



UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Córdoba

Secretaría de Extensión Universitaria

Área de Educación a Distancia

Coordinador General de Educación a Distancia:

Magíster Leandro D. Torres

Curso:

DIPLOMATURA EN DESARROLLO WEB EV 15030

Módulo:

“Diseño y Desarrollo de Páginas Estáticas”

Autor:

Ing. Ricardo Martín Espeche

Datos del Autor

Nombre: Ricardo Martín Espeche

Mail: ricardo_espeche@hotmail.com

Título Universitario: Ing. en Sistemas otorgado por Instituto Universitario Aeronáutico de Córdoba.

Antecedentes profesionales:

1999 - Administrador del DEPARTAMENTO DE DESARROLLO WEB, a cargo del desarrollo y administración del sitio Web del Instituto Universitario Aeronáutico

2002 - Jefe de Programación en KWC

2004 - Jefe de Programación Backend en Brandigital

2008 - Líder técnico en QHOM Technologies

2012 - CTO en Glooday

2014 - Socio Gerente en Invatys

Antecedentes académicos:

2000 - Profesor de cursos de Visual Basic 6 en el Instituto Adminnet.

2000 - Profesor de cursos de Sitios Web-ASP-SQL en el Instituto Adminnet.

2012 - Docente de cursos de Programación y Desarrollo de Sitios Web en la UTN

Carta al alumno

Estimado alumno

Bienvenido al Diplomado en Desarrollo WEB. Este curso tiene la finalidad de capacitarlos en lo último en tecnología web que actualmente es punta dentro del ámbito tecnológico aquí en Argentina y el mundo.

La idea de la metodología semi-presencial/distancia de dictado del diplomado, es que usted como alumno se sienta cómodo y que asimile todos los conocimientos de la guía* y la misma sea autosuficiente. Convirtiéndose el Docente en una herramienta complementaria del material.

El material fue hecho en base a tutoriales y manuales que se recopilaron de Internet, ya que al ser un tema que es tendencia en este momento, el mismo se encuentra bien desarrollado por diferentes autores que fueron precursores en estos temas. La premisa del diplomado es que ustedes tengan a su alcance una guía* didáctica y con ricos contenidos para que al finalizar el curso con ayuda del docente usted tenga los conocimientos más avanzados en la materia.

Esperando su consulta por cualquier duda que se le presente a lo largo del diplomado, lo saludo cordialmente.

Ing. Ricardo Martín Espeche

Referencias

Este material se confecciono en base a las últimas tendencias que hoy se encuentra en Internet. Para eso se buscó el mejor material y se recompilo dicha información para facilitar el aprendizaje del mismo.

El material utilizado para esta primera unidad fue:

- HTML5 y CSS3 Manuales Tecnicos (Anaya)
- El gran libro de HTML5, CSS3 y Javascript (Juan Diego Gauchat)

Índice

Introducción	5
Capítulo 1	7
Documentos HTML5	7
1.1 Componentes básicos	7
1.2 Estructura global	15
1.3 Estructura del cuerpo	20
1.4 Contenido del cuerpo	29
1.5 Nuevos y viejos elementos	38
1.6 Referencia rápida	41
Capítulo 2	43
CSS3	43
2.1 CSS y HTML	43
2.2 Conceptos básicos sobre estilos	45
2.3 Referencias	47
2.4 Referencia rápida	55
2.5 Propiedades del CSS3	57
2.6 Referencia rápida	71

Introducción

HTML5 no es una nueva versión del antiguo lenguaje de etiquetas, ni siquiera una mejora de esta ya antigua tecnología, sino un nuevo concepto para la construcción de sitios web y aplicaciones en una era que combina dispositivos móviles, computación en la nube y trabajos en red.

Todo comenzó mucho tiempo atrás con una simple versión de HTML propuesta para crear la estructura básica de páginas web, organizar su contenido y compartir información. El lenguaje y la web misma nacieron principalmente con la intención de comunicar información por medio de texto. El limitado objetivo de HTML motivó a varias compañías a desarrollar nuevos lenguajes y programas para agregar características a la web nunca antes implementadas.

Estos desarrollos iniciales crecieron hasta convertirse en populares y poderosos accesorios. A la par de este crecimiento, las mismas quisieron dar un salto para convertir la información en flujos de datos dinámicos en donde el contenido pasa a ser interactivo. Como también surgió la necesidad de generar animaciones sobre los elementos de la página, para que sean más atractivas e interactivas para el usuario. Así surgieron los componentes Flash y Java (plugins) para los browser, algo que se inserta dentro de una estructura pero que comparte con el mismo espacio en la pantalla. Pero existía un problema.... no existía comunicación e integración alguna entre aplicaciones y documentos

La falta de integración resultó ser crítica y preparó el camino para la evolución de un lenguaje que comparte espacio en el documento con HTML y no está afectado por las limitaciones de los plug-ins. Javascript, un lenguaje interpretado incluido en navegadores, claramente era la manera de mejorar la experiencia de los usuarios y proveer funcionalidad para la web. Sin embargo, después de algunos años de intentos fallidos para promoverlo y algunos malos usos, el mercado nunca lo adoptó plenamente y pronto su popularidad declinó. Los detractores tenían buenas razones para oponerse a su adopción.

En ese momento, Javascript no era capaz de reemplazar la funcionalidad de Flash o Java. A pesar de ser evidente que ambos limitaban el alcance de las aplicaciones y aislaban el contenido web, populares funciones como la reproducción de video se estaban convirtiendo en una parte esencial de la web y solo eran efectivamente ofrecidas a través de estas tecnologías. A pesar del suceso inicial, el uso de Java comenzó a declinar. La naturaleza compleja del lenguaje, su evolución lenta y la falta de integración disminuyeron su importancia hasta el punto en el que hoy día no es más usado en aplicaciones web de importancia. Sin Java, el mercado volcó su atención a Flash. Pero el hecho de que Flash comparte las mismas características básicas que su competidor en la web lo hace también susceptible de correr el mismo destino.

Mientras esta competencia silenciosa se llevaba a cabo, el software para acceder a la web continuaba evolucionando. Junto con nuevas funciones y técnicas rápidas de acceso a la red, los navegadores también mejoraron gradualmente sus intérpretes Javascript. La innovación y los increíbles resultados obtenidos llamaron la atención de más programadores. Pronto lo que fue llamado la "Web 2.0" nació y la percepción de Javascript en la comunidad de programadores cambió radicalmente.

Javascript era claramente el lenguaje que permitía a los desarrolladores innovar y hacer cosas que nadie había podido hacer antes en la web. En los últimos años, programadores y

diseñadores web alrededor del mundo surgieron con los más increíbles trucos para superar las limitaciones de esta tecnología y sus iniciales deficiencias en portabilidad. Gracias a estas nuevas implementaciones, Javascript, HTML y CSS se convirtieron pronto en la más perfecta combinación para la necesaria evolución de la web.

HTML5 es, de hecho, una mejora de esta combinación, el pegamento que une todo. HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas. A partir de ahora, HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y Javascript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales. Las barreras entre sitios webs y aplicaciones finalmente han desaparecido. Las tecnologías requeridas para el proceso de integración están listas. El futuro de la web es prometedor y la evolución y combinación de estas tres tecnologías (HTML, CSS y Javascript) en una poderosa especificación está volviendo a Internet la plataforma líder de desarrollo. HTML5 indica claramente el camino.

Capítulo 1

Documentos HTML5

1.1 Componentes básicos

HTML5 provee básicamente tres características: *estructura*, *estilo* y *funcionalidad*. Nunca fue declarado oficialmente pero, incluso cuando algunas APIs (Interface de Programación de Aplicaciones) y la especificación de CSS3 por completo no son parte del mismo, HTML5 es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5.

HTML está a cargo de la estructura, CSS presenta esa estructura y su contenido en la pantalla y Javascript hace el resto que (como veremos más adelante) es extremadamente significativo. Más allá de esta integración, la estructura sigue siendo parte esencial de un documento. La misma provee los elementos necesarios para ubicar contenido estático o dinámico, y es también una plataforma básica para aplicaciones.

Con la variedad de dispositivos para acceder a Internet, la estructura debe proveer forma, organización y flexibilidad, y debe ser tan fuerte como los fundamentos de un edificio. Para trabajar y crear sitios webs y aplicaciones con HTML5, necesitamos saber primero cómo esa estructura es construida.

Crear fundamentos fuertes nos ayudará más adelante a aplicar el resto de los componentes para aprovechar completamente estas nuevas tecnologías.

1.1.1- Un modelo no-lineal

Deberías haber oído que los entornos de hipertexto e hipermedia en el Web son considerados como no-lineales. Lo que esto pretende explicar es que en vez de ofrecer la información en una línea, de estilo página tras página, la estructura del Web permite a los visitantes de los sitios saltar de una información a otra enlazada y a otros sitios relacionados.

Esto también significa que los codificadores de HTML necesitan tener algún sentido del entorno y como crear una arquitectura efectiva dentro de este marco de trabajo. Dado que la mayoría de la gente está mentalizada en conceptos lineales y están acostumbrados a leer en una sola dirección, una página a la vez, la repentina y salvaje Web puede llegar a ser excitante e irresistible. Esto hace que la arquitectura de las páginas Web sea tan interesante para los navegantes.

En muchos casos, esto significa ofrecer una estructura lineal, página por página, dentro de los sitios Web, o crear sitios basados en estructuras jerárquicas que resulten familiares. Esto controla el entorno y da a la gente una sensación de seguridad de que el hipertexto y el

hipermedia no son lugares no-lineales repletos de saltos, y el usuario final estará mucho menos confundido.

Una sencilla forma de hacerse una idea de este complejo concepto, no es más que fijarse en la última página Web que hayas visto y que tuviera varios enlaces, navegación y banners de publicidad.

¿A dónde vas? ¿Harás clic en el banner inmediatamente? Si lo haces, saldrás del sitio, y perderás todo el contenido que estás visualizando.

¿Lees todo el material primero y después escoges un enlace del menú de navegación? Si haces esto, cuál escogerías? Posiblemente decidirás atacar el sitio a modo lineal, primero leyendo el contenido, moviéndote desde el link situado más arriba o más a la izquierda, y seguir adelante. O, posiblemente, te aventuras en hacer un clic aleatorio en un enlace de navegación.

De cualquier forma, estas opciones te muestran claramente cómo el Web es un entorno lleno de posibilidades... ¡y confusión!

1.1.2- Estándares

Los estándares son reglas formales que deben pasar un examen riguroso por un comité. En caso del HTML, dicho comité se refiere al World Wide Web Consortium (W3C). Este grupo, es el cuerpo gubernamental de expertos académicos, profesionales e industriales que estudian, comentan, y finalmente determinan lo que debe ser publicado como un estándar.

Los estándares son importantes, son las guías con las cuales los navegadores así como codificadores de HTML deberían estar trabajando. De todas formas, hay una mala interpretación con el HTML, donde algunos estándares no podrían mantener los cambios, así que se han excedido con ellos. Esto significa que las codificaciones, usando el HTML para hacer diseños en todas las plataformas y todos los navegadores, deberían de trabajar alrededor de un estándar que les asegurara y usara métodos convencionales para crear sitios Web efectivos.

Una convención es simplemente una forma común de hacer una tarea en particular. A diferencia de un estándar, el cual formaliza la forma en que el grupo de la comunidad responde a la información dentro del estándar, una convención es el reflejo de las prácticas de esa comunidad.

La organización que puede ayudar a darte flexibilidad es la World Wide Web Consortium. Este consorcio fue formado el 14 de diciembre de 1994. Es una organización independiente e internacional. La tarea del consorcio es tratar la estandarización del HTML, así como varios protocolos y lenguajes relacionados con el Web, incluyendo HTTP, URL, FTP, Gopher, WAIS, NNTP, SGML y SGL.

Un desarrollo interesante de este tipo de evolución y las reglas formalizadas del comité es que mientras la W3C ha sido el líder creciente en ofrecer sistemas de información de Internet, la Web ha sido, tal como describimos antes, la popularidad del formato. Algunos temas del consorcio han sido dejados de lado para centrarse más en las necesidades de satisfacer a los

entusiastas del Web y animar a los desarrolladores para una mayor flexibilidad. Algunos de los cambios propuestos incluyen mejoras en los protocolos. El HTTP: el método de obtención del HTML, está siendo tratado para importantes cambios. Los nuevos cambios propuestos hablan de tecnologías que mejorarán el diseño y funcionalidad de los sitios Web.

1.1.3 ANTES DE EMPEZAR

Para fomentar una sólida comprensión del HTML, miraremos la estructura del lenguaje propiamente dicho. El funcionamiento de la sintaxis del HTML es análogo a la sintaxis de la gramática española, pero, por supuesto, mucho más sencillo.

El HTML es en realidad un lenguaje muy lógico. Ciertamente, hay excepciones a las reglas, y modificaciones o interpretaciones de algunas de éstas. De todas formas, una vez que asimiles la estructura básica, empezarás a notar que el HTML es simplemente un conjunto de piezas lógicas que hacen de él un lenguaje muy cómodo de usar. Mientras que la evolución es un estado natural del HTML, los cambios normalmente no afectan a las reglas básicas. Mientras que varios elementos de este lenguaje se hacen obsoletos y otros nuevos son añadidos, la sintaxis, o la estructura correcta, raramente sufren cambios.

Para comprender el concepto de sintaxis, piense en una frase, esta tiene un sujeto y un verbo. Los adjetivos y los adverbios se añaden para aportar cualidades y comportamiento, haciendo la frase más comprensiva. El HTML es, en el fondo, parecido a una simple frase. De hecho, los componentes del HTML siguen el mismo concepto de sujeto, verbo y adjetivos/adverbios.

1.1.3.1 Etiquetas

Lo primero que hay que saber es el funcionamiento de los documentos HTML, que significa Hyper Text Markup Language. Son líneas de texto modificadas con unas marcas, llamadas etiquetas o tags.

Las etiquetas que encierran el texto son la de apertura `<xxx>` y la de cierre `</xxx>`. Fíjate cómo no hay espacios entre los signos de mayor y menor, y el lugar y la orientación de la barra diagonal en la etiqueta de cierre. Hay etiquetas que requieren la de cierre (como `<A>` y ``), otras que pueden tener etiqueta de cierre o no (como `<P>`, que no siempre requiere `</P>`) y otras que no llevan nunca etiqueta de cierre (como ``, la cual jamás lleva ``). Para saber cuándo se quiere etiqueta de cierre, necesitarás conocer todas y cada una de ellas, lo irás aprendiendo con la experiencia.

1.1.3.2 Atributos

Los atributos modifican las acciones de las etiquetas. Muchas etiquetas pueden permanecer solas, mientras que otras necesitan un atributo para darles funcionalidad. Por ejemplo, `<HTML>` nunca lleva atributos, mientras que `<BODY>` puede llevarlos o no (por ejemplo, `<BODY BGCOLOR...>`).

Ten en cuenta que los atributos siempre van en la etiqueta de apertura, nunca en la de cierre. Es absurdo poner algo del estilo de `</BODY BGCOLOR=...>`.

1.1.3.3 Valores

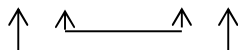
Definen el atributo al que se lo hemos aplicado. El valor puede ser una palabra (por ejemplo `<P ALIGN="center">...</P>`) o un número de pixels o porcentaje (como en `<TABLE WIDTH="40%" HEIGHT="500">...</TABLE>`). A veces un número no define un número de pixels concreto, por ejemplo, al cambiar el tamaño de las fuentes (`...`) el número es simplemente orientativo, no tiene ninguna relación con el tamaño final, y a veces puede ser en formato hexadecimal (16 números, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F...), como en `...`. Podrás conocer el funcionamiento de estos números más adelante, cuando expliquemos los atributos que los requieren.

1.1.3.4 Simetría

Poner las etiquetas en el orden correcto nos permitirá una edición del documento más rápida y efectiva

Por ejemplo: `<I>Hola</I>` y `<I>Hola</I>` daría el mismo resultado, pero es más recomendable usar lo segundo: puedes apreciar una simetría horizontal.

`<I>Hola</I>`



1.1.4 MANEJO DE CARACTERES

Habrás observado en el ejemplo anterior, que en los textos de los mismos no hay acentos, ni ñes, ni símbolos de abrir interrogación o exclamación. Esto es debido a los distintos juegos de caracteres que manejan los ordenadores.

Las máquinas manejan la información en formato binario (es decir, en unos y ceros). Estos, a su vez, forman números, los cuales se traducen en letras. ¿Cómo? Mediante tablas. Podemos asignar el valor 64 a la letra a, el 65 a la b, etc...

