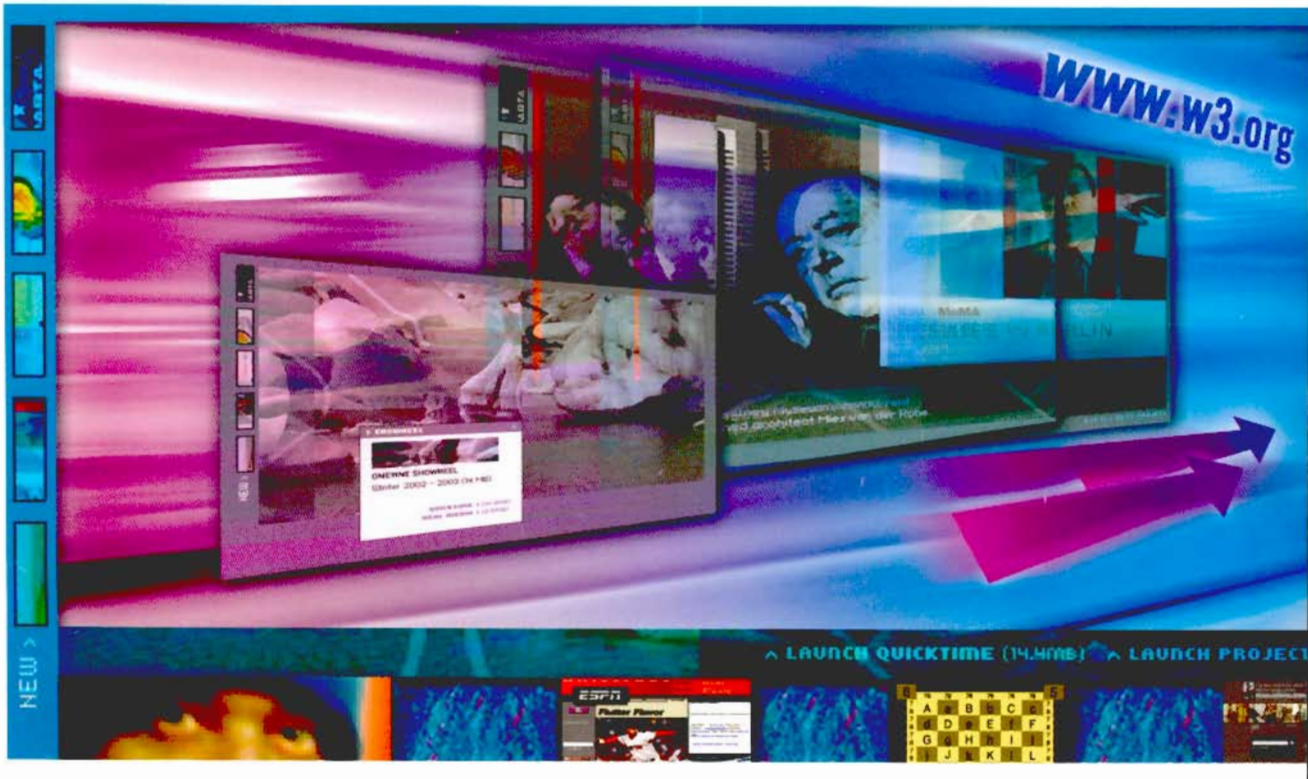
Diseño con estándares Web

Diseñe páginas para la red del futuro.



JEFFREY ZELDMAN

ANAYA MULTIMEDIA
DISEÑO Y CREATIVIDAD

Índice de contenidos

Prefacio

Hubo una época no muy lejana en la que a los conductores no les importaba arrojar botellas vacías por las ventanillas de sus coches. Años más tarde, estos mismos ciudadanos se dieron cuenta de que no era la forma correcta de eliminar la basura. La comunidad de diseño Web ha sufrido un cambio de actitud similar y los estándares Web son una de las claves de esta transformación.

La historia de nuestro medio ha consistido en resolver los problemas actuales a costa del mañana. En este libro comprobará que el enfoque "hágalo hoy, cobre mañana" ha dejado de ser necesario y productivo, y que los problemas cotidianos se pueden resolver sin generar dilemas por el camino. También se desmentirá la idea de que el diseño con estándares implica no llegar a todos los usuarios. De hecho, a menudo significa lo contrario.

Un tamaño no siempre es para todos

En este libro describiremos algunas de las formas en las que los estándares resuelven problemas habituales de diseño y producción. No hay ningún libro en el que se puedan abarcar todos los problemas y soluciones, y puede que haya otros autores que empleen un enfoque completamente opuesto a los problemas que analizaremos aquí. El libro se centra en resolver necesidades prácticas e inmediatas para poder anticipar requisitos futuros. Las técnicas e ideas presentadas en este libro se han probado y han resultado ser de gran utilidad para mi agencia de diseño y en mi experiencia como consultor, y dichas ideas, o variaciones sobre las mismas, se han utilizado en miles de sitios vanguardistas.

No todos los lectores utilizarán todas las ideas descritas en el libro de forma inmediata. Si su estilo se centra en cuadrículas, puede que no le sirvan las reglas de diseño descritas en el capítulo sobre cambios de diseño de CSS. Si el aspecto de su sitio está orientado a navegadores 4.0, puede que le interesen las técnicas híbridas descritas en los capítulos sobre diseño híbrido con XHTML y CSS, y que las técnicas de diseño con CSS le sean indiferentes.

Cualquier diseñador, programador o propietario de un sitio que sea inteligente apreciará las nociones generales que se incluyen en el libro. Los estándares resultan esenciales para cualquier medio. Como en última instancia el software utilizado para ver la Web es compatible con estándares, parece razonable conocerlos y aprender a utilizarlos correctamente. De esta forma se ahorra tiempo y dinero, se reduce la carga de trabajo, se aumenta la vida útil de un sitio y se ofrece un mejor acceso a los contenidos.

Este último aspecto es muy importante para todo el que quiera llegar a un público amplio, sobre todo con el aumento del acceso a Internet no tradicional. También tiene sus implicaciones legales, debido al creciente número de países y de estados que crean y aplican regulaciones de accesibilidad. Los estándares Web y la accesibilidad le permitirán que su sitio cumpla estas normas.

Teoría frente a práctica

No obstante, algunas de las ideas y técnicas recogidas en este libro están sujetas a debate. Si es un obseso de los estándares (en el sentido más amplio de la palabra), puede que no acepte el uso de XHTML hasta que todos los navegadores admitan correctamente el envío de documentos XHTML como por ejemplo `application/xhtml+xml` en lugar de `text/html`. Encontrará más detalles al respecto en la obra "Sending XHTML as Text/HTML Considered Harmful" de Ian Hickson (http://www.hixie.ch/advocacy/xhtml).

Si está de acuerdo con su punto de vista, puede optar por utilizar HTML 4.01 por ahora o puede configurar su servidor Web para que envíe `application/xhtml+xml` a los navegadores que lo entiendan y `text/html` a los que no lo hagan (htpp://lists.w3.org/Archives/Public/www-archive/2002Dec/0005.html). En este libro, hemos evitado este tipo de problemas debido a nuestra predisposición a realizar el trabajo en condiciones actuales, predisposición que creo que compartirá la mayoría de los lectores.

El futuro de los diseños híbridos

Del mismo modo, los fanáticos de las CSS pueden desdeñar la idea de un diseño híbrido CSS/tablas. Los métodos de diseño híbridos se ofrecen para los que los necesitan. En concreto, se ofrecen para los diseñadores cuyos trabajos deben ser igual de buenos (y prácticamente idénticos) tanto en navegadores antiguos e incompatibles como en los más recientes y compatibles.

Puede que estos métodos no sean necesarios por mucho tiempo. Mientras escribo esta página, ESPN.com ha cambiado y utiliza diseño CSS (véase la figura 1). Diez millones de lectores visitan esta página diariamente.

Figura 1.

¿Los estándares Web suponen un riesgo para los sitios comerciales? ESPN.com no lo ve así (www.espn.com). En febrero del 2003, el conocido sitio sobre deportes se volvió a lanzar con diseño CSS.

Cuando un sitio comercial de este tamaño adopta técnicas de diseño CSS, quiere decir que la llegada de los estándares Web está cerca. Aunque en un principio el uso de estándares en este sitio no era purista ni perfecto, su director artístico Mike Davidson y su equipo consiguieron algunos logros:

- La posición de todas las CSS. No hay tablas de diseño, a excepción de los elementos de los patrocinadores, que no competen al equipo de diseño.
- No hay etiquetas de fuentes.
- Ancho de banda, ancho de banda, ancho de banda. El peso del marcado y del código es la mitad de lo que lo era antes, mientras que se muestra una página mucho más rica (con mayor marcado estructural, como veremos en un capítulo posterior, se podría ahorrar incluso más ancho de banda).
- Se utiliza una única hoja de estilo para todos los navegadores, por lo que no se utiliza ni se necesita detección alguna. El sitio es prácticamente idéntico en todos los navegadores que admiten `getElementByID`, incluyendo el nuevo navegador Safari de Apple, todavía en versión beta.

ESPN.com ha obtenido significativos beneficios y ha ahorrado ancho de banda a sus lectores por utilizar muchas de las técnicas descritas aquí. Pero la primera aplicación de CSS carecía de algunas de las ideas que mencionaremos en el libro (puesto que, después de todo, no se había publicado cuando ESPN.com cambió su diseño). El sitio utiliza más detección de navegadores

de la necesaria. El marcado falla en la presentación y no realiza validación. La accesibilidad se podría mejorar y la CSS podría ser más lógica y compacta.

Cuando lea este libro, ESPN.com habrá mejorado algunos o todos estos aspectos. Pero incluso si no lo hace, ¿la botella está medio llena o medio vacía? Al tomar esta decisión, sin duda le seguirán sitios de menor relevancia comercial y menos conocidos. Cuando un sitio con 10 millones de lectores diarios se decanta por un diseño CSS, es una victoria para el diseño y los métodos de desarrollo basados en estándares Web.

Una semana antes de que ESPN.com diera el gran salto, DevEdge de Netscape (véase la figura 2) volvió a nacer como escaparate de los estándares. Este sitio siempre ha ofrecido información sólida sobre programación Web, incluyendo tutoriales sobre el uso correcto de los estándares, pero su construcción desmerece su contenido. Con el cambio de diseño basado en estándares, el sitio siguió su propio ejemplo y empezó a actuar como modelo.

Entre las opciones que incluía destacamos un diseño CSS sin tablas, estilos que los usuarios pueden seleccionar, menús desplegables basados en estándares y URL de enlaces que se visualizan en función de la conveniencia del lector a través de la propiedad de contenidos CSS.

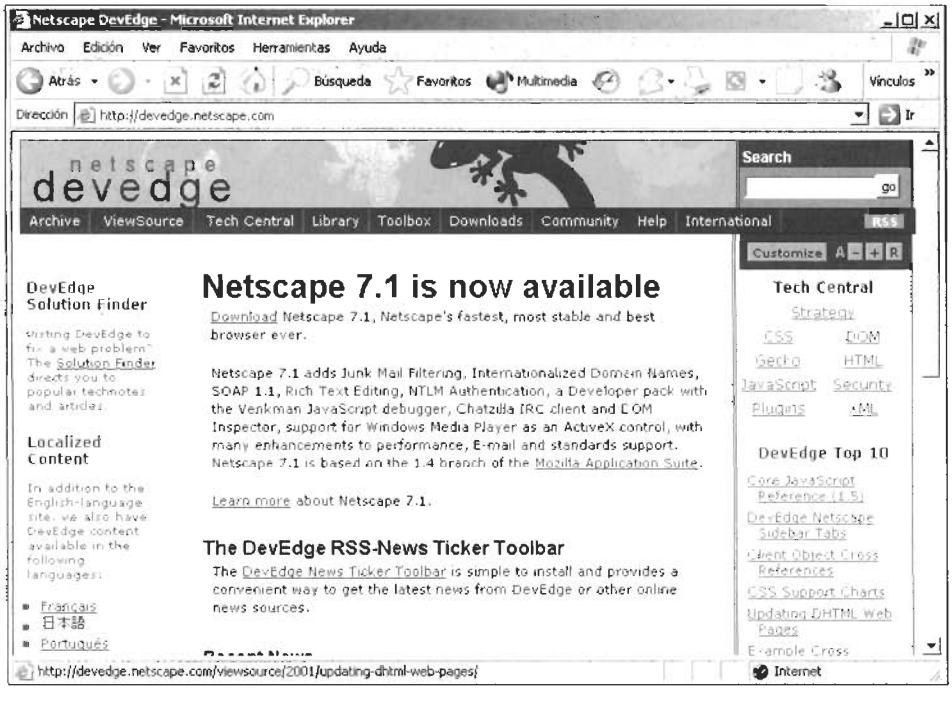


Figura 2.

En febrero del 2003, DevEdge de Netscape siguió su propio ejemplo y cambió a un diseño y un marcado basado en estándares (http://devedge.netscape.com/).

Un conjunto continuo de normas, no inflexible

Como mencionaremos en el libro, los estándares Web no constituyen un conjunto inflexible de normas. Al pasar a los estándares Web, puede que no consiga una separación perfecta entre estructura y presentación en su primer sitio o incluso en el quinto. Puede que con los primeros esfuerzos por conseguir la accesibilidad, sólo obtenga el mínimo requerido por la WAI Priority 1 y no siempre con exactitud.

Lo importante es empezar. El miedo a la imperfección puede inmovilizar a los incautos de la misma forma que la vergüenza por nuestro físico nos impide ir al gimnasio. Pero no se empieza a perder los primeros kilos hasta que se realiza el primer esfuerzo físico, por humilde que sea. Del mismo modo, nuestros sitios no conseguirán la deseada compatibilidad si no empezamos en algún momento. La eliminación de etiquetas de fuentes puede ser un comienzo. También puede reemplazar el marcado no estructural por etiquetas `<h. . .>` y `<p>` con más sentido. Se trata de una forma excelente de empezar y, como consecuencia, nos detendremos extensamente en la descripción del marcado moderno y de XHTML.

Muestre, no venda

En ocasiones, los diseñadores se centran en la parte comercial de los estándares. Con el paso de los años, he recibido cientos de cartas de diseñadores que quieren utilizar estándares pero "que sus clientes no les dejan". Si los estándares son algo continuo, ¿cómo es posible que los clientes se opongan? Por ejemplo, incluso el sitio en el que se utilicen más tablas puede validarse con HTML 4.01 o XHTML 1.0 Transicional, y seguir cumpliendo la U.S. Section 508 o las directivas de accesibilidad WAI Priority 1. Ningún cliente se opondría a un sitio accesible sin errores.

Lo que la mayoría de los diseñadores preocupados con los estándares sostiene es que se ven limitados en cuanto a las posibilidades de un determinado proyecto. Por ejemplo, no pueden utilizar un diseño CSS puro en un proyecto concreto porque el cliente (o su jefe) utiliza Netscape 4, cuya compatibilidad con CSS es, cuando menos, incompleta. Puede que sea cierto, pero no impide escribir un marcado válido y CSS correcto, y utilizar el método que describiremos en un capítulo posterior para conseguir un aspecto y un funcionamiento aceptable en distintas generaciones de navegadores.

Mi agencia es muy clara en lo que respecta a los estándares Web y a la accesibilidad pero no acerca de los métodos empleados o si cumplen un imaginario decálogo de pureza de estándares. Aplicamos el método que mejor se adapta al proyecto. Para venderlo, disponemos de dos opciones:

- En nuestras propuestas, indicamos explícitamente las tecnologías que se van a utilizar, con una descripción simple y escueta. Por ejemplo, "para el marcado se utilizará XHTML 1.0 Transitional, un estándar actual". Una vez que el cliente ha aceptado la propuesta y ha firmado el contrato, se utiliza el "permiso" concedido para emplear el estándar indicado y

no se necesita nada más. Si existe la posibilidad de que una decisión afecte al resultado visual en navegadores más antiguos, también se indica en la propuesta.

- Al comenzar el trabajo y mostrar las distintas fases del mismo al cliente, el debate tecnológico se reduce al mínimo, incluso aunque se trate de un cliente con amplios conocimientos técnicos. Cuando conseguimos un nuevo diseño con un tercio del ancho de banda de su predecesor y que conserva el formato (incluso el formato avanzado), independientemente de cuántas veces se haya cambiado o actualizado, no decimos que ha sido gracias a CSS. Decimos que hemos diseñado un sistema que protege el formato y que reduce el ancho de banda. Si el cliente cree que sabemos lo que hacemos y nos ofrece otros proyectos, nos damos por satisfechos.

Deje que su trabajo se encargue de vender

Cuando Hillman Curtis Inc. y Happy Cog colaboraron en el cambio del sitio de Fox Searchlight Pictures, nuestro principal contacto era un experimentado diseñador y programador Web. Sin embargo, desconocía algunos de los métodos híbridos CSS/tablas que empleamos. Continuamente se sorprendía por la velocidad del sitio.

Naturalmente, al realizar la entrega final, incluimos una guía de estilos, en la que explícitamente se describían los aspectos técnicos del sitio. De esta forma, no fue necesario vender lo que habíamos hecho ya que los resultados lo habían hecho por nosotros.

Cuando sea conocido por realizar este tipo de trabajos, los clientes le buscarán por sus resultados y tendrá mucho menos que vender. Aunque Happy Cog tiene una opinión agnóstica en lo que se refiere a los diseños CSS híbridos frente a los puros, en todos mis sitios personales se utiliza el diseño CSS y cada vez más en los sitios de menor tamaño de la agencia. Incluso hemos descubierto que resulta más sencillo realizar diseños exploratorios en CSS que en Photoshop; es más sencillo porque se ahorra tiempo y se reducen los pasos empleados.

Un proyecto reciente de nuestro cliente Clear Channel Entertainment requería diferentes variaciones de diseño, como sucede inicialmente en muchos proyectos. Le dijimos al cliente que se trataba de diseños CSS, a lo que respondió que ya lo sabía. Cuando estos métodos se convierten en algo habitual (en parte por su aplicación en sitios comerciales como ESPN.com, DevEdge.com y Wired.com, así como en sitios públicos y gubernamentales, como veremos en un capítulo posterior), no hace falta mencionar su utilización, como tampoco hace falta mencionar que se emplean imágenes GIF o JPEG.

Cómo vender persona a persona

Hemos descrito lo que sucede en el caso de una agencia Web, pero lo mismo se aplica a trabajos domésticos. Trate de evitar atascarse en discusiones sobre navegadores base y otros problemas de la vieja escuela. Elija las especificaciones adecuadas, descríbalas

brevemente en la documentación que le solicite su superior y póngase manos a la obra.

Dos años antes de escribir este libro, ofrecí una conferencia a un grupo de programadores Web de una importante organización gubernamental norteamericana. Les interesaban los estándares Web pero lamentaban la dependencia de su agencia de un navegador antiguo e incompatible, en el que muchos pensaban que era imposible utilizar estándares de forma interna (esa opinión no tenía fundamento; recuerde, los estándares Web son algo continuo).

Al escribir esta introducción, visité la agencia de nuevo para dar otra charla y comprobé que el ambiente había cambiado. La mayoría utilizaba Netscape 7. Algunos seguían con Netscape 4, ya que algunas de las aplicaciones se habían diseñado para aprovechar las ventajas de `document.layers`, la fuerza oculta del DHTML propietario de Netscape. Para algunos de los miembros del equipo, estas aplicaciones resultaban fundamentales. Pero en lugar de darse por vencidos, a los asistentes a la conferencia les interesaba saber cómo migrar dichas aplicaciones al DOM del W3C.

Vientos de cambio

Este tipo de cambios se producen en todas partes, en ocasiones a gran rapidez y en otros casos más lentamente. Los cambios se producen siempre que la gente inteligente se enfrenta a la tarea de crear o actualizar contenidos Web. Prácticamente desapercibido entre las preocupaciones económicas y políticas y la losa de las fechas de entrega diarias, nuestro entendimiento de cómo funciona la Web y cómo debería diseñarse está sujeto a una profunda y continua metamorfosis. En un futuro no muy lejano, los estándares Web se analizarán y se estudiarán tan ampliamente como el uso de la Web y la arquitectura de la información, y se considerarán tan esenciales como estas disciplinas, ya que resultan igual de fundamentales para la salud de nuestros sitios y del medio en el que trabajamos.

Este libro es extenso y se ha elaborado con sumo cuidado, pero apenas describe superficialmente la importancia de los estándares para la Web. Hay mucho más sobre CSS, sobre marcado accesible y estructurado, y sobre DOM de lo que se pueda incluir en este libro o una sola referencia. Y como ya hemos mencionado, hay más formas de analizar los problemas descritos que las utilizadas por este autor.

Meta a dos diseñadores en una misma habitación y tendrá tres opiniones. Es poco probable que dos diseñadores estén de acuerdo sobre tipografía, navegación o colores. Lo mismo se aplica a los estándares. A este respecto hay tantas divergencias como usuarios y teóricos.

No existe ningún libro que pueda ofrecer todo a los lectores y éste es una mera señal en un viaje cuyo camino debe encontrar personalmente. Este libro merecerá le pena si le ayuda a comprender cómo funcionan conjuntamente las tecnologías de estándares para crear sitios compatibles, al tiempo que le ofrece algunos consejos que pueden resultarle de utilidad.

Me topé con los estándares Web tras tres años de diseñar sitios a la antigua usanza y tardé otros cinco años en alcanzar los conocimientos en los que se basa este libro. Puede que no esté de acuerdo con parte de lo que haya dicho en estas páginas.

Puede que yo también lo haga dentro de seis meses o de dos años. Lo importante es no limitarse a las diferencias o rechazarlo todo de plano por no estar seguro sobre algún aspecto concreto. Lo importante es comenzar a aplicar cambios para que sus proyectos lleguen a más usuarios durante más tiempo y, a menudo, con el menor coste.

Si no es ahora, ¿cuándo? Si no lo hace, ¿quién lo hará?

Introducción

Este libro es para diseñadores, programadores, propietarios y administradores que quieren que sus sitios cuesten menos, funcionen mejor y lleguen a un público más amplio, no sólo en los navegadores, lectores de pantalla y dispositivos inalámbricos actuales, sino también en los del próximo año, el siguiente y en años futuros.

Muchos de nosotros hemos aguantado la obsolescencia que parece formar parte del rápido avance tecnológico de la Web. Siempre que aparece una nueva versión de un navegador o un nuevo dispositivo de Internet parece que afecta negativamente al sitio que acabamos de crear (o de pagar).

Diseñamos únicamente para volver a diseñar. A menudo, volvemos a diseñar no para añadir opciones demandadas por los usuarios o para aumentar el uso de un sitio, sino simplemente para estar al día de los navegadores y dispositivos que parecen adelantarse a nuestros ciclos de planificación y desarrollo.

Incluso cuando un nuevo navegador o dispositivo se apiada de nuestro sitio y no le afecta, las denominadas técnicas de compatibilidad inversa, que utilizamos para que nuestros sitios se comporten de la misma forma y tengan el mismo aspecto en todos los navegadores, se cobran un tributo de sobrecarga humana y financiera.

Estamos tan acostumbrados que lo consideramos normal; es el precio que tiene trabajar con la Web. Es un coste que la mayoría no se puede permitir (en caso de que fuera posible).

Costes en espiral, disminución de beneficios

El código redundante, los diseños de tablas anidadas, las etiquetas `<font>` y otros elementos redundantes duplican y triplican el ancho de banda necesario incluso para los sitios más sencillos.

Los usuarios lo pagan y se ven obligados a esperar eternamente a que se carguen las páginas (o se cansan de esperar y abandonan. Algunos esperan pacientemente para descubrir que cuando finalmente se carga el sitio, no pueden acceder a él).

Los servidores pagan el precio consumiendo 60 k por página cuando bastaría con 20 y nosotros pagamos a las empresas de alojamiento (o aumentamos nuestros presupuestos en IT) para hacer frente al ancho de banda que devoran nuestras páginas. Cuando más visitantes tenemos, mayor es el coste. Para procesar nuestros diseños, nuestras bases de datos realizan más consultas de las necesarias, lo que aumenta aún más el gasto.

En última instancia, nos vemos forzados a contratar servidores adicionales para satisfacer la demanda, no del creciente número de visitantes sino del exceso de marcado y de código.

Mientras tanto, las tarifas por hora que pagamos a los programadores para diseñar el código de nuestros sitios de seis formas diferentes (y después para que escriban más código que se ajuste a la versión de cada visitante) pueden aumentar los costes de tal forma que simplemente nos quedamos sin fondos. En esto, aparece un nuevo navegador o dispositivo inalámbrico y, como no hay liquidez para cubrir los gastos que supone crear el código para dicho navegador o dispositivo, el ciclo de obsolescencia comienza de nuevo.

Muchos de nosotros hemos visitado sitios con un nuevo navegador y se nos ha solicitado un navegador "moderno" que es mucho más antiguo que el que estamos utilizando.

No quiere decir necesariamente que los propietarios y programadores del sitio sean estúpidos o poco considerados, simplemente han agotado su presupuesto de actualización perpetua y no les queda nada que ofrecer.

En otros casos, el problema no es la falta de fondos sino la falta de conocimiento y de no saber cuál es el mejor beneficio de una inversión.

El diseño de Connected Earth, cuyo eslogan es "How communication shapes the world", se modificó recientemente con un coste de 1 millón de libras esterlinas (aproximadamente 1'6 millones de euros).

A pesar de esta cantidad dilapidada en la programación del sitio, prácticamente es incompatible con todos los nuevos navegadores. Deniega el acceso a los usuarios de Mozilla (véase la figura 1), Netscape 6/7 y Opera (véase la figura 2).

Como el sitio tampoco es compatible con sistemas operativos que no sean Windows, sucede lo mismo con los usuarios de Internet Explorer para Macintosh.

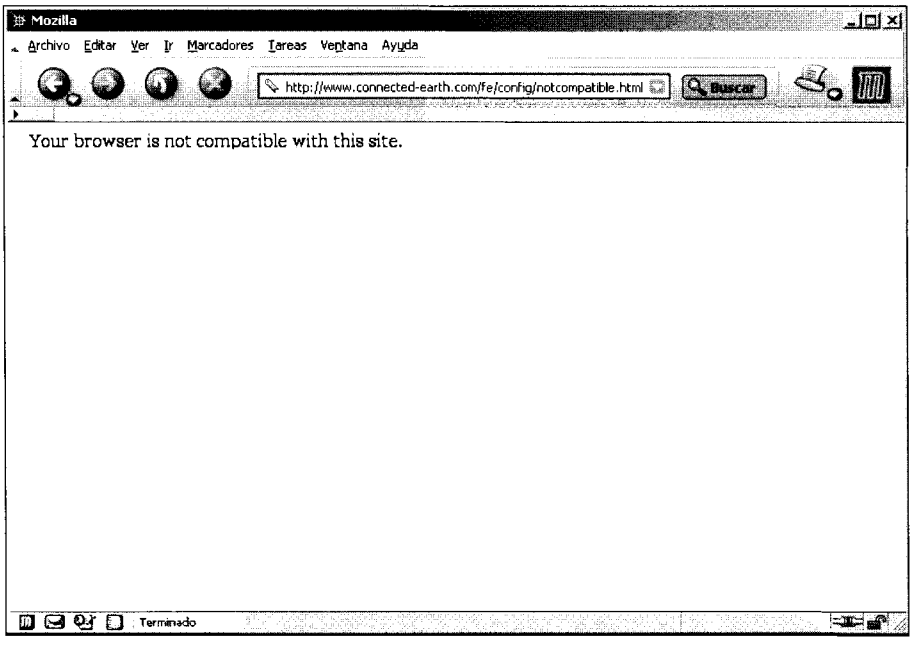


Figura 1.

A pesar el ingente presupuesto para su desarrollo, Connected Earth no es compatible con la mayoría de los navegadores modernos. No admite usuarios de Mozilla (en la imagen), Netscape 6/7 y Opera de ninguna plataforma ni a los usuarios de Internet Explorer para Mac OS (www.connected-earth.com).

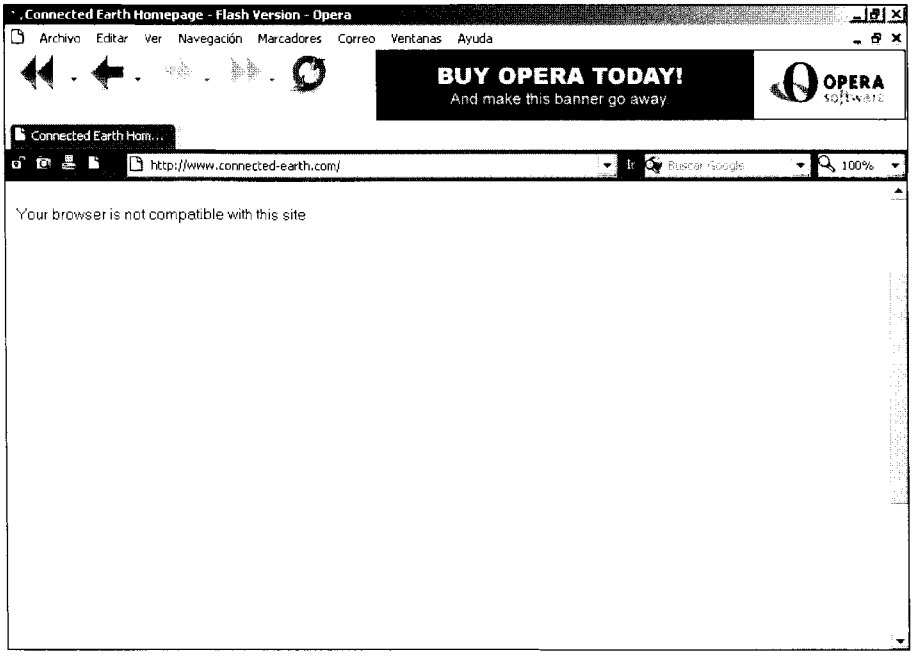


Figura 2.

El mismo sitio visto (o, para ser más precisos, no visto) en Opera 7 para Windows XP. Los problemas técnicos de Connected Earth se podrían haber evitado si hubieran aplicado los métodos descritos en este libro.

Se trate de un sitio que se ha quedado obsoleto debido a que el gasto de actualizarlo continuamente haya superado el presupuesto o de un sitio que se haya creado obsoleto debido a especificaciones anticuadas, el resultado es el mismo: la pérdida de un creciente número de clientes potenciales. Después de todo, independientemente de la inversión, el objetivo es atraer visitantes, no rechazarlos.

Existe una solución.

El fin de la obsolescencia

Las tecnologías creadas por el World Wide Web Consortium y otros organismos de estándares admitidos por la mayoría de los navegadores y dispositivos actuales permiten que se puedan diseñar y crear sitios con un funcionamiento duradero, aunque cambien dichos estándares y navegadores. Los veteranos del sector curtidos en mil batallas se mostrarán escépticos ante esta afirmación, pero en este libro le mostraremos cómo funciona.

Aprenderá a escapar del ciclo "crear, descartar, volver a crear", a diseñar para el presente y el futuro de la Web, y para superar la barrera del escritorio, sin excluir usuarios potenciales y sin malgastar tiempo y recursos económicos en soluciones propietarias a corto plazo que contienen las semillas de su propia obsolescencia.

¿Qué es el W3C?

Creado en 1994, el World Wide Consortium (W3C) (http://www.w3.org/) trabaja con especificaciones y directrices con la intención de promover la evolución de la Web y garantizar que las tecnologías Web funcionan correctamente. El consorcio está formado aproximadamente por unas 500 organizaciones. Su director, Tim Bernes-Lee (http://www.3w.org/People/Berners-Lee/), inventó la Web en 1989. Entre las especificaciones desarrolladas por el W3C destacan HTML, CSS, XML, XHTML y el Modelo de objetos de documento (DOM), entre muchas otras.

Durante años, el W3C denominó a estas especificaciones "Recomendaciones", lo que puede que haya alentado inadvertidamente a empresas miembro como Nestcape y Microsoft a implementar especificaciones del W3C de forma menos rigurosa. En su aparición en 1998, el Web Standards Project (www.webstandards.org) convirtió las Recomendaciones fundamentales del W3C en "estándares Web", una maniobra empresarial que ayudó a que la compatibilidad precisa y completa con estas especificaciones fuera algo esencial en cualquier navegador o dispositivo de Internet (como veremos más adelante).

Entre otros organismos de estándares podemos destacar a la ECMA (European Computer Manufacturers Association), responsable del lenguaje ECMAScript que familiarmente se denomina JavaScript estándar, como veremos en un capítulo posterior.

No se trata de un libro para puristas ni teóricos, sino para usuarios prácticos que quieren hacer las cosas bien. Va dirigido a los profesionales creativos que buscan consejo y una serie de técnicas probadas que les permitan modificar sus conocimientos y conseguir crear magníficos sitios Web que funcionen para un mayor número de visitantes y clientes.

Con este libro, los diseñadores y programadores podrán modificar rápidamente sus métodos existentes de creación de sitios Web que funcionen en varios navegadores y dispositivos y no simplemente en unos cuantos. Al mismo tiempo, evitarán la obsolescencia perpetua producto de técnicas de marcado y de codificación propietarias.

Los propietarios y administradores de sitios que lean este libro tendrán la oportunidad de ahorrar dinero en especificaciones que únicamente perpetúan el ciclo de pérdidas. Sabrán cómo escribir documentos de requisitos que les permitan obtener sitios con compatibilidad directa.

¿Qué es la compatibilidad directa?

¿Qué queremos decir con compatibilidad directa? En definitiva que, correctamente diseñado y creado, cualquier documento que se publique en la Web pueda funcionar en diferentes navegadores, plataformas y dispositivos de Internet, así como en los nuevos dispositivos y navegadores que aparezcan en el futuro. Los estándares abiertos hacen realidad esta posibilidad.

Como atracción añadida, el diseño y creación con estándares reduce los costes de producción y mantenimiento mientras que los sitios resultan más accesibles para aquellos con necesidades especiales (traducimos: más clientes y menos costes, mejora de las relaciones públicas, menor probabilidad de problemas relacionados con la accesibilidad).

En cuanto a los estándares Web, se refieren a lenguajes estructurales como XHTML y XML, lenguajes de presentación como CSS, modelos de objeto como el DOM del W3C y lenguajes de secuencia de comandos como ECMAScript, que describiremos en este libro junto a muchos otros.

Promovidos por comités de expertos, estas tecnologías se han diseñado cuidadosamente para ofrecer los mayores beneficios al mayor número de usuarios Web. De forma conjunta, dichas tecnologías forman el mapa para un desarrollo Web racional, accesible, sofisticado y de bajo coste. (Una indicación para los propietarios y administradores de sitios: no se preocupen por los capítulos técnicos del libro. Basta con que sus empleados y vendedores los entiendan.)

Al hablar de navegadores compatibles con estándares nos referimos a productos como Mozilla, Netscape 6+, MSIE 5+/Mac, MSIE6+/Win y Opera 7+ que comprenden y admiten XHTML, XML, ECMAScript y el DOM. Se preguntará si estos navegadores resultan perfectos a la hora de admitir todos y cada uno de estos estándares. Evidentemente no. No existe ningún software de ninguna categoría que se haya diseñado sin errores. Es más, los estándares son

sofisticados por sí mismos y la forma en que interactúan entre sí resulta de gran complejidad.

Sin embargo, los navegadores modernos son tan sólidos que podemos descartar métodos antiguos, trabajar mejor y satisfacer a un mayor número de usuarios. Y como por naturaleza resultan inclusivos, incluso podemos adaptar a los usuarios de navegadores y dispositivos antiguos, pero con compatibilidad directa.

Sin normas, sin dogmas

No tiene entre sus manos un libro religioso ni dogmático. No existe una forma perfecta de diseñar un sitio Web, ni un único método de incorporar estándares a su trabajo. En este libro no abogamos por un cumplimiento estricto de lo estándares en detrimento de enfoques progresivos que puedan resultar más indicados para determinados sitios o tareas. No sería justo decir que todas las recomendaciones del W3C son perfectas.

En esta obra le mostraremos lo que debe saber para superar los ocasionales problemas de compatibilidad en Internet Explorer, Netscape Navigator y otros navegadores modernos. También le ofreceremos estrategias para lidiar con los navegadores más antiguos que parte de su público pueda utilizar, incluyendo ese tipo tan tozudo de la oficina de la esquina.

No le mentiremos. Algunos métodos propietarios resultan más sencillos de utilizar que determinadas especificaciones del W3C. Por ejemplo, la creación de secuencias de comandos propietarias sólo para IE con los métodos abreviados de Microsoft como `innerhtml` puede resultar más rápida y sencilla que crear las secuencias de comandos para todos los navegadores con ayuda del DOM estándar del W3C.

Aun así, desde un punto de vista empresarial, tiene más sentido diseñar el código para todos los navegadores que sólo para uno, técnica que emplea el DOM.

Del mismo modo, aunque describamos los beneficios del marcado estructural y de XHTML, no pretendemos que todas las etiquetas de todas las páginas de todos los sitios siempre tengan que ser estructurales. Y no le obligaremos a cambiar todos los sitios de HTML a XHTML, aunque esperamos que las ventajas de XHTML le hagan reconsiderar la idea de realizar dicha transición lo antes posible.

El autor de este libro es conocido por recomendar el diseño de páginas Web con hojas de estilo en cascada (CSS) en lugar de las tradicionales tablas HTML siempre que sea posible (y siempre que se disponga del público adecuado). El estándar CSS resuelve muchos problemas, tanto para programadores como para lectores.

El diseño CSS es el futuro y ya se utiliza en muchos sitios, desde entidades corporativas muy visitadas como Wired (www.wired.com) (véase la figura 3), importantes motores de búsqueda (www.alltheweb.com), hasta sitios del sector público y personales.

Figura 3.

Es importante. Es corporativo. Y está diseñado con los estándares Web HTML y CSS (www.wired.com).

No obstante, los diseños con tablas no han desaparecido y siguen siendo la herramienta adecuada en muchos trabajos de diseño, trabajos que se pueden realizar sin sacrificar el cumplimiento de los estándares o la compatibilidad directa. Aunque las ventajas del diseño CSS son indudablemente atractivas, se trata de su sitio y de su decisión.

De acuerdo con algunos puristas de los estándares, tecnologías propietarias populares como Flash y QuickTime no tienen sitio en la Web. Esta actitud puede ser válida para los teóricos, pero el resto tenemos cosas que hacer y tecnologías como Flash y QuickTime son las mejores herramientas para determinadas tareas. En este libro no hablaremos mucho sobre dichas tecnologías a excepción de su relación con el marcado y la accesibilidad, pero no porque estemos en contra de las mismas, sino porque simplemente se escapan a los objetivos del libro.

Puede que se haya encontrado con defensores de los estándares Web que le desaconsejen el uso del formato propietario GIF o que le acusen de hereje si opta por diseñar sus encabezados con texto GIF en lugar de XHTML con CSS. A menudo, es aconsejable utilizar encabezados XHTML y suele resultar la mejor opción. Pero en el mundo real, lo que importa es la marca. Un tipo de letra correctamente ajustado en formato GIF (o en PNG, no propietario) puede resultar más

indicado para su sitio que utilizar CSS para obtener fuentes Arial, Verdana o Georgia de las carpetas Sistema de los usuarios. Se trata de su sitio. Es su decisión.

Práctica, no teoría

No tenemos nada en contra de los teóricos cuya pasión promueve la creación de estándares Web. Más bien lo contrario. Los admiramos profundamente y tenemos la suerte de conocerlos y de haber aprendido de algunos de ellos.

Pero este libro está dirigido a diseñadores y programadores, y a los clientes y empresarios que contratan sus servicios. Nuestro análisis de los estándares Web se basará en el contexto de aspectos de diseño, contenido y marketing. Nuestro objetivo es guiarle por el universo de los estándares Web para que adopte sus propias decisiones, las únicas con las que cualquier diseñador o cliente estará de acuerdo.

Si en este libro encuentra un ápice de dogmatismo o sostiene un punto de vista inflexible, es el siguiente: el coste empresarial actual es demasiado elevado. Nadie que lea este libro puede permitirse diseñar los sitios Web del presente con los inconsistentes métodos del pasado.

Las técnicas de la vieja escuela tuvieron su importancia cuando los estándares Web estaban por escribirse, mientras que otros apenas se admitían en los principales navegadores. Pero eso es agua pasada. Del mismo modo, la creación del código de un sitio de seis formas diferentes parecía una práctica razonable producto del boom de Internet y de presupuestos escandalosamente excesivos. Pero eso también es agua pasada.

XHTML, XML, CSS, ECMAScript y el DOM se han creado para durar. En sí mismos no constituyen un fin, sino componentes de una solución racional a problemas que han sufrido los propietarios y programadores de sitios desde la aparición de la etiqueta `<blink>`.

El Standards Web Project

Creado en 1998, el Standards Web Project consiguió el fin de la guerra de los navegadores convenciendo a Netscape. Microsoft y a otros fabricantes a que admitieran de forma precisa y completa especificaciones (estándares) que redujeran el coste y la complejidad del desarrollo, y que garantizaran un acceso sencillo y asequible para todos. Además de los creadores de navegadores, en la actualidad el grupo trabaja con fabricantes de herramientas de programación como Macromedia y con propietarios y programadores de sitios. El Standards Web Project es una coalición de base popular. Sus actividades son completamente voluntarias y no tienen ánimo de lucro.

¿Realmente es necesario?

Para que los sitios Web se desprendan de sus votos tradicionales y pasen al siguiente nivel, los diseñadores y programadores deben aprender a utilizar estándares Web, y

los propietarios y administradores de sitios deben saber cómo los estándares pueden ser útiles para sus empresas.

La revelación de los estándares Web no se manifestará por sí misma. Es poco probable que los empresarios registren el sitio Web del W3C, descifren sus documentos de estilo protegido e intuitivamente descubran lo que crípticos acrónimos como XHTML y CSS pueden significar para su beneficio. Del mismo modo, los diseñadores y programadores sin apenas tiempo, luchando por cumplir los plazos, no pueden pararse a escudriñar listas de correo y cursos prácticos en línea en busca de explicaciones comprensibles sobre las cambiantes tecnologías Web.

En el caso de los estándares, era necesario escribir este libro. Como cofundador del Web Standards Project, el trabajo me atrapó a mí. Me llamo Jeffrey. Tengo un ratón. También utilizo la fórmula editorial "nosotros", aunque no sea absolutamente necesario. Lo adopté cuando empecé a publicar sitios Web en 1995. Creo (creemos) que al lector no le molestará este hábito.

Probablemente preferiría leer sobre gráficos y diseño animado, sobre las nuevas teorías de arquitectura de sitios y sobre la facilidad de uso en lugar de sobre los argumentos de cambio tecnológico de la Web. Yo también preferiría escribir sobre estos temas. Pero malgastaríamos nuestros esfuerzos en diseño, arquitectura y facilidad de uso si nuestros sitios dejaran de funcionar en el Navegador X o el Dispositivo Y. No se harían películas sin un acuerdo del sector cinematográfico respecto a la velocidad de los fotogramas, las lentes y las técnicas de grabación de sonido. Del mismo modo, la salud del diseño y de la programación Web depende de la adopción de estándares Web. Sin estos estándares, no existirá una verdadera facilidad de uso ni un enfoque coherente del diseño.

En términos de aceptación, la Web despegó con mayor rapidez que ningún otro medio. Pero su éxito comercial precedió al desarrollo de estándares de la industria, lo que nos puso en la peligrosa posición de crear productos (sitios Web) que continuamente se quedaban obsoletos debido a las innovaciones de los navegadores y dispositivos. Estábamos tan ocupados en la producción que no teníamos tiempo para cuestionar los métodos de trabajo. Hoy en día, si pretendemos continuar con el trabajo y la producción, es necesario cuestionar y modificar nuestros métodos.

Los estándares Web son las herramientas con las que podemos diseñar y generar sofisticados sitios de gran belleza que funcionen mañana igual que como lo hacen hoy. En este libro, explicaremos la función de cada uno de los estándares y cómo utilizarlos de forma conjunta para crear sitios con compatibilidad directa. El resto es cosa suya.

Jeffrey Zeldman
New York City
2003

Parte I
Houston, tenemos un problema

Capítulo 1
El 99'9 por ciento de los sitios Web están obsoletos

En la actualidad, existe una enfermedad que afecta prácticamente a todos los sitios de la Web, desde las páginas personales más sencillas hasta los sitios multimillonarios de gigantes corporativos.

De forma astuta y maliciosa, esta enfermedad pasa desapercibida ya que está basada en normas de la industria. Aunque sus propietarios y administradores no lo sepan todavía, el 99'9 por ciento de todos los sitios Web son obsoletos.

Estos sitios pueden funcionar correctamente en navegadores de escritorio cuyos nombres acaban en 4 o 5. Pero fuera de estos entornos tolerantes ante los fallos, los síntomas de la enfermedad han empezado a aparecer.

En las versiones modernas de Microsoft Internet Explorer, el navegador Opera de Opera Software, Netscape Navigator y Mozilla (el navegador de código abierto basado en Gecko cuyo código se encuentra detrás de Navigator, CompuServe, AOL para OS X, AOL China y otros entornos de navegación), los diseños cuidadosamente construidos empiezan a tambalearse y los comportamientos costosamente generados han dejado de funcionar.

Con la evolución de estos conocidos navegadores, el rendimiento de los sitios continúa su deterioro.

En navegadores sin marca, en lectores de pantalla utilizados por usuarios discapacitados y en dispositivos no tradicionales cada vez más populares desde Palm Pilots hasta teléfonos móviles compatibles con la Web, muchos de estos sitios nunca han funcionado y no lo hacen ahora, mientras que otros

funcionan a la perfección. En función de las necesidades y del presupuesto, los propietarios y programadores de los sitios bien han ignorado estos navegadores y dispositivos o bien los han admitido tras detectar su presencia y les han añadido marcado y código personalizado como si se tratara de navegadores convencionales.

Para comprender la futilidad de esta obsoleta práctica de la industria y para comprobar cómo aumenta continuamente el coste y la complejidad de la programación Web sin llegar a alcanzar los objetivos propuestos, debemos analizar los navegadores modernos y ver en qué se diferencian con los navegadores incompatibles del pasado.

Navegadores modernos y estándares Web

A lo largo de este libro, cuando hablemos de navegadores modernos o compatibles con estándares, nos estaremos refiriendo a navegadores que comprenden y admiten HTML y XHTML, Hojas de estilo en cascada (CSS), ECMAScript y el Modelo de objetos de documento (DOM) del W3C. De forma conjunta, se trata de los estándares que nos permiten desprendernos del marcado de presentación y de lenguajes de secuencia de comandos incompatibles, y de la obsolescencia perpetua que generan.

Al cierre de esta edición, entre dichos navegadores se incluían Mozilla 1.0 y superior, Netscape Navigator 6 y superior, Microsoft Internet Explorer 6 y superior para Windows, Microsoft Internet Explorer 5 y superior para Macintosh y el navegador Opera 7 de Opera Software. En una sección posterior del libro encontrará un gráfico en el que se enumera y compara la primera hornada de navegadores compatibles. No se trata de una lista exhaustiva. Cualquier intento de enumerar todos los navegadores compatibles con estándares resultaría inútil. Aunque utilizaremos el término "compatible con estándares", no olvide lo que mencionamos en un capítulo anterior: no existe ningún navegador totalmente compatible con estándares.

La ausencia de navegadores perfectos no es suficiente para evitar el uso de estándares. En la actualidad hay millones de usuarios de Internet Explorer 5 o 5.5 para Windows. Desde el punto de vista de los estándares, estos navegadores son inferiores a IE6/Windows, Netscape 6+, etc. ¿Significa esto que si entre su público hay usuarios de IE5/Windows debería olvidarse de los estándares Web? ¿Significa que debería obligar a los usuarios de IE5/Windows a que se actualicen? Creemos que no. El diseño y la programación orientada a estándares no deberían equivaler a un diseño exclusivo para los últimos navegadores.

Del mismo modo, el uso de XHTML y CSS necesariamente no implica mandar a paseo a los usuarios de Netscape 4. Es poco probable que un sitio correctamente diseñado y generado con estándares tenga el mismo aspecto, píxel a píxel, en Netscape 4 que en la mayoría de los navegadores compatibles. De hecho, en función de su método de diseño, puede tener un aspecto completamente diferente. Y no pasa nada. Explicaremos las razones y las causas en un capítulo posterior.

Un nuevo código para un nuevo trabajo

Los navegadores modernos no son simplemente nuevas versiones del mismo producto. Difieren completamente de sus predecesores y, en muchos casos, se han creado desde cero. Mozilla, Netscape 6/7 y navegadores similares basados en Gecko no son nuevas versiones de Netscape Navigator 4. IE5/Mac no es una versión actualizada de IE4/Mac. Opera 7 no está basado en el mismo código de las versiones anteriores del navegador. Estos productos se han creado con código nuevo para realizar un trabajo nuevo; en concreto, para cumplir con la mayor precisión los estándares Web que analizaremos en este libro.

Por el contrario, los navegadores de los 90 se centraban en tecnologías propietarias (sólo para Netscape, sólo para Microsoft) y pasaban por alto los estándares. Los navegadores antiguos ignoraban completamente algunos de estos estándares, pero no suponía un problema de programación demasiado serio. Si por ejemplo los navegadores no admitían el estándar PNG, los programadores no utilizaban imágenes PNG. Sin problema. El problema aparecía cuando estos navegadores admitían los estándares de forma parcial e incorrecta. La compatibilidad incorrecta con estándares tan básicos como HTML generaba un entorno de publicación Web insostenible que, a su vez, degeneró en métodos de producción insostenibles.

Cuando el apéndice de un paciente se inflama, un cirujano cualificado realiza una operación de apendicitis completa. Imagine, por el contrario, que un estudiante en prácticas eliminara la mitad del apéndice, se cargara aleatoriamente algún que otro órgano y que se olvidara de coser al paciente. Pedimos disculpas por los ejemplos pero así era la compatibilidad con estándares en los navegadores antiguos: peligrosamente incompleta, incompetente y una amenaza para la salud de la Web.

Si Netscape 4 ignoraba las reglas CSS aplicadas al elemento `<body>` y añadía cantidades aleatorias de espacios en blanco a todos los elementos estructurales de la página y si IE4 procesaba correctamente la etiqueta pero no los espacios en blanco, se preguntará qué tipo de CSS era seguro escribir. Algunos programadores optaron por no escribir CSS. Otros simplemente escribían una hoja de estilo para compensar los fallos de IE4 y otra completamente diferente para solucionar los de Netscape 4. Para solventar las diferencias de fuentes e interfaz de usuario entre plataformas, los programadores podían escribir varias hojas de estilo en función de si el visitante del sitio utilizaba Windows o Macintosh (mucho peor si utilizaba Unix o Linux).

Estos problemas CSS eran una gota en el océano, al igual que CSS. Los navegadores no se ponían de acuerdo sobre HTML, sobre la representación de tablas o sobre los lenguajes de secuencia de comandos utilizados para añadir interactividad a la página. No existía una forma única de estructurar el contenido de una página (bueno, en realidad sí la había, pero ningún navegador la admitía). No había una forma correcta de producir el diseño de una página (bueno, en realidad sí, pero ningún navegador la admitía) ni de añadir un comportamiento sofisticado a un

sitio (de hecho sí la había, pero ningún navegador la admitía).

En su intento de solucionar estas incompatibilidades, los diseñadores y programadores decidieron crear versiones personalizadas de marcado y código (no estándar) para cada uno de los distintos navegadores defectuosos existentes. Era todo lo que se podía hacer en aquel momento si queríamos crear sitios que funcionaran en más de un navegador o sistema operativo. En la actualidad sería incorrecto ya que los navegadores modernos admiten los mismos estándares abiertos. Aun así, la práctica persiste, consumiendo innecesariamente recursos escasos, fragmentando la Web y generando sitios inaccesibles e inútiles.

El problema de las versiones

La creación de varias versiones de marcado no estándar y de código no estándar, cada una en función del comportamiento no estándar de un determinado navegador u otro, es la causa de la perpetua obsolescencia que inunda la mayoría de los sitios y a sus propietarios. Los postes de la portería se alejan y las reglas del juego sufren un cambio continuo.

Aunque resulta costosa, fútil e insostenible, la práctica persiste incluso cuando no resulta necesaria. Al enfrentarse a un navegador que admite estándares, muchos programadores lo tratan como si no lo hiciera. Por esta razón, escriben secuencias de comandos de detección que husmean en IE6 y le añaden secuencias de comandos exclusivas de Microsoft aunque IE6 puede procesar ECMAScript estándar y el DOM. Tras ello, se ven obligados a escribir secuencias de comandos de detección especiales (y código especial) para navegadores Netscape modernos que también pueden procesar ECMAScript estándar y el DOM.

Como sugiere el ejemplo, gran parte de la búsqueda de navegadores y dispositivos, y gran parte de la creación de versiones individuales resulta innecesaria en el entorno actual compatible con estándares. De hecho, es peor que innecesaria. Incluso con actualizaciones constantes, que no todos los propietarios de sitios se pueden permitir, las secuencias de comandos de detección pueden fallar.

Por ejemplo, en Windows, el navegador Opera se identifica a sí mismo como Explorer. Lo hace para evitar ser bloqueado por los sitios (como sucede con muchos sitios de entidades bancarias) que buscan IE. Pero es muy probable que las secuencias de comandos escritas exclusivamente para navegadores IE fallen en el navegador Opera. Cuando Opera se identifica como IE (la configuración predeterminada al realizar la instalación) y cuando los programadores escriben secuencias de comandos sólo para IE, el error del sitio y la frustración de los usuarios son evidentes. Los usuarios pueden cambiar su cadena de agente usuario (AU) y obligar a que Opera se identifique honestamente en lugar de intentar pasar por IE. Pero no hay muchos usuarios que conozcan esta opción y tampoco deberían.

Además de las secuencias de comandos propietarias, los programadores diseñan marcado de presentación que duplica el

ancho de banda necesario para ver u ofrecer una página al tiempo que reduce su accesibilidad para motores de búsqueda y navegadores, y dispositivos no tradicionales. A menudo estas estrategias provocan el mismo problema que pretenden resolver: la representación inconsistente entre un navegador y otro (véase figura 1.1).

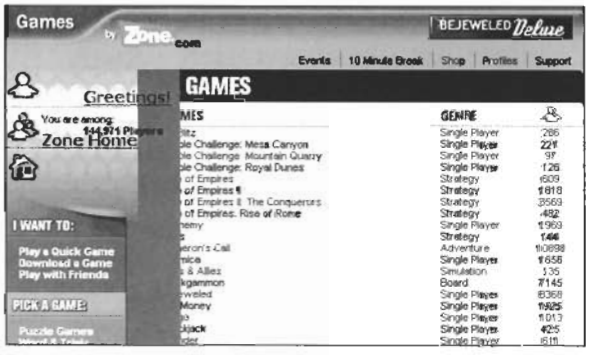


Figura 1.1.

MSN Game Zone (zone.msn.com/blog.asp) utiliza siete hojas de estilo externas y aun así no se muestra correctamente en la mayoría de los navegadores modernos. También emplea 14 secuencias de comandos (la mayoría en línea), incluyendo una potente detección de navegador. Sigue sin funcionar. El uso de varias versiones de código para solventar un problema no suele ser la solución.

Con el uso de varias versiones aumentan los costes y los problemas. Los sitios DHTML producidos para las especificaciones de secuencia de comandos propietarias (e incompatibles) de Netscape 4 e IE4 no funcionan en la mayoría de los navegadores modernos. ¿Debería el propietario del sitio invertir más para solucionar el problema y pedir a los programadores que creen un quinta o sexta versión del sitio? ¿Qué ocurre si no hay presupuesto para dicha versión? Muchos usuarios quedarán excluidos.

Del mismo modo, los programadores pueden dedicar gran cantidad de tiempo y de recursos a una versión "inalámbrica" del sitio, sólo para descubrir que el lenguaje de marcado inalámbrico que han utilizado se ha quedado obsoleto o que su versión inalámbrica falla en un nuevo dispositivo. Algunos solucionan este último problema creando otra nueva versión. Otros publican embarazosos mensajes en los que prometen admitir el nuevo dispositivo lo antes posible.

Incluso cuando recurren a tecnologías de estándares Web como XHTML y CSS, los diseñadores y programadores acostumbrados a métodos antiguos no captan la diferencia. En lugar de utilizar estándares para evitar el uso de múltiples versiones, muchos programadores crean varios archivos CSS específicos para un navegador o una plataforma que se caen por su propio peso (véanse figuras 1.1 y 1.2).

Estas prácticas suponen un gasto de tiempo y dinero. Ninguno de estos factores abunda y se han visto especialmente reducidos desde el inicio de la recesión económica occidental. Y lo que es peor, estas costosas prácticas no han contribuido a solucionar el problema. Los sitios siguen sin funcionar y los usuarios siguen bloqueados.

Pensamiento inverso

Diseccione cualquier sitio importante de la era del 2003, desde Amazon hasta Microsoft.com, desde Sony a ZDNet.

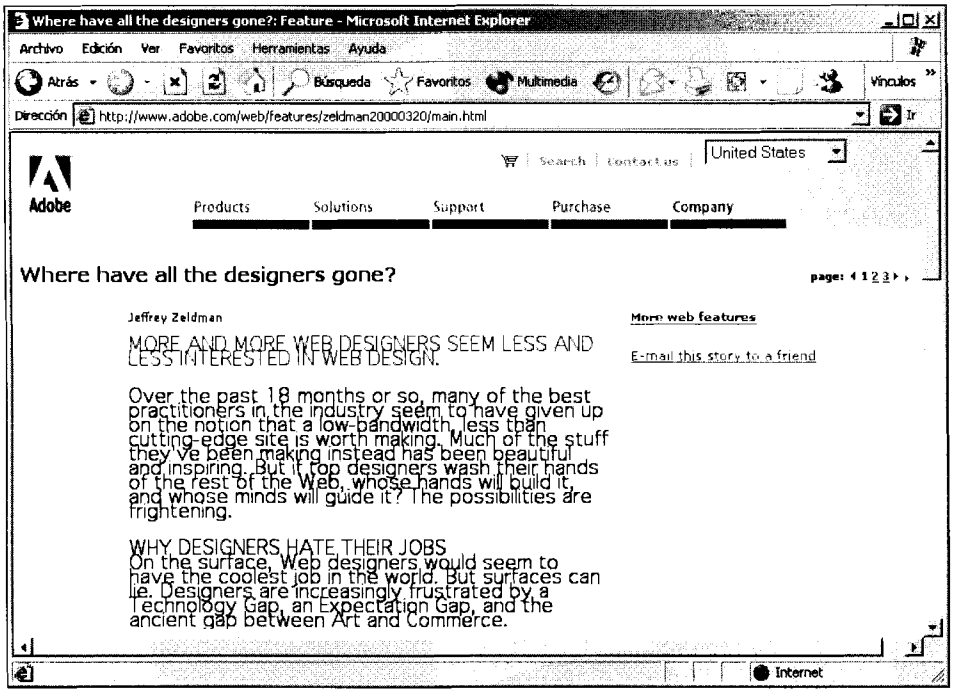


Figura 1.2.

El contenido del sitio Web de Adobe (www.adobe.com) sufre de un interlineado incorrecto y hace que las líneas de texto se solapen. En lugar de crear una o dos hojas de estilo vinculadas que funcionen en todos los navegadores, los programadores se vanagloriaron y crearon estilos específicos de un navegador y una plataforma que interactúan de forma pobre. (Al cierre de esta edición, Adobe ya había solventado este problema.) Los diseñadores acostumbrados a métodos antiguos no entienden que suele bastar con CSS y XHMTL para eliminar la necesidad del uso de varias versiones.

Examine su tortuoso marcado no estándar, su ActiveX y JavaScript propietarios (que a menudo incluyen secuencias de comandos de detección incompletas) y su uso de CSS, en caso de que las utilicen. Es increíble que estos sitios puedan funcionar en un navegador.

Estos sitios funcionan en los principales navegadores antiguos ya que las primeras cuatro o cinco generaciones de Netscape Navigator y de Microsoft Internet Explorer apenas toleraban el marcado no estándar y el código específico del navegador; de hecho, alentaban un diseño y un uso de secuencias de comandos propietarias descuidado en una batalla por dominar el espacio del navegador.

A menudo, los sitios no compatibles con estándares funcionan en navegadores antiguos porque sus propietarios han invertido en costosas herramientas de publicación que solventan las diferencias entre navegadores

mediante la generación de múltiples versiones no estándar acordes a las condiciones de determinados navegadores y plataformas, como mencionamos en un apartado anterior. Este método agota la paciencia del usuario ya que malgasta ancho de banda al dividir el código, al utilizar tablas profundamente anidadas, píxeles espaciados y otros problemas de las imágenes, así como con etiquetas y atributos desfasados o inválidos.

División del código

El código es lo que escriben los programadores para crear productos de software, sistemas operativos o prácticamente cualquier cosa de nuestro mundo digital. Cuando más de un grupo de programadores trabaja en un proyecto, el código se puede "dividir" en varias versiones incompatibles, especialmente si cada grupo de programación intenta resolver un problema diferente o se ajusta a una planificación distinta. Esta inconsistencia y falta de control centralizado suele resultar negativa.

En este libro, al utilizar división de código, nos referiremos al proceso de crear múltiples versiones de código incompatible para cumplir las necesidades de los navegadores que no admiten ECMAScript estándar y el DOM (consulte un apartado anterior).

Al mismo tiempo, esta multiplicidad de versiones agota el ancho de banda del propietario del sitio, lo que supone un coste excesivo. Cuanto mayor es un sitio y más tráfico tiene, más dinero se gasta en llamadas al servidor, redundancias, problemas con imágenes, código y marcado innecesariamente complejo.

Las cifras negativas resultan difíciles de corregir, pero si un sitio reduce su marcado en un 35 por ciento, los costes de ancho de banda se reducirán en la misma cantidad. Una organización que gaste 2.500 € al año se ahorraría 875 €. Otra que gaste 160.000 €, se ahorraría 56.000 €.

La página inicial de Yahoo (véase figura 1.3) tiene millones de accesos al día. Cada byte que se malgasta en problemas de diseño de HTML obsoleto se multiplica por un número astronómico de visitas de páginas, lo que genera gigabytes de tráfico que sufren los servidores de Yahoo y que añade costes abismales a su sobrecarga. Si simplemente Yahoo cambiara sus desfasadas etiquetas `<font>` que se alimentan de ancho de banda (véase figura 1.4), con CSS positivo para el ancho de banda, el coste de servir las páginas se reduciría considerablemente y, en consecuencia, aumentarían los beneficios de la empresa. Se preguntará por qué no ha realizado este cambio.

Únicamente podemos deducir que la empresa quiere que su sitio se vea exactamente igual en navegadores de 1995 que no admiten CSS como lo hace en los más modernos. La ironía es que a nadie, aparte de la dirección de Yahoo, le importa el aspecto que tiene.

El tremendo éxito del sitio se debe al servicio que ofrece, no a la belleza visual de su diseño (que es inexistente).

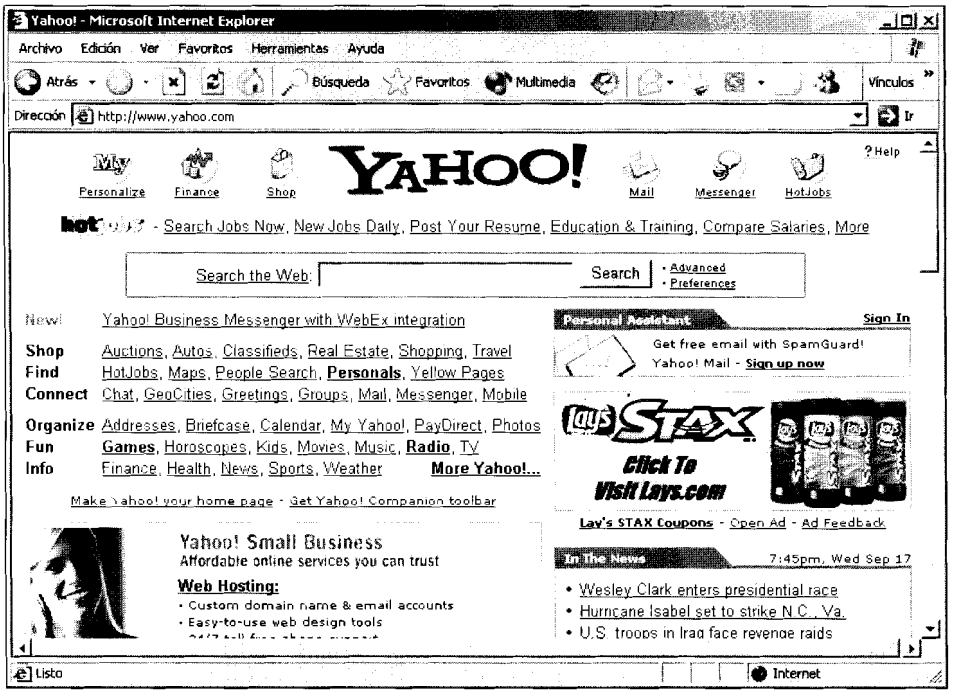


Figura 1.3.

Aspecto de Yahoo (www.yahoo.com). Sencillo.

Que esta empresa gasta una cantidad increíble de ancho de banda para ofrecer un aspecto operativo y funcional que nadie admira, realmente dice todo lo que debe saber sobre la estrecha forma de pensar de los programadores que se decantan por una compatibilidad inversa en lugar de por la razón, el uso y sus propios beneficios.

Marcado obsoleto: el coste para los propietarios de sitios

Imagine que el código y el marcado de una página Web de la vieja escuela ocupa 60 K. Dicho esto, al reemplazar las anticuadas etiquetas `<font>` y demás basura de presentación y propietaria por un marcado limpio y estructural, y algunas reglas CSS, la misma página puede reducirse a 30 K. (En mi agencia, podemos cambiar 60 K de marcado por 22 K o menos. Pero nos quedaremos con esta cifra más conservadora que supone un ahorro en ancho de banda del 50 por ciento). Veamos una serie de ejemplos.

Terminator T1

Supuesto: Un sitio Web de una pequeña empresa autogestionada o del sector público tiene un continuo flujo de visitantes, en determinados momentos de varios cientos. Después de reducir el tamaño de la página a la mitad y convertir el marcado de presentación a XHTML estructural, la organización se ahorra 1.500 € al mes.

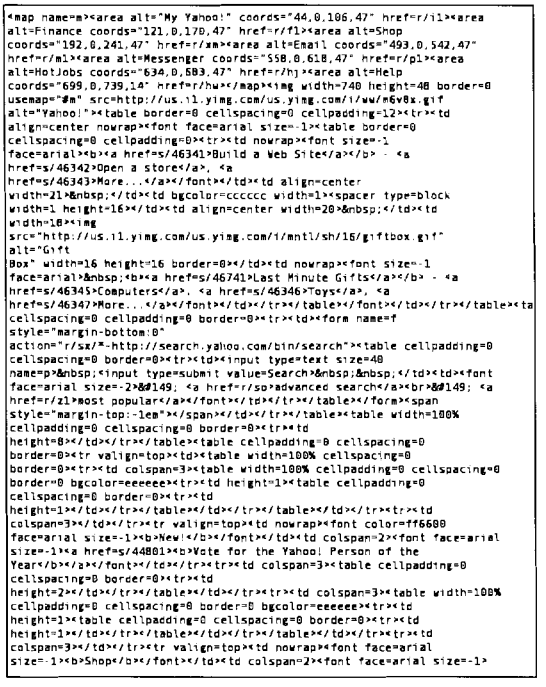


Figura 1.4.

Diseccione una página de Yahoo (Ver>Código fuente) y descubrirá que el código y el marcado utilizados para crear este sitio Web de aspecto tan simple es increíblemente complejo y rebuscado.

Funcionamiento: Para servir a su público antes de la conversión, el sitio autogestionado necesitaba dos líneas T1, cada una con un coste de 1.500 € al mes (coste normal de una línea T1 de 1,544 megas por segundo). Después de reducir el tamaño de los archivos en un 50 por ciento, la organización descubre que puede bastar con una sola línea T1, lo que elimina 1.500 € de los gastos operativos mensuales. Además de ahorrar ancho de banda, tendrán menos gastos en hardware. Cuando más sencillo sea el marcado, con mayor rapidez llegará al usuario. Cuanto antes llegue, el servidor sufrirá una menor carga, y se necesitarán menos servidores que comprar, servir y reemplazar, especialmente en el caso de servidores que procesan contenidos dinámicos dirigidos por bases de datos, es decir, todos los sitios comerciales y con contenidos más modernos.

Metered Megabytes

Supuesto: Con el aumento de la popularidad de un sitio alojado comercialmente, sus propietarios se ven obligados a pagar una inesperada multa de transferencia de archivos cada mes, de cientos o incluso miles de euros. Al reducir a la mitad el tamaño de los archivos, la cantidad mensual que deben abonar se convierte de nuevo en una cuota asequible y razonable.

Funcionamiento: Muchos servicios de alojamiento comercial asignan a sus usuarios una determinada cantidad mensual "gratuita" de ancho de banda para transferencia de archivos, por ejemplo, 3 GB. Si no llega a esa cantidad, pagará la cuota mensual habitual. Si la supera, tendrá que pagar más, en ocasiones mucho más.

En uno de los casos más infames que se recuerda, la empresa de alojamiento Global Internet Solutions cobró al diseñador independiente Al Sacui 16.000 € adicionales después de que su sitio no comercial `Nosepilot.com` superara el límite mensual permitido. Es un ejemplo extremo. Sacui pudo evitar pagar argumentando que el servicio de alojamiento había cambiado sus condiciones sin notificárselo a sus clientes (http://thewebfairy.com/gisol/), pero después de una dura batalla legal. ¿Quién quiere arriesgarse a pagar cuentas desorbitadas o enfrentarse en batallas legales a una empresa de alojamiento?

Evidentemente, no todas las empresas de alojamiento cobran estas cantidades excesivas por el exceso de transferencia de archivos. Por ejemplo, `Pair.com`, cobra actualmente 1'5 céntimos por megabyte excedido. Un sitio de pequeño tamaño alojado en Pair con bajo tráfico puede que se ahorre 200 € al año. Son los sitios de mayor tamaño con un tráfico superior los que más se ahorran al reducir el tamaño de los archivos. Ya se trate de sitios de mayor o menor tamaño, con millones de visitantes o con sólo unos pocos, cuanto más reducidos sean los archivos, menor será el ancho de banda y menor será la probabilidad de sufrir los costes de transferencia de archivos de su empresa de alojamiento. Por cierto, conviene decantarse por una empresa de alojamiento que permita transferencias de archivos ilimitadas en vez de una que le penalice por crear un sitio con gran aceptación.

Marcado condensado frente a marcado comprimido

Después de ofrecer una conferencia sobre estándares Web, se me acercó un programador del público que sostenía que las ventajas en ancho de banda que suponía el uso de marcado limpio y bien estructurado apenas suponía diferencia alguna para las empresas que comprimían su HTML.

Además de condensar el marcado al escribirlo de forma limpia (es decir, utilizando estructuras semánticas en lugar de métodos de diseño HTML anticuados), puede comprimir digitalmente el marcado en algunos entornos de servidores. Por ejemplo, el servidor Web Apache incluye un módulo `mod_zip` que reduce el HTML en el lado del servidor. En el navegador del usuario, el código HTML se vuelva a expandir.

El programador con el que hablé me puso el siguiente ejemplo. Si `Amazon.com` gasta 40 K en etiquetas de fuente anticuadas y demás basura pero utiliza `mod_zip` para comprimirlo hasta 20 K, el marcado de Amazon representaría un gasto menor que lo que sugería mi conferencia (y este libro).

En realidad, Amazon no usa `mod_zip`. De hecho, esta herramienta apenas se utiliza en Web comerciales, posiblemente debido a la carga adicional necesaria para comprimir páginas antes de enviarlas. Dicho esto, cuanto menor sea el tamaño del archivo, menor será la compresión. Si se ahorra dinero al comprimir una página de 80 K a 40 K, se ahorrará más si la reduce de 40 K a 20 K. El ahorro en una sola sesión de visita de páginas puede parecer mínimo, pero su valor es acumulativo. Con el tiempo, se pueden reducir considerablemente los costes operativos y evitar gastos adicionales (por ejemplo, puede que no sea necesario contratar líneas T1 adicionales para superar los problemas de ancho de banda).

El ahorro de ancho de banda es una de las ventajas del uso de marcado limpio y bien estructurado, pero una de las que los clientes y los contables aprecian, lo que también es aplicable tanto para los que comprimen su código HTML como para el resto de nosotros.

Compatibilidad inversa

Si pregunta a un programador lo que quiere decir con el término compatibilidad inversa le dirá que significa admitir a todos los usuarios. Nadie puede poner en duda un argumento como éste.

Sin embargo, en la práctica, significa el uso de marcado y código no estándar propietario (u obsoleto) para garantizar que todos los usuarios ven lo mismo, utilicen IE2 o Netscape 7. Considerada el Santo Grial de la programación profesional, la compatibilidad inversa parece perfecta en teoría. Pero el coste es demasiado elevado y la práctica siempre se ha basado en una mentira.

No existe la verdadera compatibilidad inversa. Siempre hay un punto de interrupción. Por ejemplo, ni Mosaic (el primer navegador visual) ni Netscape 1.0 admiten diseños basados en tablas HTML. Por definición, los que utilicen estos navegadores antiguos no tendrían la misma experiencia visual que los que vean la Web a través de navegadores menos antiguos como Netscape 1.1 o bien MSIE2.

Los programadores y clientes que sostienen el apoyo a la compatibilidad inversa inevitablemente especifican un navegador base, como por ejemplo Netscape 3 y confiesan que es el navegador más antiguo compatible con sus sitios (los usuarios de Netscape 2 no están de enhorabuena). Para cumplir su compromiso de compatibilidad con un navegador base, los programadores dividen su marcado con diferentes elementos no estándar específicos del navegador que aumentan el peso de la página.

Al mismo tiempo, los programadores escriben múltiples secuencias de comandos para que los navegadores que han seleccionado admitan y utilicen detección de navegadores para añadir a cada navegador el código más indicado. De esta forma, se incrementa más el tamaño de las páginas, la carga de los servidores y se garantiza que la carrera contra la obsolescencia perpetua continúa hasta que se queden sin fondos o sus empresas quiebren.

El bloqueo de usuarios es malo para el negocio

Mientras que algunas empresas reducen sus propios beneficios para garantizar que sus sitios se vean exactamente igual en los navegadores más antiguos que en los nuevos, otras han decidido que sólo importa un navegador. En un esfuerzo equivocado por reducir gastos, se ha diseñado un mayor número de sitios para que sólo funcionen en Internet Explorer y, en ocasiones, sólo en Windows, lo que bloquea aproximadamente entre un 15 y un 25 por ciento de los clientes y visitantes potenciales (véanse figuras 1.5, 1.6, 1.7, 1.8 y 1.9).

No pretendemos que comprenda el modelo empresarial de una compañía que rechazaría un cuarto de sus clientes potenciales. Y el número de clientes perdidos por este enfoque tan limitado sobresalta a cualquier empresario racional o agente no corporativo con un cierto interés en servir al público. Según los datos estadísticos recopilados por NUA Internet Surveys (www.nua.ie/surveys), más de 650'6 millones de personas utilizaron la Web en septiembre del 2002. Haga las cuentas.

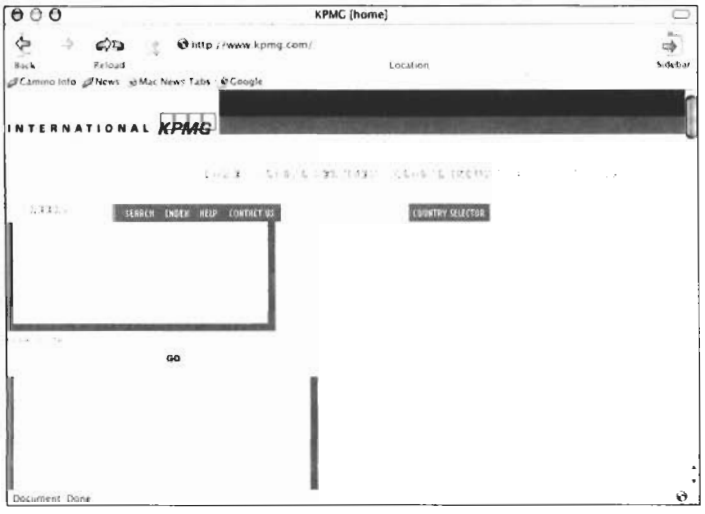


Figura 1.5.

Página inicial de KMPG (www.kmpg.com) como se ve en Navigator. O mejor, como no se ve en Navigator, gracias al estúpido código sólo para IE.

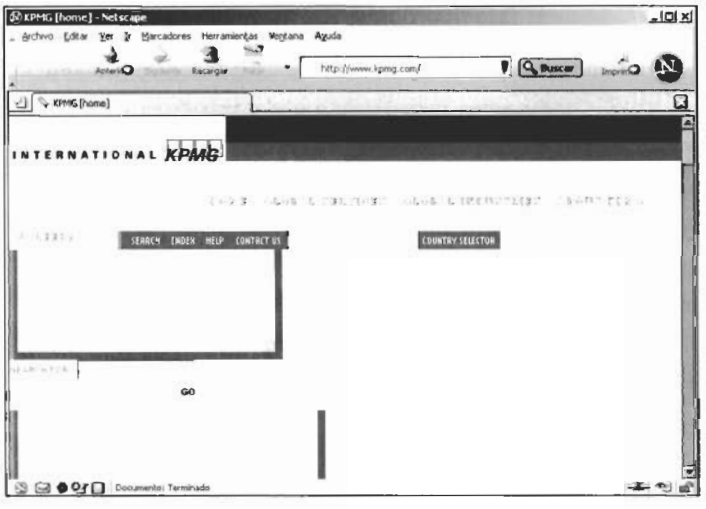


Figura 1.6.

KMPG resulta igual de inútil en Netscape 7. Suponemos que a la empresa tampoco le importan estos clientes.

Figura 1.7.

Bien, si este sitio es sólo para IE5, lo probaremos en IE5 Macintosh Edition. Tampoco funciona para estos usuarios (el sitio funciona en IE5/Macintosh la mitad de las ocasiones y falla en las restantes, sin motivo aparente).

Figura 1.8.

El mismo sitio visto en IE6/Windows, dónde finalmente parece funcionar.

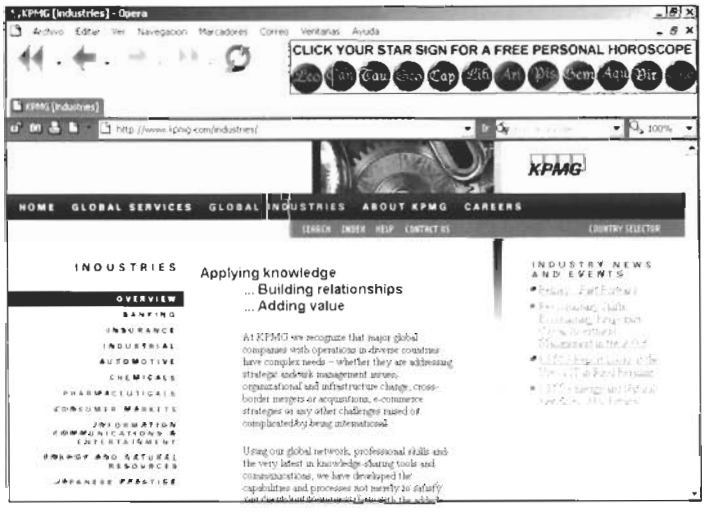


Figura 1.9.

Para ser honestos, el sitio parece funcionar con Opera 7 para Windows, cuando se identifica como IE (si se identifica como Opera, el sitio no funciona).

Imagine que no le importa perder un 25 por ciento de los usuarios que puedan visitar su sitio. El enfoque sólo para IE sigue sin tener sentido ya que no garantiza que IE (ni siquiera la categoría de navegadores de escritorio como tal) mantenga su dominio del espacio Web.

Algunos años antes de escribir este libro, el navegador Navigator de Netscape gozaba de una cuota de mercado mayor que la que Internet Explorer tiene actualmente. En aquel momento, se suponía que era el único navegador que importaba y los programadores diseñaban el código correspondiente para el mismo. Un número indeterminado de millones de dólares después, el mercado sufrió un cambio. Los sitios sólo para Netscape se desecharon en la cuneta digital de la superautopista de la información por la que viajaban.

¿Tendrán los sitios sólo para IE el mismo destino? Aunque parezca inconcebible, no se

puede saber. En la Web, lo único constante son los cambios. El creciente uso de dispositivos de Internet no tradicionales y la tendencia a diseñar en función de un determinado navegador de escritorio a expensas del resto de navegadores y dispositivos empieza a parecer una decisión empresarial inútil, lo que realmente es.

Además, como comprobará en este libro, los estándares permiten que el diseño para todos los navegadores y dispositivos sea tan sencillo como para uno solo. Entre el coste de crear versiones con compatibilidad inversa y la fútil falta de previsión que conlleva el diseño para un solo navegador, los estándares Web constituyen el único enfoque de programación con cierto sentido.

Ni las técnicas de creación de versiones que suponen una pérdida de dinero ni la decisión deliberada de admitir un solo navegador o plataforma contribuirán a que los sitios actuales funcionen en los navegadores del futuro o prosperen en el mundo cambiante que se encuentra más allá del escritorio. Si estas prácticas continúan, los costes y la complejidad aumentarán de forma que sólo las grandes empresas puedan permitirse el lujo de crear sitios Web.

En nuestro esfuerzo por ofrecer las mismas posibilidades en entornos de navegación incompatibles, para que la Web parezca impresa y funcione como un software de escritorio, hemos perdido su verdadero potencial como medio completo y multifuncional accesible para todos.

Lo perdimos cuando los diseñadores y programadores, en su intento por mantenerse al día de las demandas de producción durante el breve boom de Internet, aprendieron formas de creación de sitios no estándar específicas de los navegadores, lo que nos devuelve de nuevo a la obsolescencia.

Pero mientras lee estas líneas, el periodo de obsolescencia está llegando a su fin, llevándose consigo multitud de sitios. Si posee, administra, diseña o genera sitios Web, está de enhorabuena.

Camino a Villaidiotez

A principios de 1997, era muy común escribir JavaScript para navegadores de Netscape y JScript (un lenguaje similar a JavaScript) para navegadores de Microsoft. También era muy habitual utilizar JavaScript (que sólo funcionaba en Netscape) y ActiveX (que sólo funcionaba en IE/Windows) para enviar a cada navegador el código que necesitaba. Es lo que hicimos con los navegadores 3.0.

Esta práctica no resultó positiva en absoluto para los navegadores menores como Opera y no funcionaba correctamente para los usuarios de Internet Explorer en Macintosh, aunque sí para la "mayoría" de los usuarios, razón por la que rápidamente se convirtió en la norma de la industria. Si queríamos crear páginas Web activas que hicieran algo más y tuvieran un aspecto correcto, no teníamos otra opción que seguir estos procedimientos.

Ese mismo año, aparecieron los navegadores 4.0 para Netscape y Microsoft, cada uno de ellos haciendo gala de potentes opciones de HTML dinámico (DHTML) que, como habrá adivinado, eran completamente incompatibles entre sí. También eran incompatibles

con versiones anteriores de los mismos productos (lo que funcionaba en Netscape 4 no lo hacía en Netscape 3), sin mencionar que eran totalmente incompatibles con otros navegadores menores que dócilmente admitían especificaciones básicas como HTML en lugar de crear sus propios lenguajes y atributos.

Se preguntará si todo esto era lo correcto. Para Netscape y Microsoft sí lo era, algo que también pensaban muchos diseñadores y programadores. Aquéllos que no estaban de acuerdo no tenían otra opción que apretar los dientes y quejarse de las versiones necesarias para crear un sitio aceptablemente "profesional".

¿Cómo lo codifico? Las diferentes técnicas

Había DHTML para Netscape 4. Había DHTML incompatible para Internet Explorer 4 que prácticamente sólo funcionaba en Windows. No había JavaScript que no fuera DHTML para Netscape 3 y código de Microsoft que no fuera DHTML para IE3. Podía haber código adicional para navegadores menores y anteriores o puede que no lo hubiera. En definitiva, incluso en la página Web menos interesante se necesitaban más tenedores que en un restaurante italiano.

Algunos programadores se limitaban personalmente a dos versiones (una para IE4 y otra para Netscape 4) y exigían que los usuarios utilizaran un navegador 4.0. Otros, con presupuestos más reducidos, lo apostaban todo a un navegador y normalmente perdían la apuesta (y a veces la camisa).

El Web Standards Project, que apareció después de que lo navegadores 4.0 llegaran al mercado, estimó que la necesidad de escribir cuatro o más versiones incompatibles de todas las funciones aumentaba en un 25 por ciento el coste de diseño y programación de cualquier sitio Web, coste que normalmente soportaba el cliente.

Algunos programadores respondieron a esta afirmación encogiéndose de hombros. La Web era atractiva, atractiva, atractiva y los clientes estaban dispuestos a pagar, pagar y pagar. Entonces, ¿por qué las grandes agencias Web se preocupaban por el elevado coste de las distintas versiones de código y de marcado? Entonces estalla la burbuja de Internet, los presupuestos Web se hunden o se congelan y las agencias empiezan a desaparecer o a disminuir en número. De repente, prácticamente nadie se podía permitir económicamente las distintas versiones de código.

Mientras la industria se tambaleaba por los despidos y los cierres, apareció una nueva generación de navegadores que admitían un DOM común, el que había creado el W3C. ¿Qué significó este desarrollo? Significaba la desaparición de las versiones y que una nueva era de diseño y programación basados en estándares estaba al llegar. ¿Cómo respondió nuestra economía a estas noticias tan deseadas? Manteniendo la división de código, programando sólo para IE/Windows o cambiando a Macromedia Flash.

Para un negocio repleto de visionarios, la industria Web puede resultar especialmente oscura.

Cosas buenas para marcado malo

Al inicio de la educación de un programador informático, aprende la frase "Si se guarda basura, se tiene basura". Los lenguajes como C y Java no sólo recomiendan una práctica de codificación adecuada, lo exigen.

Del mismo modo, entre lo primero que aprende un diseñador es que la calidad de los materiales originales determina la eficacia del producto final. Si se empieza con una fotografía de gran calidad a alta resolución, el aspecto impreso o del grafico Web será el correcto. Si intenta diseñar con una instantánea de baja calidad o con una imagen Web a baja resolución, el resultado final no estará a la altura. Se puede convertir un EPS de gran calidad en un logotipo de una página Web optimizada correctamente, pero no se puede convertir un GIF a baja resolución en una Web de alta calidad, una copia impresa o un logotipo de televisión.

Pero los principales navegadores tradicionales no funcionan de esta forma. Descuidados hasta el punto del absurdo, engullen marcado dividido y vínculos erróneos a archivos originales de JavaScript sin pestañear y, en la mayoría de los casos, muestran el sitio como si se hubiera creado correctamente. Esta laxitud ha animado a diseñadores y programadores de interfaces de usuario a desarrollar malos hábitos que prácticamente desconocen. Al mismo tiempo, ha persuadido a programadores de software intermedio y de servidor para que vean tecnologías como XHTML, CSS y JavaScript como algo primitivo.

Es poco probable que aquéllos que no respetan una herramienta la utilicen correctamente. Analicemos el siguiente fragmento de código, obtenido de un costoso sitio de comercio electrónico de una empresa que compite en un mercado feroz y que reproducimos aquí en toda su supuesta gloria:

```
<td width="100%"><ont face=
"verdana,helvetica,arial" size="+1"
color="#CCCC66"><span class="header">
<b>Join now!</b></span>_</ont></td>
```

La etiqueta `<ont>` es un errata de la anticuada etiqueta `<font>`, una errata que se repite miles de veces a lo largo del sitio, gracias a una eficaz herramienta de publicación. Aparte de este error, el marcado puede resultarle familiar. Incluso puede parecerse al marcado de su sitio. En el contexto de esta página Web, realmente bastaría con lo siguiente:

```
<h3>Join now!</h3>
```

Junto con la regla correcta de una hoja de estilo, el marcado anterior, más sencillo y más estructural, sirve exactamente para lo mismo que el complicado marcado no estándar e inválido, al tiempo que ahorra ancho de banda para servidores y usuarios, y facilita la transición a un sitio más flexible que se centre en marcado basado en XML. El mismo sitio de comercio electrónico incluye el siguiente vínculo dividido de JavaScript:

```
<script language=JavaScript1.1src=
"http://foo.com/Params.richmedia=
yes&etc"></script>
```

Entre otros problemas, el atributo de lenguaje se mezcla erróneamente con la etiqueta original, es decir, al navegador se le indica que utilice un lenguaje de secuencia de comandos inexistente ("JavaScript1.1src").

De forma racional, el sitio tendría que fallar, alertar a los programadores de su error e instarles a que lo solucionen. Hasta hace poco, el JavaScript de este sitio funcionaba en navegadores importantes, con lo que se perpetuaba el ciclo de sitios incorrectamente diseñados y los navegadores que los utilizaban. No hay que extrañarse de que los programadores con experiencia consideren a la programación de interfaces de usuario algo que no respetan y que no les importa.

El marcado basura puede perjudicar seriamente la salud de un sitio

Cuando los nuevos navegadores cumplen con los estándares Web, aumenta el rigor en lo que respecta a lo que esperan de diseñadores y programadores, así como la intolerancia con respecto a código y marcado dividido. La máxima "Si se guarda basura, se tiene basura" empieza a adueñarse del mundo de los navegadores, por lo que el conocimiento de los estándares Web se hace necesario para todo el que quiera diseñar o producir sitios Web.

El daño no es irreparable. Podemos diseñar y generar sitios Web que funcionen en diferentes navegadores, plataformas y dispositivos, y solucionar los problemas de obsolescencia incorporada y bloqueo de usuarios al tiempo que preparamos el camino hacia una Web desarrollada de forma más racional, potente y accesible.

El remedio a la enfermedad de la obsolescencia incorporada se puede encontrar en un conjunto básico de tecnologías comúnmente admitidas que, de forma colectiva, se denominan, estándares Web. Al aprender a diseñar y generar con estándares Web, podemos garantizar la compatibilidad directa de todos los sitios que creemos.

La promesa de los estándares Web de escribir una vez y publicarlo en todas partes es más que un deseo: se está haciendo realidad en la actualidad, con el uso de los métodos que analizaremos en este libro. Aunque los actuales navegadores punteros admiten finalmente estos estándares y métodos, el mensaje no ha llegado todavía a muchos diseñadores y programadores, y muchos sitios nuevos se siguen levantando sobre las arenas movedizas del marcado y el código no estándar. Esperemos que con este libro todo cambie.

El remedio

Después de una larga lucha que ha enfrentado a diseñadores y programadores contra los fabricantes de los principales navegadores, por fin podemos emplear técnicas que garanticen el aspecto y el comportamiento de nuestros sitios, no sólo en el navegador de un determinado fabricante, sino en todos.

Promovidas por los miembros del W3C y de otros organismos de estándares, y admitidas en navegadores actuales desarrollados por Netscape, Microsoft, Opera y otras empresas, tecnologías como CSS, XHTML, ECMAScript (la versión estándar de JavaScript) y el DOM W3C permiten a los diseñadores realizar las siguientes tareas:

- Tener un mayor control sobre aspectos de diseño, disposición y tipografía de navegadores gráficos de escritorio, al tiempo

que permiten a los usuarios modificar la presentación para ajustarla a sus necesidades.

- Desarrollar comportamientos sofisticados que funcionen en diferentes navegadores y plataformas.
- Cumplir las normas y directrices de accesibilidad sin sacrificar el aspecto visual, el rendimiento o la sofisticación.
- Admitir varios navegadores sin preocuparse de crear diferentes versiones, a menudo sin apenas división de código.
- Admitir dispositivos no tradicionales, desde dispositivos inalámbricos y teléfonos móviles compatibles con la Web objeto de deseo de adolescentes y ejecutivos, hasta lectores Braille y de pantalla utilizados por discapacitados, sin preocuparse de crear diferentes versiones.
- Conseguir sofisticadas versiones impresas de cualquier página Web, a menudo sin necesidad de crear versiones de la página "aptas para impresión", ni de depender de carísimos sistemas de publicación propietarios para crear dichas versiones.
- Separar estilo de estructura y comportamiento, con lo que se consiguen creativos diseños complementados por una rigurosa estructura documental y se facilita la modificación de la función de documentos Web en flujos de trabajo de publicación avanzados.
- Realizar la transición entre HTML, el lenguaje Web del pasado, al marcado basado en XML del futuro, mucho más potente.
- Garantizar que los sitios diseñados y creados de esta forma funcionen correctamente en los navegadores actuales compatibles con estándares y lo hagan de forma aceptable en los antiguos (incluso si no se representan píxel a píxel de la misma forma).
- Garantizar que los sitios diseñados de esta forma funcionarán en los futuros navegadores y dispositivos, incluyendo aquéllos que todavía no se han creado. Es la promesa de la compatibilidad directa.
- ... y mucho más, como comprobará en este libro.

Antes de que aprenda a conseguir estos objetivos con ayuda de los estándares, es necesario analizar los métodos antiguos que se verán sustituidos para saber exactamente cómo estas técnicas desfasadas perpetúan el ciclo de obsolescencia, como veremos en el siguiente capítulo.

Capítulo 2
Diseño y creación con estándares

Se preguntará cómo se las apañaban los diseñadores y programadores para crear sitios antes de que se inventarán los estándares Web y antes de que los navegadores los admitieran. Como podían. Tomemos como ejemplo `Suck.com`, uno de las primeras publicaciones periódicas independientes de la Web (véase figura 2.1). Utilizaba una afilada forma de escribir y sabía cómo mostrar su contenido diario en primera página, para que los lectores no se lo perdieran. Hoy en día parece obvio, pero a mediados de los 90, cuando apareció Suck, la mayoría de los sitios sepultaban sus contenidos entre la página de introducción, las páginas de inicio, las de bienvenida, las declaraciones de intenciones y los confusos índices de contenidos.

El énfasis de Suck en el texto directo fue un soplo de aire fresco en una época en la que la mayoría de los sitios comerciales envolvían sus contenidos en excesivas metáforas literarias. Del mismo modo, el escueto y austero aspecto operativo y visual de Suck destacaba en una época en la que muchos sitios eran ejercicios excesivos de diseño entre biseles metálicos y tecnología. Brillos góticos y partes sin diseñar confluían gracias a administradores de sistemas y autores HTML autodidactas. En aquel momento, muchos sitios utilizaban todos los dispositivos que ofrecía Netscape 1.1 a los aspirantes a artistas, desde el repetitivo mosaico de fondo hasta la etiqueta `<center>` propietaria, y en algunos sitios todavía persisten estas técnicas (véase figura 2.2). En un entorno en el que primaba cuanta más cantidad mejor, Suck destacaba por atreverse a hacer menos.

Para conseguir el original y austero aspecto centrado en contenidos de Suck, los autores

Carls Steadman y Joey Anuff tuvieron que pasar por algunos aros. HTML carecía de herramientas de diseño, por una buena razón. Tal y como lo concibió Tim Berners-Lee, el físico que inventó la Web, HTML era un lenguaje de marcado estructurado (http://www.w3.org/MarkUp/html-spec) derivado de SGML, no un lenguaje de diseño como PostScript de Adobe o el estándar de hojas de estilo en cascada (todavía no se había aprobado CSS como recomendación W3C y, tras aprobarlo, pasarían cuatro largos años hasta que los navegadores lo admitirán como si se tratara de una especificación).

Se preguntará cómo controlaron Steadman y Anuff la presentación de su sitio. Lo hicieron con creatividad, inventiva y muchos metros de cinta aislante digital.

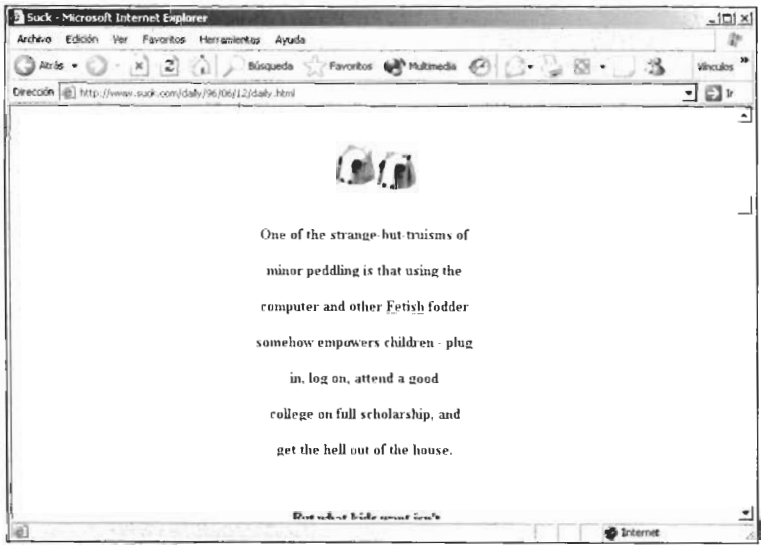


Figura 2.1.

Un sitio decididamente brillante en los albores de la Web comercial (www.suck.com).

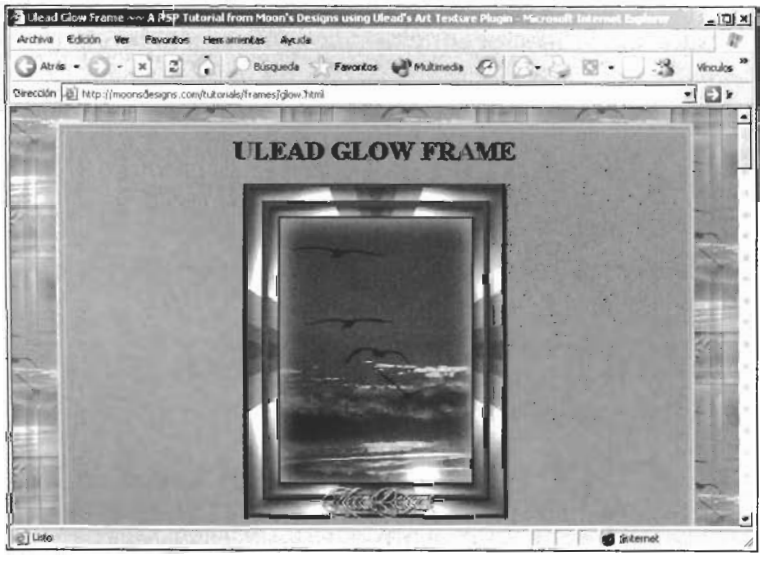


Figura 2.2.

El curso práctico Ulead Glow Frame de Moon's Design (moondesigns.com) recupera el diseño Web típico de mediados de los 90. Los contenidos se centran en una tabla HTML que también está centrada. Se aplica un repetitivo mosaico de fondo a la tabla y otro a la página en la que ésta se incluye.

Pasar por el aro

Para crear el aspecto de Suck, Steadman y Anuff escribieron una secuencia de comandos de Perl que contaba los caracteres del texto y que añadía una etiqueta de párrafo <p> como retorno del carro cuando se pasaba un determinado número de caracteres:

```
<p>One of the strange-but-truisms of
<p>minor peddling is that using the
<p>computer and other Fetish fodder
<p>somehow empowers children - plug
<p>in, log on, attend a good
<p>college on full scholarship, and
<p>get the hell out of the house.
```

Tras ello, toda la producción se envolvía en etiquetas <tt> para obligar a los primeros navegadores gráficos (básicamente a Netscape 1.1) a aplicar una fuente de un solo espacio como Courier o Monaco al texto.

El resultado era un rudimentario control tipográfico y una imitación forzada del ajuste entre caracteres. Estos intentos HTML ofrecían la única forma de conseguir efectos de diseño en 1995 (el ejemplo de la figura 2.1 es de 1996, después de un cambio de diseño más orientado a gráficos de Suck, aun así bastante minimalista; el diseño original ya no está disponible).

Los diseñadores Web empleaban métodos creativos similares que obligaban a HTML a generar efectos de diseño y se impartían en las primeras biblias de diseño Web por autores como Lynda Weinmann y David Siegel. Los creadores de HTML chasqueaban la lengua ante esta deformación de HTML, pero los diseñadores no tenían otra opción ya que los clientes exigían cuidados productos Web.

Muchos diseñadores siguen empleando métodos como éstos a diario y muchos libros siguen pregonando estas técnicas desfasadas y, en la Web del presente, contraproducentes. Un excelente libro sobre diseño Web del 2002 aconsejaba a sus lectores que para controlar la tipografía utilizaran etiquetas de fuentes y secuencias de comandos HTML. Las etiquetas de fuentes llevan tiempo en desuso (lenguaje de W3C para decir "no usen esta basura anticuada") y no se pueden crear secuencias de comandos de HTML, pero este tipo de consejos erróneos y sin sentido persisten en obras de diseño Web de distribución mundial, lo que perpetúa la ignorancia y el disparate.

El coste del diseño antes de los estándares

Al manipular HTML de forma creativa, Suck había conseguido un aspecto característico con un coste doble: el sitio excluía a ciertos lectores y resultaba difícil de actualizar.

En los primeros lectores de pantalla Mom-and-Pop (navegadores de audio para discapacitados visuales), la voz que leía el texto de Suck se detenía cada pocas palabras debido a la continua presencia de etiquetas de párrafo, lo que interrumpía el ritmo de los editoriales de Suck, tan brillantemente argumentados:

> One of the strange-but-truisms of ...
> [molesta pausa]
>
> minor peddling is that using the ...
> [molesta pausa]

computer and other Fetish fodder ... [molesta pausa]

somehow empowers children—plug ... [molesta pausa]

in, log on, attend a good ... [molesta pausa]

college on full scholarship, and ... [molesta pausa]

get the hell out of the house.

Ya bastante difíciles de analizar en condiciones ideales, las enrevesadas estructuras de frases de Suck se volvían incomprensibles cuando se veían interrumpidas por etiquetas de párrafo no semánticas. Estos problemas de audio se hacían insoportables para los usuarios de lectores de pantalla y les impedía utilizar el sitio.

Si los trucos HTML que consiguieron que el diseño funcionara en navegadores gráficos frustraban a un número desconocido de usuarios, también suponían un problema para los creadores de Suck cada vez que tenían que actualizar el sitio.

Como el diseño dependía de partes de Perl o HTML, era imposible crear una plantilla. Las entregas diarias de Suck suponían horas de trabajo de producción. Con el aumento de la popularidad del sitio, que acabaría con la compra corporativa del mismo, sus creadores se vieron obligados a contratar no sólo escritores adicionales sino también un equipo de productores. La labor manual de la producción de Suck era incompatible con la necesidad de publicar diariamente.

En un mundo más perfecto, estas dificultades se habrían confinado a la época a la que pertenecen. Se convertirían en anécdotas de los albores de la programación Web comercial. Cuando admiramos la ingenuidad de los primeros diseñadores, sonreímos al pensar que la programación fuera tan retorcida. Pero a pesar de la aparición de los estándares, la mayor parte de la producción comercial sigue dependiendo de soluciones extrañamente laboriosas y sufriendo de los problemas que generan estos métodos. Esta práctica está tan extendida que muchos programadores y diseñadores ni siquiera la ponen en duda.

Sitios modernos, técnicas antiguas

Saltemos de 1995 al 2001 para analizar un sitio Web contemporáneo que anuncia el Gilmore Keyboard Festival (www.thegilmore.com) (véase figura 2.3). Es un sitio agradable, confeccionado con técnicas de diseño de tablas muy trabajadas que durante mucho tiempo han sido normas de la industria.

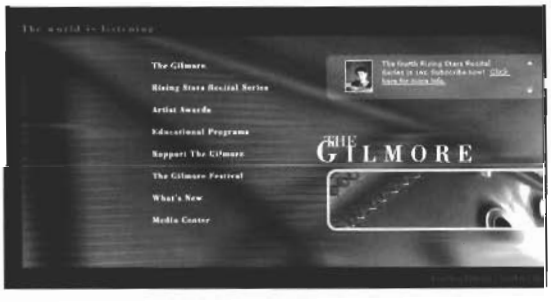


Figura 2.3.

Sitio Web de The Gilmore Keyboard Festival (www.thegilmore.com). Una delicia para los ojos pero un suplicio para actualizar o mantener.

Aparte del hecho de que el sitio de Gilmore asume el tamaño del monitor y la ventana del navegador del visitante, ¿qué es lo que falla en la esta imagen? Desde el punto de vista del propietario, el sitio tiene algunos puntos débiles:

- **Penalización financiera por cambios:** Si cambia algo relacionado con el festival, por ejemplo si se añaden nuevos recitales al cartel de este año, el sitio no se puede actualizar por medio de un sencillo enlace de texto. Los diseñadores Web tampoco pueden añadir enlaces adicionales al mapa de imágenes GIF de texto que actúa como menú de navegación del sitio. La tabla HTML que combina las distintas imágenes en un todo aparente (véase figura 2.4) se vendría abajo si se cambiara el tamaño de cualquiera de las imágenes integrantes.

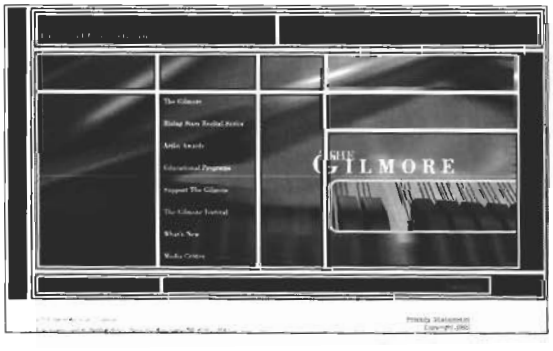


Figura 2.4.

El mismo sitio con márgenes y bordes CSS añadidos para revelar los métodos de construcción (y los posibles problemas de mantenimiento).

Por ello, incluso el más mínimo cambio en el sitio supondría un coste significativo. Sería necesario volver a diseñar, cortar y optimizar los gráficos, y habría que escribir de nuevo el marcado de tablas junto con el marcado de mapa de imágenes asociado y el código JavaScript. Cuando una tarea tan básica como añadir un enlace requiere horas de trabajo, hay que preguntarse si los métodos de producción normativos han caducado.

- **Exclusión de multitud de posibles visitantes:** Aunque sea menos evidente pero no menos importante, tal y como está implementado el sitio, resulta inaccesible para los usuarios de lectores de pantalla, navegadores de texto, Palm Pilots, teléfonos móviles habilitados para la Web y para aquéllos que utilicen navegadores convencionales pero que desactiven las imágenes. Estos usuarios sólo tendrán problemas, pueden perder sus entradas o algo peor. Si se ve en un entorno de navegación no gráfico (véase figura 2.5), el contenido total de la página es el siguiente:

```
[INLINE]
[INLINE] [INLINE]
[USEMAP]
[INLINE] Invite a friend | Contact us
Irving S. Gilmore International
Keyboard Festival
The Epic Center 359 S. Kalamazoo Mall
Suite 101, Kalamazoo,
MI 49007-4843 Privacy Statement
Copyright 2003
```

¿Es `INLINE INLINE INLINE` el mensaje que los promotores de The Gilmore quieren transmitir? Lo dudamos. ¿Es de ayuda para los posibles visitantes? Evidentemente no.

No todos los visitantes frustrados pasarán por alto esta frustración. En teoría, un amante de la música discapacitado puede

reclamar que el sitio le ha discriminado. No somos abogados y no tenemos nada en contra de este sitio que evidentemente se ha creado con las mejores intenciones.

Tampoco queremos decir que no se deban usar imágenes o que estemos en contra de la belleza. Por el contrario, las imágenes resultan vitales, la belleza es necesaria y los creadores del sitio de The Gilmore han hecho un magnífico trabajo de creación de una experiencia estética en línea. Dicha experiencia no puede ser inaccesible.

Este diseño podría conservarse y resultar accesible para todos los usuarios. Pero no lo es (todavía).

Aunque de una belleza poco usual, el sitio de The Gilmore emplea métodos de diseño y construcción bastante típicos. Y los problemas que sufre el sitio a causa de dichos métodos también son típicos. La mayoría descubrimos que los diseños de tablas que tan cuidadosamente hemos realizado se rompen cuando el cliente quiere realizar cambios, como hacen todos los clientes. O se lo cobramos al cliente o asumimos el coste.

Cuando las limitaciones de nuestro presupuesto son drásticas, por ejemplo durante una crisis económica, el cliente puede demandar soluciones rápidas que destruyan la elegancia y claridad de la presentación existente.

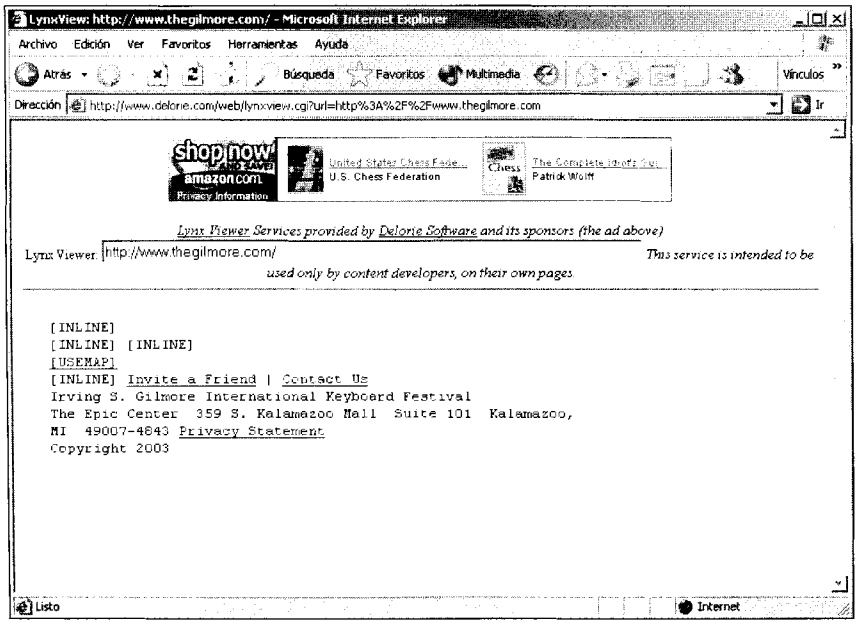


Figura 2.5.

En un navegador no gráfico, The Gilmore no proporciona información alguna, texto, enlaces, nada. Para conseguir esta captura de pantalla se ha utilizado un emulador en línea gratuito de Lynx (www.delorie.com). Si quiere ver cómo aparecen sus páginas Web en un entorno no gráfico, pruébelas en Lynx, con un emulador de Lynx o en un lector de pantalla como JAWS (o con los tres).

Por ejemplo, si se añadieran tres simples enlaces de hipertexto en la parte superior o inferior del sitio, los visitantes podrían acceder a nuevas secciones del mismo, pero también resultaría confuso. (El visitante se preguntaría por qué están estos enlaces separados del resto, si son más importantes que los otros, si son menos importantes, etc.)

La inclusión de dichos enlaces destrozaría los efectos estéticos del sitio tan cuidadosamente controlados y disminuiría la imagen de marca de un festival tan serio lo que, a su vez, reduciría el valor de mercado del sitio.

Al igual que los creadores del sitio de The Gilmore, muchos nos damos cuenta de que los diseños antiguos no funcionan correctamente. Pueden resultar adecuados para la mayoría de los navegadores más conocidos en condiciones normales y con una configuración normal de las preferencias. Pero al superar dichas condiciones normales, puede que nuestros sitios dejen de comunicar. Como mínimo, este tipo de inaccesibilidad reduce los clientes potenciales.

Los propietarios y diseñadores de The Gilmore no son únicos. Cumplen normas de la industria que llevan tiempo establecidas. Los problemas que generan estas normas son los mismos a los que se enfrenta la mayoría de los sitios Web actuales. Son los problemas a los que se enfrenta cualquier sitio creado en función de una serie de navegadores conocidos en vez de haberse diseñado para facilitar un acceso universal a través de estándares Web. También son los problemas de cualquier sitio que une la presentación a la estructura forzando a HTML a crear diseños.

Afortunadamente, existe otra forma de diseñar y crear sitios Web, una forma que resuelve los problemas generados por los métodos anticuados sin sacrificar los beneficios estéticos y de marca que ofrecen dichos métodos: los estándares Web.

La trinidad de los estándares Web

En la figura 2.6 puede comprobar cómo los estándares Web resuelven el problema que hemos mencionado. Separan cualquier página Web en tres componentes diferentes: estructura, presentación y comportamiento.

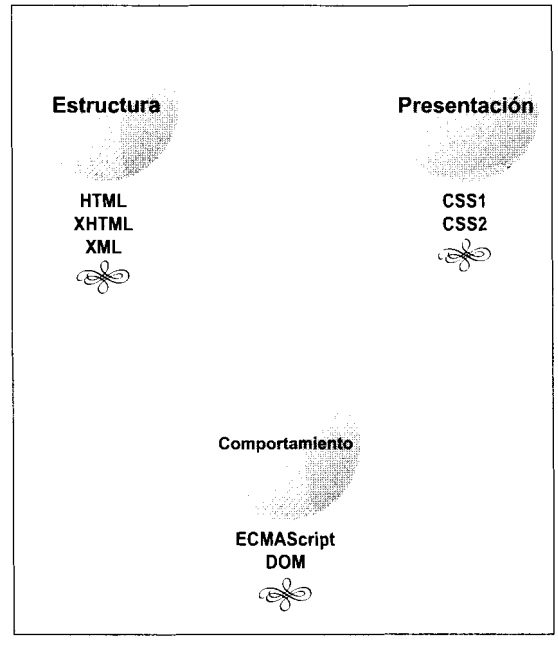


Figura 2.6.

Estructura, presentación y comportamiento, los tres componentes de cualquier página Web en el mundo de los estándares Web.

Estructura

Un lenguaje de marcado (XHTML: http://www.w3.org/TR/xhtml1) contiene datos de texto con un formato supeditado a su significado estructural: título, título secundario, párrafo, lista numerada, lista de definiciones, etc.

En la Web, este texto formaría parte de una lista de definición <dl>. El subtítulo "Estructura" se marcaría como título de definición <dt>. El párrafo que está leyendo se incluiría en las etiquetas de datos de definición <dd>:

```
<dl>
   <dt>
      Estructura
   </dt>
   <dd>
   Un <em>lenguaje de marcado</em>
   (<a href=http://www.w3.org/TR/ xhtml1>
   XHTML</a>) contiene datos de texto con
   un formato supeditado a su significado
   estructural: título, título
   secundario, párrafo, lista numerada,
   lista de definiciones, etc.
   </dd>
   <dd>
   En la Web, este texto formaría parte
   de una lista de definición. El
   subtítulo "Estructura" se marcaría
   como título de definición. El párrafo
   que está leyendo se incluiría en las
   etiquetas de datos de definición.
   </dd>
</dl>
```

Por otra parte, los dos párrafos podían depender de un solo elemento <dd>:

```
<dl>Estructura</dl>
<dd>
   <p>Un <em>lenguaje de marcado</em>
   (<a href=http://www.w3.org/ TR/xhtml1>
   XHTML</a>) contiene datos de texto con
   un formato supeditado a su significado
   estructural: título, título
      secundario, párrafo, lista
numerada,
   lista de definiciones, etc.</p>
   <p>En la Web, este texto formaría
parte de una lista de definición. El
subtítulo "Estructura" se marcaría como
título de definición. El párrafo que está
leyendo se incluiría en las etiquetas de
datos de definición.</p>
</dd>
```

XML (http://www.w3.org/TR/2000/REC-xml-20001006), el lenguaje de marcado extensible, ofrece más opciones pero por ahora nos limitaremos a XHTML, un lenguaje de marcado transicional y recomendación W3C actual que funciona como HTML prácticamente en todos los navegadores y dispositivos de Internet.

