USERS

Incluye prácticas paso a paso

Diseño web con <HTML & CSS>

Creación de sitios atractivos y profesionales

- Aplicaciones y herramientas necesarias
- Estructura del sitio
- Definición de estilos para textos
- Optimización de imágenes
- ¡Y mucho más!

RU RedUSERS

CONÉCTESE CON LOS MEJORES LIBROS DE COMPUTACIÓN

GUÍA PRÁCTICA PARA LA CREACIÓN DE BLOGS PROFESIONALES

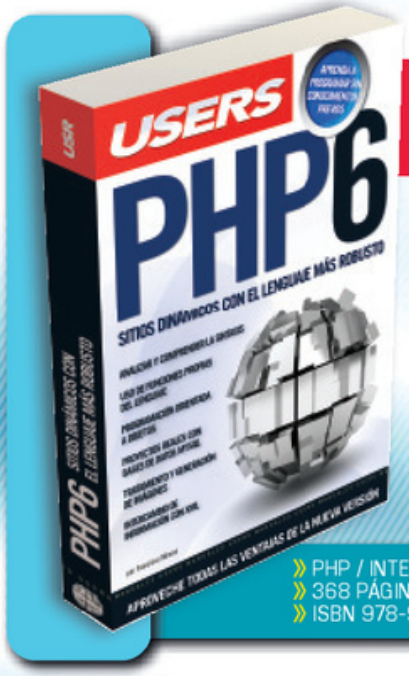
» INTERNET / HOME
» 352 PÁGINAS
» ISBN 978-987-1773-18-3

CREACIÓN, EXPANSIÓN Y MONETIZACIÓN DE BLOGS ATRACTIVOS

» INTERNET / HOME
» 352 PÁGINAS
» ISBN 978-987-1347-96-4

ENTIENDA EL CAMBIO, APROVECHE SU POTENCIAL

» DESARROLLO
» 320 PÁGINAS
» ISBN 978-987-1773-79-4



APRENDA A CREAR SITIOS DINÁMICOS CON EL LENGUAJE MÁS ROBUSTO

» PHP / INTERNET
» 368 PÁGINAS
» ISBN 978-987-663-039-9

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

MÁS INFORMACIÓN / CONTÁCTENOS

usershop.redusers.com +54 (011) 4110-8700 usershop@redusers.com

*SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // **VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

Diseño Web con HTML & CSS

Creación de sitios atractivos
y profesionales

TÍTULO: Diseño web con HTML y CSS
COLECCIÓN: Desde Cero
FORMATO: 15 x 19 cm
PÁGINAS: 192

ISBN 978-987-1857-66-1

Diseño web con HTML y CSS / coordinado por Gustavo Carballeiro. -
1a ed. - Buenos Aires : Fox Andina; Dalaga, 2012.
v. 26, 192 p. ; 19x15 cm. - (Desde cero)
ISBN 978-987-1857-66-1

1. Informática. I. Carballeiro, Gustavo, coord.
CDD 005.3

RedUSERS.com

Prólogo
al contenido

A medida que Internet creció, los usuarios se fueron convirtiendo en expertos navegantes que comenzaron a demandar cada vez una mayor cantidad de soluciones basadas en la Web.

Estos requerimientos de los internautas generaron oportunidades de trabajo que fueron aprovechadas, en un comienzo, solo por profesionales con estudios relacionados con la programación. Sin embargo, cuando la apariencia de los sitios cobró casi tanta importancia como su funcionalidad, esos técnicos se quedaron sin respuestas para dar a los clientes, y entonces recurrieron a los diseñadores gráficos.

Las herramientas de desarrollo y de diseño crecieron hasta permitir la creación de sitios con amplias funcionalidades, y con aspecto sobresaliente y llamativo para los navegantes.

En este nuevo mundo, podremos crecer profesionalmente sin realizar más inversión que la de nuestro capital intelectual. Para eso, deberemos adquirir los conocimientos técnicos que se convertirán en un negocio rentable cuando les sumemos una cuota de nuestra propia creatividad y buenas ideas.

A pesar de esta evolución, sin dudas, la base en todo diseño web hace referencia al conocimiento y manejo de HTML. En este sentido, esta obra permite adquirir todos esos conocimientos, ya que propone un aprendizaje guiado a través de los distintos conceptos y prácticas que debemos manejar para convertirnos en expertos en HTML.

Comencemos, entonces, el fascinante recorrido hacia el éxito en el desarrollo de sitios profesionales; suerte en el camino.

El libro **de un vistazo**

Esta obra reúne todos los conocimientos teóricos y prácticos necesarios para apoyar la tarea de creación de sitios web. Revisaremos en profundidad el uso y el potencial de HTML y CSS, aprendiendo su manejo en diversas situaciones prácticas.

CAPÍTULO 1
PRIMEROS PASOS

Aquí repasaremos las características principales del diseño web, conoceremos su historia y evolución, así como también, la estructura básica de un sitio y algunos consejos que debemos tener en cuenta para que nuestras creaciones sean exitosas.

CAPÍTULO 2
DEL DISEÑO AL HTML

Este capítulo nos acompañará a través de la implementación de la estructura HTML que necesitamos para plasmar una idea de diseño. Conoceremos las herramientas que intervienen en este proceso y crearemos nuestro primer documento HTML.

CAPÍTULO 3
ESTRUCTURA DEL SITIO

Esta sección nos introducirá en los conceptos básicos y avanzados relacionados con la estructura de un sitio web. Revisaremos las tendencias en diseño web, incorporaremos bordes y armaremos la estructura de un sitio.

CAPÍTULO 4
TEXTOS

En esta etapa profundizaremos en los conceptos relacionados con la incorporación de textos en la estructura de un sitio web. Aprenderemos a crear títulos, y nos familiarizaremos con los párrafos y el marcado básico. También daremos los primeros pasos en la aplicación de estilos CSS.

CAPÍTULO 5
IMAGEN

El uso de imágenes es fundamental para lograr que un sitio web sea atractivo. En este capítulo conoceremos los detalles necesarios para trabajar con las propiedades de imagen, realizar mapas de imágenes y optimizarlas para usar en la Web.

CAPÍTULO 6
ENLACES

Los enlaces serán analizados en detalle a lo largo de este capítulo. Conoceremos su definición y clasificación, así como también aprenderemos a crear enlaces dentro de la herramienta que usamos para diseñar el sitio web.

CAPÍTULO 7
LISTAS

En este capítulo podremos conocer y aprender a utilizar las listas, veremos ejemplos de uso y aprenderemos a generarlas. Conoceremos el procedimiento adecuado para personalizar listas con CSS y crearemos menús de navegación.

CAPÍTULO 8
FORMULARIOS

Los formularios son importantes para establecer contacto con nuestros visitantes. Por eso, en este capítulo conoceremos los detalles en torno a su creación y uso. Además, nos dedicaremos a crear un formulario completo utilizando el lenguaje PHP.

Contenido **del libro**

CAPÍTULO 3
ESTRUCTURA DEL SITIO 061

CAPÍTULO 4
TEXTOS 087

CAPÍTULO 5
IMAGEN 105

CAPÍTULO 6
ENLACES 121

CAPÍTULO 7
LISTAS 141

CAPÍTULO 8
FORMULARIOS 165

SERVICIOS AL LECTOR 187

Introducción a
Diseño web con HTML y CSS

Como sabemos, la era digital generó nuevas y fascinantes oportunidades de trabajo. Esta obra es la puerta de acceso a una de las áreas que más posibilidades y satisfacciones brindan en el ámbito web: el desarrollo de sitios. En este libro, los mejores consejos y contenidos son expuestos al lector, junto a los fundamentos de este arte y, al mismo tiempo, todos los secretos para que triunfemos ante cualquier desafío que nos toque encarar, convirtiéndonos en excelentes profesionales de la Web.

A través de cada uno de sus capítulos, podremos conocer y dominar HTML, así como también las principales tecnologías relacionadas. En sus páginas encontraremos un recorrido por todos los elementos teóricos fundamentales en esta profesión y veremos la forma de llevar estos conceptos a la práctica ayudados por detalladas explicaciones paso a paso.

Sin dudas, este aprendizaje sentará bases sólidas, y nos llevará a diseñar y desarrollar sitios web profesionales que resalten nuestro trabajo.

Los contenidos expuestos en esta obra están dirigidos a todos aquellos que quieran desarrollar sitios, incluso a quienes no cuentan con educación formal previa en la materia.

Por un lado, los desarrolladores web aumentarán su oferta de servicios, y se convertirán en profesionales con conocimientos avanzados de HTML. Por otra parte, los estudiantes de Diseño Gráfico serán trabajadores mucho más completos cuando finalicen sus estudios, ya que lograrán implementar sitios con buen diseño, alta funcionalidad y códigos optimizados. Finalmente, los aficionados a la Web tendrán la posibilidad de participar en ella de forma activa mediante la creación de sus propios sitios, y serán capaces de diseñarlos también para terceros, con lo cual transformarán el objeto de su devoción en una tarea rentable y productiva.

En suma, el material que tienen delante de ustedes es una obra imperdible para todo tipo de usuarios, quedan en buenas manos.

Capítulo 1

Primeros pasos

Aquí analizaremos los primeros pasos para iniciarnos en el desarrollo de sitios web utilizando HTML.

RedUSERS.com

El diseñador **web**

Actualmente, es imposible referirse a un solo tipo de perfil de diseñador web, y mucho menos, a un solo tipo de persona. La creación de sitios web se ha convertido en una disciplina en la que se plantean proyectos que requieren de una conjunción de conocimientos **técnicos** (como lenguajes de programación y servidores, entre otros), **conceptuales** (usabilidad y accesibilidad, entre otros) y también **estéticos** (como percepción y legibilidad).

Atrás quedó la época en la que para realizar un sitio web completo alcanzaba con tener un mínimo conocimiento de HTML y del tratamiento de imágenes. Ahora es importante que un diseñador sepa cómo planificar un sitio web y que tenga presentes los aspectos técnicos de diagramación y de estructura al momento de ponerse a diseñar.

LAS HERRAMIENTAS DE TRABAJO

Para realizar su trabajo, un profesional del diseño y el desarrollo de sitios web necesita contar con herramientas de software, que debe utilizar con criterio para sacar el máximo provecho de ellas, además de conocer los fundamentos de su tarea.

Debe ser un usuario intermedio de PC o Mac, utilizar los navegadores web y el correo electrónico, y estar al tanto de la existencia y el funcionamiento de las redes sociales.

Si bien existen muchas herramientas de diseño en el mercado, lo cierto es que hay un estándar bien definido de lo que se usa. Basta con echar una mirada a las solicitudes de trabajo que aparecen en los medios para saber qué es lo que se pretende hoy en día de un diseñador web.

Un diseñador web tiene que saber **HTML** y **CSS**, y tiene que trabajar con **Dreamweaver** para armar los sitios en forma correcta. Debe utilizar las herramientas principales de diseño, como **Photoshop**, **Illustrator** y **Fireworks**, que permiten trabajar en forma profesional y con la mejor calidad. Además, resultan importantes los conocimientos de diseño y animación con Flash, así como también la programación en ActionScript 3.

FIGURA 1. **La suite de programas más famosos para diseñar sitios. Adobe CS5, está disponible para Windows y para Mac OSX.**

LAS HABILIDADES DEL DISEÑADOR

Aunque es posible decir que un buen diseñador web es aquel que sabe utilizar las herramientas con las que desarrolla sus proyectos, este es solo un aspecto técnico que no resulta fundamental para tener éxito. La buena fortuna de un diseñador web depende, en gran medida, de sus conocimientos estéticos y conceptuales.

También tiene que estar inmerso en el mundo virtual de Internet, y conocer las últimas tendencias y tecnologías aunque no las aplique por sí mismo, ya que así podrá comunicarse para solicitar su implementación e interactuar con otros profesionales.

Como vemos, la tarea más difícil de un buen diseñador web no radica en aprender a utilizar una aplicación, sino en saber utilizarla con criterio y estar al tanto de todo lo que influye en su medio de trabajo, como la aparición de nuevas técnicas de diseño, cambios en las tendencias y en los estilos, o nuevos dispositivos y utilidades que simplifiquen su trabajo.

El éxito de un diseñador web depende de sus conocimientos técnicos, estéticos y conceptuales.

En este sentido, debe saber utilizar los programas más conocidos de diseño pero, sobre todo, saber determinar en qué momento conviene aplicar una u otra herramienta según lo que requieran los proyectos web en los que esté involucrado.

TAREAS DEL DISEÑADOR

Para definir de forma más clara el perfil del diseñador web, podemos realizar una división de sus tareas sobre la base de las diferentes etapas en las que se podría ver involucrado dentro de la realización de un proyecto web.

Si pensamos en la concepción del diseño de un sitio, es inevitable que sea capaz de realizar un **resumen** o **brief** de trabajo en donde pueda determinar

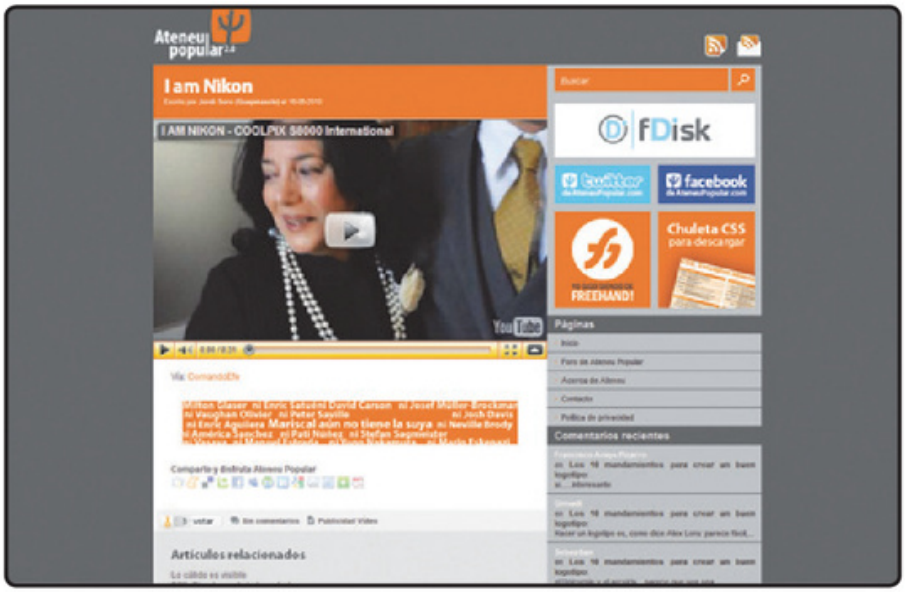


FIGURA 2. **Es muy importante que visitemos sitios donde encontremos información sobre las últimas tendencias y novedades.**

las necesidades estéticas y técnicas del proyecto, y según esto, comenzar a diseñar. En este momento, será preciso que disponga de herramientas para confeccionar el diseño. Si el sitio es **HTML**, puede trabajar con una aplicación como Photoshop, Illustrator o Fireworks, que ya mencionamos. En cambio, si el sitio es en **Flash**, deberá determinar la mejor forma de crear un diseño optimizado, utilizar las herramientas gráficas del programa y conocer su lenguaje de programación, **ActionScript**.

Para las etapas posteriores al diseño de los sitios, es necesario que sepa maquetar en HTML y utilizar CSS, y para estas tareas, Dreamweaver resulta una herramienta esencial.

Por último, es importante que tenga conocimientos sobre la forma de alojar un sitio en un servidor, ya sea por medio de Dreamweaver o mediante un programa especial para **FTP** (File Transfer Protocol), como, por ejemplo, FileZilla.

Un diseñador necesita contar con muchos conocimientos para poder crear un sitio de calidad. En esta obra, veremos todas esas herramientas e, incluso, obtendremos los mejores consejos que nos brinda la experiencia de los expertos que investigaron y escribieron sobre este tema.

Evolución del diseño web

Como usuarios de Internet, navegamos decenas de sitios por día con el objetivo de revisar nuestro correo, informarnos, compartir imágenes, comprar productos, contar lo que estamos haciendo o, simplemente, divertirnos. Hemos aprendido a utilizar estos sitios casi sin darnos cuenta, haciendo uso de interfaces que, en general, podemos entender y dominar sin grandes dosis de esfuerzo o atención.

Los elementos que hoy componen una página web y su diseño han evolucionado a través del tiempo. En sus comienzos, surgieron gracias al uso de metáforas basadas en el cine, la televisión, los libros o las galerías de arte, del mismo modo en que anteriormente los sistemas operativos tomaron la metáfora del **escritorio**. El uso de estos recursos ayudó a los usuarios a familiarizarse con las funcionalidades de estos productos interactivos y marcó el inicio de una disciplina que, tiempo más tarde, se convertiría en un trabajo interdisciplinario, frenético y lleno de inventiva.

UN RECORRIDO POR LA HISTORIA DEL DISEÑO WEB

El diseño web comenzó a dar sus primeros pasos en la década del 90, y hoy, a casi 20 años de su nacimiento, podemos decir que es una disciplina que ha tenido un crecimiento agitado, colmado de cambios, posibilidades, búsquedas e innovaciones. Esta metamorfosis fue la respuesta al ritmo vertiginoso de las necesidades y las tendencias que fueron mostrando los usuarios de Internet y a la evolución de

los factores tecnológicos que atravesó el medio web, entre los que podemos mencionar los siguientes:

- Avances en la velocidad y el tipo de conexión a Internet.
- Evolución del hardware: monitores, tarjetas de video y procesadores, entre otros componentes.
- Crecimiento del software, principalmente, el que entrega para su uso libre.
- Estandarización de los lenguajes según las normas W3C (World Wide Web).
- Masificación en el uso de los buscadores.
- Aparición de herramientas y aplicaciones web que ayudan a mejorar los procesos, los tiempos y la calidad de producción.

A partir de ahora, delinearemos los diferentes períodos del diseño web y analizaremos algunos ejemplos característicos de cada uno.

PRIMEROS PASOS

En 1991, Tim Berners-Lee publicó la primera página web. Su contenido estaba conformado únicamente por **texto** e **hipervínculos** que hacían posible la navegación entre páginas. Esta era una pequeña referencia sobre qué era la **World Wide Web** y qué se podía hacer en ella, y en menos de un año, recibió dos millones de visitas.

FIGURA 4. La primera página web, publicada en 1991 por Tim Berners-Lee, estaba compuesta solo por texto e hipervínculos.

En esta primera etapa, el foco estaba puesto en la tarea de enviar o recibir datos. Las páginas solían ser hechas por programadores, y se caracterizaban por tener una interfaz lineal y limitada. Su diseño estaba generalmente diagramado a una sola columna y no se observaba el uso de cuadrículas que ayudaran a organizar la información.

FIGURA 3. La evolución del hardware incide en el avance de otras tecnologías. Un monitor monocromo no nos permite disfrutar de imágenes, por ejemplo.

El acceso a Internet se efectuaba, principalmente, a través de conexiones **dial-up**, con una velocidad de navegación muy lenta, factor limitante del peso que podían tener los archivos. Por eso, las páginas de esa época contenían muy pocas imágenes, en general, dispuestas en orden consecutivo.

CREACIÓN DEL WORD WIDE WEB CONSORTIUM (W3C)

En 1994, el físico inglés Tim Berners-Lee fundó el consorcio W3C, organismo destinado a establecer reglas y pautas que ayudarían a unificar criterios y a definir el futuro del diseño web.

En 1995 se publicó el estándar 2.0 de **HTML** (HyperText Markup Language), que a pesar de su nombre, es el primer estándar oficial de ese lenguaje. Las primeras versiones de HTML solo permitían dar a un texto el formato de encabezado, de párrafo y de hipervínculo (mediante **etiquetas** básicas). En las versiones posteriores, se incorporó el uso de imágenes y de tablas, y de la mano de estos avances, fueron apareciendo varios navegadores web.

FIGURA 5. **Logo del W3C (www.w3c.org), que aún hoy es de gran importancia para el diseño web.**

El diseño web se convirtió en un trabajo interdisciplinario, frenético y lleno de inventiva.

ICONOS, BOTONES Y BANNERS

El período en el cual aparecieron estos elementos comenzó alrededor del año 1996, momento en el que sucedieron varias cosas simultáneamente; veamos:

- Desarrollo de nuevos recursos gráficos: **iconos** que reemplazan palabras, **banners** que conforman cabeceras, **botones** con volumen que invitan a la interactividad y **fondos** que invaden las páginas. En el afán de experimentar y aprovechar los nuevos recursos, los desarrolladores web, conocidos en esa época como **webmasters**, empezaron a incluirlos de manera excesiva en sus diseños, lo que dio origen a páginas recargadas.

HTML

HTML (HyperText Markup Language) es el lenguaje básico con el que se escribe la mayoría de las páginas web actuales. Está compuesto por etiquetas delimitadas por paréntesis angulares (<,>), que se encargan de describir la estructura y el contenido.

• Gracias a la definición de estándares HTML, se dieron los primeros pasos hacia la escritura de código semántico, compuesto por etiquetas bien estructuradas que describen el contenido.

FIGURA 6. **Página principal del buscador AltaVista, lanzado en 1995. Fue el primero que permitió realizar búsquedas de texto sobre una base de datos.**

• Para ordenar la información, comenzó a implementarse el uso de **tablas**, uno de los elementos más polémicos de HTML. Con este elemento, la diagramación de las páginas se enriqueció sobre las bases del

diseño editorial (libros y revistas) e hizo uso de múltiples **columnas**. Aparecieron las primeras aplicaciones web basadas en el uso de tablas para la creación de páginas, lo que fomentó la adopción de la técnica por parte de los diseñadores. Si bien algunos desarrolladores la siguen utilizando, se trata de una técnica obsoleta y nada recomendable, concebida originalmente para mostrar información tabular.

• Esta también fue la era del **spacer.gif**, una imagen transparente e invisible conformada por un píxel cuadrado. Su uso ganó popularidad rápidamente al permitir forzar y controlar los espacios vacíos dentro de una tabla, necesarios para separar los contenidos.

• En el terreno del hardware, se avanzó en términos de resolución y definición del color, tanto en los monitores como en las tarjetas gráficas. Esto trajo importantes mejoras en la calidad del diseño web.

• Comenzó a vislumbrarse lo que posteriormente se conocería como **guerra de los navegadores**, con

CSS

Aunque lo conoceremos y utilizaremos más adelante en esta obra, vale aclarar que CSS es el lenguaje de hojas de estilo en cascada creado para controlar el aspecto visual de los documentos HTML. Es la mejor forma de separar los contenidos y su presentación.

las primeras diferencias en la adaptación de los estándares por parte de los dos principales browsers: Internet Explorer y Netscape Navigator.

• A finales de 1996, W3C publicó la primera recomendación oficial de CSS (**Cascading Style Sheets**), conocida como **CSS nivel 1**, cuya adopción formal se produjo más adelante.

LA IRRUPCIÓN DE FLASH

Unos años más tarde, una aplicación surgida con el nombre **FutureSplash Animator** (hoy Adobe Flash) ganó popularidad entre los desarrolladores. Este software, que nació con una interfaz sencilla compuesta por una **línea de tiempo** y herramientas muy básicas, progresó hasta convertirse en una aplicación que permitió controlar el diseño y animar páginas web sin las limitaciones del código HTML de la época.

La versatilidad de **Flash** permitió crear páginas dotadas de animación e interactividad, gracias a la edición de **fotogramas** y a la manipulación de objetos por separado. Hay quienes opinan que esta herramienta contribuyó a la evolución del diseño web, aunque muchos desarrolladores también sostienen que su uso desmedido no hizo más que degenerarlo.

En 1998 se publicó la segunda recomendación oficial de CSS, conocida como **CSS nivel 2**. La versión de CSS que utilizan los navegadores actuales es CSS 2.1, una revisión de CSS 2 que aún se está elaborando. La siguiente recomendación de CSS, denominada **CSS nivel 3**, continúa en desarrollo desde 1998, y hasta el momento solo se han publicado borradores.

EL DISEÑO WEB ACTUAL

A partir de la aparición de nuevos navegadores, de las mejoras en los existentes, de la evolución en la velocidad de las conexiones y del desarrollo de nuevos dispositivos de navegación (celulares, PDA y consolas de juegos, entre otros), surgió una nueva generación de desarrolladores que deben tener en cuenta

FIGURA 7. Interfaz de FutureSplash Animator (el precursor de Flash), utilizado para crear animaciones sencillas basadas en el uso de vectores.

FIGURA 8. **Página principal de uno de los sitios realizados en Flash más populares de la época: EYE4U.**

tanto las restricciones como las múltiples posibilidades existentes a la hora de crear sitios.

La necesidad de reproducir los contenidos en los nuevos dispositivos y los avances constantes en los estándares web produjeron un nuevo impulso de sitios basados en **XHTML** (**HTML** y **XML**) combinado con **CSS** para separar cada vez más el **diseño** del **contenido**.

Por otro lado, la adopción de **JavaScript** (un lenguaje de programación sencillo capaz de crear contenidos interactivos) para la creación de **transiciones** relegó el uso de Flash a sitios de animación cada vez más específicos o al uso puntual de la tecnología en elementos como reproductores de video o banners.

A lo largo de los años, los webmasters fueron reemplazados por equipos interdisciplinarios de trabajo compuestos por programadores, arquitectos de la información, maquetadores y diseñadores web. Todos ellos construyen sitios basándose en las mejores prácticas, que también evolucionan continuamente.

La colaboración fluida entre pares, el acceso a tecnologías web de código abierto (gestores de contenidos), la disponibilidad de recursos gráficos en línea, y la incorporación de herramientas de testeo y de desarrollo modificaron el trabajo del diseñador web, y elevaron a un nivel muy superior la calidad de los productos finales.

En la actualidad, vemos que el trabajo realizado por el diseñador web alcanzó un nivel muy elevado y mejoró la calidad de los productos finales.

Tipología de los sitios web

Producto del avance tecnológico y del creciente nú mero de personas que acceden a Internet, aparecie ron variadas herramientas y servicios que es posibl utilizar online. Gracias a esto, muchos aspectos d nuestra vida cotidiana se ven directamente influer ciados por la proliferación de los sistemas web, com nuestros hábitos de compra, de pago de servicios de comunicación, entre otros.

Para ofrecer estas utilidades, existen diversos tipos de sitios, que conoceremos a continuación.

SITIOS DINÁMICOS

Aunque podríamos pensar que un sitio dinámico es aquel que tiene animaciones o que es veloz en algún aspecto, este concepto se refiere al sitio en el que, al hacer clic para acceder a una página, esta se arma según nuestro pedido. En los sitios estáticos, en cambio, las páginas están armadas, esperando que alguien acceda a ellas.

El **armado a pedido** es realizado por dos partes: el motor de armado y la información con la que este genera la página. El **motor de armado** está creado con un lenguaje de programación, y la **información** suele provenir de una base de datos. Las principales ventajas de un sitio dinámico son la facilidad de actualización y carga de datos.

Encontramos sitios dinámicos en los casos que mencionamos a continuación:

- **Foros**: plataformas donde los usuarios escriben consultas, comentarios y respuestas que son almacenados en la base de datos.
- **Blogs y microblogs**: sistemas que nos permiten tener un espacio personal, a partir de una plantilla prediseñada, en el que podemos compartir y comentar información como **música** (www.myspace.com), **videos** (www.youtube.com), **imágenes**

WORDPRESS

Dentro del listado de blogs y microblogs que analizamos, WordPress merece una mención extra, ya que ofrece, en forma gratuita, los archivos fuente que conforman el blog para descargar, instalar y personalizarlos completamente, y así utilizar el blog en cualquier dominio y hosting.

(www.fotolog.com y www.flickr.com), textos y multimedia (www.blogspot.com y http://wordpress.com), y textos reducidos (http://twitter.com).

- **Wikis**: plataformas de creación de contenidos en forma participativa, como la enciclopedia http://wikipedia.org o http://commons.wikimedia.org, un repositorio de archivos de uso gratuito.
- **Portales de noticias**: sitios con contenidos generados, comúnmente, por periódicos de edición impresa. En este caso, los usuarios no pueden encargarse de editar los contenidos, pero sí podrán dejar comentarios en algunas noticias.
- **E-commerce**: sitios como www.ticketek.com y www.amazon.com ofrecen entradas a espectáculos o productos como libros y películas en sus páginas, donde el usuario puede elegir, entre otros detalles, la cantidad y el tipo de artículo, la forma de envío y la de pago, que suele ser mediante tarjeta de crédito.
- **Redes sociales**: plataformas en las que podemos generar un perfil con datos personales, intereses, profesión y educación. A partir de él, es posible contactarnos con personas que tengan intereses similares o nuestra misma profesión, por ejemplo. También permiten agregar álbumes de fotos, videos, links y comentarios, estos últimos, tanto en el perfil propio como en el de los contactos.

APLICACIONES WEB

Muchos sitios ofrecen **aplicaciones online** en forma gratuita, entre las que encontramos editores de texto, de sonido y de imágenes; planillas de cálculo y agenda. La última tendencia es que las aplicaciones no estén instaladas en nuestras computadoras sino disponibles como un **servicio online**. El ejemplo más referencial es Gmail.com, que ofrece muchos de estos servicios y, además, permite personalizar colores, logos, fondos y la disposición de las interfaces.

INTERCAMBIO Y ALOJAMIENTO DE ARCHIVOS

La calidad de las imágenes, los videos y el audio disponibles en las computadoras ha aumentado, lo que provoca cambios en el modo de distribuir y transferir estos archivos, cada vez más pesados. Sitios como **www.rapidshare.com**, **www.dropbox.com** o **www.yousendit.com** nos permiten almacenar y enviar archivos de gran tamaño.

SITIOS ESTÁTICOS

Son sitios donde los usuarios no pueden crear ni editar los contenidos. Los casos más comunes de este tipo son los siguientes:

- **Sitios experimentales**: en estos casos se busca generar nuevas experiencias de navegación para el usuario, al producir interacciones no convencionales (**http://dontclick.it**), recorridos 3D (**www.vatican.va/various/cappelle/sistina_vr/index.html**) o navegación mediante la cámara web (**www.davidlindseywade.com/#/portfolio/portfolio/19**).
- **Newsletters y flyers**: son minisitios publicitarios que están alojados en un servidor y son enviados por e-mail con el fin de promover algún producto o servicio.

Los avances de la tecnología generan nuevos modos de transmitir archivos y novedosos servicios para los usuarios.

FIGURA 9. **En este portfolio de fotografías, podemos ir navegando de una imagen a otra interactuando con la cámara web conectada a nuestra computadora.**

Tecnologías de los sitios web

Es habitual que a diario naveguemos por diversos sitios para revisar nuestro correo electrónico, leer las noticias, o hacer alguna compra o pago virtual. A simple vista, todos los sitios que encontramos parecieran estar hechos de la misma manera, aunque si hacemos un análisis en detalle, descubriremos que en ellos intervienen diversas tecnologías que se combinan de diferentes modos para lograr las funcionalidades y prestaciones que brindan al usuario. Conozcamos las principales.

LENGUAJES DE EJECUCIÓN DEL LADO DEL USUARIO

En estos sitios, cuando el usuario accede a una página, los archivos que la componen son **descargados** e **interpretados** directamente por el navegador (Firefox o Internet Explorer). Pueden estar creados con las siguientes tecnologías:

HTML y XHTML: la mayoría de los sitios por los que navegamos se crean con estos lenguajes. Estas páginas se componen por una estructura de **etiquetas** que tienen significados semánticos (títulos importantes, párrafos o imágenes) para que los buscadores (como Google o Bing, por ejemplo) vinculen un

```
<h1>.: Portfolio, Dise&ntilde;o Web</h1>

<div id="trabajos">

  <div class="work" id="workA">
    <h1>Diseño web, Auto Neumen</h1>
        <ul>
          <li><a href="http://www.diegozamora.com.ar/neumen" rel="nofollow" target="_blank">Ver online</a></li>
          <li><a href="portfolioWeb/sitioWebNeumen.html" target="_self">Ver detalles</a></li>
        </ul>

        <br class="clear" />

      <p>Desarrollo integral de dise&ntilde;o y programaci&oacute;n del sitio web, tambi&eacute;n se realiz&oacute; un ajuste en la marca para la imagen corporativa y un nuevo dise
&ntilde;o en la comunicaci&oacute;n en vía p&uacute;blica.</p>
    </div>

      <div class="work" id="workB">
        <h1>Dise&ntilde;o web, New Fast Copy</h1>
          <ul>
            <li><a href="http://www.newfastcopy.com.ar" rel="nofollow" target="_blank">Ver online</a></li>
            <li><a href="portfolioWeb/sitioWebNewFastCopy.html" target="_self">Ver detalles</a></li>
          </ul>

          <br class="clear" />

        <p>Se realiz&oacute; tanto el Dise&ntilde;o como la programaci&oacute;n del sitio web. Se trata de una empresa de servicios empresariales de primera l&iacute;nea y con gran
experiencia y velocidad, todos valores que fueron contemplados a la hora de diseñar la interactividad y la interfase. </p>
      </div>
```

FIGURA 10. **Aquí se ve el código con el que está conformada parte de una página, con etiquetas que indican títulos principales, párrafos, links e imágenes.**

HTML vs. XHTML

Aunque lo veremos en detalle más adelante, la diferencia entre HTML y XHTML radica, fundamentalmente, en que el último requiere una sintaxis más estricta en comparación con el primero, en el que pueden omitirse las etiquetas de cierre, por ejemplo, sin que esto provoque fallas.

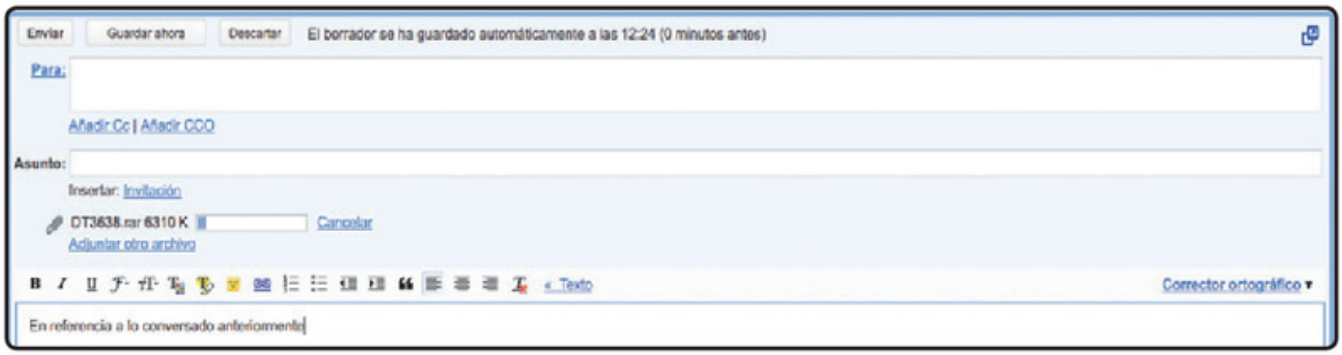


FIGURA 11. **Gmail permite adjuntar un archivo, ver la progresión de la carga y escribir en el cuerpo del correo al mismo tiempo que se produce un autoguardado del mensaje en borradores. Todo facilitado por AJAX.**

determinado texto existente en la página al significado semántico que le da la etiqueta que lo contiene.

JavaScript/AJAX: es un lenguaje de programación usado para generar interfaces de usuario mejoradas, con más interactividad. Las sentencias y funciones de este código pueden estar en el mismo archivo HTML o en un archivo externo.

AJAX es una técnica que se usa vinculando JavaScript a XML. JavaScript requiere la información alojada en el servidor en formato XML para generar cambios o agregar más funcionalidades en la página, sin interferir con su visualización ni comportamiento.

Flash y ActionScript: Flash se asomó al mundo de Internet como un programa destinado a generar vistosas animaciones vectoriales que se podrían incorporar en las páginas. En su evolución, desarrolló un potente leguaje de programación que, hoy en día, permite crear sitios y aplicaciones web dinámicas, con conectividad hacia otros lenguajes y plataformas. También brinda la posibilidad de generar entornos con gran interactividad e impacto visual, además de manejar audio y video.

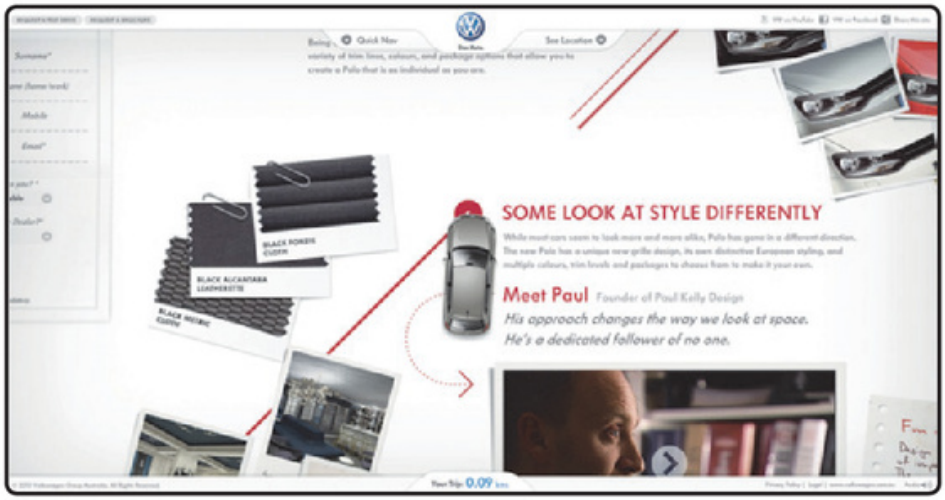


FIGURA 12. **Este sitio de alto impacto visual permite tener interactividad tanto con el mouse como con el teclado.**

ALMACENAMIENTO DE DATOS

Cuando se genera un sitio dinámico, es necesario contar con un repositorio de información. Para esto, disponemos de algunas alternativas que mencionamos a continuación:

XML (Extensible Markup Language): es un lenguaje de sintaxis similar a HTML, pero no tiene etiquetas predefinidas con significados particulares, sino que estas son establecidas por el usuario de acuerdo con la información que necesite almacenar. Suele usarse vinculado a Flash y ActionScript.

Bases de datos: una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En el entorno web, las bases más utilizadas son SQL (SQL Server pertenece a Microsoft) y MySQL (Open Source). Se las usa para almacenar los datos que componen un sitio web, tales como textos e imágenes, entre otros. Suelen utilizarse vinculadas a ASP y PHP.

LENGUAJES DE EJECUCIÓN DEL LADO DEL SERVIDOR

Son lenguajes que ejecuta el servidor para crear una página cuando el usuario accede a un sitio. Es decir, buscan la información en una base de datos y, tomándola como referencia, generan archivos HTML que serán finalmente visualizados por el usuario. Conozcamos las alternativas más difundidas:

- **PHP**: es un lenguaje de programación al que pueden agregarse diversas librerías para otorgarle funcionalidades y usarse sin costo alguno. Es compatible con la mayoría de los servidores web y sistemas operativos, aunque suele implementarse en entornos Linux y con bases de datos MySQL, lo cual evita por completo los costos de licencias.

- **ASP**: es un lenguaje de programación creado por Microsoft, que se ejecuta solo en servidores Windows. Su funcionamiento es igual al de PHP pero se utiliza, en general, vinculado a bases de datos SQL. La desventaja que presenta es que no es

FIGURA 13. Wikipedia es un entorno colaborativo creado en PHP, donde los usuarios crean la información, que es almacenada en base de datos y, luego, mostrada ante el requerimiento de cada persona.

gratuito y que debe implementarse en un entorno 100% Windows, que también tiene un costo.

APLICACIÓN DE ESTAS TECNOLOGÍAS

Un caso en el que podemos ver muchas de estas tecnologías vinculadas es **YouTube**. En él se utiliza un reproductor con funciones especiales de video, creado en **Flash**, y en la interfaz de la página se pueden postear nuevos videos, crear perfiles y dejar comentarios.

Por su parte, en los sitios experimentales que generan nuevas situaciones de interacción y de experiencia con el usuario, al mismo tiempo que buscan un gran impacto visual sin limitaciones en cuanto al diseño, tanto la interfaz como las situaciones interactivas son creadas en Flash y ActionScript de forma estática. Claro que Flash también puede leer lenguajes de marcado y bases de datos para generar sitios dinámicos.

Si hacemos un análisis en detalle, descubriremos que en un sitio intervienen diversas tecnologías.

La estructura **de las páginas**

Es hora de comenzar a presentar los aspectos relacionados con los elementos que conforman una **interfaz**, la manera en la que estos se estructuran, y los criterios que debemos seguir para realizar una página atractiva y funcional.

Cada sitio web tiene un **objetivo comunicacional** y, debido a eso, todos sus aspectos visuales y la articulación de sus elementos deben garantizar que se cumpla ese objetivo.

Es necesario que el internauta comprenda el objetivo del sitio en pocos segundos. El tiempo es tirano, y el navegante está expuesto a una innumerable cantidad de sitios, una avalancha de información que hace imprescindibles la claridad del mensaje y la buena legibilidad.

LA FORMA DE NAVEGACIÓN

A la hora de diseñar un sitio, tenemos la libertad de elegir qué forma de navegación utilizaremos, pero debemos hacerlo con criterio y adecuándonos a cada caso en particular. Veamos, a continuación, algunos

YOUTUBE

En YouTube podemos ver que el reproductor de video y sus controles están creados con Flash y ActionScript. El resto del entorno visual es HTML, pero tanto la información de los videos relacionados como los comentarios dejados por los usuarios están almacenados en una base de datos, lo que indica que se trata de un sitio creado con PHP o ASP.

sitios que proponen formas novedosas de navegación, con intervenciones poco comunes por parte de los usuarios.

- **Navegación sin clics**: **www.dontclick.it** propone una navegación sin clics, haciendo la elección de lo que deseamos con solo aguardar unos segundos sobre el elemento al que queremos acceder.
- **Navegación vertical**: todo el contenido se presenta en una sola página, que se mueve hacia arriba o hacia abajo según las opciones que vayamos eligiendo. Podemos ver ejemplos en **www.volll.com**, **www.kitfolio.com** y **www.pojeta.cz**.
- **Navegación original**: en **http://doorstepdairy.com** encontramos que la navegación se realiza mediante el movimiento de un elemento, en este caso, un pequeño camión que debe arrastrarse con el mouse.

Si imaginamos una interfaz de navegación con movimiento vertical para un sitio con muchas secciones y que ofrezca varios productos, el largo necesario para la página sería excesivo. Si nos moviéramos del último producto al comienzo, la sensación al ver pasar palabras, imágenes y colores a gran velocidad quizás resultaría desagradable. Esto nos demuestra que no todos los recursos disponibles son aplicables a cualquier sitio, y si a pesar de eso los utilizamos, tal vez el usuario no tenga una buena percepción de nuestra propuesta.

FIGURA 14. **Todo el contenido del sitio está ubicado en una sola página muy larga, que se muestra por partes en la ventana del navegador.**

FIGURA 15. En este sitio aparecen diversas opciones a medida que arrastramos el pequeño camión con el mouse.

¿QUÉ ES LA ESTRUCTURA?

Nadie espera encontrar un menú en el medio de un párrafo, ni tampoco un buscador o un logo en el pie de la página. Aunque sea de forma inconsciente, todos esperamos que los elementos estén dispuestos de cierta manera, donde los buscamos en primer lugar cuando accedemos al sitio.

Un usuario tiene la capacidad de predecir el sitio, buscando de forma rápida lo que le interesa. Luego de reconocer la disposición de los elementos en la página de inicio, los seguirá buscando en el mismo lugar dentro de las diferentes secciones. Por eso, necesitamos mantener estándares de estructura para darle una coherencia visual a la distribución del contenido en todas las páginas. Por supuesto que también podremos tomarnos pequeñas libertades de diseño para darle un ritmo especial a la navegación.

ENCABEZADO O HEADER

El **header** es uno de los aspectos principales que encontramos en un sitio web, ya que es lo primero que vemos. Ocupa todo el ancho superior del sitio y es uno de los elementos que se mantienen casi sin ninguna variación dentro de todas las páginas. Veamos, a continuación, los elementos que pueden estar presentes dentro del header de un site:

Todos los elementos y los aspectos visuales de un sitio deben garantizar que se cumpla la comunicación

• **Logo**: dentro del header se sitúa el logotipo, es decir, la identidad gráfica, el sello distintivo de una empresa. Como es fundamental diferenciarse, el logo ocupa un lugar protagónico en el extremo superior izquierdo. Esto no es casual: está comprobado mediante estudios que este es el primer lugar donde los usuarios dirigen su mirada.

Además de ser un elemento gráfico importante, al logo se le otorga una funcionalidad extra: posee un enlace al inicio del sitio, convirtiéndose en un atajo para que el usuario pueda volver a la página principal.

• **Menú**: en el header también encontramos la **barra de navegación**, que contiene los enlaces para acceder a las diferentes secciones del sitio. Los elementos del menú deben estar expuestos de modo sencillo, intuitivo y bien visible para que el usuario tenga acceso rápido a la información y siempre encuentre lo que busca.

Cuando un sitio tiene pocos enlaces, todos pueden estar en el menú principal. En cambio, en uno más complejo (de más de seis o siete páginas), aparece el **menú principal** para navegar por las secciones más importantes y, luego, **submenús** para moverse dentro de cada una. Estos submenús deben mantener coherencia temática y lógica en su agrupación. Por

FIGURA 16. Cada vez son más los sitios con diseños de vanguardia que le otorgan un papel protagónico al header.

FIGURA 17. Plantear una buena organización de contenidos da como resultado una navegación clara.

ejemplo, si tenemos un sitio de venta de productos de electrónica, a nadie se le ocurriría buscar los productos dentro de la sección Contacto.

• **Menú de rastros**: como mencionamos antes, este menú es recomendable en los sitios que tienen muchas secciones y subsecciones, ya que informa al usuario su ubicación dentro del sitio. Por ejemplo, si estamos en la subsección **Aventuras**, dentro de la subsección **Playstation 3**, que a su vez está dentro de la sección **Videojuegos**, deberíamos de informarle al usuario algo como **Videojuegos/Playstation 3/Aventuras**. Esto representa de una manera visual la jerarquía del sitio y, en la mayoría de los casos, lo encontramos antes del contenido principal.

Es importante que el usuario pueda identificar de inmediato la sección en la cual se encuentra.

• **Identificación de secciones**: es importante que el usuario sepa en qué sección se encuentra y que pueda identificarla de inmediato. En combinación con el menú de rastros, la identificación de secciones debería de tener mayor relevancia tanto en su ubicación como visualmente, y por lo general, se la coloca por debajo del menú principal de secciones, como encabezado de la sección a la que identifica.

• **Sliders**: este recurso interactivo ha ganado terreno gracias a las nuevas tendencias del diseño 2.0, donde los headers son cada vez más grandes, y la gráfica, cada vez más iconográfica. Los sliders captan la

mirada por su movimiento e interactividad, mostrando un mensaje de forma simple, rápida y eficaz. Para hacerlo, se valen de iconos, colores e información bien jerarquizada, con un diseño simple y atractivo.

FIGURA 18. **Modernos, prácticos y muy utilizados, los sliders dan dinamismo y transmiten mensajes claros de una manera efectiva.**

• **Buscadores internos**: en sitios grandes, estos buscadores suelen ubicarse arriba a la derecha o en los sidebars. Sirven para encontrar de forma rápida algo puntual. Esto quiere decir que no se accede a la información a partir de la navegación, sino que se ingresa una palabra exacta o aproximada a lo que se desea encontrar y se presiona el botón **Buscar** o similar para obtener los resultados.

CONTENIDO

Como mencionamos, está comprobado que el recorrido visual que realiza el visitante comienza por el extremo superior izquierdo (donde es usual que encontremos el logo) y luego se fija en el centro de la página, donde está el contenido principal.

Aunque la organización varía entre diseños, veremos denominadores comunes para jerarquizar los diferentes tipos de información. El uso de destacados es un gran punto de tensión, así como también los

UN BUEN DISEÑADOR

Aunque es posible decir que un buen diseñador web es aquel que sabe utilizar las herramientas con las que desarrolla sus proyectos, este es solo un aspecto técnico que no resulta fundamental para tener éxito. La buena fortuna de un diseñador web depende de sus conocimientos.

títulos, que son resaltados mediante el uso de fondos, bullets, iconos, u otros elementos, para luego continuar por la lectura de párrafos o de información secundaria.

BARRA LATERAL O SIDEBAR

Este elemento gráfico sirve para organizar contenidos importantes del sitio. Se puede ubicar a la derecha del cuerpo principal, a la izquierda o en ambos lados. Contiene enlaces externos e internos,

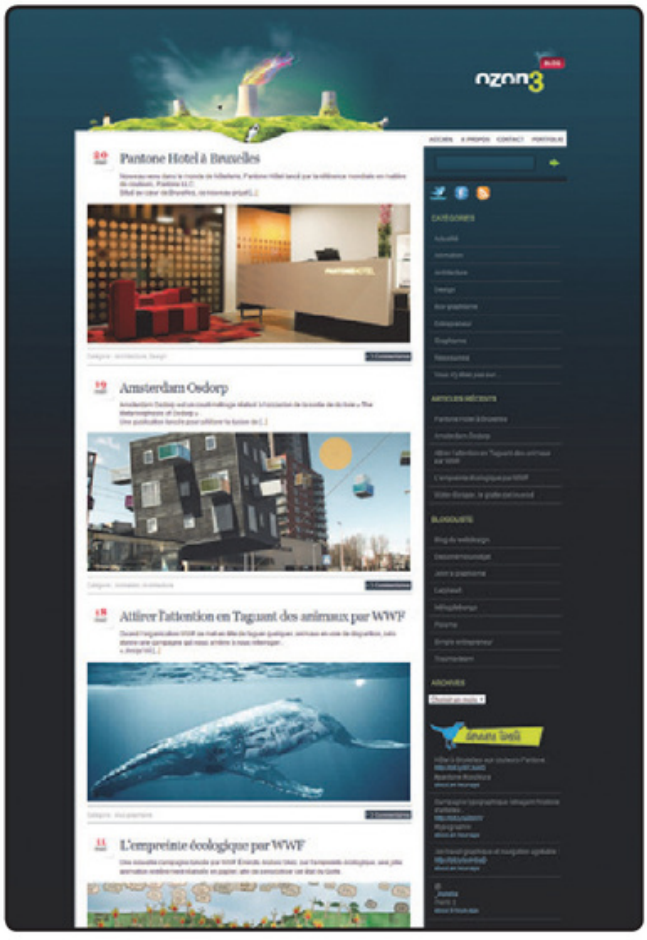
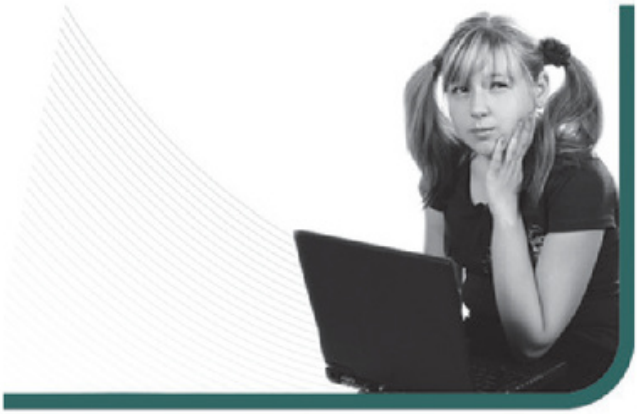


FIGURA 19. **Los sidebars son infaltables en las redes sociales y en los blogs, donde su rol es claramente organizativo, debido a la gran cantidad de información que se necesita comunicar.**



información adicional organizada por categorías y, actualmente, se utiliza también para incluir publicidades (banners, links, etcétera).

PIE DE PÁGINA O FOOTER

Así como los headers han dado un salto de lo tradicional a lo vanguardista, los footers también siguieron esos pasos. Pasaron de contener información como políticas de privacidad de uso y derechos de autor, a ser contenedores de elementos relevantes, como enlaces, mapas de sitios (para ayudar a la indexación en Google), información y formularios de contacto, y los infaltables links a las redes sociales, que nos invitan a unirnos, compartir, ver perfiles y hasta visualizar los últimos posts de Twitter.

Todos los elementos mencionados determinan lo que llamamos una **interfaz web**. Los contenidos, la forma de navegación, los elementos de identificación y las acciones que podemos generar dentro de un sitio son parte de su interfaz.

Al crear una interfaz, es importante que, para transmitir un mensaje claro, rápido y de forma sencilla, respetemos ciertas convenciones de estructura e innovemos desde el diseño y las tecnologías.

FIGURA 20. Actualmente, los footers son algo más que un buen cierre para el sitio. Su uso y actualización marca tendencia.

La tarea más difícil de un buen diseñador web no radica en aprender a utilizar una aplicación, sino en saber utilizarla con criterio y estar al tanto de todo lo que influye en su medio de trabajo, como la aparición de nuevas técnicas de diseño, cambios en las tendencias y en los estilos, o nuevos dispositivos y utilidades que simplifiquen su tarea. Debe saber utilizar los programas más conocidos de diseño y determinar en qué momento conviene uno u otro.

CONOCIMIENTOS

Como vemos, un diseñador necesita contar con muchos conocimientos para poder crear un sitio de calidad. En esta obra, veremos todas esas herramientas e, incluso, obtendremos los mejores consejos que nos brindarán la experiencia que necesitamos.

Multiple **choice**

1 ¿Qué conocimientos debe manejar un diseñador web?

a- Estéticos.
b- Estéticos, conceptuales y técnicos.
c- Estéticos, conceptuales y de diagramación.
d- Conceptuales y técnicos.

2 ¿Qué significa W3C?

a- Word Wide Web Consortium.
b- Word Wide Web.
c- Consortium Web.
d- Word Wide Consortium.

3 ¿Qué es el spacer.gif?

a- Una imagen transparente con forma redonda.
b- Una imagen de color negro.
c- Una imagen con forma redonda.
d- Una imagen transparente e invisible.

4 ¿Qué caracteriza a un sitio dinámico?

a- Qué su contenido se actualiza cada cierto tiempo.
b- Que posee videos.
c- Al acceder a una página, esta se arma según nuestro pedido.
d- Que contiene animaciones.

5 ¿Qué es un sitio estático?

a- Donde los usuarios no pueden crear ni editar contenidos.
b- Donde los contenidos no se actualizan en forma constante.
c- Sitios que no poseen animaciones.
d- Sitios que solo poseen texto.

6 ¿Qué espacio utiliza el header de un sitio?

a- Utiliza todo el ancho superior del sitio.
b- Usa el ancho inferior del sitio.
c- Hace uso de la zona central del sitio.
d- Utiliza la columna lateral izquierda del sitio.

Respuestas: 1-b, 2-a, 3-d, 4-c, 5-a, 6-a

Capítulo 2

Del diseño al HTML

En este capítulo conoceremos de qué forma podemos plasmar un diseño de sitio web utilizando el lenguaje HTML.

RedUSERS.com

Herramientas para **desarrollar HTML**

El proceso de crear páginas se denomina, en la jerga, **maquetación**, y da como resultado tener nuestro diseño convertido en un sitio navegable. Luego, en caso de que el proyecto lo requiera, faltará la adición de código JavaScript, AJAX, ASP o PHP para hacer que las páginas dinámicas obtengan contenido de las bases de datos.

Un problema que suele presentarse al desarrollar HTML es lograr que el diseño sea visualizado de igual forma en la mayoría de los navegadores de Internet. Este inconveniente se debe, principalmente, a que algunos fabricantes no han incorporado en sus productos los estándares que determinan la estructura de la Web. En consecuencia, tendremos que aplicar **parches** en el código CSS y HTML para que la maqueta sea compatible con los navegadores más frecuentes: Internet Explorer 6, 7 y 8; Mozilla Firefox 3.5 y 3.6; Chrome 5.0 y Safari 4.0. Esta práctica se denomina **cross-browsing**.

Es importante distinguir entre las distintas versiones de cada navegador, ya que cada uno interpreta el código de forma diferente. Si nos aseguramos de que nuestro sitio sea compatible con los siete navegadores mencionados, el 96% de los usuarios de Internet podrá ver sin problemas el trabajo que estamos desarrollando.

La maquetación será exitosa si el código respeta los estándares CSS de la W3C sin alterar el diseño original

HERRAMIENTAS PARA GENERAR CÓDIGO HTML Y CSS

La aplicación más potente del mercado para generar código HTML es Adobe Dreamweaver CS5, que comenzaremos a estudiar en las próximas páginas. Sin embargo, además de ella, existen otras que podemos utilizar para redactar y generar código CSS y HTML para nuestros sitios.

Si bien es cierto que cualquier editor de texto plano, como el Bloc de notas de Windows, es suficiente para redactar código HTML y CSS, nos convendrá recurrir a programas diseñados especialmente para tal fin, ya que en ellos tendremos ayudas visuales (como el uso de distintos colores para el texto) que nos orientarán en la tarea y simplificarán el trabajo.

NOTEPAD++

Notepad++ (**http://notepad-plus-plus.org**) es una aplicación que bien podríamos definir como una versión muy mejorada del editor de texto o el Bloc de notas incluido en todos los sistemas Windows. Si no estamos acostumbrados a redactar código de programación web **a ciegas**, no será conveniente utilizarlo, ya que su principal desventaja es que no cuenta con un preview incorporado. De todas maneras, desde un simple atajo de teclado, podemos visualizar el HTML en los navegadores de Internet que tengamos instalados.