El problema está en que cada ordenador es de un fabricante distinto y puede adoptar una tabla diferente al resto. Para evitarlo existen diversos estándares y el más extendido es el ASCII. De hecho, actualmente todos los ordenadores tienen la misma tabla ASCII para los primeros 127 caracteres. Pero esa tabla no contiene vocales con acento, ni ñes, ni símbolos de abrir interrogación o exclamación... Esto nos pasa por dejar que los norteamericanos sean quienes construyan las computadoras.


El HTML 2.0 eligió como tabla estándar la ISO-Latin-1, que comparte con la ASCII los 127 caracteres e incluye unos cuantos más hasta el número 255.

1.1.4.1 Caracteres extendidos en HTML

La manera de incluir los caracteres extendidos (cuyo número está más allá del 127) consiste en encerrar el código entre los caracteres `&` y `;`. Así pues, lo siguiente:

`½`

nos debería dar un medio ($\frac{1}{2}$). También existe una serie de sinónimos para poder recordar con más facilidad estos caracteres. Así, por ejemplo, `½` también se puede escribir como `½`. Vamos a ver algunos de estos códigos, los más útiles a la hora de escribir en español:

Código	Resultado
<code>&aacute;</code> , <code>&Aacute;</code> , <code>&eacute;</code> , <code>&Eacute;</code> ,...	á, Á, é, É, í, Í, ó, Ó, ú y Ú
ñ y ñ	ñ y ñ
<code>&iquest;</code>	¿
<code>&iexcl;</code>	¡
<code>&ordm;</code>	º
<code>&ordf;</code>	ª
<code>&trade;</code>  o <code>&#153;</code>	™ ◯ ™
<code>&copy;</code>	©
<code>&reg;</code>	®
<code>&nbsp;</code>	(espacio en blanco que no puede ser usado para saltar de línea)

1.1.4.2 Juego de caracteres

Este apartado puedes saltarlo si quieres, basta que sepas que todos los visores de páginas Web actuales soportan todos los caracteres gráficos de la especificación ISO 8859-1, que permiten escribir textos en la mayoría de los países occidentales. Si tienes que insertar en tu página Web algún carácter fuera de lo normal, lee este apartado.

Se han definido dos formas de representar caracteres especiales usando solamente el código ASCII de 7 bits. Estos caracteres pueden ser representados por un código numérico o una entidad cuando deseemos que aparezcan en el documento "tal cual".

A continuación están las principales entidades:

Carácter	Código	Entidad	Carácter	Código	Entidad	Carácter	Código	Entidad	Carácter	Código	Entidad
!	!	--	"	"	--	#	#	--	\$	$	--
%	%	--	&	&	--	'	'	--	((--
))	--	*	*	--	+	+	--	'	,	--
-	-	--	.	.	--	/	/	--	:	:	--
;	;	--	<	<	--	=	=	--	>	>	--
?	?	--	@	@	--	[[--	\	\	--
]]	--	^	^	--	_	_	--	`	`	--
{	{	--		|	--	}	}	--	~	~	--
&	 	nbsp	¡	¡	ixcl	¢	¢	cent	£	£	pound
¤	¤	curren	¥	¥	yen	¦	¦	brvbar	§	§	Sect
¨	¨	uml	(c)	©	copy	ª	ª	ordf	«	«	laquo
¬	¬	not		­	shy	(r)	®	reg	—	¯	macr
°	°	deg	±	±	plasm	²	²	sup2	³	³	sup3
´	´	acute	µ	µ	micro	¶	¶	para	·	·	middot
¸	¸	cedil	¹	¹	sup1	º	º	ordm	»	»	raquo
1/4	¼	frac14	1/2	½	frac12	3/4	¾	frac34	¿	¿	quest
À	À	Agrave	Á	Á	Aacute	Â	Â	Acirc	Ã	Ã	Atilde
Ä	Ä	Auml	Å	Å	Aring	Æ	Æ	AEling	Ç	Ç	Ccedil
È	È	Egrave	É	É	Eacute	Ê	Ê	Ecirc	Ë	Ë	Euml
Ì	Ì	Igrave	Í	Í	Iacute	Î	Î	Icirc	Ï	Ï	Iuml
Ð	Ð	Eth	Ñ	Ñ	Ntilde	Ò	Ò	Ograve	Ó	Ó	Oacute
Ô	Ô	Ocirc	Õ	Õ	Otilde	Ö	Ö	Ouml	×	×	times
Ø	Ø	Oslash	Ù	Ù	Ugrave	Ú	Ú	Uacute	Û	Û	Ucirc
Ü	Ü	Uuml	Ý	Ý	Yacute	Þ	Þ	Thorn	ß	ß	szlig
à	à	agrave	á	á	aacute	â	â	acirc	ã	ã	atilde

ä	ä auml	å	å aring	æ	æ aeling	ç	ç ccedil
è	è egrave	é	é eacute	ê	ê ecirc	ë	ë euml
ì	ì igrave	í	í iacute	î	î icirc	ï	ï iuml
ð	ð eth	ñ	ñ ntilde	ò	ò ograve	ó	ó oacute
ô	ô ocirc	õ	õ otilde	ö	ö ouml	÷	÷ divide
ø	ø oslash	ù	ù ugrave	ú	ú uacute	û	û ucirc
Ü	ü uuml	ý	ý yacute	þ	þ thorn	ÿ	ÿ yuml

1.1.4.3 Caracteres de control

En el HTML existen cuatro caracteres de control, que se usan para formar etiquetas, establecer parámetros, etc... Para poder emplearlos sin riesgo deberíamos escribir los siguientes códigos:

Código	Resultado
<	<
>	>
&	&
"	"

Si se quiere poner el carácter "<" en el texto deberían usar < (ASCII decimal 60) para evitar la posible confusión con el comienzo de una etiqueta (delimitador de apertura de etiqueta inicial). Análogamente, se debería usar > (ASCII decimal 62) en el texto en lugar de ">" para evitar problemas con agentes de usuario antiguos que lo interpretan incorrectamente como el final de una etiqueta (delimitador de cierre de una etiqueta) cuando aparece dentro de valores de atributos entrecomillados.

Se debe usar & (ASCII decimal 38) en lugar de "&" para evitar la confusión con el comienzo de una referencia de caracteres (delimitador de apertura de una referencia a entidades). Se debería usar también & en valores de atributos, ya que las referencias de caracteres están permitidas dentro de valores de atributos CDATA.

Algunas veces se usa la referencia a entidades de caracteres " para codificar las comillas dobles ("), ya que este carácter puede utilizarse para delimitar los valores de los atributos.

1.1.4.4 Mayúsculas, espacios y comillas

La forma correcta de poner una etiqueta es la siguiente:

```
<IMG SRC="hola.gif">
```

La utilización de mayúsculas y minúsculas es indiferente en la etiqueta y el atributo (no así en el valor, no siempre puede cambiarse indistintamente). Recomendamos ponerlo como en el ejemplo, etiqueta y atributo en mayúsculas y el valor en minúscula.

Fíjate en las comillas. Se recomienda ponerlas siempre, aunque si el valor contiene sólo un simple número o una simple palabra no suele ser necesario. Por ejemplo, `width=200` y `width="200"` daría el mismo resultado, pero se recomienda poner siempre las comillas (`ALT=Aquí pone hola` está mal, hay que poner `ALT="Aquí pone hola"`).

El uso de los espacios de esta manera `< IMG SRC = " hola.gif " >` o de esta otra `<IMGSRC="hola.gif">` es totalmente incorrecto. El único espacio debe estar situado entre etiqueta/atributo y atributo_con_valor/atributo_con_valor. Por ejemplo:

```
<IMG SRC="hola.gif" ALT="hola">
```

Si queremos poner espacios en el texto normal y corriente, deberemos usar el carácter especial ` `. Por ejemplo:

```
Esta es mi página WEB
```

y

```
Esta    es mi página WEB
```

se verían de la siguiente forma:

```
Esta es mi página WEB
```

Si queremos poner lo segundo, con cuatro espacios en blanco entre medio habrá que poner:

```
Esta&nbsp;&nbsp;&nbsp;&nbsp; es mi página WEB
```

Y, para finalizar:

```
este es un texto <B>en negrita</B>
```

y

```
este es un texto<B> en negrita</B>
```

daría un resultado idéntico, aunque se recomienda usar lo primero.

1.2 Estructura global

Los documentos HTML se encuentran estrictamente organizados. Cada parte del documento está diferenciada, declarada y determinada por etiquetas específicas. Ahora vamos a ver cómo construir la estructura global de un documento HTML y los nuevos elementos semánticos incorporados en HTML5.

1.2.1 DOCTYPE

En primer lugar necesitamos indicar el tipo de documento que estamos creando. Esto en HTML5 es extremadamente sencillo:

```
<!DOCTYPE html>
```

Esta línea debe ser la primera línea del archivo, sin espacios o líneas que la preceden. De esta forma, el modo estándar del navegador es activado y las incorporaciones de HTML5 son interpretadas siempre que sea posible, o ignoradas en caso contrario.

1.2.2 HTML

Luego de declarar el tipo de documento, debemos comenzar a construir la estructura HTML. Como siempre, la estructura tipo árbol de este lenguaje tiene su raíz en el elemento .

Este elemento envolverá el resto del código:

```
<!DOCTYPE html>
<html lang="es">
</html>
```

El atributo lang en la etiqueta de apertura es el único atributo que necesitamos especificar en HTML5. Este atributo define el idioma del contenido del documento que estamos creando, en este caso es por español.

El lenguaje HTML usa las etiquetas para construir páginas web. Estas etiquetas HTML son palabras clave y atributos rodeados de los signos mayor y menor (por ejemplo,). En este caso, **html** es la palabra clave y *lang* es el atributo con el valor es. La mayoría de las etiquetas HTML se utilizan en pares, una etiqueta de apertura y una de cierre, y el contenido se declara entre ellas. En nuestro ejemplo, <HTML> indica el comienzo del código HTML y </HTML> indica el final. Nuestro código será insertado entre estas dos etiquetas.

Cabe destacar que es indistinto escribir las etiquetas en minúscula o mayúscula.

1.2.3 HEAD

Continuemos construyendo nuestra plantilla. El código HTML insertado entre las etiquetas tiene que ser dividido entre dos secciones principales. Al igual que en versiones previas de HTML, la primera sección es la cabecera (<HEAD>) y la segunda el cuerpo (<BODY>). El siguiente paso, por lo tanto, será crear estas dos secciones en el código usando los elementos y ya conocidos. El elemento va primero, por supuesto, y al igual que el resto de los elementos estructurales tiene una etiqueta de apertura y una de cierre:

```
<!DOCTYPE html>
<html lang="es">
  <head>
  </head>
</html>
```

La etiqueta no cambió desde versiones anteriores y su propósito sigue siendo exactamente el mismo. Dentro de las etiquetas definiremos el título de nuestra página web, declararemos el set de caracteres correspondiente para los *meta data*, proveeremos información general acerca del documento e incorporaremos los archivos externos de los estilos, códigos Javascript o incluso imágenes necesarias para generar la página en la pantalla.

Excepto por el título (que se verá en parte superior de la ventana del browser) y algunos íconos, el resto de la información incorporada en el documento entre estas etiquetas es invisible para el usuario.

1.2.4 BODY

La siguiente gran sección que es parte principal de la organización de un documento HTML es el cuerpo. El cuerpo representa la parte visible de todo documento y es especificado entre etiquetas . Estas etiquetas tampoco han cambiado en relación con versiones previas de HTML:

```
<!DOCTYPE html>
<html lang="es">
  <head>
  </head>
  <body>
  </body>
</html>
```

Hasta el momento tenemos un código simple pero con una estructura compleja. Esto es porque el código HTML no está formado por un conjunto de instrucciones secuenciales. HTML es un lenguaje de etiquetas, un listado de elementos que usualmente se utilizan en pares y que pueden ser anidados. Más adelante en este capítulo veremos que más etiquetas son insertadas entre estas últimas conformando una estructura de árbol con como su raíz.

1.2.5 META

Es momento de construir la cabecera del documento. Algunos cambios e innovaciones fueron incorporados dentro de la cabecera, y uno de ellos es la etiqueta que define el juego de caracteres a utilizar para mostrar el documento.

Ésta es una etiqueta que especifica cómo el texto será presentado en pantalla:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1">
  </head>
  <body>
  </body>
</html>
```

La innovación de este elemento en HTML5, como en la mayoría de los casos, es solo simplificación. La nueva etiqueta para la definición del tipo de caracteres es más corta y simple. Por supuesto, podemos agregar etiquetas como *description* o *keywords* para definir algunos aspectos de la página web, como se vé en el siguiente ejemplo:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1">
    <meta name="description" content="Ejemplo de HTML5">
    <meta name="keywords" content="HTML5, CSS3, Javascript">
  </head>
  <body>
  </body>
</html>
```

Hay varios tipos de etiqueta que pueden ser incluidas para declarar información general sobre el documento, pero esta información no es mostrada en la ventana del navegador, es solo importante para motores de búsqueda y dispositivos que necesitan hacer una vista previa del documento u obtener un sumario de la información que contiene. Como comentamos anteriormente, aparte del título y algunos íconos, la mayoría de la información insertada entre las etiquetas no es visible para los usuarios. En el código anterior, el atributo *name* dentro de la etiqueta especifica su tipo y *content* declara su valor, pero ninguno de estos valores es mostrado en pantalla.

En HTML5 no es necesario cerrar etiquetas simples con una barra al final, pero recomendamos utilizarlas por razones de compatibilidad. El anterior código se podría escribir de la siguiente manera:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1"/>
    <meta name="description" content="Ejemplo de HTML5"/>
    <meta name="keywords" content="HTML5, CSS3, Javascript"/>
  </head>
  <body>
  </body>
</html>
```

1.2.6 TITLE

La etiqueta `<title>`, como siempre, simplemente especifica el título del documento, y no hay nada nuevo para comentar:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1">
    <meta name="description" content="Ejemplo de HTML5">
    <meta name="keywords" content="HTML5, CSS3, JavaScript">
    <title>Este texto es el título del documento</title>
  </head>
  <body>
  </body>
</html>
```

Normalmente el texto que se encuentra entre las etiquetas `<title></title>`, será el texto mostrado en la barra superior de la ventana del navegador.