Cuando se crea correctamente (sin errores, ni etiquetas o atributos ilegales), el marcado XHTML es completamente portable. Funciona en navegadores Web, lectores de pantalla, navegadores de texto, dispositivos inalámbricos, etc.

El marcado también puede incluir estructuras adicionales que el diseñador considere necesarias. Por ejemplo, el contenido y la navegación se pueden marcar como tal e incluir en etiquetas con el correspondiente título:

```
<div id="content">[Aquí va el
contenido.]</div>
<div id="navigation">[Aquí va el menú de
navegación.]</div>
```

El marcado también contiene objetos incrustados como imágenes, presentaciones en Flash o películas QuickTime junto con etiquetas y atributos que presentan equivalentes en texto para los que no pueden ver estos objetos en su entorno de navegación.

Presentación

Los lenguajes de presentación (CSS1: http://www.w3.org/TR/REC-CSS1; CSS2: http://www.w3.org/TR/REC-CSS2) aplican formato a una página Web y controlan la tipografía, el color, la disposición, etc.

En muchos casos, CSS puede reemplazar diseños de tablas HTML de la vieja escuela. En todos los casos, sustituye etiquetas de fuente no estándar y elementos desfasados que malgastan ancho de banda como el siguiente:

```
<td bgcolor="#FFCC00" align="left"
valign="top"><br><br><br> </td>
```

Esto se puede reemplazar por una celda de tabla sin adornos, por una celda de tabla con un atributo de clase o por nada.

Como la presentación se separa de la estructura, se puede cambiar una sin afectar negativamente a la otra. Por ejemplo, puede aplicar el mismo formato a varias páginas o modificar texto y enlaces sin que afecte al diseño. Puede cambiar el XHTML en cualquier momento sin miedo a alterar el diseño ya que el texto es simplemente texto, no tiene una doble función como lenguaje de diseño.

Del mismo modo, puede modificar el diseño sin tocar el marcado. ¿Los lectores se han quejado del tamaño de tipo de letra de su sitio? Basta con cambiar una regla en la hoja de estilo global para que en la totalidad del sitio se reflejen automáticamente las modificaciones. ¿Necesita una versión que se pueda imprimir? Escriba una hoja de estilo de impresión y sus páginas se imprimirán a la perfección, independientemente de cómo se vean en la pantalla del navegador Web del PC (en un apartado posterior explicaremos cómo se hace).

Comportamiento

Un modelo de objeto estándar (el DOM de W3C en http://www.w3.org/DOM/DOMTR#dom1) funciona con CSS, XHTML y ECMAScript 262 (http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM), la versión estándar de JavaScript, lo que le permite crear sofisticados comportamientos y efectos que funcionen en diferentes navegadores y plataformas. Se acabó el JavaScript sólo para Netscape. Se acabó el ActiveX y el JavaScritp sólo para IE/Windows (como mencionamos en el capítulo anterior).

En función de los objetivos y del público del sitio, los diseñadores y programadores pueden obtener todas las prestaciones de los estándares Web si separan la estructura de la presentación y el comportamiento. O pueden optar por crear sitios de transición que combinen métodos antiguos y nuevos, como por ejemplo mezclar diseños de tablas XHTML con control CSS de tipografía, márgenes, interlineado y colores.

En marcha

Si Suck siguiera en activo hoy en día, los estándares Web como XHTML y CSS permitirían que sus responsables se centraran en escribir. Una plantilla XHTML básica se encargaría de la estructura de documentos. CSS controlaría el aspecto operativo y visual sin necesidad de un diseño adicional para

cada ejemplar, aparte de la preparación de las imágenes concretas de cada artículo. Se podrían utilizar etiquetas de párrafo para determinar el inicio y el fin de los párrafos, no para forzar espacios verticales entre cada línea de texto (CSS se encargará de todo).

En navegadores gráficos como IE, Mozilla/Netscape y Opera, las hojas de estilo garantizarían que el aspecto de Suck es el que pretende el diseñador. El XHTML estructurado permitiría ver el contenido de Suck no sólo en dichos navegadores sino también en PDA, lectores de pantalla y navegadores de texto sin las correspondientes molestias no estructurales de párrafos falsos y similares propias de un marcado utilizado como herramienta de diseño.

Como sitio centrado en contenidos, `Suck.com` sería un candidato perfecto para un cambio XHTML/CSS mediante el que el contenido y el estilo se conseguirían a través de la tecnología correcta: CSS para el diseño y XHTML para el contenido estructurado. Pero también se podría beneficiar de un enfoque de transición: sencillas tablas XHTML para la colocación de las principales áreas de contenido y CSS para el resto.

Los creadores del sitio The Gilmore no se beneficiarían del uso de plantillas, por lo menos no en la página inicial del sitio debido a su diseño actual. Pero podrían comunicar su diseño con CSS y conservar el ancho de banda al tiempo que permitirían al equipo de diseño cambiar una sección de la página sin tener que modificar todo el formato.

Podrían utilizar la propiedad de fondo de CSS para ubicar su imagen principal como un solo archivo JPEG en lugar de una docena de fragmentos de imagen (véase figura 2.4) y podrían utilizar uno o varios gráficos de menú con cualquiera de los métodos de posicionamiento de CSS, incluyendo algunos que funcionaran en navegadores 4.0 que no admiten CSS de forma completa.

Los creadores de The Gilmore también podrían utilizar XHTML y atributos de accesibilidad como `alt.title` y `longdesc` para garantizar que el contenido del sitio es accesible para todos en lugar de para unos pocos (véase figura 2.5). Con ayuda de métodos de transición (CSS y tablas) u otros más estrictos (sólo CSS), se podrían haber conseguido páginas con menos gráficos en las partes interiores del sitio.

Ventajas de los métodos de transición

Los métodos de transición compatibles con XHTML y CSS suponen una gran mejora con respecto a lo que tenemos hoy en día y nos permiten resolver los problemas que hemos mencionado hasta el momento. Con las técnicas de transición, CSS para tipografía, color, márgenes, etc., y tablas XHTML para diseños básicos, se aumenta la accesibilidad, la compatibilidad y la viabilidad a largo plazo, aunque requiere más tiempo y esfuerzo (y en la mayoría de los casos, más ancho de banda) que un enfoque CSS puro sin tablas.

Happy Cog (www.happycog.com), el sitio Web de mi agencia, es de tipo transaccional (véanse figuras 2.7, 2.8 y 2.9). Combina técnicas de diseño de tablas XHTML con CSS1 y CSS2, y sencillas secuencias de comandos basadas en DOM. El sitio es compatible con

los estándares XHTML 1.0 Transational y CSS, así como con los requisitos de accesibilidad de la Section 508 y WAI Priority Guidelines (como veremos más adelante).

Al cumplir estos aspectos, el sitio Happy Cog garantiza su viabilidad a largo plazo con la evolución de navegadores y estándares. Al utilizar algunos métodos de la vieja escuela (principalmente diseños de tablas), consigue mantener su aspecto en los nuevos navegadores (véase figura 2.7) y representarse de forma aceptable en clientes de la era de las guerras de los navegadores como Netscape 4 (véase figura 2.8), cuya compatibilidad con CSS no es completa. Al mismo tiempo, la utilización por parte de Happy Cog de sencillas técnicas de marcado estructural y de accesibilidad permite que su contenido llegue a navegadores no gráficos (véase figura 2.9), lectores de pantalla y los nuevos dispositivos inalámbricos.

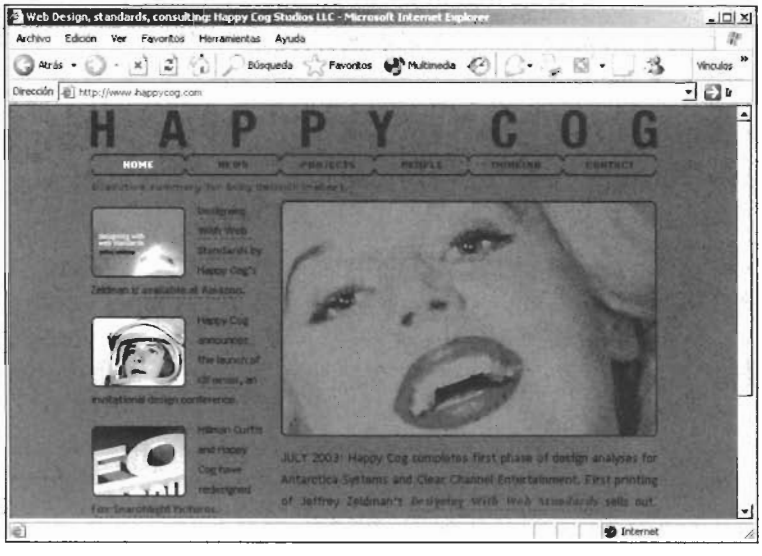


Figura 2.7.

Happy Cog, el sitio empresarial de este autor (www.happycog.com) es un ejercicio de compatibilidad directa de transición que combina diseños de tablas XHTML con CSS y el DOM. Cuando se reproduce en un navegador moderno, se puede apreciar su verdadera presentación.

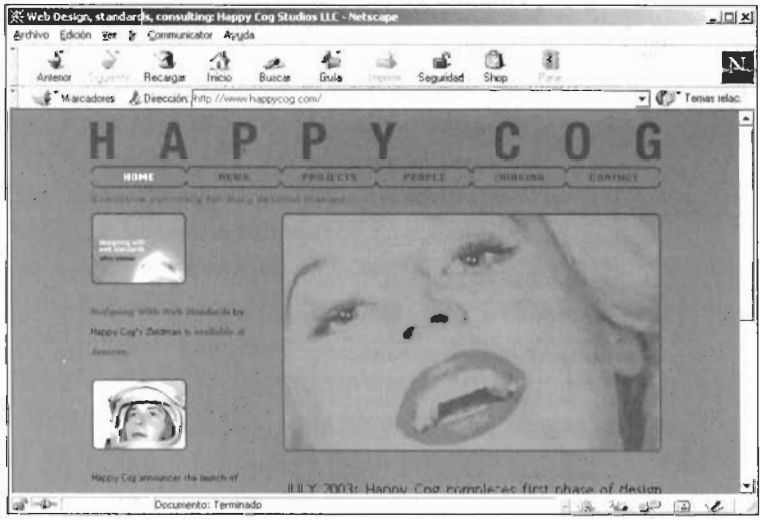


Figura 2.8.

Al abrir el sitio en un navegador antiguo (Netscape 4), cuya compatibilidad con CSS es escasa, la mayor parte de la presentación de Happy Cog se reproduce intacta. Se pierden algunos detalles del diseño, pero no nos importa ni tampoco a los usuarios de Netscape 4, acostumbrados a cierta falta de precisión en la mayor parte de los sitios que visitan.

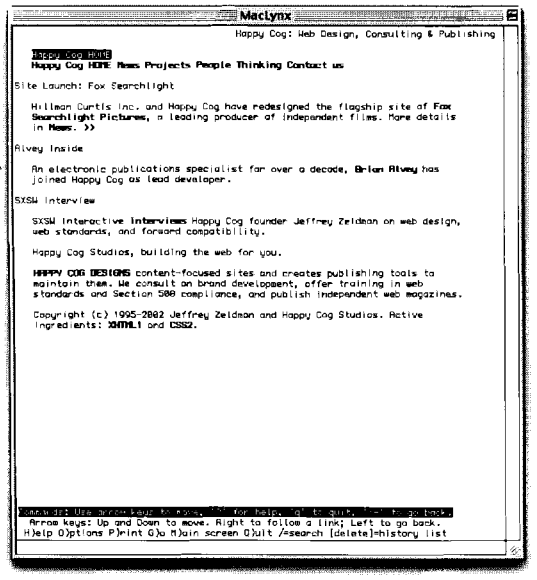


Figura 2.9.

De nuevo Happy Cog, esta vez en Lynx, un navegador de sólo texto. Como se supone, el contenido del sitio es completamente accesible cuando se separa de la presentación visual. Compárelo con la figura 2.5, en la que ninguno de los contenidos de The Gilmore resulta accesible más allá de un navegador grafico de escritorio. Happy Cog no es mejor que The Gilmore, simplemente utiliza estándares para garantizar su accesibilidad, algo que The Gilmore no hace (al menos todavía).

La estrategia de transición aplicada por Happy Cog ofrece compatibilidad inversa y directa, y es una táctica apropiada para muchos sitios actuales. Pero para obtener el máximo beneficio del diseño y creación con estándares Web, es necesario un profundo cambio en la forma de pensar y en la metodología. El objetivo del cambio de producción, conservando la integridad y portabilidad del contenido, y de proporcionar el nivel correcto de diseño a diferentes clientes, no sólo consiste en cómo deben funcionar las cosas, sino también en cómo funcionan hoy en día, cuando diseñamos y creamos con estándares.

La capacidad para separar estructura de presentación y comportamiento es la base de este nuevo enfoque de diseño. Es la forma en que se diseñarán todos los sitios en el futuro (a menos que se trate de sitios sólo de Flash) y ya se ha empleado en sitios directamente compatibles, como los que vamos a analizar.

El Web Standards Project: portabilidad en acción

El Web Standards Project (WaSP, Proyecto de estándares Web) (véase figura 2.10) apareció en 1998 para persuadir a Netscape, Microsoft y otros fabricantes de navegadores de que admitieran los estándares que describimos en este libro. Llevó mucho tiempo, perseverancia y estrategia (o, en otras palabras, lloramos, suplicamos y rogamos) pero al final se consiguió. El WaSP sostenía que la compatibilidad a través de estándares comunes era completamente necesaria si la Web quería evolucionar.

Cuando los navegadores empezaron a admitir los estándares (como veremos en un capítulo posterior), el proyecto de los estándares Web volvió a aparecer en el 2002 para alentar a los diseñadores y programadores a que los usaran y dominaran las prestaciones de estas tecnologías por las que tanto se había luchado. Como muestra del alcance de la misión del grupo, desde algo puramente evangélico hasta convertirse en un recurso educativo, se volvió a escribir y diseñar el sitio.

Nota: Una de las ironías de la lucha por la compatibilidad con estándares en los navegadores es que Netscape y Microsoft son miembros del W3C que contribuyeron significativamente a la creación de estándares Web y aun así hubo que convencerlos para que admitieran las mismas tecnologías que ellos habían ayudado a crear. Para que vea.

Como se esperaba, el sitio tiene un aspecto agradable en navegadores compatibles con estándares (véase figura 2.10).

También resulta aceptable en navegadores más antiguos y menos compatibles (véase figura 2.11). Pero el sitio transciende el espacio de navegación basado en el PC sin necesidad de marcado, código o detección de dispositivos adicionales o alternativos (se acabaron las versiones).

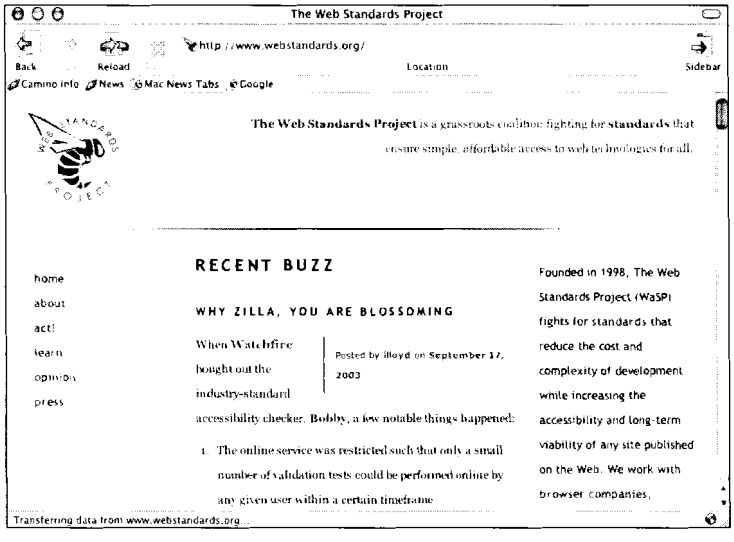


Figura 2.10.

La página inicial del Web Standards Project vista en Camino, un navegador basado en Gecko para Mac OS X (www.web-standards.org). Su diseño CSS se reproduce de forma idéntica en todos los navegadores modernos compatibles con estándares. Pero todavía hay más.

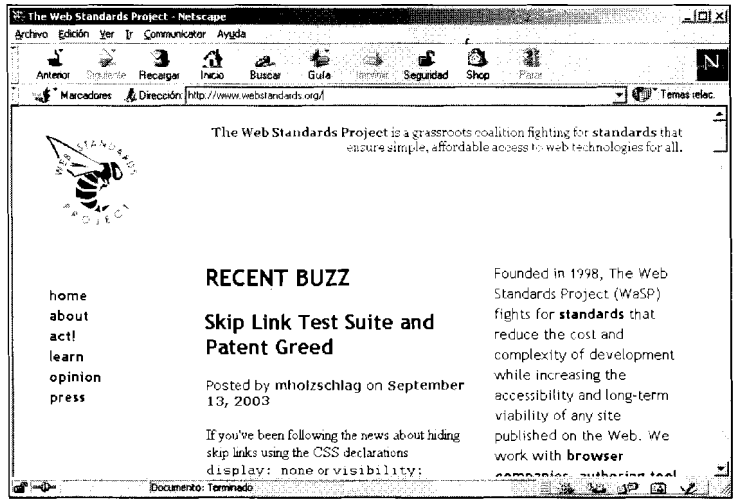


Figura 2.11.

El mismo sitio con un aspecto decente y un funcionamiento aceptable en Netscape 4, ejemplo de incompatibilidad con estándares. No fue necesaria una versión para Netscape 4.

Un documento para todo

El proyecto de estándares Web se ha diseño con XHTML 1.0 Strict. CSS se ha utilizado para el diseño. No hay versiones PALM ni WAP. No se necesitan múltiples versiones; cuando se diseña y se crea con estándares, un documento vale para todo.

La figura 2.12 muestra `webstandards.org` tal y como se ve en un reproductor Palm. En la figura 2.13, se reproduce el mismo sitio en un PocketPC de Microsoft. El caso más extraño de todos, la figura 2.14, muestra cómo funciona perfectamente el sitio en un equipo manual Newton, el predecesor de Apple de la Palm Pilot. Grant Hutchinson, que capturó la imagen de Newton, nos dijo que no hay nada como ver un sitio moderno utilizando un navegador poco sistemático en un sistema operativo antiguo.

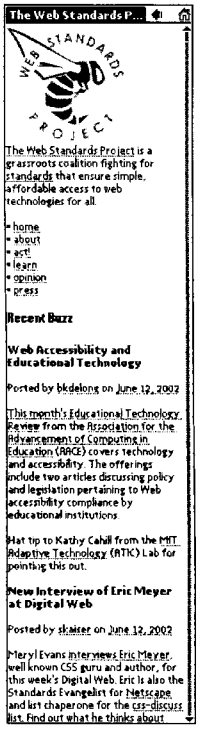


Figura 2.12.

El mismo sitio otro día, reproducido en un Palm Pilot. Sin WAP. Captura de pantalla cortesía de Porter Glendinning (www.serve.com/apg/).

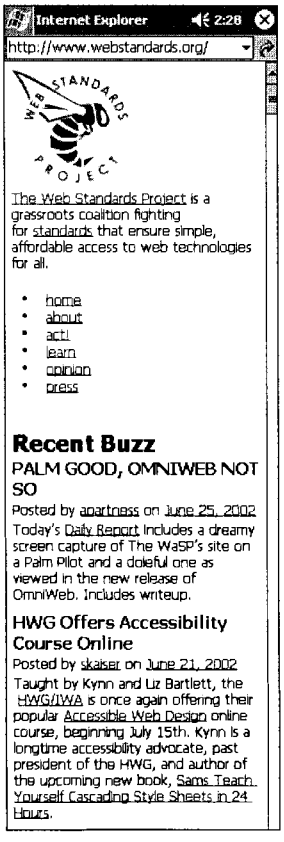


Figura 2.13.

El mismo sitio visto en un PocketPC de Microsoft. Captura de pantalla cortesía de Anil Dash (www.dashes.com/anil/).

Esto sonará como música celestial para cualquier diseñador o propietario de un sitio que quiera llegar al mayor número de visitantes con el mínimo esfuerzo. Una estricta compatibilidad con XHTML y un uso inteligente de CSS libera a diseñadores y programadores de tener que crear múltiples versiones.

En las tres últimas imágenes, el menú DHTML que aparece en la parte izquierda de la pantalla en un navegador Web de escritorio se convierte en una lista numerada convencional en la pantalla de un Palm, PocketPC o Newton. Se debe a que en realidad el menú DHTML es una lista numerada (sin ordenar). CSS modifica su aspecto en los navegadores de escritorio compatibles. El

menú cambia de página a página gracias a SSI (comandos ejecutados en el servidor).

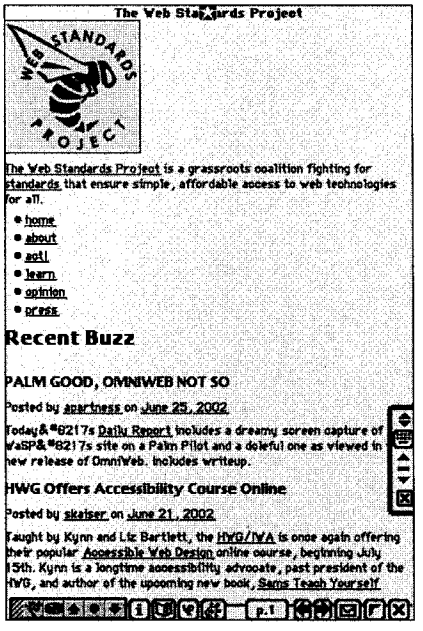


Figura 2.14.

De nuevo Webstandards.org, en esta ocasión en un equipo manual Newton de Apple. Captura de pantalla cortesía de Grant Hutchinson (www.splorp.com).

A List Apart: una página, varias vistas

A List Apart (www.alistapart.com), la revista en línea de este autor "para los creadores de sitios Web", se convirtió a diseño CSS en febrero del 2001. El diseño (véase figura 2.15) se adaptó a partir del aspecto funcional y visual "HTML minimalista" del sitio que previamente se había conseguido con ayuda de tablas HTML.

Los lectores de ALA contrarios al tratamiento de texto predeterminado del sitio pueden cambiar a un tipo de letra de mayor tamaño y más legible gracias a un conmutador de hojas de estilo diseñado por Paul Sowden. En el número 126 de ALA (www.alistapart.com/issues/126/) se explica cómo funciona y se incluye el código abierto que puede utilizar y adaptar para crear conmutadores de hojas de estilo para sus sitios Web (en capítulos posteriores encontrará más información al respecto).

En la figura 2.15 puede ver el aspecto de A List Apart en navegadores compatibles como IE5+, Mozilla, Netscape 6+ y Opera 5+. En otros entornos, el sitio adopta un aspecto completamente diferente.

En la figura 2.16 se muestra el mismo sitio reproducido en un navegador no compatible con CSS, en este caso Netscape Navigator 4. El sitio resulta perfectamente legible y utilizable.

En realidad, algunos lectores prefieren este nuevo aspecto. Inmediatamente después del cambio de diseño, el uso de Netscape 4 por parte de los visitantes de ALA aumentó temporalmente. Parecía que estos usuarios preferían una página simple que su procesador pudiera procesar al diseño anterior que únicamente ponía en evidencia las debilidades de Netscape 4.

Aquéllos que sostienen que los estándares son perjudiciales para los usuarios de navegadores antiguos pueden considerar a esta experiencia un indicador de justo lo contrario. Cuando se utilizan correctamente, los estándares son útiles para todos.

Una fuente de inspiración

Todas las hojas de estilo utilizadas en el sitio A List Apart (y en `zeldman.com`) son de código abierto y las puede utilizar y adaptar de forma gratuita en sus propios sitios. Para buscar las hojas de estilo de un sitio, debe seleccionar la opción **Ver>Código fuente** en su navegador, anotar la ubicación de los archivos CSS indicados en la parte `<head>` del documento y, tras ello, cortar y pegar dicha ubicación en la barra de direcciones de su navegador. Por ejemplo, si el XHTML de la etiqueta `<head>` del documento es el siguiente:

```
<style type="text/css" media="all">
@import "/styles/basic.css ";
</style>
```

o

```
<link rel="StyleSheet"
href="/styles/basic.css"
type="text/css" media="screen" />
```

tendría entonces que introducir http://www.domain.com/styles/basic.css en la barra de direcciones de su navegador. La mayoría de los navegadores mostrará la hoja de estilo en texto sencillo que puede cortar y pegar en su editor HTML o descargar en su disco duro.

Para ahorrar tiempo, puede que le interese visitar www.favelets.com e instalar la aplicación de marcadores View CSS Style Sheets en la barra de herramientas de su navegador. De esta forma puede cargar las hojas de estilo de cualquier sitio en nuevas ventanas del navegador con un solo clic.

El cambio de diseño CSS de ALA lo realicé en colaboración con Todd Fahrner y Tanek Çelik, participantes de los grupos de trabajo CSS de W3C. Farhner es un experto en CSS y miembro del comité de dirección de The Web Standards Project. Çelik es un ingeniero de navegadores que trabaja en Microsoft, creador de Favelets y responsable del motor de representación Tasman que controla la edición para Macintosh de IE5 (en un apartado posterior describiremos estos motores y la compatibilidad con estándares).

En la figura 2.16, apreciará que la barra de navegación parece haber desaparecido. De hecho, simplemente se ha desplazado a la parte inferior de la página. El texto aparece con el tipo y el tamaño determinados por las preferencias del usuario. No se utiliza detección de navegadores alguna para conseguir este formato simplificado. En su lugar, la hoja de estilo que controla el aspecto en pantalla del sitio está vinculada de tal forma que los navegadores no compatibles la ignoran (como explicaremos más adelante).

No hay tablas ni otros elementos no estructurales que estorben al contenido. La única concesión de diseño realizada es el atributo de color de fondo aplicado a la etiqueta `<body>` para que los gráficos del encabezado coincidan con el resto de la página en navegadores incompatibles con CSS.

Figura 2.15.

A List Apart, la revista en línea para diseñadores Web del autor de este libro, vista en un navegador compatible con CSS (www.alistapart.com). ALA adoptó un diseño sólo de CSS en febrero del 2001. Tras ello, siguieron cientos de sitios.

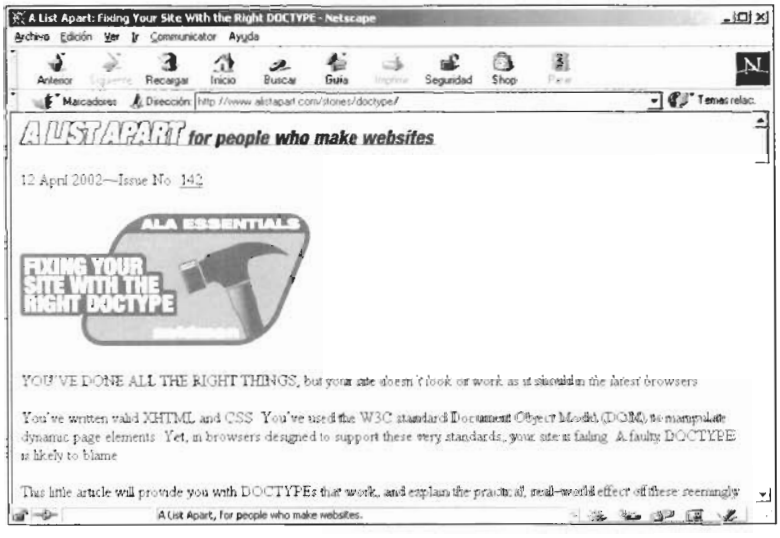


Figura 2.16.

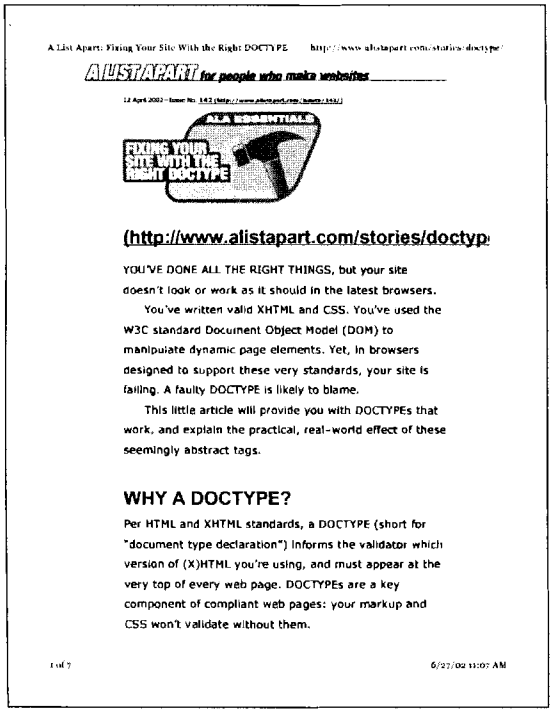
El mismo sitio visto en un navegador 4.0. No hay diseño CSS, no hay problemas. El sitio resulta perfectamente legible y utilizable.

Diseño más allá de la pantalla

Por último, pero no por eso menos importante, en la figura 2.17 puede ver el aspecto de un artículo de ALA al imprimirlo. Como puede comprobar, se ha eliminado la barra lateral, se han optimizado las fuentes y colores para la impresión y se muestra el URL de todos los enlaces, aparezcan o no en la versión en pantalla.

Esto se ha conseguido gracias a una hoja de estilo de impresión distinta diseñada por Eric Meyer, creada a partir de una hoja de estilo de impresión anterior creada por Todd Fahrner y por el autor de este libro. Meyer es un experto en CSS y autor del libro Meyer on CSS. Su artículo para ALA, "Going to Print" (www.alistapart.com/stories/goingtoprint/) explica las técnicas necesarias

para crear una hoja de estilo de impresión, técnicas que analizaremos en capítulos posteriores.


A List Apart: Fixing Your Site With the Right DOCTYPE http://www.alistapart.com/stories/doctype/

ALISTAPART for people who make websites

12 April 2002 - Issue No. 142 (http://www.alistapart.com/issues/142/)

ALA ESSENTIALS

FIXING YOUR SITE WITH THE RIGHT DOCTYPE

(http://www.alistapart.com/stories/doctyp

YOU'VE DONE ALL THE RIGHT THINGS, but your site doesn't look or work as it should in the latest browsers.

You've written valid XHTML and CSS. You've used the W3C standard Document Object Model (DOM) to manipulate dynamic page elements. Yet, in browsers designed to support these very standards, your site is failing. A faulty DOCTYPE is likely to blame.

This little article will provide you with DOCTYPEs that work, and explain the practical, real-world effect of these seemingly abstract tags.

WHY A DOCTYPE?

Per HTML and XHTML standards, a DOCTYPE (short for "document type declaration") informs the validator which version of (X)HTML you're using, and must appear at the very top of every web page. DOCTYPEs are a key component of compliant web pages: your markup and CSS won't validate without them.

1 of 7 6/27/02 11:07 AM

Figura 2.17.

De la Web al papel: los artículos de ALA se pueden imprimir al instante gracias a hojas de estilo de impresión.

Por el momento, el concepto más importante que debe asimilar es que con un solo documento, una hoja de estilo de impresión, el sitio ALA ya no necesita producir diferentes versiones aptas para la impresión. Con toda probabilidad, tampoco será necesario en sus propios sitios. Existe una posible excepción si se trata de sitios que emplean formatos de artículos multipágina, como www.wired.com u O'Reilly Network (www.oreilly.com). Este tipo de sitios requieren una página apta para impresión simplemente para mantener todo el artículo en una misma página. Sin embargo, también pueden beneficiarse del uso de una hoja de estilo de impresión.

A continuación repasaremos los beneficios que pueden obtener los dos sitios que acabamos de mencionar.

Menos tiempo y dinero, más público

Si el diseño y la creación con estándares significa que ya no necesita crear varias versiones de todos los sitios, resulta evidente que la reducción de tiempo y costes puede ser considerable:

- Se acabaron las versiones sólo para Netscape.
- Se acabaron las versiones sólo para IE.
- Se acabaron las versiones básicas para navegadores antiguos.
- En muchos casos, se acabaron las versiones WAP o WML.
- En muchos casos, se acabaron las versiones especiales para impresión.
- Se acabó el husmear en navegadores y plataformas, y recurrir al servidor para obtener diferentes componentes optimizados para navegadores o dispositivos.

Aunque muchas organizaciones quieran llegar a usuarios de dispositivos inalámbricos o no tradicionales, no se pueden permitir el gasto que supone crear versiones inalámbricas o de sólo texto. Gracias a los estánda-

res XHTML y CSS, no es necesario. Sin mover un dedo, estas organizaciones podrán llegar a nuevos lectores y clientes.

Un cumplimiento estricto de los estándares también es una forma magnífica de solucionar problemas de accesibilidad. Si su sitio funciona en un Palm Pilot, es muy probable que también lo haga en un lector de pantalla como Jaws aunque, evidentemente, tendrá que probarlo para confirmarlo y puede que tenga que dedicarle más tiempo si quiere que sea verdaderamente accesible. Analizaremos estos aspectos en un capítulo posterior.

El futuro

Los lenguajes de marcado basados en XML, harán que la Web actual sea cosa de niños. Pero no podemos crear la Web del mañana si utilizamos normas de diseño y programación del ayer.

Hay dos formas: compatibilidad directa transicional (una suculenta mezcla de técnicas tradicionales y otras basadas en estándares) y compatibilidad directa estricta basada en una completa (o casi completa) separación de estructura, presentación y comportamiento.

La compatibilidad directa transicional acepta la realidad del entorno mixto de navegación actual. Es muy indicada para proyectos en los que la marca es una prioridad y los navegadores no compatibles constituyen una parte importante del público de destino. La compatibilidad directa estricta, como su nombre indica, se ciñe al espíritu de los estándares, es el enfoque más compatible y proporciona las mayores ventajas cuando se utiliza en los contextos adecuados. A continuación, analizaremos con mayor detalle lo que supone cada uno de estos enfoques.

Compatibilidad directa transicional

Ingredientes

- XHTML válido para marcado (también se puede utilizar HTML 4.01).
- CSS válidas para controlar la tipografía, los colores, los márgenes, etc.
- Ligero uso de tablas XHTML en diseños, lo que evita la anidación ya que CSS se encarga de parte del trabajo.
- Opcional: etiquetas estructurales aplicadas a celdas de tablas (facilita el uso de CSS y secuencias de comandos, y contribuye con la transición prevista para el próximo año a diseños CSS menos centrados en tablas).
- JavaScript/ECMAScript basado en DOM, posiblemente con división de código para aceptar las versiones 4.0 de IE y Navigator.
- Atributos y pruebas de accesibilidad.

Recomendable para

La compatibilidad directa transicional es recomendable para sitios visitados por un alto porcentaje de navegadores 4.0 y anteriores que simplemente no admiten adecuadamente CSS, ni el DOM. También resulta válida en aquellos casos en los que las tablas

son más indicadas para el diseño que CSS. El enfoque transicional se aplica en la bibliotecas filiales de The New York Public Library (véase figura 2.18) para acomodar a una amplia comunidad de usuarios de Netscape 4.0 y, al mismo tiempo, cumplir con los estándares XHTML y CSS, sin dejar de lado la accesibilidad y la viabilidad a largo plazo.

Ventajas

- Compatibilidad inversa racional. Se pueden crear sitios que se muestren con una calidad razonable incluso en los navegadores más antiguos. Siempre se verán mejor en navegadores nuevos y más compatibles (se puede argumentar que incluso es más indicado ya que sutilmente promueve la puesta al día).
- Compatibilidad directa. Los sitios seguirán funcionando en futuros navegadores y dispositivos.
- Se empieza a preparar el camino a la posterior transición hacia un marcado basado en XML y un diseño CSS puro.

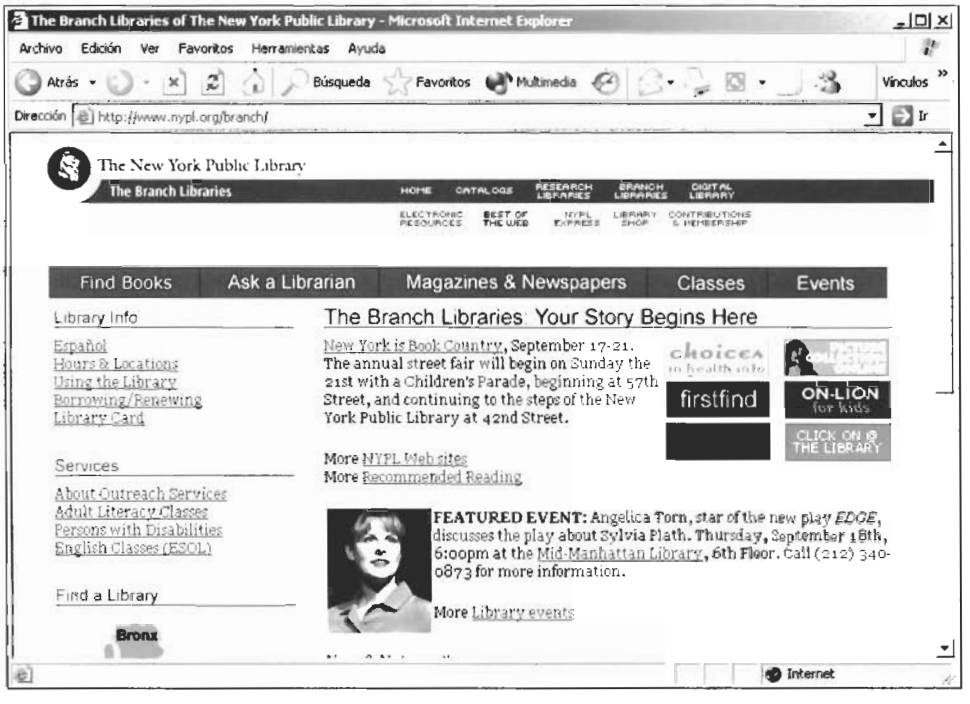


Figura 2.18.

La New York Public Library (www.nypl.org/branch/) adopta un enfoque transicional de los estándares Web: XHTML válido con un ligero uso de tablas para el diseño, accesibilidad de Prioridad 1 de la WAI y CSS para tipografía, márgenes y colores. Compatible tanto directa como inversamente, la compatibilidad directa transicional es una conversión de estándares sencilla que resulta indicada para muchas organizaciones.

- Se reducen los problemas de mantenimiento en los navegadores actuales ya que se elimina el marcado y el código basura.
- Se aumenta la accesibilidad. Se reducen los problemas relacionados con la accesibilidad y la pérdida de clientes y riesgos de litigio.
- Se restablece parcialmente la estructura de documentos en los documentos (y decimos parcialmente porque el marcado todavía incluye algunas estructuras de diseño).
- Se restablece la elegancia, la claridad y la simplicidad del marcado, con la consecuente reducción del tamaño de los archivos, un menor gasto del ancho de banda y una reducción de los costes de entrega, producción y mantenimiento.

Inconvenientes

- La estructura y la presentación siguen agrupadas de forma conjunta, lo que dificulta y aumenta el coste del mantenimiento y la actualización de los sitios.
- Por el mismo motivo, resultará más complicado y más costoso crear sitios diseñados de esta forma en futuros sistemas de administración de contenidos basados en XML. Es poco probable que se convierta en un problema para sitios de pequeño tamaño pero sí para sitios de contenidos a gran escala y sitios comerciales que cuenten con cientos o miles de páginas generadas dinámicamente.

Compatibilidad directa estricta

Ingredientes

- Separación completa entre estructura y presentación y comportamiento.
- CSS válidas utilizadas en el diseño. Las tablas se utilizan solamente para su función original: la presentación de datos en formato tabular como los que se incluyen en hojas de cálculo, libretas de direcciones, listados de eventos, etc.
- XHTML válido 1.0 Strict o Transitional utilizado para el marcado.
- Énfasis en la estructura. No hay restos de presentación en el marcado (estricto) o el menor número posible de restos de presentación en el marcado (de transición).
- Etiquetado estructural/abstracción de los elementos de diseño ("menú" en lugar de "Cuadro verde").
- Secuencias de comandos basadas en DOM para el comportamiento. División limitada del código, únicamente si resulta necesario.
- Atributos y comprobación de accesibilidad.

Recomendable para

La compatibilidad directa estricta es recomendable para todos aquellos sitios que no sean visitados por un alto porcentaje de navegadores no compatibles (habitualmente 2.0 y anteriores). Esta opción permite que los navegadores no compatibles puedan acceder

al contenido, pero es menos adecuada para representar el comportamiento en dichos navegadores.

Ventajas

- Compatibilidad directa. Aumento de la compatibilidad en navegadores y dispositivos existentes y futuros (incluyendo dispositivos inalámbricos).
- Transición hacia formas más avanzadas de marcado basado en XML.
- Llega a un mayor número de usuarios con menos trabajo.
- No hay versiones.
- Menos problemas de accesibilidad (en caso de que se produzca alguno). Generalmente, el contenido de los sitios diseñados de esta forma es accesible para todos.
- Restablece la elegancia, la simplicidad y la lógica del marcado.
- Restablece la estructura de documentos en los documentos.
- La producción y el mantenimiento resultan más rápidos y menos costosos. Como cuesta menos producir y mantener un sitio, se puede evitar la carga de presupuestos reducidos mientras que los presupuestos más extensos (si los hubiere) se pueden dedicar al diseño, programación, arte, fotografía, edición y comprobación de la utilidad del sitio.
- Resulta más sencillo de incorporar en sistemas de edición dinámicos y de administración de contenidos dirigidos por plantillas.
- El formato CSS permite conseguir diseños que no se pueden realizar con tablas HTML.
- Los sitios funcionarán en posteriores navegadores y dispositivos.

Inconvenientes

- Es muy probable que el aspecto de los sitios sea muy simple en navegadores antiguos.
- La compatibilidad con CSS en los navegadores es imperfecta. Puede que se necesiten algunas modificaciones.
- Algunas técnicas que resultan más fáciles de aplicar con tablas HTML se complican (o resultan imposibles) si se utiliza un diseño CSS. Puede que sea necesario modificar algunos diseños.
- Algunos navegadores anteriormente compatibles (como las versiones de Opera anteriores a la 7, por ejemplo) puede que se bloqueen con comportamientos basados en DOM.
- Los comportamientos basados en DOM no funcionarán en los principales navegadores 4.0 o anteriores, ni en lectores de pantalla, navegadores de texto y la mayoría de dispositivos inalámbricos. Será necesario usar etiquetas `<nonscript>` y CGI para conseguir funcionalidad adicional en dichos navegadores y dispositivos.

En capítulos posteriores veremos cómo funcionan los estándares (tanto individual como colectivamente) y ofreceremos consejos y estrategias para resolver problemas empresariales y de diseño relacionados con los distintos tipos de programación Web. Pero antes de eso, nos detendremos a considerar algunas de las dudas que puede que hayan surgido.

Si los estándares aumentan la compatibilidad, mejoran la accesibilidad y racionalizan la producción y el mantenimiento, evitan que se malgaste de ancho de banda y reducen los costes, ¿por qué los diseñadores y programadores no los usan de forma correcta y coherente en todos los sitios que crean?

¿Por qué no reclaman todos los clientes el cumplimiento de los estándares Web de la misma forma que en las antiguas películas de cárceles los presos hacían sonar sus tazas contra los barrotes de la celda? ¿Por qué hemos tenido que escribir un libro como éste para que lo lea, además de todos sus clientes, colegas, jefes y vendedores? ¿Por qué no se utilizan más los estándares Web?

Por una extraña coincidencia, en el siguiente capítulo nos ocuparemos de esto.

Capítulo 3
El problema de los estándares Web

Los estándares Web tienen la llave para acceder a un diseño y a una programación Web accesible y de bajo coste, algo que no se puede saber si tomamos como referencia la mayor parte de los sitios de los últimos años, tanto creativos como comerciales. En este capítulo analizaremos algunos de los motivos por los que los estándares Web no se han incorporado todavía a la práctica normativa de todas las agencias de diseño y divisiones Web, y siguen sin ser componentes obligatorios de todos los proyectos o propuestas de sitios.

Si prefiere leer historias sobre el éxito de los estándares Web puede consultar un capitulo posterior. En otro capítulo encontrará técnicas para ponerse manos a la obra. Pero si necesita ayuda para vender los estándares a sus colegas o si simplemente quiere saber cómo una industria puede conseguir estándares sin utilizarlos, se encuentra en el capítulo correcto.

Un aspecto agradable, un código repelente

A mediados del 2002, con otros seis miembros de la comunidad de nuevos medios, formé parte del jurado de los Eighth Annual Communication Arts Interactive Awards (www.commarts.com), posiblemente el concurso de diseño más prestigioso del sector. Los sitios y proyectos participantes se encontraban entre los de mejor diseño y programación de ese año.

En un primer momento, los jueces dedicamos 10 semanas a la revisión de miles de sitios Web y CD-ROM, para reducir el abanico a unos cientos de finalistas de los que

apenas 50 quedaron seleccionados como ganadores. La votación final tuvo lugar en Bay Area, donde estuvimos confinados durante una semana. Hasta la elección de los ganadores, nadie podía salir. Al final de la semana, habíamos seleccionado 47 proyectos ganadores y por fin nos vimos liberados.

Para celebrar el final de la votación (y la consiguiente libertad), quedé con un amigo para cenar en San Francisco. Le interesaba mucho todo lo del concurso y tenía ciertos conocimientos sobre programación Web.

Me preguntó si había quitado puntos a los sitios que no eran compatibles con estándares. A lo que le respondí que ninguno lo era.

Una aplastante realidad. De los miles de sitios a concurso, ninguno se había creado en HTML estructural válido. Muchos de ellos eran visualmente impactantes (véase figura 3.1) y estaban brillantemente programados (véase figura 3.2); algunos ofrecían contenidos bien escritos y sólo algunos resultaban originales. Pero ninguno de ellos mostraba un atisbo de marcado estructural válido, CSS compactas o secuencias de comandos basadas en estándares.

Más de la mitad de los sitios enviados se habían programado enteramente en Flash. Del resto, la mayoría sólo funcionaba en navegadores 4.0, únicamente en IE4 o en Netscape 4. Algunos únicamente funcionaban en Windows. De los cientos de finalistas, la mayoría de ellos estaban producidos con mucho lujo (y de forma cara) y todos, con su propia forma de representar los mejores esfuerzos profesionales de la industria, apenas usaban los estándares Web lo más mínimo.

Objetivos comunes, medios comunes

Los sitios Web enviados a Communication Arts eran muy diversos en lo que respecta a objetivos de creación y de marcado, pero la mayoría compartía ciertos objetivos subyacentes, los mismos que los suyos o los míos. Todos queremos que nuestros sitios atraigan al público de destino, promuevan la participación, resulten sencillos de comprender y de utilizar, y que transmitan todo lo bueno de nuestra organización, producto o servicio, no sólo en palabras, sino también a través del aspecto y del funcionamiento del sitio.

A todos nos gustaría obtener el máximo rendimiento del dinero de nuestros presupuestos. Queremos que nuestros sitios funcionen para el mayor número de usuarios y en el mayor número de entornos posibles. Intentamos evitar incompatibilidades con navegadores y plataformas, y estar al menos un paso por delante de la guadaña del cambio tecnológico.

A todos nos gusta crear un sitio que funcione correctamente el día de mañana sin sufrir una continua y costosa carga tecnológica, como comentamos en el capítulo anterior. Preferíamos dedicar nuestro escaso tiempo a actualizar contenidos y a añadir servicios en lugar de volver a diseñar el código de nuestros sitios cada vez que aparece un nuevo dispositivo o navegador.

Los estándares son la clave para conseguir estos objetivos. La pregunta es por qué no han invadido con fuerza la comunidad de programación.

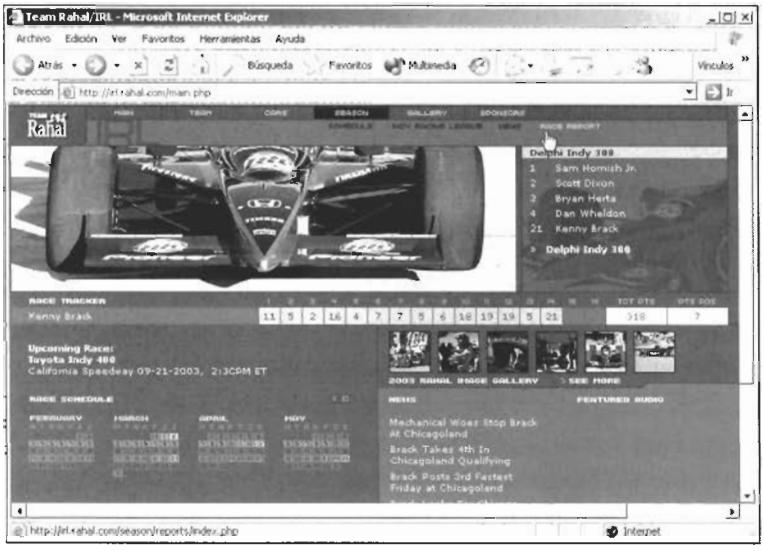


Figura 3.1.

Team Rahal, Inc (www.rahal.com), uno de los ganadores del Eighth Annual Communication Arts Festival, es de gran belleza, impresionante e incompatible con estándares.

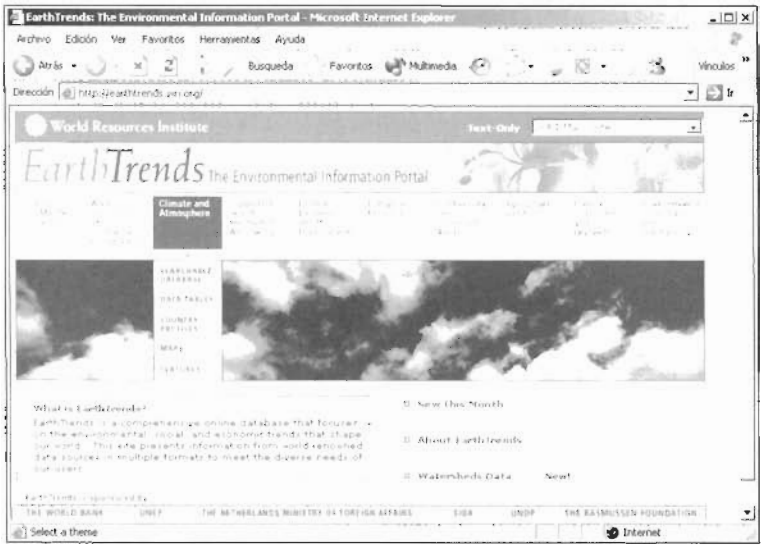


Figura 3.2.

World Resources Institute (http://earthtrends.wri.org/), otro ganador visualmente atractivo, sabiamente programado pero que tampoco utiliza estándares Web.

Percepción frente a realidad

Por un lado, como ocurre con la accesibilidad, muchos diseñadores opinan erróneamente que los estándares Web son de algún modo hostiles o antiéticos para las necesidades del buen diseño gráfico. Por otro, los que crean estándares Web no se encargan de venderlos: los sitios de W3C o ECMA (véase figura 3.3), visual y arquitectónicamente pedestres, no ofrecen ningún atractivo ni inspiración para los estilistas gráficos ni para los diseñadores orientados a consumidores. Y el aspecto poco profesional de estos sitios no contribuye a combatir el mito de los estándares como algo poco ético para el diseño visual. Únicamente los sitios de gran belleza que utilizan estándares (véase figura 3.4) pueden invertir esta falsa percepción.

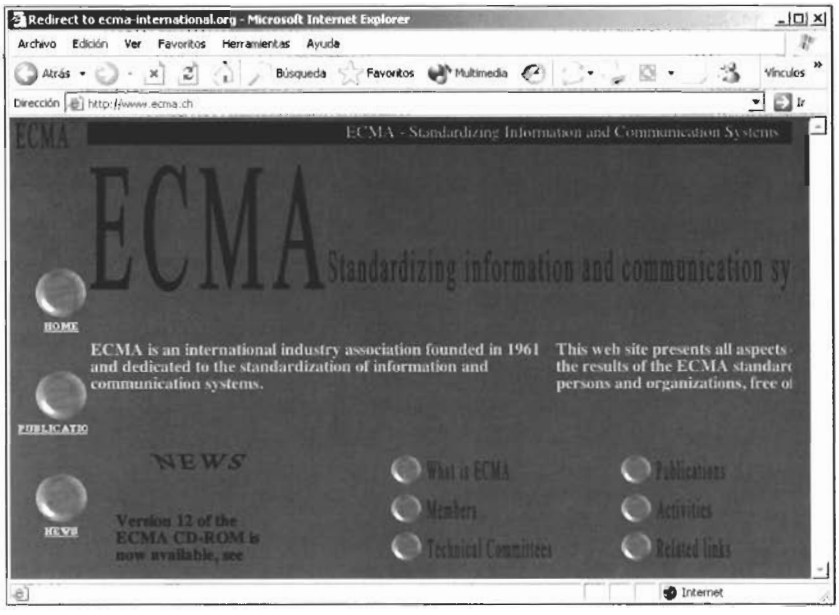


Figura 3.3.

ECMA (European Computer Manufacturers Association) es un organismo de estándares y cuna de grandes pensadores. Desafortunadamente, ninguno de ellos posee la mínima competencia en lo que se refiere a diseño gráfico o arquitectura de sitios pensados para los usuarios. Por ello, es poco probable que el sitio de ECMA (www.ecma.ch) aliente a los diseñadores a que aprendan algo sobre ECMAScript u otros estándares (compárelo con las figuras 3.1 y 3.2, que representan el tipo de aspecto que inspira a los diseñadores visuales; más adelante encontrará más información sobre ECMA).

Figura 3.4.

Kaliber 10000 (K10k), un portal de diseño que se lee con avidez (www.k10k.net), se ha construido con XHTML válido, CSS y el Modelo de objetos de documento (DOM) de W3C. Aunque no fue el primero en adoptar los estándares, el uso que hace de los mismos es importante, ya que muestra a la comunidad de diseño que son compatibles con un buen diseño gráfico.

Al mismo tiempo, los programadores y diseñadores que no se han parado a aprender las diferentes variedades del lenguaje de secuencia de comandos propietario, puede que no le vean el sentido a aprender algo nuevo, o puede que estén muy ocupados con JSP, ASP o .NET como para pensar en cambiar sus técnicas básicas. Los que dependen de editores WYSIWYG para realizar su trabajo tendrán un motivo diferente para no utilizar estándares. Principalmente, dependen de editores WYSIWYG; por lo tanto, es poco probable que sepan que ahora dichos editores admiten estándares. Multitud de experimentados programadores utilizan herramientas WYSIWYG como Dreamweaver y GoLive, pero también lo hacen otros no tan experimentados a los que les resultaría imposible crear una sencilla página Web sin acceso a dichas herramientas. Por último, decir que recientemente los navegadores han empezado a contar con cierta compatibilidad con estándares. Muchos profesionales Web están tan acostumbrados a hacer las cosas a la fuerza que no se han dado cuenta que los navegadores han cambiado. En primer lugar nos detendremos en esta última razón.

Año 2000: el año de los navegadores

Con la aparición de IE5 para Macintosh en marzo del 2000, el mundo (o al menos la parte del mundo que utiliza Mac) obtuvo algo más que una muestra de lo que significan los estándares Web. IE5/Mac admitía XHTML, ECMAScript, prácticamente toda la especificación CSS1, gran parte de CSS2 y la mayoría del DOM. También podía representar XML puro, aunque no está claro si esto le interesaría a alguien (en un capítulo posterior podrá encontrar más información al respecto o bien puede visitar Bugzilla en la direccion http://bugzilla.mozilla.org/show_bug.cgi?id=64945, donde verá algunos de los enfoques que han adoptado para solucionar el problema de XML en los navegadores).

IE5/Mac: cambios y zoom

IE5/Mac estaba de tal forma en armonía con los estándares que variaba su representación y rendimiento en función del `<!DOCTYPE>` que aparecía en la parte superior del marcado de la página, una técnica denominada conmutación de DOCTYPE que analizaremos en capítulos posteriores. En pocas palabras, con el DOCTYPE correcto, una página funcionaría y se mostraría como los estándares Web dicen que debería. Con un DOCTYPE antiguo o parcial (o inexistente), la página se representaría en modo de compatibilidad inversa para evitar afectar a sitios no compatibles con estándares, es decir, para ser amable con el 99'9 por ciento de los sitios comerciales de la Web, al menos hasta el momento (véase figura 3.5). IE5/Mac también incluía una característica denominada Zoom de texto (véase figura 3.6) que permitía a los usuarios aumentar o reducir cualquier texto Web, incluyendo el configurado en píxeles por medio de CSS, lo que solventaba el eterno problema de la accesibilidad. Antes de IE5/Mac, sólo el navegador Opera permitía a los usuarios aumentar o reducir texto Web, incluido el configurado en píxeles. Opera aplicaba el zoom en toda la página, gráficos y demás elementos, un innovador enfoque del conflicto entre lo que el diseñador desea y lo que el usuario puede necesitar (véanse figuras 3.7, 3.8 y 3.9).

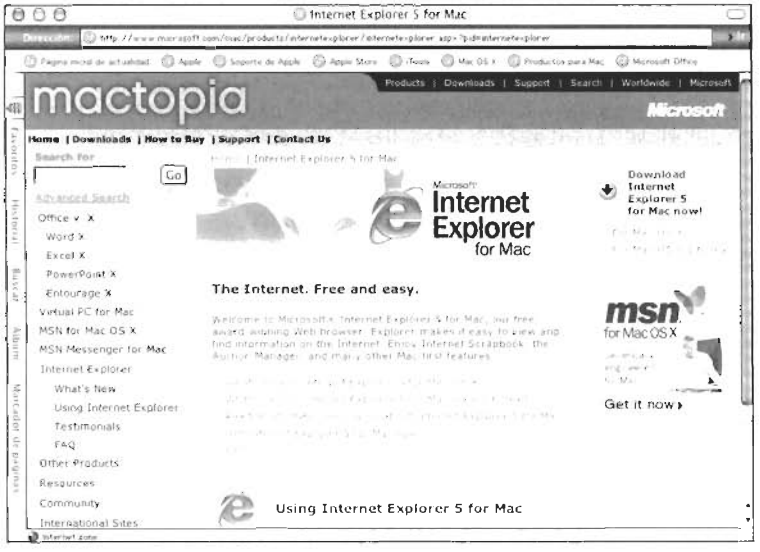


Figura 3.5.

Hola mundo, es IE5 para Macintosh, el primer navegador en utilizar correctamente la mayoría de los estándares Web y cuyas innovaciones se asentaron en productos de la competencia (www.microsoft.com). Algunas de estas innovaciones pasaron incluso a IE para Windows. Pero desafortunadamente no todas ellas.

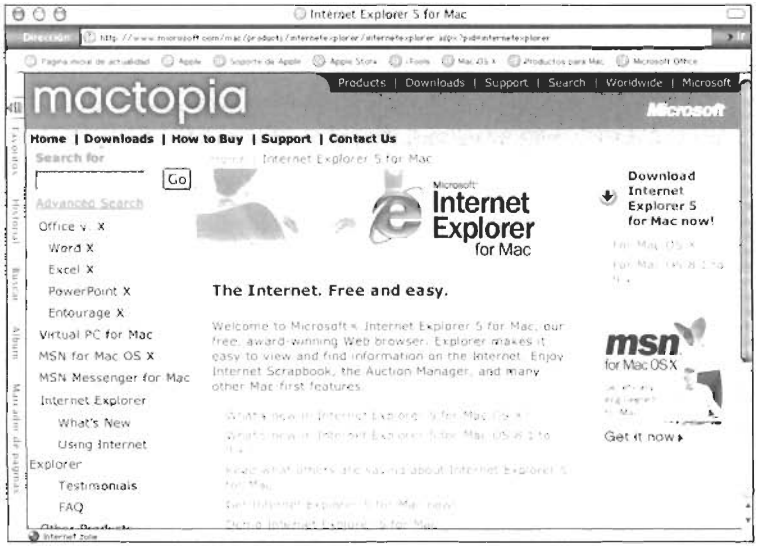


Figura 3.6.

El Zoom de texto de IE5/Mac en acción. Con tan sólo pulsar una tecla de comando o un menú desplegable, los usuarios pueden aumentar o reducir el texto de cualquier página Web, ya sea texto configurado en píxeles, puntos, centímetros o cualquier otra unidad relativa o absoluta. Las imágenes de la página no se ven alteradas, sólo cambia el tamaño del texto. Esta opción pronto pasó a Netscape, Mozilla, Camino y otros importantes navegadores compatibles con estándares. Desafortunadamente, tres años después de que IE para Macintosh presentara esta función, IE para Windows sigue sin ofrecerla.

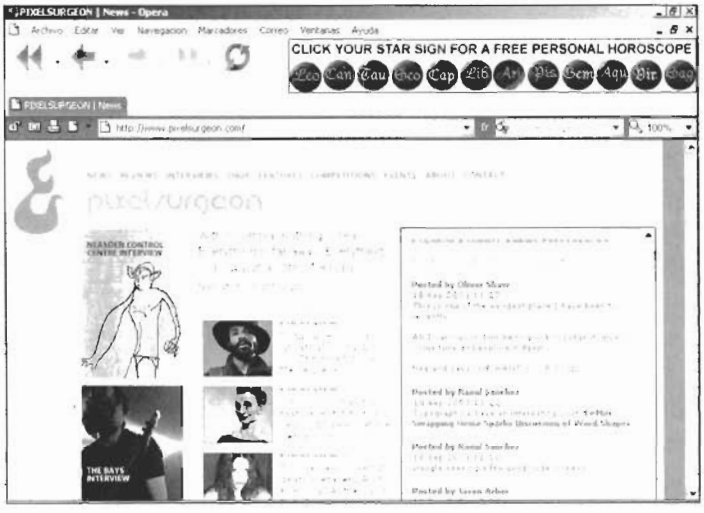


Figura 3.7.

PixelSurgeon (www.pixelsurgeon.com/news/), un importante portal de diseño y hermano de K10k (véase figura 3.4) visto en el navegador Opera 7 de Opera Software a tamaño real.

Figura 3.8.

El mismo sitio aumentado a 200 por ciento con ayuda de la opción Zoom de página de Opera. Para los usuarios impedidos visualmente, esta opción constituye una forma de leer pequeños botones gráficos y otros gif de texto cuyas minúsculas fuentes serían un problema de accesibilidad.

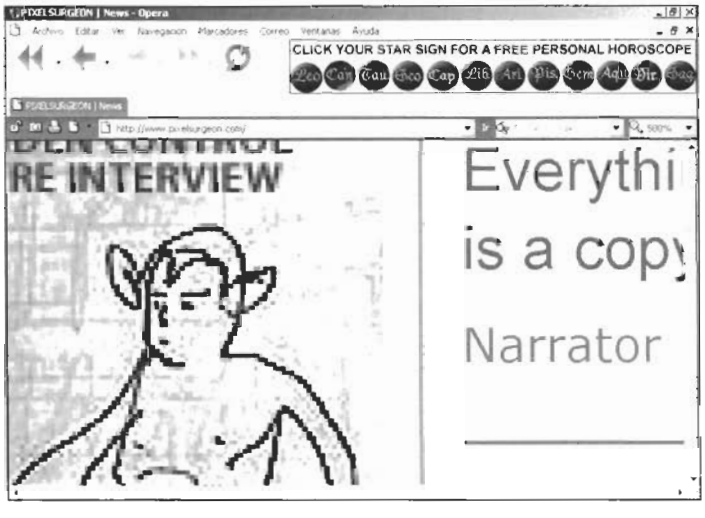


Figura 3.9.

El mismo sitio otra vez, aumentado al 500 por ciento. Para los diseñadores, esta función le permite estudiar con más detalle los elementos visuales de un sitio sin tener que guardarlos primero en forma de capturas de pantalla para después abrirlos en Photoshop.

El temerario movimiento de Netscape

Tras el lanzamiento de IE5/Mac aparecieron multitud de navegadores compatibles con estándares. Netscape 6 y su pariente de código abierto Mozilla admitían XML, XHTML, CSS, ECMAScript y el DOM en las diferentes plataformas informáticas. También usaba la conmutación de DOCTYPE y ofrecía Zoom de texto, y se había diseñado para ser un navegador totalmente compatible.

Para conseguir una compatibilidad con estándares total, como recomendaba WaSP, Netscape se deshizo temerariamente de su navegador Navigator/Communicator 4.0 y de cualquier atisbo de código legado que incluyera, para empezar desde cero. La creación de un navegador completamente nuevo le llevó más tiempo que si se hubiera actualizado uno ya existente. Netscape perdió una parte considerable de su cuota de mercado durante los años que duró la creación de Mozilla/Netscape 6, que comenzó en 1998 pero que no consiguió un producto comercial hasta finales del 2000 (y podemos decir que no obtuvo un producto comercialmente viable hasta el 2002).

Estos administradores e ingenieros no estaban locos. Evidentemente creían, al igual que WaSP, que el nuevo navegador estaría listo en un año. Cuando este año se convirtió en dos y luego en tres, los administradores e ingenieros resistieron con una determinación heroica hasta ver completado su trabajo.

Muchas empresas habrían abandonado un proyecto como éste, habrían encogido los hombros y habrían comercializado un navegador 5.0 no estándar construido con código legado en lugar de sacrificar tiempo y cuota de mercado ante un competidor tan fiero como Microsoft. Aunque los accionistas no estén de acuerdo, la administración y los ingenieros de Netscape merecen nuestro agradecimiento por anteponer la compatibilidad y la salud futura de la Web a beneficios a corto plazo y a su propio interés.

Se abren las compuertas

El siguiente en aparecer fue Opera 6, sin conmutación de DOCTYPE ni DOM, pero compatible con el resto de estándares. Opera renunció a la conmutación de DOCTYPE ya que, entre todos los navegadores comerciales, era el único que siempre había intentado mostrar páginas acordes a las especificaciones del W3C. Por ello, los creadores de Opera no vieron el sentido de ofrecer un modo de compatibilidad inversa (Opera 5 y 6 no admitían el DOM W3C estándar pero Opera 7, que apareció en el 2002, sí).

Por último, Microsoft lanzó IE6 para Windows, un navegador que se puso al día con la precisa representación CSS de su versión para Macintosh y que ofrecía gran compatibilidad para XML, ECMAScript y el DOM, y que combinaba IE5/Mac, Mozilla y Netscape 6+ al ofrecer conmutación de DOCTYPE. (En el lanzamiento del navegador, la prensa especializada en favor de Windows se fijó finalmente en la conmutación de DOCTYPE y otorgó a IE6/Windows el mérito que se merecía.)

IE6/Windows no implementaba Zoom de texto, pero esta opción, aunque muy deseable desde el punto de vista de la accesibili-

dad, es una innovación, no un estándar. En IE6/Windows los fondos fijos de CSS eran incorrectos y también presentaba un fallo (aún lo tiene) que puede dividir los diseños CSS que utilizan la propiedad `float` (como veremos en un capítulo posterior). A pesar de todo esto, se trataba de un navegador altamente compatible y bien diseñado, y su predecesor IE5.x/Windows se le acercaba bastante en cuanto a funcionamiento se refiere.

Ninguno de estos navegadores era perfecto (ningún programa lo es), pero cada uno de ellos era un logro que demostraba el compromiso adquirido por la compatibilidad a través de estándares. Nadie, ni siquiera el proyecto de estándares Web, hubiera pensado que estas empresas llegarían tan lejos y conseguirían tanto. Con los principales navegadores admitiendo una especie de paridad en su cumplimiento de los estándares, los diseñadores y los programadores podían utilizar diseños CSS y otras técnicas basadas en estándares con total libertad.

¿Demasiado poco, demasiado tarde?

La aparición de navegadores corporativos con una sólida compatibilidad fue una excelente noticia para los usuarios y los creadores de la Web. Pero cuando finalmente llegaron estas noticias, muchos diseñadores y programadores se habían convencido de que los estándares Web eran un sueño, y muchos habían abandonado la idea de implementarlos correctamente. Es fácil entender por qué.

2000-2001: LA VANGUARDIA DE LOS ESTÁNDARES

Antes del lanzamiento de IE6/Windows, una vanguardia de diseñadores y programadores comenzaron a utilizar diseños CSS y otras técnicas basadas en estándares, incluyendo la programación de secuencias de comandos basada en DOM. Algunos conseguían forzar a IE5/Windows a que representara CSS de forma precisa y desactivaron CSS en navegadores 4.0 porque no podían representarlas de forma correcta. Otros adoptaron un enfoque progresivo, permitiendo que todos los navegadores tuvieran algo que mirar mientras reservaban el diseño completo para los nuevos navegadores más compatibles. En un capítulo posterior nos ocuparemos de esta vanguardia del diseño. Sus soluciones y estrategias se analizarán en otra parte del libro.

CSS: la primera ronda es gratis

La especificación CSS1 apareció en Navidad de 1996. Meses después, apareció IE3, con una rudimentaria compatibilidad con CSS entre sus características. La compatibilidad con CSS (inexistente en Netscape 3) otorgó al navegador de Microsoft el primer atisbo de credibilidad en una época de dominio de Netscape Navigator. IE3 admitía lo bastante de CSS como para deshacerse de las etiquetas `<font>` no estándar y empezar a experimentar con márgenes, interlineado y otros rudimentos del diseño CSS. Sorprendidos con lo que vieron en las páginas de demostración de Microsoft (véase figura 3.10), en

las que se exponían las nuevas prestaciones del navegador, muchos diseñadores se adentraron por primera vez en el diseño CSS.

La compatibilidad con CSS de IE3 fue un arriesgado primer paso, pero como todos los primeros pasos tenía fallos y era incorrecto. Los que utilizaban CSS por primera vez estaban exultantes por la libertad creativa que les ofrecía pero pronto se desanimaron por los primeros errores de IE que podían inutilizar una página. Por ejemplo, bajo determinadas circunstancias, las imágenes de una página controlada por CSS se situaban encima del texto en lugar de hacerlo a un lado del mismo. Para que se haga una idea de cómo funcionaba, ponga su mano sobre este párrafo e intente leer a través de la misma.

La solución de este primer problema de representación CSS de IE 3 consistía en incluir todas las imágenes y todos los párrafos en una tabla de celda, con lo que se duplicaba el tamaño de la página al tiempo que se derrotaba la función de CSS (para controlar el diseño sin tablas y sin exceso de ancho de banda). Los diseñadores concluyeron que CSS no estaba preparado todavía, una determinación que parecía razonable dada la ausencia de compatibilidad con CSS en el líder del mercado Netscape 3.

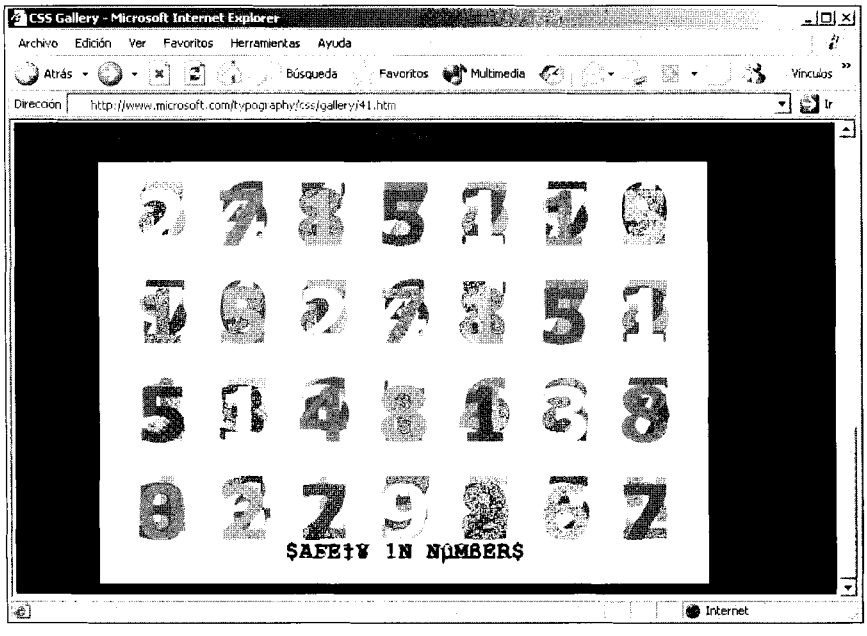


Figura 3.10.

Una página de la galería CSS de 1998 de Microsoft (http://www.microsoft.com/typography/css/gallery/). El texto y todos los demás efectos de diseño están creados totalmente en CSS, sin imágenes GIF ni JPEG. IE podía mostrar estos efectos; Netscape 3 (líder del mercado en aquel entonces) no. La CSS de la galería usaba valores incorrectos requeridos por el imperfecto motor CSS de IE3 y la compatibilidad general con estándares era escasa, pero el genio había salido de la lámpara. Viendo lo que CSS podía hacer, muchos de nosotros no volvimos a mirar atrás.

Los navegadores erróneos conducen a prácticas erróneas

Aparecieron entonces los navegadores 4.0. Aunque con errores e incompleto, IE4 mejoró considerablemente la compatibilidad con CSS que ofrecía IE3. Netscape 4 ofrecía CSS por primera vez en una implementación de última hora tan dispersa que pospuso la adopción de CSS otros dos años más.

Para ser justos, la compatibilidad con CSS de Netscape 4 era mucho mejor que la de IE3 (http://www.webreview.com/style/css1/leaderboard.shtml). Pero aunque prácticamente nadie utiliza IE3 hoy en día, millones de usuarios siguen utilizando Netscape 4. Por ello, muchos propietarios de sitios se vieron obligados a admitir Netscape 4 y confundieron "compatibilidad", algo positivo, con "una igualdad de píxeles perfecta y un comportamiento idéntico", algo negativo, ya que limita a los programadores y les obliga a escribir código erróneo y marcado sin utilidad.

La maldición de la representación de legado

Entre los principales fallos de Netscape 4 destacamos las representaciones de legado y la falta de herencia.

Diseñadas para abstraer presentación de estructura, las CSS no asumen cómo se supone que deben representarse los elementos o incluso qué lenguaje de marcado se va a utilizar, aunque navegadores y otros agentes se encargan normalmente de dichas suposiciones. (Algunos navegadores modernos utilizan CSS para reforzar sus suposiciones y permiten que las hojas de estilo del diseñador las reemplace). De forma predeterminada, en la mayoría de los navegadores, sin CSS, el título `<h1>` tendría un gran tamaño y aparecería en negrita, con márgenes verticales (espacio en blanco) por encima y por debajo.

CSS le permite cambiar todo esto. Con CSS, `<h1>` puede tener un tamaño reducido, aparecer en cursiva y carecer de márgenes si el diseñador lo cree conveniente. Desafortunadamente no ocurre lo mismo en Netscape 4, que añade sus representaciones legadas predeterminadas a todas las reglas CSS que especifique el diseñador. Si la CSS indica que no debe haber ningún espacio en blanco por debajo del título, Netscape 4 lo añadirá de todas formas.

Cuando los diseñadores aplicaron CSS al marcado HTML estándar, descubrieron que IE4 hacía principalmente lo que se le decía, mientras que Netscape 4 trastocaba todos los diseños.

Algunos diseñadores abandonaron CSS. Otros (entre los que me incluyo) solucionaron inicialmente el problema descartando el marcado estructural, utilizando construcciones como `<div class="headline1">` en lugar de `<h1>`. Con esto se resolvió el problema de representación a expensas de la estructura de documentos y la semántica y, en consecuencia, se antepusieron logros a corto plazo en lugar de viabilidad a largo plazo, con los problemas que esto conlleva.

Ahora se ven los malos resultados de dichos problemas.

Este autor ha abandonado hace tiempo la práctica de deformación estructural de documentos, pero un gran número de diseñadores y programadores siguen escribiendo marcado basura en nombre de la compatibilidad inversa con Netscape 4. Esta práctica normativa es problemática y genera problemas de utilización al tiempo que se bloquean los esfuerzos para normalizar y racionalizar flujos de trabajo dirigidos por datos.

Los sistemas de administración de contenidos, herramientas de edición y editores Web visuales (conocidos como editores WYSIWYG) desarrollados durante la era de los navegadores 4.0 están repletos de marcado sin sentido que aumenta considerablemente la dificultad y los gastos de adaptación de los sitios a los estándares actuales o la preparación de contenido legado para bases de datos dirigidas por XML. En sitios de gran tamaño creados por varios diseñadores y programadores, cada uno puede usar diferentes etiquetas no estándar, lo que imposibilita agrupar todos los datos y cambiar su formato de acuerdo a un esquema de mayor utilidad. (Imagine una biblioteca pública en la que los libros no estén clasificados mediante un sistema decimal sino de acuerdo a las preferencias de distintos catalogadores, cada uno con sus propias normas.)

Fuera del ámbito de los navegadores gráficos, el marcado estructural sin sentido también reducía la utilidad de las páginas. Para un usuario de una Palm, un teléfono Web o bien un lector de pantalla `<div class="headline1">` es texto sencillo, no un título. Por ello, compramos o diseñamos sistemas de administración de contenidos que sustituyen un conjunto de etiquetas por otro cuando bastaría con un solo conjunto de etiquetas estándar. O también podríamos obligar a los usuarios de Palm, teléfonos Web y lectores de pantalla a que vean el marcado estructural y adivinen su significado.

Podemos agradecer a Netscape 4 (y a nuestra propia voluntad por aceptar sus errores) por enfangarnos en este desastre. Ahora se explica por qué los ingenieros de Netscape y Mozilla siguieron trabajando en el proyecto de cuatro años de Mozilla. Realmente no tenían ningún producto legado que mereciera la pena utilizar como referencia.

Herencia del viento

Netscape 4 tampoco comprendía ni era compatible con la herencia, el brillante concepto subyacente que otorga a CSS toda su fuerza. CSS racionaliza la producción y reduce el ancho de banda al permitir la aplicación de reglas generales en el árbol de documentos a menos que el diseñador especifique lo contrario.

Por ejemplo, en CSS, puede aplicar un tipo de fuente, un tamaño y un color al selector `<body>` y la misma fuente, tamaño y color se mostrarán en todos los secundarios de dicha etiqueta, desde `<h1>` a `<p>`, pero no en Netscape 4. En Netscape 4, 2+2 es igual a 2+2 no a 4.

Los programadores experimentados consiguieron solucionar la falta de compatibilidad del navegador con respecto a la herencia escribiendo reglas redundantes:

```
body, td, h1, p   {font-family: verdana,
arial, helvetica, sans-serif;}
```

En el ejemplo anterior, los selectores `td`, `h1` y `p` son redundantes porque cualquier navegador compatible aplica a estos elementos secundarios el mismo estilo que tenga el elemento principal.

Los programadores menos experimentados aplicaron sus normas de forma completa, lo que aumentaba la redundancia y provocaba un gasto de ancho de banda mayor:

```
body  {font-family: verdana, arial,
       helvetica, sans-serif;}
td    {font-family: verdana, arial,
       helvetica sans-serif;}
h1    {font-family: verdana, arial,
       helvetica sans-serif;}
p     {font-family: verdana, arial,
       helvetica sans-serif;}
```

y así sucesivamente. Era una pérdida de ancho de banda para usuarios y servidores, pero bastaba para realizar el trabajo. Otros programadores concluyeron que CSS no funcionaba en Netscape 4 (tenían razón) o que era imperfecto (no tenían razón, pero esta suposición se extendió rápidamente).

Netscape 4 presentaba otros errores relacionados con CSS, los suficientes como para llenar las Páginas Amarillas de cualquier ciudad, pero con éstos nos basta para hacernos una idea general y también para retrasar la adopción del estándar CSS.

El Sr. Comportamiento

Junto con este fallo de CSS, los primeros navegadores no se ponían de acuerdo en una forma de facilitar comportamiento sofisticado a través de secuencias de comandos. Todos los navegadores disponían de un Modelo de objetos que les indicaba el tipo de comportamientos que se podía aplicar a los objetos de la página. Netscape 4 contaba con un modelo propietario `document.layers`. IE4 utilizaba su propio modelo propietario `document.all`. Ninguno admitía el DOM del W3C, que estaba en proceso. Los programadores que querían aplicar comportamientos sofisticados (incluso básicos) a sus sitios, tenían que diseñar dos tipos de código que se ajustara a dichos navegadores. Para admitir a navegadores anteriores (compatibilidad inversa) era necesario más código y pasar por un mayor número de aros, como mencionamos en un capítulo anterior.

Los principales navegadores ni siquiera se ponían de acuerdo en un lenguaje común de secuencias de comandos. Al principio, Netscape inventó JavaScript y prometió lanzarlo como estándar para que otros fabricantes de navegadores pudieran admitirlo. Pero durante años, a pesar de su promesa, Netscape se aferró al secreto de JavaScript, ya que lo consideraba una ventaja con respecto a la competencia. (Si Navigator se convertía en el único navegador compatible con JavaScript, por qué nadie diseñaría para un navegador de la competencia menos potente, pensaba Netscape. En su lugar, es muy probable que Microsoft hubiera hecho lo mismo. De hecho, lo hizo con su tecnología ActiveX propietaria.)

Para competir, Microsoft invirtió el código de JavaScript y lo modificó, algo inevitable en este tipo de proyectos. El lenguaje resultante funcionaba como JavaScript pero no exactamente como JavaScript. El nuevo lenguaje era lo suficientemente diferente como para molestar al usuario. Microsoft lo denominó JScript. Mientras tanto, preparaba

una tecnología diferente llamada ActiveX que supuestamente proporcionaría una funcionalidad perfecta en todas las versiones de su navegador IE pero realmente sólo funcionaba bien en Windows, donde se sigue utilizando para cosas como buscar componentes extraviados.

JScript, JavaScript y ActiveX. En nombre de la compatibilidad inversa y entre navegadores, los programadores se encontraron con varias parejas de baile, y parecía que ninguna bailaba al mismo son, algo que sufrieron los clientes en forma de costes continuos de programación y pruebas.

Por fin un lenguaje de secuencias de comandos duradero

Con el tiempo, ECMA ratificó una versión estándar de JavaScript que modestamente llamaron ECMAScript (www.ecma.ch/ecma1/STAND/ECMA-262.HTM). Tras ello, el W3C emitió un DOM estándar. A la larga, Netscape y Microsoft admitieron ambos estándares, pero no antes de que años de dura incompatibilidad hubieran convertido a los programadores en expertos en técnicas de secuencias de comandos y modelos de objetos propietarios no compatibles, y de que hubieran convencido a los propietarios de sitios de que la programación Web siempre sería un asunto conflictivo. De ahí los sitios sólo para IE, la detección incorrecta de secuencias de comandos y, en algunos casos, el abandono de los estándares Web en favor de soluciones propietarias como Flash.

Por cierto, en caso de que se pregunte a qué se dedica la ECMA, no se moleste en buscar en el complicado y confuso sitio de la organización (véase figura 3.3). Lo que le interesa saber es que es la Asociación Europea de Fabricantes Informáticos y también un cuerpo de estándares que, al contrario que el W3C, denomina "recomendaciones" a las tecnologías que desarrolla en lugar de llamarlas "estándares". Los sitios confusos y las denominaciones desconcertantes son otra de las razones por la que los estándares lo han tenido difícil para conseguir aceptación en la Web.

Sitios confusos, denominaciones desconcertantes

Fíjese en la especificación CSS2 presentada por el W3C (véase figura 3.11). CSS2 es un potente lenguaje de presentación estándar creado para facilitar las necesidades de los diseñadores, algo que no se aprecia al examinar esta página.

Tiene una presentación tan poco alentadora como la página que hizo su tío el del pueblo en una tarde con ayuda de Microsoft FrontPage y un editor de imágenes de 50 €.

Conteniendo la sensación de terror, intente leer y comprender la especificación a pesar de su poco atractivo aspecto. Después de todo, el W3C está formado por científicos, no por diseñadores. Lo que importan son las palabras, ¿verdad? Veinte minutos después, con los ojos bizcos, se dirigirá hasta una tienda de informática en línea y comprará Macromedia Flash.

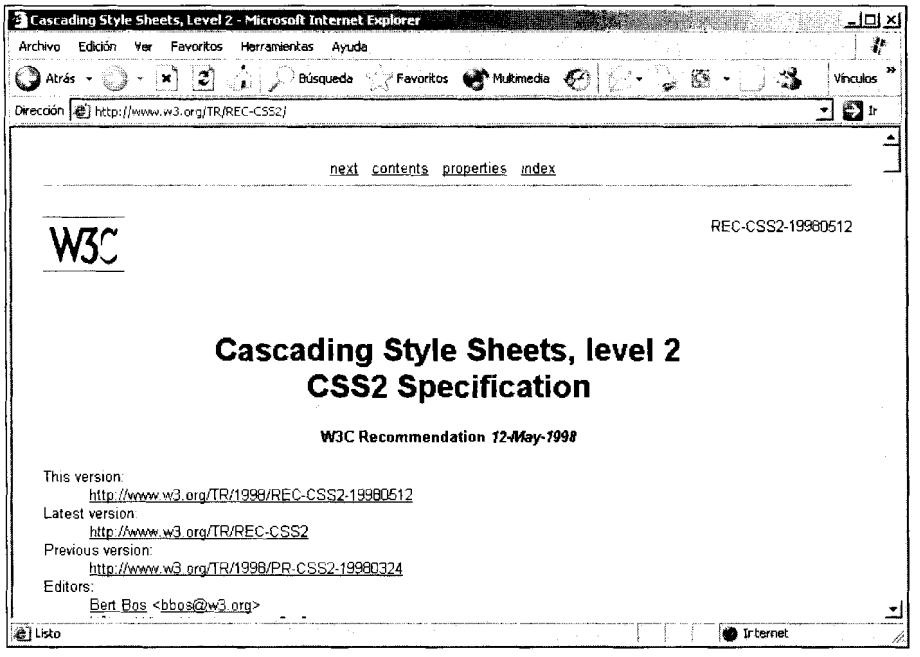


Figura 3.11.
La especificación CSS2 por el W3C (www.w3.org/TR/REC-CSS2/). Un prometedor lenguaje de presentación presentado de forma nada inspirada.

Para ser honestos, no sólo el W3C no parece muy ducho en diseño gráfico o arquitectura de la información, sino que tampoco está muy acostumbrado a escribir cursos prácticos adecuados para los diseñadores. El sitio W3C está formado por una serie de precisos documentos técnicos creados por importantes pensadores y científicos expertos, lo que siempre han querido que sea.

En la sección "How to Read W3C Specs" (www.alistapart.com/stories/readspec/) J. David Eisenberg, autor de O'Reilly y miembro del comité directivo de WaSP (y uno de los editores técnicos de este libro), lo explica de esta forma: "Cuando se buscan respuestas, se busca un manual del usuario o una guía de referencia; se quiere utilizar la tecnología. Pero éste no es el objetivo de una especificación W3C. El objetivo de una especificación es indicar a los programadores que implementarán la tecnología en cuestión, qué características debe incluir y cómo deben implementarlas. Es la diferencia entre el manual del propietario de su coche y los manuales de reparación."

Por definición, el W3C se dirige a ingenieros, no al público en general. No trata de explicar o vender los estándares que crea. Como dijimos antes, ni siquiera los denomina estándares, aunque realmente lo sean. (En recientes apariciones en la prensa y en partes del sitio W3C, el Consorcio ha empezado a utilizar la palabra "estándar" en lugar de "recomendaciones". Un buen comienzo.)

Al contrario de lo que sucede con una corporación, el W3C no se beneficia de que utilicemos un estándar Web y aconseja a sus empresas miembro que no realicen patentes (http://www.w3.org/TR/2002/WD-patent-policy-20021114/) ni cobren nada por los estándares Web o sus componentes.

Académicos frente a economistas

Alejado del mundo en el que las empresas luchan entre sí para conseguir eliminar a su adversario, el W3C constituye un espacio que se centra en el potencial de la Web en lugar de en la presión de la competencia. Las actividades del W3C son extrañas, hechas por extraños y para extraños, y su sitio refleja el énfasis del grupo en aspectos científicos en detrimento del estilo o de la facilidad de uso para los clientes.

El problema es que diseñadores, programadores y propietarios de sitios se preocupan enormemente del estilo y más aun de la facilidad de uso para los clientes. En sus propios sitios, nunca publicarían intencionadamente texto complejo o arcaico que confundiera a los lectores, nunca colocarían sus contenidos más importantes en sitios inaccesibles y nunca presentarían deliberadamente sus contenidos en un marco antiestético que obligara a sus visitantes a pulsar el botón Atrás.

Psicológicamente, la gente que se preocupa de la estética, la claridad y la facilidad de uso, y que se pasan el día intentando aplicar dichos atributos en sus propios proyectos Web, probablemente no piense que un sitio de aspecto amateur repleto de arcaicos documentos técnicos tenga la llave del futuro. ¿En qué cree esta gente? Cree en astutas presentaciones de gigantes corporativos.

Los consorcios sugieren, las empresas venden

En Occidente, cuando nos enfrentamos a un problema empresarial o creativo, lo solucionamos abriendo una aplicación de software y recurrimos a corporaciones líderes en el mercado en busca de los productos que necesitamos. Los propietarios de sitios comprueban la salud de sus empresas abriendo hojas de cálculo con formato de Microsoft Excel. Los diseñadores crean logotipos en Adobe Illustrator y animaciones en Macromedia Flash, y preparan imágenes y diseños Web en Adobe Photoshop. Para cada problema existe una categoría de software; para cada categoría, un líder.

Aunque los diseñadores y programadores Web pioneros aprendieron a crear páginas en el Bloc de notas, los que llegaron después recurrían a productos de edición visual (herramientas WYSIWYG) para realizar sus tareas creativas. Cuando los lenguajes de secuencia de comandos propietarios y los modelos de objetos incompatibles dificultaron aun más el desarrollo, productos como Macromedia Dreamweaver y Adobe GoLive ayudaron a muchos profesionales a crear sitios funcionales al tiempo que ocultaban las complejidades subyacentes. ¿Cómo se podían admitir distintos navegadores? Pulse el botón derecho y el software se encargará de todo.

Estos editores visuales líderes de su categoría eran sofisticados y potentes. Pero hasta hace poco, el código y el marcado que generaban no tenía nada que ver con los estándares Web. Al igual que los programadores, Dreamweaver y GoLive escribían secuencias de comandos específicas para un navegador y estructuras de marcado no semánticas.

Dreamweaver MX y GoLive 6 son mucho más compatibles que sus versiones anterio-

res (como veremos en un capítulo posterior), aunque requieren más esfuerzos de programación. ¿Y dónde buscan los usuarios estos esfuerzos? Acuden a sitios de productos atractivos y bien escritos que actúan como manuales de usuario en línea y albergan el desarrollo de las comunidades de Dreamweaver y GoLive. Hablaremos sobre estos sitios en breve.

Concienciados por los productos o concienciados por los estándares

Mientras las empresas se esforzaban por concebir sencillas soluciones de tipo "pulse un botón" para los problemas del diseño de aplicaciones, lo mismo ocurría con el software intermedio y de servidor. Las herramientas de edición propietarias y los productos de base de datos de grandes marcas como IBM, Sun, Lotus y Microsoft, y de pequeñas y valientes empresas como Allaire (ahora parte de Macromedia) ofrecían la funcionalidad necesaria en un momento en el que los estándares como XML y el DOM seguían debatiéndose en un comité.

No construimos nuestro coche partiendo desde cero así que tampoco deberíamos construir nuestro sitio Web de esta forma. Firme aquí, compre ahora y si algo no funciona lo solucionaremos en la siguiente actualización. Para los programadores dependientes de fechas de entrega tenía sentido al igual que lo tuvo considerar a HTML como una herramienta de diseño: básicamente, consistía en cumplir las necesidades de los clientes al instante.

Por ello, la preocupación por los productos en lugar de la preocupación por los estándares dominaba la forma de pensar de muchos programadores Web y, en concreto, de muchos diseñadores Web. Por cada diseñador o programador que recurría a las especificaciones W3C, había 10 que obtenían su información de los sitios Web de Netscape, Microsoft, Macromedia (véase figura 3.12), Adobe (véase figura 3.13) y otros grandes (e inteligentes) empresas.

Al contrario que `w3.org`, estos sitios se crean para servir a los consumidores (diseñadores y programadores profesionales), para estrechar el vínculo entre empresa y cliente, y para mejorar la imagen corporativa. Por ello, estos sitios tienden a estar bien diseñados (o al menos de forma correcta) en función de las especificaciones de sus marcas corporativas, y sus cursos prácticos se escriben para que resulten fácilmente comprensibles para un público profesional.

No hace falta decir que estos cursos prácticos enfatizan la eficacia de los productos de la empresa. Cuando estas empresas mencionan los estándares Web, probablemente es para proclamar que sus productos son más compatibles que los de sus competidores. Después de todo, el objetivo de estos sitios es que valoremos el producto que acabamos de adquirir y que estemos dispuestos a comprar la próxima actualización.

En definitiva, muchos profesionales de la Web, diseñadores y programadores así como sus clientes y empleados, saben bastante acerca de soluciones propietarias y bastante menos sobre estándares Web.

No hay muchos que se den cuenta, quizás de forma tangencial, que aliarse en exclusiva con una determinada empresa o línea de productos puede destinarlos para siempre a un ciclo de gastos interminable: desde actualizaciones necesarias a costosas personalizaciones, formación, consultoría y mucho más. Después de todo, es cómo funcionan los negocios.

Existe un producto concreto que merece especial atención, como veremos a continuación.

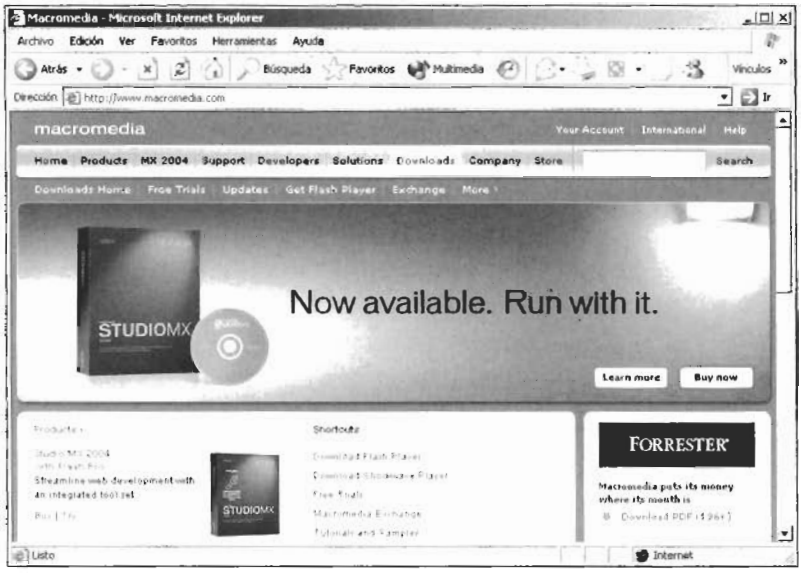


Figura 3.12.

Al igual que el W3C, Macromedia crea valiosas herramientas para diseñadores y programadores (www.macromedia.com). Al contrario que el W3C, Macromedia vende estas herramientas y trabaja duro para ganarse a la comunidad de diseñadores y para hablar el idioma de éstos en su sitio Web.

Figura 3.13.

Al igual que su rival Macromedia, Adobe (www.adobe.com) interactúa continuamente con sus usuarios, les pregunta cuáles son sus necesidades y cambia sus productos en función de esto. Como su rival, también se esfuerza duramente por hablar el idioma de los diseñadores en su sitio. Compare esta imagen con las figuras 3.3 y 3.11.

La palabra que empieza por F

De todas las soluciones propietarias del mercado que las corporaciones han intentando vender, ninguna ha tenido tanto éxito como Macromedia Flash (véase figura 3.14). El producto apareció como un humilde componente bajo el nombre FutureSplash que permitía a los diseñadores añadir gráficos basados en vectores y animaciones a sus páginas.

Los diseñadores apenas prestaron atención a FutureSplash pero Macromedia, como empresa inteligente que es, reconoció inmediatamente su potencial. Adquirió el complemento y la herramienta de creación asociada, le cambió el nombre por Flash y lo transformó en un completo y flexible entorno de creación dirigido por un lenguaje de programación similar a JavaScript denominado ActionScript.

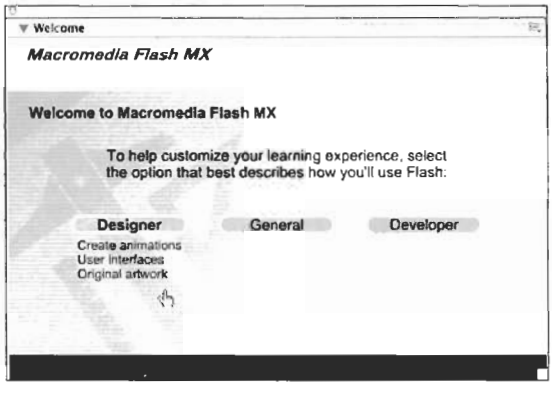


Figura 3.14.

Bienvenido a Macromedia Flash. El entorno de creación Flash puede ser complejo, profundo y difícil, pero Macromedia hace todo lo posible por guiar a diseñadores y programadores de la mano cuando éstos empiezan a utilizar el producto.

Macromedia también consiguió crear un culto sobre la programación con Flash.

El valor de Flash

Mientras que los lenguajes de secuencia de comandos y modelos de objeto incompatibles de los navegadores 4.0 causaron estragos y aumentaron los costes, Flash 4 y su potente lenguaje de programación funcionaba igual de bien en Navigator, IE y Opera, y prácticamente con la misma facilidad en Mac OS, Linux y Unix que como lo hacía en Windows. Para muchos diseñadores, era el adiós al HTML y CSS para chapuceros navegadores 4.0 y una calurosa bienvenida a Flash.

El uso de logotipos en movimiento, tediosas pantallas de carga e introducciones interminables y no deseadas, hizo que Flash se creara una mala fama inicial entre los usuarios. Pero el abuso de la potencia de las nuevas herramientas se convirtió con el tiempo en sofisticados ejercicios creados por gente como One9nine (véase figura 3.15) o Juxt Interactive (véase figura 3.16), entre otros. Agencias con menos talento y menos innovadoras se subieron precipitadamente al carro de Flash y crearon sitios mucho menos atrayentes, pero no se puede culpar al mal carpintero del martillo y los clavos. Flash devoraba el suculento espacio de las aplicaciones del mismo modo que el navegador de Microsoft se merendaba a Netscape.

Aunque Flash resultaba apropiado para muchos proyectos, en otros no lo era, y Flash 4 presentaba lamentables problemas de utilización y accesibilidad que preocupaban terriblemente a los usuarios sin que los programadores y los clientes se dieran cuenta.

Entre las voces más críticas se encontraba Jakob Nielsen (http://www.useit.com) del grupo de consultores Nielsen Norman.

En el 2002, Macromedia solucionó sus problemas y mejoró considerablemente las funciones de accesibilidad y utilización en su actualizado producto Flash MX, y contrataron a Nielsen como consultor. (Si Microsoft y Netscape hubieran sido igual de inteligentes y hubieran contratado a sus críticos más fieros, este autor estaría tumbado en alguna playa en lugar de penar con este libro, pero vamos al grano.) En las manos adecuadas, Flash facilita completas experiencias interactivas que resultarían difíciles de emular con el uso de marcado estándar, CSS, SVG y el DOM.

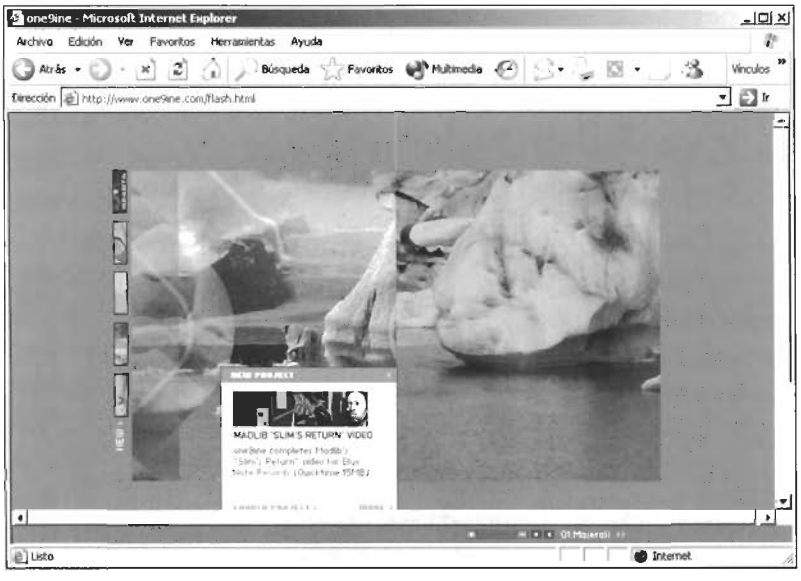


Figura 3.15.

Ponga una potente herramienta en manos de diseñadores visuales de gran talento y, ¿qué es lo que obtiene? Experiencias difíciles de igualar. One9ine (www.one9ine.com) es una tienda con base en Nueva York cuyo diseño se ejecuta casi en su totalidad en Flash.

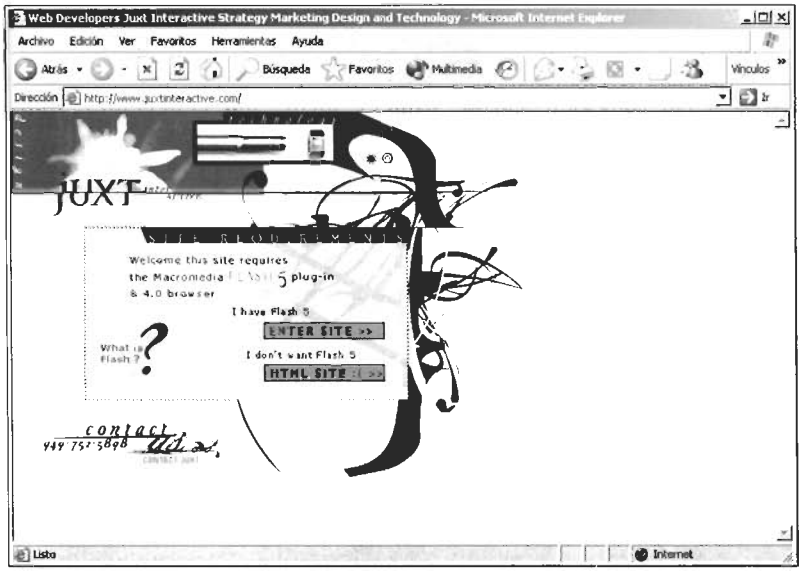


Figura 3.16.

Juxt Interactive (www.juxtinteractive.com) con base en California es otro de los pioneros en el uso de Flash cuyos experimentados diseños contribuyeron a desterrar la idea de que Flash era un truco solamente capaz de crear molestas introducciones.

Nota: SVG es una aplicación XML y un lenguaje vectorial estándar que permite crear animaciones y secuencias de comandos, y que es compatible con otros estándares Web. Al momento de escribir este libro, ninguno de los navegadores más conocidos incluía compatibilidad nativa con SVG y para utilizarlo se requiere un componente, como ocurre con Flash. (El navegador Amaya de W3C admite SVG de forma limitada y algunas variedades de Mozilla se pueden compilar con compatibilidad SVG parcial, aunque estos entornos no se encuentran entre los de mayor popularidad.)

En la actualidad, si quiere crear complejas interfaces similares a aplicaciones, es más sencillo hacerlo en Flash, con su base de instalación y su entorno de programación único.

Puede que un día tenga más sentido crear estas aplicaciones con ayuda de una combinación de XML, XHTML, CSS, ECMAScript, SVG y el DOM.

El problema con Flash

El principal problema relacionado con Flash es que resulta poco indicado para muchos sitios comerciales y de contenido. Aun así, los programadores lo utilizan en estas situaciones incorrectas porque Flash permite utilizar elegantes presentaciones que convencen a los clientes de la buena inversión que han realizado y porque, resulten satisfactorios o no, los sitios en Flash quedan muy bien en la cartera de una empresa.

En sitios de noticias, portales, sitios de compras, institucionales o comunitarios, revistas, directorios y demás que ponen el énfasis en el texto o implican una interactividad práctica, suele ser más adecuado utilizar XHTML, CSS y otros estándares.

Sin embargo, muchos programadores venden Flash en lugar de estos estándares, no porque resulte más indicado para los objetivos del proyecto sino porque lo despachan enseguida y los resultados atraen a los clientes. Es similar, a grandes rasgos, a las agencias de publicidad que se centran en trabajos que no venden esperando acumular premios de diseño.

El otro problema con Flash

El otro problema con Flash es que algunos diseñadores están tan enamorados de él que se han olvidado de cómo utilizar los estándares Web, en caso de que lo supieran. Como resultado, uno se encuentra con presentaciones en Flash que sólo funcionan en uno o dos navegadores. Los mismos archivos de Flash funcionarían en cualquier navegador que contara con el componente, pero los sitios se han creado de forma tan lamentable que muchos usuarios son incapaces de acceder al contenido de Flash. Incluso hay sitios de Flash que requieren IE simplemente para cargar una presentación en Flash. Es como pedirle que use una televisión Sanyo (y no una Sony) para poder ver la programación vía satélite.

Cuando a estas agencias se les pide que diseñen un sitio tradicional basado en estándares, vuelven a los métodos antiguos que hemos descrito anteriormente y a menudo ceden estos proyectos a los diseñadores más jóvenes para que los de mayor experiencia se puedan centrar en proyectos de Flash.

XML, XHTML, CSS y el DOM no son insípidas tecnologías para principiantes, sino potentes y maduros estándares capaces de ofrecer al usuario enriquecedoras experiencias. No tengo nada en contra de las agencias que se especializan en trabajos de Flash de gran belleza y funcionalidad. Simplemente me gustaría que se prestara el mismo cuidado y atención en el 90 por ciento restante del diseño y la programación. Pero no tengo que venderle nada. Ya ha comprado este libro.

La compatibilidad es un insulto

El otro obstáculo a la aceptación de los estándares Web es la creencia equivocada de que los estándares disminuyen de algún modo la creatividad, limitan al diseñador o generan experiencias menos útiles para el usuario en comparación con los métodos propietarios de la vieja escuela. ¿De dónde proviene esta creencia?

Puede que provenga de programadores que hayan intentado utilizar estándares Web en navegadores 4.0 o anteriores y que, con toda razón, no están contentos con los resultados. Pero los días de incompatibilidad con estándares se han acabado.

El poder del lenguaje para moldear percepciones

La frase "estándares Web" puede tener la culpa. El Web Standards Project la acuñó como acto de propaganda. Buscábamos un conjunto de palabras que ofrecería a los fabricantes de navegadores exactamente lo que estaba en juego, un conjunto de palabras cuyo imperativo ético subyacente recordara a los fabricantes de navegadores su compromiso con las tecnologías que habían ayudado a crear y que rogaban admitir. Necesitábamos una frase que transmitiera a programadores, clientes y periodistas técnicos la necesidad urgente de tecnologías fiables, implementadas de forma consistente y válidas para toda la industria. "Recomendaciones" no se ajustaba. "Estándares" sí, o eso pensábamos.

Apenas disponíamos de presupuesto y nuestras esperanzas eran escasas, aunque tuvimos éxito. En la actualidad, empresas como Netscape, Microsoft, Adobe y Macromedia luchan por el cumplimiento de los estándares y se jactan de ello por ser una función esperada y deseada, como la tracción a las cuatro ruedas. Pero aunque estas empresas lo hayan adoptado, muchos diseñadores no lo han hecho.

Algunos confunden los estándares Web con un conjunto impuesto y arbitrario de normas de diseño (al igual que muchos piensan lo mismo de la facilidad de uso, desafortunadamente como muchos consultores). Deberían explicarle a esos diseñadores que los estándares Web no tienen nada que ver con directrices estéticas externas.

Si no la frase "estándares Web", puede que la palabra "cumplimiento" sea equivocada. Los diseñadores quieren cumplir con sus visiones creativas, no con un complejo conjunto de normas tecnológicas. Deberían saber que dichas normas no tienen nada que ver con el aspecto operativo y visual de un sitio; simplemente permiten que los navegadores comuniquen lo que el diseñador ha creado. Del mismo modo, los clientes quieren cumplir con los objetivos definidos por sitios corporativos o institucionales basados en requisitos del mercado y análisis de usuarios. De nuevo, diremos que los estándares Web sólo pueden ayudar garantizando que los sitios funcionen para el mayor número de usuarios y de plataformas.

El problema de la inspiración

Los diseñadores y clientes pueden desalentarse por la falta de inspiración visual (en ocasiones rallando la hostilidad hacia el diseño) de algunos sitios en los que se utilizan estándares Web o en los que se jactan de su cumplimiento de alguna especificación W3C. Encontraremos el mismo problema cuando describamos la accesibilidad. (Algunos sitios de accesibilidad no son agradables de ver, pero el problema es de los diseñadores de los mismos, no de la accesibilidad, que no tiene connotaciones visuales. Lo mismo ocurre con los estándares Web, incluso si el aspecto operativo y visual del sitio Web del W3C o de ECMA no motive a los diseñadores para que aprendan más sobre XML y CSS2.)

El concurso Wthremix (véase figura 3.17) que apareció en diciembre del 2002, intentaba generar parte de ese interés estético perdido. Sus creadores explican de esta forma sus objetivos:

> *El W3C crea potentes estándares y directrices para que la Web sea más racional y se mejore lo que recibe el usuario. Tecnologías como XML, CSS, XHTML, el DOM y directrices como la Iniciativa de accesibilidad Web pueden ayudarnos a crear sitios Web más completos que funcionen mejor para todos. Pero el W3C está formado por gente muy extraña, no por diseñadores, programadores, escritores y especialistas de la información que se preocupan de los clientes.*
>
> *Como resultado, las potentes tecnologías y directrices del W3C se ven atrapadas en un sitio poco elegante que resulta menos atractivo y menos útil de lo que debería. Nos preguntábamos si el sitio del W3C se podría transformar en otro más atractivo, mejor organizado y más sencillo de utilizar y comprender. Éste es el motivo de este concurso.*

El concurso

Wthremix es un desafío de diseño para los programadores y un desafío de programación para los diseñadores. La idea es muy sencilla: volver a diseñar la página inicial del W3C. Confeccionar un diseño y una navegación intuitivos, organizar el contenido pensando en el usuario y crear un aspecto estético que refleje la importancia e influencia de la institución. Muéstrenos cómo cambiaría el diseño de la página, cómo debería comunicarse con los usuarios y, para demostrarlo, utilice XHTML sin tablas y CSS, y cumpla el nivel de accesibilidad 1 de WAI.

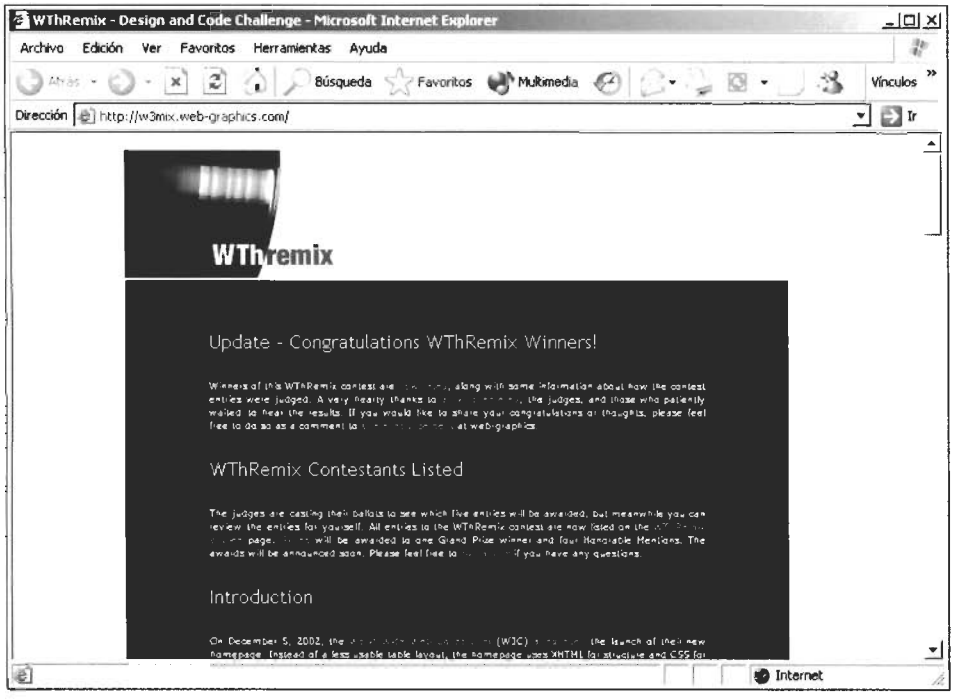


Figura 3.17.

El concurso Wthremix que apareció en diciembre del 2002 retaba a diseñadores y programadores a volver a diseñar el sitio del W3C (http://w3mix.web-graphics.com/).

Otros problemas

Puede que algunos desconfíen de los estándares Web debido a experiencias negativas con la primera versión defectuosa de Netscape 5 o de los errores de IE6 que siguen sin corregirse un año después de que apareciera el navegador.

Puede que otros, intrigados por la promesa de los estándares Web, hayan convertido un sitio de XML a XHTML para descubrir que sus diseños se mostraban de forma diferente en Netscape 6, Mozilla e IE6. En un capítulo posterior explicaremos a qué se debe e incluiremos soluciones rápidas y sencillas para que su sitio vuelva a funcionar. Pero si no conoce estas soluciones, puede que desconfíe de los estándares Web y que prefiera agachar la cabeza y continuar utilizando métodos no estándar obsoletos que tan bien funcionaban en los navegadores del pasado.

No se deje atrapar por el lado oscuro. Aunque la ignorancia y los perjuicios están tan presentes en el diseño Web como en cualquier otra disciplina humana, los estándares Web han venido para quedarse y este libro para ayudarle.

Capítulo 4
XML conquista el mundo (y otras historias del éxito de los estándares Web)

Antes de continuar, conviene contrarrestar las malas vibraciones del capítulo anterior y explicar cómo los estándares Web se han establecido firmemente dentro y fuera de Internet. A pesar de los malentendidos que impiden su adopción en determinados sectores, los estándares Web han ganado la batalla en muchos frentes y están cambiando la tecnología, las empresas y la edición dentro y fuera de la Web.