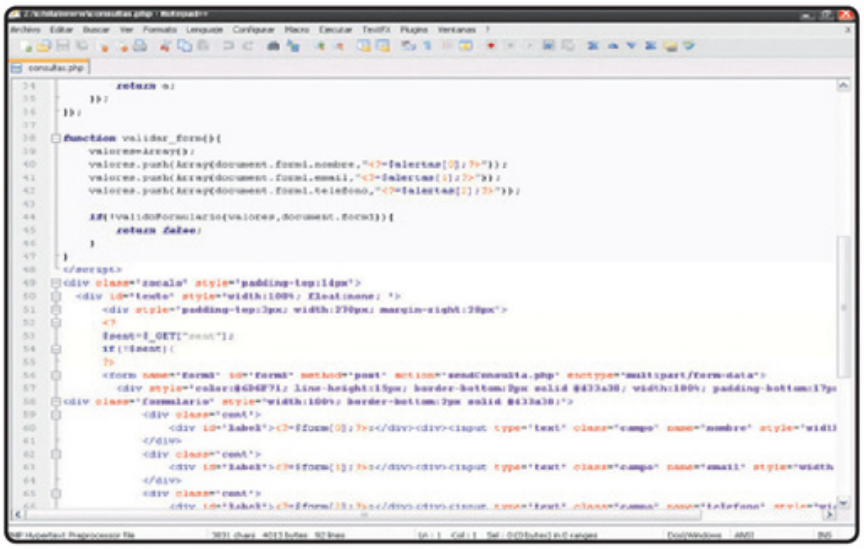


FIGURA 1. Interfaz de Notepad++, donde vemos como ejemplo el código de una página escrita en PHP.

Entre los principales puntos a favor de esta aplicación encontramos los siguientes:

- Es gratuita.
- Sintaxis coloreada.
- Numeración de línea.
- Muy liviana, consume pocos recursos del sistema.
- Código **autocompletable** en diferentes lenguajes.
- Es posible abrir y gestionar archivos de texto pesados (por ejemplo, logs de acceso a servidores) sin perder estabilidad.

TOPSTYLE PRO

TopStyle Pro (**www.topstyle4.com**) es uno de los mejores editores de código CSS y, en general, es recomendable utilizarlo como complemento del editor de HTML principal que empleemos para trabajar. Entre sus principales ventajas podemos mencionar:

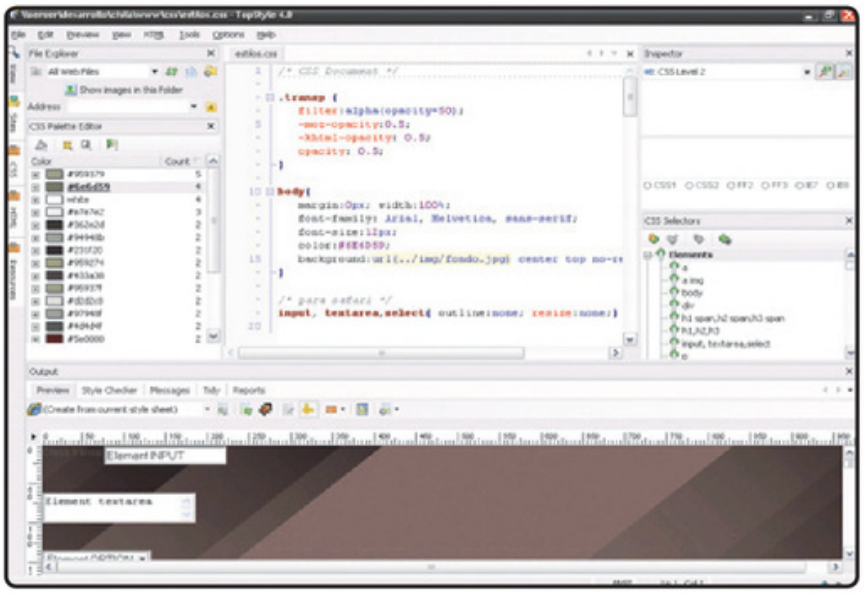


FIGURA 2. Área de trabajo de TopStyle Pro. El panel `CSS Palette Editor`, ubicado a la izquierda, muestra los colores utilizados en nuestra hoja de estilos y cuántas veces aparece cada uno.

- Preview del código escrito en tiempo real.
- Validador de estilos CSS.
- Sintaxis autocompletable del código que estamos escribiendo.

¿POR QUÉ DREAMWEAVER?

La elección de Adobe Dreamweaver como principal herramienta de trabajo se debe a que es una aplicación muy potente para desarrollar sitios web respetando los estándares CSS y HTML. Con ella, podemos diseñar sitios de forma visual o directamente sobre el código, y tenemos la posibilidad de trabajar con distintos entornos de programación, como XHTML, CSS, XML, JavaScript, AJAX, PHP y ASP.

FIGURA 3. **Dreamweaver es la aplicación de la suite Adobe Web Premium especialmente desarrollada para la creación de sitios web.**

Dreamweaver nos permite configurar sitios y, de esta manera, definir un ámbito de desarrollo y testeo para los diseños y aplicaciones web que generemos, sin necesidad de usar un programa externo para tal fin.

Además, esta nueva versión de la herramienta (CS5) cuenta con una funcionalidad muy destacable: el **asistente de codificación inteligente**. Su función es optimizar la velocidad de escritura de código, dado que el programa reconoce y sugiere **strings** de código en diferentes lenguajes de programación. Estas sugerencias aparecen tanto en funciones predefinidas del lenguaje como en otras personalizadas que hayamos usado en el mismo desarrollo.

Por otro lado, con Adobe Dreamweaver tenemos una ventaja simple pero fundamental sobre otros editores de HTML: podemos insertar tags corriendo solamente un comando del programa, como `Insertar/Imagen`.

Adobe **Dreamweaver CS5**

La flexibilidad es una de las cualidades más destacadas de Dreamweaver. Esta aplicación puede ser usada por aquellas personas que tengan poca experiencia en el armado de una página HTML, porque al insertar elementos desde los menús, el programa se encarga de redactar a la perfección el código HTML y CSS. También es útil para usuarios avanzados, ya que permite manipular manualmente los tags con asistencias de edición excepcionales, que aceleran el proceso de redacción de código.

Luego de iniciar Dreamweaver por primera vez, es recomendable definir la distribución del **espacio de trabajo**. Esto determinará la ubicación de los distintos menús y opciones dentro de la aplicación, y es una selección que quedará registrada para las próximas veces que lo usemos. Para hacerlo, nos dirigimos al menú **Ventana/Diseño del espacio de trabajo**.

La distribución llamada **Clásico** es sumamente flexible e intuitiva, y resulta familiar para quienes vienen trabajando con versiones anteriores de este programa. Sin embargo, cada usuario elegirá la que le resulte más adecuada.

EL ÁREA DE TRABAJO

Cuando abrimos un documento (o creamos uno nuevo, como veremos más adelante), por defecto aparece la pantalla dividida en dos zonas principales: **Código** y **Diseño**.

Como su nombre lo indica, en el área denominada **Código**, la cual se ubica en la parte superior, veremos el **código HTML** que corresponde a la página. Aquí también podremos escribir cualquier otro código de programación web que necesitemos (como ASP, PHP, JavaScript, etcétera). Por su parte, en la sección **Diseño**, que se encuentra en la parte inferior, se muestra la previsualización (**preview**) del código interpretado.

El resto de la pantalla se compone de la siguiente manera: en la parte superior encontraremos el menú y la barra de herramientas; en la derecha, la barra de ventanas; y en la parte inferior, la barra de propiedades HTML y CSS. Veremos estas secciones en detalle en las próximas páginas, donde nos encargaremos de conocer las partes más importantes que presenta la interfaz del programa.

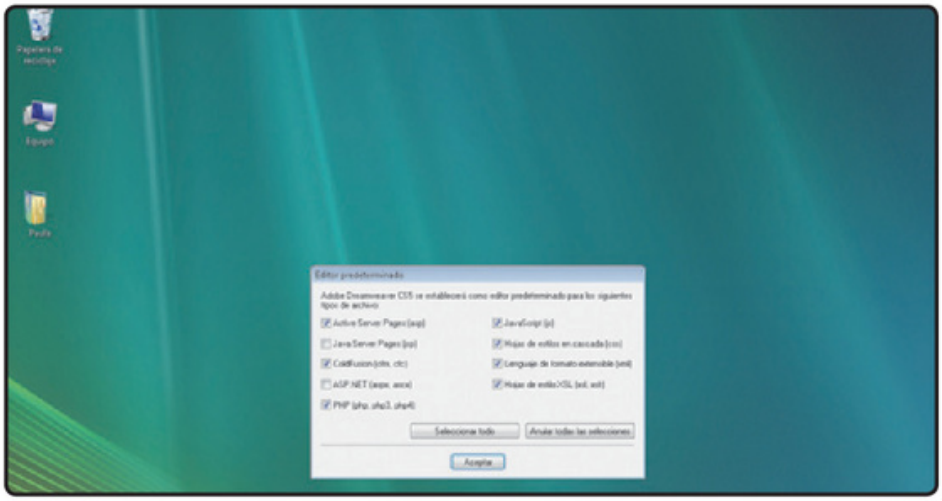


FIGURA 4. **La primera vez que ejecutamos Dreamweaver, debemos definir si será el editor predeterminado para los tipos de archivo que soporta.**

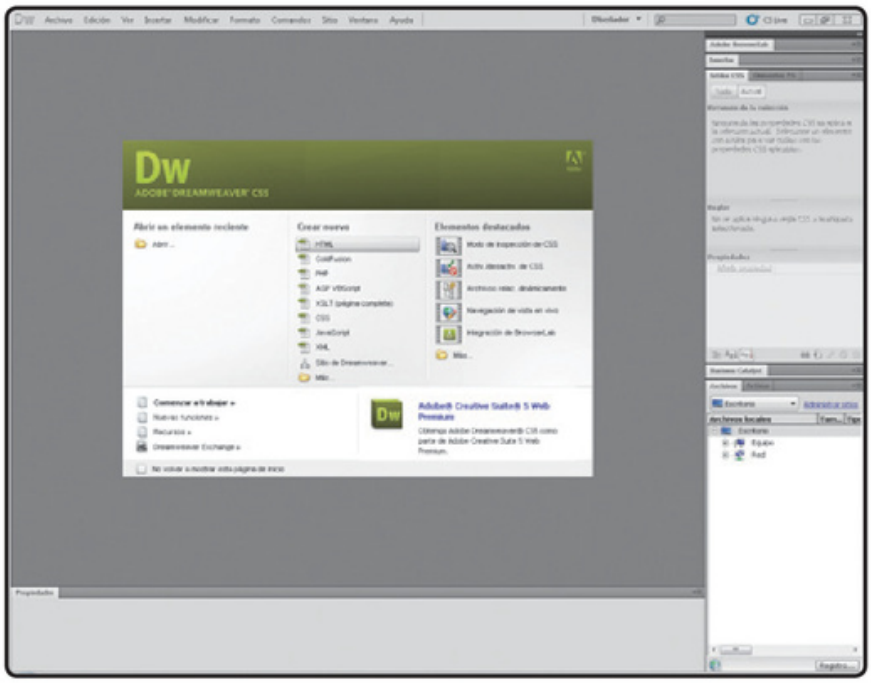


FIGURA 5. Al abrirse, Dreamweaver ofrece una lista de acciones que podemos realizar. Tenemos la opción de evitar que se abra este cuadro marcando la casilla ubicada en el ángulo inferior izquierdo.

NUEVAS CARACTERÍSTICAS

Adobe Dreamweaver CS5 incluye mejoras respecto a su antecesor; las más destacadas son las siguientes:

- **Inspección de hojas de estilo:** se pueden cambiar las propiedades CSS sin necesidad de escribir código, simplemente, desde un panel ubicado a la derecha del área de trabajo.
- **Integración con Adobe BrowserLab:** este es un servicio en línea de Adobe que permite probar y comparar nuestro desarrollo en diferentes navegadores de Internet.
- **Sugerencias de código PHP personalizado:** se trata de un asistente para escribir sintaxis adecuada de nuestras funciones personalizadas de PHP.

Aunque trabajaremos con CSS más adelante, vale destacar en este caso la nueva característica de Dreamweaver CS5 que realiza la **inspección de CSS** en tiempo real. Gracias a esta funcionalidad, es posible cambiar las propiedades de una hoja de estilos sin necesidad de interpretar código. Desde el panel **`Estilos CSS`**, podremos modificar colores y otros parámetros de manera muy sencilla.

INTEGRACIÓN ENTRE APLICACIONES DE LA SUITE

Además de las ventajas de cada uno de los programas de la suite Adobe Web Premium, trabajar con el paquete completo ofrece beneficios adicionales. Entre ellos, permite crear un objeto inteligente de imagen a partir de la inserción de un archivo de Photoshop o Illustrator.

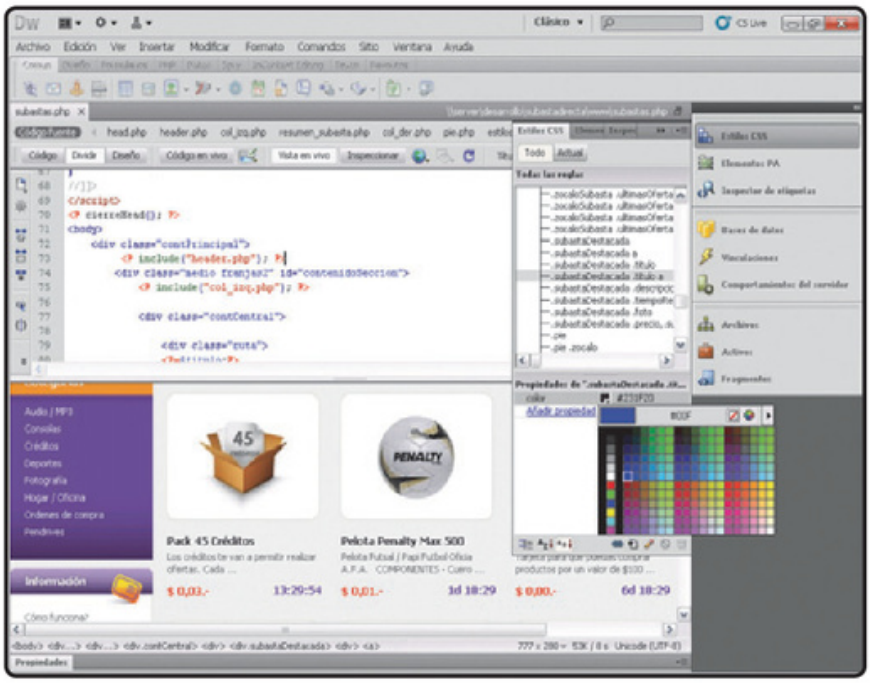


FIGURA 6. **Luego de seleccionar un estilo, buscamos la propiedad que queremos modificar y elegimos el nuevo valor que deseemos darle.**

PRESTAR ATENCIÓN AL CÓDIGO

Aunque podemos trabajar exclusivamente en la vista **Diseño**, es conveniente utilizar la pantalla dividida de Dreamweaver. De esta forma, a medida que hagamos el diseño de una página, también veremos el código que se va generando. En etapas de aprendizaje, esto nos ayudará a familiarizarnos con la sintaxis HTML y a relacionar cada uno de los elementos que insertemos con sus etiquetas y atributos HTML (veremos estos temas sobre el final de la clase). Más adelante, cuando tengamos más conocimientos, ver el código nos permitirá cuidar que este cumpla con los estándares definidos por la W3C (**www.w3.org**).

Si bien este es un tema que veremos en detalle en la próxima clase, cabe adelantar que, al trabajar con la interfaz dividida, encontraremos que algunas palabras comienzan con el signo **&** (**ampersand**). Tendremos que familiarizarnos con estos caracteres para poder leer HTML con naturalidad, ya que se utilizan para indicarle al navegador el uso de algunos símbolos o palabras especiales.

¿Qué es **HTML**?

Toda página web está conformada por un código estructurado, denominado **HTML** (HyperText Markup Language). Básicamente, este cuenta con una serie de instrucciones para indicarle al navegador que estemos utilizando la manera en que la página debe ser visualizada y representada.

Una de las características de este lenguaje es su **simplicidad**, dado que para crear un documento HTML, lo único que necesitamos es un editor de texto como WordPad o el Bloc de notas de Windows. Con estas dos sencillas herramientas, es posible estructurar una página web por medio de la inclusión de las **etiquetas** (**tags**) propias del lenguaje. Estas indican

el comienzo y el final de los elementos que integran el documento, desde las características de los textos hasta los elementos multimedia, como imágenes, videos y sonidos.

Aunque HTML es un lenguaje, no lo es de programación, sino que es un **lenguaje de marcado**. Se trata de un sistema de etiquetas que indica al navegador, por ejemplo, cuándo subrayar un título, cómo definir el color del texto o dónde insertar una imagen. Si el navegador detectara algún error en el código que debe representar, mostraría el documento tal como lo hubiera interpretado.

XHTML

La sigla proviene de EXtensible Hypertext Markup Language, y es un lenguaje que cuenta con un etiquetado más estricto que HTML, con lo cual permite una correcta interpretación del contenido del documento. Posee una mayor flexibilidad, ya que logra ser representado por todos los navegadores, incluyendo

HTML es un lenguaje de marcado con etiquetas que le indican al navegador lo que debe mostrar

los de dispositivos portátiles, como PDAs y teléfonos celulares, entre otros.

Gracias a XHTML, el mismo documento es útil para todos los soportes; es decir, no es necesario desarrollar un documento para navegadores web y otro diferente para contenido móvil, por ejemplo.

Además, los elementos correspondientes al diseño (como colores, tipos de fuente o fondos, entre otros) se representan en **hojas de estilo en cascada (CSS)** por separado; de esta forma, podemos tener un código mucho más limpio y de menor tamaño, el cual puede ser seguido con más facilidad.

FIGURA 7.
El código XHTML tiene una menor cantidad de errores de compatibilidad y una mayor integración con las diferentes plataformas de visualización de contenidos web.

Diferencias entre **HTML y XHTML**

La evolución de HTML ha sido compleja, y avanzó sobre diferentes temáticas, que cambiaron aspectos del lenguaje año tras año hasta llegar a la versión actual (HTML 4.0). Esta incluye elementos que en el pasado no estaban integrados y que fueron surgiendo gracias a los diferentes fabricantes de navegadores web.

Sin embargo, hoy en día no se navega por la Web únicamente desde una computadora, sino que cada vez se utilizan más los dispositivos móviles y de otro tipo. Debido a esto, se hizo necesario crear un mecanismo que permitiera manejar la información de manera más estandarizada y para que fuera universalmente accesible.

Como sabemos, XHTML es el nuevo lenguaje estándar generalizado para la Web desde el año 2000. Como está basado en **XML** (Extensible Markup Language), es posible considerarlo como una combinación entre HTML y XML.

BENEFICIOS DE XHTML

Los principales beneficios de XHTML frente a otros lenguajes son los siguientes:

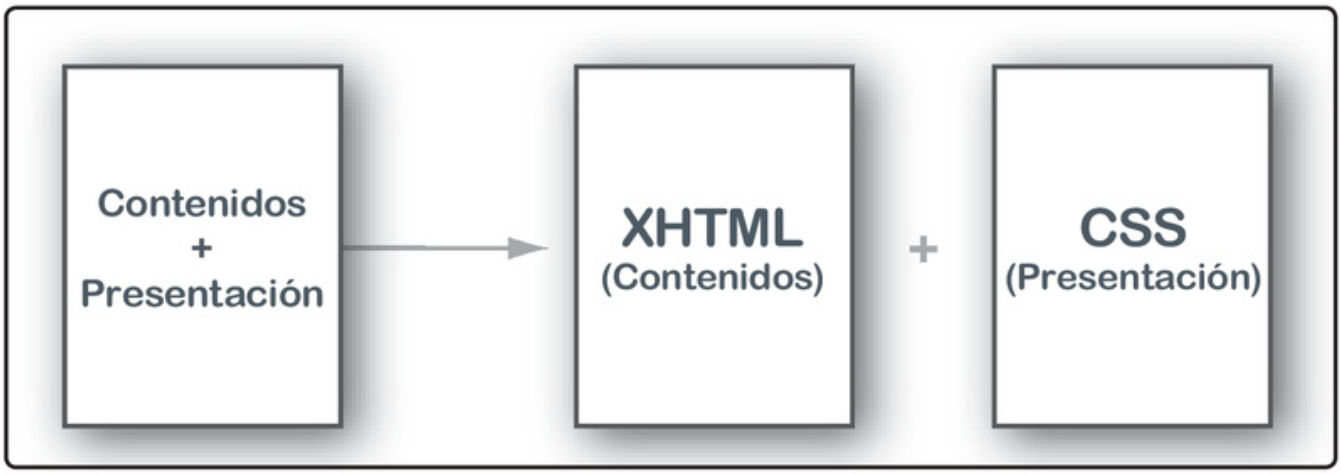


FIGURA 8. **HTML incluye los contenidos y la forma de presentación. XHTML solo trata con contenidos, que se representan según lo indicado en CSS.**

XHTML VÁLIDO

Muchas herramientas de diseño web aún no producen código XHTML completamente válido, es decir, que cumpla con los estándares de la W3C. Es por eso que, antes de publicar una página web en Internet, debemos realizar su comprobación en el sitio **http://validator.w3.org.**

- Compatibilidad con navegadores antiguos: XHTML fue diseñado con el objetivo de cumplir con los requerimientos de todos los browsers.
- Reducción de errores: como mencionamos anteriormente, al ser más estricto, elimina una gran cantidad de errores propios de HTML.
- Facilidad de edición: por ser un código más estructurado –que, además, separa los elementos del diseño en hojas de estilo–, es un código más legible para editar.
- Compatibilidad con estándares web: es compatible con los nuevos estándares sugeridos por la W3C (World Wide Web-Consortium).

Estructura básica **de una página**

Por su estructura, las páginas web pueden dividirse en dos partes: **head** (cabecera) y **body** (cuerpo). El head es el encargado de determinar el tipo de documento y sus propiedades, además de otras características no visibles para el usuario que visita la página. Por su parte, el body es el responsable de darle a la página los elementos visibles, es decir, el texto, las imágenes, las animaciones Flash y todo lo que el usuario puede observar en pantalla.

EL HEAD

Esta sección contiene directivas que el navegador necesita para leer el documento correctamente. Entre estos elementos, encontramos el título de la página, las **meta etiquetas**, las **funciones** JavaScript y los **enlaces** de referencia.

Podemos mencionar siete etiquetas principales en la cabecera, las cuales se encuentran presentes en casi todas las páginas web: **DOCTYPE, html, title, meta DESCRIPTION, meta KEYWORDS, link** a hojas de estilo en cascada (CSS) y **link externo** a funciones JavaScript. De esta forma, un encabezado básico en una página web estaría conformado como vemos en la siguiente porción de código:

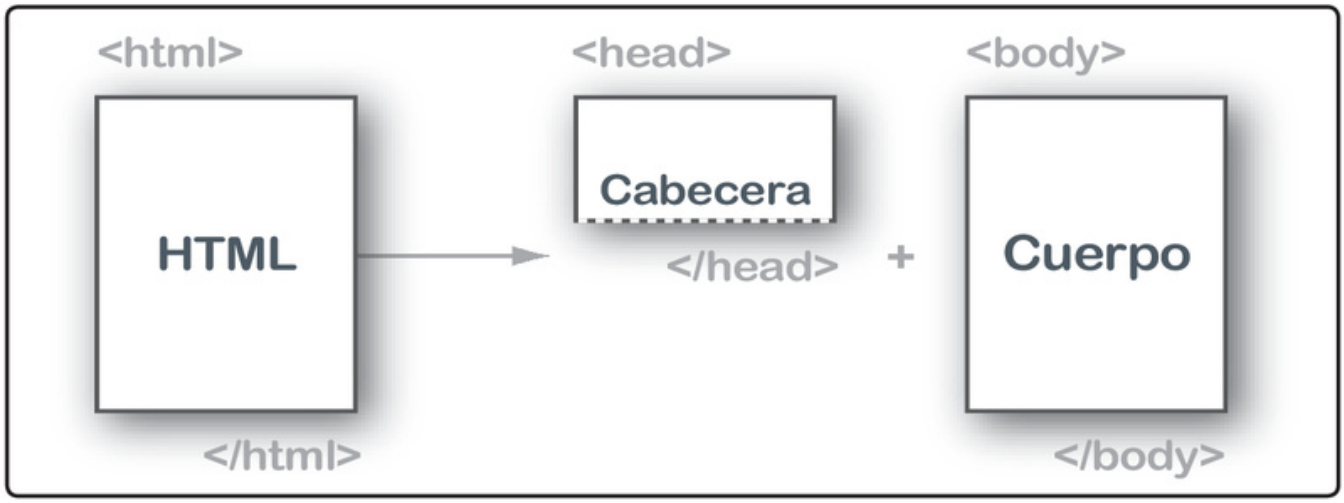


FIGURA 9. **Con sus dos partes integradas, el documento HTML le comunica al navegador cómo debe mostrar la información que contiene.**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Título de la Página Web</title>
<meta name="DESCRIPTION" content="Descripción mínima de la Página Web que debe
    contener preferentemente alrededor de 70 caracteres.">
<meta name="KEYWORDS" content="palabras, claves, que, identifican, a, la, web,
    separadas, por, coma">
<link href="estilos.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="script.js"></script>
</head>
```

En este ejemplo, observamos la definición del tipo de documento en la etiqueta **DOCTYPE**. Aunque no es obligatoria y puede ser omitida, debemos saber que su falta puede dar lugar a imperfecciones en la interpretación por parte del navegador. A continuación, la página está contenida entre las etiquetas **<html>** y **</html>** como apertura y cierre de su código fuente. Luego, aparece la apertura del encabezado a través de la etiqueta **<head>** y el título que será visualizado en la barra de título del navegador, definido entre las etiquetas **<title>** y **</title>**. Le siguen la descripción del sitio y las palabras clave que determinan su temática, y después, los enlaces a las referencias: por un lado, a las hojas de estilo (CSS), y por el otro, a una función JavaScript.

EL BODY

El elemento body, cuya apertura es **<body>** y su cierre **</body>**, se introduce luego del cierre de la etiqueta **</head>**. Le indica al navegador los elementos que debe incluir, así como también la manera de hacerlo, es decir, su organización en el espacio de la página web. Luego del tag de cierre del contenido, aparece la etiqueta **</html>**, que corresponde al final de la página.

El elemento body se introduce luego del cierre de la etiqueta </head>

ETIQUETAS Y POSICIONAMIENTO

Es muy importante definir correctamente las meta etiquetas DESCRIPTION y KEYWORDS junto con el título de la página web. Estos tres factores son considerados indispensables para la adecuada categorización de los buscadores y, por lo tanto, fundamentales para el posicionamiento en ellos.

Etiquetas y atributos: **definición**

El HTML está integrado por elementos que, en conjunto, forman la base de su estructura. De esta manera, cuando definimos una etiqueta, estamos indicando que, en función de ella, se represente algo, que puede ser visible al usuario o no.

DEFINIR LAS ETIQUETAS

Para definir una etiqueta, debemos realizar una apertura y un cierre. Entre ambos, tendremos el contenido que se adaptará a las condiciones que ella disponga. Por ejemplo, para subrayar un texto, utilizaríamos la siguiente línea de código:

```
<u>Texto Subrayado</u>
```

Existen 93 etiquetas estándar que tienen una representación específica, lo que nos facilita la realización de un documento si las utilizamos como herramientas para introducir la información que deseamos mostrar.

LOS ATRIBUTOS

Los atributos son directivas que nos permiten definir valores para cada una de las etiquetas agregadas en el código fuente. Son introducidos en las etiquetas con el objeto de otorgar algunas características puntuales, como el alto, el ancho, un borde, etcétera. Un atributo puede definir desde valores numéricos, como la altura de un determinado bloque, hasta colores y estilos de letra.

Todo atributo contiene un **valor**, que debe estar encerrado entre comillas.
Por ejemplo:

```
<a href="enlace" rel="de que se tratará
   el enlace">Aquí el enlace</a>
```

Siempre debemos recordar que las etiquetas deben llevar una apertura y un cierre, porque así lograremos informar cuál es el contenido que debe ser representado a través de las declaraciones que hacemos en ellas.

Cuando declaramos por completo la etiqueta, podemos observar, entonces, cuatro interventores: la etiqueta propiamente dicha, los atributos, los valores de los atributos y el contenido en cuestión, como en el siguiente ejemplo:

```
<etiqueta atributo1="numérico" atributo
  2="alfanumérico">Contenido</etiqueta>
```

ETIQUETAS OBSOLETAS

Aunque podemos utilizar cualquiera de las etiquetas disponibles en HTML, en este caso recomendamos no emplear algunas en particular, ya que se consideran obsoletas. Entre ellas, podemos mencionar applet, basefont, center, dir, font, isindex, menu, s, strike, u.

FIGURA 10. **Dreamweaver muestra, con diferentes colores, las distintas partes del código.**

Elementos **HTML**

En lo que respecta al lenguaje HTML, se considera a los elementos como las indicaciones o partes que el navegador interpretará más adelante. Durante la programación de un sitio web, encontraremos varios tipos de elementos que nos permitirán modelarlo a nuestro gusto.

UN REPASO POR LOS PRINCIPALES ELEMENTOS

Si bien hay una gran cantidad de elementos, a continuación citaremos algunos de los más utilizados:

- **<p>**: permite dividir nuestro un texto en párrafos. Si preferimos realizar un salto de línea en vez de comenzar un nuevo párrafo, recurrimos al elemento **
**.
- **<ul>**: corresponde a una **lista no numerada**. En ella, cada elemento debe estar identificarlo con la etiqueta **<li>**.
- **<ol>**: en este caso, corresponde a una **lista numerada** y también se acompaña a cada elemento de la lista con la etiqueta **<li>**.
- **<h1>, <h2>,…,<h6>**: identifican a los **encabezados** del sitio web. Su tamaño e importancia varía; 1 es el mayor y 6 es el menor.
- **<a>**: esta etiqueta indica la referencia a algún recurso, y es acompañada por el atributo **"href"**, que sirve para definir la dirección del vínculo.

CLASIFICACIÓN DE ELEMENTOS

Podemos distinguir tres tipos de elementos en el lenguaje HTML:

- **Llenos:** estos elementos cuentan con una apertura y un cierre. El cierre es idéntico a la apertura pero cuenta con una barra diagonal (**/**). Por ejemplo, si queremos agregar una línea de texto en donde una frase aparezca resaltada en negrita, ingresamos lo siguiente:

```
<p>Aquí una línea de texto <b>con una
  parte resaltada en negrita</b></p>
```

- **Vacíos:** en este caso, los elementos no requieren de un cierre. Un ejemplo de este tipo es la etiqueta **
**, que da la posibilidad de realizar un salto de línea.
- **Con argumentos:** aquí se encuentran los elementos que poseen atributos definidos; por ejemplo:

```
<table width="700px">Una tabla de 700
píxeles de ancho</table>
```

CLASIFICACIÓN DE ELEMENTOS

El lenguaje HTML clasifica los elementos en dos grandes grupos: en línea (**inline**) y de bloque (**block**). La diferencia entre ambos radica en la posición que abarcan sobre el **espacio** disponible en la página. Los elementos de **bloque** siempre comienzan en una **línea nueva**, de modo que ocupan todo el espacio disponible de ella, hasta el final. Por su parte, los elementos en línea ocupan únicamente el espacio necesario para su visualización.

Otra diferencia que es posible encontrar entre ambos grupos es su contenido: mientras que los de bloque admiten en su interior otros elementos de bloque y en línea, los elementos en línea no admiten a los de bloque en su contenido. En términos específicos, esto significa que un elemento de bloque solo puede aparecer como contenido en otro elemento de bloque; por otro lado, un elemento en línea puede aparecer en otros en línea, así como también en los de bloque.

TIPOS DE ELEMENTOS HTML

TIPO DE ELEMENTO	ELEMENTO
Elementos en línea	a, abbr, acronym, b, basefont, bdo, big, br, cite, code, dfn, em, font, i, img, input, kbd, label, q, s, samp, select, small, span, strike, strong, sub, sup, textarea, tt, u, var
Elementos de bloque	address, blockquote, center, dd, dt, dir, div, dl, fieldset, form, frame-set, h1, h2, h3, h4, h5, h6, hr, isindex, li, menu, noframes, nos-cript, ol, p, pre, table, tbody, td, tfoot, th, thead, tr, ul
Elementos que pueden ser considerados en línea o de bloque según el caso	button, del, iframe, ins, map, object, script

Tabla 1. **Elementos HTML.**

RECOMENDACIONES

Es muy interesante observar cómo diferentes maneras de introducir el código pueden hacer variar su tamaño en cuanto a líneas, sin modificar la estética del sitio web. Siempre es aconsejable que nuestro sitio web tenga su código validado por la W3C y que éste sea liviano, para acelerar la interpretación de los navegadores.

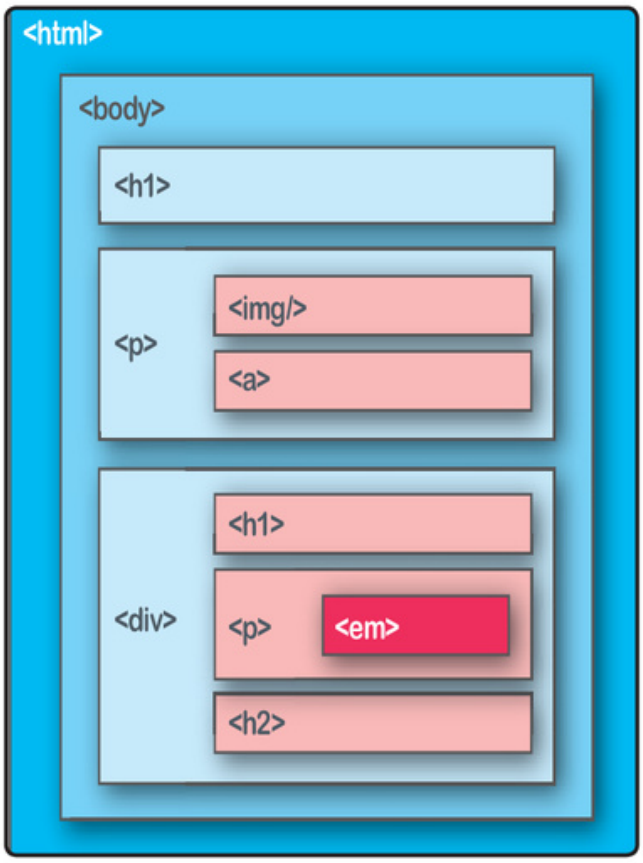


Figura 11. **Los elementos permiten ver fácilmente las partes del código y mejoran la posterior visualización en los navegadores.**

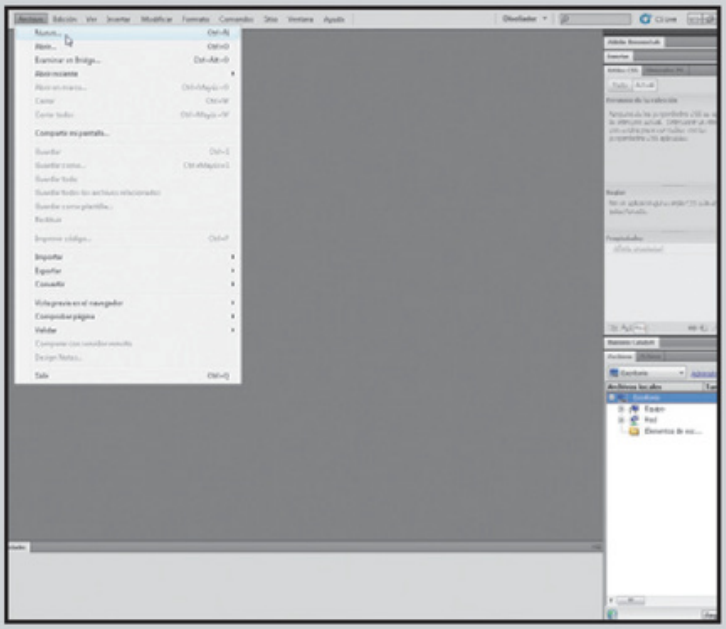
Primer documento HTML en **Dreamweaver**

En este apartado, realizaremos, a través de un paso a paso detallado, un nuevo documento en HTML utilizando Dreamweaver como aplicación.

Adobe Dreamweaver brinda la posibilidad de crear documentos en una gran variedad de formatos, al mismo tiempo que permite abrir todo tipo de archivos basados en texto; como ejemplo podemos mencionar ASP, PHP, JavaScript y hojas de estilo en cascada (CSS), aunque no hayan sido creados utilizando este programa. Además, tengamos en cuenta que podemos abrir y editar archivos de código fuente Visual Basic, Java y C#. De esta forma se amplía el rango de acción y las opciones de esta aplicación, haciendo mucho más eficiente las tareas efectuadas por el desarrollador.

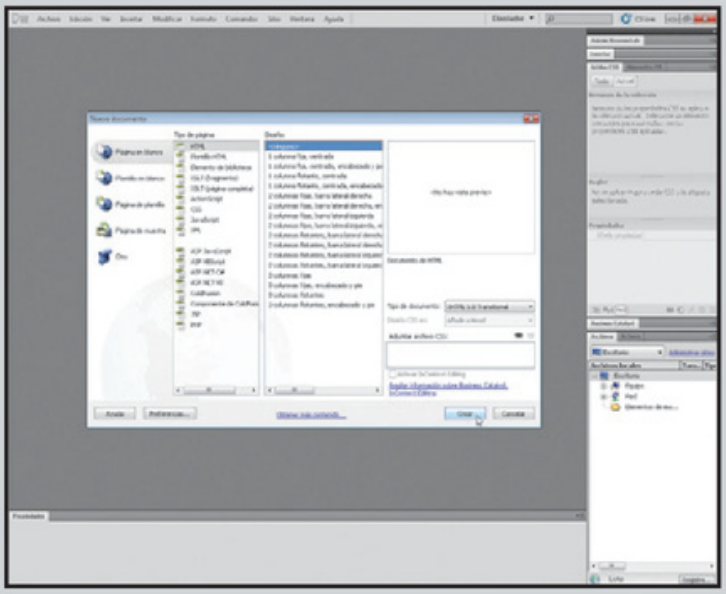
PASO A PASO /1
Crear un documento HTML

1



Para crear un documento HTML, puede hacer clic en la opción **HTML** que aparece en la sección **Crear nuevo**, del cuadro de bienvenida del programa. Otra alternativa, si ya tiene el programa abierto, es dirigirse al menú **Archivo/Nuevo...**.

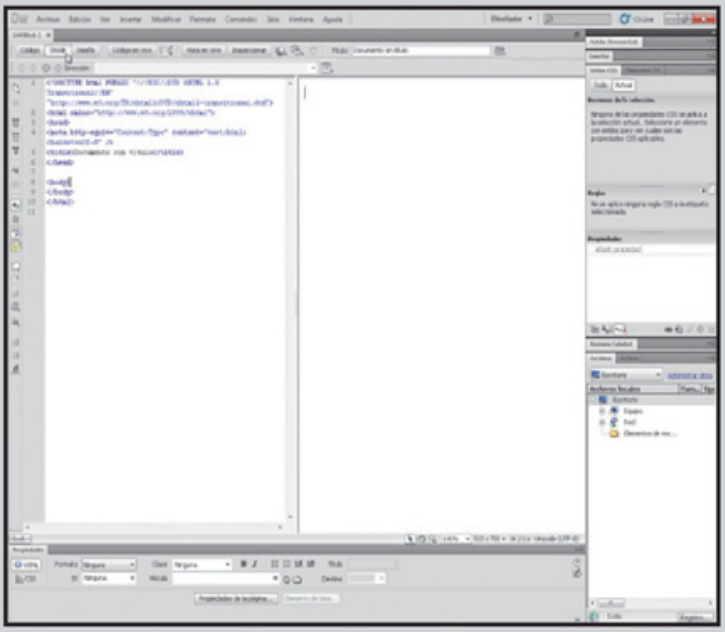
2



Desde el cuadro que se presenta, puede escoger entre una gran cantidad de formatos y, al crear un documento nuevo, optar por alguno de los diseños de página predefinidos o tomar un documento basado en una plantilla existente. En este caso, dentro de la pestaña **Página en blanco**, elija **HTML** en el recuadro **Tipo de página**. En la sección **Diseño** seleccione **Ninguno** y presione **Crear**.

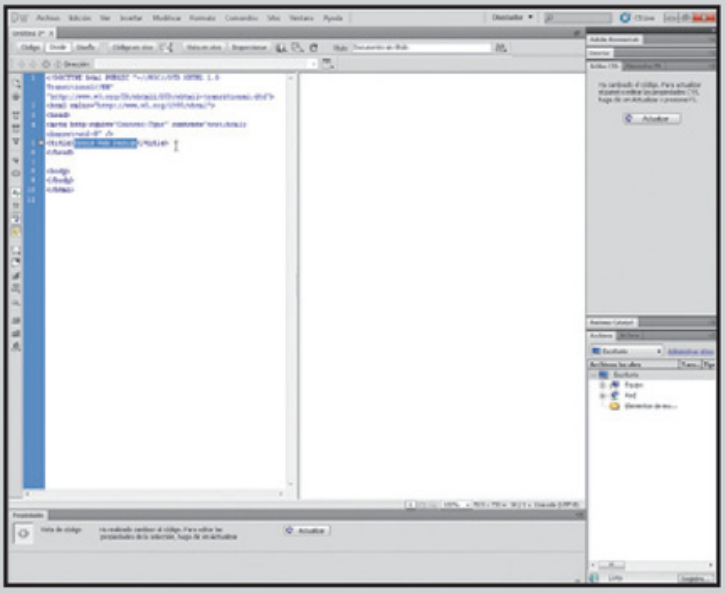
PASO A PASO /1
(Cont.)

3



Aparece en pantalla nuestro documento bajo el nombre "Untitled-1". Puede elegir cómo lo visualizará desde los botones **Código**, **Dividir** o **Diseño**, según lo que desee ver. En este caso, recurra a la vista dividida para apreciar más fácilmente los cambios que haga durante el trabajo.

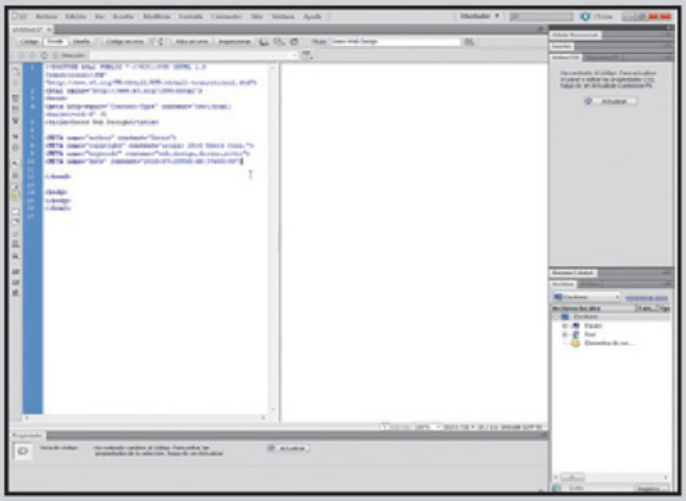
4



Al comenzar a preparar el nuevo documento, revise el código de la izquierda hasta identificar la etiqueta **<title>**. Para colocar el título del sitio, escriba, entre las etiquetas **<title>** y **</title>**, el texto que figurará en la ventana del navegador. En este caso, escriba "**Users Web Design**".

PASO A PASO /1 (Cont.)

5

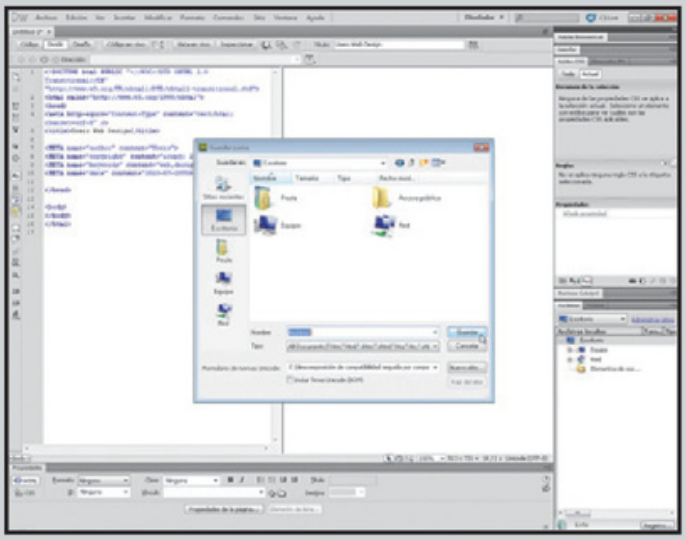


También puede crear otras etiquetas esenciales para el header, como las de descripción, autor, etcétera. En este caso, ingrese el siguiente código antes del tag de cierre:

</head>:

```
<META name="author" content="Users">
<META name="copyright" content="&copy; 2010 Users Corp.">
<META name="keywords" content="web,design,diseno,sitio">
<META name="date" content="2010-07-20T08:49:37+00:00">
```

6



Por último, almacene el documento en nuestro disco a través del menú **Archivo/Guardar como...**. En el cuadro que aparece, coloque el nombre que le dará y presione el botón **Guardar**.

Atributos

Como ya hemos visto en secciones anteriores, cada documento HTML se presenta estructurado por los elementos, los cuales se encargan de indicar al navegador de qué manera debería verse el sitio web que está abriendo actualmente.

Cuando se precisa más información en los elementos, entran en juego los denominados **atributos**, que añaden datos extra para ser interpretados por los navegadores.

Los atributos se incluyen siempre en las etiquetas de apertura de los elementos, y la forma de introducirlos es la siguiente:

```
<img src=" logo.jpg" alt="Logo"></img>
```

Como podemos observar en el ejemplo anterior, lo primero que aparece en el atributo es su nombre, seguido por el carácter igual (=) y, por último, el valor del atributo incluido entre comillas. En este caso, hemos definido una imagen con la etiqueta **img** y, en su propiedad **scr**, especificamos el nombre del archivo de imagen.

La combinación entre las distintas etiquetas y los diferentes atributos nos abre un gran abanico de posibilidades en cuanto a las opciones de visualización de un sitio web. De esta forma, podemos modificar algunas de las características visuales del sitio recurriendo exclusivamente al código fuente, sin que sea necesaria la inclusión de pesadas imágenes que harían retrasar la carga de la página en el navegador, y aburrir al visitante.

La combinación de etiquetas y atributos nos permite lograr distintas visualizaciones de un sitio

LOS CUATRO GRUPOS DE ATRIBUTOS

Si bien cada etiqueta HTML cuenta con sus posibles atributos ya definidos, podemos distinguir algunos que son comunes en casi todas ellas. Es por esto que HTML divide los atributos en cuatro grupos diferenciados por sus funcionalidades: básicos, para internacionalización, de eventos y de foco. A partir de ahora, conoceremos en detalle los correspondientes a cada grupo.

LOS ATRIBUTOS MÁS FRECUENTES: BÁSICOS

Estos atributos son los que se presentan en el código fuente de los documentos HTML con mayor frecuencia, y la razón es bien clara: se utilizan en la mayoría de las etiquetas existentes. En el grupo de los atributos básicos podemos encontrar los siguientes:

- **Id**: identifica de manera única a cada elemento dentro de un documento HTML.
- **Class**: determina la clase CSS que afectará al elemento.
- **Style**: determina de manera directa las propiedades CSS que afectarán al elemento.
- **Title**: añade un título al elemento. Esto es de utilidad para mejorar la accesibilidad del documento, ya que su nombre es visualizado al situar el cursor sobre el elemento.

Cabe aclarar que los atributos **id** y **class** pueden contener letras, números y guiones medios y bajos como valor, pero no pueden empezar con números. Como recomendación, no se deberían utilizar letras como la ñ, ya que podrían no funcionar en algunos navegadores.

LOS ATRIBUTOS DE IDIOMA

Son utilizados para definir y adaptar los elementos insertados en el documento a un idioma específico. Se los emplea en páginas que muestran su contenido en varios idiomas o en aquellas que quieren especificar en qué idioma está su documento. En este grupo encontramos los siguientes tres atributos:

- **Lang**: indica el idioma del elemento.
- **Xml:Lang**: indica el idioma del elemento, aunque en este caso tiene más prioridad que el atributo anterior; es decir, prevalece este atributo por sobre **Lang**.
- **Dir**: indica la dirección del texto. Se lo utiliza al trabajar con idiomas que no se escriben de izquierda a derecha.

LOS ATRIBUTOS DINÁMICOS

Encontramos dentro de este grupo muchos atributos, debido a que son los encargados de exhibir los **efectos** y las **animaciones** de las páginas que incluyen lenguaje JavaScript. Veamos, a continuación, cuáles son estos atributos:

- **Onclick**: establece la acción que se va a realizar cuando se haga un clic sobre el elemento.

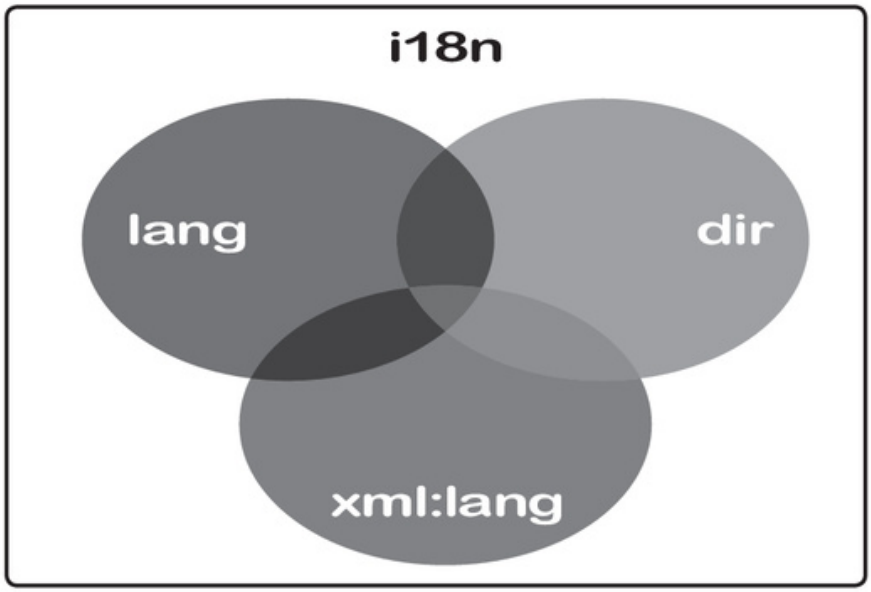


Figura 12. A este grupo de atributos también se lo denomina i18n, que es la abreviatura de la palabra internacionalización, utilizando el número de letras que tiene el término entre la "i" inicial y la "n" final.

- **Ondblclick**: establece la acción que se realiza cuando se hace un doble clic sobre el elemento.
- **Onmousedown**: se ejecuta la acción cuando se detecta el botón del mouse presionado.
- **Onmouseup**: se ejecuta la acción cuando se detecta que el botón fue soltado.
- **Onmouseover**: establece la acción cuando se detecta que el cursor se sitúa sobre el elemento.
- **Onmousemove**: establece la acción cuando se detecta que el cursor se encuentra en movimiento sobre el elemento.
- **Onmouseout**: se ejecuta la acción cuando el cursor abandona el elemento.
- **Onkeypress**: se ejecuta la acción cuando se presiona una tecla del teclado.
- **Onkeydown**: se ejecuta la acción cuando se detecta que está pulsada la tecla del teclado.
- **Onkeyup**: se ejecuta la acción cuando la tecla del teclado fue soltada.

El grupo de atributos de eventos solo es utilizado cuando el documento HTML incorpora el lenguaje **JavaScript**, ya que a través de ellos, es posible lograr que el sitio web responda ante la interacción del usuario. Por ejemplo, podemos verlos en acción y descubrir su potencial al visitar sitios en los que se emplean menús de navegación JavaScript, los cuales despliegan su lista de elementos cuando nos situamos sobre cada uno de los botones.

Junto a JavaScript, los atributos de evento ayudan a dar interacción a los documentos HTML

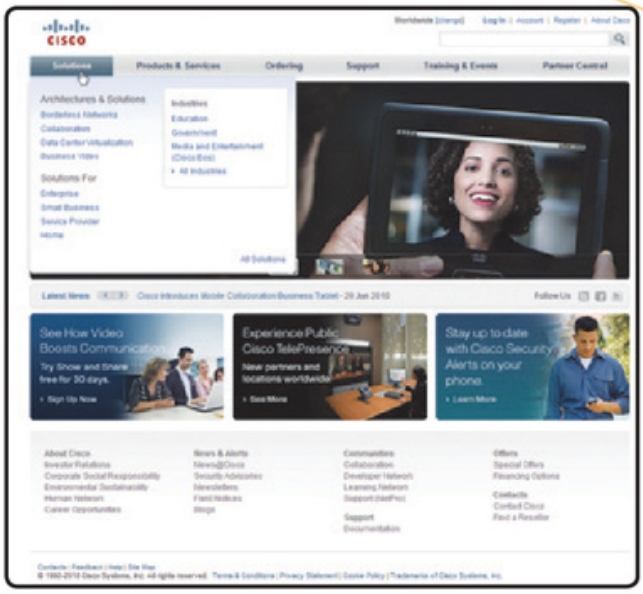


FIGURA 13. **Los atributos de eventos son muy importantes para las aplicaciones web actuales. En www.cisco.com podemos verlos en acción.**

LOS ATRIBUTOS DE SELECCIÓN

El foco aparece durante la interacción del visitante en el sitio. Estos atributos nos permiten asignar acciones que se realizarán en cada caso. Cuando nos referimos a detectar o perder el foco, estamos hablando de que el usuario ha seleccionado o ha deseleccionado el elemento. Por ejemplo, si tenemos un formulario web, diríamos que cuando el usuario está escribiendo en uno de sus campos, este elemento tiene el foco. Cuando continúa hacia otro campo, el elemento anterior ha perdido el foco, ya que lo tiene el elemento actual que está usando. Veamos cuáles son los atributos de foco:

- **AccessKey**: establece una tecla de acceso rápido a un determinado elemento.
- **TabIndex**: establece el orden de tabulación del documento; su valor está entre 0 y 32.767.
- **OnFocus**: se ejecuta la acción cuando se detecta

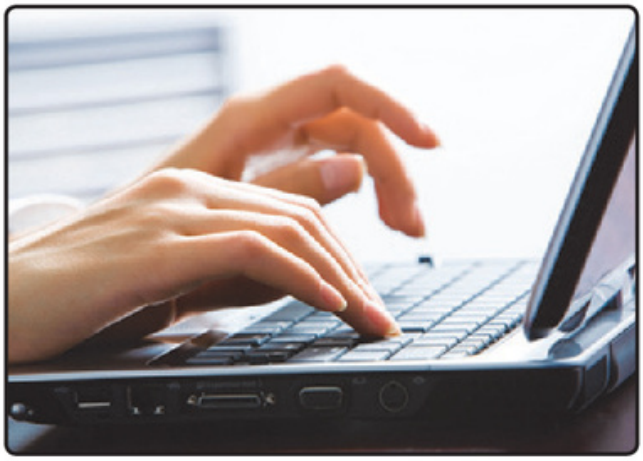
el foco sobre el elemento, lo que significa que actúa cuando el elemento es seleccionado.

- **OnBlur**: se ejecuta la acción cuando se detecta que el elemento ha perdido el foco; es decir, actúa cuando el elemento se ha deseleccionado.

El atributo **accessKey** asigna teclas de acceso rápido a los elementos de un sitio web, lo cual tiene un problema implícito: los navegadores. Este inconveniente consiste en que, si bien HTML define qué teclas se deben presionar para acceder al elemento en cuestión, Internet Explorer interpreta al acceso rápido a través de la tecla **<ALT>**, mientras que Mozilla Firefox interpreta esta tecla a través de la combinación **<ALT + SHIFT>**, el navegador Safari realiza la acción a través de la tecla **<CTRL>** y, por último, Opera la interpreta a través de la combinación **<SHIFT + ESC>**.

Todo esto quiere decir que el acceso a cada parte de una página utilizando las teclas depende, exclusivamente, del navegador que esté utilizando el usuario; por lo tanto, debemos tener en cuenta estas diferencias al crear el sitio web.

El atributo tabIndex exhibe su mayor funcionalidad cuando estamos manejando formularios de login o de registro



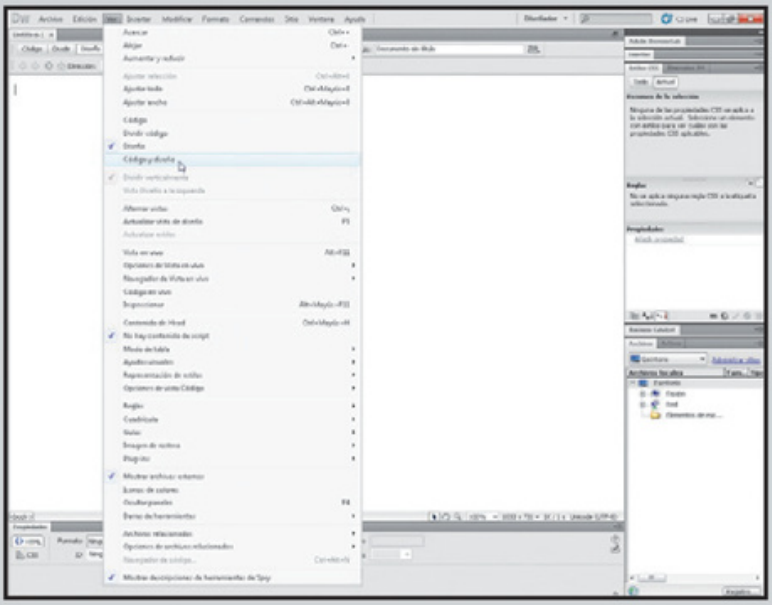
El atributo **tabIndex**, por su parte, exhibe su mayor funcionalidad cuando estamos manejando, por ejemplo, los formularios de login o de registro. Cuando un usuario se enfrenta a un formulario en el que debe ingresar sus datos en diferentes campos, al utilizar la tecla **<TAB>** para moverse entre ellos, lo hará en el orden indicado mediante este atributo. Para decirlo de otra manera, al definir el atributo **tabIndex** de cada elemento, estaremos definiendo la secuencia en la que el usuario entrará en cada campo si no recurre al mouse, es decir, si pasa de un campo a otro pulsando **<TAB>**.

Primeras etiquetas **HTML**

Veremos ahora cómo escribir código en Dreamweaver y observaremos los cambios que se producen en la sección que muestra la vista de diseño. Aunque ya creamos y modificamos algunas etiquetas para configurar el título de la página y agregar tags descriptivos, aún no hemos generado modificaciones en el contenido de ella.

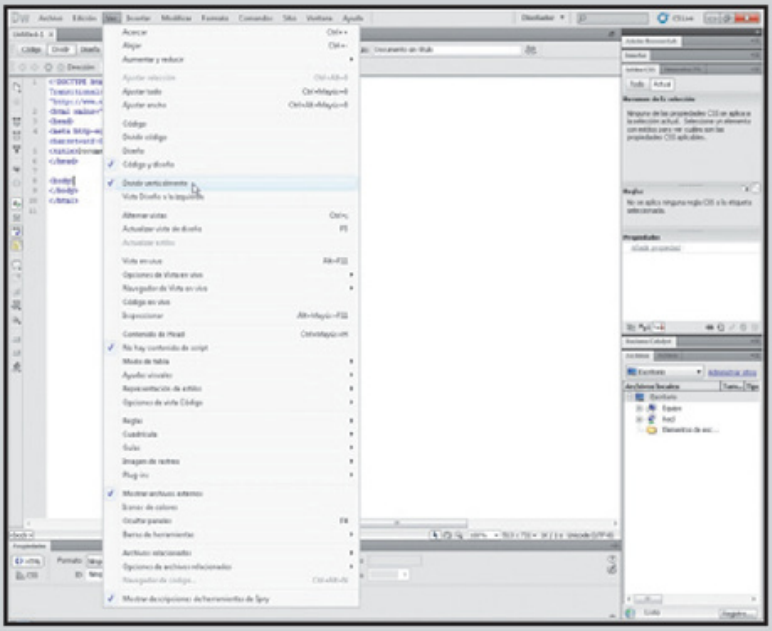
PASO A PASO /2
Escribir código en Dreamweaver

1



Para escribir etiquetas en Dreamweaver desde el código, seleccione en la barra de vistas de la interfaz la que dice **Dividir**.
Otra forma de hacerlo es ir al menú **Ver** y allí elegir **Código y diseño**.

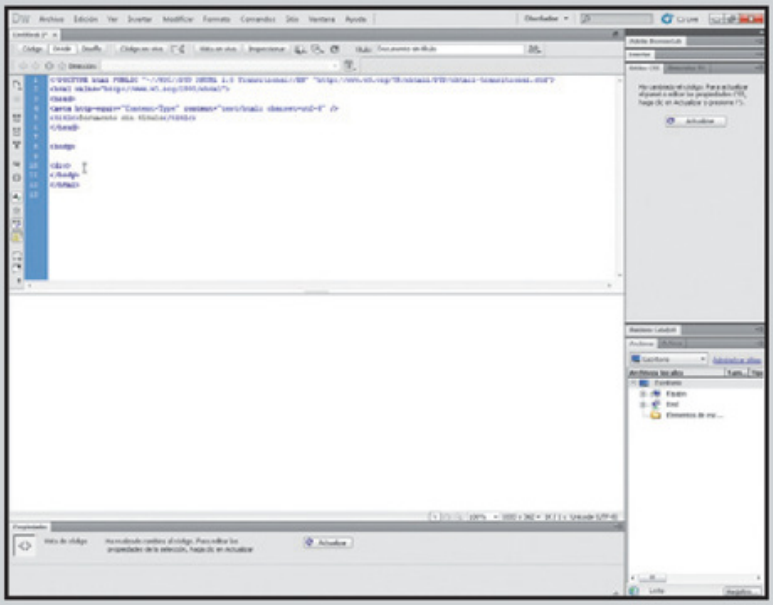
2



Luego, para que no se corten las líneas de código y el trabajo resulte más cómodo, en el mismo menú haga clic en **Dividir verticalmente**, de modo que la división de la pantalla sea horizontal.

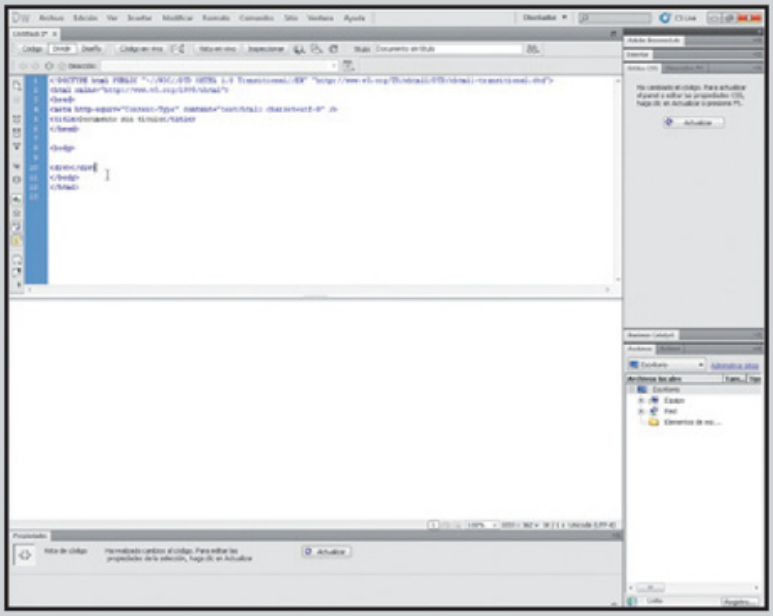
PASO A PASO /2
(Cont.)

3



Dentro de la etiqueta **body**, presione la tecla **<ENTER>** y comience a escribir la primera etiqueta, que puede ser cualquiera de las existentes. En este caso, será ** **.

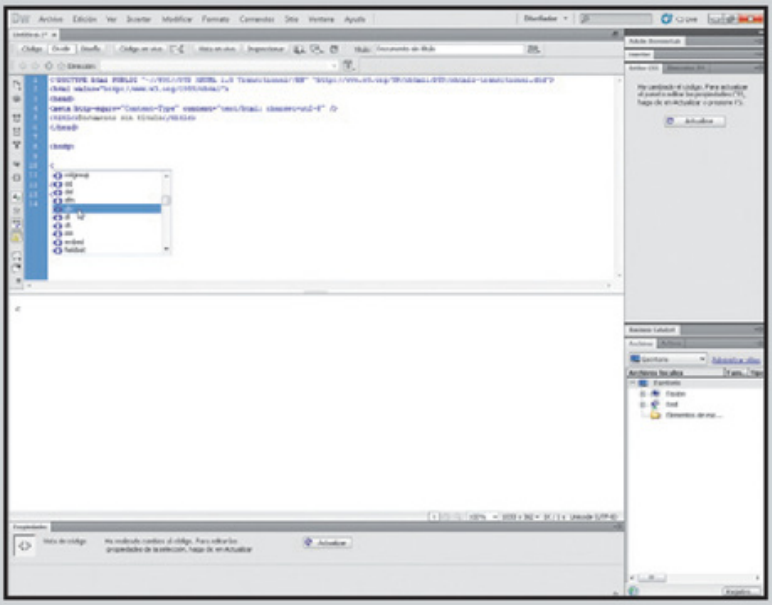
4



Las etiquetas llevan una apertura y un cierre, lo que significa que, cuando abra una etiqueta, debe cerrarla. Para cerrar ** **, escriba ** **.
Este cierre debe aparecer al final del contenido de la etiqueta; es decir, el contenido tiene que ir entre la apertura y el cierre.

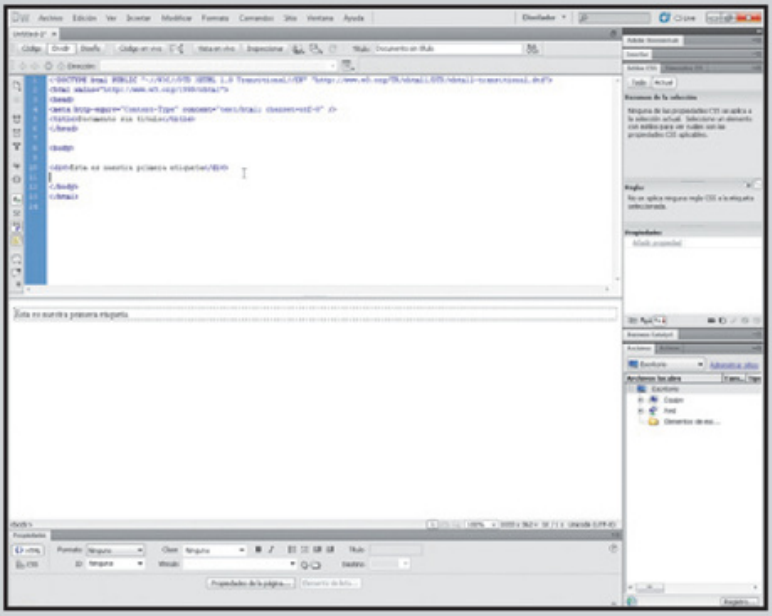
PASO A PASO /2
(Cont.)

5



Dreamweaver da la posibilidad de utilizar un gestor de etiquetas. Este gestor aparece al colocar el signo **<** para abrir una etiqueta. Seleccione la etiqueta en el gestor, ayudado por el mouse o los cursores de dirección, y una vez hecho esto, presione la tecla **<ENTER>** y escriba el signo **>**.

6



Cuando termine de escribir el contenido de la etiqueta, bastará con escribir **</** para que Dreamweaver la cierre automáticamente con su utilidad de autocompletado. Luego, podrá ver el contenido que haya incorporado en la página.

Hasta aquí, vimos cómo escribir etiquetas en el código de Dreamweaver. Luego conoceremos la forma adecuada en que podemos incorporar, y la forma de utilizar y modificar cada una de ellas.

Multiple **choice**

1 ¿Qué da como resultado la maquetación?

a- Un bosquejo del diseño.
b- Un diseño preliminar.
c- Un diseño convertido en sitio navegable.
d- Un sitio sin los enlaces.

2 ¿Qué es cross-browsing?

a- Generación de estilos CSS.
b- Aplicación de parches en el código CSS y HTML.
c- Aplicación de enlaces.
d- Aplicación de animaciones.

3 ¿Qué herramienta sirve para generar código HTML?

a- Notepad++.
b- Notepad.
c- Microsoft Word.
d- Microsoft WordPad.

4 ¿Cuál es la acción que debemos realizar al ejecutar Dreamweaver por primera vez?

a- Ejecutar el asistente de configuración.
b- Abrir un archivo de ejemplo.
c- Definir la distribución del espacio de trabajo.
d- Cambiar el idioma predeterminado.

5 Mencione algún beneficio de XHTML sobre otros lenguajes.

a- Más sentencias disponibles.
b- Más herramientas de desarrollo compatibles.
c- Facilidad de instalación.
d- Facilidad de edición.

6 ¿En que momento se introduce el elemento body?

a- Después del cierre de head.
b- Antes del cierre de head.
c- En el pie de la página.
d- Cuando aparezca la etiqueta html.

Respuestas: 1-c, 2-b, 3-a, 4-c, 5-d, 6-a.

Capítulo 3

Estructura del sitio

Aquí veremos las partes que componen la estructura de un sitio web, su descripción y los elementos que contienen.

RedUSERS.com

Tendencias en el **diseño de fondos**

El fondo de un sitio puede darle un toque distintivo y actuar como contexto de la información. Trabajado de manera apropiada, podrá reflejar una determinada época, representar el concepto de minimalismo o, simplemente, dar aire al diseño.

Podemos categorizar los fondos según su **aspecto visual**, es decir, el tratamiento gráfico que se aplique; o según su relación con el resto de los elementos de la interfaz. Veamos las opciones del primer grupo.

- **Colores plenos**: son fondos sencillos que transmiten orden y claridad visual. Son pertinentes en los sitios que necesitan priorizar los contenidos, como los de fotografías o tipografías. La aplicación de estos fondos es sencilla, ya que podemos crearlos con la declaración de la propiedad `background-color` en el elemento `body` de la hoja de estilos.

- **Gradientes**: otorgan mayor profundidad al diseño y se convierten en un detalle que eleva su calidad visual. Para aplicarlos, es aconsejable repetir una pequeña imagen a modo de patrón, utilizando las propiedades `background-image` y `background-repeat`. La dirección del gradiente debe ser vertical u horizontal, no diagonal.

- **Texturas**: son apropiadas para aquellos diseños en los que queremos representar una metáfora o transmitir una cierta atmósfera. Por ejemplo, el uso de papeles viejos denotará el paso del tiempo, mientras que la elección de una textura de acero transmitirá conceptos como asepsia, modernidad o volumen. Las texturas suelen ser aplicadas como una gran imagen de fondo, utilizando la propiedad `background-image`, y también es posible hacerlo como `patrón`, repitiendo una pequeña imagen.

El fondo puede ser un condimento especial para dar un toque distintivo y contexto para la información del sitio

FIGURA 1. En los sitios con fondos de color pleno se destaca el contenido por sobre el contexto, como vemos en www.davidfooks.com.

FIGURA 2. El uso de texturas enriquece el lenguaje gráfico, transmitiendo diferentes conceptos o ideas. En www.gomammoth.co.uk encontramos un ejemplo de uso de textura.

• **Fotografías:** si encontramos una fotografía apropiada, podemos lograr un ambiente muy realista al utilizarla como fondo. Habitualmente, la foto utilizada recibe un importante retoque y se aplica de manera fija a través de la propiedad **`background-attachment`**. Los contenidos quedarán por encima de ella y, al hacer scroll, se deslizarán independientemente del fondo. Recordemos que no es aconsejable aplicar una fotografía a modo de patrón, porque su repetición genera saltos visuales.

• **Ilustraciones:** los fondos creados a partir de ilustraciones suelen tener un toque de personalidad u originalidad. Sin embargo, esta técnica en general transmite un espíritu algo infantil, que no siempre resulta apropiado. Como sabemos, las ilustraciones pueden aplicarse como patrones y también como imágenes de fondo y, a diferencia de las fotografías, están compuestas por pocos colores, por lo que pueden ser optimizadas con formato **`GIF`** o **`PNG`**, de peso reducido.

FIGURA 3. Las fotografías crean un entorno realista, difícil de lograr mediante otras técnicas. Encontramos un gran ejemplo en www.alexarts.ru/en.

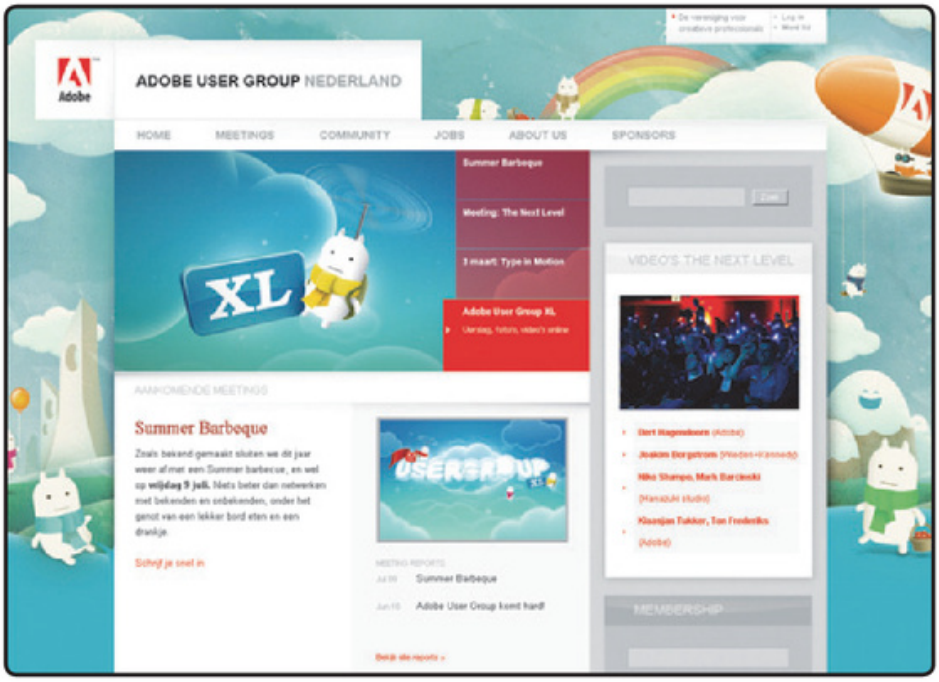


FIGURA 4. Las ilustraciones crean ambientes muy ricos. Su creatividad trasciende los límites que proporcionan otras técnicas, como vemos en www.adobeusergroup.nl.

RELACIÓN DEL FONDO CON LOS ELEMENTOS

Además del tratamiento gráfico, para crear la atmósfera adecuada en un sitio debemos tener en cuenta la relación que los fondos mantienen con el resto de los elementos, ya que podemos encontrar varias formas de aplicación.

- **Fondo y contenido en diferentes niveles visuales**: el fondo permanece en un segundo nivel, actuando como marco para el resto de los contenidos del sitio. El contenido principal se ubica dentro de su propio contenedor, que se coloca por delante del fondo, como si se tratara de un plano más cercano al espectador. En ocasiones, el fondo del contenedor se aplica con transparencias, acercando visualmente ambos planos.

- **Fondo y contenidos integrados visualmente**: al integrar el fondo con los elementos, disminuye la sensación de profundidad que otorga el uso de diferentes planos. Como estos sitios no están enmarcados por contenedores, transmiten sensación de aire, apertura y, en algunos casos, horizontalidad. Por lo

PATRONES

Esta técnica consiste en repetir –en el eje X, Y o en ambos– un módulo cuadrado o rectangular para componer un fondo de mayor tamaño. Los patrones pueden ser creados a partir de texturas, fotografías o gráficos, y su ventaja es que el peso total es inferior.

FIGURA 5.
En www.pixelimagedubai.com vemos un sitio cuyo fondo actúa como marco de los contenidos principales.

general, el trabajo de implementación del fondo se realiza con mayor cuidado, dado que el riesgo de que los elementos queden desfasados es más notorio que en la implementación de fondos ubicados en diferentes planos.

• **Fondos compuestos por varios fondos**: aquí se aprovecha el fondo para separar los contenidos. De esta manera, se utiliza, por ejemplo, una ilustración para el cabezal y un color pleno u otra imagen como fondo para el cuerpo. Así se logra un corte visual entre ambos sectores.

En los fondos compuestos se aprovecha el fondo para separar los contenidos

FIGURA 6. The Blizzards (www.theblizzards.ie) presenta un fondo que se integra a los contenidos principales, lo que disminuye el nivel de profundidad.

Propiedades y aplicación de **fondos**

El fondo de un elemento HTML puede controlarse a través de la propiedad **background**, que contempla **background-color**, **background-image**, **background-repeat**, **background-attachment** y **background-position**.

Vale aclarar que, cuando hablamos de fondo, nos referimos a la zona que ocupa el contenido. No se verán afectados los bordes ni los márgenes, pero sí el padding, si este existiera, ya que forma parte del contenido.

BACKGROUND-COLOR

Este atributo se usa para determinar el color de fondo de cualquier elemento HTML. Es posible asignarle un valor hexadecimal o RGB, establecerlo por el nombre del color o, simplemente, usar el valor **transparent**.

En el siguiente ejemplo, al elemento **<body>** le asignaremos color gris, y a la clase **caja**, naranja. En cambio, si al elemento **caja** no le asignáramos un color determinado, este adoptaría el establecido por defecto, que es **transparent**.

En la actualidad, los sitios de vanguardia poseen grandes ilustraciones e imágenes en el fondo

```
body{
        background-color: gray;
}

.caja{

        width: 200px;
        height: 200px;
        background-color: orange;
}
```

BACKGROUND-IMAGE

Esta propiedad establece una imagen de fondo para cualquier elemento HTML. En este caso, se indica dentro del código la ruta de ubicación de la imagen, es decir, el lugar donde se encuentra el archivo. También es posible usar un color de fondo al aplicar una imagen, ya que estas propiedades pueden emplearse juntas (la imagen se posicionará delante del color de fondo). La sintaxis de esta propiedad es: **url ("ruta a la carpeta donde se ubica la imagen / nombre del archivo y su extensión")**. Veamos un ejemplo:

```
body {
        background-color:#FFFFFF;
        background-image: url ("img/
bg-b.jpg");

        margin:0;
        padding:0;
}
```

FIGURA 7. **En www.themepresso.com podemos ver un ejemplo de cómo queda aplicada una imagen como fondo del ←body→.**

BACKGROUND-REPEAT

Otro de los aspectos del fondo que controlamos con CSS está relacionado con el tamaño de la imagen y de los elementos que la contienen. Si la imagen es más grande que el elemento HTML que la contiene, esta se verá cortada. En cambio, si es más chica, se repetirá automáticamente en el eje X (horizontal) y en el Y (vertical) formando un mosaico, a menos que se establezcan otros atributos.

Esta repetición puede controlarse a través de los valores de la propiedad **Background-repeat**:

- **repeat-x**: repite la imagen horizontalmente.
- **repeat-y**: repite la imagen verticalmente.
- **no-repeat**: no repite la imagen.
- **repeat**: repite la imagen en X e Y.

En el siguiente código se utiliza una imagen como fondo del contenedor y, dado que no se indica otra cosa, esta se repetirá en ambos sentidos:

Con una repetición solo en el eje X, la imagen se repite horizontalmente

```
#contenedor{
        width: 300px;
        height: 300px;
        background-image: url(../img/
textura.jpg);
        float: left;
}
```

En cambio, al asignarle la repetición solo en el eje X, la imagen se repite horizontalmente:

```
#contenedor{
        width: 300px;
        height: 300px;
        background-image: url(../img/
textura.jpg);
        background-repeat: repeat-x;
        float: left;
}
```

Un recurso muy utilizado al maquetar sitios es el empleo de una pequeña imagen de fondo que se repita

FIGURA 8. Arriba, podemos ver una imagen repitiéndose en ambos ejes del contenedor. A la derecha, solo se repite en el eje X.

en el eje X y, luego, un color pleno. Esto es común en sitios que son muy largos, ya que es posible usar una imagen vistosa o un gradiente y luego, si el contenido es muy extenso, solo se visualizará el color de fondo.

BACKGROUND-ATTACHMENT

Esta propiedad no es muy utilizada. Su efecto es mantener fija la imagen de fondo de un elemento mientras el contenido se desplaza por delante de ella cuando nos movemos con el scroll.

FIGURA 9. En www.anthonynolan.org, la imagen del degradé se repite en la parte superior del sitio; después se usa un color plano para continuar con el alto de la página.

BACKGROUND-POSITION

A través de esta propiedad, indicamos la posición de la imagen de fondo de un elemento. Es posible definir este parámetro a partir de **palabras clave**:

- **Ubicación horizontal**: **left** (izquierda), **center** (centro), **right** (derecha).
- **Ubicación vertical**: **top** (arriba), **center** (centro), **bottom** (abajo).

O mediante medidas como las siguientes:

- **Porcentajes**: fija la distancia en porcentaje desde la posición inicial, que es 0,0 y está ubicada en el margen superior izquierdo. Su valor por defecto es 50%.
- **Píxeles**: fija la distancia en píxeles desde la posición inicial, que es 0,0 y está ubicada en el margen superior izquierdo.
- **Inherit**: hereda la posición del elemento padre.

En el siguiente ejemplo podemos ver cómo se pueden combinar ambas posiciones utilizando las palabras clave. La primera hace referencia a la ubicación en X, y la segunda, a la posición en Y:

```
body {
        background-image:url("../
imagenes/bg.png");
        background-position:left top;
        background-repeat:no-repeat;
}
```

También podemos utilizar para el background la propiedad **shorthand**, que permite definir, primero, el color; a continuación, la ruta de la imagen; luego, su repetición y, finalmente, su posición en X e Y. Veamos un ejemplo:

```
body {
        background: #FFFFFF url("grafik/
topBack.png") repeat-x scroll 0 0;
}
```

El background de un sitio

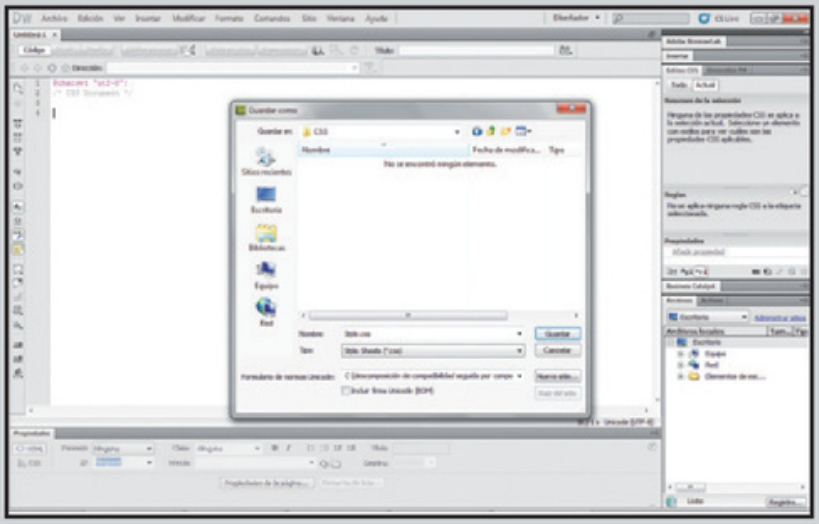
En este apartado, veremos en seis pasos las formas de colocar un background.

El background de un sitio es importante, tanto como el sitio en sí mismo; es el que generará los contrastes entre el contenido y los distintos contenedores. En este caso, explicaremos cómo colocar nuestro background en el sitio, pero antes de hacerlo, será importante abrir el archivo de nuestro fondo en el programa de diseño que utilicemos, como Photoshop o Illustrator, y analizar cómo conviene usarlo con el código CSS y HTML para integrarlo al sitio.

Luego de esto, tendremos que darnos a la tarea de dividir la imagen de la forma que resulte más conveniente. En este caso, cortamos un trozo de header de 16 píxeles de ancho por 56 píxeles de alto, una sección de body de 123 píxeles de ancho por 123 píxeles de alto y, finalmente, hacemos un recorte del footer de 56 píxeles de ancho por 552 píxeles de alto.

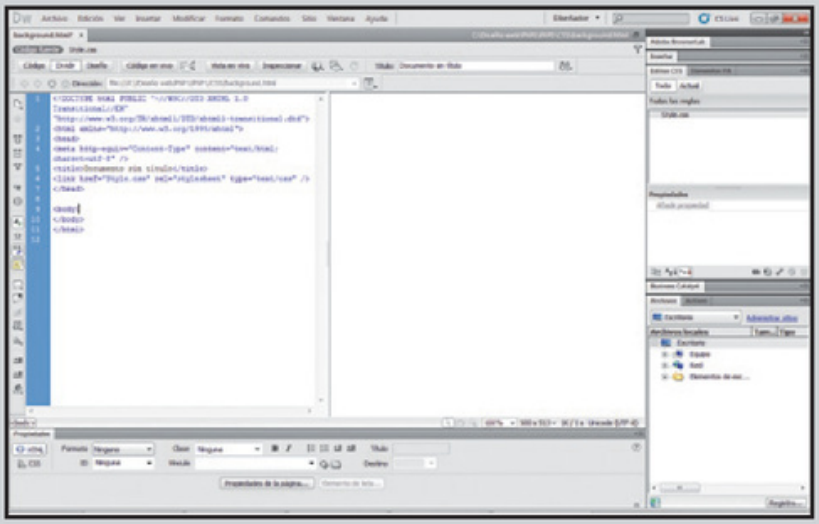
PASO A PASO /1
Crear el background de un sitio

1



Luego de guardar los recortes como archivos JPG, abra Dreamweaver y cree un nuevo archivo CSS desde **Archivo/Nuevo…**, eligiendo **CSS** en el recuadro **Tipo de página** y presionando **Crear**. Vaya a **Archivo/Guardar como…** y, en la ventana que aparece, elija la carpeta donde almacenará la hoja de estilos. Escriba **style** en el campo **Nombre** y elija **Style Sheets (*.css)** en **Tipo**. Para terminar, presione **Guardar** y cierre el archivo.

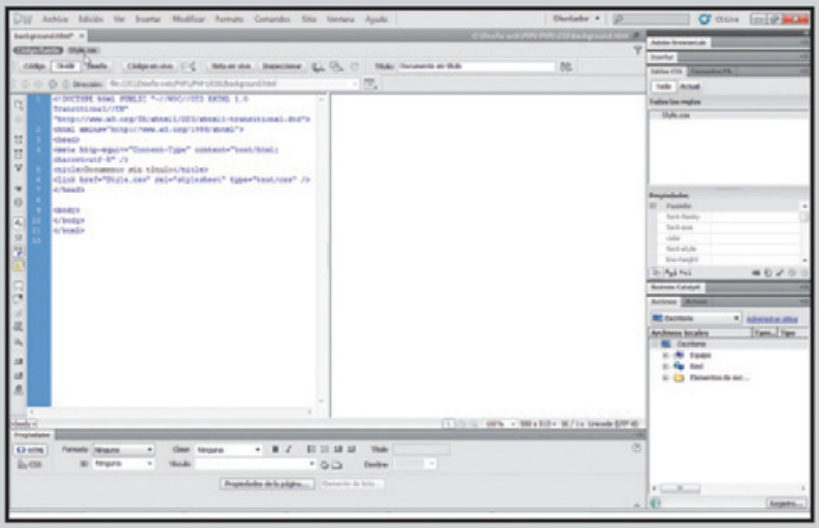
2



Genere un archivo HTML desde **Archivo/Nuevo…**, eligiendo **HTML** en el recuadro **Tipo de página** y presionando **Crear**. Después, almacene el archivo en su disco desde **Archivo/Guardar como…**. En el nuevo documento, dentro del panel **Estilos CSS**, haga clic en el icono **Adjuntar hojas de estilo**. En la ventana **Vincular hoja de estilos externa** que aparece, presione **Examinar…** y ubique la hoja de estilo creada anteriormente (**style.css**). Luego de que la seleccione, haga clic en **Aceptar**.

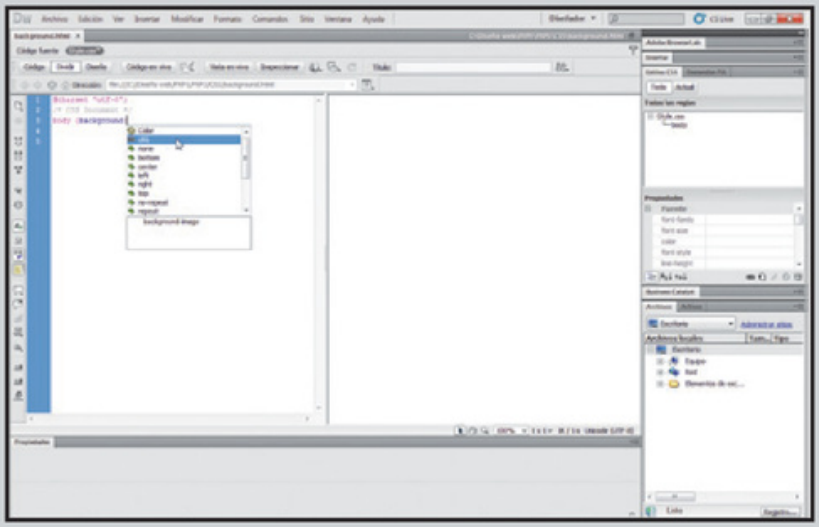
PASO A PASO /1 (Cont.)

3



Vuelva a la ventana **Vincular hoja de estilos externa**, donde verá la ruta y el nombre del archivo CSS. Presione **Aceptar** y, luego, guarde el archivo HTML para salvar los cambios realizados.
A continuación, haga clic en el botón **style.css** para comenzar a trabajar en el archivo CSS correspondiente.

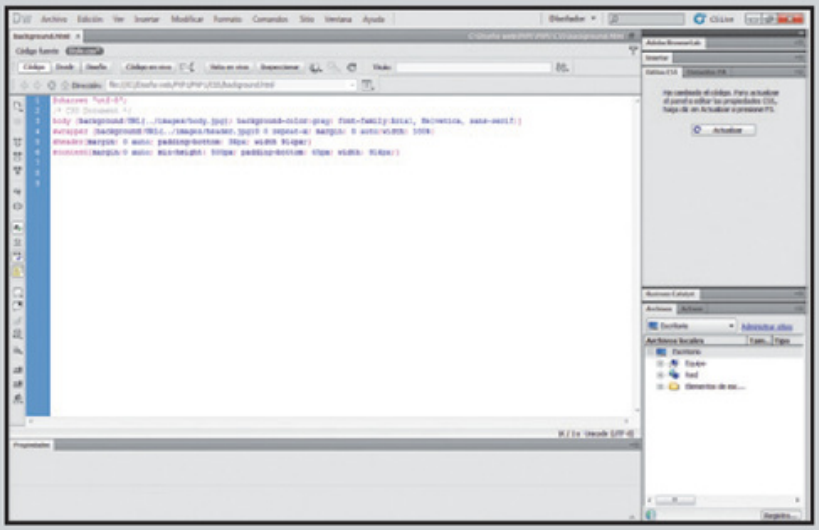
4



Ahora va a definir el body. Para hacerlo, escriba **body { background: url(../imagenes/body.jpg); background-color: gray; font-family:Arial, Helvetica, sans-serif;}**. Luego de ingresar la propiedad **background:**, al tipear el espacio, Dreamweaver le ofrecerá opciones. Si elige **URL**, podrá buscar el archivo con la imagen de fondo del header, de manera sencilla. Si no, bastará con que usted mismo escriba la ruta.

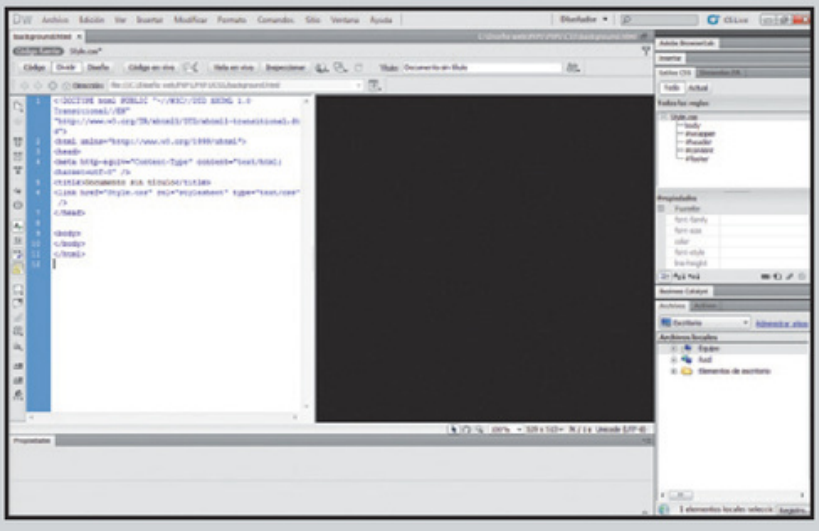
PASO A PASO /1 (Cont.)

5



Para definir el wrapper, escriba **`#wrapper { background: url(../imagenes/header.jpg)0 0 repeat-x; margin: 0 auto; width: 100%;}`**. A continuación, ingrese **`#header { margin: 0 auto; padding-bottom:36px; width: 914px;}`** para generar el header y, finalmente, establezca el área de contenido con **`#content { margin: 0 auto; min-height: 500px; padding-bottom:65px; width: 914px;}`**. El hecho de no darle al header una altura determinada le permitirá colocar la navbar en su interior.

6



Para terminar, defina el footer con **`#footer { background: url(../imagenes/footer.jpg)0 0 repeat-x; min-height: 552px; margin: 0 auto; width: 100%;}`**.

En este caso, fijes la altura, ya que el footer solo contendrá elementos que están definidos, lo que lo hace invariable. Al hacer clic en **`Código fuente`**, ver{a que se aplica el estilo al body de la página.

Más adelante en esta clase, cuando aprendamos a crear un layout, veremos cómo debemos generar los divs que darán estructura al sitio para que incorporen estas reglas y, así, la página luzca como deseamos en cada una de sus áreas.

Bordes (CSS)

Cuando trabajamos con HTML, debemos imaginar que cualquier elemento está inscripto en un rectángulo, cuyos bordes podemos controlar. La propiedad genérica es **Border** y afecta a los cuatro bordes que posee la figura (imaginaria), contemplando los atributos de ancho, estilo y color (**border-width**, **border-style** y **border-color**). Esta propiedad vale para cualquier elemento HTML. A continuación, veremos cada una de sus propiedades en detalle, así como la forma de aplicarla en los diferentes lados.

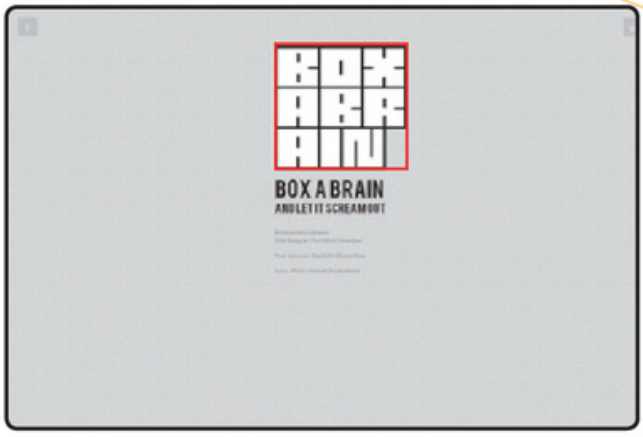


FIGURA 10. **En las capturas observamos que, aunque el diseño sea circular o irregular, la imagen siempre está circunscripta dentro de un cuadrado.**

ANCHO DE BORDES

El **grosor** de los bordes puede ser modificado a través de las siguientes propiedades:

- **border-top-width**: controla el borde superior.
- **border-right-width**: controla el borde derecho.
- **border-bottom-width**: controla el borde inferior.
- **border-left-width**: controla el borde izquierdo.

A cada una de estas propiedades podemos asignarle alguno de los valores permitidos, que son:

- **thin**: grosor fino.
- **medium**: especifica un grosor normal. Viene determinado por defecto.
- **thick**: grosor ancho.
- **inherit**: toma el ancho del borde del elemento padre.
- **pixeles**: este es, tal vez, el más utilizado, ya que permite tener mayor control y precisión sobre el grosor que deseamos obtener.

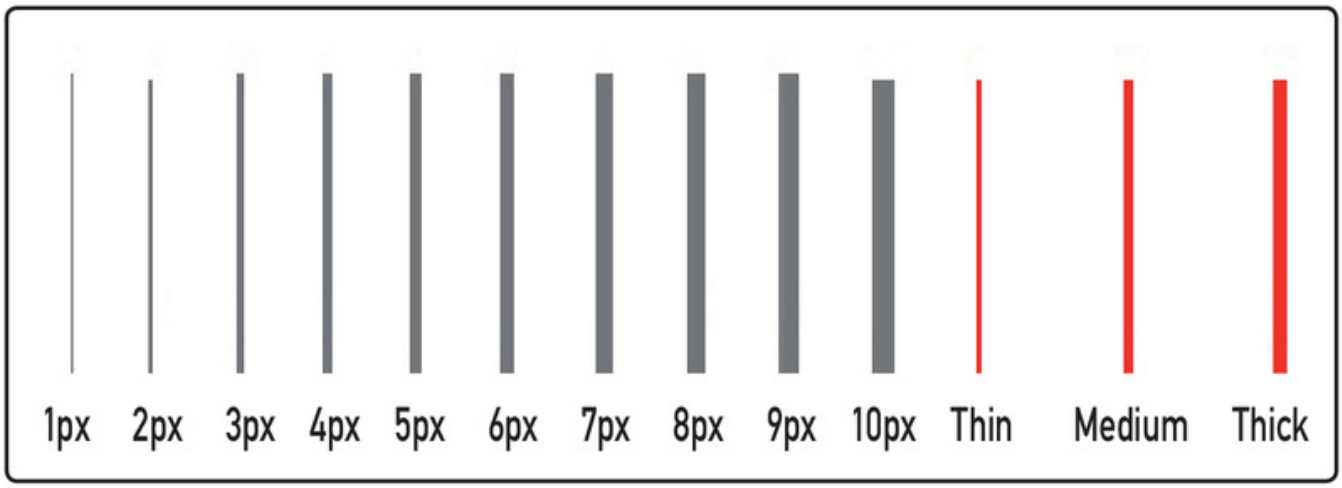


FIGURA 11. **Distintos ejemplos de grosores que podemos aplicar en los bordes. Las medidas en píxeles son determinadas por el usuario; las otras son estándar de CSS.**

A la propiedad **border-width** podemos asignarle varios valores en forma resumida, lo que nos ahorra bastante tiempo. Esta modalidad debe tenerse siempre en cuenta, ya que su sintaxis se mantiene igual y es aplicable a muchos atributos.

```
.caja{
        border-width:8px;
        width: 200px;
        height: 200px;
        background-color: gray;
        border-color: red;
        border-style: solid;
}
```

Por otro lado, cuando se indican dos valores, el primero se aplica a los bordes superior e inferior; y el otro, a los bordes derecho e izquierdo.

```
.caja{
        border-width:8px 1px;
        width: 200px;
        height: 200px;
        background-color: gray;
        border-color: red;
        border-style: solid;
}
```

Cuando se indican tres valores, el primero se aplica al borde superior; el segundo, al derecho e izquierdo; y el último, al inferior.

```
.caja{
        width: 200px;
        height: 200px;
        border-width:12px 1px 5px;
        background-color: gray;
        border-color: red;
        border-style: solid;
}
```

COLOR DE LOS BORDES

Los bordes admiten la asignación de colores a través de las siguientes propiedades:

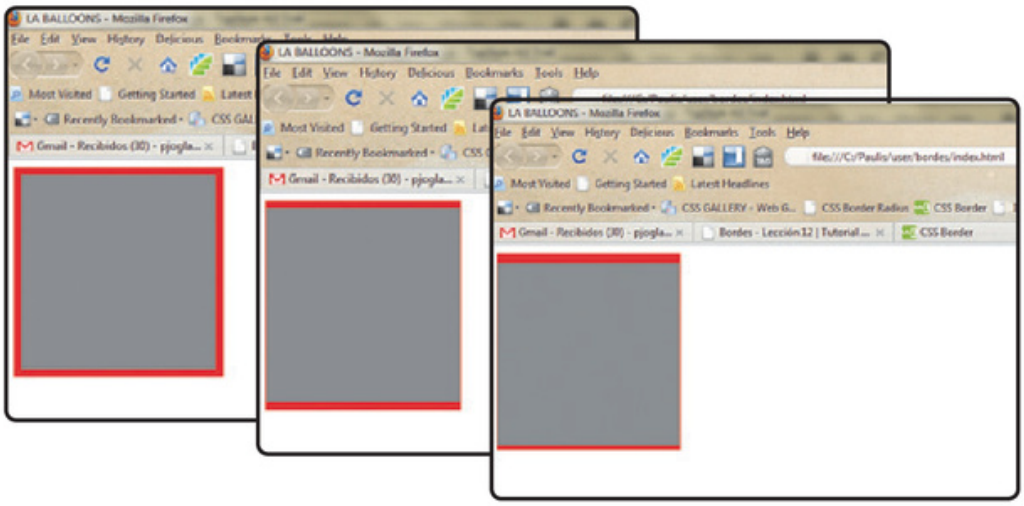
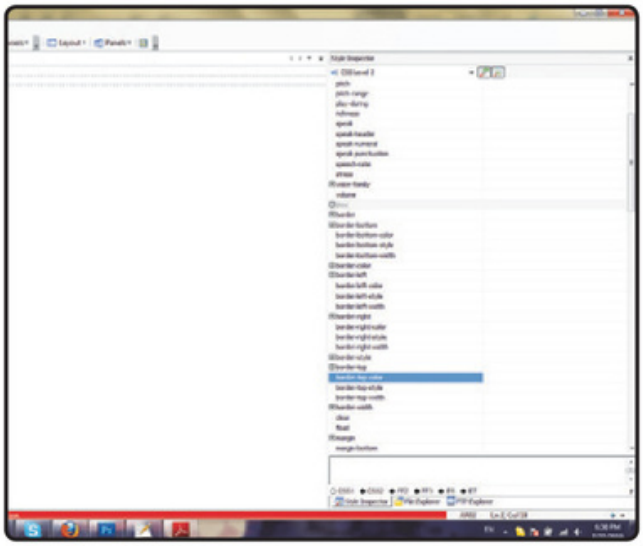


FIGURA 12. Aquí vemos el resultado de indicar un solo valor para la propiedad `border-width`. En el segundo ejemplo aplicamos dos valores y, finalmente, tres.

- **`border-top-color`**: controla el color para el borde superior.
- **`border-right-color`**: controla el color para el borde derecho.
- **`border-bottom-color`**: controla el color para el borde inferior.
- **`border-left-color`**: controla el color para el borde izquierdo.

Los valores disponibles para el color de los bordes son los que están en notación **`hexadecimal`**, por ejemplo, #123456; o los predefinidos como, White, Black, Red, etcétera.



ESTILO DE LOS BORDES

La tercera opción que podemos modificar de los bordes es el estilo. Su propiedad genérica es **`border-style`** y se asigna de la siguientes manera:

- **`border-top-style`**: controla el estilo para el borde superior.
- **`border-right-style`**: controla el estilo para el borde derecho.
- **`border-bottom-style`**: controla el estilo para el borde inferior.
- **`border-left-style`**: controla el estilo para el borde izquierdo.

Los valores disponibles para el color de los bordes se encuentran en notación hexadecimal

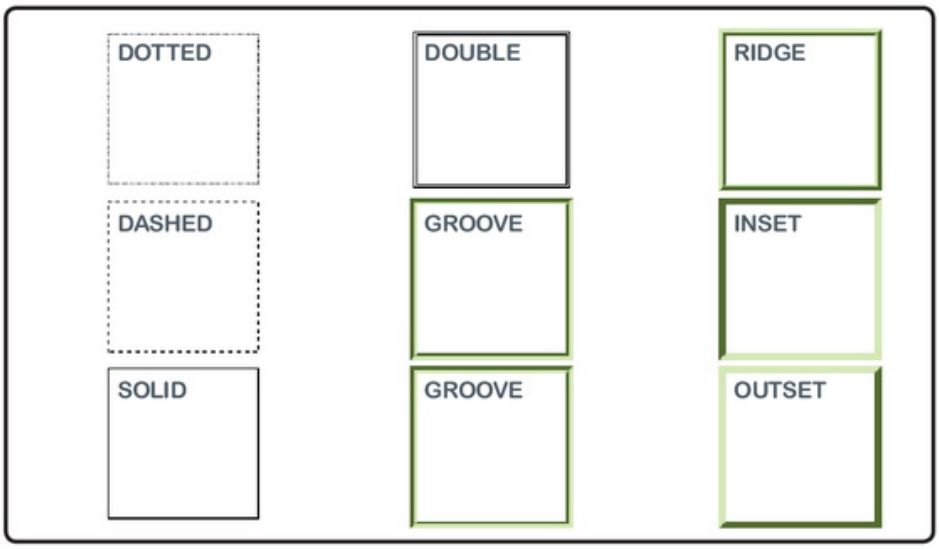


FIGURA 13. Aquí vemos los diferentes estilos de bordes que podemos aplicar mediante CSS.

Como en otros casos, podemos usar el método de escritura llamado **shorthands** para abreviar múltiples reglas CSS. De esta forma, en una sola línea asignamos varios valores a una propiedad; en el caso de los bordes, el ancho, el estilo y el color. Por ejemplo, si tuviésemos nuestro código con las siguientes declaraciones:

```
border-top-style:dotted;
border-top-width:5px;
border-top-color: White;
```

Podríamos abreviarlo de la siguiente manera:

```
border-top: 5px dotted White;
```

Si quisiésemos el mismo estilo para los cuatro bordes, tendríamos el código:

```
border-top: 5px dotted White;
border-right: 5px dotted White;
border-bottom: 5px dotted White;
border-left: 5px dotted White;
```

Gracias al método abreviado, podríamos resumirlo en una sola línea:

```
border: 5px dotted White;
```

Hacer la declaración de esta forma es lo más frecuente: ahorra tiempo de escritura y acorta el código.

PROBAR Y APRENDER

En `www.w3schools.com/css` podemos manipular el código HTML y ver, automáticamente, qué influencia visual tiene cada cambio que hagamos. Esto es muy bueno para entender bien el manejo de los diferentes atributos y su sintaxis.

Tableless

En sus comienzos, el lenguaje HTML tenía múltiples carencias, ya que no se pensaba en la Web como en un multimedio con diseños creativos, sino que la idea era transferir y compartir datos científicos. Por esta razón, no era posible alinear una página hasta darle la forma que se quisiera, sino que, simplemente, se ordenaba y se jerarquizaba el texto.

Por esta razón, los diseñadores comenzaron a usar lo único que les permitía formar de alguna manera una estructura: las **tablas**. Este recurso funcionó durante un tiempo, pero con la aparición de distintos navegadores, motores de búsqueda y diversas maneras en que la gente empezó a pensar la Web, fueron dejadas de lado.

¿POR QUÉ DEJAMOS DE LADO LAS TABLAS?

Esto se debe a que, fundamentalmente, las tablas se utilizan para presentar **contenido tabulado**, y su estructura está conformada por **filas** y **columnas**.

Si las utilizamos para la estructura de un sitio, nos encontraremos con múltiples problemas a la hora de diagramarlo, ya que deberemos **anidar** varias tablas para conseguir la disposición deseada, y tendremos que colocar el diseño y el contenido en el interior del sitio, algo que resulta poco práctico desde el diseño. Además, si maquetamos el sitio con tablas, tendremos otras desventajas:

- El contenido del HTML se vuelve mucho más pesado.
- El servidor tarda más en acceder y, por lo tanto, habrá menos gente en el sitio.
- Los buscadores indexan el sitio de manera incorrecta, porque a veces leen su contenido de modo confuso, debido a la forma en la que el contenido se distribuye en el código.
- El rediseño del sitio resulta incómodo y, en muchos

FIGURA 14. Aquí podemos apreciar un prototipo realizado con tablas. A la izquierda, la página abierta en un navegador; y arriba, el código con tablas.

casos, inaccesible, lo que hace que debamos iniciar el proceso desde cero.

- La forma de pensar la estructura del sitio es complicada y hasta impráctica a la hora de efectuar su implementación.

TABLELESS

Por los motivos que mencionamos, el uso de tablas en la estructura web se está dejando cada vez más de lado, ya que se fue migrando a **CSS+HTML/XHTML**.

En los inicios del código **CSS**, no todos los navegadores lo soportaban de forma completa y, a veces, ni siquiera lo hacían de manera parcial. Esto hizo que el paso de las tablas a una estructura dinámica constituida en su mayoría por etiquetas tomara más tiempo. Pero desde hace unos cinco o seis años, casi todos los navegadores soportan el código CSS en su totalidad, lo que permite que haya cada vez más sitios desarrollados de esta manera.

Hace más de una década que el W3C (World Wide Web Consortium) se ha encargado de normalizar los **estándares web**, lo que facilita la accesibilidad y la correcta visualización de los sitios en los distintos navegadores, en tanto estos últimos cumplan las normativas que sugiere el W3C.

Veamos, a continuación, las ventajas de utilizar CSS y dejar de lado las tablas:

- Podemos tener un **código más liviano** (en muchos casos, hasta en un 50%); por lo tanto, se favorece el tiempo de acceso del servidor.
- Es posible mantener dentro del código una

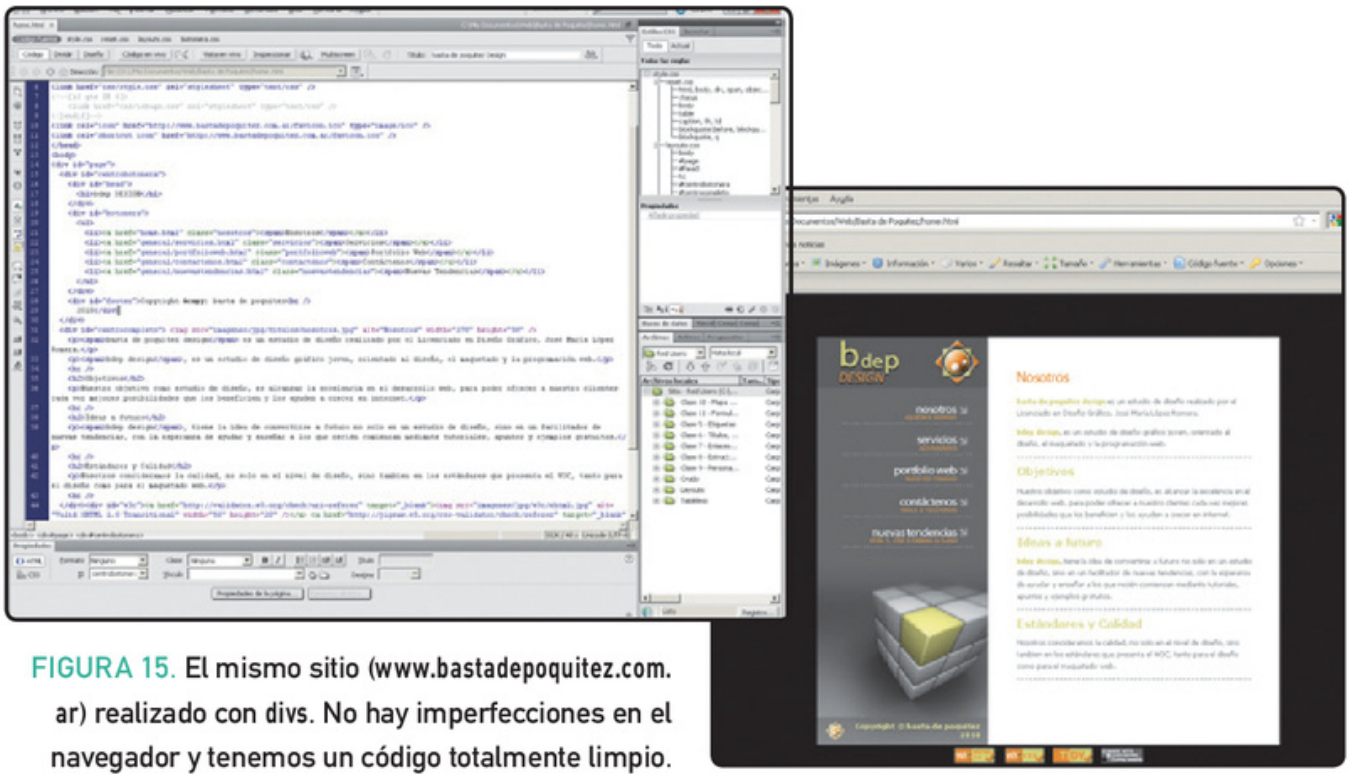


FIGURA 15. El mismo sitio (www.bastadepoquitez.com.ar) realizado con divs. No hay imperfecciones en el navegador y tenemos un código totalmente limpio.

CSS nos permite mantener dentro del código una estructura más ordenada, que solo lleva el contenido, y no, el diseño

estructura más ordenada y para nada confusa, ya que **solo hay contenido** y **no diseño**. Esto facilita la indexación del contenido web en los distintos buscadores.

- Los navegadores muestran el sitio de manera correcta, incluso, en dispositivos móviles, como Palms y teléfonos. En cambio, si utilizamos tablas, esos pequeños equipos no lo mostrarán de forma correcta.
- El diseño con CSS nos permite hacer un **cambio parcial o total** del sitio de manera fluida y práctica. Es así que se genera una optimización en el armado y el mantenimiento o rediseño.

A pesar de estas ventajas, la implementación del sistema tableless ha demorado debido a varios factores. Por un lado, las empresas que desarrollaban los navegadores no querían o no podían cumplir las normas para que CSS fuese implementado de forma óptima. Por su parte, los diseñadores, los desarrolladores y los programadores web consideraban tedioso tener que aprender un código nuevo, sobre todo, porque no se sabía si sería totalmente aceptado e implementado. Finalmente, quienes encargaban la realización de sus sitios no se preocupaban por la forma en la que estos se estructuraban, ya que lo importante era que se vieran y funcionaran.

Afortunadamente, esto ha cambiado en la actualidad. Las empresas desarrolladoras de navegadores, quienes producen los sitios (tanto diseñadores, como desarrolladores y programadores), las empresas y los usuarios se han ido dando cuenta de que la evolución a nivel web no cesará y es progresiva.

Ya no es lo mismo que algo solo funcione, sino que el funcionamiento debe ser correcto, y su estructura, simple, ordenada y de fácil modificación. Gracias a estas características, avanzamos paulatinamente hacia un código con mayor potencialidad y que nos brindará nuevos recursos. En este camino, el maquetado tableless es un paso hacia esa nueva forma de pensar, manejar y utilizar la Web.

Posicionamiento de **elementos**

Como ya vimos, el lenguaje HTML considera a todos los elementos como cajas. Si estas no se ven afectadas por reglas CSS de posicionamiento que sobrescriben su ubicación natural, se colocan en la página según su flujo normal.

Además de las propiedades predefinidas que poseen los elementos, también hay otras características que los navegadores tienen en cuenta al posicionar un elemento dentro de una página. Algunas son:

- Qué tipo de elemento es: si es en línea o en bloque.
- Cuáles son las medidas del elemento: ancho y alto.
- Las dimensiones de la ventana del navegador.
- Las relaciones entre los elementos de la página.

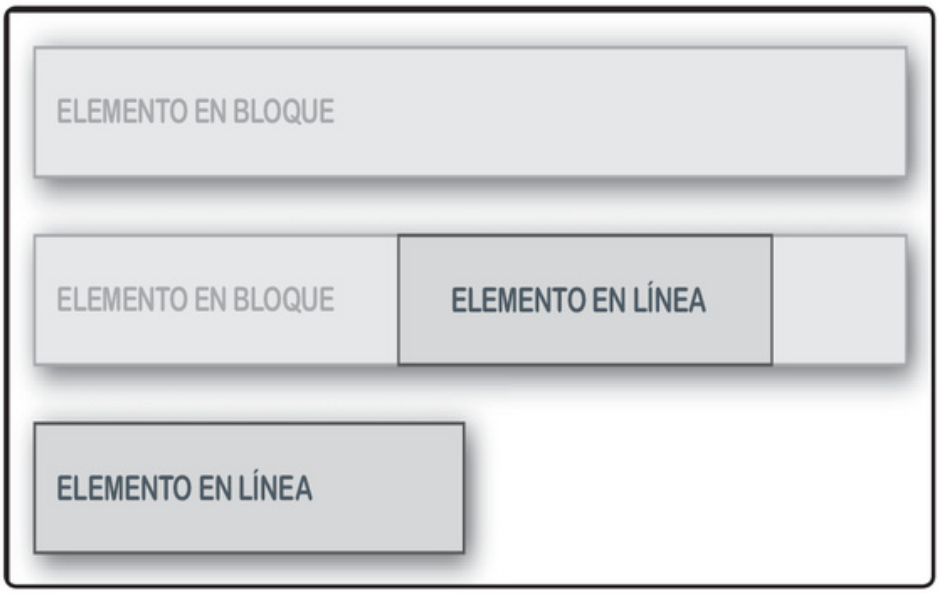


FIGURA 16. **Visualización en los navegadores del posicionamiento normal de los elementos en bloque y en línea.**

Según el estándar de CSS, existe un subgrupo compuesto por cinco propiedades empleadas para posicionar las cajas dentro de un documento HTML de diferentes maneras.

POSICIONAMIENTO NORMAL O ESTÁTICO

Es el que tienen por defecto todos los elementos de HTML según el flujo normal de la página. Si no se indica ninguna propiedad que afecte este parámetro, solo se tiene en cuenta si el elemento es en bloque o en línea.

Como ya vimos, la regla general es que los elementos en bloque pueden ubicarse dentro de otros elementos en bloque, pero nunca dentro de aquellos que son en línea. Estos últimos, en cambio, no tienen restricciones, ya que pueden ser posicionados dentro de elementos tanto en línea como en bloque.

En general, el ancho de los elementos en bloque es definido por el del elemento que los contiene. Entonces, cuando estos elementos se ubican dentro de otro del mismo tipo, este último se denomina **contenedor**, y es el que define la posición y el tamaño de sus cajas interiores.

Si el elemento en bloque no se encuentra dentro de otro elemento, entonces su tamaño y posición son definidos por la etiqueta **<body>**.

Los elementos en línea, en cambio, se ubican uno al lado del otro, y cuando no queda espacio disponible, ya sea en la etiqueta **<body>** o en otro elemento contenedor, entonces el elemento baja y ocupa una nueva línea.

POSICIONAMIENTO RELATIVO

Este tipo de posicionamiento es considerado una variable del posicionamiento normal. Por eso, toma como referencia la ubicación original del posicionamiento estático y, luego, se desplaza en los ejes X o Y, según se indique en el valor de las propiedades declaradas.

El conjunto de propiedad:valor utilizado para controlar este posicionamiento es **position:relative**; después, para manejar la distancia del desplazamiento en medidas concretas, se completa con alguna o varias de las siguientes propiedades: **top**, **right**, **bottom** y **left**. Todas ellas pueden tener valores tanto positivos como negativos, por ejemplo, **left: -10px;**.

Analicemos un ejemplo para comprender el comportamiento de esta propiedad. El código CSS es:

```
.caja1 {
        position: relative;
        top: 30px;
        left: 15px;
        background-color:#d5f9f4;
        width: 50px;
        height: 50px;
        margin:20px;
}
.caja2 {
        position: normal;
        background-color:red;
        width: 50px;
        height: 50px;
        margin:20px;
        left:10px;
}
```

El bloque de código HTML es el que presentamos a continuación:

```
<div class="caja1">caja1</div>
<div class="caja2">caja2</div>
<div class="caja2">caja2</div>
```

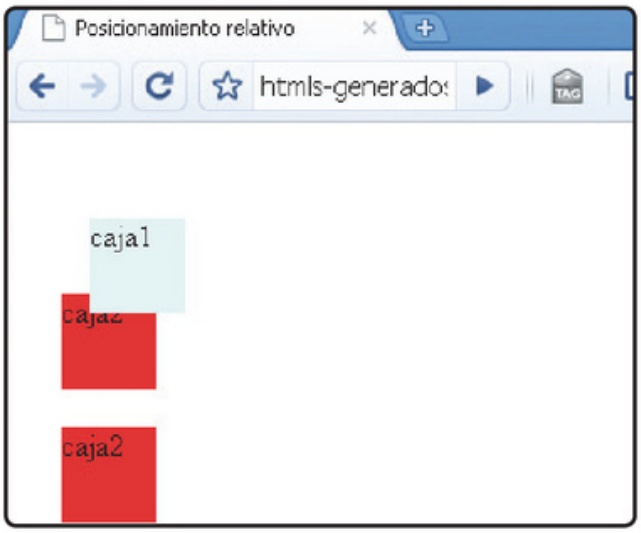


FIGURA 17. **Visualización del elemento caja1 con posicionamiento relativo en relación a dos elementos caja2 con posicionamiento normal.**

En el ejemplo anterior, la **caja1** se desplaza vertical y horizontalmente respecto de su posición normal. Como el resto de las cajas no modifica su posición, se produce una superposición o solapamiento entre elementos. También podemos observar que el resto de las cajas tampoco ocupa el espacio liberado por la **caja1**.

POSICIONAMIENTO ABSOLUTO

Al asignarle posicionamiento absoluto a un elemento, este se ubica en la página de manera independiente al resto de ellos. Es como si pasara a colocarse en un plano más cercano al espectador, y dejara en un segundo plano a los demás, que no se ven alterados por su presencia.

Con posicionamiento absoluto, un elemento se ubica en forma independiente

Para controlar la ubicación de los elementos con este posicionamiento, también se utiliza la propiedad **position**, esta vez, acompañada del valor **absolute**. Al igual que el posicionamiento relativo, se combina con las propiedades **top**, **right**, **bottom** y **left**.

A diferencia del anterior, el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor. Analicemos el ejemplo anterior, cambiando el valor del posicionamiento relativo por otro absoluto. Veamos el CSS:

```
.caja1 {
        position: absolute;
        top: 50px;
        left: 20px;
        background-color:#d5f9f4;
        width: 50px;
        height: 50px;
        margin:20px;
}
.caja2 {
        position: normal;
        background-color:red;
        width: 50px;
        height: 50px;
        margin:20px;
        left:10px;
}
```

El código HTML sería:

```
<div class="caja1">caja1</div>
<div class="caja2">caja2</div>
<div class="caja2">caja2</div>
```

FIGURA 18. **Visualización de un elemento con posicionamiento absoluto (caja1) en relación a dos cajas con posicionamiento relativo (caja2).**

La mayor diferencia respecto del posicionamiento relativo es que, al desplazarse un elemento, el resto de los que están en la página pasan a ocupar su lugar, ya que se comportan como si esta caja no existiera en el flujo normal.

POSICIONAMIENTO FIJO

Esta es una variante del posicionamiento absoluto, que no se utiliza con demasiada frecuencia en la maquetación de páginas web. La diferencia más importante es que, al recibir esta regla, el elemento afectado se convierte en una caja fija e inamovible, independiente del resto de los elementos y que no se ve afectado si el usuario utiliza el scroll de la página.

Las propiedades que se aplican son las mismas que en los dos posicionamientos anteriores, pero el valor para este caso es **fixed**. Veamos un ejemplo en CSS:

```
.caja1 {
        position: fixed;
        background-color:#d5f9f4;
        width: 50px;
        height: 50px;
}
```

Como podemos notar, para el posicionamiento relativo, absoluto, fijo y estático de una caja se utiliza siempre la propiedad **position:**, acompañada, o no, de **relative**, **absolute**, **fixed**, **inherit**.

POSICIONAMIENTO FLOTANTE

Este tipo de posicionamiento del estándar CSS tiene características particulares, y aunque es el más complejo de comprender, es uno de los más utilizados en la maquetación web. Consiste en desplazar la caja todo lo posible hacia la derecha o hacia la izquierda del espacio disponible en la página.

La propiedad de posicionamiento utilizada en este caso es **float:**, y los valores que podemos asignarle son cuatro: **left**, **right**, **none** o **inherit**. Las cajas con posicionamiento flotante dejan de pertenecer al

flujo normal de la página y se ubican lo más que pueden hacia la izquierda o hacia la derecha, según el valor asignado.

COMPORTAMIENTO DE LAS CAJAS FLOTANTES

Las cajas que poseen la propiedad **float** no se superponen entre sí, dado que siempre tienen en cuenta las demás cajas existentes en la página. Suponiendo que hay dos cajas con posicionamiento flotante de valor **left**, estas se ubicarán hacia la izquierda de la página, una al lado de la otra. Más a la izquierda, en primer término, irá la que primero se haya declarado en el código HTML, y luego, la otra. De no contar con espacio suficiente para colocarse en la misma línea, el segundo elemento saltará a la línea siguiente. El mismo caso se puede tomar en cuenta para las cajas con posicionamiento de valor **right**.

Veamos un ejemplo CSS:

```
.caja1 {
        float: left;
        background-color: #d5f9f4;
        width: 80px;
        height: 80px;
        margin: 10px;
        padding: 10px;
}
.caja2 {
        float:left;
        background-color: #d5f556;
        width: 80px;
        height: 80px;
        margin: 10px;
        padding: 10px;
}
```

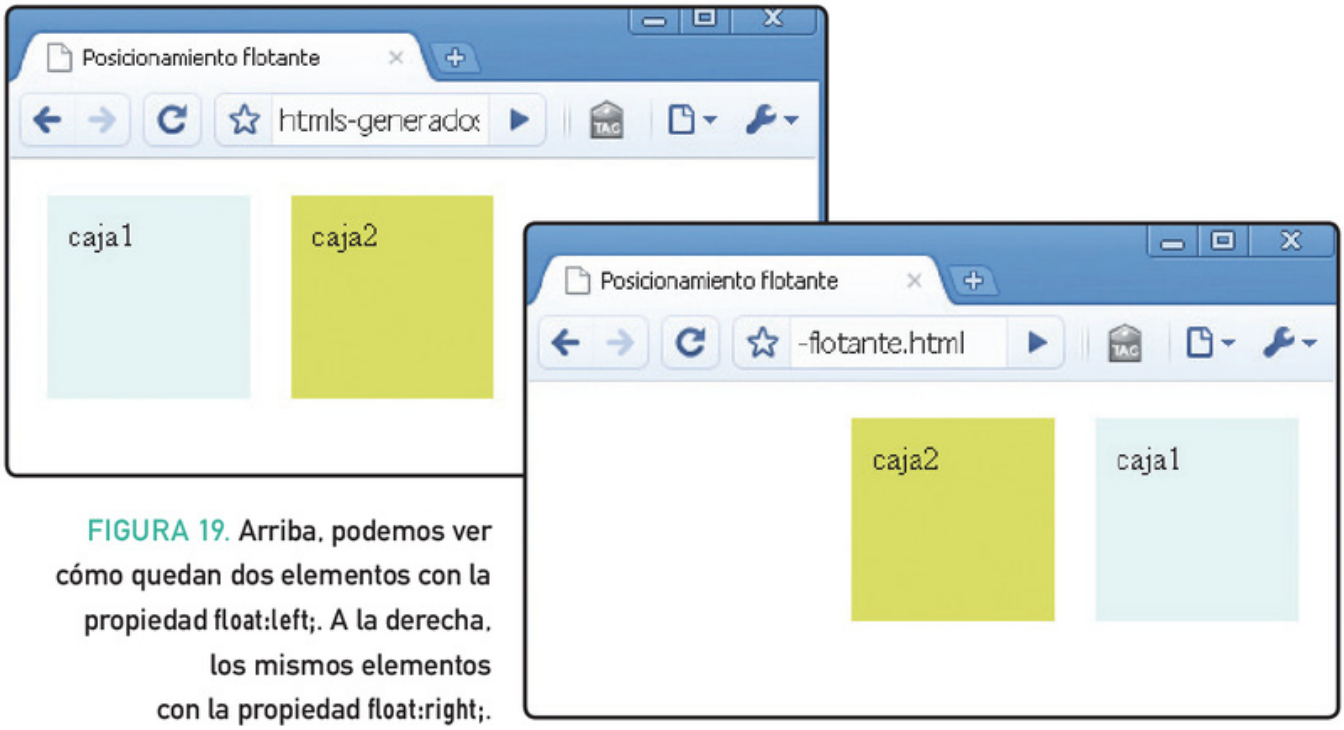


FIGURA 19. Arriba, podemos ver cómo quedan dos elementos con la propiedad float:left;. A la derecha, los mismos elementos con la propiedad float:right;.

El HTML sería:

```
<div class="caja1">caja1</div>
<div class="caja2">caja2</div>
```

CLEARFIX

La propiedad **float** de CSS es muy utilizada para posicionar elementos, pero, en ocasiones, presenta un problema. Si el elemento está ubicado dentro de otro (por ejemplo, un div sin la propiedad **float** definida), este contenedor no envuelve al elemento flotante dentro de ella y colapsa. Veamos un código HTML de ejemplo:

```
<div style="border:1px solid black;
 padding:20px;">Elemento 1 sin float
<div style="float:left; width:200px;
 background-color:#beeff4;
 padding:20px;">Elemento 2 (ubicado
dentro del elemento 1) CON float</div>
</div>
```

Para solucionar este comportamiento, existe un **hack** de CSS para que los elementos flotantes dentro de un div se adapten lo mejor posible a su elemento. La solución consiste en crear una clase **clearfix** en nuestro CSS y aplicarla a los elementos contenedores, tal como muestra el siguiente ejemplo CSS:

```
.clearfix:after {
  content:".";
  display:block;
  height:0;
  clear:both;
```

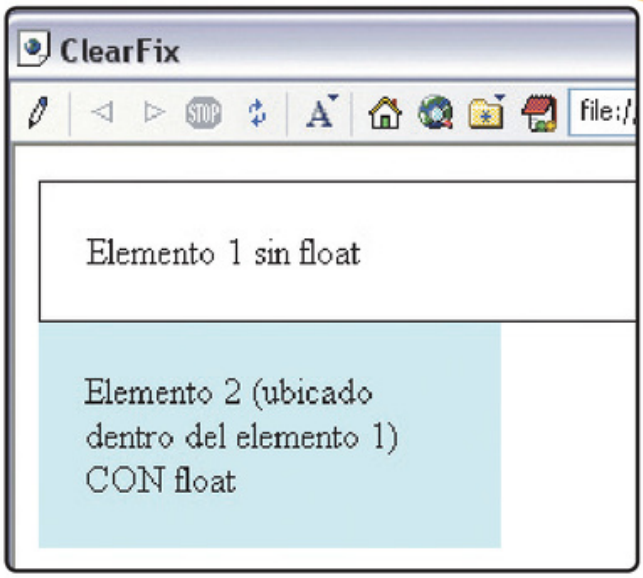


FIGURA 20. Visualización del colapso de elementos que aparece al utilizar la propiedad float de CSS.

```
  visibility:hidden;
}

.clearfix {display:inline-block;}
/* Hide from IE Mac \*/
.clearfix {display:block;}

/* End hide from IE Mac */
#divflotante {
  float:left;
}
```

El código HTML es el siguiente:

```
<div id="contenedor" class="clearfix">
  <div id="divflotante">
    Div flotante.
  </div>
</div>
```

Multiple **choice**

1 ¿Cuál es la ventaja de los gradientes?

a- Otorgan mayor profundidad al diseño.
b- Entregan una mejor compatibilidad.
c- Son más livianos.
d- Transmiten cierta atmósfera.

2 ¿Qué atributo se usa para determinar el color del fondo de un elemento HTML?

a- Background.
b- Background-color.
c- Color.
d- Transparent.

3 ¿Con qué atributo repetimos una imagen en el fondo?

a- Image-repeat.
b- Repeat.
c- Background-repeat.
d- Repeat-x.

4 ¿Qué propiedad afecta a los bordes de un elemento?

a- Image.
b- Element.
c- Color.
d- Border.

5 ¿Con qué propiedad controlamos el color del borde superior de un elemento?

a- Border-top-color.
b- Border-color.
c- Top-color.
d- Image-top-color.

6 Mencione alguna desventaja del uso de tablas.

a- No permiten usar textos.
b- No son compatibles con algunos navegadores.
c- El contenido HTML se vuelve más pesado.
d- No podemos usar imágenes.

Respuestas: 1-a, 2-b, 3-c, 4-d, 5-a, 6-c.

Capítulo 4

Textos

Las opciones para trabajar con textos y aprovecharlos en nuestros diseños serán analizadas en este capítulo.

RedUSERS.com

Texto: párrafos y títulos

Como ya sabemos, en sus primeras versiones las páginas solo estaban conformadas por algunos títulos y unos pocos párrafos, dentro de los cuales se marcaban los hipervínculos y algunos contenidos que se querían resaltar.

Las primeras especificaciones oficiales del lenguaje HTML definidas por el W3C nacieron para ayudar a los desarrolladores a estructurar y marcar los textos mediante el uso de etiquetas específicas, pensadas para asignarle mayor jerarquía y un significado lógico a cada uno de sus elementos.

ESTRUCTURACIÓN DEL TEXTO

Para estructurar correctamente los contenidos, sugerimos comenzar por analizar el texto, identificando y diferenciando los elementos que lo componen. Títulos, subtítulos y párrafos son los que vamos a etiquetar en primer término. Como referencia, podemos basarnos en el tratamiento que se le da al texto en cualquier ámbito editorial; por ejemplo, el capítulo de un libro o la noticia de un diario. Veamos ahora cuáles son las etiquetas adecuadas para definir esta estructura.

PÁRRAFOS

Una de las etiquetas más utilizadas en la maquetación de páginas es **<P>**, cuya función es delimitar los **párrafos** correspondientes a un texto. Para aplicarla –al igual que la mayoría de las etiquetas–, debemos

A través de CSS, podemos sobrescribir las propiedades que algunas etiquetas tienen por defecto, como los márgenes

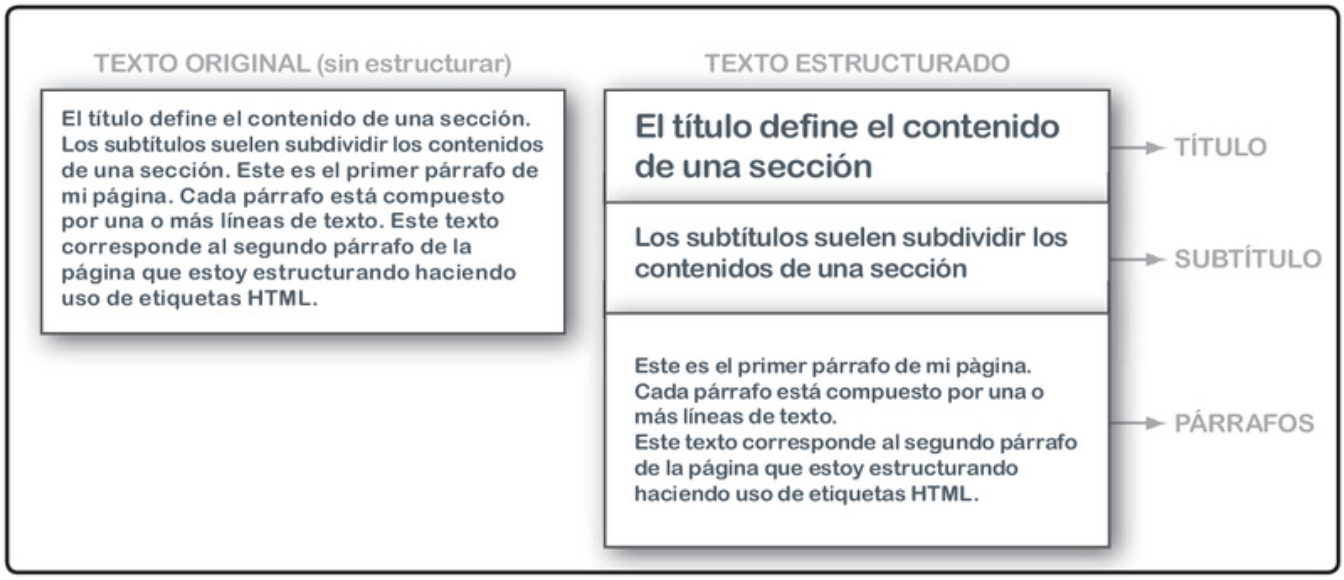


FIGURA 1. **En este texto, podemos identificar tres elementos principales: un título, un subtítulo y dos párrafos.**

encerrar con ella el texto correspondiente a cada una de esas secciones, de la siguiente manera:

```
<p>Este es el primer párrafo de mi
  página. Cada párrafo está compuesto por
 una o más líneas de texto.</p>

<p>Este texto corresponde al segundo
 párrafo de la página que estoy
 estructurando haciendo uso de
 etiquetas HTML.</p>
```

Como la etiqueta **<P>** es un elemento en **bloque**, cada párrafo comienza en una nueva línea, y deja, por defecto, un salto de línea al finalizar, lo que ayuda a separar visualmente los párrafos entre sí.

TÍTULOS

Como habitualmente las páginas están definidas por secciones más complejas que los párrafos, surge la necesidad de incorporar títulos y subtítulos que delimiten y diferencien estas secciones.

El estándar HTML define seis etiquetas que se asignan según el nivel de importancia conceptual que cada título tiene dentro del contexto en el que se encuentra. En orden jerárquico descendente, estas etiquetas son **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>** y **<h6>**.

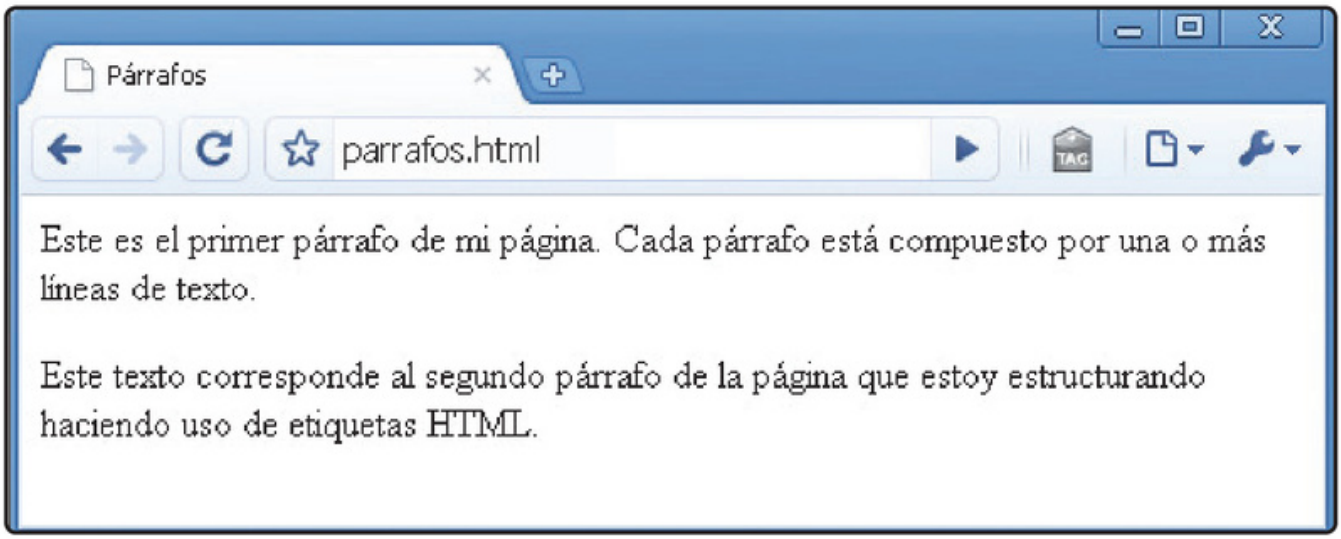


FIGURA 2. **Visualización de una página compuesta por dos párrafos encerrados por la etiqueta** <p>.

ELECCIÓN DE UN TÍTULO

La elección de un título adecuado es muy importante a la hora de atraer visitantes a nuestro sitio web. Cada sección debe contar con un título que describa y a la vez invite a los usuarios a recorrer su contenido. Un título mal seleccionado podría ser fatal para la página.

La etiqueta **<h1>** se utiliza para definir el título más importante de la página, en tanto que se usa **<h6>** para el de menor jerarquía. Esto no significa que todas las jerarquías deban usarse en la maquetación de un sitio web, porque esto dependerá de cuán rico sea el contenido y, en consecuencia, de su estructura. En páginas comunes, suele ser suficiente el uso de tres o cuatro niveles de títulos.

Si al ejemplo anterior le añadimos dos títulos con sus correspondientes etiquetas, obtenemos la sintaxis que vemos a continuación:

```
<h1>El título define el contenido de
  una sección</h1>
<h2>Los subtítulos suelen subdividir
  los contenidos de una sección.</h2>
<p>Este es el primer párrafo de mi
  página. Cada párrafo está compuesto por
  una o más líneas de texto.</p>
<p>Este texto corresponde al segundo
  párrafo de la página que estoy
  estructurando haciendo uso de
  etiquetas HTML.</p>
```

Los títulos, al igual que los párrafos, son elementos en bloque. Debido a esto, siempre comienzan en una nueva línea y se separan visualmente de los elementos que se disponen a continuación.

ESPACIOS EN BLANCO Y SALTOS DE LÍNEA

Para el lenguaje HTML, los espacios en blanco o los

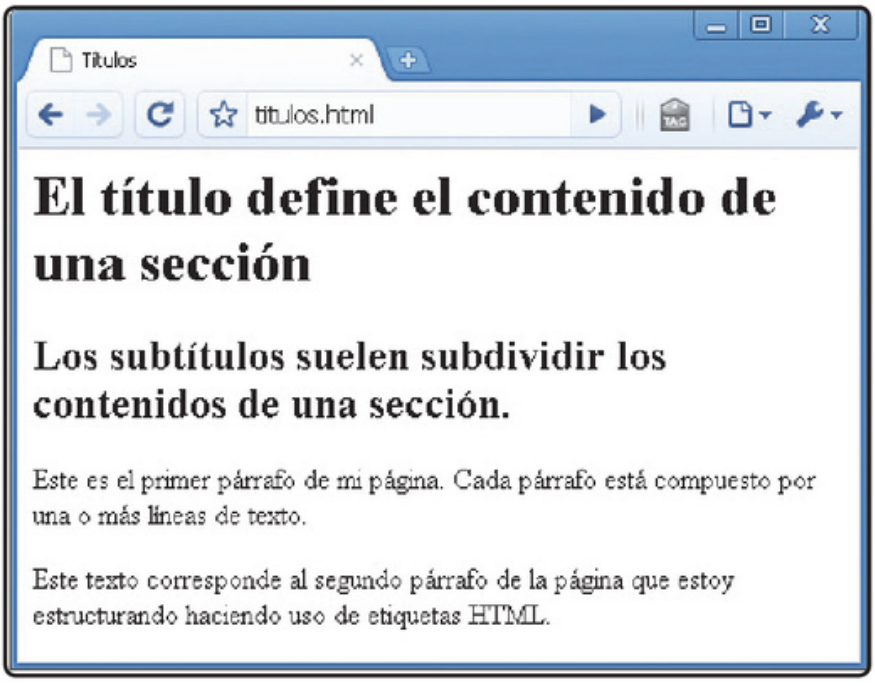


FIGURA 3. **Visualización en un browser de la aplicación de las etiquetas** <p>, <h1> **y** <h2>.

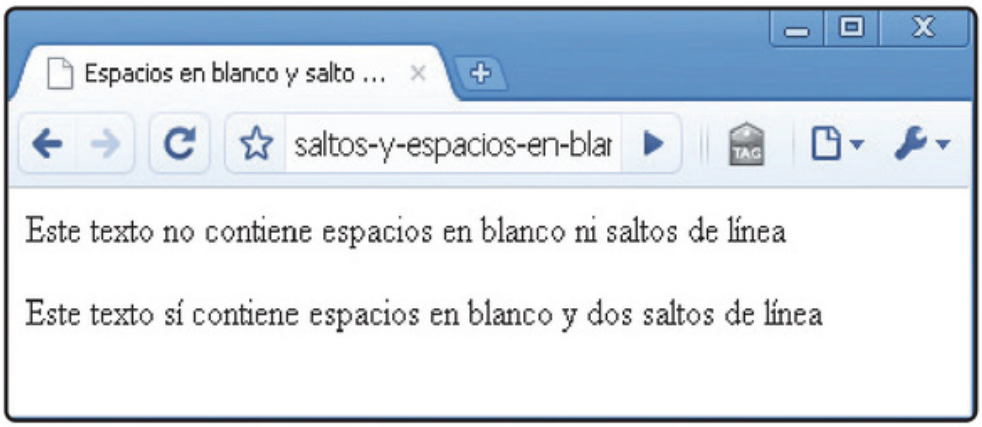


FIGURA 4. **Visualización de un texto con espacios en blanco y saltos de línea sin formateo.**

saltos de línea que aparecen dentro de una página y no reciben un tratamiento especial son ignorados al visualizarse en un navegador. La excepción son aquellos espacios compuestos por un solo carácter, que, comúnmente, separan a las palabras entre sí.

Para comprender exactamente de qué estamos hablando, podemos comparar el código escrito en un editor HTML con su visualización en un navegador. El código escrito sería el siguiente:

```
<p>Este texto no contiene espacios
   en blanco ni saltos de línea</p>
<p>Este texto sí contiene espacios
   en blanco y dos saltosde línea</p>
```

En este caso, ninguno de los espacios o saltos definidos en el código ha sido respetado por el navegador. Sin embargo, si a este código le aplicamos las etiquetas específicas que veremos a continuación, podremos lograr que los espacios sean respetados.

ESPACIOS

Para forzar los espacios en blanco entre palabras, debemos agregar el código **` `** tantas veces como sean necesarias. Tengamos en cuenta que esta porción de código genera el espacio en blanco correspondiente a un solo carácter.

Para que el ejemplo que vimos anteriormente se visualice de forma correcta, deberíamos agregar

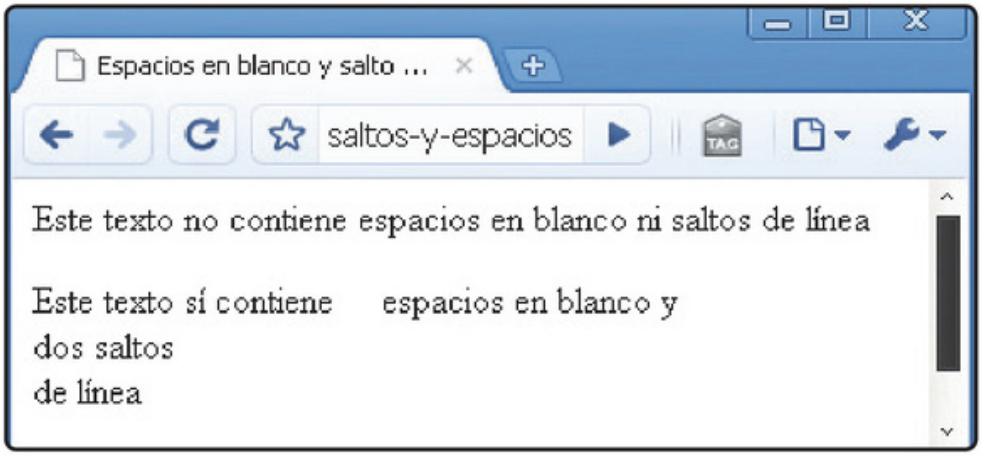


FIGURA 5. **Visualización de un texto con espacios en blanco y saltos de línea, con el formateo de texto aplicado correctamente.**

nuevas etiquetas y nuevo código, como vemos a continuación:

```
<p>Este texto no contiene espacios
  en blanco ni saltos de línea</p>
<p>Este texto sí contiene
       
  espacios en blanco y <br />dos
  saltos<br /> de línea</p>
```

Marcado básico y caracteres especiales

Con frecuencia, los textos contienen palabras o frases destacadas en negrita o cursiva, o incluyen citas textuales de otros contextos.

La mayoría de las etiquetas de marcado que utilizaremos para crear esos estilos son elementos **en línea**, para que no se produzca un salto de página, aunque existen algunas excepciones. Comencemos a analizar cuáles son.

ETIQUETA <EM>

Es un elemento en línea que se utiliza para reforzar la importancia que tiene un determinado texto dentro de un párrafo. Como resultado, los navegadores muestran los textos encerrados por esta etiqueta de manera **cursiva**, como vemos en el siguiente ejemplo:

```
<p>Este es un ejemplo, de cómo se
  utiliza la etiqueta &lt;em&gt; para
  destacar <em>este texto</em> dentro
  de un párrafo .</p>
```

ETIQUETA <STRONG>

Conceptualmente, esta etiqueta es similar a ****, pero se diferencia de ella en que esta se utiliza para remarcar el texto más importante del párrafo. También es un elemento en línea, y su visualización en los navegadores es como la negrita. Su uso es como vemos a continuación:

```
<p>Este es un ejemplo, de cómo se
 utiliza la etiqueta &lt;strong&gt;
 para destacar <strong>que este
 texto es el más importante</strong>
 del párrafo.</p>
```

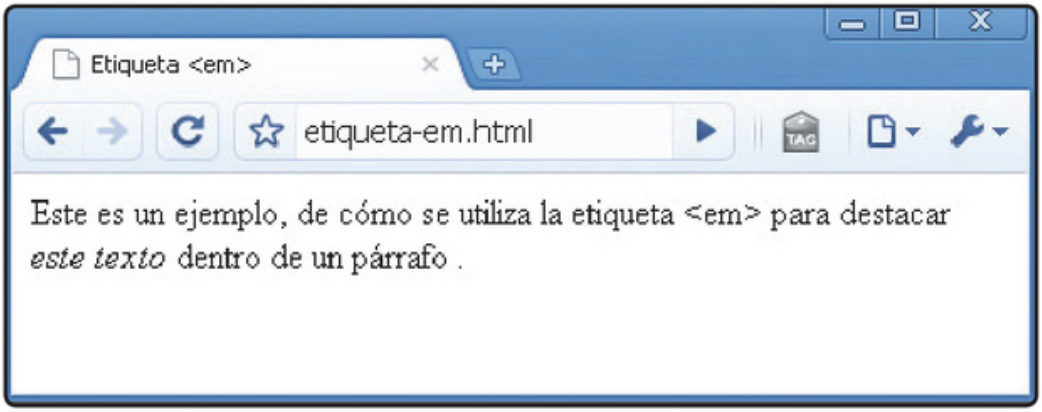


FIGURA 6. Visualización de un texto resaltado mediante el uso de la etiqueta dentro de un párrafo.

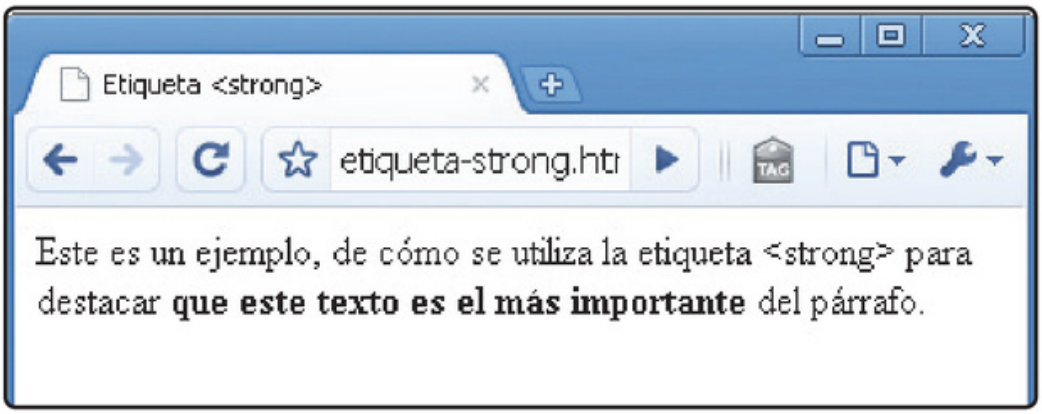


FIGURA 7.
Visualización de un texto destacado mediante el uso de la etiqueta `<strong>` dentro de un párrafo.

ETIQUETAS <INS> Y <DEL>

El uso de estas etiquetas no es demasiado frecuente. Se aplican para remarcar, dentro de un contenido, aquellas modificaciones que se han realizado respecto de su versión original.

Concretamente, **<ins>** indica qué textos fueron insertados (se ven con un subrayado) y **<del>**, cuáles fueron eliminados (se presentan de manera tachada). Ambas etiquetas son elementos en línea, y su aplicación puede complementarse mediante el uso del atributo **Datetime**, útil para adicionar información sobre la fecha y la hora en las que se realizaron los cambios. Vemos un ejemplo de la aplicación de estas etiquetas:

<ins> se encarga de indicarnos qué textos han sido insertados, mientras que <del> nos indica cuáles han sido los textos eliminados

```
<p>En este párrafo, <ins datetime=
  "20091026" cite="http://www.librosweb.
  es/mas_informacion.html"> se
  crearon nuevas palabras</ins> <del
  datetime="20091025" cite="http://www.
  librosweb.es/ mas_informacion.html">
  y se eliminaron algunas la
  versión original</del> para
  ejemplificar el uso de las etiquetas
  &lt;ins&gt; y &lt;del&gt;</p>
```

ETIQUETAS <B> Y <STRONG>

Si bien aplicando las etiquetas <b> y ←strong→ obtenemos idéntico resultado visual, la diferencia más importante entre ellas es conceptual. La función de ←b→ es meramente visual, mientras que ←strong→ señala que el texto abarcado por ella es el más importante dentro de un párrafo.

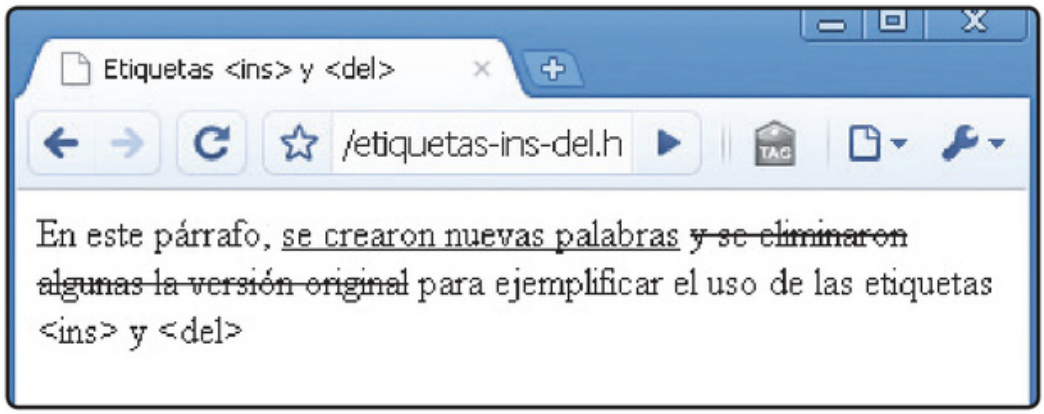


FIGURA 8. Visualización del uso de las etiquetas `<ins>` y `<del>` dentro de un párrafo.

ETIQUETAS <BLOCKQUOTE>

A diferencia de las anteriores, este es un elemento en bloque, utilizado para marcar las **citas textuales** que fueron extraídas de otros contextos. Se complementa con el atributo **cite**, que permite indicar la URL de la que se obtuvo la cita textual. Veamos un ejemplo del uso de esta etiqueta:

```
<p>Si en un texto específico, queremos
  indicar que un contenido, corresponde
  a una cita textual, debemos hacerlo
  como muestra</p>
<blockquote cite="http://www.w3.org/TR/
  html401/struct/text.html">"este texto
  que además, estamos encerrando entre
  comillas"</blockquote>
```

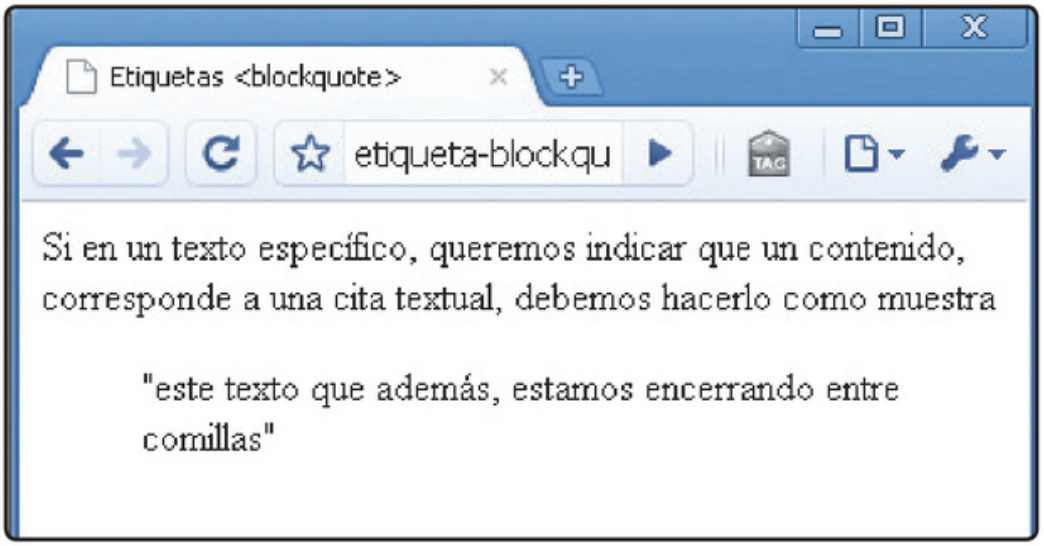


FIGURA 9. Visualización del uso de la etiqueta `<blockquote>` para indicar una cita textual dentro de un texto.

CODIFICACIÓN DE CARACTERES

Además de utilizar el marcado, al escribir un texto en un documento HTML, tenemos que ingresar algunos caracteres de manera especial, ya que no son reconocidos por los navegadores de la misma forma en que lo haría un editor de texto.

Las principales razones por las que esto sucede es que algunos de estos caracteres son usados para escribir las etiquetas HTML, de modo que los navegadores los interpretan como código en vez de tomarlos como texto normal. Por otro lado, aquellos caracteres que están por fuera del alfabeto inglés también pueden ocasionar problemas en su visualización; por ejemplo, ñ, á, ç, ¿, ¡, etcétera.

Para evitar el primero de los dos inconvenientes, al escribir determinados caracteres, debemos reemplazarlos por la entidad correspondiente del siguiente cuadro:

Por ejemplo, si queremos escribir correctamente el texto "Si escribimos en lenguaje HTML, Los paréntesis angulares <, > tal cual son, no son visualizados correctamente en los navegadores", debemos hacerlo como se muestra a continuación, para que se vea como corresponde:

```
<p> Si escribimos en lenguaje html,
  Los par&eacute;ntesis angulares &lt;,
  &gt tal cual son, no son visualizados
  correctamente en los navegadores</p>
```

El segundo problema es un poco más complejo de resolver, ya que deviene del proceso de creación de un sitio web en el que intervienen diferentes responsables. Si no se acuerda en el tipo de codificación que se va a utilizar (normalmente, llamado **charset**), pueden intervenir codificaciones distintas que generarán inconvenientes en la visualización de los caracteres.

ENTIDADES PARA REPRESENTAR SIGNOS

ENTIDAD	CARÁCTER	DESCRIPCIÓN	TRADUCCIÓN
&lt;	<	less tan	Signo de menor que
&gt;	>	more than	Signo de mayor que
&amp;	&	ampersand	Ampersand
&quot;	"	quotation mark	Comillas
	(espacio en blanco)	non-breaking space	Espacio en blanco
&apos;	'	apostrophe	Apóstrofo

TABLA 1. **Algunos de estos caracteres son reconocidos como código por los navegadores.**

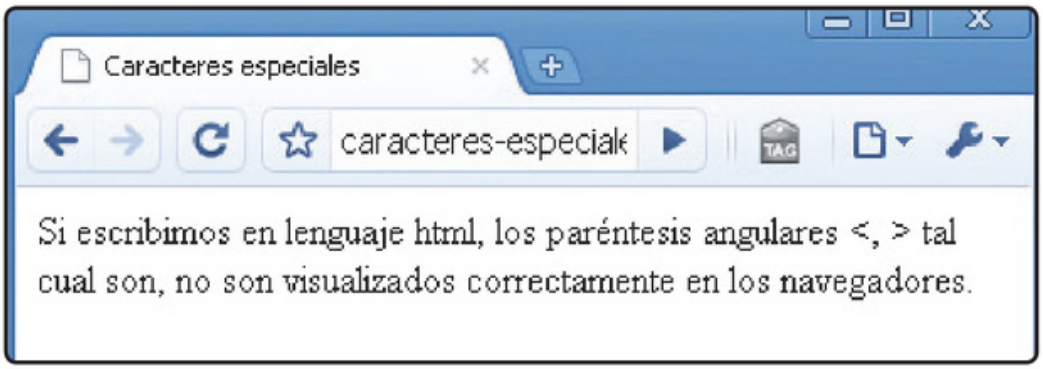


FIGURA 10. Aquí, además de la entidad de los paréntesis, tenemos que utilizar la correspondiente a la "e" acentuada.

La solución acertada consiste en emplear el mismo `charset` en todos los procesos involucrados, por ejemplo, UTF-8 o ISO-8859.

Cabe mencionar que la codificación que más se adecua a los hispanohablantes es la ISO-8859-1, y que UTF-8 es el charset por defecto en cada estándar de la W3C.

Como no siempre es posible utilizar el mismo charset, para asegurar una correcta visualización, dentro del texto podemos optar por reemplazar todos los caracteres por su entidad correspondiente.

ENTIDADES PARA REEMPLAZAR CARACTERES

ENTIDAD	CARÁCTER	DESCRIPCIÓN OFICIAL
ñ	ñ	latin letter n with tilde
Ñ	Ñ	latin capital n letter with tilde
á	á	a acute
é	é	e acute
í	í	i acute
ó	ó	o acute
ú	ú	u acute
Á	Á	A acute
É	É	E acute
Í	Í	I acute
Ó	Ó	O acute
Ú	Ú	U acute
€	€	Euro

TABLA 2. Representación correcta de los caracteres especiales o acentuados.

Por lo tanto, para escribir el texto "Es probable que existan problemas de visualización, cuando existen caracteres acentuados dentro de un párrafo.", deberíamos reemplazar los caracteres especiales de la siguiente forma:

```
<p> Es probable que existan problemas
    de visualizaci&oacute;n,
    cuando existencaracteres acentuados
  dentro de un p&aacute;rrafo. </p>
```

CSS es un lenguaje de hojas de estilo que nos permite definir parámetros de un sitio web

CSS (Cascading Style Sheets)

CSS es el lenguaje de **hojas de estilos** creado para definir determinados parámetros de un sitio web. Con él, se separa, por un lado, el contenido HTML, definiendo la función que tendrá cada elemento (como podría ser encabezado, párrafo, título, etcétera), y luego, se otorgan los aspectos visuales de cada uno de ellos.

Los objetivos de usar hojas de estilo son los siguientes:

- Definir los aspectos visuales de un sitio web.
- Homogeneizar estos parámetros en todo el sitio.

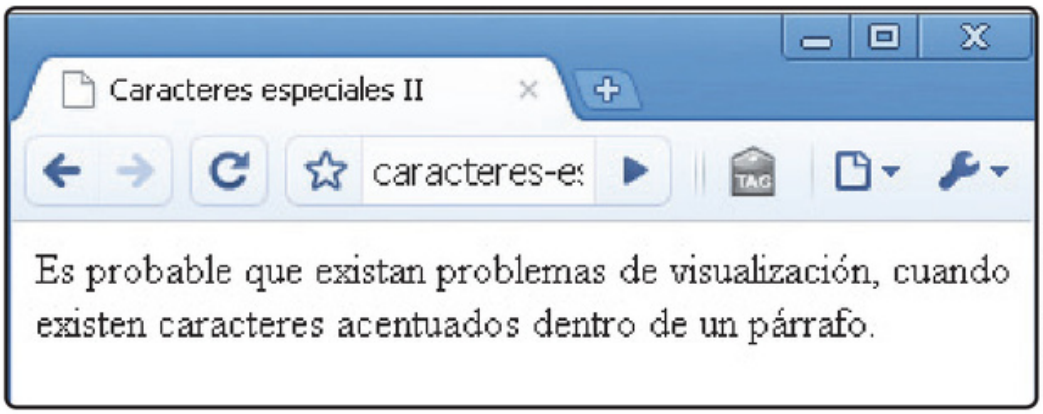


FIGURA 11. Si reemplazamos los caracteres acentuados, el texto se visualiza correctamente.

ENTIDADES HTML Y CHARSET

Las entidades HTML están compuestas por una secuencia de caracteres que devuelve un determinado carácter. Un charset es una norma que define cómo se verán ciertos caracteres en toda la página HTML.

• Actualizar o modificar su apariencia en forma clara.

Algunos de los aspectos que es posible manipular con CSS son:

• Color de fondo o imagen de fondo.
• Tipografía (tamaño, familia, color, etc.).
• Aspectos de los links y sus estados (rollover, enlaces visitados y activos).
• Márgenes de los elementos, bordes, y otros.
• Efectos JavaScript.

¿CÓMO APLICAR CSS A NUESTRAS PÁGINAS?

Entre las formas más comunes de aplicar CSS a un documento HTML, encontramos las siguientes:

• **CSS en el mismo documento**: esta forma se emplea cuando los estilos son específicos para un documento HTML o cuando son muy pocos. En este caso, se deben incluir dentro de una etiqueta **<style>** y siempre dentro de **<head>**, como vemos a continuación:

```
<head>

<style type= text/css>
 h1{
         color: Blue;
         font-size:12px;
 }
</style>

</head>
```

Tengamos presente que una de las premisas fundamentales de CSS es la practicidad a la hora de modificar valores, y aplicar a todo el sitio el mismo

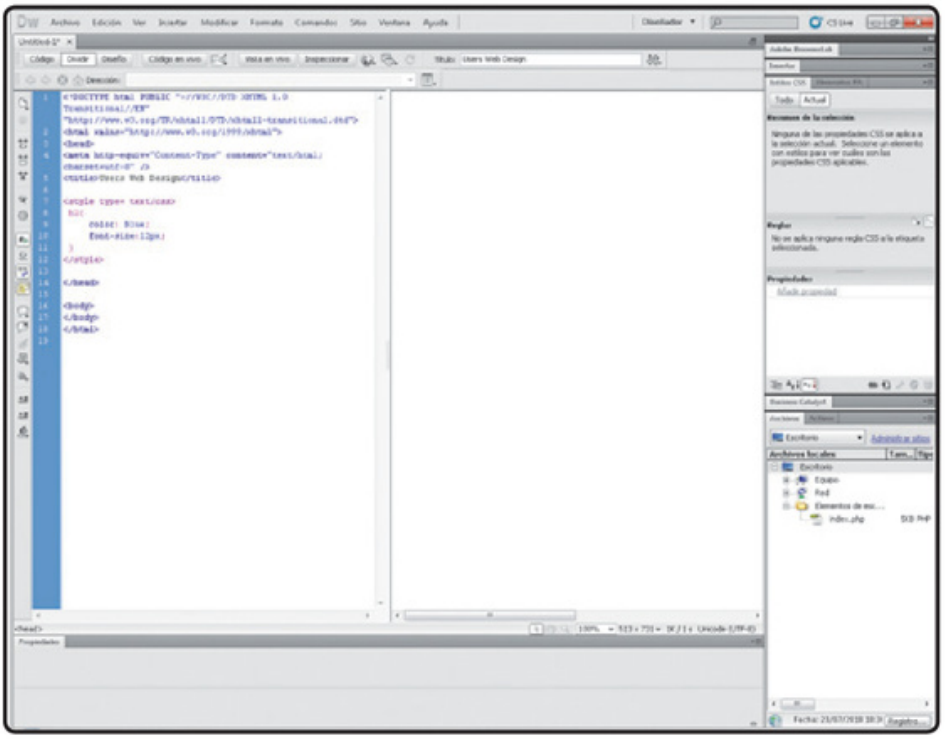


FIGURA 12. Al incorporar estilos de esta forma, solo afectaremos el documento HTML sobre el que trabajemos.

estilo. Sin embargo, si aplicamos CSS de esta manera, el cambio que realicemos solo afectará al archivo HTML en el que estemos trabajando.

• **CSS como archivo externo**: esta es la forma más usual y práctica, que consiste en aplicar un archivo CSS externo relacionado al HTML a través de la etiqueta **<link>**. Así, podremos aplicar todos los archivos CSS que hagan falta para darle estilos al documento:

```
<head>

<link href="css/styles.css"
 rel="stylesheet" type="text/css" />

</head>
```

Este es el modo de trabajo más habitual, ya que, aplicando el mismo archivo CSS a las diferentes páginas de un sitio, garantizaremos la homogeneidad de estilos. Esto también permite que, al variar algún valor de cualquier selector, este se aplique a todo el sitio de forma rápida y sencilla.

Un CSS externo permite actualizar los estilos del sitio de forma sencilla, modificando solo el archivo CSS

• **CSS en elementos HTML**: es un método que no resulta recomendable, aunque es necesario conocerlo. Este tipo de aplicación de CSS solo se usa en circunstancias en las que, específicamente, es necesario aplicar un estilo a un elemento determinado. Por ejemplo:

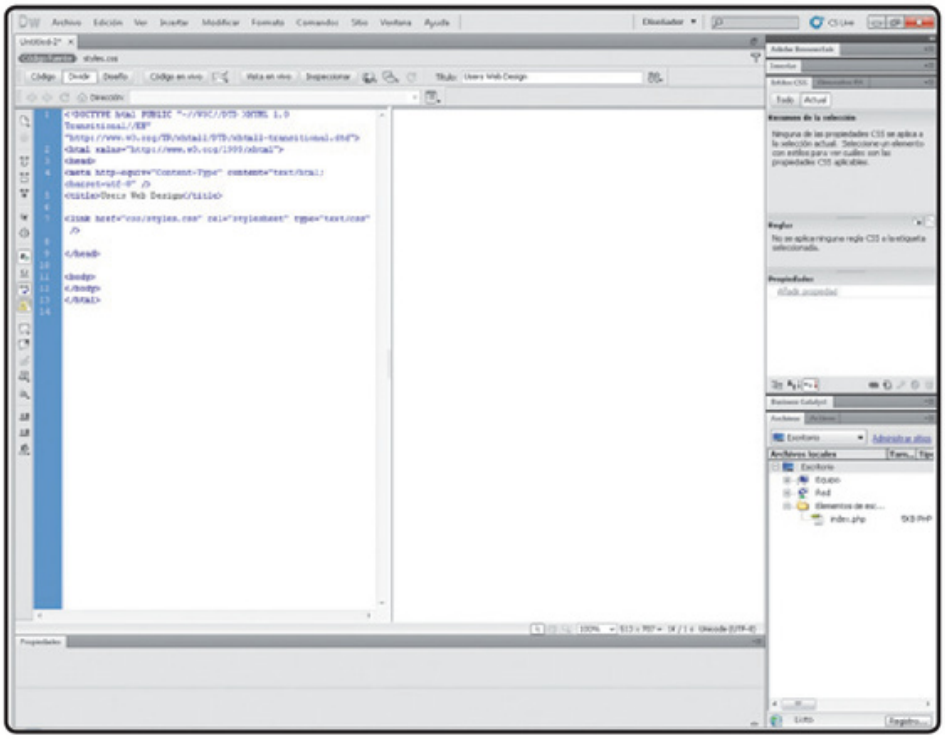


FIGURA 13. Si enlazamos un archivo CSS de esta forma, al hacer cambios en él, los veremos aplicados en todas las páginas donde esté enlazado.