1.2.7 LINK

Otro importante elemento que va dentro de la cabecera del documento es `<link>`. Este elemento es usado para incorporar estilos, códigos Javascript, imágenes o iconos desde archivos externos. Uno de los usos más comunes para es la incorporación de archivos con estilos CSS:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1">
    <meta name="description" content="Ejemplo de HTML5">
    <meta name="keywords" content="HTML5, CSS3, JavaScript">
    <title>Este texto es el título del documento</title>
    <link rel="stylesheet" href="misestilos.css">
  </head>
  <body>
  </body>
</html>
```

En HTML5 ya no se necesita especificar qué tipo de estilos estamos insertando, por lo que el atributo *type* fue eliminado. Solo necesitamos dos atributos para incorporar nuestro archivo de estilos: *rel* y *href*. El atributo *rel* significa “relación” y es acerca de la relación entre el documento y el archivo que estamos incorporando por medio de *href*. En este caso, el atributo *rel* tiene el valor *stylesheet* que le dice al navegador que el archivo *misestilos.css* es un archivo CSS con estilos requeridos para presentar la página en pantalla.

Un archivo de estilos es un grupo de reglas de formato que ayudarán a cambiar la apariencia de nuestra página web (por ejemplo, el tamaño y color del texto). Sin estas reglas, el texto y cualquier otro elemento HTML sería mostrado en pantalla utilizando los estilos estándar provistos por el navegador. Los estilos son reglas simples que normalmente requieren solo unas pocas líneas de código y pueden ser declarados en el mismo documento. No es estrictamente necesario obtener esta información de archivos externos pero es una práctica recomendada. Cargar las reglas CSS desde un documento externo (otro archivo) nos permitirá

organizar el documento principal, incrementar la velocidad de carga y aprovechar las nuevas características de HTML5.

1.3 Estructura del cuerpo

La estructura del cuerpo (el código entre las etiquetas `<body></body>`) generará la **parte visible del documento**. Este es el código que producirá nuestra página web. HTML siempre ofreció diferentes formas de construir y organizar la información dentro del cuerpo de un documento. Uno de los primeros elementos provistos para este propósito fue `<table>`. Las tablas permitían a los diseñadores acomodar datos, texto, imágenes y herramientas dentro de filas y columnas de celdas, incluso sin que hayan sido concebidas para este propósito. Más adelante, gradualmente, otros elementos reemplazaron su función, permitiendo lograr lo mismo con menos código, facilitando de este modo la creación, permitiendo portabilidad y ayudando al mantenimiento de los sitios web.

La estructura quedaría de la siguiente manera:

```
<HTML>
  <HEAD>
  ..
  </HEAD>

  <BODY>
  ..
  </BODY>
</HTML>
```

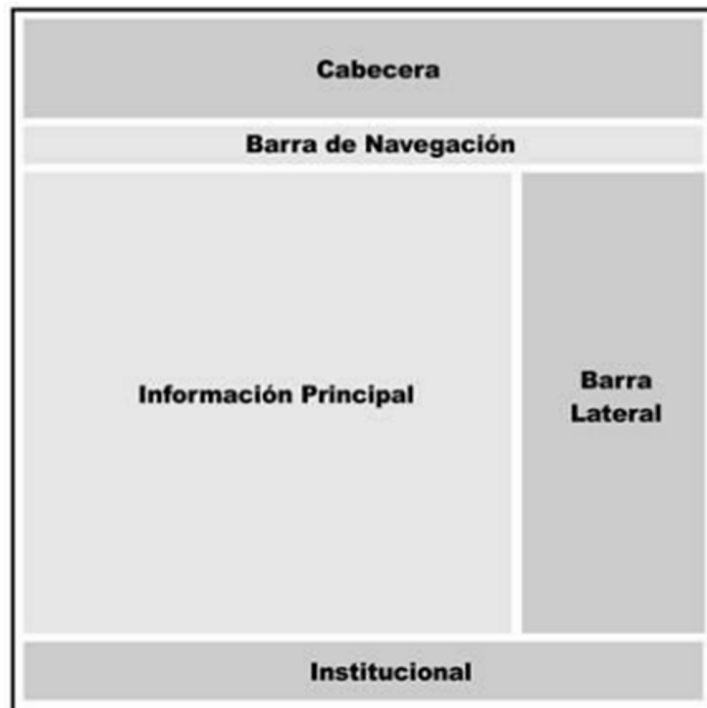
El elemento `<div>` (conocido como capa) comenzó a dominar la escena. Con el surgimiento de webs más interactivas y la integración de HTML, CSS y Javascript, el uso de las caps se volvió una práctica común. Pero este elemento, así como el `<table>`, no provee demasiada información acerca de las parte del cuerpo que está representando.

Luego de la revolución de los dispositivos móviles y el surgimiento de diferentes formas en que la gente accede a la web, la identificación de cada parte del documento es una tarea que se ha vuelto más relevante que nunca. Considerando todo lo expuesto, HTML5 incorpora nuevos elementos que ayudan a identificar cada sección del documento y organizar el cuerpo del mismo. En HTML5 las secciones más importantes son diferenciadas y la estructura principal ya no depende más de los elementos `<div>` o `<table>`.

Cómo usamos estos nuevos elementos? depende de nosotros, pero las palabras clave otorgadas a cada uno de ellos nos dan ayuda a entender sus funciones. Normalmente una página o aplicación web está dividida entre varias áreas visuales para mejorar la experiencia del usuario y facilitar la interactividad. Las palabras claves que representan cada nuevo elemento de HTML5 están íntimamente relacionadas con estas áreas, como se verá más adelante.

1.3.1 Organización

La siguiente figura representa un diseño común encontrado en la mayoría de los sitios webs estos días. A pesar del hecho de que cada diseñador crea sus propios diseños, en general podremos identificar las siguientes secciones en cada sitio web estudiado:



En la parte superior, referido como **Cabecera**, se encuentra el espacio donde usualmente se ubica el logo, título, subtítulos y una corta descripción del sitio web o la página. Inmediatamente debajo, podemos ver la **Barra de Navegación** en la cual casi todos los desarrolladores ofrecen un menú o lista de enlaces con el propósito de facilitar la navegación a través del sitio.

Los usuarios son guiados desde esta barra hacia las diferentes páginas o documentos, normalmente pertenecientes al mismo sitio web. El contenido más relevante de una página web se encuentra, en casi todo diseño, ubicado en su centro. Esta sección presenta información y enlaces valiosos.

La mayoría de las veces es dividida en varias filas y columnas. En la figura anterior se utilizaron solo dos columnas: **Información Principal** y **Barra Lateral**, pero esta sección es extremadamente flexible y normalmente diseñadores la adaptan acorde a sus necesidades insertando más columnas, dividiendo cada columna entre bloques más pequeños o generando diferentes distribuciones y combinaciones. El contenido presentado en esta parte del diseño es usualmente de alta prioridad.

En el diseño de ejemplo, **Información Principal** podría contener una lista de artículos, descripción de productos, entradas de un blog o cualquier otra información importante, y la **Barra Lateral** podría mostrar una lista de enlaces apuntando hacia cada uno de esos ítems. En un blog, por ejemplo, esta última columna ofrecerá una lista de enlaces apuntando a cada entrada del blog, información acerca del autor, etc...

En la base de un diseño web clásico siempre nos encontramos con una barra más que aquí llamamos **Institucional**. La nombramos de esta manera porque esta es el área en donde normalmente se muestra información acerca del sitio web, el autor o la empresa, además de algunos enlaces con respecto a reglas, términos y condiciones y toda información adicional que el desarrollador considere importante compartir. La barra **Institucional** es un complemento de

la **Cabecera** y es parte de lo que se considera estos días la estructura esencial de una página web, como podemos apreciar en el siguiente ejemplo:

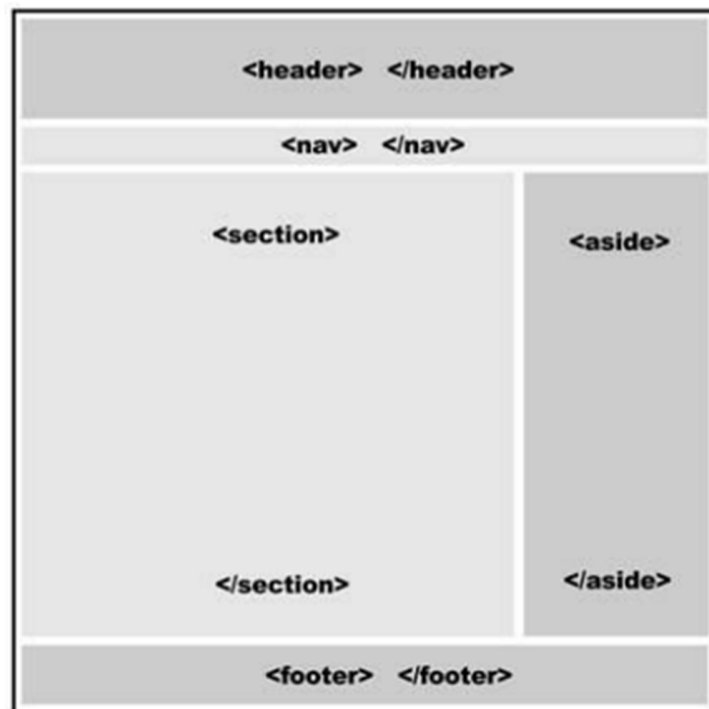


La imagen es una representación de un blog normal. En este ejemplo se puede claramente identificar cada parte del diseño considerado anteriormente.

1. Cabecera
2. Barra de Navegación
3. Sección de Información Principal
4. Barra Lateral
5. El pie o la barra Institucional

Esta simple representación de un blog nos puede ayudar a entender que cada sección definida en un sitio web tiene un propósito. A veces este propósito no es claro pero en esencia se encuentra siempre allí, ayudándonos a reconocer cualquiera de las secciones descritas anteriormente en todo diseño. HTML5 considera esta estructura básica y provee nuevos elementos para diferenciar y declarar cada una de sus partes.

A partir de ahora podemos decir al navegador para qué es cada sección:



La imagen muestra el típico diseño presentado anteriormente, pero esta vez con los correspondientes elementos HTML5 para cada sección (incluyendo etiquetas de apertura y cierre).

1.3.2 <HEADER>

Uno de los nuevos elementos incorporados en HTML5 es `<header>`. El elemento no debe ser confundido con `<head>` usado antes para construir la cabecera del documento. Del mismo modo que `<head>`, la intención de es proveer información introductoria (títulos, subtítulos, logos), pero difiere con respecto a `<head>` en su alcance. Mientras que el elemento `<head>` tiene el propósito de proveer información acerca de todo el documento, `<header>` es usado solo para el cuerpo o secciones específicas dentro del cuerpo:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1">
    <meta name="description" content="Ejemplo de HTML5">
    <meta name="keywords" content="HTML5, CSS3, JavaScript">
    <title>Este texto es el título del documento</title>
    <link rel="stylesheet" href="misestilos.css">
  </head>
  <body>
    <header>
      <h1>Este es el título principal del sitio web</h1>
    </header>
  </body>
</html>
```


En el código, definimos el título de la página web utilizando el elemento `<header>`. Recuerde que esta cabecera no es la misma que la utilizada previamente para definir el título del documento. La inserción del elemento representa el comienzo del cuerpo y por lo tanto de la parte visible del documento.

1.3.3 <NAV>

Siguiendo con nuestro ejemplo, la siguiente sección es la Barra de Navegación. Esta barra es generada en HTML5 con el elemento `<nav>`:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1">
    <meta name="description" content="Ejemplo de HTML5">
    <meta name="keywords" content="HTML5, CSS3, JavaScript">
    <title>Este texto es el título del documento</title>
    <link rel="stylesheet" href="misestilos.css">
  </head>
  <body>
    <header>
      <h1>Este es el título principal del sitio web</h1>
    </header>
    <nav>
      <ul>
        <li>principal</li>
        <li>fotos</li>
        <li>videos</li>
        <li>contacto</li>
      </ul>
    </nav>
  </body>
</html>
```

Como se puede apreciar en el código, el elemento `<nav>` se encuentra dentro de las etiquetas `<body>` pero es ubicado después de la etiqueta de cierre de la cabecera (`</header>`), no dentro de las etiquetas `<header>`. Esto es porque `<nav>` no es parte de la cabecera sino una nueva sección.

Anteriormente dijimos que la estructura y el orden que elegimos para colocar los elementos HTML5 dependen de nosotros. Esto significa que HTML5 es versátil y solo nos otorga los parámetros y elementos básicos con los que trabajar, pero cómo usarlos será exclusivamente decisión nuestra. Un ejemplo de esta versatilidad es que el elemento `<nav>` podría ser insertado dentro del elemento `<header>` o en cualquier otra parte del cuerpo. Sin embargo, siempre se debe considerar que estas etiquetas fueron creadas para brindar información a los navegadores y ayudar a cada nuevo programa y dispositivo en el mercado a identificar las partes más relevantes del documento.

Para conservar nuestro código portable y comprensible, recomendamos como buena práctica seguir lo que marcan los estándares y mantener todo tan claro como sea posible. El elemento `<nav>` fue creado para ofrecer ayuda para la navegación, como en menús principales o grandes bloques de enlaces, y debería ser utilizado de esa manera.

1.3.4 <SECTION>

Siguiendo nuestro diseño estándar nos encontramos con las columnas que en la figura llamamos **Información Principal** y **Barra Lateral**.



Como explicamos anteriormente, la columna Información Principal contiene la información más relevante del documento y puede ser encontrada en diferentes formas (por ejemplo, dividida en varios bloques o columnas). Debido a que el propósito de estas columnas es más general, el elemento en HTML5 que especifica estas secciones se llama simplemente `<section>`:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1">
    <meta name="description" content="Ejemplo de HTML5">
    <meta name="keywords" content="HTML5, CSS3, JavaScript">
    <title>Este texto es el título del documento</title>
    <link rel="stylesheet" href="misestilos.css">
  </head>
  <body>
    <header>
      <h1>Este es el título principal del sitio web</h1>
    </header>
    <nav>
      <ul>
        <li>principal</li>
        <li>fotos</li>
        <li>videos</li>
        <li>contacto</li>
      </ul>
    </nav>
```

```

        <section>
        </section>
    </body>
</html>

```

Al igual que la **Barra de Navegación**, la columna **Información Principal** es una sección aparte. Por este motivo, la sección para **Información Principal** va debajo de la etiqueta de cierre </nav>.

1.3.5 <ASIDE>

En un típico diseño web la columna llamada **Barra Lateral** se ubica al lado de la columna **Información Principal**. Esta es una columna o sección que normalmente contiene datos relacionados con la información principal pero que no son relevantes o igual de importantes.

En el diseño de un blog, por ejemplo, la **Barra Lateral** contendrá una lista de enlaces. La información dentro de esta barra está relacionada con la información principal pero no es relevante por sí misma. Siguiendo el mismo ejemplo podemos decir que las entradas del blog son relevantes pero los enlaces y las pequeñas reseñas sobre esas entradas son solo una ayuda para la navegación pero no lo que al lector realmente le interesa.

En HTML5 podemos diferenciar esta clase secundaria de información utilizando el elemento <aside>:

```

<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1">
    <meta name="description" content="Ejemplo de HTML5">
    <meta name="keywords" content="HTML5, CSS3, JavaScript">
    <title>Este texto es el título del documento</title>
    <link rel="stylesheet" href="misestilos.css">
  </head>
  <body>
    <header>
      <h1>Este es el título principal del sitio web</h1>
    </header>
    <nav>
      <ul>
        <li>principal</li>
        <li>fotos</li>
        <li>videos</li>
        <li>contacto</li>
      </ul>
    </nav>
    <section>
    </section>
    <aside>
      <blockquote>Mensaje número uno</blockquote>
      <blockquote>Mensaje número dos</blockquote>
    </aside>
  </body>
</html>

```

El elemento <aside> podría estar ubicado del lado derecho o izquierdo de nuestra página de ejemplo, la etiqueta no tiene una posición predefinida. El elemento <aside> solo describe la información que contiene, no el lugar dentro de la estructura. Este elemento puede estar ubicado en cualquier parte del diseño y ser usado siempre y cuando su contenido no sea considerado como el contenido principal del documento. Por ejemplo, podemos usar <aside> dentro del elemento <section> o incluso insertado entre la información relevante, como en el caso de una cita.

1.3.6 <FOOTER>

Para finalizar la construcción de la plantilla o estructura elemental de nuestro documento HTML5, solo necesitamos un elemento más. Ya contamos con la cabecera del cuerpo, secciones con ayuda para la navegación, información importante y hasta una barra lateral con datos adicionales, por lo tanto lo único que nos queda por hacer es cerrar nuestro diseño para otorgarle un final al cuerpo del documento.

HTML5 provee un elemento específico para este propósito llamado <footer>:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="iso-8859-1">
    <meta name="description" content="Ejemplo de HTML5">
    <meta name="keywords" content="HTML5, CSS3, JavaScript">
    <title>Este texto es el título del documento</title>
    <link rel="stylesheet" href="misestilos.css">
  </head>
  <body>
    <header>
      <h1>Este es el título principal del sitio web</h1>
    </header>
    <nav>
      <ul>
        <li>principal</li>
        <li>fotos</li>
        <li>videos</li>
        <li>contacto</li>
      </ul>
    </nav>
    <section>
    </section>
    <aside>
      <blockquote>Mensaje número uno</blockquote>
      <blockquote>Mensaje número dos</blockquote>
    </aside>
    <footer>
      Derechos Reservados &copy; 2010-2011
    </footer>
  </body>
</html>
```

En el típico diseño de una página web la sección llamada Institucional será definida por etiquetas <footer>. Esto es debido a que la barra representa el final (o pie) del documento y esta parte de la página web es normalmente usada para compartir información general sobre el autor o la organización detrás del proyecto.

Generalmente, el elemento `<footer>` representará el final del cuerpo de nuestro documento y tendrá el propósito descrito anteriormente. Sin embargo, `<footer>` puede ser usado múltiples veces dentro del cuerpo para representar también el final de diferentes secciones (del mismo modo que la etiqueta `<header>`).

1.4 Contenido del cuerpo

Los elementos HTML5 estudiados hasta el momento nos ayudan a identificar cada sección del diseño y asignar un propósito intrínseco a cada una de ellas, pero lo que es realmente importante para nuestro sitio web se encuentra en el interior de estas secciones.

La mayoría de los elementos ya estudiados fueron creados para construir una estructura para el documento HTML que pueda ser identificada y reconocida por los navegadores y nuevos dispositivos. Aprendimos acerca de la etiqueta `<body>` usada para declarar el cuerpo o parte visible del documento, la etiqueta `<header>` con la que agrupamos información importante para el cuerpo, la etiqueta `<nav>` que provee ayuda para la navegación del sitio web, la etiqueta `<section>` necesaria para contener la información más relevante, y también `<aside>` y `<footer>` para ofrecer información adicional de cada sección y del documento mismo. Pero ninguno de estos elementos declara algo acerca del contenido.

Todos tienen un específico propósito estructural. Más profundo nos introducimos dentro del documento más cerca nos encontramos de la definición del contenido. Esta información estará compuesta por diferentes elementos visuales como títulos, textos, imágenes, videos y aplicaciones interactivas, entre otros. Necesitamos poder diferenciar estos elementos y establecer una relación entre ellos dentro de la estructura.

1.4.1 Elementos HTML4

1.4.1.1 Formato del párrafo

Las etiquetas de párrafo `<P>` nos servirán para iniciar un nuevo párrafo de texto. No necesita la etiqueta `</P>` al final si no están alineados de ninguna manera. Los párrafos pueden alinearse. En este caso sí que necesitan su correspondiente etiqueta de cierre `</P>`. Son las siguientes:

```
<P ALIGN="LEFT"> Para alinear el texto con el margen izquierdo:
```

Este párrafo está alineado a la izquierda

```
<P ALIGN="CENTER"> Para centrar el texto:
```

Este párrafo está centrado.