En este capítulo, nos detendremos en primer lugar en el estándar Web de mayor éxito desde HTML: el Lenguaje de marcado extensible (XML). Se trata de un formato de datos que se ha adoptado prácticamente de forma universal y que se ha adaptado para satisfacer complejas necesidades. Veremos cómo XML ayuda a conservar la viabilidad de los productos de software en un mercado de continuos cambios, soluciona el problema actual de las empresas dirigidas por datos y ha permitido la aparición de una nueva generación de aplicaciones y servicios Web interoperativos.

También veremos cómo los estándares Web han facilitado la distensión y han promovido la cooperación entre competidores dentro del sector de los navegadores. Describiremos cómo las herramientas profesionales de creación Web que antes ignoraban los estándares han acabado por adoptarlos. Veremos cómo los sitios personales de obcecados diseñadores y programadores muestran una mayor aceptación del diseño CSS, marcado XHTML y un mayor cumplimiento de la Iniciativa de accesibilidad en la Web (WAI) y las directivas de accesibilidad de la sección 508.

Todas estas historias de éxito tienen algo en común: los estándares están ganando

aceptación porque funcionan. Cuantos más estándares se acepten, mejor será su funcionamiento y más fácil será el camino que se nos presenta.

El lenguaje universal (XML)

En un capítulo anterior mencionamos cómo la pobre compatibilidad de los primeros navegadores convenció a muchos diseñadores y programadores de que los estándares eran castillos en el aire, lo que provocó que algunos se decantaran por métodos obsoletos y contraproducentes, y que otros se aferraran a Flash ante la exclusión de HTML, CSS y JavaScript. Puede que los capítulos anteriores le hayan convencido de que los estándares Web se enfrentan a una encarnizada batalla en lo que se refiere a aceptación y uso correcto. ¿Qué haremos entonces con XML?

El estándar del Lenguaje de marcado extensible (www.w3.org/TR/2000/REC-xml-20001006) se presentó en febrero de 1998 y sacudió los cimientos de la industria del software (véase figura 4.1). Por primer vez, se ofrecía un formato universal y adaptable para estructurar documentos y datos, no sólo en la Web, sino en todas partes. Y fue recibido con gran aceptación.

Comparación entre XML y HTML

Aunque XML se basa en la misma tecnología principal que creó HTML (al igual que HTML, utiliza etiquetas, atributos y valores para aplicar formato a documento estructurados), difiere bastante del venerable lenguaje de marcado al que intenta reemplazar.

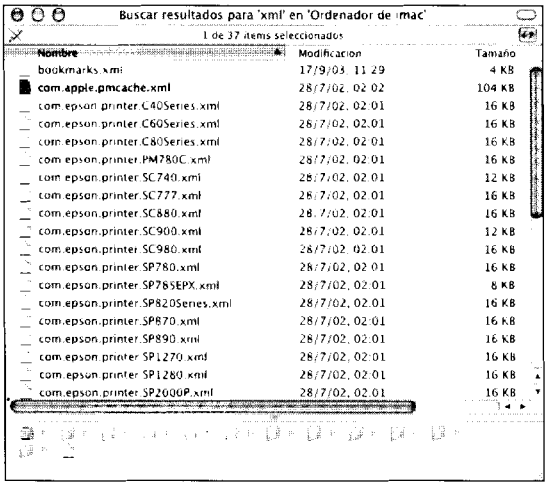


Figura 4.1.

¡Mamá, hay XML en mi ordenador! Realice una rápida búsqueda en un Macintosh y encontrará miles de archivos XML. Algunos almacenan preferencias del sistema operativo, mientras que otros controlan sus impresoras. Algunos son componentes básicos de una aplicación, como Acrobat. iPhoto, iTunes, Eudora, Internet Explorer, Mozilla, Camino, Flash MX, Dreamweaver MX y otros. XML es un estándar Web que supera los límites de la Web.

HTML es un lenguaje básico para marcar páginas Web. Dispone de un número fijo de etiquetas y de un pequeño conjunto de normas relativamente inconsistentes. En HTML, es necesario cerrar algunas etiquetas, otras no y puede que le interese cerrar algunas sí y otras no, en función de cómo se encuentre. Esta inconsistencia permite que cualquiera pueda crear una página Web, aunque no sepa lo que está haciendo y, evidentemente, ésa era la idea.

En aquellos días parecía una buena idea, cuando la Web necesitaba contenidos básicos

y poco más. Pero en la sofisticada Web actual, en la que las páginas se suelen crear por medio de herramientas de publicación y el contenido fluye de bases de datos a páginas Web y de dispositivos inalámbricos a impresoras, la falta de normas uniformes en HTML impide cambiar la función de los datos. Es muy sencillo convertir texto en HTML, pero es muy complicado convertir datos marcados en HTML a otro formato.

Del mismo modo, HTML es simplemente un lenguaje de formato pero sin mucha conciencia de sí mismo. No contiene información sobre el contenido al que aplica formato, lo que limita las posibilidades de volver a utilizar dicho contenido en otras partes. Y, evidentemente, es un lenguaje estrictamente para la Web.

Por el contrario, el marcado basado en XML, está controlado por normas coherentes y puede traspasar los límites de la Web. Al marcar un documento en XML, no sólo lo preparamos para mostrarlo en una página Web. Lo codificamos en etiquetas comprensibles en cualquier entorno compatible con XML.

Un padre, varios hijos

Específicamente, XML es un lenguaje para crear otros lenguajes. Siempre que se cumplan las normas, los bibliotecarios tienen total libertad para crear marcado XML cuyas etiquetas personalizadas faciliten las operaciones de catalogación. Las compañías discográficas pueden crear marcado XML cuyas etiquetas incluyan datos sobre el artista, la grabación, el compositor, el productor, derechos de autor, etc. Los compositores pueden organizar sus partituras en un lenguaje de marcado XML personalizado denominado MusicML. (A partir de ahora, cuando nos refiramos a creación de marcado XML, diremos "escribir XML".)

Estos lenguajes XML personalizados se denominan aplicaciones y, como todos son XML, son compatibles entre sí. Es decir, un analizador XML puede comprender todas estas aplicaciones y éstas pueden intercambiar datos entre sí con mucha facilidad. Por ello, los datos de una base de datos XML de una compañía discográfica pueden acabar en el catálogo de grabaciones de una biblioteca sin esfuerzo y sin errores, y sin preocuparse de las incompatibilidades del software.

Un ingrediente básico del software profesional y de consumo

La posibilidad de aplicar formato, comprender e intercambiar datos, ha convertido a XML en algo tan omnipresente como la Coca-Cola. XML no sólo almacena contenido de bases de datos en línea y corporativas sino que también se ha convertido en la lengua franca de programas de base de datos como FileMaker Pro y otro software no orientado a bases de datos, desde aplicaciones de diseño hasta productos empresariales como Microsoft Office y OpenOffice, cuyos formatos de archivos nativos se basan en XML.

El sistema operativo Macintosh OS X basado en Unix de Apple almacena sus preferencias en XML. Gigantes del diseño impreso como Quark XPress 5.0 y Adobe InDesign 2.0 importan y exportan XML y admiten la creación de plantillas basadas en XML.

Editores Web visuales como Macromedia Dreamweaver MX y Adobe GoLive 6 también recurren a XML, lo que facilita (o al menos hace posible) intercambiar datos entre la página impresa, el diseño Web y la base de datos que controla su almacenamiento en línea o directorio global.

No contentos con simplemente analizarlo, algunos productos están formados por XML. Macromedia Dreamweaver MX está construido con archivos XML disponibles para el usuario final (véanse figuras 4.2, 4.3 y 4.4), lo que permite modificar el programa si se modifican esos archivos (www.alistapart.com/stories/dreamweaver/). La personalización de Dreamweaver de esta forma y la venta de dichas versiones personalizadas a colegas que utilicen Dreamweaver se ha convertido en una especie de industria artesanal.

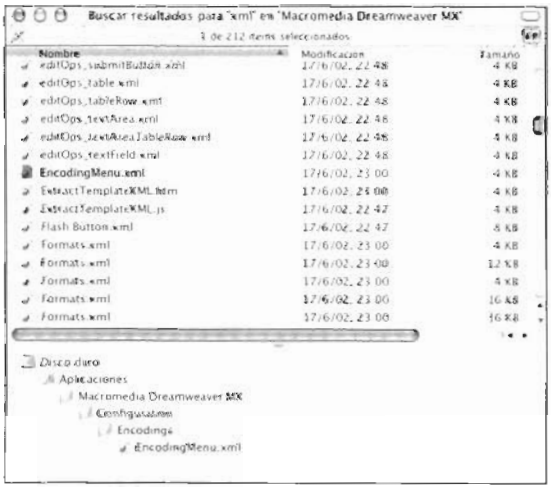


Figura 4.2.

Dreamweaver MX, una conocida herramienta de programación Web, está formada por archivos cuyos formatos resultarán familiares a los programadores Web. Bajo la caperuza, Dreamweaver está formado por archivos XML...

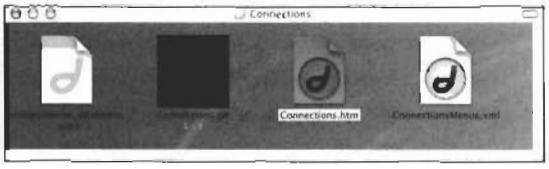


Figura 4.3.

...junto con archivos GIF, HTML y JavaScript. Al igual que los archivos de un sitio Web, los componentes de Dreamweaver se organizan en subdirectorios.

Figura 4.4.

Los inteligentes usuarios de Dreamweaver pueden editar estos archivos para personalizar el productor. En el ejemplo, un usuario modifica los métodos abreviados de teclado predeterminados de Dreamweaver. Para ello cambia un archivo XML denominado menus.xml.

El software de consumo también está encantado con XML. El administrador de información personal de su PC, Mac o PDA lee y escribe XML o puede modificarse para que lo haga a través de productos de terceros como el analizador XML Aefred para Palm (www.xml.com/pub/ r/216). Cuando su cámara digital añade la fecha a una instantá-

nea y registra sus dimensiones, el tamaño de archivo y este tipo de datos, lo hace en XML. Cada vez que su padre le envía esas fotos de 7 MB de sus vacaciones, probablemente esté enviando datos con formato XML junto a las bellas imágenes de la puesta de sol.

Incluso programas de administración de imágenes como iPhoto (véase figura 4.5) de Apple es capaz de entender XML. Cuando imprime una instantánea del primer encuentro del pequeño Oscar con un cachorro, las mejillas sonrojadas de Oscar y los brillantes ojos del perrito se representan correctamente gracias a los valores de impresión almacenados como datos XML por el sistema operativo Macintosh OS X.

Más famoso que la MTV

¿Por qué XML ha capturado la imaginación de fabricantes tan dispares y ha conseguido implantarse en sus productos? XML combina estandarización con extensibilidad (la posibilidad de personalizar), convertibilidad (la posibilidad de transformar datos de un formato a otro) y un intercambio de datos relativamente perfecto entre una aplicación XML o producto compatible con XML y otro.

Como estándar abierto que no depende de patentes ni royalties, XML aniquila los formatos propietarios desfasados de aceptación limitada y gastos incorporados. Al incluir XML en un producto de software o al desarrollar un lenguaje personalizado basado en XML, el W3C no le cobra cuota alguna. Es más, la aceptación de XML es vírica. Cuantos más vendedores se enganchan a XML, con mayor rapidez se extiende a otros vendedores y resulta más sencillo pasar datos del producto de un fabricante a otro.

Además, XML funciona. Se acabaron los días en los que los compañeros de trabajo le consideraban un gurú por conseguir datos delimitados por tabulaciones de un producto e importarlos en otro (a menudo con cierta pérdida de datos y gran cantidad de cambios manuales de formato).

Figura 4.5.

El programa iPhoto de Apple (www.apple.com/iphoto/), un componente gratuito de su sistema operativo OS X, utiliza XML para organizar datos de bibliotecas de fotografías, recordar parámetros de impresión y mucho más.

XML ayuda a los vendedores a crear productos cuya compatibilidad alienta a los consumidores a que trabajen de forma más inteligente, no a que trabajen más. Los consumidores responden con sus libros de bolsillo.

No es una panacea, pero sale en la tele

No queremos que piense que XML es la panacea para todos los problemas de software. Los datos de un JPEG están mejor expresados en formato binario que en texto. Tampoco queremos decir que todos los paquetes de software del mercado utilicen XML, aunque la mayoría de las aplicaciones profesionales y productos de consumo lo hagan, y sus cifras aumentan cada día más. Ni siquiera queremos decir que todos los programas que presumen de admitir XML lo hagan a la perfección.

Pero correctamente implementado o no, XML ha transformado la industria del software así como el lugar de trabajo.

Incluso los fabricantes de productos que no admiten XML empiezan a pensar que deberían hacerlo. En abril de 2002, desanimados por las escasas ventas y un mercado fragmentado, un grupo de proveedores de televisión y tecnologías interactivas se unieron bajo la bandera de iTV Production Standards Initiative. Su misión, descubrir y apoyar un estándar basado en XML para que los productores pudieran escribir contenidos interactivos y distribuirlos en las principales plataformas (www.allnetdevices.com/developer/news/2002/04/09/itv_firms.html).

¿Le suena familiar? Es exactamente lo que manifestó el The Web Standards Project sobre los estándares Web durante las guerras de los navegadores a finales de los 90.

Cinco formas de generar datos

En la Web, XML es el formato cada vez más utilizado por profesionales, programadores y especialistas de contenidos que deben trabajar con datos almacenados en grandes sistemas corporativos o institucionales. A continuación, se enumeran cinco razones para seleccionar XML, muchas de las cuales le resultarán familiares tras el análisis anterior:

- Al igual que ASCII, XML es un formato de archivo único y universal que interactúa a la perfección con otros.

- Al contrario que ASCII (o HTML), XML es un formato inteligente consciente de sí mismo. No sólo almacena datos, también puede almacenar datos sobre datos (metadatos), lo que facilita la realización de búsquedas y otras funciones.

- XML es un lenguaje extensible, que permite ser personalizado para adaptarse a las necesidades de cualquier institución, empresa o categoría empresarial o académica, y que puede generar lenguajes adicionales basados en XML que realicen tareas específicas, como sindicación de datos o entrega de servicios Web.

- XML se basa en normas que garantizan la coherencia de la transferencia de datos a otras bases de datos, la conversión en otros formatos o la manipulación por parte de otras aplicaciones XML.

- A través de protocolos XML adicionales y lenguajes de ayuda basados en XML, los datos XML se pueden transferir automáticamente a una amplia variedad de formatos, desde páginas Web a catálogos impresos e informes anuales. Esta posibilidad de transformación es con lo que soñaban los programadores antes de que apareciera XML. Tampoco las grandes corporaciones ignoran las características de ahorro que ofrece XML.

Un filón de inventos

Los siguientes cuatro ejemplos y las enseñanzas que se desprenden de éstos, sugerirán la profundidad de la aceptación de XML en la Web y pondrán de manifiesto cómo la continua aparición de nuevos lenguajes y protocolos derivados de XML solucionan los problemas que acuciaron incluso al más brillante de los programadores.

Transformaciones de hoja de estilos extensible (www.w3.org/TR/xslt)

Este lenguaje de marcado basado en XML puede extraer y ordenador datos XML y aplicarles un formato HTML o XHTML, listos para poder verlos en línea. Si lo prefiere, XSLT puede transformar sus datos a PDF o texto, o utilizarlos para mantener actualizado un gráfico o una imagen empresarial similar en formato SVG. Incluso puede hacer todo esto a la vez.

Puede consultar el tutorial de J. David Eisenberg "Using XML" (www.alistapart.com/stories/usingxml/).

Infraestructura para la descripción de recursos (www.w3.org/RDF/)

Este lenguaje basado en XML constituye una estructura coherente para aplicaciones que intercambian metadatos en la Web. En términos prácticos, RDF integra catálogos y directorios de biblioteca; recopila y sindica noticias, software y todo tipo de contenidos, y facilita la comunicación entre distintos tipos de colecciones (como por ejemplo colecciones personales de fotografías y música, un ejemplo que ofrece el sitio del W3C). RDF también puede controlar software. Si dispone del navegador Mozilla en su escritorio, abra sus carpetas y busque en ellas. Encontrará archivos RDF (y CSS) que ayudan a funcionar al navegador. Fíjese en concreto en las carpetas de perfiles. Cada perfil tiene su propio conjunto de archivos basados en XML.

Resumen enriquecido de datos (http://backend.userland.com/rss092)

El Resumen enriquecido de datos (RSS) es un vocabulario XML para describir sitios Web, creado originalmente por Dan Libby para completar el portal My Netscape de AOL/Netscape. Cuando AOL perdió interés en abril del 2001, la empresa UserLand Software de Dave Winer continuó con la especificación. En la actualidad, RSS se utiliza en miles de sitios, lo que lo convierte en uno de los formatos XML de mayor aceptación en la Web (véase figura 4.6).

XML-RPC (www.xmlrpc.com)

Otra de las innovaciones de UserLand Software, XML-RPC, es una especificación y un conjunto de implementaciones que permi-

ten que el software ejecutado en diferentes sistemas operativos y diferentes entornos pueda realizar llamadas de procedimiento en Internet.

Entre otras cosas, XML-RPC se puede usar para automatizar tareas de administración de sitios en herramientas de publicación Web.

Herramientas de publicación Web para el resto de los mortales

Como muestra este breve resumen, lo mismo que los productos de software compatibles con XML descritos anteriormente hacen por un precio, los lenguajes basados en XML lo hacen gratis si caen en manos de programadores inteligentes. A su vez, estos programadores suelen crear nuevos productos para facilitar las necesidades de otros diseñadores, programadores y autores.

Programas de publicación personal como Movable Type (www.movabletype.org), utilizado tanto por editores conocedores de XML como por otros menos técnicos para mantener periódicos Web, nuevos sitios y registros Web, emplean XML-RPC para facilitar la administración del sitio y RSS XML para sindicar y distribuir automáticamente contenidos a otros sitios compatibles con XML (véase figura 4.7). Mientras que Movable Type permite a sus usuarios publicar, XML permite a Movable Type existir.

Movable Type es sólo uno de los diferentes productos de publicación que utilizan XML para gestionar y sindicar contenidos. Entre otros podemos destacar a Radio UserLand (véase figura 4.8), UserLand Frontier (http://frontier.userland.com/) y Blogger de Pyra Software (www.blogger.com). Estos productos muestran un aumento continuo de su popularidad, mientras que una segunda oleada de editores personales, incluyendo aquéllos que se perdieron la revolución Web inicial, descubren las ventajas de compartir sus pensamientos en línea.

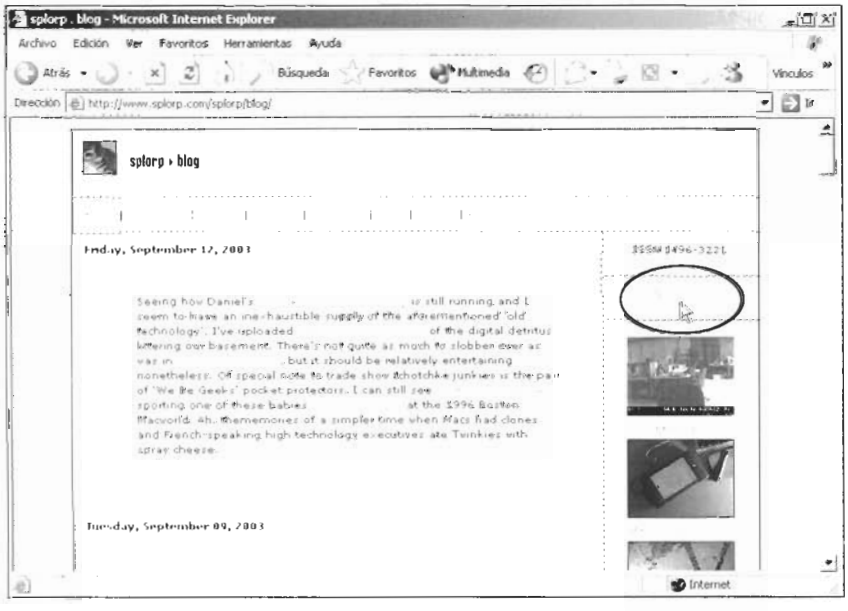


Figura 4.6.

El registro Web, o publicación periódica personal, de Splorp.com, ofrece RSS XML, lo que permite sindicar fácilmente el contenido del sitio (www.splorp.com). Por cierto, el sitio asegura que "splorp" es el ruido que se produce al coger lasaña con una cuchara. Ahora ya lo sabe.

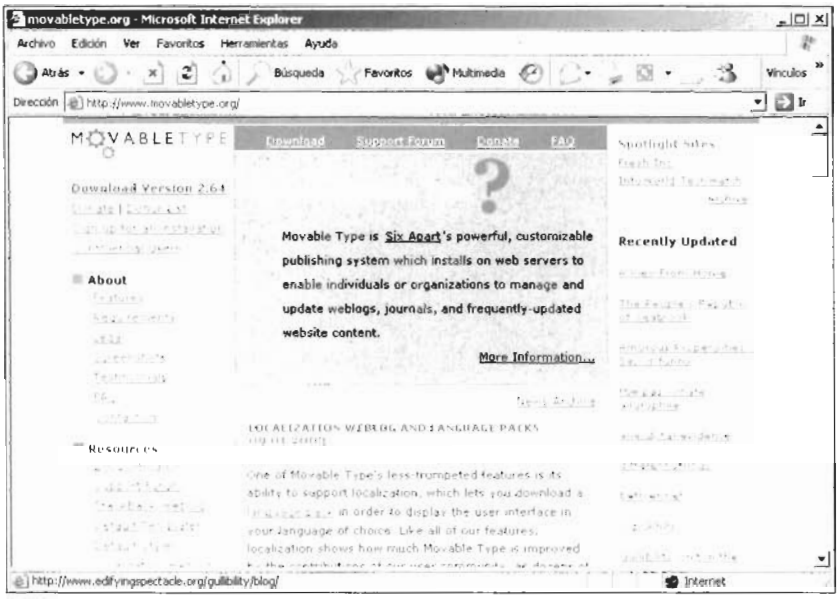


Figura 4.7.

Movable Type, una herramienta de publicación Web, utiliza derivados de XML para facilitar la administración de sitios y ofrecer contenidos a otros sitios compatibles con XML (www.movabletype.org).

Figura 4.8.

Radio UserLand (http://radio.userland.com/) utiliza derivados de XML como SOAP para eliminar el fastidio de la publicación Web y poder centrarse en escribir, incluir imágenes y establecer vínculos a sitios de su agrado.

Por esta razón, con el aumento de la publicación personal, también lo hace XML, no sólo entre los programadores más sofisticados sino también entre los que nunca han oído hablar del estándar XML y a los que les costaría escribir XML (o, en ocasiones, incluso HTML) por su cuenta.

Lo que vale para los independientes vale para los líderes del mercado. Flash MX importa, exporta y trabaja con XML, lo que añade los beneficios empresariales del intercambio de datos basado en estándares a la potencia creativa propietaria de la conocida herramienta de diseño de Macromedia. Los programado-

res pueden utilizar los mismos datos XML para controlar versiones de Flash y no de Flash, lo que les permite ahorrar tiempo y dinero a la vez que se utilizan mejor los recursos.

A su servicio

La lógica de XML también está detrás del emergente mercado de servicios Web. El Protocolo de acceso a objetos simple basado en XML (www.develop.com/soap/) facilita el intercambio de información en un entorno de red descentralizado e independiente de plataformas, de acceso a servicios, objetos y servidores, y para codificar, descodificar y procesar mensajes. El poder subyacente de XML permite que SOAP pueda superar la complejidad de múltiples plataformas y productos.

SOAP es el único protocolo en el prometedor mundo de los servicios Web (www.w3.org/2002/ws/), una categoría que grandes empresas como IBM (www-106.ibm.com/developerworks/webservices/) y Microsoft (www.microsoft.com/net/) ansían poseer. Los pequeños programadores independientes de código abierto ofrecen una visión competitiva de los servicios Web centralizados, en la que no domina ninguna empresa. David Rosman define los servicios Web de esta forma:

> *Los servicios Web son componentes de software reutilizables basados en XML y protocolos relacionados que permiten una interacción prácticamente gratuita en el ecosistema empresarial. Se pueden utilizar de forma interna en procesos de integración de aplicaciones rápidos y de bajo coste o se pueden poner a disposición de los clientes, proveedores y asociados en Internet. (Fuente: www.dangerous-thinking.com/ stories/2002/02/16/webServicesDefined.html.)*

XML controla la mayor parte de los protocolos de servicios Web y la compatibilidad que incorpora es lo que permite la presencia de dichos servicios. Mientras XML sea gratuito para todos, no es necesario que ninguna empresa (independientemente de su importancia y su poder) domine esta categoría.

Aplicaciones XML en nuestro sitio

XML es el lenguaje en el que se basan los Gráficos vectoriales escalables (www.w3.org/TR/SVG/) y el Lenguaje de marcado de hipertexto extensible (www.w3.org/TR/2002/REC-xhtml1-20020801/). Los ilustradores que exportan el logotipo de sus clientes en formato SVG y los autores Web que crean sus páginas en XHTML utilizan XML, lo sepan o no.

Las normas comunes a todas las formas de XML contribuyen a que estos formatos funcionen de forma conjunta y con otros tipos de XML, por ejemplo con XML almacenado en una base de datos. Un gráfico SVG puede alterarse automáticamente en respuesta a una búsqueda generada por un visitante o se puede actualizar continuamente en función de datos entregados por un proveedor de noticias XML.

El sitio de un canal televisivo local puede utilizar esta posibilidad para mostrar noticias en directo sobre el tráfico. Mientras

acaba un atasco y empieza otro, los proveedores de noticias pueden obtener esta información del servidor, donde se le aplicará un formato de contenido de texto legible para el usuario en XHTML y un mapa de tráfico actualizado en SVG. Al mismo tiempo, los datos se pueden sindicar en RDF o RSS para compartirlos con otras organizaciones informativas o se pueden utilizar con SOAP para que las autoridades respondan al respecto.

Aunque están basados en XML, los gráficos SVG se pueden crear muy fácilmente en productos como Adobe Illustrator 10 (www.adobe.com/products/illustrator/main.html). Al igual que los gráficos vectoriales de Flash, las imágenes creadas en SVG pueden llenar incluso los monitores de mayor tamaño sin apenas utilizar ancho de banda. Y estos gráficos, como otros componentes estándar de páginas Web, se pueden manipular con ayuda de ECMAScript y el DOM, sin mencionar que, de forma predeterminada, se puede acceder al contenido textual SVG e incluso se puede seleccionar con el cursor independientemente de cómo se haya estirado o deformado.

Todavía en pañales

En la actualidad, la potencia de SVG está de algún modo limitada por la necesidad de utilizar un componente (www.adobe.com/svg/) como ocurre con Flash. Y este componente no funciona con la misma calidad en todas las plataformas y navegadores. Cuando los navegadores admitan de forma nativa SVG, su capacidad para añadir interactividad visual basada en estándares a todos los sitios Web será mucho más completa.

También en la actualidad, la compatibilidad con XML que ofrecen los navegadores se encuentra todavía en pañales. Aunque XML es el motor de multitud de programas, bases de datos y servicios Web, son escasos los navegadores que pueden mostrar archivos XML puros y la creación de aplicaciones XML supera las posibilidades de muchos diseñadores y propietarios de sitios.

La comunidad de programadores ha resuelto este último problema con la creación de lenguajes, protocolos y productos basados en XML que todos podamos utilizar entre bastidores. El W3C ha solucionado el problema de compatibilidad con XML de los navegadores mediante la combinación de la conocida simplicidad de HTML con las prestaciones extensibles de XML en el estándar de marcado XHTML que analizaremos en un capítulo posterior.

Compatible por naturaleza

Como comparten un padre común y están sujetos a las mismas normas, todas las aplicaciones XML son compatibles entre sí, lo que permite a los programadores manipular un conjunto de datos XML a través de otro para desarrollar nuevas aplicaciones XML en función de las necesidades, sin temer a la incompatibilidad.

Omnipresente en el software profesional y de consumo actual, ampliamente utilizado en la programación Web, esencial para el creciente mercado de los servicios Web y con compatibilidad directa por diseño, XML resuelve el problema de obsolescencia descrito en capítulos anteriores. XML ha tenido

un éxito que nadie había imaginado ya que soluciona las peores pesadillas de incompatibilidad y los problemas tecnológicos.

Los fabricantes de software, poco dispuestos a arriesgarse a perder clientes diferenciándose del resto, reconocen que al admitir XML, sus productos pueden funcionar con otros y ser viables en un mercado sujeto a continuos cambios. Ejecutivos y profesionales de la informática, reacios a renunciar a sistemas legados propietarios para conservar el valioso rehén de datos de sus organizaciones, pueden resolver sus problemas mediante la conversión a XML. Los pequeños programadores independientes pueden competir con las grandes empresas dominando el poder de XML, que recompensa las ideas, no los presupuestos.

En un mundo dirigido por datos, los formatos propietarios ya no sirven, si lo hicieron alguna vez. XML equilibra el terreno de juego e invita a todos a participar. XML es un estándar Web y funciona.

Y ése es el sello de un estándar: que funcione, que permita realizar un trabajo y que se pueda utilizar con otros estándares. Llámelo interoperabilidad (el término que le asigna el W3C) o simple cooperación entre componentes. Sea cual sea el nombre, XML supone una importante mejora con respecto a los días de las tecnologías Web propietarias. Bajo el hechizo de los estándares, los competidores habían aprendido a cooperar.

Una nueva era de cooperación

Como hemos mencionado varios cientos de veces, Microsoft, Netscape y Opera admiten, finalmente, los mismos estándares en sus navegadores. Como inesperada consecuencia de su cooperación tecnológica, estos otrora fieros competidores han aprendido a trabajar conjuntamente de otras formas sorprendentes.

En julio del 2002, Microsoft remitió al grupo de trabajo HTML del W3C una serie de pruebas HTML y aserciones en apoyo al Ten Suite Development HTML 4.01 del W3C (http://lists.w3.org/Archives/Public/www-qa-wg/ 2002Jul/0103.html). La contribución se realizó en nombre de Microsoft, Openwave Systems, Inc. y America Online, Inc., propietarios de Netscape y Mozilla. Opera Software Corporation (fabricantes del navegador Opera) y The Web Standards Project también la revisó.

Suites de prueba y especificaciones

Las suites de prueba W3C permiten a los fabricantes de navegadores determinar si su software es compatible con un estándar o requiere modificaciones adicionales. No existía ninguna suite de prueba para HTML 4.01 (el lenguaje de marcado que también es la base de XHTML 1.0). En ausencia de dicha suite de prueba, los fabricantes de navegadores que querían cumplir con dichos estándares tenían que cruzar los dedos y esperar lo mejor.

Es más, al no existir dicha suite de prueba, los fabricantes de estándares se encontraban en una posición extraña. ¿Cómo podían estar seguros de que una de sus tecnologías podía solucionar correctamente los problemas que supuestamente tenía que solucionar si

carecían de métodos prácticos para probarla? Es como diseñar un coche en papel sin disponer de un taller que fabrique lo que hemos planificado.

Para interés de los creadores de estándares y los fabricantes de navegadores, dicha suite debió aparecer mucho antes.

Importancia de la suite

Cuando Microsoft tomó la iniciativa para corregir el problema generado por la ausencia de una suite de prueba, optó por no actuar en solitario e invitó a sus competidores y a un grupo externo (WaSP) a que participaran en el esfuerzo de los estándares. La misma importancia tuvo la aceptación de dicha oferta por parte de los competidores y del grupo externo. El trabajo se remitió sin gravámenes de patentes o beneficios, indicando que los trabajos resultantes o derivados serían propiedad exclusiva del W3C. Ni Microsoft ni sus competidores pretendían obtener un céntimo de sus problemas.

En el orden normal de las cosas, Microsoft no suele preocuparse por lo que le conviene a AOL/Netscape ni éstos tampoco se preocupan por ayudar a Microsoft, y ninguno de los dos se devana los sesos para descubrir qué es lo que le interesa a Opera. Y estas empresas no se embarcan en proyectos desinteresados que les hagan perder dinero. Sin embargo, ahí las tenemos, actuando de forma conjunta por el bien de la Web y centrándose no sólo en nuevas tecnologías propietarias sino en el humilde HTML 4.

Ignorado por la prensa especializada, este desapercibido evento significa un cambio brutal. Hemos recorrido un largo camino desde los días en los que los fabricantes de navegadores se centraban en tecnologías "de un solo navegador" a expensas de los estándares Web y de los usuarios Web para de esta forma arruinar el mercado a sus competidores. Los fabricantes de navegadores siguen innovando, evidentemente, y esperan que compremos sus productos en detrimento de los de sus rivales. Pero esta nueva predisposición a trabajar de forma conjunta demuestra lo comprometidos que están en lo que respecta a la compatibilidad a través de los estándares Web y cómo ha cambiado el sector desde los días de las guerras de los navegadores (1996-1999).

No se crea el cuento

De vez en cuando, los periódicos y las publicaciones especializadas se aferran a algún cambio en el mercado de los navegadores para afirmar que la guerra se ha reiniciado. Sucedió en junio del 2002, cuando AOL cambió sus usuarios CompuServe de un navegador basado en IE a otro basado en Mozilla/Netscape. "¡Cambio en el mercado Web preocupa a los programadores!" proclamaban las publicaciones especializadas. "¡La Segunda Guerra de los Navegadores!", afirmaban las páginas de economía de los periódicos. Meses más tarde, aparecieron titulares parecidos cuando AOL cambió a sus usuarios de Mac OS X a un navegador basado en Gecko. No se crea el cuento.

En la economía de las noticias actual, los editores deben vender ejemplares si desean conservar sus empleos. Las crisis y los conflictos siempre venden, y los instintos de la prensa amarilla poco tienen que ver con la

cruda y pragmática realidad. Independientemente de los cambios ocasionales de las cuotas de mercado, las guerras de los navegadores se han acabado hace tiempo y ningún editor desea que se recrudezcan.

La Web se construirá sobre el lecho de las tecnologías analizadas en este libro y admitidas por los principales navegadores. AOL/Netscape y Microsoft tendrán sus roces de vez en cuando, pero la competencia entre ellos ha cambiado considerablemente de los reinos de los navegadores a escenarios que no nos conciernen (a excepción de FrontPage, que analizaremos a continuación).

La Web cumple la mayoría de edad

Aunque no tan milagroso (ni tan importante) como un plan de paz de la ONU, el conjunto de pruebas HTML sigilosamente presentado al W3C por Microsoft y sus enemigos más encarnizados del sector refleja un cambio permanente en el futuro cercano de la Web. Cuando los competidores cooperan de esta forma, significa que el medio del que dependen ha cumplido la mayoría de edad.

Lo mismo sucede en cualquier sector maduro. Las compañías discográficas que se odian entre sí aunarán esfuerzos pacíficamente para presentar un nuevo estándar de la industria y para protegerse de las amenazas de su sustento (como el intercambio de archivos de música por la red). Que Microsoft, AOL/Netscape y Opera puedan trabajar de forma conjunta nos demuestra que la Web ha madurado. Lo que les ha unido (una suite de prueba del W3C) nos indica el origen de esta madurez. Nuestro medio ha crecido gracias a los estándares que se describen en este libro.

En los próximos años podemos anticipar una mayor cooperación y un aumento de la compatibilidad con los estándares. También nos podemos relajar sabiendo que los sitios construidos con estándares continuarán funcionando en los navegadores de estos fabricantes hoy, mañana o dentro de 10 años. Sus acciones conjuntas demuestran que están tan comprometidos con la compatibilidad directa como cualquier diseñador o propietario de un sitio podría imaginar.

Otro ejemplo del compromiso de un fabricante de navegadores con los estándares. Netscape apoya económicamente a un pequeño equipo de evangelistas de los estándares cuya misión es trabajar para mejorar los estándares en navegadores y en sitios Web, y que publican soluciones para las diferentes plataformas basadas en estándares abiertos, no para ser vistas en Netscape ni otros parches propietarios.

Estándares Web y herramientas de creación

Diseñados en el momento álgido de las guerras de los navegadores, editores visuales profesionales líderes del mercado como Macromedia Dreamweaver y Adobe GoLive solucionaron inicialmente el problema de la incompatibilidad entre navegadores generando marcado y código optimizado para navegadores 3.0 y 4.0.

Cuando los navegadores se basaban en etiquetas HTML inválidas no estándar, Dreamweaver y GoLive les proporcionaban lo que necesitaban. Si cada navegador tenía su propio Modelo de objetos de documento

incompatible, controlado por su propio lenguaje de secuencia de comandos propietario, Dreamweaver y GoLive diseñaban el código en función de ello.

De esta forma, Dreamweaver y GoLive eran igual de inocentes que los programadores que creaban a mano el mismo tipo de marcado y de código. Como explicamos en un capítulo anterior, los diseñadores de sitios hacían lo que tenían que hacer cuando los estándares estaban por desarrollarse y los navegadores por admitirlos. La máxima "Dales lo que piden" tenía sentido en aquella época, pero ya no es una estrategia válida. Cuando los navegadores apoyaron a los estándares, herramientas como Dreamweaver y GoLive tendrían que haber hecho lo mismo. Hoy lo han conseguido.

La Task Force de Dreamweaver

La Task Force Dreamweaver del Web Standards Project se creó en el 2001 para trabajar con los ingenieros de Microsoft en un intento de mejorar el cumplimiento de los estándares y la accesibilidad de los sitios producidos con Dreamweaver, el líder del mercado entre los editores Web visuales profesionales. La historia de la Task Force se recoge en www.webstandards.org/act/campaign/dwtf/.

Entre los principales objetivos del grupo, definidos por Rachel Andrew y Drew McLellan, destacamos los siguientes:

- Dreamweaver debería producir marcado válido (el marcado válido utiliza solamente etiquetas y atributos estándar y no contiene errores, como explicaremos en un capítulo posterior).
- Dreamweaver debería permitir elegir entre versiones de XHTML y HTML, e incluir una DTD válida para cada opción. (Una DTD, o Definición de tipo de documento, indica al navegador el tipo de marcado que se ha utilizado para crear una página Web.)
- Dreamweaver respetaría la DTD de un documento y produciría marcado y código de acuerdo a la misma.
- Dreamweaver permitiría a los usuarios crear documentos Web accesibles para todos.
- Dreamweaver representaría CSS2 con un nivel de precisión adecuado para que las páginas con formato CSS pudieran funcionar en el entorno visual de Dreamweaver.
- Dreamweaver no alteraría diseños CSS válidos añadiendo elementos estilísticos sin consentimiento del usuario.
- Los usuarios de Dreamweaver deberían confiar en que sus páginas creadas con Dreamweaver serían válidas y tendrían un alto nivel de accesibilidad.

Éstos y otros objetivos se materializaron en mayo del 2002 con el lanzamiento de Dreamweaver 2002. Al evaluar el producto que habían ayudado a perfilar, la Task Force descubrió que Dreamweaver

- Producía marcado válido.
- Ayudaba a los usuarios a crear sitios accesibles.

- Reproducía CSS2 con un nivel de precisión razonable (aunque la ubicación de CSS siempre resulta problemática).
- Evitaba alterar diseños CSS.
- Promovía la validación de XHTML y CSS (pruebas automatizadas del cumplimiento de los estándares).
- Respetaba y promovía los estándares Web.

Nota: Puede leer la evaluación completa del producto de la Dreamweaver Task Force en la dirección www.webstandards.org/act/campaign/dwtf/mxassessed.html.

Las herramientas WYSIWYG cumplen la mayoría de edad (dos de tres no está nada mal)

En lo que respecta a este capítulo, basta con decir que Dreamweaver MX hace algo más que alabar en su compatibilidad con los estándares Web y la accesibilidad. Sus prestaciones y su compatibilidad con los estándares son equivalentes con la de los navegadores actuales.

Aunque el WaSP no pudo trabajar con Adobe, esta empresa mejoró de forma independiente y sustancial la compatibilidad con los estándares en su producto profesional de creación Web visual GoLive (www.adobe.com/products/golive/main.html).

Además de CSS y XHTML, GoLive es compatible con el Lenguaje de integración multimedia sincronizado (www.w3.org/AudioVideo/), el estándar del W3C para crear presentaciones multimedia accesibles. Aquéllos que utilizan uno de estos dos editores Web visuales líderes del mercado pueden crear, de forma sencilla, sitios accesibles compatibles con los estándares.

FrontPage: incompatible por diseño

Un tercer producto de la categoría, Microsoft FrontPage, está poco extendido entre los profesionales pero sí lo utilizan usuarios semiprofesionales, ya que viene incluido con otros productos de Microsoft. Los profesionales del sector público pueden verse limitados a utilizar FrontPage porque el presupuesto no permita un editor Web adicional.

"Ya tenemos un editor Web", le dirán los gerentes. Están equivocados. FrontPage no es un editor Web visual. Es un editor de páginas para Internet Explorer.

Aunque Microsoft fabrica navegadores compatibles con estándares, su herramienta FrontPage vomita marcado y código propietario, y genera sitios que sólo se pueden ver y sólo funcionan en Internet Explorer. El resultado no estándar de FrontPage no se debe a la ineptitud. El problema es una característica, aunque no diseñada para beneficio del usuario.

Bajo declaración jurada, Bill Gates admitió que había enviado un memorando en el que indicaba al grupo de software de Office, en

el que se incluye FrontPage, que dejara de crear documentos compatibles con otros productos (es decir, que dejara de crearlos basados en estándares). Gates no quería que productos de la competencia pudieran abrir, guardar y modificar documentos generados por Microsoft. Como resultado, si utiliza FrontPage para diseñar o programar sitios Web, prácticamente se asegura de que sólo funcionen correctamente en Internet Explorer.

Pero todavía hay esperanza. En julio del 2002, UsableNet anunció que había integrado su herramienta de accesibilidad LIFT en Microsoft FrontPage (www.usablenet.com/frontend/onenews.go?news_id=45), con lo que alentaba a los usuarios de FrontPage a crear sitios accesibles (LIFT también se incluye en Dreamweaver MX). Aunque no es una garantía de que la compatibilidad con estándares sea una de las nuevas características de FrontPage, este nuevo énfasis en la accesibilidad es al menos un pequeño paso en esta direccion.

Sin embargo, por el momento, si quiere crear sitios accesibles compatibles con estándares, la mejor opción es crearlos manualmente en un editor de texto o con ayuda de Dreamweaver o GoLive. Utilice FrontPage únicamente si está preparado para editar a mano todas las etiquetas y atributos que genere el producto.

La emergencia del diseño CSS

La especificación CSS1 se creó en 1996 para acabar con el HTML de presentación y separar el aspecto de un sitio de los datos que contenía. En el 2000, la mayoría de los principales navegadores era capaz, al fin, de mostrar correctamente diseños en CSS. Pero los diseñadores y programadores se resistían a adoptar CSS ya que una parte importante de sus visitantes utilizaba navegadores 4.0 y anteriores que no eran compatibles. Había que hacer algo para que los estándares Web resultaran seguros para los grandes diseñadores y programadores. El Web Standards Project decidió que eran necesarias tácticas de guerra de guerrillas.

La campaña de actualización de navegadores

En febrero del 2001, el Web Standards Project inició su campaña de actualización de navegadores (www.webstandards.org/act/campaign/buc/). Como su nombre indica, el objetivo de esta campaña era que los consumidores probaran nuevos navegadores con mayor compatibilidad con estándares, con lo que se convencía a los diseñadores para que utilizaran estándares en lugar de HTML y código propietario.

En algunos casos, la campaña alentaba a los programadores a que actuaran como si todo su público ya utilizara navegadores compatibles con estándares única y exclusivamente. Un fragmento de JavaScript en la etiqueta `<head>` de un documento podía demostrar el uso del DOM en cualquier navegador que accediera a dicha página. Si el navegador usaba el DOM, también entendía la página. En caso contrario, se dirigía al usuario a otra página que su navegador pudiera entender.

Esa página aconsejaba al visitante que actualizara su navegador con una versión más reciente de IE, Netscape u Opera, u otros

navegadores compatibles. También le indicaba por qué dicha actualización mejoraría su utilización de la Web.

Al contrario de las antiguas campañas de marketing de tipo "Se aconseja verlo en...", la campaña de actualización de navegadores del WaSP no creía en los fabricantes. No nos importaba qué navegador descargara el usuario, siempre que fuera uno compatible con estándares.

Esta técnica de fuerza bruta sólo era recomendada para sitios que utilizaran correctamente el diseño CSS y el DOM, y sólo para sitios no comerciales que podían permitirse el hecho de enviar usuarios a otros sitios. A los diseñadores y programadores que participaron se les invitó a crear sus propias páginas de actualización de navegadores personalizadas en función de las necesidades de sus visitantes, a utilizar estas técnicas exclusivamente en sitios válidos y compatibles con estándares, y a valorar detenidamente las ventajas e inconvenientes de tomar esta decisión.

Uso y abuso de las actualizaciones de navegadores

Con demasiada frecuencia hubo programadores que utilizaron la técnica para alejar a los usuarios de sitios que ni remotamente cumplían los estándares. Para completar el daño, en lugar de crear sus propias páginas de actualización, estos programadores los enviaban a las del WaSP. Como habrá adivinado, estos esfuerzos resultaban en la frustración de los consumidores y no en que probaran navegadores compatibles, como se deseaba.

Entre los culpables se encontraba un sitio que incluía imágenes de animadoras. `Raiderettes.com` (véase figura 4.9), uno de los que más referencia hacía a la página de la campaña de actualización de navegadores del Web Standards Project. Su aspecto es aceptable, pero al intentar validar su marcado (http://validator.w3.org/), se genera el siguiente informe:

```
Fatal Error: no document type declaration;
will parse without validation.
I could not parse this document, because
it uses a public identifier that is not
in my catalog.
```

`Raiderettes.com`, como otros muchos sitios comerciales, había usado el código de redirección de navegadores del WaSP, no en beneficio de los estándares, sino simplemente como medio para rechazar navegadores incapaces de procesar sus menús DHTML. No hace falta decir que el Web Standards Project recibió multitud de correos de hombres enfurecidos por la testosterona que esperaban disfrutar con las señoritas, no aprender sobre CSS y ECMAScript. No hay nada más desagradable que el rencor de un hombre al que se le ha negado su momento voyeur (a excepción, quizás, de ese hombre).

Una actualización más amable

Aparte de estos fracasos, la campaña tuvo mucho éxito ya que animó a miles de diseñadores y programadores a que dieran una oportunidad a los estándares Web. También despertó la conciencia acerca de los estándares en la comunidad empresarial, captando la atención de la prensa especializada. Y, de vez en cuando, persuadió a los consumidores para que descargaran navegadores compatibles.

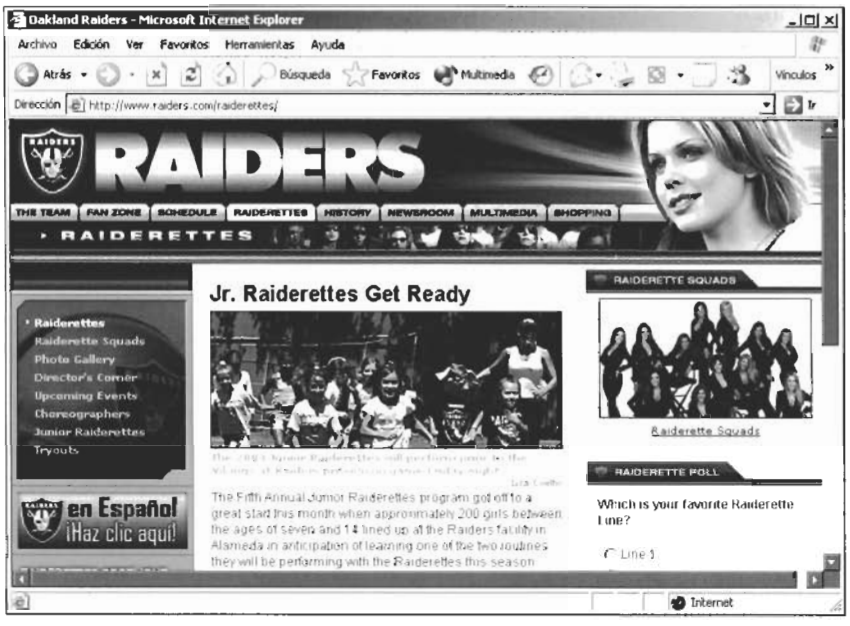


Figura 4.9.

Raiderettes.com, sitio oficial del equipo de animadoras Oakland Raider (www.raiderettes.com). ¿Qué tienen que ver las biografías de animadoras y las fotos de modelos con los estándares Web? Siga leyendo.

Aunque la campaña de actualización de navegadores cobró notoriedad gracias a sus travesuras de bloqueo de navegadores, constituyó un esfuerzo múltiple por ofrecer tácticas adecuadas para los usuarios que recibió menos apoyo de la prensa pero que tuvo efectos positivos. Para apoyar a la nueva campaña y demostrar su lado más amable con el usuario, A List Apart (www.alistapart.com) cambió su diseño utilizando CSS y HTML válido (que después dejaría paso a XHTML), como mencionamos en un capítulo anterior.

En un intento por demostrar la flexibilidad de la campaña, ALA evitó excluir deliberadamente a ningún visitante. Por el contrario, el sitio simplemente ocultaba su diseño a los navegadores no compatibles. Los que usaran IE5+, Netscape 6, Opera 5+ y navegadores con la misma compatibilidad, obtenían diseño más contenidos. Los que utilizaban navegadores antiguos sólo obtenían contenidos. Los usuarios de navegadores no compatibles también veían un mensaje de "actualización del navegador" que se ocultaba a los usuarios de navegadores con mayor compatibilidad. Los estándares funcionaban a pleno rendimiento y todo el mundo era bienvenido.

Para atraer el interés de los diseñadores, ALA disfrazó su enfoque apto para el usuario en forma de propaganda, describiendo su cambio de diseño como si fuera un manifiesto y haciendo hincapié en que su diseño se ocultaría a los navegadores incompatibles con CSS, a los que la revista, indignada, mandaba al infierno (véanse figuras 4.10 y 4.11).

Figura 4.10.

"Al infierno con los navegadores antiguos". Este manifiesto retaba a los diseñadores a utilizar CSS en sus diseños, el DOM en su código y dejaran que el marcado fuera marcado en lugar de forzar a HTML para que se encargara del diseño (www.alistapart.com/stories/tohell/).

Figura 4.11.

La página central en CSS de ALA No. 99, como apareció en febrero del 2001, coincidiendo con el inicio de la campaña de actualización de navegadores (www.alistapart.com/stories/99/).

A List Apart retó a sus 70.000 lectores semanales a que dejaran de poner excusas y que empezaran a incorporar estándares en sus sitios. Como se aprecia en la figura 4.11, se había escrito un prólogo en una servilleta durante una conferencia Web y después se había rediseñado en CSS. El texto es éste:

> *Dentro de seis meses, un año o dos, a lo sumo, todos los sitios Web se diseñarán con estándares que separan la estructura de la presentación. (O se crearán con Flash 7.) Podemos contemplar cómo nuestros conocimientos se quedan obsoletos o podemos empezar a aprender técnicas basadas en estándares ahora mismo.*
>
> *De hecho, como las últimas versiones de IE, Navigator y Opera ya admiten muchos estándares Web, si estamos dispuestos a abandonar la idea de que la compatibilidad inversa es una virtud, podremos dejar de poner excusas y empezar a usar los estándares.*
>
> *En ALA, comenzando con el número 99, ya lo hemos hecho. Únase a nosotros.*

Comienza la riada

El editorial dio en el clavo. En cuestión de días, los sitios independientes, uno tras otro, empezaron a convertir sus diseños a CSS y su estructura a XHTML. El sitio diario de Todd Dominey (véase figura 4.12) es una muestra representativa de este tipo de conversiones por su elección de la tecnología, mientras que la calidad de escritura y diseño se encuentra por encima de la media. Profesional de los medios afincado en Atlanta, Dominey utiliza Flash en su galardonado sitio (véase figura 4.13) y CSS y XHTML en su registro Web diario.

Figura 4.12.

Las dos caras de Todd Dominey, un nuevo diseñador. Su acertado sitio diario (mostrado en la imagen) utiliza XHTML para la estructura de los documentos y CSS para el diseño (www.whatdoiknow.org).

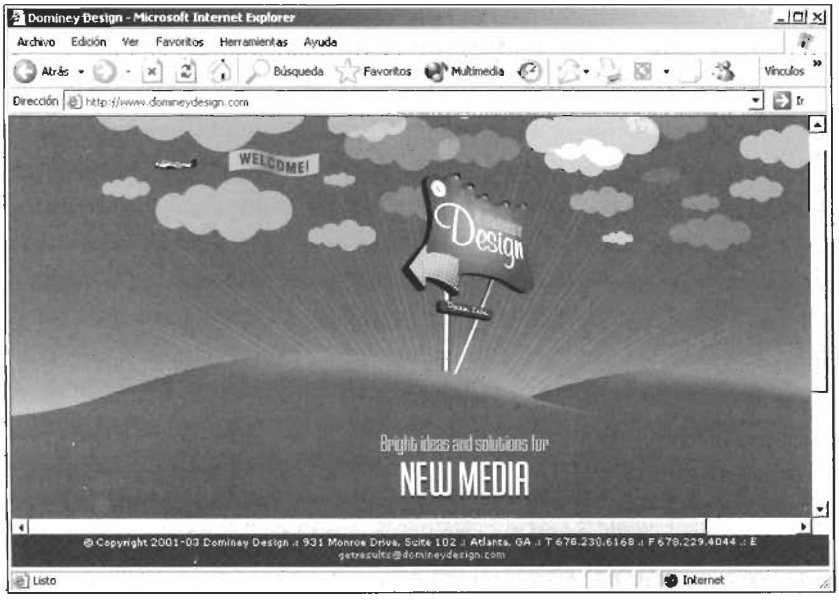


Figura 4.13.

El sitio por el que Dominey ha recibido multitud de galardones utiliza Flash (www.domineydesign.com).

En este último sitio, Dominey explica la razón de la conversión a los estándares Web:

> *Este sitio utiliza técnicas de diseño XHTML/CSS. No hay espaciadores GIF transparentes. No hay filas. No hay columnas. No hay elementos sobrantes. El motivo es muy sencillo: soy un nuevo diseñador y necesitaba un lugar para experimentar con técnicas de diseño avanzadas que el trabajo con los clientes no me permite emplear (sólo en algunas excepciones). Las técnicas de diseño están en la cresta de la ola y se actualizarán continuamente en los nuevos navegadores. Si utiliza, como mínimo, Internet Explorer 5.0 (Mac y Windows) o superior, además de Mozilla, Netscape 6 y Opera, no tendrá problema. Con cualquier otro navegador anterior o beta, como iCab y OmniWeb, seguramente los tendrá. Si todo lo que ve es un texto azul sobre un fondo gris, necesita desesperadamente actualizar su navegador.*
>
> *www.whatdoiknow.org/about.html.*

Innumerables conversiones y los sitios de ayuda en las que se asientan

En los dos años posteriores a que la campaña de actualización de navegadores y de cambio de diseño de ALA impactara en la comunidad de diseñadores en forma de de gancho doble, infinidad de sitios personales y de registros Web adoptaron CSS/XHTML. (Quizás no como un gancho doble. Somos diseñadores, no fanáticos de la lucha, por lo que nuestros símiles deportivos son dudosos.)

Para facilitar el repentino interés por el diseño CSS, varios sitios personales publicaron diseños CSS de código abierto. ¿No comprende el funcionamiento de CSS? ¿Duda de la aplicación de estilos a sus páginas? Visite cualquiera de estos sitios para aprender o para copiar y pegar:

- Layout Reservoir de Blue Robot (véase figura 4.14).
- Técnicas de diseño CSS de Eric Costello (http://glish.com/css/).
- Página de diseños visuales "Little Boxes" de Owen Brigg (véase figura 4.15).

Si sus ambiciones son más agresivas, Eric Meyer le ofrece "css/edge" (véase figura 4.16), donde se incluyen revolucionarias técnicas de diseño CSS que únicamente funcionan en los navegadores con mayor compatibilidad. Por naturaleza, no se pueden utilizar dichas técnicas en la mayoría de los trabajos cliente, pero sirven de base para el trabajo que realice en años venideros.

La mayoría de los diseñadores CSS se conforman con crear sitios que se puedan ver correctamente en IE5 o superior y que sufran pequeños cambios (o que únicamente muestren el contenido sin diseño) en Netscape 4. Real World Style de Mark Newhouse (véase figura 4.17) cuenta con diseños CSS que funcionan aceptablemente en Netscape 4 y otros navegadores antiguos.

Si quiere utilizar CSS pero no se puede permitir prescindir de navegadores antiguos, en Real World Site encontrará soluciones para ello.

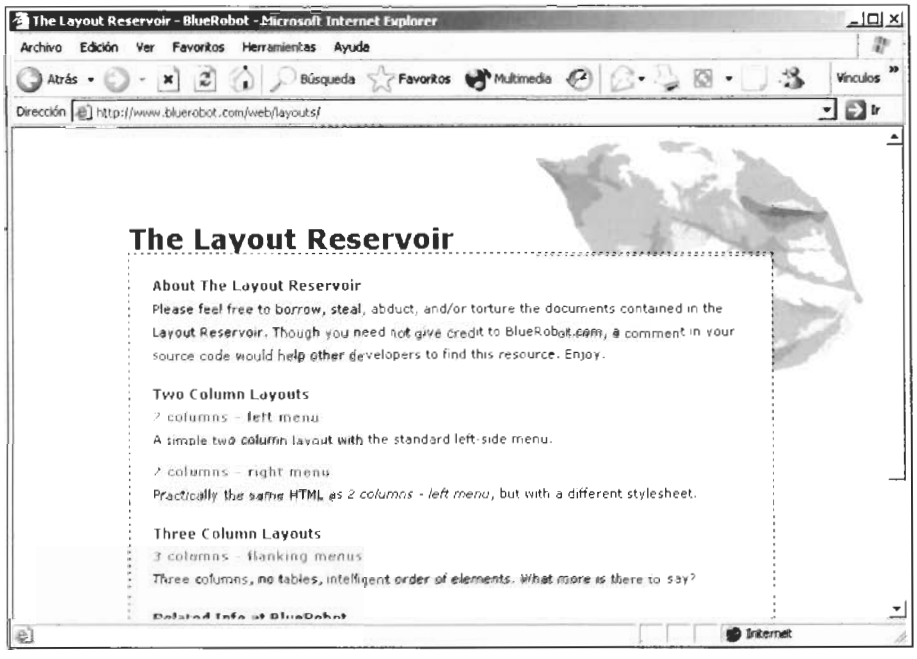


Figura 4.14.

Layout Reservoir de Blue Robot ofrece sencillos y útiles diseños CSS, que puede usar de forma gratuita (www.bluerobot.com/web/layouts/).

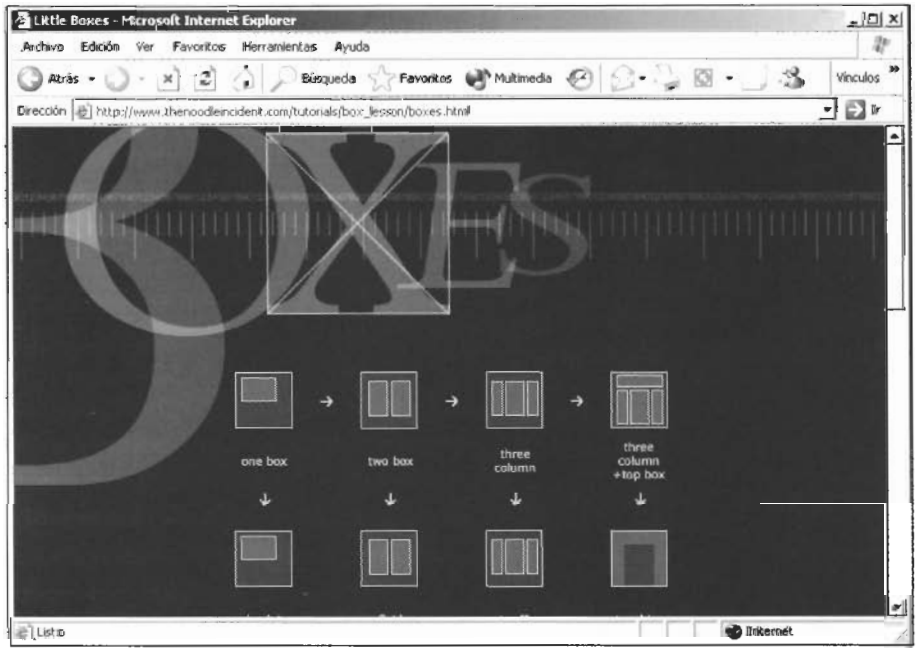


Figura 4.15.

Página de diseños visuales "Little Boxes" de Owen Brigg (http://www.thenoodleincident.com/tutorials/box_lesson/boxes.htm). Seleccione un diseño, pulse sobre el mismo y consiga la CSS utilizada para crearlo (que puede utilizar y adaptar en sus propios proyectos Web).

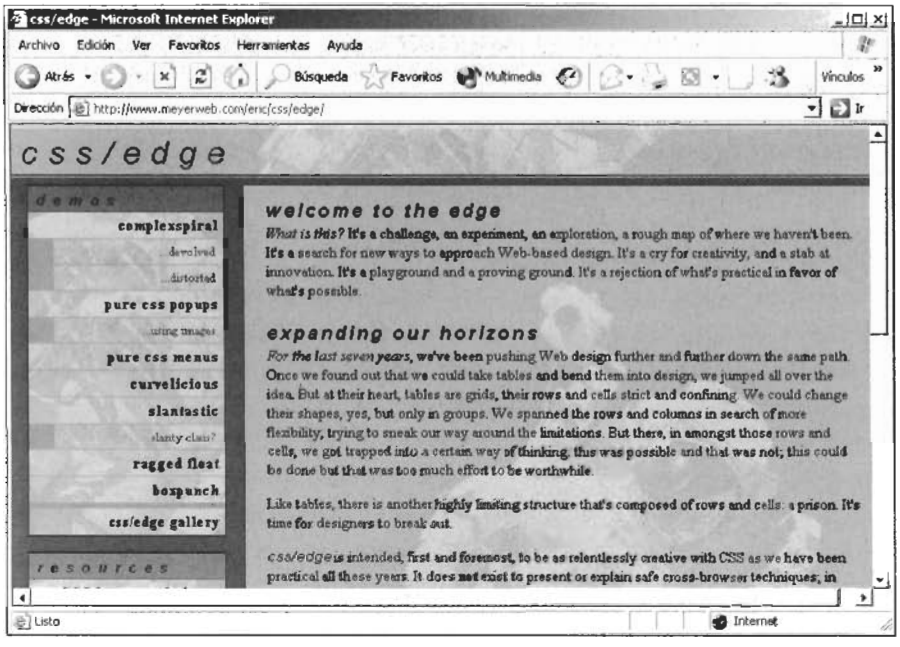


Figura 4.16.

css/edge de Eric Meyer incluye revolucionarias técnicas que sólo funcionan en los navegadores más modernos y con mayor compatibilidad.

Figura 4.17.

En el otro lado del espejo, nos encontramos con Real World Style de Mark Newhouse (www.realworld-style.com). Ofrece diseños CSS de código abierto que funcionan aceptablemente incluso en navegadores tan antiguos como Netscape 4. (Evidentemente, en navegadores más modernos y compatibles, el funcionamiento es mejor.)

Caprichos justificados

Con la aparición de CSS/XHTML en el mundo de los sitios personales e independientes, se añadió a la mezcla la accesibilidad a través del cumplimiento con la sección 508 o las directrices de accesibilidad WAI. Al cierre de la edición de este libro, los principales sitios independientes se agrupaban en dos categorías (como indican las producciones duales de Todd Dominey):

- Sitios basados en estándares y con compatibilidad directa que utilizan CSS en el diseño y XHTML en la estructura, y que cumplen con la sección 508 u otras directrices de accesibilidad.

- Sitios de Flash que aprovechan las prestaciones de ActionScript (un lenguaje de secuencia de comandos de Macromedia basado en ECMAScript estándar).

Conviene decir que si su sitio personal no recurre a los estándares Web ni a Flash, muchos de los integrantes de la comunidad de diseño pensarán que está fuera de onda. Esta opinión puede parecer caprichosa y trivial, y de hecho lo es. Pero al dominar los errores de la naturaleza humana se consigue un objetivo superior: la adopción de técnicas aparentemente directas en los principales sitios.

El liderazgo proviene de los individuos

Los sitios personales e independientes siempre han superado las barreras y sus innovaciones siempre han inspirado cambios en los sitios más conocidos. JavaScript, los marcos y las ventanas emergentes aparecieron en primer lugar en los sitios personales más sofisticados. Cuando nadie se interesaba por visitar dichos sitios, los diseñadores comerciales reconocieron que resultaba seguro incorporar JavaScript, marcos y ventanas emergentes en sus trabajos. Por esta razón, ahora vemos menús desplegables DHTML en sitios de comercio electrónico (www.coach.com/index_noflash.asp) y molestos anuncios emergentes en la ración diaria de www.nytimes.com (lo que no siempre es agradable).

La transferencia de tecnologías de sitios independientes a sitios corporativos será mejor en esta ocasión ya que, ahora, la sofisticación se ha centrado en estándares y accesibilidad compatible. Auspiciado por un lado por sitios personales de gran inspiración y, en otro, por las nuevas leyes, el cambio ya ha comenzado.

El corporativismo de los estándares Web

En el 2001, Estados Unidos y Canadá publicaron una serie de directrices en las que exigían que los sitios gubernamentales se diseñaran con estándares Web y fueran accesibles. A esta iniciativa se sumaron los gobiernos de Reino Unido y Nueva Zelanda, así como numerosos estados americanos.

En el 2002, los principales sitios relacionados con el gobierno, incluyendo Texas Parks & Wildlife (www.tpwd.state.tx.us) y la página de Juneau, Alaska (véase figura 4.18) se convirtieron a estándares Web e incluyeron diseño CSS.

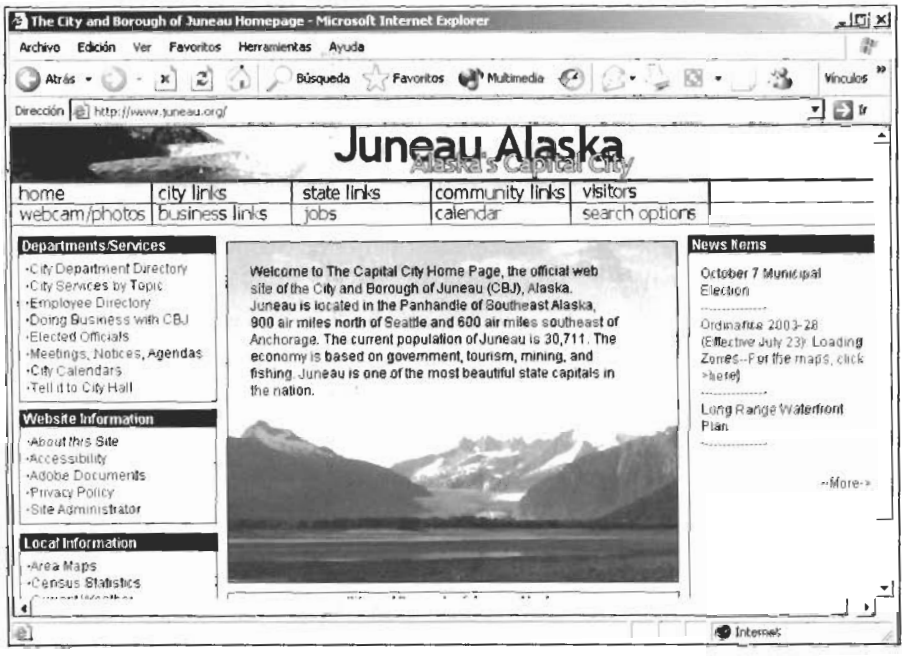


Figura 4.18.
Bienvenido a Juneau, Alaska, cuna del turismo, la pesca, la minería y las hojas de estilo (www.juneau.org). Se trata de uno de los primeros sitios públicos que se desprendió de los marcos de la vieja escuela en favor de un diseño CSS y un marcado XHTML.

Texas Parks & Wildlife explicaba de esta forma su conversión (www.tpwd.state.txt.us/standards/tpwbui.htm):

> *Texas Parks & Wildlife, como agencia estatal, de acuerdo al código administrativo 201.12, debe diseñar el código de sus páginas Web según los estándares Web definidos por el W3C. El motivo de estos requisitos es la accesibilidad. Las páginas Web deben ser accesibles para todos los usuarios, independientemente de la tecnología utilizada para acceder a las mismas o de las incapacidades de los usuarios.*
>
> *Pero si las páginas no tienen el mismo aspecto para todos los usuarios, ¿cómo pueden ser accesibles?*
>
> *El hecho de que sean accesibles no significa que todo el mundo vea lo mismo. La accesibilidad está relacionada con el contenido y la información. Implica que todo el contenido (texto, imágenes y multimedia) esté disponible para el usuario. Para llevarlo a cabo y cumplir con los estándares, TPW utiliza Hojas de estilo en cascada para separar el contenido de la presentación. De esta forma puede ofrecer páginas de mayor calidad y un contenido dinámico.*
>
> *Los estándares de código Web establecidos por el W3C permitieron la creación de páginas accesibles. Al utilizar estándares W3C como las Hojas de estilo en cascada y XHTML, Texas Parks & Wildlife empezó a crear páginas Web compatibles.*

Los programadores de la página Capital City de Alaska ofrecieron un razonamiento similar (www.juneau.org/about/stylesheets.php):

Nuestra intención es convertir el sitio CBJ de una estructura de marcos al estándar Hojas de estilo en cascada (CSS). Este estándar nos ofrece una mayor libertad de diseño mientras que aumenta la accesibilidad del contenido para nuestros usuarios. Al utilizar hojas de estilo, podemos ofrecer nuestros contenidos prácticamente a cualquier navegador y plataforma informática, incluyendo dispositivos manuales y ADA, desde la misma página en lugar de utilizar duplicados de las mismas de sólo texto u orientados a impresión. Del mismo modo, nos permite evitar muchas de las limitaciones que supone nuestro diseño basado en marcos.

El único inconveniente es que muchos navegadores antiguos no son compatibles con el uso de hojas de estilo. Sin embargo, la ventaja de este inconveniente es que las hojas de estilo son mucho más amables con los navegadores no compatibles de lo que lo son los marcos con los navegadores no compatibles con marcos. Los navegadores incompatibles con hojas de estilo muestran el contenido de una página Web CSS en un formato de texto. Aunque su aspecto no sea el mejor, se muestra toda la información, incluyendo vínculos e imágenes. De hecho, al utilizar hojas de estilo, el diseñador Web puede controlar el orden en el que se muestra el contenido en un navegador incompatible sin que esto afecte a la forma en que se muestra el sitio en navegadores compatibles. Esta característica es una de las principales ventajas de las Hojas de estilo en cascada, lo que permite que las páginas Web se puedan emplear tanto en navegadores nuevos como en los antiguos. Como diseñador de sitios Web, prefiero que los usuarios digan que el aspecto de mis páginas Web es "un poco raro" a que digan que no funcionan.

La mayoría de las versiones más modernas de los navegadores más conocidos son compatibles con CSS. Si su navegador actual no es compatible con hojas de estilo, debe descargar uno que lo sea. En sistemas más antiguos (con procesadores más lentos y menor memoria), puede utilizar el navegador Opera, ya que sus requisitos de sistema son inferiores a los de sus competidores.

Los sitios comerciales dan el salto

En el 2002, los principales sitios comerciales también empezaron a convertirse a las técnicas CSS y XHTML que sus programadores habían desdeñado anteriormente por razones de "compatibilidad inversa".

En julio de ese mismo año, el motor de búsqueda Lycos Europe cambió a marcado XHTML y diseño CSS, seguido de cerca, en diciembre, por HotBot, propiedad de American Lycos.

En ese mismo periodo, el motor de búsqueda superrápido AllTheWeb (véase figura 4.19) también se convirtió a CSS y XHTML, y añadió hojas de estilo definidas por el usuario a sus opciones de personalización.

Al cierre de la edición de este libro, Google y Yahoo todavía no han secundado la iniciativa de estos competidores menores, pero como hemos dicho antes, ésa es la naturaleza del cambio: los sitios independientes son los primeros en arriesgarse y los grandes les siguen cuando saben que es seguro hacerlo.

La conversión de Wired Digital

En septiembre del 2003, el conocido sitio Wired Digital se remodeló y se convirtió en un sitio compatible con estándares: XHTML para datos y CSS para la presentación (véanse figuras 4.20 y 4.21). El director Douglas Bowman (véase figura 4.22) supervisó el cambio de diseño y elevó la compatibilidad con los estándares a la prioridad principal.

Como sitio con gran cantidad de tráfico y una gran reputación en cuanto al entendimiento y el uso de la tecnología Web, Wired ha sido un faro en la comunidad de programadores. Su conversión a los estándares Web envió un claro mensaje a la industria, indicando que había llegado la hora de apostar por la compatibilidad directa en lugar de otras preocupaciones.

Figura 4.19.

AllTheWeb, el motor de búsqueda candidato a derrocar a Google (si es que alguien puede), adoptó la estructura XHTML y diseño CSS en 2002 (www.alltheweb.com).

Figura 4.20.

Wired Digital (www.wired.com) se convirtió a los estándares en el 2002, y utilizó XHTML para la estructura y CSS para su diseño.

Figura 4.21.

Al ver la misma página en un navegador no compatible (en este caso, Netscape 4), el acceso al contenido es total, pero se pierde el diseño, utilizando la misma técnica implementada en el cambio de diseño de A List Apart del 2001. También se ofrecía un enlace a la campaña de actualización de navegadores del The Web Standards Project para recomendar el cambio a navegadores más modernos.

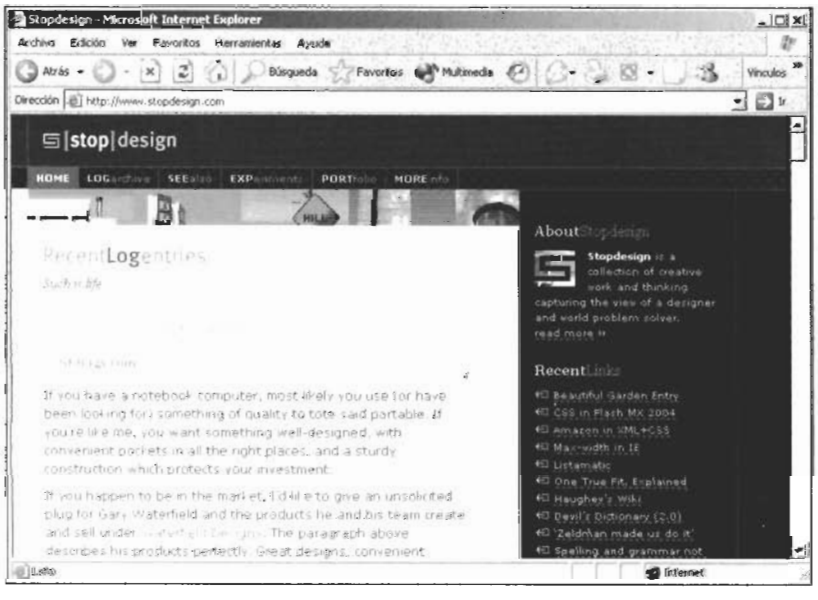


Figura 4.22.

Aunque muchos diseñadores y programadores participaron en el cambio de diseño de Wired Digital, su orientación hacia los estándares se debía principalmente a la labor evangélica de Douglas Bowman, que posteriormente abandonó Wired para dedicarse a su propia empresa, Stop Design. El sitio de Stop Design (www.stopdesign.com) también utiliza brillantemente el diseño CSS y un XHTML bien estructurado.

Los sistemas de publicación y los estándares

La validación XHTML/CSS completa apenas se consigue en sitios comerciales de gran escala, aunque las plantillas iniciales y los clientes y fabricantes se comprometan con el apoyo de las especificaciones W3C. En realidad, los clientes y fabricantes con esta preocupación son escasos. Pero aquéllos que sí se preocupan a menudo encuentran que todos sus esfuerzos se ven reducidos por dispositivos obsoletos, bases de datos comprometidas y todo tipo de bloqueos similares.

Como hemos mencionado anteriormente, los diseñadores y programadores independientes no tienen problema alguno para adoptar las especificaciones sobre CSS y XHTML del W3C. Algunos incluso disfrutan con la modificación de sitios comerciales a los que no están afiliados y en los que aplican las especificaciones en cuestión de horas (como ha hecho Eric Meyer con el sitio KPMG que mencionamos en un capítulo anterior).

No resulta difícil conseguir XHTML y CSS válidos, y una accesibilidad de prioridad 1. Cualquier diseñador y programador medianamente competente puede hacerlo. El problema radica en los sistemas de gran escala y en contenidos de terceros que deben integrarse en los sitios de gran tamaño.

Para garantizar la salud y la compatibilidad de nuestros sitios, necesitamos sistemas de administración de contenidos que promuevan prácticas de accesibilidad y compatibilidad en lugar de destrozarlas. Los encargados de diseñar estos sistemas deben admitir que el hecho de comprometer la interfaz para forzar un problema de pantalla ya no es una solución aceptable. Y también necesitamos clientes dispuestos a invertir en sistemas que funcionen en favor de la compatibilidad directa. Para ello hace falta persuasión y dinero.

La persuasión es ilimitada. El dinero es un tema aparte.

En una economía saneada, las empresas invierten en I+D, en formación y en planificación. En una economía enferma, las empresas se centran en reducir gastos, eliminar personal y procesos, y se preocupan en poder abrir al día siguiente. Resulta muy sencillo alentar a programadores y diseñadores para que utilicen métodos compatibles: basta con mostrarles los beneficios.

Convencer a las empresas que sufren problemas de que inviertan en la salud a largo plazo de su presencia en la Web es el verdadero desafío al que se enfrentan aquéllos que quieren salvar a la Web de la oscuridad y devolverla a la luz.

Los ingredientes equivocados

En un principio, el lanzamiento de Wired Digital se vio ensombrecido por los errores de validación. Algunos de estos errores se debían al sistema de administración de contenidos Vignette utilizado por el personal editorial de Wired. Otros se debían a contenidos publicitarios de terceros diseñados con métodos no válidos y un incorrecto procesamiento de URL. Mi agencia, Happy Cog, se enfrentó a obstáculos similares, y puede que también se le presenten en sus propios proyectos. Una vez conseguidas las plantillas compatibles, el uso de un sistema de administración de contenidos (CMS) no actualizado añade marcado basura y código propietario, lo que reduce la compatibilidad del sitio. Si añade estos ingredientes a su comida, ya no será un plato sano.

En el caso de Wired, estos problemas se solucionaron rápidamente (principalmente por la insistencia de Bowman), poco después de su aparición. Wired Digital era un sitio comercial a gran escala que cumplía perfectamente los estándares. No todas las empresas disponían de los recursos tecnológicos para solucionar los errores generados por Vignette y similares, ni tampoco se lo podían permitir.

Para que la Web progrese, las herramientas de publicación deben ser compatibles con los estándares (como se indica a continuación). Los propietarios y administradores de sitios deben convencer a los distribuidores CMS de la importancia de la compatibilidad, al igual que los diseñadores y programadores convencieron a los fabricantes de navegadores. Si hay suficientes clientes para ello, puede que los distribuidores acepten actualizar sus herramientas y la consecución de la compatibilidad con estándares será mucho más fácil.

Adopción de estándares mediante medios tradicionales

No toda la compatibilidad con estándares requiere diseños CSS y no todas las empresas e instituciones dispuestas a aprovechar las ventajas de la compatibilidad que ofrecen los estándares Web necesitan un paso tan avanzado. En un capítulo anterior describimos un método intermedio de cumplimiento de los estándares que mezclaba lo antiguo y lo nuevo en una combinación de compatibilidad directa. Se trata de un método que atrae a empresas e instituciones que prefieren un cambio paulatino pero firme a la conversión evangélica. Resulta especialmente apto para organizaciones con una amplia base de navegadores antiguos y no compatibles. Una de estas instituciones es la New York Public Library que, en el 2001 y bajo la dirección del entonces coordinador Web Carrie Bickner, convirtió sus Branch Libraries (www.nypl.org/branch/) a marcado XHTML 1.0 y utilizó una combinación de diseños tradicionales basados en tablas y Hojas de estilo en cascada para la presentación. La biblioteca también cumple con las directrices de accesibilidad U.S. Section 508 para aplicar todos sus patrones de forma más eficaz.

Simultáneamente con la conversión del sitio a los estándares, la biblioteca publicó una guía de estilo (www.nypl.org/styleguide/), escrita por Bickner y un servidor, en la que se explicaban los requisitos de marcado y diseño de todos los proyectos de bibliotecas filiales, y que incluía un tutorial sobre XHTML y CSS.

En lugar de confinar la guía de estilo al sitio interno de la biblioteca, NYPL optó por hacerla pública, a la espera de que ayudaría a miles de diseñadores y programadores a que adoptaran XHTML y CSS, y que alentaría a otras instituciones a que admitieran los estándares. Al cierre de la edición de este libro, ambos frentes han mostrado un considerable éxito.

Al igual que en un principio despertaron la imaginación de los diseñadores independientes, actualmente los estándares inspiran a los diseñadores y programadores institucionales, que ven los estándares Web y la accesibilidad como las claves para conseguir llegar a todo el público.

El W3C entra en acción

En un principio, la política del W3C consistía en publicar sus estándares, no en forzarlos. Pero la era de pasividad del W3C llega a su fin. En el 2001, el W3C creó el grupo Quality Assurance (http://www.w3.org/QA/) para mejorar su relación con la comunidad de diseñadores y programadores, y para garantizar que las especificaciones W3C se usaran e implementaran correctamente. También comenzó a publicar una serie de artículos para explicar y promocionar sus especificaciones.

El borrador del W3C (www.w3.org/WAI/bcase/benefits.hmtl) sobre los beneficios empresariales del diseño accesible basado en estándares destaca una mayor cuota de mercado y un público más amplio, la racionalización de la eficacia (con reducción de costes), una menor fiabilidad legal y una clara demostración de la responsabilidad social. Merece la pena leer, imprimir y compartir este artículo.

Si las ventajas que apunta el artículo del W3C le resultan familiares, sin duda habrá leído este libro en lugar de simplemente hojear sus páginas en un estado de semihipnosis. Si las ventajas le parecen atractivas, es que lo son. Los estándares y la accesibilidad aumentan el público de destino a la vez que reducen los gastos. Si esta propuesta no atrae el interés de algún empresario, seguramente no viva en el mismo planeta que nosotros.

Resumen

Los estándares como XML han cambiado la cara de las empresas. La compatibilidad con los estándares ha conseguido acabar con las guerras de los navegadores y con el peaje impuesto sobre la accesibilidad y la viabilidad a largo plazo, marcando el comienzo de una nueva era de cooperación incluso entre los más fieros competidores. Estándares como XHTML y CSS, admitidos en los principales navegadores y herramientas de creación profesionales, han cambiado los métodos de diseño de sitios a mejor; no sólo en los más sofisticados sitios independientes y personales, sino también en sitios gubernamentales, institucionales y comerciales. Los estándares Web ganan la partida, y aquí estamos nosotros.

Llega el momento de dejar los halagos y ponernos manos a la obra. El siguiente capítulo lanza la pelota a su cancha (ya le dijimos que no somos muy buenos con las metáforas deportivas...).

Parte II
Diseño y creación

Capítulo 5
Marcado moderno

En los capítulos anteriores, describimos los problemas creativos y empresariales generados por los métodos de diseño Web anticuados, analizamos los beneficios del diseño y la creación por medio de estándares, y definimos someramente los avances del medio. En el resto del libro, pasaremos de lo general a lo particular, y el mejor punto de partida es detenernos por segunda vez en los fundamentos del marcado Web.

Muchos diseñadores y programadores se resistirán a ello. Probablemente los que hemos dedicado más de unas semanas al diseño de sitios profesionales sepamos todo lo necesario sobre HTML. ¿Los diseñadores y programadores no tienen que aprender nuevos y más potentes lenguajes en su tiempo libre? Por ejemplo, ¿no es más importante estudiar tecnologías del lado del servidor como PHP, ASP o ColdFusion (como veremos más adelante) que malgastar el tiempo en pensar en rudimentos como tablas HTML o etiquetas de párrafo?

La respuesta es sí y no. Las tecnologías del lado del servidor son de vital importancia para crear sitios dinámicos que respondan a las peticiones de los usuarios.

Incluso los tradicionales sitios informativos se pueden beneficiar si se almacenan sus contenidos en bases de datos y se recuperan cuando sea necesario a través de PHP o de tecnologías similares. De hecho, al igual que los estándares Web analizados en este libro, los lenguajes de secuencia de comandos del lado del servidor realizan una función similar al abstraer datos de la interfaz. Al igual que los estándares como CSS liberan al diseñador de la necesidad de recluir cualquier fragmento de contenido en celdas de

tabla sin sentido semántico y que consumen ancho de banda, lo mismo hacen lenguajes como PHP y ASP, que liberan a los creadores de sitios de tener que crear todas las páginas a mano.

Pero las páginas Web generadas dinámicamente no servirán de nada si son inaccesibles, incompatibles con diferentes navegadores y dispositivos, o están repletas de marcado basura. Si dichas páginas dinámicas no se representan en algunos navegadores o dispositivos, o tardan más de 60 segundos en cargarse a través de una conexión telefónica cuando deberían tardar 10 segundos, eso quiere decir que las tecnologías del lado del servidor no funcionan como deberían. No debe optar por una opción o por otra, sino combinar las dos. Las tecnologías del lado del servidor y las bases de datos crean sitios más potentes, pero lo que ofrecen dichos sitios es contenido que funciona mejor si se estructura de manera semántica y limpia. Y es lo que nos falta a muchos de nosotros (y a muchos de los sistemas de administración de contenidos de los que dependemos).

¿Qué es PHP?

PHP (www.php.net) es un lenguaje general de secuencia de comandos de código abierto especialmente indicado para la programación Web y que se puede incrustar en XHTML. Su sintaxis se basa en C, Java y Perl, y resulta relativamente sencillo de aprender. PHP, que significa Preprocesador de hipertexto, cuenta con numerosas prestaciones pero su fama se debe a una en concreto: al utilizar PHP con una base de datos MySQL (www.mysql.com), los diseñadores y programadores pueden crear sitios Web dinámicos y diseñar aplicaciones Web.

PHP es un proyecto de la Apache Software Foundation (www.apache.org) y es gratuito, una de las razones de su popularidad (otra es la amplia variedad de herramientas de perfiles y depuración). Los programadores de código abierto y los diseñadores Web independientes están enamorados de este lenguaje y continuamente crean aplicaciones basadas en PHP que distribuyen de forma gratuita. Por ejemplo, Refer de Dean Allen (www.textism.com/tools/refer/) realiza el seguimiento de los visitantes que han seguido un enlace de un tercero hasta su sitio (véase figura 5.1). URL Cleaner de Dan Benjamin (www.hivelogic.com/urlcleaner.php) arregla direcciones Web no válidas creadas por otros lenguajes de secuencia de comandos como ColdFusion (véase figura 5.2).

PHP se puede ejecutar con software de servidor de Microsoft, pero se suele utilizar con el servidor Apache. Apache funciona tanto en Windows como en UNIX. El sistema operativo Mac OS X de Apple basado en UNIX incluye PHP y el servidor Apache, como en la mayoría de las distribuciones de Linux.

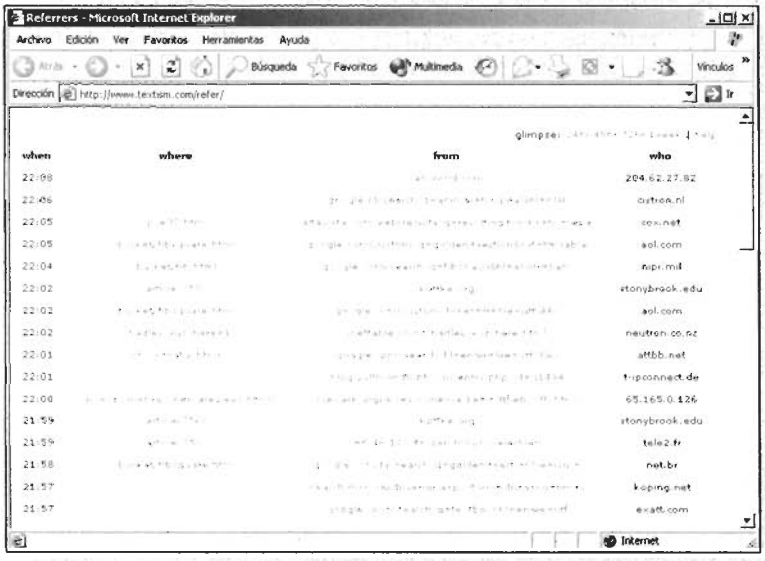


Figura 5.1.

Por medio del lenguaje de secuencia de comandos PHP, Refer de Dean Allen (www.textism.com/tools/refer/) realiza el seguimiento de los visitantes que han seguido un vínculo de terceros a nuestro sitio, y cualquier diseñador o programador puede utilizarlo de forma gratuita.

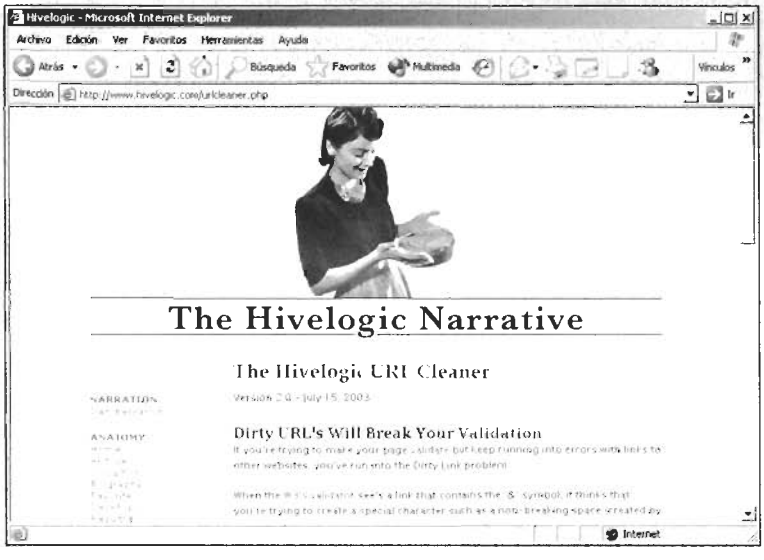


Figura 5.2.

URL Cleaner de Dan Benjamin (www.hivelogic.com/urlcleaner.php), creado en PHP, arregla direcciones Web no válidas.

Las Páginas de servidor activo de Microsoft o ASP (www.asp.net) y Macromedia ColdFusion (www.macromedia.com/software/coldfusion/) son dos lenguajes de secuencia de comandos muy conocidos que se utilizan para ofrecer contenidos Web dinámicos. De estos tres lenguajes de secuencia de comandos, sólo PHP es gratuito. Si tenemos en cuenta que todas las reglas generales son falsas, ASP se suele implementar en un entorno de desarrollo de Microsoft. ColdFusion se utiliza habitualmente en combinación con otras herramientas de programación de Macromedia, como Dreamweaver y Flash, y PHP suele ser la opción de los rebeldes y los independientes.

Por otra parte, líderes del mercado como Yahoo se han decantado por PHP (www.theopenenterprises.com/story/TOE20021031S0001), lo que sugiere que se trata de un lenguaje robusto y escalable, y que su carácter gratuito lo hace muy atractivo tanto para grandes como para pequeñas empresas. El cambio de Yahoo a PHP y a otras herramientas de código abierto demuestra también la futilidad de intentar aplicar reglas generales a algo tan vasto y tan volátil como es la programación Web.

Cada uno de estos tres lenguajes de secuencia de comandos se ha utilizado en sitios de pequeño y gran tamaño. Cada uno tiene sus ventajas y su legión de fanáticos devotos. Las comparaciones y las críticas de los tres lenguajes superan las pretensiones de este libro. No obstante conviene mencionar que los lenguajes de secuencia de comandos generan extensos URL con multitud de símbolos &, lo que no es aconsejable para HTML y XHTML. En HTML y XHTML, el símbolo & se utiliza para denotar entidades, como `&#8217;`, que es el símbolo Unicode para el apóstrofe tipográfico. En concreto, ColdFusion parece incumplir este estándar. Se puede solucionar por medio de una función de ColdFusion denominada `URLEncodedFormat()`. ASP cuenta con una función similar, `HTMLEncode`. En ambos casos, los programadores pueden (y deben) evitar el problema y pasar sus URL a través de la función antes de representarlos.

Las Páginas de Java Server (JSP), otra tecnología dinámica, se suelen emplear en sistemas empresariales de gran tamaño y escapa a los objetivos de este libro.

La vergüenza secreta del marcado incorrecto

Cuanto mayor éxito se haya conseguido en el campo del diseño Web y mayor tiempo se haya pasado en el mismo, menos probable será que nos demos cuenta del coste oculto que supone el marcado incorrecto. En la primera década del sector, el diseño Web era como dar de comer a un grupo de melindrosos niños. Para crear sitios que funcionaran, tuvimos que aprender a acomodar los requisitos dietéticos de cada navegador. Los navegadores actuales se alimentan de lo mismo, pero muchos profesionales no lo saben y siguen añadiendo golosinas al suflé.

Los alimentos incorrectos afectan a las arterias, a los dientes y disminuyen la energía de los que los consumen. Un marcado erróneo es igual de dañino para las necesidades a corto plazo de sus usuarios y para la salud a largo plazo de los contenidos. Pero hasta hace poco, este hecho se ocultaba debido a la tolerancia al marcado basura que mostraban los navegadores más conocidos, como comentamos en un capítulo anterior.

En este capítulo, y en los siguientes, investigaremos la olvidada naturaleza del marcado limpio y semántico, y aprenderemos a pensar de forma estructural en lugar de considerar al marcado Web como una herramienta de diseño de segunda. Al mismo tiempo, examinaremos XHTML, el lenguaje estándar de marcado para páginas Web, describiremos sus objetivos y sus ventajas, y desarrollaremos una estrategia para convertir HTML en XHTML.

Por una extraña coincidencia, la creación correcta de XHTML aboga por un marcado estructural y desaconseja los errores de presentación. En XHTML 1.0 Transitional, estos errores han desaparecido, lo que significa que puede utilizarlos si los necesita pero es aconsejable conseguir los mismos efectos de diseño de otra forma (por ejemplo, con ayuda de CSS). En XHTML 1.0 y 1.1, se prohíben estas técnicas de presentación: si las utiliza, su página no pasará el servicio de validación de marcado del W3C, conocido familiarmente como Validator (véase figura 5.3). (Si no está familiarizado con esta herramienta, pronto lo estará, cuando empiece a diseñar y a crear con estándares.)

VALIDACIÓN

El servicio de validación del W3C (http://validator.w3.org/) puede probar páginas Web creadas con HTML 4.01, XHTML 1.0 y XHTML 1.1 para garantizar que cumplen con las especificaciones. Su servicio de validación CSS (http://jigsaw.w3.org/css-validator/) desempeña la misma función con las hojas de estilo. El Web Design Group de `htmlhelp.com` cuenta con un servicio de validación de marcado igualmente fiable (http://www.htmlhelp.com/tools/validator/). Estos tres servicios se ofrecen gratuitamente. A lo largo del libro nos ocuparemos de éstas y otras herramientas esenciales (y gratuitas).

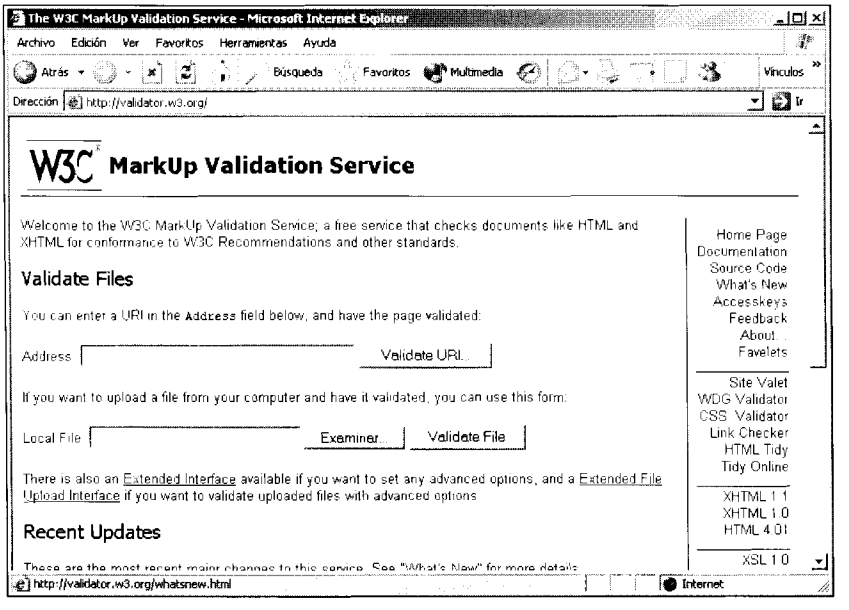


Figura 5.3.

El servicio gratuito de validación en línea del W3C (http://validator.w3.org/) lo utilizan miles de diseñadores y programadores para garantizar que sus sitios cumplen con los estándares Web. ¿Hemos dicho que es gratis?

Siempre que seleccione XHTML Strict o Transitional, descubrirá que todo lo que sabe es incorrecto. Los saltos de línea (<br>) que ha desperdigado como copos de nieve para conseguir una lista; los encabezados que ha implementado para forzar la representación en lugar de aplicar un tamaño jerárquico dentro de un contorno; las imágenes GIF de píxeles transparentes que ha utilizado para crear espacios en blanco. Comprobará que tiene que solucionar estos aspectos y muchos otros.

En lugar de elementos de presentación, tendrá que empezar a pensar de forma estructural. Deje que el marcado sea marcado. Incluso en diseños de transición que utilicen algunas tablas de presentación y otros elementos en desuso, aprenderá a trabajar más con CSS, a desterrar colores de celda de tablas complejos y redundantes, y atributos de alineación de su código XHTML, y reemplazarlos con un par de normas en una hoja de estilo global. Al aprender el nuevo lenguaje de marcado Web, también puede desprenderse de años de hábitos incorrectos. Empecemos pues.

¿Una reformulación de qué?

Según el W3C "XHTML (http://www.w3.org/TR/xhtml1/) es una reformulación de HTML en XML". En un lenguaje más sencillo pero menos preciso, XHTML es un lenguaje de marcado basado en XML con un aspecto operativo y visual similar al de HTML, con pequeñas pero significativas diferencias. Para los navegadores Web y otros agentes, XHTML funciona igual que HTML; aunque algunos de los nuevos y sofisticados navegadores pueden tratarlo de forma diferente, como veremos en un capítulo posterior. Para diseñadores y programadores, el hecho de escribir XHTML es igual que el de escribir HTML, pero con normas propias más estrictas y algún elemento nuevo, como veremos a continuación.

En un capítulo anterior, describimos XML, o Lenguaje de marcado extensible (http://www.w3.org/XML/) como un superlenguaje de marcado con el que los programadores pueden desarrollar otros lenguajes de marcado personalizados. XHTML (Lenguaje de marcado de hipertexto extensible) es uno de ellos. XHTML 1.0 es la primera versión con mayor compatibilidad inversa de XHTML, por lo que es la más fácil de aprender y la que menos problemas genera en navegadores y otros agentes.

Existen numerosas aplicaciones y protocolos basados en XML, pero su popularidad se debe parcialmente a su capacidad para intercambiar y transformar datos sin apenas esfuerzo y sin problemas de compatibilidad, una virtud que comparte con XHTML. Entre estos protocolos encontramos Gráficos vectoriales escalables (http://www.w3.org/TR/SVG/), Lenguaje de integración de multimedia sincronizado (http://www.w3.org/TR/REC-smil/), Protocolo de acceso a objetos simple (http://www.w3.org/TR/SOAP/), Marco de descripción de recursos (http://www.w3.org/RDF/) y Plataforma de preferencias de privacidad (http://www.w3.org/TR/P3P/).

Cada uno de estos protocolos (y muchos otros) desempeña una importante función en la Web, pero ninguno es tan importante para

los diseñadores y programadores como XHTML, y ninguno resulta tan sencillo de aprender.

Se preguntará por qué reformular HTML en XML. Por un lado, XML es coherente donde HTML no lo es. En XML, si se abre una etiqueta, es necesaria volverla a cerrar. En HTML, algunas etiquetas se cierran siempre, otras nunca se cierran y otras se pueden cerrar o no según decida el programador. Esta falta de coherencia puede causar problemas prácticos. Por ejemplo, algunos navegadores se niegan a mostrar una página HTML que deje celdas abiertas aunque según HTML sea correcto dejarlas así. XHTML le obliga a cerrar todos los elementos, lo que evita problemas en el navegador, ahorra horas de pruebas y depuración, e impide que malgastemos neuronas intentando recordar qué etiquetas hay que cerrar y cuáles no.

Lo que es más importante, los lenguajes y las herramientas basadas en XML son la joya del presente de la Web y la clave del futuro. Si crea su marcado en un lenguaje basado en XML, su sitio funcionará mejor con otros lenguajes, aplicaciones y protocolos basados en XML.

Si XML es tan importante, se preguntará por qué crear un lenguaje de marcado basado en XML que funcione como HTML. XML es potente y persuasivo, pero no se pueden utilizar datos XML puros en la mayoría de los navegadores Web y esperar que los utilicen de forma inteligente, como por ejemplo que los muestren en una página Web con un formato agradable (véase figura 5.4). En definitiva, XHTML es una tecnología puente, que combina la potencia de XML (aproximadamente) con la simplicidad de HTML (básicamente).

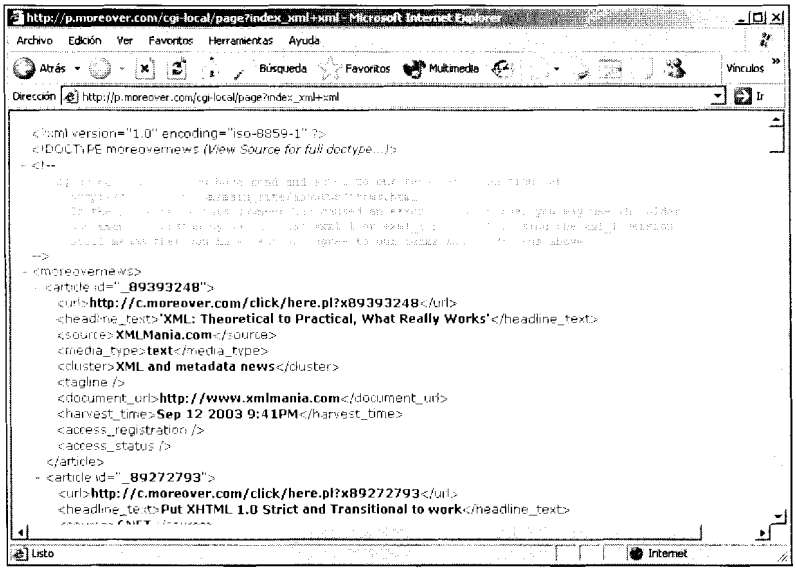


Figura 5.4.

Un ejemplo de documento XML (http://p.moreover.com/cgi-local/page?index_xml+xml) visto en un navegador moderno. Algunos navegadores representan XML como texto y otros como texto más etiquetas. Ninguno de estos enfoques es especialmente útil. Por el contrario, prácticamente todos los navegadores y dispositivos saben qué hacer con XHTML, lo que lo convierte en algo útil y potente.

Resumen ejecutivo

En definitiva, XHTML es XML que funciona como HTML, en navegadores nuevos y antiguos, y que funciona como se espera en la mayoría de los dispositivos de Internet, desde Palm hasta teléfonos Web, pasando por lectores de pantalla, lo que le convierte en una solución portátil, práctica y económica.

Resulta muy sencillo aprender XHTML y utilizarlo como un HTML más sencillo para principiantes que no han adquirido malos hábitos y posiblemente más complicado para aquéllos que se apuntaron al carro del diseño y la programación Web en los años 90.

XHTML es el estándar de marcado actual (que ha reemplazado a HTML 4) y se ha diseñado para generar un estructura lógica y rigurosa de contenido Web, que funciona correctamente con otros estándares Web, como CSS y el DOM, y que se puede combinar con lenguajes, aplicaciones y protocolos basados en XML presentes y futuros. Enumeraremos todos los beneficios de XHTML después del siguiente inciso, breve pero necesario.

¿Qué XHTML es el adecuado?

En este capítulo, y a lo largo del libro, nos centraremos en XHTML 1.0 y haremos hincapié en XHTML 1.0 Transitional, que es la variedad más permisiva de XHTML, la más compatible con los métodos de programación existentes, y la que resulta más sencilla de aprender y de actualizar.

La mayoría de los fanáticos de los estándares prefiere XHTML 1.1 Strict, opinión muy aceptable, aunque es menos compatible con el pasado y cuenta con un tipo MIME que provoca fallos de comportamiento en los navegadores más conocidos. Además, la conversión de páginas Web antiguas a XHTML 1.1 Strict lleva más tiempo que la conversión a XHTML 1.0 Transitional. Para muchos de los lectores de este libro, XHTML 1.0 Transitional probablemente sea la mejor opción para los próximos años.

Al cierre de la edición de este libro, un borrador de la especificación XHTML 2.0 se había presentado a la comunidad de programadores para su comentario. De acuerdo a su configuración actual, esta nueva especificación se acerca a un ideal semántico y se aleja de los métodos de programación Web existentes. Por diseño, el borrador XHTML 2.0 actual no es compatible inversamente con HTML o XHTML 1.0. Abandona las convenciones habituales como el elemento `IMG` (en su lugar se utiliza `OBJECT`), la etiqueta `<BR>` (sustituida por un nuevo elemento `LINE`) y el vínculo ancla (en cuyo lugar se presenta ahora una tecnología denominada `HLINK`). Estos aspectos pueden cambiar cuando este libro llegue a sus manos, o puede que se concreten de forma real.

Algunos programadores han acogido la especificación XHTML 2 propuesta con poco entusiasmo, mientras que otros lo han hecho con cierta mofa. Incluso otros la han adoptado con cierto sigilo, esperando a ver que pasa. Y muchos profesionales Web siguen trabajando como esclavos sin saber nada de esto y preguntándose para qué sirven las opciones de accesibilidad de Dreamweaver.

Falta por ver qué parte de la especificación propuesta se conservará en la recomendación final. También falta por ver si los programadores y los creadores de navegadores se decantarán por XHTML 2 o lo ignorarán. Como la especificación propuesta está lejos de su versión final y no es admitida por ningún navegador todavía, su futura existencia es interesante pero no relevante para este libro ni para nuestros objetivos, razón por la que recomendamos XHTML 1.0.

Por último, si el pensamiento de que la especificación XHTML 2.0 propuesta no es compatible de forma inversa le hace preguntarse si XHTML es compatible de forma directa, debería saber que ningún navegador ni dispositivo tiene pensado abandonar su compatibilidad con XHTML 1. A este respecto, ningún fabricante de navegadores tiene pensado dejar de admitir HTML 4, a pesar de nuestras reticencias sobre HTML incorrecto, que presentamos en un capítulo anterior. Los sitios correctamente creados con la especificación HTML 4.01 seguirán funcionando durante años. Lo mismo sucede con los sitios creados correctamente en XHTML 1. La elección entre estas dos especificaciones (HTML y XHTML) se basa en diez puntos clave que resumimos a continuación.

Diez razones para realizar la conversión a XHTML

1. XHTML es el estándar de marcado actual, que ha sustituido a HTML 4.

2. XHTML se ha diseñado para funcionar correctamente con otros lenguajes de marcado, aplicaciones y protocolos basados en XML, ventaja de la que carece HTML.

3. XHTML es más coherente que HTML, por lo que es menos probable que provoque problemas de funcionamiento y representación.

4. XHTML 1.0 es un puente a futuras versiones de XHTML. Si el borrador de la especificación XHTML 2 alcanza el estado de recomendación final, resultará más sencillo adaptarse a la misma (si así lo decide) desde XHTML 1.0 que desde HTML.

5. Los navegadores antiguos se sienten tan cómodos con XHTML como con HTML. A este respecto, XHTML no presenta ninguna ventaja especial pero tampoco ningún inconveniente.

6. A los nuevos navegadores les encanta XHTML (en especial XHTML 1.0) y muchos le conceden un tratamiento especial que no se ofrece a páginas creadas con HTML 4. En muchos casos, esto hace que XHTML sea más predecible que XML.

7. XHTML funciona con la misma corrección en dispositivos inalámbricos, lectores de pantalla y otros agentes de usuario especializados que en navegadores de escritorio tradicionales, y en muchos casos elimina la necesidad de crear versiones de marcado inalámbrico especializadas y permite que los sitios lleguen a un mayor número de usuarios con menos trabajo y a menor precio. No podemos garantizar la relación causa efecto, pero muchos sitios HTML se ven

cargados de versiones inalámbricas, versiones sólo para texto y páginas especiales para impresión, mientras que la mayoría de los sitios XHTML no sufren estos estorbos: un documento vale para todo. (En la mayoría de los casos, un documento sirve para todo si cuenta con el estilo correcto para varios medios, función que desempeña CSS.)

8. XHTML forma parte de una familia de estándares Web (en la que se incluye CSS y el DOM del W3C) que le permite controlar el comportamiento y el aspecto de páginas Web entre diferentes plataformas, navegadores y dispositivos.

9. La creación en XHTML puede ayudarle a abandonar el hábito de escribir marcado de presentación lo que, a su vez, le puede ayudar a evitar problemas de accesibilidad e incoherencias de representación entre navegadores de escritorio de diferentes fabricantes. (Si escribe XHTML estructural y coloca todos sus elementos visuales en CSS, donde deberían estar, ya no tendrá que preocuparse por las diferencias entre navegadores de Netscape y de Microsoft, como por ejemplo, celdas de tabla vacías a las que se han aplicado diferentes anchos.)

10. Al crear en XHTML se acostumbrará a probar sus trabajos mediante servicios de validación de marcado, que le ahorrarán horas de pruebas y depuración, y le ayudarán a evitar los principales errores de accesibilidad, como la no inclusión de un atributo alt en todas las etiquetas `<IMG>`. En capítulos posteriores encontrará más información al respecto.

Cinco razones para no cambiar a XHTML

1. Cobra por hora.

2. Disfruta creando diferentes versiones para todas las páginas de todos los navegadores y dispositivos posibles.

3. Hay una voz en su interior que le dice que no lo haga.

4. Piensa dejar el sector de la Web.

5. No conoce las reglas de XHTML.

Afortunadamente, podemos hacer algo con esta última razón, como veremos en el siguiente capítulo.

Capítulo 6
XHTML: la reestructuración de la Web

El título de este capítulo podría haber sido "XHTML: Reglas sencillas, directrices simples". Por un lado, las reglas y directrices analizadas en este capítulo son simples y sencillas. Por otra parte, "simple" y "sencillo" son a los libros de diseño Web como "nuevo" y "gratis" a la publicidad de un supermercado, manidos recursos para atraer la atención pero muy eficaces, que estimulan el interés y hacen que la gente se preste a probar un producto.

Y, en este capítulo, evidentemente queremos estimular el interés y hacer que se pruebe un producto. La razón es que, una vez comprendidas las ideas simples y sencillas de este capítulo, pensará en la forma en que funcionan las páginas Web y comenzará a cambiar la forma de crear sus propias páginas. Y con ello no queremos decir que utilice las etiquetas de este año en lugar de las del año pasado. Queremos que realmente cambie su forma de pensar (y de trabajar).

Por otra parte, también pensamos en un título como "Cómo conseguir la exclusividad con el método único en un cegador destello de iluminación ", pero nos pareció excesivo. Realmente de lo que trata XHTML (y este capítulo) es de la reestructuración, por lo que hemos utilizado el otro título.

En este capítulo analizaremos los conceptos básicos de XHTML y describiremos los mecanismos y las implicaciones del marcado estructural frente al marcado de presentación. Si ha incorporado los estándares Web a su diseño o a su programación, le resultará familiar este material. Pero incluso los más reticentes se sorprenderán cuando descubran las sorpresas que esconde este capítulo. Bueno, puede que no se lleguen a sorprender.

Digamos simplemente que captarán su atención. Con eso nos conformamos.

Conversión a XHTML: reglas sencillas, directrices simples

La conversión de HTML tradicional a XHTML 1.0 Transitional es rápida e indolora, siempre que se cumplan una serie de normas y directrices. Si ha escrito HTML, podrá escribir XHTML. Si nunca ha escrito HTML, no tendrá problemas para hacerlo. Nos ceñiremos a los conceptos básicos (sencillos y simples). Veamos las reglas de XHTML.

El DOCTYPE y el espacio de nombres correctos

Los documentos XHTML se inician con elementos que indican a los navegadores cómo deben interpretarlos y a los servicios de validación cómo probarlos. El primero de éstos es la declaración DOCTYPE (abreviatura de tipo de documento). Este útil elemento indica al servicio de validación qué versión de XHTML o HTML se está utilizando. Por motivos que únicamente conocen los miembros del comité W3C, la palabra DOCTYPE siempre se escribe en mayúsculas.

¿Por qué un DOCTYPE?

XHTML permite a los diseñadores y programadores crear diferentes tipos de documento, cada uno regido por reglas diferentes. Las reglas de cada tipo de documento se definen en las especificaciones XHTML en un texto denominado definición de tipo de documento (DTD). La declaración DOCTYPE informa a los servicios de validación y a los navegadores qué DTD se ha aplicado al crear el marcado. A su vez, esta información indica a dichos servicios de validación y navegadores cómo procesar la página.

Las declaraciones DOCTYPE son un elemento clave de las páginas Web compatibles. El marcado y las CSS no se validan a menos que el código fuente XHTML comience con un DOCTYPE correcto. Además, el DOCTYPE que se seleccione determina la forma en la que se representa un sitio en la mayoría de los navegadores modernos. Puede que los resultados le sorprendan. En capítulos posteriores analizaremos el impacto de DOCTYPE en Internet Explorer y en navegadores basados en Gecko como Netscape, Mozilla y Camino.

XHTML 1 ofrece tres opciones DTD y tres posibles declaraciones DOCTYPE:

- **Transitional:** Una DTD muy cómoda pero ligeramente descuidada, cuyo lema es "vive y deja vivir" (como veremos en un apartado posterior).
- **Strict:** Una DTD severa pero misteriosamente reservada que le obliga a utilizar elementos y atributos de marcado de presentación.
- **Frameset:** La DTD de los 90. Le permite utilizar marcos en su diseño.

¿Qué DOCTYPE es el adecuado?

De las tres variedades que acabamos de presentar, XHTML 1.0 Transitional es el más parecido al HTML que todos conocemos. Y,

como tal, es el único que permite el uso de estructuras de marcado de presentación y de otros elementos y atributos obsoletos.