```
<p style="color: black; font-family:
 Verdana;">Un párrafo de texto.</p>
```

SINTAXIS DE ATRIBUTOS Y PROPIEDADES EN CSS

Cuando creamos un estilo, estamos definiendo todas las características que tendrá un elemento HTML, es decir, su apariencia visual, su **estética**.

La sintaxis de CSS es muy simple y se mantiene siempre de la misma manera. Consta de un **selector**, que indica a qué elemento o etiqueta se le aplicará el estilo; y de **declaraciones** (una sola o varias), formadas por el nombre de la **propiedad** y un **valor**. Ambas partes, selector y declaración, determinan una **regla CSS**.

Es muy importante tener en cuenta la forma de escribir los estilos: primero, el **selector** y, luego, las **declaraciones**, que siempre van entre llaves ({ }). La declaración está formada por la **propiedades**, seguidas por dos puntos (**:**); y luego están los **valores**. Si hay más de una declaración, entonces utilizamos un punto y coma (**;**) y la nueva declaración.

FIGURA 14. **Esta es la sintaxis de CSS utilizada con todas las propiedades.**

Para que el código sea legible y fácil de recorrer con la vista, suele ponerse una declaración bajo la otra. Entonces, un archivo CSS quedaría de la siguiente manera:


```
body{
  color: #666;
  font-size: 11px;
  font-family: "Trebuchet MS", Arial,
    sans-serif;
  background-color: #4bc5ee;
  background-image: url(../img/bg_body.
    jpg);
}
```

Selectores

Nuestra hoja de estilos CSS contendrá distintos tipos de **selectores**, que son los que le indican al navegador de Internet dónde aplicar los estilos que definimos.

Cada regla CSS definida puede aplicarse a cuantos elementos HTML necesitemos y, a su vez, cada elemento HTML puede utilizar cualquier cantidad de reglas que hayamos creado. Aunque estos conceptos son un tanto confusos al principio, se irán aclarando a medida que avancemos y practiquemos con ellos.

SELECTOR UNIVERSAL

Este selector se usa para aplicar estilos a todos los elementos HTML de una página, y se indica mediante un asterisco.

Los selectores le indican al navegador de Internet dónde aplicar los estilos definidos

```
* {
        height:50px;
}
```

El ejemplo anterior establece un alto de 50 píxeles para todos los elementos de las páginas a las que se encuentre vinculada nuestra hoja de estilos. Aunque es una forma sencilla de trabajar, no es muy utilizada porque no es común que el mismo estilo pueda aplicarse a todos los elementos de una página.

SELECTOR DE TIPO O ETIQUETA

Se utiliza para aplicar estilos a todos los elementos HTML que coincidan con el nombre del selector. Para usarlo, solo es necesario nombrar el elemento HTML (tag) sin los caracteres de apertura y de cierre, es decir, sin < ni />. Por ejemplo, a continuación definimos un borde de color gris para todos los elementos **div** de nuestra página:

```
div {
        border:1px solid #CCC;
}
```

Si queremos aplicar el mismo estilo a más de un elemento HTML, podemos encadenar los selectores, separándolos por una coma. Con el siguiente ejemplo veremos un borde gris en todos los elementos **div**, en todas las imágenes y en todos los párrafos de la página:

```
div, img, p{
        border:1px solid #CCC;
}
```

En las hojas de estilo complejas, suelen agruparse los estilos comunes a varios elementos en una única regla y, luego, se definen los estilos específicos de cada elemento HTML por separado. Siguiendo con el ejemplo anterior, podemos lograr que todos los elementos **div**, las imágenes y los párrafos de nuestra página se vean con un borde gris, y que solo los elementos **div** tengan una altura de 50 píxeles. Veamos cómo:

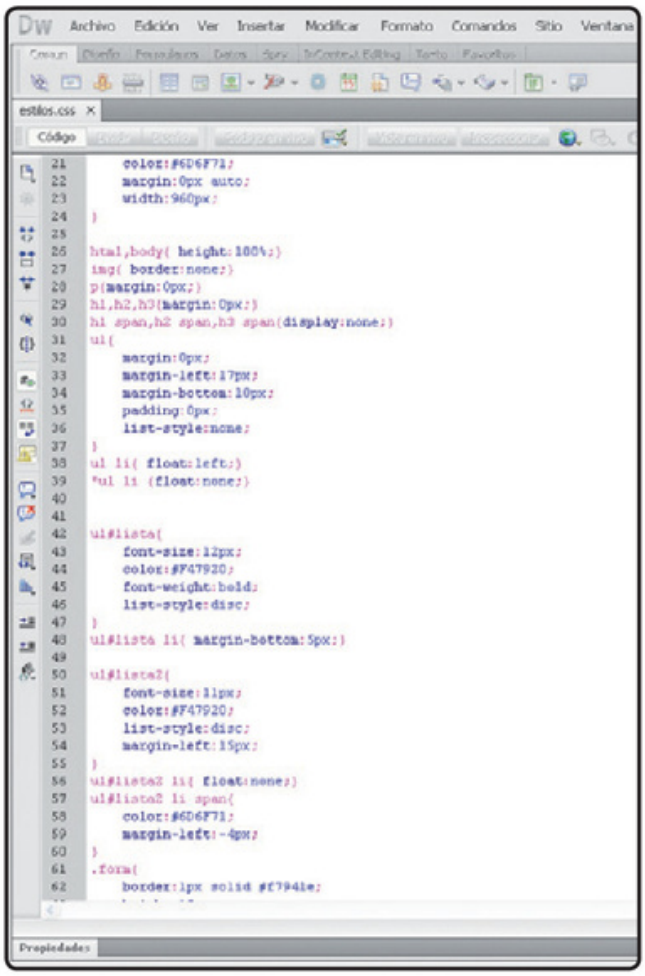


FIGURA 15. **Hoja de estilos del sitio www.namuntu.com.**

```
div, img, p{
        border:1px solid #CCC;
}
div{
        height:50px;
}
```

SELECTOR DESCENDENTE

Utlizamos este selector para seleccionar un elemento HTML que se encuentra dentro de otro del mismo tipo.

Los selectores descendentes siempre están formados por dos o más selectores, separados entre sí por un espacio. El último selector indica el elemento HTML al que se le debe aplicar el estilo. A continuación, aplicamos color rojo a todos los elementos **** que estén dentro de un párrafo (**<p>**). A aquellos que estén por fuera de los párrafos, no se les aplicará este estilo:

```
p span{
        color:#F00;
}
```

SELECTOR DE CLASE

Se define utilizando un punto al comienzo del selector, forma en la que el navegador de Internet lo distingue de los otros tipos de selectores. Supongamos que tenemos el siguiente código HTML:

```
<ul>
        <li>Primer lugar</li>
        <li>Segundo lugar</li>
        <li>Tercer lugar</li>
</ul>
```

Si quisiéramos aplicar un estilo solo al primer elemento **<li>**, nos sería imposible hacerlo con los tipos de selectores que conocimos hasta ahora. Una de las soluciones es aplicar el atributo **[class]** sobre el elemento HTML al que le queremos aplicar el estilo. Entonces, nuestro código HTML quedaría así:

```
<ul>
        <li class="principal">Primer
   lugar</li>
        <li>Segundo lugar</li>
        <li>Tercer lugar</li>
</ul>
```

Luego, definimos el estilo (en este caso, fijamos la tipografía de color rojo), utilizando un selector de clase, de la siguiente manera:

```
.principal{
        color:#F00;
}
```

De igual modo, cualquier elemento HTML que tenga definido su atributo **[class]** como principal, tendrá su tipografía de color rojo. Por ejemplo:

```
<span class="principal">Hola</span>
<p class="principal">Chau</p>
```

Ahora bien, si quisiéramos la tipografía roja solo en los párrafos cuyo atributo **[class]** ha sido definido como principal, deberíamos definir la regla CSS así:

```
p.principal{
        color:#F00;
}
```

FIGURA 16. Arriba, podemos ver el sitio www.namuntu.com con diferentes estilos aplicados en los elementos HTML. A la derecha, el mismo sitio sin la hoja de estilos.

SELECTOR DE ID

Este tipo de selector es igual al de clase, aunque lo aplicaremos a un único elemento HTML de nuestra página, porque no se permite usar varios atributos **id** con el mismo valor.

Para un solo elemento HTML, también podríamos utilizar un selector de clase, pero en ese caso nos convendrá emplear este selector de ID porque es más eficiente.

Para utilizar este tipo de selector, debemos definir el atributo **[id]** del elemento HTML al que le queremos aplicar nuestro estilo. La sintaxis es muy parecida a la del selector de clase, pero en vez de un punto, utilizaremos el numeral o **sharp** (#). En nuestro ejemplo, veremos el código HTML:

```
<ul>
        <li>Primer lugar</li>
        <li id="nuevo">Segundo
   lugar</li>
        <li>Tercer lugar</li>
</ul>
```

Y el respectivo código CSS:

```
#nuevo{
        color:#F00;
}
```

Al igual que en los selectores de clase, podemos restringir el alcance del selector combinando el de ID con otros, por ejemplo:

```
a#importante{
        font-size:14px;
}
```

Hasta aquí, hemos visto los distintos tipos de selectores que utilizaremos en nuestra actividad como diseñadores profesionales de páginas web. Este tipo de definiciones resultan complejas al principio, pero con el correr de las horas de práctica y trabajo, nos iremos encontrando, cada vez con mayor frecuencia, con este tipo de selectores, lo que hará que nos familiaricemos con ellos y que hagamos su lectura con suma naturalidad.

Multiple **choice**

1 **¿Qué etiqueta nos permite delimitar los párrafos?**

a- <V>
b- <P>
c- <Z>
d- <W>

2 **¿Para qué se usa la etiqueta <h1>?**

a- Para agregar un título a una imagen.
b- Para crear una tabla.
c- Para integrar una imagen.
d- Para definir el título más importante de la página.

3 **¿Cómo forzamos un espacio en blanco entre palabras?**

a- •nbsp
b-
c- %nbsp
d- $nbsp

4 **¿Qué hace la etiqueta ?**

a- Muestra el texto en cursiva.
b- Muestra el texto en negrita.
c- Muestra el texto tachado.
d- Oculta el texto.

5 **¿Qué carácter obtenemos con &oacute?**

a- é.
b- ú.
c- ó.
d- á.

6 **¿Qué carácter obtenemos con &ntilde?**

a- w.
b- x.
c- Ñ.
d- ñ.

Respuestas: 1-b, 2-d, 3-b, 4-a, 5-c, 6-d.

Capítulo 5

Imagen

En este capítulo, revisaremos la forma en que podemos agregar y trabajar con imágenes en nuestros diseños web.

RedUSERS.com

Imágenes en HTML

Prácticamente, todos los sitios web contienen imágenes, en mayor o menor cantidad. Entre ellas, encontramos dos tipos: las de contenido y las de complemento o adorno.

Las imágenes de **contenido** son las que agregan información adicional a la información textual que tengamos en la página, como fotos, gráficos, ilustraciones, etcétera. Las incluiremos en el código HTML mediante la etiqueta **<img>**.

Las imágenes de **complemento** o **adorno** son las que utilizaremos para hacer, por ejemplo, cajas con bordes redondeados, fondos con degradé, iconos en listados o en links, etcétera. No deberíamos incluirlas en el código HTML, sino a través de hojas de estilo CSS. Esto responde a dos razones principales. En primer lugar, cuando queramos modificar el estilo del sitio, lo haremos directamente desde la hoja CSS. En segundo, les quitaremos tarea a los robots de los buscadores al indexar solo las imágenes que son importantes para el sitio.

La etiqueta que utilizaremos para incluir imágenes en el código HTML es <img>

ATRIBUTOS OBLIGATORIOS DE LA ETIQUETA <IMG>

Como dijimos, la etiqueta que utilizaremos para incluir imágenes en el código HTML es **<img>**, que tiene dos atributos requeridos, u obligatorios, y cuatro opcionales.

El primer atributo obligatorio es **src**, que proviene de la abreviatura de source (fuente, en inglés). A través de él, le indicamos al navegador la URL de la imagen que queremos mostrar. Esta puede ser absoluta o relativa, tal como vimos cuando conocimos las rutas.

El segundo atributo obligatorio es **alt**, que se utiliza para describir el contenido de la imagen en un texto breve de hasta 1024 caracteres.
Veamos, a continuación, un ejemplo de uso de la etiqueta **<img>**:

```
<img src="logo.jpg" alt="Nombre de la
  empresa" />
```

ALT

Originalmente, el atributo `alt` se utilizaba como sustituto de la imagen en caso de que esta no pudiera ser mostrada por diferentes razones (estar deshabilitada desde el navegador, error de conexión, etcétera). Un uso importante que le damos hoy es para ayudar a los buscadores.

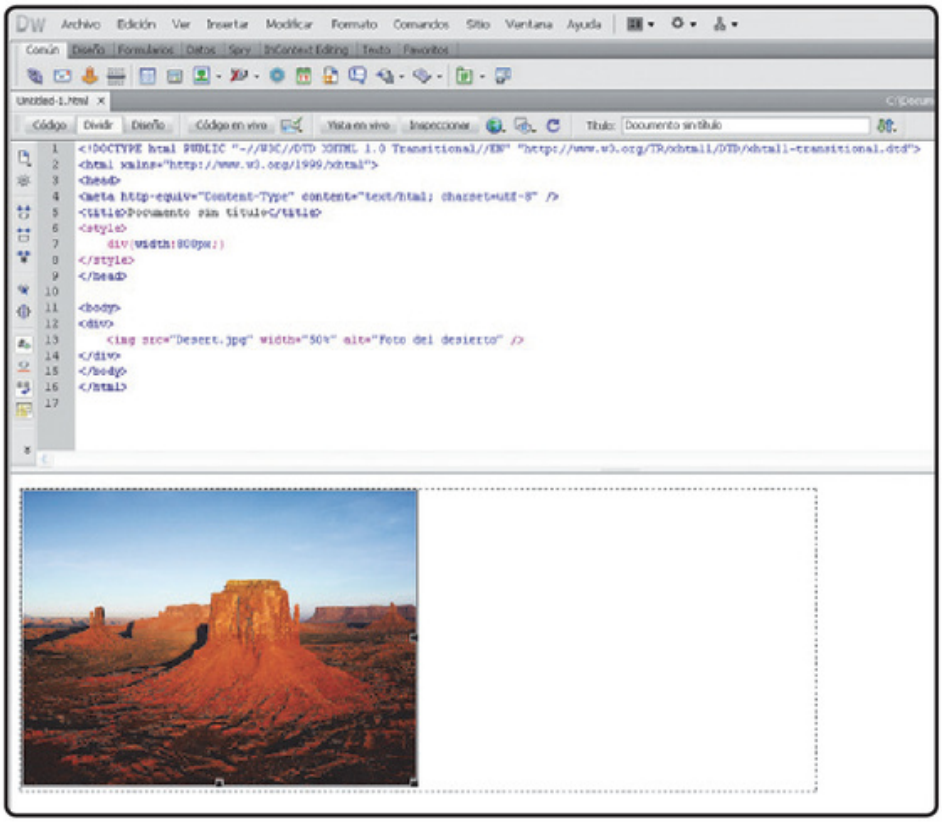


FIGURA 1. En este ejemplo, definimos una imagen cuyo ancho es la mitad del de su elemento contenedor, es decir, 400 píxeles.

ATRIBUTOS OPCIONALES

El atributo **longdesc** es muy poco usado y sirve para especificar dónde encontraremos más información sobre la imagen. Está pensado para complementar el límite de 1024 caracteres que tiene el atributo **alt**.

Veamos un ejemplo:

```
<img src="unafoto.jpg" alt="Esta es una
  descripción de la foto"
  longdesc="servicios/mas_info.html" />
```

El atributo **name** sirve para establecerle un nombre al elemento imagen, en tanto que **width** y **height** se aplican para definir el ancho y el alto con el que se muestran las imágenes en la página HTML.

Estos atributos pueden parecer un tanto contradictorios porque, como hemos visto antes, todo lo referido al aspecto gráfico de la página debe definirse en la hoja de estilos CSS. Por lo tanto, en principio, el ancho y el alto de una imagen también deberían estar definidos allí. No obstante, es virtualmente imposible que CSS determine los anchos y los altos de todas las imágenes que tengamos en un sitio, ya que la

hoja de estilos aumentaría su tamaño según la cantidad de imágenes presentes, lo que resultaría contraproducente. Por esta razón, los atributos **width** y **height** son la excepción a la regla de que el código HTML no debe hacer referencia al aspecto gráfico de los elementos. Si el valor de estos atributos es un **número entero**, el navegador de Internet interpretará que hacemos referencia a una medida en **píxeles**. A continuación, vemos una imagen con un ancho de 320 píxeles y un alto de 240 píxeles:

```
<img src="images/foto1.jpg" alt="Foto de
  una persona" width="320" height="240"
  />
```

También es posible definir el ancho y el alto de una imagen con **medidas relativas**, indicando un **porcentaje**. En este ejemplo, mezclamos los dos tipos de medidas que podemos utilizar: el ancho aquí equivale a la mitad del ancho del elemento contenedor (en este caso, un div), y el alto será de 240 píxeles.

```
<div>
        <img src="images/foto2.jpg"
 alt="Foto de otra persona" width="50%"
 height="240" />
</div>
```

Los valores que definimos en los atributos **width** y **height** no tienen por qué coincidir con los valores de ancho y alto reales de la imagen. Sin embargo, si estos valores no coinciden, las imágenes se mostrarán deformadas y con un aspecto que lucirá desprolijo.

En caso de que definamos solo uno de los dos atributos, el navegador de Internet calculará el restante para que se mantenga la proporción original de la imagen.

Si bien **<img>** es una etiqueta HTML que no contiene cierre, para que la página XHTML sea válida, todas las etiquetas deben estar cerradas. Por lo tanto, para cerrar la etiqueta de imagen, utilizamos **/>**.

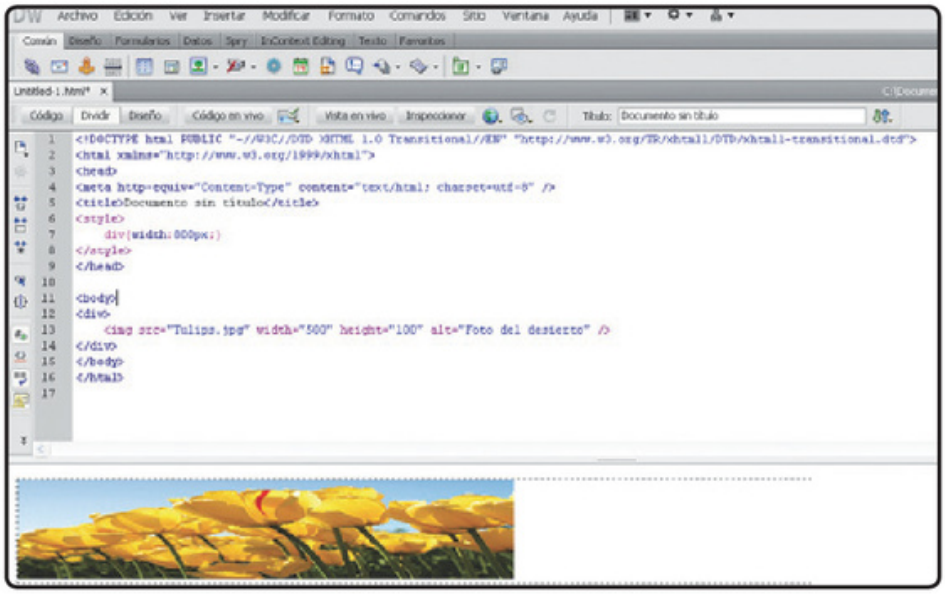


FIGURA 2. **Mal uso del ancho y el alto de la imagen. Los valores asignados están forzando el tamaño de la imagen, con lo cual la deforman.**

Tipos de **imágenes**

Los tres formatos de imágenes más comunes son **GIF**, **JPG** y **PNG**. No podemos decir que uno de ellos sea mejor que otro porque, en realidad, elegiremos el adecuado según la imagen que vayamos a insertar en el HTML.

GIF

La sigla GIF proviene de Graphics Interchange Format, o formato de intercambio de gráficos. Este formato utiliza un máximo de 256 colores, por lo que no es adecuado para fotografías de mediana y alta resolución. En cambio, es muy usado para imágenes con grandes áreas de un mismo color, pequeños iconos o **bullets** e imágenes de complemento o

No existe un formato de archivo de imagen mejor que otro, sino que cada uno tiene un uso ideal

adorno, como cajas con bordes redondeados y degradés cortos.

La principal ventaja del formato **GIF** es que permite reemplazar un color por transparencia, dándonos mayor flexibilidad a la hora de aplicarlo sobre un fondo HTML. Además, en archivos de baja resolución, su ventaja sobre **JPG** es que, al no comprimirlos, conserva con mayor nitidez la imagen por mostrar.

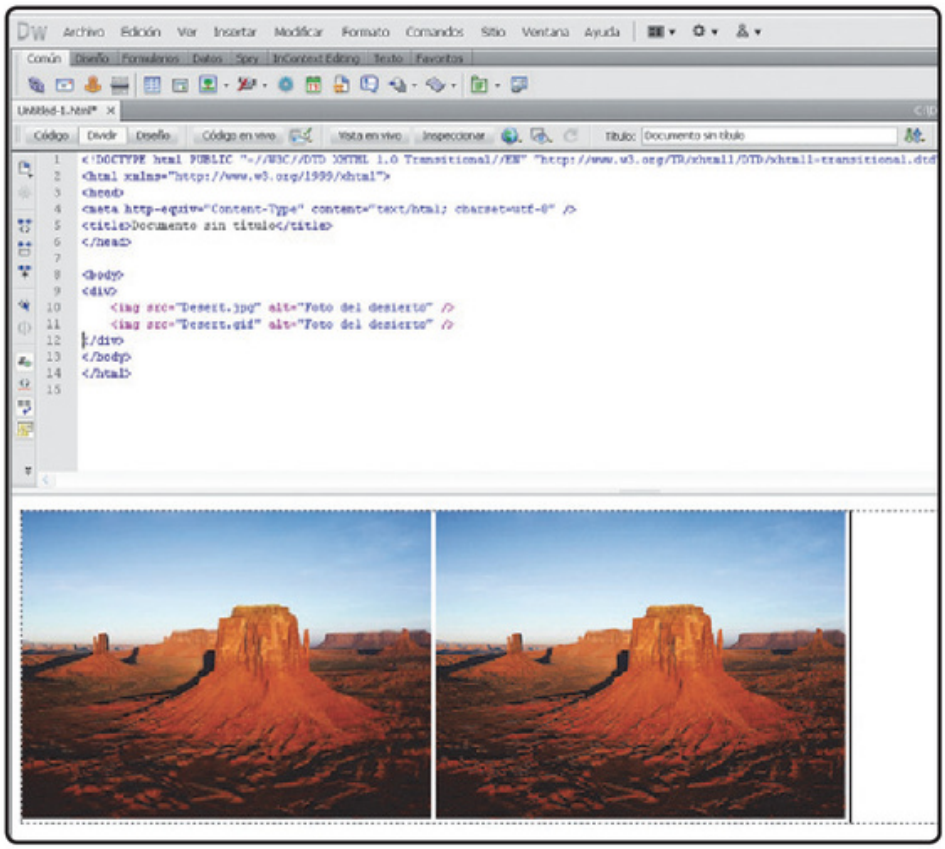
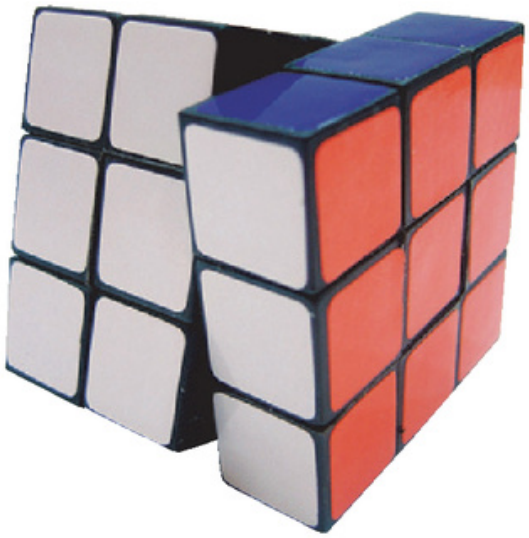


FIGURA 3. A simple vista, estas imágenes parecen iguales. Sin embargo, la de la izquierda es un JPG que pesa 23 KB; y la de la derecha, un GIF cuyo peso es de 59 KB.



JPG

También conocido como JPEG, por Joint Photographic Experts Group, o grupo de expertos en fotografía, este formato soporta hasta 16 millones de colores, de modo que resulta ideal para fotografías o gráficos.

Tiene una muy buena relación entre el peso del archivo y la calidad de la imagen, lograda mediante su algoritmo de compresión, que reemplaza varios píxeles por un área del mismo tamaño, con un color promedio. Cuanta mayor compresión de imagen tengamos, más calidad resignaremos, pero ganaremos en la reducción del peso del archivo.

Una de las ventajas del formato **JPG** es lo que se conoce como **modo progresivo**, que los navegadores interpretan mostrando, en primer lugar, una versión en baja calidad de la imagen, para luego ir mejorándola a medida que se va cargando.

PNG

La sigla de este formato proviene de Portable Network Graphics, o gráficos portátiles de red. Este formato es muy usado, ya que las imágenes se comprimen sin pérdidas visibles de calidad.

La principal ventaja de los **PNG** es que admiten imágenes de 24 bits, incorporando un **canal alpha** que da la posibilidad de utilizar transparencias totales o parciales dentro del archivo.

La desventaja es que no es compatible con todos los navegadores. En algunos más viejos (como Internet Explorer 6), al usar **PNG** veremos la imagen pero no su transparencia, que será reemplazada por un cuadro gris.

Este formato también soporta el **modo progresivo** pero, además, tiene soporte para **metadata** (o metadatos), información adicional que poseen los archivos **PNG** para ser indexados por los motores de búsqueda. Esta información no es visible en la imagen, sino que está oculta dentro del código fuente del archivo.

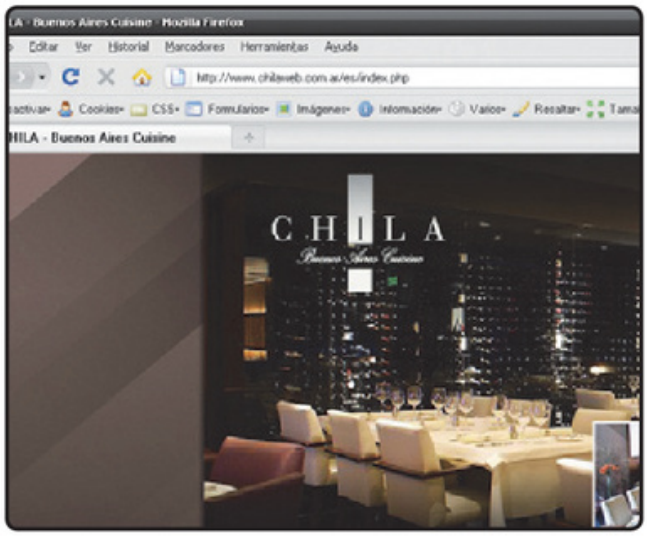


FIGURA 4. En www.chilaweb.com.ar, el logo ubicado arriba a la izquierda es un PNG cuya transparencia se aplica sobre el fondo de imagen.

Optimización de imágenes **para la Web**

Los diseñadores gráficos están acostumbrados a trabajar con resoluciones de 300 dpi (dots per inch, puntos por pulgada). En diseño web, debemos darles importancia al ancho y al alto de las imágenes que utilizaremos, y no tanto a la resolución, ya que la de cualquier monitor es siempre de aproximadamente 72 dpi.

Si utilizamos 300 dpi para una imagen que incluiremos en una web, obtendremos imágenes demasiado pesadas y que no se verán mejor que si tuvieran el mínimo de resolución.

Una imagen pesada ocupará mayor espacio en el servidor, generará más tráfico de datos, tardará más tiempo en cargarse y, probablemente, agote a los usuarios del sitio cuando tengan que esperar su carga.

El peso de un archivo de imagen dependerá, directamente, de la cantidad de colores, su tamaño y la resolución. Entonces, cualquier acción que recorte estos parámetros implicará una reducción en el peso del archivo y, por lo tanto, del tiempo de descarga necesario para visualizarla en cualquier navegador de Internet. Al guardar el archivo, debemos elegir el formato adecuado entre los tres más habituales para uso web: **GIF**, JPG y **PNG**.

JPG VS. GIF

El formato **JPG** soporta 256 colores como mínimo y 16 millones como máximo, por lo que la reducción de peso por este lado es irrelevante. Lo que permite ajustar mejor el peso de un archivo **JPG** es la **compresión**. Al aplicarla, es aconsejable ir probando cuál es la mayor compresión que podemos obtener sin corromper visualmente la imagen.

En cambio, para los archivos **GIF**, la cantidad de colores es importante, ya que podemos tener 256 como máximo. En este caso, mediante la reducción de colores, podremos disminuir el peso del archivo.

72 dpi

Cuando cambiemos el ancho y el alto de las imágenes por medio de algún programa de edición, deberemos asegurarnos de que, además del tamaño, estemos modificando la resolución para que esta sea de 72 dpi.

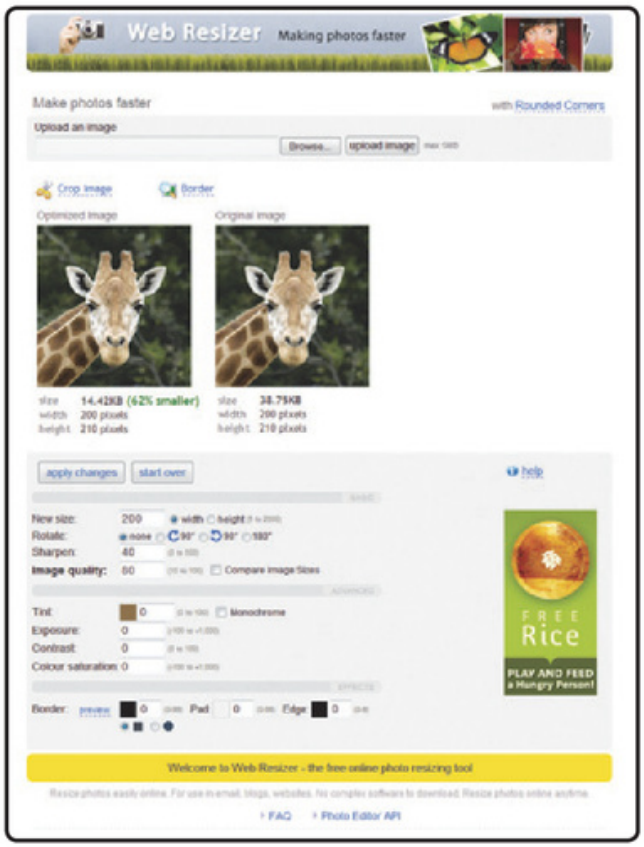


FIGURA 5. **Web Resizer (www.webresizer.com) es un servicio web que permite optimizar imágenes sin recurrir a ninguna aplicación.**

Muchas veces, encontraremos que, visualmente, no existe diferencia entre un archivo **GIF** con 256 colores y otro con 128, y de esa forma, podremos reducir el peso hasta en un 60%.

Aunque Photoshop es una gran herramienta a la hora de optimizar imágenes para la Web, existen otras utilidades online y offline que permiten realizar esta tarea. Entre las opciones online, una muy recomendable es Web Resizer (**www.webresizer.com**), que se utiliza para reducir la resolución de una imagen (y, por consiguiente, su peso) como Web Service, sin tener que descargar nada.

Propiedades de las imágenes **en CSS**

Cuando se empezaron a utilizar imágenes en los sitios web, estas tenían muy pocas propiedades. Sin embargo, con el correr del tiempo, se expandieron y se volvieron más flexibles, gracias al empleo de las reglas CSS.

Al insertar una imagen en el espacio que le hemos designado dentro del HTML, esta obtendrá, por defecto, solo dos propiedades. La primera es **[width]**, que determinará su ancho en píxeles; y la segunda es **[height]**, para la altura.

Otra que podemos agregarle a la etiqueta **<img>** es la propiedad o atributo **[alt]**, que sin duda es una de las más importantes, ya que con esta información, si el navegador no consigue cargar la imagen, mostrará un texto que hará referencia a ella.

PROPIEDADES CSS

Las propiedades que le otorga CSS a una imagen son variadas. No solo podemos modificar **[width]**

Photoshop es una aplicación importante pero existen utilidades web que nos ayudarán, como www.webresizer.com

y **[height]**, sino que también podemos darle propiedades de **margin** para establecer su marginado e, incluso, aplicarle propiedades de borde mediante **[border]**. Esto resulta útil si la imagen es un enlace, dado que, por defecto, dicho elemento tiene atribuida la propiedad **[border: 1px solid]**, que podemos hacer que no se vea o que únicamente se vea en uno de sus lados.

También podemos utilizar la propiedad **[float]**, que nos permitirá hacer flotar la imagen y ubicarla donde consideremos más conveniente. Esta opción es práctica cuando queremos colocar algo al lado de la imagen sin necesidad de generar un div, sino, simplemente, dándoles a las etiquetas contiguas la propiedad **[float]**.

Por su parte, **[resize]** permite redimensionar, con lo cual nos da la posibilidad de mostrar la imagen en un tamaño diferente del original. Esta es una propiedad que se ha agregado a CSS3.

Para finalizar, hablaremos de una propiedad que, en muchos casos, resulta controvertida. Se trata de **[background]**, mediante la cual una imagen puede ser ubicada o posicionada como background dentro de casi cualquier etiqueta. En este caso, si lo usamos como fondo, podremos generar mosaicos por repetición, escribir encima de él y hasta producir sprites con una única imagen.

Aunque es una práctica muy utilizada para generar los fondos de los sitios o, incluso, los menús

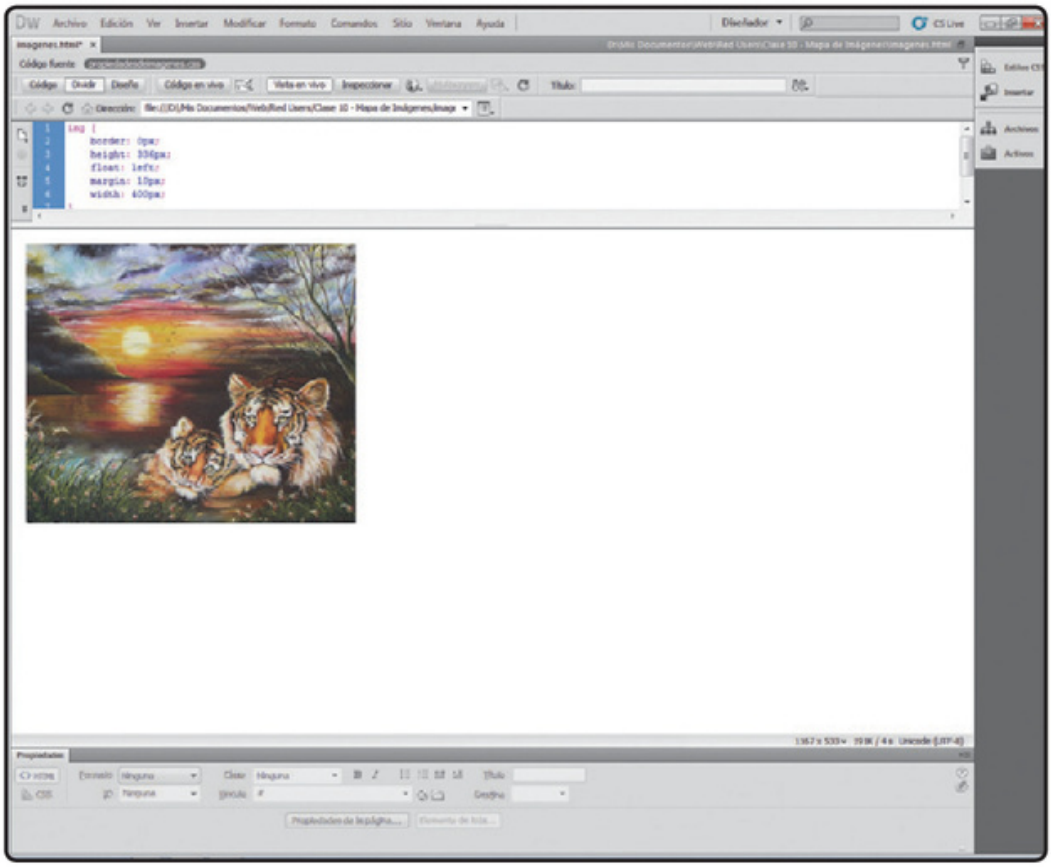


FIGURA 6. Aquí podemos observar en Dreamweaver una imagen con algunas propiedades en el código CSS.

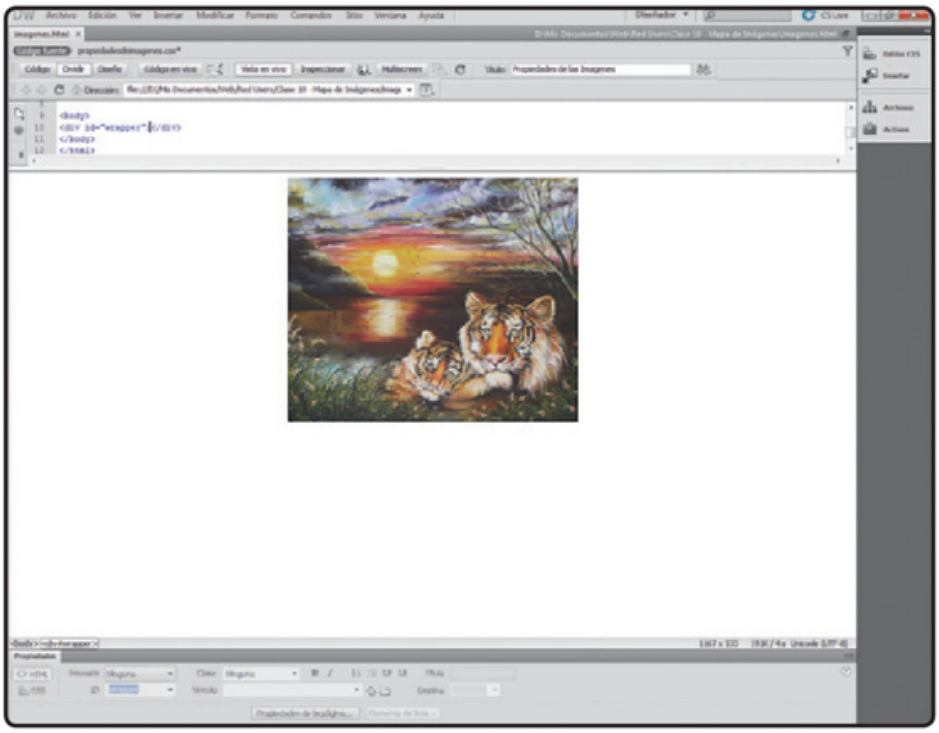


FIGURA 7. Aquí podemos observar la misma imagen colocada como background. En el código HTML veremos que no hay ninguna etiqueta `<img>`.

de navegación, debemos tener en cuenta que no es favorable para los motores de búsqueda, ya que les impide indexar la imagen como información. Por este motivo, debemos utilizar esta propiedad solo cuando es realmente necesario, para no restar la posibilidad de lograr un mejor posicionamiento en los buscadores. Recordemos que estos, a la hora de indexar, ingresan todo aquello que puedan leer como información, algo que no sucede cuando una imagen está colocada con la propiedad **`[background]`**).

CSS sprites

La técnica de **sprites** de imágenes se ha empleado durante años, incluso, antes de que se la pusiese en práctica para desarrollar partes de un sitio web. En sus inicios, fue utilizada para generar los movimientos básicos de los videojuegos elaborados en 2D, que mediante programación algorítmica, mostraban al personaje moviéndose y haciendo lo que nosotros deseábamos. Un ejemplo concreto de esto es el juego Mario Bros, que a lo largo de más de una década, usó sprites en sus diversas versiones.

¿QUÉ SON LOS SPRITES DE IMÁGENES?

Se trata de una técnica que consiste en tener varias construcciones de imágenes dentro de una única imagen. En el ámbito web, esto podría significar incluir varias posiciones de un mismo botón, como

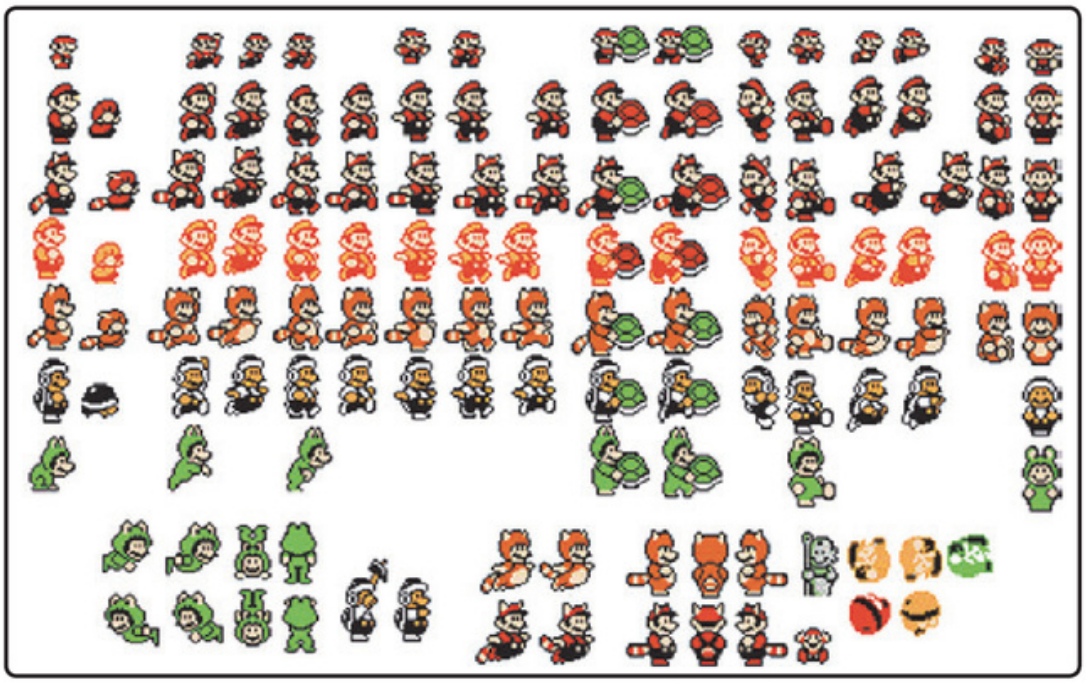


FIGURA 8. En esta imagen, podemos observar el sprite utilizado para el juego Super Mario 3.

su estado off, hover y on. Otra utilidad es colocar, en una sola imagen, todas aquellas que utilizaremos repetidas veces en un sitio web, tal como hace Google, en su buscador, o Yahoo Mail, en su casilla de correo.

¿CÓMO SE UTILIZAN LOS SPRITES DE IMÁGENES?

Si bien al principio el uso de sprites no es sencillo, con algo de práctica y atención, se transformará en una herramienta útil y de fácil implementación. Para utilizarla, primero debemos colocar la imagen como **background**; es decir, si se trata de una botonera, la imagen funcionará como fondo de ella. Luego, mediante la propiedad **[background-position]**, vamos asignando las distintas ubicaciones que se mostrarán de la imagen mediante los ejes **X** e **Y**. El primero corresponde a la línea horizontal, también llamada **ancho** de la imagen; y el segundo, a la línea vertical, o **alto** de la imagen. Tengamos en cuenta que pueden establecerse con valores positivos o negativos.

VENTAJAS DE ESTA TÉCNICA

La técnica de sprites es muy efectiva y, cuando nos familiaricemos con ella, es probable que sea una de las primeras opciones que evaluemos a la hora de realizar una gran cantidad de imágenes que se repitan en un sitio, tanto para los botones como para alguna porción del fondo (es menos empleada para este último fin, pero hay quienes la aplican).

Los ejes X e Y pueden tener valores tanto positivos como negativos, algo fundamental en la técnica de sprites

Como ventajas significativas, debemos mencionar que, al tener todo agrupado en una, dos o tres imágenes, su carga es más rápida. La transferencia de datos entre el servidor y la máquina del usuario se acelera, lo que le brinda al visitante una mayor velocidad de respuesta al acceder al sitio.

Dos ejemplos que debemos tener en cuenta son los mencionados anteriormente: el buscador de Google y Yahoo Mail.

El primero solo utiliza dos imágenes para generar la representación gráfica. En este caso, nos concentraremos en la imagen con la que se generan los resultados de búsqueda. Esta no tiene más que 150 píxeles de ancho por 106 de alto con formato **PNG** y fondo transparente, lo que permite colocarla sobre cualquier tipo de fondo (también podría haber sido un **GIF** con fondo transparente). Esta imagen está constituida por la marca, las flechas hacia adelante y hacia atrás, las correspondientes a arriba y abajo,

FIGURA 9. Podemos observar el sprite que se utiliza para el buscador Google (www.google.com); se trata de un archivo PNG.

UTILIDADES WEB

Si no tenemos ganas de crear las imágenes de sprites por nosotros mismos, podemos recurrir a sitios de Internet que brindan este servicio. Dos de ellos son www.csssprites.com y www.css-sprit.es. Ambos generadores brindan una gran cantidad de opciones para configurar.

Reducir la cantidad de peticiones al servidor disminuye la velocidad de carga del sitio

la cruz para cerrar una búsqueda, las estrellas de favoritos, el signo más y el signo menos, los signos de conversación y, finalmente, **Google Checkout**. Para analizar mejor este ejemplo, podemos decir que Google utiliza, en el caso de las O amarillas del logo, un **[background-position]** que se repite de acuerdo con la cantidad de páginas que tenga el resultado. Con esta técnica, la carga se genera cuando ingresamos por primera vez al buscador.

Por otro lado, tenemos el ejemplo de Yahoo Mail. Esta empresa utiliza una única imagen para generar no solo los botones, sino también los fondos de los tabs de la casilla de correo. La optimización en la carga podría denominarse única, pues hace que el sitio resulte accesible para todos los usuarios.

Recordemos que la creación de un sitio web no significa, simplemente, lograr un buen diseño, y optimizar el código HTML, CSS, PHP o el que usemos. También implica optimizar las imágenes, porque si son bien utilizadas, harán que el sitio sea más amigable y tenga un concepto efectivo de usabilidad.

Mapa **de imágenes**

Los mapas de imágenes son **áreas** que se colocan sobre una imagen de un sitio web para generar más de un enlace en ella. Antiguamente, no eran reconocidos por los navegadores, y se debía recurrir al servidor para utilizarlos. En la actualidad, en cambio, se los elabora mediante código HTML y casi todos los browsers los admiten. Como la definición del mapa de imágenes se encuentra dentro del archivo HTML, es rápidamente accesible y permite, en muchos casos, trabajar sin conexión.

Este recurso es sencillo de usar y resulta ideal cuando queremos utilizar una porción de una imagen como enlace. A diferencia de lo que muchos diseñadores creen, los mapas sí soportan **comportamientos (scripts)**, ya que a todos ellos les podemos asignar acciones, desde el estado **off** hasta el **on**.

Estos mapas tienen algunas ventajas en relación con otras formas de diagramar o actualizar un sitio, entre ellas, las siguientes:

- Se cargan automáticamente aunque la imagen no lo haya hecho, y pueden utilizarse mientras tengan su texto de referencia.
- Se adaptan fácilmente a distribuciones complicadas o complejas, evitándonos cortar o separar las porciones de la imagen que emplearemos como enlace.

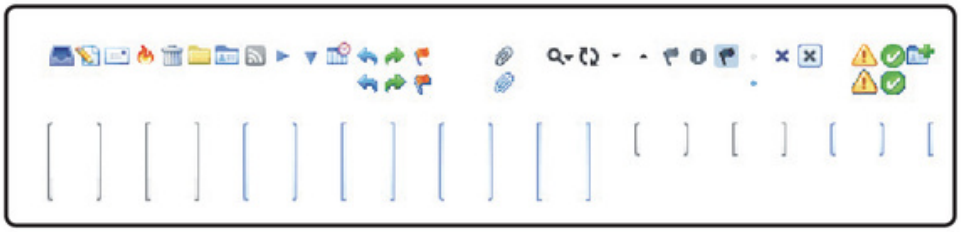


FIGURA 10. **Esta es una pequeña porción del sprite que se utiliza en el sitio de Yahoo Mail (www.mail.yahoo.com).**

• Podemos volver a usarlos una vez definidos, de modo que ahorramos mucho tiempo a la hora de ubicarlos.
• Al aplicarlos en toda una barra de navegación, por ejemplo, solo deberemos cambiar la imagen, porque si los elementos se mantienen en el mismo lugar, no será preciso reubicar o modificar el mapa, y así tampoco tendremos que editar el archivo HTML.

Los mapas de imágenes están compuestos por dos partes. La primera es la imagen que se utiliza dentro del sitio, y la segunda está dada mediante la etiqueta o tag **<map>**, que delimita el espacio empleado como enlace.

Estos mapas de imágenes están restringidos por un espacio geométrico del que obtendremos las dimensiones mediante las **coordenadas X**, para el **ancho**, e **Y**, para la **altura**, teniendo en cuenta que el vértice superior izquierdo es **0,0**. Para obtener manualmente las coordenadas donde queremos ubicar el mapa, debemos recurrir a un programa como Photoshop.

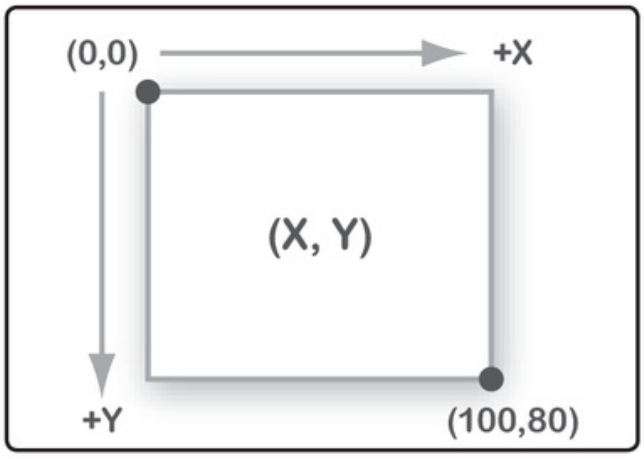


FIGURA 11. **Este recuadro mide 100 por 80 píxeles, contados desde el ángulo superior izquierdo. Entonces, la coordenada X es 100, y la Y es 80, para delimitar el ángulo inferior derecho.**

Cada etiqueta **<map>** posee la etiqueta **<area>**, que puede ser una sola o más, dependiendo de cuántos enlaces necesitemos en una misma imagen. La etiqueta **<area>** tiene algunos atributos que le son propios y que enumeramos a continuación:

• El atributo **[alt]** muestra un texto cuando situamos el mouse encima del área, a diferencia de lo que sucede con el **[alt]** de las imágenes, que nos permite nombrarlas si el navegador no puede cargarlas.
• El atributo **[shape]** indica el tipo de área. Esta puede ser **[rect]** si se trata de un área recta, como un cuadrado o un rectángulo, que siempre tendrá dos pares de coordenadas: X1, Y1, X2, Y2. Por su parte, **[circle]** es un área circular, que tendrá un par de coordenadas seguidas del radio: X1, Y1, R.

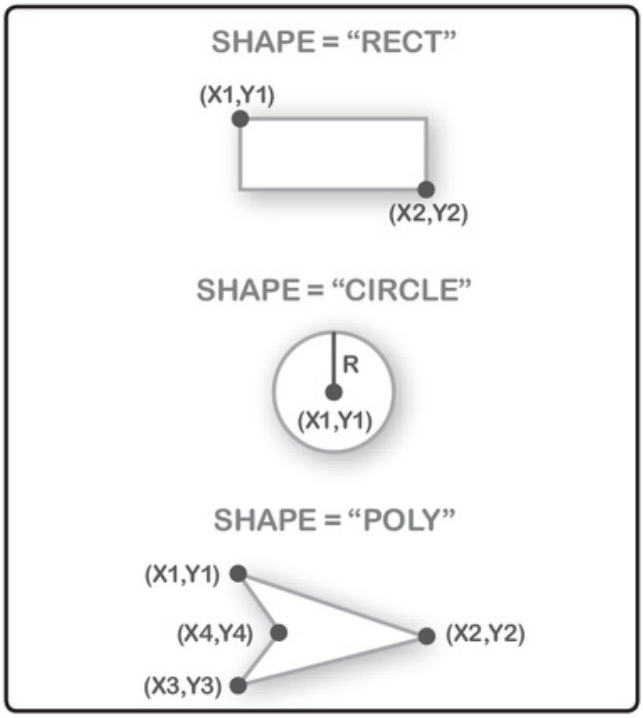


FIGURA 12. **En esta imagen podemos observar los tres tipos de área que se le pueden asignar al atributo [shape].**

Finalmente, **[poly]** es un área editable que puede tener la forma que deseemos, con la cantidad de coordenadas que necesitemos. Por ejemplo: X1, Y1, X2, Y2, X3, Y3, X4, Y4.

- El atributo **[coords]** corresponde a los valores numéricos que definen el área en donde se encontrará el mapa de imágenes; son sus coordenadas de posicionamiento.
- El atributo **[href]**, como en el resto de las etiquetas, se utiliza para enlazar el mapa de imágenes con el archivo o dirección que queramos relacionar.

Debemos decir, además, que para que la imagen actúe como mapa, es preciso colocar **[usemap="nombre_del_mapa"]** dentro de su etiqueta.