```
<P ALIGN="RIGHT"> Para alinear el texto con el margen derecho:
```

Este párrafo está alineado a la derecha

```
<P ALIGN="JUSTIFY"> Para alinear el texto a los dos márgenes. Es ignorado por muchos navegadores, pero algunos no, como el Explorer 4.0.
```

Eso es un ejemplo el alineamiento justificado. Si se observa este breve párrafo con exploradores como el Microsoft Internet Explorer 4.0 podrás observar que los dos extremos, izquierdo y derecho, están alineados. Recuerda que para poder apreciar este efecto en los párrafos de texto, necesitas escribir más de 2 líneas.

Si se quiere hacer un salto de línea o retorno de carro, habrá que usar la etiqueta `
`. Por ejemplo:

```
<P>Esto<BR>es un párrafo
```

Se vería así:

Esto

es un párrafo

Dos
 equivalen a un <P>. Estos dos ejemplos retornan el mismo resultado:

```
<P>Esto es un párrafo<P>Esto es otro
```

```
<P>Esto es un párrafo<BR><BR>Esto es otro
```

También puede usarse la etiqueta <NOBR>...</NOBR>. Es para no permitir el salto de línea, es decir, para hacer líneas muy largas en páginas en las que haya que usar la barra de desplazamiento horizontal.

1.4.1.2 Cabeceras

Sirven para poner títulos en nuestras páginas. Van entre las etiquetas <Hx> y </Hx>, donde x es un número del 1 al 6. El 1 es la fuente de mayor tamaño y 6 la de menor. El que corresponde al tamaño estándar del texto es 3. Estos números no tienen nada que ver con el número de píxeles ni con los números de la etiqueta que veremos posteriormente.

La siguiente tabla, muestra los diferentes tipos de etiquetas con su respectivo resultado:

Etiqueta	Resultado
<H1> ... </H1>	Cabecera de nivel 1
<H2> ... </H2>	Cabecera de nivel 2
<H3> ... </H3>	Cabecera de nivel 3
<H4> ... </H4>	Cabecera de nivel 4
<H5> ... </H5>	Cabecera de nivel 5
<H6> ... </H6>	Cabecera de nivel 6

Estas etiquetas se pueden definir como de formato de párrafo pero por su importancia he preferido tratarlas aparte. No resulta recomendable utilizarlas para aumentar o disminuir el tamaño del tipo de letra, ya que cada navegador los muestra de manera diferente. Se usan para dividir correctamente en secciones nuestra página, tal y como se hace en un documento de texto normal.

1.4.1.3 Cambiando el tipo de letra

Todas estas etiquetas nos permiten cambiar de una manera u otra el aspecto del tipo de letra que estemos utilizando y se pueden utilizar con tiras de caracteres dentro de un párrafo.

Etiqueta	Utilidad	Resultado
 ... 	Pone el texto en negrita.	Soy un texto negro como el carbón
<I> ... </I>	Representa el texto en cursiva.	<i>Estoy ladeado</i>
<U> ... </U>	Para subrayar algo.	<u>Como soy muy importante estoy subrayado</u>
<S> ... </S>	Para tachar.	A mí, en cambio, nadie me quiere

1.4.1.5 Otros elementos

Por último, debemos estudiar algunas cosas que no son texto y que podemos incorporar a nuestra página.

Etiqueta	Utilidad	Resultado
<HR>	Inserta una barra horizontal.	<hr/>

	Salto de línea.	Hay un antes y un después de saltar a otra línea
<!-- ... -->	Comentarios.	Esto se escribe y

1.4.1.6 La etiqueta *Anchor*

Un vínculo tiene dos extremos (llamados en inglés *anchors*, *anclas*) y una dirección. El vínculo comienza en el "ancla de origen" (*origen del vínculo*) y apunta al "ancla destino" (*destino del vínculo*), que puede ser cualquier recurso de la Web (p.ej., una imagen, un video clip, un archivo de sonido, un programa, un documento HTML, un elemento dentro de un documento HTML, etc.).

Para incorporar un enlace hay que utilizar esta etiqueta. Todo lo que encerremos entre <A> y , ya sea texto o imágenes, será considerado como enlace y sufrirá dos modificaciones:

1. Se visualizará de manera distinta en el navegador. El texto aparecerá subrayado y de un color distinto al habitual, y las imágenes estarán rodeadas por un borde del mismo color que el del texto del enlace.
2. Al pulsar sobre el enlace, seremos enviados al documento que apuntaba el enlace.

Para que el enlace sirva para algo debemos especificarle una dirección. Lo haremos de la siguiente manera:

```
Para continuar haga clic <A HREF="direccion">aqui</A>.
```

La dirección estará en formato URL (*Uniform Resource Locator*) y lo contiene el atributo HREF.

Un método para ayudar en la navegación de una misma página es hacer enlaces dentro de ella. Esta técnica hace que los navegantes accedan de una manera más rápida a la información, sin duda muchos lo apreciarán. Consiste en hacer clic en algún texto o imagen y desplazarse a otra parte dentro de la misma página.

Para ello debemos marcar en nuestra página las diferentes secciones en las que se divide, insertando para ello los llamados “Marcadores”. Lo haremos con el parámetro “name”:

```
<A name="sección1"></A>
```

Esta instrucción marca el inicio de una sección dentro de nuestra página. La sección se llamará sección1. Para hacer un enlace que salte al marcador de esta sección dentro de nuestra página lo haríamos de la siguiente forma:

```
<A href="#sección1">Primera Parte</A>
```

1.4.1.7 Listas desordenadas

La etiqueta nos permite presentar listas de elementos sin orden alguno. Cada elemento de la lista irá normalmente precedido por un círculo. Por ejemplo,

```
<UL>
  <LI>Primer elemento
  <LI>Segundo elemento
</UL>
```

1.4.1.8 Listas ordenadas

La etiqueta nos permite presentar listas de elementos ordenados de menor a mayor. Normalmente cada elemento de la lista irá precedido por su número en el orden. Por ejemplo,

```
<OL>
  <LI>Primer elemento
  <LI>Segundo elemento
```

```
</OL>
```

1.4.1.9 Listas de definiciones

Este es el único tipo de lista que no utiliza la etiqueta . Al tener como objetivo presentar una lista de definiciones, de modo que tiene que representar de manera distinta el objeto definido y la definición. Esto se hace por medio de las etiquetas <DT> y <DD>:

```
<DL>

  <DT>Primer elemento<DD>Es un elemento muy bonito.

  <DT>Segundo elemento<DD>Este, en cambio, es peor.

</DL>
```

1.4.1.10 Imágenes

Para incluir gráficos e imágenes en nuestras páginas utilizaremos la etiqueta

```
<IMG>
```

de esta manera:

```
<IMG SRC="fichero_grafico" ALT="descripcion">
```

El parámetro SRC especifica el nombre del fichero que contiene el gráfico. Los formatos estándar en la red son el GIF y el JPG. Las últimas versiones de Netscape y Explorer aceptan también el formato PNG.

El parámetro ALT especifica el texto que se mostrará en lugar del gráfico en aquellos navegadores que no sean capaces de mostrarlos (como el Lynx) y en el supuesto de que el usuario los haya desactivado. Algunos navegadores lo muestran cuando pasamos el ratón por encima de la imagen. Es por eso que, aunque algunos usuarios no lo lleguen a ver nunca, conviene ponerlo siempre. De hecho, el estándar HTML 4.0 obliga a hacerlo.

Existen dos atributos que, aunque opcionales, conviene indicar siempre: la altura y el ancho del gráfico en pixels. De este modo, el navegador puede mostrar un recuadro del tamaño de la imagen mientras la va leyendo de la red y así poder mostrar el resto de la página correctamente mientras tanto.

```
<IMG SRC=graficos/dwnldns.gif" ALT="Netscape 4.0" WIDTH="88" HEIGHT="31">
```

1.4.2 Elementos HTML5

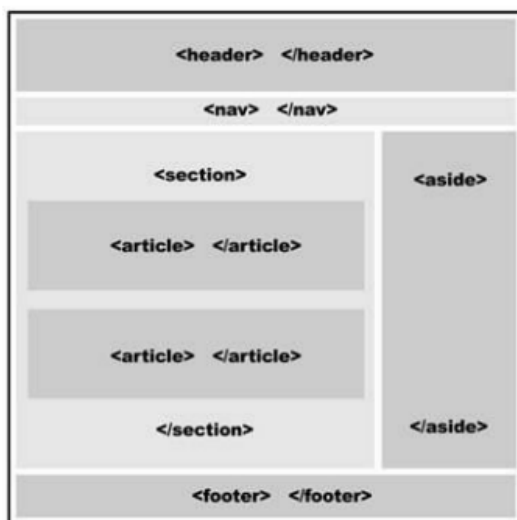
1.4.2.1 <article>

El diseño considerado anteriormente es el más común y representa una estructura esencial para los sitios web estos días, pero es además ejemplo de cómo el contenido clave es

mostrado en pantalla. Del mismo modo que los blogs están divididos en entradas, sitios web normalmente presentan información relevante dividida en partes que comparten similares características. El elemento `<article>` nos permite identificar cada una de estas partes:

```
<section>
  <article>
    Este es el texto de mi primer mensaje
  </article>
  <article>
    Este es el texto de mi segundo mensaje
  </article>
</section>
```

Como puede observarse en el código, las etiquetas `<article>` se encuentran ubicadas dentro del elemento `<section>`. Las etiquetas `<article>` en nuestro ejemplo pertenecen a esta sección, son sus hijos, del mismo modo que cada elemento dentro de las etiquetas `<body>` es hijo del cuerpo. Y al igual que cada elemento hijo del cuerpo, las etiquetas `<article>` son ubicadas una sobre otra, como es mostrado en la imagen.



El elemento `<article>` no está limitado por su nombre (no se limita, por ejemplo, a artículos de noticias). Este elemento fue creado con la intención de contener unidades independientes de contenido, por lo que puede incluir mensajes de foros, artículos de una revista digital, entradas de blog, comentarios de usuarios, etc...

Lo que hace es agrupar porciones de información que están relacionadas entre sí independientemente de su naturaleza. Como una parte independiente del documento, el contenido de cada elemento `<article>` tendrá su propia estructura.

Para definir esta estructura, podemos aprovechar la versatilidad de los elementos `<header>` y `<footer>` estudiados anteriormente. Estos elementos son portables y pueden ser usados no

solo para definir los límites del cuerpo sino también en cualquier sección de nuestro documento:

```
<section>
  <article>
    <header>
      <h1>Título del mensaje uno</h1>
    </header>
    Este es el texto de mi primer mensaje
    <footer>
      <p>comentarios (0)</p>
    </footer>
  </article>
  <article>
    <header>
      <h1>Titulo del mensaje dos</h1>
    </header>
    Este es el texto de mi segundo mensaje
    <footer>
      <p>comentarios (0)</p>
    </footer>
  </article>
</section>
```

Los dos mensajes insertados en el código fueron contruidos con el elemento `<article>` y tienen una estructura específica. En la parte superior de esta estructura incluimos las etiquetas `<header>` conteniendo el título definido con el elemento `<h1>`, debajo se encuentra el contenido mismo del mensaje y sobre el final, luego del texto, vienen las etiquetas `<footer>` especificando la cantidad de comentarios recibidos.

1.4.2.2 `<hgroup>`

Dentro de cada elemento `<header>`, en la parte superior del cuerpo o al comienzo de cada `<article>`, incorporamos elementos `<h1>` para declarar un título. Básicamente, las etiquetas `<h1>` son todo lo que necesitamos para crear una línea de cabecera para cada parte del documento, pero es normal que necesitemos también agregar subtítulos o más información que especifique de qué se trata la página web o una sección en particular.

De hecho, el elemento `<header>` fue creado para contener también otros elementos como tablas de contenido, formularios de búsqueda o textos cortos y logos. Para construir este tipo de cabeceras, podemos aprovechar el resto de las etiquetas H, como `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` y `<h6>`, pero siempre considerando que por propósitos de procesamiento interno, y para evitar generar múltiples secciones durante la interpretación del documento por parte del navegador, estas etiquetas deben ser agrupadas juntas. Por esta razón, HTML5 provee el elemento `<hgroup>`:

```
<section>
  <article>
    <header>
      <hgroup>
```

```

        <h1>Título del mensaje uno</h1>
        <h2>Subtítulo del mensaje uno</h2>
    </hgroup>
    <p>publicado 10-12-2011</p>
</header>
Este es el texto de mi primer mensaje
<footer>
    <p>comentarios (0)</p>
</footer>
</article>
<article>
    <header>
        <hgroup>
            <h1>Título del mensaje dos</h1>
            <h2>Subtítulo del mensaje dos</h2>
        </hgroup>
        <p>publicado 15-12-2011</p>
    </header>
    Este es el texto de mi segundo mensaje
    <footer>
        <p>comentarios (0)</p>
    </footer>
</article>
</section>

```

Las etiquetas H deben conservar su jerarquía, lo que significa que debemos primero declarar la etiqueta <h1>, luego usar <h2> para subtítulos y así sucesivamente. Sin embargo, a diferencia de anteriores versiones de HTML, HTML5 nos deja reusar las etiquetas H y construir esta jerarquía una y otra vez en cada sección del documento. En el ejemplo del código, agregamos un subtítulo y datos adicionales a cada mensaje. Los títulos y subtítulos fueron agrupados juntos utilizando <hgroup>, recreando de este modo la jerarquía <h1> y <h2> en cada elemento <article>.

Navegadores y programas que ejecutan y presentan en la pantalla sitios webs leen el código HTML y crean su propia estructura interna para interpretar y procesar cada elemento. Esta estructura interna está dividida en secciones que no tienen nada que ver con las divisiones en el diseño o el elemento <section>. Estas son secciones conceptuales generadas durante la interpretación del código. El elemento <header> no crea una de estas secciones por sí mismo, lo que significa que los elementos dentro de <header> representarán diferentes niveles e internamente pueden generar diferentes secciones. El elemento <hgroup> fue creado con el propósito de agrupar las etiquetas H y evitar interpretaciones incorrectas por parte de los navegadores.

1.4.2.3 <figure> y <figcaption>

La etiqueta <figure> fue creada para ayudarnos a ser aún más específicos a la hora de declarar el contenido del documento. Antes de que este elemento sea introducido, no podíamos identificar el contenido que era parte de la información pero a la vez independiente, como ilustraciones, fotos, videos, etc... Normalmente estos elementos son parte del contenido

relevante pero pueden ser extraídos o movidos a otra parte sin afectar o interrumpir el flujo del documento. Cuando nos encontramos con esta clase de información, las etiquetas `<figure>` pueden ser usadas para identificarla:

```
<article>
  <header>
    <hgroup>
      <h1>Título del mensaje uno</h1>
      <h2>Subtítulo del mensaje uno</h2>
    </hgroup>
    <p>publicado 10-12-2011</p>
  </header>
  Este es el texto de mi primer mensaje
  <figure>
    
    <figcaption>
      Esta es la imagen del primer mensaje
    </figcaption>
  </figure>
  <footer>
    <p>comentarios (0)</p>
  </footer>
</article>
```

En el código, en el primer mensaje, luego del texto insertamos una imagen. Esta es una práctica común, a menudo el texto es enriquecido con imágenes o videos. Las etiquetas `<figure>` nos permiten envolver estos complementos visuales y diferenciarlos así de la información más relevante.

También en el código se puede observar un elemento extra dentro de `<figure>`. Normalmente, unidades de información como imágenes o videos son descriptas con un corto texto debajo. HTML5 provee un elemento para ubicar e identificar estos títulos descriptivos. Las etiquetas `<figcaption>` encierran el texto relacionado con `<figure>` y establecen una relación entre ambos elementos y su contenido.

1.5 Nuevos y viejos elementos

HTML5 fue desarrollado con la intención de simplificar, especificar y organizar el código. Para lograr este propósito, nuevas etiquetas y atributos fueron agregados y HTML fue completamente integrado a CSS y Javascript. Estas incorporaciones y mejoras de versiones previas están relacionadas no solo con nuevos elementos sino también con cómo usamos los ya existentes.

1.5.1 <mark>

La etiqueta <mark> fue agregada para resaltar parte de un texto que originalmente no era considerado importante pero ahora es relevante acorde con las acciones del usuario. El ejemplo que más se ajusta a este caso es un resultado de búsqueda. El elemento <mark> resaltará la parte del texto que concuerda con el texto buscado:

```
<span>Mi <mark>coche</mark> es rojo</span>
```

Si un usuario realiza una búsqueda de la palabra “coche”, por ejemplo, los resultados podrían ser mostrados con el código. La frase del ejemplo representa los resultados de la búsqueda y las etiquetas <mark> en el medio encierran lo que era el texto buscado (la palabra “coche”). En algunos navegadores, esta palabra será resaltada con un fondo amarillo por defecto, pero siempre podemos sobrescribir estos estilos con los nuestros utilizando CSS.

En el pasado, normalmente obteníamos similares resultados usando el elemento . El agregado de <mark> tiene el objetivo de cambiar el significado y otorgar un nuevo propósito para éstos y otros elementos relacionados:

- es para indicar énfasis (reemplazando la etiqueta <i> que utilizábamos anteriormente).
- es para indicar importancia.
- <mark> es para resaltar texto que es relevante de acuerdo con las circunstancias.
- debería ser usado solo cuando no hay otro elemento más apropiado para la situación.

1.5.2 <small>

La nueva especificidad de HTML es también evidente en elementos como <small>. Previamente este elemento era utilizado con la intención de presentar cualquier texto con letra pequeña. La palabra clave referenciaba el tamaño del texto, independientemente de su significado. En HTML5, el nuevo propósito de <small> es presentar la llamada letra pequeña, como impresiones legales, descargos, etc...