El atributo `target` del vínculo `HREF` es uno de dichos elementos. Si quiere que las páginas enlazadas se abran en nuevas ventanas, o aunque no lo quiera pero se lo exijan sus clientes, Transitional es la única DTD XHTML que le permite hacerlo con dicho atributo:

```
<p>Visit <a href="http://
www.whatever.org" target=
"_blank">whatever.org</a> in a new
window.</p>

<p>Visit <a href="http://
www.whatever.org/" target=
"bob">whatever.org</a> in a named new
window.</p>
```

Para abrir páginas vinculadas en nuevas ventanas con ayuda de XHTML 1.0 Strict, sería necesario escribir JavaScript y también comprobar que los vínculos funcionan en un entorno incompatible con JavaScript. La necesidad de abrir páginas enlazadas en nuevas ventanas no es lo importante. Lo importante es que con XHTML 1.0 Transitional puede hacerlo y de forma sencilla.

XHTML 1.0 Transitional también tolera el uso de colores de fondo aplicados a celdas de tablas y otras posibilidades que se deberían conseguir realmente por medio de CSS en lugar de con marcado. Si su declaración DOCTYPE indica que ha utilizado XHTML 1.0 Strict pero su página incluye el atributo `bgcolor`, los servicios de validación lo marcarán como error y algunos navegadores no compatibles lo ignorarán (es decir, no mostrarán el color de fondo). Por el contrario, si declara que está utilizando XHTML 1.0 Transitional, `bgcolor` no se marcará co-mo error y los navegadores sí lo mostrarán.

En definitiva, XHTML 1.0 Transitional es la DTD idónea para aquellos diseñadores que realizan la transición hacia los modernos estándares Web. Por algo se denomina Transitional.

Se podría discutir si XHTML 1.0 Strict es la mejor opción para los diseñadores y programadores en proceso de cambio, del mismo modo que se podría discutir que alistarse en los Marines sea la mejor opción para perder esos kilos de más. El salto de HTML 4 a XHTML 1.0 Strict sirve de experiencia para saber que el marcado se debe utilizar para la estructura, no para los efectos visuales, y puede que en su caso sea la opción correcta. Pero, para los objetivos de este capítulo (y para los de la mayoría de los lectores), utilizaremos XHTML 1.0 Transitional. Veamos su declaración DTD:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
```

El DOCTYPE se utiliza en documentos que incluyen un elemento `<frameset>`; de hecho, debe utilizar este DOCTYPE en sus documentos `<frameset>`.

La declaración DOCTYPE debe escribirse en la parte superior de cualquier documento XHTML, por delante de cualquier otro código o marcado. Precede al elemento `<title>`, al elemento `<head>`, a los elementos `meta` y a los enlaces a archivos de hoja de estilo y JavaScript. Evidentemente, aparece antes que los contenidos. En definitiva, la

declaración DOCTYPE precede a todos los elementos.

Los lectores expertos en estándares se preguntarán por qué no hemos mencionado un elemento opcional que aparece antes que la declaración DOCTYPE: el prólogo XML opcional. Lo veremos más adelante.

El espacio de nombres aparece después del DOCTYPE

Inmediatamente después de la declaración DOCTYPE debe incluir una declaración de espacio de nombres XHTML que mejora al antiguo elemento `<html>`:

```
<html xmlns="http://www.w3.org/1999/
xhtml" xml:lang="en" lang="en">
```

En XML, un espacio de nombres es una colección de tipos de elemento y de nombres de atributo asociados a una DTD concreta, y la declaración de espacio de nombres le permite identificar a su espacio de nombres haciendo referencia a su ubicación en línea, en este caso www.w3.org/1999/xhtml. Los dos atributos adicionales, en orden de aparición inverso, indican que su documento se ha escrito en inglés y que la versión de XML utilizada también está escrita en inglés.

Una vez determinadas las declaraciones DOCTYPE y de espacio de nombres, una página XHTML 1.0 Transitional empezaría de esta forma:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/
xhtml" xml:lang="en" lang="en">
```

Nota: Si no le gusta escribir personalmente el código y el marcado que se incluye en el libro, puede entonces utilizar el código que se ofrece en `zeldman.com`, `alistapart.com` o bien `webstandards.org`, y copiarlo y pegarlo como desee.

Declare su tipo de contenido

Para que los navegadores los interpreten correctamente y para que pasen las pruebas de validación de marcado, todos los documentos XHTML deben declarar el tipo de codificación de caracteres utilizado para crearlos, ya sea Unicode, ISO-8859-1 (también conocido como Latin-1) o cualquier otro.

Si no está familiarizado con la codificación de caracteres o nunca ha oído hablar de ISO-8859-1, no se preocupe, ya que lo veremos más adelante. Por el momento, basta con saber que hay tres formas de indicar a los navegadores el tipo de codificación de caracteres utilizado, pero que sólo una de ellas funciona de forma fiable, y no es la que recomienda el W3C.

El prólogo XML (y cómo evitarlo)

Muchas páginas XHTML comienzan por un prólogo XML opcional, también conocido como declaración XML. Cuando se utiliza, este prólogo precede a las declaraciones DOCTYPE y de espacio de nombres que mencionamos anteriormente y su misión es especificar la versión de XML y el tipo de

codificación de caracteres empleado en la página.

El W3C recomienda empezar todos los documentos XML, incluidos los documentos XHTML, con un prólogo XML. Para especificar la codificación ISO-8859-1 (Latin-1) se utilizaría el siguiente prólogo XML:

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
```

Por ahora, nada complicado. La etiqueta indica al navegador que se está utilizando XML versión 1.0 y que la codificación de caracteres es ISO-8859-1. La única diferencia o novedad de esta etiqueta es el signo de interrogación que aparece al principio y al final de la misma.

Desafortunadamente, muchos navegadores, incluso los más conocidos, no pueden procesar su prólogo XML. Después de embeberse con este elemento XML, se tambalean, se desesperan y se hunden, se convierten en la vergüenza de su familia y pierden su lugar en la sociedad.

Realmente, los navegadores salen impunes. Son los visitantes los que sufren el funcionamiento incorrecto de un sitio. En algunos casos, los sitios son completamente invisibles para los visitantes e incluso pueden colapsar el navegador de éstos. En otros, el sitio no se bloquea, pero se muestra de forma incorrecta (como cuando IE6/Windows se encuentra con uno de estos prólogos).

Afortunadamente, existe una solución. En lugar de este problemático prólogo, se puede especificar la codificación de caracteres por medio de un elemento de tipo de contenido (Content-Type) que se debe añadir al elemento `<head>`. Por ejemplo, para especificar la codificación ISO-8859-1 escriba esto:

```
<meta  http-equiv="Content-Type"
content="text/html;
charset=_ISO-8859-1" />
```

El inicio de su documento XHTML tendría esta forma:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/
xhtml">
     <head>
<title>Transitional Industries: Working
for Change</title>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1" />
     </head>
```

Si trabaja en un sitio internacional con multitud de caracteres no ANSI, puede crearlo en Unicode y añadir el siguiente elemento `Content-Type` a su marcado:

```
<meta http-equiv="Content-Type"
content="text/html;charset=UTF-8" />
```

Como mencionamos anteriormente, no dude en utilizar el código de `zeldman.com`, `alistapart.com` o `webstandards.org`, y copiarlo y pegarlo.

Puede que se quiera olvidar de todos los detalles de la descripción anterior. Muchos diseñadores desconocen estos problemas aparte de las etiquetas que deben copiar y pegar en sus plantillas, y llevan una vida feliz y productiva.

Aparte de un tema más que analizaremos brevemente a continuación, ya hemos pasado la peor parte de este capítulo. ¡Enhorabuena! El resto será sencillo.

Etiquetas en minúscula

Al contrario que HTML, XML discrimina entre mayúsculas y minúsculas, por lo que XHTML también lo hace. Todos los nombres de atributos y elementos XHTML deben escribirse en minúscula o el documento no se podrá validar. (La validación garantiza que una página no tiene errores. Si se ha olvidado de los servicios de validación gratuitos que ofrecen el W3C y el Web Design Group, puede consultar un capítulo anterior.)

Analicemos un elemento HTML típico:

```
<TITLE>Transitional Industries: Our
Privacy Policy</TITLE>
```

Comprobará que se trata de un elemento `TITLE` y comprobará que se trata de la página Privacy Policy (Directiva de privacidad) que nadie lee, aparte de los del departamento jurídico. La traducción de este elemento a XHTML es tan sencilla como cambiar de mayúsculas a minúsculas:

```
<title>Transitional Industries: Our
Privacy Policy</title>
```

Del mismo modo, `<P>` se convierte en `<p>`, `<BODY>` en `<body>` y así sucesivamente.

Evidentemente, si el HTML original utiliza nombres de elementos y atributos en minúscula, no es necesario cambiarlos. Pero prácticamente todos hemos aprendido a escribir los nombres de atributos y elementos HTML en mayúscula, por lo que tendremos que cambiarlos a minúscula cuando cambiemos a XHTML.

Editores HTML muy conocidos como BareBones, BBEdit, Optima-Systems PageSpinner y Allaire HomeSite le permiten convertir automáticamente nombres de etiquetas y atributos a minúsculas, algo que también puede hacer con la herramienta HTML gratuita Tidy.

No se preocupe de los valores de los atributos ni del contenido

En el ejemplo anterior, únicamente se ha convertido a minúsculas el nombre del elemento (`title`). El texto restante se ha conservado como estaba, con las mayúsculas iniciales. Es más, el contenido del elemento de título se podría haber dejado en mayúsculas (TRANSITIONAL INDUSTRIES: OUR PRIVACY POLICY) y seguirá siendo XHTML válido, aunque resulte una molestia para la vista.

Los nombres de elementos y atributos deben escribirse en minúsculas pero los valores de atributos y el contenido no. A continuación le ofrecemos ejemplos de XHTML perfectamente válido:

```
<img src="/images/whopper.jpg" alt="Big
John catches a whopper." />

<img src="/images/WHOPPER.JPG" alt="Big
John catches a whopper." />

<img src="/images/whopper.jpg" alt="Big
John catches a Whopper and fries." />
```

Debe saber que, en función del software de su servidor, el nombre de archivo del atributo `src` puede discriminar entre mayúsculas y minúsculas, pero a XHTML no le importa. Por otra parte, los valores de ID y de clases, sí discriminan entre mayúsculas y minúsculas.

Preste especial atención a nombres de atributo que mezclen mayúsculas y minúsculas. Si utiliza una herramienta WYSIWYG como

Dreamweaver de Macromedia o un editor de imágenes como ImageReady de Adobe para generar imágenes cambiantes de JavaScript, tendrá que cambiar `onMouseOver` por `onmouseover`. En serio.

Esto puede causar problemas:

```
onMouseOver="changeImages
```

En cambio esto es perfectamente válido:

```
onmouseover="changeImages
```

La hora de Tidy

Sin duda, la forma más sencilla de crear páginas XHTML válidas es escribirlas desde cero. Pero gran parte del diseño Web es en realidad un rediseño, y a menudo tendrá que actualizar páginas existentes. Los proyectos de cambio de diseño son la oportunidad perfecta para migrar a XHTML y no es necesario hacerlo a mano. La herramienta HTML gratuita Tidy (véase figura 6.1), le permite convertir de forma rápida HTML a XHTML válido.

Tidy es una creación del fanático de los estándares Dave Raggett y en la actualidad se mantiene como programa de código abierto por Source Forge (http://tidy.sourceforge.net/), aunque también existen versiones personalizadas. Por ejemplo, Terry Teague (http://www.geocities.com/terry_teague/ tidy.html) ha desarrollado la versión para Mac OS, véase la figura 6.1.

Existen versiones en línea de Tidy así como binarios que se pueden descargar para Windows, UNIX, diferentes distribuciones de Linux, Mac (OS 9 y OS X9) y otras plataformas. Algunas versiones funcionan como complementos para mejorar las prestaciones de programas Web existentes. Por ejemplo, BBTidy es un complemento para el editor BBEdit (X)HTML de BareBones Software.

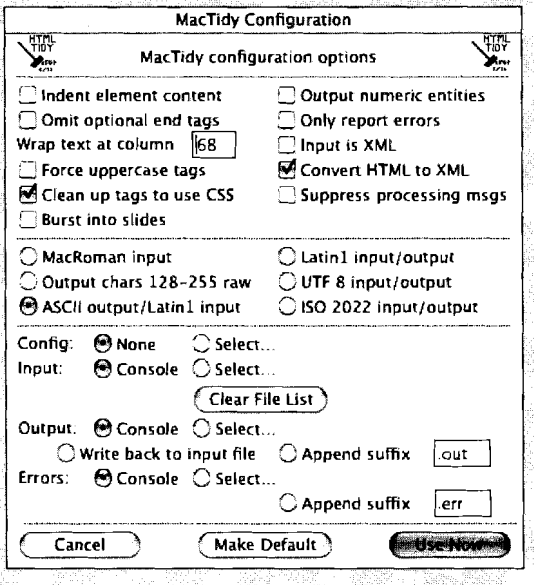


Figura 6.1.

Su aspecto no es muy agradable, pero su precio sí. La herramienta HTML Tidy (http://tidy.sourceforge.net/) es gratuita y le permite convertir páginas de HTML a XHTML. Fíjese en la opción Convert HTML to XHTML *(Convertir HTML en XHTML). Así de sencillo. Existen versiones de Tidy prácticamente para todos los sistemas operativos (en este caso, la imagen se corresponde a la versión para Mac OS X).*

Cada versión ofrece opciones diferentes y, por ello, incluye documentación diferente. El aspecto de Tidy puede parecer simple, pero es una potente herramienta, y su manual puede evitarle mucho sufrimiento.

Todos los valores de atributo entre comillas

En HTML, no era necesario incluir los valores de los atributos entre comillas, pero en XHTML sí (`height="55"`, no `height=55`). Es prácticamente todo lo que tenemos que decir al respecto. La la la. Qué buen día hace...

Bueno, hay algo más que decir. Imagine que en el valor del atributo se incluye material entre comillas. Por ejemplo, el valor del atributo `alt` tiene que ser "The Happy Town Reader's Theater Presents "A Christmas Carol."". Se podría hacer de esta forma:

```
<img src="/images/carol.jpg" alt="The
Happy Town Reader's Theater presents 'A
Christmas Carol.'" />
```

Si prefiere usar las secuencias de caracteres de escape para aplicar apóstrofes y comillas simples y dobles de forma tipográficamente correcta, podría utilizar algo como:

```
<img src="/images/carol.jpg" alt="The
Happy Town Reader&#8217;s Theater
presents &#8216;A Christmas
Carol.&#8217;" />
```

Al encerrar los atributos entre comillas, debe separarlos con espacios en blanco. Este ejemplo es erróneo:

```
<hr width="75%"size="7" />
```

Nota: El validador del W3C tiene que procesar tanto HTML como XHTML y su analizador no detecta este error. Cualquier analizador diseñado para trabajar con XML lo puede detectar.

Si por alguna extraña razón tiene que utilizar comillas en un valor de atributo, utilice `&quot;`, como se indica a continuación:

```
<img src="/images/hello.jpg" alt="Mrs.
O'Hara says, &amp;quot;Hello&amp;quot; to us." />
```

Puede utilizar apóstrofes para encomillar un atributo; si necesita un apóstrofe incrustado, utilice `&apos;`:

```
<img src="/images/hello.jpg" alt='Mrs.
O&amp;apos;Hara says, "Hello" to us.' />
```

En HTML, el uso de comillas en los valores de atributo era opcional pero muchos lo aplicábamos, por lo que la conversión a XHTML no suele suponer mucho trabajo. En un intento de recortar su ancho de banda, muchos sitios comerciales evitaban el uso de comillas en los valores de atributo. Cuando pasen a XHTML, estos sitios tendrán que empezar a aplicar las comillas.

HTML Tidy puede añadir comillas a todos los valores de atributo de forma automática. De hecho, puede realizar automáticamente todas las operaciones de conversión mencionadas en este capítulo.

Todos los atributos requieren valores

Todos los atributos deben tener valores. Por ello, los atributos del siguiente fragmento HTML

```
<td nowrap>
<input type="checkbox" name="shirt"
value="medium" checked>
<hr noshade>
```

deben contar con valores. El valor debe ser idéntico al nombre del atributo.

```
<td nowrap="nowrap">
<hr noshade="noshade" />
<input type="checkbox" name="shirt"
value="medium" checked="checked" />
```

Lo sabemos. Parece raro, se hace raro y cuesta acostumbrarse a ello.

Cierre todas las etiquetas

En HTML, se podían abrir muchas etiquetas, como <p> y <li>, sin necesidad de cerrarlas. Este fragmento es perfectamente aceptable en HTML pero inválido en XHTML:

```
<p>Este HTML es aceptable pero sería
inválido en XHTML.
<p>Me olvidé de cerrar las etiquetas de
párrafo
<p>Pero a HTML no le importa. ¿Por qué no
cambiarán estas reglas?
```

En XHTML, toda etiqueta que se abra debe cerrarse:

```
<p>Este HTML es aceptable y también es
XHTML válido.</p>
<p>Cierro las etiquetas después de
abrirlas.</p>
<p>Soy especial y me siento bien al
respecto.</p>
```

Esta regla tiene más sentido que el enfoque confuso e incoherente de HTML, y puede ayudarle a evitar problemas que nadie necesita. Por ejemplo, si no cierra sus etiquetas de párrafo, puede que en algunos navegadores se produzcan problemas al representar las CSS. XHTML le obliga a cerrar las etiquetas y, al hacerlo, garantiza que las páginas funcionen como deben.

Cierre también las etiquetas vacías

En XHTML, incluso las etiquetas vacías como por ejemplo <br> e <img> deben cerrarse por medio de una barra /> al final de las mismas:

```
<br />
<img src="zeldman.gif" />
```

Fíjese en la barra /> al final de la etiqueta de salto XHTML. Fíjese también en la barra /> al final de la etiqueta de imagen. En ambos casos, antes de la barra se añade un espacio en blanco para evitar la confusión en navegadores desarrollados antes del estándar XHTML. No es ciencia ficción.

Tampoco exige mucho trabajo. Las últimas versiones de BBEdit, PageSpinner y HomeSite añaden de forma automática el espacio y la barra a las etiquetas vacías si se indica que se está trabajando con XHTML (véase figura 6.2). Lo mismo sucede con herramientas de edición Web como Dreamweaver y GoLive.

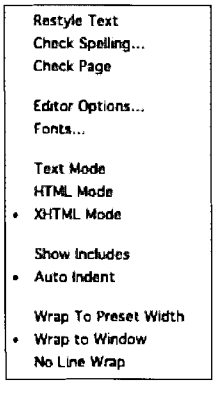


Figura 6.2.

El editor PageSpinner de Optima System se puede utilizar en modo XHTML por medio del menú desplegable. Convierte a minúsculas los nombres de elementos y valores de atributos, y cierra las etiquetas vacías con un espacio y una barra (www.optima-system.com/pagespinner/).

Naturalmente, para que siga siendo válido y accesible, el elemento de imagen del segundo ejemplo debe incluir también un elemento `alt` y tampoco vendría mal un atributo `title`:

```
<img src="zeldman.gif" alt="Jeffrey
Zeldman, autor de Diseño con estándares
Web." title="Jeffrey Zeldman, elegante
diseñador Web y autor multimillonario de
Diseño con estándares Web, actualmente en
su edición 400." />
```

Esto sí es un buen ejemplo de XHTML.

No incluya guiones dobles en los comentarios

Los guiones dobles sólo deben utilizarse al principio y al final de un comentario XHTML, lo que significa que lo siguiente ya no es válido:

```
<!--Inválido -- como el siguiente
separador clásico. -->
<!------------------------------------>
```

Puede reemplazar los guiones interiores por signos de igualdad (=) o añadir espacios entre los mismos:

```
<!-- Válido - - como el siguiente
separador -->
<!--==================================-->
```

Codifique todos los caracteres < y &

Todos los signos menor que (<) que no formen parte de una etiqueta deben codificarse como `<` y todos los símbolos & que no formen parte de una entidad deben codificarse como `&`. De esta forma

```
<p>She & he say that x < y when z = 3.</p>
```

se marcaría como

```
<p>She &amp; he say that x &lt; y when
z = 3.</p>
```

El servicio de validación del W3C mostrará mensajes de advertencia en el marcado sin codificar pero un analizador XML generará un error grave.

Nota: Es recomendable codificar el signo > como `>`. Aunque nunca tenga que codificarlo (excepto en alguna circunstancia realmente extraña), se utiliza por motivos de simetría y el marcado resultará más sencillo de leer si lo emplea.

Repasemos a continuación las reglas de XHTML que hemos aprendido.

Resumen ejecutivo: las reglas de XHTML

- Abra el documento con el DOCTYPE y el espacio de nombres correctos.
- Declare su tipo de contenido por medio de un elemento de contenido `META`.
- Escriba todos los nombres de elementos y atributos en minúscula.
- Todos los valores de atributo deben ir entre comillas.
- Asigne valores a todos los atributos.
- Cierre todas las etiquetas.

- Cierre las etiquetas vacías con un espacio y una barra.
- No añada guiones dobles dentro de un comentario.
- Asegúrese de que el signo menor que (<) y el símbolo & se codifican como `<` y `&`.

En una lista tan breve como ésta, las reglas de XHTML parecen sencillas y simples, y de hecho lo son. Nos queda un aspecto adicional que tratar antes de pasar a la parte divertida.

Codificación de caracteres

Al leer la segunda regla de XHTML en el apartado anterior (Declare su tipo de contenido) puede que se haya preguntado para qué debe declarar su tipo de contenido. Incluso puede que se haya preguntado qué es un tipo de contenido. Las respuestas a estas preguntas las encontrará más adelante. Puede que se pregunte si merece la pena leer esta parte tan tediosa. Evidentemente, la respuesta es afirmativa. Si hemos tenido que escribirla, tiene que leerla. Es lo justo. Además, puede que aprenda algo.

Unicode y otros conjuntos de caracteres

El conjunto de caracteres predeterminado para documentos XML, HTML y HTML 4.0 es Unicode (http://www.w3.org/International/O-unicode.html), un estándar definido por el Unicode Consortium (www.unicode.org). Se trata de un completo conjunto de caracteres que proporciona un número exclusivo a cada carácter, independientemente de la plataforma, programa o lenguaje. Por esta razón, es lo más parecido a un alfabeto universal, aunque se trate de un esquema de asignación numérico, no alfabético.

Aunque Unicode es el conjunto de caracteres predeterminado para los documentos Web, los programadores pueden utilizar el que consideren que mejor se adapta a sus necesidades. Por ejemplo, los sitios Web europeos y americanos suelen utilizar codificación ISO-8859-1 (Latin-1). Se estará preguntando qué significa esta codificación o de dónde proviene. Bueno, para ser honestos, seguro que no se lo está preguntando, pero como necesitamos un punto de transición hemos pensado que lo haría.

ISO 8859

ISO 8859 es una serie de conjuntos de caracteres gráficos multilenguaje estandarizados codificados con 8 bits que se utilizan para escribir en lenguajes alfabéticos y, el primero de ellos, ISO-8859-1 (también denominado Latin-1) se utiliza para asignar caracteres europeos occidentales a Unicode.

Entre los conjuntos de caracteres ISO 8859 se incluyen, entre otros, Latin 2 (Europa del Este), Turco, Griego, Hebreo y Nórdico.

El estándar ISO 8859 se creó a mitad de los 80 por la ECMA, con el respaldo del Organización internacional de estándares (ISO). Ahora ya lo sabe.

Asignación de un conjunto de caracteres a Unicode

Independientemente del conjunto de caracteres que utilice, para asignarlo al estándar Unicode debe declarar su codificación de caracteres, como mencionamos anteriormente en la segunda regla de XHTML (verá que todo esto tiene sentido). Los sitios disponen de tres formas de declarar su codificación de caracteres:

- Un administrador de sitios (el chico de los sistemas) puede definir la codificación a través de los encabezados HTTP devueltos por el servidor Web. El W3C recomienda este enfoque, pero apenas se usa, posiblemente porque los administradores prefieren jugar a juegos en red que andar procesando encabezados HTTP.

- Para documentos XML (incluido XHTML), un diseñador o programador puede utilizar el prólogo XML opcional para especificar la codificación. También es una recomendación del W3C, pero hasta que aumente el número de navegadores que pueda procesarla correctamente, no se la recomendamos.

- En documentos HTML o XHTML, un diseñador o programador puede especificar la codificación por medio del elemento `meta` de tipo de contenido. Al contrario que el método del administrador de servidor (que falla cuando éste se olvida de hacer su trabajo) y del método del prólogo XML (que falla cuando los navegadores no lo pueden procesar), nos podemos fiar del método del tipo de contenido. Es el enfoque que recomendamos anteriormente; ahora sabe por qué.

¡Enhorabuena! Ha leído la parte más pesada del libro o al menos de este capítulo, o probablemente de esta página. Ahora empieza lo bueno. A partir de este punto, comenzaremos a cambiar la forma de diseñar y crear sitios Web.

Una cura estructural recomendada para todos

La programación con XHTML es algo más que cambiar de mayúsculas a minúsculas y añadir barras al final de las etiquetas `<br>`. Si simplemente se tratara de cambiar el formato de las etiquetas, nadie se preocuparía y, en lugar de estándares Web, este libro estaría repleto de deliciosas recetas de tofu. Como el pastel de tofu con grosellas. ¡Uuummmm! Está mucho mejor si utiliza crema de queso en lugar de tofu. Y azúcar, mucho azúcar. Y mantequilla y huevos, no se olvide de los huevos. Pero discrepamos. Para aprovechar las ventajas de XHTML, debe pensar en su marcado como en algo estructural no en términos visuales.

Cómo marcar un documento para que tenga sentido, no estilo

Recuerde que siempre que sea posible, debe utilizar CSS en sus diseños. En el mundo de los estándares Web, el marcado XHTML no tiene que ver con la presentación sino con la estructura interna del documento. Los documentos bien estructurados tienen tanto sentido en un Palm o en un lector de pantalla como en un navegador gráfico de escritorio. Los documentos bien estructurados

también tienen sentido visual en navegadores de escritorio antiguos incompatibles con CSS o en navegadores más modernos en los que se ha desactivado CSS por el motivo que sea.

No todos los sitios pueden desprenderse de sus diseños de tablas HTML. El W3C, los inventores de CSS, no realizaron la conversión hasta diciembre del 2002. Es más, incluso los puristas más acérrimos de los estándares no siempre pueden separar la estructura de la presentación, al menos no en XHTML 1. Pero esta separación entre estructura y presentación (entre datos y diseño) es un ideal al que debemos acercarnos y del cual se pueden aprovechar incluso los diseños híbridos y temporales. A continuación le ofrecemos algunos consejos para que empiece a pensar más estructuralmente.

Color dentro de los contornos

En el colegio, nos obligaban a escribir redacciones en un formato estándar. Después vinieron los diseñadores y, oh, qué bien nos sentimos una vez liberados del yugo de los restrictivos contornos y una vez que nos hemos adentrado en los exclusivos reinos de la expresión personal. (Puede que nuestros folletos y sitios comerciales no fueran tan exclusivos ni tan personales. Pero al menos no teníamos que preocuparnos por los contornos. ¿O sí?)

Realmente, según HTML, debemos siempre estructurar nuestro contenido textual en jerarquías organizadas (contornos). Antes de que los navegadores admitieran CSS no podíamos hacerlo y conseguir diseños comercialmente válidos pero en la actualidad podemos ofrecer una correcta estructura documental subyacente sin tener que pagar multas de diseño.

Al marcar texto para la Web o al convertir documentos de texto existentes a páginas Web, debe pensar en términos de los contornos tradicionales:

```
<h1>Tema</h1>
<p>Introducción</p>
<h2>Punto secundario</h2>
<p>Texto importante</p>
```

Evite el uso de elementos HTML obsoletos como las etiquetas `<font>` o de elementos sin sentido como `<br />` para simular una estructura lógica que no existe. Por ejemplo, no haga esto:

```
<font size="7">Tema</font><br />
Introducción <br /><br />
<font size="6">Punto
secundario</font><br />
Texto importante <br />
```

Utilice elementos por su significado, no por su aspecto

Muchos de nosotros nos hemos habituado a marcar texto como `<h1>` cuando simplemente queremos que tenga un determinado tamaño, o como `<li>` cuando queremos incluir una viñeta por delante del mismo. Como mencionamos en un capítulo anterior, tradicionalmente los navegadores han impuesto atributos de diseño sobre los elementos HTML. Estamos acostumbrados a pensar que `<h1>` significa grande, que `<li>` significa viñeta y que `<blockquote>` equivale a "sangrar este texto". Muchos hemos garabateado nuestra parte de HTML en la que se utilizan elementos estructurales para forzar efectos de presentación.

Del mismo modo, si un diseñador quiere que todos los titulares tengan el mismo tamaño, puede configurarlos todos como <h1> aunque no tenga sentido estructural y es el tipo de cosa que el consultor de accesibilidad Jakob Nielsen denominaría pecado sino estuviera tan ocupado con los colores de los enlaces:

```
<h1>Éste es el título principal, o lo
sería si hubiera organizado mi material
textual en forma de contorno.</h1>

<h1>Éste no es el título principal pero
quería que tuviera el mismo tamaño que el
anterior y no sé cómo utilizar CSS.</h1>

<h1>Esto no es un título ni nada. Pero
realmente quería que el texto de esta
página tuviera el mismo tamaño, algo muy
importante para mi visión creativa. Si
supiera cómo utilizar CSS, podría
conseguir mi diseño sin sacrificar la
estructura del documento.</h1>
```

Debemos empezar a utilizar los elementos por lo que significan, no por su aspecto. De hecho, <h1> puede tener el aspecto que queramos. Por medio de CSS, <h1> puede tener un tamaño reducido y un grosor normal (Roman), el texto <p> puede ser mayor y en negrita, <li> puede utilizarse sin viñetas (o puede emplear una imagen PNG, GIF o JPEG de un gato, de un perro o del logotipo de la empresa) y así sucesivamente.

A partir de hoy, utilizaremos CSS para determinar el aspecto de estos elementos. Incluso podemos modificarlo en función de su posición dentro de una página o un sitio. Ya no será necesario, si es que alguna vez lo fue, utilizar <li> para algo que no sea su función esencial (indicar que un elemento forma parte de una lista de varios elementos).

CSS abstrae totalmente la presentación de la estructura, lo que permite aplicar cualquier estilo a cualquier elemento que desee. En un navegador compatible con CSS, los seis niveles de titulares (h1-h6) pueden tener el mismo aspecto si el diseñador así lo desea:

```
h1, h2, h3, h4, h5, h6      {
     font-family: georgia, palatino,
     "New Century Schoolbook", times,
     serif;
     font-weight: normal;
     font-size: 2em;
     margin-top: 1em;
     margin-bottom: 0;
     }
```

Se preguntará la razón de todo esto. Puede hacerlo para imprimir un determinado aspecto operativo y visual en los navegadores gráficos al tiempo que conserva la estructura de los documentos en navegadores de texto, dispositivos inalámbricos y boletines informativos HTML por correo.

No queremos adelantarnos y explicar técnicas CSS en un capítulo sobre XHTML. Simplemente queremos mostrarle que la estructura de los documentos y la presentación visual son dos aspectos diferentes, y que los elementos estructurales deben utilizarse para transmitir la estructura no para forzar un estilo.

Utilice elementos estructurales en lugar de elementos sin sentido

Como habíamos olvidado, o puede que nunca lo supiéramos, que la función de los elementos HTML y XHTML es comunicar el sentido estructural, muchos de nosotros nos hemos acostumbrado a escribir marcado sin estructura alguna.

Por ejemplo, a menudo se utiliza el siguiente marcado para añadir una lista a una página:

```
elemento <br />
otro elemento <br />
un tercer elemento <br />
```

Pruebe a utilizar una lista, ordenada o sin ordenar:

```
<ul>
<li>elemento</li>
<li>otro elemento</li>
<li>un tercer elemento</li>
</ul>
```

Puede pensar que si utiliza `<li>` aparecerá una viñeta. Consulte el apartado anterior. CSS no asume en absoluto al aspecto que deben tener los elementos. Espera a que se lo indiquemos. La desactivación de viñetas es lo mínimo que CSS puede hacer. Puede hacer que una lista parezca un texto normal de un párrafo o una barra gráfica de navegación, con efectos de imágenes cambiantes y todo.

Por ello, utilice elementos de lista para el marcado de listas. Del mismo modo, utilice `<strong>` en lugar de `<b>`, `<em>` en lugar de `<i>`, etc. De manera predeterminada, la mayoría de los navegadores muestran `<strong>` como `<b>` y `<em>` como `<i>`, con lo que se crea el efecto visual deseado sin afectar a la estructura del documento. Aunque CSS no asume en absoluto la representación de ningún elemento, los navegadores sí lo hacen, y nunca hemos encontrado ningún navegador que mostrara `<strong>` como otra cosa que no fuera negrita (a menos que la CSS de algún diseñador le indique lo contrario). Si le preocupa que en algún navegador no se pueda ver el elemento `<strong>` como negrita, puede escribir una regla CSS como la siguiente:

```
strong     {
     font-weight: bold;
     font-style: normal;
     }
```

Al utilizar marcado estructural como `<strong>` se evita que los usuarios de navegadores de texto y otros dispositivos alternativos descarguen marcado sin sentido a su entorno de navegación (¿Qué significa `<b>` para un lector de Braille?) Además, los elementos orientados a la presentación como `<b>` y `<i>` probablemente desaparezcan en las próximas versiones de XHTML. Jugará sobre seguro si utiliza en su lugar elementos estructurales.

Elementos visuales y estructura

Los estándares Web no sólo se aplican a las tecnologías que utilizamos sino que también afectan a la forma en que las utilizamos. El hecho de escribir marcado XHTML y de utilizar CSS en las labores de diseño no implica necesariamente que un sitio sea más accesible o portátil, o que utilice menos ancho de banda. Se abusa de XHTML y CSS como se ha hecho con cualquier tecnología Web anterior. Con marcado XHTML incorrecto, el usuario malgasta su tiempo igual que lo hacía con HTML incorrecto. Una CSS excesiva no es el sustituto perfecto para el HTML de presentación; simplemente es un mal menor.

Las directrices expuestas en el apartado anterior pueden ayudarle a evitar el uso de

complejas estructuras internas sin significado semántico. Pero se preguntará qué sucede con elementos visuales corporativos, como las barras de navegación entre sitios, que suelen incluir un logotipo. ¿Pueden ser estructurales? ¿Deben serlo?

La respuesta a la primera pregunta es afirmativa. Los elementos visuales, incluyendo las barras de navegación, pueden ser estructurales. Por ejemplo, como hemos mencionado anteriormente, CSS puede mostrar una lista ordenada o sin ordenar como una barra de navegación, con sus botones e imágenes cambiantes.

La respuesta a la segunda pregunta es que no es necesario que los elementos visuales como las barras de navegación sean estructurales en diseños híbridos o de transición. Si dichos diseños evitan los errores y emplean una correcta estructura en las principales zonas de contenido, si su XHTML y CSS son válidos, y se ha primado la accesibilidad, habrá logrado cumplir con los estándares y no tendrá que avergonzarse de nada. (Bueno, puede que se avergüence de algo, pero no de la forma en que ha diseñado su sitio Web.)

En el siguiente capítulo analizaremos las diferencias entre creaciones buenas y malas (y si en las creaciones incorrectas se utiliza XHTML y CSS o no) y aprenderemos a crear marcado correcto y compacto en componentes no estructurales de diseños híbridos y de transición.

Capítulo 7
Páginas más firmes y compactas: estructura y metaestructura en marcado estricto e híbrido

Pase lo que pase, no ignore este capítulo. Le servirá para mejorar sus conocimientos, eliminar partes sobrantes de sus páginas Web y a comprender mejor la diferencia entre marcado y diseño. Los conceptos de este capítulo le resultarán sencillos de seguir y pueden significar una profunda diferencia en el rendimiento de sus sitios y en la forma de diseñarlos, producirlos y actualizarlos.

En este capítulo aprenderá a escribir marcado lógico y compacto que le permitirá reducir su ancho de banda hasta un 50 por ciento, lo que se reflejará en el tiempo de carga de sus sitios y reducirá los problemas y las preocupaciones. Para conseguir estas ventajas eliminaremos elementos de presentación de nuestro XHTML y aprenderemos a evitar muchas de las prácticas habituales menos recomendadas.

Estas prácticas negativas afectan a muchos de los sitios de la Web, especialmente a aquéllos que incorporan algún tipo de CSS en diseños basados esencialmente en tablas. En la mayoría de los casos se puede hacer de forma incorrecta, aunque la experiencia de los diseñadores esté demostrada en muchos otros aspectos. Es un problema de igualdad de oportunidades, ya que puede aparecer tanto en sitios en los que el código se ha creado manualmente como en aquéllos creados por medio de herramientas de edición visuales como Dreamweaver y GoLive.

En este capítulo enumeraremos los errores más comunes para que pueda reconocerlos y protegerse de los mismos, y aprenderemos a solucionarlos. También nos familiarizaremos con el atributo de identificador exclusivo (`id`), la "nota adhesiva" de los estándares

Web, y veremos cómo se puede utilizar para escribir XHTML ultracompacto, ya se trate de diseños híbridos o de transición.

¿Todos los elementos deben ser estructurales?

En un capítulo anterior mencionamos que no es del todo incorrecto que los elementos de navegación de sitios de transición no siempre sean estructurales. Profundizando más al respecto, diremos que incluso en diseños CSS algunos componentes pueden no ser estructurales, al menos no en el sentido "titular, párrafo, lista" que mencionamos en dicho capítulo.

Sin embargo, estos elementos pueden ser meta-estructurales. Es decir, pueden marcarse por medio de elementos estructurales genéricos y atributos estructurales específicos que indiquen la función estructural que desempeñan dentro del diseño del sitio. No se hace por motivos teóricos sino para conseguir beneficios reales como la reducción del ancho de banda y para facilitar el mantenimiento del sitio.

La sección Daily Report de `zeldman.com` (véase figura 7.1) utiliza CSS para el diseño y XHTML para la estructura. Y prácticamente todos los elementos de este marcado son estructurales, desde listas hasta párrafos, pasando por la etiqueta de dirección utilizada para denotar, como habrá supuesto, la dirección.

Los gráficos de navegación que se encuentran en la parte superior de la página no son estructurales, de acuerdo a la definición que utilizamos en un capítulo anterior. Se trata simplemente de imágenes vinculadas, ubicadas por medio de un elemento de nivel de bloque XHTML (`div`) al que el diseñador del sitio ha asignado un nombre por medio del atributo de identificador exclusivo (`id`).

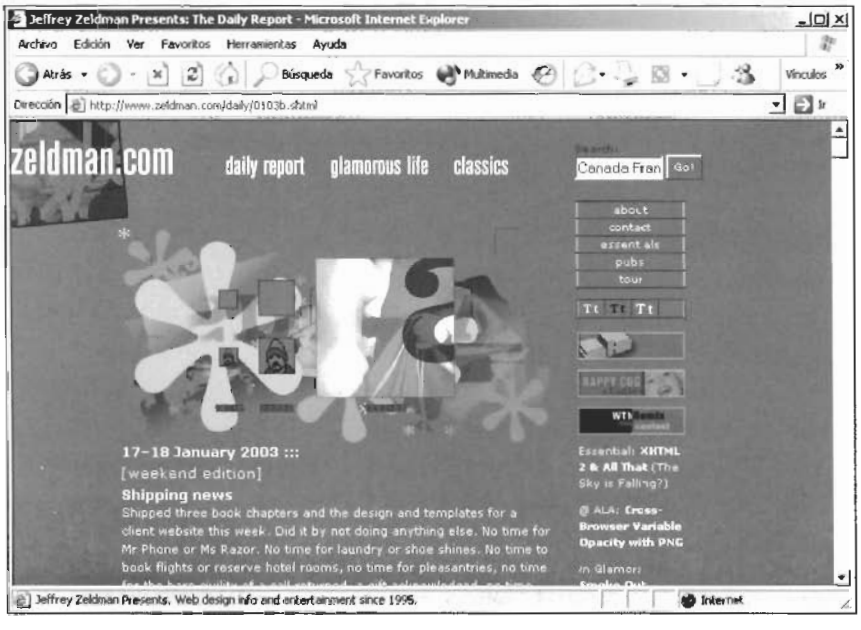


Figura 7.1.

La sección Daily Report de zeldman.com utiliza CSS para el diseño y XHTML para el marcado. Los gráficos de navegación de la parte superior no son estructurales. ¿O sí?

Más adelante definiremos nuestros términos y veremos cómo funcionan estos componentes de forma conjunta para crear una metaestructura y para reducir el peso de la página y controlar el diseño sin tener que recurrir a marcado de presentación.

div, id y otros asistentes

En este capítulo y en los siguientes haremos referencia al elemento `div` y al atributo `id`. Si se utilizan correctamente, `div` es un gran asistente para el marcado estructurado mientras que `id` es un increíble herramienta que le permitirá crear XHTML compacto, aplicar CSS correctamente y añadir sofisticados comportamientos a su sitio por medio del Modelo de objeto de documento (DOM) estándar. El W3C define estos elementos y otros dos importantes componentes HTML/XHTML de esta forma:

> *Los elementos `DIV` y `SPAN`, junto con los atributos `id` y `class`, constituyen un mecanismo genérico para añadir estructura a los documentos. Estos elementos definen el contenido en línea (`SPAN`) o en bloques (`DIV`) pero no realizan imposiciones de presentación al contenido. Por ello, los autores pueden utilizarlos junto a las hojas de estilo para adaptar HTML a sus necesidades y preferencias.*

Un mecanismo genérico para estructuras específicas

Todos los usuarios de HTML están familiarizados con elementos como la etiqueta de párrafo o `h1`, pero no tanto con `div`. La clave para entender el elemento `div` se encuentra en la frase del W3C "un mecanismo genérico para añadir estructura".

> **¿Qué es div?**
>
> Es un momento tan bueno como cualquier otro para explicar que `div` es una abreviación de división. Cuando se agrupan una serie de vínculos, se denomina división de un documento. El contenido sería otra división, la parte jurídica en el pie de la página otra, etc.

En el ejemplo de `zeldman.com`, los principales gráficos de navegación se encierran en un `div` ya que no forman parte de ningún gráfico, lista u otro elemento estructural general. Pero en un sentido más amplio y específico, estas imágenes no desempeñan una función estructural, aparte de la de navegación. Para subrayar su función, al `div` que contiene las imágenes se le asigna el `id` exclusivo `navprin`, nombre que este autor se ha inventado como abreviatura de "navegación principal".

Puede utilizar cualquier nombre. Gladys o naranja son perfectamente válidos con las reglas de XHTML. Pero los nombres metaestructurales (los nombres que explican la función realizada por los elementos que contienen) son más indicados. No se sentiría muy bien si denomina "naranja" a una parte de su sitio y el cliente decide decantarse por el azul. Incluso se sentiría peor si tiene que revisar sus hojas de estilo para tratar de recordar si Gladys era una zona de navegación, una barra lateral, un formulario de búsqueda, etc.

Además, la creación de etiquetas `id` estructurales como "menú", "contenido" o "formulariodebúsqueda" le recuerdan que el marcado

no es diseño y que una página bien estructurada puede tener el aspecto que deseemos. Se trate de diseños CSS puros o diseños híbridos de CSS y tablas, si se acostumbra a asignar nombres metaestructurales a componentes esenciales de la página (como son las zonas de navegación, contenido y búsqueda), acabará por dejar de crear y pensar en marcado de presentación.

id frente a class

El atributo `id` no es nuevo en XHTML; tampoco lo es el atributo `class` ni los elementos `span` y `div`. Todos provienen de HTML. El atributo `id` asigna un nombre exclusivo a un elemento. Cada nombre sólo se puede utilizar en la misma página. (Por ejemplo, si una página contiene un `div` con el `id` "contenido", no puede haber otro `div` u otro elemento con el mismo nombre.) Por el contrario, el atributo `class` se puede utilizar repetidamente en la misma página (por ejemplo, cinco párrafos de una página pueden compartir el nombre de clase "pequeño" o "nota"). El siguiente marcado le ayudará a clarificar la distinción entre `id` y `class`:

```
<div id="searchform">Aquí van los
componentes del formulario.</div>
<div class="blogentry">Aquí va una
entrada de registro Web.</div>
```

En estos ejemplos, `div id="searchform"` se utiliza para incluir en un bloque la zona de la página que contiene el formulario de búsqueda, mientras que `div class= "blogentry"` se puede utilizar para incluir en un bloque todas las entradas en un registro Web (lo que se conoce familiarmente como blog).

Sólo hay un formulario de búsqueda en la página, por lo que se selecciona `id` para esa instancia en concreto. Pero un registro Web puede tener varias entradas, por lo que en ese caso se utiliza el atributo `class`. Si el registro Web puede pasar sin utilizar estos `div` y, por el contrario, puede utilizar estructuras de titular y de párrafo convencionales, mucho mejor. La sección Daily Report de `zeldman.com` se ha denominado registro Web (no por nosotros) pero sus entradas están marcadas con elementos `h3`, `h4` y `p` genéricos.

La teoría del Post-it

Puede que le resulte útil pensar en el atributo `id` como si fuera un Post-it. Utilizamos estas notas adhesivas en el frigorífico para acordarnos de comprar leche. Las ponemos en el teléfono para acordarnos de llamar a un cliente que se retrasa en los pagos. Las ponemos en un montón de facturas para acordarnos de pagarlas antes del día 15 del mes.

El atributo `id` es similar a estas notas adhesivas ya que etiqueta una zona determinada del marcado para recordarnos que dicha zona requiere un tratamiento especial. Para aplicar este tratamiento especial, es necesario escribir una o varias normas en una hoja de estilo o algunas líneas de código en un archivo JavaScript que haga referencia a ese `id` concreto. Por ejemplo, puede que su archivo CSS tenga normas especiales que sólo se apliquen a elementos del `div` con el `id "formulariodebúsqueda"`. O puede que contenga normas que únicamente se apliquen a una tabla cuyo `id` sea `"barra-demenú"`. (Si ha prestado atención, habrá

adivinado que una tabla con un `id "barra-demenú"` es una tabla que actúa como barra de menú. En capítulos posteriores crearemos un sitio que utilice este tipo de tabla y este `id`.)

Cuando se utiliza un valor de un atributo `id` como imán para un determinado conjunto de reglas CSS, se denomina selector CSS. Existen diversas formas de crear selectores, como veremos en capítulos posteriores, pero `id` resulta especialmente útil y versátil.

El poder de id

El atributo `id` es increíblemente potente. Entre otras tareas, puede realizar las que enumeramos a continuación:

- Como selector de hoja de estilo (consulte el apartado anterior), lo que nos permite crear XHTML preciso y reducido.
- Como ancla de destino para vínculos de hipertexto, lo que reemplaza al desfasado atributo de nombre (o coexiste con éste por motivos de compatibilidad inversa).
- Como medio para hacer referencia a un determinado elemento de una secuencia de comandos basada en DOM.
- Como nombre de un objeto declarado.
- Como herramienta para procesamientos generales (en un ejemplo del W3C, "para identificar campos al extraer datos de páginas HTML a una base de datos, traducir documentos HTML a otros formatos, etc.").

Normas de id

Un valor `id` debe empezar por una letra o un guión bajo; no puede empezar por un número. El servicio de validación W3C no detectará este error, pero un analizador XML sí. Al mismo tiempo, si trata de usar un `id` con JavaScript con la forma `documento.nombreid.valor`, debe utilizar como nombre una variable JavaScript válida, es decir, debe empezar con una letra o un guión bajo, seguido por letras, números y guiones bajos. No se permiten espacios en blanco ni guiones. A este respecto, no es aconsejable el uso de guiones bajos en nombres de clases ni `id`, debido a las antiguas limitaciones de CSS (y de algunos navegadores).

Por último, para los lectores más fanáticos, diremos que se puede empezar un nombre de `id` o de clase con un número si éste se escapa, es decir, si se utiliza la correcta secuencia de caracteres de escape Unicode para representar dicho número. Pero nadie lo hace.

Con la potencia llega la responsabilidad

Los elementos y atributos que hemos analizado, a menudo se utilizan incorrectamente, lo que cuadruplica el tamaño de las páginas en lugar de reducirlo y eliminar todos los restos de estructura y no añadir metaestructura a los elementos estructurales genéricos de XHTML. Después de leer este capítulo, podrá reconocer estas formas erróneas y sabrá cómo evitarlas.

Atrévase a hacer menos

Una vez analizados los elementos XHTML de propósito general (en especial `div` e `id`), volvamos de nuevo al ejemplo de `zeldman.com` (véase figura 7.1). Las cuatro imágenes del menú se mantienen con ayuda de un elemento `div` genérico cuya estructura metaestructural exclusiva es `"primenav"`.

```
<div id="primenav">[Aquí van las imágenes
vinculadas.]</div>
```

Lo que separa a este método de las técnicas más familiares es que el elemento de nivel de bloque no contiene instrucciones de presentación: sin ancho, sin alto, sin color de fondo, sin alineación vertical u horizontal. Nuestro humilde elemento de nivel de bloque puede desprenderse de todos los aspectos en los que insistían los componentes de diseño de la vieja escuela.

Se preguntará entonces cómo sabe el navegador dónde añadir las imágenes.

Retomemos brevemente el análisis de los selectores CSS. `primenav` se utiliza como uno de estos selectores. En una de las hojas de estilo del sitio, un regla CSS indica que el elemento `primenav` no tiene margen, es decir, que está rodeado por espacio en blanco. Como `<div id="primenav">` es el primer elemento visible que aparece en el código XHTML, los navegadores compatibles lo sitúan en la esquina superior izquierda de la página.

Las cuatro imágenes que controla `primenav` se muestran en línea, una tras otra. No hace falta una tabla para ubicar estas imágenes unas con respecto a las otras. Tampoco es necesario asignar a cada imagen su propia regla CSS de ubicación exclusiva. El orden en el que se enumeran las imágenes en XHTML es el orden en el que aparecerán en la página.

Gracias a la lógica del orden de los elementos, evitamos la confusión de los diseños de tabla y la complejidad de las hojas de estilo incorrectas (como veremos a continuación), con lo que los usuarios del sitio reciben un diseño fiable que se carga rápidamente ya que no contiene marcado o CSS excesivos.

A continuación le mostramos el marcado de `zeldman.com`, sin los elementos JavaScript, con los nombres de las imágenes y de las rutas, para que resulte más claro:

```
<div id="primenav">
<a href="/"><img src="/i/home.gif"
width="150" height="100" border="0"
alt="Home" /></a>
<a href="/"><img src="/i/dailyreport.gif"
width="140" height="100" border="0"
alt="The Daily Report" /></a>
<a href="/glamorous/"><img src="/i/
glamorouslife.gif" width="110"
height="100" border="0" alt="My
Glamorous Life" /></a>
<a href="/classics/"><img src="/i/
classic.gif" width="100" height="100"
border="0" alt=" Classics,
1995-2002" /></a>
</div>
```

Comprobará que los atributos de anchura y altura de las imágenes, aunque son útiles, no son estrictamente necesarios. Comprobará también que, gracias a CSS, el atributo de borde es totalmente prescindible, aunque lo hayamos incluido en el ejemplo por los navegadores antiguos. Nuestro marcado podría ser tan limpio y sencillo como el siguiente:

```
<div id="primenav">
<a href="/"><img src="/i/home.gif"
alt="Home" /></a>
<a href="/"><img src="/i/dailyreport.gif"
alt="The Daily Report" /></a>
<a href="/glamorous/"><img src="/i/
glamorouslife.gif" alt="My Glamorous
Life" /></a>
<a href="/classics/"><img src="/i/
classic.gif" alt=" Classics,
1995-2002" /></a>
</div>
```

Puede comparar y contrastar la claridad del marcado anterior con la versión típica de diseño de tablas que mostramos ahora:

```
<table border="0" cellpadding="0"
cellspacing="0" width="500">
<tr>
<td valign="top" width="150" height="100"
bgcolor="#339999">
<a href="/"><img src="/i/home.gif"
width="150" height="100" border="0"
alt="Home" /></a>
</td>
<td valign="top" width="140" height="100"
bgcolor="#339999">
<a href="/"><img src="/i/dailyreport.gif"
width="140" height="100" border="0"
alt="The Daily Report" /></a>
</td>
<td valign="top" width="110" height="100"
bgcolor="#339999">
<a href="/glamorous/"><img src="/i/
glamorouslife.gif" width="110"
height="100" border="0" alt="My
Glamorous Life" /></a>
</td>
<td valign="top" width="100" height="100"
bgcolor="#339999">
<a href="/classics/"><img src="/i/
classic.gif" width="100" height="100"
border="0" alt=" Classics,
1995-2002" /></a>
</td>
</tr>
</table>
```

No hay color. Pero el marcado compacto y la metaestructura no se limitan a los diseños CSS. También pueden mejorar y racionalizar diseños basados en tablas.

Diseños híbridos y marcado compacto: ventajas e inconvenientes

El miedo a aprender a diseñar con CSS o la imposibilidad de utilizarlo en un determinado proyecto no son motivos para evitar el uso de los estándares Web. Si combina diseños de tabla racionalizados con CSS básicas y XHTML estructural y accesible, conseguirá un enfoque híbrido y de transición que funcione.

CSS sigue evolucionando, los navegadores siguen aprendiendo a admitir CSS1 y CSS, y algunas ideas de diseño funcionan mejor cuando sus elementos básicos se controlan con sencillas tablas XHTML. Pero el marcado de tablas no es lo mismo que el marcado estúpido, y los diseños híbridos que utilizan determinados componentes no estructurales se diferencian del HTML innecesariamente ostentoso con el que se siguen construyendo muchos de los sitios actuales.

Advertencia: Hemos mencionado que algunos componentes de la interfaz de los sitios híbridos pueden no ser estructurales. No significa que el marcado estructural no tenga su importancia en el resto de los sitios; los párrafos se seguirán marcando como párrafos, las listas como listas, etc. Tampoco es una invitación a producir menús de navegación u otros componentes por medio de los caducos trucos descritos en este libro. Sean estructurales o

no, todos los componentes deben producirse con marcado claro y compacto, y con CSS claras. El marcado limpio es el verdadero objetivo de este capítulo.

Asignación de nombres a elementos incorrectos

En este apartado, veremos algunas formas erróneas de construir sitios y explicaremos por qué estos métodos son contraproducentes. También inventaremos etiquetas para diferentes técnicas especialmente indeseables que se utilizan con demasiada frecuencia. Con la lectura de este apartado no sólo nos desprenderemos de malos hábitos de trabajo, sino que también podremos detectar dichos errores en el trabajo de colegas y distribuidores.

Una vez leído este apartado y tras asimilar los sencillos conceptos del mismo, cuando sus colegas o distribuidores traten de utilizar determinados tipos de marcado sin sentido, podrá hacérselo saber por medio de las descriptivas etiquetas que inventaremos en este capítulo. Sus colegas y distribuidores desarrollarán un nuevo respeto por el marcado, un nuevo respeto por sus opiniones y, sobre todo, un nuevo malestar cuando se encuentre con ellos.

Realmente, con los nombres asignados a las prácticas de creación incorrectas que describiremos a continuación no pretendemos mofarnos de nadie, sino simplemente identificar y eliminar dichas prácticas de nuestro trabajo. Esperemos que también le resulten útiles.

Errores comunes del marcado híbrido

Imagine el diseño de un sitio basado en tablas (véase figura 7.2) con una barra de menú (véase figura 7.3) que cambia de estado (véase figura 7.4) para indicar la posición del visitante cuando cambie de una página a otra. Evidentemente, el logotipo del sitio será un elemento gráfico, con toda seguridad una imagen GIF. Las etiquetas de menú (EVENTS, ABOUT, CONTACT, etc.) también pueden ser imágenes o sencillos vínculos de hipertexto. Si se selecciona el hipertexto, los diseñadores Web de la vieja escuela inundarán el marcado con etiquetas de fuentes para controlar el aspecto del texto (tamaño, tipo, color) y atributos desfasados para controlar la alineación, las propiedades de los bordes y el color de fondo de las celdas de las tablas en las que se incluyen estas etiquetas de menú. Los diseñadores modernos utilizarían CSS, pero los resultados no serán mucho mejores si se utiliza un enfoque incorrecto.

Incluso si se usan imágenes GIF en lugar de hipertexto, el diseñador buscará la forma de distinguir el aspecto de la tabla de menú de otras estructuras de tabla del sitio. Por ejemplo, parece que en la tabla de menú (véanse figuras 7.3 y 7.4) se necesitan bordes negros, mientras que otras estructuras de tabla de la página no utilizan bordes. En el peor de los casos, el diseñador incluiría la tabla de menú dentro de otra tabla para conseguir el efecto de contorno negro. Era la técnica habitual en 1997. Los diseñadores actuales pueden combinar el marcado sobrante con un exceso de CSS para diferenciar la tabla de menú de otras tablas de la página. No es una solución mejor.

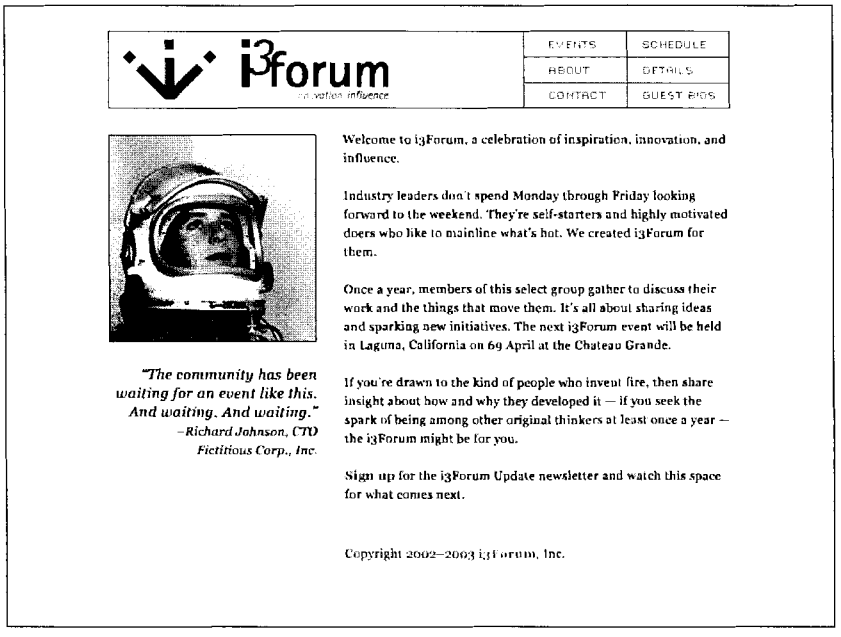


Figura 7.2.

Diseño del sitio de la conferencia i3Forum (que crearemos en capítulos posteriores).

Figura 7.3.

El menú, tal y como aparecerá cuando el visitante llegue a la página de inicio. El logotipo se resalta con un fondo blanco.

Figura 7.4.

Cuando el visitante cambia a la página EVENTS, se resalta la etiqueta EVENTS.

Clasitis: el sarampión del marcado

¿Cómo podemos indicar a las celdas de las tablas de navegación que se muestren de forma diferente al resto de las tablas de la página? ¿O cómo obligamos a los vínculos de estas celdas a que se muestren de forma diferente a otros vínculos? No con etiquetas y atributos de presentación obsoletos como `bgcolor` y `font`, como ya habrá imaginado. No queremos nada como esto:

```
<td align="left" valign="top"
bordercolor="black" bgcolor="white">
<font family="verdana, arial"
size="2"><b>
<a href="/events/">Events</a></b>
</font></td> etc.
```

Pero tampoco queremos aplicar una clase a todos los elementos que requieran un tratamiento especial. Ya hemos visto demasiado de este tipo de técnicas:

```
<td class="whitewithblackborder">
<span class="menuclass"><b>
<a href="/events/">Events</a></b>
</span></td> etc.
```

Incluso hemos visto, con perdón, este tipo de basura:

```
<td class="whitewithblackborder"><span
class="menuclass"><font class=
"menufontclass"><b>
<a href="/events/">Events</a></b>
</font></span></td> etc.
```

Podemos denominar a este estilo de marcado clasitis. En un sitio afectado por clasitis, a todas las etiquetas les salen sus propias clases; es el sarampión del marcado. Oscurece el significado al tiempo que añade un peso innecesario a las páginas.

Esta dolencia data de la época en que aparecieron los navegadores semicompatibles con CSS y se debe a la forma infantil en la que muchos diseñadores entendían el funcionamiento de CSS.

Desafortunadamente, muchos todavía no han superado esa concepción infantil de CSS, bien porque se han decantado por otra tecnología o porque la herramienta de edición visual que utilizan aplica una clase a todas las etiquetas que genera, y han aprendido a utilizar CSS a través del código que generan estas herramientas.

La clasitis es tan nociva como lo ha sido la etiqueta `<font>`; en un marcado correcto apenas se necesita.

LOS EDITORES VISUALES Y LA CLASITIS

Incluso los editores Web visuales más sofisticados tienden a generar clases innecesarias al igual que muchos gérmenes, principalmente porque son editores visuales y no personas. Las personas pueden abstraer lo general de lo específico. Al aplicar un estilo a un párrafo en CSS, sabemos que queremos que todos los párrafos tengan el mismo estilo. Pero un editor visual no puede saber nuestras intenciones. Si, utilizando uno de estos editores creamos cinco párrafos con la fuente Verdana 11, la herramienta asignará una clase a cada uno de los párrafos. Las últimas versiones de Dreamweaver y GoLive son sofisticadas, potentes y compatibles con los estándares. Pero es recomendable editar manualmente los resultados para evitar la clasitis y la divitis.

Divitis

En sus mejores momentos, la clasitis se puede injertar sobre un marcado que debería ser estructural:

```
<p class="noindent">Forma incorrecta de
diseñar páginas Web.</p>
<p class="indentnomargintop">No se
necesitan todas estas clases.</p>
<p class="indentnomargintop">Los
diseñadores con clase evitan este
problema.</p>
<p class="indentnomargintop">Clase
rechazada.</p>
```

Es un ejemplo de lo que una herramienta de edición visual más sofisticada y más compatible con estándares puede generar (como

mencionamos anteriormente). En otros casos, la clasitis se exagera con una condición mucho más seria, lo que hemos denominado divitis:

```
<div class="primarycontent"><div
class="yellowbox"><div class="heading">
<span class="biggertext">Welcome</span>
to the Member page!</div><div class=
"bodytext">Welcome returning members.
</div><div class="warning1">If you are
not a member <a href="/gohere/"
class="warning2">go here</a>!</div>
</div></div>
```

En este caso carecemos de estructura, únicamente hay multitud de bytes de marcado no estructural y, por lo tanto, sobrante. Si visita este sitio con un Palm o un navegador de texto, no sabrá la relación que tienen los diferentes elementos entre sí. Principalmente, porque no están relacionados en absoluto. La divitis sustituye la ingesta de estructura por calorías vacías de elementos sobrantes.

Las clasitis y la divitis son como las notas innecesarias que toca un músico novato, cuando en realidad debería servir de apoyo al cantante o solista principal. Son como adjetivos inútiles en un texto. Son las malas hierbas del jardín del significado.

Si reduce la clasitis de su marcado comprobará que las páginas Web afectadas reducen su tamaño a la mitad. Si evita la divitis, podrá escribir marcado compacto y estructural que funcione tanto en un navegador de texto como en su navegador de escritorio favorito.

Hágalo con rigor para emprender el camino hacia la creación de páginas Web más inteligentes y compatibles con estándares.

Los div son correctos

Puede que algunos piensen que he afirmado que los elementos `div` son incorrectos pero que he utilizado uno en el primer ejemplo de este capítulo (las imágenes de navegación de `zeldman.com`) y, que por ello, les he mentido.

De hecho, nunca he afirmado que los elementos `div` fueran incorrectos, ni tampoco lo ha dicho el W3C, que es el responsable de su creación (si no se lo cree, consulte un apartado anterior). Un `div` es una unidad de marcado muy indicada para crear bloques de estructura en un sitio.

La divitis se manifiesta al utilizar `div` para reemplazar elementos completamente válidos (y más adecuados). Si ha escrito un párrafo, entonces se trata de un párrafo y debe marcarse como tal, no como un `div` de la clase de texto.

El título de nivel superior debe ser `<h1>` no `<div class"headline">`. Seguro que aprecia la diferencia.

¿Por qué un div?

Muchos diseñadores utilizan elementos `div` no estructurales para reemplazar cualquier cosa, desde títulos hasta párrafos. Muchos adquirieron este hábito cuando descubrieron que los navegadores 4.0, en especial los de Netscape, envolvían los espacios en blanco no deseados en elementos estructurales como `<h1>`, pero que no afectaba a los diseños que utilizaban únicamente elementos `div` no estructurales (como vimos en un capítulo anterior).

Para evitar que los detalles de diseño desaparezcan en navegadores 4.0, muchos diseñadores insisten en utilizar elementos `div` no estructurales, con la consecuente exclusión de párrafos, títulos, etc. El precio que se paga es excesivo. Con esta práctica, los sitios se vuelven impenetrables para un número cada vez mayor de usuarios inalámbricos, telefónicos, alternativos y de lectores de pantalla. Y también se duplica o triplica el ancho de banda, independientemente de qué navegador o dispositivo se utilice. No merece la pena.

Otro ejemplo clásico de divitis se produce cuando un diseñador se infecta con el virus "las tablas son nocivas, CSS no" y sustituye 200 toneladas de marcado de tablas por 200 toneladas de elementos `div` anidados. No ha conseguido nada (excepto un sentimiento de autocomplacencia) y puede que incluso haya dificultado la edición del documento.

Enamorados del elemento id

Si el HTML de presentación no es válido, si tampoco lo son la clasitis y la divitis, se preguntará cómo podemos aplicar reglas de diseño diferentes a nuestra zona de navegación en un diseño híbrido que combine tablas y CSS. Podríamos hacerlo con ayuda de un `id`. Asigne una etiqueta metaestructural exclusiva a la tabla en la que se incluya el menú:

```
<table id="menu">
```

Posteriormente, al escribir la hoja de estilo, tendrá que crear un selector de menú y un conjunto asociado de reglas CSS que indiquen a las celdas de la tabla, a las etiquetas de texto y a todos los demás elementos cómo se deben representar exactamente. Si opta por utilizar enlaces de hipertexto para construir el menú, puede hacerlo sin dificultad alguna:

```
<td><a href="/events/">Events</a></td>
```

Se preguntará dónde está el resto. Bien, no hay ningún resto, aparte de las demás celdas de tabla necesarias. Al etiquetar la tabla con el `id "menu"` (o el nombre que seleccione) y al utilizar `"menu"` como selector CSS en la hoja de estilo que escriba posteriormente en el proceso de diseño, se eliminan todos los elementos sobrantes de los que hemos hablado.

No hay clasitis. No hay divitis. No se necesitan las obsoletas etiquetas `<font>`. No es necesario aplicar redundantes atributos de altura, anchura, alineación y color de fondo que consumen ancho de banda en cada instancia del elemento `<td>`. Con una nota adhesiva digital (el `id "menu"`) hemos preparado el marcado compacto y limpio para su procesamiento de presentación en una hoja de estilo independiente.

CSS controla todos los aspectos estéticos de estas celdas, incluyendo el color de fondo (y la imagen de fondo, en caso de que se utilice), el tratamiento de los bordes (si hay), el espacio en blanco, la alineación vertical y horizontal, y los efectos de imágenes cambiantes. Estos últimos efectos suelen incluir cambios en el color del fondo, en el color de los bordes o en ambos. Se admiten en la mayoría de los navegadores actuales y no afectan a los navegadores que no los admiten. Del mismo modo, CSS controla el aspec-

to del vínculo, incluyendo la fuente, el color, el tamaño y otras muchas variables.

¿Qué pasa con las imágenes?

Se preguntará si esta combinación de un `id` exclusivo, XHTML compacto y una hoja de estilo funciona igual en menús basados en tablas que utilizan imágenes como en los que utilizan enlaces de hipertexto. Me alegro que se haga esa pregunta. Funciona igual de bien. Basta con crear la tabla únicamente con los elementos que necesita, hacer lo mismo con las imágenes y asegurarse de que todos los elementos de imagen cuentan con un atributo `alt` que se pueda utilizar.

De hecho, se pueden combinar imágenes GIF transparentes con reglas CSS para crear sutiles efectos de imágenes cambiantes sin necesidad de JavaScript. Pero nos estamos adelantando a los acontecimientos.

Como parece que este capítulo está dedicado a los nombres, hemos denominado a la combinación de XHTML y CSS dirigida por `id`, método de texto limpio, cuando utiliza básicamente elementos de menú de hipertexto y método de imagen limpia cuando utiliza principalmente menús controlados por imágenes. Si este libro formara parte de un siniestro conglomerado global, estos enfoques se registrarían y se patentarían, y la mera alusión a este capítulo le costaría una suculenta cuota de licencia. Pero no somos así. Puede utilizar estos métodos con total libertad y asignarles los nombres que desee.

Conviene mencionar otro aspecto para comprender cómo los diseños de tablas pueden ser más inteligentes y menos nocivos.

Eliminación de celdas de tablas redundantes

Ahora, analizaremos el caso del sitio Web The Marine Center (véase figura 7.5), diseñado y producido en el 2003 para el mayor proveedor mundial de peces y corales raros. Comprobará que el diseño del sitio se divide en diferentes zonas según la función de las mismas. Por ejemplo, en la columna de la derecha se anuncian los artículos actualmente disponibles, invita al visitante a participar en la lista de correo del sitio e incluye funciones de navegación y de búsqueda. El área de contenidos de la izquierda enumera las novedades, ofrece una serie rotativa de imágenes e invita al lector a detenerse en la columna mensual sobre cuentos de peces.

Este tipo de diseños suele incluir cada componente dentro de su propia celda de tabla, de forma similar al sitio The Gilmore que estudiamos en un capítulo anterior. En un diseño típico creado mediante la división de componentes de Photoshop o la ubicación de elementos en un editor visual, la imagen de mayor tamaño se incluiría en su propia celda de tabla, al igual que las fotos de menor tamaño, el encabezado grafico de la sección Fish Tales y así sucesivamente. Las celdas restantes incluyen imágenes GIF transparentes para controlar la cantidad de espacio horizontal y vertical entre las secciones.

Se utilizan, por tanto, docenas de celdas, lo que crea peligrosas dependencias. Por ejemplo, si los especialistas en contenidos del sitio escriben demasiadas líneas de introducción una semana, la imagen del pez de mayor tamaño se desplazaría hacia abajo en la

página y con este desplazamiento pueden aparecer espacios no deseados en la columna de la derecha. El sitio malgastará gran cantidad de ancho de banda, no por contener imágenes, sino por depender de tablas complejas con sus correspondientes atributos de presentación asociados.

Aunque The Marine Center utiliza técnicas de diseño híbridas, evita los problemas asociados habitualmente a las mismas ya que restringe al mínimo el uso de tablas. La cuadrícula de tablas básica crea las columnas de la izquierda y de la derecha. Los elementos se añaden a estas columnas en orden de aparición, y los detalles de espacio en blanco y alineación se controlan exclusivamente por medio de CSS. En un capítulo posterior analizaremos los métodos CSS. Por el momento, limítese a utilizar el menor número de elementos de tablas posible. Al combinar el menor número de tablas con CSS, comprobará que muchas de las técnicas que consideraba fundamentales no lo son.

Métodos desfasados

Por motivos de referencia (y en caso de que siga utilizando alguno de estos métodos), finalizaremos este capítulo con la descripción de algunos de los métodos obsoletos de producción, enumerados según su aparición en la Web.

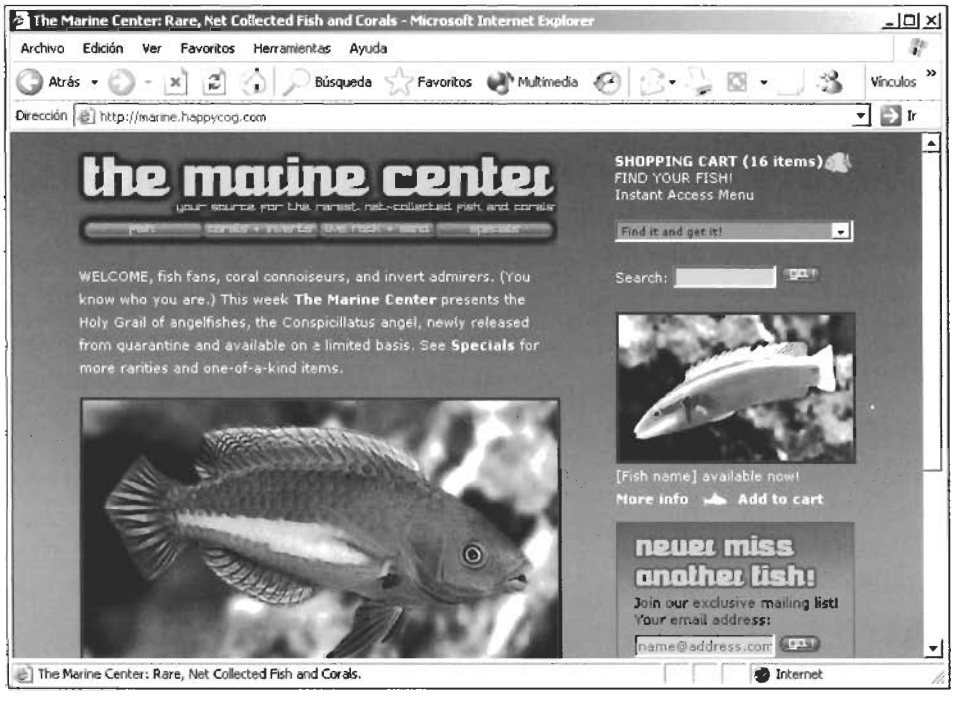


Figura 7.5.

Pescados en el menú: The Marine Center (www.themarinecenter.com) ofrece peces y corales de gran rareza y un cumplimiento híbrido de los estándares.

El año del mapa

En los albores del diseño Web comercial, se utilizaban mapas de imágenes para conseguir barras de navegación como la de la figura 7.3 o como la que aparece en la parte superior e inferior de la figura 7.5. Los primeros mapas de imágenes estaban formados por cinco partes:

- Un archivo de Photoshop (o de Canvas o cualquier otro editor de imágenes) que literalmente era una fotografía de una interfaz de usuario, guardado en formato GIF o, posteriormente, en JPEG.
- Un archivo que contenía las coordenadas de cada región activa de la imagen junto con el URL a cada región activa vinculada.
- Un programa CGI, normalmente escrito en PERL e instalado en un directorio especial del servidor, cuya labor consistía en traducir los clics de ratón del usuario en los URL especificados en el archivo.
- Un atributo ISMAP que se añadía al elemento de imagen en el código fuente de la página.
- Un ancla de hipertexto alrededor del elemento de imagen que apuntaba a la ubicación del programa CGI.

El HTML solía tener este aspecto:

```
<A HREF="/cgi-bin/imagemap/image.map">
<IMG SRC="/images/image.gif" ISMAP></A>
```

Éste era el famoso mapa de imágenes del lado del servidor de principios de los 90. Recibía este nombre porque la secuencia de comandos CGI del servidor del propietario del sitio se encargaba de traducir los clics de ratón del usuario en los URL asignados. Evidentemente, el diseñador también trabajaba lo suyo. El diseño de este tipo de elementos no era nada fácil, pero era la única forma de conseguir el efecto visual deseado.

Las asignaciones de imágenes del lado del servidor se vieron reemplazadas por asignaciones de imágenes del lado del cliente. En este caso, la traducción de clics en coordenadas se producía en el cliente, es decir, en el navegador del visitante, en lugar de en el servidor. Este tipo de asignaciones no requería un archivo diferente; en su lugar, las coordenadas se incrustaban en el HTML de la página, con un aspecto similar al que mostramos a continuación:

```
<map name="navigation">
<area shape="rect" coords="0,400,75,475"
href="index.html">
<area shape="rect" coords="401,500, 425,
525" href="events.html">
...
</map>
```

Las asignaciones de imágenes del lado del cliente resultaban más sencillas y, en consecuencia, la opción del lado del servidor se descartó con el tiempo. El progreso es genial, ¿verdad?

Descontentos con la asignación de imágenes

Los programadores que utilizaban asignaciones de imágenes y que querían mantener a los usuarios al día de sus cambios de ubicación dentro de la jerarquía del sitio (véanse figuras 7.3 y 7.4) no tenían más opción que

crear un archivo de navegación diferente para cada página. Asediados, a los usuarios de módem no les quedaba más remedio que descargar un nuevo archivo de imagen para cada página. Si el programador era hábil, también cambiaba el archivo en cada página, de forma que no se pudiera cambiar la parte de la imagen que pertenecía a la página actual.

Se necesitaba una tonelada métrica de ancho de banda para conseguir estos efectos y como entre 1994 y 1996 la velocidad de marcado no era excesiva, los usuarios estaban descontentos. El incipiente movimiento por la facilidad de uso, viendo la frustración de los usuarios, declaró que las imágenes eran generalmente incorrectas.

En la actualidad, prácticamente nadie utiliza asignaciones de imágenes. Aun así, debido a estas primeras experiencias, algunos consultores y sus seguidores permanecen contrarios a las imágenes y al diseño gráfico. Es como renegar de los automóviles porque una vez un Hispano Suiza asustó a nuestro caballo. Pero no estamos aquí para discutir ninguna opinión (a excepción de aquéllas que malinterpretan o tergiversan los estándares Web).

Sin acceso, sin estructura

Las asignaciones de imágenes permitían conseguir un determinado aspecto y funcionalidad, pero únicamente funcionaban en navegadores gráficos de escritorio en los que se habían activado las imágenes y exigían del usuario tanto movilidad física (para utilizar el ratón) y una vista perfecta (para ver las imágenes). Podría haberse mejorado la accesibilidad por medio del atributo `alt`, pero normalmente no se hacía.

En lo que nos respecta, las asignaciones de imágenes no transmitían un sentido estructural. Simplemente se trataba de imágenes sobre las que se podía pulsar si se disponía del equipo adecuado (incluyendo el equipo humano). Se había creado el precedente del HTML no estructural.

El método de cortar y trocear

Entre los principales defectos de estas asignaciones se encontraba su incapacidad para ahorrar ancho de banda y almacenar en caché los componentes de imagen recurrentes. Los diseñadores comenzaron a cortar y a trocear sus componentes de Photoshop, a exportar los segmentos de imagen en formato GIF o JPEG y a utilizar tablas HTML para ubicar todas las piezas. (Dudamos que necesite que le refresquemos la memoria a este respecto pero, por si acaso, puede consultar los primeros capítulos.)

El método de cortar y trocear no era estructural, pero ahorra ancho de banda para el servidor y para el usuario, ya que almacenaba los elementos recurrentes en la caché del navegador en el disco duro del usuario. Y se podía mejorar la accesibilidad por medio del atributo `alt` de la etiqueta de la imagen:

```
<a href="/events/">
<img src="/images/events.gif"
alt="Events">
</a>
```

Como este método facilitaba la creación de un determinado aspecto y funcionalidad sin la sobrecarga de ancho de banda de las

asignaciones de imágenes, los diseñadores y programadores rápidamente lo adoptaron.

El final del método de cortar

Al principio, aplanábamos las capas de Photoshop, seleccionábamos a mano cada fragmento por medio de la herramienta de marco rectangular, y los copiábamos y pegábamos en un nuevo archivo independiente. Utilizábamos complementos de terceros para exportar cada archivo en formato GIF. Escribíamos las tablas HTML a mano y rezábamos para que los fragmentos seleccionados manualmente se combinaran a la perfección. Batíamos nuestra propia mantequilla y zurcíamos nuestros propios calcetines.

Pero sucedieron varias cosas.

Netscape amplió el modelo de objeto JavaScript que había inventado en 1996 para incluir la posibilidad de intercambiar imágenes en respuesta a los movimientos del ratón del usuario. Nació la imagen cambiante. Algunos aprendimos a utilizar JavaScript para incluir estados de imágenes cambiantes en nuestros diseños de tabla codificados a mano. Otros copiaban y pegaban de sitios pioneros como Project Cool. Algunos hacíamos ambas cosas.

Mientras tanto, Photoshop fortalecía sus opciones Web, a lo que Adobe respondió rápidamente con su aplicación ImageReady, que en un principio se vendía de forma independiente pero que ahora se integra con Photoshop. Con ImageReady, se pueden arrastrar guías de Photoshop y dejar que el programa se encargue de cortar y trocear. Incluso puede escribir el marcado de tablas HTML y, en versiones posteriores, también genera imágenes cambiantes JavaScript.

Fireworks, la herramienta de creación de imágenes Web de Macromedia, hacía lo mismo (y sigue haciéndolo) y tanto Macromedia como Adobe empezaron a ofrecer herramientas de edición Web WYSIWYG que podían crear tablas e imágenes cambiantes aunque no supiéramos crear el código de las mismas a mano.

En la actualidad, los programadores disponen de lo mejor de ambos mundos: un agradable efecto visual con todas las labores duras realizadas por el ordenador (que, al contrario que los seres humanos, nunca se cansa de realizar estos trabajos).

En defensa de los diseños de tablas de navegación

Las tablas HTML (o XHTML) que contienen imágenes GIF y JavaScript siguen siendo una norma del diseño y la programación Web actuales. Algunos defensores de los estándares no están de acuerdo ya que no son estructurales y algunos consultores las desdeñan como desdeñan todo lo que tenga que ver con el diseño gráfico.

Pero como hemos tratado de mostrarle en este capítulo (y que veremos con más detalle en capítulos posteriores), estas tablas siguen siendo una herramienta muy adecuada para diseños híbridos y de transición que requieren un determinado aspecto y funcionalidad. Su principal defecto es que se necesita mucho trabajo para cambiarlas cuando un sitio revisa su arquitectura o actualiza su aspecto operativo y visual. Este defecto penaliza a

los diseñadores y propietarios de sitios, pero no afecta a los usuarios.

Las tablas no son la única forma de conseguir este tipo de efectos, pero se pueden utilizar en diferentes entornos de navegación, se pueden reproducir con marcado compacto y accesible, y suelen malgastar menos ancho de banda que las alternativas sin imágenes que muchos diseñadores probaron en los años 90.

Los detalles redundantes de las tablas redundantemente detalladas

En un intento por ahorrar ancho de banda, aumentar la accesibilidad y tener sentido para los motores de búsqueda (que no pueden leer imágenes), a mediados de los 90, algunos diseñadores Web dejaron de utilizar texto GIF y comenzaron a imitar diseños de cortar y trocear mediante la creación de tablas de navegación en HTML y la inclusión de éstas en diseños de tabla. Los navegadores todavía no eran compatibles con CSS, pero las extensiones HTML propietarias sí. Por esta razón, se obtiene basura como la del siguiente ejemplo, que puede gastar más ancho de banda que las imágenes a las que pretende reemplazar:

```
<!-- Begin outermost table. This creates
the black border effects. -->
<TABLE WIDTH=80% BORDER=0 CELLSPACING=1
CELLPADDING=0>
<TR>
<TD WIDTH=100% VALIGN=TOP
BGCOLOR="#000000">
<!-- Begin actual navigational content
table. -->
<TABLE WIDTH=100% HEIGHT=100% BORDER=0
CELLSPACING=1 CELLPADDING=0>
<TR>
<TD WIDTH=25% HEIGHT=50 VALIGN=TOP
BGCOLOR="#FF9900"><FONT SIZE=2><BR><BR>
<FONT FACE="GENEVA, ARIAL, HELVETICA"><B>
<A HREF="item1.html">Menu Item 1</A>
</B></FONT><BR><BR></FONT></TD>
<TD WIDTH=25% HEIGHT=50 VALIGN=TOP
BGCOLOR="#FF9900"><FONT SIZE=2><BR><BR>
<FONT FACE="GENEVA, ARIAL, HELVETICA"><B>
<A HREF="item2.html">Menu Item 2</A>
</B></FONT><BR><BR></FONT></TD>
<TD WIDTH=25% HEIGHT=50 VALIGN=TOP
BGCOLOR="#FF9900"><FONT SIZE=2><BR><BR>
<FONT FACE="GENEVA, ARIAL, HELVETICA"><B>
<A HREF="item3.html">Menu Item 3</A>
</B></FONT><BR><BR></FONT></TD>
<TD WIDTH=25% HEIGHT=50 VALIGN=TOP
BGCOLOR="#FF9900"><FONT SIZE=2><BR><BR>
<FONT FACE="GENEVA, ARIAL, HELVETICA"><B>
<A HREF="item4.html">Menu Item 4</A>
</B></FONT><BR><BR></FONT></TD>
</TR>
</TABLE>
<!-- End actual navigational content
table. -->
</TD>
</TR>
</TABLE>
<!-- End outermost visual effects
creation table. -->
```

Este tipo de marcado apesta pero sigue utilizándose en muchos sitios comerciales, incluyendo los de algunos líderes del sector, aunque sea un paso atrás en lo que a la estética del marcado se refiere. (Compare este marcado con el que crearemos en capítulos posteriores.) Muchos se sobreponen al susto almacenando fragmentos de marcado de presentación en los registros de sus bases de datos. De esta forma, si se intenta realizar un cambio de diseño con marcado limpio y esencialmente estructural, el viejo marcado basura, que proviene de la base de datos, lo destruirá.

Las organizaciones que han utilizado estas técnicas sólo pueden emerger de la Edad de piedra digital con grandes esfuerzos y presupuestos. Como la producción de este tipo de

sitios no tiene futuro, tarde o temprano dichas organizaciones tendrán que tomar una decisión. Si trabaja en una de estas organizaciones, esperamos que este libro le ayude a tomar dicha decisión.

La llegada de la CSS incorrecta

En 1997, cuando MSIE3 comenzaba a admitir parcialmente CSS1, algunos de nosotros, intrigados por la promesa del W3C de un lenguaje de diseño, enfocamos en un principio nuestra labor con los mismos detalles redundantes y no estructurales que habíamos derrochado en nuestros diseños de tablas sin imágenes.

En lugar de realizar vínculos a una hoja de estilo que podía almacenarse en caché en el disco duro del usuario, utilizábamos estilos en línea como los del ejemplo del siguiente apartado que malgastaban tanto ancho de banda como los antiguos atributos `font` y `bgcolor`. En un intento por admitir navegadores no CSS, a menudo utilizábamos las etiquetas de fuente y otros elementos sobrantes de cualquier forma, lo que duplicaba el ancho de banda que consumían nuestros diseños. Y no hace falta decir que no nos preocupábamos lo más mínimo por la estructura de los documentos.

El verdadero poder de CSS para separar la estructura de la presentación no estaba disponible en los navegadores y, lo más importante, tampoco en las mentes de la mayoría de los diseñadores Web profesionales (y en muchos casos sigue siendo así). Los diseñadores no estaban preparados para pasar a CSS ya que no existía para ellos.

A la descripción anterior de la clasitis y la divitis puede añadir su predecesor, mucho más maligno ya que incluye todos los estilos CSS en línea, con lo que se pierden las ventajas de CSS: ahorrar ancho de banda y conservar el significado mediante la aplicación de estilos globales a marcado estructurado en todo el sitio. Hemos recuperado el siguiente fragmento de CSS/XHTML incorrecto de 1997 para que compruebe su aspecto y para que evite escribir algo parecido.

Como si fuera 1997

El siguiente fragmento de código es un ejemplo perfecto del tipo de CSS incorrecto (combinado con un marcado erróneo) que muchos de nosotros escribíamos en 1997. También es el tipo de CSS que generaban las herramientas de edición visual de entonces, en caso de que se pudieran configurar para ello:

```
<font color="#FFFFFF">|</font>
 
 
<a style="color:#FFFFFF;
text-decoration:none;" href="/isapi/
gosearch.asp?target=/us/default.asp"
target="_top"
onmouseover="this.style.color='#FFCC00';"
onmouseout="this.style.color=
'#FFFFFF';">Search</a> 
```

Aunque parezca increíble, esta pequeña muestra de deshecho no se ha rescatado de los archivos de diseño del 97, tampoco es un trabajo de un estudiante, ni siquiera el trágico producto de un perdido sitio comercial. Se obtuvo, tal y como lo ve, de la página principal de Microsoft (véase figura 7.6) el 7 de enero del 2003, y se ha actualizado continuamente hasta ese día.

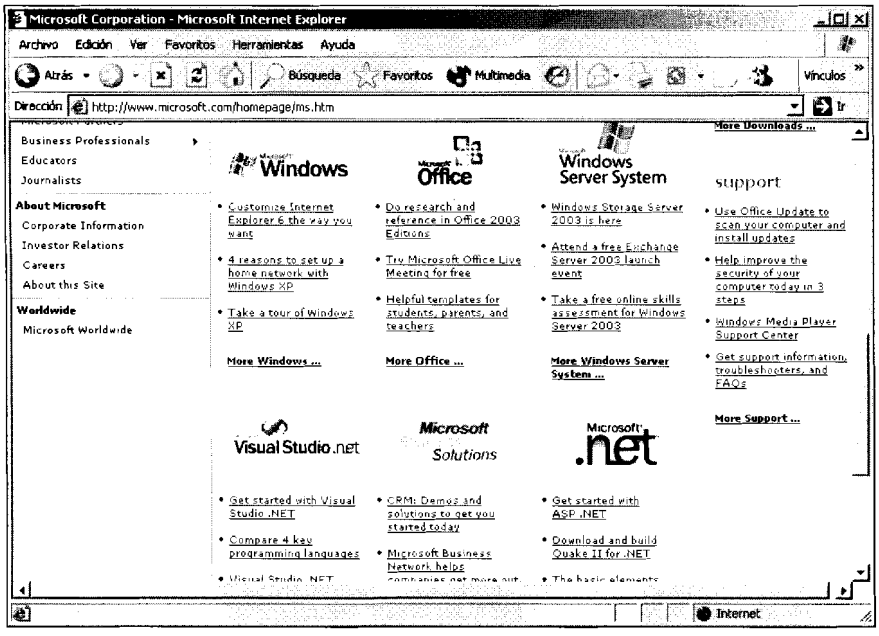


Figura 7.6.

¿A dónde le gustaría ir hoy? A 1997, a juzgar por el marcado y CSS utilizados en la página de Microsoft en el 2003 (www.microsoft.com).

Como la redundancia es igual de incorrecta en libros como en código, no explicaremos lo que le pasa a este marcado. Si todavía no lo sabe, significa que alguno de los dos no ha hecho los deberes.

Microsoft es miembro del W3C, un contribuidor clave a las especificaciones CSS y XHTML, y el fabricante de navegadores compatibles que hace tiempo han abandonado el gusto por HTML y CSS como los utilizados en su página principal. Puede consultar un capítulo anterior sobre una nueva era de cooperación y preguntarse el porqué de esta vuelta a los peores métodos de marcado y CSS de los 90.

Afortunadamente, a finales de 1997 y principios de 1998, la mayoría de los diseñadores habían desechado este tipo de creaciones de encefalograma plano. Desafortunadamente, lo que adoptamos como alternativa era casi igual de malo (clasitis y divitis). Pero el horizonte se despeja.

Un paso adelante

Los diseños de tablas pueden crearse correctamente, ser ligeros, accesibles y compatibles con los estándares. Y los diseños CSS pueden hacer más: aligerar nuestra carga de trabajo y la del servidor, separar la estructura de la presentación para aumentar el alcance y la utilidad de un sitio, y proporcionar a los diseñadores visuales nuevas herramientas para desbloquear su creatividad. En los siguientes capítulos consolidaremos lo que hemos aprendido hasta ahora y comenzaremos nuestro viaje a CSS con la creación de un sitio compacto y compatible.

Capítulo 8
Ejemplos de XHTML: un diseño híbrido (parte I)

Este capítulo y los siguientes forman un pequeña unidad. Para empezar, nos pondremos manos a la obra y aplicaremos lo que hemos aprendido hasta ahora sobre XHTML para crear el marcado de un proyecto de diseño real. El marcado que crearemos será parcialmente estructural, parcialmente de transición y completamente compatible con los estándares. Tras ello, describiremos los fundamentos de las hojas de estilo en cascada (CSS) para usuarios principiantes e intermedios. Por último, nos centraremos más en CSS y utilizaremos los nuevos conocimientos para completar nuestro proyecto. Este tipo de aprendizaje práctico no es muy distinto a aprender a nadar lanzándose directamente al agua, aunque preferimos presentarlo como si aprendiera francés visitando París. Al crear el proyecto, aprenderemos a incorporar accesibilidad a nuestro marcado (y, tras ello, a nuestros sitios).

Ventajas de los métodos de transición utilizados en estos capítulos

En este capítulo, comenzaremos con la creación de un diseño híbrido y de transición que combine técnicas de diseño de tablas tradicionales (pero en este caso racionalizadas) con mejoras de marcado textual estructurado y accesibilidad. Las técnicas empleadas en este proyecto y que describiremos en los siguientes capítulos resultan idóneas para bibliotecas y otras instituciones públicas, así como para pequeñas empresas y organizaciones cuyos objetivos sean los siguientes:

- Gestionar grandes cantidades de contenidos con un presupuesto limitado.
- Admitir una amplia variedad de navegadores y dispositivos.

- Conservar el ancho de banda de sus visitantes (y el suyo propio).
- Iniciar la transición hacia los estándares Web con métodos de publicación fiables, económicos y sencillos de implementar.

Hojas de estilo en lugar de JavaScript

Al finalizar estos capítulos, habremos creado una plantilla compatible con estándares para el sitio i3Forum (véase figura 8.1). La plantilla final creada en estos capítulos se puede ver en el servidor de Happy Cog, en http:// i3.happycog.com/. El sitio terminado, creado por medio de estas plantillas, se encuentra en http://i3forum.com/.

En este capítulo, estableceremos nuestro marcado. En un capítulo posterior añadiremos CSS para controlar el color, el tamaño y la posición de los elementos. (Antes nos detendremos para aprender los conceptos básicos de CSS.) Entre otras cosas, la CSS que crearemos nos permitirá obtener los efectos de imágenes cambiantes de menú que normalmente se consiguen con imágenes y JavaScript. No hay nada incorrecto en el uso de imágenes o JavaScript, pero al utilizar texto HTML y CSS, ahorramos ancho de banda al tiempo que el sitio será más accesible para una amplia variedad de entornos, incluyendo lectores de pantalla y navegadores de texto, navegadores no de escritorio (como PDA y navegadores basados en telefonía) y navegadores incompatibles con CSS.

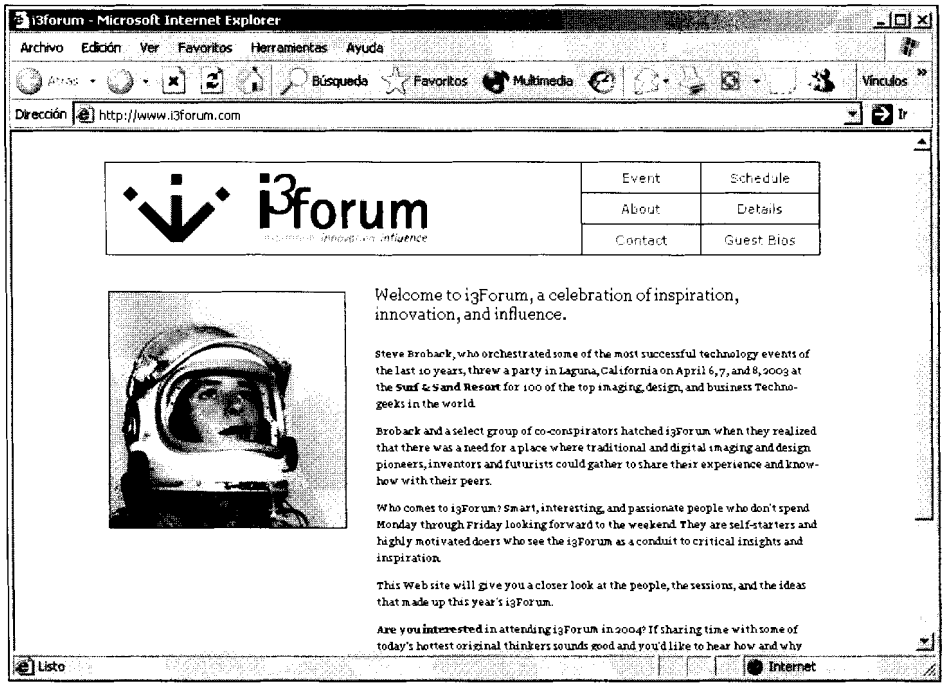


Figura 8.1.

La plantilla terminada, como aparecerá tras completar los procesos descritos en los siguientes capítulos.

Enfoque básico (resumen)

El diseño de i3Forum tiene como objetivo conseguir una identidad concisa y atrayente sin excesos, y su XHTML es igual de sencillo. Está formado por dos tablas XHTML, centradas y mejoradas y controladas con ayuda de CSS. La primera tabla se encarga del menú de navegación y, la segunda, del contenido (véase figura 8.2).

El XHTML de las tablas se describirá a continuación. Pero seguramente haya reparado en un aspecto preliminar. Habitualmente, este tipo de diseños suele utilizar una sola tabla, con elementos `rowspans` y `colspans` para las diferentes filas y columnas. Si usáramos ImageReady de Adobe para cortar y trocear automáticamente los componentes de Photoshop utilizados para diseñar el sitio (y para vender el diseño al cliente), ImageReady representaría toda la página en una sola tabla. Se preguntará entonces por qué hemos utilizado dos.

Tablas separadas: ventajas de accesibilidad y CSS

Si se saltó el análisis sobre `div`, `id` y otros asistentes de capítulo anterior, puede que le interese volver a consultarlo antes de continuar. Al dividir nuestro diseño en dos tablas podemos dominar el poder del atributo `id` para lo siguiente:

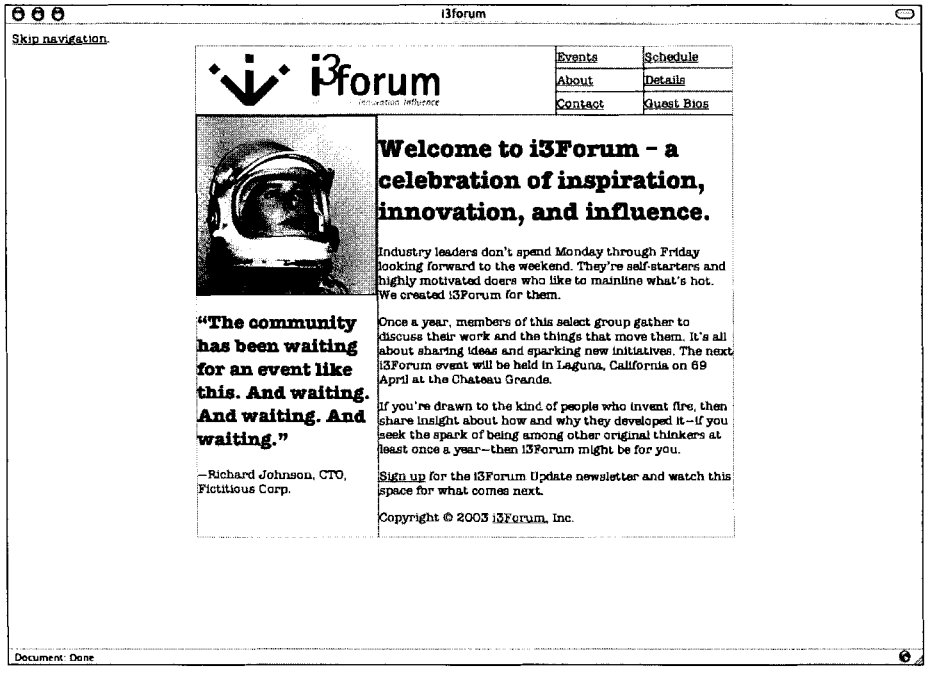


Figura 8.2.

La plantilla que crearemos en este capítulo, en la que se ha desactivado CSS y se han activado los bordes. Fíjese en la línea ligeramente más gruesa entre el menú y las zonas de contenido, donde se juntan las dos tablas.

- Racionalizar la CSS que crearemos en un capítulo posterior.
- Aplicar determinadas mejoras de accesibilidad.
- Etiquetar estructuralmente cada tabla en función de la tarea que desempeñe, lo que permitirá que sea más fácil cambiar el diseño y reemplazar las tablas XHTML de presentación por `div` con estilos CSS.

El elemento de resumen de tabla

Al dividir el diseño en dos tablas, también podemos añadir un atributo de resumen a cada una:

```
<table id="nav" summary="Elementos de
navegación" … etc. >
<table id="content" summary="Contenido
principal." … etc. >
```

El atributo de resumen es invisible para navegadores de escritorio convencionales como IE y Netscape. Pero el software de lectura de pantalla utilizado por visitantes con carencias visuales entiende el atributo de resumen y puede leer su valor en alto. En nuestro caso, el lector de pantalla dirá "Elementos de navegación" y "Contenido principal". Los lectores de pantalla bien diseñados permiten a los usuarios ignorar la tabla si creen que no les interesa.

Por esta razón, la creación de resúmenes de tabla es una correcta estrategia de respaldo a la accesibilidad para llegar a usuarios que puedan pasar por alto el vínculo para ignorar la navegación, Skip navigation, que describiremos más adelante.

Estructura de páginas e id

Hemos asignado un valor de atributo `id` a cada una de las tablas en función de la labor estructural que realizan: navegación o contenido. De esta forma, posteriormente podremos escribir reglas CSS compactas que se puedan aplicar a toda la tabla y evitar las temidas clasitis y divitis.

También nos permite incluir un vínculo para ignorar la navegación en la parte superior del marcado.

El qué y el por qué del vínculo para ignorar la navegación

Como su nombre indica, este vínculo permite a los usuarios saltarse la navegación y acceder directamente a la tabla de contenido por medio de un vínculo de ancla.

El atributo `id` con el valor `content` proporciona el ancla a la que realizaremos el enlace:

```
<div class="hide"><a href="#content"
title="Skip navigation." accesskey=
"2">Skip navigation</a>.</div>
```

El hecho de ignorar la navegación no es un requisito urgente para los usuarios Web normales, que pueden centrar su atención en determinadas partes de una página Web simplemente con fijarse en dichas partes e ignorando las que no les interesan.

Pero los usuarios con problemas de visión que utilizan lectores de pantalla experimentan la Web de forma lineal, un vínculo por vez. Puede resultar frustrante para dichos usuarios soportar un constante fluir sonoro de vínculos de menú cada vez que carguen

una página de nuestro sitio. El vínculo Skip Navigation les permite evitar este problema.

Este vínculo también puede ser de utilidad para los usuarios que utilicen navegadores PDA no compatibles con CSS o teléfonos Web. Evitarán tener que desplazarse por infinidad de vínculos cada vez que carguen una nueva página. Por último, el vínculo Skip Navigation puede beneficiar a los usuarios con discapacidades físicas, aunque no es un método perfecto (encontrará más información en un apartado posterior).

Skip Navigation y acceskey

Nuestro vínculo Skip Navigation permite a los visitantes con problemas de visión o que utilicen navegadores incompatibles con CSS acceder directamente al contenido de la segunda tabla, cuyo atributo `id` es `content` (y también su vínculo de ancla):

```
<table id="content" …> etc.
```

En dichos entornos no visuales o incompatibles con CSS, el vínculo está disponible en la parte superior de la página (véanse figuras 8.3 y 8.4).

En un capítulo posterior crearemos una regla para ocultar dicho vínculo en navegadores compatibles con CSS (aunque lo hemos incluido a continuación, por si acaso se impacienta):

```
.hide    {
    display: none;
    }
```

Debido a esta regla, los visitantes que utilicen navegadores modernos en los que hayan activado CSS no verán el vínculo Skip Navigation, aunque no necesitan verlo ya que no necesitan la funcionalidad del mismo, como hemos indicado en un apartado anterior. Los lectores de pantalla que ignoren CSS leerán fácilmente el contenido del `div` y, por lo tanto, informarán a los usuarios con problemas de visión de que pueden saltarse la aburrida recitación de los vínculos restantes. (Desafortunadamente, algunos lectores de pantalla obedecen a las CSS aunque sus usuarios no puedan verlas).

Evidentemente hay una excepción. Un usuario con dificultades de movilidad, que utilice un navegador compatible con CSS para ver el sitio, puede que quiera ignorar el área de navegación y acceder directamente al contenido. La mayoría de los usuarios con dificultades de movilidad puede ver la página Web entera de una sola vez (excepto aquellos usuarios con dificultades físicas y visuales). Pero para navegar a esa página, los usuarios impedidos utilizan el teclado o un dispositivo de entrada alternativo.

El hecho de tener que atravesar vínculos de navegación no deseados puede convertirse en una molestia o en algo peor.

¿Cómo pueden estos usuarios ignorar la navegación si no pueden ver el vínculo Skip Navigation en sus navegadores? Hemos incluido esta opción por medio del atributo `accesskey`, que funciona incluso cuando no se puede ver el vínculo Skip Navigation en el navegador.

Desafortunadamente, el método es imperfecto como veremos a continuación.

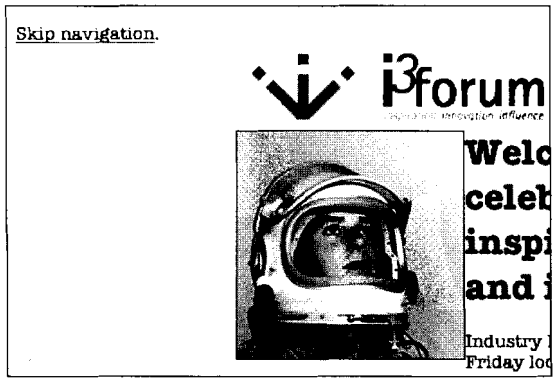


Figura 8.3.

En un navegador incompatible con CSS (o un navegador compatible pero en el que se haya desactivado CSS), el vínculo Skip Navigation es claramente visible en la parte superior de la página.

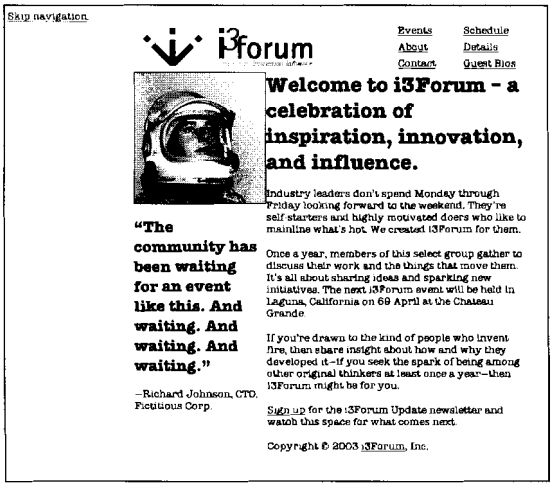


Figura 8.4.

El vínculo Skip Navigation en contexto, en nuestro diseño en el que se ha desactivado CSS.

accesskey: buenas y malas noticias

El atributo `acceskey` permite a los usuarios navegar en sitios Web por medio del teclado en vez de con el ratón. Para asignar un atributo `accesskey` a un elemento, basta con declararlo, como en el fragmento XHTML anterior, que volvemos a reproducir con el correspondiente atributo y valor en negrita:

```
<a href="#content" title="Skip
navigation." accesskey="2">Skip
navigation</a>.
```

En nuestro marcado, hemos asignado al vínculo Skip Navigation el `accesskey` 2. Por ello, para ignorar la navegación, basta con que el visitante pulse **2** en su teclado. Como suele ocurrir con las mejoras de accesibilidad, el marcado necesario suele ser sencillo de escribir y no afecta al diseño visual del sitio. En este caso, es tanto negativo como positivo.

¿Cómo sabe el visitante que tiene que pulsar **2** en su teclado? Ninguno de los navegadores más utilizados muestra las asignaciones de letra de `accesskey`. Ni tampoco los menos utilizados.

accesskey e iCab

Al cierre de esta edición, solamente iCab (véase figura 8.5), un navegador de Macintosh, muestra visualmente las asignaciones de letra de accesskey. La mayoría de los navegadores Web no son usuarios de Macintosh y la mayoría de los usuarios de Macintosh no utilizan iCab.

Para empeorar las cosas, iCab no puede mostrar la asignación de `accesskey` si el vínculo Skip Navigation se oculta por medio de CSS. Por el momento, iCab sigue sin admitir CSS1, la primera recomendación CSS del W3C, publicada en 1996.

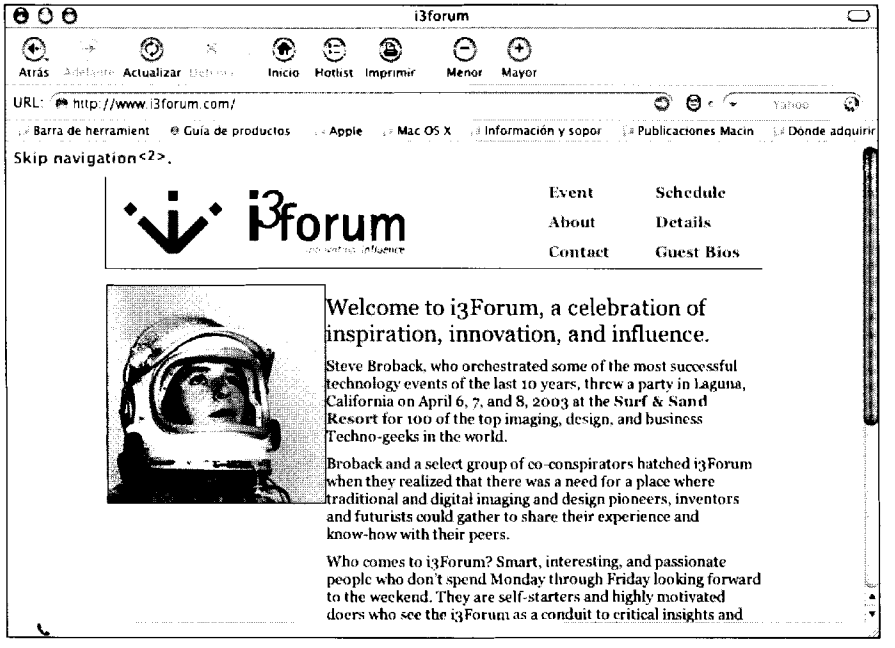


Figura 8.5.

De todos los navegadores del mundo, sólo iCab para Macintosh (http://www.icab.de/) muestra nuestra accesskey 2, para indicar al usuario que puede saltarse la navegación si pulsa 2 en su teclado.

En definitiva, aunque iCab es un navegador interesante y su compromiso para admitir HTML 4 es impresionante (no bromeamos, la compatibilidad de iCab con HTML 4 es magnífica), iCab no va a solucionar el problema de `accesskey`.

Dos posibilidades utópicas para accesskey

Evidentemente, la mayoría de usuarios que pueden beneficiarse de `accesskey` no disponen de ningún modo para saber qué letras o números `accesskey` deben pulsar y, por lo tanto, no pueden beneficiarse de nada. Por esta razón, la inclusión de `accesskey` en su marcado es un tanto idealista.

Si el W3C recomendara asignaciones estándar de `accesskey` para funciones universales como las de ignorar la navegación (y si los diseñadores y programadores siguieran estas recomendaciones), los usuarios siempre sabrían qué teclas pulsar. Sería algo muy indicado.

Por otra parte, los fabricantes de navegadores podrían optar por aumentar su compatibilidad con `accesskey` y mostrar los valores `accesskey` visualmente si el usuario decide activar esta opción de accesibilidad en sus preferencias. IE para Windows cuenta con una opción de accesibilidad que permite a los usuarios ignorar el tamaño de las fuentes en cualquier página Web. También podría incluir una opción para mostrar los valores de `accesskey` en todo momento.

Debemos admitir que es ciertamente utópico que el W3C estandarice los métodos abreviados de `accesskey` a corto plazo y parece utópico que cualquiera de los principales fabricantes de navegadores (y mucho menos todos ellos) dediquen su tiempo y sus recursos para que una opción se vea en todo

momento. Sin embargo, seguiremos utilizando `accesskey`. Algunos usuarios pueden acceder al código fuente para comprobar los valores `accesskey` utilizados y utilizar las correspondientes teclas para navegar. Esperemos que todo sea más sencillo para estos usuarios.

Atributos id adicionales

Además de los nombres principales de atributo `id` (`nav` y `content`), en nuestra primera pasada por el marcado del sitio también asignamos nombres de atributo `id` exclusivos a cada una de las celdas de la tabla de navegación. Bastará con dos celdas para que el método resulte claro:

```
<td width="100" height="25" id="events">
<a href="events.html">Events</a></td>
<td width="100" height="25"
id="schedule"><a href="schedule.html">
Schedule</a></td>
```

También asignamos valores de atributo `id` a cada una de las dos divisiones principales de la tabla de contenido, es decir, a la barra lateral (`id="sidebar"`) y a la zona de contenido principal (`id="primarycontent"`).

A continuación le mostramos el núcleo de la tabla de contenidos, con gran parte de los datos eliminados por motivos de claridad. Los valores y atributos `id` se muestran en negrita:

```
<table id="content" etc.>
<tr>
<td width="200" id="sidebar">
Aquí va la barra lateral.
</td>
<td width="400" id="primarycontent">
Aquí va el contenido principal.
</td>
```

Hemos añadido un nombre de atributo `id` en las filas secundarias de la barra de navegación. Por ello, la segunda fila de botones de navegación tiene el siguiente valor `id`:

```
<tr id="nav2">
```

Y, como habrá imaginado, la tercera fila de botones de navegación tiene este otro valor `id`:

```
<tr id="nav3">
```

¿Cuánto es demasiado?

Los dos últimos nombres de atributo `id` (`nav2` y `nav3`) no son necesarios para los objetivos de este diseño, aunque pueden resultar útiles algún día, por ejemplo para realizar un cambio de diseño. ¿Debemos incluirlos o no? Si los incluimos, sólo se añaden algunos bytes adicionales al XHTML, aunque podríamos también no incluirlos.

Si en el sitio definitivo la barra de navegación se encuentra en un archivo SSI diferente (o en un registro exclusivo gestionado por PHP, JSP, ColdFusion o ASP), el cliente podría editar dicho archivo en cualquier momento y cambiar la totalidad del sitio con tan sólo modificar un archivo. Si el cliente pretende utilizar tecnologías del lado del servidor, puede que no tenga mucho sentido incluir `nav2` y `nav3`. Por otra parte, si no se utilizan estas tecnologías y el marcado de los menús se repite manualmente en todas las páginas, puede que sea más seguro incluir `nav2` y `nav3` para evitar posibles errores de búsqueda y reemplazo en un cambio de diseño posterior. Y eso es lo que hemos hecho.

Marcado de primera pasada: igual que el marcado de última pasada

En ésta y en las páginas siguientes, se incluye la primera pasada por el marcado del sitio desde <body> hasta </body>. También es nuestra última pasada por el marcado del sitio. Todos los ajustes de diseño posteriores se controlarán en su totalidad por CSS. Es una de las ventajas que tiene descargar la mayor cantidad de elementos de presentación en las hojas de estilo, incluso en un diseño híbrido como el nuestro. Para ahorrar espacio en este libro, hemos sustituido la copia de cuerpo utilizada en la plantilla por texto marcador de posición genérico.