```
<img src="nuestra_imagen.jpg"
  alt="Imagen" width="356" height="341"
  border="0" usemap="#Imagen" />
<map name=" Imagen" id=" Imagen ">
<area shape="rect"
  coords="44,139,124,189"
  href="enlace_al_sitio_que_deseemos"
  alt="Rectángulo" />
<area shape="circle" coords="71,70,37"
  href=" enlace_al_sitio_que_deseemos "
  alt="Círculo" />
<area shape="poly"
coords="287,236,222,309,209,296,214,291"
  href=" enlace_al_sitio_que_deseemos"
  alt="Polilínea o Línea adaptable" />
</map>
```

Los mapas de imágenes son realmente útiles si sabemos aplicarlos y utilizarlos. En muchos casos, nos ahorrarán bastante tiempo que perderíamos cortando imágenes para enlazarlas. Las formas rectas y circulares son las más sencillas de crear usando Dreamweaver, y aunque la herramienta con la que podemos dibujar una polilínea lleva un poco más de tiempo de manejo, no difiere del modo de uso de, por ejemplo, la **[Pluma]** de Adobe Illustrator.

FIGURA 13. En http://theall-knowing4.blogspot.com encontramos un ejemplo de mapa de imágenes combinado con CSS para lograr el mouseover.

Multiple **choice**

1 **¿Para qué se puede usar el atributo longdesc?**

a- Para eliminar una imagen.
b- Para controlar la posición de una imagen.
c- Para agregar una imagen.
d- Para encontrar información sobre una imagen.

2 **¿Cuáles son los formatos de imagen más comunes?**

a- AVI, TIFF y JPG.
b- TIFF, JPG y PNG.
c- GIF, JPG y PNG.
d- WMV, WMA y JPG.

3 **¿Cuál es la característica más destacada de PNG?**

a- Su mejor calidad.
b- La posibilidad de utilizar transparencias.
c- La posibilidad de usar más colores.
d- Su mayor compatibilidad.

4 **¿Cuántos colores soporta el formato JPG?**

a- Hasta 16 millones de colores.
b- Hasta 256 colores.
c- Hasta 20 millones de colores.
d- Hasta 2 millones de colores.

5 **¿Qué técnica permite tener varias construcciones de imágenes dentro de una única imagen?**

a- Animación en 2D.
b- Optimización de imágenes.
c- Sprites de imágenes.
d- Diagramación.

6 **¿Qué atributo muestra un texto cuando ponemos el mouse encima de un área?**

a- Alt.
b- Crtl.
c- Alt-u.
d- Alt-x.

Respuestas: 1-d, 2-c, 3-b, 4-a, 5-c, 6-a.

Capítulo 6

Enlaces

En este capítulo aprenderemos la forma adecuada de trabajar con enlaces en nuestros diseños.

RedUSERS.com

Enlaces
o hipertextos

El término **hipertexto** se refiere a un tipo de texto electrónico que representa, además de una nueva tecnología informática, una novedosa manera de edición.

Gran parte del éxito del lenguaje HTML se debe a la posibilidad de **conectar** la información verbal y no verbal (visual, sonora, animación, etcétera) a través del uso de **enlaces** o **hipertextos** que, como nodos conformadores de una gran red, ponen a disposición de los usuarios recorridos alternativos no secuenciales.

Lo más novedoso de este concepto es, precisamente, la libertad que se le da al usuario de moverse por la Web, siguiendo sus propios intereses y eligiendo qué información consumir o qué acción ejecutar como parte de un trayecto alternativo entre muchos otros.

ORIGEN DEL CONCEPTO DE HIPERTEXTO

El origen de la idea de **hipertexto** se remonta al año 1945, cuando **Vannevar Bush**, Jefe del Departamento de Investigación y Desarrollo Científico de EE.UU, publicó el artículo "As we may

FIGURA 1. **El artículo original "As we may think" se encuentra disponible en los archivos de The Atlantic (www.theatlantic.com).**

ORIGEN DE HTML

En la definición del lenguaje HTML, se tomaron algunas características que ya existían para la publicación digital de contenidos, entre ellas, el hipertexto. Es por eso que las siglas HTML provienen de las palabras HyperText Markup Language.

La mente humana no procesa naturalmente la información de manera secuencial, sino a través de asociaciones

think", en el que criticaba los métodos utilizados hasta ese entonces para gestionar la información.

Allí detalla que la estructura secuencial de los documentos –influenciada por la estructura verbal del discurso– es la causante de que los métodos de la época fueran incapaces de procesar grandes cantidades de información de manera correcta. La razón es simple: la mente humana no procesa naturalmente la información de modo secuencial, sino que lo hace a través de **asociaciones**.

Si bien este autor nunca llegó a utilizar el término hipertexto, fue el primero en tener noción de lo que era la **multisecuencialidad** (no linealidad) de este elemento.

ENLACES BÁSICOS

Técnicamente, los enlaces son bloques compuestos por fragmentos de textos u otros elementos que vinculan dos recursos. Estos recursos suelen ser páginas web, aunque también se emplean enlaces para **relacionar** imágenes, para **descargar** documentos (PDF, Word, etc.) o para **ejecutar** acciones, como enviar un e-mail.

La sintaxis de un enlace está compuesta por la etiqueta **<A>…</A>**, dentro de la cual siempre debe declararse el atributo **href** o **name** para indicar el destino del enlace.

Atributo HREF

Es el más importante para esta etiqueta y se usa para **enlazar** diferentes URL. Veamos cómo sería la sintaxis correcta de un enlace simple:

```
<A HREF="URL">Texto del enlace</A>
```

En este caso, **Texto del enlace** es el contenido que se visualizará en el navegador, y **"URL"** es el destino al que apuntará el link.

Atributo name

Este atributo se implementa para nombrar secciones dentro de una misma página web, y suele ser útil para contenidos extensos, en los que el usuario podrá ir de una parte a otra sin hacer scroll.

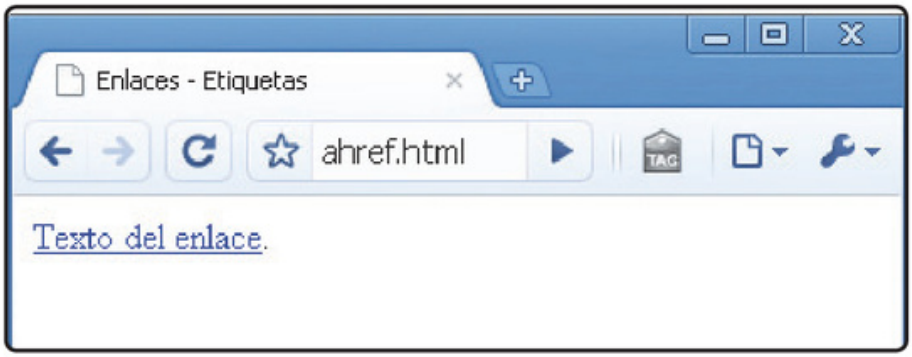


FIGURA 2. **Visualización de un enlace sin estilos (por defecto).**

Para aprender a utilizarlo, en primera instancia, crearemos un enlace de destino con el atributo **name**, asignándole un valor al azar, que nos servirá como identificador para el siguiente paso.

Luego, desde el enlace de origen, haremos referencia a este enlace anteponiendo el símbolo **[#]** al valor que le hayamos dado a **name**. Por ejemplo:

- Enlace de destino: `<A NAME="parte1">Primera parte</A>`
- Enlace de origen: `<A HREF="#parte1">Ir a la primera parte</A>`

De esta manera, al hacer clic sobre el enlace de origen, el navegador se posicionará sobre el de destino.

URL (localizador uniforme de recursos)

Para comenzar a crear enlaces un poco más complejos, es necesario familiarizarnos con el concepto de **URL** (del inglés, Uniform Resource Locator), que, además de ser esencial para crear enlaces, también suele aplicarse a otros elementos.

Protocolo es el mecanismo que usan los navegadores para acceder a los recursos

Las URL determinan que cada página HTML publicada en Internet posea un nombre único que las distinga del resto. Esto es lo que permite que los enlaces apunten inequívocamente a una determinada página. Por ejemplo, si queremos acceder a la página de Wikipedia, debemos escribir en el navegador la siguiente cadena de texto: **www.wikipedia.org**, que es la URL completa de la página principal de su sitio.

Las URL poseen una estructura destinada a que los navegadores encuentren los recursos disponibles de manera eficiente. La mayoría de las que son simples están conformadas por tres partes: protocolo, dominio y directorio; aunque existen otras más complejas:

FIGURA 3. **Las URL están conformadas por tres partes que le indican al navegador cómo llegar a la página solicitada.**

- **Protocolo**: es el mecanismo que usan los navegadores para acceder a los recursos. Todos los sitios web utilizan **http://**, y a aquellos que requieren seguridad se les agrega una letra "s" al final (**https://**).
- **Dominio**: especifica el nombre de la computadora (por lo general, es un nombre de dominio o una

dirección IP) donde se encuentra alojado el contenido que compone un sitio web.
• **Directorio**: luego del dominio, se encuentra el o los directorios separados por barras ("/"), que definen el recorrido o ruta por seguir para llegar hasta el recurso.

Enlaces relativos y absolutos

Todo sitio web posee, comúnmente, numerosos enlaces. Estos pueden apuntar a otras páginas o recursos dentro mismo sitio o alojados en sitios externos. Esto categoriza a los enlaces en dos grandes grupos:

• **Internos**: son aquellos que apuntan a recursos o páginas ubicados dentro del mismo sitio.
• **Externos**: son los que enlazan un recurso o una página ubicada por fuera del sitio de origen. Una característica clave de este tipo de enlaces es que, al hacer clic sobre ellos, el navegador abandona el sitio en el que se encuentra para localizar el nuevo sitio.

Por otro lado, existe una clasificación que, además de tener en cuenta si los enlaces son externos o internos, tiene que ver con la manera en la que se escribe la sintaxis de las URL:

• **Absolutas**: son las que incluyen la URL completa, con protocolo, dominio y directorios. Por lo general, se utilizan para los enlaces externos, ya que necesitan toda la información de la localización del recurso.
• **Relativas**: por lo general, este tipo de URL se utiliza al crear enlaces internos. Solo indica los **directorios** y, a partir de esa información, los navegadores pueden adivinar los datos faltantes (**protocolo y dominio**) para localizar el recurso.

Por ejemplo, si desde la página ubicada en **http://www.wikipedia.org/ruta1/ruta2/pagina1.html** quisiéramos acceder a otra presente en **http://www.ejemplo.com/ruta1/ruta2/pagina2.html**, no necesitaríamos escribir la URL absoluta, ya que ambas páginas poseen el mismo dominio y utilizan el mismo protocolo. Por lo tanto, podemos prescindir de esa parte de la URL y escribirla de manera relativa. Entonces, las dos formas de escribir la misma URL serían:

URL absoluta: **http://www.ejemplo.com/ruta1/ruta2/pagina2.html**
URL relativa: **/ruta2/pagina2.html**

Y el enlace aplicado:

```
<A HREF="/ruta2/pagina2.html">Texto del
  enlace correspondiente a la página
  1</A>
```

Una de las ventajas de utilizar enlaces relativos es que podemos cambiar la dirección de un sitio web sin tener que rescribir las URL.

Al usar enlaces relativos, si cambia la dirección del sitio, no tendremos que rescribir todos los hipervínculos

Existen varios tipos de URL relativas. Veamos cada uno de ellos analizando ejemplos de enlaces que nos ayudarán a comprender cómo deben crearse:

• Cuando el **origen** y el **destino** del enlace están en el **mismo directorio**, el protocolo, el dominio y los directorios son exactos. La única diferencia es el nombre del recurso enlazado, por lo que la URL relativa será solamente este nombre.

Origen: **http://www.misitio.com/ruta1/ruta2/pagina1.html**
Destino: **http://www.misitio.com/ruta1/ruta2/pagina2.html**
URL relativa: **pagina2.html**

• Cuando el destino del enlace está en un nivel superior que el de su origen, aunque ambas páginas están muy cerca, es necesario indicar esa superioridad de nivel al escribir la URL. La manera correcta de hacerlo es escribir el nombre del recurso anteponiéndole dos puntos y una barra (**../**), que le indicarán al navegador que debe subir un nivel para localizar el recurso.

Origen: **http://www.misitio.com/ruta1/ruta2/pagina1.html**
Destino: **http://www.misitio.com/ruta1/pagina2.html**
URL relativa:**../pagina2.html**

Si el destino se encuentra dos o más niveles por encima del origen, debemos incluir **../** tantas veces como niveles existan. Por ejemplo, si los niveles fueran dos, la URL relativa se escribiría así:

Origen: **http://www.misitio.com/ruta1/ruta2/ruta3/pagina1.html**
Destino: **http://www.misitio.com/ruta1/pagina2.html**
URL relativa:**../../pagina2.html**

• Si el destino del enlace está en un **nivel inferior** al de su origen, debemos escribir la URL indicando el nombre del recurso anteponiéndole el nombre del directorio que lo contiene:

ASPECTOS DE LOS ENLACES

Podemos reconocer fácilmente los enlaces porque, al pasar el cursor por encima de ellos, este se transforma en una manito con un dedo señalador. Si los enlaces no tienen estilos que modifican su aspecto visual, por defecto se muestran como textos subrayados de azul.

Es conveniente describir todos los directorios que existen a partir del servidor, anteponiendo una barra (/)

Origen:**http://www.misitio.com/ruta1/pagina1.html**
Destino: **http://www.misitio.com/ruta1/ruta2/pagina2.html**
URL relativa: **ruta2/pagina2.html**

En este caso, no es necesario utilizar barras o puntos, pero, al igual que en el caso anterior, debemos escribir la cantidad de directorios de diferencia que haya entre ambos:

Origen: **http://www.misitio.com/ruta1/pagina1.html**
Destino: **http://www.misitio.com/ruta1/ruta2/ruta3/ruta4/pagina2.html**
URL relativa: **ruta2/ruta3/ruta4/ruta2/pagina2.html**

• Si el origen y el destino del enlace comparten el mismo dominio pero se encuentran muy alejados, podríamos utilizar los dos métodos anteriores para simplificar la tarea. Sin embargo, lo conveniente es describir todos los directorios que existen a partir del servidor, anteponiendo una barra (**/**) .

De esta forma, los navegadores vuelven hasta la raíz del dominio y buscan el recurso a partir de allí, siguiendo el orden de los directorios que hemos especificado.

Origen: **http://www.misitio.com/ruta1/ruta2/ruta3/pagina1.html**
Destino: **http://www.misitio.com/ruta9/pagina2.html**
URL relativa: **/ruta9/pagina2.html**

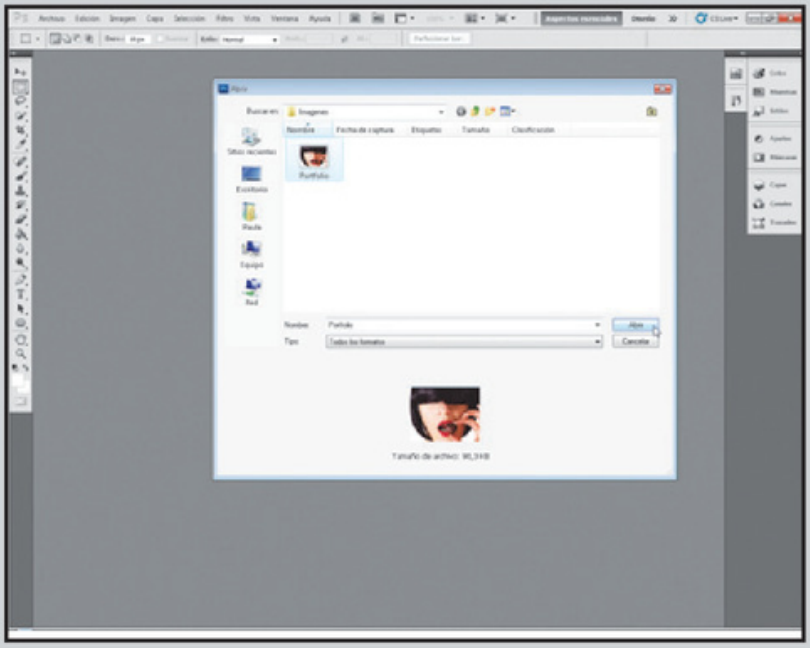
Enlaces **en Dreamweaver**

En este apartado veremos los distintos tipos de enlaces que se pueden generar a través de Dreamweaver. Como en otros casos, el programa nos simplificará bastante su creación. Veamos cómo hacerlo.

Adobe Dreamweaver brinda la posibilidad de generar enlaces desde el manejador gráfico ubicado en la barra **[Propiedades]** o desde el código. Seguramente, al principio nos sentiremos más cómodos con la primera opción, pero la segunda nos resultará ideal para escribir más rápido y sin recurrir al mouse.

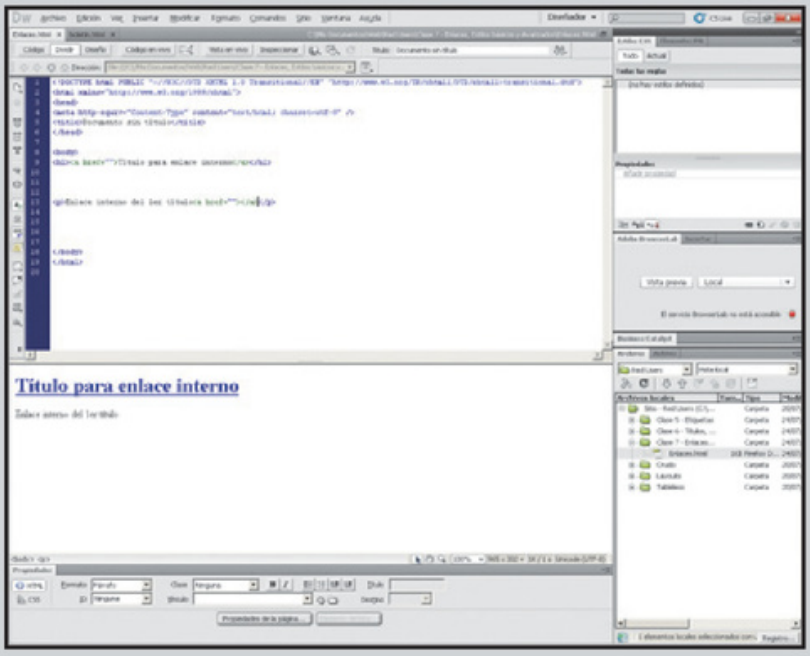
PASO A PASO /1
Crear enlaces

1



Lo primero que debe hacer para preparar el documento en el que va a armar el enlace será crear un texto con la etiqueta **<h1>** y otro con **<p>**.

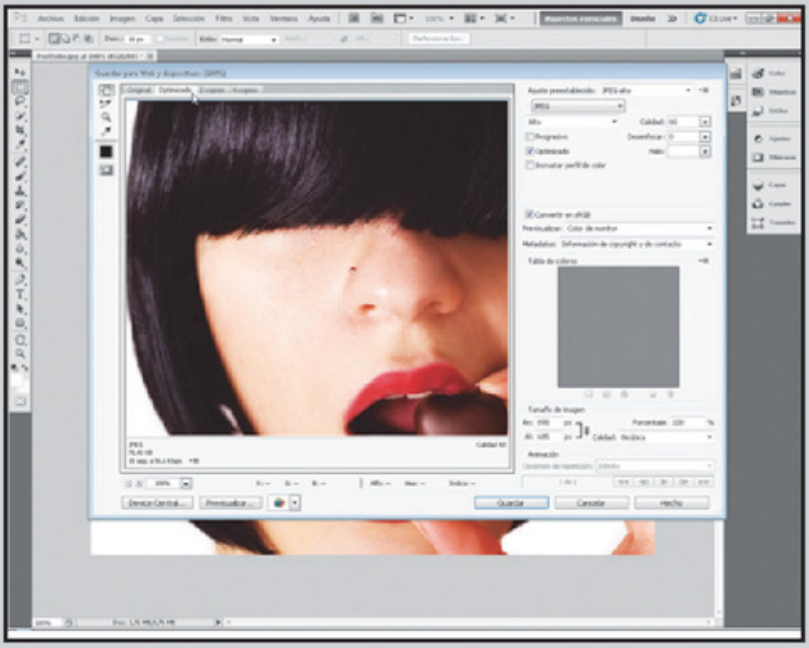
2



Para generar el enlace, utilice la etiqueta **<a>**, seguida del parámetro **href**. Debe tener en cuenta que, en un enlace interno, tanto **<h1>** como **<p>** tienen que incluir una etiqueta **<a>**.

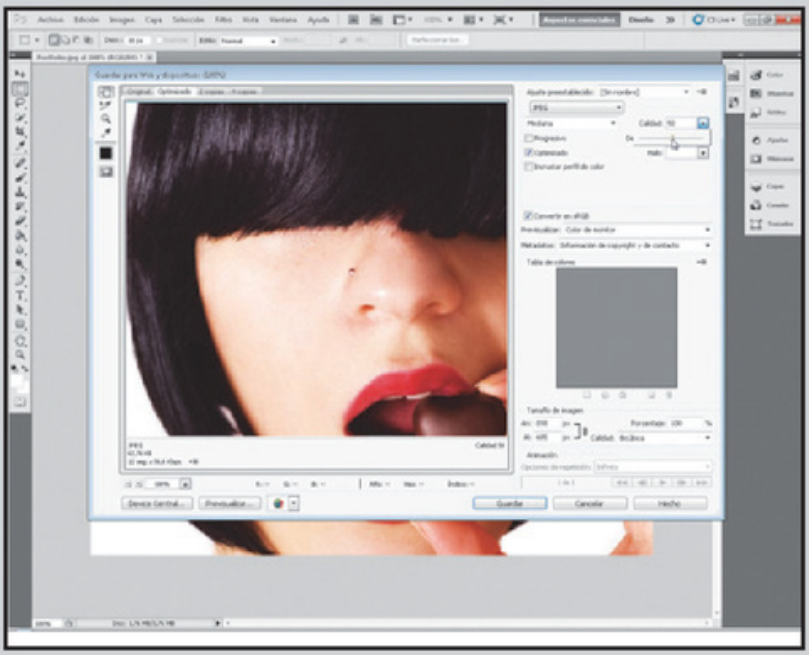
PASO A PASO /1
(Cont.)

3



A continuación, dentro del parámetro **href** de la etiqueta **<a>** correspondiente a **<h1>**, coloque el signo **#** seguido del nombre **texto**. Luego, en la etiqueta **<a>** correspondiente a **<p>**, ponga **texto** dentro del parámetro **name**. De esta forma, conseguirá que la etiqueta **<h1>** enlace con **<p>** al hacer clic (tengas en cuenta que en **name** la palabra **texto** no debe llevar **#**).

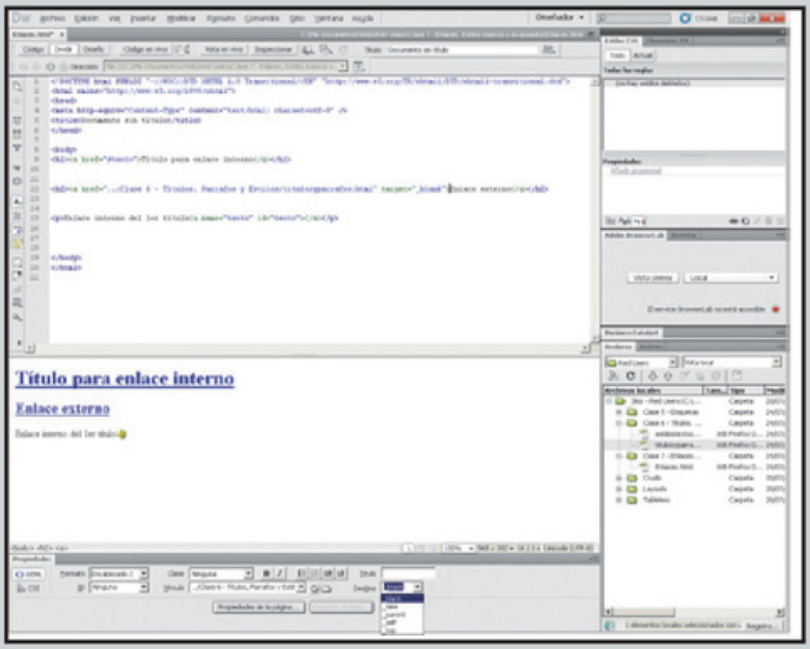
4



Para generar un enlace externo al documento, use una etiqueta **<h2>** y vincule su texto a otro archivo externo de carácter HTML. Entonces, seleccione el texto y utilice el manejador gráfico que se encuentra en la barra **[Propiedades]**, moviendo el mouse hasta el archivo de destino del enlace.

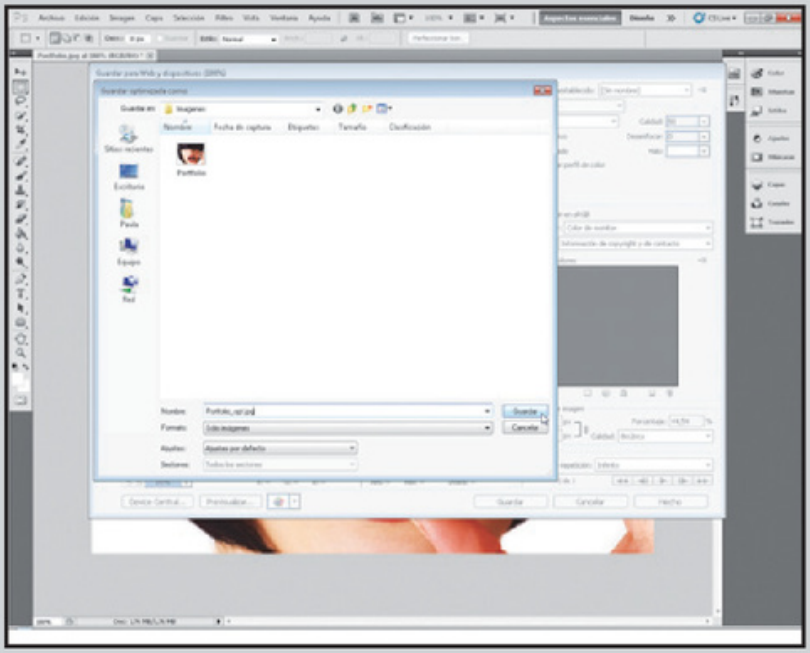
PASO A PASO /1 (Cont.)

5



Puede indicarle al enlace externo el lugar que tendrá al abrirse. Si bien por defecto la etiqueta **<a>** reemplazará el archivo HTML que está viendo por el nuevo, mediante las opciones del desplegable **[Destino]** puede indicar que este se muestre. Como puede observar, la propiedad **target** indicará en qué ventana debe abrirse el enlace.

6



En este caso, deje la etiqueta **<a>** sin ningún destino, porque si bien el enlace es externo, se encuentra dentro de su propio sitio, y lo más conveniente es que una página (archivo HTML) reemplace a la otra. No sucede lo mismo cuando el enlace generado debe enlazar otro tipo de archivo, como podrían ser **PDF**, **RAR** o **ZIP**, entre otros.

Cuando creamos los archivos de nuestros sitios, siempre debemos recordar que sus nombres no pueden tener espacios ni caracteres especiales, porque si no, pueden ser tomados como un error dentro de los estándares. Además, a la hora de crear vínculos, es más seguro trabajar evitando el uso de esos caracteres y de los espacios.

Unidades de medida

Las medidas se determinan con valores numéricos que pueden ser **enteros**, **decimales**, **positivos** o **negativos**. Luego de ellos, debemos poner la unidad de medida. Si indicamos el valor numérico en 0 (cero), podemos no especificar la unidad de medida; pero si el valor numérico es distinto de cero y no indicamos ninguna unidad, la mayoría de los navegadores de Internet ignorará la regla. Este es un error habitual de los diseñadores que recién comienzan a manejar CSS, y es importante que le prestemos atención para evitar cometerlo.

A continuación, analizaremos tres casos de estilos aplicados a un párrafo. En el primero, la regla CSS está mal aplicada, porque no contiene una unidad de medida:

```
p{ width: 500; }
```

A continuación, vemos dos reglas correctas:

```
p{ padding: 0; }

p{ margin-left: 10px; }
```

Las unidades se dividen en dos grupos: relativas y absolutas. Conozcamos las características de cada tipo.

UNIDADES RELATIVAS

Estas unidades se definen en función de la medida de algún otro elemento HTML. Las relativas son más flexibles que las absolutas, ya que pueden adaptarse a diferentes medios. Además, al incrementar o reducir la medida de referencia, todas las medidas relativas se incrementarán o se reducirán proporcionalmente, y por lo general, conservarán las relaciones de tamaño de nuestro diseño.

Las unidades relativas son **em**, **ex** y **px**. Las dos primeras son proporcionales a los tamaños de tipografía utilizados, y la última es proporcional a la resolución de pantalla del usuario.

La unidad **em** tiene su origen fuera del ámbito del CSS y se define, inicialmente, como el ancho de la letra eme mayúscula (M) de la tipografía que se esté utilizando. En CSS, **em** hace referencia a la unidad de medida de la tipografía que esté utilizando el **elemento HTML padre**. Por ejemplo, si

Las medidas se determinan con valores numéricos, que pueden ser enteros, decimales, positivos o negativos

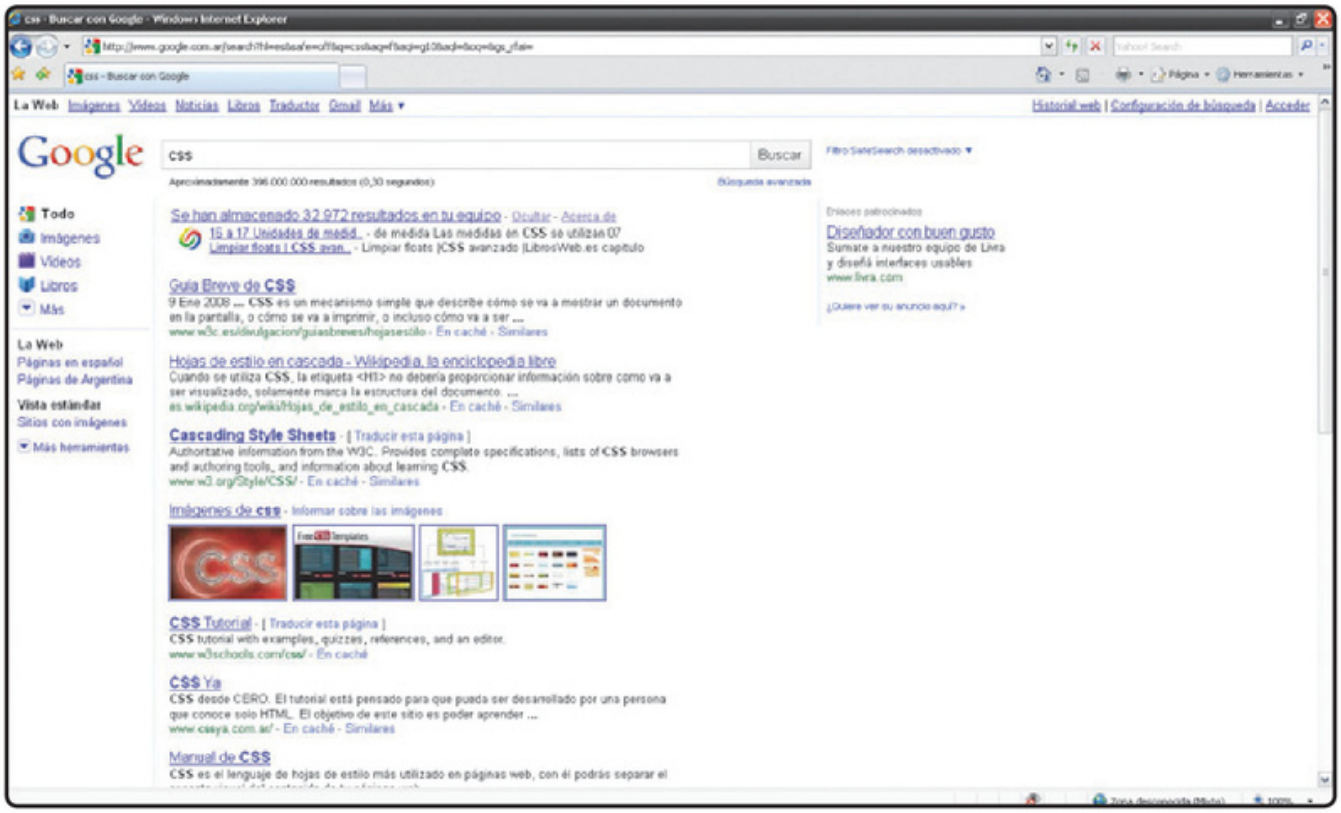


FIGURA 4. **Aquí vemos los resultados de búsqueda de Google con el tamaño de tipografía por defecto de nuestro navegador (mediano).**

El uso de unidades relativas permite mantener las proporciones de un diseño

definimos el siguiente código, la tipografía de nuestro párrafo será un 20% mayor a la de su elemento padre; en este caso, el elemento **<body>**:

```
body{ font-size:10px; }
p{ font-size:1.2em; }
```

Para calcular el tamaño que tendrá la tipografía del párrafo, debemos hacer una sencilla cuenta: 10 (tamaño de letra del **<body>**) * 1.2 (tamaño relativo de letra del párrafo). Esta operación da 12, que es el tamaño en **px** que tendrá la tipografía del párrafo.

Aunque al principio todo esto puede parecer muy complicado, con la práctica nos daremos cuenta de que es sencillo, e incorporaremos estas cuentas de proporción en nuestro trabajo habitual.

Como ya mencionamos, la principal ventaja que tiene el uso de unidades relativas es que nos permiten mantener las proporciones de un diseño independientemente del tamaño por defecto que esté utilizando el usuario que visita nuestra página.

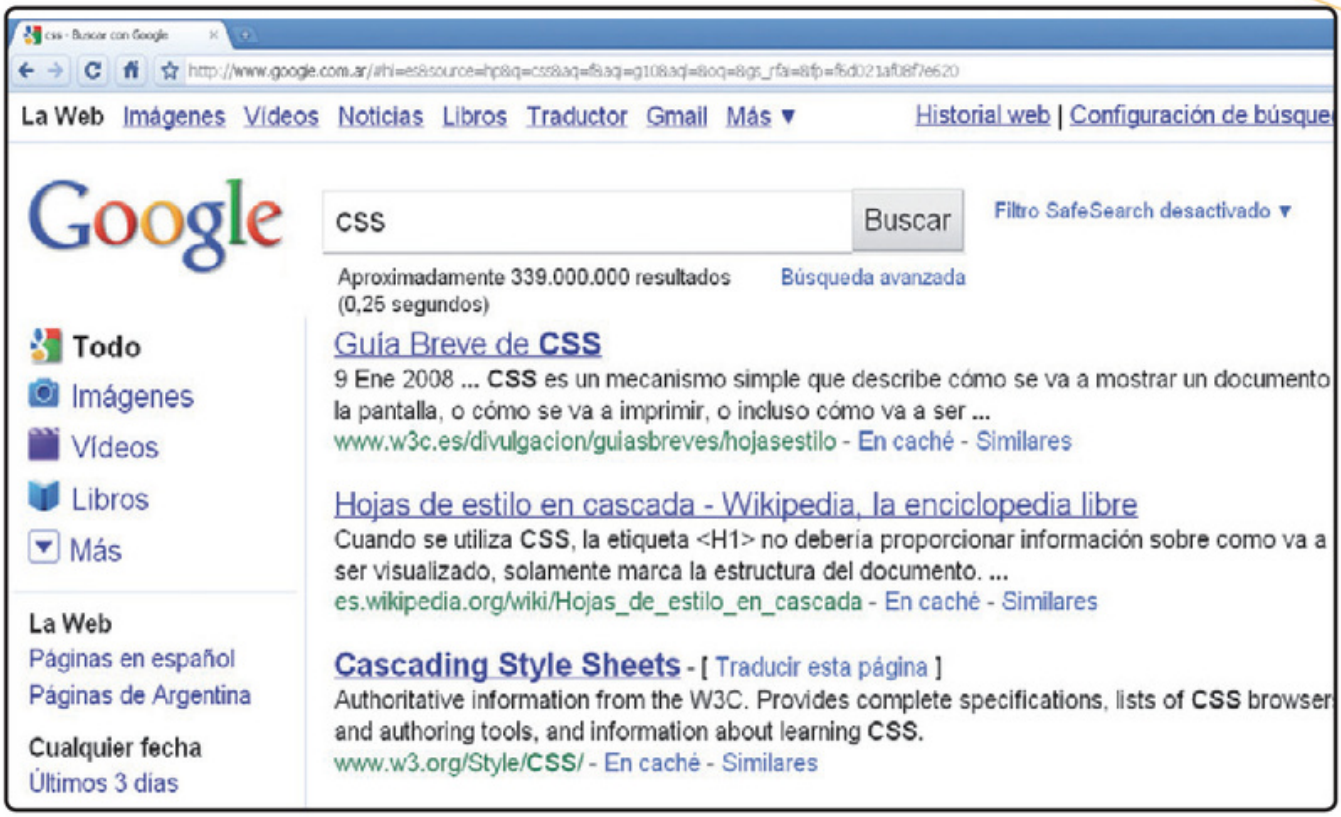


FIGURA 5. **Al establecer un tamaño de texto muy grande en nuestro navegador, vemos que Google aplica unidades relativas, dado que, cuando agrandamos el tamaño del texto, todo aumenta de manera proporcional.**

La lógica de la unidad **ex** es exactamente igual a la de la unidad **em**, solo que se toma como referencia la altura de la letra equis minúscula (x).

En CSS, lo más común es utilizar las unidades **px** y **em**, donde **px** (píxel) es relativa respecto a la pantalla del usuario.

PORCENTAJES

Otra unidad relativa en CSS son los porcentajes. Esta unidad de medida está formada por un valor positivo seguido del símbolo %. Los porcentajes se utilizan, generalmente, para definir el ancho de los elementos. Podemos ver un claro ejemploen la porción de código que se presenta a continuación:

```
div#principal{ width:600px; }
div#contenido{ width:50%; }
```

Donde nuestro código HTML es el siguiente:

```
<div id="principal">
        <div id="contenido">
                ...
        </div>
</div>
```

En este caso, el **[div]** cuyo atributo **[ID]** es **[contenido]** tendrá 300 px de ancho, porque mide el 50% (la mitad) de su elemento padre, que es el **[div]** cuyo atributo **[ID]** es **[principal]**.

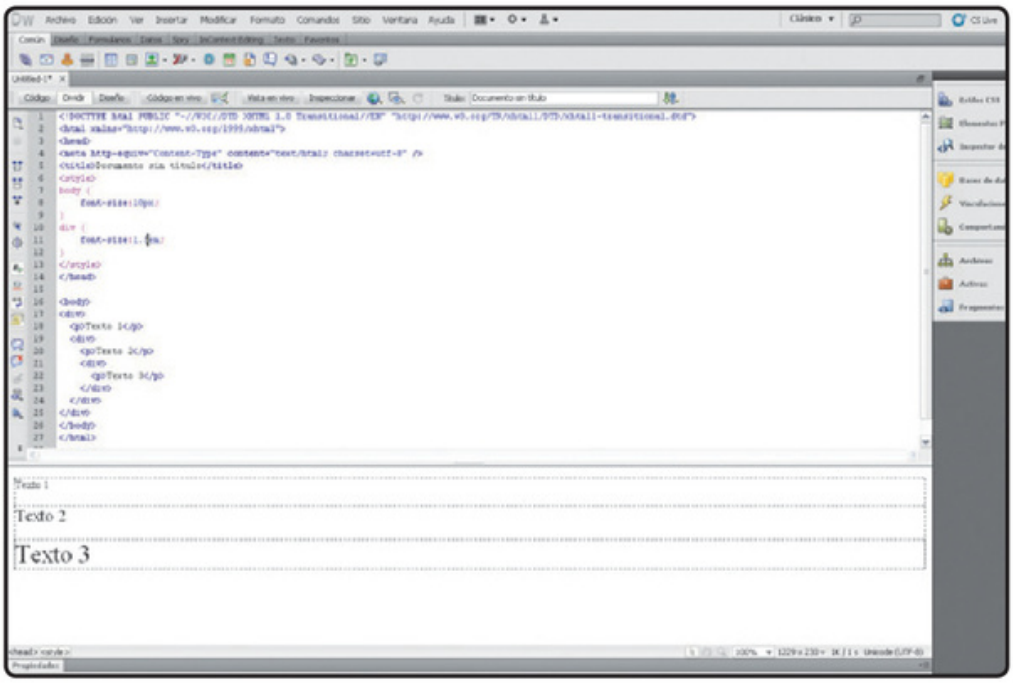


FIGURA 6. En Dreamweaver se presenta el problema de que el texto se ve cada vez más grande. Para evitarlo, debemos ser más específicos con las relaciones de los elementos HTML.

Un problema habitual que tendremos al utilizar unidades relativas es que el texto se verá cada vez más chico o más grande. Por ejemplo, si nuestro código CSS es:

```
body { font-size:10px; }
div{ font-size:1.5em; }
```

Y el HTML es el siguiente:

```
<body>
<div>
<p>Texto 1</p>
<div>
<p>Texto 2</p>
<div>
<p>Texto 3</p>
</div>
</div>
</div>
</body>
```

En este caso, **[Texto 1]** tendrá un tamaño de 15 px (10 * 1.5 = 15); **[Texto 2]** tendrá un tamaño de 22.5 px (10 * 1.1 * 1.1 = 22.5) y **[Texto 3]**, de 33.75 px (10 * 1.5 * 1.5 * 1.5 = 33.75).

Para solucionar este tema, debemos ser más específicos con las relaciones de los elementos HTML, usando los atributos **[id]** y **[class]** vistos anteriormente.

UNIDADES ABSOLUTAS

Las unidades absolutas definen las medidas de forma específica, debido a que sus valores reales no dependen ni hacen referencia a los de otros elementos HTML. Las unidades absolutas son:

- **in**, del inglés inches, pulgadas (una pulgada equivale a 2,54 centímetros)
- **cm**, centímetros
- **mm**, milímetros

La propiedad que se encarga de controlar la altura es height, que determina el alto de la caja

- **pt**, puntos (un punto equivale a 1/72 pulgadas, es decir, alrededor de 0,35 milímetros)
- **pc**, picas (una pica equivale a 12 puntos, es decir, alrededor de 4,23 milímetros)

La mayoría de estas unidades absolutas están en desuso, a excepción de los puntos (**pt**), que se utilizan para indicar el tamaño de tipografía de los documentos que vamos a imprimir.

RECOMENDACIONES

Por lo general, es aconsejable utilizar unidades relativas siempre que sea posible, ya que esto hará que nuestra página sea más flexible para ser visualizada en diferentes dispositivos.

Lo habitual es emplear la unidad **px** (píxel) y porcentajes (**%**) para definir los elementos principales de las páginas, y **em** y porcentajes para el tamaño de la tipografía.

Propiedades **de tamaño**

La altura y la anchura de un elemento HTML están controladas por los atributos llamados **width** y **height** en CSS, que se traducen como ancho y alto, respectivamente. Conozcamos cómo se utiliza cada una de ellos.

ANCHO (WIDTH)

La propiedad que controla la anchura es **width** y se emplea para determinar el ancho de la caja. Veamos su uso en el código CSS:

```
#caja {
width: 60px;
}
```

El código HTML sería:

```
<div id="caja">Esta sería la caja
  contenedora de un div con 60px
  de ancho
</div>
```

Mediante líneas punteadas, en el primer ejemplo de uso de **width** vemos que, al no tener un ancho especificado, el párrafo emplea todo del elemento contenedor. En el segundo caso se le asigna el 50% de su elemento padre y, luego, se expresa en ems y en píxeles.

ALTO (HEIGHT)

La propiedad que controla la altura es **height** y se emplea para determinar el alto de la caja. A

continuación, veremos cómo se establecen sus valores para un elemento HTML. En este caso, será para el ** ** con un **[ID]** llamado **[caja]**. Veamos el código CSS:

```
#caja {
height: 60px;
}
```

El código HTML es:

```
<div id="caja"> Esta sería la caja
 contenedora de un div con 60px de alto

</div>
```

En la imagen donde aparece el alto modificado, en el primer caso no se le asigna valor de alto, y actúa de la misma manera que cuando no se le asigna un ancho. Luego, a continuación, le asignamos un porcentaje al alto que no es bien interpretado por el navegador (estos tienen dificultades para interpretar este tipo de medida en el alto). Para finalizar, aplicamos un valor relativo en ems y píxeles.

Para definir el tamaño, es posible emplear, en ambos casos, cualquiera de las unidades estándar que explicaremos a continuación, aunque es común que siempre las encontremos especificadas en píxeles:

- **Inherit**: toma los valores del elemento padre. Si este no tiene altura explícita, se le asigna valor **auto**.
- **Porcentaje**: se calcula tomando de referencia las medidas que tiene asignado el elemento padre.
- **Píxeles**: se le asigna el valor en píxeles.

Los atributos width y height no pueden tener valores negativos, ya que expresan el tamaño del elemento

- **Auto**: este es el valor que se le asigna al elemento cuando no tiene una altura determinada por CSS. En este caso, el navegador calcula la dimensión sobre la base del contenido y el espacio disponible.

Ninguno de estos atributos puede tener valores negativos, ya que expresan el tamaño que tendrá el elemento. Ambos pueden aplicarse a cualquier elemento, excepto a aquellos en línea (**inline**) que no sean imágenes. Recordemos que los elementos inline no generan una nueva línea en el flujo del texto y, por eso, sería inútil asignarles valores de ancho y alto.

Por otro lado, cuando definimos las medidas de una caja o elemento, debemos tener en cuenta que dichos valores se verán afectados por los de bordes, padding y margen que también se hayan aplicado.

Cuando trabajamos con cajas y queremos lograr dos elementos con las mismas dimensiones, pero por ejemplo, aplicando un borde de 3 píxeles, debemos hacer lo siguiente. En primer lugar, para conseguir el mismo tamaño debemos restar, tanto al alto como al ancho, 6 píxeles, que corresponden a 3 píxeles por cada lado (horizontal o vertical). Es decir, si al lado derecho le quitamos 3 y al

izquierdo otros 3, tenemos 6 píxeles menos de ancho. Lo mismo sucede con el alto, para compensar el grosor de los bordes. Con la misma metodología restarmos 10 px de padding en la segunda caja, dado que ambos factores afectan de forma directa las dimensiones de una capa.

LAYOUT LÍQUIDOS

Como hemos mencionado anteriormente, una de las medidas que podemos utilizar para el ancho y el alto de los elementos es el porcentaje.

Aunque prevalece el uso de las medidas definidas en píxeles, por ser más específicas y exactas, existen los llamados **layout líquidos,** en los que se emplean medidas porcentuales para la estructuración del sitio. En estos layouts, para definir los anchos de los elementos se utilizan unidades relativas, como porcentajes o ems, por lo que los elementos se adaptan a la pantalla que emplee el usuario y a su resolución. De esta forma, se aprovecha al máximo el espacio disponible.

A pesar de esta ventaja, son más utilizados los layouts fijos, aunque su uso puede generar algunas incomodidades. Por ejemplo, en primera instancia hay que determinar para qué resolución se va a maquetar el sitio, y si un usuario tiene una resolución diferente, verá el sitio distinto.

En este caso, debemos tener en cuenta que corremos el riesgo de que aparezca el scroll horizontal en la página, que como sabemos, se trata de un elemento

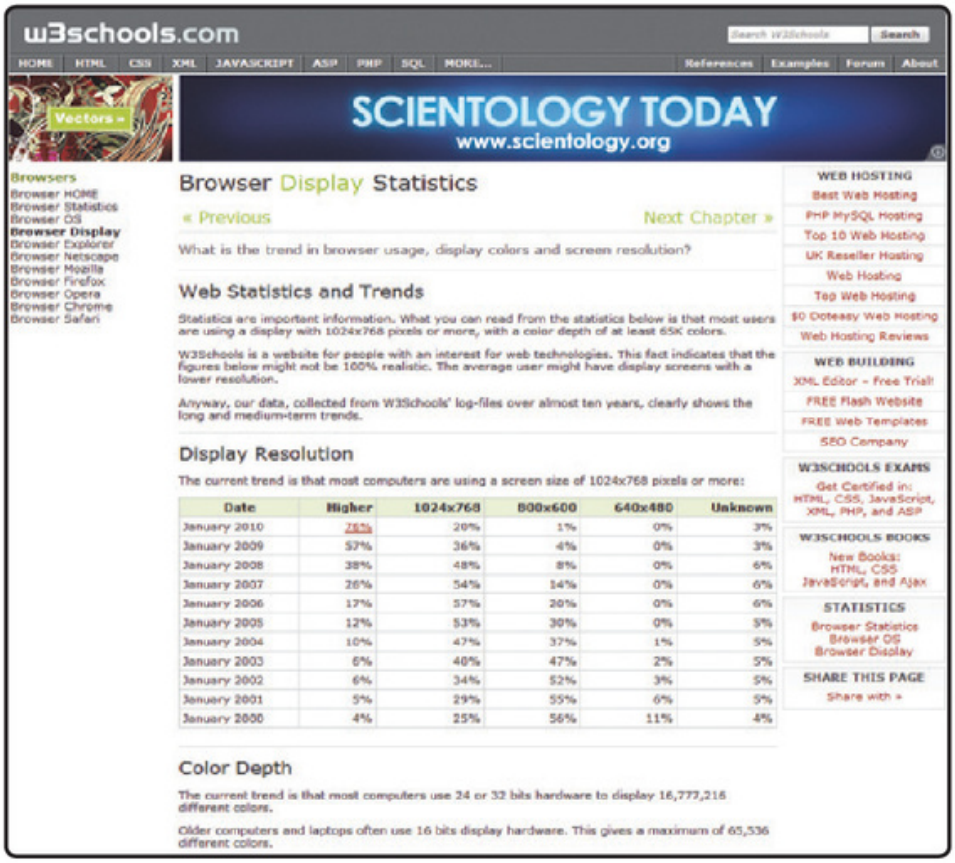


FIGURA 7. En www.w3schools.com encontramos un cuadro con las resoluciones más utilizadas a lo largo de los últimos años.

no es muy agradable. En la actualidad, se está trabajando para 1024 x 768.

Enlaces básicos y avanzados

Como ya vimos, todos los sitios poseen diversos enlaces. Cada uno de los estados de un enlace puede ser maquetado con distintas propiedades, dependiendo de la función que va a cumplir.

Hay varias maneras de maquetar un **enlace** y de vincularlo, dependiendo de su función. Entender el funcionamiento y la estructura de un **enlace** puede facilitar mucho el código semántico y hacerlo, además, un código más breve, limpio y de fácil acceso. Veamos, a continuación, las tres formas más comunes de crear enlaces:

• La primera es el **enlace tipográfico**, aquel que se utiliza desde los comienzos de Internet, y que consta de un texto con propiedades de color y de subrayado especiales. En este caso, nuestro código CSS estaría definido de la siguiente manera:

```
a:link { color: #ededed;text-decoration:
 underline; }
```

• La segunda forma de definir un enlace es mediante una **imagen**. En la actualidad, se la utiliza comúnmente en galerías de imagen, aunque hay quienes siguen empleando esta forma fuera de los sitios de ese tipo. En esta clase de enlaces, el código en el HTML se vería de la siguiente manera:

```
<a href="ruta_de_mi_nueva_página"><img
  src="nombre_de_la_imagen" width="249"
  height="32" /></a>
```

• Finalmente, la forma más habitual y correcta consiste en definir un **enlace por bloque**. De esta manera, obtendremos un enlace con diseño no solo mediante imagen sino también con tipografía. En general, este modo de diseñar se utiliza para las **navbar** (barras de navegación) o **botoneras**. El código semántico que incorporamos en nuestro código CSS es el siguiente:

```
ul { Definir las propiedades }
ul li { Definir las propiedades
  de la lista }
ul li a { Definir las propiedades
  del enlace }
ul li a .nombre_de_la_clase
 { Definir las propiedades de la clase }
ul li a span { Definir las propiedades
   de la etiqueta span }
ul li a:hover { Definir las propiedades
   de la pseudo-clase para efectuar
   el comportamiento }
```

En el código HTML, lo definimos de esta forma:

```
<ul><li><a href="enlace" class="nombre_
  de_la_clase"><span>Contenido
  tipográfico</span></a></li></ul>
```

Los enlaces pueden tener diferentes características, y son jerarquizados de acuerdo con su importancia y a su relevancia. La jerarquización de un enlace tiene que ver con el diseño y la ubicación que le otorgamos dentro del sitio.

Los diseños de los enlaces deben ser pensados en sus distintos estados, ya que, de esta forma, podremos saber si los espacios y las posiciones realmente están bien seleccionados y cuidados.

Los links que definimos como texto suelen ser, en su mayoría, los términos de contrato, las políticas de privacidad, las direcciones de e-mail (que se definen como tudirecciondemail) y los textos para ver más (view more) de los sitios dinámicos. También se usan en los sidebars de las webs dinámicas.

Los enlaces generados por bloques la mayoría de las veces suelen contener tipografías que no son estándar (no vienen instaladas por defecto con el sistema operativo), sombras, fondos realizados con degradados o con imágenes y, otras veces, con efectos visuales.

PROPIEDADES EN LOS ESTADOS DEL ENLACE

Las propiedades que podemos utilizar en los enlaces son diversas y variadas, y todas pueden aplicarse en los distintos estados de estos. Por ese motivo, recordemos que, como dijimos anteriormente, el link debe ser pensado en sus cuatro estados. Veamos las propiedades que tenemos disponibles:

- **background**: son aquellas que hacen referencia al fondo. Se utilizan cuando queremos colocar una imagen como botón en un bloque generado mediante una lista ordenada o desordenada, o cuando queremos utilizar un color de la misma forma.
- **color**: la usamos cuando queremos cambiar el color del enlace en alguno de sus estados o en todos ellos.
- **font**: sirve para generar cambios tipográficos, ya sea de familia tipográfica, variaciones en el cuerpo, inclinación o interlineado.
- **text-decoration**: nos permite quitar o colocar subrayado, tachado o, incluso, una sobrelínea.
- **visibility**: lo utilizamos cuando no queremos que se vea el texto de un bloque generado por lista.

En los enlaces es posible utilizar una gran cantidad de propiedades

Para finalizar, debemos tener en cuenta que el lenguaje HTML y CSS tienen un gran potencial, que día a día se desarrolla aún más con el objetivo de poder prescindir de ciertos objetos a la hora de construir enlaces y hasta el sitio completo. Esto es importante porque se han comenzado a implementar las etiquetas para HTML 5 y CSS 3, que nos permitirán generar los enlaces para navbar o botoneras sin necesidad de utilizar imágenes en muchos casos, como cuando queremos hacer uso de colores degradados, aplicar sombras o permitir el uso de tipografías no estandarizadas, que podrán ser alojadas en el servidor, leídas y cargadas por CSS. También nos permitirán generar composiciones con más de un fondo para producir enlaces o fondos web.

Multiple **choice**

▶1 ¿Para qué sirve Media Encoder?

a- Para efectuar más de una exportación.
b- Para controlar el uso de las capas.
c- Para eliminar una capa.
d- Para importar elementos al proyecto.

▶2 ¿Cómo se conoce también a la placa digitalizadora?

a- Transformadora.
b- Tarjeta de video.
c- Ubicadora.
d- Capturadora.

▶3 Mencione una parte de la placa digitalizadora.

a- Conector RJ-45.
b- Conector tipo BNC.
c- Memoria RAM.
d- Procesador.

▶4 ¿Qué significa DV?

a- Video en alta definición.
b- Audio y video.
c- Digital video.
d- Elemento de video.

▶5 ¿Para qué sirve Encore?

a- Para importar elementos.
b- Para codificar videos.
c- Para optimizar la codificación.
d- Para crear menús.

▶6 ¿Qué podemos agregar desde el panel Menú?

a- Botones, texto e imágenes.
b- Solo botones.
c- Títulos.
d- Capas.

Respuestas: 1-a, 2-d, 3-b, 4-c, 5-c, 6-a

Capítulo 7
Listas

El manejo de listas es importante para incorporar elementos adicionales en nuestros diseños; veremos cómo agregarlas.

RedUSERS.com

Listas: **definición**

Debido a la gran utilidad que tienen, las listas son uno de los elementos HTML que con mayor frecuencia encontraremos en los sitios web. Es importante conocer sus ventajas y las razones por las que es conveniente utilizarlas.

Las listas nos permiten mostrar la información de forma clara, porque ordenan visualmente el contenido de un sitio. Gracias a esto, el ojo puede recorrerlo con mayor rapidez y, así, focalizar en el contenido que es de interés para el usuario.

Por otra parte, en sitios con contenidos muy extensos, una lista sirve para organizar y separar algunos elementos y, de este modo, evitar lo que podría ser un gran bloque de texto. De esta manera, su lectura se hace más llevadera, y lo importante es más fácil de encontrar y reconocer.

Para analizar el uso de listas podemos visitar el sitio de un periódico, por ejemplo, **www.clarin.com**. Este diario argentino tiene un gran volumen de información que se encuentra segmentada y organizada por temáticas, secciones e importancia, entre otras categorías. Para lograr una diagramación clara del contenido, es importante utilizar elementos que le simplifiquen al lector la tarea de hallar lo que está buscando. Para esto, además de valernos de títulos, subtítulos, viñetas, recuadros, destacados y copetes, también emplearemos listas.

El uso de listas es sencillo y necesario para que el sitio respete los estándares. Este recurso puede usarse en sus formas **ordenadas** o **desordenadas**, y también **anidarse** (es decir, poner una lista dentro de otra) para conseguir diferentes niveles de lectura.

Más adelante veremos cómo deben organizarse las listas y cuál es su estructura. Por ahora, presentemos

FIGURA 1. En www.copenhagen.chopeh.com encontramos un ejemplo donde el menú se estructura a partir de una lista desordenada.

el código utilizado en www.copenhagen.chopeh.com para crear el menú:

```
<ul id="nav">
   <li class="home"><a href="index.
    php">Home</a></li>
   <li class="about"><a href="about.
     php">About Me</a></li>
   <li class="work"><a href="work.
     php">My Work</a></li>
   <li class="help"><a href="help.
     php">Help Me</a></li>
   <li class="contact"><a href="contact.
     php">Contact Me</a></li>
   <li><a class="download" target="_
     blank" href="CV-Pete-Lacey.pdf">
     Download CV</a></li>
  </ul>
```

La manipulación de listas a través de CSS nos brinda la posibilidad de cambiarles totalmente el estilo, para lograr estéticas muy atractivas y vanguardistas que están bastante lejos de la idea que comúnmente se tiene de ellas.

Por ejemplo, en www.pingdom.com/free encontramos en el banner principal un listado en dos columnas, cada uno de cuyos ítem contiene una imagen de fondo que funciona como icono. El uso de imágenes

Gracias a las listas, el ojo recorre la información de manera más rápida y focaliza en lo importante

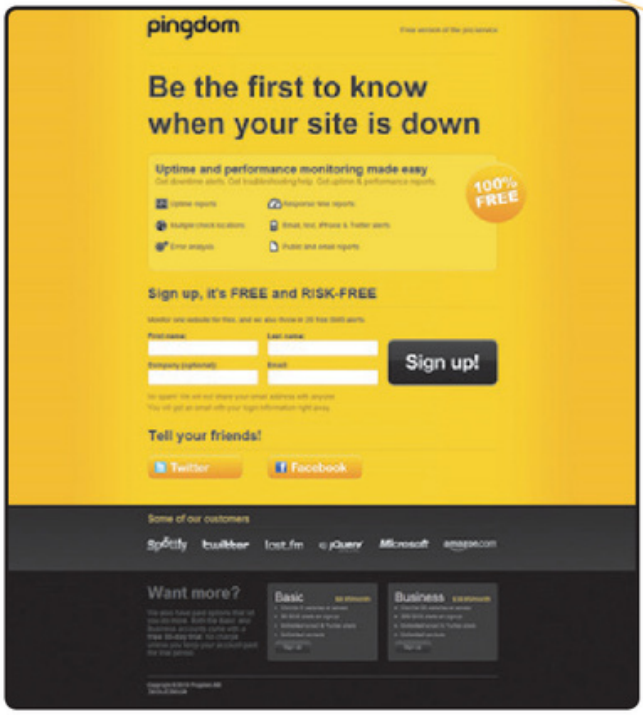


FIGURA 2. En www.pingdom.com/free podemos ver dos formas de uso de este recurso: una en el listado central, y otra en el footer.

que representen el tema listado es bastante frecuente y enriquece la apariencia visual del sitio. En el mismo sitio, dentro del footer, vemos dos listas con el clásico bullet circular, el uso más estándar de este recurso. En este caso, la información es secundaria, menos importante que la anterior, por lo cual la lista tiene una apariencia menos atractiva, que no compite visualmente con la principal.

Como podemos notar, las listas pueden usarse y manipularse a través de CSS de acuerdo con nuestras necesidades, otorgándoles mayor o menor relevancia según su tratamiento.

TRES MOTIVOS PARA UTILIZAR LISTAS

Aunque podemos conseguir los mismos resultados

mediante el uso de CSS aplicado a cualquier elemento HTML, la diferencia al utilizar listas está en su practicidad; veamos las razones.

En principio, aplicar un estilo a los elementos `<li>` de una lista hace que, luego, si queremos cambiarlo, nos resulte mucho más sencillo hacerlo. Esto se debe a que no tendremos todos los elementos separados, en cuyo caso deberíamos cambiar el estilo en cada uno de ellos.

Aunque más adelante conoceremos con más detalles los tipos de listas y sus características, cabe mencionar que, si tenemos una lista ordenada y queremos cambiar de lugar un elemento, el navegador interpretará que es una `<ol>` (lista ordenada) y, automáticamente, numerará los elementos. En cambio, si hubiésemos escrito los números a mano, tendríamos que renumerarlos uno por uno, algo realmente incómodo, sobre todo, en listas muy extensas.

El uso de listas sirve para que el lector las interprete como tal, lo que ayuda mucho a la lectura y a la ubicación de contenido, dado que las listas poseen, además, una estructura semántica.

Todo esto significa que las listas deben implementarse con las etiquetas que tienen este fin, es decir, `<ol>` y `<ul>`. Para nosotros, esta forma de trabajo aumentará la practicidad en el manejo de estos elementos. Para los usuarios son una ayuda porque alivian mucho la lectura, en especial, en sitios que tienen una gran carga de contenido, cuya información puede ser jerarquizada.

Si tenemos un sitio con bloques de texto muy extensos, y otro con información destacada, listas, textos en negrita y diferentes tamaños de tipografías, sin dudas será mucho más sencillo el recorrido visual del segundo, tanto para dar un vistazo general como para realizar una lectura detallada.

Es importante tener en cuenta todos estos elementos que pueden ayudar a enriquecer nuestras interfaces, no solo porque resultan agradables y estéticos sino también por su utilidad comunicacional.

Listas ordenadas y no ordenadas

Ya hemos mencionado que, en ciertas ocasiones, debemos recurrir a listas ordenadas cuando precisamos, por ejemplo, numerar los ítem.

En cambio, en otros casos, simplemente necesitamos mejorar la presentación de la información y, entonces, recurrimos a listas no ordenadas.

LISTAS ORDENADAS

Estas listas se caracterizan por estar encabezadas por algún tipo de elemento que deja implícito su orden, como podrían ser números o letras.

Todos los elementos pertenecientes a una lista ordenada se encuentran dentro de las etiquetas **<OL>** y **</OL>**, que significan Ordered Lists. La primera le indica al navegador que comienza una lista ordenada, en tanto que la última señala que esta termina. A partir de la primera etiqueta, se comienza la numeración, y si no indicamos otra cosa a través de CSS, se la implementará con el sistema decimal, que es el más difundido.

Dentro de las etiquetas que vimos, encontramos los elementos que componen la lista. Cada uno debe tener, al comienzo, la etiqueta **
**, que significa list ítem; y terminar con **
**. Como podemos imaginar, este tipo de listas no utiliza imágenes ni bullets, ya que su propósito es solo mostrar un orden.

Es posible manipularlas a través de CSS por medio de la propiedad **List-style-type**, que puede tomar los siguientes valores:

- decimal-leading-zero: enumera los elementos a partir de 01.
- decimal: enumera los elementos a partir de 1.
- lower-roman: lista los elementos identificándolos con números romanos en minúsculas.
- upper-roman: lista los elementos identificándolos con números romanos en mayúsculas.
- lower-alpha: identifica los elementos con letras a partir de la "a", en minúsculas.
- upper-alpha: identifica los elementos con letras a partir de la "a", en sus versiones mayúsculas.

Veamos, a continuación, un ejemplo del código HTML con la estructura de una lista ordenada:

```
<ol>
        <li>Elemento número uno</li>
        <li>Elemento número dos</li>
        <li>Elemento número tres</li>
        <li>Elemento número cuatro</li>
        <li>Elemento número cinco</li>
</ol>
```

A través de CSS, controlaremos, en este caso, el tipo de numeración y otros aspectos que ya conocemos, como el padding de cada elemento, el color, la familia tipográfica y su tamaño. Veamos el ejemplo:

```
ol{
        list-style-type: decimal;
```

OTRAS FORMAS DE LISTAR

Las explicadas anteriormente son las maneras más habituales para listar, pero también existen, por ejemplo, **lower-greek**, que identifica los elementos con números griegos; o las propiedades **hebrew**, **armenian**, **hiragana**, **hiragana-iroha** y **katakana-iroha**, entre otras.