```
<small>Derechos Reservados &copy; 2011 MinkBooks</small>
```

1.5.3 <cite>

Otro elemento que ha cambiado su naturaleza para volverse más específico es <cite>. Ahora las etiquetas <cite> encierran el título de un trabajo, como un libro, una película, una canción, etc...

```
<span> Amo la película <cite>Tentaciones</cite></span>
```

1.5.4 <address>

El elemento <address> es un viejo elemento convertido en un elemento estructural. No necesitamos usarlo previamente para construir nuestra plantilla, sin embargo podría ubicarse perfectamente en algunas situaciones en las que debemos presentar información de contacto relacionada con el contenido del elemento <article> o el cuerpo completo.

Este elemento debería ser incluido dentro de <footer>, como en el siguiente ejemplo:

```
<article>
  <header>
    <h1>Título del mensaje </h1>
  </header>
  Este es el texto del mensaje
  <footer>
    <address>
      <a href="http://www.jdgauchat.com">JD Gauchat</a>
    </address>
  </footer>
</article>
```

1.5.5 <time>

En cada <article> de nuestro último código, incluimos la fecha indicando cuándo el mensaje fue publicado. Para esto usamos un simple elemento <p> dentro de la cabecera (<header>) del mensaje, pero existe un elemento en HTML5 específico para este propósito. El elemento <time> nos permite declarar un texto comprensible para humanos y navegadores que representa fecha y hora:

```
<article>
  <header>
    <h1>Título del mensaje dos</h1>
    <time datetime="2011-10-12" pubdate>publicado 12-10-2011</time>
  </header>
  Este es el texto del mensaje
</article>
```


En el código, el elemento `<p>` usado en ejemplos previos fue reemplazado por el nuevo elemento `<time>` para mostrar la fecha en la que el mensaje fue publicado. El atributo `datetime` tiene el valor que representa la fecha comprensible para el navegador (timestamp). El formato de este valor deberá seguir un patrón similar al del siguiente ejemplo: 2011-10-12T12:10:45. También incluimos el atributo `pubdate`, el cual solo es agregado para indicar que el valor del atributo `datetime` representa la fecha de publicación.

1.6 Referencia rápida

En la especificación HTML5, HTML está a cargo de la estructura del documento y provee un grupo completo de nuevos elementos para este propósito. La especificación también incluye algunos elementos con la única tarea de proveer estilos.

Esta es una lista de los que son más relevantes:

- `<header>` Este elemento presenta información introductoria y puede ser aplicado en diferentes secciones del documento. Tiene el propósito de contener la cabecera de una sección pero también puede ser utilizado para agrupar índices, formularios de búsqueda, logos, etc...
- `<nav>` Este elemento indica una sección de enlaces con propósitos de navegación, como menús o índices. No todos los enlaces dentro de una página web tienen que estar dentro de un elemento `<nav>`, solo aquellos que forman partes de bloques de navegación.
- `<section>` Este elemento representa una sección general del documento. Es usualmente utilizado para construir varios bloques de contenido (por ejemplo, columnas) con el propósito de ordenar el contenido que comparte una característica específica, como capítulos o páginas de un libro, grupo de noticias, artículos, etc...
- `<aside>` Este elemento representa contenido que está relacionado con el contenido principal pero no es parte del mismo. Ejemplos pueden ser citas, información en barras laterales, publicidad, etc...
- `<footer>` Este elemento representa información adicional sobre su elemento padre. Por ejemplo, un elemento
- `<footer>` insertado al final del cuerpo proveerá información adicional sobre el cuerpo del documento, como el pie normal de una página web. Puede ser usado no solo para el cuerpo sino también para diferentes secciones dentro del cuerpo, otorgando información adicional sobre estas secciones específicas.
- `<article>` Este elemento representa una porción independiente de información relevante (por ejemplo, cada artículo de un periódico o cada entrada de un blog). El elemento `<article>` puede ser anidado y usado para mostrar una lista dentro de otra lista de ítems relacionados, como comentarios de usuarios en entradas de blogs, por ejemplo.
- `<hgroup>` Este elemento es usado para agrupar elementos H cuando la cabecera tiene múltiples niveles (por ejemplo, una cabecera con título y subtítulo).
- `<figure>` Este elemento representa una porción independiente de contenido (por ejemplo, imágenes, diagramas o videos) que son referenciadas desde el contenido principal. Esta es información que puede ser removida sin afectar el fluido del resto del contenido.
- `<figcaption>` Este elemento es utilizado para mostrar una leyenda o pequeño texto relacionado con el contenido de un elemento `<figure>`, como la descripción de una imagen.
- `<mark>` Este elemento resalta un texto que tiene relevancia en una situación en particular o que ha sido mostrado en respuesta de la actividad del usuario.
- `<small>` Este elemento representa contenido al margen, como letra pequeña (por ejemplo, descargos, restricciones legales, declaración de derechos, etc...).

- `<cite>` Este elemento es usado para mostrar el título de un trabajo (libro, película, poema, etc...).
- `<address>` Este elemento encierra información de contacto para un elemento `<article>` o el documento completo. Es recomendable que sea insertado dentro de un elemento `<footer>`.
- `<time>` Este elemento se utiliza para mostrar fecha y hora en formatos comprensibles por los usuarios y el navegador.

Capítulo 2

CSS3

2.1 CSS y HTML

Como aclaramos anteriormente, la nueva especificación de HTML (HTML5) no describe solo los nuevos elementos HTML o el lenguaje mismo. La web demanda diseño y funcionalidad, no solo organización estructural o definición de secciones. En este nuevo paradigma, HTML se presenta junto con CSS y JavaScript como un único instrumento integrado.

Ahora es momento de analizar CSS, su relevancia dentro de esta unión estratégica y su influencia sobre la presentación de documentos HTML. Oficialmente CSS nada tiene que ver con HTML5. CSS no es parte de la especificación y nunca lo fue. Este lenguaje es, de hecho, un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML. Al comienzo, atributos dentro de las etiquetas HTML proveían estilos esenciales para cada elemento, pero a medida que el lenguaje evolucionó, la escritura de códigos se volvió más compleja y HTML por sí mismo no pudo más satisfacer las demandas de diseñadores.

En consecuencia, CSS pronto fue adoptado como la forma de separar la estructura de la presentación. Desde entonces, CSS ha crecido y ganado importancia, pero siempre desarrollado en paralelo, enfocado en las necesidades de los diseñadores y apartado del proceso de evolución de HTML.

La versión 3 de CSS sigue el mismo camino, pero esta vez con un mayor compromiso. La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño. Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta es la razón por la que cada vez que mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas.

En este momento las nuevas características incorporadas en CSS3 están siendo implementadas e incluidas junto al resto de la especificación en navegadores compatibles con HTML5. En este capítulo, vamos a estudiar conceptos básicos de CSS y las nuevas técnicas de CSS3 ya disponibles para presentación y estructuración. También aprenderemos cómo utilizar los nuevos selectores y pseudo clases que hacen más fácil la selección e identificación de elementos HTML.

2.1.1 Modelos de caja

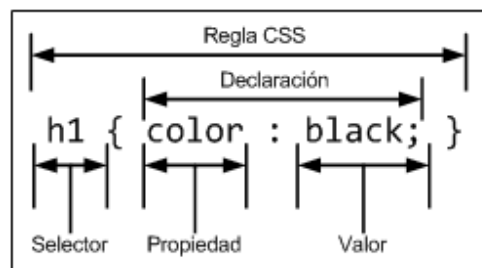
Para aprender cómo podemos crear nuestra propia organización de los elementos en pantalla, debemos primero entender cómo los navegadores procesan el código HTML. Los navegadores consideran cada elemento HTML como una caja.

Una página web es en realidad un grupo de cajas ordenadas siguiendo ciertas reglas. Estas reglas son establecidas por estilos provistos por los navegadores o por los diseñadores usando CSS, este tiene un set predeterminado de propiedades destinados a sobrescribir los estilos provistos por navegadores y obtener la organización deseada. Estas propiedades no son específicas, tienen que ser combinadas para formar reglas que luego serán usadas para agrupar cajas y obtener la correcta disposición en pantalla. La combinación de estas reglas es normalmente llamada modelo o sistema de disposición. Todas estas reglas aplicadas juntas constituyen lo que se llama un modelo de caja.

Existe solo un modelo de caja que es considerado estándar estos días, y muchos otros que aún se encuentran en estado experimental. El modelo válido y ampliamente adoptado es el llamado Modelo de Caja Tradicional, el cual ha sido usado desde la primera versión de CSS. Aunque este modelo ha probado ser efectivo, algunos modelos experimentales intentan superar sus deficiencias, pero la falta de consenso sobre el reemplazo más adecuado aún mantiene a este viejo modelo en vigencia y la mayoría de los sitios webs programados en HTML5 lo continúan utilizando.

2.1.2. Glosario básico

CSS define una serie de términos que permiten describir cada una de las partes que componen los estilos CSS. El siguiente esquema muestra las partes que forman un estilo CSS muy básico:



Los diferentes términos se definen a continuación:

- Regla: cada uno de los estilos que componen una hoja de estilos CSS. Cada regla está compuesta de una parte de "selectores", un símbolo de "llave de apertura" ({}), otra parte denominada "declaración" y por último, un símbolo de "llave de cierre" ({}).
- Selector: indica el elemento o elementos HTML a los que se aplica la regla CSS.
- Declaración: especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- Propiedad: característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
- Valor: establece el nuevo valor de la característica modificada en el elemento.

Un archivo CSS puede contener un número ilimitado de reglas CSS, cada regla se puede aplicar a varios selectores diferentes y cada declaración puede incluir tantos pares propiedad/valor como se desee.

2.2 Conceptos básicos sobre estilos

Antes de comenzar a insertar reglas CSS en nuestro archivo de estilos y aplicar un modelo de caja, debemos revisar los conceptos básicos sobre estilos CSS que van a ser utilizados en el resto del libro.

Aplicar estilos a los elementos HTML cambia la forma en que estos son presentados en pantalla. Como explicamos anteriormente, los navegadores proveen estilos por defecto que en la mayoría de los casos no son suficientes para satisfacer las necesidades de los diseñadores. Para cambiar esto, podemos sobrescribir estos estilos con los nuestros usando diferentes técnicas.

2.2.1 Estilos en línea

Una de las técnicas más simples para incorporar estilos CSS a un documento HTML es la de asignar los estilos dentro de las etiquetas por medio del atributo style.

El siguiente código muestra un documento HTML simple que contiene el elemento <p> modificado por el atributo style con el valor font-size: 20px. Este estilo cambia el tamaño por defecto del texto dentro del elemento <p> a un nuevo tamaño de 20 pixeles.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Este es el título del documento</title>
  </head>
  <body>
    <p style="font-size: 20px">Mi texto</p>
  </body>
</html>
```

Usar la técnica demostrada anteriormente es una buena manera de probar estilos y obtener una vista rápida de sus efectos, pero no es recomendado para aplicar estilos a todo el documento. La razón es simple: cuando usamos esta técnica, debemos escribir y repetir cada estilo en cada uno de los elementos que queremos modificar, incrementando el tamaño del documento a proporciones inaceptables y haciéndolo imposible de mantener y actualizar. Solo imagine lo que ocurriría si decide que en lugar de 20 pixeles el tamaño de cada uno de los elementos <p> debería ser de 24 pixeles. Tendría que modificar cada estilo en cada etiqueta <p> en el documento completo.

2.2.2 Estilos embebidos

Una mejor alternativa es insertar los estilos en la cabecera del documento y luego usar referencias para afectar los elementos HTML correspondientes:

```
<!DOCTYPE html>
<html lang="es">
<head>
```

```

        <title>Este texto es el título del documento</title>
        <style>
            p { font-size: 20px }
        </style>
    </head>
    <body>
        <p>Mi texto</p>
    </body>
</html>

```

El elemento `<style>` permite a los desarrolladores agrupar estilos CSS dentro del documento. En versiones previas de HTML era necesario especificar qué tipo de estilos serían insertados. En HTML5 los estilos por defecto son CSS, por lo tanto no necesitamos agregar ningún atributo en la etiqueta de apertura `<style>`.

El código resaltado tiene la misma función que la línea de código embebida, pero esta vez no tuvimos que escribir el estilo dentro de cada etiqueta `<p>` porque todos los elementos `<p>` ya fueron afectados. Con este método, reducimos nuestro código y asignamos los estilos que queremos a elementos específicos utilizando referencias.

2.2.3 Archivos externos

Declarar los estilos en la cabecera del documento ahorra espacio y vuelve al código más consistente y actualizable, pero nos requiere hacer una copia de cada grupo de estilos en todos los documentos de nuestro sitio web. La solución es mover todos los estilos a un archivo externo y luego utilizar el elemento `<link>` para insertar este archivo dentro de cada documento que los necesite.

Este método nos permite cambiar los estilos por completo simplemente incluyendo un archivo diferente. También nos permite modificar o adaptar nuestros documentos a cada circunstancia o dispositivo. En el Capítulo 1, estudiamos la etiqueta `<link>` y cómo utilizarla para insertar archivos con estilos CSS en nuestros documentos. Utilizando la línea `<link rel="stylesheet" href="misestilos.css">` le decimos al navegador que cargue el archivo `misestilos.css` porque contiene todos los estilos necesarios para presentar el documento en pantalla.

Esta práctica fue ampliamente adoptada por diseñadores que ya están trabajando con HTML5. La etiqueta `<link>` referenciando el archivo CSS será insertada en cada uno de los documentos que requieren de esos estilos:

```

<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Este texto es el título del documento</title>
        <link rel="stylesheet" href="misestilos.css">
    </head>
    <body>
        <p>Mi texto</p>
    </body>
</html>

```

2.3 Referencias

Almacenar todos nuestros estilos en un archivo externo e insertar este archivo dentro de cada documento que lo necesite es muy conveniente, sin embargo no podremos hacerlo sin buenos mecanismos que nos ayuden a establecer una específica relación entre estos estilos y los elementos del documento que van a ser afectados.

Cuando hablábamos sobre cómo incluir estilos en el documento, mostramos una de las técnicas utilizadas a menudo en CSS para referenciar elementos HTML. En el siguiente código, el estilo para cambiar el tamaño de la letra referenciaba cada elemento `<p>` usando la palabra clave `p`. De esta manera el estilo insertado entre las etiquetas `<style>` referenciaba cada etiqueta `<p>` del documento y asignaba ese estilo particular a cada una de ellas.