Fíjese también que en este marcado hemos utilizado vínculos relativos de referencia a archivos de imagen (`img src="images/logo.gif"`) en lugar de vínculos absolutos (`img src= "/images/logo.gif"`), ya que estamos trabajando desde nuestro escritorio y no desde el servidor. El marcado final utilizará vínculos de referencia absolutos. (Los URL absolutos son más fiables que los URL relativos ya que no se dividen si la ubicación de los archivos cambia: por ejemplo, si `/events.html` se cambia a `/events/`, la referencia absoluta a `/images/logo.gif` seguirá funcionando. Además, los URL absolutos permiten solucionar un error de CSS que aparecía en algunos navegadores antiguos que equivocaban las referencias de archivo relativas en las hojas de estilo.)

Técnicamente, el marcado definitivo difiere ligeramente del marcado de primera pasada ya que sustituye las referencias de archivo relativas por referencias absolutas. No es preocupante para muchos de nosotros, pero siempre hay algún lector que revisa el código de las páginas para comprobar las afirmaciones realizadas en un libro.

Puede que le resulte más sencillo ver el código fuente en su fuente original. Como hemos mencionado anteriormente en este capítulo, puede encontrar el proyecto en http://i3.happycog.com/.

Marcado de navegación: la primera tabla

A continuación se incluye la sección de navegación, a partir del elemento body. Para mantener el interés, le diremos que esta parte del marcado, aunque se puede validar, comete un pecado de pureza de marcado que no es de presentación. Veamos si lo puede detectar.

```
<body bgcolor="#ffffff">
<div class="hide"><a href="#content"
title="Skip navigation."
accesskey="2">Skip navigation</a>.</div>
<table id="nav" summary="Navigation
elements" width="600" border="0"
align="center" cellpadding="0"
cellspacing="0">
<tr>
<td rowspan="3" id="home" width="400">
<a href="/" title="i3Forum home
page."><img src="images/logo.gif"
width="400" height="75" border="0"
alt="i3forum home" /></a></td>
<td width=""100" height="25" id="events">
<a href=""events.html">Events</a></td>
<td width="100" height="25"
id="schedule"><a href="schedule.html">
Schedule</a></td>
</tr>
<tr id="nav2">
<td width="100" height="25" id="about">
<a href="about.html">About</a></td>
```

```
<td width="100" height="25" id="details">
<a href="details.html">Details</a></td>
</tr>
<tr id="nav3">
<td width="100" height="25" id="contact">
<a href="contact.html">Contact</a></td>
<td width="100" height="25"
id="guestbios"><a href="guestbios.html">
Guest Bios</a></td>
</tr>
</table>
```

Presentación, semántica, pureza y pecado

¿Es un verdadero fanático de los estándares? ¿Ha descubierto el gran pecado de nuestro XHTML? La principal ofensa aparece en la primera línea. Se utiliza el obsoleto atributo `bgcolor` (color de fondo) en el elemento del cuerpo para especificar, incluso en navegadores no compatibles con CSS, que el color de fondo de la página es el blanco (`#ffffff`). Veámoslo de nuevo:

```
<body bgcolor="#ffffff">
```

Si escribimos marcado de la vieja escuela como éste, nos veremos fuera de la academia de los estándares. Después de todo, CSS nos permite especificar el color de fondo del cuerpo y el W3C recomienda que para ello se utilice CSS, no HTML ni XHTML. Para muchos de los fanáticos de los estándares, el uso de `bgcolor` es un pecado.

Un libro de transición para un periodo de transición

Para los fanáticos de los estándares que se pasan horas y horas discutiendo sobre los males del marcado de presentación en las listas de correo del W3C, lo que hemos hecho en este ejemplo es maligno y nocivo.

A este respecto, también hemos pecado al utilizar tablas como contenedores de datos tabulares, al especificar anchuras y alturas en las celdas de tabla, y al configurar márgenes de imágenes como cero en el marcado. De hecho, para algunos, todo este capítulo es un pecado. Puede que algunos fanáticos de los estándares no piensen mucho en este libro, para ser francos. En su opinión, deberíamos enseñar a escribir marcado semántico en lugar de permitir que los lectores piensen que es correcto utilizar tablas en sus diseños.

Pero en realidad es correcto. Puede que dentro de unos años no lo sea, cuando los diseñadores utilicen y los navegadores admitan versiones totalmente semánticas de XHTML y versiones enriquecidas de CSS y SVG. Pero nuestro libro es un libro de transición para un periodo de transición. Los estándares Web no son un conjunto de leyes inmutables, sino un camino repleto de opciones y decisiones. En nuestra opinión, los que insisten en la pureza absoluta del entorno actual de navegación y de estándares, hacen tanto daño a la adopción generalizada de los estándares Web como los que nunca han oído hablar del marcado estructural y de CSS.

Concesiones para los navegadores antiguos

Se preguntará por qué hemos utilizado el atributo `bgcolor` a pesar de su maldad. El sitio híbrido que vamos a crear no asume nada con respecto a los navegadores usados por sus usuarios. En un navegador antiguo no compatible con CSS, si el color de fondo predeterminado se define con cualquier otro color que no sea el blanco, el logotipo GIF transparente del sitio se verá afectado

por halos provocados por los píxeles del borde que no coinciden. Ningún cliente quiere que su logotipo tenga este aspecto en un navegador, incluso si el resto del sitio se ve perfectamente en navegadores antiguos.

Un conocido navegador antiguo no compatible con CSS utilizó un gris medio como color de fondo predeterminado. Nuestro logotipo no se suaviza con gris, sino con blanco. Si no hubiéramos definido el color de fondo por medio del atributo `bgcolor`, nuestro logotipo no se vería correctamente en estos entornos de navegación.

En el mundo real, puede que no le importe el aspecto de su sitio en un navegador 2.0 ó 3.0. Y por esa regla de tres, puede que no le importe cómo se vea en un navegador 4.0, ni tampoco a su jefe o a su cliente. Las técnicas semánticamente impuras utilizadas en este capítulo no tratan de crear la misma experiencia visual en todos los navegadores. En un navegador no compatible con CSS, nuestro diseño no se verá mejor que como aparece en la figura 8.3. Y no pasa nada.

Hemos utilizado tablas en este sitio y hemos incluido `bgcolor` para mostrarle que estos compromisos se pueden realizar en XHTML Transitional 1.0 y que el sitio seguirá siendo válido. También lo hemos hecho para sugerir que cualquier esfuerzo por incluir estándares en nuestro trabajo, aunque sea un esfuerzo comprometido (pero realista) que utilice marcado de presentación, merece la pena.

Marcado de contenido: la segunda tabla

La tabla de contenido es la que aparece después de la de presentación y resulta bastante evidente para cualquiera que alguna vez haya escrito HTML o XHTML. Debe fijarse en dos aspectos principales: la compactación del marcado y el uso de `id`:

```
<table id="content" summary="Main
content." width="600" border="0"
align="center" cellpadding="0"
cellspacing="0">
<tr>
<td width="200" id="sidebar"
valign="top">
<img src="images/astro.jpg" width="200"
height="200" border="0" alt="i3Forum.
Breeding leadership." title="i3Forum.
Breeding leadership." />
<h2>Subhead</h2>
<p>Text</p>
</td>
<td width="400" id="primarycontent">
<h1>Headline</h1>
<p>Copy.</p>
<p>Copy.</p>
<p>Copy.</p>
<p>Copy.</p>
<div id="footer">
<p>Copyright &copy; 2003 <a href="/"
title="i3forum home page.">i3Forum</a>,
Inc.</p>
</div>
</td>
</tr>
</table>
</body>
```

En el siguiente capítulo analizaremos los fundamentos de CSS y, en otro capítulo posterior, utilizaremos CSS para añadir control visual a nuestro sitio híbrido.

Capítulo 9
Conceptos básicos de CSS

En el capítulo anterior comenzamos a crear un sitio híbrido, de transición y compatible con estándares. En el capítulo siguiente, finalizaremos nuestra experiencia con las Hojas de estilo en cascada (CSS). Pero primero analizaremos algunos aspectos básicos. En este capítulo, repasaremos los fundamentos de la gramática CSS, analizaremos algunos conceptos no tan básicos y terminaremos con una descripción de un método de diseño CSS que es diferente al utilizado por la mayoría de los diseñadores y a los que se incluyen en muchos libros. Incluso si ya está familiarizado con CSS puede que le interese.

Presentación de CSS

El W3C define CSS como "un mecanismo simple para añadir estilo (por ejemplo, fuentes, colores, espacios) a documentos Web" (www.w3.org/Style/ CSS/). Conviene mencionar algunos detalles que se omiten en este resumen:

- CSS es un lenguaje de diseño estándar para la Web. Controla los colores, la tipografía y el tamaño y ubicación de elementos e imágenes.
- Aunque es preciso y potente, CSS resulta muy sencillo de crear a mano, como veremos en este capítulo.
- CSS no malgasta ancho de banda. Un solo archivo CSS puede controlar el aspecto de todo un sitio, formado por miles de páginas y cientos de megabytes.
- Durante mucho tiempo, los creadores de CSS (W3C) han intentado que reemplace a los diseños HTML basados en tablas o en marcos, pero como veremos en un

capítulo posterior, también puede funcionar correctamente en un diseño híbrido y de transición.

- El diseño CSS puro, combinado con XHTML estructural, puede ayudar a los diseñadores a separar la presentación de la estructura, lo que mejora la accesibilidad de los sitios y facilita su mantenimiento, como veremos en el siguiente apartado.

Ventajas de CSS

Hay un proverbio ruso que dice que la repetición es la madre del aprendizaje. Así que perdónenos si le recordamos que CSS, al igual que otros estándares Web, no se creó por motivos abstractos y tampoco para un futuro lejano. Si lo utiliza correctamente en el presente, obtendrá prácticas ventajas como las que mencionamos a continuación:

- Conserva el ancho de banda del usuario, lo que acelera la carga de páginas, especialmente en conexiones telefónicas.
- Reduce la sobrecarga del servidor y del ancho de banda, con el consiguiente ahorro económico (consulte un capítulo anterior).
- Reduce el tiempo de diseño y programación. La producción de un sitio como el del ejemplo nos llevó sólo dos horas, y dedicamos gran parte de este tiempo a trabajar en los propios capítulos. (Este ahorro de tiempo se refiere únicamente al proceso de programación. La creación del contenido y los elementos artísticos requieren más tiempo.)
- Reduce las operaciones de actualización y mantenimiento:
  - Los encargados del contenido no tendrán que preocuparse por tablas complejas, etiquetas de fuentes y otros componentes de diseño de la vieja escuela que pueden dividir el texto cuando se actualicen. Como estos elementos ya no existen (o son escasos), no hay nada que se divida.
  - Los diseñadores, programadores y agencias no tendrán que preocuparse por que los clientes dividan el sitio.
  - Los cambios globales se pueden realizar en cuestión de minutos. Si por ejemplo el texto aparece demasiado oscuro, basta con modificar un par de reglas del archivo CSS para que el cambio se refleje inmediatamente en todo el sitio.
- Aumenta la compatibilidad mediante el cumplimiento de las recomendaciones del W3C (estándares Web).
- Aumenta la accesibilidad ya que elimina algunos, muchos o todos los elementos de presentación del marcado.

Anatomía de estilos

En este apartado, presentaremos los elementos básicos de CSS. Este libro no es un extenso manual de referencia sobre CSS. Un manual de referencia sobre CSS superaría el tamaño de este libro, aunque uno de nuestros libros favoritos cabe perfectamente en un bolsillo.

A continuación, comenzaremos a diseccionar la anatomía de los estilos. Aprenderemos más sobre CSS aplicando estos conceptos en un capítulo posterior y, a lo largo de todo el libro, seguiremos hablando de CSS.

Selectores, declaraciones, propiedades y valores

Una hoja de estilo está formada por una o varias reglas que controlan cómo deben representarse los elementos seleccionados. Una regla CSS se divide en dos partes: un selector y una declaración:

```
p    { color: red; }
```

En este regla, p es el selector, mientras que `{ color: red; }`, entre llaves, es la declaración. Las declaraciones, por su parte, están formadas por dos elementos: una propiedad y un valor. En la declaración anterior, `color` es la propiedad y `red` el valor.

Opciones

En lugar de la palabra `red`, podríamos haber utilizado el valor hexadecimal (color Web) `#ff0000`:

```
p    { color: #ff0000; }
```

Habríamos ahorrado algunos bytes utilizando una abreviatura de CSS que tiene el mismo significado:

```
p    { color: #f00; }
```

También podríamos haber utilizado RGB de dos formas:

```
p    { color: rgb(255,0,0); }
p    { color: rgb(100%,0%,0%); }
```

El cero es opcional, excepto cuando no lo es

Al utilizar porcentajes RGB, el signo aparece aunque el valor sea cero. En CSS, esto no sucede siempre. Por ejemplo, al especificar un tamaño de 0 píxeles, no es necesario que detrás del 0 se incluya px.

Muchos consideramos que no es correcto escribir 0px, 0in, 0pt o 0cm. El cero es cero. A nadie le importa la unidad de medida si el valor es cero. Sin embargo, al especificar porcentajes RGB, el valor cero requiere el signo del porcentaje. No nos pregunte, sólo trabajamos aquí. Al igual que la regla que prohíbe utilizar guiones bajos en nombres de clase y de `id`, la insistencia del signo del porcentaje en RGB es un hecho de CSS, cuya lógica, si existe, se ha perdido en las brumas de la historia.

No queríamos comenzar la anatomía de los estilos con quejas sobre incoherencias y excepciones, pero tendrá que soportarlo mientras sigamos utilizando los colores para describir los fundamentos de CSS.

Declaraciones múltiples

Es incorrecto especificar un color sin especificar también un color de fondo y al contrario:

```
p    { color: #f00; background: white; }
```

Se puede utilizar el valor `transparent` para evitar que el fondo de una regla se superponga sobre el de otra:

```
p    { color: #f00; background:
transparent; }
```

Fíjese en que una regla puede estar formada por más de una declaración y que se utiliza punto y coma para separar una declaración de la siguiente.

PELIGRO. FONDOS TRANSPARENTES Y NETSCAPE 4

Le advertimos que, en muchos casos, Netscape 4.x interpreta `transparent` como negro. Si por el contrario utiliza `background: inherit;`, el color será verde. Se debe a que Netscape 4 procesa cadenas no reconocidas de caracteres y las pasa a la fuerza por su analizador hexadecimal incluso aunque claramente se trata de valores hexadecimales no válidos. (IE/Windows puede que hiciera lo mismo en versiones anteriores a la 4.0, pero no estamos seguros.)

Como el valor del color de fondo predeterminado es `transparent` lo indiquemos o no, si su público está formado principalmente por usuarios de Netscape 4, puede que tenga que evitar la declaración del color de fondo. El servicio de validación del W3C emitirá una advertencia si lo hace, pero es preferible una advertencia de validación que un sitio Web desfigurado. Repetimos que sólo debe considerar esta opción si su público está formado esencialmente por usuarios de Netscape 4.x.

La salud del punto y coma

Continuamos con las incoherencias y las excepciones. La última regla de una declaración no puede acabar en punto y coma, y muchos diseñadores eliminan el punto y coma final en series de declaraciones (ya que el punto y coma funciona como separador, no como terminador).

Sin embargo, los autores CSS más experimentados añaden punto y coma al final de todas las declaraciones. Lo hacen en parte por motivos de coherencia, pero principalmente para evitar dolores de cabeza al añadir y eliminar declaraciones de reglas existentes. Si todos los pares propiedad-valor acaban en punto y coma, no tendrá que preocuparse cuando cambie sus declaraciones de un lugar a otro.

Espacio en blanco y diferencia entre mayúsculas y minúsculas

Normalmente, las hojas de estilo suelen incluir más de una regla y la mayoría de las reglas contienen más de una declaración. Si se utilizan espacios en blanco, resulta más sencillo realizar el seguimiento de declaraciones múltiples y modificar las hojas de estilo:

```
body  {
   color: #000;
   background: #fff;
   margin: 0;
   padding: 0;
   font-family: Georgia, Palatino, serif;
   }

p    {
font-size: small;
   }
```

El espacio en blanco, o la ausencia del mismo, no afecta a la forma en que se representa la CSS en los navegadores y, al contrario que XHTML, CSS no distingue entre mayúsculas

y minúsculas. Obviamente, existe una excepción: los nombres de clase y de `id` distinguen entre mayúsculas y minúsculas cuando están asociados a un documento HTML. En este caso, `myText` y `mytext` no coinciden. La propia CSS discrimina entre mayúsculas y minúsculas, pero puede que el lenguaje del documento no.

En http://devedge.netscape.com/viewsource/2001/css-class-id/ encontrará más información al respecto.

Valores alternativos y genéricos

Un diseñador Web puede especificar fuentes para todo un sitio como se indica a continuación:

```
body   {
    font-family: "Lucida Grande",
    Verdana, Lucida, Arial, Helvetica,
    sans-serif;
    }
```

Fíjese en que las fuentes con más de un nombre ("Lucida Grande") se deben incluir entre comillas rectas ASCII y que detrás de la comilla de cierre se añade una coma, no por delante, para enfado de los diseñadores acostumbrados a colocar la coma dentro de las comillas.

Las fuentes se utilizan en el orden en que se enumeran. Si el equipo del usuario contiene la fuente Lucida Grande, el texto utilizará dicha fuente.

En caso contrario, se utilizará Verdana. Si no se encuentra Verdana, se utiliza Lucida. Y así sucesivamente. Se preguntará por qué las fuentes están en orden.

Uso del orden para admitir diferentes plataformas

El orden importa. Lucida Grande puede encontrarse en Mac OS X. Verdana se incluye en todos los sistemas modernos de Windows, en Mac OS X y en sistemas operativos Mac más antiguos. Si se hubiera enumerado primero Verdana, en equipos bajo Mac OS X se utilizaría Verdana en lugar de Lucida Grande.

Con las dos primeras fuentes, Lucida Grande y Verdana, el diseñador ha cubierto las necesidades de la mayoría de los usuarios (de Windows y de Macintosh). Tras éstas, sigue Lucida para usuarios de UNIX y Arial para los de sistemas antiguos de Windows. Helvetica se utiliza en sistemas antiguos de UNIX. Si ninguna de las fuentes enumeradas está disponible, el estilo genérico sans serif garantiza que se utilice cualquier fuente sans serif disponible. En el hipotético caso de que el equipo del usuario no cuente con ninguna fuente sans serif, se utiliza en su lugar la fuente predeterminada del navegador.

No es una ciencia perfecta

No pretendemos afirmar que Lucida, Verdana, Arial y Helvetica son equivalentes en lo que respecta a su belleza, su elegancia, su altura o su idoneidad para utilizarlas en pantalla (o que todas las formas de Helvetica tengan la misma calidad). Nuestro objetivo no es crear experiencias visuales idénticas para todos los usuarios; la plataforma, el navegador, el tamaño del monitor, la resolución, la calidad y otros parámetros hacen que sea imposible. Simplemente tratamos de garantizar que todos los visitantes tengan la

mejor experiencia que sus condiciones le permitan y que dichas experiencias sean lo más parecidas de un usuario a otro.

Selectores agrupados

Cuando varios elementos comparten propiedades estilísticas, se puede aplicar una misma declaración a varios selectores si éstos se agrupan en una lista delimitada por comas:

```
p, td, ul, ol, ul, li, dl, dt, dd    {
      font-size: small;
   }
```

Esta posibilidad resulta muy útil cuando se utilizan navegadores antiguos que no comprenden la herencia CSS.

La herencia y sus descontentos

En función de CSS, las propiedades las heredan elementos secundarios de sus elementos principales. Pero no siempre sucede así. Veamos la siguiente regla:

```
body   {
      font-family: Verdana, sans-serif;
   }
```

Llegados a este punto del libro, sabrá tan bien como nosotros qué significa esta regla. El elemento `body` del sitio utilizará Verdana si dicha fuente se encuentra disponible en el sistema del visitante; en caso contrario, se utilizará una fuente sans-serif genérica.

Secundarios

Con respecto a la herencia, lo que se aplica al elemento de nivel superior (en este caso, al elemento `body`), se aplica a sus secundarios (como por ejemplo `p`, `td`, `ul`, `ol`, `li`, `dl`, `dt` y `dd`). Sin necesidad de añadir reglas adicionales, todos los elementos secundarios del elemento `body` se mostrarán en Verdana o en una fuente sans-serif genérica, así como sus secundarios y los secundarios de sus secundarios. Y es lo que sucede en la mayoría de los navegadores modernos.

Pero no sucede en los agentes de usuario creados durante las guerras de los navegadores, cuando la compatibilidad con los estándares no era una de las prioridades de los fabricantes. Por ejemplo, no sucede en Netscape 4, que ignora la herencia y también ignora las reglas aplicadas al elemento `body`. (IE/Windows, hasta la versión IE6, presenta un problema similar en el que los estilos de fuente se ignoran en las tablas.)

Sea amable con Netscape 4

Afortunadamente, se puede solucionar este problema de los navegadores antiguos si se usa lo que denominamos el principio de redundancia "Sea amable con Netscape 4":

```
body   {
      font-family: Verdana, sans-serif;
   }
p, td, ul, ol, ul, li, dl, dt, dd    {
      font-family: Verdana, sans-serif;
   }
```

Puede que los navegadores 4.0 no entiendan la herencia, pero sí entienden los selectores agrupados. Y todos tendrán su ración de Verdana. Se preguntará si este esfuerzo duplicado (la creación de una regla redundante) supone un gasto de ancho de banda para el usuario. Pues sí. Pero merece la pena utilizarlo.

Por cierto, Netscape 4 no es el único navegador antiguo que desperdicia la herencia.

Solamente es el único que sigue utilizando una minoría leal.

¿La herencia es una maldición?

Imagine que no quiere que los elementos secundarios hereden la fuente Verdana y prefiere que, por ejemplo, los párrafos se muestren en Times. No hay problema. Basta con crear una regla más específica para p (en negrita en el siguiente párrafo) para reemplazar a la regla principal:

```
body   {
       font-family: Verdana, sans-serif;
   }
td, ul, ol, ul, li, dl, dt, dd   {
       font-family: Verdana, sans-serif;
   }
p   {
       font-family: Times, "Times New
       Roman", serif;
   }
```

Y todos tendrán su ración de Times. Vale, no nos repetiremos más.

Selectores contextuales (descendientes)

Podemos evitar la clasitis y conseguir que el marcado sea correcto si hacemos que el estilo de un elemento dependa del contexto en el que aparece. Los selectores que aplican este tipo de reglas se denominaban selectores contextuales en CSS1 ya que dependían del contexto para aplicar una regla. En CSS2 se denominan selectores descendientes, pero sus efectos son los mismos independientemente del nombre que reciban.

Para mostrar en cursiva un texto marcado como y evitar que se muestre en negrita cuando aparezca dentro de un elemento de lista, podríamos utilizar lo siguiente:

```
li strong   {
       font-style: italic;
       font-weight: normal;
   }
```

¿Qué hace una regla como ésta?

```
<p><strong>Soy negrita y no soy cursiva
porque no aparezco en un elemento de
lista. La regla no me afecta.</strong>
</p>
<ol>
<li><strong>Soy cursiva y con un grosor
Roman (normal) porque aparezco dentro de
un elemento de lista.</strong></li>
<li>Soy un texto normal en esta lista.
</li>
</ol>
```

Probemos con la siguiente CSS:

```
strong   {
       color: red;
       }
h2     {
       color: red;
       }
h2 strong   {
       color: blue;
   }
```

y su efecto en el marcado:

```
<p>The strongly emphasized word in this
paragraph is <strong>red</strong>.</p>
<h2>This subhead is also red.</h2>
<h2>The strongly emphasized word in this
subhead is <strong>blue</strong>.</h2>
```

Probablemente no utilice selectores contextuales para poner en cursiva texto de una lista que normalmente aparecería en negrita. Podría utilizar estos selectores para crear sofisticadas mejoras de diseño, como por ejemplo para añadir una imagen de fondo a un elemento XHTML normal, junto con el suficiente espacio en blanco para evitar que el texto se solape sobre la imagen. Pero

podría crear estos efectos con selectores id contextuales o de clase.

Selectores id y selectores id contextuales

En los diseños modernos, los selectores id se suelen utilizar en selectores contextuales:

```
#sidebar p   {
      font-style: italic;
      text-align: right;
      margin-top: 0.5em;
      }
```

El estilo precedente se aplicará sólo a los párrafos que aparezcan en un elemento con el id sidebar. Dicho elemento probablemente será un div o una celda de tabla, aunque también podría ser una tabla o algún otro elemento de nivel de bloque. Incluso podría ser un elemento en línea, como o , aunque estos usos serían realmente extraños y además inválidos, ya que no se puede anidar un elemento <p> dentro de . Independientemente del elemento que utilice, debe ser el único elemento de la página con el id sidebar. En un capítulo anterior encontrará más información al respecto.

Un selector, múltiples usos

Aunque el elemento sidebar sólo aparecerá una vez en cada página del marcado, el selector id se puede utilizar como selector contextual o descendiente tantas veces como sea necesario:

```
#sidebar p   {
      font-style: italic;
      text-align: right;
      margin-top: 0.5em;
      }
#sidebar h2   {
      font-size: 1em;
      font-weight: normal;
      font-style: italic;
      margin: 0;
      line-height: 1.5;
      text-align: right;
      }
```

En este caso, los elementos en sidebar reciben un tratamiento especial distinto al de otros elementos p de la página y los elementos h2 de sidebar reciben un tratamiento especial distinto al de otros elementos h2 de la página.

El selector es independiente

No se debe utilizar el selector id contextualmente. Puede ser independiente:

```
#sidebar   {
      border: 1px dotted #000;
      padding: 10px;
   }
```

En función de esta regla, el elemento de página con el id sidebar tendrá un borde negro de puntos (#000) de un píxel de grosor, con un espacio en blanco de 10 píxeles a su alrededor.

Selectores de clase

La discriminación de clases es algo terrible en la vida real, pero en las hojas de estilo es algo correcto. En CSS, los selectores de clase se indican por medio de un punto:

```
.fancy   {
      color: #f60;
      background: #666;
   }
```

Cualquier elemento de la clase fancy mostrará texto naranja (#f60) sobre un fondo

gris (#666). Por esta razón, `<h1 class="fancy">Boy Howdy!</h1>` y `<p class="fancy">Yee haw!</p>` comparten este chocante esquema de color.

Al igual que `id`, las clases también se pueden utilizar como selectores contextuales:

```
.fancy td   {
      color: #f60;
      background: #666;
   }
```

En el ejemplo anterior, las celdas de tablas incluidas en los elementos de mayor tamaño con el nombre de clase `fancy` mostrarán un texto naranja sobre un fondo gris. (El elemento de mayor tamaño con el nombre `fancy` puede ser una fila de tabla o un `div`.)

También se pueden seleccionar elementos en función de sus clases:

```
td.fancy   {
      color: #f60;
      background: #666;
   }
```

En este ejemplo, las celdas de tabla de la clase `fancy` serán naranjas con un fondo gris.

```
<td class="fancy">
```

Puede asignar la clase `fancy` sólo a una celda de tabla o a todas las que desee. Las etiquetadas de esta forma serán de color naranja con un fondo gris. A las celdas de tabla que no se les asigne la clase `fancy` no se verán afectadas por esta regla. Del mismo modo, un párrafo de la clase `fancy` no será de color naranja con un fondo gris, ni tampoco lo será ningún otro elemento de esta clase. El efecto se limita a las celdas de tabla con la etiqueta `fancy` debido a la forma en que hemos establecido la regla (utilizando el elemento `td` para seleccionar la clase `fancy`).

Combinación de selectores para crear sofisticados efectos de diseño

Los selectores de clase, `id` y contextuales se pueden combinar para crear sutiles o sorprendentes efectos visuales. En el sitio The Marine Center diseñado por Happy Cog, la imagen de marca de un pez (`lister2.gif`) sustituye al aburrido punto blanco que se encuentra en las listas sin ordenar convencionales (véase figura 9.1).

Figura 9.1.

Los selectores de clase, id y contextuales se pueden combinar para crear efectos visuales. En el sitio The Marine Center (http://marine.happycog.com/), el aburrido punto blanco que se encuentra en la mayoría de las listas sin ordenar se ha sustituido por una imagen de un pez.

A continuación se reproduce la regla CSS que indica a las listas sin ordenar de la clase `inventory` que muestren una imagen en lugar de un aburrido punto negro (y que muestren un disco en los navegadores más antiguos que no son capaces de representar la imagen):

```
ul.inventory {
      list-style: disc url(/images/
common/lister2.gif) inside;
   }
```

Y aquí, en forma abreviada para ajustarlo a la página, el marcado sobre el que se basa:

```
<ul class="inventory">
<li><a href="/angelfish">Angelfish</a>
(67 items)</li>
<li><a href="/anglers">Anglers/
Frogfish</a> (35 items)</li>
<li><a href="/anthias">Anthias</a>
(5526 items)</li>
<li><a href="/basslets">Basslets</a>
(15 items)</li>
</ul>
```

Los usuarios de IE/Windows y Opera/Windows obtienen una opción especial: el sitio muestra primero un punto y, tras ello, lo rellena con la imagen del pez.

El efecto es similar al de una animación de Flash o JavaScript, pero es puramente accidental y es el resultado del orden en el que IE para Opera y para Windows carga y muestra los componentes de una página Web. En otros navegadores, los usuarios simplemente ven los peces.

Si ha leído con atención, comprenderá que una lista sin ordenar de una clase diferente no mostrará la imagen del pez, ni tampoco aparecerá si la clase se aplica a cualquier otro elemento que no sea una lista sin ordenar.

Las imágenes del pez también aparecen en la columna de compras del sitio (véase figura 9.2), gracias a un elemento en línea (`span`) de la clase `cartier`:

```
.cartier   {
      padding-left: 5px;
      background: transparent
url(/images/common/cartfish.gif)
no-repeat top left;
   }
```

Figura 9.2.

Las imágenes del pez también aparecen en la columna de compras del sitio. No se han utilizado etiquetas <img> para crear estos efectos. Consulte el texto de explicación para saber cómo se ha conseguido.

Puede estudiar técnicas adicionales si analiza el código fuente que se encuentra en el servidor, en http://marine.happycog.com/, donde se almacenan las plantillas originales del sitio (véase figura 9.3).

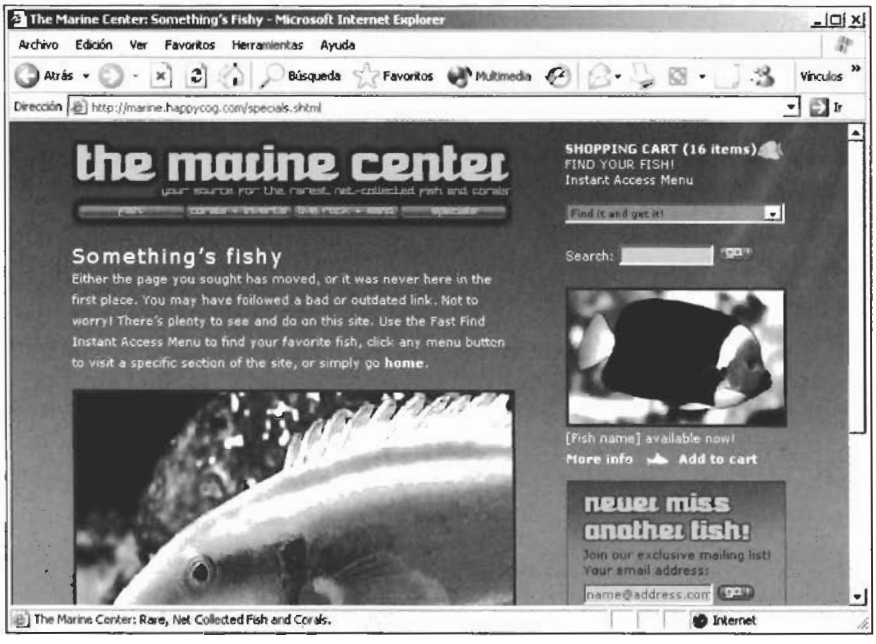


Figura 9.3.

Entre las fotografías de peces y las imágenes de peces generadas por CSS, todo el mundo sabe de qué trata este sitio. Incluso si se produce el error 404, el visitante sigue viendo un marco con peces.

Un paso adelante

En el siguiente capítulo aprenderemos más sobre la gramática CSS pero, por el momento, nos detendremos para responder a una duda: ¿dónde añadimos la CSS? ¿En medio del XHTML? ¿En un archivo separado? (Una pista: siga leyendo.)

Estilos externos, incrustados y en línea

Las hojas de estilo se pueden aplicar a una página Web de tres formas diferentes: externa, incrustada o en línea. Comenzaremos con la mejor.

Hojas de estilo externas

Una hoja de estilo externa (archivo CSS) es un documento de texto que se encuentra separado de las páginas XHTML que controla. Para utilizar dicho archivo CSS, la página XHTML hace referencia a la misma por medio de un vínculo al documento o importándolo en el elemento de estilo (también en el documento). El aspecto de un vínculo de hoja de estilo es:

```
<link rel="StyleSheet" href="/styles/
mystylesheet.css"
type="text/css" media="all" />
```

La directiva `@import`, utilizada para importar una hoja de estilo, tiene el siguiente aspecto:

```
<style type="text/css" media="all">
@import "/styles/mystylesheet.css";
</style>
```

o este:

```
<style type="text/css" media="all">
@import url("/styles/mystylesheet.css");
</style>
```

Mayor potencia, menor coste

Ya sean vinculadas o exportadas, las hojas de estilo externas ofrecen la mayor potencia al menor coste. Una vez descargada una hoja de estilo externa en la caché de un usuario, permanece activa y puede controlar el diseño de una, de varias, de cientos o de cientos de miles de páginas sin tener que volverla a descargar. Algo realmente útil.

Vinculación e importación como selectores de navegador

Prácticamente todos los navegadores compatibles con CSS admiten el método de vinculación, incluso los más antiguos cuya compatibilidad con CSS es mínima. Por el contrario, el método `@import` sólo funciona en navegadores 5.0 y posteriores. Por esta razón, los diseñadores pueden utilizar `@import` para ocultar hojas de estilo a los navegadores 4.0.

No implica necesariamente que se oculte CSS a todos los navegadores, como mencionamos en capítulos anteriores. En realidad, es justo lo contrario. El hecho de que algunos navegadores acepten `@import` y otros no, nos permite servirlos a todos. ¿Qué le parece?

Como se puede utilizar más de una hoja de estilo en un sitio, los diseñadores pueden incluir los estilos básicos en una hoja de estilo vinculada y los estilos sofisticados en una hoja importada. La hoja de estilo básica vinculada, visible para todos los navegadores incluso para los que pretenden comprender CSS, proporciona elementos de diseño básicos como las fuentes utilizadas y el color del texto, del fondo y de los vínculos. La hoja importada, cuyos estilos sólo son visibles para los navegadores más compatibles, contiene reglas CSS adicionales que los nuevos navegadores comprenden y los antiguos no.

El XHTML podría tener este aspecto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/
xhtml">
  <head>
<title>Your Title Here</title>
<link rel="StyleSheet" href="/css/
basic.css" type="text/css" media="all" />
<style type="text/css" media="all">
@import "/css/sophisto.css";
</style>
<meta http-equiv="Content-Type"
content="text/html; charset=
ISO-8859-1" />
</head>
```

Al adoptar este enfoque por capas, se puede admitir a todos los usuarios sin bloquear el contenido a nadie y sin la agonía de la detección de navegadores (analizaremos esta técnica con más detalle en un apartado posterior).

Estas ventajas de ancho de banda reducido para usuarios y servidores, junto con la posibilidad de admitir varios navegadores compatibles, sólo están disponibles si se utilizan hojas de estilo externas. Cuando terminemos el diseño del sitio i3Forum, utilizaremos hojas de estilo externas para aprovechar estas ventajas. No somos tontos.

Hojas de estilo incrustadas

En lugar de vincular o importar uno o varios archivos de hoja de estilo, un diseñador pue-

de incrustar las reglas de la hoja de estilo en el elemento head de una página XHTML 1, utilizando el elemento de estilo que mostramos a continuación (resaltado en negrita):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/
xhtml">
  <head>
<title>i3forum</title>
<style type="text/css">
<!--
body    {
        background: white;
        color: black;
        }
-->
</style>
<meta http-equiv="Content-type"
content="text/html; charset=
iso-8859-1" />
</head>
```

Al contrario que los estilos vinculados o importados, los estilos incrustados no ofrecen ahorro de ancho de banda ya que el usuario debe cargar una nueva hoja de estilo incrustada siempre que abra una nueva página. Incluso si la hoja de estilo incrustada es la misma en todas las páginas, el usuario tendrá que descargarla. Se preguntará por qué un diseñador utilizaría una hoja de estilo incrustada. A continuación enumeramos algunas razones:

- El sitio está formado por una sola página. No hemos dicho que sea habitual, sólo que es posible.

- El público del diseñador vive en un agujero temporal y utiliza IE3 para visitar el sitio. IE3 fue el primer navegador mínimamente compatible con CSS. No admitía hojas de estilo externas. Bueno, si lo pensamos tampoco es una razón muy probable.

- El diseñador ha utilizado hojas de estilo externas para controlar todo el sitio, pero tiene que crear reglas adicionales para una página concreta. Es una razón excelente para crear una hoja de estilo incrustada. De hecho, es la única forma de conseguir determinados efectos visuales. Cuando diseñemos la hoja de estilo de i3Forum en el siguiente capítulo (ya casi llegamos), añadiremos un estilo incrustado a cada página del sitio para controlar la forma en que se resaltan los elementos de menú cuando el usuario cambia de una página a otra.

- El diseñador crea la hoja de estilo y necesita ver inmediatamente el efecto de los cambios aplicados a la misma. Es otra excelente razón para crear una hoja de estilo incrustada.

Al diseñar un sitio (ya sea híbrido o CSS puro), tiene mucho sentido incrustar la hoja de estilo en el elemento <head> de la página en la que estemos trabajando. Tras ello, cuando obtenga el diseño deseado, puede copiar los estilos a un archivo CSS externo y eliminar el estilo incrustado de su marcado.

Durante el proceso detallado en el siguiente capítulo utilizaremos una hoja de estilo incrustada ya que ofrece la forma más sencilla y rápida de realizar el trabajo. (Cambie una regla, vuelva a cargar la página y compruebe si obtiene lo que esperaba.) Cuando nuestra CSS funcione como queremos, la eliminaremos del elemento <head>, la guardaremos en un archivo CSS externo dentro

de un subdirectorio `a/css` en el servidor y estableceremos un vínculo al mismo para ahorrar ancho de banda en el sitio terminado.

Estilos en línea

CSS se puede aplicar a un determinado elemento si se utiliza el atributo de estilo de dicho elemento, como se muestra en el siguiente ejemplo:

```
<h1 style="font-family: verdana, arial,
sans-serif; "> Headline</h1>
<img style="margin-top: 25px;">
```

Como habrá imaginado, los estilos en línea no ahorran ancho de banda; en realidad, añaden ancho de banda a todas las páginas en las que se utilizan y suelen ser tan dañinos como las etiquetas `<font>`.

El estilo de las páginas Web no debe basarse principalmente en CSS en línea. Pero si se utilizan con respeto y discreción, una CSS en línea puede ser una herramienta muy útil. Los estilos en línea son el retoque de las CSS

El método de diseño "El mejor caso posible"

En el pasado, cuando creábamos diseños con marcado de presentación, probábamos nuestro trabajo en el navegador más antiguo que tuviéramos en el disco duro. Para que el aspecto en estos navegadores fuera el correcto, creábamos tablas profundamente anidadas, utilizábamos elementos `div` no estructurales en lugar de elementos estructurales como `h1`, `h2`, `li` y `p`; y todo aquello que hemos dejado de hacer. Cuando conseguíamos que el aspecto del sitio fuera el correcto en el viejo navegador, lo probábamos en un navegador nuevo, donde también tenía un aspecto correcto, pero con gasto terrible de ancho de banda y semántica. Muchos diseñadores Web siguen utilizando esta práctica. Pero el coste es demasiado elevado y el método ya no es productivo.

Por el contrario, puede escribir su CSS en una hoja de estilo incrustada y probarla en un navegador de confianza, como Mozilla, Chimera, Netscape 7, IE6/Windows, IE5/Mac u Opera 7 (por nombrar algunos). De esta forma, creará páginas accesibles, con bajo ancho de banda y compatibles que permiten que el marcado sea marcado y que utilicen CSS correctamente.

Cuando esté satisfecho con lo que ha diseñado, pruebe la página en otros navegadores compatibles. Debería tener el mismo aspecto en todos ellos. En caso contrario, tendrá más trabajo que hacer. Puede que una de sus CSS esté incorrectamente escrita y que su navegador preferido comprenda lo que quiere decir, como cuando nos entienden cuando hablamos con la boca llena.

De estilos incrustados a estilos externos: el método de las dos hojas

Cuando el aspecto operativo y visual del sitio sea el correcto en todos los navegadores compatibles a su disposición, no debe abrir su peor navegador antiguo. Por el contrario, copie sus reglas CSS a un nuevo archivo con el nombre `basic.css`, cargue el archivo en el directorio `/css/` de su servidor, elimine

la hoja de estilo incrustada de la página y vincúlelo al elemento head del documento:

```
<link rel="StyleSheet" href="/css/
basic.css" type="text/css" media="all" />
```

Vacíe la caché, apague y reinicie los navegadores, y pruebe de nuevo para asegurarse de que no ha eliminado por accidente ninguna regla al pasar de CSS incrustadas a CSS externas.

Cómo probar y admitir navegadores no compatibles

Si todavía está contento, llegó la hora de abrir cualquier navegador no compatible que quiera admitir. Existe una remota posibilidad de que el sitio tenga un aspecto correcto en el peor navegador. En caso contrario, tendrá que crear un nuevo documento vacío con el nombre sophisto.css y comenzar a transferir las reglas CSS cuya sofisticación sospeche que afecta al navegador antiguo. Al copiar las reglas sofisticadas al archivo sophisto.css, elimínelas de basic.css. (Evidentemente, puede usar cualquier nombre. Hemos empleado basic y sophisto, pero puede servir cualquier otro.)

Cargue sophisto.css en su subdirectorio /css/ y vincule esta segunda hoja de estilo externa por medio de @import:

```
<style type="text/css"
media="all">@import "/css/sophisto.css";
</style>
```

Vacíe la caché del navegador antiguo, vuelva a cargar la página y compruebe si se muestra de forma aceptable. Puede que lo haga. En caso contrario, tendrá que cambiar alguna regla más de basic.css a sophisto.css.

Llegará un momento en que su sitio se vea perfectamente en los navegadores compatibles y tenga un aspecto atractivo en las reliquias anteriores a los estándares.

Por ejemplo, el sitio Fox Searchlight (www.foxsearchlight.com), diseñado por Hillman Curtis en colaboración con Happy Cog, utiliza el método de las dos hojas para presentar el aspecto operativo y visual del sitio en todos los navegadores, a la vez que oculta los estilos problemáticos a los navegadores 4.0. En uno de estos navegadores, el sitio es menos compacto, pero su aspecto es correcto y se puede utilizar. Al igual que el método de diseño del apartado anterior nos ahorra tener que crear código según las limitaciones del mínimo denominador común, el método de las dos hojas nos evita tener que excluir el menor denominador común. (Visite foxsearchlight.com y compárelo en varios navegadores.)

Referencias de archivo absolutas y relativas

Al crear la CSS inicial, utilizamos referencias de archivo relativas ya que estábamos trabajando en nuestro escritorio, no en un servidor. La hoja de estilo final que crearemos en el siguiente capítulo utilizará referencias de archivo absolutas (no relativas) para evitar problemas en navegadores antiguos incapaces de comprender los vínculos relativos en CSS.

El hecho de que en navegadores antiguos se produzcan problemas con los vínculos relativos de los archivos CSS es otra buena razón para probar nuestro trabajo en navegadores

fiables en lugar de hacerlo en otros más antiguos. Si se emplea el método antiguo (es decir, primero se prueba en el peor navegador), si ha diseñado las páginas sin conexión y, consecuentemente, ha utilizado referencias de archivo relativas, el navegador arruinará dichas referencias, las imágenes no aparecerán y puede que tarde horas en descubrir qué es lo que falla en la hoja de estilo, cuando en realidad no falla nada.

Ventajas de ambos métodos

El método de diseño "El mejor caso posible" nos evita tener que crear código según las limitaciones del mínimo denominador común. Podemos crear un diseño comercial, correcto y compatible sin necesidad de escribir toneladas de marcado sobrante.

El método de las dos hojas de estilo evita tener que excluir el mínimo denominador común o que los usuarios de los navegadores menos comunes no tengan nada que mirar en los mismos. Ofrecemos a estos usuarios la mejor experiencia visual posible en estas limitadas circunstancias, no les decimos que su navegador es inservible y no les odiamos por obligarnos a deformar nuestro XHTML y nuestras CSS (porque no hemos hecho nada de eso).

No se trata de un marcado semántico, ni de un diseño CSS puro, no es la vanguardia de los estándares Web y tampoco la única forma de diseñar un sitio moderno. Pero es una buena técnica para introducir las ventajas de los estándares en un mercado en transición. Como Klatuu dijo a los habitantes de la Tierra "Puede que no sea perfecto, pero es un sistema y funciona".

El método de las dos hojas no se limita a los diseños híbridos. Puede resultar igual de eficaz a la hora de comunicar diseños CSS puros a diferentes generaciones de navegadores compatibles. En cualquier caso, el objetivo es el mismo: crear una experiencia completa para cada usuario, hasta el punto que permita su navegador o dispositivo.

Ya estamos preparados para utilizar los conocimientos adquiridos sobre CSS y para completar el diseño del sitio que comenzamos en un capítulo anterior. En el siguiente, eso es lo que haremos precisamente. Acompáñenos.

Capítulo 10
CSS en acción: un diseño híbrido (parte II)

En un capítulo anterior creamos el marcado híbrido para el sitio i3Forum y combinamos elementos estructurales como `<h1>`, `<h2>` y `<p>` con componentes no estructurales (tablas XHTML utilizadas para definir la rejilla básica).

También utilizamos resúmenes de tablas, `accesskey` y un vínculo Skip Navigation para que el sitio fuera más accesible en entornos de navegación no tradicionales.

En este capítulo, completaremos nuestra labor de producción y aplicaremos CSS para obtener efectos de diseño que complementen la marca y aumenten el atractivo del sitio sin depender de texto GIF, imágenes cambiantes JavaScript, imágenes GIF con píxeles de separación, construcciones de celdas anidadas profundamente u otros vestigios de diseño Web de la vieja escuela.

En la figura 10.1 puede ver la plantilla de la página de inicio tal y como aparece tras la primera pasada con una hoja de estilo. Como ocurre con todo el diseño, el uso de CSS es un proceso iterativo. En este capítulo, completaremos nuestra CSS en dos pasadas. La primera se encarga del 90 por ciento de lo necesario, mientras que la segunda permite corregir errores y añadir los retoques finales.

Preparación de las imágenes

Aunque el sitio se ha diseñado en Photoshop, no se trata del típico trabajo de cortar y trocear. En la figura 10.2 puede ver las seis imágenes utilizadas para crear la totalidad del sitio. Tres de ellas se utilizarán en la parte frontal: la fotografía del astronauta, la del perro y el GIF transparente del logotipo

que aparecerá en la esquina superior izquierda de la barra de menú.

Las tres imágenes restantes se usarán como fondos. Arrof.gif es una imagen derivada del logotipo que se ubicará en el fondo como marca de agua. Bgpat.gif (véase figura 10.3), formada por píxeles de un solo color alternados con píxeles transparentes, se utilizará para crear efectos de color de fondo en la barra de menú. Nopat.gif, un fondo de color blanco, que sustituye a bgpat.gif en los efectos de imagen cambiante CSS y que también se puede utilizar para indicar la posición del visitante dentro de la jerarquía del sitio por medio de un estilo incrustado en cada página al final del proyecto (como veremos a continuación).

Establecimiento de parámetros básicos

Una vez ubicadas las imágenes y con nuestra composición original como guía, podemos comenzar a utilizar CSS para establecer parámetros de diseño básicos. Sabemos que el fondo del sitio será blanco, que el texto utilizará diferentes tamaños de Georgia (o una fuente serif equivalente), que en la parte inferior de la zona de contenido se añadirá una marca de agua basada en el logotipo y que será necesario reforzar determinados valores de espacio en blanco por medio de imágenes GIF de espaciado o elementos sobrantes adicionales. Hagámoslo realidad.

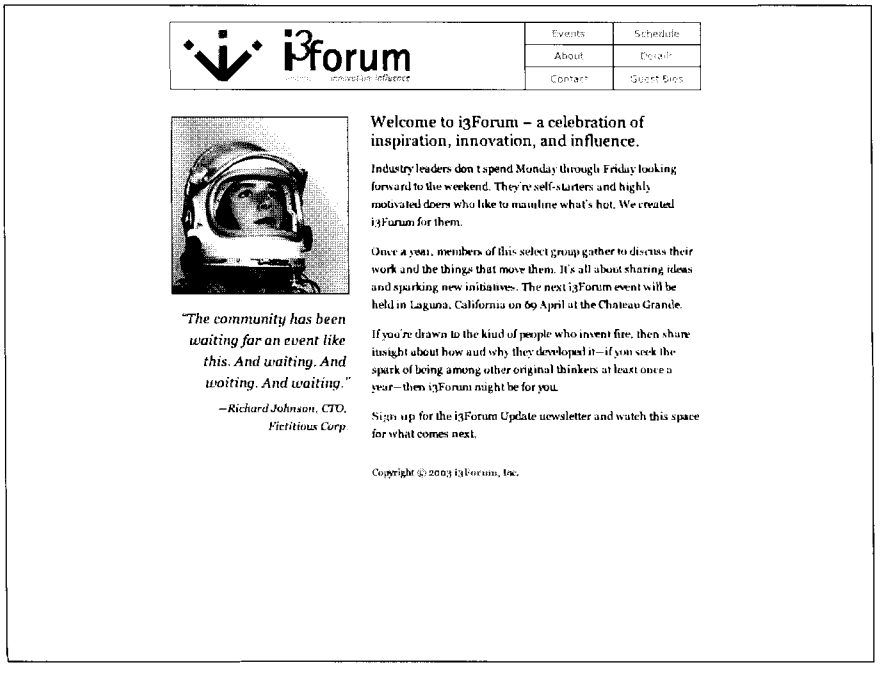


Figura 10.1.

La plantilla tal y como aparece en nuestra primera pasada con CSS. Se añaden los elementos, los tamaños, las fuentes y los colores, pero los fondos no consiguen ocupar los botones de menú del lado derecho. Se necesita un poco más de trabajo.

La imagen innecesaria

Estrictamente hablando, una de nuestras seis imágenes no es necesaria. Nopat.gif, la imagen del fondo blanco, es superflua (véase figura 10.2). Una regla CSS que especificara `background: white` tendría el mismo efecto (y luego utilizaremos una regla como ésta en vez de nopat.gif).

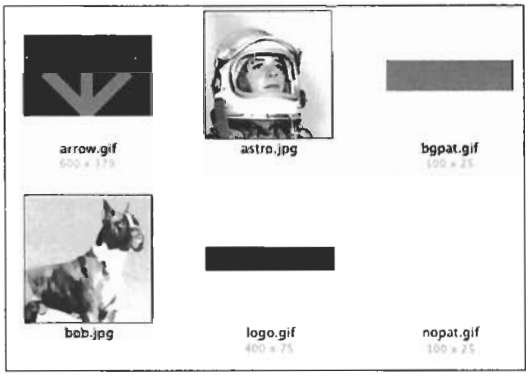


Figura 10.2.

Sólo se han utilizado seis imágenes en este sitio, tres de ellas como fondos. Las técnicas de cortar y trocear en Photoshop e ImageReady no son necesarias.

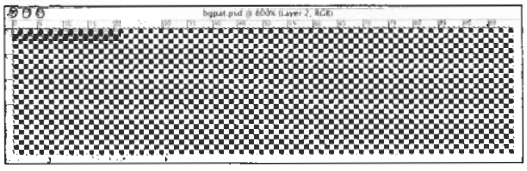


Figura 10.3.

La imagen GIF de fondo con píxeles alternos aumentada un 800 por ciento y con un fondo negro, añadido para los objetivos de este libro, para que resalte sobre los píxeles transparentes.

Sin embargo, para la barra de navegación, hemos utilizado este fondo, para que vea cómo se consigue un cambio de imagen de fondo en CSS.

En una ejecución que implique el uso de fondos de marca de agua intercambiados o con texturas, se necesitan dos imágenes. Por esta razón, será necesario escribir el tipo de reglas CSS que describiremos en esta parte del capítulo.

Estilos generales, más sobre abreviaturas y márgenes

En nuestra primera regla, definimos los colores básicos de la página y los márgenes superior e inferior:

```
body    {
        color: #000;
        background: #fff;
        margin: 25px 0;
        padding: 0;
        }
```

Según esta regla, todo el texto será de color negro (`#000`) sobre un fondo blanco (`#fff`). Los colores se describen en una abreviatura de CSS, como explicamos en el capítulo anterior (`#000` es la abreviatura de `#000000`; `#fff` equivale a `#ffffff`). Sólo se pueden utilizar estas abreviaturas para reemplazar pares de caracteres; así, `#fc0` es lo mismo que `#ffcc00`. No se pueden emplear en ausencia de pares de caracteres. Por ejemplo, no existe una abreviatura para un color que no sea seguro con la Web, como `#f93C7a`.

Abreviaturas y relojes

La primera regla también establece un margen superior e inferior de 25 px, así como márgenes izquierdos y derechos de 0 píxeles. Es una versión abreviada de lo siguiente:

```
margin: 25px 0 25px 0;
```

A su vez, este ejemplo es una versión abreviada de:

```
margin-top: 25px;
margin-right: 0;
margin-bottom: 25px;
margin-left: 0;
```

En CSS, los valores se asignan en el mismo orden que las cifras principales de un reloj: 12 en punto (margen superior), 3 en punto (margen derecho), 6 en punto (margen inferior) y 9 en punto (margen izquierdo). Si queremos que nuestra página tenga un margen superior de 25 píxeles, un margen derecho de 5, un margen inferior de 10 y un margen izquierdo de un 30 por ciento de la anchura total, la regla tendría este aspecto:

```
margin: 25px 5px 10px 30%;
```

Cuando los márgenes verticales son idénticos al inferior y al superior (como ocurre en este sitio, aproximadamente de 25 píxeles) y cuando los márgenes verticales coinciden con el izquierdo y el derecho (como sucede en este sitio, aproximadamente 0 píxeles), podemos ahorrar algunos bytes de ancho de banda del usuario si utilizamos una construcción como:

```
margin: 25px 0;
```

Como mencionamos en un capítulo anterior, el valor 0 no requiere una unidad de medida. 0px es igual que 0 cm ó 0 in.

Por último, nuestra regla define el relleno con el valor 0 para poder utilizar Opera, que utiliza relleno en lugar de márgenes para forzar los bordes de la página.

Cómo ocultar y bloquear

En el siguiente paso, con dos sencillas reglas, conseguimos varios objetivos útiles:

```
.hide   {
        display: none;
        }

img     {
        display: block;
        border: 0;
        }
```

La primera regla crea una clase denominada `hide` que se puede utilizar para que elementos y objetos sean invisibles en navegadores compatibles con CSS. Como explicamos en un capítulo anterior, utilizaremos esta característica para ocultar nuestro vínculo Skip Navigation en los navegadores modernos a la vez que haremos que esté disponible para usuarios de lectores de pantalla, navegadores de texto y navegadores basados en PDA y en teléfonos no compatibles con CSS. (Al cierre de la edición de este libro, algunos navegadores basados en PDA admitían CSS parcialmente, pero de forma tan pobre como lo hacían los navegadores de escritorio 3.0 y 4.0. Afortunadamente, muchas de estas implementaciones habrán mejorado cuando este libro llegue a sus manos.)

La regla de las imágenes

La segunda regla resulta más útil y menos problemática. En primer lugar, `display: block;` indica que todas las imágenes de la

página se representarán como elementos de nivel de bloque en lugar de elementos en línea. Si no está familiarizado con estos términos, le ofrecemos dos sencillos ejemplos. Los párrafos son elementos de nivel de bloque; la obsoleta etiqueta `<i>` (cursiva) es un elemento en línea. Los elementos de nivel de bloque existen en su propio cuadro y están seguidos por un retorno del carro. Los elementos en línea forman parte del flujo, sin retornos del carro ni cuadros propios.

Al indicar al navegador que procese las imágenes como elementos de nivel de bloque, evitamos tener que escribir `<br>` o bien `<br clear="all">`, o basura similar antes y después de nuestras imágenes, y también evitamos tener que incluir dichas imágenes en sus propias celdas de tabla para proteger los requisitos de espacio de nuestro diseño. (En un capítulo posterior encontrará más información al respecto pero, si no puede esperar, puede visitar la dirección http://devedge.netscape.com/viewsource/2002/img-table/.)

Parece todo tan sencillo que muchos lo pasarán por alto sin pararse a pensar en lo que están haciendo, pero la asignación explícita de un estado `inline` o `block` a un elemento es una herramienta increíblemente potente. Los vínculos ordinarios, al convertirse en elementos de nivel de bloque por medio de CSS, se pueden transformar en botones, por ejemplo. En una regla que veremos más adelante, mediante el uso de un selector adicional podremos añadir espacio en blanco vertical específico a las imágenes que se encuentren en una determinada zona del diseño, con lo que por medio de unas líneas de CSS conseguiremos lo que de otro modo requeriría un marcado repleto de cortes, trozos, celdas anidadas e imágenes GIF de espaciado.

Vuelva a leerlo. Queremos enseñarle a conseguir diseños que parezcan estar formados por 50 celdas de tabla y docenas de imágenes cortadas pero que simplemente utilizan algunas líneas de marcado y un par de reglas CSS. Vale. Ya lo hemos dicho.

Tras ello, la declaración `border: 0;` desactiva los bordes de las imágenes, por lo que no será necesario escribir `border="0"` en nuestro marcado. (Si nos preocupara el aspecto del sitio en navegadores incompatibles con CSS, tendríamos que escribir `border="0"` en nuestro marcado. Lo hicimos y, en un capítulo anterior, explicamos las razones por lo que lo hicimos.)

Aplicación de color a los vínculos (presentación de las pseudoclases)

En HTML de presentación, controlamos los colores de los vínculos por medio de atributos del elemento `body` como `vlink="#CC3300"`. En el diseño Web moderno, nuestro elemento `body` puede estar desprovisto de adornos y, en su lugar, podemos recurrir a CSS. Para hacerlo todo más sencillo, CSS añade un estado especial al vínculo, relacionado con los estados visitado y activo que aprendimos en los 90. CSS también le permite hacer algo más que simplemente cambiar los colores de un vínculo.

CSS denomina a estos estados de ancla selectores de pseudoclases. Según la forma de pensar de CSS, una clase real es la que se especifica de forma explícita con un atributo

class=. Una pseudoclase es la que depende de las acciones del usuario o del estado del navegador (:hover, :visited). También existen pseudoelementos (:before y :after). En cualquier caso, con las cuatro reglas que mostramos a continuación, podemos controlar los colores de los vínculos y mucho más (véanse figuras 10.4 y 10.5).

```
a:link {
      font-weight : bold;
      text-decoration : none;
      color: #c30;
      background: transparent;
      }
a:visited {
      font-weight : bold;
      text-decoration : none;
      color: #c30;
      background: transparent;
      }
a:hover {
      font-weight : bold;
      text-decoration : underline;
      color: #f60;
      background: transparent;
      }
a:active {
      font-weight : bold;
      text-decoration : none;
      color: #f90;
      background: transparent;
      }
```

Datos adicionales sobre vínculos y selectores de pseudoclase

En las cuatro reglas anteriores, lo único que puede parecerle nuevo es la propiedad text-decoration. Cuando su valor es none, no se utiliza subrayado. Cuando el valor es underline, lo ha adivinado, el vínculo aparece subrayado, como sucedía con todos los vínculos a mediados de los 90. Cuando el valor es overline, aparece una línea sobre el texto en lugar de por debajo del mismo.

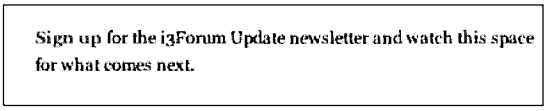


Figura 10.4.

El color del vínculo es rojo oscuro, en negrita y sin subrayado, de acuerdo a la regla CSS a:link.

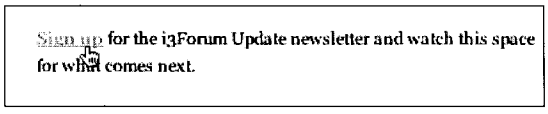


Figura 10.5.

Cuando el ratón del usuario se desplaza sobre el vínculo, su color se ilumina en naranja y se subraya, de acuerdo a la regla CSS a:hover.

Puede combinar líneas por encima y por debajo de un texto, como se indica a continuación:

```
text-decoration: underline overline;
```

Se preguntará que aspecto tendría esto. Sería similar al texto vinculado dentro de un cuadro con parte superior e inferior pero sin lados. Sabemos de dos diseñadores Web, uno de ellos somos nosotros, que han utilizado este efecto al menos en una ocasión. Conviene mencionar dos aspectos más antes de que abandonemos la tierra de los vínculos.

El uso de LVHA

Fíjese en esto. Algunos navegadores ignorarán una o varias reglas de pseudoclases de elementos de ancla si éstas no aparecen en el orden especificado anteriormente: vínculo, visitado, sobre y activo (en inglés, LVHA). Si

cambia el orden se tendrá que atener a las consecuencias. En la dirección http://www.meyerweb.com/eric/css/link-specificity.html encontrará una explicación detallada de por qué este orden tiene su importancia.

Pseudotrampas en IE/Windows

Debe saber que incluso en su última y mejor encarnación (al menos al cierre de la edición de este libro), Internet Explorer para Windows presentaba algunos problemas con las pseudoclase `hover` y `active`. Los estados `hover` tienden a bloquearse. Utilice el botón **Atrás** en su copia de Internet Explorer para Windows y es muy probable que el último vínculo sobre el que haya pasado el ratón continúe en este estado. Al mismo tiempo, es muy probable que descubra que el vínculo sobre el que ha pulsado para avanzar siga en su estado activo. Probablemente no le haga mucha gracia.

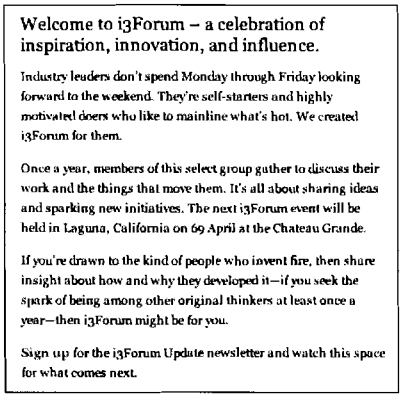
Como el color activo también se bloquea, si aplica imágenes de fondo a la pseudoclase `a:active`, IE/Windows las reproducirá erróneamente. Si entre su público se encuentran usuarios de IE/Windows (¿y cuándo no?), puede optar por evitar la clase `a:active`. O también puede optar, como hemos hecho nosotros, por no hacer nada creativo al respecto.

Si este bloqueo de los estados de vínculo es un error o algo útil, depende de a quién se lo pregunte. Probablemente haya doscientos millones de usuarios de IE/Windows que ya estén acostumbrados y que piensen que la Web funciona de esta forma.

Introducción de otros elementos comunes

En el siguiente bloque de código, vemos que nuestra vieja amiga, la regla "Sea amable con Netscape 4" (consulte el capítulo anterior), se ha utilizado para indicar al navegador que todo el sitio debe emplear la fuente Georgia o una fuente sans-serif alternativa (véase figura 10.6).

```
p, td, li, ul, ol, h1, h2, h3, h4, h5,
h6   {
        font-family: Georgia, "New
Century Schoolbook", Times, serif;
        }
```


Welcome to i3Forum – a celebration of inspiration, innovation, and influence.

Industry leaders don't spend Monday through Friday looking forward to the weekend. They're self-starters and highly motivated doers who like to mainline what's hot. We created i3Forum for them.

Once a year, members of this select group gather to discuss their work and the things that move them. It's all about sharing ideas and sparking new initiatives. The next i3Forum event will be held in Laguna, California on 69 April at the Chateau Grande.

If you're drawn to the kind of people who invent fire, then share insight about how and why they developed it—if you seek the spark of being among other original thinkers at least once a year—then i3Forum might be for you.

Sign up for the i3Forum Update newsletter and watch this space for what comes next.

Figura 10.6.

Las fuentes se especifican con una sola regla aplicada a varios selectores (p, td, li, ul, ol, h1, h2, h3, h4, h5 y h6). El tamaño y el espacio en blanco se controlan por medio de reglas y selectores adicionales.

Georgia, una fuente de pantalla de Microsoft diseñada por el ganador de la medalla Type Directors Club (TDC), Matthew Carter, por ser legible incluso a tamaño reducido, se encuentra en la práctica totalidad de los sistemas de Windows y Macintosh. New Century Schoolbook se incluye en la mayoría

de sistemas UNIX y Times se puede encontrar en sistemas informáticos paleolíticos. Si todo lo demás falla, siempre podemos acudir a una fuente serif genérica.

En la siguiente regla, indicamos al navegador que los títulos deben tener un tamaño superior al tamaño de fuente predeterminado del usuario. Cuando no se especifica un tamaño de fuente base, el navegador considera que el tamaño de fuente predeterminado del usuario es de 1 eme. (No importa si el tamaño de fuente predeterminado del usuario es de 12 o 48 píxeles. Seguirá siendo 1 eme.) Para que el título sea mayor que el tamaño predeterminado, lo definiremos como 1,15 eme. También insistiremos en que el título tenga un grosor normal (no en negrita):

```
h1    {
      font-size: 1.15em;
      font-weight: normal;
      }
```

No es necesario indicar al navegador que `h1` debe definirse con la fuente Georgia. Ya lo hicimos en la regla anterior.

A continuación, utilizamos el selector `html` para añadir más detalle a nuestro estilo `p`. Todos los elementos de la página, a excepción del propio `html`, son elementos secundarios de `html`. Podríamos haber escrito `p` en lugar de `html p`, pero queríamos que sintiera que había hecho una buena inversión con este libro. Veamos la regla, seguida por las pertinentes explicaciones:

```
html p    {
          margin-top: 0;
          margin-bottom: 1em;
          text-align: left;
          font-size:  0.85em;
          line-height: 1.5;
          }
```

En este caso, indicamos que los párrafos no tienen espacio en blanco en la parte superior (lo que nos permite reducir los títulos y los subtítulos), 1 eme de espacio en blanco en la parte inferior (lo que evita que se solapen unos con otros) y que son ligeramente más reducidos (0.85 eme) que el tamaño de fuente predeterminado del usuario (utilizando el mismo razonamiento aplicado a `h1` anteriormente, pero en dirección inversa).

Información adicional sobre tamaños de fuente

Resulta algo complicado especificar tamaños de fuente relativos como hemos hecho en la regla anterior. En concreto, resulta complicado especificar que las fuentes deben ser ligeramente más pequeñas que el tamaño predeterminado del usuario ya que éste puede utilizar un tamaño predeterminado especialmente reducido. De ser así, puede que el texto sea demasiado pequeño.

Por ejemplo, si el usuario ha especificado Verdana de 11 píxeles como fuente predeterminada, nuestro texto reducido puede tener un tamaño de 9 o 10 píxeles, algo incómodo para leer extensos pasajes de texto. El usuario que sufra este aspecto puede ajustar el diseño si cambia el tamaño de fuente de su navegador, pero a algunos usuarios les resulta molesto y algunos ni siquiera saben cómo cambiar el tamaño de un texto.

En la mayoría de las configuraciones originales de los sistemas, los tamaños de fuente predeterminados son tan exagerados como la parte menos atractiva de un caballo, y un tamaño de 0.85 eme puede resultar más que perfecto. Si el usuario tiene defectos visuales

y define un tamaño de texto mucho mayor que el predeterminado, la fuente seguirá pareciéndole correcta y verá que es ligeramente inferior de lo normal. Pero si un usuario de Windows selecciona un tamaño pequeño como predeterminado para su navegador o si un usuario de Mac configura su navegador a 12px/72ppp, puede que nuestro texto se vea demasiado pequeño, lo que provocará el enfado del usuario o, lo más probable, que salga del sitio frustrado y furioso.

Por otra parte, podríamos haber utilizado un valor de píxeles para el tamaño de nuestro texto:

```
font-size: 13px;
```

Podríamos haberlo hecho y crear también el interlineado por medio de una abreviatura CSS:

```
font: 13px/1.5 Georgia,"New Century
Schoolbook", Times, serif;
```

Al contrario que los tamaños relativos basados en emes, los tamaños basados en píxeles son un 99'9 por ciento dependientes en todos los navegadores y plataformas. Y si un tamaño basado en píxeles es demasiado pequeño para un determinado usuario, lo puede ajustar por medio de las herramientas de zoom de texto o de página de cualquier navegador moderno del mundo menos en uno. Por desgracia, dicho navegador es IE/Windows, el navegador más utilizado en la actualidad.

Esto significa que si utilizamos píxeles para controlar con seguridad el tamaño de las fuentes, nos arriesgamos a que el texto resulte inaccesible para los usuarios de IE/Windows con defectos visuales. Pero si intenta usar emes para solucionar este problema, como hemos hecho en nuestro sitio, frustrará a los visitantes que han reducido su preferencia de tamaño de fuente predeterminado para compensar el hecho de que el valor predeterminado de fábrica resulta excesivo para la mayoría de los humanos.

En definitiva, independientemente de lo que haga, siempre habrá alguien frustrado. Una vez diseñamos un sitio sin configurar ningún tamaño de fuente. Pensamos que todos los usuarios estarían contentos al fin. Pero, por el contrario, recibimos cientos de cartas y quejas de que el tamaño del texto era demasiado grande. En un capítulo posterior encontrará más información sobre las penas y glorias del tamaño de las fuentes.

La maravillosa altura de línea

Fíjese de nuevo en la declaración de altura de líneas en la regla que estamos analizando:

```
line-height: 1.5;
```

`Line-height` es un término de CSS para denominar al interlineado. Una altura de línea de 1.5 es lo mismo que un espaciado del 150 por ciento. Se podría haber marcado como 1.5 eme, pero no era necesario.

Antes de CSS, sólo podíamos simular el interlineado si utilizábamos la etiqueta de párrafo de forma no estructural (como comentamos en un capítulo anterior), si utilizábamos `<pre>` o si añadíamos imágenes GIF de píxeles de espaciado entre todas las líneas de texto, forzando el texto en celdas de tabla de anchos absolutos y rezando para que las imágenes de píxel de espaciado

invisibles se descargaran correctamente. En caso contrario, el visitante vería imágenes GIF de marcado de posición divididas, en lugar del maravilloso interlineado.

Pero no merece la pena ni siquiera pensar en ello. CSS resuelve este problema para siempre.

Alineación a la izquierda

Por último, se preguntará por qué hemos especificado la alineación del texto a la izquierda en la regla anterior. La respuesta es sencilla. Si no lo hubiéramos hecho, IE6/Windows podría centrar el texto debido a un problema interno. IE5/Windows no padece este defecto, ni ningún otro navegador. Este comportamiento parece ser aleatorio. Muchos elementos que no se han alineado específicamente a la izquierda en CSS se mostrarán correctamente en IE6/Windows. Pero otros no lo harán. Y nunca se puede saber cuáles serán. Si utiliza la alineación a la izquierda podrá evitar este problema.

Configuración del pie de página

Llegados a este punto, ya se habrá familiarizado con la jerga de CSS y entenderá la siguiente regla:

```
#footer p   {
      font-size: 11px;
      margin-top: 25px;
      }
```

Sin necesidad de que le digamos que utiliza el `id` exclusivo `footer` como selector y que cualquier párrafo dentro de `footer` se definirá con un tamaño de fuente de 11 píxeles y tendrá 25 píxeles de espacio en blanco por encima.

Tampoco le recordaremos que el navegador sabe qué fuente debe utilizar ya que lo hemos establecido en una regla anterior.

Disposición de la división de las páginas

El siguiente conjunto de reglas establece las divisiones básicas de las páginas. Las hemos incluido de forma conjunta en nuestro archivo CSS para que las labores de modificación y de cambio de diseño resulten más sencillas, y las hemos precedido de un comentario que nos recuerde, o que explique a un colega que tenga que modificar posteriormente nuestra hoja de estilo, para qué sirve dicho conjunto de reglas. Si está familiarizado con los comentarios de HTML, comprobará que es muy similar, pero con una convención ligeramente distinta basada en la programación en C.

Tras el comentario, incluimos una regla que establece que la zona de contenido principal tendrá 25 píxeles de espacio en blanco a la izquierda y en la parte superior (véase figura 10.7), y otra regla que añade una imagen de fondo no repetitiva (`arrow.gif`) en la parte inferior y en el centro de la tabla que tenga el `id content` (véase figura 10.8).

Conseguir este efecto por medio del obsoleto atributo de imagen de fondo de la etiqueta de tabla de celda resultaría difícil, por no decir imposible. Por un lado, la imagen de fondo tendría que cubrir las dos celdas de tabla. Por ello, tendríamos que dividir la imagen de fondo en fragmentos, asignar cada uno de los fragmentos a una tabla de celda diferente y rezar para que todos los fragmentos se alinearan.

Figura 10.7.

El espaciado vertical entre la zona de navegación y el contenido del cuerpo, y el espacio en blanco horizontal entre la zona de la fotografía de la barra lateral y el texto del cuerpo se controlan por una misma regla que se aplica al selector de contenido principal. No se necesitan imágenes GIF de espaciado ni celdas de tablas.

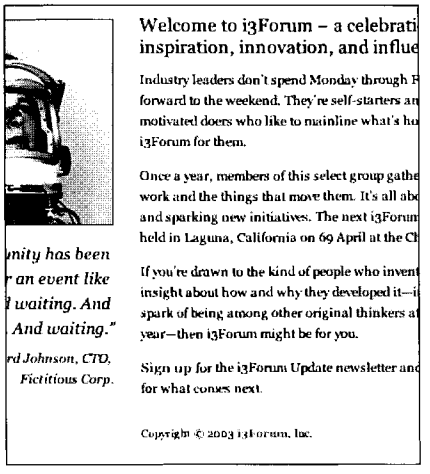


Figura 10.8.

Una imagen de una flecha, derivada del logotipo, ancla el área de contenido y actúa como su marca de agua gracias a una declaración de fondo aplicada al selector de contenido. Como el selector engloba a toda la tabla, la flecha puede cubrir las dos celdas de tabla (la barra lateral y la zona de contenido principal). Un efecto tan sencillo resultaría muy difícil de conseguir, sino imposible, con métodos de la vieja escuela.

Por otra parte, la obsoleta etiqueta de imagen de fondo en HTML adopta, de manera predeterminada, forma de mosaico, y no se puede evitar. Tendríamos que utilizar dos imágenes transparentes de la misma altura que las celdas de tabla en las que se incluyen y rezar para que el usuario no cambie el tamaño del texto, ya que anularía la alineación de la altura de dichas celdas.

También podríamos haber aplicado la misma altura a todas las páginas, lo que limitaría la cantidad de texto que el cliente puede añadir o eliminar de las mismas. Nuestro cliente no estaría muy de acuerdo.

Con CSS, no tendremos que volver a preocuparnos de este aspecto. La lectura de las reglas le llevará mucho menos tiempo del que ha tardado en leer estos párrafos:

```
/* Divisiones de página básicas */
#primarycontent    {
    padding-left: 25px;
    padding-top: 25px;
    }
#content {
    background: transparent url(images/
arrow.gif)center bottom no-repeat;
    }
```

Seguidamente, definimos reglas para la barra lateral:

```
/* Atributos de representación de la
barra lateral */
#sidebar p    {
    font-style: italic;
    text-align: right;
    margin-top: 0.5em;
    }
#sidebar img    {
    margin: 30px 0 15px 0;
    }
#sidebar h2    {
    font-size: 1em;
    font-weight: normal;
    font-style: italic;
```

```
margin: 0;
line-height: 1.5;
text-align: right;
}
```

La primera regla indica que los párrafos dentro del elemento con el `id` exclusivo `sidebar` se alinearán a la derecha, se mostrarán en cursiva y tendrán un margen superior (espacio en blanco) y la mitad de la altura del tamaño de fuente (0.5 eme). Si le resulta familiar es porque lo incluimos en la descripción del capítulo anterior sobre selectores `id` en CSS.

La segunda regla indica que las imágenes dentro del elemento con el `id` exclusivo `sidebar` tendrán un margen superior de 30 píxeles, un margen inferior de 15 píxeles y no tendrán espacio en blanco adicional a los lados (véase figura 10.9). Anteriormente habíamos hecho referencia a esta regla, en el apartado sobre la regla de las imágenes. Ya hemos llegado a ella.

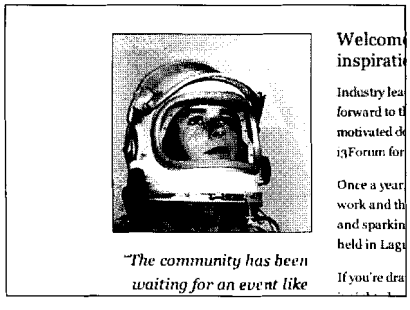


Figura 10.9.

Los cuidadosamente seleccionados valores de espacio en blanco vertical por encima y por debajo de la fotografía de la barra lateral se consiguen al aplicar valores de margen superior e inferior a #sidebar img. Las imágenes dentro del div sidebar obedecen a estos valores de espacio en blanco; el resto de imágenes no.

Reglas como ésta hacen que merezca la pena vivir ya que nos liberan de la necesidad de utilizar múltiples celdas vacías e imágenes GIF de espaciado para crear los espacios en blanco.

La tercera regla de este apartado aplica a los títulos `h2` un aspecto similar a las citas de una revista (especialmente en entornos textuales como Windows Cleartype y Mac OS X Quartz) en lugar de parecer títulos HTML (véase figura 10.10). Especifica que el texto `h2` dentro de `sidebar` tendrá un tamaño de 1 eme, un grosor normal, en cursiva y alineado a la derecha, y que tendrá el mismo valor de altura de línea (1.5) que el texto restante de la página.

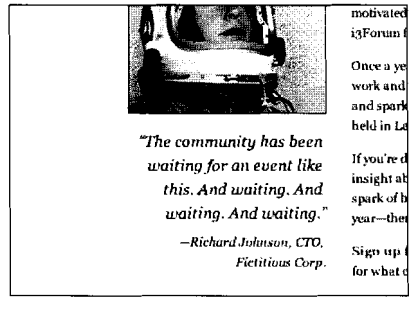


Figura 10.10.

Citas de la barra lateral, marcadas como títulos de segundo nivel (h2); sin embargo parecen citas de revista, no títulos HTML.

Elementos de navegación: primera pasada

Hasta el momento, lo hemos hecho todo correctamente. Todas las reglas que hemos escrito se representan como esperábamos en el navegador compatible con estándares que

hemos utilizado para probar nuestro trabajo. Conseguir correctamente la barra de navegación va a resultar ligeramente más complicado. En nuestra primera pasada, que veremos a continuación, conseguiremos determinadas características y perderemos algunas otras:

```
/* Componentes de la barra de
navegación */
table#nav    {
    border-bottom: 1px solid #000;
    border-left: 1px solid #000;
    }
```

Esta regla indica a la tabla con el `id nav` que debe crear un efecto de borde negro sólido de 1 píxel en su parte inferior y a la izquierda (pero no en la parte superior ni a la derecha).

```
table#nav td    {
    font: 11px verdana, arial,
sans-serif;
    text-align: center;
    vertical-align: middle;
    border-right: 1px solid #000;
    border-top: 1px solid #000;
    }
```

No hace falta explicar que esta regla especifica Verdana de 11 píxeles como la fuente de texto de menú preferida y que rellena los elementos de borde rechazados por la regla anterior (en la parte derecha y en la parte superior). Si hubiéramos indicado a la tabla que creara un efecto de borde por los cuatro lados, las celdas de tabla habrían añadido un píxel adicional de borde en la parte derecha y en la parte superior, lo que sería una vergüenza para nuestra familia y algo muy triste para todos los usuarios.

La regla anterior también indica que el texto se debe centrar horizontalmente en todas las celdas y que debe alinearse verticalmente en el centro de las mismas, similar al recurso de presentación `td valign="middle"` de la vieja escuela que todos conocemos. (Nos parecía que era lo correcto pero, en la segunda pasada, lo eliminaremos.)

```
table#nav td a    {
    font-weight: normal;
    text-decoration: none;
    display: block;
    margin: 0;
    padding: 0;
    }
```

En la regla anterior, comprobará que hemos indicado a los vínculos cómo deben comportarse y, para ello, hemos utilizado una sofisticada cadena de selectores contextuales. Las instrucciones del selector son "aplica la siguiente regla sólo a los vínculos incluidos en celdas de tabla y sólo si se encuentran en una tabla cuyo identificador exclusivo sea `nav`".

Como hemos mencionado anteriormente, también utilizamos la declaración CSS `display: block` para convertir los humildes vínculos XHTML en elementos de nivel de bloque que ocupen la totalidad de las celdas de tabla (al menos, es lo que esperamos).

```
#nav td a:link, #nav td a:visited {
    background: transparent url(images/
bgpat.gif)repeat;
    display: block;
    margin: 0;
    }
#nav td a:hover {
    color: #000;
    background: white url(images/
nopat.gif)repeat;
    }
```

Los dos últimas reglas utilizan selectores contextuales e `id` para controlar las pseudoclases vínculo, visitado y sobre, y rellenan las dos primeras clases con nuestra imagen

de color de fondo con píxeles alternos (véase figura 10.3). También utilizan una imagen de fondo blanco para el estado sobre/imagen cambiante.

Si volvemos a la figura 10.1, podemos comprobar todo lo que hemos conseguido correcta e incorrectamente en nuestra primera pasada de estilo. Todo lo que pretendíamos conseguir lo hemos conseguido, a excepción de la barra de navegación. El logotipo es correcto, el color de fondo se ha aplicado sin problemas (véase figura 10.11) y los efectos del ratón sobre el vínculo funcionan como esperábamos (véase figura 10.12).

Figura 10.11.

Los efectos de imagen cambiante de CSS en acción, obtenidos gracias a las diferentes pseudoclases de estado aplicadas a las celdas de la tabla con el id nav. En este caso, se reproduce al estado predeterminado de un gráfico de menú.

Figura 10.12.

Efectos de imagen cambiante de CSS, segunda parte. Cuando el cursor del usuario se sitúa sobre un gráfico de menú, el fondo se vuelve blanco. Sin necesidad de JavaScript (y no tenemos nada en contra de JavaScript).

IMÁGENES INNECESARIAS II: ESTA VEZ ES ALGO PERSONAL

Si recuerda, en una sección anterior con un título similar, mencionamos que el fondo de color blanco no es realmente necesario para esta ejecución. Pues bien, el fondo de color blanco no es realmente necesario para esta ejecución.

En lugar de:

```
#nav td a:hover {
    color: #000;
    background: white url(images/
nopat.gif)repeat;
    }
```

podríamos haber escrito la siguiente regla:

```
#nav td a:hover {
    color: #000;
    background-image: none;
    }
```

Al eliminar la imagen de los estados de vínculo (`background-image: none;`) se generaría el mismo efecto de imagen cambiante de un fondo de color blanco con una imagen menos por la que preocuparse y un menor consumo de ancho de banda. Más adelante en este capítulo crearemos los efectos concretos para cada una de las páginas, sin recurrir al GIF de color de fondo blanco.

Sin embargo, como los cambios de imágenes de fondo en CSS son muy atractivos y como puede que quiera dominar sus prestaciones en su propio proyecto, las hemos utilizado para generar el efecto de imagen cambiante en nuestra barra de navegación.

Pero el patrón de fondo predeterminado (enlace, visitado) únicamente rellena parte de cada elemento de menú de la derecha y queremos que rellene todo el espacio. Nos sentimos vulnerables y avergonzados. En un primer intento de una solución CSS definitiva, resolveremos este problema pero aparecerán otros nuevos.

CSS de la barra de navegación: primer intento, segunda pasada

En el primer intento de la segunda pasada, especificaremos los tamaños en los efectos de los vínculos:

```
#nav td a:link, #nav td a:visited {
    background: transparent url(images/
bgpat.gif)repeat;
    display: block;
    margin: 0;
    width: 100px;
    height: 25px;
    }
```

Como esperábamos, los botones de la parte derecha se rellenan completamente, pero el logotipo se ve afectado ya que su fondo tiene ahora 100 x 25 píxeles (véase figura 10.13). 100 x 25 es el valor indicado para los botones de pequeño tamaño, pero no resulta correcto para el logotipo (de 400 x 75).

De algún modo, los cambios que hemos realizado han eliminado la alineación vertical que definimos anteriormente. Los elementos se ignoran verticalmente dentro de las celdas, pero sus contenidos no. Como se aprecia en la figura 10.13, el texto de los botones se sitúa en la parte superior de las celdas en lugar de alinearse verticalmente en el centro. Este tipo de presentación es la misma en todos los navegadores compatibles con CSS probados. No se trata de un error, sino de un comportamiento inesperado que no encaja con el modelo de diseño CSS.

Figura 10.13.

Un paso adelante, dos pasos atrás. Al especificar tamaños en pseudoclases nav, los fondos se rellenan completamente pero el logotipo se ve afectado. También destruye la alineación vertical que definimos anteriormente. Ahora el texto se sitúa en la parte superior de las celdas.

Afortunadamente, al crear el marcado en un capítulo anterior, asignamos a cada celda un identificador exclusivo. `Home` es el `id` de la celda en la que se incluye nuestro logotipo. Se preguntará si podemos utilizar `home` para crear un conjunto adicional de reglas que reemplacen a las empleadas para crear los botones de 100 x 25. Claro que podemos:

```
td#home a:link, td#home a:visited {
    background: transparent url(images/
bgpat.gif) repeat;
    width: 400px;
    height: 75px;
    }
td#home a:hover {
    background: white url(images/
nopat.gif) repeat;
    width: 400px;
    height: 75px;
    }
```

Estas nuevas reglas rellenan el logotipo correctamente y el sitio es casi perfecto. Pero la pérdida del comportamiento que

esperábamos con `vertical-align: middle` sigue siendo inaceptable. Lo corregiremos en la pasada final.

CSS de la barra de navegación: pasada final

En la pasada final, obtenemos todo lo planeado:

```
/* Componentes de la barra de
navegación */
table#nav    {
    border-bottom: 1px solid #000;
    border-left: 1px solid #000;
    }
table#nav td    {
    font: 11px verdana, arial,
sans-serif;
    text-align: center;
    border-right: 1px solid #000;
    border-top: 1px solid #000;
    }
table#nav td a    {
    font-weight: normal;
    text-decoration: none;
    display: block;
    margin: 0;
    padding: 0;
    }
#nav td a:link, #nav td a:visited {
    background: transparent url(/images/
bgpat.gif) repeat;
    display: block;
    margin: 0;
    width: 100px;
    line-height: 25px;
    }
#nav td a:hover {
    color: #f60;
    background: white url(/images/
nopat.gif) repeat;
    }
td#home a:link img, td#home a:visited
img {
    color: #c30;
    background: transparent url(/images/
bgpat.gif) repeat;
    width: 400px;
    height: 75px;
    }
td#home a:hover img {
    color: #f60;
    background: white url(/images/
nopat.gif) repeat;
    width: 400px;
    height: 75px;
    }
```

¿Qué ha cambiado? Hemos eliminado la instrucción `vertical-align: middle`. Tras ello, hemos suprimido la línea que indicaba que los botones tenían una altura de 25 píxeles y la hemos sustituido por la siguiente:

```
line-height: 25px;
```

La altura de línea ha rellenado los 25 píxeles tal y como lo hizo la altura, pero también ha colocado correctamente el texto en el centro vertical de cada botón. Haría falta ser un genio en CSS para saber por qué ha funcionado este método mejor que el otro. Pero lo importante es que ha funcionado.

Últimos pasos: estilos externos y el efecto "Usted está aquí"

Para completar el sitio y enviárselo al cliente, necesitamos dos pasos más. En primer lugar, es necesario cambiar los estilos incrustados a un archivo CSS externo y eliminar la hoja de estilo incrustada, como explicamos en el capítulo anterior. Tras ello, tendremos que crear un efecto "Usted está aquí" (véanse figuras 10.14 y 10.15) para que el visitante sepa en qué página se encuentra. Recuerde que no vamos a cambiar el marcado. Queremos conseguir este efecto con ayuda de CSS, sin necesidad de aplicar clases adicionales a nuestra barra de navegación.

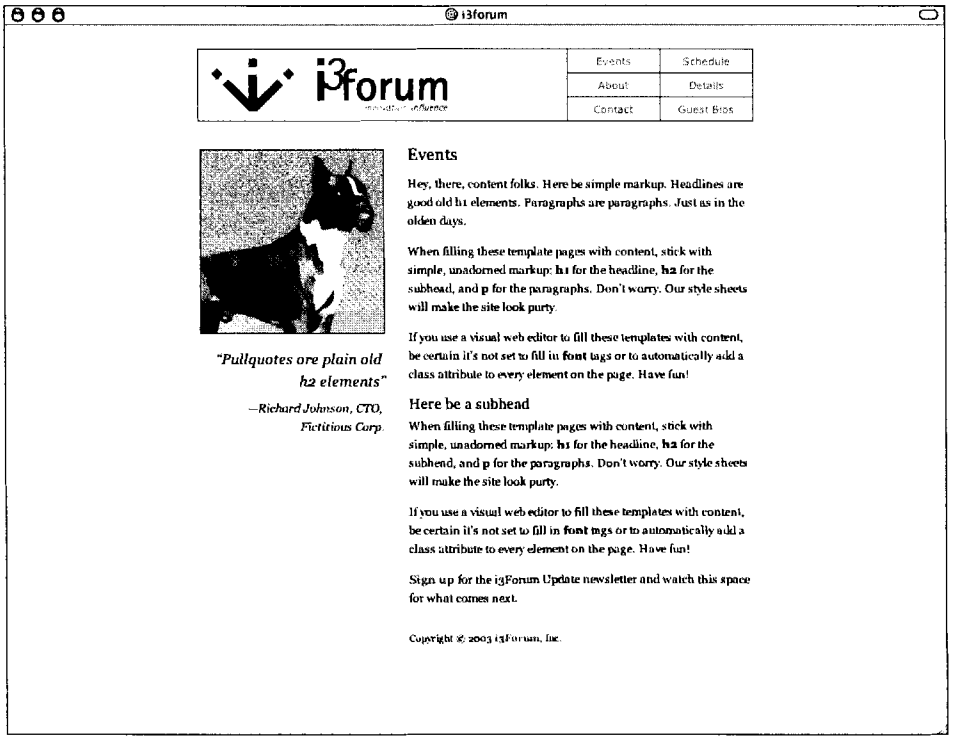


Figura 10.14.
El efecto "usted está aquí" en la plantilla de la página Events.

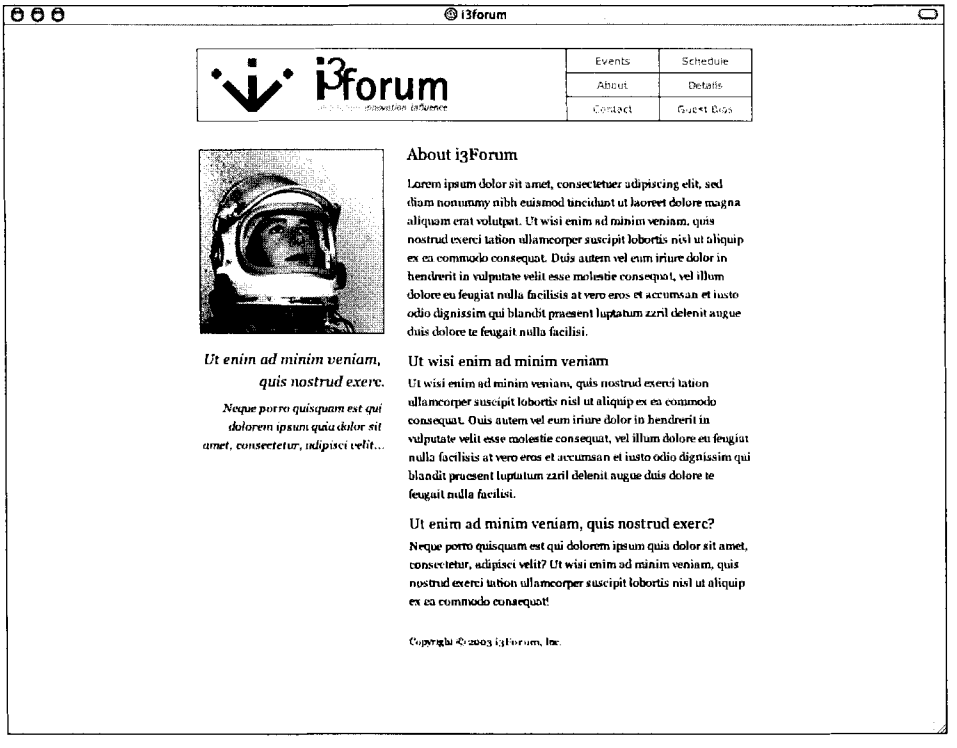


Figura 10.15.
El efecto "usted está aquí" en la plantilla de la página About.

Este tipo de efecto es muy sencillo de conseguir. Una vez eliminados los estilos incrustados, todas las páginas de plantilla obtienen sus datos CSS mediante un vínculo a una hoja de estilo externa como la siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/
xhtml">
  <head>
<title>i3forum</title>
<link rel="StyleSheet" href="/css/i3.css"
type="text/css" media="all" />
```

Basta con utilizar los elementos de estilo para añadir una hoja de estilo incrustada a cada página. Dicha hoja de estilo contendrá una sola regla: una regla que invierte la presentación habitual del botón de menú de dicha página. Es más fácil verlo que explicarlo.

En la página Events, la regla incrustada es la siguiente, con el selector importante resaltado en negrita:

```
<style type="text/css" media="screen">
td#events a:link, td#events a:visited {
    color: #c30;
    background: #fff;
    }
</style>
```

Fíjese en que hemos utilizado `background: #fff;` en lugar de `nopat.gif` para crear el resalte del fondo blanco, como prometimos anteriormente.

En la página About, la regla incrustada es la siguiente:

```
<style type="text/css" media="screen">
td#about a:link, td#about a:visited {
    color: #c30;
    background: #fff;
    }
</style>
```

Todas las páginas del sitio cuentan con una regla como ésta; por lo tanto, todas las páginas indican al visitante dónde se encuentra sin necesidad de modificar ni una línea de marcado. Se preguntará por qué no cambiamos el marcado de una página a otra o por qué no creamos una clase `thispage` para este indicador y la utilizamos para reemplazar el estilo de menú de dicho vínculo. Evidentemente se podría hacer y, de hecho, se hace a menudo.

Pero si no modificamos el marcado de navegación de una página a otra, se pueden añadir datos continuamente mediante inclusiones del lado del servidor, un enfoque muy útil para sitios de pequeño y mediano tamaño como el que queremos crear.

En www.zeldman.com, utilizamos esta técnica para modificar un indicador "usted está aquí" de una barra de navegación CSS página a página y utilizamos SSI para añadir los mismos datos XHTML a todas las páginas. Pero eso es otro sitio y otra historia, y hemos llegado al final de otro capítulo.

En el siguiente, nos adentraremos en las profundidades de CSS, veremos algunos de los problemas de los navegadores y sus respectivas soluciones, descubriremos cosas que los navegadores hacen correctamente (pero que puede que no nos interesen) y analizaremos obstinados elementos que resisten a todos los intentos de uso de estándares Web.

Capítulo 11
Cómo trabajar con navegadores (parte I). Conmutación de DOCTYPE y modo Estándares

¿Cómo saben los navegadores actuales si hemos creado un sitio con compatibilidad directa? ¿Cómo pueden representar correctamente un sitio basado en estándares cuando también tienen que admitir decentemente sitios anticuados diseñados con métodos desfasados?

La respuesta es que la mayoría de los navegadores modernos recurren a nuestro viejo amigo el DOCTYPE, que conocimos en un capítulo anterior, para cambiar de modo Estándares (con el que los sitios funcionan de acuerdo a las especificaciones del W3C) y modo de compatibilidad inversa (con el que los sitios se crean en función del comportamiento de diferentes navegadores incompatibles). Si le interesa controlar el comportamiento y el aspecto de sus sitios, éste es el capítulo correcto.

Para mantener el interés, el modo Estándares en navegadores Gecko como Mozilla y Netscape funciona de forma ligeramente diferente al modo Estándares en Internet Explorer, y estas pequeñas diferencias pueden tener un profundo e inesperado efecto en nuestros diseños. Para suavizar los problemas potenciales (y para que todo cobre más interés), los navegadores Gecko más recientes, como Mozilla 1.01+ y Netscape 7, añaden un tercer modo con un funcionamiento similar al modo Estándares de IE. (Para que la vida sea incluso más interesante, Opera 7 ha implementado la conmutación de DOCTYPE, aunque su funcionamiento se desconoce por el momento).

Los navegadores Gecko denominan a este tercer modo similar al de IE "Casi Estándares". El nombre no pretende insultar al modo Estándares de IE, al menos es lo que

pensamos, sino indicar que, según la opinión de los expertos en Gecko, el comportamiento visual que esperamos de nuestros navegadores difiere de las recomendaciones expuestas por las especificaciones CSS.

En este capítulo explicaremos el funcionamiento de los diferentes modos y le ofreceremos un sencillo método para garantizar el aspecto correcto de un sitio independientemente de la forma en que los navegadores interpreten CSS y otras especificaciones. Si ha convertido un sitio de transición (tablas más CSS) a XHTML, ha comprobado que después de hacerlo algunos navegadores tratan el diseño de forma diferente y se ha preguntado por qué los navegadores han cambiado su diseño y qué puede hacer para deshacer el cambio, este capítulo tiene su nombre escrito.

En el siguiente apartado veremos por qué la mayoría de los navegadores necesitaba un interruptor para cambiar de modo Estándares a modo de compatibilidad inversa, y cómo se ha elegido al humilde DOCTYPE para ello. Si es el tipo de persona al que le gusta saber de dónde vienen las cosas y por qué son lo que son, siga leyendo. Si prefiere ignorar la información previa y pasar directamente a los consejos y las técnicas, puede saltarse las siguientes páginas.

La saga de conmutadores DOCTYPE

A finales de los 90, cuando los fabricantes de navegadores se dieron cuenta de que la compatibilidad completa y precisa con los estándares Web era importante para los clientes que diseñaban y creaban sitios Web, se preguntaron cómo podían crear nuevos navegadores que admitieran los estándares correctamente sin que ello afectara a los sitios antiguos y no compatibles.

Después de todo, las versiones existentes de Netscape y Explorer habían persuadido a millones de diseñadores a adaptarse a sus particularidades concretas, incluyendo extensiones HTML propietarias, implementaciones CSS parciales e incorrectas y lenguajes de secuencia de comandos específicos del fabricante. Microsoft y Netscape estaban dispuestos a mejorar su compatibilidad con los estándares, pero no si ello significaba perder billones de dólares en sitios existentes. Para un fabricante de navegadores, sería un suicidio.

Puede que con un ejemplo comprenda el dilema de los fabricantes de navegadores.

A mediados de los años 90, cuando las primeras versiones de Internet Explorer comenzaban a admitir CSS parcialmente, hicieron algunas cosas de forma incorrecta, como en el caso del modelo de cuadro (que veremos en un capítulo posterior). Los primeros borradores siempre son duros y es de elogiar que Microsoft empezara a admitir CSS a finales de 1997.

Pero elogiable o no, la interpretación incorrecta del modelo de cuadro CSS por parte de Microsoft supuso un problema. Miles de diseñadores ya habían aprendido el modelo de cuadro incorrecto utilizado por IE4.x e IE5.x, y habían adaptado su CSS para que se mostrara adecuadamente en dichas

versiones de Internet Explorer. Si versiones posteriores de IE, no digamos los navegadores de otros fabricantes, eran compatibles con el modelo de cuadro con mayor precisión, los diseños existentes se vendrían abajo, para enfado del cliente, del creador y de los usuario ¿Qué se podía hacer?

Un interruptor para activar y desactivar estándares

Incluso antes de que Netscape y Microsoft acordaran crear navegadores que admitieran los estándares con mayor precisión, un héroe desconocido había resuelto el problema del procesamiento de sitios creados con métodos no compatibles. Ese héroe era el experto en interfaces de usuario Todd Fahrner, participante en los grupos de trabajo de CSS y HTML del W3C y cofundador del proyecto The Web Standards Project. Comprobará que el nombre de Fahrner aparece continuamente en el libro.

A principios de 1998, en una oscura lista de correo del W3C (todas las listas de correo del W3C son ciertamente oscuras, pero el adjetivo añade un toque de misticismo a la historia), Fahrner propuso que los fabricantes de navegadores incorporaran una especie de interruptor que permitiera activar y desactivar la representación de los estándares. También sugirió que la presencia o ausencia de un DOCTYPE se podría utilizar como dicho interruptor.

Según el razonamiento de Fahrner, si el marcado de una página comenzaba con un DOCTYPE, era más que probable que el autor tuviera ciertos conocimientos y hubiera hecho un esfuerzo por cumplir con los estándares. Los navegadores analizarían este tipo de páginas Web en función de las especificaciones del W3C. Por el contrario, una página sin DOCTYPE no pasaría las pruebas de validación del W3C (http://validator.w3.org/) y probablemente se hubiera creado con métodos de la vieja escuela. Los navegadores la procesarían en consecuencia, es decir, representarían la página como lo hacen los navegadores antiguos no compatibles.

Pero quedaba un problema: no había navegadores compatibles con los estándares. El mundo tendría que esperar dos años para ver si la conmutación de DOCTYPE era la clave de la compatibilidad inversa y directa. (Pero en nuestro caso no tendremos que esperar ni dos minutos. Basta con leer el siguiente párrafo.)

La aparición del interruptor

En Marzo del 2000, Microsoft lanzó al mercado IE Macintosh Edition, cuyo motor de representación Tasman, creado por el fanático de los estándares e ingeniero de Microsoft, Tantek Çelik, ofrecía una compatibilidad sustancialmente precisa y completa con los estándares, incluyendo CSS, XHTML y el DOM. IE5/Macintosh incluía Zoom de texto para mejorar la accesibilidad y fue el primer navegador en usar la conmutación de DOCTYPE para cambiar de un modo a otro.

Ocupados en la creación de sus propias innovaciones y en comercializar su propia familia de navegadores compatibles con estándares, los ingenieros que trabajaban en

navegadores basados en Gecko sabían que dos elementos eran correctos (conmutación de DOCTYPE y Zoom de texto) cuando los vieron. Los navegadores Gecko, entre los que destacaban Netscape 6, Mozilla y Camino, que apareció poco después de IE5/Macintosh, incluían conmutación de DOCTYPE y Zoom de texto, junto con una compatibilidad rigurosa y detallada con los estándares, posibilitada por el motor de representación Gecko/Mozilla (http://www.mozilla.org/newlayout/).

SIN CAMBIOS PARA OPERA

Antes de la versión 7.0, el navegador Opera de Opera Software no seguía estas normas; siempre trataba de representar las páginas en modo compatible con los estándares, independientemente de cómo se hubieran creado. Las versiones de Opera anteriores a la 7.0 tampoco admitían el DOM del W3C y, por lo tanto, los sitios compatibles que utilizan secuencias de comandos basadas en el DOM no funcionaban correctamente en los navegadores antiguos de Opera. Afortunadamente, los usuarios de Opera forman un selectivo grupo que suele descargar versiones mejoradas del programa tan pronto como aparecen. Y, como hemos mencionado anteriormente, Opera 7 incluye ahora conmutación de DOCTYPE, aunque la empresa no ha documentado aún su funcionamiento. (No sabemos si funciona como la conmutación DOCTYPE de IE, como la de Mozilla o como la de Netscape. Tendremos que esperar para saberlo.)

Cuando IE6/Windows se unió a este grupo, también admitía la conmutación DOCTYPE y añadía una propiedad DOM que podía mostrar si el modo Estándares se había activado en un determinado documento Web (http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/compatmode.asp).

Control del rendimiento de los navegadores: el conmutador DOCTYPE

Como hemos mencionado anteriormente, la mayoría de los navegadores modernos utilizan la presencia o la ausencia de determinados DOCTYPE para cambiar entre una compatibilidad rigurosa con los estándares y una compatibilidad inversa tolerante a fallos. Las implementaciones y los detalles de los navegadores varían, pero el funcionamiento básico de la conmutación DOCTYPE se ajusta a los predicados que definió Todd Fahrner en 1998, cuando los navegadores compatibles con estándares eran un sueño para los programadores. A continuación resumimos brevemente su funcionamiento:

- Un DOCTYPE XHTML que incluye un URI completo (una dirección Web completa), indica a los navegadores IE y Gecko que la página se represente en modo Estándares, con lo que nuestras CSS y XHTML se consideran especificaciones W3C. Algunos DOCTYPE HTML 4 también activan el modo Estándares, como veremos más adelante. En modo Estándares, el navegador asume que sabemos lo que estamos haciendo.

- Si se utiliza un DOCTYPE incompleto o desfasado, o no se utiliza ningún DOCTYPE, los mismos navegadores activan el modo de compatibilidad inversa, y asumen (probablemente con razón) que hemos escrito un marcado inválido y anticuado, y un código no estándar específico del navegador. En este modo, los navegadores intentan analizar la página mediante compatibilidad inversa, reproducen las CSS con el aspecto que tendrían en IE4 o 5 y recurren a un DOM propietario específico del navegador.

Para controlar la decisión que toman los navegadores, basta con incluir u omitir un DOCTYPE completo. Así de sencillo. Bueno, casi.

Tres modos para la hermana Sara

Por motivos que explicaremos más adelante, los nuevos navegadores Gecko pueden seleccionar entre tres modos. El modo de compatibilidad inversa (descrito anteriormente), el modo Casi estándares (equivalente al modo Estándares en IE) y el modo Estándares (ligeramente distinto al modo Estándares de IE en un par de aspectos concretos). Al mismo tiempo, algunos DOCTYPE que desencadenan el modo Estándares en los primeros navegadores Gecko, desencadenan el modo Casi estándares en los nuevos navegadores Gecko.

A los no iniciados puede parecerle realmente confuso pero, como comprobarán en las siguientes páginas, resulta brutalmente confuso. Sin embargo, hay motivos legítimos para ello e, igual de importante, existen sencillas soluciones para garantizar el correcto aspecto no sólo en navegadores que utilicen los diversos métodos de conmutación DOCTYPE, sino también en navegadores como Opera 5/6 que no lo hacen. Prepárese y siga leyendo. Realizaremos este viaje juntos.

Antes de analizar las diferencias entre los navegadores, veremos qué es lo que tienen en común.

DOCTYPE completos e incompletos

Los navegadores que utilizan conmutación DOCTYPE buscan DOCTYPE completos. Es decir, buscan DOCTYPE que incluyan una dirección Web completa como http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd. El siguiente DOCTYPE activa el modo Estándares en cualquier navegador moderno que utilice conmutación DOCTYPE:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-strict.dtd">
```

Sí, se trata de un DOCTYPE Strict y sí, hasta el momento hemos promovido el uso de XHTML Transitional. En este caso hemos utilizado Strict para ilustrar nuestro ejemplo sin demasiadas advertencias, de las que obtendremos muchas. Únicamente queremos decir que un DOCTYPE completo como el anterior sirve para desencadenar el modo Estándares.

Sin embargo, muchos escribimos DOCTYPE incompletos que desencadenan el modo de compatibilidad inversa en lugar del modo Estándares deseado. Y también los incluyen de forma predeterminada muchos de los

programas de creación que utilizamos. Por ejemplo, muchas herramientas de creación añaden DOCTYPE como el siguiente, derivado del propio sitio del W3C:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Strict//EN"
     "DTD/xhtml1-strict.dtd">
```

¿En qué se diferencia esta cadena de texto de la anterior?

Si se fija en la última parte del DOCTYPE anterior (`"DTD/xhtml1-strict.dtd"`) comprobará que se trata de un vínculo relativo en lugar de un URI completo. De hecho, es un vínculo relativo a un documento DTD del sitio del W3C. A menos que haya copiado esta DTD y la haya añadido a su propio servidor en el directorio indicado, algo que nadie hace, el vínculo relativo no apunta a nada. Como la DTD se encuentra en el sitio del W3C pero no en el nuestro, el URI se considera incompleto, por lo que el navegador muestra nuestro sitio en modo de compatibilidad inversa.

(Se preguntará si los navegadores actuales tratan de localizar y cargar la DTD. Pues no. Simplemente reconocen el DOCTYPE incompleto en su base de datos y activan el modo de compatibilidad inversa. ¿Significa esto que las páginas de `w3.org` que utilizan este DOCTYPE URI incompleto también se representarán en dicho modo? Parece que debería ser así. Aunque la ironía es algo delicioso de contemplar, no lo sabemos con certeza pero, en cualquier caso, no es nuestro problema.)

Volvamos de nuevo a la versión completa que desencadena el modo Estándares deseado:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-strict.dtd">
```

Comprobará que incluye un URI completo al final de la etiqueta. Podríamos copiar dicho URI y pegarlo en la barra de dirección de cualquier navegador Web y, si lo hiciéramos, podríamos leer la DTD XHTML 1.0 Strict en su incomprensible totalidad. Como el URI en el extremo final de este DOCTYPE indica una ubicación válida en la Web, los navegadores con conmutación DOCTYPE consideran que se trata de un DOCTYPE completo y representan nuestra página en modo Estándares.

Como información adicional, debe saber que Internet Explorer cambia a modo Estándares en presencia de cualquier DOCTYPE XHTML, incluya o no un URI completo. Sin embargo, un DOCTYPE XHTML que no incluya un URI completo, es técnicamente nulo. Por ello, en el siguiente análisis, le aconsejamos encarecidamente que utilice un URI completo en su DOCTYPE. (Después de todo, no tendría sentido cambiar a modo Estándares si nuestra página Web no es válida.)

Compatibilidad inversa de los prólogos en IE6/Windows

Siempre hay una excepción a la regla. Incluso con un DOCTYPE XHTML completo, IE6/Windows activará el modo de compatibilidad inversa si incluimos el prólogo XML opcional. En serio. Por esta razón, en un capítulo anterior desaconsejamos la inclusión de este tipo de prólogos. ¿Ve como todo llega tarde o temprano?

Listado completo de DOCTYPE XHTML completos

A continuación, enumeramos los DOCTYPE XHTML completos que activan los modos Estándares o Casi estándares en navegadores con conmutación de DOCTYPE. Se preguntará qué es el modo Casi estándares. Sea paciente. Pronto lo veremos.

DTD XHTML 1.0

XHTML 1.0 Strict: Activa el modo Estándares en todos los navegadores que admiten la conmutación de DOCTYPE. No funciona en versiones de Opera anteriores a la 7 ni en versiones de IE/Windows anteriores a la 6.0.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-strict.dtd">
```

XHTML 1.0 Transitional: Activa el modo Estándares en navegadores IE compatibles (IE6+/Windows, IE5+/Macintosh). También lo activa en la primera generación de navegadores Gecko (Mozilla 1.0 y Netscape 6) y el modo Casi estándares en navegadores Gecko actualizados (Mozilla 1.01, Netscape 7+ y Camino 0.6+). No funciona en versiones de Opera anteriores a la 7 ni en versiones de IE/Windows anteriores a la 6.0.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset: Activa el modo Estándares en navegadores IE compatibles (IE6+/Windows, IE5+/Macintosh). También lo activa en la primera generación de navegadores Gecko (Mozilla 1.0 y Netscape 6) y el modo Casi estándares en navegadores Gecko actualizados (Mozilla 1.01, Netscape 7+ y Camino 0.6+). No funciona en versiones de Opera anteriores a la 7 ni en versiones de IE/Windows anteriores a la 6.0. La conmutación de DOCTYPE puede afectar a la forma de representar los marcos pero, sí esto sucede, no está documentado por ningún navegador.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Frameset//EN"
     "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-frameset.dtd">
```

DTD XHTML 1.1

XHTML 1.1 (Strict por definición): Activa el modo Estándares en todos los navegadores que admiten conmutación de DOCTYPE. No funciona en versiones de Opera anteriores a la 7 ni en versiones de IE/Windows anteriores a la 6.0.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.1//EN"
     "http://www.w3.org/TR/xhtml11/DTD/
xhtml11.dtd">
```

Truco: Si no le agrada tener que escribir a mano este tipo de comandos (¿y a quién sí?), puede copiar el código y pegarlo desde la página A List Apart (http://www.alistapart.com/stories/doctype/).

Resumen ejecutivo: desencadenadores DOCTYPE XHTML en IE y Gecko

Veamos lo que hemos aprendido hasta el momento:

- Cualquier DOCTYPE XHTML 1 ó 1.1 completo activa el modo Estándares en versiones compatibles de Internet Explorer (IE5+/Macintosh, IE6+/Windows) y en los primeros navegadores Gecko (Netscape 6, Mozilla 1.0).
- Un DOCTYPE XHTML Strict completo tiene el mismo efecto en todos los navegadores anteriores y también en los últimos navegadores Gecko (Mozilla 1.01+, Netscape 7 y Camino 0.6+).
- Un DOCTYPE XHTML 1.0 Transitional o Frameset completo activa el modo Casi estándares en los últimos navegadores Gecko.
- Los DOCTYPE incompletos que tienen URI relativos activan el modo de compatibilidad directa en todos los navegadores que admiten conmutación de DOCTYPE. También generan errores de validación CSS, pero no generan errores de validación XHTML debido a un error del servicio de validación XHTML del W3C o a una diferencia entre sus servicios de validación XHTML y CSS, según a quién pregunte. Bueno, olvide todo lo que hemos dicho excepto que los DOCTYPE incompletos activan el modo de compatibilidad inversa y desencadenan errores de validación CSS.
- Muchas herramientas de creación añaden DOCTYPE incompletos de forma predeterminada, por lo que activan el modo de compatibilidad inversa y generan errores de validación CSS.

Este autor y los integrantes del Web Standards Project se han puesto en contacto con los fabricantes de las herramientas de creación más conocidas para avisarles de la necesidad de incluir DOCTYPE completos de forma predeterminada. También hemos solicitado al W3C que publique un listado de DOCTYPE completos en su sitio Web, www.w3.org. Por el momento, para garantizar que su sitio activa el modo Estándares, puede que tenga que modificar manualmente el DOCTYPE generado por su herramienta de creación cuando diseñe una nueva página XHTML.

Conmutación DOCTYPE: el peligro está en los detalles

La conmutación de DOCTYPE no se limita a XHTML. Como hemos mencionado anteriormente, algunos DOCTYPE HTML completos también activan el modo Estándares. Por ejemplo, IE activa el modo Estándares (y los nuevos navegadores basados en Gecko hacen lo propio con el modo Casi estándares) en presencia de un DOCTYPE HTML 4.01 Strict completo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.01//EN"
     "http://www.w3.org/TR/html4/
strict.dtd">
```

Pero tanto en navegadores IE como en los basados en Gecko, un DOCTYPE HTML 4.0 completo activa el modo de compatibilidad inversa. La diferencia radica en 0.1. Si estas incoherencias le sugieren que es más aconsejable crear los sitios en XHTML que en HTML, es lo que hemos recomendado hasta ahora.