```
        padding: 20px;
        font-family: arial;
        font-size: 15px;
        color: olive;
}
ol li{
        padding-left: 20px;
}
```

1. Elemento número uno
2. Elemento número dos
3. Elemento número tres
4. Elemento número cuatro
5. Elemento número cinco

FIGURA 3. **Aquí podemos apreciar cómo se vería en un navegador la lista ordenada con estilo decimal creada con el primer código.**

Cuando trabajamos con listas, es fundamental cerrar tanto la lista como cada uno de los elementos **<li>**. Si olvidamos hacerlo, los sitios no validarán su código HTML y, además, tendremos problemas con el código CSS. Por ejemplo, puede ocurrir que el estilo de nuestros elementos **<li>** se propague por todo el sitio si no cerramos esta etiqueta, o que la lista adopte otras posiciones con respecto a los elementos adyacentes si no la finalizamos.

FIGURA 4. **El uso de este tipo de listas es muy frecuente en sitios de enciclopedias o diccionarios, como www.wordreference.com.**

LISTAS NO ORDENADAS

La listas no ordenadas tienen la misma estructura de las anteriores, excepto que las etiquetas que las encierran son **
** y **
** (**unordered list**). Estas listas se diferencian de las ordenadas porque, en vez

de identificar los elementos a través de números o letras, en general se emplean bullets o imágenes. La estructura de esta lista es la siguiente:

```
<ul>
        <li>Primer elemento de nuestra
           lista</li>
        <li>Segundo elemento de nuestra
           lista</li>
        <li>Tercero elemento de nuestra
           lista</li>
        <li>Cuarto elemento de nuestra
           lista</li>
        <li>Quinto elemento de nuestra
           lista</li>
</ul>
```

Para una lista no ordenada, el código CSS que utiliza un cuadrado como bullet es el siguiente:

```
ul{
        list-style-type: square;
        font-family: arial, helvetiva,
            verdana, sans-serif;
        font-size: 15px;
        color: #4b0082;
}
```

Ya hemos adelantado que muchos menús de navegación se generan a partir del uso de listas. A continuación, veremos un ejemplo de ese uso, en el que debemos prestar atención al lugar donde se colocan la etiqueta **<a>** y su cierre:

```
<ul>
     <li><a ref="http://paginaejemplo.
          com.ar">Primer elemento de
          nuestra lista</a></li>
     <li><a ref=" http://paginaejemplo.
          com.ar">Segundo elemento de
          nuestra lista</a></li>
     <li><a ref=" http://paginaejemplo.
          com.ar">Tercero elemento de
          nuestra lista</a></li>
     <li><a ref=" http://paginaejemplo.
          com.ar">Cuarto elemento de
          nuestra lista</a></li>
</ul>
```

LISTAS ANIDADAS

Usamos esta denominación para definir una lista que está dentro de otra. Puede aplicarse a listas tanto ordenadas como no ordenadas, de modo que podemos crear una combinación de ambas. Lo que obtenemos de esta forma es una lista dentro de otra:

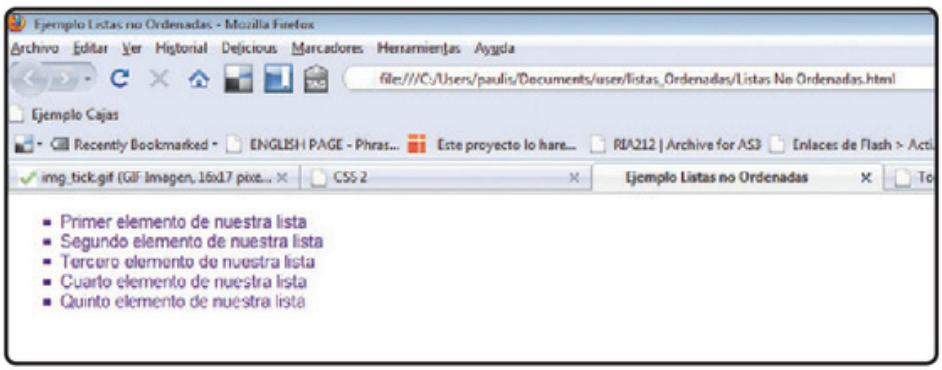


FIGURA 5. Este es el resultado que obtenemos luego de aplicar el código a una lista no ordenada.

una contenedora, que es la principal; y otra secundaria, que estará indentada con respecto a la primera.

En este tipo de listas, es fundamental cerrar y abrir las etiquetas donde corresponde, para así lograr la estructura correcta. Veamos un ejemplo de lista anidada:

```
<ul>
 <li>Letras del alfabeto
  <ol>
   <li>Letra A</li>
   <li> Letra B</li>
   <li> Letra D</li>
  </ol>
  </li>
 <li>Colores</li>
 <li>Números</li>
 <li>Otros</li>
</ul>
```

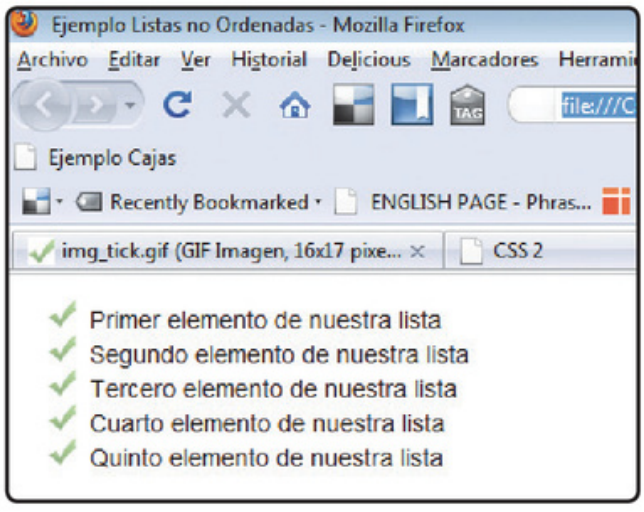


FIGURA 6. **En este caso, podemos ver una lista no ordenada que utiliza una imagen para decorar cada uno de los elementos.**

¿Dónde utilizamos listas?

Cuando pensamos en listas, tal vez lo primero que nos viene a la mente es una sucesión de palabras sin formato, una debajo de la otra.

Aunque pensemos que las listas no se usan demasiado o que no son de utilidad, son elementos muy utilizados y totalmente controlables a través de CSS.

A continuación, veremos ejemplos de sitios que utilizan listas y estudiaremos cómo las implementan.

LISTAS ESTÁNDAR

Comencemos por la tradicional lista con un bullet circular. Podemos verla en **www.mycurli.com**, donde se la utiliza para hacer un punteo del texto principal y, además, en el footer, donde conforma tres columnas para enumerar los diferentes links.

Las listas son elementos muy utilizados y totalmente controlables a través de CSS.

LISTAS CON IMÁGENES

Es muy frecuente utilizar listas para los links de navegación de un sitio. Como ejemplo, en **www.redvelvetart.com** notamos que el menú principal está compuesto por un listado de links. Cada <li> tiene, en su interior, un elemento <a> al que se le aplica la imagen icónica mediante CSS.

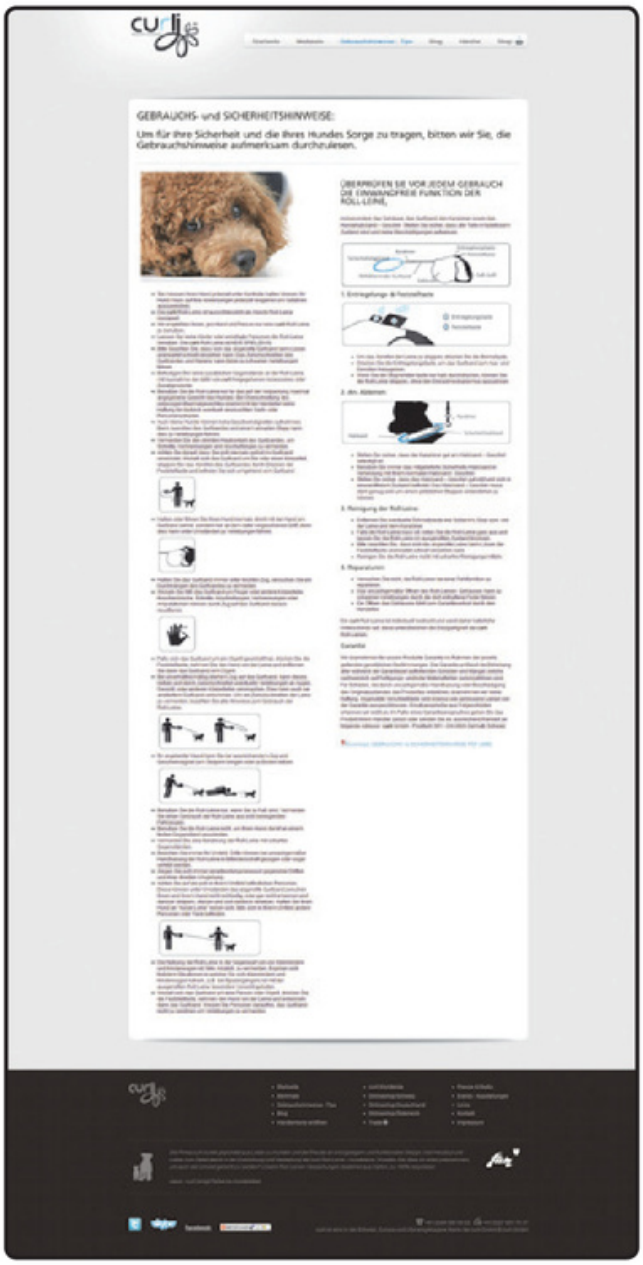


FIGURA 7. **Este sitio (www.mycurli.com) muestra diferentes formas de hacer uso de listas con bullets.**

En este caso, podemos darnos cuenta de que en una sola imagen encontramos todos los estados del botón, es decir, el botón en descanso y el rollover, que también se define por CSS mediante la propiedad denominada **hover**.

De la misma manera se tratan los otros menús, con excepción de los tags en el ángulo inferior izquierdo, en donde no se utilizan imágenes sino fuentes.

FIGURA 8. **El sitio www.redvelvetart.com recurre a imágenes para decorar varias listas, que funcionan como menús.**

Hallamos otro ejemplo de este tipo en el sitio de Adobe (**www.adobe.com/es**), donde podemos ver dos columnas de listas, cada una con una miniatura que representa el programa al que se hace mención. Luego hay un listado de noticias, cada una de las cuales representa un link hacia su ampliación. Por debajo, aparece un menú horizontal; en él, cada link está flotado hacia la izquierda, con margin y padding configurados para generar la separación entre ellos.

En **www.storypixel.com** podemos ver listas en varios lugares. Primero tenemos el menú principal, y

FIGURA 9. **El sitio de Adobe (www.adobe.com/es) utiliza las imágenes de los iconos de cada programa para decorar la lista de productos.**

luego aparecen las categorías, a las cuales, mediante CSS, se les aplica una miscelánea como background de etiqueta, de la siguiente manera:

```
#nav_categories li {
        background:url
         ("http://s3.amazonaws.com/
         creativeharvest/blt-tag-a.gif")
         no-repeat left 6px transparent;
}
```

En este ejemplo vemos el uso de la propiedad shorthand para definir, en una misma línea, el uso de una imagen como background, la asignación de no-repeat para que la imagen sea una sola (y no obtener un mosaico como resultado), una posición en X y en Y, y el color de fondo transparente.

Por otro lado, en el menú **My Diet**, cada link está compuesto por una imagen de fondo, que se utiliza mediante un código donde se define la clase del link, se aplica una imagen como background y se establece su altura:

```
#my-diet-list li.item-flash {
background:url("http://s3.amazonaws.com/
creativeharvest/my-diet-flash.gif")
no-repeat scroll left top transparent;
height:80px;
}
```

FIGURA 10. **En el sitio www.storypixel.com encontramos diferentes formas de utilizar imágenes aplicadas a las listas.**

LISTAS EN MENÚ DE NAVEGACIÓN

En www.revistaohlala.com tenemos un claro ejemplo de cómo usar listas para realizar un menú. En esta estructura, el primero y el último link se controlan con CSS de manera separada del resto, es decir, aplicándoles clases diferentes:

```
<ul>
        <li class="first">
<a href="">Link1</a></li>
        <li><a href="">Link2</a></li>
        <li><a href="">Link3</a></li>
        <li><a href="">...</a></li>
        <li class="tab-sedal">
<a href="">Link4</a></li>
</ul>
```

Las distintas longitudes de cada botón están determinadas por el largo de la palabra. Cada **<li>** tiene un margin-left, que si fuera utilizado para todos los links, se aplicaría también al primero, y entonces este no quedaría alineado. Por otro lado, el último link posee una clase especial donde se aplican una imagen y un color diferente como background.

El resto de los botones posee el mismo código CSS, que es el siguiente:

```
ul#nav li {
        background-color:#7F7F7F;
        float:left;
        height:19px;
        margin-left:2px;
        padding:6px 25px;
        text-align:center;
}
```

FIGURA 11. **En muchos casos, necesitaremos aplicar distintos estilos a los diferentes elementos de una lista. Encontramos un ejemplo de esto en www.revistaohlala.com.**

En www.zaum.co.uk primero vemos listas con **<li>**, con un elemento debajo del otro; luego, tanto en el menú como en el footer, bajo los títulos **Who**, **What**, **Where** y **Why**, encontramos listas con **<li>** flotados hacia la izquierda.

En el menú tenemos el siguiente código, que muestra la sección seleccionada con un borde blanco debajo:

```
#top-nav ul li a.active {
border-bottom:3px solid #FFFFFF;
}
```

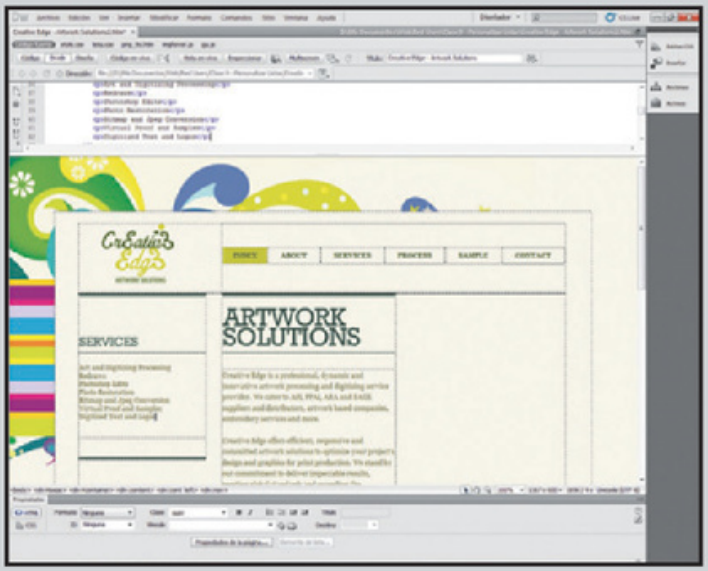
FIGURA 12. Además de usarlas en el menú, www.zaum.co.uk utiliza listas para presentar otros grupos de elementos.

Personalizar **listas**

Como sabemos, las listas son una herramienta muy utilizada a la hora de diseñar un sitio web. Se trata de elementos que nos permiten enumerar puntos que queremos que el visitante tenga en cuenta, pero también nos dan la posibilidad de realizar diversos tipos de botoneras.

PASO A PASO /1
Personalizar listas

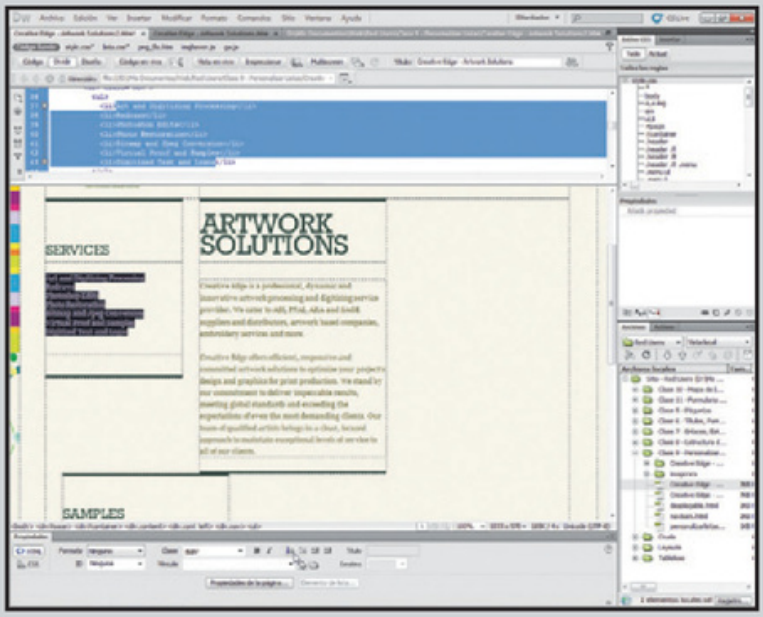
1



Luego de crear un documento HTML y un archivo de hoja de estilos llamado **`llsta.css`**, escriba la cantidad de párrafos que quiere convertir en lista. Las listas no necesariamente deben ser extensas. Aquí, los párrafos que transformará se encuentran por debajo de **`SERVICES`**, dentro de un div con una clase denominada **`nav`**.

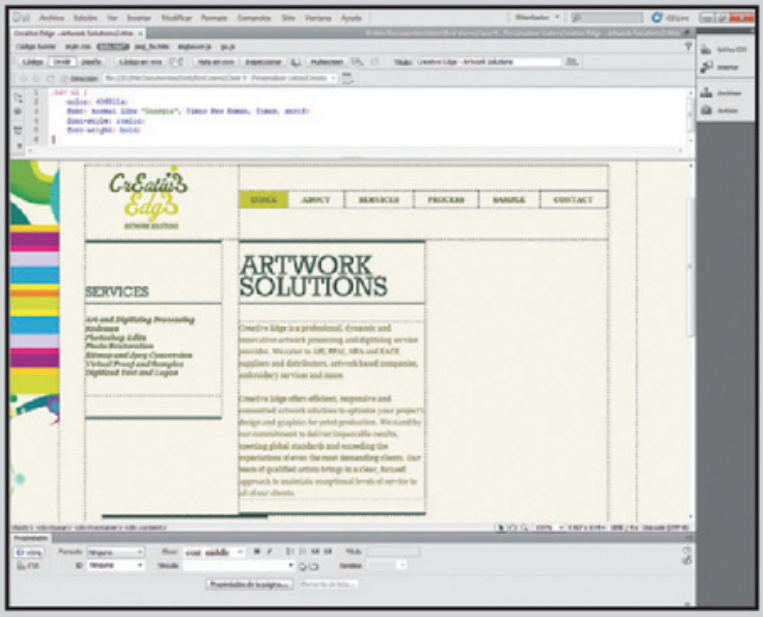
PASO A PASO /1
(Cont.)

2



Seleccione todos los párrafos que quiere convertir en listas. Luego, diríjase a la barra **Propiedades** y haga clic en **Lista sin ordenar**. Observe cómo cambian el código y el aspecto.

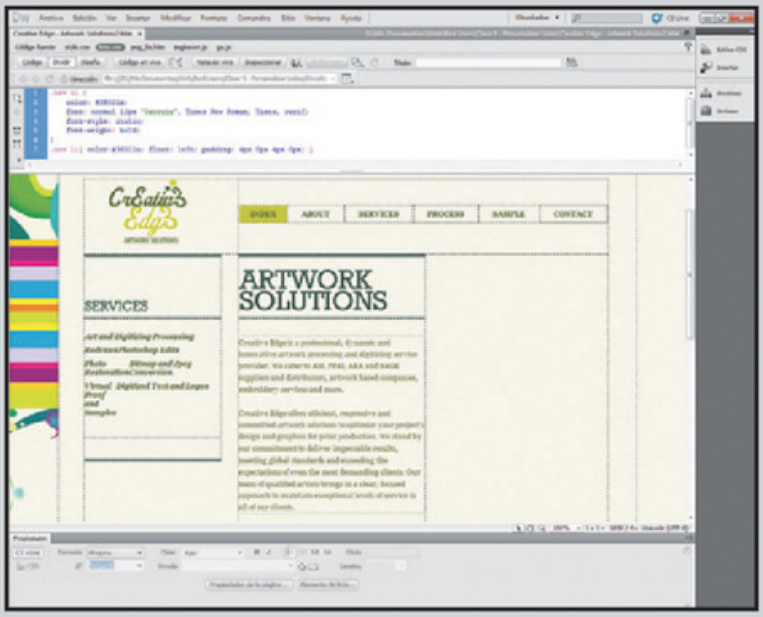
3



Para comenzar a darle propiedades a la lista, vaya al archivo **lista.css** y, dentro de él, escriba `.nav ul { color: #36511a; font: normal 12px "Georgia", Times New Roman, Times, serif; font-style: italic; font-weight: bold; }`. Aquí puede ver aplicadas las propiedades que le dará a la lista desordenada a nivel general.

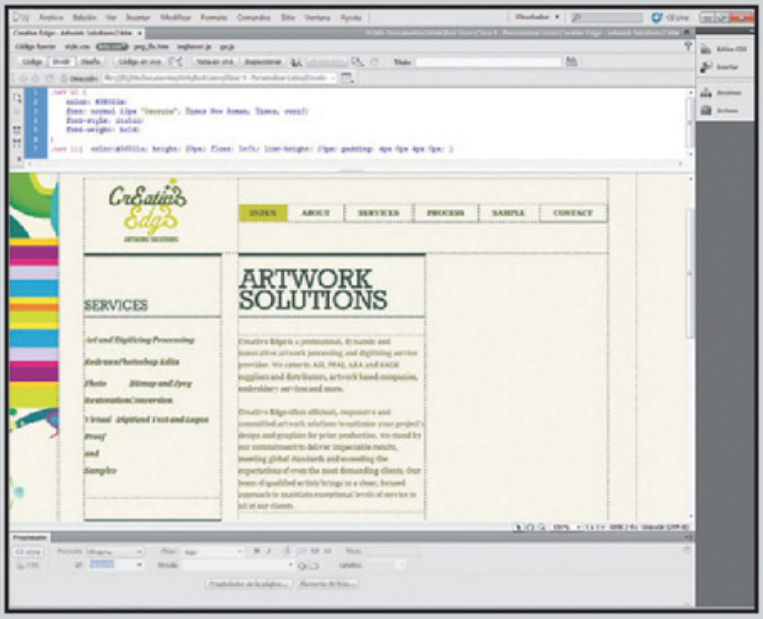
PASO A PASO /1
(Cont.)

4



Ahora, pase a mejorar la presentación de las etiquetas **<li>**. Para hacerlo, a la hoja de estilos agréguele el texto **`.nav li { color: #36511a; float: left; padding: 4px 0px 4px 0px; }`** para que las etiquetas **<li>** se ubiquen una al lado de la otra, que tengan un color determinado y una distancia particular entre ellas.

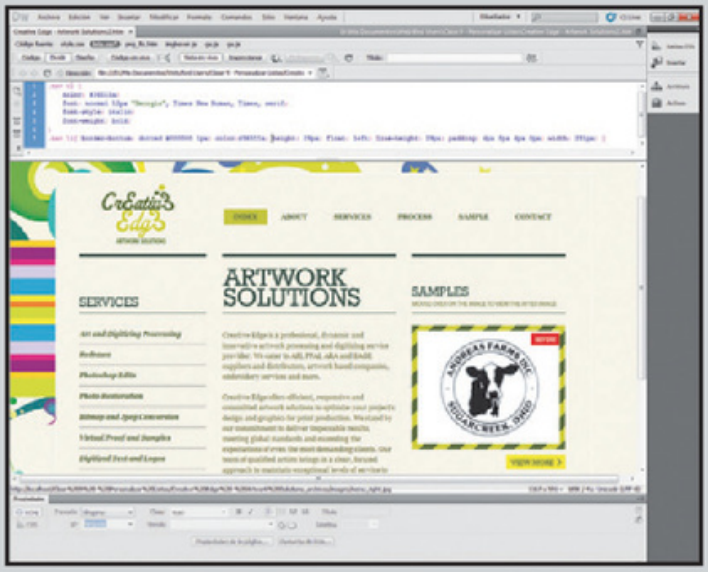
5



Para aumentar el espacio entre elementos, agregue en el código anterior un alto a la etiqueta **<li>**, mediante la aplicación de una altura en cada línea. El código es el siguiente: **`.nav li { color: #36511a; height: 29px; float: left; line-height: 29px; padding: 4px 0px 4px 0px;}`**.

PASO A PASO /1
(Cont.)

6



Para finalizar, a la etiqueta **<li>** dele un borde y un ancho, para así tener la lista diseñada y ordenada como desea. El código semántico debe quedar así: **.nav li{ border-bottom: dotted #000000 1px; color:#36511a; height: 29px; float: left; line-height: 29px; padding: 4px 0px 4px 0px; width: 231px; }**.

Como dijimos al principio, las listas son una herramienta muy útil cuando necesitamos enumerar cierta información. También podemos observar que poseen múltiples aplicaciones, ya que nos permiten generar navbars e, incluso, pueden usarse para ciertas estructuras en reemplazo de las tablas.

Creación de menús **de navegación**

La creación de menús de navegación mediante **listas** es una de las prácticas más comunes en la actualidad. Como ya vimos, cuando generamos una **lista desordenada**, por defecto el navegador muestra la viñeta **bullet**, en tanto que si creamos una **lista ordenada**, la viñeta es **decimal**.

Ya sabemos que una de las ventajas de utilizar reglas CSS es que podemos cambiar las características de las listas y controlar sus propiedades básicas, que

son **List-style-type** para definir el estilo de la viñeta en la lista, **List-style-image** para cambiar la imagen de la viñeta y **List-style-position** para ubicar la viñeta por fuera del texto (**outside**) o por dentro de él (**inside**).

La modificación de estas propiedades permite obtener una gran variedad de resultados, ajustando el menú al diseño que hayamos realizado. Para hacerlo de forma simple, CSS nos brinda el shorthand **list-style**, cuyo objetivo es escribir menos, dado que permite aplicar todas las propiedades de la lista en un formato abreviado.

CREACIÓN DE UN MENÚ DE NAVEGACIÓN VERTICAL

Como mencionamos, las **listas**, además de cumplir su función natural, se aplican en la creación de **navbars** verticales u horizontales. Para este caso, lo primero que debemos hacer es definir la etiqueta **<ul>** dentro del código CSS, quitándole el estilo a la lista; el código es **#botonera ul { list-style: none; }**. Luego, definimos la etiqueta **<li>** mediante el código **#botonera ul li { font: normal 12px/12px Arial, Helvetica, Sans-serif; }**, que permite elegir la tipografía que se usará si las imágenes no se cargan.

Para finalizar, definimos la etiqueta **<a>** y armamos las clases para cada botón, repitiendo la creación de la **clase** y aplicando las propiedades del **background** según la cantidad de botones que tengamos.

```
#botonera ul li a { height: 50px;
text-decoration: none; width: 142px; }
 .quienessomos { background: url
 (../imagenes/jpg/botonera/quienessomos.
 jpg) center top;}
```

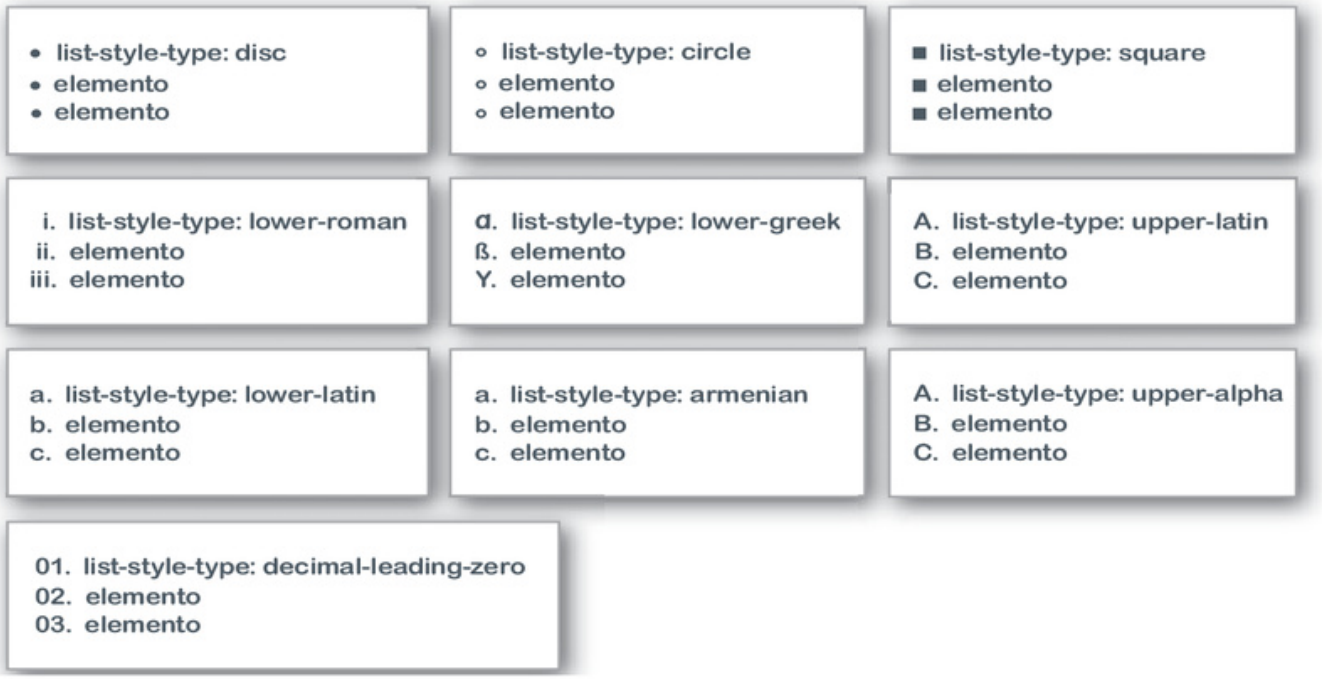


FIGURA 13. Aquí observamos ejemplos de los distintos tipos de viñetas que podemos aplicar a las listas.

Dentro de nuestras listas, hemos escrito el nombre de cada botón. Cuando damos a esos botones las propiedades con reglas CSS, notaremos que ese primer texto (que se utiliza cuando se ejecutan navegadores que no muestran los estilos) se superpone con el estilo. Por este motivo, vamos a agregar una etiqueta **span** de modo que el texto se vuelva invisible. Además, asignaremos las propiedades en el estado **hover** del botón:

```
#botonera ul li a span { text-decoration:
none; visibility: hidden; }
#botonera ul li a:hover { background-
position: center bottom; }
```

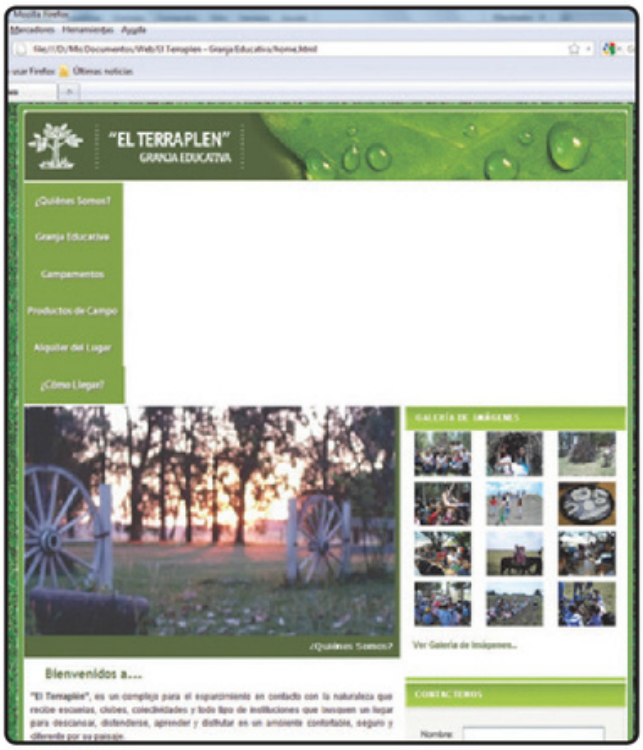


FIGURA 14. En el sitio de El Terraplén (www.granjaelterraplen.com.ar), encontramos un menú vertical creado con las propiedades vistas en nuestro ejemplo.

CREACIÓN DE UN MENÚ DE NAVEGACIÓN HORIZONTAL

A partir de cualquier menú vertical, es posible crear uno horizontal. Para lograrlo, el cambio más importante lo generamos en la etiqueta **<a>**, aunque también pueden hacerse modificaciones en **<li>** o en ambas. Esto último es recomendable debido a que algunas versiones de IE mostrarán y cumplirán los cambios de propiedades dependiendo de dónde se encuentren estos. Entonces, tanto en la etiqueta **<li>** como en **<a>**, agregamos la propiedad **float: left**. Las etiquetas deberían quedarnos así:

```
#botonera ul li { float: left; }
#botonera ul li a {
float: left;
        height: 50px;
        text-decoration: none;
        width: 142px;
}
```

FIGURA 15. En esta imagen se muestra el menú vertical convertido en horizontal, gracias a las propiedades que se han colocado más arriba.

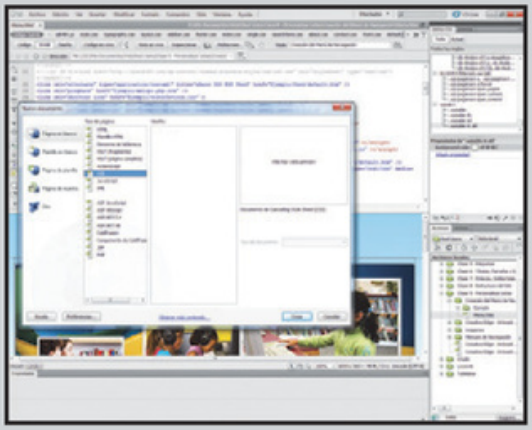
Además de los menús verticales y horizontales básicos, si utilizamos las reglas CSS para las pseudo-clases **hover** y **active**, podríamos crear navbars avanzadas que tuviesen pestañas, sombras, tipografías no estándar e, incluso, submenús internos que funcionen y se desplieguen (esto con la ayuda del lenguaje JavaScript). Recordemos que las botoneras o navbars deben estar contenidas dentro de un **div** o, si es en **HTML5**, dentro de la etiqueta **nav**.

Crear un menú **de navegación**

Como vimos hasta aquí, en la actualidad las listas son fundamentales para la creación de navbars, también llamadas menús de navegación. Aunque más adelante veremos cómo generar el menú de nuestro sitio, aquí exploraremos la creación y la manipulación de listas para obtener un menú más complejo.

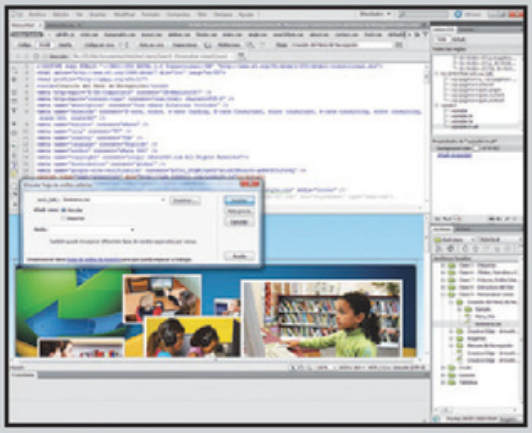
PASO A PASO /2
Crear menú de navegación

1



Lo primero que hará será crear un archivo CSS que contendrá el menú de navegación. Para hacerlo, vaya a **Archivo/Nuevo...**, elija **CSS** en **Tipo de página** y presione **Crear**. Luego, guarde el archivo como **botonera.css**.

2



Ya creó un documento HTML y lo guardó. Ahora, diríjase al panel derecho **Estilos CSS** y haga clic en **Adjuntar hoja de estilo**. Cuando se abre **Adjuntar hoja de estilos externa**, presione **Examinar…**, busque el archivo **botonera.css** y pulse **Aceptar**. Finalmente, vuelva a hacer clic en **Aceptar**.

PASO A PASO /2
(Cont.)

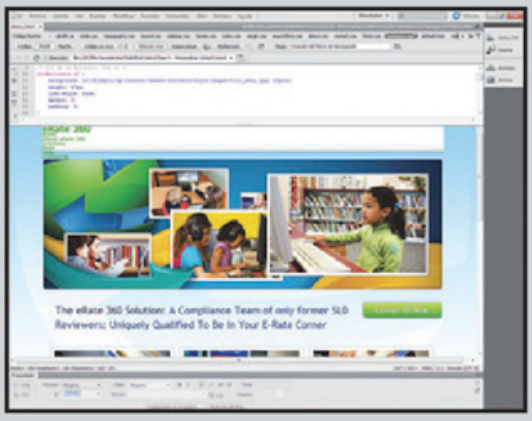
3

Desde el panel **Insertar**, cree un div llamado **botonera** y en él escriba el texto de los botones que formarán la navbar. En este caso, además, incluya una línea vacía al comienzo y otra al final del texto. Seleccione los elementos y, en la barra inferior **Propiedades**, presione **Lista sin ordenar**.

4

Dentro de las etiquetas **<li>**, coloque cada enlace mediante etiquetas **<a>**, excepto en el logo del sitio, al que previamente debe ponerle una etiqueta **<h1>**. Además, coloque una etiqueta **class="izquierdo"** en el espacio en blanco inicial, una **class="derecho"** en el espacio en blanco final y **class="sans-divider"** en la que tiene el logo o marca.

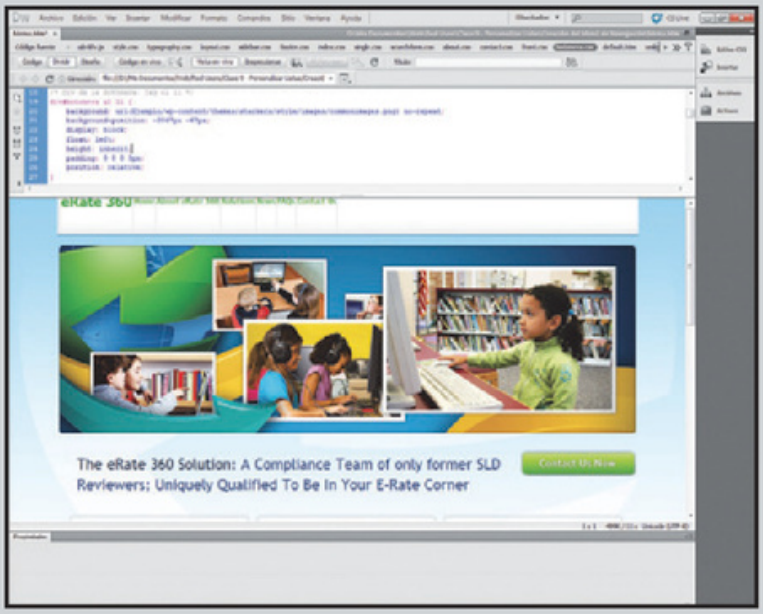
5



Vaya al archivo **botonera.css** y allí escriba **div#botonera {display: block; margin: 0 auto; padding: 0 0 0 2px; width: 988px;}** para definir las características del div creado en el paso anterior. Defina las propiedades de la etiqueta **<ul>** escribiendo **div#botonera ul {background: url(Ejemplo/wp-content/themes/starkers/style/images/tile_menu.jpg) repeat; height: 67px; list-style: none; margin: 0; padding: 0;}**, la URL de la imagen es la ubicación que le haya dado al fondo.

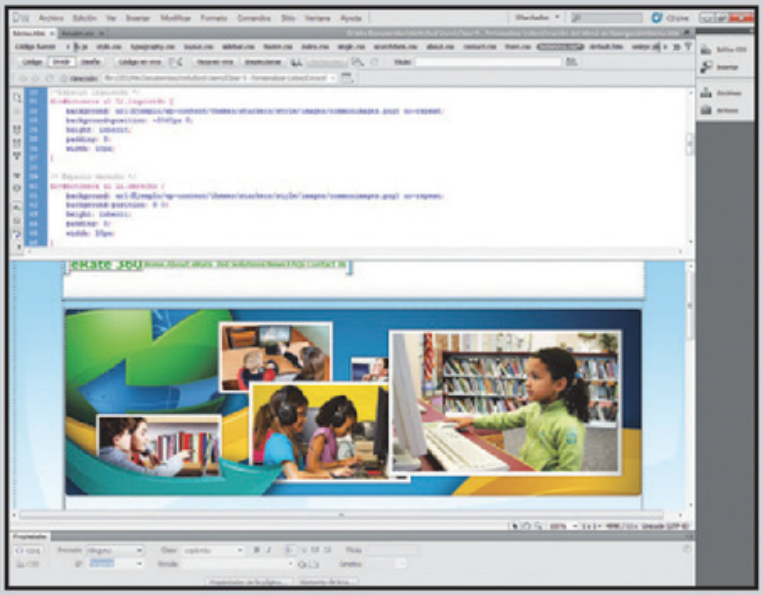
PASO A PASO /2
(Cont.)

6



Defina la etiqueta **<li>** de la botonera con **div#botonera ul li {background: url(Ejemplo/wp-content/themes/starkers/style/images/commonimages.png) no-repeat; background-position: -2047px -67px; display: block; float: left; height: inherit; padding: 0 0 0 3px; position: relative;}**.

7



Para definir los espacios en blanco escriba el código **div#botonera ul li.izquierdo {background: url(Ejemplo/wp-content/themes/starkers/style/images/commonimages.png) no-repeat; background-position: -2041px 0; height: inherit; padding: 0; width: 10px;}**, para el lado izquierdo; y **div#botonera ul li.derecho {background: url(Ejemplo/wp-content/themes/starkers/style/images/commonimages.png) no-repeat; background-position: 0 0; height: inherit; padding: 0; width: 10px;}**.

PASO A PASO /2
(Cont.)

8

Ahora, defina la clase **.sans-divider** con el código **div#botonera ul li.sans-divider{background: none; padding: 0;}**.

9

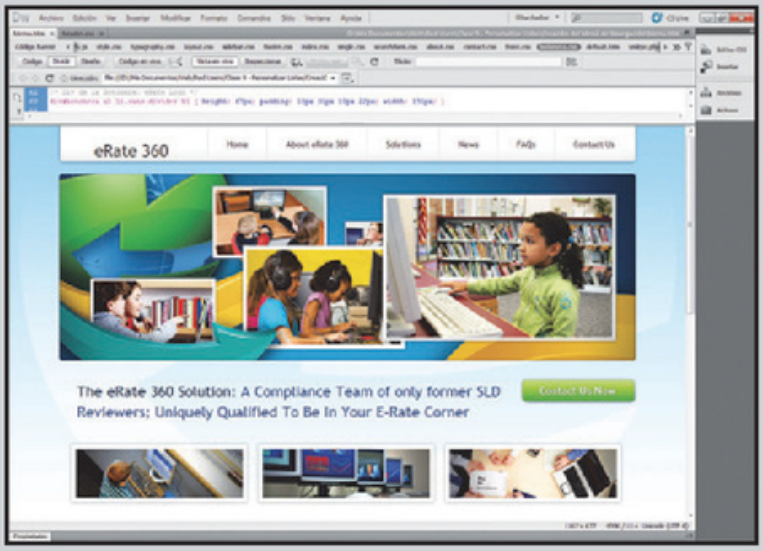
La etiqueta **<a>** debe ser definida con **div#botonera ul li.sans-divider a {background:none; padding:0;}** y **div#botonera ul li a{color:#121212; display: block; font: normal 1.4em "Trebuchet MS", Verdana, Geneva, Arial, Helvetica, sans-serif; height: 67px; line-height: 65px; padding: 0 31px;}**.

PASO A PASO /2
(Cont.)

10

Luego de aplicar la etiqueta **<a>**, defina todos sus estados escribiendo en el archivo **botonera.css** lo siguiente: **div#botonera ul li a:hover {background: url(Ejemplo/wp-content/themes/starkers/style/images/tile_menuhover.jpg) repeat; text-decoration: none;}**. Con este código, modifica todos los botones en su estado **hover**.

11



Lo próximo que definirá será el tamaño que deberá tener la etiqueta **<li>** de la marca o logo. En este caso, escriba **div#botonera ul li.sans-divider h1{height: 67px; padding: 10px 31px 10px 22px; width: 191px;}**.

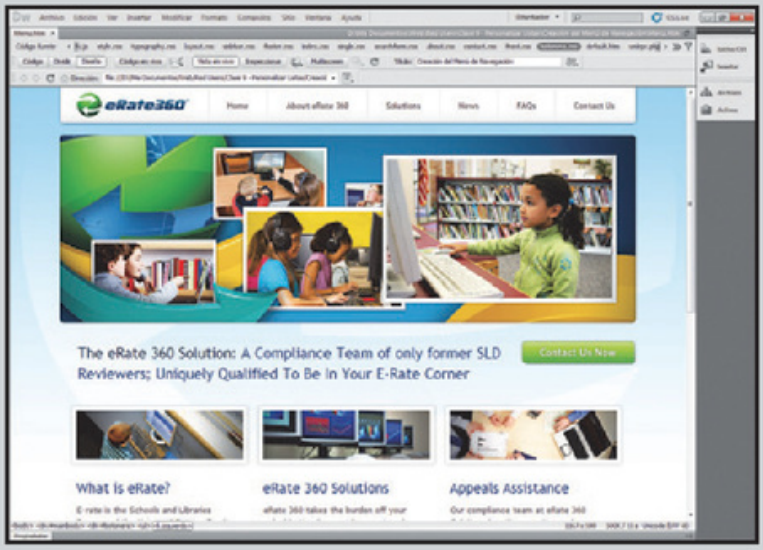
Hemos creado de esta forma nuestro primer menú con listas, para conocer mejor uno de los múltiples usos que tiene este elemento. Tengamos en cuenta que los menús o botoneras deben ser diseñados

PASO A PASO /2
(Cont.)

12

Pase a definir ahora las propiedades de la etiqueta **<h1>** cuando tiene en su interior una etiqueta **<a>** mediante **div#botonera ul li.sans-divider h1 a {background: url(Ejemplo/wp-content/themes/starkers/style/images/commonimages.png) no-repeat; background-position: -450px -300px; height: 44px; overflow: hidden; text-indent: -9999px;}**.

13



Una vez que haya realizado los pasos anteriores, podrá decir que tiene su primera botonera o menú armado. Haga clic en **Vista en vivo** para ver su funcionamiento.

previamente y que, antes de comenzar a maquetar, hay que tener en claro lo que se quiere obtener, ya que de eso dependerá la forma en la que asignemos las propiedades adecuadas.

Multiple **choice**

1 ¿Qué son los metadatos?

a- Textos para títulos.
b- Parámetros relacionados con un recurso.
c- Datos sobre el video.
d- Nombre de las capas de un proyecto.

2 ¿Cómo encontramos los metadatos?

a- En la información de las capas.
b- En el panel de animación.
c- En las propiedades del texto.
d- En el panel Metadatos.

3 ¿Con qué aplicación podemos subtitular diálogos?

a- Premiere.
b- After Effects.
c- Encore.
d- Flash.

4 ¿Desde qué panel aplicamos los estilos?

a- Animación.
b- Capas.
c- Estilos.
d- Tareas.

5 ¿Cómo agregamos sombra a un objeto?

a- **`Objeto/Agregar sombra`**
b- **`Objeto/Sombra/Nueva`**
c- **`Objeto/Agregar`**
d- **`Sombra/Agregar sombra`**

6 ¿Cuál es el punto de partida para la autoría de un DVD?

a- La planificación previa.
b- La edición del menú.
c- La elección de los elementos.
d- La creación e las capas.

Respuestas: 1-b, 2-d, 3-c, 4-c, 5-a, 6-a.

Capítulo 8
Formularios

Aquí conoceremos qué son los formularios y de qué forma podemos agregarlos a un sitio web.

RedUSERS.com

Formularios

Los formularios web son muy similares a los impresos en papel y poseen su misma finalidad o función. Estos elementos permiten transmitir información entre un equipo cliente (el del usuario) y el servidor.

Pueden ser de muy diversos tipos, dependiendo del objetivo que deban cumplir. Entre las opciones más comúnmente utilizadas, encontramos los de contacto, los de suscripción, los de pedido, los de votación, los de encuesta y los que se utilizan para ingresar un currículum en una bolsa de trabajo.

Todos ellos se realizan en cuatro partes: la primera es la estructura dada por el código HTML, la segunda es la estética otorgada por el código o las reglas CSS, la tercera es la validación e interacción del lado del cliente (mediante los eventos y el código JavaScript) y la cuarta es la programación del lado del servidor mediante el código PHP (hablaremos de este lenguaje más adelante).

Según su complejidad, encontramos dos clases de formularios:

- **Básicos**: son aquellos que constan de muy pocas opciones y que están elaborados, principalmente, con etiquetas `<form>` e `<input>`. Dentro de este grupo, podemos mencionar los formularios de suscripción o de login, entre otros.
- **Avanzados**: son aquellos formularios complejos que están elaborados no solo por las etiquetas `<form>` e `<input>`, sino que además utilizan `<fieldset>` para agrupar o emblocar la información. Suelen requerir, además, la etiqueta `<label>` para generar una descripción de lo que se espera que se coloque en los campos que hay que completar. Ejemplo de este caso son los formularios de registro, los de compra y los de ingreso de currículum, entre otros.

Los formularios son la principal vía de comunicación entre los visitantes y los dueños de un sitio

Si analizamos los formularios según su finalidad, podemos hacer la siguiente división:

- **De contacto**: son los más utilizados en los sitios web. En ellos se piden los datos mínimos para poder comunicarnos con los responsables del sitio y, por lo general, sirven para evacuar alguna duda que tengamos.
- **De suscripción**: se utilizan cuando queremos que nos llegue información relacionada con algún tema de nuestro interés o que nos llama la atención. En la mayoría de los casos, solo nos solicitan una dirección de e-mail.
- **De registro**: son el segundo grupo de formularios más utilizados en los sitios web, después de los de contacto. En ellos debemos colocar nuestros datos personales, un nombre de usuario y una contraseña, para tener acceso a ciertas partes del sitio restringidas a quienes no se registren. Son muy habituales para formar parte de redes sociales, abrir cuentas de e-mail o participar en foros.

FIGURA 1. Con pocos elementos, los formularios de login son necesarios para controlar el acceso de los usuarios, como vemos en www.deploycms.com y en www.bundlehunt.com.

- **De login**: permiten acceder a los sitios en los que nos hemos registrado previamente. Suelen pedirnos un nombre de usuario y una contraseña.
- **De compra**: son aquellos que debemos completar cuando nos encontramos dentro de las llamadas **tiendas online**. Estos formularios nos pedirán no solo nuestros datos personales, sino también información sobre tarjeta de crédito o cuenta bancaria.
- **De encuesta**: estos formularios se utilizan en las páginas que brindan un servicio online. En general, su finalidad es ilustrar la experiencia del usuario en el sitio para mejorarla en el futuro.
- **De CV**: se encuentran en las páginas de bolsas de trabajo o de ciertas empresas y, como su nombre lo indica, nos permite colocar todos nuestros datos para postularnos a un puesto laboral. En algunos de ellos, podemos incluir una foto o el archivo con nuestro propio CV.

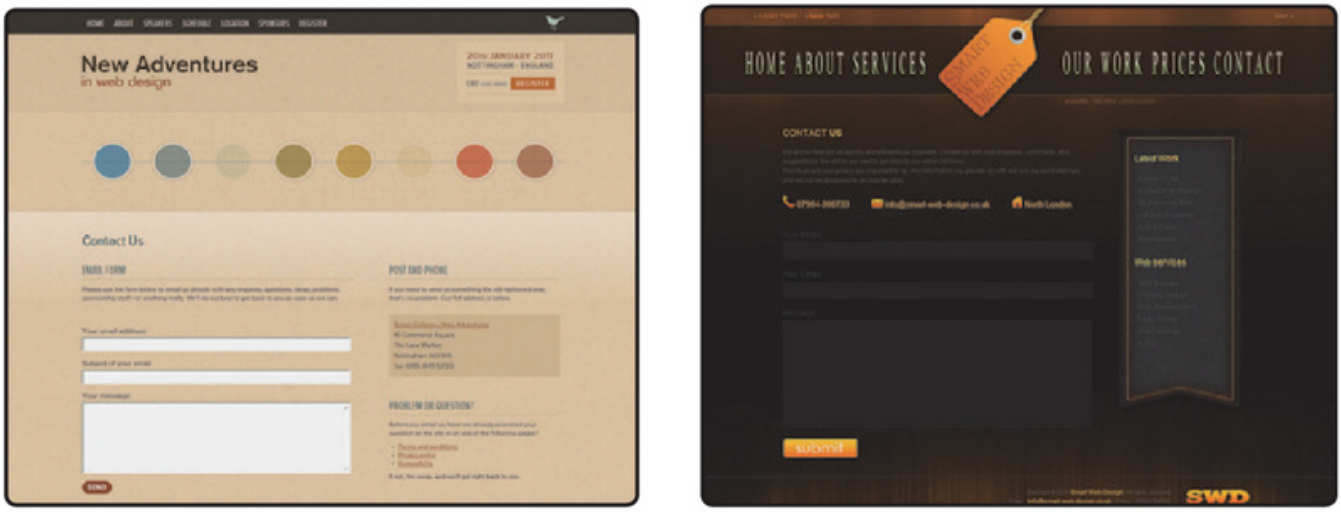


FIGURA 2. Los formularios de contacto son los más utilizados en cualquier tipo de sitio. Aquí vemos los de www.simplycreate.net y www.smart-web-design.co.uk.

CAPTCHA

Captcha es una prueba de seguridad destinada a diferenciar entre los humanos y las máquinas. Consiste en pedir la introducción de los caracteres que se muestran en una imagen distorsionada (imagen movida, caracteres tachados o incompletos, entre otras posibilidades).

Su finalidad más importante es evitar que los robots (pequeños programas que tienen por misión utilizar ciertos servicios para generar correo o cuentas spam) utilicen servicios de registro en cuentas de e-mail o redes sociales.

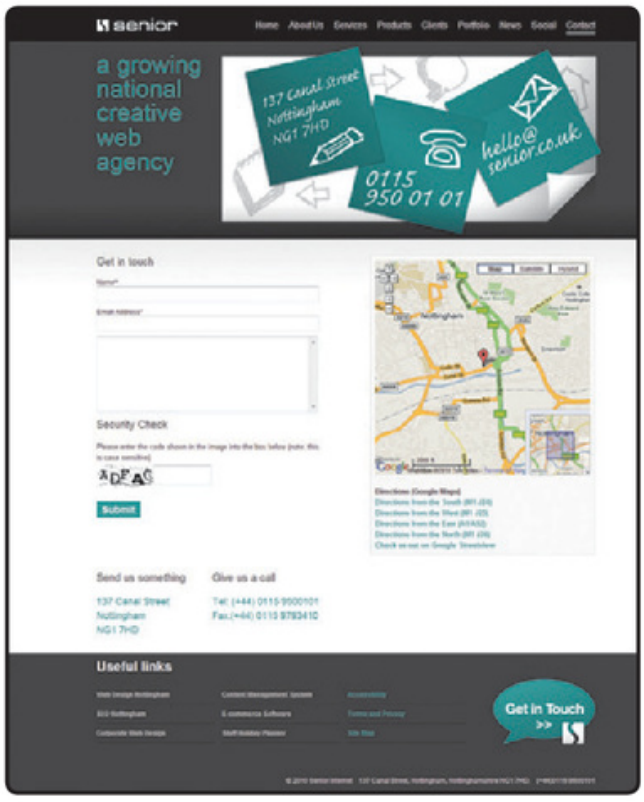


FIGURA 3. **Muchos formularios incluyen captcha, un mecanismo para evitar que sean completados por procesos automáticos. Encontramos un ejemplo en www.senior.co.uk.**

COMPOSICIÓN DE UN FORMULARIO

Aunque en las próximas páginas conoceremos en detalle cada uno de los elementos que integran un formulario, vamos a adelantar que, como sucede con otras páginas, este elemento está compuesto por una serie de etiquetas. La primera es `<form>`, que se utiliza como la estructura básica de todo formulario. Dentro de ella definimos los atributos, los métodos y la acción que llevará a cabo este recurso. Luego, está la etiqueta `<input>`, que puede tener varios usos diferentes:

- `Text`: una caja de texto simple para textos cortos.
- `Radio`: para seleccionar una única opción dentro de una casilla circular.
- `Checkbox`: para seleccionar una o más opciones dentro de una casilla cuadrada.
- `Submit`: propiedad que se asigna a las etiquetas `<input>` que hacen referencia a un botón (botones de envío, de borrado, etcétera).
- `File`: botón que permite buscar archivos en nuestro equipo y subirlos al sitio.
- `Reset`: botón para indicar al navegador que debe llevar los campos del formulario a su configuración predefinida (borra lo que se ha completado).

Encontramos también una etiqueta `<textarea>`, un campo de texto destinado a contener grandes cantidades de caracteres. La etiqueta `<select>`, por su parte, tiene por finalidad darnos a elegir la opción que queramos dentro de una lista, por lo cual tendrá en su interior la etiqueta `<option>`. Por otro lado, `<label>` se ocupa de mostrarnos la referencia del campo de texto y, finalmente, `<fieldset>` sirve para agrupar los distintos tipos de datos según cómo resulte más conveniente.

Esta contiene por defecto una etiqueta `<legend>`, que utilizaremos para colocar el título que le daremos al grupo de datos.

XFORM

Fue diseñado como una alternativa a los formularios tradicionales y representa la próxima generación de formularios HTML/xHTML. XForm posee muchas características avanzadas, ya que es un formulario que puede actuar en tiempo real cuando nos da a elegir las opciones: si hacemos clic en una lista desplegable, es probable que se estén pidiendo los datos de ella al servidor en el momento en el que lo hacemos. Este formulario puede especificar cómo serán validados los distintos datos a medida que se van ingresando, y no suele utilizar scripts separados.

En la actualidad, XForm no es soportado por la mayoría de los navegadores.

EL CLIENTE Y EL SERVIDOR

Más adelante profundizaremos en el funcionamiento y la programación de los formularios, pero en este punto es importante mencionar que estos pueden ser combinados con scripts para volverse dinámicos, es decir, funcionar.

Entre los scripts llamados **del lado del cliente**, el más estandarizado es el lenguaje **JavaScript**. Los que trabajan **del lado del servidor** son un poco más complejos, dado que permiten crear todo tipo de tareas para lograr un uso más minucioso del formulario desarrollado, que puede abarcar la autenticación de un inicio de sesión, el envío y el almacenamiento de información en una base de datos o la corrección ortográfica del envío de un e-mail.

Elementos de los formularios

Los formularios son la manera más utilizada para que el usuario de un sitio se comunique con el administrador o dueño. Allí, el visitante encuentra una serie de campos de texto que debe completar, y botones que se encargarán de que el mensaje sea procesado y enviado al administrador del sitio o a un programa que lo hará automáticamente.

Existen distintos modos de hacer que el formulario que tenemos en nuestro sitio web funcione. Por un lado, podemos usar la opción que brindan muchos servidores, que consiste en alojar el contenido del formulario completado en un archivo dentro del servidor. Por otro, podemos aprender a programar en **PHP** o **ASP** para que el contenido del formulario, una vez procesado, llegue a la dirección de e-mail que deseemos.

Todos los formularios tienen por estructura básica una etiqueta denominada `<form>`, que se cierra con `</form>`. Dentro de ella, vamos a colocar el contenido que deseemos. Esta etiqueta posee ciertos atributos que veremos a continuación:

- `action`: define la acción que debe realizar el formulario. Tenemos dos formas de usar este atributo. La primera es asociarlo a un archivo que se encargará de procesar el contenido del formulario. En este caso, el código es `<form action="mi archivo .php o .asp"></form>`. La segunda alternativa consiste en enviar el formulario sin procesar a una dirección de e-mail, de la siguiente

manera: `<form action=`mailto:midireccion@de-correo.com`></form>`.

- **method**: se ocupa de especificar cómo debe ser enviado el contenido del formulario. La primera manera de hacerlo es mediante el método denominado **GET**. Este utiliza la barra de direcciones del navegador para enviar el contenido, volviéndolo en cierto sentido público, porque se visualiza en dicha barra. Este sistema no soporta el envío de imágenes ni más de 500 caracteres. Está muy difundido en las páginas web dinámicas para la elaboración de consultas que se encuentran asociadas a una base de datos.

La segunda opción es el método denominado **POST**, el más utilizado para enviar el contenido de un formulario porque lo hace sin recurrir a la barra de navegación del browser, con lo cual oculta el contenido. Además, permite adjuntar y enviar imágenes, y la cantidad de caracteres es, por decirlo de alguna forma, ilimitada.

- **enctype**: se usa dependiendo de cómo procesemos el contenido del formulario. Si lo hacemos a través de un programa automáticamente, el atributo no se utiliza. En cambio, si el contenido será enviado a una dirección de correo electrónico, se coloca **text/plain** para que el mail llegue en texto plano. Por ejemplo: **`<form action=" mi archivo .php o .asp (o dirección de mail)" method="post" enctype="text/plain">`**.

A partir de ahora, conoceremos los distintos elementos que componen los formularios. Todos ellos, como cuadros de texto y botones, pueden denominarse **campos de texto** y **controles de formularios**, respectivamente.

FIGURA 4. Este sitio contra la pobreza infantil (www.noslajugamosolocambiamos.org) presenta, en su página principal, un formulario con el estilo completamente ajustado al diseño general.

ETIQUETA INPUT

Se utiliza para textos cortos, se asigna mediante **<input>** y podemos aplicarle dos atributos: **type** y **name**. La etiqueta se define de la siguiente manera: **<input type="text" name="nombre" value="">**. Con este ejemplo, creamos una caja de texto corto, cuyo contenido será **nombre**. Al atributo **name** es posible colocarle cualquier palabra que deseemos, aunque es aconsejable que esté relacionada con su contenido.

El atributo **name** es muy importante, debido a que es el nombre que utilizaremos para tomar la información del campo de texto cuando procesemos el formulario. No debemos utilizar caracteres especiales o latinos para nombrar los campos, y siempre tenemos que estar de acuerdo con el programador que trabajará en el sitio, ya que la aplicación usada para enviar el formulario debe tener, entre sus **variables**, el nombre que nosotros le otorgamos al atributo **name**.

También es fundamental definir el atributo **type**, ya que la etiqueta **<input>** posee múltiples usos y es este atributo el que determinará la función que cumplirá. Los valores que podemos darle a **type** son: **text**, **password**, **radio**, **checkbox**, **submit**, **reset**, **file**, **hidden**, **image** y **button**.

Un atributo menos importante, pero que también podemos definir, es **size**, que nos permitirá establecer el tamaño del campo de texto o etiqueta en un número de caracteres visibles. Si no definimos este atributo, cuando el campo de texto se muestre, lo hará en un tamaño preestablecido por el navegador que estemos empleando.

El atributo **maxlegth** nos permite definir la cantidad máxima de caracteres que se pueden colocar dentro del campo de texto. De esta forma, si le asignamos cierto valor, una vez que sea alcanzado, el navegador impedirá seguir escribiendo.

FIGURA 5. Los campos del formulario de esta casa de té (www.tsaateashop.com) cambian su color a verde cuando los estamos editando.

Aunque es un error común confundir **maxlegth** con **size**, debemos recordar que **size** simplemente asigna un tamaño a la etiqueta, mientras que **maxlegth** restringe la cantidad de caracteres que podemos ingresar.

El atributo size asigna un tamaño a la etiqueta, y maxlegth limita los caracteres que se pueden escribir

El atributo **value** sirve para otorgarle un valor inicial al campo si no queremos que aparezca vacío. En algunas ocasiones, se lo utiliza para detallar, mediante un ejemplo, la forma en la que debe completarse el campo de texto, como vemos a continuación: **<input type="text" name="nombre" value="Escriba su nombre" />**. El atributo **value** siempre funciona dentro de una etiqueta **<form>**, nunca fuera de ella.

TEXTO OCULTO

Otro caso en el que usamos el elemento **<input>** es cuando deseamos ocultar el texto que queremos escribir por medio de asteriscos o círculos. Para esto, la única diferencia estará en el atributo **type**, en donde debe figurar **password**. Este atributo es muy utilizado cuando queremos generar campos para claves o logins. Veamos un ejemplo: **<input type="password" name="contrasena" value="" />**.

ETIQUETA RADIO O BOTÓN RADIO

El tipo **radio** nos permite hacer elegir una única opción de una lista. La etiqueta empleada es **<input>**, pero con su atributo **type** establecido como **radio**. Al usar este elemento, debemos generar los textos y los saltos de línea en el código HTML. Veamos un ejemplo:

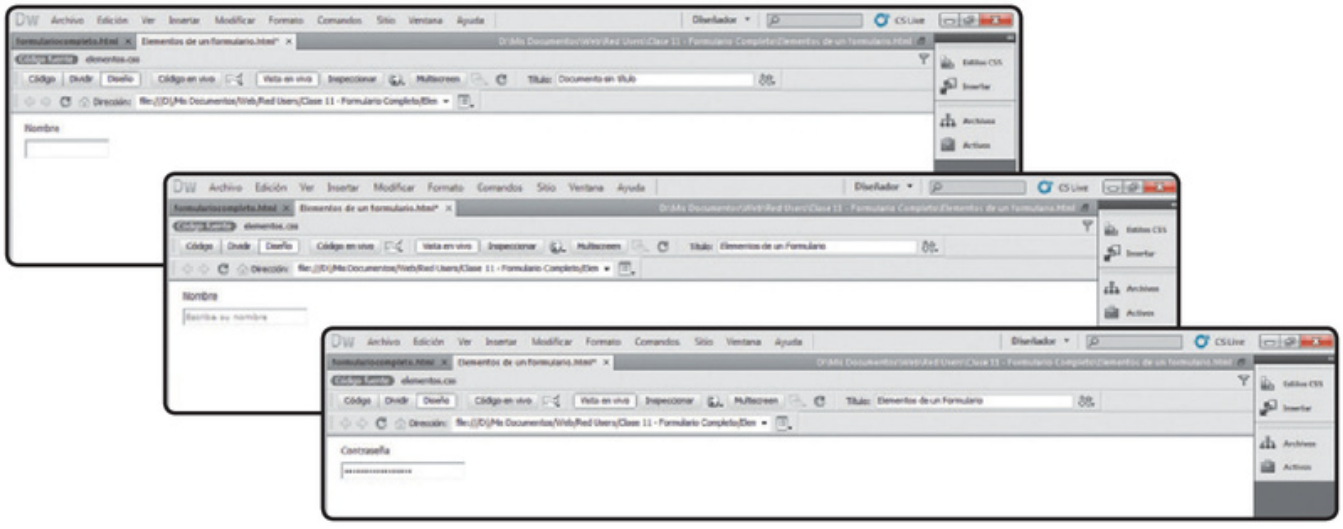


FIGURA 6. Aquí podemos encontrar campos de texto utilizados de distintas maneras: tal como aparece por defecto, con texto de ayuda y configurado para ocultar los caracteres.

FIGURA 7. El formulario que encontramos en www.quinttodigital.com es simple pero aparece en todas las páginas, ya que está incluido en el footer.