Existen varios métodos para seleccionar cuáles elementos HTML serán afectados por las reglas CSS:

- referencia por la palabra clave del elemento
- referencia por el atributo `id`
- referencia por el atributo `class`

Más tarde veremos que CSS3 es bastante flexible a este respecto e incorpora nuevas y más específicas técnicas para referenciar elementos, pero por ahora aplicaremos solo estas tres.

2.3.1 Referenciando con palabra clave

Al declarar las reglas CSS utilizando la palabra clave del elemento afectamos cada elemento de la misma clase en el documento. Por ejemplo, la siguiente regla cambiará los estilos de todos los elementos `<p>`:

```
p { font-size: 20px }
```

Esta es la técnica presentada previamente en el código. Utilizando la palabra clave `p` al frente de la regla le estamos diciendo al navegador que esta regla debe ser aplicada a cada elemento `<p>` encontrado en el documento HTML. Todos los textos envueltos en etiquetas `<p>` tendrán el tamaño de 20 píxeles.

Por supuesto, lo mismo funcionará para cualquier otro elemento HTML. Si especificamos la palabra clave `span` en lugar de `p`, por ejemplo, cada texto entre etiquetas `` tendrá un tamaño de 20 píxeles:

```
span { font-size: 20px }
```


¿Pero qué ocurre si solo necesitamos referenciar una etiqueta específica? ¿Debemos usar nuevamente el atributo style dentro de esta etiqueta? La respuesta es no. Como aprendimos anteriormente, el método de Estilos en Línea (usando el atributo style dentro de etiquetas HTML) es una técnica en desuso y debería ser evitada. Para seleccionar un elemento HTML específico desde las reglas de nuestro archivo CSS, podemos usar dos atributos diferentes: id y class.

2.3.2 Referenciando con el atributo id

El atributo id es como un nombre que identifica al elemento. Esto significa que el valor de este atributo no puede ser duplicado. Este nombre debe ser único en todo el documento. Para referenciar un elemento en particular usando el atributo id desde nuestro archivo CSS la regla debe ser declarada con el símbolo # al frente del valor que usamos para identificar el elemento:

```
#texto1 { font-size: 20px }
```

La regla en el código será aplicada al elemento HTML identificado con el atributo id="texto1". Ahora nuestro código HTML lucirá de esta manera:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Este texto es el título del documento</title>
    <link rel="stylesheet" href="misestilos.css">
  </head>
  <body>
    <p id="texto1">Mi texto</p>
  </body>
</html>
```

El resultado de este procedimiento es que cada vez que hacemos una referencia usando el identificador texto1 en nuestro archivo CSS, el elemento con ese valor de id será modificado, pero el resto de los elementos <p>, o cualquier otro elemento en el mismo documento, no serán afectados.

Esta es una forma extremadamente específica de referenciar un elemento y es normalmente utilizada para elementos más generales, como etiquetas estructurales. El atributo id y su especificidad es de hecho más apropiado para referencias en Javascript, como veremos en más adelante.

2.3.3 Referenciando con el atributo class

La mayoría del tiempo, en lugar de utilizar el atributo id para propósitos de estilos es mejor utilizar class. Este atributo es más flexible y puede ser asignado a cada elemento HTML en el documento que comparte un diseño similar:

```
.texto1 { font-size: 20px }
```

Para trabajar con el atributo class, debemos declarar la regla CSS con un punto antes del nombre. La ventaja de este método es que insertar el atributo class con el valor texto1 será suficiente para asignar estos estilos a cualquier elemento que queramos:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <p class="texto1">Mi texto</p>
  <p class="texto1">Mi texto</p>
  <p>Mi texto</p>
</body>
</html>
```

Los elementos <p> en las primeras dos líneas dentro del cuerpo del código en el código tienen el atributo class con el valor texto1. Como dijimos previamente, la misma regla puede ser aplicada a diferentes elementos en el mismo documento. Por lo tanto, estos dos primeros elementos comparten la misma regla y ambos serán afectados por el estilo especificado en el documento css. El último elemento <p> conserva los estilos por defecto otorgados por el navegador.

La razón por la que debemos utilizar un punto delante del nombre de la regla es que es posible construir referencias más complejas. Por ejemplo, se puede utilizar el mismo valor para el atributo class en diferentes elementos pero asignar diferentes estilos para cada tipo:

```
p.texto1 { font-size: 20px }
```

En el código creamos una regla que referencia la clase llamada texto1 pero solo para los elementos de tipo <p>. Si cualquier otro elemento tiene el mismo valor en su atributo class no será modificado por esta regla en particular.

2.3.4 Referenciando con cualquier atributo

Aunque los métodos de referencia estudiados anteriormente cubren un variado espectro de situaciones, a veces no son suficientes para encontrar el elemento exacto. La última versión de CSS ha incorporado nuevas formas de referenciar elementos HTML. Uno de ellas es el Selector de Atributo. Ahora podemos referenciar un elemento no solo por los atributos id y class sino también a través de cualquier otro atributo:

```
p[name] { font-size: 20px }
```

La regla en el código cambia solo elementos <p> que tienen un atributo llamado name. Para imitar lo que hicimos previamente con los atributos id y class, podemos también especificar el valor del atributo:

```
p[name="mitexto"] { font-size: 20px }
```

CSS3 permite combinar “=” con otros para hacer una selección más específica:

```
p[name^="mi"] { font-size: 20px }  
p[name$="mi"] { font-size: 20px }  
p[name*="mi"] { font-size: 20px }
```

Si usted conoce Expresiones Regulares desde otros lenguajes como Javascript o PHP, podrá reconocer los selectores utilizados en el código. En CSS3 estos selectores producen similares resultados:

- La regla con el selector ^= será asignada a todo elemento <p> que contiene un atributo name con un valor comenzado en “mi” (por ejemplo, “mitexto”, “micasa”).
- La regla con el selector \$= será asignada a todo elemento <p> que contiene un atributo name con un valor finalizado en “mi” (por ejemplo “textomi”, “casami”).
- La regla con el selector *= será asignada a todo elemento <p> que contiene un atributo name con un valor que incluye el texto “mi” (en este caso, el texto podría también encontrarse en el medio, como en “textomicasa”).

En estos ejemplos usamos el elemento <p>, el atributo name, y una cadena de texto al azar como “mi”, pero la misma técnica puede ser utilizada con cualquier atributo y valor que necesitemos. Solo tiene que escribir los corchetes e insertar entre ellos el nombre del atributo y el valor que necesita para referenciar el elemento HTML correcto.

2.3.5 Referenciando con pseudo clases

CSS3 también incorpora nuevas pseudo clases que hacen la selección aún más específica.

```
<!DOCTYPE html>  
<html lang="es">  
  <head>  
    <title>Este texto es el título del documento</title>  
    <link rel="stylesheet" href="misestilos.css">  
  </head>  
  <body>  
    <div id="wrapper">  
      <p class="mitexto1">Mi texto1</p>  
      <p class="mitexto2">Mi texto2</p>  
      <p class="mitexto3">Mi texto3</p>  
      <p class="mitexto4">Mi texto4</p>  
    </div>  
  </body>  
</html>
```

Miremos por un momento el nuevo código HTML del código. Contiene cuatro elementos <p> que, considerando la estructura HTML, son hermanos entre sí e hijos del mismo elemento <div>. Usando pseudo clases podemos aprovechar esta organización y referenciar un elemento específico sin importar cuánto conocemos sobre sus atributos y el valor de los mismos:

```
p:nth-child(2){
background: #999999;
}
```

La pseudo clase es agregada usando dos puntos luego de la referencia y antes del su nombre. En la regla del código referenciamos solo elementos <p>. Esta regla puede incluir otras referencias. Por ejemplo, podríamos escribirla como .mi clase:nth-child(2) para referenciar todo elemento que es hijo de otro elemento y tiene el valor de su atributo class igual a mi clase. La pseudo clase puede ser aplicada a cualquier tipo de referencia estudiada previamente.

La pseudo clase nth-child() nos permite encontrar un hijo específico. Como ya explicamos, el documento HTML del código anterior tiene cuatro elementos <p> que son hermanos. Esto significa que todos ellos tienen el mismo padre que es el elemento <div>. Lo que esta pseudo clase está realmente indicando es algo como: “el hijo en la posición...” por lo que el número entre paréntesis será el número de la posición del hijo, o índice. La regla está referenciando cada segundo elemento <p> encontrado en el documento.

Usando este método de referencia podemos, por supuesto, seleccionar cualquier hijo que necesitemos cambiando el número de índice. Por ejemplo, la siguiente regla tendrá impacto sólo sobre el último elemento <p> de nuestra plantilla:

```
p:nth-child(4){
background: #999999;
}
```

Como seguramente se habrá dado cuenta, es posible asignar estilos a todos los elementos creando una regla para cada uno de ellos:

```
{
margin: 0px;
}
p:nth-child(1){
background: #999999;
}
p:nth-child(2){
background: #CCCCCC;
}
p:nth-child(3){
background: #999999;
}
p:nth-child(4){
background: #CCCCCC;
}
```

La primera regla del código usa el selector universal * para asignar el mismo estilo a cada elemento del documento. Este nuevo selector representa cada uno de los elementos en el cuerpo del documento y es útil cuando necesitamos establecer ciertas reglas básicas. En este caso, configuramos el margen de todos los elementos en 0 pixeles para evitar espacios en blanco o líneas vacías como las creadas por el elemento <p> por defecto.

En el resto del código usamos la pseudo clase `nth-child()` para generar un menú o lista de opciones que son diferenciadas claramente en la pantalla por

Para agregar más opciones al menú, podemos incorporar nuevos elementos `<p>` en el código HTML y nuevas reglas con la pseudo clase `nth-child()` usando el número de índice adecuado. Sin embargo, esta aproximación genera mucho código y resulta imposible de aplicar en sitios webs con contenido dinámico. Una alternativa para obtener el mismo resultado es aprovechar las palabras clave `odd` y `even` disponibles para esta pseudo clase:

```
*{
margin: 0px;
}
p:nth-child(odd){
background: #999999;
}
p:nth-child(even){
background: #CCCCCC;
}
```

Ahora solo necesitamos dos reglas para crear la lista completa. Incluso si más adelante agregamos otras opciones, los estilos serán asignados automáticamente a cada una de ellas de acuerdo a su posición. La palabra clave `odd` para la pseudo clase `nth-child()` afecta los elementos `<p>` que son hijos de otro elemento y tienen un índice impar. La palabra clave `even`, por otro lado, afecta a aquellos que tienen un índice par.

Existen otras importantes pseudo clases relacionadas con esta última, como `first-child`, `last-child` y `only-child`, algunas de ellas recientemente incorporadas. La pseudo clase `first-child` referencia solo el primer hijo, `last-child` referencia solo el último hijo, y `only-child` afecta un elemento siempre y cuando sea el único hijo disponible. Estas pseudo clases en particular no requieren palabras clave o parámetros, y son implementadas como en el siguiente ejemplo:

```
*{
margin: 0px;
}
p:last-child{
background: #999999;
}
```

Otra importante pseudo clase llamada `not()` es utilizada para realizar una negación:

```
:not(p){
margin: 0px;
}
```

La regla del código asignará un margen de 0 píxeles a cada elemento del documento excepto los elementos `<p>`. A diferencia del selector universal utilizado previamente, la pseudo clase `not()` nos permite declarar una excepción. Los estilos en la regla creada con esta pseudo clase serán asignados a todo elemento excepto aquellos incluidos en la referencia entre paréntesis. En lugar de la palabra clave de un elemento podemos usar cualquier otra referencia que deseemos. En el próximo listado, por ejemplo, todos los elementos serán afectados excepto aquellos con el valor `mitexto2` en el atributo `class`:

```
:not(.mitexto2){  
margin: 0px;  
}
```

Cuando aplicamos la última regla al código HTML, el navegador asigna los estilos por defecto al elemento <p> identificado con el atributo class y el valor mitexto2 y provee un margen de 0 pixeles al resto.

2.3.6 Nuevos selectores

Hay algunos selectores más que fueron agregados o que ahora son considerados parte de CSS3 y pueden ser útiles para nuestros diseños. Estos selectores usan los símbolos >, + y ~ para especificar la relación entre elementos.

```
div > p.mitexto2{  
color: #990000;  
}
```

El selector > está indicando que el elemento a ser afectado por la regla es el elemento de la derecha cuando tiene al de la izquierda como su padre. La regla en el Listado 2-22 modifica los elementos <p> que son hijos de un elemento <div>. En este caso, fuimos bien específicos y preferenciamos solamente el elemento <p> con el valor mitexto2 en su atributo class.

El próximo ejemplo construye un selector utilizando el símbolo +. Este selector referencia al elemento de la derecha cuando es inmediatamente precedido por el de la izquierda. Ambos elementos deben compartir el mismo padre:

```
p.mitexto2 + p{  
color: #990000;  
}
```

La regla del código afecta al elemento <p> que se encuentra ubicado luego de otro elemento <p> identificado con el valor mitexto2 en su atributo class.

El último selector que estudiaremos es el construido con el símbolo ~. Este selector es similar al anterior pero el elemento afectado no necesita estar precediendo de inmediato al elemento de la izquierda. Además, más de un elemento puede ser afectado:

```
p.mitexto2 ~ p{  
color: #990000;  
}
```

La regla del código afecta al tercer y cuarto elemento <p> de nuestra plantilla de ejemplo. El estilo será aplicado a todos los elementos <p> que son hermanos y se encuentran luego del elemento <p> identificado con el valor mitexto2 en su atributo class. No importa si otros elementos se encuentran intercalados, los elementos <p> en la tercera y cuarta posición aún serán afectados. Puede verificar esto último insertando un elemento mitexto

luego del elemento <p> que tiene el valor mitexto2 en su atributo class. A pesar de este cambio solo los elementos <p> serán modificados por esta regla.

2.4 Referencia rápida

En HTML5 la responsabilidad por la presentación de la estructura en pantalla está más que nunca en manos de CSS. Incorporaciones y mejoras se han hecho en la última versión de CSS para proveer mejores formas de organizar documentos y trabajar con sus elementos.

2.4.1 Selector de atributo y pseudo clases

CSS3 incorpora nuevos mecanismos para referenciar elementos HTML.

2.4.1.1 Selector de Atributo

Ahora podemos utilizar otros atributos además de id y class para encontrar elementos en el documento y asignar estilos. Con la construcción `palabraclave[atributo=valor]`, podemos referenciar un elemento que tiene un atributo particular con un valor específico. Por ejemplo, `p[name="texto"]` referenciará cada elemento `<p>` con un atributo llamado name y el valor "texto". CSS3 también provee técnicas para hacer esta referencia aún más específica. Usando las siguientes combinaciones de símbolos `^=`, `$=` y `*=` podemos encontrar elementos que comienzan con el valor provisto, elementos que terminan con ese valor y elementos que tienen el texto provisto en alguna parte del valor del atributo. Por ejemplo, `p[name^="texto"]` será usado para encontrar elementos `<p>` que tienen un atributo llamado name con un valor que comienza por "texto".

2.4.1.2 Pseudo Clase :nth-child()

Esta pseudo clase encuentra un hijo específico siguiendo la estructura de árbol de HTML. Por ejemplo, con el estilo `span:nth-child(2)` estamos referenciando el elemento `` que tiene otros elementos `` como hermanos y está localizado en la posición 2. Este número es considerado el índice. En lugar de un número podemos usar las palabras clave odd y even para referenciar elementos con un índice impar o par respectivamente (por ejemplo, `span:nth-child(odd)`).

2.4.1.3 Pseudo Clase :first-child

Esta pseudo clase es usada para referenciar el primer hijo, similar a `:nth-child(1)`.

2.4.1.4 Pseudo Clase :last-child

Esta pseudo clase es usada para referenciar el último hijo.

2.4.1.5 Pseudo Clase :only-child

Esta pseudo clase es usada para referenciar un elemento que es el único hijo disponible de un mismo elemento padre.

2.4.1.6 Pseudo Clase :not()

Esta pseudo clase es usada para referenciar todo elemento excepto el declarado entre paréntesis.

2.4.2 Selectores

CSS3 también incorpora nuevos selectores que ayudan a llegar a elementos difíciles de referenciar utilizando otras técnicas.

2.4.2.1 Selector >

Este selector referencia al elemento de la derecha cuando tiene el elemento de la izquierda como padre. Por ejemplo, `div > p` referenciará cada elemento `<p>` que es hijo de un elemento `<div>`.