Si necesita información adicional sobre DOCTYPE que activan los diferentes modos en los nuevos navegadores Gecko, consulte

el artículo "Gecko's 'Almost Standards' Mode" en http://devedge.netscape.com/viewsource/2002/ almost-standards/. Si el nivel de detalle que ofrece le parece excesivo, deje de leerlo y haga como que nunca lo ha leído. Como comprobará en el siguiente apartado, no tenemos que preocuparnos por estas complejidades siempre que creemos nuestros sitios en XHTML 1.0 Transitional o Strict, y que apliquemos las sencillas soluciones CSS que describiremos a continuación.

Celebremos la diversidad de navegadores (o al menos aprendamos a vivir con ella)

El modo Estándares de Gecko difiere del de IE en determinados aspectos, y esta diferencia puede resultar desagradable si no la esperamos. La diferencia es especialmente apreciable y resulta especialmente desagradable en diseños híbridos (CSS más tablas) cuando se reproducen en navegadores Gecko de primera generación.

Después de convertir de HTML 4.01 Transitional a XHTML 1.0 Transitional y sin realizar cambios adicionales en el marcado a excepción de los descritos en capítulos anteriores, los diseñadores esperan que sus diseños tengan el mismo aspecto que siempre han tenido. Sin embargo, sus diseños pueden tener un aspecto inesperadamente distinto en los primeros navegadores Gecko como Netscape 6.0 y Mozilla 1.0. Si los diseñadores hubieran realizado la conversión a XHTML Strict, sus diseños se verían de forma diferente en todos los navegadores Gecko, tanto los nuevos como los antiguos. Hay un motivo basado en estándares para el cambio y también una sencilla forma de solventarlo.

El problema del hueco de las imágenes en Gecko

En la figura 11.1 se muestra la sección The Daily Report (www.zeldman.com), tal y como se ha visto durante años, cuando el diseño del sitio se basaba en técnicas de transición que combinaban CSS y tablas.

Después de realizar la conversión de HTML 4.01 Transitional a XHTML 1.0 Transitional (y cambiar de un DOCTYPE HTML 4.01 completo a un DOCTYPE XHTML 1.0 Transitional completo) el aspecto del sitio era el mismo que siempre había tenido en Internet Explorer, tanto para Windows como para Macintosh. Pero en los navegadores Gecko, aparecían extraños y misteriosos huecos entre las imágenes de las tablas que controlaban el menú de navegación del sitio (véanse figuras 11.2 y 11.3).

Una o dos reglas para vincularlo todo: uso de CSS para desactivar los huecos

Existe un motivo perfectamente válido para este inesperado comportamiento (como veremos más adelante). Pero antes de adentrarnos en la teoría, explicaremos por qué hemos recuperado el estado correcto de la navegación (véase figura 11.4) tanto en navegadores Gecko como IE. Ha bastado con dos líneas de CSS:

```
img {display: block;}
.inline {display: inline;}
```

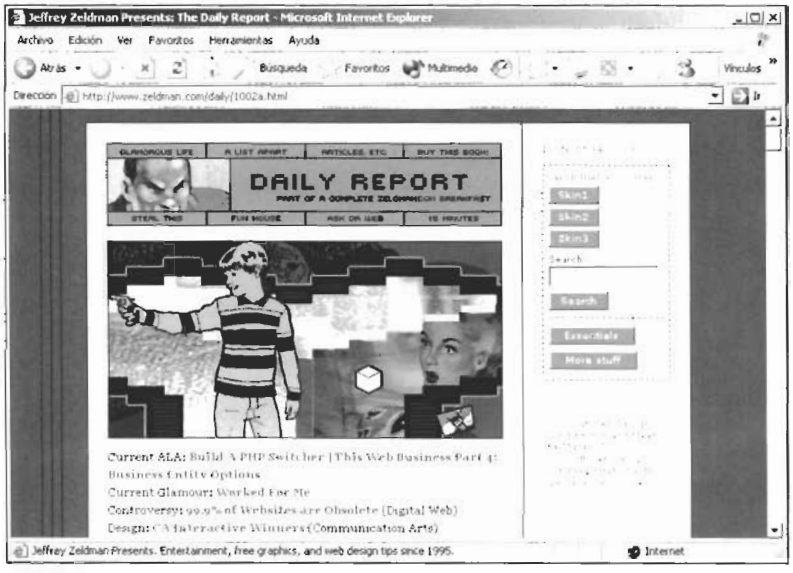


Figura 11.1.

Este anticuado diseño híbrido de tablas y CSS en www.zeldman.com empleaba un DOCTYPE completo para activar el modo Estándares. Después de realizar la conversión de HTML a XHTML, el aspecto era el mismo que siempre había tenido en Internet Explorer (www.zeldman.com/daily/1002a.html).

Figura 11.2.

En los primeros navegadores Gecko, antes de aplicar la solución CSS descrita en este capítulo, la tabla de navegación del sitio mostraba misteriosos huecos no deseados.

Figura 11.3.

La misma imagen se ha aumentado para que los huecos se aprecien con mayor claridad. Los primeros navegadores Gecko no se equivocaban al representar la tabla de esta forma, pero lo hacían de forma no esperada.

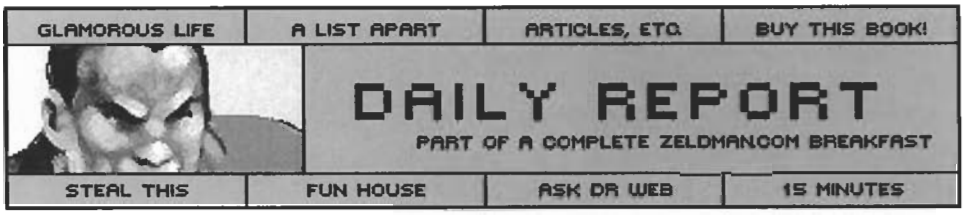


Figura 11.4.

Por medio de dos sencillas reglas CSS eliminamos los huecos no deseados en los navegadores Gecko. En el modo Casi estándares utilizado por los navegadores Gecko posteriores, este problema no aparecía y no se necesitaba solución alguna. Pero, de todas formas, conviene utilizarlas, como explicaremos a continuación.

La primera regla cierra los huecos. Para ello, indica a todos los navegadores que traten a las imágenes como elementos de nivel de bloque y no como elementos en línea (en un capítulo anterior encontrará mas información sobre las diferencias entre estos elementos). Pero, en caso de que queramos que alguna de nuestras imágenes se muestre en línea, la segunda regla (`.inline`) crea una clase para ello. Las imágenes que se muestren en línea tendrán el siguiente marcado:

```
<img class="inline" … etc. />
```

Existen diversas formas de obtener este resultado. Por ejemplo, algunos diseñadores prefieren escribir una regla que especifique que cuando una celda de tabla contenga sólo una imagen, ésta se muestre como elemento de nivel de bloque:

```
td img {display: block;}
```

Con esta sencilla regla, los menús de navegación que suelen incluir una única imagen por celda de tabla se representarán como elementos de nivel de bloque y sin huecos, mientras que el resto de imágenes del sitio se mostrarán en línea. La ventaja de este método alternativo es que utiliza una regla en lugar de dos, lo que ahorra en ancho de banda y elimina la necesidad de añadir clases a ciertas imágenes, lo que aumenta este ahorro.

Cada uno de los métodos funciona mejor en unos diseños que en otros. El método que seleccione depende de su diseño. Un consejo para principiantes: pruebe un método. Si no funciona, pruebe con el otro. Puede que consiga un tercer método completamente personal.

El nacimiento del modo Casi estándares

La creación de sencillas reglas CSS como las que hemos presentado anteriormente apenas nos llevará tiempo. La incorporación de las mismas a sus prácticas de diseño evitará que tenga que preocuparse por los diferentes navegadores. Sin embargo, cuando apareció por primera vez el problema de los huecos de las imágenes y cuando parecía que sólo se producía al convertir de HTML a XHTML, algunos diseñadores llegaron a la conclusión de que XHTML era defectuoso, mientras que

otros determinaron que Netscape 6 y otros navegadores Gecko no funcionaban o eran incorrectos. De hecho, la inclusión de espacios en blanco en imágenes sin estilo era una de las características de Gecko, no un error (como veremos en breve). No obstante, no era una característica que los diseñadores comprendieran desde un principio y tampoco una característica que fuera necesaria.

Para contentar a los diseñadores en proceso de transición, los ingenieros de Gecko/Netscape crearon un modo Casi estándares que se comportaba como el modo Estándares de IE.

Líneas base y espacio en blanco vertical en el modo Estándares de Gecko

Como mencionamos en el apartado anterior, los navegadores IE y de Gecko se diferencian en el tratamiento que hace el modo Estándares de las imágenes en relación a la cuadrícula de una página Web. En un modo Estándares puro, Gecko considera que las imágenes son elementos en línea a menos que en nuestra hoja de estilo especifiquemos que son elementos de nivel de bloque. Todos los elementos en línea, como el texto, se sitúan sobre una línea base para que las formas descendentes de las letras minúsculas, como y, g o j, tengan espacio suficiente. El tamaño y la ubicación de dicha línea base depende del tamaño y de la fuente del elemento contenedor, por ejemplo, del tamaño y de la fuente del texto de un párrafo que contenga la imagen en línea.

En la figura 11.5 podemos ver una imagen en línea sobre su línea base. En el modo Estándares de Gecko, siempre se añade un espacio en blanco por debajo de las imágenes en línea para que las letras descendentes del bloque contenedor tengan espacio suficiente, haya texto o no en dicho bloque. Si necesita más información al respecto, consulte el artículo de Eric Meyer en Netscape DevEdge, escrito antes de la creación del modo Casi estándares, pero no por ello menos importante (http://devedge.netscape.com/viewsource/2002/img-table/).

Figura 11.5.

Todo el texto se sitúa en una línea base para que las partes descendentes de las letras tengan espacio suficiente. En el modo Estándares de Gecko, todos los elementos en línea, incluyendo las imágenes, comparten la línea base con el elemento contenedor (en este caso, un párrafo de una frase).

Es decir, en este nuevo modo, no aparecían los inesperados huecos. Como este comportamiento inesperado se producía principalmente en diseños de transición, los ingenieros determinaron que los DOCTYPE XHTML Transitional invocarían el modo Casi estándares de Gecko, mientras que los DOCTYPE Strict continuarían activando el modo Estándares más riguroso de Gecko. (Después de todo, si escribe marcado estricto que no sea de presentación, probablemente no incluya las imágenes en celdas de tabla.)

Limitaciones del modo Casi estándares

Si todos los usuarios de Gecko actualizaran rutinariamente sus navegadores con las últimas versiones, el modo Casi estándares resolvería el problema de los huecos de las imágenes en diseños de transición. Muchos usuarios de Gecko actualizan religiosamente sus navegadores pero algunos, por ejemplo los usuarios de CompuServer, no lo hacen. (Los usuarios de CompuServe actualizan sus navegadores cuando su proveedor de servicios les envía un nuevo disco de instalación.) Por ello, tiene sentido escribir reglas CSS como las que hemos descrito anteriormente. De esta forma, podremos prever resultados inesperados en navegadores actuales y futuros que no sean IE ni Gecko, que pueden ser tan rigurosos en la forma de procesar imágenes y otros elementos en línea como lo es Gecko en modo Estándares. Además, reducirá el marcado de presentación y los comportamientos normativos del navegador asociados al mismo, y le ayudará a que piense en CSS.

De "Viva la diferencia" a "Esto es una mier..."

Un comportamiento correcto, aunque inesperado, es una cosa. Los fallos y los errores son caso aparte. En el siguiente capítulo analizaremos los fallos habituales de los navegadores y aprenderemos a solucionarlos. También describiremos aspectos fundamentales del diseño CSS.

Capítulo 12
Cómo trabajar con navegadores (parte II). Modelos de cuadro, fallos y soluciones

"Crear una vez, publicar en todas partes" es el Grial del diseño y la programación basados en estándares. No aprendemos a crear correctamente en XHTML para ganar una medalla de oro. Lo hacemos para que nuestros sitios funcionen en navegadores de escritorio, navegadores de texto, lectores de pantalla y dispositivos manuales, tanto hoy, como mañana y como dentro de 10 años. Del mismo modo, no utilizamos CSS exclusivamente para objetivos a corto plazo, como la reducción del ancho de banda para recortar los gastos mensuales. Lo hacemos principalmente para que nuestros sitios tengan el mismo aspecto en Nestcape 14.0 como el que tienen en la actualidad y que el marcado de presentación innecesario no impida la experiencia de los usuarios en entornos no CSS.

Los agentes de usuario compatibles con los estándares trasladan la compatibilidad directa desde el reino de los deseos hasta estrategias de diseño racionales y sostenibles. Si la Web se viera principalmente en navegadores 4.0 de Microsoft, sólo podrían conseguir la compatibilidad directa los sitios más rudimentarios, no todos.

Si los principales agentes de usuario que tuvieron éxito con los navegadores 4.0 hubieran continuado con su apoyo a las tecnologías propietarias a expensas de los estándares base, el futuro de la Web como plataforma abierta sería dudoso.

Afortunadamente, se puede afirmar que todos los navegadores comercializados en los últimos años son compatibles con los estándares. Pero algunos son más compatibles que otros. En este capítulo trataremos de solucionar las diferencias de compatibilidad.

El modelo de cuadro y sus descontentos

Con la publicación de CSS1 en 1996, el W3C propuso que cualquier objeto de una página Web se incluya en un cuadro cuyas propiedades puedan controlar los diseñadores por medio de reglas. No importa que el objeto sea un párrafo, una lista, un título, una imagen o un elemento genérico de nivel de bloque como `<div id="navigation">`. Si se puede añadir a una página Web a través del marcado, se incluirá en un cuadro.

En la figura 12.1 puede ver las cuatro áreas del cuadro CSS: contenido, relleno, borde y margen. El contenido (de color gris oscuro en la imagen) es la zona más interna, seguida por el relleno (de color gris claro), el borde (de color negro) y el margen (de color blanco). Estas zonas deberían resultarle familiares. En un capítulo anterior, al crear un diseño híbrido para el sitio i3Forum, especificamos valores de borde, relleno y margen para los diferentes elementos de la página. También asignamos valores de tamaño a la zona de contenido al definir reglas como la siguiente (los valores de contenido se resaltan en negrita):

```
td#home a:link img, td#home a:visited
img {
    color: #c30;
    background: transparent url(/images/
bgpat.gif) repeat;
    width: 400px;
    height: 75px;
    }
```

En esta regla, la anchura de la zona de contenido es de 400 píxeles y la altura es de 75 píxeles. Comprobará que no escribimos esto:

```
content: 400px;
```

Ni tampoco esto:

```
content-width: 400px;
content-height: 75px;
```

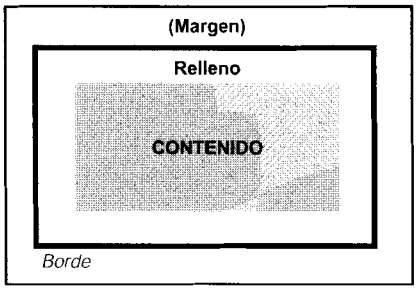


Figura 12.1.

Las cuatro áreas del cuadro CSS genérico: contenido, relleno, borde y margen. (El borde exterior del margen se ha oscurecido artificialmente para que se vea mejor.)

En CSS no existen estas reglas. Los valores de borde, de los márgenes y de relleno se asignan por nombre; la zona de contenido no. Al comenzar el aprendizaje de CSS, puede asumir que `width: 400 px` se aplica a todo el cuadro (sin incluir el margen). Después de todo, es la forma en la que funcionan los programas de diseño de páginas, la forma en que piensan los diseñadores y la forma en que los usuarios entienden los diseños. Si creamos un diseño CSS que contenga dos elementos `div` próximos entre sí y asignamos a cada uno una anchura del 50 por ciento de la ventana del navegador del visitante, seguramente espere que ambos conserven dicho valor al añadir relleno y bordes. Pero CSS no funciona de esta forma.

Del mismo modo, si fuera un fabricante de navegadores que tuviera que implementar CSS, aplicaría incorrectamente `width: 400px` a todo el cuadro (sin incluir el

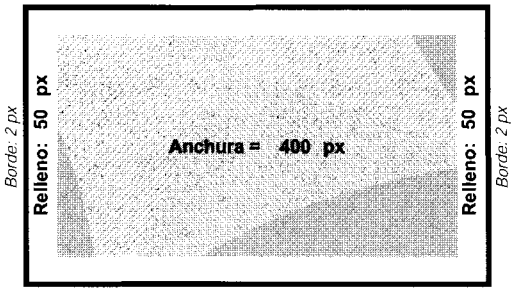
margen) en lugar de limitarlo a la zona de contenido como dictan las especificaciones. Resultaría muy sencillo cometer un error de este tipo, sobre todo si implementáramos CSS1 mientras la especificación sigue reciente. No olvide este aspecto; lo retomaremos más adelante.

De hecho, el modelo de cuadro CSS es más sofisticado y, de algún modo, más problemático, que lo que el sentido común, las normas de diseño gráfico y las primeras (e incorrectas) implementaciones de navegadores le pueden hacer creer.

Funcionamiento del modelo de cuadro

Según CSS, a cada una de las cuatro áreas (contenido, relleno, borde y margen) se les puede asignar valores y estos valores son acumulativos. Para determinar la anchura total del cuadro, debe sumar los valores de relleno, de contenido y de borde (véase figura 12.2). Si la anchura del contenido es de 400 píxeles, el relleno es de 50 píxeles en cada lado y el borde es de 2 píxeles, la anchura total será de 504 píxeles (400 de contenido + 2 del borde izquierdo + 50 del relleno izquierdo + 50 del relleno derecho + 2 del borde derecho = 504).

A CSS no le importan los tipos de valores que seleccionemos. Por ejemplo, podemos especificar que la anchura del contenido sea el 67 por ciento de la anchura de la ventana del navegador del usuario en el momento en que se vea la página, que el relleno sea 5 eme y el borde 1 píxel. El tamaño total sería difícil de determinar y dependería de la ventana del navegador del visitante y del tamaño predeterminado del texto (1 eme).



Anchura total: 2px + 50px + 400px + 50px + 2px = 504px

Figura 12.2.

El modelo de cuadro en funcionamiento. Los valores de contenido, relleno y borde se suman para conseguir la anchura total del cuadro.

Hay más sobre el modelo de cuadro de lo que hemos explicado hasta el momento. Por un lado, los márgenes superior e inferior de elementos verticalmente adyacentes se solapan entre sí. (Prepare un café doble o algo más fuerte y lea con atención http://www.w3.org/TR/REC-CSS2/box.html.) Por otra parte, los valores de tamaño carecen de sentido si el cuadro está vacío. Si especificamos que un cuadro vacío debe ocupar el 100 por cien de la altura y la anchura de la página, ocupará realmente el 0 por ciento ya que no hay nada en su interior. Difiere del funcionamiento de los marcos HTML y también de la forma en que en ocasiones funcionan los diseños de tabla en navegadores 4.0. Si trata de conseguir efectos de color mediante el relleno de dos elementos de nivel de bloque adjuntos con colores de fondo (pero sin datos reales), la técnica funcionará en marcos y puede que lo haga en diseños de tabla vistos en navegadores 4.0 pero no lo hará en

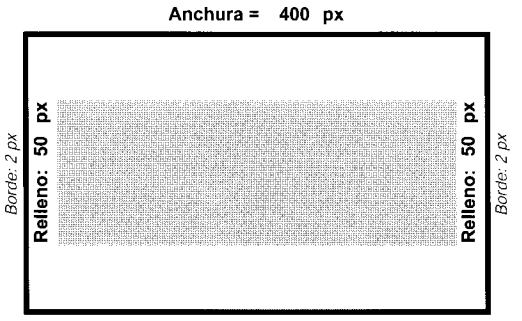
CSS. En este sentido, paradójicamente, no se puede separar la presentación de la estructura ya que, si no hay datos, no puede haber presentación. (Si un árbol no se cae, no hay bosque.)

Fallos del modelo de cuadro

Si ha prestado atención, ya se habrá percatado de cómo falla el modelo de cuadro en las primeras implementaciones CSS como las comprendidas entre IE4 e IE5.5/Windows. En estos navegadores y en IE/Mac hasta la versión 5.0, los valores de anchura y altura destinados únicamente a la zona de contenido se aplican a la totalidad del cuadro (sin incluir los márgenes).

En la figura 12.3 podemos ver el modelo de cuadro. Para seguir con el ejemplo anterior, en IE4-5.5/Windows, si la anchura del contenido es de 400 píxeles, el relleno es de 50 por lado y el borde es de 2 píxeles, la anchura total será de 400 píxeles y no tendrá los 504 píxeles correctos, y la zona de contenido se reducirá a 296 píxeles (400-50-50-2-2). Cuanto mayor sean los valores de relleno y de los bordes, mayor será la desviación del diseño de nuestras pretensiones y más se reducirá la zona de contenido.

Cientos de millones de usuarios siguen utilizando IE5.x/Windows y miles de diseñadores han adoptado el modelo de cuadro incorrecto de IE5.x/Windows y lo utilizan en sus diseños, lo que genera resultados erróneos en navegadores que interpretan correctamente el modelo de cuadro, como por ejemplo IE6/Windows, IE5/Macintosh, Netscape 6, Netscape 7, Mozilla, Camino, Kmeleon, Safari, Opera 5, Opera 6 y Opera 7.



Modelo de cuadro incorrecto en IE. Anchura general: 400 px.

Figura 12.3.

El modelo de cuadro falla en las versiones de Internet Explorer comprendidas entre la 4.0 y la 5.5. 400 + 2 + 50 + 50 + 2 = 400. Aritmética incorrecta pero puede que esté bien pensado.

Por el contrario, los diseñadores que comprenden el modelo de cuadro y realizan sus creaciones de acuerdo a las especificaciones, producen los resultados deseados en los navegadores que acabamos de mencionar pero obtendrán resultados nada agradables en IE5.5/Windows y anteriores, utilizados, como ya hemos mencionado, por cientos de millones de usuarios.

Existe una solución. De hecho, existen varias. Pero la más utilizada proviene, de todos los posibles lugares, de un ingeniero de Microsoft. Pero antes de solucionar el fallo del modelo de cuadro, nos detendremos a analizar sus virtudes.

En defensa del modelo de cuadro fallido

El modelo de cuadro en IE4/5.x/Windows (y también en todas las versiones de IE/Macintosh previas a la 5.0) es incuestiona-

blemente defectuoso e incorrecto según la especificación CSS, pero no es estúpido. En cierto modo, este modelo de cuadro resulta más sencillo de utilizar y es más práctico que la especificación que determina la CSS. El modelo de cuadro CSS funciona correctamente cuando se utiliza para crear diseños basados en valores de píxel absolutos, pero su complejidad puede que lo haga problemático, poco intuitivo y en ocasiones contraproducente al intentar crear diseños basados en porcentajes que cambien para ajustarse al monitor del visitante.

Imagine un sencillo diseño con dos celdas en el que la anchura de las columnas de la izquierda y la derecha añaden hasta un 100 por ciento de la anchura del monitor del visitante. Resulta trivial crear este tipo de diseños con tablas HTML pero no tan sencillo por medio de CSS. Para crear un diseño de dos columnas en CSS, un ser humano crearía dos elementos `div` y utilizaría la propiedad CSS `float` para situar uno junto al otro:

```
#nav  {
      width: 35%;
      height: 100%;
      background: #666;
      color: #fff;
      margin: 0;
      padding: 0;
      }

#maintext  {
      width: 65%;
      height: 100%;
      color: #000;
      background: #fff;
      margin: 0;
      padding: 0;
      float: right;
      }
```

En este caso tenemos dos divisiones de página: una para la navegación y otra para el texto principal. `float: right` indica a la columna de texto principal que flote hacia la derecha de la columna de navegación. La columna de navegación debe tomar el 35 por ciento de la anchura de la ventana del navegador del usuario. La zona de texto principal recibe el 65 por ciento restante. Entre ambas, las dos divisiones de página ocuparán el 100 por ciento de la ventana (35 + 65 = 100).

En la figura 12.4 se puede apreciar que el enfoque funciona siempre que no se utilice ni relleno ni bordes. Los bordes son opcionales y dependen del diseño, pero el relleno es fundamental para evitar que aparezca texto ilegible como el que se incluye en esta ilustración. Cuando un diseñador añade relleno y bordes a un diseño básico, lado a lado como éste, la página se desmorona. En algunos navegadores, los elementos se solapan (véase figura 12.5). En otros, como las cifras suman más del 100 por cien, la anchura del diseño es excesiva para poder representarse en el monitor, por lo que el lector tendrá que desplazarse horizontalmente. Independientemente del tamaño del monitor utilizado, el diseño siempre será demasiado ancho para el mismo.

El funcionamiento se puede mejorar si definimos el relleno y la anchura en porcentajes, que se escalarán para ajustarse a la ventana del navegador del visitante. Pero la combinación precisa de porcentajes con rellenos basados en longitud es esencialmente imposible. Para corregir este problema, CSS3 contiene una propuesta de propiedad de tamaño de cuadro para que los diseñadores puedan decidir qué modelo de cambio de tamaño quieren emplear con un determinado elemento.

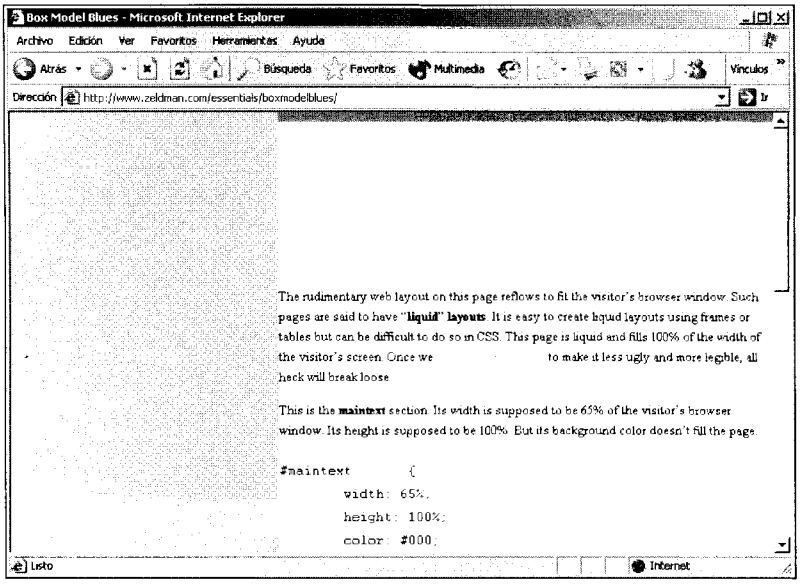


Figura 12.4.

Con CSS resulta más complicado conseguir diseños flexibles basados en porcentajes que cambien para ajustarse al monitor del visitante. En este diseño, los dos elementos div se colocan de forma contigua (http://www.zeldman.com/essentials/boxmodelblues/).

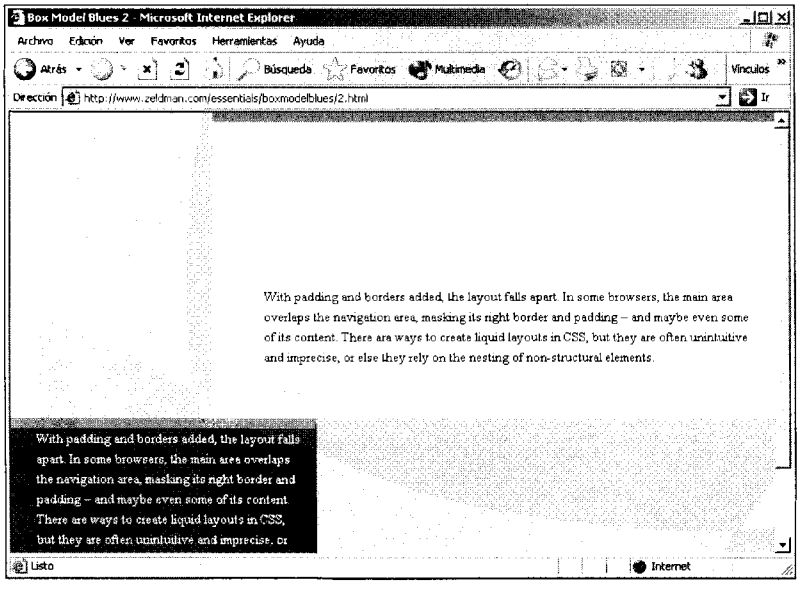


Figura 12.5.

Cuando añadimos relleno y bordes, el diseño se desmorona. En algunos navegadores los elementos se solapan. En otros, la página resulta demasiado ancha para la pantalla.

Si el modelo de cuadro CSS funcionara como la implementación incorrecta en versiones anteriores de IE/Windows, no necesitaríamos una propiedad de cambio de tamaño de cuadro. Y, evidentemente en la actualidad, no disponemos de dicha propiedad en nuestros navegadores.

En ausencia de dicha propiedad, los diseñadores suelen hacer trampas mediante la definición de `div` sin relleno, añadiendo subelementos en aquéllos que contienen a los `div` y asignando márgenes a dichos subelementos para imitar el deseado efecto de relleno. El problema con esta solución es

que genera extraños elementos de marcado que no son estructurales, ya que únicamente existen para solucionar problemas de diseño (en capítulos posteriores encontrará más información sobre estos subelementos). Si añadimos demasiados elementos de este tipo, nuestro marcado comenzará a parecerse a la divitis que comentamos en un capítulo anterior.

Hay que reconocer que estos enfoques son ciertamente puristas. Aparte de los bytes adicionales de ancho de banda, no hay nada malo en crear y anidar elementos no estructurales para solucionar problemas de diseño. En términos prácticos, el daño es mínimo. Pero es la misma mentalidad obsoleta que ocasionó el problema. (¿Problemas de diseño? ¡Escriba código adicional!) Con los estándares, esta forma de pensar desaparece para siempre. Pero la complejidad y las limitaciones del diseño CSS1/2 en ocasiones nos obligan a hacer lo mismo que queríamos evitar con su creación.

Con el modelo de cuadro IE4/5/Windows podemos crear diseños líquidos que combinen valores de longitud y porcentaje sin anidar `div` innecesarios o tener que devanarnos los sesos. No queremos afirmar que IE4/5/Windows sea el epítome de la compatibilidad con los estándares ya que seguramente no lo sea. Únicamente queremos decir que el modelo de cuadro incorrecto tiene sus peculiaridades.

Existe una técnica libre de divitis que nos permite crear diseños líquidos que combinen longitud y porcentaje en CSS, pero no es una técnica en la que ningún ser humano pensaría para hacer el trabajo y, como mucho, es una ciencia inexacta. El truco consiste en crear un único `div`, en utilizar la página como segundo `div` y en negociar con ambos por medio de `float`. Como es imposible calcular las dimensiones totales exactas, es necesario adivinar los números, con lo que nunca se conseguirá la suma exacta de 100 por cien.

Cuando la revista A List Apart se convirtió de tablas HTML a diseño CSS en febrero del 2001, se necesitaron tres diseñadores, incluyendo el aquí presente, para determinar la forma de hacerlo. Los otros dos eran los expertos Fahrner (que contribuyó a la especificaciones CSS) y Çelik (colaborador de las especificaciones CSS y creador del motor de representación Tasman de IE5/Mac). El diseño original basado en tablas (véase figura 12.6) estaba formado básicamente por dos columnas con una zona de contenido y una barra lateral a la derecha con una anchura pétrea. Entre los tres, conseguimos el mismo efecto en CSS (véase figura 12.7) por medio del método nada intuitivo mencionado en el párrafo anterior. La anchura total presenta valores que nunca suman 100 por cien. No debería resultar tan complicado crear un diseño de dos columnas que combinara elementos líquidos y fijos.

En la actualidad resulta más sencillo ya que los diseñadores disponen de la experiencia de A List Apart y de otros sitios CSS, y pueden copiar o modificar sus técnicas. Sepa que CSS 2.1 resuelve muchos de estos problemas y que las propuestas de CSS3 resuelve la mayoría. También debe saber que, al cierre de la edición de este libro, los navegadores principalmente admiten CSS1 con partes de CSS2.

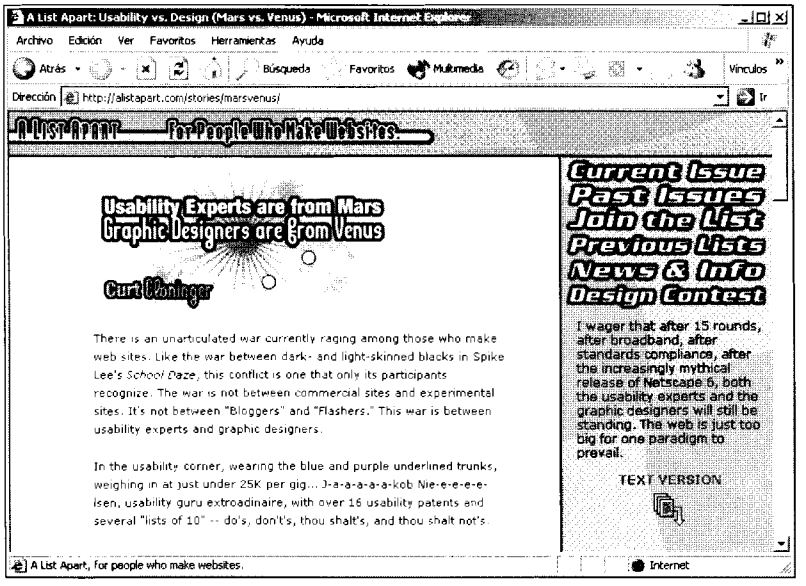


Figura 12.6.

El diseño original de la revista A List Apart (www.alistapart.com) consistía en celdas de tabla que cambiaban para ajustarse al monitor del visitante.

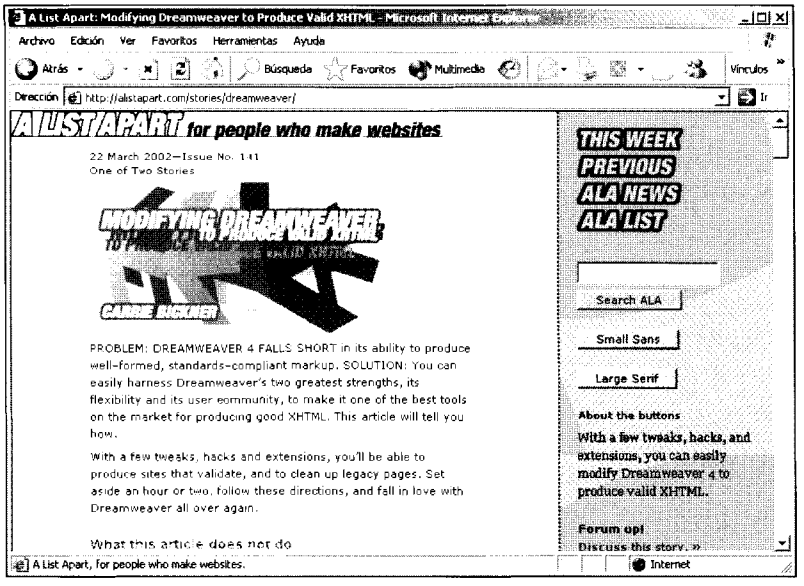


Figura 12.7.

Se necesitaron tres diseñadores CSS (dos de ellos reconocidos expertos) para determinar la forma de conseguir un aspecto operativo y visual similar por medio de hojas de estilo, durante el cambio de diseño CSS del sitio en febrero del 2001.

La complejidad del modelo de cuadro CSS1 sigue siendo un punto de bloqueo para los que quieren crear determinados tipos de diseños líquidos y va en contra de la forma de pensar de los diseñadores.

En una entrevista del 2002, el experto en DOM, Peter-Paul Koch, decía lo siguiente:

"Lógicamente, un cuadro se mide de lado a lado. Imagine un cuadro físico de cualquier tipo. Añada a su interior algo de menor tamaño. Pida a alguien que mida la anchura del cuadro. Medirá la distancia entre los lados del cuadro (los bordes). A nadie se le ocurriría medir el contenido del cuadro. Los diseñadores Web que crean cuadros para

almacenar el contenido, se preocupan por la anchura visible del cuadro, sobre la distancia de un borde a otro. Los bordes, y no el contenido, son las claves visuales para el usuario del sitio. A nadie le interesa la anchura del contenido."

Compleja o no, la especificación que tenemos es la que debemos utilizar. A los fabricantes de navegadores no se les pide que mejoren la especificación sino que la implementen de forma correcta y completa, y que todos los navegadores hagan lo mismo, ya sean actuales o anticuados. Se preguntará, entonces, cómo podemos crear diseños que funcionen con la misma corrección en navegadores que no entienden el modelo de cuadro como en los que sí lo hacen.

El problema del modelo de cuadro: seguridad de las CSS por motivos democráticos

Tantek Çelik, nombre que ya ha aparecido en este capítulo varias veces, nos ofrece una solución muy usada al problema generado por implementaciones incorrectas del modelo de cuadro. Esta solución se aprovecha de un error de análisis CSS en IE5/Windows para aplicar una anchura que admita IE5/Windows y, tras ello, reemplazarla con el valor real (véase figura 12.8). En este ejemplo, tomado de http://www.tantek.com/CSS/Examples/boxmodelhack.html, el valor correcto de la zona de contenido es 300 píxeles, pero IE/Windows lo malinterpreta y resta 100 píxeles de relleno y de bordes.

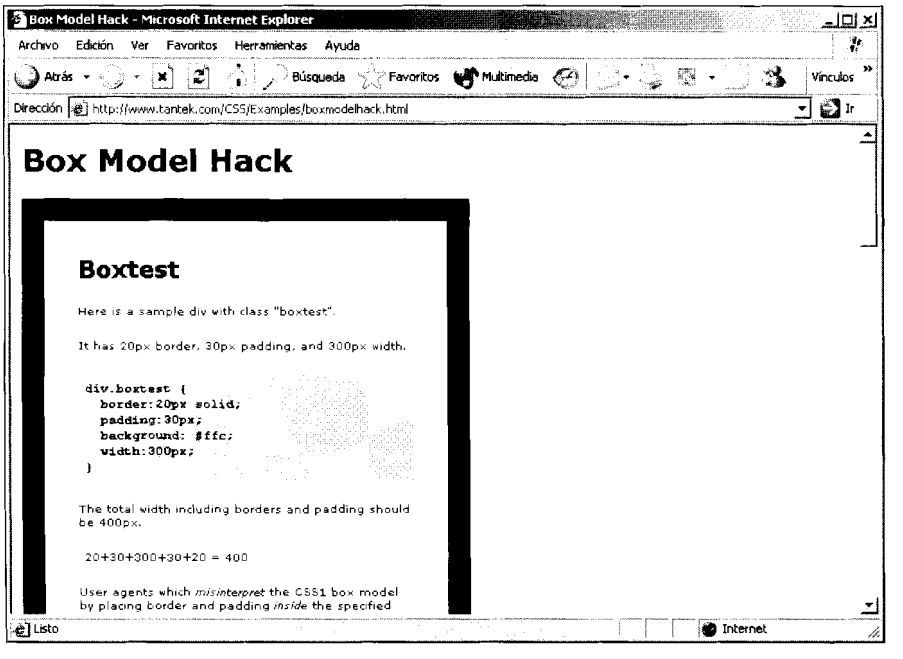


Figura 12.8.

La solución al modelo de cuadro de Tantek Çelik resuelve el problema y también resulta muy útil para corregir otros problemas de los navegadores (http://www.tantek.com/CSS/Examples/boxmodelhack.html).

En primer lugar, le mostramos la regla CSS sin corregir como la hubiéramos escrito para todos los navegadores:

```
div.boxtest {
    border:20px solid;
    padding:30px;
    background: #ffc;
    width:300px;
}
```

Los navegadores que comprendan CSS correctamente crearán un cuadro con una anchura total de 400 píxeles (20 + 30 + 300 + 30 + 20 = 400). Pero las versiones desfasadas de IE/Windows restan los valores de relleno y de borde de los 300 píxeles de la zona de contenido, lo que genera un cuadro de 300 píxeles de ancho y una zona de contenido de 200 píxeles de ancho (300 - 20 - 30 -30 -30 = 200). Por ello, les proporcionamos un valor falso de 400 píxeles (el tamaño total que deseamos) y les confundimos con dos líneas que contienen selectores CSS que IE4/5/Windows no comprenden (estas dos líneas se resaltan en negrita).

```
div.content {
    width:400px;
    voice-family: "\"}\"";
    voice-family:inherit;
    width:300px;
```

Para terminar, proporcionamos el valor correcto, los 300 píxeles. IE5.x/Windows deja de leer cuando se topa con reglas con no comprende. Los navegadores más compatibles siguen leyendo y utilizan este valor correcto.

Los navegadores de Opera (al menos los anteriores a Opera 7) presentan un problema adicional, ya que comprenden los selectores CSS2 y el modelo de cuadro pero padecen el mismo error de análisis que IE/Windows. Por ello, los navegadores de Opera también pueden utilizar los valores falsos, y no es lo que pretendemos. Tanket resuelve este problema mediante la creación de una regla "Sea amable con Opera" adicional:

```
html>body .content {
    width:300px;
}
```

Puede que Opera pase por alto el valor correcto en la regla anterior, pero utilizará el de la regla adicional ya que es más específico y, en CSS, la regla más específica prevalece sobre la menos específica (por ello, una regla `#menu` o prevalece sobre una regla `p`).

Desde su aparición a finales del 2000, esta solución se ha utilizado en miles de sitios para facilitar el diseño CSS. Para ello, se proporciona a los navegadores desfasados los valores incorrectos que requiere su modelo de cuadro incorrecto, a la vez que se proporcionan valores precisos para los agentes de usuario más compatibles. Resulta más complicado de explicar (y de leer en esta página) que de utilizar en la práctica. En nuestro caso lo utilizamos constantemente.

Sepa que esta solución no es necesaria cuando el relleno y los bordes se definen como cero. En función de la sensibilidad de su diseño y de los requisitos de un determinado proyecto, puede que tampoco sea necesario cuando los valores de relleno sean mínimos, el diseño no sea demasiado estricto y no le importe que el tamaño del cuadro varíe en unos píxeles de un navegador a otro. (En algunos diseños, también puede hacer trampas si selecciona valores que deliberadamente sumen menos de 100 por cien.)

El juego del escondite en los navegadores

La solución que hemos presentado, también conocida como método Tantek, depende en un error de análisis para engañar a las versiones desfasadas de IE/Windows para que admitan correctamente el modelo de cuadro. Como mencionamos en un capítulo anterior, el método `@import` para vincular a una hoja de estilo engaña a los navegadores 4.0 y anteriores para que rechacen dicha hoja de estilo. Con éstos y otros métodos, utilizamos los estándares para admitir los estándares. Puede parecer extraño, pero sólo significa que permitimos a los navegadores que ignoren lo que no comprenden. Y no lo hacemos mediante métodos de la vieja escuela como la división de código, sino dejando que los estándares básicos hagan lo mejor que saben hacer. En la dirección http://w3development.de/css/hide_css_from_browsers/summary/ encontrará métodos adicionales para ocultar diferentes elementos a distintos navegadores compatibles.

El error del espacio en blanco en IE/Windows

Para no aburrirle con demasiada teoría, analizaremos un error más sencillo con una solución menos compleja, el error del espacio en blanco en IE/Windows. En la figura 12.9 se muestran los estragos que este error puede producir en diseños híbridos pero que es igual de devastador en diseños CSS puros.

Los dos fragmentos de marcado que mostramos a continuación tienen la misma funcionalidad. Sin embargo, debido a la forma en que utilizan el espacio en blanco (o no lo hacen), se reproducirán de forma diferente en un navegador que intente equivocadamente analizar el espacio en blanco en XHTML. De esta forma:

```
<td><img src="foo.gif" /></td>
```

se representará de forma diferente a:

```
<td>
 <img src="foo.gif" />
</td>
```

El segundo ejemplo, el que incluye espacio en blanco en su marcado, puede generar huecos no deseados en la página Web, como se aprecia en la figura 12.9. Lo mismo sucede con el siguiente ejemplo. El siguiente marcado (sin espacio en blanco)

```
<td><a href="#foo"><img src="foo.gif"
/></a></td>
```

puede tener un aspecto diferente en su navegador que éste otro con la misma funcionalidad

```
<td>
 <a href="#foo">
  <img src="foo.gif" />
 </a>
</td>
```

Se preguntará a qué se debe. El error de espacio en blanco era un problema conocido en Netscape Navigator desde la versión 3.0 (o posiblemente antes). Cuando Microsoft se decidió a crear un navegador competente, sus ingenieros imitaron gran parte del comportamiento de Netscape incluyendo, desafortunadamente, algunos de sus errores.

Figura 12.9.

Un día más, un problema de representación más. En esta ocasión, se trata del error de espacio en blanco en IE/Windows. La imagen se ha ampliado para mostrar los detalles.

Al cierre de esta edición, los navegadores IE/Windows hasta IE6 continúan imitando este error de Netscape. ¿La solución? Eliminar el espacio en blanco del marcado.

El error de espacio en blanco puede resultar igual de dañino en diseños CSS como en diseños híbridos de CSS y tablas. Veamos la siguiente lista, obtenido de una versión de enero del 2003 de `zeldman.com`:

```
<div id="secondarynav">
<ul>
<li id="secondarytop"><a href="/about/"
title="History, FAQ, bio,
etc.">about</a></li>
<li id="contact"><a href="/contact/"
title="Write to us.">contact</a></li>
<li id="essentials"><a href="/essentials/
" title="Vital info."
>essentials</a></li>
<li id="pubs"><a href="/pubs/"
title="Books and articles."
>pubs</a></li>
<li id="tour"><a href="/tour/"
title="Personal appearances."
>tour</a></li>
</ul>
     ...</div>
```

La CSS (que veremos dentro de unos párrafos) transforma la lista en botones que el usuario puede pulsar, como se indica en la figura 12.10. Pero en IE/Windows, el espacio en blanco del marcado desarma el espaciado y el posicionamiento de los botones. Para solucionar este error, es necesario eliminar el espacio en blanco de esta forma:

```
<div id="secondarynav"><ul><li
id="secondarytop"><a href="/about/"
title="History, FAQ, bio, etc.">about
</a></li><li id="contact"><a href="/
contact/" title="Write to us.">contact
</a></li><li id="essentials"><a
href="/essentials/" title="Vital
info.">essentials</a></li><li
id="pubs"><a href="/pubs/" title="Books
and articles.">pubs</a></li><li
id="tour"><a href="/tour/"
title="Personal appearances.">tour</a>
</li></ul> ...</div>
```

Resulta más difícil de editar y de leer que la versión normal que utiliza espacio en blanco, pero no tenemos otra opción si queremos que nuestro sitio se vea correctamente en los navegadores más habituales. Al eliminar el espacio en blanco, hemos reducido el tamaño del archivo en un par de bytes, lo que es un pequeño consuelo. La CSS que genera estos efectos de botón de menú se reproduce a continuación y puede verse en http://www.zeldman.com/c/sophisto.css:

```
#secondarynav ul {
    list-style: none;
    padding: 0;
    margin: 0;
    border: 0;
    }
```

```
#secondarynav li   {
    text-align: center;
    border-bottom: 1px solid #066;
    width: 100px;
    margin: 0;
    padding: 0 1px 1px 1px;
    font: 10px/12px verdana, arial,
sans-serif;
    color: #cff;
    background: #399;
    }

#secondarytop   {
    border-top: 1px solid #066;
    }

#secondarynav li a {
    display: block;
    width: 96px;
    font-weight: normal;
    padding: 1px;
    border-left: 2px solid #6cc;
    border-right: 2px solid #6cc;
    background-color: #399;
    color: #cff;
    text-decoration: none;
    }

#secondarynav li a:hover {
    font-weight: normal;
    border-left: 2px solid #9cc;
    border-right: 2px solid #9cc;
    background-color: #5aa;
    color: #fff;
    text-decoration: none;
    }
```

Utilizaremos marcados y reglas prácticamente idénticas en nuestro cambio de diseño CSS en un capítulo posterior.

Si necesita más información sobre cómo se puede emplear CSS para convertir listas estructuradas en elementos de navegación gráficos, puede consultar el artículo de Mark Newhouse, "CSS Design: Taming Lists" (http://www.alistapart.com/stories/taminglists/).

El error float en IE6/Windows

Anteriormente en este capítulo explicamos cómo la propiedad `float` de CSS permite situar un elemento junto a otro. Por ejemplo, con la siguiente regla, un elemento `div` con el `id` exclusivo `maintext` se desplazaría a la izquierda de una barra lateral o de otro elemento de la parte derecha:

```
#maintext   {
    float: left;
    }
```

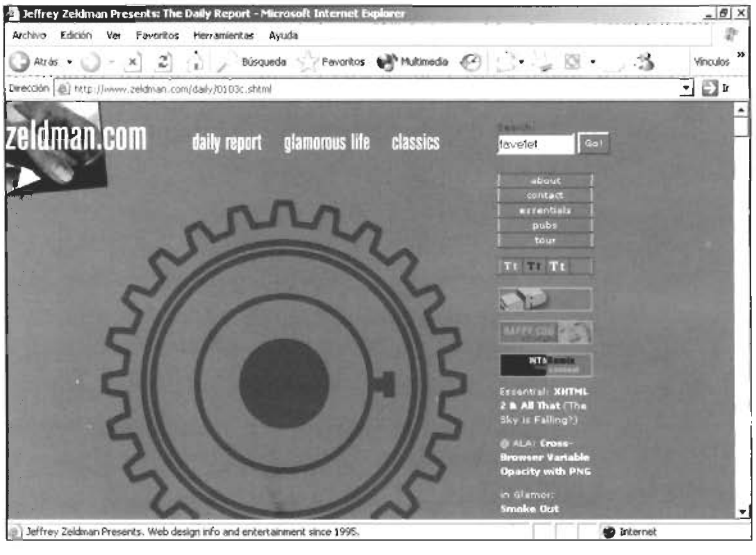


Figura 12.10.

Al aplicar reglas CSS a una lista sin ordenar, se crea un efecto visual de botones de navegación. El marcado de la lista debe compactarse para evitar el error de espacio en blanco en IE/Windows (www.zeldman.com).

Esperemos que el texto de `maintext` sea breve y conciso. IE6/Windows sufre un desafortunado problema. Los pasajes de texto extensos incluidos en un `div` flotante se recortan artificialmente, lo que impide que el lector vea el texto completo y también hace que desaparezca la barra de desplazamiento. Para poder ver la totalidad del texto o recuperar la barra de desplazamiento del navegador, el lector tiene que actualizar la página o pulsar **F11** dos veces rápidamente. ¿Los visitantes de su sitio pulsan **F11** rutinariamente cuando cargan cualquier página Web? Los nuestros tampoco.

El problema afecta a un desconocido (y esperemos que reducido) porcentaje de usuarios de IE6, pero es como decir que sólo afecta a un reducido porcentaje de la población china. El error se detectó por primera vez en el 2001, en la versión beta de IE6/Windows, y los ingenieros de Microsoft han tratado de solucionarlo. Sin embargo, al cierre de la edición de este libro, el error sigue sin corregirse.

Puede que los ingenieros de Microsoft sean incapaces de reproducir el error en laboratorio o de determinar su origen, pero la comunidad independiente de diseño parece haber descubierto la fuente del problema, así como su correspondiente solución. Siga leyendo.

Aférrese a los viejos valores

En apariencia, IE6 sufre para calcular la altura de los elementos de nivel de bloque. Eddie Traversa de http://dhtmlnirvana.com/ descubrió que IE6/Windows almacena en caché los valores que calcula en una página de un sitio y que, tras ello, aplica incorrectamente dichos valores a otras páginas. Por ejemplo, si el `div maintext` tuviera 300 píxeles en la primera página de un sitio y 1400 en la siguiente, IE6 únicamente mostraría los primeros 300 píxeles. Al cargar manualmente cada página se corrige este error, ya que se vacía la caché y se elimina el valor inicial, hasta que el lector acceda a otra página y comience un nuevo ciclo de valores incorrectos y texto ilegible.

Corregido con una secuencia de comandos

El primer paso consiste en reconocer que existe un problema. El segundo consiste en diagnosticar la naturaleza exacta de dicho problema. En el tercer paso, el programador Aaron Boodman de `Youngpup.net` escribió una función de JavaScript que, en presencia de IE6/Windows, itera una propiedad existente del bloque flotante, lo que provoca que la página vuelva a fluir y se muestre correctamente. Veamos la secuencia de comandos de Aaron:

```
if (document.all && window.attachEvent)
window.attachEvent("onload", fixWinIE);
function fixWinIE() {
    if (document.body.scrollHeight <
    document.all.content.offsetHeight) {
       document.all.content.style.display
       = 'block';
    }
}
```

Si lo desea, puede copiar y pegar estas líneas de JavaScript de la dirección http://www.alistapart.com/tightmen.js. Con estas líneas también podrá crear diseños CSS que utilicen `float` sin miedo. En su propia versión, sustituya las referencias a `.content` por el nombre de su `div`. Por ejemplo, si su

página incluye contenido flotante en un `div` con el nombre `maintext`, su código debería ser el siguiente:

```
if (document.all && window.attachEvent)
window.attachEvent("onload", fixWinIE);
function fixWinIE() {
   if (document.body.scrollHeight <
   document.all.maintext.offsetHeight) {
      document.all.maintext.style.display
      ='block';
   }
}
```

Otra técnica consiste en evitar todo esto por medio del posicionamiento CSS absoluto para imitar `float`, un método que aplicaremos en un capítulo posterior.

Flash y QuickTime: ¿objetos de deseo?

Muchos de nosotros incrustamos objetos multimedia como películas de Flash y QuickTime en nuestros sitios. No existe una forma compatible con estándares de hacerlo que funcione de forma fiable en diferentes navegadores y plataformas. Para comprender la razón, le contaremos una historia de orgullo y venganza tan melodramática como cualquier obra de Shakespeare o cualquier ópera italiana. Bueno, quizás no tan melodramática, pero casi.

Objetos incrustados: una historia de orgullo y venganza

Cuando a los creadores de los primeros navegadores Mosaic y Netscape se les ocurrió la brillante idea de permitir a los diseñadores incluir imágenes en las páginas Web, ampliaron HTML con la creación de una etiqueta `<img>` específicamente para sus navegadores. El W3C no lo aprobaba. Aconsejó a los creadores Web que utilizaran elementos `<object>`. Pero millones de sitios Web después, la etiqueta `<img>` seguía en circulación y la compatibilidad con el elemento `<object>` del W3C era inexistente.

En esto, apareció el complemento FutureSplash (más tarde denominado Flash) junto con otros elementos multimedia como las películas Real y QuickTime. De nuevo, el W3C sugirió que se debería utilizar la etiqueta `<object>` para incrustar este tipo de contenidos en una página Web. Pero, en su lugar, Netscape inventó la etiqueta `<embed>` y, cuando comenzaron a aparecer otros navegadores, también admitieron esta etiqueta de Netscape.

En opinión de Netscape y Microsoft, sus clientes esperaban que la Web funcionara como un espacio multimedia y era responsabilidad de los fabricantes de navegadores satisfacer este deseo a través de la innovación, aunque no consiguieran cuota de mercado en el intento.

En opinión del W3C, los fabricantes de navegadores creaban sus propias etiquetas e ignoraban especificaciones perfectamente correctas y estándar. Y, por ello, no tenía sentido crear especificaciones útiles y abiertas si las empresas miembro del W3C no prestaban atención a las mimas. En los años posteriores, el W3C se vengaría por partida doble de aquéllos que habían ignorado sus magníficos estándares.

(Bueno, está bien, no se vengaría de nadie ni nada parecido. Simplemente hicieron lo que

les mandaron y lo que creían que era lo correcto. Simplemente establecieron estándares de marcado que tenían sentido. Pero tiene más gracia si lo contamos de esta otra forma. Perdónenos.)

La doble venganza del W3C

El primer acto de venganza del W3C fue excluir la etiqueta <embed> de todas las especificaciones HTML oficiales, a pesar de que cientos de miles de diseñadores la utilizaban en millones de sitios. <embed> no se incluyó en HTML 3.2. Tampoco en HTML 4 ni HTML 4.01, cuyo único objetivo era incluir las etiquetas más utilizadas que no se habían incluido en HTML 4. Y como XHTML 1 se basaba en HTML 4.0, <embed> tampoco formaba parte de XHTML. Por esta razón, cualquier sitio que utilizara <embed> no podía validarse como HTML o XHTML (y sigue siendo así).

Es verdad. Ha leído bien. Millones de sitios que incrustaban elementos multimedia no podían validarse en función de una especificación del W3C ya que el W3C nunca reconoció el elemento <embed>.

Y como la herida seguía doliendo, y los deseos de venganza no desaparecían, el W3C eliminó el humilde elemento <img> de su especificación XHTML 2.0 original. ¿Quiere imágenes? Utilice <object>. ¿Quiere Flash? Utilice <object>. ¿Quiere QuickTime? Utilice <object>. Es lo que dice el W3C.

El problema es que la compatibilidad con <object> es mínima en algunos navegadores modernos y nula en los más antiguos. Y los diseñadores no van a dejar de utilizar Flash o de incrustar otros tipos de contenidos multimedia porque el W3C se oponga a las etiquetas que utilizan. Cuando un diseñador cree en los estándares Web y los utiliza pero necesita incrustar contenidos más completos, ¿qué puede hacer?

Incrustación de objetos multimedia y compatibilidad con estándares

En noviembre del 2002, Drew McLellan de Dreamweaverfever.com y el Web Standars Project llevaron a cabo un experimento. En un artículo para la revista A List Apart, Drew descartaba el marcado inválido utilizado universalmente para incrustar contenidos de Flash en una página Web y lo sustituyó por XHTML ligero y compatible (http://www.alistapart.com/stories/flashsatay/). Se desprendió del elemento <embed> no válido y reemplazó el marcado de estilo IE como el siguiente:

```
classid="clsid:D27CDB6E-AE6D-11cf-96B8-
444553540000"
```

por marcado compatible como este:

```
type="application/x-shockwave-flash"
```

El HTML que genera Flash al publicar una película suele tener este aspecto:

```
<object classid="clsid:D27CDB6E-AE6D-
11cf-96B8-444553540000"
   codebase="http://
download.macromedia.com
   /pub/shockwave/cabs/flash/
swflash.cab#version=6,0,0,0"
      width="400" height="300" id="movie"
align="">
      <param name="movie" value=
"movie.swf">
   <embed src="movie.swf" quality="high"
width="400"
   height="300" name="movie" align=""
   type="application/x-shockwave-flash"
```

```
   pluginspage="http://
www.macromedia.com/go/getflashplayer">
</object>
```

El HTML es pesado e inválido, pero funciona en cualquier navegador en el que se haya instalado el componente Flash. Drew lo redujo al siguiente XHTML ligero y válido:

```
<object type="application/x-shockwave-
flash" data="movie.swf"
    width="400" height="300">
    <param name="movie" value=
"movie.swf" />
</object>
```

No sólo se trata de una versión limpia, ligera y válida, sino que también reproduce películas de Flash en navegadores de Netscape e IE. Desafortunadamente, estas películas no se reproducen de forma fluida en IE/Windows debido a un error. Sin embargo, no es realmente grave si su contenido de Flash se limita a algunos K. Es más importante si los archivos de Flash son de mayor tamaño.

Drew resolvió este problema de IE/Windows utilizando una pequeña película de Flash para cargar otra de mayor tamaño. Parecía que definitivamente se había encontrado una forma compatible con estándares de incrustar contenidos multimedia.
Después de probar el método de Drew en todos los navegadores y plataformas disponibles, A List Apart publicó el artículo.

Una pega: fallos de objetos

Lamentablemente, una vez publicado el artículo, se descubrió otro problema. En lugar del contenido Flash incrustado, algunos visitantes veían una zona de texto vacía (véanse figuras 12.11 y 12.12).

Sucedía principalmente en navegadores IE/Windows (5, 5.5 y 6) aunque la mayoría de los usuarios de IE/Windows podían ver el contenido de Flash sin problemas. Los mismos problemas se apreciaban en Konqueror y Mozilla para Linux. Al igual que el error `float` de IE6 descrito anteriormente, el error de objeto afectaba a un porcentaje desconocido de usuarios.

¿Cuántos usuarios afectados? ¿Cientos? ¿Miles? ¿Millones? El número era tan importante que el foro de debate asociado al artículo se colapsó con informes de error (http://www.alistapart.com/stories/flashsatay/discuss/).

Según una comparación publicada sobre implementaciones de objeto en diferentes navegadores (http://www.student.oulu.fi/~sairwas/object-test/results/), IE/Windows y otros agentes de usuario que en ocasiones no representaban la película Flash, supuestamente debían comprender y procesar correctamente el elemento `<object>`.

IE/Windows presenta problemas similares cuando `<object>` se utiliza para incrustar una imagen, como recomienda XHTML 2. El método `<object>` de Drew parecía funcionar para la mayoría de los usuarios, pero puede que no para todos. Como se supone que `<object>` funciona y como la hace prácticamente en todos los navegadores la mayor parte del tiempo, algunos diseñadores utilizan el método `<object>`.

Pero para otros, la necesidad de incrustar contenidos multimedia sigue siendo incompatible con el objetivo del cumplimiento con los estándares.

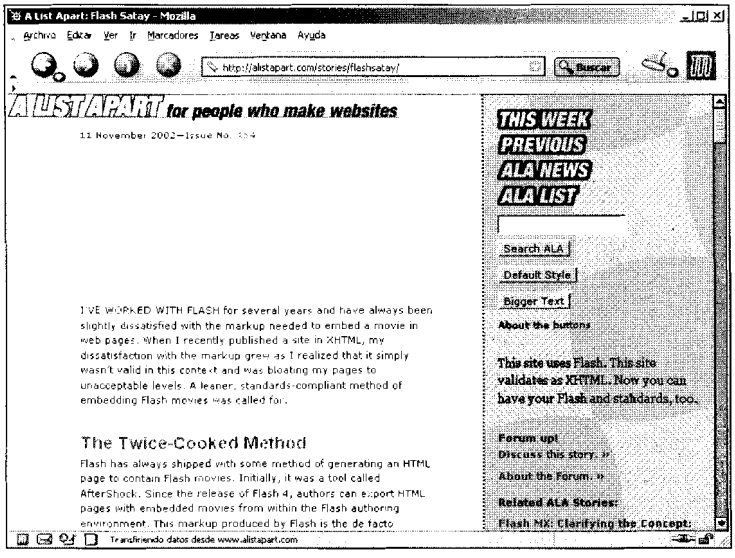


Figura 12.11.

El método de Drew utiliza marcado completamente válido para incrustar contenido Flash pero, misteriosamente, en algunos navegadores no funciona, como en el caso de Mozilla., aunque Flash esté instalado (http://www.alistapart.com/stories/flashsatay/).

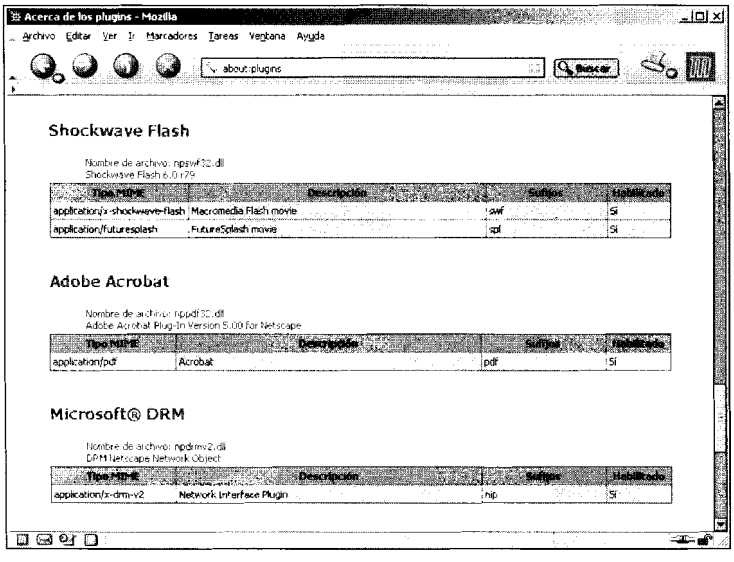


Figura 12.12.

Flash 5 está instalado y el marcado es válido, pero para este usuario en concreto, el archivo de Flash no se reproduce.

Cómo hacer trampas con document.write de Java

En un esfuerzo por incrustar contenido Flash y conseguir validación XHTML, algunos navegadores han usado la detección de navegadores de JavaScript y `document.write` para que los servicios de validación del W3C pasaran por alto la etiqueta `<embed>` inválida:

```
<!-- utilizado para crear xhtml válido
con la etiqueta embed -->
<script type="text/javascript">
//<![CDATA[
if (navigator.mimeTypes &&
navigator.mimeTypes["application/
x-shockwave-flash"]){
document.write('<embed src="/media/
yourflashmovie.swf" ...
```

El funcionamiento de esta técnica es la siguiente. Flash se representa correctamente

en todos los navegadores compatibles con JavaScript (a menos que el usuario se haya creído que JavaScript es el hombre del saco de los riesgos de seguridad y lo haya desactivado) y se engaña al servicio de validación del W3C para que crea que el marcado de la página es correcto.

Pero como dijimos al principio del capítulo, no creamos XHTML correcto para conseguir una medalla de oro. Engañar al servicio de validación no es lo mismo que conseguir la compatibilidad.

En la remotamente improbable circunstancia de que su cliente (o su jefe) le exija la compatibilidad con los estándares además de contenidos multimedia incrustados, el método JavaScript le permitirá cumplir ambos requisitos de forma aparente. Pero le recomendamos que sea honesto con su jefe o su cliente, que le explique los problemas descritos anteriormente con la mayor sencillez posible y que acepte que, por el momento, las partes de su sitio que utilicen contenidos multimedia incrustados no se validarán.

Con tan sólo mirar un periódico comprobará que vivimos en un mundo imperfecto, en el que las ideas nobles se enfrentan diariamente a la cruda realidad. La imposibilidad de conseguir la compatibilidad con XHTML y, al mismo tiempo, incrustar contenido Flash o QuickTime, es la menor de nuestras preocupaciones.

Cuando el procesamiento de `<object>` mejore en los navegadores, podremos utilizarlo sin temor para añadir contenidos multimedia a sitios perfectamente compatibles.

Un mundo de soluciones

Las soluciones nos permiten cumplir la promesa de los estándares (crear una vez, publicar en todas partes) a pesar de que ningún navegador sea perfecto y que algunos sean menos perfectos que otros. De esta forma, las soluciones nos liberan para que podamos mejorar el contenido, el diseño y el uso de nuestros sitios en lugar de malgastar nuestro tiempo en tecnologías propietarias sin salida.

Pero no todo el mundo está de acuerdo con las soluciones, ya sean prácticas o beneficiosas. Según algunas opiniones, si los navegadores no admiten de forma completa o correcta una especificación del W3C, pues peor para los usuarios de dicho navegador, o simplemente se evita el estándar. Esta perspectiva implica muchos problemas, como los que enumeramos a continuación:

- Si nos limitamos a las especificaciones que todos los navegadores admiten de forma completa y precisa, no podremos crear ninguna página Web. Ni siquiera HTML 3.2 se admite de forma universal y precisa.

- Llegar a entender los estándares lleva su tiempo, y el tiempo es un bien muy escaso en los mercados competitivos. En ocasiones, las presiones comerciales obligan a los ingenieros a comercializar productos antes de que hayan comprobado todos los detalles de una determinada especificación. Incluso pueden llegar a comercializar software sabiendo que es defectuoso. (Netscape 6.0 e Internet Explorer 6.0 incluían errores que sus

ingenieros conocían, como el error `float` de IE6 que mencionamos anteriormente.) Si fuéramos excesivamente puristas con respecto al uso de soluciones, los visitantes de nuestros sitios sufrirían innecesariamente.

- En ocasiones, las especificaciones son vagas en determinados aspectos, y algunas cambian después de publicarse. Es el caso de CSS2. Como mencionamos algunas páginas atrás, CSS2 se revisó en CSS2.1 en el 2002 porque la versión original contenía ciertas ideas oscuras y que no funcionaban. Cuando se mueven los postes de la portería, los equipos titubean. Del mismo modo, en la especificación CSS1 original, el factor de escala entre palabras clave de tamaños de fuente adyacentes era de 1.5. De todos los navegadores, Netscape 4 era el único que implementaba la escala de palabras clave exactamente como indicaba el W3C. Qué ocurrió. Que las fuentes pequeñas eran demasiado pequeñas y que las fuentes grandes eran demasiado grandes. La especificación cambió posteriormente.

- La mayoría de nosotros tenemos que compensar los errores de agentes de usuario antiguos, en especial de aquellos agentes que se utilizan masivamente, como ocurre con IE5.x/Windows, con su modelo de cuadro CSS incorrectamente implementado.

Los lectores que teman que al recomendar este tipo de soluciones les queremos conducir a un universo moralmente cuestionable, deben saber que todas las soluciones compartidas en este capítulo son perfectamente compatibles. No estamos escribiendo código propietario ni marcado basura para superar las limitaciones de los navegadores. Estamos utilizando los estándares para admitir los estándares. Por ejemplo, en la solución del problema del modelo de cuadro, utilizamos selectores CSS2 para garantizar que los navegadores con modelos de cuadro incorrectos procesen correctamente nuestros requisitos de diseño CSS1. Eliminamos espacio en blanco innecesario de nuestro marcado e intentamos crear texto legible y accesible para todos.

No hay nada malo con estas técnicas y sí muchos aspectos positivos. Nos permiten utilizar los estándares en la actualidad en lugar de fijar la mirada en un horizonte lejano repleto de software de navegación perfecto. Cuando los navegadores mejoren, y los navegadores antiguos caigan en desuso, necesitaremos menos soluciones, aunque continuaremos con su aplicación para garantizar que nuestros sitios tengan tanta compatibilidad inversa como directa. En el siguiente capítulo finalizaremos nuestro breve análisis de las penas y glorias de los navegadores, y examinaremos uno de los aspectos básicos del diseño: el control de la tipografía.

Capítulo 13
Cómo trabajar con navegadores (parte III). Tipografía

Junto con la ubicación y el color, la tipografía es una herramienta de diseño básica y fundamental. Los diseñadores impresos dedican años al estudio de la historia y la aplicación de la tipografía. Aprenden a distinguir entre tipos de letra que, para los no iniciados, parecen idénticos, como Arial y Helvetica. Cuando estos diseñadores de educación tradicional llegan a la Web, con sus limitados y contradictorios conjuntos de herramientas tipográficas, suelen estar menos equipados para navegar por sus aguas que aquéllos con una formación en diseño menos tradicional.

El tamaño importa

Windows, UNIX y Macintosh incluyen diferentes tipos de fuente con diferentes resoluciones predeterminadas y, a menudo, con diferentes estilos de representación predeterminados, desde pixelados a suavizados (como ocurre en Mac OS 9), hasta tipos de letra tan suaves que se parecen al texto de Photoshop (como ocurre en Mac OS X Jaguar de forma predeterminada y en Windows XP con Cleartype, pero sólo si el usuario activa esta opción). Por esta razón, la antigua etiqueta `<font size=3>` tiene significados diferentes en función del sistema operativo, no sólo en tamaño, sino también en apariencia. Desafortunadamente, la mayoría de los métodos CSS de cambio de tamaño de fuentes también provocan un cambio de tamaño de un sistema operativo a otro, a pesar de los esfuerzos de los fabricantes de los principales navegadores por reducir o eliminar los problemas entre plataformas mediante un tamaño de fuente predeterminado estándar.

Control de usuario

Además de las diferencias entre plataformas, de los métodos desfasados y de las representaciones CSS, la Web difiere de la impresión en que el usuario supuestamente tiene cierto control sobre lo que ve. La acomodación de los usuarios resulta tan compleja con los estándares como lo era con los métodos de la vieja escuela, debido a los problemas que mencionaremos en este capítulo (y ya verá si los mencionamos). También resulta complicado que los diseñadores con formación tradicional acepten la premisa del control de usuario. Desafortunadamente, los métodos CSS (emes, porcentajes, palabras clave de tamaño de fuente) que deben encargarse del control de usuario, sufren los problemas que aparecen entre navegadores y entre plataformas. Durante un breve periodo de tiempo, estos problemas se solucionaron con la buena voluntad de los fabricantes de navegadores, ya que acordaron admitir de forma uniforme un tamaño estándar entre plataformas. Pero los nuevos navegadores echan por tierra estos logros, lo que dificulta la tarea de equilibrar los requisitos de diseño con la necesidad de control del usuario.

En este capítulo, analizaremos la historia y los problemas de la tipografía Web, y estudiaremos dos métodos de configuración a través de los estándares Web. Ambos métodos funcionan, pero ninguno es perfecto. También mencionaremos otros métodos que son menos perfectos y que generan más problemas. El 13 es el número de la mala suerte; los diseñadores preocupados por el diseño y también por los usuarios se enfrentan a decisiones duras y, en ocasiones, desafortunadas, como veremos en este capítulo. Analizaremos las técnicas, sus ventajas y sus riesgos, y concesiones ocasionales. Como siempre, la decisión de qué concesiones son las adecuadas para su público es completamente suya.

Miedos de la vieja escuela

Tim Berners-Lee, el inventor de la Web y fundador del W3C, imaginó su creación como un medio para el intercambio de documentos de texto; por esta razón, no incluyó control tipográfico en el lenguaje estructurado HTML.

Como mencionamos en uno de los primeros capítulos, los diseñadores Web utilizaban `<tt>`, `<pre>`, `<blockquote>` y otras etiquetas sin sentido semántico para cambiar de tipo de letra, conseguir efectos de posición y simular el interlineado. Tras ello, comenzaron a utilizar imágenes GIF o Flash de texto, una práctica que se mantiene en la actualidad (véase figura 13.1), a menudo a cargo de diseñadores a los que les resultaba difícil aceptar las limitaciones de CSS y XHTML, y las concesiones inherentes al enfrentamiento entre las necesidades del diseñador y de los usuarios.

En 1995, con la aparición masiva de sitios comerciales, era necesario hacer algo para el menos contar con herramientas tipográficas básicas. Netscape nos ofreció la etiqueta `<font>`, cuyo atributo era el tamaño. Podíamos especificar cifras (`<font size=2>`) o cifras absolutas basadas en los valores predeterminados del usuario (`<font size=+1>`).

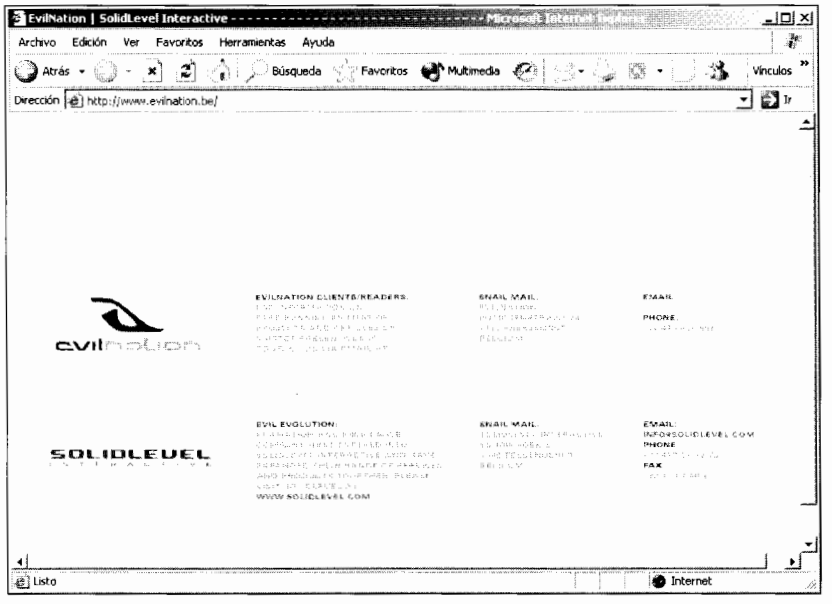


Figura 13.1.

Todas las palabras de esta página (http://www.evilnation.be/) son vectores Flash, no texto real. Por ello, no se puede buscar, no es accesible y tampoco se puede copiar ni pegar. Los diseñadores que se preocupan por la tipografía a menudo tienen dificultades para aceptar las limitaciones de XHTML y CSS, y las concesiones de control de usuario.

Los diseñadores abandonaron con rapidez las etiquetas de párrafo y otros elementos estructurales, y controlaron sus diseños mediante la combinación de `<font size...>` con etiquetas `<br>`. Para no quedarse fuera, Microsoft nos proporcionó `<font face...>`.

La mayoría de los lectores de este libro recordarán aquella época y también recordarán los problemas, sobre todo los originados por las diferencias entre plataformas. Windows asumía un tamaño base predeterminado de 16 píxeles a 96 ppp (puntos por pulgada). Macintosh, muy unido al diseño impreso, asumía un tamaño de 12 píxeles a 72 ppp, basado en el estándar PostScript. Los tamaños de fuente que resultan perfectos en una plataforma, resultan demasiado grandes o demasiado pequeños en la otra.

Los diseñadores de Mac lo achacaban a la estupidez de Windows.

Los diseñadores de Windows lo achacaban a la estupidez de Macintosh.

Puntos de diferencia

Seguidamente aparecieron las primeras implementaciones de CSS como la de IE3, lo que alivió ligeramente el problema entre plataformas. Los puntos son unidades de impresión, no de pantalla, pero los diseñadores están familiarizados con los puntos y muchos optaron por utilizarlos para especificar el texto Web que empleaban. En el mundo de Windows, las fuentes de 7 puntos tenían una altura de 9 píxeles, el umbral más bajo de legibilidad. En Mac, las fuentes de 7 puntos tenían una altura de 7 píxeles, lo que las hacia ilegibles, y tan inútiles como una barba en un bebé.

En 1997, `Microsoft.com` seleccionó fuentes de 7 puntos (véase figura 13.2) para

promocionar las habilidades CSS de su nuevo navegador IE3 para Windows y Macintosh. Era como invitar a alguien a una presentación de una película y desenfocar el proyector antes de proyectarla. Las fuentes eran igual de ilegibles en IE4.x y en Netscape 4 para Macintosh, pero no debido a problemas del navegador, sino a las diferencias entre plataformas.

Todd Fahrner, que sería el padre de la conmutación DOCTYPE (consulte un capítulo anterior), añadió la figura 13.2 a su sitio personal, para demostrar que los puntos son una unidad inútil de CSS en términos de diseño de pantalla (aunque resultan correctos para hojas de estilo impresas).

Fahrner también apuntó que el texto definido en puntos y píxeles no se podía cambiar de tamaño (en aquel momento) por medio de la utilidad de ajuste de tamaño de fuente de los navegadores. No se debía a que CSS1 considerara los puntos y los píxeles unidades fijas. (CSS1 no definía ningún método de cambio de tamaño como imposible de llevar a cabo.) Simplemente se trataba de una decisión tomada por los primeros fabricantes de navegadores para empezar a admitir CSS. El usuario podía cambiar las fuentes definidas en emes y porcentajes, pero IE3, IE4 y Netscape 4 generaban terribles errores cuando se utilizaba este tipo de fuentes. Fahrner no tardaría en proponer una solución para alguno de estos problemas.

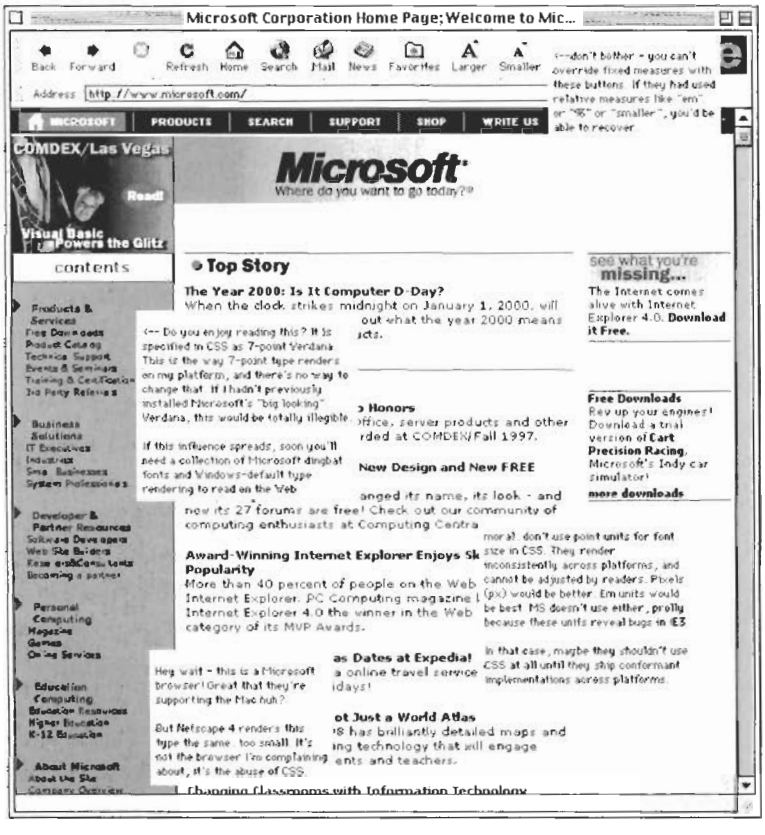


Figura 13.2.

En esta captura de pantalla de mediados de los 90, Todd Fahrner documentó problemas de una incorrecta utilización de CSS que incluía fuentes de pantalla especificadas en puntos (http://style.cleverchimp.com/font_size/points/font_wars.GIF). Este tipo de fuentes era ilegible en Macintosh debido a las diferencias entre plataformas.

En el momento en que se tomó la captura de pantalla anterior (véase figura 13.2), la única unidad de tamaño que funcionaba correctamente en diferentes navegadores y plataformas era el píxel, que no se podía cambiar de tamaño. Cinco años después, la única unidad fiable sigue siendo el píxel, y sigue sin poderse cambiar de tamaño en IE/Windows.

Al fin un tamaño estándar pero, ¿por cuánto tiempo?

En un esfuerzo por superar las diferencias entre plataformas, los fabricantes de Netscape, de los navegadores de Microsoft y de Mozilla, se unieron a finales de 1999 y decidieron estandarizar un tamaño de fuente predeterminado de 16 píxeles o 96 ppp en todas las plataformas. Al hacerlo, los fabricantes de navegadores y los usuarios podían evitar los problemas generados por las diferencias de tamaño entre plataformas y el texto ilegible e inútil.

Netscape 6+, Mozilla, IE5+/Macintosh e IE/Windows ofrecían a todos los usuarios el mismo parámetro de tamaño de fuente predeterminado. Mientras que el usuario no modificara sus preferencias, tanto puntos, como emes, porcentajes y palabras clave de tamaño de fuente funcionarían de la misma forma entre las distintas plataformas, y ningún usuario se vería innecesariamente afectado por la ignorancia de un diseñador o programador sobre las diferencias entre plataformas. Seguía habiendo errores y problemas de herencia en la implementación por parte de algunos navegadores de porcentajes y emes (como veremos más adelante) pero el primer escollo se había salvado.

Desafortunadamente, los fabricantes de navegadores no explican por qué hacen lo que hacen, y los usuarios de la Web no estudian problemas de diseño (ni deberían). La ventaja de un tamaño estándar se evaporó cuando los usuarios cambiaron sus preferencias y los diseñadores no se enteraron de nada, literalmente.

El estándar de tamaño de fuente de 16 píxeles/96 ppp

En 1998, en una lista de correo del W3C, Todd Fahrner propuso maliciosamente la estandarización de un valor predeterminado de 16 píxeles para su utilización en Windows. En el 2000, su recomendación fue adoptada por los principales fabricantes de navegadores. Aunque las preocupaciones de Fahrner eran de naturaleza práctica (quería asegurarse de que las fuentes se pudieran leer en línea), en la siguiente cita se refiere a algo que puede parecerle extraño.

CSS se veía limitada a una abstracción en la que la longitud media del brazo de un usuario Web resultaba fundamental para definir el tamaño de un píxel. En serio. De cualquier modo, al presentar su idea sobre un tamaño de fuente estandarizado entre plataformas, Fahrner se encargó de cubrir el importante problema de la longitud del brazo con temas más terrenales como la facilidad de uso. Escribió esto:

"Desde Mosaic, el valor de tamaño de fuente predeterminado en los principales navegadores se ha definido en 12 puntos. Propongo volver a definir este valor predeterminado en 16 píxeles... El valor actual se rasteriza de forma diferente en las distintas plataformas. En Mac, lo hace a 12 píxeles (con una resolución lógica fijada en 72 ppp). En PC Wintel, lo hace de forma predeterminada a 16 píxeles (con una resolución lógica predeterminada de 96 ppp).

Todos los valores de tamaño de fuente escalables funcionan con respecto a esta inconsistente rasterización base. Para un diseñador, significa que la única forma de sugerir un tamaño de fuente (coherente entre las diferentes plataformas) es utilizar unidades de píxel CSS, que el usuario no puede escalar y, por lo tanto, no son óptimamente portables ni indicadas para el usuario.

La medida correctora adecuada consiste en que los navegadores Mac (y X11) rompan con la tradición e incorporen el valor predeterminado de un texto normal de 16 píxeles, en lugar de los 12 puntos. Evidentemente, está sujeto a los cambios que realice el usuario, pero un valor inicial que sea coherente al menos conseguirá que el uso de valores de tamaño de fuente escalables por parte de los diseñadores resulte menos problemático, ya que cualquier variación con respecto al tamaño predeterminado se deberá expresamente a la preferencia del usuario y no a las caprichosas diferencias legadas de los sistemas operativos.

Si los diseñadores tienden a creer que un valor de 16 píxeles es excesivo para utilizarlo como base, ¿por qué sugerirlo como predeterminado?

Una de las razones es por pura conveniencia. Mac es una plataforma minoritaria, aunque con una fuerte presencia en el campo del diseño Web (yo utilizo Mac). Es poco realista pensar que los navegadores Windows/X11 cambiarán sus valores predeterminados para que coincidan con la curiosa limitación de Mac a una resolución lógica de 72 ppp.

El estándar CSS1 de 1996 sugiere un valor de 1/90" para un píxel de referencia, extrapolado desde un ángulo visual de 0'0227 grados de la longitud del brazo. Se supone que los agentes de usuario deben escalar los píxeles correctamente si la resolución física varía significativamente con respecto a este valor. Un píxel de referencia de 1/90" sugiere una rasterización de 12 puntos en 15 píxeles, no en 16, aunque se acerca mucho más a 16 que a 12. Como ningún sistema operativo ni agente de usuario actual asume una resolución lógica de 90 ppp (ni tampoco implementa la escala de píxeles), el valor del píxel de referencia debe adjuntarse a 1/96". Es más sencillo preservar al ángulo visual sugerido de 0'0227 grados si asignamos al usuario de referencia una mayor longitud de brazo.

Los diseñadores piensan que un valor de 16 píxeles es excesivo porque están acostumbrados al tamaño base de 12 píxeles

de sus equipos Mac. La legibilidad tiene un 90 por ciento de familiaridad."

`http://lists.w3.org/Archives/Public/www-style/1998Dec/0030.html`

Dos años más tarde, la primera generación de navegadores con cierta compatibilidad con estándares, se unieron a la recomendación de Fahrner. En Internet Explorer, en Netscape y en Mozilla, tanto para Windows como para Macintosh, se adoptó como tamaño de texto predeterminado 16 píxeles/96 ppp, y el aumento de legibilidad se celebró por todo el mundo (excepto cuando los usuarios lo volvían a cambiar).

Con el tiempo, los esfuerzos de Fahrner consiguieron que el W3C admitiera un tamaño de píxel de referencia predeterminado relacionado con el concepto de 16 píxeles/96 ppp, como se puede comprobar en el borrador de la CSS 2.1.

En el siguiente fragmento, comprobará que el W3C sigue preocupado por la longitud media del brazo del usuario. Deberíamos estar agradecidos de que sólo hablen de la longitud de un brazo:

Se recomienda que el píxel de referencia sea el ángulo visual de un píxel en un dispositivo con una densidad de píxeles de 96 ppp y a una distancia del lector equivalente a la longitud de un brazo. Para una longitud de brazo nominal de 28 pulgadas, el ángulo visual sería de aproximadamente 0'0213 grados (http://www.w3.org/TR/CSS21/syndata.html#length-units).

Aunque el píxel de referencia estándar parecía preparar el camino para un tamaño predeterminado estándar, los dos no están oficialmente relacionados. El W3C se ha pronunciado sobre el primer aspecto, no sobre el segundo.

Todo lo bueno desaparece con un clic

Un porcentaje desconocido de usuarios de Macintosh, disgustados por los tamaños de texto que percibían (correctamente) de forma excesiva y nada atractiva, recuperaron inmediatamente el antiguo valor de 12 píxeles y 72 ppp en sus navegadores, lo que anuló el intento de estandarización y les impidió de nuevo poder leer el texto de muchos sitios. En la Web, se supone que el usuario está al mando, incluso cuando no sabe qué es lo que más le conviene.

Cuando los seguidores de este cambio a los 12 píxeles son diseñadores, puede que consigan sitios con un aspecto perfecto en Macintosh pero que sufran en el dominante espacio de Windows. Muchos diseñadores que utilizan Macintosh emplean equivocadamente los 12 píxeles y 72 ppp y, como resultado, su trabajo (o su público) son los que lo sufren.

Evidentemente, muchos usuarios de Windows también opinan que el valor predeterminado de 16 píxeles es excesivo y, por ello, configuran sus navegadores para ver

todos los sitios Web con un parámetro inferior al que la norma propone. Cuando los usuarios de Windows y Mac hacen este tipo de cosas, no tiene efecto en los sitios que utilizan píxeles para especificar el tamaño de las fuentes, pero sí en los que emplean emes, porcentajes y (en menor medida) palabras clave CSS de tamaño de fuente. Examinaremos estos dilemas a lo largo del capítulo.

La reacción equivocada al cambio en los navegadores

Los usuarios que cambiaron a una fuente de tamaño inferior al estándar simplemente buscaban la comodidad. No eran los únicos que no entendían la importancia del tamaño estándar, ni tampoco se esperaba que lo hicieran. Muchos profesionales de la Web tampoco lo entendieron, en parte debido a que Netscape y Microsoft no lo anunciaron.

En concreto, la Browser Quirks Brigade, que creía que los navegadores siempre difieren en apariencia y comportamiento, y que la forma de resolver estas diferencias era combinar detección de navegadores con división de código y sopa de etiquetas, rápidamente hizo lo que le habían enseñado.

Antes del 2000, en lugar de utilizar píxeles para garantizar que el texto se representaba con el mismo tamaño prácticamente en todos los navegadores y plataformas, estos programadores habían utilizado puntos. Como los puntos carecían de sentido en términos de pantalla y se implementaban de forma diferente en las dos principales plataformas informáticas, dichos programadores crearon múltiples hojas de estilo basadas en puntos, recurriendo a la detección de plataformas para que una hoja de estilo basada en puntos sirviera para los usuarios de Windows y otra para los de Macintosh. A menudo, era mucho más complicado, pero todavía no hemos desayunado.

El pez que se muerde la cola. CSS condicional

Con la aparición en el 2000 de los navegadores compatibles con estándares que admitían el mismo tamaño de fuente predeterminado y la misma resolución en todas las plataformas, los diseñadores Web tuvieron la oportunidad de abandonar la detección de navegadores y las CSS condicionales basadas en puntos. En lugar de eso, la Browser Quirks Brigade actualizó sus secuencias de comandos y utilizó archivos CSS condicionales adicionales. Condicionales adicionales suena divertido, ¿verdad? Pero los resultados no lo eran.

En lugar de funcionar como se esperaba, estas secuencias de comandos ampliadas y archivos adicionales iban en contra de su objetivo principal, lo que generaba páginas ilegibles o con un formato extraño como hemos mencionado en capítulos anteriores. Las secuencias de comandos y las hojas de estilo erróneamente concebidas que solían utilizar se comportaban como un camarero robotizado estropeado.

Navegadores: Todos cenaremos pollo.

Camarero: Muy bien, entonces dos hamburguesas y un plato vegetariano.

Navegadores: No, pollo. Tomaremos todos pollo, gracias.

Camarero: Perdone un segundo (dirigiéndose a un determinado navegador). Sé lo que usted quiere. Quiere helado con su hamburguesa

Navegadores: Pollo, pollo, todos queremos pollo.

Camarero: ¿Verdad que quiere helado? Seguro que sí, ¿a quién no le gusta el helado?

Navegadores: (Comienzan a desesperarse) Pollo.

Camarero: ¡Y nata montada! Aquí tienen, tengan un buen día y paguen al salir.

La mayoría de los programadores responsables de estas pesadillas no eran estúpidos. A menudo, tenían grandes conocimientos, una gran experiencia y (seamos honestos) eran profesionales muy bien remunerados. Sus clientes pagaban grandes cantidades de dinero por sitios innecesariamente complicados que nunca funcionaban correctamente y que requerían un mantenimiento constante y caro para que continuaran fallando de formas más complejas y a mayor precio. Parece algo que nadie en su sano juicio desearía pero era la norma habitual (y en muchos casos continúa siéndolo).

Con el tiempo, se acababan los fondos y se evaporaban los presupuestos, y tanto el propietario como los visitantes se estancaban con un sitio obsoleto, como mencionamos en los primeros capítulos. Acuciados por la sombra del desastre y en un intento de anticiparse a lo inevitable, algunos programadores incluían un aviso de tipo "Se aconseja verlo con..." en su página inicial. Es como añadir una advertencia de las autoridades sanitarias a un paquete de tabaco y esperar que se convierta en un frasco de vitaminas.

Este caos se puede evitar si comprendemos que los navegadores admiten estándares comunes y que la mejor solución suele ser la más sencilla. (Es decir, utilizar la misma hoja de estilo en todos los navegadores y evitar el uso de puntos excepto en impresión.) Afortunadamente ha comprado este libro y desafortunados todos aquéllos que creen que todos los navegadores deben actuar de forma diferente y que con un presupuesto desbordante y un doctorado en morderse la cola, se puede resolver el inexistente problema de la diferencia entre plataformas mediante circunvoluciones tipo Escher. Vaya frase.

Chimera y Safari: magnífico rendimiento, avergonzados por el tamaño

Al cierre de la edición de este libro, los nuevos navegadores para Mac OS X, aunque excelentes e inspiradores en muchos aspectos, desafortunadamente deshacen el camino andado en el 2000 cuando los tamaños de fuente estándar eran una preocupación. El navegador Chimera basado en Gecko (Navigator) y el navegador Safari basado en HTML de Apple, en sus versiones beta al cierre de esta edición, se han distribuido ampliamente y han ganado gran cantidad de adeptos. Ambas versiones ofrecen una sólida compatibilidad con CSS y otros estándares, y Chimera tiene un mérito especial ya que combina la compatibilidad con los estándares de Mozilla y Gecko con la suave

representación de texto y las avanzadas opciones de usuario de OS X. Cuando lea este libro, puede que Chimera haya cambiado su nombre por Camino, por causas legales (http://www.mozilla.org/projects/camino/).

Desafortunadamente, ni Chimera ni Safari cuentan en la actualidad con el tamaño de fuente predeterminado de 16 píxeles. En ambos viene configurado a 14 píxeles, que ni es el estándar ni el valor predeterminado de Macintosh anterior al 2000. Parece que los fabricantes hacen apuestas compensatorias y proporcionan a los usuarios de Mac fuentes que no llegan al estándar pero que esperan que no sean demasiado pequeñas para representar correctamente el texto.

Puede que sea lo que desean los usuarios de Macintosh, pero es una pendiente peligrosa. De hecho, en los paneles de preferencias (véanse figuras 13.3 y 13.4) de ambos navegadores resulta complicado cambiar a 16 píxeles, ya que no se ofrece como opción predeterminada, aunque se puede introducir el valor manualmente si el usuario sabe cómo hacerlo y está dispuesto a ello.

Algunos usuarios cambian los tamaños de fuente predeterminados de sus navegadores, pero muy pocos basan dichas modificaciones en una conciencia arcana y preternatural de los estándares. Por ello, en ambos navegadores, con su configuración predeterminada, a menos que el texto de una página Web se haya especificado en valores de píxel, los usuarios de Mac tendrán una vez más que lidiar con texto de tamaño reducido que puede resultar ilegible.

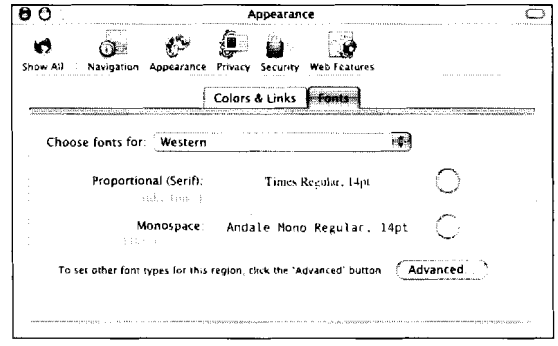


Figura 13.3.

El navegador Camino (Navigator) basado en Gecko para OS X ofrece una magnífica compatibilidad con los estándares pero, desafortunadamente, no incluye el tamaño de texto predeterminado de 16 píxeles. Tampoco en el panel de preferencias se indica la forma de cambiar a este tamaño (en caso de que los usuarios supieran hacerlo).

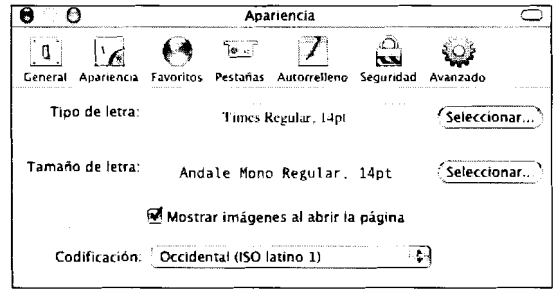


Figura 13.4.

El navegador beta Safari de Apple, sorprendentemente compatible con los estándares, también incluye un tamaño de fuente no estándar y en su panel de preferencias en tan poco probable que el usuario sepa seleccionar el parámetro estándar como en el de Camino. Una vez más, los usuarios de Mac tendrán que soportar texto a menor tamaño.

El efecto de tamaños no estándar en tratamientos de texto accesibles

En las cinco imágenes siguientes (véanse figuras 13.5, 13.6, 13.7, 13.8, 13.9 y 13.10) se muestra el sitio i3Forum que creamos en capítulos anteriores, en el que se utilizan emes para que los usuarios puedan cambiar el tamaño del texto en cualquier navegador, incluso en el desesperadamente tozudo Internet Explorer para Windows. Apreciará variaciones en la altura de la página puesto que cada navegador cuenta con diferentes profundidades de interfaz de usuario y que hay una pérdida de espacio vertical en la versión gratuita de Opera (véase figura 13.8) ya que incluye anuncios publicitarios para subvencionar el coste de programación.

A pesar de estas diferencias, se puede apreciar que el texto tiene el mismo tamaño en Netscape 7/Macintosh (véase figura 13.5), IE5/Macintosh (véase figura 13.6), IE6/Windows (véase figura 13.7) y Opera 7/Windows (véase figura 13.8).

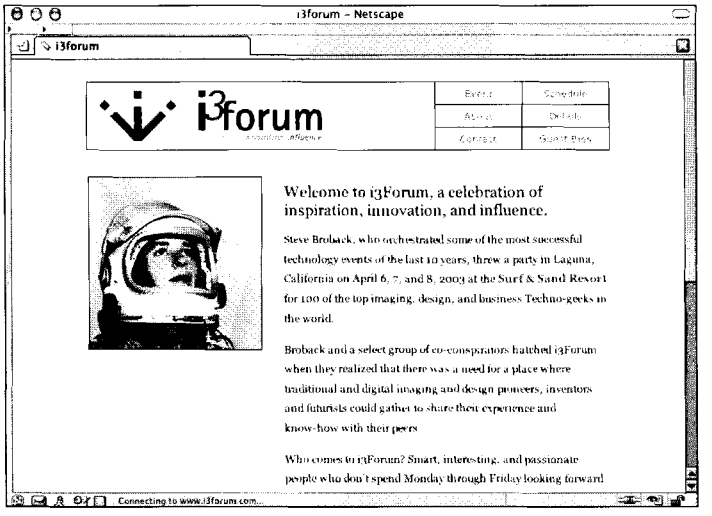


Figura 13.5.

El sitio i3Forum, creado en capítulos anteriores, utiliza emes para controlar el tamaño de las fuentes. En este caso lo vemos en Netscape 7 para Macintosh (www.i3forum.com).

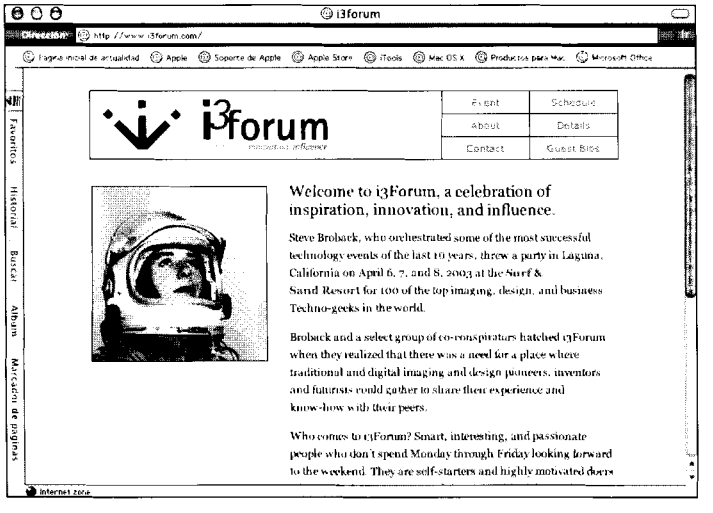


Figura 13.6.

El mismo sitio de nuevo, en IE5/Mac. Aunque la página parece ligeramente más profunda debido a diferencias del navegador, el tamaño del texto es idéntico.

Esto sugiere que la detección de navegadores y plataformas, así como el uso de CSS condicional para ofrecer diferentes tamaños de punto para las distintas plataformas, es una carga que ningún diseñador debe desear y que ningún cliente debería padecer.

También puede comprobar que el texto es ligeramente más pequeño en Chimera/Camino (véase figura 13.9) y en Safari (véase figura 13.10) que en los otros. De hecho, el texto es más pequeño, un problema que se aprecia con mayor claridad en fuentes sarif como la que hemos utilizado en este ejemplo (Georgia) que en fuentes sans serif, que resulta legible incluso con tamaños pequeños. Puede que a algunos usuarios de Camino y Safari les resulte difícil leer el sitio, pero pueden cambiar el tamaño de la fuente por medio del zoom de texto, con lo que resolverán el problema. Pero no todos los usuarios conocen la función de zoom de texto, por lo que para algunos el tamaño inicial puede parecerles molesto.

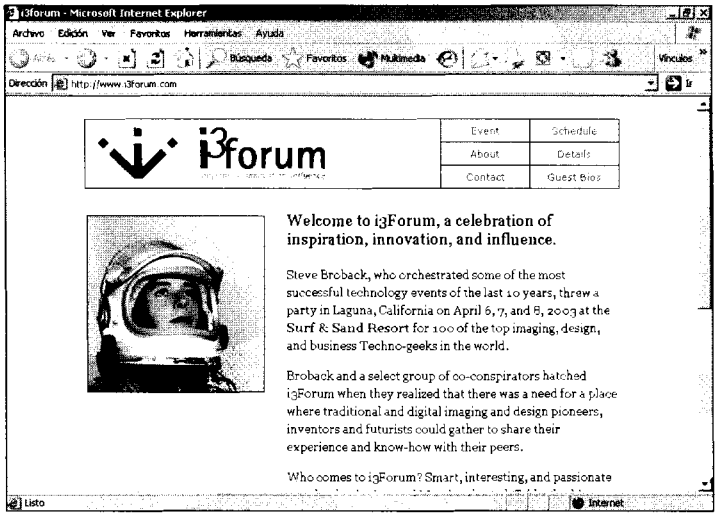


Figura 13.7.

Cuando cambiamos a Windows XP e IE6, el texto permanece con el mismo tamaño.

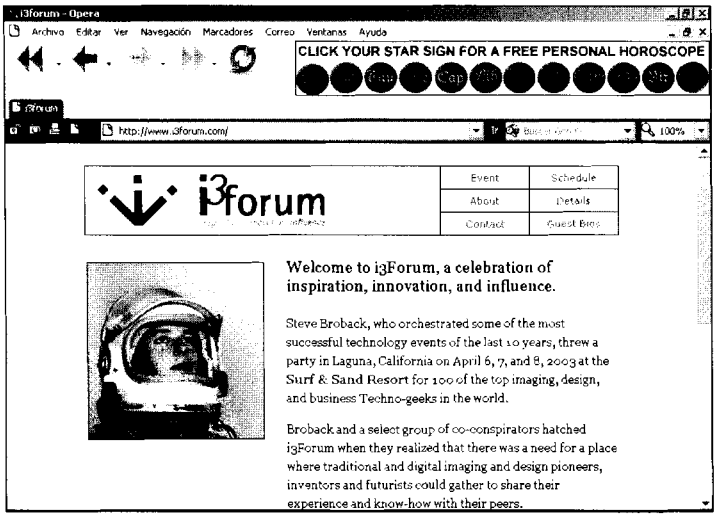


Figura 13.8.

Del mismo modo, cuando el sitio se ve en Opera 7, el tamaño del texto se mantiene. La ventaja del parámetro de tamaño de fuente predeterminado utilizado por todos estos navegadores debería resultar evidente. El texto es legible y se puede cambiar de tamaño en todos ellos.

Sólo nos queda esperar que los brillantes ingenieros concienciados por los estándares que se encargan de Camino y Safari, sepan apreciar las ventajas del valor predeterminado de 16 píxeles y 96 ppp, y que lo hagan en una actualización. (Cuando lea este libro, puede que Safari ya haya cambiado al valor predeterminado.) Al usar tamaños relativos basados en eme, hacemos lo que los defensores de la accesibilidad han aconsejado durante tanto tiempo, pero no funcionará correctamente a menos que todos los navegadores admitan el mismo tamaño de fuente predeterminado. Tampoco funcionará correctamente en otras circunstancias, como veremos en breve.

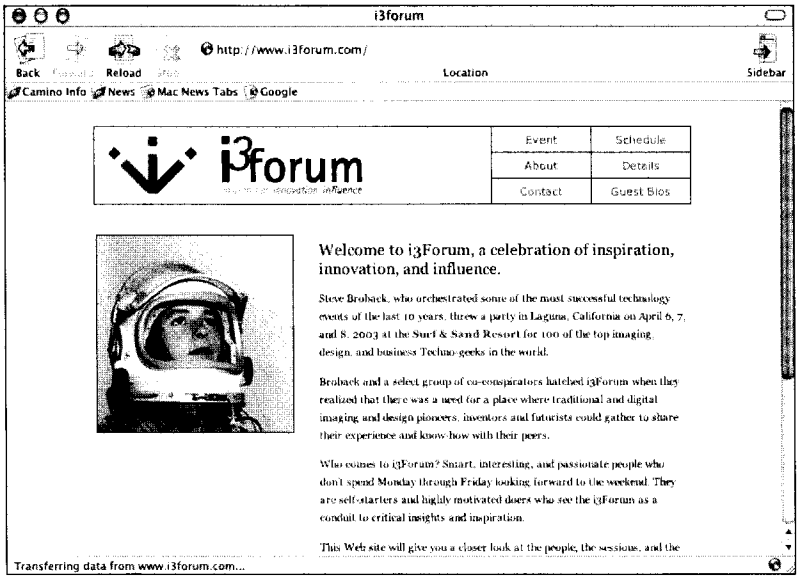


Figura 13.9.

Desafortunadamente, en el magnífico navegador Camino (Navigator), el texto parece más pequeño y resulta difícil de leer. Los usuarios pueden cambiar el tamaño del texto, pero no deberían tener que hacerlo y puede que algunos ni siquiera sepan cómo hacerlo.

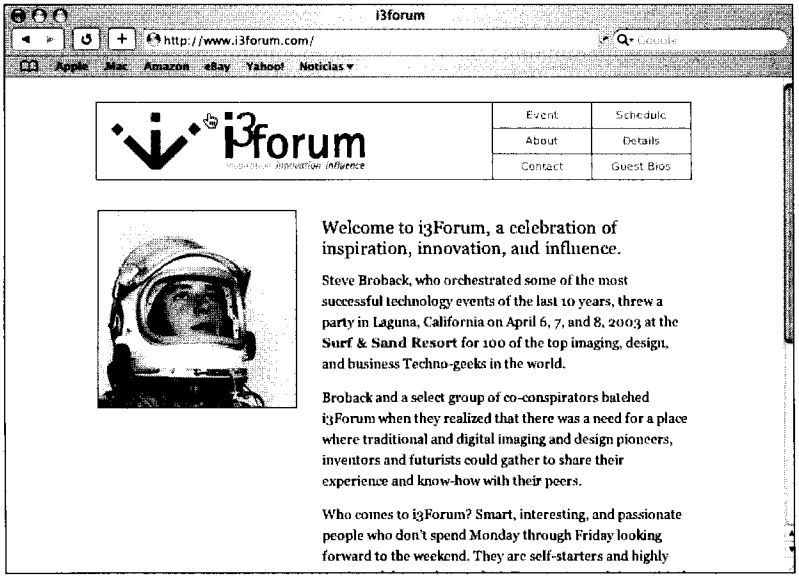


Figura 13.10.

Al igual que Camino, la versión beta del navegador Safari de Apple reproduce el texto a un tamaño inferior al predeterminado. Sólo nos queda esperar que los ingenieros de estos dos navegadores consideren las ventajas del valor predeterminado de 16 píxeles y 96 ppp.

La angustia de las emes

Los defensores de la accesibilidad y los creadores de CSS siempre han estado de acuerdo en que la solución eran las emes. Desafortunadamente, a menudo son la solución para el fracaso. Después de escuchar todas las conferencias y de leer todos los libros y artículos, se sentirá avergonzado si utiliza cualquier otra cosa que no sea eme para especificar el tamaño de sus fuentes. Pero la magnífica teoría de las emes se diluye en la práctica y no sólo en los navegadores que no admiten el tamaño de fuente predeterminado.

Por otro lado, está el problema de los navegadores antiguos. Netscape 4 ignora las unidades eme y ex que se aplican al texto, aunque las respeta extrañamente cuando se utilizan para altura de línea. IE3 trata a las unidades eme como píxeles. De esta forma, 2 eme se traduce incorrectamente por 2 píxeles. Aunque prácticamente nadie utiliza IE3, ahí está el problema.

Del mismo modo, los navegadores antiguos desperdician la herencia en elementos anidados cuyo tamaño se expresa en unidades eme. Como cada vez es menor el número de usuarios de Netscape 4, no perderemos el tiempo en detallar cómo dicho navegador procesa incorrectamente el tamaño relativo de elementos anidados. Basta con que sepa que si admite navegadores antiguos y utiliza unidades eme (sobre todo en elementos anidados), se adentra (y hace lo mismo con sus usuarios) en un doloroso universo (véase figura 13.11).

Opciones del usuario y unidades eme

Un problema más habitual con las unidades eme es que los usuarios suelen reducir el tamaño de fuente predeterminado, como habrá comprobado en este capítulo. Los usuarios de Mac recuperan el valor de 12 píxeles y 72 ppp. Los de Windows utilizan la opción Pequeña en lugar de Mediana en los menús Ver>Tamaño de texto. Estos cambios hacen que cualquier texto con un tamaño menor de 1 eme resulte más pequeño de lo esperado y demasiado reducido para leerlo. En el 2002, Owen Briggs, experto en CSS/DHTML, probó todos los métodos de cambio de texto disponibles en una amplia variedad de navegadores y plataformas para descubrir cuál fallaba y cuál funcionaba correctamente. 264 capturas de pantalla después, a pesar de que su intención era demostrar que las unidades eme siempre eran viables, descubrió exactamente lo contrario (véase figura 13.11).

Text as 0.8em.	Text as 0.8em.	Text as 0.8em.
Text as 0.8em with base 100%	Text as 0.8em with base 100%	Text as 0.8em with base 100%
Text as keyword small.	Text as keyword small.	Text as keyword small.
Text as keyword small, ALA style.	Text as keyword small, ALA style.	Text as keyword small, ALA style.

Figura 13.11.

¿Qué hay en una eme? Todo menos coherencia de tamaño entre navegadores y plataformas (http://www.thenoodleincident.com/tutorials/box_lesson/font/).

Las unidades eme funcionan correctamente siempre que no se especifique el texto por debajo del tamaño predeterminado del usuario. Funcionan correctamente siempre que los usuarios no modifiquen sus preferencias. Pero la mayoría de los diseñadores y muchos clientes están a favor de fuentes más pequeñas y muchos diseños las exigen. Muchos usuarios consideran que el tamaño predeterminado de 16 píxeles es incómodo para la lectura normal y por ello cambian la configuración de sus preferencias. Cuando se utilizan unidades eme para diseñar sitios, los esfuerzos de reducción del diseñador se unen a los del usuario, lo que genera un texto difícil de leer o completamente ilegible.

Cuando definimos fuentes pequeñas con unidades eme (o porcentajes), chocamos con un universo de parámetros de preferencia del usuario imposible de controlar y de conocer con antelación. Lo que parece elegante en nuestro monitor no lo es en el del usuario. Si comete este acto en nombre de la accesibilidad, se está engañando.

En el sitio i3Forum, hemos tratado de minimizar los posibles daños mediante la utilización de tamaños ligeramente inferiores a 1 eme. Pero puede variar en función del usuario.

Por otra parte (http://www.alistapart.com/stories/dao/), si el cliente y la estética lo permiten, puede diseñar todos sus sitios con fuentes normales o de mayor tamaño definidas con unidades eme. De esta forma evitará problemas de accesibilidad causados por el tamaño. Pero pocos diseños funcionan con un tamaño predeterminado de 16 píxeles y superior, y puede que algunos usuarios se quejen del desagradable aspecto del sitio debido al excesivo tamaño del texto. Si la moraleja es que no se puede contentar a todo el mundo, la moraleja adicional es que es menos probable que pueda contentar a todo el mundo si utiliza unidades eme para especificar el tamaño de su texto.

Algunos evangelistas de los estándares y algunos defensores de la accesibilidad optarán por no creer en lo que hemos dicho, al igual que mucha gente optó por creer que la llegada del hombre a la Luna en 1969 fue un engaño. No sé si fue T.S. Elliot o Woody Allen quien dijo que la gente no quiere demasiada realidad. Fuese quien fuese, tenía razón.