```
<input type="radio" name="opciones"
  value="1" />Opción 1
<br />
<input type="radio" name=" opciones"
  value="2" />Opción 2
<br />
<input type="radio" name="opciones"
  value="3" />Opción 3
<br />
<input type="radio" name=" opciones"
  value="4" />Opción 4
<br />
```

Como vemos en nuestro ejemplo, el atributo **name** es el mismo en todos los casos, pero **value** es diferente. Por lo tanto, cuando recibamos el e-mail sobre la opción seleccionada, veremos lo siguiente: **Opciones = 2**.

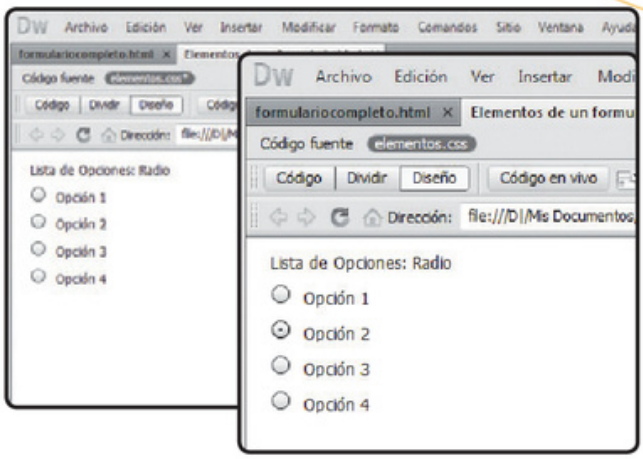


FIGURA 8. En este caso, podemos ver una lista de opciones sin selección predeterminada, a la izquierda; y la segunda opción elegida por defecto, a la derecha.

Es posible **preseleccionar** una opción mediante el atributo **checked**. Para lograr la selección automática de la segunda opción, el código es **<input type="radio" name="lista" value="2" checked />Opción 2**.

CHECKBOX O CAJA DE SELECCIÓN O VALIDACIÓN

El tipo **checkbox** es más conocido como caja de validación, que puede ser activada o desactivada. A diferencia de los botones radio, en este caso es posible seleccionar más de una opción al mismo tiempo. El código para usarla es: **<input type="checkbox" name="validacion" />Caja de Validación**.

Al igual que con **radio**, en este caso es posible seleccionar una opción de manera predefinida mediante el atributo **checked**. La información seleccionada puede ser indicada en nuestro correo como **on** u **off**, según corresponda.

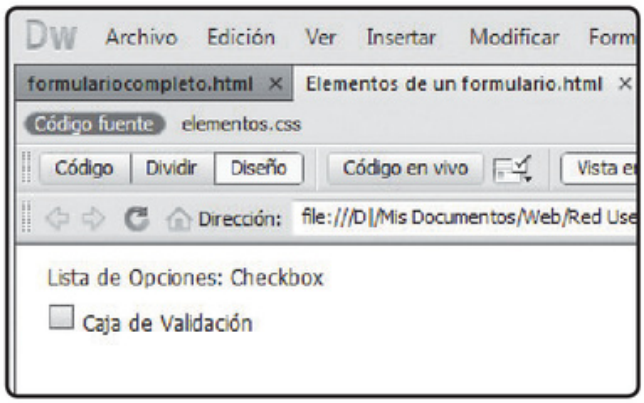


FIGURA 9. **Aunque podemos usar una sola caja de validación, es frecuente utilizar varias agrupadas, para que el usuario elija distintas opciones.**

BOTÓN SUBMIT O DE ENVÍO DE FORMULARIO

El tipo **submit** se utiliza para enviar el formulario una vez que esté completo. Se diferencia en estructura solo en el atributo **type**, como vemos a continuación: **<input type="submit" value="Enviar" />**. Con **value** aplicamos al botón **submit** el mensaje que queremos que se muestre.

BOTÓN RESET O DE BORRADO DE FORMULARIO

El tipo **reset** se emplea para borrar el contenido del formulario si nos equivocamos al completarlo, ya que se vacían todos los campos y se los vuelve a su inicio. La forma de uso es: **<input type="reset" value="Borrar" />**, y es de empleo optativo.

BOTÓN FILE O DE ARCHIVOS ADJUNTOS

El atributo **type** con valor **file** se usa para permitir el envío de archivos mediante el formulario de contacto. Su código es: **<input type="file" name="adjunto" size="30" />**. Es importante que en este botón le demos un valor al atributo **size**, para establecer el tamaño de la caja en la que se mostrará el texto con la ruta al archivo.

CAMPO HIDDEN O DATOS OCULTOS

El tipo **hidden** no es muy utilizado en sitios HTML, pero sí en aquellos que necesitan cierto tráfico con el servidor. Permite enviar datos adicionales para acelerar el proceso; por ejemplo, los formularios muchas veces se envían a dos cuentas de e-mail. Es un campo oculto que el usuario no podrá ver, a menos que observe el código fuente. Su uso es: **<input type=hidden name="sitio" value="www.tuweb.com">**.

FIGURA 10. **En el sitio de Mooze (www.moozedesign.com), el formulario está modificado para lograr que se integre completamente con el diseño.**

BOTÓN IMAGE

El tipo **image** sirve para colocar una imagen en reemplazo del botón **submit**. El código es: **<input type="image" src="imagenes/typeimage.jpg" alt="Submit " />**, donde podemos observar que aparecen dos atributos no utilizados hasta ahora. El primero es **src**, que permite seleccionar el archivo que emplearemos para que se muestre en reemplazo de **submit**. El segundo es **alt**, que se usa solo cuando el navegador no puede mostrar la imagen.

FIGURA 11. **En este formulario (www.mustardmonkey.co.uk) podemos ver varios elementos, incluyendo un botón con imagen para efectuar el registro.**

TYPE BUTTON O BOTÓN COMÚN

El atributo **type** con valor **button** es la definición de un botón normal. No tiene gran utilidad, aunque se define en algunos formularios si se quiere utilizar JavaScript para ocultar o mostrar determinados contenidos. Su uso es: **<input type=button value=" botón">**.

Hacer que los botones sean reemplazados por imágenes nos permite convertirlos en parte del diseño

ETIQUETA PARA TEXTO LARGO

La etiqueta **textarea** sirve para que el usuario pueda escribir más de una línea en el campo de texto. Se inserta como **<textarea>** y debe cerrarse al finalizar.

Suele utilizarse para que los usuarios envíen un mensaje o un comentario, o cuando deben completar alguna opción que necesita un desarrollo un poco más complejo que una línea.

No se emplea para textos cortos, y posee los atributos **name**, **rows** y **cols**. El atributo **rows** define el número de líneas que se podrán escribir en el campo de texto, mientras que **cols** define el número de columnas en el que este puede dividirse. Un ejemplo de su uso es: **<textarea name="comentario" rows="10" cols="20"></textarea>**.

En la etiqueta **textarea** no es posible definir el atributo **value**, por lo que si queremos dejar algún comentario para guiar al usuario, tendremos que hacerlo entre la etiqueta de apertura y la de cierre, como vemos a continuación: **<textarea name="comentario" rows="10" cols="20">Escriba su comentario…</textarea>**.

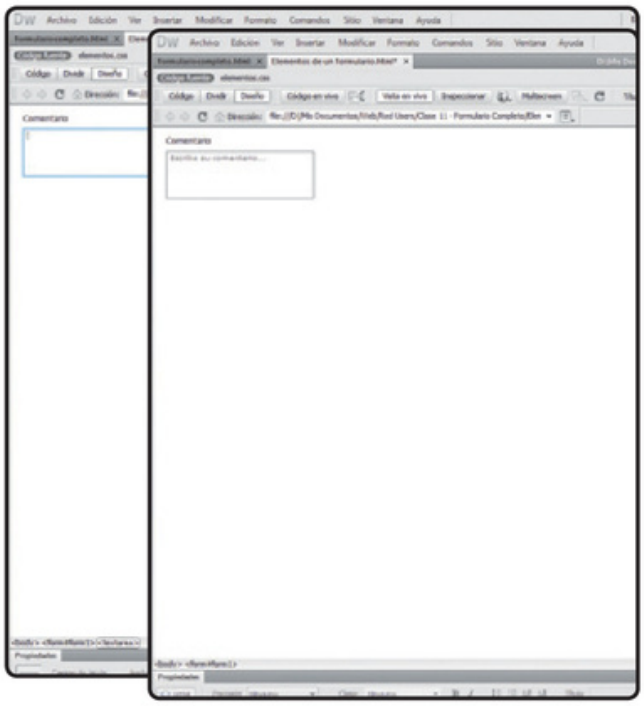


FIGURA 12. La etiqueta textarea, tal como aparece por defecto y configurada para que muestre un mensaje al usuario.

LISTA DE OPCIONES

La etiqueta **select** también es conocida como **lista de opciones**, ya que permite elegir entre una serie de alternativas predefinidas por el diseñador del formulario. Está compuesta por una etiqueta **<select>** y su correspondiente cierre. Podemos establecer su nombre mediante el atributo **name**, determinando las opciones disponibles por una etiqueta **<option>** y su respectivo cierre. Veamos el código de ejemplo:

```
<select name="opciones">
<option>Opción 1</option>
<option>Opción 2</option>
<option>Opción 3</option>
<option>Opción 4</option>
</select>
```

Las listas son muy usadas para el lugar de residencia y el año de nacimiento en los formularios de registro

FIGURA 13. Una vez más, en www.heartwoodhall.com encontramos un formulario incorporado al diseño, tanto por su ubicación como por su estilo.

La etiqueta **select** puede tener tres atributos más. Por un lado, tenemos **size**, que se utiliza para definir el tamaño de la caja que posee la lista de opciones. Esto se traduce en la cantidad de opciones que se mostrarán a la vez, aunque de esta forma perderemos el campo desplegable y aparecerá, en el lado derecho de la etiqueta **select**, una barra de desplazamiento.

Por otro lado, podemos definir **multiple**, para que sea posible seleccionar más de un elemento de la lista, utilizando la tecla **<SHIFT>** o **<CTRL>**, de

acuerdo con la versión de Windows que tengamos. Aunque no es recomendable utilizar este atributo, porque puede ser desconocido para el usuario que esté visitando el sitio, veamos un ejemplo:

```
<select name="opciones" size="3"
   multiple>
<option>Opción 1</option>
<option>Opción 2</option>
<option>Opción 3</option>
<option>Opción 4</option>
</select>
```

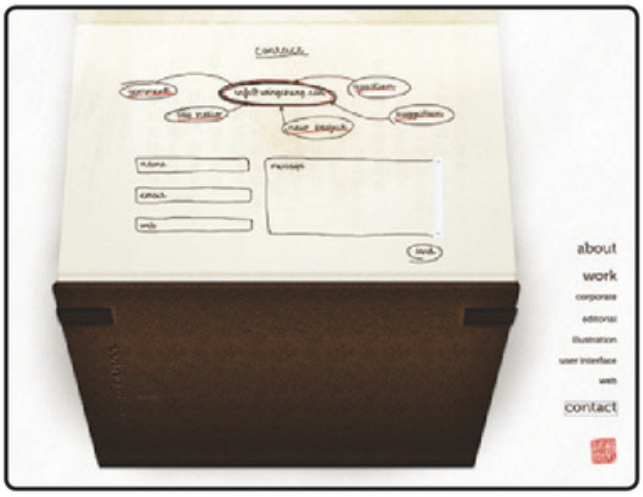


FIGURA 14. **En este caso, el formulario respeta perfectamente el diseño del sitio (www.wingcheng.com).**

El último atributo es **selected**, cuya función, como su nombre lo indica, es hacer que la opción elegida aparezca seleccionada por defecto. Su uso es **<option selected>Opción 2</option>**.

Por otra parte, el atributo **value** nos permite asociar una letra o un número al valor que elijamos de la lista. Es una función muy útil para aquellos formularios que

deben ser procesados con posterioridad. Mediante el código **<option value="1">Opción 1</option>**, cuando el formulario sea enviado por correo electrónico, se mostrará **Opción = 1**.

BLOQUES DE ELEMENTOS

Los bloques de elementos nos dan la posibilidad de agrupar los distintos elementos de los formularios. Ya no necesitamos aplicar etiquetas externas al formulario para hacerlo (en una época, se utilizaban tablas, y luego se empleaban etiquetas div para agrupar en bloque los elementos; todas ellas, externas, que no pertenecen al form).

ETIQUETAS FIELDSET Y LEGEND

La etiqueta **<fieldset>** sirve para agrupar los distintos elementos de un formulario e, incluso, agregarles un título mediante la etiqueta **<legend>**. Esta última etiqueta permite utilizar el atributo **align** con las propiedades **center**, **top**, **right**, **bottom** y **left**, según como queramos mostrar el título.
Por defecto, **<fieldset>** genera un recuadro, y **<legend>** se coloca en la parte superior izquierda de este, cortando la línea continua que lo conforma.

```
<fieldset>
  <legend>Bloque de elementos</legend>
  Escribe lo que quieras
  <input type="text" size="20">
</fieldset>
```

FIGURA 15. **Aquí podemos observar claramente cómo están agrupados varios elementos mediante esta etiqueta.**

ETIQUETA LABEL

Permite colocar texto en su interior, que queremos mostrarle al usuario. Es muy usada para presentar el texto que corresponde a la etiqueta **<input>**. Por ejemplo, si el formulario tiene un campo llamado **nombre**, este texto estaría en el interior de **<label>**.

```
<label>Ingrese su E-Mail:
  <input type="text" name="email">
</label>
```

FIGURA 16. **Si observamos el código, veremos que el texto que hace referencia a aquello que se debe completar en el campo input está dentro de la etiqueta** <label>.

Introducción **a PHP**

PHP es un acrónimo recursivo que significa **PHP Hypertext Pre-processor**. Su creación se remonta al año 1995, y como todo lenguaje, pasó por diferentes versiones, algunas más populares que otras. La más reciente es la 5.3.3 (lanzada en julio de 2010), aunque una de las más usadas fue la 4.4.8. Si bien la diferencia entre ellas se concentra en funciones avanzadas que no veremos aquí, en nuestros ejemplos asumimos que el trabajo se realizará en servidores con la versión 5.2.6 o superior.

PHP es uno de los lenguajes más difundidos porque, prácticamente, todos los sistemas operativos para servidores lo soportan, aunque entre ellos el de mayor uso y robustez es Linux.

Podemos incluir el código PHP dentro del HTML de la página en la que estemos trabajando. Para indicar que iniciaremos una secuencia de código PHP, usamos la etiqueta **<?php** y, para señalar su cierre, la **?>**. Cuando incorporamos código PHP al HTML, es fundamental almacenar el archivo con extensión PHP, ya que esta es la manera en que el servidor reconocerá el tipo de página y podrá interpretar el código.

Cuando incluimos código PHP, los archivos no pueden tener extensión HTML, sino que deben ser PHP

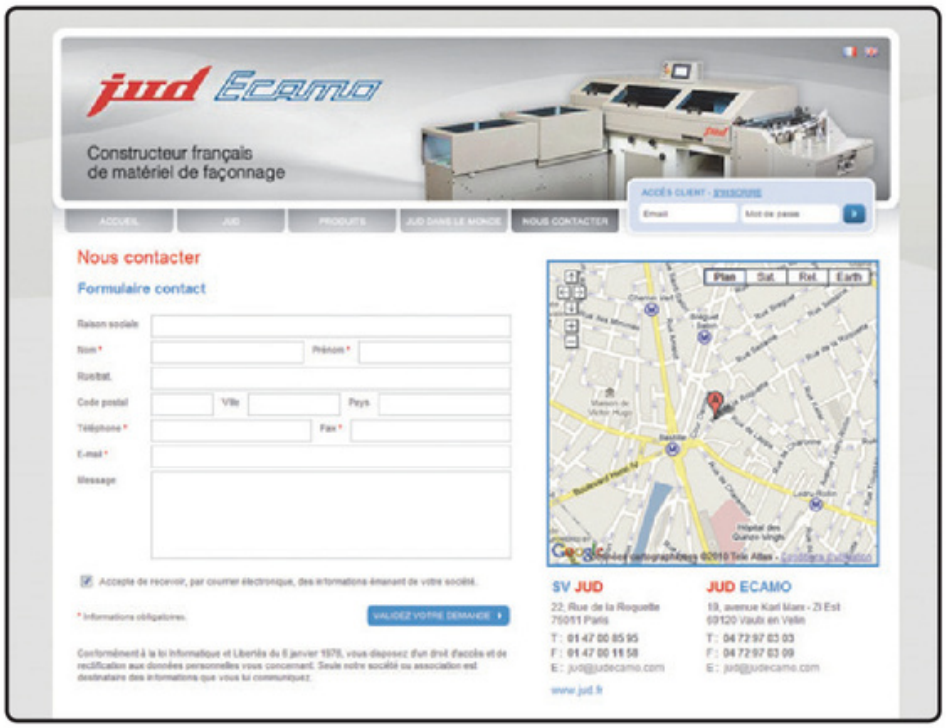


FIGURA 17. El formulario de contacto que encontramos en www.jud-ecamo.com está realizado en PHP.

LENGUAJES DE SERVIDOR

Estos lenguajes son los que se interpretan y ejecutan en el lugar donde está alojado el sitio. Cada vez que un usuario accede a una página, el servidor ejecuta las líneas de código antes de que el contenido sea mostrado en el navegador de Internet del visitante.

El código de los lenguajes de servidor es inaccesible para el usuario. Como PHP es un lenguaje de este tipo, el usuario no podrá ver el código, sino que su navegador lo recibirá ya interpretado.

LENGUAJES DE CLIENTE

Son aquellos que se interpretan y ejecutan en la computadora o dispositivo del usuario. En este caso, el encargado de interpretar y ejecutar los comandos es el navegador de Internet que esté utilizando quien visita el sitio. El lenguaje de cliente más utilizado es **JavaScript**, actualmente masificado por su **framework** más conocido: **jQuery**.

PROCESO DE EJECUCIÓN EN PHP

Gracias a PHP, podremos mostrar contenido

FRAMEWORK

Los frameworks simplifican el desarrollo de una aplicación web mediante la automatización de funciones que se usan con frecuencia. Gracias a ellos, las operaciones complejas quedan encapsuladas para que, luego, nosotros apliquemos operaciones sencillas y fáciles de entender.

dinámico en nuestro sitio web; es decir, aquel que está almacenado en una base de datos o en variables, que a través de la interpretación de PHP, se traduce en HTML en el navegador. Este proceso es relativamente fácil de comprender, y podemos resumirlo en estos pasos:

1. El navegador de Internet del usuario solicita una página PHP determinada.
2. El servidor localiza el documento pedido y, a través del **intérprete de PHP**, ejecuta el código que esté en esa página.
3. Una vez que se ejecuta el código PHP, se genera el resultado como código HTML.
4. El resultado en HTML es enviado al usuario que solicitó la página y se muestra en su navegador de Internet.

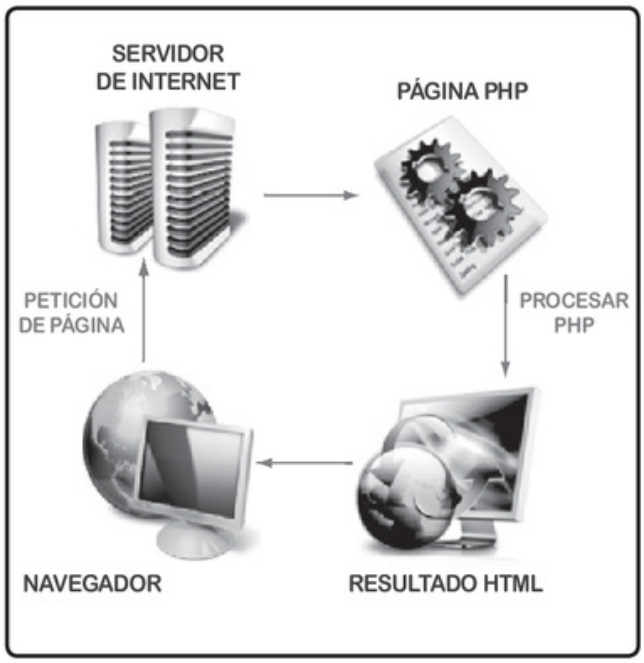


FIGURA 18. En este esquema, podemos ver cómo funciona el proceso de ejecución de un archivo PHP alojado en un servidor.

A diferencia HTML, una página PHP no puede verse de manera local

Es importante entender que una página PHP no puede visualizarse de manera local como sí sucede con las HTML; es decir, no podemos abrirlas directamente desde el navegador de Internet. Para ver una página PHP de forma correcta necesitamos tener instalado un servidor en nuestra computadora o conectarnos a uno externo.

FIGURA 19. El catálogo de productos de la tienda Hemline (www.shophemline.com) está desarrollado en PHP, como podemos ver en la barra de direcciones del navegador.

VARIABLES

Una variable es una posición en la memoria del servidor, reservada para asignar cualquier valor o dato.

A diferencia de otros lenguajes de programación, en PHP no es estrictamente necesario declarar las variables, ya que solo debemos anteponer el símbolo $ al nombre de aquella que queremos utilizar.

Las variables en PHP son **case sensitive**, es decir, el lenguaje diferencia entre las minúsculas y las mayúsculas. Por ejemplo, **`$miVariable=1`** no es lo mismo que **`$mivariable=1`**, y PHP las interpretará como dos variables diferentes.

Una variable puede almacenar distintos tipos de datos. En PHP, las opciones son las siguientes:

- **`Integer`**: números enteros positivos y negativos.
- **`Double`**: números decimales positivos y negativos.
- **`String`**: cadenas de texto.
- **`Boolean`**: valores true o false.
- **`Array`**: tipo de variable de múltiples posiciones.
- **`Object`**: tipo especial de dato complejo.

VARIABLES DEFINIDAS

En PHP existen diferentes tipos de variables predefinidas por el lenguaje que utilizaremos para manejar la información en las distintas páginas de un sitio. Las dos que más utilizaremos son **`$_GET`** y **`$_POST`**, y las veremos con mayor detalle más adelante. Mientras tanto, a continuación veamos nuestro primer ejemplo de script en PHP, dentro del código HTML:

```
<html>
<head>
<title>Mi primer PHP</title>
</head>
<body>
<?php
$miVariable="Hola Mundo!";
```

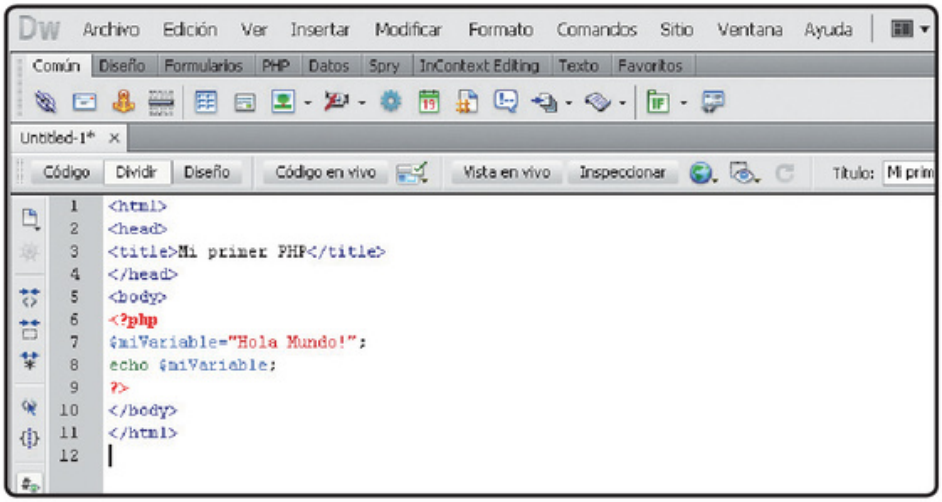


FIGURA 20. Adobe Dreamweaver CS5 colorea la sintaxis según el lenguaje que usemos y cada comando dentro de él. Esto es muy útil y nos ayuda en la lectura del código.

```
echo $miVariable;
?>
</body>
</html>
```

En el ejemplo anterior vemos que la palabra **miVariable** comienza con el símbolo **$**. Como mencionamos anteriormente, de esta forma le estamos indicando a PHP que se trata de una variable. También observamos que al final de cada línea hay un punto y coma **;**, para indicar a PHP que allí finaliza una línea de código. Por último, la función **echo** sirve para mostrar en pantalla el contenido de una variable o una cadena de texto.

ESTRUCTURAS DE CONTROL

La estructura de control que utilizaremos con mayor frecuencia es **if** (en inglés, el condicional si). Esta analiza una condición y, en caso de que se cumpla, ejecuta un script; de lo contrario, ejecuta otro. Por ejemplo, el código que se muestra a continuación se traduce de la siguiente manera: si la variable "edad" es mayor o igual al valor 18, mostrar la frase "Eres mayor de edad"; y si la variable edad no cumple con la condición mencionada, mostrar la frase "Eres menor de edad".

```
<?php
if($edad>=18){
        echo "Eres mayor de edad";
}else{
        echo "Eres menor de edad";
}
?>
```

INCLUDES

En muchas ocasiones, tendremos la necesidad de incluir código desde un archivo externo, que puede ser PHP o HTML. Esto sucede, por ejemplo, cuando precisamos incluir en todas las páginas de un sitio

FIGURA 21. PHP también se puede utilizar para crear formularios de registro, como el que encontramos en www.jumpinteractivo.com.

un archivo HTML que contiene un menú de navegación práctico.

```
<body>
<?php
include("menu.html");
?>
</body>
```

La ventaja de utilizar **include** es que, si necesitamos realizar un cambio sobre el archivo agregado, no tendremos que hacerlo más de una vez. Siguiendo el ejemplo, si queremos agregar una opción al menú, no hará falta modificar cada una de las páginas HTML, dado que bastará con hacerlo en la página **menu.html**.

VECTORES

Aunque el empleo de vectores es un tema complejo, debemos mencionar que un array (por vector, en inglés) es una variable que contiene múltiples posiciones para almacenar datos. Para acceder a los datos guardados en un vector, existen dos formas. La primera consiste en indicar el índice del vector; por ejemplo, **$miVector0;**. La segunda es indicar la clave (o key) del valor al que queremos acceder, por ejemplo, **$miVector"dato";**.

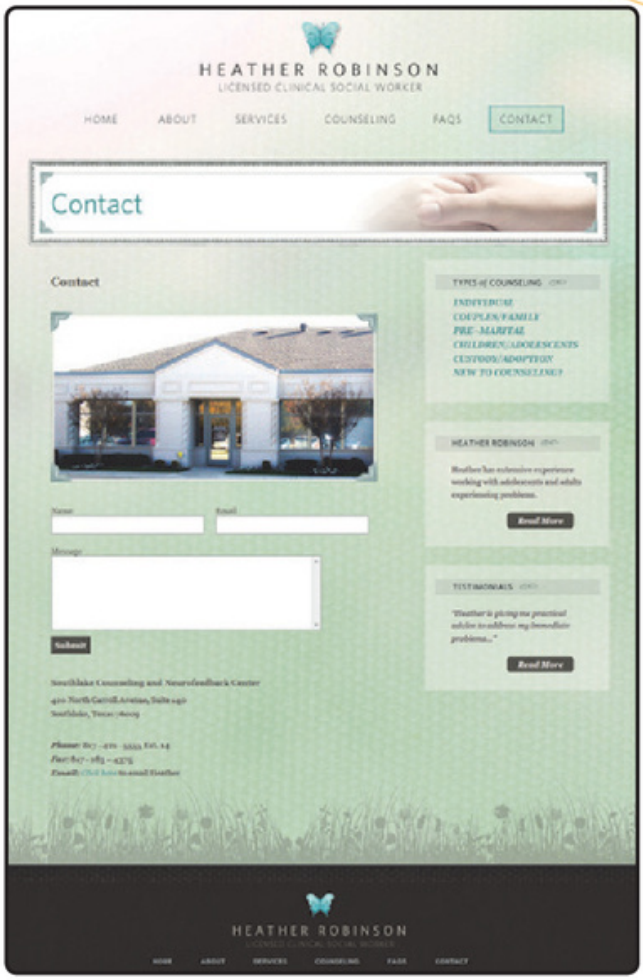


FIGURA 22. **Al hacer clic en el botón** `Submit` **de la página de contacto de www.hrobinsoncounseling.com, se envía la información a** `Mailer.php`.

Las posiciones de los vectores siempre comienzan en 0 (cero), lo que significa que uno que contenga diez posiciones, irá de la posición cero a la nueve.

FUNCIONES

Las herramientas más importantes de cualquier lenguaje de programación son las funciones. Recordemos que en PHP, podemos definir nuestras propias funciones y, también, utilizar muchas que ya están definidas por el lenguaje.

Una que usaremos muy a menudo es **mail()**, que, como su nombre lo indica, sirve para enviar contenido por correo electrónico. Se define de la siguiente manera: **mail ($destinatario,$asunto,$mensaje);**.

VARIABLES ENTRE PÁGINAS

Muchas veces, nos encontraremos ante la necesidad de transmitir variables de una página a otra. Para hacerlo, existen los métodos **POST** y **GET**. Esta transmisión de una variable entre páginas se conoce, en la jerga, como "pasar una variable".

Para pasar una variable por **GET**, debemos indicarla en la URL de la página a la que estamos accediendo. La primera variable va precedida del carácter **?** (símbolo de interrogación), y a partir de la segunda, se emplea **&** (ampersand). Por ejemplo: **miPagina.php?nombre=Juan&apellido=Perez**. En este caso, el archivo **miPagina.php** recibirá por **GET** la variable **nombre**, cuyo valor es **Juan**; y **apellido**, cuyo valor es **Perez**.

Luego, dentro del código de **miPagina.php**, por medio de PHP interpretamos esas variables que estamos pasando, utilizando la variable definida **$_GET**, que actúa como vector. Vemos entonces que **$_GET** es, en realidad, un vector cuyas claves o keys son las diferentes variables que le hemos pasado a la página.

```
<?php
$nom=$_GET"nombre";
$ape=$_GET"apellido";
?>
```

El uso de **POST** es similar, solo que para pasar variables no utilizamos la URL de la página de destino sino un formulario. Luego, interpretamos esas variables con la variable definida **$_POST**. Veremos que **$_POST** también es un vector cuyas claves son las diferentes variables que le hemos pasado a la página a través de un formulario.

```
<?php
$nom=$_POST"nombre";
$ape=$_POST"apellido";
?>
```

Recordemos que para pasar una variable por el método GET, debemos indicarla en la URL de la página a la que estamos accediendo

FIGURA 23. El sitio www.namuntu.com utiliza un formulario para hacer reservas de alquiler de los equipos. Los datos le llegan al webmaster por e-mail.

El uso de **POST** es más seguro y discreto, ideal para la transmisión de datos, por ejemplo, de un formulario de contacto. En cambio, **GET** se usa para pasar pocas variables y de información poco relevante para el usuario.

CONCATENAR

El carácter **.** (punto) se utiliza para concatenar cadenas de texto en PHP. Por ejemplo, con el siguiente código obtenemos como resultado el texto **20/10**:

```
<?
$dia=20;
$mes=10;
echo $dia."/".$mes; //
?>
```

Hasta aquí llegamos con los primeros pasos en PHP, un lenguaje extenso y con infinitas posibilidades. Para estudiar y conocer más sobre él, es aconsejable visitar su sitio, **www.php.net**, donde encontraremos novedades y un manual completo con todas sus funciones.

Se trata de un manual que contiene principalmente una completa referencia de cada una de las funciones del lenguaje PHP, las cuales debemos conocer para enfrentar el desarrollo de sitios web con esta herramienta. También contiene una referencia del lenguaje y sus posibilidades de uso, explicaciones de algunas de las características importantes de PHP e información suplementaria que puede servirnos como material de consulta.

Es importante mencionar que este manual puede descargarse en diferentes formatos, para lo cual visitaremos la dirección web **www.php.net/download-docs.php**. Allí seleccionamos el formato que más nos convenga para descargar el contenido a la computadora. Por otra parte, también es posible navegar a través de sus secciones en la misma página web.

Multiple **choice**

1 **¿Qué elemento nos permite transmitir información entre el cliente y el servidor?**

a- Formularios.
b- E-mail.
c- Listas.
d- Protocolo.

2 **Según su complejidad ¿Qué tipos de formularios encontramos?**

a- Estáticos y dinámicos.
b- Simples y continuos.
c- Básicos y complejos.
d- Básicos y avanzados.

3 **¿Qué tipo de formulario es usado en las tiendas online?**

a- De compras.
b- De login.
c- De registro.
d- De pago.

4 **¿Qué es captcha?**

a- Una imagen.
b- Una firma electrónica.
c- Una prueba de seguridad.
d- Una selección de recursos.

5 **¿Cómo se llama la caja para textos cortos?**

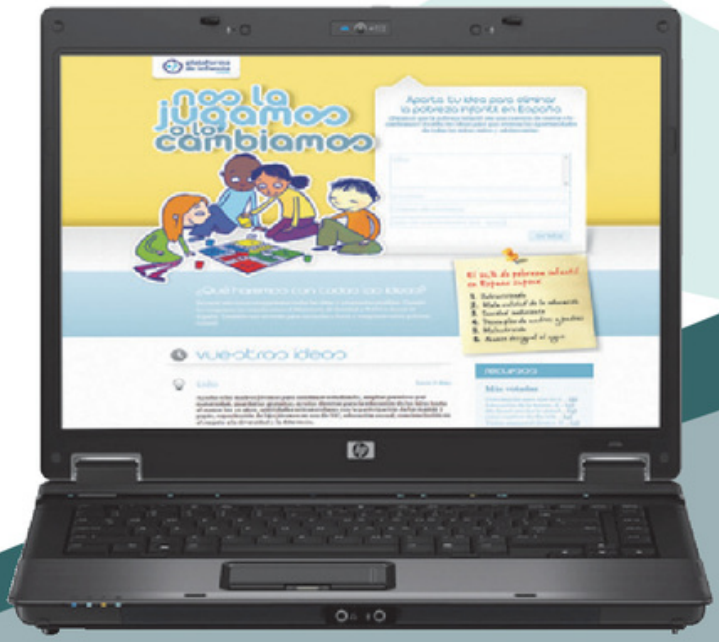
a- Label.
b- Text.
c- Radio.
d- Checkbox.

6 **¿Cómo se denomina la alternativa a los formularios tradicionales que representa la próxima generación de formularios HTML/ xHTML?**

a- xHTML.
b- XForm.
c- FormX.
d- Formularios dinámicos.

Respuestas: 1-a, 2-d, 3-a, 4-c, 5-b, 6-b.

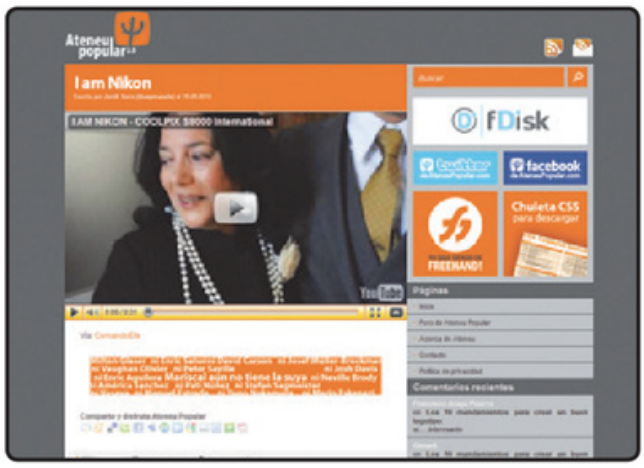
Servicios **al lector**



Con el índice temático podremos realizar búsquedas específicas a partir de los términos clave del libro.

RedUSERS.com

Índice **temático**



F

G

H

I

L

M

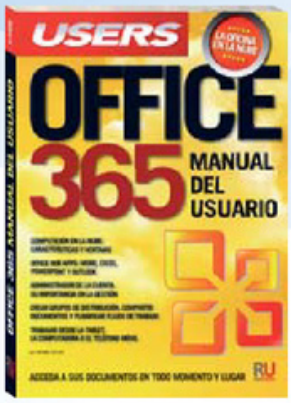
N

P

S

T

X

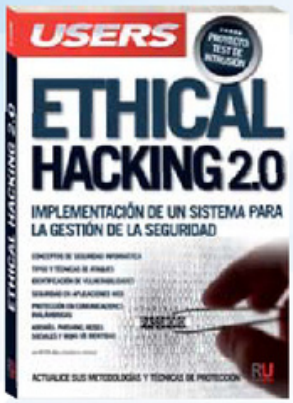


Una obra ideal para aprender todas las ventajas y servicios integrados que ofrece Office 365 para optimizar nuestro trabajo.

→ 320 páginas / ISBN 978-987-1857-65-4

Esta obra presenta las mejores aplicaciones y servicios en línea para aprovechar al máximo su PC y dispositivos multimedia.

→ 320 páginas / ISBN 978-987-1857-61-6



Esta obra va dirigida a todos aquellos que quieran conocer o profundizar sobre las técnicas y herramientas de los hackers.

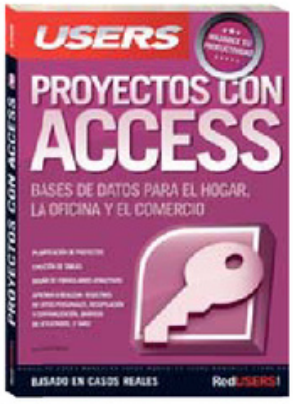
→ 320 páginas / ISBN 978-987-1857-63-0

Este libro se dirige a fotógrafos amateurs, aficionados y a todos aquellos que quieran perfeccionarse en la fotografía digital.

→ 320 páginas / ISBN 978-987-1857-48-7

En este libro encontraremos una completa guía aplicada a la instalación y configuración de redes pequeñas y medianas.

→ 320 páginas / ISBN 978-987-1857-46-3



Esta obra está dirigida a todos aquellos que buscan ampliar sus conocimientos sobre Access mediante la práctica cotidiana.

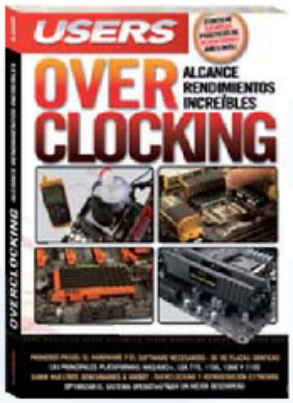
→ 320 páginas / ISBN 978-987-1857-45-6

Este libro nos introduce en el apasionante mundo del diseño y desarrollo web con Flash y AS3.

→ 320 páginas / ISBN 978-987-1857-40-1

Esta obra presenta un completo recorrido a través de los principales conceptos sobre las TICs y su aplicación en la actividad diaria.

→ 320 páginas / ISBN 978-987-1857-41-8



Este libro está dirigido tanto a los que se inician con el overclocking, como a aquellos que buscan ampliar sus experiencias.

→ 320 páginas / ISBN 978-987-1857-30-2

CONÉCTESE CON LOS MEJORES LIBROS DE COMPUTACIÓN

Esta obra nos presenta a las dos herramientas más poderosas para la creación de sitios web profesionales de la actualidad. A través de procedimientos paso a paso, nos muestra cómo armar un sitio real con Dreamweaver y Fireworks sin necesidad de conocimientos previos.

» DISEÑO / DESARROLLO
» 320 PÁGINAS
» ISBN 978-987-663-022-1

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

MÁS INFORMACIÓN / CONTÁCTENOS

usershop.redusers.com +54 (011) 4110-8700 usershop@redusers.com

*SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // **VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

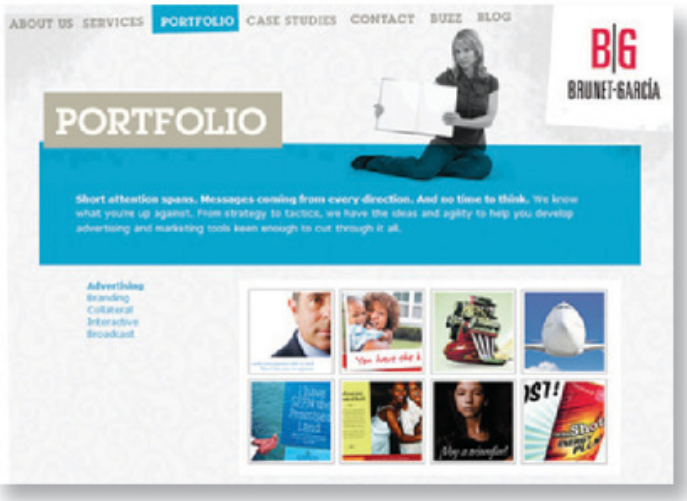
Diseño web con HTML & CSS

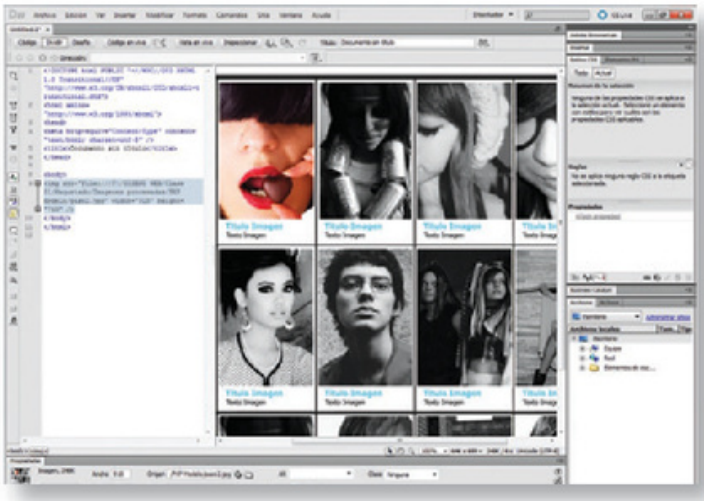
¡Últimas tendencias en diseño!

A la hora de diseñar y desarrollar en Internet, los lenguajes HTML y CSS se posicionaron como la piedra fundamental para el armado de sitios web. Esta obra está dirigida a diseñadores gráficos, programadores y aficionados que quieran formar parte del apasionante universo del diseño y desarrollo web, aplicando las tecnologías de vanguardia.

Dentro del libro encontrará:

¿Qué tipos de sitios existen? / Principios básicos sobre el Diseño Web / La salida laboral / Etiquetas HTML en Dreamweaver / El background de nuestro sitio / Posicionamiento de elementos en CSS / Texto: codificación de caracteres y aplicación de estilos / Imagen: tipos y formatos / Enlaces: estilos básicos y avanzados / Listas: ejemplos y recomendaciones / Reglas para hacer CSS accesible / ¿Cómo hacer funcionar nuestros formularios?





Profesor en línea

Ante cualquier consulta técnica relacionada con el libro, puede contactarse con nuestros expertos: profesor@redusers.com

Si desea más información sobre nuestros productos:
Servicio de Atención al Lector
usershop@redusers.com

RU RedUSERS

ISBN 978-987-1857-66-1

9 789871 857661 >