2.4.2.2 Selector +

Este selector referencia elementos que son hermanos. La referencia apuntará al elemento de la derecha cuando es inmediatamente precedido por el de la izquierda. Por ejemplo, `span + p` afectará a los elementos `<p>` que son hermanos y están ubicados luego de un elemento ``.

2.4.2.3 Selector ~

Este selector es similar al anterior, pero en este caso el elemento de la derecha no tiene que estar ubicado inmediatamente después del de la izquierda.

2.5 Propiedades del CSS3

La web cambió para siempre cuando unos años atrás nuevas aplicaciones desarrolladas sobre implementaciones Ajax mejoraron el diseño y la experiencia de los usuarios. La versión 2.0, asignada a la web para describir un nuevo nivel de desarrollo, representó un cambio no solo en la forma en que la información era transmitida sino también en cómo los sitios web y nuevas aplicaciones eran diseñados y construidos.

Los códigos implementados en esta nueva generación de sitios web pronto se volvieron estándar. La innovación se volvió tan importante para el éxito de cualquier proyecto en Internet que programadores desarrollaron librerías completas para superar las limitaciones y satisfacer los nuevos requerimientos de los diseñadores.

La falta de soporte por parte de los navegadores era evidente, pero la organización responsable de los estándares web no tomó las tendencias muy seriamente e intentó seguir su propio camino. Afortunadamente, algunas mentes brillantes siguieron desarrollando nuevos estándares en paralelo y pronto HTML5 nació. Luego del retorno de la calma (y algunos acuerdos de por medio), la integración entre HTML, CSS y Javascript bajo la tutela de HTML5 fue como el caballero bravo y victorioso que dirige las tropas hacia el palacio enemigo.

A pesar de la reciente agitación, esta batalla comenzó mucho tiempo atrás, con la primera especificación de la tercera versión de CSS. Cuando finalmente, alrededor del año 2005, esta tecnología fue oficialmente considerada estándar, CSS estaba listo para proveer las funciones requeridas por desarrolladores (aquellas que programadores habían creado desde años atrás usando códigos Javascript complicados de implementar y no siempre compatibles).

En un intento por reducir el uso de código Javascript y para estandarizar funciones populares, CSS3 no solo cubre diseño y estilos web sino también forma y movimiento. La especificación de CSS3 es presentada en módulos que permiten a la tecnología proveer una especificación estándar por cada aspecto involucrado en la presentación visual del documento. Desde esquinas redondeadas y sombras hasta transformaciones y reposicionamiento de los elementos ya presentados en pantalla, cada posible efecto aplicado previamente utilizando Javascript fue cubierto.

Vamos a generar la siguiente plantilla para estudiar cada una de las propiedades del CSS3:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Nuevos Estilos CSS3</title>
<link rel="stylesheet" href="nuevocss3.css">
</head>
<body>
<header id="principal">
<span id="titulo">Estilos CSS Web 2.0</span>
</header>
</body>
</html>
```

2.5.1 Compatibilidad

En este momento, algunas pocas propiedades del CSS3, se encuentra en estado experimental en algunos navegadores. Para aplicarla efectivamente a sus documentos, debe declararla con los correspondientes prefijos. Los prefijos para los navegadores más comunes son los siguientes:

- -moz- para Firefox.
- -webkit- para Safari y Chrome.
- -o- para Opera.
- -khtml- para Konqueror.
- -ms- para Internet Explorer.
- -chrome- específico para Google Chrome.
-

2.5.2 Border-radius

Por muchos años diseñadores han sufrido intentando lograr el efecto de esquinas redondeadas en las cajas de sus páginas web. El proceso era casi siempre frustrante y extenuante. Todos lo padecieron alguna vez. Esta es la razón por la que, entre todas las nuevas posibilidades e increíbles propiedades incorporadas en CSS3, la que exploraremos en primera instancia es border-radius:

```
body {
    text-align: center;
}
#principal {
    display: block;
    width: 500px;
    margin: 50px auto;
    padding: 15px;
    text-align: center;
    border: 1px solid #999999;
    background: #DDDDDD;
    -moz-border-radius: 20px;
    -webkit-border-radius: 20px;
    border-radius: 20px;
}
#titulo {
    font: bold 36px verdana, sans-serif;
}
```

La propiedad border-radius en este momento es experimental por lo que debemos usar los prefijos -moz- y -webkit- para que funcionen en navegadores basados en motores Gecko y WebKit, como Firefox, Safari y Google Chrome. Si todas las esquinas tienen la misma curvatura podemos utilizar un solo valor. Sin embargo, como ocurre con las propiedades margin y padding, podemos también declarar un valor diferente por cada una:

```
#principal {
```

```

display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-border-radius: 20px 10px 30px 50px;
-webkit-border-radius: 20px 10px 30px 50px;
border-radius: 20px 10px 30px 50px;
}

```

Como puede ver en el código, los cuatro valores asignados a la propiedad `border-radius` representan diferentes ubicaciones. Recorriendo la caja en dirección de las agujas del reloj, los valores se aplicarán en el siguiente orden: esquina superior izquierda, esquina superior derecha, esquina inferior derecha y esquina inferior izquierda. Los valores son siempre dados en dirección de las agujas del reloj, comenzando por la esquina superior izquierda.

Al igual que con `margin` o `padding`, `border-radius` puede también trabajar solo con dos valores. El primer valor será asignado a la primera y tercera esquina (superior izquierda, inferior derecha), y el segundo valor a la segunda y cuarta esquina (superior derecha, inferior izquierda).

También podemos dar forma a las esquinas declarando un segundo grupo de valores separados por una barra. Los valores a la izquierda de la barra representarán el radio horizontal mientras que los valores a la derecha representan el radio vertical. La combinación de estos valores genera una elipsis:

```

#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-border-radius: 20px / 10px;
-webkit-border-radius: 20px / 10px;
border-radius: 20px / 10px;
}

```

2.5.3 Box-shadow

Ahora que finalmente contamos con la posibilidad de generar bonitas esquinas para nuestras cajas podemos arriesgarnos con algo más. Otro muy buen efecto, que había sido extremadamente complicado de lograr hasta este momento, es sombras. Por años diseñadores han combinado imágenes, elementos y algunas propiedades CSS para generar sombras. Gracias a CSS3 y a la nueva propiedad `box-shadow` podremos aplicar sombras a nuestras cajas con solo una simple línea de código:

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;
  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
  border-radius: 20px;
  -moz-box-shadow: rgb(150,150,150) 5px 5px;
  -webkit-box-shadow: rgb(150,150,150) 5px 5px;
  box-shadow: rgb(150,150,150) 5px 5px;
}
```

La propiedad box-shadow necesita al menos tres valores. El primero, es el color. Este valor fue construido aquí utilizando la función rgb() y números decimales, pero podemos escribirlo en números hexadecimales también. Los siguientes dos valores, expresados en pixeles, establecen el desplazamiento de la sombra. Este desplazamiento puede ser positivo o negativo. Los valores indican, respectivamente, la distancia horizontal y vertical desde la sombra al elemento. Valores negativos posicionarán la sombra a la izquierda y arriba del elemento, mientras que valores positivos crearán la sombra a la derecha y debajo del elemento. Valores de 0 o nulos posicionarán la sombra exactamente detrás del elemento, permitiendo la posibilidad de crear un efecto difuminado a todo su alrededor.

La sombra que obtuvimos hasta el momento es sólida, sin gradientes o transparencias (no realmente como una sombra suele aparecer). Existen algunos parámetros más y cambios que podemos implementar para mejorar la apariencia de la sombra.

Un cuarto valor que se puede agregar a la propiedad ya estudiada es la distancia de difuminación. Con este efecto ahora la sombra lucirá real. Puede intentar utilizar este nuevo parámetro declarando un valor de 10 pixeles, como en el siguiente ejemplo:

```
box-shadow: rgb(150,150,150) 5px 5px 10px;
```

Agregando otro valor más en pixeles al final de la propiedad desparramará la sombra. Este efecto cambia un poco la naturaleza de la sombra expandiendo el área que cubre. A pesar de que no recomendamos utilizar este efecto, puede ser aplicable en algunos diseños.

El último valor posible para box-shadow no es un número sino más bien una palabra clave: inset. Esta palabra clave convierte a la sombra externa en una sombra interna, lo cual provee un efecto de profundidad al elemento afectado.

```
box-shadow: rgb(150,150,150) 5px 5px 10px inset;
```

El estilo mostrará una sombra interna alejada del borde de la caja por unos 5 pixeles y con un efecto de difuminación de 10 pixeles.

2.5.4 Text-shadow

Ahora que conoce todo acerca de sombras probablemente estará pensando en generar una para cada elemento de su documento. La propiedad box-shadow fue diseñada especialmente para ser aplicada en cajas. Si intenta aplicar este efecto a un elemento , por ejemplo, la caja invisible ocupada por este elemento en la pantalla tendrá una sombra, pero no el contenido del elemento. Para crear sombras para figuras irregulares como textos, existe una propiedad especial llamada text-shadow:

```
...
#titulo {
    font: bold 36px verdana, sans-serif;
    text-shadow: rgb(0,0,150) 3px 3px 5px;
}
```

Los valores para text-shadow son similares a los usados para box-shadow. Podemos declarar el color de la sombra, la distancia horizontal y vertical de la sombra con respecto al objeto y el radio de difuminación.

En el código anterior una sombra azul fue aplicada al título de nuestra plantilla con una distancia de 3 pixeles y un radio de difuminación de 5.

2.5.5 @font-face

Obtener un texto con sombra es realmente un muy buen truco de diseño, imposible de lograr con métodos previos, pero más que cambiar el texto en sí mismo solo provee un efecto tridimensional. Una sombra, en este caso, es como pintar un viejo coche, al final será el mismo coche. En este caso, será el mismo tipo de letra.

El problema con las fuentes o tipos de letra es tan viejo como la web. Usuarios regulares de la web a menudo tienen un número limitado de fuentes instaladas en sus ordenadores, usualmente estas fuentes son diferentes de un usuario a otro, y la mayoría de las veces muchos usuarios tendrán fuentes que otros no. Por años, los sitios webs solo pudieron utilizar un limitado grupo de fuentes confiables (un grupo básico que prácticamente todos los usuarios tienen instalados) y así presentar la información en pantalla.

La propiedad @font-face permite a los diseñadores proveer un archivo conteniendo una fuente específica para mostrar sus textos en la página. Ahora podemos incluir cualquier fuente que necesitemos con solo proveer el archivo adecuado:

```
#titulo {
    font: bold 36px MiNuevaFuente, verdana, sans-serif;
    text-shadow: rgb(0,0,150) 3px 3px 5px;
}
@font-face {
    font-family: 'MiNuevaFuente';
    src: url('font.ttf');
}
```

La propiedad `@font-face` necesita al menos dos estilos para declarar la fuente y cargar el archivo. El estilo construido con la propiedad `font-family` especifica el nombre que queremos otorgar a esta fuente en particular, y la propiedad `src` indica la URL del archivo con el código correspondiente a esa fuente. En el código, el nombre `MiNuevaFuente` fue asignado a nuestro nuevo tipo de letra y el archivo `font.ttf` fue indicado como el archivo correspondiente a esta fuente.

Una vez que la fuente es cargada, podemos comenzar a usarla en cualquier elemento del documento simplemente escribiendo su nombre (`MiNuevaFuente`) especificamos que el título será mostrado con la nueva fuente o las alternativas `verdana` y `sans-serif` en caso de que la fuente incorporada no sea cargada apropiadamente.

2.5.6 Gradiente lineal

Los gradientes son uno de los efectos más atractivos entre aquellos incorporados en CSS3. Este efecto era prácticamente imposible de implementar usando técnicas anteriores pero ahora es realmente fácil de hacer usando CSS. Una propiedad `background` con algunos pocos parámetros es suficiente para convertir su documento en una página web con aspecto profesional:

```
#principal {
    display: block;
    width: 500px;
    margin: 50px auto;
    padding: 15px;
    text-align: center;
    border: 1px solid #999999;
    background: #DDDDDD;
    -moz-border-radius: 20px;
    -webkit-border-radius: 20px;
    border-radius: 20px;
    -moz-box-shadow: rgb(150,150,150) 5px 5px 10px;
    -webkit-box-shadow: rgb(150,150,150) 5px 5px 10px;
    box-shadow: rgb(150,150,150) 5px 5px 10px;
    background: -webkit-linear-gradient(top, #FFFFFF, #006699);
    background: -moz-linear-gradient(top, #FFFFFF, #006699);
}
```

Los gradientes son configurados como fondos, por lo que podemos usar las propiedades `background` o `backgroundimage` para declararlos. La sintaxis para los valores declarados en estas propiedades es `linear-gradient` (posición inicio, color inicial, color final). Los atributos de la función `linear-gradient()` indican el punto de comienzo y los colores usados para crear el gradiente. El primer valor puede ser especificado en pixeles, porcentaje o usando las palabras clave `top`, `bottom`, `left` y `right` (como hicimos en nuestro ejemplo). El punto de comienzo puede ser reemplazado por un ángulo para declarar una dirección específica del gradiente:

```
background: linear-gradient(30deg, #FFFFFF, #006699);
```

También podemos declarar los puntos de terminación para cada color:

```
background: linear-gradient(top, #FFFFFF 50%, #006699 90%);
```

2.5.7 Gradiente radial

La sintaxis estándar para los gradientes radiales solo difiere en unos pocos aspectos con respecto a la anterior. Debemos usar la función `radial-gradient()` y un nuevo atributo para la forma:

```
background: radial-gradient (center, circle, #FFFFFF 0%, #006699 200%);
```

La posición de comienzo es el origen y puede ser declarada en píxeles, porcentaje o una combinación de las palabras clave `center`, `top`, `bottom`, `left` y `right`. Existen dos posibles valores para la forma (`circle` y `ellipse`) y la terminación para el color indica el color y la posición donde las transiciones comienzan.

2.5.8 RGBA

Hasta este momento los colores fueron declarados como sólidos utilizando valores hexadecimales o la función `rgb()` para decimales. CSS3 ha agregado una nueva función llamada `rgba()` que simplifica la asignación de colores y transparencias. Esta función además resuelve un problema previo provocado por la propiedad `opacity`.

La función `rgba()` tiene cuatro atributos. Los primeros tres son similares a los usados en `rgb()` y simplemente declaran los valores para los colores rojo, verde y azul en números decimales del 0 al 255. El último, en cambio, corresponde a la nueva capacidad de opacidad. Este valor se debe encontrar dentro de un rango que va de 0 a 1, con 0 como totalmente transparente y 1 como totalmente opaco.

```
#titulo {  
    font: bold 36px MiNuevaFuente, verdana, sans-serif;  
    text-shadow: rgba(0,0,0,0.5) 3px 3px 5px;  
}
```

Este ejemplo demuestra cómo los efectos son mejorados aplicando transparencia. Reemplazamos la función `rgb()` por `rgba()` en la sombra del título y agregamos un valor de opacidad/transparencia de 0.5. Ahora la sombra de nuestro título se mezclará con el fondo, creando un efecto mucho más natural.

2.5.9 HSLA

Del mismo modo que la función `rgba()` agrega un valor de opacidad a `rgb()`, la función `hsla()` hace lo mismo para la función `hsl()`.

La función `hsla()` es simplemente un función diferente para generar colores, pero es más intuitiva que `rgba()`. Algunos diseñadores encontrarán más fácil generar un set de colores personalizado utilizando `hsla()`. La sintaxis de esta función es: `hsla(tono, saturación, luminosidad, opacidad)`.

```
#titulo {  
    font: bold 36px MiNuevaFuente, verdana, sans-serif;  
    text-shadow: rgba(0,0,0,0.5) 3px 3px 5px;  
    color: hsla(120, 100%, 50%, 0.5);  
}
```

Siguiendo la sintaxis, tono representa el color extraído de una rueda imaginaria y es expresado en grados desde 0 a 360. Cerca de 0 y 360 están los colores rojos, cerca de 120 los verdes y cerca de 240 los azules. El valor saturación es representado en porcentaje, desde 0% (escala de grises) a 100% (todo color o completamente saturado). La luminosidad es también un valor en porcentaje desde 0% (completamente oscuro) a 100% (completamente iluminado). El valor 50% representa luminosidad normal o promedio. El último valor, así como en `rgba()`, representa la opacidad.

2.5.10 Outline

La propiedad `outline` es una vieja propiedad CSS que ha sido expandida en CSS3 para incluir un valor de desplazamiento. Esta propiedad era usada para crear un segundo borde, y ahora ese borde puede ser mostrado alejado del borde real del elemento.