Se preguntará entonces qué es lo que la gente quiere. Puede que quieran los dos métodos que describiremos más adelante, que parecen funcionar mejor que los que hemos tratado hasta el momento, aunque tengan sus propios problemas.

Los píxeles demuestran que los píxeles funcionan

Entre todos los valores CSS, con algunas excepciones que analizaremos a continuación, las humildes unidades de píxel (px) funcionan tanto en navegadores antiguos como en nuevos, en navegadores compatibles e incompatibles, y en cualquier plataforma que se le ocurra. 13 px son 13 px se vean en Windows, Mac OS o Linux/UNIX. Si configura su texto en píxeles, tendrá el mismo aspecto en Gecko/Mac OS X (véase figura 13.12), IE6/Windows (véase figura 13.13) y Opera 7/Windows (véase figura 13.14).

Figura 13.12.

Los comienzos del cambio de diseño de zeldman.com en febrero del 2000 (http://www.zeldman.com). El texto se ha definido en píxeles y, por lo tanto, tiene el mismo tamaño en Camino para Mac OS X (en la imagen)...

Figura 13.13.

... como en IE6 para Windows XP. No sólo eso...

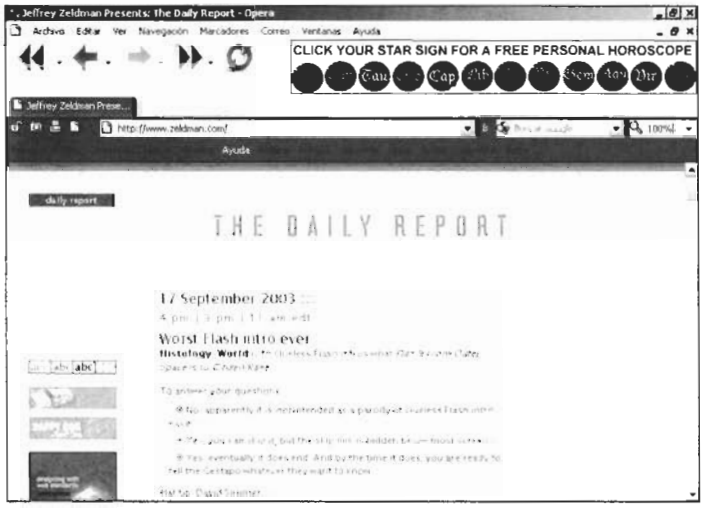


Figura 13.14.

... incluso tiene el mismo aspecto en Opera 7/Windows, a pesar de las quejas de algunos teóricos de que Opera siempre abstrae valores de píxel.

Las ventajas no se limitan a las fuentes. Las imágenes están creadas en píxeles. Si la fotografía de nuestro producto es de 200 px por 200 px y utilizamos CSS para especificar un margen izquierdo de 100 px, sabemos que la relación de tamaño entre la imagen y el margen será 2:1, y también sabemos que todos los usuarios verán dicha relación de espacio. En Happy Cog, denominamos pixelismo a la posibilidad de crear relaciones basadas en tamaño. Dichas relaciones forman parte no sólo de diseños basados en cuadrículas sino también de diseños basados en reglas, que analizaremos en un capítulo posterior.

En diseños atractivos, los píxeles son naturales. Si decimos que el texto de nuestro título tiene 25 píxeles de altura, que nuestro margen izquierdo es de 100 px y que el cuerpo de texto es de 11 px, todos los usuarios verán lo mismo, aunque el tamaño real de lo que vean dependerá de la resolución de su PC y del tamaño de su monitor.

La unidad más pequeña: absolutamente relativo

A una resolución de 1200 x 870, un solo píxel parece mayúsculo en un monitor de 22 pulgadas y minúsculo en uno de 17. CSS considera que un píxel es una unidad relativa ya que los monitores y las resoluciones varían. Es cierto. También es cierto que 10 píxeles son 10 píxeles, desde una esquina de la pantalla a la otra, y de una pantalla a otra. El píxel siempre es la unidad más pequeña de una determinada pantalla, lo que le convierte en el átomo del diseño de nuevos medios. La mayoría de los diseñadores y la mayoría de los usuarios piensan en los píxeles como en unidades absolutas. Al contrario de lo que sucede con la mayoría de unidades, el píxel resulta familiar para todos los usuarios de la Web independientemente de su nivel de sofisticación. Lo conocen porque han pasado años viendo la televisión.

Si el estándar CSS ve los píxeles abstractamente (recuerde el asunto de la longitud media del brazo del usuario Web), los fabricantes de navegadores no lo hacen. Entienden que el píxel es lo que todos pensamos: el punto más pequeño de la pantalla.

Por esta razón, prácticamente todos los navegadores, desde los mejores hasta los pasables, admiten los píxeles de la misma forma, y si utiliza píxeles para especificar el tamaño de sus fuentes, podrá ver lo que espera en cualquier navegador y en cualquier plataforma (como también harán sus usuarios). Como ocurre con cualquier regla, hay excepciones: una sofisticada y vagamente preocupante, y otra más simple e insignificante.

Cuando los píxeles fallan en Opera

Algunas versiones del navegador Opera representan constantemente tamaños más pequeños de los especificados. Por ejemplo, si especifica un texto de 11 px, Opera 5 para Macintosh lo mostrará aproximadamente a 10 píxeles.

Håkon Lie es el técnico jefe de Opera. También es el padre de CSS, a quien muchos diseñadores con cierta sensibilidad están agradecidos. Se dice que es el científico que sostiene que CSS debería definir los píxeles

de forma abstracta (lo de la longitud del brazo) y no como los vemos el resto de los mortales (el punto más pequeño de la pantalla). Puede que tengamos que estarle menos agradecidos por eso, y aliviados por que la mayoría de los fabricantes de navegadores haya optado por entender los píxeles como hacemos todos nosotros.

Algunos sostienen que la forma de Opera de procesar a menor tamaño los valores de píxeles se basa en la abstracción de la longitud de brazo media, en vez de decir que se trata de un error de la versión para Macintosh que parece que Opera no quiere corregir. Puede que estos intelectuales tengan razón pero hemos comprobado que Opera 7 para Windows procesa los píxeles de la misma forma que el resto de navegadores (véase figura 13.14).

Considerando que en la actualidad Opera 7 admite la conmutación DOCTYPE, nuestros lectores intelectuales se preguntarán si Opera 7 trata a los píxeles de la misma forma que otros navegadores sólo en modo de compatibilidad inversa, y los trata de forma abstracta en modo estándares. Estos lectores sobrevaloran el problema. En la figura 13.14 se demuestra que Opera 7 trata a los píxeles de la misma forma que el resto de nosotros los comprendemos; la página en cuestión es XHTML 1.0 Transitional válido con diseño CSS. Por esta razón, nos inclinamos a creer que Opera ha tomado la determinación de representar los píxeles como lo hacen otros navegadores, independientemente de las objeciones teóricas.

En definitiva, cuando utilizamos píxeles para especificar el tamaño de las fuentes, la mayoría de los usuarios de Opera verán lo que queremos que vean, aunque los que utilicen versiones anteriores lo vean con un tamaño ligeramente inferior.

Cuando los píxeles fallan en Netscape 4.x o en otros navegadores

Existe otra excepción a la fiabilidad infalible de los píxeles. En algunas versiones de Netscape 4, los valores de píxel se malinterpretan. En una o dos actualizaciones incrementales de una o dos plataformas, el viejo caballo de guerra olvida cómo debe interpretar las unidades de píxel, como cuando nuestro abuelo se olvida de nuestros nombres. No nos pregunte en qué versiones sucede. ¿Será 4.73 para Linux o 4.79 para Windows 98? Ni lo sabemos ni nos importa. La mayoría de las versiones de Netscape 4 interpretan correctamente los píxeles, aunque malinterpreten todo lo demás.

Este extraño comportamiento de Netscape no se basa en preocupaciones teóricas abstractas. Simplemente se trata de un error, que los usuarios pueden evitar si actualizan a la siguiente versión de esta reliquia o, preferiblemente, si lo hacen a un navegador de Netscape que se haya comercializado este siglo. Algunos usuarios no lo harán. Y puede que uno de ellos sea su jefe o su cliente.

Si aunque les golpee en la cabeza con el objeto pesado más cercano no consigue iluminarlos con las ventajas de un navegador compatible con estándares, empiece a buscar otro trabajo. (De todas formas, después de golpear a su jefe en la cabeza tendrá que buscar otro trabajo. ¿En qué está pensando?)

El problema con los píxeles

El problema relacionado con los píxeles ya se ha analizado en este capítulo y se reduce a lo siguiente. En IE/Windows, los usuarios no pueden cambiar el tamaño del texto defi-nido en píxeles. Si ha usado una fuente de 9 píxeles en el texto de su sitio, muchos visitantes pulsarán el botón **Atrás** de sus navegadores. Incluso una fuente de 11 píxeles puede resultar demasiado pequeña para al-gunos, en función del tipo de fuente seleccionado (por ejemplo, Verdana de 11 px tiene menos quejas que Times de 11 px), el tamaño y la resolución del monitor, de la vista del usuario, del grado de contraste entre fondo y frente y de la presencia o ausencia de fondos que puedan distraer la atención del usuario. Para una persona con una visión inferior a 20/20, el problema puede resultar muy molesto. Para los que tengan defectos de visión más graves, puede ser mucho peor.

También puede que haya un ingeniero de CAD que navegue por la red con su monitor de 32 pulgadas, 4000 por 3000 y que inunde listas de correo de diseño Web con enconadas quejas sobre texto de tamaño ridículo. Si realmente quisiera solucionar su problema, utilizaría un navegador que admitiera funciones de zoom de texto o de página. Si optara por utilizar IE/Windows, podría activar esta opción para ignorar los tamaños de fuentes (que describiremos más adelante). Incluso podría escribir una hoja de estilo como la siguiente:

```
html, body {font-size: 1em !important;}
```

Pero seguramente prefiera quejarse de este problema que solucionarlo de alguna de estas formas.

Como diseñador, es responsable de los problemas que sufran sus usuarios, incluso de los problemas extraños como los de aquéllos que ven la Web a una resolución anormalmente exagerada. El problema con los píxeles es que, como sucede con cualquier otro método de cambio de tamaño de fuentes, inevitablemente provocan el descontento de algún usuario.

Sin tamaño para algunos

Al cierre de la edición de este libro, el zoom de texto ya tiene tres años. La opción de zoom de texto resuelve el problema de los píxeles. El equipo de ingeniería de navegadores Mac de Microsoft inventó esta opción. Pensará que esta opción debería formar parte de IE/Windows. Se equivoca. Hemos luchado por ello. Lo hemos suplicado. Y estamos convencidos de que el mundo se consumirá en llamas de destrucción apocalíptica antes de que Microsoft incorpore el zoom de texto y otras opciones de usuario avanzadas de IE5/Macintosh a su navegador bandera de Windows.

Lo que ofrece IE para Windows (véanse figuras 13.15 y 13.16) es una opción para ignorar los tamaños de fuentes especificados en páginas Web. Oculta en las profundidades de Opciones>General>Accesibilidad, esta desconocida opción permite a los usuarios con defectos de visión (o a los que usen monitores con resoluciones superelevadas) ver todo el texto Web a un tamaño apropiado para ellos. Por lo menos permite a los usuarios con defectos visuales que pueden encontrar esta opción, ver todo el texto Web a un tamaño apropiado para ellos.

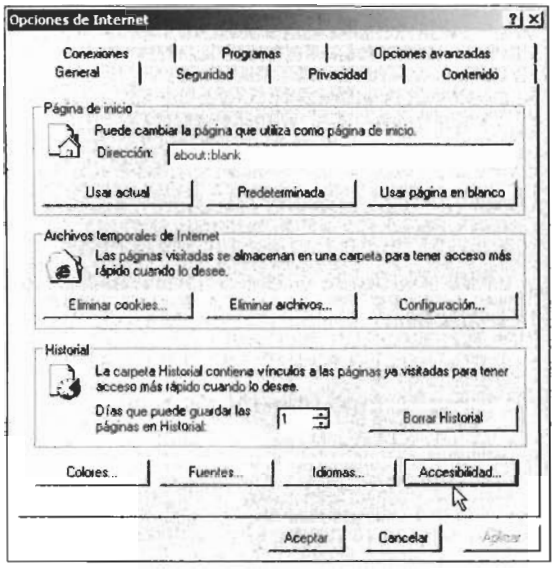


Figura 13.15.

Oculta en IE/Windows se encuentra una opción para omitir los tamaños de fuentes especificados en páginas Web. El usuario debe acceder primero a Opciones de Internet, a la ficha General y, tras ello, hacer clic en Accesibilidad.

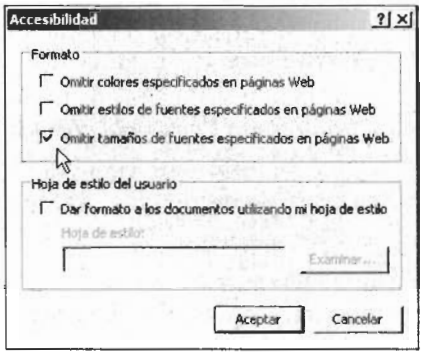


Figura 13.16.

Seguidamente, el usuario debe marcar la opción Omitir tamaños de fuentes especificados en páginas Web, pulsar Aceptar y salir del cuadro de diálogo.

La opción es encomiable. Pero este enfoque de todo o nada no es lo mismo que la opción de zoom de texto (en IE5+/Macintosh, Netscape, Mozilla, Camino, Kmeleon y Safari desde la versión Millennium) o de zoom de página (en Opera desde el principio).

Las relaciones de tamaño de texto transmiten el significado que se pierde cuando los usuarios no pueden ver los tamaños de fuente. La pérdida es aceptable si el diseñador se ha tomado la molestia de escribir marcado estructural. Pero, en la actualidad, el marcado no estructural domina la producción Web, como todos sabemos. Con la divitis y la clasitis que ya hemos explicado, las relaciones de texto se pierden cuando los tamaños se desactivan mediante este método de fuerza bruta. El zoom de texto y el zoom de página conservan las relaciones de texto que permiten a los lectores obtener el significado del contenido, aunque éste se haya creado incorrectamente. IE/Windows es un navegador correcto y MSIE6 lo es en especial. Pero el hecho de ocultar el tamaño de fuentes no es una solución para permitir a los usuarios aumentar o reducir el tamaño de texto en función de sus necesidades.

Considerando el actual dominio de IE/Windows y su resistencia a que los usuarios escalen el texto definido en píxeles, seguramente se vea tentado a abandonar la posibilidad de utilizar los fiables píxeles entre plataformas. Pero puede que no tenga que hacerlo. En un capítulo anterior describimos una técnica que permite a los diseñadores utilizar píxeles sin castigar a los usuarios de Windows con problemas de visión.

Mientras tanto, ofrecemos otro posible enfoque que permite el cambio de tamaño en todos los navegadores (incluido IE para Windows) y que evita la mayoría, aunque no todos, los problemas asociados a emes y porcentajes.

El método de las palabras clave de tamaños de fuentes

Desconocidas y poco utilizadas, CSS1 (y, posteriormente, CSS2) ofrecen siete palabras clave de tamaños de fuentes que permiten controlar los tamaños de texto sin el absolutismo de los píxeles ni los problemas de herencia, de plataformas y de opciones del usuario relacionados con emes y porcentajes. A continuación le mostramos las siete palabras clave. Su significado es obvio para cualquiera que alguna vez haya comprado una camiseta (http://www.w3.org/TR/CSS2/fonts.html#value-def-absolute-size):

```
xx-small
x-small
small
medium
large
x-large
xx-large
```

Por qué las palabras clave superan a emes y porcentajes

Como hemos mencionado en un apartado anterior, al utilizar emes o porcentajes siempre se corre el peligro de que sus valores se multipliquen, lo que genera texto demasiado grande o demasiado pequeño. Por el contrario, los valores de palabras clave no cambian aunque los elementos se aniden. Si `<body>` es `small`, `<div>` es `small` y `<p>` es `small`, y `p` se encuentra dentro de `div`, que se incluye en `body`, los tres valores `small` no se combinan (como sucede con emes y porcentajes) y el resultado sigue siendo legible. Es más, el resultado sigue siendo `small` (no `x-small` o `xx-small`). Las emes y los porcentajes se combinan. Los valores de palabras clave no.

Además, al menos en navegadores Gecko y versiones modernas de IE, `xx-small` nunca puede ser inferior a 9 px, lo que significa que nunca será ilegible. Evidentemente, el texto puede ser difícil de leer para algunos usuarios, pero no es lo mismo que ser ilegible.

Al igual que las emes, las palabras clave se basan en el tamaño de fuente predeterminado del usuario. Al contrario que las emes, las palabras clave nunca descienden por debajo del umbral de resolución indicado. Si el tamaño predeterminado del usuario es de 10 píxeles (improbable, pero posible), `x-small` sería 9 px y `xx-small` también. Evidentemente, en este caso, se perdería la diferencia de tamaño entre `x-small` y `xx-small`. Pero no perderíamos a nuestros lectores.

Parece perfecto, ¿verdad? Podemos especificar tamaños sin enfrentarnos a la incapacidad de IE/Windows para cambiar el tamaño de los píxeles y sin inflingir diminutas fuentes ilegibles en ningún visitante. Las palabras clave de tamaños de fuentes parecen equilibrar la accesibilidad con la necesidad de control del usuario. ¿Qué podría ir mal? Una pista: los navegadores.

Problemas iniciales con las implementaciones de palabras clave

Las siete palabras clave de tamaños de fuentes se podrían haber implementado correcta y uniformemente, pero no fue así. Desafortunadamente, las palabras clave se implementaron inepta o incorrectamente en los primeros navegadores que trataron de admitirlas aunque, en un caso, se implementaron correctamente con resultados tan negativos que provocaron el cambio de una versión posterior del estándar CSS.

Mientras que Netscape 4 ignoraba las palabras clave CSS, Netscape 4.5 y superiores, e Internet Explorer 3 las representaban a tamaños ilegibles. Por ejemplo, Netscape 4.5 e IE3 representaban `xx-small` a seis píxeles, tres píxeles por debajo del umbral de legibilidad.

En el caso de Netscape, los ingenieros se guiaron por una recomendación del W3C, que indicaba que todos los tamaños debían ser 1'5 veces mayores que el tamaño anterior. Si `small` equivalía a 10 píxeles. `medium`, un tamaño mayor, sería 15 píxeles. `x-small` tendría que equivaler a 6'6667 píxeles (10/1'5) y, en cuanto a `xx-small`, mejor no pregunte. El W3C cambió su recomendación por el valor de escala 1'2 después de que se demostrara que el valor 1'5 no funcionaba correctamente. Pero si una cosa no funcionaba correctamente era la representación precisa de la especificación en Netscape 4.

Netscape 4.5 hizo lo correcto y siguió las recomendaciones del W3C a la letra. Por ello, fue ejemplarmente castigado. Fue una de las pocas ocasiones en que Netscape 4.x utilizó correctamente un estándar. (Recuerdo otras dos ocasiones, con consecuencias igual de desafortunadas. Netscape se negó a que los nombres de clases e ID incluyeran guiones bajos, y también se negó a que los nombres de clases e ID empezaran por números. Era un comportamiento correcto en CSS1 y, de nuevo, se puso a Netscape como un trapo.) Netscape recibió lo suyo cuando se equivocó con los estándares y volvió a recibir cuando los aplicó correctamente. Seguro que se compadece de los pobres ingenieros de Netscape 4.

El error de palabras clave en IE/Windows

Mientras tanto, IE4/Windows, IE5/Windows e incluso IE5.5/Windows aplicaban incorrectamente las palabras clave, de alguno modo u otro. En estos navegadores, existe una desconexión lógica entre la palabra clave y la forma en que se representa. `Small` tiene un tamaño medio y `medium` es mayor de lo normal, y así sucesivamente. Se preguntará cómo pudo ocurrir algo tan extraño.

Los ingenieros que desarrollaron IE para Windows trataban de hacer lo correcto. Seguramente recuerde las siete etiquetas `<font size>` de Netscape que mencionamos anteriormente: `<font size=1>`, `<font size=4>`, etc. Los programadores de IE asignaron directamente las siete palabras clave CSS a los siete tamaños de fuente de Netscape. En muchos aspectos, era lo lógico. Hay que reconocer el mérito del equipo de programación por intentar admitir las palabras clave de forma que tuviera

sentido para los diseñadores que anteriormente habían sufrido los tamaños de fuentes en Nestcape.

El problema, evidentemente, es que los tamaños no se asignan a las palabras clave. En los navegadores antiguos, `<font size=3>` es el tamaño predeterminado o `medium` que el usuario especifica en sus preferencias. En el marcado HTML extendido de Netscape, se asume `<font size=3>` a menos que se especifique otro tamaño. Lógicamente, un tamaño predeterminado debería asignarse a la palabra clave CSS medium. Desafortunadamente, en el esquema original de IE/Win-dows, `<font size=3>` se asigna a `small` en lugar de a `medium`, porque `small` es el tercer tamaño si contamos desde la parte inferior de la lista.

El progreso se evidencia, pero el uso lo elude

Con el tiempo, IE5/Macintosh, Netscape 6+, Mozilla e IE6 aplicaron correctamente las palabras clave. Pero cuando esto sucedió, millones de usuarios utilizaban IE5 y Netscape 4, navegadores que aplicaban dichas palabras clave de forma incorrecta. Si los diseñadores utilizaban palabras clave CSS como debían, la representación se perdía en IE4-5/Windows (y también en Netscape 4, aunque de forma menos importante). Si, de forma deliberada, los diseñadores utilizan incorrectamente las palabras claves para que la representación fuera adecuada en IE4-5/Windows, ésta se perdía en todos los navegadores y en las últimas versiones de IE/Windows. ¿Qué podían hacer los diseñadores? Lo que hicimos la mayoría de nosotros fue considerar que las palabras clave de tamaños de fuentes de CSS eran simplemente otra opción que no funcionaba.

Las palabras clave crecen: el método Fahrner

Otra vez más fue Todd Fahrner (que debería cobrar derechos de todos los diseñadores Web del planeta) quien ideó una solución. Y no fue otra que el problema del modelo de cuadro que describimos en un capítulo anterior, creado por Tantek Çelik (cuya estatua debería presidir nuestros parques algún día). Puede leer la evolución y los detalles del método Fahrner en el artículo "Size Matters" de A List Apart (http://www.alistapart.com/stories/sizematters/) y también puede comprobar la variación que hace Happy Cog del mismo en http://www.happycog.com/thinking/colophon.html.

El método funciona de esta forma:

- Se ocultan estilos a los navegadores 4.0 por medio de la directiva `@import`. Como estos navegadores no entienden dicha directiva, ignoran la hoja de estilo importada que contiene las reglas CSS, que de todas formas únicamente sabrían desperdiciar. De esta forma se permite que los usuarios de navegadores 4.0 vean el texto en el tamaño que elijan como predeterminado o, en caso de que sea necesario, se les proporcionan valores de píxeles en una hoja de estilo vinculada.

- Se utiliza el problema del modelo de cuadro para proporcionar valores de palabra clave de tamaño de fuente falsos a IE5/Windows y valores correctos a los

navegadores más compatibles, al igual que hicimos para proporcionar valores falsos de márgenes, relleno y contenido a los diferentes navegadores en un capítulo anterior.

- Se añade una regla CSS más específica para proteger a Opera de uno de estos errores (como describimos en un capítulo anterior).

El método en acción

A continuación incluimos un sencillo ejemplo. Queremos que el texto de párrafo tenga un tamaño `small` (un paso por debajo de `medium`). Los navegadores compatibles nos proporcionarán dicho valor si lo solicitamos. Pero es necesario engañar a IE5/Windows. Si pedimos a IE que nos proporcione `x-small`, nos dará lo que la mayoría de los navegadores considera que es `small`. Veamos la regla:

```
p    {
font-size:    x-small;
        /* valor falso para WinIE4/5 */
    voice-family: "\"}\"";
        /* engañe a WinIE4/5 para que
piense que la regla se ha terminado */
    voice-family: inherit;
        /* recupérese del truco */
    font-size:    small;
        /* valor para los navegadores más
compatibles */
    }
```

Como hicimos en el capítulo anterior, añadimos una regla para ser amable con Opera. Como el selector de esta regla es más específico que el de la regla anterior, Opera le prestará atención y se evita que utilice el valor falso destinado únicamente para IE4-5/Windows:

```
html>p    {
    font-size:    small;
        /* sea amable con Opera */
    }
```

Siempre que lo hagamos en una hoja de estilo importada, Netscape 4 no lo verá y no se confundirá. Si tiene que aplicar el tamaño a varios selectores, resulta muy sencillo. Para ver cómo hacerlo, puede consultar la hoja de estilo en http:// www.happycog.com/c/sophisto.css y adaptarla a sus propias necesidades.

Objeciones

El método anterior funciona bastante bien pero las palabras clave de tamaños de fuentes padecen el mismo problema que afecta a las emes y que ya hemos analizado previamente. Cualquier valor inferior al tamaño de fuente predeterminado del usuario puede ser demasiado pequeño para leerlo cómodamente. Suele suceder en los mismos tres jinetes del Apocalipsis tipográfico:

- Los usuarios de Windows que configuran su navegador con la opción Pequeña en lugar de Mediana (Ver>Tamaño de texto).

- Los usuarios de Mac que cambian al cómodo valor de 12 píxeles y 72 ppp (o cualquier valor por debajo de 16 píxeles).

- Los usuarios de nuevos navegadores como Camino y Safari que no admiten el tamaño predeterminado de 16 píxeles y 96 ppp, que tan útil resulta entre las distintas plataformas.

Cuando se aplica la directiva `@import` y el método de Tantek a las palabras clave de

tamaños de fuente, los usuarios evitan el daño que pueden inflingir las emes y se aseguran de que ninguna fuente tendrá un tamaño inferior a 9 píxeles. Pero si el usuario ha configurado un tamaño de fuente diminuto, `medium` será diminuto, al igual que todos los valores que le siguen.

En busca de la fuente utilizable

Trece años de madurez de la Web y no existe forma alguna de definir fuentes que, fiable y coherentemente, ofrezcan el control que requieren los diseñadores y la libertad que necesitan los usuarios para cambiar los valores de tamaño iniciales de cualquier diseño. Podría existir una forma si IE/Windows permitiera a los usuarios cambiar el tamaño del texto en píxeles, como han hecho IE5/Macintosh, Mozilla, Netscape, Navigator y Kmeleon en los últimos tres años y Opera siempre ha hecho. Pero a Moby (el técnico, no la ballena) le saldrá pelo antes de que Microsoft incluya la tecnología que ha inventado en su cliente de Windows.

No se trata de un problema de los estándares. El zoom de texto no se corresponde a los parámetros de CSS o de ningún otro estándar Web. No existe ninguna especificación ni ninguna ley que diga que los usuarios deben poder cambiar el texto de las páginas Web. Lo dice el sentido común, pero no es un estándar.

No piense que se trata de un problema exclusivo para los diseñadores que se basan en los estándares. Aquéllos que evitan pensar en los estándares Web lo pasan peor y consiguen resultados mucho menos predecibles.

Tampoco tienen nada que envidiar los que diseñan exclusivamente en Flash. También padecen los problemas de resolución del monitor y se enfrentan a retos de accesibilidad más graves que el resto de nosotros. Es más, incluso la posibilidad que tiene el usuario de aumentar el contenido de Flash por medio de menús contextuales no garantiza la legibilidad, en especial cuando el contenido de Flash se confina en marcos cuyos bordes pueden ocultar zonas a la vista.

Se preguntará cuál es la solución. En ocasiones tendrá que usar píxeles y proporcionar algún artilugio de tamaño de texto dirigido por el DOM para solucionar las limitaciones de IE/Windows. En otros casos, tendrá que aplicar el método Fahrner para crear palabras clave de tamaños de fuentes accesibles, que se puedan cambiar de tamaño en cualquier navegador. Usted paga y usted decide. Esto ha sido todo en lo que respecta al capítulo. Como dice Clint Eastwood, "¿Se siente afortunado?"

Capítulo 14
Conceptos básicos de accesibilidad

La accesibilidad y los estándares tienen mucho en común. Ambos tratan de garantizar que nuestro trabajo resulte útil para el mayor número posible de lectores, visitantes y clientes. La accesibilidad está tan unida al resto de estándares que hemos descrito en este libro que, en los años 90, el W3C publicó una Iniciativa de accesibilidad Web (WAI) para aconsejar a los creadores Web acerca de las estrategias necesarias para conseguirla (http://www.w3.org/WAI/GL/).

La WAI ofrece tres niveles estandarizados de acceso, desde el rápidamente alcanzable (Prioridad 1), pasando por el que requiere más trabajo (Prioridad 2) para terminar con el nivel principal (Prioridad 3). El objetivo de los tres niveles es la accesibilidad que, al igual que las otras formas de cumplimiento de los estándares que hemos descrito anteriormente, es algo continuo, no una decisión de tipo todo o nada. Aunque no estemos preparados para realizar la conversión a diseños CSS, podemos probar nuestros sitios y cumplir con los estándares si utilizamos los métodos híbridos de diseño que hemos analizado en capítulos anteriores. De esta forma, con un pequeño y razonable esfuerzo, cualquiera de nosotros, incluso los novatos en el tema de la accesibilidad, puede obtener un nivel de Prioridad 1 o similar. Al hacerlo, conseguiremos que nuestros sitios estén disponibles para todos aquéllos para los que antes no lo estaban.

En muchos países existen leyes que proscriben la negación de acceso a los discapacitados, y muchos países han aplicado dichas leyes a los nuevos medios a través de edictos de accesibilidad Web como U.S. Section 508. Algunas de estas leyes nacionales se basan en la Prioridad 1 de la iniciativa WAI, pero

otras picotean aleatoriamente en el bufé del acceso. Las leyes de algunos países resultan confusas para los miembros de la WAI, que han dedicado años al estudio de los problemas de acceso. Algunas leyes resultan vagas y otras realmente prácticas. Muchas leyes han sido ignoradas por los organismos que las han creado. Con razón los diseñadores y los programadores están confusos.

En este capítulo, nos abriremos paso entre la confusión, rechazaremos los mitos que han aturdido a muchos profesionales y le ofreceremos un repaso práctico de accesibilidad aplicada y de sentido común. También analizaremos algunas herramientas que puede utilizar para incorporar accesibilidad a sus diseños y comentaremos las limitaciones de dichas herramientas.

No se podría abarcar el campo de la accesibilidad en un solo capítulo. Afirmar lo contrario sería un insulto no sólo para los expertos en accesibilidad sino también para aquéllos que la necesitan. De hecho, apenas existen libros que ahonden en lo que brevemente describiremos en este capítulo y, de hecho, algunos contribuyen sin saberlo a aumentar la confusión de los diseñadores sobre la accesibilidad y su hostilidad hacia el acceso.

Acceso mediante libros

Muchos libros sobre accesibilidad con buenas intenciones desprenden fuego y azufre. Y el olor del azufre no inspira a los diseñadores. Con demasiada frecuencia, estos libros únicamente contienen desagradables ejemplos, o totalmente irreales, de sitios accesibles, junto con consejos poco prácticos de tipo "nunca especifique tamaños de fuente". Algunos son hostiles al diseño. Otros no tienen experiencia en la programación de sitios comerciales. Después de leer estos libros, los diseñadores pueden pensar que la accesibilidad es irrelevante.

Otros libros están bien documentados y se basan en una profundidad apasionada. Son los que merecen el tiempo del lector. Pero no son recomendables para el profesional Web general ya que están dirigidos a lectores que sufren algún tipo de discapacidad. Este tipo de libros suelen dedicar gran parte de su tiempo a presentar métodos de entrada alternativos y a evaluar los méritos y deméritos de los agentes de usuario alternativos. Es muy probable que los diseñadores no discapacitados se sientan alienados o teman que, de algún modo, también se vean afectados. El miedo a la ceguera, a la parálisis o a otras discapacidades es el responsable, parcialmente, de la incomodidad de algunos diseñadores como el propio concepto de accesibilidad y estos libros no contribuyen a que eludan este perjuicio.

Nuestras recomendaciones son las siguientes:

- ***Building Accessible Web Sites*, Joe Clark:** No sólo el mejor y más completo libro escrito hasta el momento sobre accesibilidad Web. También se encuentra entre los libros de diseño Web más irresistibles jamás escritos: ingenioso, dogmático y dinámico. En nuestra extraña línea de trabajo, tenemos acceso a la mayoría de los nuevos libros de diseño y de informática. Pocos son completos, menos son completamente lúcidos y muchos menos

nos transmiten que estemos ante una gran obra de comunicación. Devoramos el libro de Clark de punta a cabo, como si se tratara de la última novela de James Ellroy o de Raymond Chandler. Y, tras ello, lo volvimos a leer. Con este libro, no sólo aprenderá todo lo que necesita saber sino que también lo leerá por placer.

Abarca todos los aspectos, desde los fundamentos de la creación de atributos `alt` utilizables hasta las complejidades del titulado de medios. Joe Clark, a quien Atlantic Monthly denominó "el rey de los títulos cerrados", ha dedicado 20 años al campo del acceso de medios, y se nota. Disfruta de una posición única para guiar al lector de forma clara y confiada desde lo principal a lo detallado, ofreciendo estrategias de accesibilidad en fases que se ajustan a cualquier limitación presupuestaria o de tiempo, clarifica las regulaciones y desenmascara los mitos. Es más, Clark presta la misma atención a la estética del diseño como al acceso, y demuestra que ambos aspectos son compatibles. Sin duda, un libro que debe leer.

- ***Constructing Accessible Web Sites*, varios autores:** Escrito por diversos expertos en la materia como Jim Thatcher, Shawn Layton Henry, Paul Bohman y Michael Burks, este libro de orientación práctica es una especie de compendio de los mejores artículos de revistas, cada uno sobre un determinado aspecto del universo del acceso, analizado de forma precisa y detallada. Entre los diferentes temas abordados se incluye material fundamental sobre las limitaciones del software de prueba de accesibilidad de botones, analiza posibles métodos de implementación de la accesibilidad en grandes empresas y ofrece detallados consejos sobre la creación accesible en Flash MX.

  Aunque carece de la ventaja del libro de Clark de una única voz, el coro que ofrece *Constructing Accessible Web Sites* es convincente por naturaleza. Ambos libros exigen un hueco en la estantería de cualquiera que diseñe, cree, posea o gestione sitios Web. Entre otras ventajas, la lectura de estos libros le servirá para rechazar muchas ideas equivocadas, y tristemente generalizadas, como las que analizaremos a continuación.

Confusión extendida

Presentados con la noción de la accesibilidad, muchos diseñadores, programadores y propietarios de sitios tienden a perorar incomprensibles aforismos sobre la forma de complacer a sus clientes. Informados con leyes de accesibilidad como la U.S. Section 508 del Workforce Investment Act, a menudo se adentran en lo que sólo podemos denominar incontinencia mental.

El genio metió el pie

En más de una ocasión, dirigiéndose a un público profesional, hemos escuchado cómo un diseñador Web altamente respetado respondía a la pregunta de un espectador sobre accesibilidad con tonterías como la siguiente: "Creamos materiales de altísimo

nivel para la elite de consumidores a la que quiere llegar nuestro cliente. Eso de la accesibilidad, es una pequeña parte de nuestro mercado y, eh, a nuestro cliente no le importa perder algunos adeptos. Quiero decir, que nuestro cliente fabrica pantallas de televisión de alta resolución. Y los ciegos no creo que las vayan a comprar este año (risas)".

De hecho, puede que los invidentes compren dichas pantallas de televisión para algún miembro de su familia que sí pueda ver si el sitio les permite leer las especificaciones y utilizar los formularios de compra en línea. Es más, la gran mayoría de los usuarios Web con dificultades de visión no son completamente invidentes. La mayor parte de los que requieren mejoras de acceso suelen ser usuarios con una visión reducida, daltónicos o simplemente sufren una ligera miopía, y cualquiera de ellos puede querer comprar una gran pantalla de televisión de calidad.

El multimillonario ciego

Por otra parte, los navegadores son, en efecto, usuarios ciegos. En una conferencia a la que asistimos recientemente, uno de los oradores apuntó que el motor de búsqueda Google es el principal usuario ciego de la red y que este usuario realiza recomendaciones en forma de resultados de búsqueda para una tonelada métrica de usuarios cada minuto del día.

Visto de otra forma (perdón por lo de visto), el número total de lectores de Google lo convierte en una especie de multimillonario ciego. ¿Cuántos sitios dirían que no a clientes potenciales dispuestos a gastar miles de millones de dólares? Si lo vemos de una tercera forma, sólo los invidentes norteamericanos constituyen un grupo similar a la población de las áreas metropolitanas de Nueva York y Los Angeles. No sería ningún acierto excluir a toda la población de estas zonas de nuestros sitios.

El acceso no se limita a los usuarios con dificultades de visión

Pero el acceso no se limita a los usuarios con dificultades de visión. Aquéllos con dificultades de movimiento (con parálisis parciales o totales) puede que quieran comprar un televisor por la red en lugar de desplazarse molestamente a una tienda de electrodomésticos. Muchas de las mejoras de acceso se dirigen a estos consumidores y no al grupo de los cupones y el bastón del que suelen acordarse los mal informados cuando consideran el uso de la accesibilidad. Las mejoras de acceso también ayudan a los consumidores no discapacitados que tratan de comprar esa televisión mediante un Palm Pilot o un teléfono compatible con la Web.

En definitiva, el diseñador, programador o propietario de sitio que afirme que los invidentes no compran nuestros productos, no lo acaba de entender. Ellos son los que no ven la verdadera naturaleza del público que rechazan, incluyendo los millones de usuarios no discapacitados que habrían encontrado su sitio por medio de un motor de búsqueda si hubieran realizado un esfuerzo por cumplir las directrices de acceso.

Desafortunadamente, no son los únicos que no comprenden lo que significa el acceso y para quién sirve.

Una nube de ideas confusas

Con los años, hemos escuchado muchas declaraciones equivocadas de boca de profesionales Web supuestamente bien informados, como las que destacamos a continuación:

"Tío, soy diseñador. Hago cosas que tienen un buen aspecto. ¿Cómo puedo hacer cosas con un buen aspecto para un montón de gente invidente?" (La respuesta: Prácticamente todas las mejoras de diseño no afectan en absoluto al diseño visual. Se aplican al marcado, por debajo de la reluciente superficie de nuestro sitio. Tío, lo que haces puede tener un aspecto perfecto y ser completamente accesible (véase figura 14.1).

"Es problema de nuestro cliente. Si no lo incluye en la solicitud de propuesta, no tenemos que preocuparnos", afirmaba un importante programador de una de las principales y más famosas agencias Web, poco después de que se fuera a pique y sus restos los adquiriera una pequeña empresa de Asia Central. Por cierto, se equivocaba en lo de que la accesibilidad era problema del cliente.

"¿La Section 508? Eso es para el gobierno. No somos funcionarios públicos." Escuchado en una agencia de diseño a cuyo cliente se le exigió legalmente la inclusión de acceso.

"Uno de los miembros del comité lo está considerando. Tendremos un borrador preparado en cualquier momento". Un director de proyecto de una institución gubernamental norteamericana compartía esta información interna un año después de que la Section 508 se convirtiera en la ley universal.

"Estamos especializados en Dreamweaver, por lo que todo eso de la accesibilidad se resuelve en la nueva versión, ¿no?", nos dijo un diseñador. Sí y no. Dreamweaver MX cuenta con numerosas mejoras de acceso, pero hay que saber cómo utilizarlas. Sólo con abrir el programa no basta para garantizar que un sitio sea accesible.

Figura 14.1.

Este sitio (http://www.spazowham.com/) cumple con las directrices de accesibilidad de la U.S. Section 508. ¿Lo ve? Claro que no. De eso se trata.

La ley y el diseño

El volumen de confusión era ya elevado cuando la aprobación de la Section 508 de la U.S. Workforce Investment Act dio una vuelta de tuerca más. (En este apartado, escribimos desde una perspectiva norteamericana y utilizamos ejemplos norteamericanos, pero los mismos principios se aplican a cualquier lugar del mundo, independientemente de las leyes locales que lo rijan.)

La Section 508 requiere que muchos sitios acepten usuarios con discapacidades, desde lesiones de movilidad hasta una amplia variedad de lesiones visuales, y también determina lo que significa la accesibilidad. (Truco: no basta con añadir atributos `alt` a las imágenes.) Enfrentados a esta tarea, muchos profesionales Web concluyen que la accesibilidad equivale a páginas de sólo texto o a diseños simples poco atractivos. Tampoco es eso.

Las imágenes, CSS, diseños de tabla, JavaScript y otros elementos del diseño Web contemporáneo son totalmente compatibles con el cumplimiento de la Section 508; simplemente requieren una atención especial. Como comprobará en este capítulo, veremos lo que implica específicamente la accesibilidad y la Section 508, y veremos cómo puede aplicar decisiones inteligentes y utilizar las herramientas disponibles para que sus sitios lo cumplan correctamente.

La Section 508

La Section 508 (véase figura 14.2) forma parte del Rehabilitation Act de 1973, cuyo objetivo era acabar con la discriminación con los discapacitados. Promulgada por el Congreso de los Estados Unidos el 7 de agosto de 1999, la ley 105-220 (Rehabilitation Act Amendments de 1998 [http://www.usworkforce.org/wialaw.txt]), aumentaba significativamente los requisitos tecnológicos de acceso de la 508. La ley se aplicaba a equipos informáticos, faxes, fotocopiadoras, teléfonos y otros dispositivos utilizados para transmitir, recibir o almacenar información. También se aplicaba a muchos sitios Web.

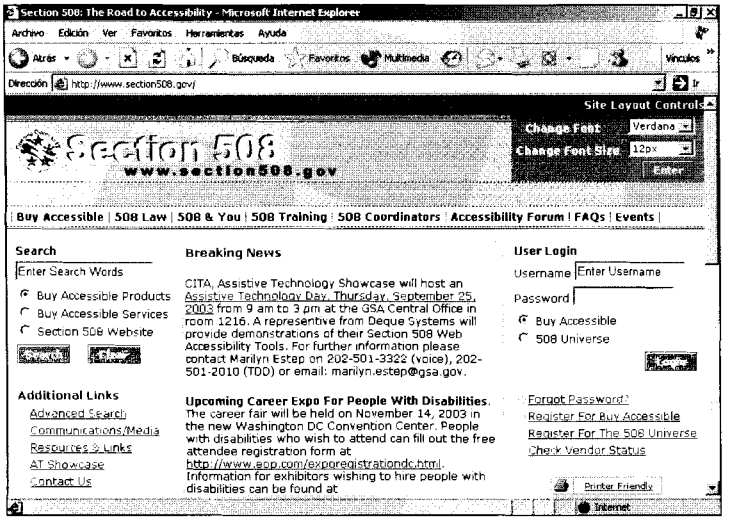


Figura 14.2.

El sitio oficial de la Section 508 cumple con las directrices que promulga (www.section508.gov). No suele ocurrir con los sitios oficiales y no ocurría originalmente con el sitio section508.gov. Fíjese en los conmutadores de tamaño de fuente en la parte superior derecho, influencia de ALA, para compensar los navegadores que no permiten a los usuarios cambiar el tamaño del texto.

La Section 508 se convirtió en ley el 21 de junio del 2001. Afecta directamente a departamentos y agencias federales, así como a los diseñadores Web que trabajan para las mismas. La ley también se aplica a proyectos financiados por el gobierno y a todos los estados que la adopten, como muchos han hecho ya. Los lectores norteamericanos pueden consultar la decisión de su estado en http://www.resna.org/taproject/policy/initiatives/508/508Stateactions.htm.

En resumen, la Section 508 se aplica a lo siguiente:

- Departamentos y agencias federales (incluyendo el Servicio de correos estadounidense).
- Actividades promovidas o financiadas por el gobierno federal.
- Actividades promovidas por los estados que hayan adoptado la regulación.

Acceso igualitario o equivalente para todos

La Section 508 requiere que todos los sitios Web bajo su jurisdicción proporcionen un acceso igualitario o equivalente para todos, incluyendo los discapacitados visuales, los discapacitados auditivos, los discapacitados físicos y los que sufren de epilepsia fotosensible.

Puede que le sorprendan los problemas a los que se enfrentan estos usuarios. Por ejemplo, un texto pequeño que no se pueda cambiar de tamaño puede impedir que los usuarios con problemas de visión puedan leer el contenido de un sitio (consulte el análisis del problema de los píxeles de un capítulo anterior). Los botones de navegación diminutos pueden resultar un estorbo para los visitantes con problemas motrices. Las páginas con parpadeos y destellos pueden provocar serios ataques a los usuarios con riesgo de epilepsia. Y podríamos continuar. La ley explica multitud de problemas de acceso habituales y sugiere, no dicta, posibles soluciones.

La Section 508 no prohíbe el uso de CSS, JavaScript, imágenes o diseños de tablas. Tampoco nos impide que incorporemos recursos como Flash o QuickTime, siempre que cumplamos determinadas directrices que veremos más adelante. Naturalmente, la mayoría de los sitios que cumplen la ley 508 (como los que cumplen los estándares) se vea estupendamente tanto en los nuevos navegadores como en los antiguos. No supone problema alguno bajo la ley ya que los usuarios Web pueden actualizar sus navegadores con tan sólo descargar las nuevas versiones, que suelen ofrecerse de forma gratuita.

El cumplimiento de las directrices de accesibilidad, junto con el cumplimiento de los estándares, no sólo aumenta la disponibilidad de nuestros sitios a millones de usuarios discapacitados, sino que también nos permite llegar a otros muchos, incluyendo los usuarios de PDA, teléfonos compatibles con la Web, navegadores minoritarios y kioscos, y atraer a muchos más por medio de motores de búsqueda.

Más visitantes. Más usuarios. Más lectores. Más miembros. Más clientes. Suena muy bien. Se preguntará entonces por qué los

diseñadores, programadores y propietarios de sitios están confusos o son hostiles a la Section 508 y a regulaciones de accesibilidad similares. Estas hostilidades se deben, principalmente, a los mitos sobre acceso que durante tanto tiempo nos han asolado. Intentaremos clarificar algunas de estas creencias equivocadas.

Mitos de accesibilidad desenmascarados

Los mitos que describimos a continuación son algunos de los que han desconcertado a muchos profesionales Web. Nuestras refutaciones tienen un alcance limitado debido a las restricciones físicas de este capítulo.

Mito: la accesibilidad obliga a crear dos versiones de un sitio

Falso. Si diseñamos con estándares Web y seguimos ciertas directrices, nuestros sitios serán tan accesibles para lectores de pantalla, Lynx, PDA y navegadores antiguos como en los navegadores más modernos y compatibles. Los estándares y la accesibilidad coinciden en que un documento Web debe servir para todos los usuarios y lectores.

Si diseñamos exclusivamente en Macromedia Director/Shockwave entonces sí tendrá que crear una versión diferente para cumplir con la Prioridad 1 de la WAI o las directrices de la Section 508. Si el acceso forma parte de sus activos, diseñe con estándares y guarde Shockwave para proyectos más indicados. Sepa que una de las próximas versiones de Macromedia Director permitirá a los programadores crear archivos Shockwave mucho más accesibles. Falta por ver si estos archivos cumplirán las directrices de acceso de la WAI o no.

Mito: una versión de sólo texto satisface el requisito de acceso igualitario o equivalente

Falso. La tecnología de adaptación ha evolucionado enormemente y prácticamente cualquier elemento de una página Web convencional puede ser total o parcialmente accesible, sin alteraciones visibles en el diseño. (Recuerde que el trabajo se realiza bajo la superficie.) Al limitar a los visitantes con problemas de visión a un sitio sólo de texto, asumimos que éstos no ven nada en absoluto o que los que sufren problemas de movilidad no utilizan imágenes. También se asume que a dichos usuarios no les interesa comprar en nuestro sitio comercial ni participar en un foro de debate en línea. En definitiva, el desfasado enfoque de sólo texto no sirve para nadie. Además, la creación y el mantenimiento de páginas sólo de texto resulta más costoso que añadir sencillas etiquetas y atributos de acceso a nuestro sitio.

Mito: la accesibilidad cuesta demasiado

Falso. ¿Qué cuesta crear un vínculo Skip Navigation o escribir un resumen de tablas como vimos en un capítulo anterior? ¿Qué cuesta escribir un breve texto `alt` para las imágenes de nuestra página? Bastan unos minutos. Puede que cobre por horas pero, a menos que cobre millones de euros a la hora, el coste de añadir más opciones de acceso

requeridas por la Prioridad 1 de la WAI o la U.S. Section 508 es insignificante, sobre todo si al hacerlo se protege de querellas antidiscriminatorias. Éstas si que cuestan.

No obstante, un cumplimiento de nivel superior que requiera un trabajo más especializado, resultará más costoso que estas sencillas tareas. Por ejemplo, cuesta mucho más crear títulos cerrados para vídeos de RealTime o QuickTime o para subtitular servidores de noticias en tiempo real. Pero son escasos los sitios que ofrecen el tipo de contenidos que requieren esta especialización. Como dijimos al principio del capítulo, el acceso es un proceso continuo, por lo que el coste del cumplimiento de sus directrices básicas es prácticamente nulo.

En sitios de gran tamaño cuyo contenido lo actualiza personal editorial (no programadores), para añadir acceso a las nuevas páginas basta con actualizar el sistema de administración de contenidos con los atributos necesarios. En un sitio dinámico, puede resultar tan sencillo como modificar las plantillas que generan las páginas. La inclusión de atributos de acceso en formularios, de resúmenes estructurales en diseños de tablas y la realización de ajustes similares en plantillas globales puede ser un coste único que sea rentable para atraer a nuevos clientes y para alejar a los posibles vampiros de la profesión legal.

Estamos demasiado ocupados gastando millones como para invertir correctamente algunos euros

Tenemos un amigo. Siempre compra CD que no tiene tiempo de escuchar y DVD que no tiene tiempo de ver. Alquila un estudio por si acaso le entran ganas de pintar aunque hace dos años que no pinta. Tiene todos los canales digitales aunque nunca ve la televisión porque sale de marcha todo el tiempo. El único inconveniente de este frenético estilo de vida consumista es un pequeño dolor en la muela inferior. Lleva dos meses con dolor de muelas, pero no se puede permitir una visita al dentista.

Probablemente opine que las prioridades de nuestro amigo son erróneas, pero su actitud no es diferente a la de muchas empresas.

Los que se quejan del coste de la accesibilidad (y que también se suelen quejar del coste de los estándares) suelen ser los mismos que se gastan miles de euros en detección de navegadores, secuencias de comandos condicionales, CSS condicional e incluso en HTML condicional, como mencionamos en un capítulo anterior. Opinan que no malgastan su dinero al enviar 10 hojas de estilo diferentes a 10 navegadores diferentes, cuando bastaría con un solo archivo CSS. ¿Y dedicar unos cuantos euros a la accesibilidad? Cuesta demasiado, dicen.

En febrero de 2003, `MSN.com` envió a Opera 7 diferente HTML y CSS del que envió a Opera 5 ó 6 (véase figura 14.3). Se preguntará si estos archivos condicionales mejoraron la representación visual en Opera. En realidad, la empeoraron (http://deb.opera.com/howcome/2003/2/msn/). Y no hace falta decir que el sitio no era totalmente accesible, según una prueba de navegadores sólo de texto (véase figura 14.4) y según Bobby de Watchfire, una servicio de validación de accesibilidad en línea gratuito (http://bobby.cast.org/bobby/bobbyServlet?URL=http%3A//www.msn.com/&gl=wcag1-aaa).

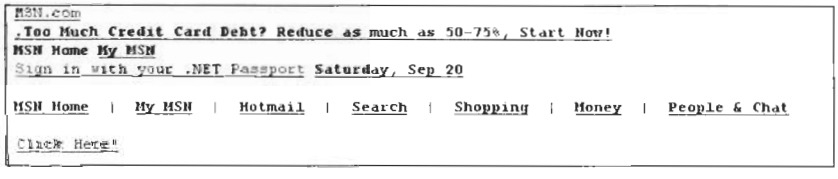


Figura 14.3.

¿Puede decir, a simple vista, que este sitio es inaccesible (www.msn.com)? Podría si fuera discapacitado, como se muestra en la figura 14.4.

Figura 14.4.

El navegador de texto Lynx revela algunos de los problemas de accesibilidad de MSN, incluyendo formularios de búsqueda en los que no se puede introducir texto, vínculos ciegos y referencias inútiles a mapas de imágenes.

Microsoft no es la única organización que malgasta fortunas en soluciones desfasadas que aumentan exponencialmente el coste de la publicación Web al tiempo que no resultan ventajosas para nadie. Tampoco es la única empresa que dilapida sus presupuestos en excesivos problemas técnicos imaginarios mientras se ahorran unos céntimos en accesibilidad.

Muchas empresas que afirman no poder permitirse la implementación ni siquiera de los niveles de accesibilidad más básicos parecen disponer de cantidades ingentes de dinero para programación de servidores, medios en tiempo real y, a menudo innecesariamente, complejo JavaScript. Si estos gastos se consideran parte del negocio, el coste mucho más reducido que supone el acceso debería considerarse de la misma forma.

Mito: la accesibilidad obliga a crear diseños primitivos y simples

Falso. Las imágenes, diseños de tabla, CSS, JavaScript, tecnologías del lado del servidor

como PHP y otros productos del diseño Web contemporáneo son perfectamente compatibles con la Prioridad 1 de la WAI y la Section 508: simplemente requieren una atención especial. También se pueden utilizar tecnologías como Flash y QuickTime, siempre que cumplan las directrices aplicables a las mismas (que analizaremos en breve). Todos los sitios producidos por Happy Cog en los dos últimos años utilizan secuencias de comandos basadas en DOM, diseños CSS o híbridos, imágenes GIF y JPEG, etc. Aun así, pasan todas las pruebas en línea de validación de acceso con bastante solvencia.

Como veremos en breve, el hecho de pasar las pruebas en línea de validación de acceso no garantiza que nuestro sitio sea realmente accesible. Existen límites en lo que respecta a las pruebas que el software puede realizar; el juicio humano debe moderar y evaluar todas estas pruebas. Por otra parte, si no pasan las pruebas de la Prioridad 1 de la WAI y la Section 508, es muy probable que nuestros sitios no sean compatibles.

Mito: según la Section 508, los sitios deben tener el mismo aspecto en todos los navegadores y agentes de usuario

Falso. ¿Cómo se conseguiría? Naturalmente, la mayoría de los sitios que cumplen la 508 se verán mejor en los nuevos navegadores que en los antiguos. No va en contra de la ley, ya que los usuarios Web pueden actualizar sus navegadores de forma gratuita por medio de descargas. El contenido debe ser accesible en Lynx; lectores de pantalla, PDA y otros agentes. El diseño visual no se puede presentar en muchos de estos entornos y no tiene por qué tener el mismo aspecto en diferentes navegadores. Los métodos de la vieja escuela que pretendían que los sitios tuvieran el mismo aspecto en todos los navegadores y en todas las plataformas son una de las razones de la cantidad de sitios inaccesibles, inválidos y difíciles de mantener de la Web.

Mito: la accesibilidad es sólo para discapacitados

Falso. Es verdad que el cumplimiento de estas leyes puede mejorar (o, en algunos casos, ofrecer por primera vez) acceso a los usuarios con graves discapacidades. Pero también ayuda a los siguientes:

- A cualquiera que utilice Palm Pilots, teléfonos móviles compatibles con la Web y otros dispositivos de navegación no tradicionales, un sector del mercado en alza.
- A los usuarios con lesiones o discapacidades temporales (por ejemplo, una muñeca rota).
- A los usuarios con problemas de visión menores y reversibles, incluyendo los nacidos después de la Segunda Guerra Mundial (un amplio segmento de la población).
- A los usuarios que accedan a sitios de forma temporal desde un entorno distinto al habitual, como por ejemplo, a través de kioscos o navegadores de funcionalidad limitada como los que se encuentran en aeropuertos y otras instalaciones públicas.

- A los propietarios de sitios que quieren llegar a cualquiera de estos usuarios en lugar de que se decanten por un sitio de la competencia.

- A los propietarios de sitios que quieran beneficiarse de los motores de búsqueda, los mayores usuarios ciegos de todos (consulte un apartado anterior).

Mito: Dreamweaver MX, Bobby de Watchfire o cualquier otra herramienta resuelve todos los problemas

Falso. Estas herramientas ayudan, pero no pueden reemplazar al juicio humano. El servicio de validación de accesibilidad Bobby de Watchfire (véase figura 14.5), el Portal Cynthia Says (http://www.contentquality.com) o LIFT de UseableNet (http://www.usablenet.com/) le pueden ayudar a probar sus problemas de cumplimiento con la WAI o la Section 508. Dreamweaver MX incluye elementos de LIFT que pueden ayudarle y animarle a crear de forma más accesible.

Pero estas pruebas y herramientas no son la panacea ni soluciones instantáneas. Piense en ellas como complementos que le pueden ayudar a formular prácticas más indicadas y a identificar áreas problemáticas concretas en los proyectos que desarrolle.

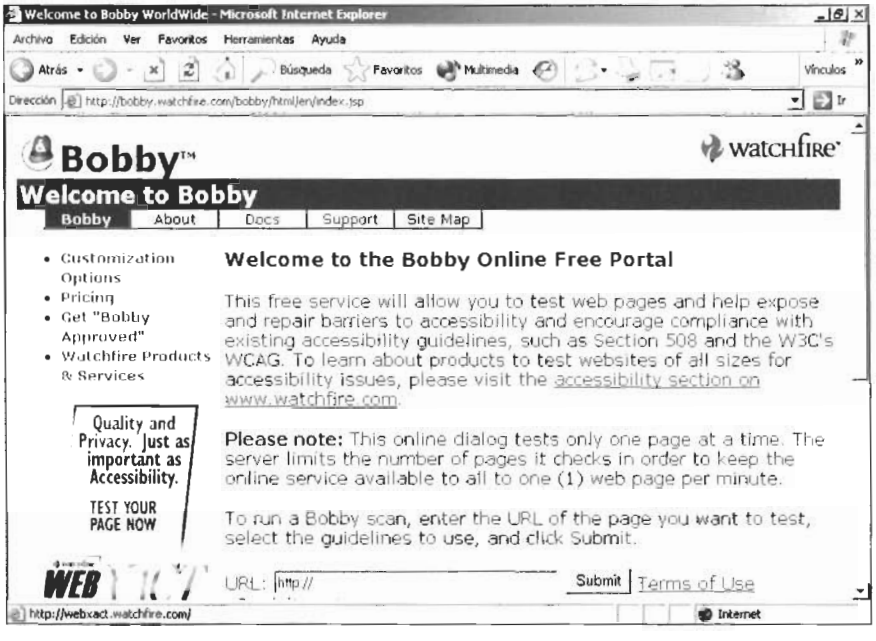


Figura 14.5.

Bobby de Watchfire (http://bobby.watchfire.com/), un servicio de validación de accesibilidad gratuito disponible en la red, le puede ayudar a aplicar en sus páginas el cumplimiento con las directrices de la WAI o la Section 508. Pero como muchos programas, WAI tiene sus limitaciones y, por ello, debe evaluar con sentido común los informes que le ofrezca.

Mito: los diseñadores pueden ignorar las leyes de accesibilidad si los clientes se lo piden

Está por ver. No conocemos ningún caso en el que un diseñador Web haya sido responsable de crear sitios inaccesibles, todavía. Pero el Departamento de Justicia estadounidense ha perseguido a arquitectos que incumplen la ley, ya sea porque se lo indiquen sus clientes o no. Puede que algún día los diseñadores Web se enfrenten a las mismas penas. Es nuestra labor la educación de los clientes (o jefes), no culparles cuando sabemos de sobra que estamos haciendo algo incorrecto.

Consejos de accesibilidad, elemento a elemento

Las siguientes directrices ofrecen enfoques que puede utilizar para que los elementos más utilizados de una página Web cumplan las recomendaciones de accesibilidad de la WAI (o del gobierno).

Imágenes

Si no incluimos el texto `alt`, los usuarios de Lynx, de lectores de pantalla y de otros navegadores poco habituales y dispositivos únicamente podrán oír o ver [IMAGEN] [IMAGEN] [IMAGEN] [IMAGEN] [IMAGEN] [IMAGEN] o algo igual de inútil. En capítulos anteriores encontrará un ejemplo visual de lo mismo. La falta de textos `alt` también se indicará como error de acceso WAI y como error de validación XHTML. Utilice el atributo `alt` del elemento `img` (http://www.w3.org/WAI/GL/WCAG20/checkpoints.html) para describir la función de cada imagen.

Nuestro querido amigo el atributo null alt

En el caso de imágenes sin significado, como GIF de espaciado (no queremos decir que siga utilizando este tipo de imágenes), utilice `alt=""`, también conocido como atributo `null alt` o texto `null alt`. No agrave los problemas de los usuarios con texto `alt` literal para todas las imágenes sin significado como `alt="pixel spacer gif"` o `alt="table cell background color gradient"`. Utilice el atributo `null alt` para las imágenes destinadas a crear efectos de diseños puramente visuales (sin significado semántico).

Utilice atributos alt que transmitan el significado a sus visitantes

Utilice atributos `alt` que transmitan el significado a sus visitantes y no para sus colegas y compañeros.

Por ejemplo, en un logotipo que funcione como vínculo a la página principal, utilice `alt="Smith Company home page"` en lugar de `alt="smith_logo_rev3"` o `alt="Smith Company logo"`. Para un usuario con problemas visuales no tiene interés alguno saber que una imagen que no puede ver es un logotipo. El hecho de que al pulsar sobre la imagen pueda acceder a la página principal tiene mucho más sentido. Si es totalmente necesario hacerlo, puede utilizar algo como esto:

```
alt="Smith Company home page [logo]"
```

Evite el software de ayuda que genera atributos `alt` automáticamente, ya que, con toda seguridad, generará inútiles textos `alt` derivados directamente de los nombres de archivo:

```
alt="smith_logo_32x32"
```

En definitiva, no utilice un robot para realizar un trabajo humano. De hecho...

No se fíe del software que realiza trabajos humanos

Si su página pasa las pruebas de validación WAI o Section 508 de Bobby, no asuma que sus atributos `alt` vayan a funcionar. Una página que utilice `alt="mickeymouse"` en todas las imágenes (`alt=""`) podría pasar las pruebas sin problema. Ningún programa de software le puede decir si sus textos `alt` son correctos. Y, francamente, no queremos vivir en un mundo en el que los programas informáticos puedan realizar este tipo de juicios. Si no sabe de qué estamos hablando, sólo tiene que ver Matrix, Blade Runner o Minority Report.

La información de pantalla alt

Algunos de los principales navegadores representan equivocadamente atributos `alt` como informaciones de pantalla cuando el ratón del usuario se sitúa sobre una imagen. Aunque millones de usuarios Web ya están acostumbrados, es una idea horrible por muchas razones. En esencia, el texto `alt` es una herramienta de accesibilidad, no un elegante truco de información en pantalla. (El atributo `title` es el adecuado para realizar elegantes trucos de información de pantalla.) El W3C indica explícitamente que el texto `alt` debe ser visible únicamente cuando las imágenes no se pueden ver:

> *El atributo `alt` especifica texto alternativo que se reproduce cuando la imagen no se puede visualizar. Los agentes de usuario deben reproducir textos alternativos cuando no puedan admitir imágenes, no puedan admitir un determinado tipo de imagen o cuando se hayan configurado para no representar imágenes.*

Ningún navegador debería representar texto `alt` redundante que describa lo que el visitante vidente ya puede ver. Pero en IE/Windows, por ejemplo, ocurre exactamente esto (véase figura 14.6), y Netscape/Windows fue el primero que lo hizo. No es nuestro problema, a no ser que nos lleve a escribir "creativos" textos `alt` que no sirvan para explicar las imágenes a aquéllos que las pueden ver, su principal objetivo.

No utilice alt en imágenes de fondo

Los principiantes en temas de acceso me suelen preguntar si es necesario escribir texto `alt` para las imágenes de fondo CSS (o HTML). Es una pregunta lógica y razonable, con una sencilla respuesta: no. De hecho, no se podría aunque lo intentáramos. No hay atributos `alt` en CSS. (Por ejemplo, no existe un atributo `alt` para una imagen de fondo en una etiqueta `<body>`.)

Si la imagen de fondo debe comunicar un significado especialmente importante que no ofrece el texto de la página, por ejemplo si el texto de la página sólo dice "Era un hombre honesto" y la imagen de fondo CSS es un retrato de Abraham Lincoln, puede intentar

incluir las palabras "Presidente Abraham Lincoln" dentro de un elemento `title` del atributo `body` o un atributo `table summary` si ha utilizado tablas. O también podría, con mucho menos sentido, incluir el texto explicativo dentro de una etiqueta `noscript` si supone que los usuarios que no pueden ver imágenes no disponen de un entorno compatible con JavaScript.

Mejor aún, puede descartar la idea. Una fotografía de Lincoln con el texto "Era un hombre honesto". ¿Qué tipo de página es ésa?

QuickTime de Apple y otros medios de reproducción de vídeo

Cuando se necesite un componente, incluya un vínculo claro al elemento necesario.

Si utiliza una imagen para realizar el vínculo al componente necesario, asegúrese de que dispone del correspondiente texto `alt`.

Para mejorar la accesibilidad del vídeo QuickTime (o REAL), utilice una herramienta de captura o un estándar Web como SMIL para crear texto y títulos descriptivos que acompañen a las pistas de audio. En el artículo "SMIL When You Play That" de A List Apart encontrará un resumen de SMIL orientado a diseñadores (http://www.alistapart.com/stories/smil/).

El titulado de vídeo requiere experiencia, tiempo y dinero. Sobre todo experiencia. Y tiempo. Y dinero. Y experiencia. Si su cliente es el que paga la reproducción de vídeo, asegúrese que el titulado se incluye en el presupuesto de producción y el plazo de entrega.

WGBH Boston es un excelente sitio en lo que respecta a la reproducción de QuickTime y Flash accesible (http://main.wgbh.org/wgbh/access/). Joe Clark ha dirigido los trabajos de vídeo accesible de BMW Films (http://www.BMWFilms.com/). Puede que estos sitios le sirvan de inspiración.

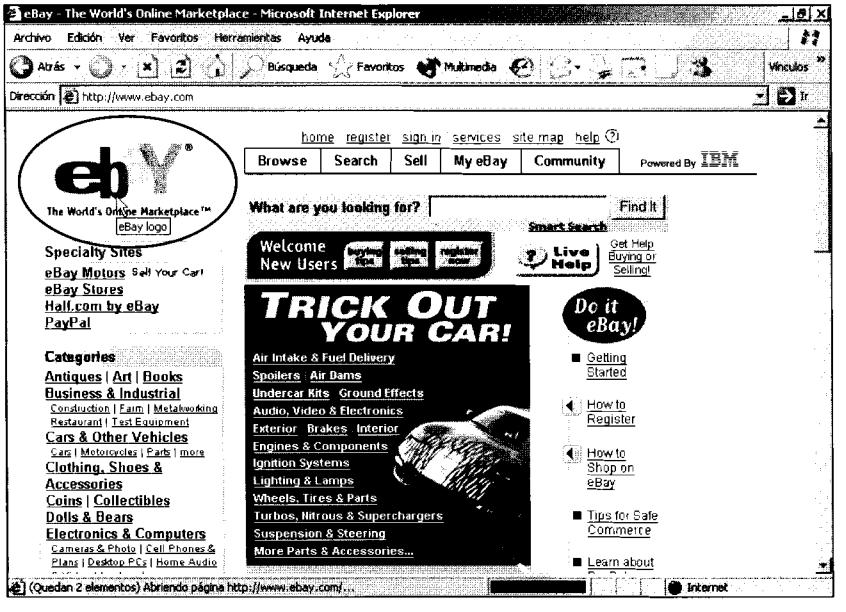


Figura 14.6.

La posición del logotipo de eBay sugiere que es un botón sobre el que se puede pulsar, pero no lo es (www.ebay.com). Su atributo alt ("logotipo eBay") es correcto pero parece redundante en navegadores como IE/Windows que muestran texto alt como información de pantalla, aunque no deberían hacerlo.

Macromedia Flash 4/5

Macromedia Flash 4 y 5 cuentan con opciones de accesibilidad limitadas, como la posibilidad de crear pistas de audio que describan botones de navegación. Si no puede actualizar su versión a Flash MX, utilice estas opciones y ofrezca alternativas HTML. Si puede, utilice Flash MX.

Macromedia Flash MX

Comercializado en abril del 2002, Macromedia Flash MX ofrece opciones de accesibilidad considerablemente mejoradas, incluyendo compatibilidad con lectores de pantalla, aunque la mayoría de éstas sólo funciona en entornos basados en Windows, ya que Flash MX se comunica con Microsoft Active Accesibility.

Los lectores de pantalla, denominados incorrectamente navegadores de voz o lectores de texto, son navegadores que reproducen el texto en voz alta. Actualmente, las mejoras de acceso de Flash MX son compatibles con dos de los principales lectores de pantalla, JAWS de Freedom Scientific (véase figura 14.7) y Windows-Eyes de GW Micro (http://www.gwmicro.com/press/flash.htm).

Flash MX cumple con varios de los requisitos especificados en la U.S. Section 508, como indicamos a continuación:

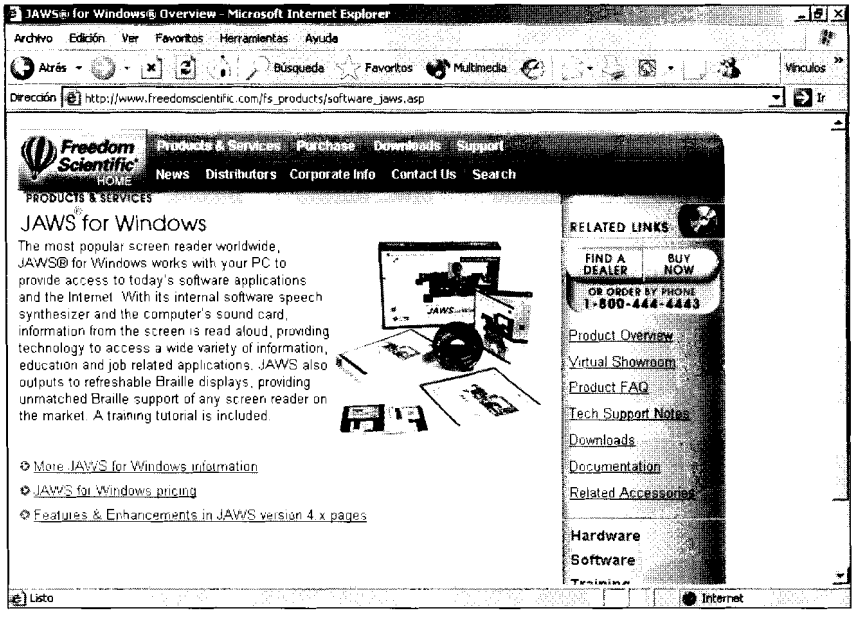


Figura 14.7.

JAWS de Freedom Scientific, un lector de pantalla, trabaja con los navegadores de Microsoft para ayudar a los discapacitados visuales a acceder a contenidos Web y a navegar de forma accesible por sitios Web creados en Flash MX (http://www.freedomscientific.com/fs_products/software_jaws.asp). Además, el sitio de Freedom Scientific cumple con la Prioridad 1 de la WAI y la U.S. Section 508, algo poco habitual en este tipo de empresas.

- Aumento del tamaño del contenido.
- Navegación sin necesidad de ratón.
- Sincronización de sonido.
- Compatibilidad con paletas de colores personalizadas.

Al mismo tiempo, soluciona los siguientes problemas identificados por la U.S. Section 508:

- Posibilidad de crear equivalentes en texto.
- Posibilidad de minimizar el daño causado por la creación incorrecta de eventos animados.
- Posibilidad de crear versiones accesibles de botones, formularios y etiquetas.
- Al igual que en XHTML, posibilidad de especificar el orden de tabulación para ayudar a los usuarios que naveguen sin ratón.

En Flash MX; el orden de tabulación se especifica por medio de ActionScript, la versión de Macromedia de ECMAScript. En un apartado posterior encontrará más información sobre cómo especificar el orden de tabulación en XHTML.

Ninguna de estas mejoras funciona si no aprende a utilizarlas y las incorpora a su proceso de creación en Flash. Tampoco se trata de mejoras particularmente sencillas de entender e implementar. Y existen determinados aspectos de accesibilidad básicos que eluden a Flash MX.

En el número 143 de A List Apart, encontrará más detalles sobre la creación accesible en Flash (http://www.alistapart.com/issues/143/). Se analizan las mejoras de acceso de Flash y sus limitaciones, en artículos de Joe Clark y del diseñador Andrew Kirkpatrick del CP/WGBH National Center for Accessible Media y, si no ha salido disparado a consultarlo, le sugerimos que lea los dos libros que recomendamos al inicio del capítulo.

Consejos adicionales sobre Flash

Cuando se necesite un componente, incluya un vínculo claro al elemento necesario.

Si utiliza una imagen para realizar el vínculo al componente necesario, asegúrese de que dispone del correspondiente texto `alt`.

Si utiliza JavaScript para detectar la presencia o ausencia de Flash, elabore un plan alternativo, es decir, un vínculo claro al elemento necesario, para aquéllos que no puedan utilizar o no utilicen JavaScript. Al mismo tiempo, si utiliza JavaScript para detectar la presencia o ausencia de Flash, por el amor de Dios, asegúrese de que sabe lo que está haciendo.

El medio se encuentra plagado de restos de secuencias de comandos de detección de navegadores y complementos, y de los cuerpos de los usuarios Web atrapados en el fuego cruzado.

Debe comprender que, a pesar de sus más sinceros esfuerzos, hay mucha gente que no podrá acceder a su contenido Flash.

Color

Si utiliza el color para denotar información (como por ejemplo, para indicar dónde se puede pulsar), refuércelo con otros métodos (por ejemplo, con vínculos en negrita o subrayados). Si ha desactivado el subrayado por medio de CSS, puede incluir vínculos en negrita que resalten sobre el texto normal. Si lo hace, evite utilizar texto en negrita sin subrayar, para de esta forma no confundir a los daltónicos sobre qué texto en negrita es un hipervínculo y qué texto en negrita es simplemente texto en negrita. Si la diferencia entre texto de hipervínculo y texto normal es evidente incluso para un daltónico (si el texto es de color negro y los vínculos son de color blanco, por poner el ejemplo más extremo), la negrita o cualquier otro método de diferenciación no serán necesarios.

Evite hacer referencia al color en el texto. La instrucción "Consulte el cuadro amarillo si necesita ayuda" no servirá de nada para los invidentes (o para los que no puedan ver el color).

Preste especial atención al crear armoniosos esquemas de color, cuyas diferencias puedan no resultar aparentes para los que sufren determinados tipos de daltonismo.

Puede visitar http://www.vischeck.com/, dirección en la que podrá comprobar cómo ven sus páginas Web los usuarios con diferentes tipos de daltonismo.

CSS

Pruebe sus páginas con y sin hojas de estilo para asegurarse de que resultan legibles de ambas formas. No se preocupe por los cambios en el diseño gráfico al desactivar los estilos, a menos que dichos cambios inutilicen el sitio.

El marcado estructurado transmite el significado cuando desaparecen las CSS

Si usa XHTML bien estructurado en la creación de sus páginas, éstas funcionarán mejor incluso cuando desactive los estilos o no haya estilos disponibles (véanse figuras 14.8 y 14.9). Los lectores recibirán el marcado estructurado con o sin CSS, como si utilizaran la característica de acceso de tamaños de fuentes de IE/Windows que describimos en un capítulo anterior. Como hemos mencionado a lo largo del libro, enfatice la estructura y evite la divitis. El marcado como el que mostramos a continuación no tiene mucho sentido cuando el usuario no puede disponer de la CSS:

```
<div class="header">Headline</div>
<div class="copy">Text</div>
<div class="copy">Text</div>
<div class="copy">Text</div>
```

No se fíe de lo que ve en un solo navegador

Escriba CSS válidas y pruébelas en varios navegadores. No escriba CSS no válidas que funcionen en un determinado navegador. Una CSS incorrecta puede convertir una página en ilegible. No todos los usuarios Web saben cómo desactivar las CSS, no digamos reemplazar una CSS con hojas de estilo de usuario y, en la actualidad, no todos los navegadores admiten hojas de estilo de usuario.

Figura 14.8.

El sitio personal del autor en un navegador compatible con CSS activadas (www.zeldman.com). Ya sabemos que lo vimos en un capítulo anterior. Paciencia.

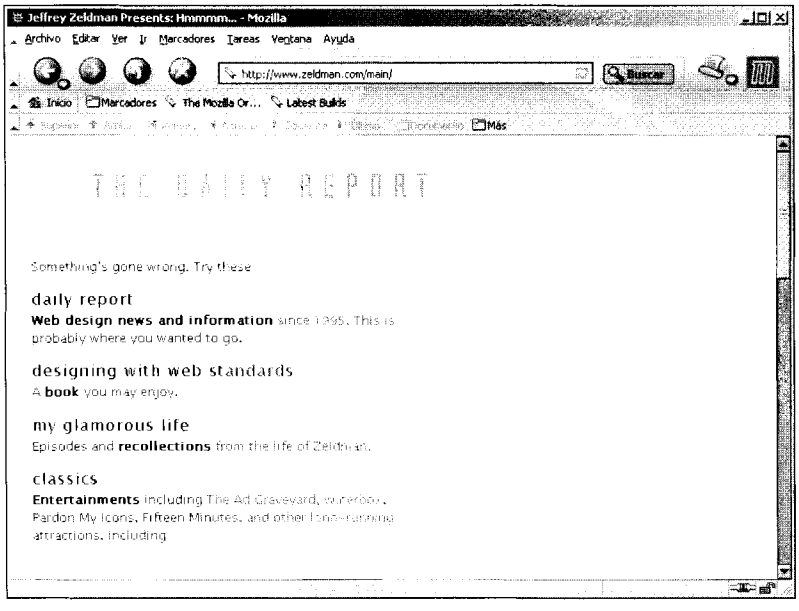


Figura 14.9.

Cuando se desactiva CSS, la relación del subtítulo con el párrafo se conserva ya que el sitio se ha creado con XHTML estructural y sencillo.

Preste atención al tamaño del texto

No asuma que lo ha hecho todo bien con tan sólo utilizar emes en lugar de píxeles (consulte un capítulo anterior). Evite el texto basado en píxeles que no se pueda cambiar de tamaño o utilice conmutadores de estilo basados en DOM o en el servidor para que los usuarios puedan cambiar el tamaño del texto aunque sus navegadores les nieguen este derecho. Los conmutadores de estilo basados en DOM se describen en un capítulo posterior.

No se fíe, repetimos, no se fíe de los resultados de las pruebas de validación de acceso

No asuma que su CSS es correcta simplemente porque su página haya superado las pruebas de accesibilidad de Bobby. Una página que utilice fuentes de 7 píxeles ilegibles puede superar dichas pruebas.

Imágenes cambiantes y otros comportamientos de secuencia de comandos

Diseñe su código para garantizar que los vínculos funcionan incluso cuando se desactiva JavaScript. Para ello, pruebe a desactivar JavaScript en su navegador.

Ofrezca alternativas para los usuarios que no utilizan el ratón

Los usuarios con defectos de movilidad pueden utilizar navegadores compatibles con JavaScript (y de hecho lo hacen) pero puede que no puedan utilizar el ratón para hacer clic o realizar otros movimientos. Proporcióneles código alternativo:

```
<input type="button" onclick=
"setActiveStyleSheet('default');
return false;" onkeypress=
"setActiveStyleSheet('default');
return false;" />
```

En este ejemplo, `onkeypress` es el equivalente de `onclick` para usuarios que no utilizan el ratón. Las dos líneas de código conviven pacíficamente. El código alternativo es invisible para los usuarios que utilizan el ratón. Es cierto que crear dos códigos para la misma función añade algunos bytes al peso total de la página. En este caso, el aumento fraccional del ancho de banda se compensa con la presencia de nuevos usuarios, en lugar de castigarlos por sus discapacidades.

Utilice noscript para los que no pueden utilizar JavaScript

La sección Daily Report de `zeldman.com` utiliza JavaScript en un elemento `form` para proporcionar acceso a páginas anteriores. A continuación reproducimos el código, en el que hemos eliminado algunos elementos para que resulte más claro. Comprobará que se utiliza tanto `onclick` como `onkeypress` para que el usuario pueda acceder al URL de una página anterior:

```
<form action="foo"><input type="button"
value="Previous Reports"
onclick="window.location='http://
www.zeldman.com/daily/0103c.shtml';"
onkeypress="window.location='http://
www.zeldman.com/daily/0103c.shtml';"/>
...
</form>
```

Resulta indicado para los que utilizan navegadores compatibles con JavaScript y que no lo han desactivado por razones religiosas. Pero, ¿qué pasa con los que no pueden utilizar JavaScript? Bastaría con un sencillo vínculo dentro de un elemento `noscript`:

```
<noscript>
<p><a href="/daily/0103c.shtml">Previous
Reports</a></p>
</noscript>
```

A continuación incluimos el código con todos los componentes:

```
<form action="foo"><input type="button"
class="butt" value="Previous Reports"
onclick="window.location='http://
www.zeldman.com/daily/0103c.shtml';"
onkeypress="window.location='http://
```

```
www.zeldman.com/daily/0103c.shtml';"
onmouseover="window.status='Previous
Daily Reports.'; return true;"
onmouseout="window.status='';return
true;" />
<noscript>
<p class="vs15"><a href="/daily/
0103c.shtml">Previous Reports</a></p>
</noscript>
</form>
```

Los que no estén familiarizados con JavaScript puede que encuentren este código un tanto desalentador, sobre todo al verlo impreso en un libro. Pero es cosa de niños, como le dirá cualquier programador experimentado. Cualquier diseñador podría escribir este código. A pesar de su rudimentaria naturaleza, sirve para usuarios Web "convencionales", para los que tienen dificultades motrices, para los que tienen problemas de visión, para los que utilizan dispositivos no tradicionales y para los del tipo paranoico que utilizan navegadores compatibles con JavaScript pero que desactivan JavaScript.

Evite al máximo las secuencias de comandos generadas

Evite el uso de secuencias de comandos generadas por Dreamweaver y GoLive, que asumen ciertas cosas con respecto a los navegadores y a las plataformas. Como mínimo, pruebe las páginas creadas de esta forma en navegadores no convencionales y con JavaScript desactivado.

Aprenda más

La interacción entre los comportamientos controlados por secuencias de comandos y la accesibilidad puede ser ciertamente compleja, y un análisis completo de la misma se escapa a los objetivos del libro. Si necesita más información sobre JavaScript, puede consultar el artículo "Working with JavaScript" de Apple Internet Developer (http://developer.apple.com/internet/ javascript/) y recursos como `webreference.com` y `scottandrew.com`, junto con los que le mostraremos en un capítulo posterior.

Formularios

Al principio del capítulo, le recomendamos dos libros. *Building Accessible Web Sites* dedica todo un capítulo a la creación de formularios accesibles en línea. *Constructing Accessible Web Sites* también dedica un capítulo a las ventajas e inconvenientes de crear este tipo de formularios. De esta coincidencia puede que se vea inclinado a deducir que la creación de formularios accesibles en línea esté de algún modo implicada. Puede que tenga razón.

No se preocupe. La mayoría de las tareas son sencillas, como la asociación de campos de formulario a las etiquetas correctas (por ejemplo, asociar la zona de texto de un formulario de búsqueda a una etiqueta Búsqueda). Simplemente existen muchas de estas tareas y la descripción de todas ellas supera los objetivos de este capítulo.

Creado lo que considera accesible, lo puede probar en Lynx (http://lynx.browser.org/) o Jaws. Los usuarios de Mac necesitarán Virtual PC (http://www.connectix.com/products/vpc6m.html) o un PC real para ejecutar Jaws. Para los usuarios de Linux, se incluye un lector de pantalla gratuito en la distribución Linux 7.0 SuSE (http://www.hicom.net/~oedipus/vicug/SuSE_blinux.html) o pueden utilizar Speakup (http://www.linux-speakup.org/speakup.html).

Mapas de imágenes

Evite los mapas de imágenes si puede, y siempre es posible. Cuando los necesite, utilice mapas de imágenes del lado del cliente con texto `alt` y ofrezca vínculos de texto redundantes. No recurra a los anticuados mapas de imágenes del lado del servidor.

Diseños de tablas

No se exceda. Escriba sencillos resúmenes de tabla como describimos en un capítulo anterior y utilice CSS para evitar la necesidad de tablas anidadas, imágenes GIF para crear espacios y basura similar, como hemos explicado anteriormente.

Y eso es todo. A pesar de que haya escuchado lo contrario, el uso de sencillos diseños de tablas no es un peligro para el acceso, ni tampoco es nada ilegal ante las directrices de la WAI o la Section 508, y tampoco condenará su alma al tormento eterno. Si el diseño CSS es una opción, selecciónela. En caso contrario, aprenda a quererse un poco más y no se preocupe tanto.

Uso de tablas para datos

Identifique los encabezados de tabla y utilice el marcado adecuado para asociar celdas de datos y celdas de encabezado que tengan dos o más niveles lógicos de encabezados de filas y columnas. En una tabla que enumere los integrantes del reparto de la película The Music Man, un encabezado de tabla podría ser `Actor` y las celdas de tablas asociadas con el mismo serían Robert Preston, Shirley Jones, Buddy Hackett, Hermione Gingold, etc.

Una persona que pueda ver y que utilice un navegador gráfico verá la conexión entre `Actor` y la columna de nombres situada directamente por debajo de la misma. Pero los lectores de pantalla requieren marcado adicional que conecte el encabezado de tabla con sus celdas de datos asociadas.

En el código incluido en http://www.w3.org/WAI/wcag-curric/sam45-0.htm comprobará cómo el grupo WAI clarifica la conexión entre encabezados y sus celdas de datos asociadas (véase figura 14.10).

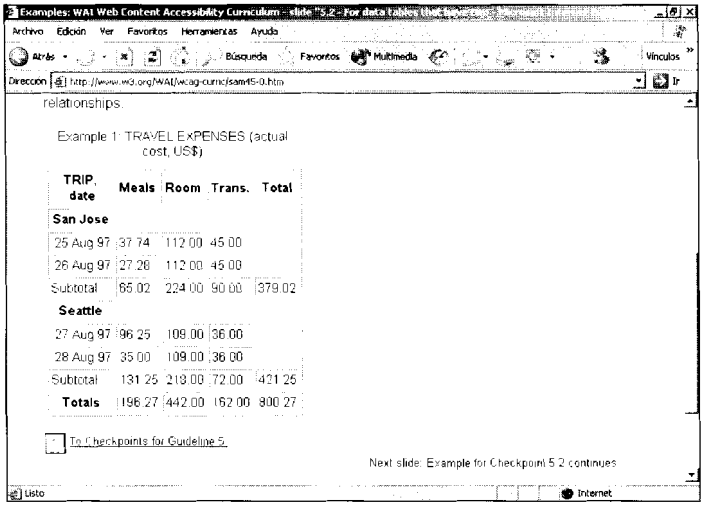


Figura 14.10.

No es muy agradable pero sirve. Una página de ejemplo de la iniciativa de accesibilidad del W3C muestra un método para asociar encabezados a celdas de datos en tablas complejas (http://www.w3.org/WAI/wcag-curric/sam45-0.htm).

Marcos, subprogramas

Simplemente, no los utilice.

Elementos intermitentes o parpadeantes

No. Pero un no rotundo. Puede que no haya utilizado una etiqueta `<blink>` o `<marquee>` desde los tiempos de FrontPage (o ni eso) pero recuerde que la prohibición de elementos intermitentes o parpadeantes se extiende también a contenidos en Flash y QuickTime.

Herramientas de la profesión

Si utiliza un editor visual para crear páginas Web, puede recurrir a diversas herramientas y componentes para simplificar el cumplimiento de las directrices de acceso:

- **SSB Insight LE y GoLive:** El componente gratuito SSB Insight LE para Adobe GoLive identifica automáticamente muchos (pero no todos) incumplimientos de accesibilidad (http://www.adobe.com/products/golive/ssb.html).

- **UseableNet LIFT y Dreamweaver:** Mencionado anteriormente, el programa LIFT de UsableNet para Macromedia Dreamweaver 4 ofrece numerosas opciones además de ayudar en el cumplimento de las directrices. Identifica automáticamente muchos (pero no todos) incumplimientos de accesibilidad. Existen versiones independientes de LIFT y dos de las más recientes incorporan las directrices de facilidad de uso recomendadas por el Nielsen Norman Group (http://www.usablenet.com/lift_dw/lift_dw.html).

- **Dreamweaver MX:** Muchas de las prestaciones de LIFT se han incorporado a Dreamweaver MX, incluyendo un analizador de validación 508, una guía de referencia 508 y herramientas para añadir funciones de accesibilidad a imágenes, tablas y marcos. Recuerde que ninguna herramienta oculta todos los problemas ni es un sustituto infalible para nuestras valoraciones y experiencias. Al igual que el martillo y los clavos, las herramientas sólo son útiles para los que saben usarlas. Al igual que los huevos, la leche y la harina..., vale, no importa, lo ha cogido.

- **Microsoft FrontPage y LIFT:** UsableNet anunció el 9 de julio del 2002 que había integrado su producto LIFT en la conocida herramienta de creación de Microsoft, FrontPage (http://www.usablenet.com/frontend/onenews.go?news_id=45).

  LIFT no impide que FrontPage genere marcado propietario no estándar pero, como ocurre en Dreamweaver MX, permitirá a los usuarios de FrontPage comprobar los problemas de acceso y les guía hacia la inclusión de opciones de accesibilidad en imágenes, tablas, marcos, etc.

Cómo utilizar Bobby

Ya utilice los productos mencionados anteriormente o diseñe el marcado y el código de su sitio a mano, el validador de accesibilidad Bobby de Watchfire debería ser su próxima

parada (http://bobby.watchfire.com/bobby/html/en/index.jsp).

Con tan sólo un botón, Bobby puede comprobar el cumplimiento con el acceso de cualquier página, aunque los detalles requieren un mayor análisis. Tanto la WAI como la Section 508 dependen de una lista de comprobación manual para garantizar el cumplimiento. Al contrario que los servicios de validación de marcado y CSS del W3C, las pruebas de validación de Bobby no le ofrecen una relación de las partes correctas ni una lista de los errores que debe corregir. Por el contrario, debe interpretar el resultado de Bobby. Ésta es la parte complicada. Pero también es la parte educativa y la que le permite ganarse el sueldo como diseñador, programador o especialista Web con conocimientos.

Listas de comprobación

Cuando la versión gratuita en línea de Bobby (o la potente versión binaria que se puede adquirir) termina de evaluar una página, indica que si ésta no incurre en una serie de problemas especificados por la Section 508, se trata de una página aprobada. Bobby nunca dice "Que bien lo ha hecho colega. Esta página es completamente compatible con las directrices de Prioridad 1 de la WAI", la Section 508 o cualquier otra especificación de acceso publicada.

Bobby no puede decir eso. Bobby es un programa. El software depende de algoritmos para comprobar la presencia de problemas comunes. Y, como hemos repetido en infinidad de ocasiones, muchos problemas se escapan al entendimiento de una máquina.

No tenemos tiempo para describir todas las situaciones que pueden producirse al probar el cumplimiento del acceso por parte de un sitio, pero este ejemplo le servirá para que entienda y aplique los tipos de listas de comprobación que genera Bobby. En una ocasión, una versión híbrida (tablas y CSS) de `zeldman.com` pasó la prueba Section 508 del programa, pero la aprobación definitiva dependía de la interpretación de una lista de comprobación generada por Bobby.

La lista de comprobación incluía lo siguiente. "Debe especificar un orden de tabulación lógico entre controles de formulario, vínculos y objetos."

Nuestro buen amigo el atributo tabindex

El atributo XHTML `tabindex` especifica el orden de tabulación de navegación entre controles de formulario. Si no crea un orden de tabulación lógico, los usuarios que dependan de la tabulación (en lugar del ratón) simplemente pasarán de vínculo a vínculo en el orden en que éstos aparezcan en el código fuente XHTML. Puede que no sea la forma más útil de guiarlos por nuestro sitio, en especial si el texto contiene multitud de vínculos o una extensa navegación que aparece en las partes iniciales del marcado.

Al igual que el vínculo Skip Navigation y `accesskey` (elementos descritos en un capítulo anterior), `tabindex` ahorra a los usuarios de lectores de pantalla los peores aspectos de la navegación en serie, lo que les permite acceder directamente al contenido que les interesa. Mientras que Skip

Navigation evita extensas listas de vínculos y `accesskey` proporciona acceso por medio de teclas (al menos en teoría) a los diferentes componentes de una página, `tabindex` proporciona métodos abreviados de acceso en serie a diferentes partes de la página, similar al índice de capítulos de un DVD, que permite a los usuarios adelantar la película hasta la persecución de coches o volver a la escena de amor.

En sitios comerciales, después de crear un orden de tabulación como describiremos más adelante, se prueba en usuarios reales. En sitios personales o sin ánimo de lucro, puede que no disponga de este lujo. Cuando las pruebas con usuarios no son posibles, puede idear un supuesto, crear un orden de tabulación basado en dicho supuesto y esperar a que los visitantes del sitio que utilicen `tabindex` le digan si ha acertado o no.

Llegada en orden de tabulación

En la figura 14.11 se puede ver el sitio híbrido antiguo. Se indica el orden de tabulación antes de utilizar `tabindex`. Cuando los usuarios pulsaban el tabulador, pasaban de un vínculo al siguiente en el orden en que dichos vínculos aparecían en el marcado. Era el comportamiento predeterminado esperado (y atroz). En la figura 14.12 puede ver el orden modificado, gracias a `tabindex`, en el que cada vez que se pulsa el tabulador, el usuario accede al punto al que quiere dirigirse. Los números se han impreso sobre la captura de pantalla para indicar el orden de tabulación, realmente no aparecen en el sitio.

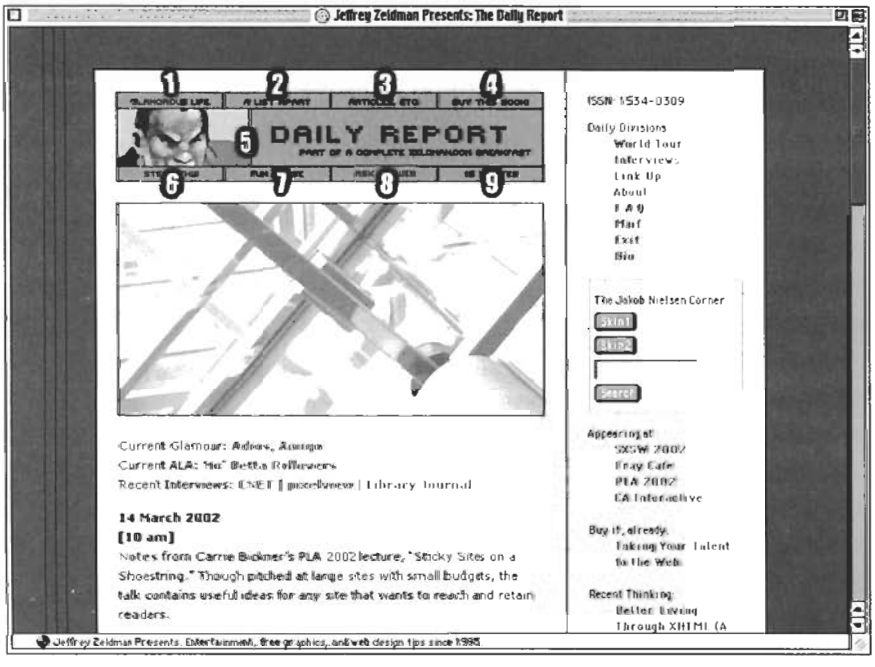


Figura 14.11.

El sitio híbrido de Zedman antes de modificar el orden de tabulación por medio de tabindex. Cuando los visitantes pulsan el tabulador, pasan de un vínculo al siguiente en el orden en que dichos vínculos aparecen en el marcado (www.zeldman.com). Los números impresos no forman parte del sitio; se han añadido para ilustrar el orden de tabulación predeterminado.

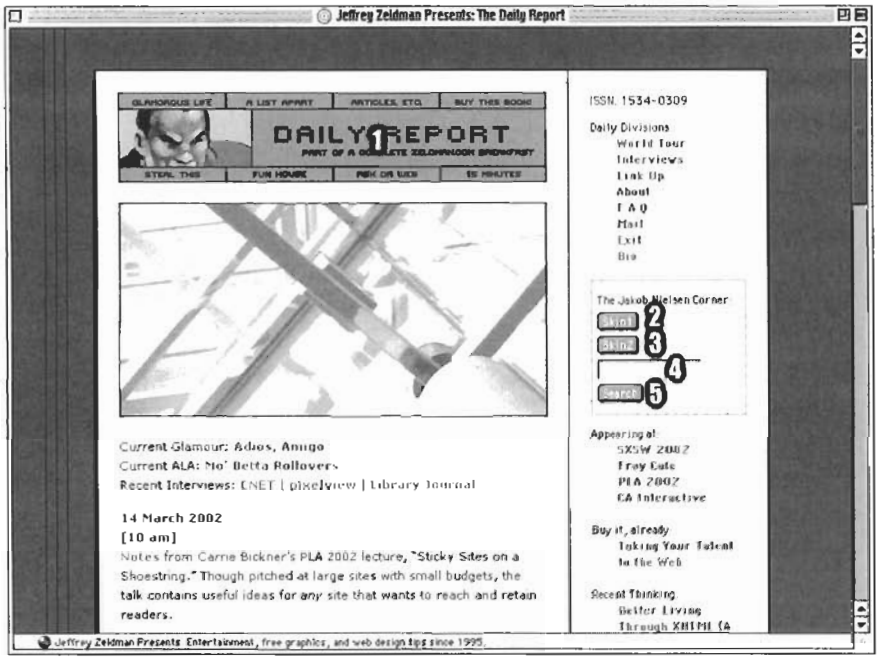


Figura 14.12.

Una vez revisado el orden de tabulación, al pulsar el tabulador el usuario accede a un componente del sitio que puede que sea el que le interese. Los números indican la secuencia de orden de tabulación modificada. El diseño visual del sitio es el de siempre. Los que no utilicen el tabulador no apreciarán cambio alguno.

Después de ponernos mentalmente en la posición de un visitante que no utilizara el ratón y de considerar qué vínculos queríamos utilizar y en qué orden, realizamos la siguiente asignación de orden de tabulación:

1. El primer clic lleva a los visitantes al encabezado de página/botón de inicio para que puedan volver a cargar la página si lo desean, confirmar su ubicación en el espacio Web y, sobre todo, volver a la página inicial si se encuentran en alguna otra parte del sitio.

2. El segundo clic lleva a los visitantes a un botón que les permite seleccionar el tamaño de fuente y la hoja de estilo predeterminados. El funcionamiento del conmutador de estilo basado en DOM se explicará en un capítulo posterior. Su función es mejorar la accesibilidad, permitiendo a los visitantes cambiar el tipo o el tamaño de la fuente (o ambos), aunque también se puede usar de forma más creativa para modificar totalmente el aspecto operativo y visual de un sitio.

3. El tercer clic cambia a un botón que permite al visitante seleccionar un tamaño de fuente que sea más legible para los discapacitados visuales, para los usuarios de Windows que han definido un tamaño de texto mediano de forma predeterminada (consulte un capítulo anterior), etc.

4. El cuarto clic centra el enfoque en el campo de entrada del formulario Search. Al especificar anteriormente el formulario Search en el flujo de vínculos del visitante, evitamos que tenga que pasar por decenas de vínculos no deseados.

5. El quinto clic del tabulador desplaza al usuario hasta el botón Search y le permite ejecutar las búsquedas que haya iniciado en el paso anterior.

6. El sexto clic, en la imagen por debajo del campo, transporta al visitante al botón Previous Report, situado en la parte inferior de la página. Este botón le permite cargar páginas antiguas en orden cronológico inverso.

7. El séptimo clic, también por debajo del campo en la imagen, lleva a los usuarios a un botón Top of the Page que les evita tener que desplazarse manualmente hasta la parte superior de la página. Resulta especialmente útil para los usuarios que no pueden desplazarse con el ratón.

Después de estas siete paradas en boxes, se retoma la tabulación normal, lo que permite a los lectores navegar por los restantes vínculos de la página en el orden en que aparecen.

Creación y comprobación

La modificación del orden de tabulación fue una operación sencilla. Simplemente asignamos un valor `tabindex` a todos los elementos que queríamos priorizar. Por ejemplo, a continuación le mostramos una versión simplificada del XHTML correspondiente a nuestro botón de tamaño de fuente predeterminado antes de cambiarlo para mejorar el acceso:

```
<form action="send">
    <input type="button" />
</form>
```

Y aquí puede ver el mismo botón después de modificarlo. (El único elemento que cambia se resalta en negrita.)

```
<form action="send">
    <input type="button" tabindex="2" />
</form>
```

El siguiente elemento de la secuencia se marcó como `tabindex="3"`, el siguiente como `tabindex="4"` y así sucesivamente. No es física quántica.

Tras realizar este ejercicio, hemos cambiado en dos ocasiones el diseño de `zeldman.com`, por medio de técnicas CSS y XHTML reestructurado. Sin embargo, puede ver los resultados del cambio de orden de tabulación, así como el código fuente, en http://www.zeldman.com/daily/1002a.html.

La mayoría de las tareas de acceso son igual de sencillas, conceptualmente y en su marcado. Entre las tareas habituales que le puede sugerir Bobby destacamos la creación de métodos abreviados de teclado para elementos de formulario y comprobar si la página es legible y utilizable cuando se desactivan las hojas de estilo. Cuando aprenda más sobre acceso, no necesitará las sugerencias de Bobby y, al implementar mejoras de accesibilidad, Bobby realizará cada vez menos sugerencias de este tipo.

En ocasiones, Bobby ofrece sugerencias irrelevantes. Por ejemplo, incluso si su sitio utiliza tablas únicamente para el diseño, Bobby responderá a la presencia de las mismas con un comentario de este tipo: "Si ésta es una tabla de datos (que no se usa únicamente para el diseño), identifique encabezados para las filas y columnas de la misma."

Del mismo modo, Bobby dirá de cualquier sitio, "Si utiliza colores para transmitir información, asegúrese de que esta información también se presenta de otro modo."

Estas declaraciones se quedan obsoletas con rapidez. Son como los murmullos aleatorios de un alma en pena poseída por un episodio sicótico. Por esta razón, algunos expertos en accesibilidad rechazan este tipo de herramientas y basan su trabajo exclusivamente en sus propios conocimientos. Pero para los que no somos expertos, siempre que estemos dispuestos a aguantar las reacciones de Bobby, puede ser de gran ayuda. La herramienta de comprobación de acceso Cynthia Says (http://www.contentquality.com), que apareció al cierre de la edición de este libro, realiza más pruebas que Bobby y puede que le resulte más agradable.

En deferencia al proverbio ruso sobre los beneficios educativos de la repetición, concluiremos este apartado recordándole una vez más que aunque Bobby le ofrezca un informe de salud favorable, puede que su sitio no sea accesible. Ignore los elementos irrelevantes de la lista de comprobación y céntrese en los relevantes, utilice su sentido común y consulte más libros sobre accesibilidad.

Una página, dos diseños

Después del estiramiento facial del orden de tabulación, nuestro sitio tenía el mismo aspecto de siempre. Pero de algún modo, el sitio tiene ahora dos diseños de interfaz de usuario: uno para usuarios de navegadores gráficos tradicionales que navegan con el ratón y otro para los que utilizan el tabulador en navegadores gráficos y no gráficos. Los dos diseños conviven pacíficamente, no requieren versiones de páginas accesibles especiales ni modificar el diseño visual que la mayoría de los usuarios percibe. (Por otra parte, si nuestro esquema de color hubiera representado el texto de forma ilegible para los usuarios con discapacidades visuales, lo habríamos cambiado y puede que, al hacerlo, también hubiéramos modificado el diseño general.)

Cómo planificar el acceso: ventajas

Aunque muchos sitios no están obligados legalmente a proporcionar acceso, puede que tengan que hacerlo algún día. Todos sabemos que las leyes cambian constantemente. También sabemos que todos estamos sujetos a las leyes, nos guste o no. Al aplicar estas mejoras a nuestros sitios, aunque no estemos obligados por la legislación actual, evitaremos tener que realizar costosos cambios en caso de que las leyes cambien el próximo año y también evitaremos el gasto de las denuncias por discriminación (y una mala publicidad).

El acceso a nuestro servicio

Una vez presentada la lógica que se esconde tras el repentino interés de muchos propietarios de sitios por la accesibilidad, diremos que el temor a las denuncias es un motivo equivocado para incorporar el acceso a nuestras técnicas de diseño. Estas mejoras abren nuestros sitios a nuevos visitantes y, ¿qué sitio no estaría dispuesto a tener más visitantes? Los que no puedan acceder a

otros sitios se verán obligados a sentirse leales al nuestro si los acogemos al realizar estos ajustes en nuestro marcado. Si otras tiendas en línea impiden el acceso de usuarios discapacitados o de los que utilicen dispositivos no tradicionales, y la nuestra los acoge, adivine quién venderá a estos clientes y quién no.

No debe olvidar que cuanto más accesible sea nuestro sitio para los visitantes discapacitados y para los usuarios de dispositivos de Internet no tradicionales, más disponible estará su contenido en Google, AllTheWeb y otros motores de búsqueda y directorios. Por el contrario, cuanto menos accesible sea nuestro sitio, menor será el tráfico que recibamos de Google y similares. `zeldman.com`, A List Apart y la mayoría de los sitios diseñados por Happy Cog en los últimos años suelen aparecer en las primeras posiciones de los resultados de los motores de búsqueda, no porque hagan nada especial, sino porque estos sitios están diseñados con marcado estructural con acceso mejorado.

Vaya, estamos intentado alcanzar un plano moral superior y sólo hemos ofrecido razones interesadas por las que implementar la accesibilidad. Veamos dos más.

La implementación de mejoras de acceso puede aumentar su concepto de diseño. Aspectos como el orden de tabulación le permitirán superar la visión del diseño como parte decorativa de un aspecto superficial (operativo y visual) y adentrarse en los reinos del flujo de usuarios, del diseño de contingencia y de la facilidad de uso general. Son aspectos sobre los que diseñadores Web, arquitectos de la información y especialistas piensan. La accesibilidad es simplemente otro aspecto de la planificación que nos llevará a crear sitios adecuados para diferentes necesidades humanas.

La implementación del acceso y la definición de una estrategia de compatibilidad pueden mejorar nuestros conocimientos de programación y ofrecernos nuevas perspectivas que nunca hubiéramos imaginado en otras circunstancias. Si aprendemos las enseñanzas de la WAI y las particularidades de la Section 508 (o de cualquier otro estándar de acceso definido legalmente), aumentará nuestro valor como diseñador Web profesional, situará a nuestra agencia en una posición privilegiada con respecto a nuestros competidores y contribuirá a que nuestros sitios lleguen a más público que nunca. Es lo que quieren todos los propietarios de sitios y a lo que aspira cualquier diseñador o programador. La utilización de la accesibilidad ayudará a que sus visitantes consigan sus objetivos, sí, pero también a que consigamos los nuestros.

Capítulo 15
Cómo trabajar con secuencias de comandos basadas en DOM

En un principio, Microsoft creó JavaScript, y estaba bien. Tras ello, Microsoft engendró JScript, que era diferente. Se produjo un enfrentamiento entre grandes armadas y las llamas de DHTML amenazaban con devastarlo todo. La salvación llegó con el nacimiento del Modelo de objeto de documento (DOM), cuya primera manifestación se denominó DOM Level 1 (http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/). Y era realmente bueno. Por primera vez, el DOM del W3C ofreció a los diseñadores y creadores un medio estándar de acceso a los datos, secuencias de comandos y capas de presentación con los que sus sitios estaban formados.

Desde entonces, el W3C ha continuado con la actualización de sus especificaciones DOM y, con la llegada del The Web Standards Project (WaSP), los navegadores han conseguido admitir al menos la mayor parte de la especificación DOM Level 1, aunque en ocasiones difieran en las formas de admitirla. (Si quiere saber cómo admite el DOM su navegador favorito, visite http://www.w3.org/2003/02/06-dom-support.html.) En este capítulo, conoceremos el DOM y analizaremos algunas de las formas que pueden ayudarnos a realizar útiles tareas, como las de mostrar u ocultar contenidos en respuesta a acciones del usuario, ofrecer opciones de personalización y accesibilidad, y crear menús dinámicos. Le prometemos que no será demasiado complejo ni técnico.

Presentación del DOM

¿Qué es el DOM? Según el World Wide Web Consortium (W3C), (http://www.w3.org/DOM/), el DOM es una interfaz

independiente de navegadores y plataformas, y de lenguaje neutral que permite "que programas y secuencias de comandos accedan y actualicen el contenido, estructura y estilo de documentos. Se puede aumentar el procesamiento del documento y los resultados de dicho procesamiento se pueden incorporar de nuevo a la página presentada."

En un lenguaje sencillo, el DOM permite que otros componentes de una página (hojas de estilo, elementos de marcado y secuencias de comandos) sean accesibles y se puedan modificar. Si nuestra página Web fuera una película, XHTML sería el guionista, CSS sería el director artístico, los lenguajes de secuencia de comandos serían los efectos especiales y el DOM sería el director que controla toda la producción.

Además, en lugar de que el servidor sufra y los conductos se bloqueen con solicitudes HTTP, la interactividad basada en DOM se produce en el lado del cliente (es decir, en el disco duro del visitante). Funciona incluso aunque la conexión finalice.

Un método estándar para que las páginas Web se comporten como aplicaciones

Aunque este aspecto se escapa a los objetivos del libro, el aspecto más atractivo de la interactividad basada en DOM es que puede imitar el comportamiento del software convencional. Por ejemplo, el visitante puede cambiar el orden de los datos tabulares si pulsa sobre el encabezado, como si se tratara de una hoja de cálculo de Excel o el Macintosh Finder (una aplicación que permite a los usuarios ordenar, copiar, cambiar de posición, cambiar de nombre, eliminar o procesar de cualquier otra forma los distintos archivos y carpetas de sus escritorios). (Véanse figuras 15.1, 15.2 y 15.3.)

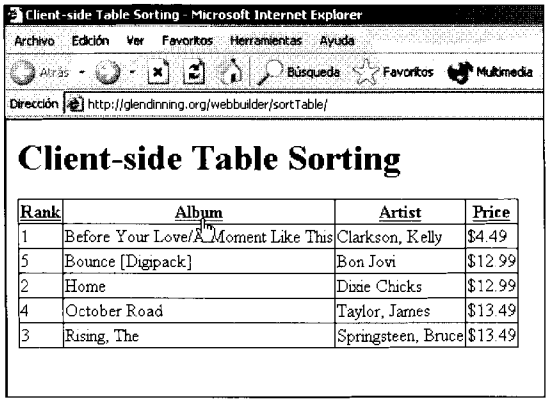


Figura 15.1.

El DOM permite que las páginas Web se comporten como aplicaciones de escritorio. En este ejemplo de Porter Glendinning, los datos se ordenan por título del disco cuando el usuario hace clic en el encabezado Album (http://glendinning.org/webbuilder/sortTable/).

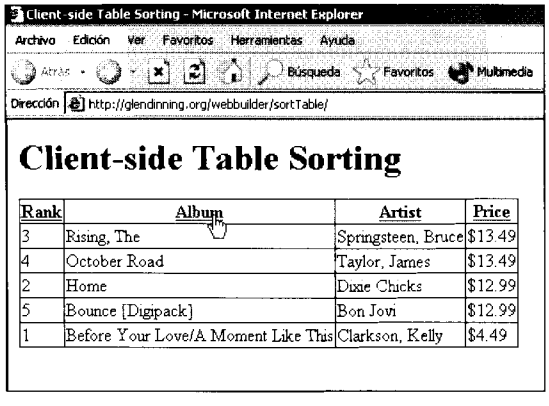


Figura 15.2.

Al hacer clic de nuevo sobre el encabezado, se invierte el orden, que se aprecia inmediatamente en la página original. No se necesita otra página para los resultados.

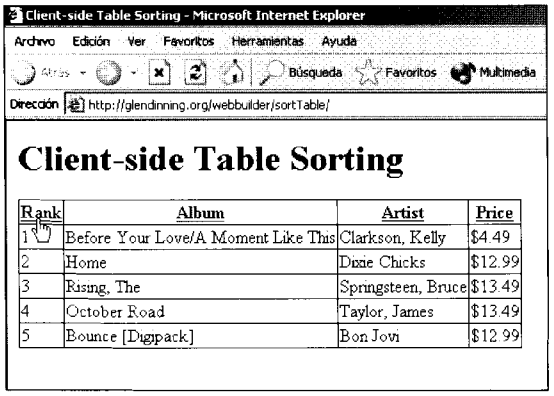


Figura 15.3.

Al hacer clic sobre el encabezado Rank se vuelve a ordenar la lista numéricamente, sin necesidad de cargar una nueva página y sin necesidad de requerir procesamiento ni contacto cliente/servidor. En la dirección http://glendinning.org/webbuilder/ encontrará ejemplos adicionales.

En las páginas Web convencionales, esta actividad requiere negociaciones entre el cliente y el servidor, secuencias de comandos en el servidor, datos y la generación y carga de una nueva página HTML de resultados cada vez que se modifique el orden. Pero con el DOM, toda la actividad se realiza aunque el usuario no esté conectado al servidor. Años después de descargar la página mostrada en las imágenes anteriores, sus datos se podrán ordenar sin necesidad de una conexión activa y sin la distracción del reemplazo de páginas.

Del mimo modo, se puede eliminar un catálogo de compra de un carro de la compra sin necesidad de notificárselo al servidor hasta que el cliente decida adquirir un artículo o buscar productos adicionales. Además, las traducciones de un idioma a otro pueden realizarse con un clic del ratón (véanse figuras 15.4, 15.5 y 15.6). El DOM se encarga de estas tareas mediante la manipulación de datos en respuesta a acciones del usuario. Puede hacerlo porque puede acceder a todos los elementos estándar de una página Web mediante cualquier agente de usuario compatible con el DOM. (Dicho de otra forma, el DOM manipula los elementos Web estándar de la misma forma que ActionScript manipula los componentes de archivos Flash.)

Aunque se trata de ejemplos modestos, sus implicaciones son profundas. Durante años, los programadores que querían crear aplicaciones de interfaz de usuario tenían que usar Java o Flash. Dichas herramientas siguen disponibles, evidentemente, pero la compatibilidad de los navegadores con el DOM permite a los programadores crear potentes y completas aplicaciones basadas en la Web generadas en su totalidad con los estándares que hemos analizado en este libro.

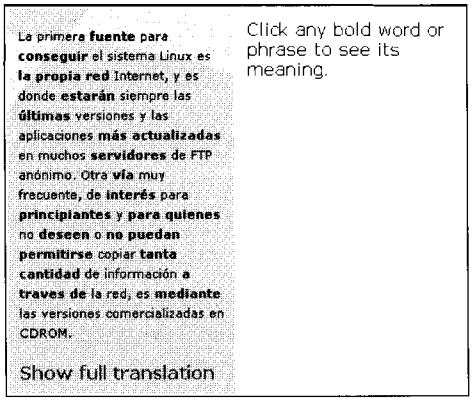


Figura 15.4.

En este ejemplo de David Eisenberg, la traducción español-inglés se consigue con un simple clic de ratón (http://www.alistapart.com/stories/domtricks3/).

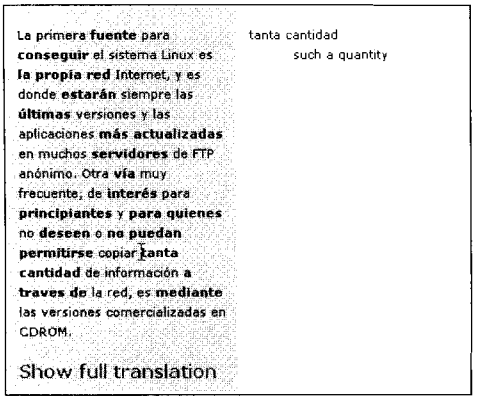


Figura 15.5.

Si lo desean, los lectores ingleses que tengan ciertos conocimientos de español pueden ver la traducción de las palabras o frases que desconozcan.

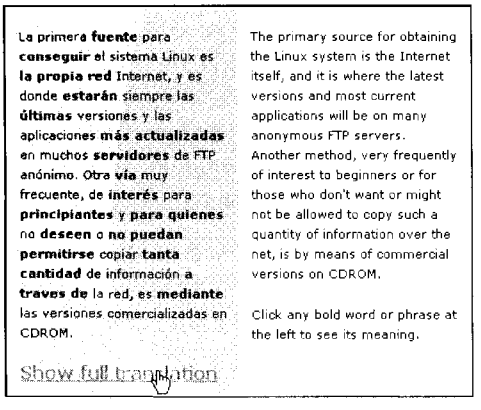


Figura 15.6.

En cambio, los que tengan más dificultades pueden optar por recibir una traducción completa. Toda la actividad se lleva a cabo en el cliente, sin necesidad de una conexión activa y sin necesidad de cargar una página de resultados diferente.

En este capítulo nos centraremos en sencillas tareas dirigidas por el DOM que resulten adecuadas a las necesidades de sitios comerciales y de contenidos. Pero la programación dirigida por el DOM, con la combinación de XML, XHTML, CSS y ECMAScript, parece preparada para moldear el futuro de la Web.

¿Dónde funciona?

Aunque con algunas diferencias, que describiremos en un apartado posterior, el DOM del W3C se admite en los siguientes navegadores:

- Netscape 6 y superior.
- Mozilla 0.9 y superior.
- Camino/Navigator 0.6 y superior.
- IE5/Windows y superior (naturalmente incluye a IE6 y superiores).
- IE5/Macintosh y superior.
- Opera 7 (y, de forma más limitada, versiones anteriores hasta Opera 4, pero dichas versiones carecen de tal cantidad de elementos que no merece la pena tenerlas en cuenta. Afortunadamente, la mayoría de los usuarios de Opera actualizan a menudo sus navegadores.)
- Kmeleon (pero le faltan algunas partes).
- Safari (basado en Kmeleon, por lo que le faltan algunas partes).
- Konqueror (con algunas omisiones; todavía no permite cambiar el valor de un nodo de texto, aunque si no sabe lo que esto significa, seguramente no trate de hacerlo, por lo que no debe preocuparse).

Desaparecidos en la interacción: entornos no basados en DOM

¿Qué falta en la lista anterior? IE4 (Macintosh y Windows). Pero prácticamente ya nadie utiliza IE4, por lo que su falta de compatibilidad con el DOM no es realmente un problema. El navegador iCab que analizamos en un capítulo anterior tampoco aparece en esta lista. Tampoco es una gran preocupación. Como los usuarios de iCab no pueden acceder a la interactividad propietaria basada en secuencias de comandos que incorpora la práctica totalidad de los sitios Web, es poco probable que se sorprendan por no poder acceder a la interactividad basada en DOM de nuestro sitio.

En la lista tampoco vemos a Netscape 4. Esto si puede ser un problema para algunos. Sin embargo, más adelante presentaremos una técnica que nos permitirá ofrecer contenido alternativo a Netscape 4 y a prácticamente cualquier navegador que responda a JavaScript pero que no comprenda el DOM.

¿Qué más falta en la lista? Los dispositivos manuales y los teléfonos Web todavía no admiten el DOM, y los navegadores de texto como Lynx nunca lo harán.

En muchos casos, puede compensar la falta de compatibilidad con el DOM de estos agentes de usuario de la misma forma que siempre ha hecho en entornos que no admitían JavaScript (porque siempre lo ha hecho, ¿verdad?)

Para admitir dispositivos incompatibles con el DOM, haga lo siguiente:

- Utilice elementos `<nonscript>` que ofrezcan acceso alternativo (un vínculo de hipertexto en lugar de un botón, por ejemplo).

- Realice pruebas en Lynx, como mencionamos en el análisis sobre accesibilidad y JavaScript de un capítulo anterior.

- Proporcione `return false` a los vínculos auténticos en lugar de falsos vínculos `javascript:` que no conducen a ninguna parte, como `<a href="javascript:" ...>`. Por ejemplo, el siguiente código se usa para activar un conmutador de estilo en `zeldman.com`. Comienza con un vínculo auténtico (`/about/switch/`) y su desencadenador `onclick` termina en `return false`:

```
<a href="/about/switch/"
onclick="setActiveStyleSheet(
'default'); return false;"
onkeypress="setActiveStyleSheet(
'default'); return false;"
accesskey="w">
<img src="/i/file.gif" width="25"
height="16" alt="Default style
(white)." title="Default style
(white)." />
</a>
```

Los navegadores que comprendan el DOM ignorarán el vínculo e invocarán la función. Los agentes de usuario no compatibles con el DOM ni con JavaScript seguirán el vínculo a la página (véase figura 15.7) que explica el conmutador y garantiza a los usuarios que no sucede nada malo. (Entre los elementos restantes del fragmento de código vemos a nuestros viejos amigos `accesskey` y `onkeypress`, que se utilizan para que la página sea más accesible, como vimos en capítulos anteriores.)

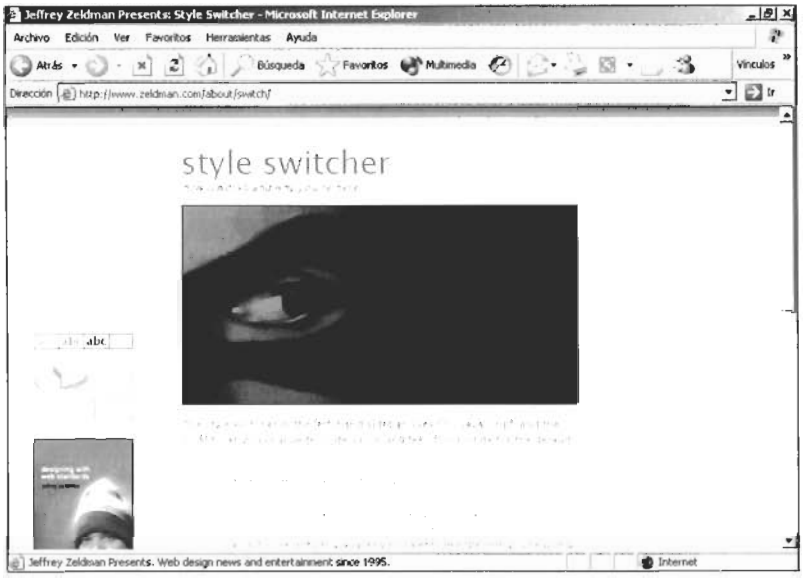


Figura 15.7.

Al iniciar un evento JavaScript con un vínculo auténtico y terminarlo con return false, el sitio puede actuar en navegadores compatibles con el DOM al tiempo que tiene en cuenta las necesidades de usuarios de dispositivos alternativos y usuarios de navegadores incompatibles con el DOM (http://www.zeldman.com/about/switch/).

La maldición de la semicuasihemicompatibilidad

Más problemático que lo que falta en la lista de agentes de usuario compatibles con el DOM es lo que ésta incluye. Por ejemplo, Opera 4, 5 y 6 creen que comprenden el DOM aunque no lo hacen, y las cadenas de agente de usuario con las que se identifican hacen que resulte imposible crear código para solucionar el problema por medio de la detección de navegadores. No somos muy partidarios de la detección de navegadores; cuando creemos que es necesario enviar a los navegadores por rutas alternativas, preferimos probar las prestaciones en lugar de cadenas de agente de usuario. Desafortunadamente, la creencia alucinatoria de Opera 5 y 6 de que comprenden el DOM también puede evitar que estos métodos funcionen. Nos pondremos manos a la obra con los detalles más adelante. Pero las versiones desfasadas se Opera no son nuestro único problema.

Los detalles del DOM

Aunque todos los navegadores modernos admiten el DOM, no todos lo hacen de la misma forma, lo que resulta un mal menor para los diseñadores que utilizan las técnicas ilustradas en este capítulo pero que se convierte en algo mucho más grave para los programadores que deben dominar el DOM para crear complejas aplicaciones basadas en la Web. Afortunadamente para éstos, en DOM Compatibility Overview de Peter-Paul Koch encontrarán detalles de dichas diferencias (http://www.xs4all.nl/~ppk/js/index.html?version5.html). Si, debido a alguna de sus numerosas actualizaciones, no encuentra la página, diríjase a http://www.xs4all.nl/~ppk/. El incansable Koch también se encarga de la lista de correo sobre DOM del W3C, entre cuyos miembros se encuentran sofisticados expertos en secuencias de comandos de todo el mundo (http://www.xs4all.nl/~ppk/js/list.html). Que aparezca W3C en el nombre no quiere

decir que sea propiedad o que dependa del W3C. Simplemente se utiliza para distinguir entre el DOM del W3C y otros, por ejemplo, para dejar claro que no se trata de una lista de correo para programadores interesados en debatir sobre el DOM de Microsoft IE 4.0/Windows.

DOM, Roba sin mirar a quién

Además de las fuentes mencionadas en el párrafo anterior, el espléndido Sr. Koch ha escrito una introducción al DOM (http://www.xs4all.nl/~ppk/js/dom1.html). Después de leerla, ya no se avergonzará al encontrar palabras y frases como nodo y estructura de árbol. Puede que siga sin saber lo que significan, pero pronto empezarán a resultarle familiares, como los nombres de ciudades del Medio Oeste a los de Nueva York.

La serie sobre DOM de David Eisenberg en A List Apart constituye una magnífica forma de acostumbrarse al DOM (http://www.alistapart.com/stories/dom/). Tras un agradable artículo de presentación y un curso práctico, Eisenberg explica cómo mostrar y ocultar componentes (http://www.alistapart.com/stories/dom2/), crear menús dinámicos (http://www.alistapart.com/stories/ domtricks2/) y actualizar el contenido de las páginas (http://www.alistapart.com/stories/domtricks3/), como se muestra en las figuras 15.4, 15.5 y 15.6.

Por favor DOM, no les hagas daño

Las secuencias de comandos basadas en DOM no son nada positivas para los navegadores con una falta total de compatibilidad con el estándar DOM del W3C. En este caso, nuestro principal problema es Netscape 4. La solución consiste en buscar compatibilidad con el DOM y, en su ausencia, proporcionar contenido alternativo o una página Web diferente.

Que en ocasiones tengamos que hacerlo es lamentable. El objetivo de los estándares es proporcionar el mismo contenido a todos los agentes de usuario. En el caso de HTML y XHTML, podemos hacer exactamente eso. Cuando se trata de CSS, como vimos en un capítulo anterior, sabemos que el método de las dos hojas nos permite enviar los mismos archivos a todos los agentes de usuario, mientras que al utilizar la directiva `@import` podemos proteger a los navegadores antiguos de estilos que no pueden procesar. Y en otros capítulos vimos cómo el modelo de cuadro puede solucionar problemas de análisis en IE5/Windows y en versiones desfasadas de Opera para proporcionar a todos las mismas hojas de estilo sin generar errores en navegadores como IE5/Windows que confunde los tamaños de fuente y el modelo de cuadro.

En todos estos casos, nos las hemos ingeniado para ofrecer a todos los mismos archivos, aunque con ocasionales parches para tapar las fugas de compatibilidad. Sin embargo, no podemos solucionar los problemas de compatibilidad con el DOM del mismo modo. Aunque IE5/Windows tergiversa el modelo de cuadro, sigue siendo un navegador CSS que puede analizar un archivo CSS. Pero Netscape 4 no es un navegador compatible con el DOM en ningún aspecto y, para ser justos con el fabricante, el W3C seguía

retocando la especificación DOM cuando apareció Netscape 4. A menos que se sea Nostradamus, no se puede admitir lo que todavía no existe.

No podemos esperar que un navegador incompatible con el DOM utilice una secuencia de comandos basada en el DOM al igual que no podemos pedirle a un pollo que utilice una calculadora. Lo que sí podemos hacer es probar el conocimiento sobre DOM y, si no existe, enviar el navegador a una página diferente (o, mejor todavía, añadir contenidos alternativos según resulte necesario por medio de SSI, ASP o cualquier otra solución intermedia).

Cómo funciona

Dori Smith (www.dori.com) del WaSP inventó el método DOM sniff al principio de 2001 como medio para implementar la campaña de actualización de navegadores de dicho grupo (que describimos en un capítulo anterior). Pero se puede usar con cualquier objetivo y, en muchos casos, como sustituto de complicadas secuencias de comando de detección de navegadores que resultan imposibles de mantener actualizadas.

El funcionamiento del método DOM sniff es muy sencillo. Una vez creada una página válida, añada la siguiente secuencia de comandos en el elemento `<head>` de su documento o en algún punto de un archivo JavaScript vinculado (.js):

```
if (!document.getElementById) {
    window.location =
        "http://www.sitio.com/
        algunapágina/"
}
```

donde `sitio.com` representa a su sitio y `/algunapágina/` a una página alternativa cuyo contenido Netscape 4 (o cualquier otro navegador no compatible con el DOM) pueda procesar. Esta página alternativa puede utilizar JavaScript para simular el comportamiento de la página estándar o puede incluir contenido sin secuencias de comandos que cualquier navegador pueda procesar.

Puede que la página alternativa ni siquiera haga eso. En su lugar, puede aconsejar al usuario que descargue Netscape 7 o un navegador compatible similar. Este enfoque de actualización está desfasado y no agradará a los usuarios de Netscape 4 que no pueden actualizarse debido a estúpidas políticas corporativas. Era la queja legítima a la campaña de actualización de navegadores del WaSP. Incluso en la actualidad, sigue siendo una queja casi legítima aunque, afortunadamente, el número de organizaciones que infligen esta reliquia incompatible con los estándares a sus empleados va decreciendo.

Sin embargo, si las aplicaciones basadas en DOM son una parte importante de nuestro diseño o incluso su razón de ser, no tiene otra opción que recomendar al visitante que vuelva a intentarlo con un navegador compatible con el DOM. En este tipo de sitios, el método DOM sniff protege a los usuarios del contenido que sus navegadores no pueden admitir al tiempo que les ofrece la posibilidad de regresar y disfrutar del contenido tras realizar la actualización.

Simplificación de sitios dinámicos: fumadores y no fumadores

Si ya sabe crear páginas dinámicas o condicionales, y si los componentes de las páginas

varían en función del agente de usuario, puede utilizar el método DOM sniff para proporcionar los correspondientes fragmentos de contenido a los navegadores incompatibles con el DOM y también puede simplificar la administración de su contenido. En lugar de enviar un tipo de documento a Internet Explorer 5 para Windows, otro para Internet Explorer 5 para Macintosh, otro para Internet Explorer 6, otra para Netscape 6 y así sucesivamente hasta el infinito, puede simplemente proporcionar dos tipos de contenido: DOM y no DOM.

Variantes de código

En un archivo .js global, el código funcionaria como hemos visto antes. Al añadir la secuencia de comandos a una página individual, la introduciría entre `<head>` y `<head/>`. En documentos XHTML Transitional 1, la secuencia de comandos sería:

```
<script type="text/javascript"
language="javascript">
<!--  //
if (!document.getElementById) {
    window.location =
        " http://www.sitio.com/
        algunapágina/"
}
    // -->
</script>
```

En documentos XHTML Strict, se podría utilizar un archivo .js global o añadir lo siguiente:

```
<script type="text/javascript">
// <![CDATA[
if (!document.getElementById) {
    window.location =
        " http://www.sitio.com/
        algunapágina/"
}
    // ]]>
</script>
```

Por qué funciona

El método DOM sniff es un binario, un conmutador, no mucho más complicado que un interruptor de encendido y apagado. Como `getElementById` es un método DOM, los navegadores compatibles lo ignoran. Por el contrario, los navegadores que no reconocen este método no son compatibles con el DOM y obedecerán la redirección, de forma que se les lleva a la página cuyo contenido pueden comprender.

Las redirecciones JavaScript deshabilitan el botón **Atrás**, lo que no es nada correcto. Pero no es necesario que siga el método DOM sniff con `window.location`. Puede seguirlo con cualquier comando que crea más indicado para los intereses de sus usuarios.

Por qué

No es necesario crear, probar y actualizar y mantener continuamente una extensa y compleja secuencia de comandos de detección de navegadores. Ya no estamos probando las cadenas de agente de usuario. Estamos probando el conocimiento de una especificación del W3C. Bastante sencillo. Bastante simple. Y también bastante barato.

Dónde falla: el no tan magnífico Opera

Como sugerimos anteriormente, el método DOM sniff falla en versiones anteriores del navegador Opera que creen que comprenden el DOM. En lugar de seguir la ruta adicional tallada para los agentes de usuario incompatibles, estas versiones antiguas de Opera se interrumpen momentáneamente por `getElementById` y se ahogan en una página que no pueden analizar correctamente.

¿Cómo podemos resolver esta ecuación? No podemos recurrir a la anticuada detección de navegadores ya que Opera, de forma predeterminada, se identifica como Internet Explorer. Si utilizamos el método DOM sniff para llevar a los usuarios de Netscape 4 a una página alternativa, lo que podemos hacer en nuestra página "estándar" es incluir una nota vinculada a dicha página alternativa e invitar a los usuarios de Opera a que la visiten. Esta solución es tan elegante como un camión Pegaso, y la comparación no es justa para los camiones Pegaso. Se entromete en la conciencia de todos los visitantes, desbarata la experiencia de los usuarios con extrañas afirmaciones de navegador y hace que nuestro sitio parezca poco profesional. No sólo eso, una nota que indique a los usuarios de Opera que hagan clic sobre la misma puede hacer que abandonen una página basada en DOM que sus navegadores sí pueden procesar.

Se preguntará cuál es la solución. Según este autor, no hacer nada. Si nuestro sitio basado en DOM falla en Opera 6, confiamos en que el usuario se percate de que puede ser la ocasión perfecta para actualizar a Opera 7 o posterior. Y basta de cosas tristes. Pongamos a prueba a nuestro prototipo para ver cómo se comporta a toda velocidad. (Nuestras metáforas sobre coches son casi tan malas como los símiles deportivos, parece.)

Cómo mostrar y ocultar

Por muchos motivos, puede que le interese ocultar parte de los contenidos de una página cuando ésta se cargue, y mostrarlos de nuevo en función de las acciones del visitante. El DOM facilita las operaciones de mostrar y ocultar contenido.

En `zeldman.com`, una barra lateral situada a la izquierda de la zona de contenido incluye enlaces a sitios de terceros. Estos enlaces pueden abrumar inicialmente al usuario, por lo que se ocultan cuando se carga la página (véase figura 15.8). Con tan sólo hacer clic en un hipervínculo se vuelven a mostrar (véase figura 15.9), en función de lo que el visitante considere oportuno. Con otro clic se vuelven a ocultar.

La técnica requiere algunos componentes. En el archivo JavaScript del sitio (http://www.zeldman.com/j/nu.js), una sencilla función que recurre a nuestro viejo amigo, `getElementById`, crea la posibilidad de activar o desactivar la representación:

```
// cambio de visibilidad
function toggle( targetId ){
   if (document.getElementById){
      target = document.getElementById(
targetId );
         if (target.style.display ==
"none"){
           target.style.display = "";
         } else {
           target.style.display = "none";
         }
   }
}
```

Un archivo de inclusión (http://www.zeldman.com/includes/outside2.html) contiene los enlaces a terceros así como el marcado y el código para cambiar su estado de representación.

En primer lugar vemos un párrafo que inicia la función de cambio, la asocia a una variable con nombre (`outside2`), proporciona un texto básico de instrucciones (`Toggle`

Externals) y la sustituye por un atributo title del vínculo de ancla:

```
<p><a href="/" onclick="toggle(
'outside2');return false;" title="Hide or
show Relevant Externals (below).">Toggle
Externals</a>.</p>
```

La lista de definición que aparece justo después de este párrafo contiene el valor del atributo title (outside2) necesario para la función e incluye una regla CSS en línea ("display: none;") que oculta la lista hasta que se modifica su estado de representación:

```
<dl id="outside2" style="display:none;">
<dt>Relevant Externals:</dt>
<dd><a href="http://www.sitio.com/"
title="Description.">Nombre del
sitio</a></dd>
etc.
```

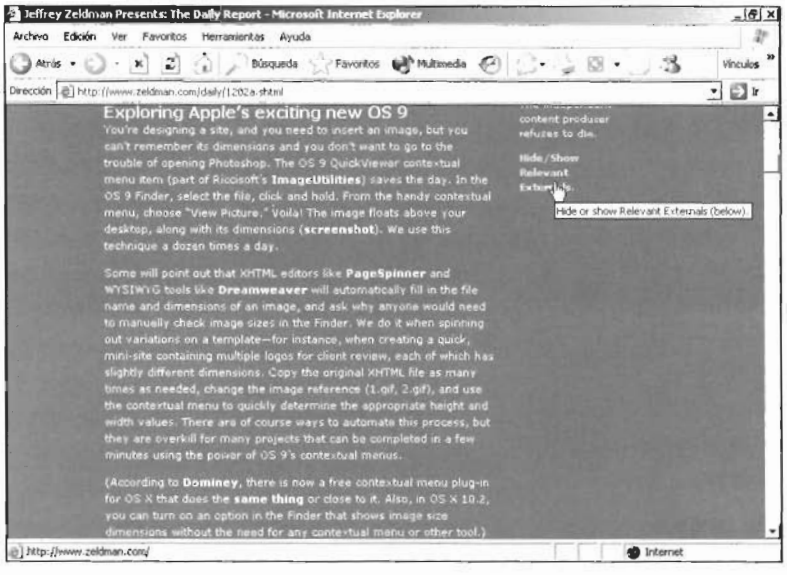


Figura 15.8.

Operaciones de ocultar y mostrar basadas en el DOM. La barra lateral de la página contiene enlaces que, inicialmente, no se ven (www.zeldman.com).

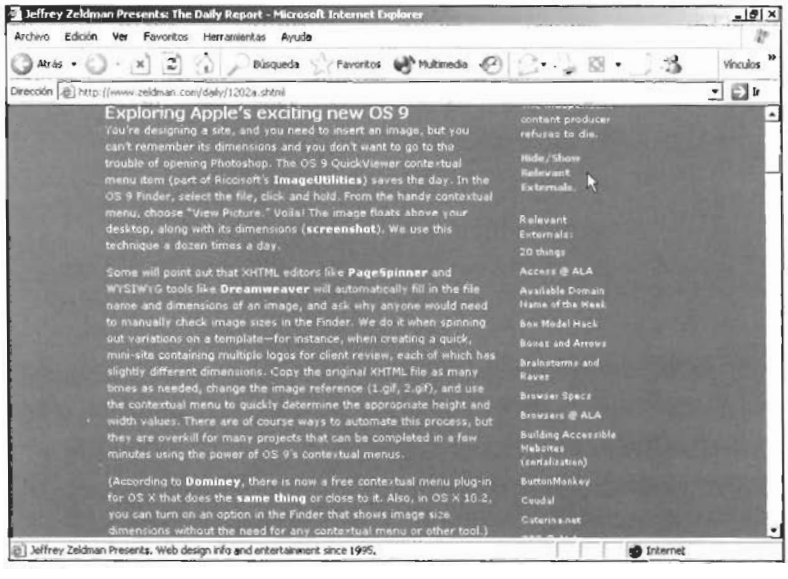


Figura 15.9.

Al pulsar en Toggle Externals, la lista aparece en pantalla.

Hemos utilizado CSS en línea (`"display: none;"`) pero no era necesario. La hoja de estilo externa del sitio podría haber incluido una regla que dijera que el contenido con el `id outside2` se desactivara de forma predeterminada y no se mostrara en pantalla. Esta regla tendría el siguiente aspecto:

```
#outside2   {
    display: none;
    }
```

Si la regla formara parte de una hoja de estilo externa, el marcado de la lista de definición sería el siguiente:

```
<dl id="outside2">
<dt>Relevant Externals:</dt>
<dd><a href="http://www.sitio.com/"
title="Description.">Nombre del
sitio</a></dd>
etc.
```

Si quiere que un elemento sea invisible pero que conserve un espacio dentro del diseño, debe utilizar `visibility: hidden`.

Combinación de mostrar/ocultar con otras técnicas

El ejemplo que acabamos de ver realiza operaciones muy básicas de ocultar y mostrar. La técnica se puede adjuntar a cualquier otra operación que desee. En la página Happy Cog Projects (véase figura 15.10), el visitante puede ver una serie de iconos, cada uno representa a un proyecto. Cuando se hace clic sobre uno de estos iconos (véase figura 15.11), se muestra el texto correspondiente a dicho proyecto. Además, el icono se ilumina como imagen cambiante.

En el principal archivo JavaScript de Happy Cog (http://www.happycog.com/j/h.js) utilizamos la misma función para mostrar y ocultar que en `zeldman.com`. El marcado de http://www.happycog.com/ es más vistoso pero conceptualmente es idéntico al que ya hemos visto. Por ejemplo, en la fila superior de iconos, utilizamos la función para cambiar la visibilidad del texto dentro de un elemento con el `id chcopy`. A continuación reproducimos el marcado y el JavaScript con una función adicional que todavía no hemos visto:

```
<a href="/projects/" onclick=
"toggle('chcopy');return false;">
<img src="/i/p/ch.gif" width="100"
height="75" border="0" alt="Charlotte
Gray." /></a>
```

Aquí está de nuevo, en esta ocasión con el JavaScript adicional, una imagen cambiante convencional en la imagen con el nombre `ch`, incluida y marcada en negrita:

```
<a href="/projects/"
onclick="toggle('chcopy');return false;"
    onmouseover="changeImages('ch',
'/i/p/cho.gif'); return true;"
    onmouseout="changeImages('ch',
'/i/p/ch.gif'); return true;">
<img name="ch" src="/i/p/ch.gif"
width="100" height="75" border="0"
alt="Charlotte Gray." /></a>
```

Más adelante en la página encontramos el contenido oculto, cuyo marcado y código debería entender sin dificultad:

```
<div id="chcopy" style="display:none">
<h2>Título de esta sección.</h2>
<p>Primer párrafo.</p>
<p>Aquí va más texto, seguido por la
posibilidad de que el lector invoque de
nuevo la función y, por lo tanto, oculte
el contenido que acaba de leer: [<a
href="/projects/"
onclick="toggle('chcopy');return
false;">Close text</a>.]</p>
</div>
```

Figura 15.10.

Las técnicas de mostrar y ocultar basadas en el DOM se pueden combinar con otros efectos (http://www.happycog.com/projects/).

Figura 15.11.

En este caso, se combinan con efectos tradicionales de imágenes cambiantes.

Se preguntará si las opciones de combinación se limitan a mostrar/ocultar con imágenes cambiantes. Por supuesto que no.

Puede combinar mostrar y ocultar con imágenes cambiantes, menús emergentes, menús desplegables y chimpancés gigantes si lo considera oportuno para sus visitantes. Y si lo considera oportuno para sus visitantes, estaríamos muy interesados en conocerlos.

Menús dinámicos (desplegables y expandibles)

Los menús DHTML tienen al menos cuatro críticas en su haber:

- Los programadores recuerdan las desmesuradas promesas de marketing, reducción del rendimiento y lamentables incompatibilidades del DHTML propietario de 1998.

- Los comités incapaces de basar la arquitectura de sitios en nada que no fuera los egos de sus miembros nos han proporcionado menús de múltiples capas que nos resultan frustrantes en lugar de guiarnos al contenido que buscamos.

- Los menús DHTML no comprobados y que fallan en nuestros navegadores han impedido que en algún momento pudiéramos utilizar un sitio.

- Los menús DHTML generados o bien WYSIWYG han resuelto con demasiada frecuencia los problemas entre diferentes navegadores y plataformas por medio de marcado no estructural y gaseosas secuencias de comandos específicas del navegador que duplican o triplican el peso de una página (con lo que aumenta el tiempo de espera para acceder a ella).

Entre métodos propietarios sin futuro, una arquitectura no orientada a los usuarios, una fase de comprobación insuficiente y su excesivo tamaño, DHTML se considera desfasado y ligeramente embarazoso. Como las animaciones GIF de Elmer, DHTML se ve como una de las molestias más inútiles de la Web y un aroma enfermizo impregna toda la categoría.

Pero el DOM corrige muchos de estos problemas ya que funciona en navegadores compatibles y utiliza marcado compacto y estructural, y CSS válidas para conseguir sus objetivos. Es uno de nuestros favoritos. En "Using Lists for DHTML menus" (http://www.gazingus.org/dhtml/?id=109) David Lindquist utiliza listas sin ordenar para crear compactos menús DHTML desplegables (véase figura 15.12) y expandibles (véanse figuras 15.13 y 15.14). Los ejemplos incluyen archivos CSS y JavaScript que puede descargar para utilizarlos o personalizarlos. (Lo sentimos, pero ni siquiera los estándares Web pueden solucionar el problema de la arquitectura de sitios basada en comités.)

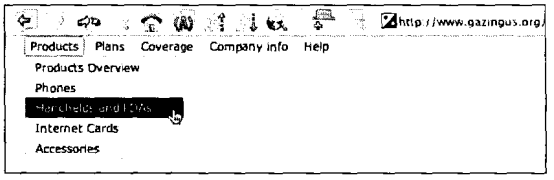


Figura 15.12.

Tiene un aspecto similar al de cualquier otro menú desplegable. Sin embargo, en lugar de código propietario y excesivo, y de marcado estructural, éste utiliza marcado de lista correcto, CSS válidas y JavaScript (http://www.gazingus.org/html/menuDropdown.html). Puede descargar los archivos de código para utilizarlos como desee.

Conmutadores de estilo: ayudan en el acceso y ofrecen opciones

En un capítulo anterior, analizamos la imposibilidad de utilizar fuentes Web sin alienar al menos a una parte de los posibles visitantes y lamentamos que, tras 13 años de evolución de la Web como medio, los píxeles sigan siendo el método más fiable para cambiar el tamaño de una fuente, y el que más problemas causa a los usuarios de IE/Windows. ¿Cómo podríamos ofrecer a nuestros visitantes distintos enfoques de fuentes para que los puedan seleccionar? ¿Cómo podríamos cambiar nuestro diseño mientras estamos en él?

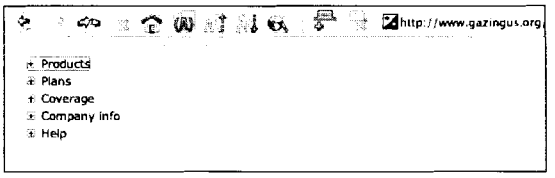


Figura 15.13.

Un menú expandible en el que se ven signos + y - sobre los que se puede pulsar (http://www.gazingus.org/html/menuExpandable2.html).

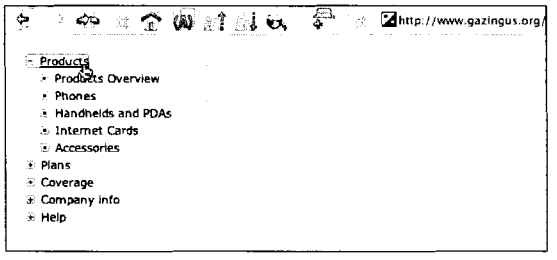


Figura 15.14.

El menú se despliega cuando el usuario hace clic. Una vez más, un marcado estructural, CSS válidas y una pizca de JavaScript se combinan a través del DOM para ofrecer algo más que la suma de las partes. Como en el caso anterior, puede descargar los archivos de código para utilizarlos como desee.

Según CSS, se puede. CSS nos permite asociar cualquier página Web no sólo a una hoja de estilo predeterminada (persistente) sino también a archivos CSS alternativos. En el interés por mejorar la accesibilidad, estas hojas de estilo alternativas pueden ofrecer una fuente de mayor tamaño o bien un esquema de color de mayor contraste (véanse figuras 15.15 y 15.16). O simplemente podrían cambiar completamente el aspecto de un sitio por motivos de lo que una vez se denominó personalización de usuarios.

El W3C recomienda que los navegadores proporcionen algún medio para que los usuarios puedan seleccionar cualquiera de estos estilos alternativos, algo que hacen los navegadores basados en Gecko como Mozilla y Netscape 7. Pero al cierre de esta edición, ningún otro navegador ofrecía esta función.

Las ventajas creativas y de accesibilidad de las hojas de estilo alternativas podrían haber sido inalcanzables para siempre. Pero en el 2001, Paul Sowden resolvió el problema aprovechando el hecho de que las hojas de estilo alternativas, al igual que cualquier otro componente de una página, son accesibles para el DOM.

En el artículo "Alternative Style: Working with Alternate Style Sheets" de A List Apart (http://www.alistapart.com/stories/alternate/), Sowden escribió un archivo JavaScript (http://www.alistapart.com/stories/alternate/ styleswitcher.js) que permitía a los visitantes del sitio cargar estilos alternativos con tan sólo hacer clic en un vínculo, un elemento de un formulario o cualquier otro componente interactivo.

Después de explicar la utilización de esta secuencia de comandos en el sitio, Sowden la lanzó en forma de código abierto. Desde entonces, miles de diseñadores y programadores lo han utilizado en sitios comerciales, públicos y privados para solucionar problemas de accesibilidad (véanse figuras 15.15 y 15.16), explorar sus efectos creativos o ambos (véanse figuras 15.17, 15.18 y 15.19). Lea el artículo de ALA, descargue el código y manos a la obra.

¿Más DOM por venir?

En este capítulo le hemos ofrecido una mínima degustación de lo que el DOM puede hacer y cómo se utiliza, a veces de forma sencilla y a veces de forma compleja, en sitios comerciales, personales y del sector público. Puede utilizar el DOM para crear completas aplicaciones Web, conseguir creativos efectos, mejorar la accesibilidad, etc. En la actualidad, no conocemos ningún libro sencillo ni ningún sitio Web sobre el tema. Sin duda se podría escribir un magnífico libro que muestre a los diseñadores cómo conseguir resultados espectaculares con el DOM y creemos que pronto aparecerá. Mientras tanto, siga mirando el cielo.

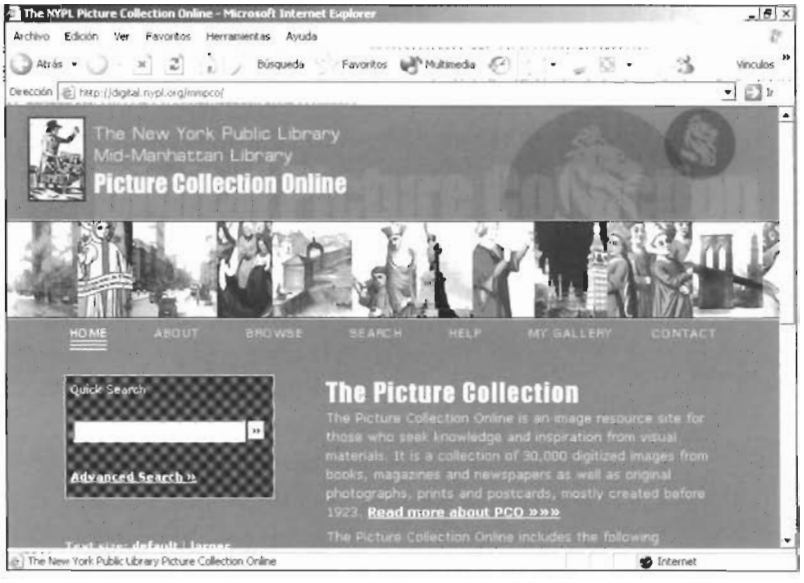


Figura 15.15.

The Picture Collection Online es uno de los muchos sitios de recursos de imágenes de The New York Public Library's (http://digital.nypl.org/mmpco/). Utiliza un conmutador de estilo para evitar posibles problemas a los usuarios con dificultades de visión.

Figura 15.16.

Cuando el usuario selecciona el tamaño larger, no sólo aumenta el tamaño de la fuente, sino que también se incrementa el contraste y se oculta el maravilloso fondo en capas (que puede resultar confuso para los usuarios con problemas de visión).

Figura 15.17.

El sitio personal de Eric Meyer utiliza un conmutador de estilo de ALA junto con otros elementos adicionales (www.meyerweb.com). En esta imagen vemos el sitio con su hoja de estilo alternativa natural. (Fíjese en el fondo fotográfico y en los efectos decorativos de la parte superior derecha.)

Figura 15.18.

En este caso, se utiliza una hoja de estilo clásica, sin fondo fotográfico ni adornos decorativos en la parte superior derecha. Se ha invertido la posición de los elementos (la navegación se ha cambiado de derecha a izquierda).

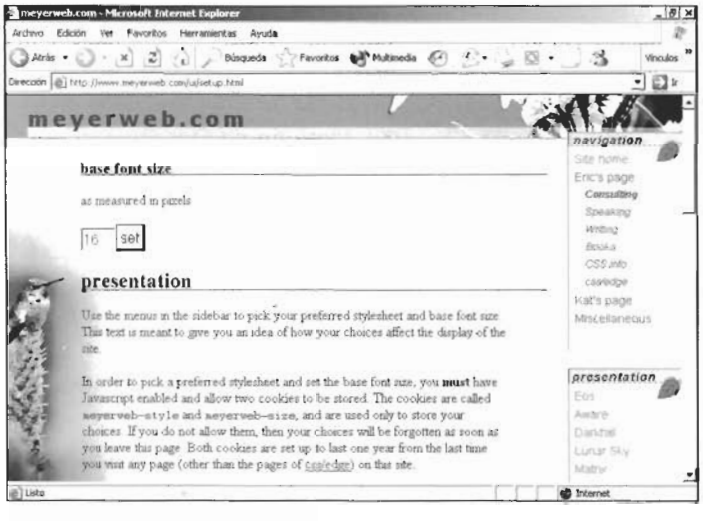


Figura 15.19.

En esta imagen, el usuario ha cambiado a la hoja de estilo natural pero ha aumentado el tamaño del texto de 11 a 16 píxeles.

Capítulo 16
Un cambio de diseño en CSS

En este capítulo, utilizaremos gran parte de lo que hemos aprendido sobre marcado, CSS, accesibilidad y el DOM, y lo pondremos en práctica en el cambio de diseño de un sitio para separar estructura de presentación. Definiremos y cumpliremos objetivos adecuados para los usuarios del sitio y, quizás, para los del suyo propio. Una vez satisfechos con el aspecto operativo y visual del sitio, crearemos un diseño alternativo basado en CSS y el DOM que el usuario pueda seleccionar.

Aunque el diseño visual del sitio que mostraremos en este capítulo no tiene nada de especial, los métodos de construcción son importantes, ya que son las técnicas que utilizaremos para crear muchos más sitios y, con el tiempo, la totalidad de los sitios que diseñemos.

En capítulos anteriores hemos analizado las piezas del puzzle de los estándares y hemos aprendido a crear sitios que combinan métodos clásicos y modernos. En este capítulo le acercaremos las prácticas de diseño de la Web del futuro.

Describiremos las técnicas utilizadas para crear diseños CSS, desde los fundamentos de la creación de las secciones de una página hasta las ideas menos básicas, como la creación de botones de menú a partir de listas estructuradas y la generación de efectos de imagen cambiante similares a JavaScript sin JavaScript e incluso sin utilizar el elemento XHTML `img`. Además de abarcar técnicas específicas, nos centraremos en formas de pensar, como por ejemplo en el diseño basado en reglas frente al diseño basado en cuadrículas.

Definición de objetivos

En este proyecto de cambio de diseño, realizado específicamente para este libro, modificaremos el aspecto operativo y visual, la estructura y la funcionalidad básica de `zeldman.com`, el sitio personal del autor. Ya capte la atención de miles de lectores o simplemente le atraiga a usted y a algún familiar, un sitio personal tiene un valor incalculable, sobre todo a la hora de experimentar con CSS y otros estándares que puede que no esté preparado para utilizar en diseños para clientes.

Un sitio personal también le puede ayudar a darse cuenta de que no está solo como diseñador. Al crear este tipo de sitios, formará parte de una floreciente comunidad de diseñadores y programadores orientados a los estándares que, sorprendentemente, se muestran deseosos de ayudarse unos a otros.

En muchas ocasiones, bloqueados por un problema de CSS, de marcado o de JavaScript, hemos compartido nuestro desconcierto en `zeldman.com`. Tras ello, los lectores envían sus mensajes de correo electrónico con la posible solución a la pregunta planteada en el sitio. Del mismo modo, al leer los sitios de otros usuarios, muchas veces hemos sentido la necesidad de enviarles una sugerencia sobre los problemas que los confunden.

No somos los únicos. Le pasa a cualquiera que escriba sobre problemas de diseño Web o programación en su sitio personal. Pruébelo, le gustará.

Carácter de marca

Creada en mayo de 1995, `zeldman.com` ofrece cursos prácticos, noticias y detalles sobre diseño Web así como cualquier otro tema de nuestro interés. El carácter de marca es irreverente, apasionado y curioso; el diseño es limpio, minimalista y caprichosamente elegante. Nuestro cambio de diseño debe conservar estos atributos.

Los diez principales objetivos

Los requisitos adicionales del cambio de diseño son menos específicos con respecto a la marca. Puede que sean tan relevantes para su sitio como para el nuestro. A continuación enumeramos los diez objetivos principales de este proyecto:

1. El sitio debe ser tan fácil de utilizar en entornos no gráficos como lo es en los mejores y últimos navegadores de Netscape, Microsoft, Opera y otros. Su contenido y sus funciones básicas deben estar disponibles para todos los navegadores y dispositivos, y su diseño debe funcionar en cualquier navegador razonablemente compatible con CSS.

2. El marcado debe validarse en función de la especificación XHTML 1.0 Transitional y debe evitarse el uso de elementos de presentación. Separaremos la estructura de la presentación, lo que significa el rechazo a `span` y `div` no semánticos cuando se pueda utilizar `p` o `h1`. No se utilizarán imágenes GIF espaciadoras ni tablas, sino es para presentar datos tabulares. Tampoco se utilizarán atributos desfasados o inválidos como `bgcolor` o

`marginheight`; no se empleará equivocadamente `<blockquote>` para conseguir formato y no se recurrirá a etiquetas `<br />` cuando un elemento estructural pueda crear el efecto visual deseado y, a la vez, transmitir significado. (Si estos objetivos le parecen confusos, consulte un capítulo anterior.)

3. Las CSS deben validarse, deben ser todo lo compactas posible y deben disponerse de forma lógica (consulte los capítulos anteriores).

4. Para conseguir nuestro objetivo de proporcionar contenidos y funciones básicas en cualquier entorno de navegación imaginable, el sitio debe ser lo más compacto posible. Con demasiada frecuencia, falla el deseo de conseguir la accesibilidad y se convierte en una simple cadena de palabras vacías. Para evitar este destino, probaremos nuestro trabajo con un estándar de accesibilidad aceptado. Hemos seleccionado U.S. Section 508 (consulte un capítulo anterior) pero se puede utilizar cualquier otro estándar. Lo importante es seleccionar un conjunto de directrices y adherirse a las mismas.

5. El sitio debe proporcionar un aspecto operativo y visual reconocible sin dilapidar ancho de banda para el visitante o el servidor con marcado excesivo, complejas secuencias de comandos o imágenes innecesarias. El sitio no debe malgastar recursos sino poseer estilo. Tampoco es aconsejable que intimide al visitante (los excesos no se corresponden con el carácter de marca). Debe convertirse en un lugar familiar, en el que se reconozca la evolución del mismo y no un cambio radical con respecto a versiones anteriores.

6. El sitio debe ofrecer interactividad visual para que parezca algo vivo.

7. Siempre que sea posible, se deben ofrecer equivalentes de los elementos dinámicos para favorecer a los usuarios de navegadores de texto o de otros dispositivos no tradicionales. (Es otra forma de decir que no utilice la interactividad como excusa de una pobre accesibilidad.)

8. El sitio debe proporcionar opciones de personalización de usuario sin perder el carácter de marca en el proceso.

9. El texto debe ser sencillo y agradable de leer, y debe convertirse en la parte principal del sitio, que es la lectura, y no las compras, los vínculos o los botones. El procesamiento cuidadoso del texto es importante para cualquier sitio, pero resulta fundamental para los que se leen como un periódico diario.

10. La navegación debe ser clara, intuitiva y obvia. Puede, y debe, disponer de cierto garbo visual, pero también debe funcionar en entornos no visuales. Los visitantes que accedan al sitio de forma lineal (por ejemplo a través de lectores de pantalla) deben disponer de la posibilidad de ignorar este aspecto. Si la navegación se puede conseguir con elementos estructurales, como por ejemplo con componentes de una lista sin ordenar, mucho mejor.

Existen objetivos adicionales, pero con éstos bastará.

Estos requisitos deben incorporarse a cualquier práctica de diseño Web y a la mayoría de los proyectos de diseño de sitios. Deben considerarse algo habitual y pasar inadvertidos. No hace falta decir nada sobre la mayoría de los requisitos, al igual que no es necesario decir que los productos de una tienda de alimentación deben ser comestibles y estar protegidos de las enfermedades, que un automóvil debe ofrecer movimiento controlado o que la ropa debe protegernos de las inclemencias del tiempo y de las miradas de los extraños. Pero debido a la extraña forma en que ha evolucionado la industria, estos objetivos están lejos de llegar a ser habituales y es necesario mencionarlos específicamente.

Método y locura

Al finalizar este capítulo, habremos creado un diseño CSS predeterminado (véase figura 16.1) y una versión alternativa (véase figura 16.2) que el usuario pueda seleccionar con tan sólo hacer clic en la interfaz de usuario. Para diseñar la versión alternativa basta con guardar el archivo CSS original con un nuevo nombre y modificar sus valores. Los colores cambian, al igual que algunas fuentes, y en la parte inferior de la pantalla se añade una imagen que no se desplaza, pero resultará muy sencillo crear el estilo alternativo. Se preguntará cuántos estilos alternativos pueden utilizarse en un mismo sitio. Todos los que quiera.

Ambos diseños parecen simples, y lo son, pero necesitaríamos un libro entero para describir los razonamientos subyacentes a todas y cada una de las reglas utilizadas en las dos hojas de estilo. Nuestro análisis se centrará en los puntos principales (y otros que resulten de interés) en lugar de enumerar y describir tediosamente todas las reglas CSS, todo el XHTML o todos los detalles de accesibilidad. Además, esta enumeración sería de algún modo redundante ya que hemos dedicado los últimos capítulos a las técnicas CSS, XHTML y de mejora de accesibilidad utilizadas en este cambio de diseño CSS.

Un pequeño desorden y su evolución

La revisión operativa y visual que describiremos en este capítulo evolucionó durante varios días y recurrió a ciertos componentes de una versión CSS anterior que también había evolucionado con el tiempo y que, ocasionalmente, cambiaba en respuesta a las acciones del usuario. No era el estilo de administración de proyectos habitual.

Al diseñar sitios comerciales, siempre se analiza el carácter de marca, se imaginan situaciones con los usuarios para basar las decisiones sobre navegación y arquitectura, y se colabora con diferentes especialistas para determinar el flujo del sitio y los requisitos técnicos y empresariales. Tras ello, durante semanas se diseña y se retoca en Photoshop, Illustrator o FreeHand.

Las revisiones siguen a las presentaciones. Las fechas se cumplen o se incumplen, los objetivos aumentan o se reducen, las lágrimas caen y se escuchan enérgicos exabruptos de un sitio a otro (bueno, para ser más precisos, de un ordenador a otro).

Tras ello viene el código, las pruebas, más código y más pruebas en servidores provisionales, hasta que esperamos que los errores más evidentes se hayan detectado y corregido. Meses después, el proyecto hace su aparición pública.

Figura 16.1.

En este ejemplo vemos la página inicial del sitio y su aspecto operativo y visual predeterminado (blanco).

Figura 16.2.

De nuevo el mismo sitio, con un aspecto operativo y visual alternativo, con un color anaranjado que puede parecer ligeramente oscuro en las páginas de este libro en blanco y negro pero que se ve magnífico en la Web.

Evolución del todo a partir de las partes

Esta metodología que acabamos de mencionar es típica de los proyectos de diseño de sitios comerciales y dirigidos a usuarios. Pero en este proyecto, no lo utilizaremos. En su lugar, sin un plan general y sin componentes que utilizar como guía, empezamos a bosquejar componentes individuales del sitio, con un diseño de nivel modular y no global. (En primer lugar, diseñe el pomo de la puerta, lo que le llevará a diseñar la puerta. Con el paso del tiempo, sin saber cómo lo ha conseguido, habrá diseñado una casa. Hemos tomado la idea de un diseño modular del experto en interfaces de usuario Jeffrey Ven, que siempre ha sido un defensor de la misma.)

Tomamos una decisión inicial: el diseño estaría formado por tres zonas principales. Después de llegar a esta conclusión, no necesitábamos un plan maestro.

En lugar del todo, nos centramos en las partes. ¿Qué aspecto tendría el dispositivo de navegación? ¿Cómo se ubicarían los párrafos en relación a los subtítulos? Si incluíamos vínculos a sitios de terceros, ¿deberían verse al cargarse el sitio o el visitante tendría la opción de activarlos y desactivarlos?

La mayor parte del diseño se creó mediante código, no con Photoshop. Y lo hicimos en público, cambiando el sitio a trompicones mientras los lectores miraban. Algunos pensaron que nos habíamos vuelto locos. Otros disfrutaban espiando por encima del hombro. Algunos mostraban sus quejas sobre algún aspecto, que aprovechábamos para obtener nuevas ideas. De esta forma descubrimos, en lugar de planificar y ejecutar, mucho de lo que analizaremos en este capítulo. En cierto sentido, todos los diseños funcionan de esta forma, pero en nuestro caso se oculta a usuarios y clientes, que solamente ven los resultados, nunca el esfuerzo. Para intentar conseguir un capítulo coherente, fingiremos que siempre supimos lo que estábamos haciendo y omitiremos los pasos intermedios y las ideas equivocadas que pudieran surgir.

Visite www.zeldman.com para asegurarse de que puede ver las últimas versiones correctas del marcado, las CSS y el código. En http://www.zeldman.com/c/wh.css y http://www.zeldman.com/c/ or.css encontrará las hojas de estilo. Al igual que cualquier otro sitio vivo, `zeldman.com` es una obra continuamente en proceso y puede que cuando lea este libro algunos aspectos del diseño hayan cambiado.

Definición de los parámetros básicos

Decidimos que nuestra plantilla tendría tres zonas principales (véase figura 16.3):

- Una barra en la parte superior de la página que actuaría como botón Inicio y que también podría incluir la navegación principal. Denominaremos a esta zona "nuevo menú".
- Una barra lateral en la izquierda que contendría la navegación secundaria, las

opciones de búsqueda, de interfaz de usuario, anuncios publicitarios y vínculos. Una etiqueta adecuada para esta zona sería "navegación secundaria".

- En la zona de contenido principal, como habrá adivinado, se incluirá el contenido principal del sitio (que esperemos que sea más interesante que esta frase). Seamos creativos y denominemos a esta zona "contenido principal".

En nuestra página XHTML, después del DOCTYPE, el espacio de nombres y los metadatos, nuestros primeros esfuerzos serán los siguientes:

```
<div id="nuevomenú"></div>
<div id="navegaciónsecundaria"></div>
<div id="contenidoprincipal"></div>
```

Cada una de las tres zonas principales fluirá con el contenido. Pero primero es necesario aplicar estilo a cada una de ellas. En nuestra hoja de estilo predeterminada (blanca), definimos parámetros de estilo generales y los agrupamos en la parte superior. Comenzaremos con la zona de nuevo menú, que se puede ver en la parte superior de la figura 16.3 y que discurre de izquierda a derecha:

```
    /* Definición de los parámetros
       generales de diseño */
#nuevomenú   {
    background: #e0861e;
    color: #000;
    border: 0;
    border-bottom: 1px dotted #000;
    margin: 0;
    padding: 0;
    text-align: left;
    height: 31px;
    }
```

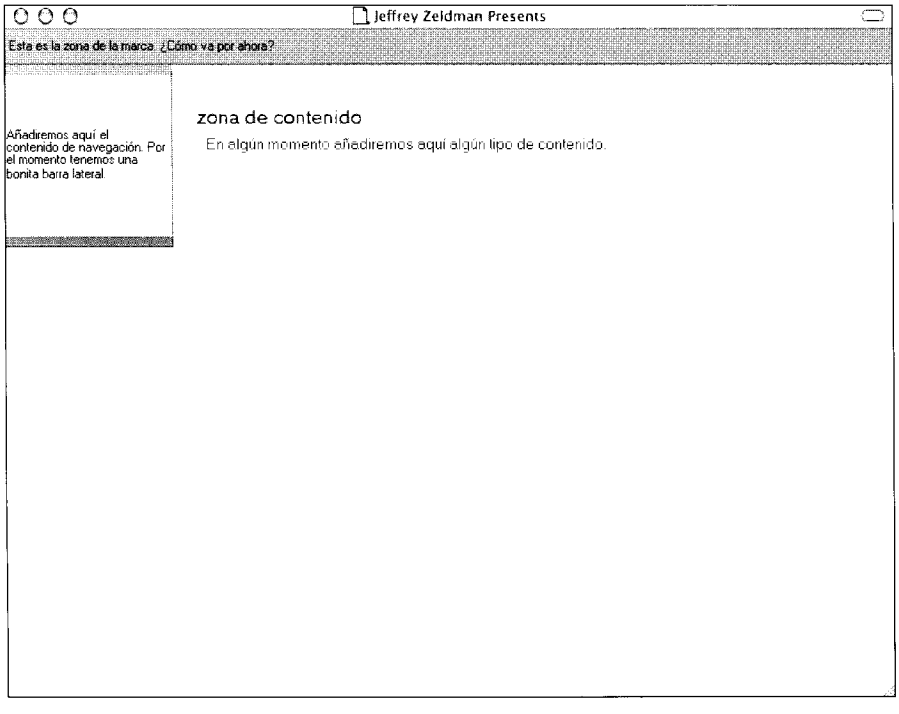


Figura 16.3.

La plantilla se dividirá en tres zonas principales.

Después de haber leído los capítulos anteriores y de empollarse el modelo de cuadro, sabrá cómo interpretar estas reglas, pero las resumiremos. La zona tiene una altura de 3 píxeles, el fondo tiene un color naranja no seguro con la Web (`#e0861e`), su contenido está deliberadamente alineado a la izquierda para evitar un error de algunas versiones de Internet Explorer y en la parte inferior se utilizan efectos de bordes con puntos. No es necesario compensar las incompatibilidades con el modelo de cuadro ya que los márgenes, el relleno y los bordes se han definido con el valor `0`. El borde de puntos de 1 píxel en la parte inferior puede que se solape 1 píxel en la zona inmediatamente superior en IE5/Windows debido a la confusión de este navegador con la forma en que interactúan los márgenes, el relleno y los bordes entre sí en el modelo de cuadro CSS. Pero nadie se dará cuenta, por lo que no debe preocuparse.

Instalación de la barra lateral

A continuación, aparece la zona de navegación secundaria o barra lateral (véase figura 16.4):

```
#navegaciónsecundaria   {
    /* float: left; */
    position: absolute;
    left: 0;
    margin: 0;
    padding: 0 25px 25px 25px;
    background: #bdcdbd url(/i/2003/
    sbbot4.gif) repeat-x bottom left;
    border: 0;
    border-top: 10px solid #95a580;
    border-right: 1px dotted #000;
    border-bottom: 1px dotted #000;
    width: 151px;  /* Valor falso para
                      IE4-5.x/Win */
    voice-family: "\"}\"";
    voice-family:inherit;
    width: 100px;    /* Valor real para
         los navegadores compatibles */
    }
html>#navegaciónsecundaria {
    width: 100px; /* Sea amable con
                     Opera */
    }
```

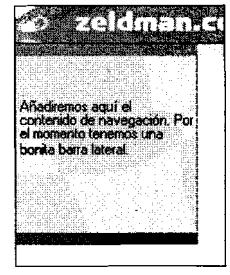


Figura 16.4.

¡Mírame! ¡Soy una barra lateral! La tira de color de la parte superior se ha creado con una propiedad de borde CSS. La tira más oscura de la parte inferior es una imagen GIF creada en Photoshop.

Hay mucho en lo que fijarse, sobre todo en una página impresa, pero ya está familiarizado con todo ello. La anchura total de la zona es de 151 píxeles (100 de contenido, con márgenes izquierdo y derecho de 25 píxeles, y un píxel adicional para el borde de puntos de la derecha). Hemos utilizado el modelo de cuadro que describimos en un capítulo anterior para proporcionar a IE5/Windows valores de su agrado y una regla de tipo Sea amable con Opera para solventar uno de los errores de análisis de este navegador. Pero suceden muchas cosas más, por lo que analizaremos alguna de ellas con más detalle, comenzando con la parte de posicionamiento.

La parte de posicionamiento

La parte de posicionamiento de la regla es la siguiente:

```
#navegaciónsecundaria    {
    /* float: left; */
    position: absolute;
    left: 0;
```

Las líneas que se encargan del posicionamiento se muestran en negrita; `/* float: left; */` es un simple comentario de lo que consiguen estas dos líneas de CSS o, si lo prefiere, lo que emulan. (Los comentarios CSS se escriben de la misma forma que en el lenguaje de programación C). La primera regla desplaza a la barra lateral del flujo normal del documento (`position: absolute;`) y la segunda, (`left: 0;`), la ubica hacia la izquierda, por debajo de la zona de menú.

Se preguntará si se podría haber hecho de otra forma. Evidentemente sí. De hecho, en una versión anterior, lo hicimos de esta otra forma, en la que el punto de diferencia se resalta en negrita:

```
#navegaciónsecundaria    {
    float: left;
    margin: 0;
    padding: 0 25px 25px 25px;
    background: #bdcdbd url(/i/2003/
    sbbot4.gif) repeat-x bottom left;
    border: 0;
    border-top: 10px solid #95a580;
    border-right: 1px dotted #000;
    border-bottom: 1px dotted #000;
    width: 151px;  /* Valor falso para
                      IE4-5.x/Win */
    voice-family: "\"}\"";
    voice-family:inherit;
    width: 100px;     /* Valor real para
                        los navegadores
                        compatibles con
                        CSS */
    }
```

Fíjese en que a la zona se le aplica `float: left`, lo que hace que se desplace hacia la izquierda. ¿Qué esperaba, algo de Shakespeare? Veamos la razón por la que hemos optado por posicionamiento absoluto en lugar de `float`.

El problema de float, parte 999

Desafortunadamente, como vimos en un capítulo anterior, algunos conocidos navegadores tienen problemas para procesar `float`, incluyendo la imposibilidad de realizar este desplazamiento flotante o la tendencia a calcular equivocadamente la altura de las zonas flotadas cuando el visitante pasa de una página a otra, lo que recorta partes y fragmentos del contenido.

Este último problema se puede compensar por medio de JavaScript, pero si lo hace puede adentrarse en un infierno de escalofriantes errores de navegador que dejaremos que otra persona describa en otro libro.

Además, al utilizar `float` en una zona, se arriesga a dividir el diseño si el monitor del usuario se ha configurado con un tamaño menor a la anchura total de la página. El uso de posicionamiento absoluto para emular `float` parece resolver el problema en la mayoría de los casos.

Los diseños CSS se basan en `float` y la compatibilidad incompleta con `float` es un verdadero problema. Es como si un coche funcionara perfectamente excepto al maniobrar con el mismo. Pero para cada problema hay una solución y la emulación de `float` por medio del posicionamiento absoluto fue la nuestra. (De hecho, Drew McLellan de `DreamweaverFever.com` y la Dreamweaver Task Force de The Web Standards Project recomendó la solución.)

CONVENCIONES DE NOMBRES DE ARCHIVO

En nuestro esfuerzo por ahorrar ancho de banda, intentamos utilizar nombres breves en lugar de directorios: `/i/` en lugar de `/images/`, `/j/` en lugar de `/javascript/` y `/c/` en lugar de `/css/` o `/styles/`.

Aplicamos el mismo enfoque a los nombres de archivo. Por ejemplo, la imagen utilizada para completar la parte inferior de la barra lateral se denomina `sbbot.gif`, en lugar de `sidebar_bottom.gif`. Cuantos menos caracteres utilicemos, menos bytes se malgastan. No empleamos esta práctica en todos los casos, pero lo intentamos.

Del mismo modo, en lugar de utilizar `logo_banner.gif` para el estado predeterminado de una imagen de logotipo y `logo_banner_over.gif` para el estado "sobre" de dicha imagen, utilizamos `lb.gif` y `lbo.gif` respectivamente. Cuando aparecen cifras en los nombres de los archivos, éstas indican el número de la versión; por ejemplo, `lbo2.gif` es un segundo borrador (La o de `lbo2.gif` es una letra, no un cero.)

Creación de barras de color

Con las reglas que permiten crear la barra de lateral en mente, fíjese en que hemos pintado un color de fondo gris verdoso (`#bdcdbd`) y que hemos situado un bloque de color sólido `#95a580` de 10 píxeles en la parte superior por medio de una propiedad CSS `border-top`, que se puede utilizar para crear un bloque de color del tamaño que deseemos:

```
border-top: 10px solid #95a580;
```

Queremos obtener un bloque sólido de 10 píxeles que coincida en la parte inferior de la zona, pero ya hemos utilizado nuestra propiedad `border-bottom` para crear el efecto de borde de puntos de 1 píxel. No se pueden utilizar dos propiedades de borde de fondo. (Existe una solución CSS para este problema pero, que sepamos, sólo funciona en IE5/Macintosh.)

Pero podemos hacer otra cosa. Podemos crear un rectángulo coloreado con el tamaño correcto en Photoshop. Podemos denominarlo `sbbot` (parte inferior de la barra lateral), guardar una copia GIF (`sbbot4.gif`) y utilizarla como fondo que se ubicará en la parte inferior izquierda y que sólo se repetirá horizontalmente (`repeat-x`), no verticalmente (`repeat-y`). Y eso, queridos amigos, es lo que hemos hecho:

```
background: #bdcdbd url(/i/2003/
sbbot4.gif) repeat-x bottom left;
```

Disponemos de una barra lateral con el color y el tamaño correctos, con un borde de puntos y barras de color de 10 píxeles de altura en la parte inferior y superior (véase figura 16.4). Hemos explotado al máximo la figura 16.4, pero explotar al máximo un solo componente es parte de un buen diseño.

El problema de la coincidencia de color

Parece que resulta imposible que la barra de color inferior, la que hemos creado en

Photoshop, coincida con la barra superior creada con la propiedad CSS de borde, aunque copiemos y peguemos el color del borde superior (`#95a580`) directamente en el selector de colores de Photoshop. Gracias a las tecnologías conflictivas, la coincidencia de colores no se puede conseguir en la Web.

A continuación le indicamos una serie de problemas que lo impiden:

- Todos los navegadores ajustan sus colores antes de representar una página. Puede que haya calibrado su monitor. Puede que incluso lo haya configurado como sRGB (http://www.w3.org/Graphics/Color/sRGB.html), un espacio de color predeterminado estándar para Internet. Pero los navegadores no son lo suficientemente inteligentes para saber si lo ha hecho o no. Modificarán los colores de todas formas. Y cada navegador los cambiará de forma diferente.

- Los sistemas de 16 bits no pueden reproducir con precisión ningún color que no sea blanco o negro. En estos sistemas, los navegadores calculan incorrectamente los colores CSS en una dirección y los colores de las imágenes en la dirección contraria, como explican David Lehn y Hadley Stern (http://hotwired.lycos.com/webmonkey/00/37/index2a.html?tw=design).

Optamos por utilizar el dilema de la coincidencia de colores como una oportunidad creativa y seleccionamos un color diferente para la barra inferior, un color más oscuro que tuviera un mayor significado de "parte inferior".

Un espacio para el contenido

La creación de la zona de contenido principal (el espacio en blanco de la figura 16.3) es bastante sencilla:

```
#contenidoprincipal    {
    border: 0;
    border-top: 10px solid #bdcdbd;
    padding: 0;
    margin: 0;
    margin-left: 150px;
    width: auto;
    }
```

De nuevo hemos utilizado CSS para crear una banda de color de 10 píxeles de altura (`border-top: 10px solid #bdcdbd;`). A pesar de la falta de un plan, el sitio empieza a contar con elementos de diseño consistentes. También hemos desactivado todos los bordes restantes, hemos definido un margen izquierdo que indica a la zona de contenido que comience donde termina la barra lateral (`margin-left: 150px;`) y hemos desactivado todos los márgenes restantes y el relleno. Y hemos indicado a nuestra zona de contenido principal que ocupe todo el espacio horizontal disponible, desde su borde izquierdo hasta el borde de la ventana del navegador, por medio de una sencilla regla (`width: auto`).

Un cuadro dentro de otro cuadro

Si quisiéramos texto de una esquina a otra que refluyera para ocupar la anchura de cualquier navegador, habríamos terminado con la definición de los parámetros básicos y podríamos sacar las bebidas y los aperitivos. Pero no es lo que queremos (queremos bebidas y aperitivos, pero no texto de una esquina a la otra) y tampoco nuestros lectores. Después de todo, un texto legible y fácil de

leer es el elemento número 9 de nuestra lista de objetivos del cambio de diseño.

Una forma para que el texto resulte más sencillo de leer, como cualquier diseñador de libros o revistas sabe, es definirlo en una columna que no sea ni demasiado ancha ni demasiado estrecha. Nos parece que 400 píxeles es la anchura correcta para la columna de texto de este diseño. Y es lo suficientemente estrecha para que, al combinar su anchura con la de la barra lateral, encaje perfectamente en todos los monitores, excepto en los más pequeños.

Pensará que basta con modificar `primarycontent` y ajustarlo a una anchura de 400 píxeles, pero no es lo que haremos. Al hacerlo, algunos navegadores desplazan todo el contenido completamente hacia la derecha de la página, mientras que otros lo sitúan por delante de la barra lateral. Tendremos que crear un contenedor de contenido de 400 píxeles de ancho y situarlo dentro de la zona de contenido principal. La regla CSS de dicho contenedor sería la siguiente:

```
#bravefourhundred    {
    margin: 0;
    border: 0;
    padding: 15px 25px;
    width: 450px; /* Valor falso para
                     IE4-5.x/Win */
    voice-family: "\"}\"";
    voice-family:inherit;
    width: 400px; /* Valor real para
                     los navegadores
                     compatibles */
    }
html>#bravefourhundred {
    width: 400px;  /* Sea amable con
                      Opera */
    }
```

Como sabe la forma de interpretar estas reglas, habrá reconocido el valor real (400 píxeles), el método Tantek para engañar a las versiones antiguas de IE para que muestren el valor correcto y la regla Sea amable con Opera. Pasemos al marcado, en el que nuestro valiente contenedor de contenido "cuatrocientos" se añade de esta forma:

```
<div id="primarycontent">
    <div id="bravefourhundred">
    Aquí va el contenido.
    </div>
</div>
```

Una pausa para descansar, (¿dónde dejamos esos aperitivos?).

Diseño basado en reglas

Ya hemos creado los cimientos en CSS y en el marcado para un diseño que funcionará en cualquier navegador que se haya comercializado en este siglo y con un contenido que funcionará en cualquier agente de usuario.

Cuando empecemos a completar nuestra plantilla con texto e imágenes de ejemplo, añadiremos reglas que controlen la presentación de los elementos de marcado seleccionados y que también controlen la forma en que interactúan entre sí. Aunque nuestra actividad parece sencilla y humilde, y lo es, lo que realmente haremos es evolucionar de unos predicados dirigidos por componentes a un tipo de diseño similar a la Web. En definitiva, nos acercaremos a un diseño basado en reglas.

No describiremos todas y cada una de las reglas de la hoja de estilo (como mencionamos anteriormente, puede encontrarlas en

http://www.zeldman.com/c/wh.css), sino que analizaremos una de ellas para introducir la explicación del diseño basado en reglas:

```
p {
    margin-top: 0;
    margin-bottom: 1em;
    font: 11px/1.5 Verdana, Trebuchet,
          Lucida, Arial, sans-serif;
    }
```

En este caso, los elementos principales son un margen superior 0 y un margen inferior de 1 eme (es decir, un retorno del carro basado en el tamaño del texto). Si no hubiéramos especificado este valor de margen inferior, los navegadores tendrían que adivinar nuestras intenciones y puede que no dejaran espacio vertical entre los párrafos.

Pero el objetivo no es simplemente evitar comportamientos inesperados en los navegadores Web. El objetivo es definir reglas que controlen el aspecto operativo y visual en el nivel modular. El diseño basado en reglas es un técnica para crear diseños modulares. Al definir un margen superior con el valor 0, permitimos que los títulos secundarios se ubiquen por encima de los párrafos (sobre todo si definimos valores 0 de margen inferior para los títulos secundarios, lo que haremos en una regla posterior). Al definir un margen inferior de 1 eme, diferenciamos el espacio entre párrafos del espacio entre títulos secundarios y párrafos.

Es un aspecto menor pero el diseño CSS está repleto de estas pequeñas decisiones, que marcan la diferencia entre un diseño agradable y controlado, y diseños cuya calidad aleatoria significa una falta de cuidado. Lo mejor de todo es que esta sensación de control se puede conseguir sin necesidad de recurrir a complicados marcados de presentación y sus costes en ancho de banda asociados.

La definición de muchas de estas reglas también nos evita tener que inundar nuestro marcado con atributos `class`. Por ejemplo, la humilde regla de párrafo que mostramos anteriormente nos evita tener que escribir algo como esto:

```
<p class="notopmargin">
```

De estos pequeños pasos se consiguen grandes ahorros en ancho de banda. Bonita frase, ¿verdad? Al definir reglas para los distintos elementos de la página, incluimos instrucciones sobre cómo interactúan entre sí dichos elementos. Por ejemplo, podríamos crear una regla que no sólo adjunte un borde negro de 1 píxel alrededor de las imágenes sino que también añada 10 píxeles de espacio en blanco por debajo de las mismas. Por medio de selectores `id`, podemos establecer este tipo de borde y espaciado para las imágenes que aparezcan en la zona A de nuestro diseño y aplicar un tratamiento de borde y espaciado completamente diferente en las de la zona B. CSS incluso nos permite variar la posición de un elemento en función de los elementos que le precedan:

```
p+p  {
     text-indent: 2em;
     margin-top: -1em;
     }
```

Esta regla indica que si un párrafo está precedido de otro párrafo (`p+p`), la primera línea del segundo párrafo se sangrará 2 eme y que se suspenderán las reglas de margen inferior habituales, para simular el estilo de

párrafo que suele emplearse en un libro (véase figura 16.5).

Podemos crear reglas para cualquier situación:

```
img+h3    {
      margin-top: 15px;
      }
```

Esta regla indica a los navegadores que dejen 15 píxeles de espacio en blanco sobre cualquier título `h3` que esté precedido por una imagen como, por ejemplo, un título `h3` cambiante situado directamente por debajo de una fotografía en la parte superior de un sitio de noticias. No todos los navegadores admiten este tipo de selector CSS pero los que no lo hacen, simplemente lo ignoran, por lo que no debe preocuparse. La gran mayoría de los diseños basados en reglas se pueden conseguir sin recurrir a selectores CSS2 que algunos navegadores no admiten todavía. El diseño basado en reglas es la suma de multitud de pequeñas reglas relacionadas entre sí.

A continuación veremos algunos aspectos más del cambio de diseño.

Un botón Inicio con efectos CSS de imagen cambiante

Anteriormente definimos una zona de "nuevo menú" de 31 píxeles de altura. Después de probar la navegación (que describiremos más adelante), descubrimos que se podría incluir en la barra lateral. ¿Qué se podía hacer entonces con la zona de "nuevo menú"? La utilizaremos para incluir un botón Inicio (véase figura 16.6).

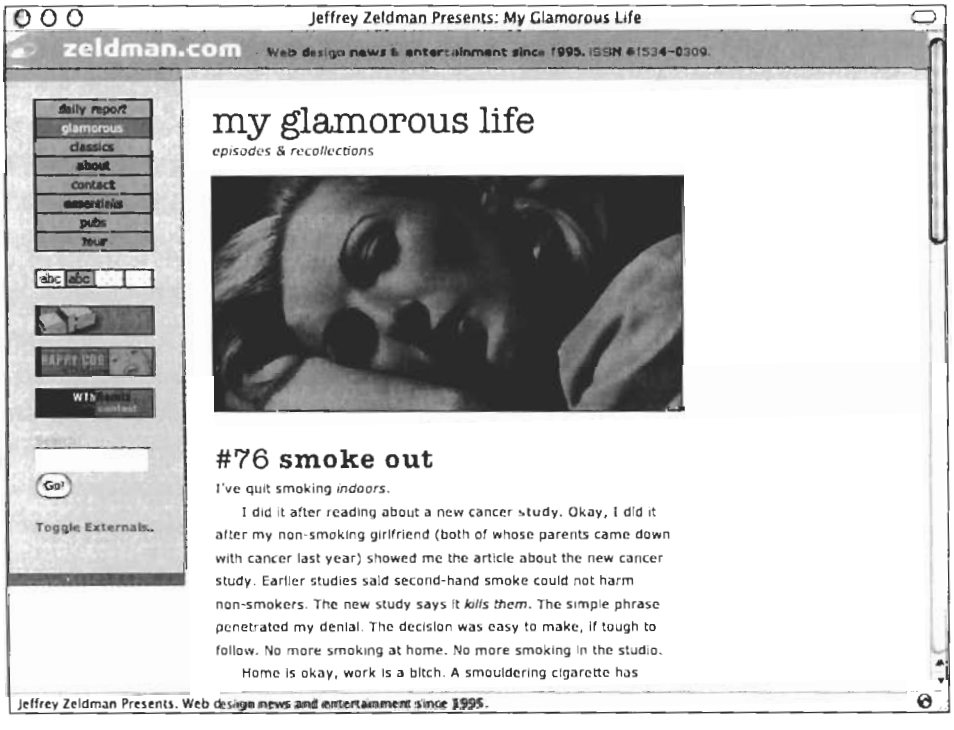


Figura 16.5. *El sangrado de párrafos de estilo impreso se puede conseguir sin necesidad de recurrir a selectores de clase.*

Figura 16.6.

Se utilizarán una o dos imágenes similares para crear un botón de inicio con efectos CSS de imagen cambiante.

El botón de inicio cambiará en respuesta a la actividad del ratón del usuario y, como mencionamos en un capítulo anterior, el efecto de imagen cambiante se consigue con ayuda de CSS, no con JavaScript. Como dijimos, las imágenes se añadirán por medio de la propiedad de fondo CSS, en lugar de la etiqueta `<img>` habitual.

En Photoshop, creamos dos imágenes similares, una de las cuales se muestra en la figura 16.6. En la imagen predeterminada, el texto del logotipo es casi blanco (`#fdf8f2`). Cuando el visitante sitúa el ratón sobre el logotipo, el texto del logotipo se vuelve totalmente blanco (`#fff`). El efecto es demasiado sutil como para reproducirlo en el libro, razón por la que hemos mostrado una figura y no dos. El fondo de ambas imágenes GIF es transparente para que el color de fondo CSS de #newmenu se pueda apreciar a través. (Recuerde que si intentamos hacer que coincida el color en Photoshop, no se verá.)

En nuestro marcado, dentro de la zona de menú, añadimos un `div` con el identificador exclusivo `bannerlogoban`:

```
<div id="bannerlogoban"><div>
```

Le aplicamos el siguiente estilo:

```
    /*  Logotipo sin imagen con imagen
        cambiante  */
#bannerlogoban {
    margin: 0;
    padding: 0;
    border: 0;
    width: 600px;
    height: 31px;
    background: url(/i/2003/parts/
                lbo2.gif) no-repeat;
}
```

La anchura es de 600 píxeles para que coincida con la de la imagen. La altura es de 31 píxeles para que coincida con la de la imagen y la del elemento contenedor (nuevo menú). La regla de imagen de fondo carga una versión previa de la imagen en estado sobre (`lbo2.gif`) por lo que no se producirá retraso alguno cuando la imagen cambiante CSS se desencadene por el movimiento del ratón del usuario.

La sustitución de imágenes de Fahrner (FIR)

En este momento disponemos de un logotipo/botón de inicio en estado "sobre" pero sobre el que no se puede pulsar y que no hace nada. Nuestra intención es que la página muestre el estado inactivo hasta que el usuario sitúe el cursor del ratón sobre la imagen.

El siguiente paso es un tanto complicado y se basa en un concepto ideado por Todd Fahrner, mencionado en numerosas ocasiones a lo largo del libro. En primer lugar, creamos una clase con el nombre `alt` con el único objetivo de que sea invisible en entornos CSS:

```
.alt  {
      display: none;
      }
```

Tras ello, creamos una serie de reglas que aplicaremos a un vínculo de ancla a la página raíz o de inicio:

```
#logoban {
    display: block;
    padding: 0;
    border: 0;
    margin: 0;
    background: url(/i/2003/parts/
                lb2.gif) no-repeat;
    width: 600px;
    height: 31px;
    }
a#logoban:hover {
    background: url(/i/2003/parts/
                lb02.gif) no-repeat;
    }
```

La primera regla carga la imagen de fondo (`lb2.gif`) y su estado inactivo. La segunda regla indica al navegador que muestre la versión activada cuando el ratón se sitúe sobre la imagen.

Seguidamente creamos el marcado necesario, de dentro a fuera. Primero creamos un texto que se vea en entornos no CSS:

```
Zeldman.com. Web design news and
entertainment since 1995. ISSN
#1534-0309.
```

Tras ello, incluimos el texto en un elemento `span` de la clase `alt` para ocultarlo de los navegadores compatibles con CSS:

```
<span class="alt">Zeldman.com. Web design
news and entertainment since 1995. ISSN
#1534-0309.</span>
```

El texto será visible en lectores de pantalla y Palm Pilots pero no se verá en navegadores de escritorio compatibles con CSS. En el siguiente paso, adjuntaremos el texto a un vínculo de ancla de nuevo a la página de inicio, aplicaremos el `id` `logoban` y añadiremos también un atributo `title`:

```
<a id="logoban" href="/"
title="Zeldman.com. Web design news
and entertainment since 1995."><span
class="alt">Zeldman.com. Web design news
and entertainment since 1995. ISSN
#1534-0309.</span></a>
```

Ya tenemos un vínculo operativo. En entornos compatibles con CSS, se representa como una imagen con efectos de imagen cambiante. En entornos no CSS, se trata simplemente de texto vinculado. Tras ello, lo incluimos todo en la regla `bannerlogoban` que creamos anteriormente:

```
<div id="bannerlogoban"><a id="logoban"
href="/" title="Zeldman.com. Web design
news and entertainment since 1995."><span
class="alt">Zeldman.com. Web design news
and entertainment since 1995. ISSN
#1534-0309.</span></a></div>
```

Es mucho marcado, pero no tanto si lo comparamos con los bytes que requiere un elemento `img`, una carga previa JavaScript, una secuencia de comandos `mouseover` de JavaScript y los controladores de eventos `onmouseover` y `onmouseout` de JavaScript que habitualmente se suelen incluir en los elementos `img`. (Además, es chulo.)

A continuación lo reproducimos tal y como aparece en la página, incluyendo el elemento "nuevo menú" principal:

```
<div id="newmenu"><div
id="bannerlogoban"><a id="logoban"
href="/" title="Zeldman.com. Web design
news and entertainment since 1995."><span
class="alt">Zeldman.com. Web design news
and entertainment since 1995. ISSN
#1534-0309.</span></a></div></div>
```

Se puede decir que podríamos haber ahorrado un paso y eliminar un `div` si hubiéramos añadido las reglas de `bannerlogoban` a `newmenu` y, tras ello, no hubiéramos incluido el `div bannerlogoban`. Pero puede que algún día queramos añadir algún elemento a la derecha del botón de inicio (o a la izquierda, aunque es menos probable) y sólo podremos hacerlo si el módulo está incluido en sí mismo.

Usos adicionales de FIR

La técnica de Fahrner se puede utilizar de diversas formas. Por ejemplo, podemos utilizarla para añadir una imagen ultra-ancha en el centro de un diseño líquido. Cuando el visitante aumente la anchura de la ventana de su navegador, se vería una mayor cantidad de la imagen. De vuelta al rancho, (es decir, en cualquier punto del cambio de diseño CSS de `zeldman.com`), utilizamos dicha técnica para crear el aspecto de tres anuncios, con efectos de imagen cambiante no sólo en las imágenes, sino también en los bordes (véase figura 16.7).

Figura 16.7.

El método Fahrner se vuelve a utilizar en otros puntos del sitio para crear el aspecto de tres anuncios, con efectos de imagen cambiante en las imágenes y otros efectos en los bordes.

La CSS inicial era similar a la que mostramos en el ejemplo anterior, a excepción de las reglas para crear los bordes. Pero no todos los navegadores actuales se han diseñado para admitir este tipo de habilidad CSS. Opera 6 sufre un problema. Netscape 7 otro. Desde las profundidades de los arrozales, pedimos refuerzos. Dicho de forma menos poética, solicitamos ayuda para solucionar algunos de estos problemas y diferencias, y Porter Glendinning se vio obligado. Entre este caballero, el señor Fahrner y nosotros, creamos una cámara de tortura CSS como la siguiente:

```
/*  Anuncios sin elementos img, gracias
    Todd y Porter  */
#banner1, #banner2, #banner3 {
    margin: 10px 0 0 0;
    padding: 0;
    width: 100px;
    height: 25px;
    }

#banner1   {
    /* Opera utiliza este fondo para el
       efecto de imagen cambiante. */
    background: url(/i/bans/
                hc100bano.gif)
                no-repeat 1px;
    }

#banner2   {
    /* Opera utiliza este fondo para el
       efecto de imagen cambiante. */
    background: url(/i/bans/
                ala100bano.gif)
                no-repeat 1px;
    }

#banner3   {
    /* Opera utiliza este fondo para el
       efecto de imagen cambiante. */
    background: url(/i/bans/zeldmix2.gif)
                no-repeat 1px;
    }

#hcban, #alban, #wtban {
    display: block;
    padding: 0;
    border: 1px solid #000;
    background: url(/i/bans/hc100ban.gif)
                no-repeat 1px; /* empiece
                a ocultarlo \*/
```

```
    background-position: 0px; /* deje de
                ocultarlo */
    width: 100px;
    height: 25px;
    voice-family: "\"}\"";   /* ¿Hace
                falta explicarlo? */
    voice-family: inherit;
    width: 98px;
    height: 23px; /* Valores para solapar
                los bordes */
    }

html>body #hcban, html>body #alban,
html>body #wtban {
    width: 98px;
    height: 23px; /* Sea amable con
                Opera */
    }

#alban {
    background-image: url(/i/bans/
                ala100ban.gif);
    }

#wtban {
    background-image: url(/i/bans/
                zeldmix.gif);
    }

a#hcban:hover {
    background-image: url(/i/bans/
                hc100bano.gif);
    border: 1px solid #ffc;
    }

a#alban:hover {
    background-image: url(/i/bans/
                ala100bano.gif);
    border: 1px solid #ffc;
    }

a#wtban:hover {
    background-image: url(/i/bans/
                zeldmix2.gif);
    border: 1px solid #ffc;
    }

.alt {
    display: none;
    }
```

Y el marcado sobre el que se asienta:

```
<div id="banner2"><a id="alban"
href="http://www.alistapart.com/"
target="eljefe" title="A List Apart,
for people who make websites."><span
class="alt">A List Apart</span></a></div>

<div id="banner1"><a id="hcban"
href="http://www.happycog.com/"
target="eljefe" title="Happy Cog Studios.
Web design and consulting."><span
class="alt">Happy Cog Studios</span>
</a></div>

<div id="banner3"><a id="wtban"
href="http://w3mix.web-graphics.com/"
target="eljefe" title="Remix the
W3C."><span class="alt">WThRemix</span>
</a></div>
```

Visualmente, este método funciona en cualquier navegador comercializado este siglo. Es más, proporciona contenido a cualquier navegador o dispositivo de Internet, nuevo o antiguo.

Si la hoja de estilo se vincula, la técnica no funcionará en el lector de pantalla JAWS que, de forma absurda, sólo lee en alto lo que se ve en un navegador después de representar la CSS. Pero si la hoja de estilo se importa, el texto no se ocultará a JAWS.

Después de salir por el otro lado del ejercicio, desde nuestro punto de vista, no tiene sentido crear efectos de borde de esta forma.

Se puede obtener el mismo resultado con mucho menos ancho de banda con tan sólo cambiar los colores de borde en Photoshop. Pero mereció la pena hacerlo para saber si se podía hacer.

Como sucede con todas las CSS que analizaremos en este capítulo, puede utilizar con total libertad la hoja de estilo original, copiarla, pegarla y adaptarla a sus necesidades.

Una barra de navegación CSS/XHTML

Volvamos a la barra lateral que dejamos sola y desvalida en la figura 16.1. El siguiente paso consiste en completarla. Por ejemplo, la completaremos con navegación para que nuestros visitantes se puedan desplazar por nuestro sitio. Conceptualmente, una barra de navegación es simplemente una lista de vínculos. Hagamos honor a dicha definición y escribamos nuestros vínculos en forma de lista estándar sin ordenar:

```
<ul>
<li id="secondarytop"><a href="/"
title="The Daily Report. Web design news
and info.">daily report</a></li>
<li id="glam"><a href="/glamorous/"
title="My Glamorous Life: Episodes and
recollections.">glamorous</a></li>
<li id="classics"><a href="/classics/"
title="Entertainments,
1995&#8211;2002.">classics</a></li>
<li id="about"><a href="/about/"
title="History, FAQ, and
suchlike.">about</a></li>
<li id="contact"><a href="/contact/"
title="Write to us.">contact</a></li>
<li id="essentials"><a href=
"/essentials/" title="Info for web
designers.">essentials</a></li>
<li id="pubs"><a href="/pubs/"
title="Zeldman&#8217;s books and
articles.">pubs</a></li>
<li id="tour"><a href="/tour/" title="We
may be coming to your town: personal
appearances.">tour</a></li>
</ul>
```

Es como cualquier persona sensata aplicaría formato a una lista sin ordenar.

Desafortunadamente, para evitar el problema del espacio en blanco que mencionamos en un capítulo anterior, debemos suprimir el espacio en blanco de esta forma:

```
<ul><li id="secondarytop"><a href="/"
title="The Daily Report. Web design news
and info.">daily report</a></li><li
id="glam"><a href="/glamorous/" title="My
Glamorous Life: Episodes and
recollections.">glamorous</a></li><li
id="classics"><a href="/classics/"
title="Entertainments,
1995&#8211;2002.">classics</a></li><li
id="about"><a href="/about/"
title="History, FAQ, and
suchlike.">about</a> </li><li
id="contact"><a href="/contact/"
title="Write to us.">contact</a></li><li
id="essentials"><a href="/essentials/"
title="Info for web
designers.">essentials</a></li><li
id="pubs"><a href="/pubs/"
title="Zeldman&#8217;s books and
articles.">pubs</a></li><li
id="tour"><a href="/tour/" title="We may
be coming to your town: personal
appearances.">tour</a></li></ul>
```

Ahora resulta más difícil de leer, ¿verdad? Pero no tenemos otra opción si nos preocupa el aspecto del sitio en Internet Explorer para Windows y, evidentemente, eso nos preocupa mucho.

Cómo añadir el estilo

Hemos definido una lista. A continuación, la transformaremos:

```
    /*  Cree los botones   */
#secondarynav ul {
    list-style: none;
    padding: 0;
    margin: 15px 0;
    border: 0;
    }

#secondarynav li   {
    text-align: center;
    border-bottom: 1px solid #000;
    width: 100px;
    margin: 0;
    padding: 0;
    font: 10px/15px Verdana, Lucida,
          Arial, sans-serif;
    color: #000;
```

```
    background: #e0861e;
    }

#secondarytop, #tertiarytop   {
    border-top: 1px solid #000;
    }

#secondarynav li a {
    display: block;
    font-weight: normal;
    padding: 0;
    border-left: 1px solid #000;
    border-right: 1px solid #000;
    background: #e0861e;
    color: #000;
    text-decoration: none;
    width: 100px;    /* Valor falso para
                        IE4-5.x/Win. */
    voice-family: "\"}\"";
    voice-family:inherit;
    width: 98px;    /* Lo ha adivinado.
          Valor correcto para los
          navegadores compatibles. */
    }

html>#secondarynav li a {
    width: 98px;    /* Sea amable con
                       Opera */
    }

#secondarynav li a:hover {
    font-weight: normal;
    border-left: 1px solid #000;
    border-right: 1px solid #000;
    background: #c90;
    color: #fff;
    text-decoration: none;
    }
```

Cuando termine de limpiar la bebida que se le ha caído sobre el libro, analizaremos calmadamente cada una de estas reglas que convierten una lista convencional en el mejor recuerdo de una novia. (No pega mucho, pero no importa.) Lo importante es que describiremos las reglas relacionadas con el tratamiento de los botones de menú.

```
#secondarynav ul {
    list-style: none;
    padding: 0;
    margin: 15px 0;
    border: 0;
    }
```

Esta regla es muy sencilla. Es broma. De hecho, en esta regla, indicamos a la lista que no muestre viñetas (`list-style: none;`). También definimos el relleno, los bordes y los márgenes izquierdo y derecho con el valor 0, y asignamos a todo el conjunto 15 píxeles de espacio en blanco en la parte superior e inferior. Tras ello:

```
#secondarynav li   {
    text-align: center;
    border-bottom: 1px solid #000;
    width: 100px;
    margin: 0;
    padding: 0;
    font: 10px/15px Verdana, Lucida,
          Arial, sans-serif;
    color: #000;
    background: #e0861e;
    }
```

Una vez definido el estilo de la parte principal (`ul`), le damos al elemento de lista secundario el cariño que se merece. El texto se centra, se ajusta la anchura (no hay valores de relleno ni de borde horizontal que puedan afectar a la anchura en navegadores que interpretan incorrectamente el modelo de cuadro) y a cada elemento de lista se le indica que genere un línea negra sólida en su parte inferior. Cuando se añada esta línea a las que describiremos más adelante, se creará el efecto de un cuadro o el contorno de un botón.

Por último, fíjese en el valor de altura de línea, que es el segundo valor que aparece después de la fuente (15 píxeles). (`Font: 10px/15px` es una abreviatura CSS. Significa lo mismo que `font-size: 10px; line-height: 15px`. Pero ya lo sabía.) Con esta regla, indicamos a cada elemento de lista, o botón, que tenga una altura de 15 píxeles y que sitúe el texto en el centro

vertical (porque así funciona la altura de línea). Para una regla de aspecto tan sencillo, es más que suficiente.

```
#secondarytop, #tertiarytop   {
    border-top: 1px solid #000;
    }
```

Esta otra regla añade un borde negro en la parte superior de un elemento con el `id` exclusivo `secondarytop` o `tertiarytop`. En este caso, solamente nos preocupa `secondarytop`. Si se fija otra vez en el marcado, bueno se lo pondremos más sencillo y se lo mostraremos de nuevo:

```
<ul><li id="secondarytop"><a href="/"
title="The Daily Report. Web design news
and info.">daily report</a></li>
```

Comprobará que el primer elemento de lista tiene el `id` `secondarytop`. Por ello, además de las reglas que aplican estilo a todos los elementos de lista de la barra lateral, éste depende de una regla adicional que le indica que se añada una línea negra (un borde de 1 píxel) en la parte superior. En otras palabras, el primer botón tendrá parte superior e inferior. Es evidente cuando lo piensa. Si aplicamos un borde superior e inferior a todos los elementos de lista, obtendremos una línea que los une, lo que no resultaría correcto. Esto, por otra parte, si lo es.

Puntos de clic y rellenos de color: estética y accesibilidad

Prácticamente hemos terminado. Tranquilo; lo siguiente parece peor de lo que es:

```
#secondarynav li a {
    display: block;
    font-weight: normal;
    padding: 0;
    border-left: 1px solid #000;
    border-right: 1px solid #000;
    background: #e0861e;
    color: #000;
    text-decoration: none;
    width: 100px;    /* Valor falso para
                        IE4-5.x/Win. */
    voice-family: "\"}\"";
    voice-family:inherit;
    width: 98px;    /* Lo ha adivinado.
           Valor correcto para los
           navegadores compatibles. */
    }
```

En esta regla, hemos definido el estilo del vínculo y le hemos indicado que se muestre en bloque y no en línea. Al combinarlo con la altura de línea que establecimos en `#secondarynav li`, el resultado es un botón con un relleno de un determinado color (mejora estética) y que se puede hacer clic en todo el botón (una importante mejora de accesibilidad para los usuarios con dificultades motrices o visuales, como indicamos en un capítulo anterior). Además, el botón realmente tiene el aspecto operativo y visual de un botón.

También hemos creado bordes izquierdo y derecho para completar el relleno del botón, y hemos utilizado los trucos habituales para persuadir a IE/Windows de que no estropee el modelo de cuadro.

```
html>#secondarynav li a {
    width: 98px;    /* Sea amable con
                       Opera */
    }
```

A continuación (redoble de tambores), la última regla que necesita nuestra fiesta de botones, el efecto de imagen cambiante:

```
#secondarynav li a:hover {
    font-weight: normal;
    background: #c90;
    color: #fff;
    text-decoration: none;
    }
```

De nuevo, por motivos estéticos y de accesibilidad, se puede hacer clic en la totalidad del botón, gracias a `display: block` que se definió en la regla de vínculo anterior, que se deriva en ésta otra más específica. Esta regla indica al color del texto y al color de fondo que deben cambiar en respuesta al cursor del ratón del usuario. No repetiremos las instrucciones sobre el borde izquierdo y el derecho ya que los efectos de bordes negros de 1 píxel se conservan de reglas anteriores.

Como toque final, como hicimos al crear nuestro diseño híbrido en un capítulo anterior, persuadimos al botón correspondiente a que presente un efecto "Usted está aquí". Para ello, añadimos un estilo incrustado específico de la página al elemento `<head>` de la misma:

```
<style type="text/css">
<!--
#secondarytop a:visited {
    font-weight: normal;
    background: #c60;
    color: #fff;
    text-decoration: none;
    }
-->
</style>
```

Fíjese en las figuras 16.8 y 16.9. El efecto "Usted está aquí" se puede apreciar en el botón daily report, con un fondo más oscuro (`#c60`) y un texto invertido (`#fff`). El efecto adicional de imagen cambiante activado por el usuario puede apreciarse en la figura 16.9.

En una nota técnica, los comentarios `<!--` que rodean a nuestra hoja de estilo constituirían un problema si el proceso de creación se realizara en XHTML 1.0 Strict, pero hemos utilizado Transitional. Estos comentarios también resultarían problemáticos si ofreceríamos las páginas como `application/xhtml+xml` pero lo estamos haciendo como `text/html`. Para entender por qué estos elementos podrían constituir un problema en XHTML Strict, consulte entonces http://devedge.netscape.com/viewsource/2003/ xhtml-style-script/.

Figura 16.8.

Una simple lista sin ordenar recibe un tratamiento de belleza cortesía de CSS. Pero espere, todavía hay más.

Figura 16.9.

El omnipresente efecto de imagen cambiante. Para el ciudadano medio, se trata de botones de navegación estándar. De hecho, simplemente se trata de marcado al que se le ha aplicado un estilo. Ahora ya lo sabe.

Al combinar la barra de navegación que acabamos de crear con los anuncios, la interfaz del motor de búsqueda y un dispositivo de conmutación de estilo que hemos creado cuando no miraba, el resultado es el que se recoge en la figura 16.10.

Figura 16.10.

La barra lateral completada.

Retoques finales

Hay mucho más sobre diseño de lo que se puede analizar en un sólo capítulo, incluso en un capítulo tan extenso como éste. Existen mejoras de accesibilidad aplicadas a elementos `form`, incluyendo el formulario de búsqueda. Están nuestros viejos amigos Skip Navigation, `tabindex` y `accesskey`. Está el diseño de un dispositivo de conmutación de estilo (que se puede apreciar en la parte central de la figura 16.10) y el código en el que se basa, tomado del conmutador de estilo de código abierto de A List Apart que mencionamos en un capítulo anterior. Está la necesidad de crear un estilo alternativo (véase figura 16.11) que, principal pero no exclusivamente, esté formado por valores de color cambiantes. Y están los detalles como la barra lateral Externals (véase figura 16.12) que se oculta de la vista hasta que el usuario decide abrirla.

Una vez terminado el trabajo, queda la inevitable depuración, aunque en este caso no es excesiva ya que los navegadores comprenden las CSS que hemos utilizado. Aun así, aparecieron algunos problemas.

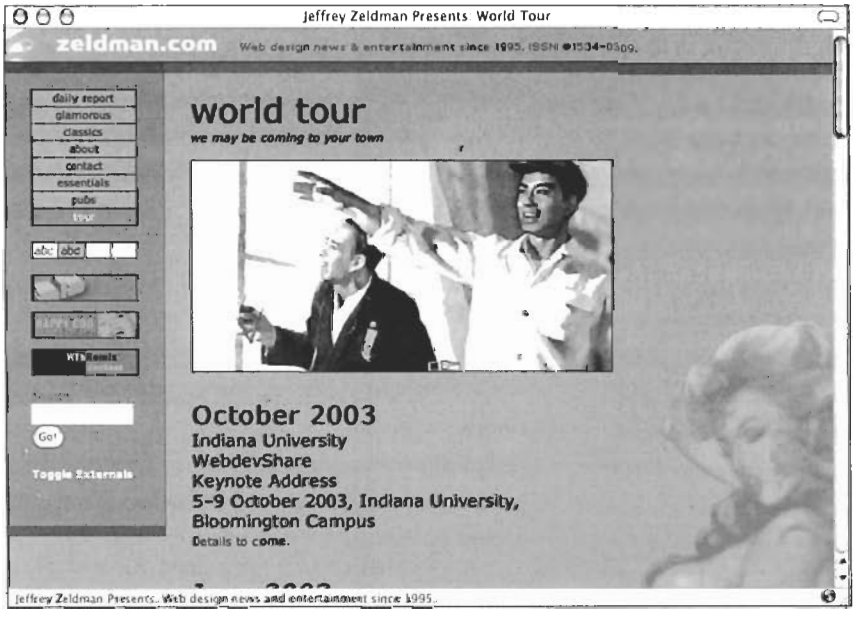


Figura 16.11.

La opción alternativa (naranja) toma forma.

Por ejemplo, el navegador Safari de Apple sufre un problema inicial de latencia de imágenes después de cambiar del diseño blanco al naranja (véanse figuras 16.13 y 16.14). Para ser justos con Safari, diremos que usamos la versión beta del navegador cuando realizamos el cambio de diseño y que, aparte de este error, procesaba el diseño de forma casi perfecta. Es más, conseguimos solucionar el problema de Safari con bastante facilidad.

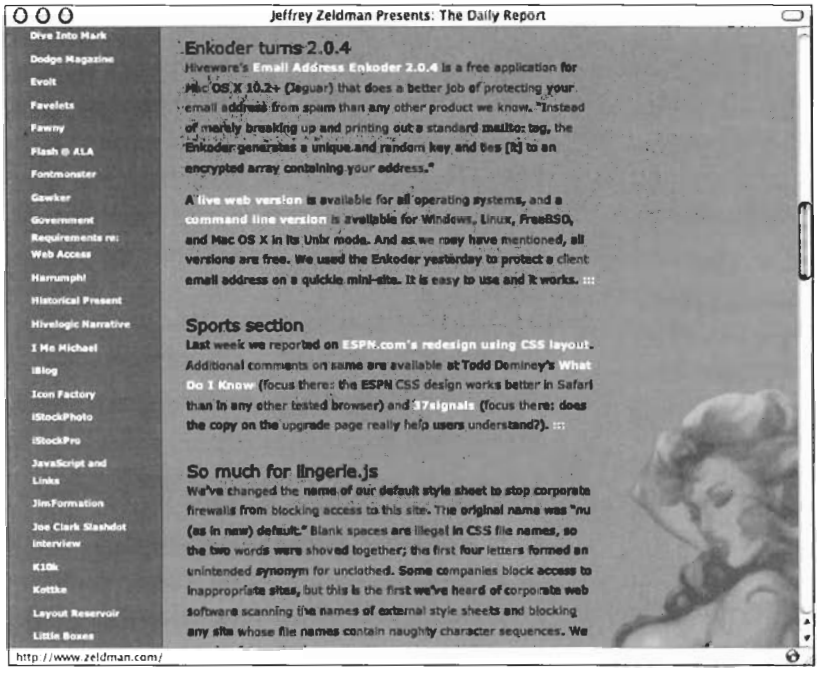


Figura 16.12.

El diseño naranja, en el que la barra lateral Externals anteriormente oculta se ha vuelto a activar. (En un capítulo anterior encontrará más información sobre cómo ocultar y mostrar contenido por medio del DOM).

Figura 16.13.

En la imagen se aprecia el error de latencia de imagen de fondo del navegador Safari de Apple.

Figura 16.14.

El problema es más evidente cuando se ve la barra lateral de forma aislada. Debería aparecer una barra de color en la parte inferior de la barra lateral. Safari muestra dos. Compárelo con las figuras 16.2 y 16.11.

Veamos lo que sucedió. Si recuerda, el diseño blanco utilizaba una imagen de fondo en la parte inferior de la barra lateral (véanse figuras 16.4 y 16.10). El diseño naranja, como no sabrá a menos que ya se haya conectado y haya visitado http://www.zeldman.com/c/or.css, no utiliza esta imagen de fondo. Pero cuando Safari cambia al diseño naranja, retiene incorrectamente la imagen de fondo del diseño blanco.

La solución era sencilla. En una revisión del diseño naranja, indicamos a todos los navegadores que no utilizaran ninguna imagen de fondo en la barra lateral. Con eso bastó.

En la figura 16.15 se puede ver los efectos del ajuste CSS en Safari.

A continuación reproducimos la regla original que escribimos para el diseño naranja:

```
background: #c60;
```

Figura 16.15.

Una rápida modificación CSS resuelve el problema de Safari.

Y, en este caso, el cambio que solucionó el problema en Safari:

```
background-color: #c60;
background-image: none;
```

Aparte de Safari, este cambio no fue necesario en ningún navegador. Pero este error de Safari resulta sencillo de solucionar y el cambio en las reglas es perfectamente válido. Sin duda la versión final de Safari no sufrirá este problema pero, mientras tanto, `background-image: none` lo soluciona y no resulta dañino. Al igual que ocurre con otros métodos de diseño Web, este tipo de depuración es el último paso, aunque en este caso de cambio de diseño la depuración, como todo lo demás, se realizó en tiempo real mientras los lectores observaban.

Tras ello hemos creado un diseño CSS sin tablas y una versión alternativa que el usuario puede seleccionar, sin rastro de marcado de presentación ni de nada similar a la locura de enviar a cada navegador un archivo CSS diferente. El sitio se carga con rapidez, tiene el mismo aspecto operativo y visual en

todos los navegadores modernos, y resulta accesible para cualquier dispositivo de Internet. El capítulo acaba aquí pero el cambio de diseño del sitio continúa. Por ejemplo, tenemos pensado crear un diseño alternativo adicional que ofrezca texto de mayor tamaño o que permita cambiar de tamaño con más facilidad. Pero ésa es otra historia y ya es hora de dejar de escucharnos y de empezar a pensar en las historias que nos quiera contar.

Parte III
Apéndice

Apéndice
Navegadores modernos: el bueno, el feo y el malo

Como explicamos al principio del capítulo, cuando hablamos de navegadores modernos o compatibles con estándares nos referimos a navegadores que comprenden y admiten HTML y XHTML, CSS, ECMAScript y el Modelo de objeto de documento (DOM) del W3C. Estos estándares básicos ayudan a diseñadores y programadores a superar los métodos de la vieja escuela (marcado de presentación y lenguajes de secuencia de comandos incompatibles) y la obsolescencia que entrañan.

Todavía no ha aparecido ningún navegador que admita los estándares a la perfección y no es probable que aparezca ninguno. Pero al inicio del nuevo milenio surgieron algunos navegadores con una compatibilidad casi completa con los estándares básicos. Cuando las versiones actualizadas de estos navegadores lleguen al mercado, serán más compatibles y tendrán menos fallos. También aumentarán su cuota de mercado.

Al cierre de esta edición, prácticamente todos los usuarios Web han actualizado sus navegadores a uno de los que describimos a continuación o a una versión posterior y mejorada. En la lista únicamente se incluyen los navegadores más conocidos y únicamente se destacan algunas de sus características (y posibles puntos débiles).

Navegadores compatibles: la primera ola

Opera 7

Año de aparición: 2002.

¿Compatible con HTML/XHTML?: Sí.

¿Compatible con CSS?: Prácticamente con la totalidad de CSS1 y gran parte de CSS2.

¿Compatible con ECMAScript/DOM?: Sí, la primera versión de Opera que lo hace.

Datos curiosos: La primera versión de Opera compatible con el DOM del W3C y la primera versión de Opera realmente compatible con los estándares. La empresa muestra un excelente historial de compatibilidad con estándares anteriores como HTML y CSS. Al igual que en versiones anteriores de Opera, Opera 7 incluye la opción de zoom de texto para que el texto y los gráficos Web sean más accesibles para los usuarios con discapacidades visuales.

MSIE5+/Macintosh

Año de aparición: 2000.

¿Compatible con HTML/XHTML?: Sí.

¿Compatible con CSS?: Prácticamente con la totalidad de CSS1 y gran parte de CSS2.

¿Compatible con ECMAScript/DOM?: Sí.

Datos curiosos: Comercializado en marzo de 2001. El primer navegador compatible con estándares del mercado, la primera versión de IE/Mac que admitía correctamente JavaScript/ECMAScript y el primer navegador de todas las plataformas que admitía correctamente el modelo de cuadro. Incluye la opción de zoom de texto para que el texto y los gráficos Web sean más accesibles para los usuarios con discapacidades visuales. Compatible con hojas de estilo de usuario.

El motor de representación Tasman del navegador, responsable de la mejora en la compatibilidad con los estándares, comienza a dar muestras de envejecimiento. La compatibilidad de IE5/Macintosh con el DOM es correcta, pero no tan completa como se podría esperar. Es un poco lento y a veces su comportamiento vuelve locos a los creadores de contenido dinámico. Aun así, el navegador admite funciones DOM básicas y su valoración media es muy positiva, desde el punto de vista de los estándares.

Netscape 6+

Año de aparición: 2001.

¿Compatible con HTML/XHTML?: Sí.

¿Compatible con CSS?: La totalidad de CSS1 y gran parte de CSS2.

¿Compatible con ECMAScript/DOM?: Prácticamente sí, pero algunos aspectos resultan extraños y la representación dinámica es lenta si la comparamos con IE/Windows.

Datos curiosos: Navegador basado en Gecko diseñado desde un principio para admitir los estándares Web CSS, XML, XHTML, DOM y ECMAScript. (Al igual que Tasman, Gecko es un motor de representación diseñado para admitir estándares Web básicos. Tasman es exclusivamente para Macintosh; Gecko es para todas las plataformas.)

Las primeras versiones de Netscape 6.0 eran ligeramente defectuosas. Las siguientes son mejores y las versiones 7.0 y superiores son sobresalientes.

Incluye zoom de texto para la accesibilidad. Admite hojas de estilo de usuario y hojas de estilo alternativas. También admite el bloqueo de anuncios, desde Netscape 7.01.

Mozilla 1.0

Año de aparición: 2002. Moz 1.0 apareció en mayo de 2002, pero el Mozilla original data de 1996 aproximadamente.

¿Compatible con HTML/XHTML?: Sí.

¿Compatible con CSS?: La totalidad de CSS1 y gran parte de CSS2.

¿Compatible con ECMAScript/DOM?: Véase la información sobre Netscape 6+.

Datos curiosos: Navegador de código abierto basado en Gecko diseñado desde un principio para admitir los estándares Web. Utilizado principalmente por fanáticos de la informática, su facilidad de uso lo hace adecuado para cualquiera. Incluye zoom de texto para mejorar la accesibilidad. Admite hojas de estilo de usuario y alternativas.

Entre los navegadores basados en Mozilla destacamos Chimera/Camino y Phoenix, pero Mozilla no vive únicamente de la navegación. Se puede utilizar Gecko y el código abierto base de Mozilla para crear nuevas aplicaciones más allá del navegador de escritorio tradicional. (Por ejemplo, existe un reproductor de televisión basado en Mozilla y Java.)

Safari

Año de aparición: Finales de 2002.

¿Compatible con HTML/XHTML?: Sí.

¿Compatible con CSS?: Aparentemente admite la totalidad de CSS1 y gran parte de CSS2, aunque la compatibilidad es en ocasiones incompleta.

¿Compatible con ECMAScript/DOM?: Prácticamente sí.

Datos curiosos: Creado por Apple Computer para sus usuarios OS X, se basa en un motor KHTML de código abierto. Ligero, rápido y preciso en la mayoría de los sitios. Aunque al cierre de esta edición se encontraba en su versión beta, millones de usuarios de Macintosh ya han adoptado Safari. Incluye zoom de texto para mejorar la accesibilidad. También incluye un botón de informe para poder solucionar errores en CSS, XHTML o de secuencias de comandos.

MSIE6/Windows

Año de aparición: 2001.

¿Compatible con HTML/XHTML?: Sí.

¿Compatible con CSS?: Prácticamente con la totalidad de CSS1 y parte de CSS2.

¿Compatible con ECMAScript/DOM?: Casi en su totalidad, pero también con muchos elementos propietarios, que hace que muchos se vean tentados a crear código exclusivamente para IE6/Windows, algo que no

tiene sentido, a menos que se quiera crear un sitio de red interna sólo para IE. Incluso por este motivo puede que sea más indicado utilizar el estándar DOM, en caso de que funciones o partes del sitio se cambien posteriormente a un espacio Web público.

Datos curiosos: La versión más compatible de IE/Windows. Actualmente, el navegador más utilizado en la Web, en parte por ser el único incorporado en un sistema operativo. Cuando se utiliza junto a Windows XP Clear Type, representa textos de forma asombrosa.

No incluye zoom de texto, zoom de página ni admite hojas de estilo alternativas. Los usuarios con defectos visuales pueden mejorar la accesibilidad del texto si utilizan la opción Omitir tamaños de fuentes especificados en páginas Web en sus preferencias de Usuario, pero esta propuesta de tipo todo o nada es menos sofisticada y menos útil que el zoom de texto o de página que ofrecen otros navegadores compatibles con los estándares.

Los usuarios de IE/Windows pueden cambiar el tamaño del texto por medio de un dispositivo incluido en el navegador, pero a algunos métodos de tamaño de texto no les afecta esta herramienta. (Por ejemplo, el texto definido en píxeles no se puede cambiar de tamaño en IE/Windows.) Por el contrario, todo el texto, independientemente de cómo se haya especificado, se puede cambiar de tamaño en IE5+/Macintosh, Mozilla, Netscape 6+, Opera y Camino.

MSIE5.5/Windows

Año de aparición: 2001.

¿Compatible con HTML/XHTML?: Sí, aunque con algunas omisiones.

¿Compatible con CSS?: Prácticamente en su totalidad (pero con algunos errores de importancia).

¿Compatible con ECMAScript/DOM?: No realmente.

Datos curiosos: Concienciado con los estándares, fue un buen navegador en su día, pero sus problemas con CSS y los lenguajes de secuencia de comandos lo hacen menos compatible que otros de los que hemos enumerado. En capítulos anteriores del libro encontrará una solución para resolver sus problemas con CSS.

MSIE5/Windows

Año de aparición: 1999.

¿Compatible con HTML/XHTML?: Sí, aunque con algunas omisiones.

¿Compatible con CSS?: Ligeramente, pero con algunos problemas y omisiones importantes.

¿Compatible con ECMAScript/DOM?: No lo suficiente.

Datos curiosos: Consulte los comentarios sobre IE5.5.

Netscape 4

Año de aparición: 1997.

¿Compatible con HTML/XHTML?: Sólo de forma parcial.

¿Compatible con CSS?: Escasa.

¿Compatible con ECMAScript/DOM?: No (no se diseñó para que lo fuera).

Datos curiosos: Comercializado en el momento álgido de las guerras de los navegadores, este otrora poderoso navegador se diseñó para que admitiera código y marcado propietario, y no estándares. Su compatibilidad con CSS era escasa, equivocada e incompleta. A este respecto, su compatibilidad con HTML básico no era nada del otro mundo. No admitía el DOM porque el DOM todavía no se había escrito y probablemente no lo hubiera admitido aunque se hubiera escrito ya que durante las guerras de los navegadores Netscape y Microsoft estaban convencidos de que solamente podrían ganar si inventaban nuevas tecnologías a expensas de los estándares.

Aunque la mayoría de los usuarios han actualizado a Netscape 6+ o han cambiado a un producto competente como MSIE u Opera, algunos no lo han hecho por diferentes razones. La persistencia de Netscape 4, con su lamentable compatibilidad con los estándares, hace que muchos diseñadores y programadores piensen que deben seguir utilizando métodos anticuados para admitir este grupo de usuarios en vías de extinción. Esperamos que este libro le sirva para saber que puede admitir a usuarios de Netscape 4 y a todos los demás, al tiempo que utiliza los estándares.

MSIE4

Año de aparición: 1997.

¿Compatible con HTML/XHTML?: Más que Netscape 4 pero no mucho.

¿Compatible con CSS?: Más que Netscape 4.

¿Compatible con ECMAScript/DOM?: No (no se diseñó para que lo fuera).

Datos curiosos: Comercializado en el momento álgido de las guerras de los navegadores, MSIE4 admitía código y marcado propietario en lugar de estándares. Prácticamente todos los usuarios de IE4 han cambiado a una versión más moderna, en parte porque Microsoft incluye el navegador en su sistema operativo. Si, por ejemplo, actualiza de Windows 95 a Windows XP, instantáneamente cambiará de IE4 a IE6. Aunque IE4 es ligeramente mejor que Netscape 4 en lo que a estándares se refiere (a pesar de lo que Microsoft le haya hecho creer), resulta un problema menor para los programadores ya que se utiliza mucho menos.

Índice alfabético

C

D

E

F

G

H

I

N

O

P

Q

R

S

T

U

V

W

X

Z