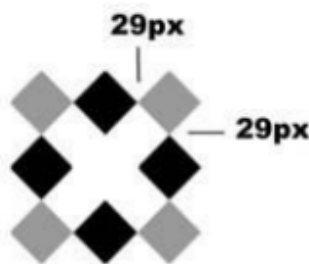
```
#principal {  
    display: block;  
    width: 500px;  
    margin: 50px auto;  
    padding: 15px;  
    text-align: center;  
    border: 1px solid #999999;  
    background: #DDDDDD;  
    outline: 2px dashed #000099;  
    outline-offset: 15px;  
}
```

Agregamos a los estilos originalmente aplicados a la caja de nuestra plantilla un segundo borde de 2 píxeles con un desplazamiento de 15 píxeles. La propiedad `outline` tiene similares características y usa los mismos parámetros que `border`. La propiedad `outline-offset` solo necesita un valor en píxeles.

2.5.11 Border-image

Los posibles efectos logrados por las propiedades `border` y `outline` están limitados a líneas simples y solo algunas opciones de configuración. La nueva propiedad `border-image` fue incorporada para superar estas limitaciones y dejar en manos del diseñador la calidad y variedad de bordes disponibles ofreciendo la alternativa de utilizar imágenes propias.

La propiedad `border-image` toma una imagen y la utiliza como patrón. De acuerdo a los valores otorgados, la imagen es cortada como un pastel, las partes obtenidas son luego ubicadas alrededor del objeto para construir el borde.



Para hacer el trabajo, necesitamos especificar tres atributos: el nombre del archivo de la imagen, el tamaño de las piezas que queremos obtener del patrón y algunas palabras clave para declarar cómo las piezas serán distribuidas alrededor del objeto.

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 29px;
  -moz-border-image: url("diamonds.png") 29 stretch;
  -webkit-border-image: url("diamonds.png") 29 stretch;
  border-image: url("diamonds.png") 29 stretch;
}
```

Con las modificaciones realizadas estamos definiendo un borde de 29 píxeles para la caja de nuestra cabecera y luego cargando la imagen `diamonds.png` para construir ese borde. El valor 29 en la propiedad `border-image` declara el tamaño de las piezas y `stretch` es uno de los métodos disponibles para distribuir estas piezas alrededor de la caja.

Existen tres valores posibles para el último atributo. La palabra clave `repeat` repetirá las piezas tomadas de la imagen todas las veces que sea necesario para cubrir el lado del elemento. En este caso, el tamaño de las piezas es preservado y la imagen será cortada si no existe más espacio para ubicarla. La palabra clave `round` considerará qué tan largo es el lado a ser cubierto y ajustará el tamaño de las piezas para asegurarse que cubren todo el lado y ninguna pieza es cortada. Finalmente, la palabra clave `stretch` estira solo una pieza para cubrir el lado completo.

En nuestro ejemplo utilizamos la propiedad `border` para definir el tamaño del borde, pero se puede también usar `border-with` para especificar diferentes tamaños para cada lado del elemento (la propiedad `border-with` usa cuatro parámetros, con una sintaxis similar a `margin` y `padding`). Lo mismo ocurre con el tamaño de cada pieza, hasta cuatro valores pueden ser declarados para obtener diferentes imágenes de diferentes tamaños desde el patrón.

2.5.12 Transform y transition

Los elementos HTML, cuando son creados, son como bloques sólidos e inamovibles. Pueden ser movidos usando código Javascript o aprovechando librerías populares como jQuery (www.jquery.com), por ejemplo, pero no existía un procedimiento estándar para este propósito hasta que CSS3 presentó las propiedades transform y transition.

Ahora ya no tenemos que pensar en cómo hacerlo. En su lugar, solo tenemos que conocer cómo ajustar unos pocos parámetros y nuestro sitio web puede ser tan flexible y dinámico como lo imaginamos. La propiedad transform puede operar cuatro transformaciones básicas en un elemento: scale (escalar), rotate (rotar), skew (inclinarse) y translate (trasladar o mover). Veamos cómo funcionan:

2.5.12.1 Transform: scale

```
#principal {
    display: block;
    width: 500px;
    margin: 50px auto;
    padding: 15px;
    text-align: center;
    border: 1px solid #999999;
    background: #DDDDDD;
    -moz-transform: scale(2);
    -webkit-transform: scale(2);
}
```

En el ejemplo partimos de los estilos básicos utilizados para la cabecera generada anteriormente y aplicamos transformación duplicando la escala del elemento. La función scale recibe dos parámetros: el valor X para la escala horizontal y el valor Y para la escala vertical. Si solo un valor es provisto el mismo valor es aplicado a ambos parámetros.

Números enteros y decimales pueden ser declarados para la escala. Esta escala es calculada por medio de una matriz. Los valores entre 0 y 1 reducirán el elemento, un valor de 1 mantendrá las proporciones originales y valores mayores que 1 aumentarán las dimensiones del elemento de manera incremental.

Un efecto atractivo puede ser logrado con esta función otorgando valores negativos:

```
-moz-transform: scale(1,-1);
-webkit-transform: scale(1,-1);
```

Dos parámetros han sido declarados para cambiar la escala de la caja principal. El primer valor, 1, mantiene la proporción original para la dimensión horizontal de la caja. El segundo valor

también mantiene la proporción original, pero invierte el elemento verticalmente para producir el efecto espejo.

Existen también otras dos funciones similares a `scale` pero restringidas a la dimensión horizontal o vertical: `scaleX` y `scaleY`. Estas funciones, por supuesto, utilizan un solo parámetro.

2.5.12.2 Transform: rotate

La función `rotate` rota el elemento en la dirección de las agujas de un reloj. El valor debe ser especificado en grados usando la unidad “deg”:

```
-moz-transform: rotate(30deg);  
-webkit-transform: rotate(30deg);
```

Si un valor negativo es declarado, solo cambiará la dirección en la cual el elemento es rotado.

2.5.12.3 Transform: skew

Esta función cambia la simetría del elemento en grados y en ambas dimensiones.

```
-moz-transform: skew(20deg);  
-webkit-transform: skew(20deg);
```

La función `skew` usa dos parámetros, pero a diferencia de otras funciones, cada parámetro de esta función solo afecta una dimensión (los parámetros actúan de forma independiente). En él, las propiedades del código anterior, realizamos una operación `transform` a la caja de la cabecera para inclinarla. Solo declaramos el primer parámetro, por lo que solo la dimensión horizontal de la caja será modificada. Si usáramos los dos parámetros, podríamos alterar ambas dimensiones del objeto. Como alternativa podemos utilizar funciones diferentes para cada una de ellas: `skewX` y `skewY`.

2.5.12.4 Transform: translate

Similar a las viejas propiedades `top` y `left`, la función `translate` mueve o desplaza el elemento en la pantalla a una nueva posición.

```
-moz-transform: translate(100px);  
-webkit-transform: translate(100px);
```

La función `translate` considera la pantalla como una grilla de píxeles, con la posición original del elemento usada como un punto de referencia. La esquina superior izquierda del elemento es la

posición 0,0, por lo que valores negativos moverán al objeto hacia la izquierda o hacia arriba de la posición original, y valores positivos lo harán hacia la derecha o hacia abajo.

En el código, movimos la caja de la cabecera hacia la derecha unos 100 píxeles desde su posición original. Dos valores pueden ser declarados en esta función si queremos mover el elemento horizontal y verticalmente, o podemos usar funciones independientes llamadas `translateX` y `translateY`.

2.5.12.5 Transformando todo al mismo tiempo

A veces podría resultar útil realizar sobre un elemento varias transformaciones al mismo tiempo. Para obtener una propiedad `transform` combinada, solo tenemos que separar cada función a aplicar con un espacio:

```
-moz-transform: translateY(100px) rotate(45deg) scaleX(0.3);  
-webkit-transform: translateY(100px) rotate(45deg) scaleX(0.3);
```

Una de las cosas que debe recordar en este caso es que el orden es importante. Esto es debido a que algunas funciones mueven el punto original y el centro del objeto, cambiando de este modo los parámetros que el resto de las funciones utilizarán para operar.

2.5.12.6 Transformaciones dinámicas

Lo que hemos aprendido hasta el momento en este capítulo cambiará la forma de la web, pero la mantendrá tan estática como siempre. Sin embargo, podemos aprovecharnos de la combinación de transformaciones y pseudo clases para convertir nuestra página en una aplicación dinámica:

```
#principal {  
    display: block;  
    width: 500px;  
    margin: 50px auto;  
    padding: 15px;  
    text-align: center;  
    border: 1px solid #999999;  
    background: #DDDDDD;  
}  
#principal:hover {  
    -moz-transform: rotate(5deg);  
    -webkit-transform: rotate(5deg);  
}
```

La regla original para la caja de la cabecera fue conservada intacta, pero una nueva regla fue agregada para aplicar efectos de transformación usando la vieja pseudo clase `:hover`. El resultado obtenido es que cada vez que el puntero del ratón pasa sobre esta caja, la propiedad `transform` rota la caja en 5 grados, y cuando el puntero se aleja la caja vuelve a rotar de

regreso a su posición original. Este efecto produce una animación básica pero útil con nada más que propiedades CSS.

2.5.12.7 Transiciones

De ahora en más, hermosos efectos usando transformaciones dinámicas son accesibles y fáciles de implementar. Sin embargo, una animación real requiere de un proceso de más de dos pasos.

La propiedad `transition` fue incluida para suavizar los cambios, creando mágicamente el resto de los pasos que se encuentran implícitos en el movimiento. Solo agregando esta propiedad forzamos al navegador a tomar cartas en el asunto, crear para nosotros todos esos pasos invisibles, y generar una transición suave desde un estado al otro.

```
#principal {
    display: block;
    width: 500px;
    margin: 50px auto;
    padding: 15px;
    text-align: center;
    border: 1px solid #999999;
    background: #DDDDDD;
    -moz-transition: -moz-transform 1s ease-in-out 0.5s;
    -webkit-transition: -webkit-transform 1s ease-in-out 0.5s;
}
#principal:hover{
    -moz-transform: rotate(5deg);
    -webkit-transform: rotate(5deg);
}
```

Como puede ver en el código, la propiedad `transition` puede tomar hasta cuatro parámetros separados por un espacio. El primer valor es la propiedad que será considerada para hacer la transición (en nuestro ejemplo elegimos `transform`). Esto es necesario debido a que varias propiedades pueden cambiar al mismo tiempo y probablemente necesitemos crear los pasos del proceso de transición solo para una de ellas. El segundo parámetro especifica el tiempo que la transición se tomará para ir de la posición inicial a la final. El tercer parámetro puede ser cualquiera de las siguientes palabras clave:

- `ease`
- `linear`
- `ease-in`
- `ease-out`
- `ease-in-out`

Estas palabras clave determinan cómo se realizará el proceso de transición basado en una curva Bézier. Cada una de ellas representa diferentes tipos de curva Bézier, y la mejor forma de saber cómo trabajan es viéndolas funcionar en pantalla. El último parámetro para la propiedad `transition` es el retardo. Éste indica cuánto tiempo tardará la transición en comenzar.

Para producir una transición para todas las propiedades que están cambiando en un objeto, la palabra clave `all` debe ser especificada. También podemos declarar varias propiedades a la vez listándolas separadas por coma.

2.6 Referencia rápida

CSS3 provee nuevas propiedades para crear efectos visuales y dinámicos que son parte esencial de la web en estos días.

- **border-radius** Esta propiedad genera esquinas redondeadas para la caja formada por el elemento. Posee dos parámetros diferentes que dan forma a la esquina. El primer parámetro determina la curvatura horizontal y el segundo la vertical, otorgando la posibilidad de crear una elipsis. Para declarar ambos parámetros de la curva, los valores deben ser separados por una barra (por ejemplo, `border-radius: 15px / 20px`). Usando solo un valor determinaremos la misma forma para todas las esquinas (por ejemplo, `border-radius: 20px`). Un valor para cada esquina puede ser declarado en un orden que sigue las agujas del reloj, comenzando por la esquina superior izquierda.
- **box-shadow** Esta propiedad crea sombras para la caja formada por el elemento. Puede tomar cinco parámetros: el color, el desplazamiento horizontal, el desplazamiento vertical, el valor de difuminación, y la palabra clave `inset` para generar una sombra interna. Los desplazamientos pueden ser negativos, y el valor de difuminación y el valor `inset` son opcionales (por ejemplo, `box-shadow: #000000 5px 5px 10px inset`).
- **text-shadow** Esta propiedad es similar a `box-shadow` pero específica para textos. Toma cuatro parámetros: el color, el desplazamiento horizontal, el desplazamiento vertical, y el valor de difuminación (por ejemplo, `text-shadow: #000000 5px 5px 10px`).
- **@font-face** Esta regla nos permite cargar y usar cualquier fuente que necesitemos. Primero, debemos declarar la fuente, proveer un nombre con la propiedad `font-family` y especificar el archivo con `src` (por ejemplo, `@fontface{font-family: Mifuentes; src: url('font.ttf') }`). Luego de esto, podremos asignar la fuente (en el ejemplo Mifuentes) a cualquier elemento del documento.
- **linear-gradient(posición inicio, color inicial, color final)** Esta función puede ser aplicada a las propiedades `background` o `background-image` para generar un gradiente lineal. Los atributos indican el punto inicial y los colores usados para crear el gradiente. El primer valor puede ser especificado en píxeles, en porcentaje o usando las palabras clave `top`, `bottom`, `left` y `right`. El punto de inicio puede ser reemplazado por un ángulo para proveer una dirección específica para el gradiente (por ejemplo, `linear-gradient(top, #FFFFFF 50%, #006699 90%)`).
- **radial-gradient(posición inicio, forma, color inicial, color final)** Esta función puede ser aplicada a las propiedades `background` o `background-image` para generar un gradiente radial. La posición de inicio es el origen y puede ser declarado en píxeles, porcentaje o como una combinación de las palabras clave `center`, `top`, `bottom`, `left` y `right`. Existen dos valores para la forma: `circle` y `ellipse`, y puntos de terminación pueden ser declarados para cada color indicando la posición donde la transición comienza (por ejemplo, `radial-gradient(center, circle, #FFFFFF 0%, #006699 200%)`).
- **rgba()** Esta función es una mejora de `rgb()`. Toma cuatro valores: el color rojo (0-255), el color verde (0-255), el color azul (0-255), y la opacidad (un valor entre 0 y 1).
- **hsla()** Esta función es una mejora de `hsl()`. Puede tomar cuatro valores: el tono (un valor entre 0 y 360), la saturación (un porcentaje), la luminosidad (un porcentaje), y la opacidad (un valor entre 0 y 1).

- **outline** Esta propiedad fue mejorada con la incorporación de otra propiedad llamada **outline-offset**. Ambas propiedades combinadas generan un segundo borde alejado del borde original del elemento (por ejemplo, `outline: 1px solid #000000; outline-offset: 10px;`).
- **border-image** Esta propiedad crea un borde con una imagen personalizada. Necesita que el borde sea declarado previamente con las propiedades **border** o **border-with**, y toma al menos tres parámetros: la URL de la imagen, el tamaño de las piezas que serán tomadas de la imagen para construir el borde, y una palabra clave que especifica cómo esas piezas serán ubicadas alrededor del elemento (por ejemplo, `border-image: url("file.png") 15 stretch;`).
- **transform** Esta propiedad modifica la forma de un elemento. Utiliza cuatro funciones básicas: **scale** (escalar), **rotate** (rotar), **skew** (inclinarse), y **translate** (trasladar o mover). La función **scale** recibe solo un parámetro. Un valor negativo invierte el elemento, valores entre 0 y 1 reducen el elemento y valores mayores que 1 expanden el elemento (por ejemplo, `transform: scale(1.5);`). La función **rotate** usa solo un parámetro expresado en grados para rotar el elemento (por ejemplo, `transform: rotate(20deg);`). La función **skew** recibe dos valores, también en grados, para la transformación horizontal y vertical (por ejemplo, `transform: skew(20deg, 20deg);`). La función **translate** mueve el objeto tantos píxeles como sean especificados por sus parámetros (por ejemplo, `transform: translate(20px);`).
- **transition** Esta propiedad puede ser aplicada para crear una transición entre dos estados de un elemento. Recibe hasta cuatro parámetros: la propiedad afectada, el tiempo que le tomará a la transición desde el comienzo hasta el final, una palabra clave para especificar cómo la transición será realizada (**ease**, **linear**, **ease-in**, **ease-out**, **ease-in-out**) y un valor que determina el tiempo que la transacción tardara en comenzar (por ejemplo, `transition: color 2s linear 1s;`).