



Nuevos elementos semánticos HTML5. Nuevas propiedades CSS3
Alejandro Amat Reina

ÍNDICE

1. INTRODUCCIÓN	2
2. HTML5	3
LA SEMÁNTICA DE LOS ELEMENTOS	3
NUEVOS ELEMENTOS ESTRUCTURALES	5
EL ELEMENTO HEADER	7
EL ELEMENTO SECTION	7
EL ELEMENTO ARTICLE	9
EL ELEMENTO NAV	10
EL ELEMENTO ASIDE	11
EL ELEMENTO FOOTER	11
EL ELEMENTO MAIN	12
ORGANIZANDO EL DISEÑO DE LA PÁGINA CON ELEMENTOS HTML5	13
EL ESQUEMA DEL DOCUMENTO	13
OTROS ELEMENTOS SEMÁNTICOS	15
LOS ELEMENTOS FIGURE Y FIGCAPTION	15
EL ELEMENTO MARK	15
EL ELEMENTO TIME	16
ELEMENTOS CON LA SEMÁNTICA REDEFINIDA	17
EL ELEMENTO SMALL	17
EL ELEMENTO CITE	17
EL ELEMENTO ADDRESS	18
CATEGORÍAS DE ELEMENTOS HTML5	18
NAVEGADORES ANTIGUOS	20
3. CSS3	22
UNA NOTA SOBRE PREFIJOS DE PROVEEDORES	22
NUEVOS SELECTORES CSS3	23
SELECTOR GENERAL DE ELEMENTOS HERMANOS	23
NUEVOS SELECTORES DE ATRIBUTO	24
PSEUDO-ELEMENTOS	24
NUEVAS PSEUDO-CLASES	25
NUEVAS PROPIEDADES CSS3	26
BORDES CON ESQUINAS REDONDEADAS (BORDER-RADIUS)	28
SOMBRAS EN LAS CAJAS (BOX-SHADOW)	29
SOMBRAS EN LOS TEXTOS (TEXT-SHADOW)	30
FONDOS CON GRADIENTE	31
TRANSPARENCIAS	33
OUTLINE	35
BORDER-IMAGE	36

1. Introducción

En los temas anteriores hemos visto una introducción a HTML5 y CSS3 haciendo especial hincapié en aquellas cosas que no han cambiado con respecto a las versiones anteriores de estas tecnologías.

En este tema, comenzaremos a estudiar las nuevas características de ambas. Veremos los motivos por los que se hacía necesaria la introducción de nuevos elementos HTML, qué nuevos elementos se han introducido y por qué, así como otros elementos que ya existían cuya semántica ha sido redefinida.

En lo que respecta a CSS, veremos los nuevos selectores CSS3, pseudo-elementos, pseudo-clases, etc. También veremos algunas nuevas características que nos facilitarán la labor a la hora de crear diseños más complejos, como es el caso de los bordes redondeados, sombras en los textos y en las cajas, transparencias, imágenes de borde, etc.

2. HTML5

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 ayudan 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.

A continuación, veremos cuáles fueron los motivos que llevaron a los desarrolladores de HTML5 a crear nuevos elementos.

La semántica de los elementos

HTML no es más que una forma de describir documentos que contienen hipervínculos, documentos que están unidos entre sí como parte de una red de conocimiento. Cuando introducimos un elemento HTML en un documento, lo hacemos por un propósito determinado, cada elemento tiene un significado, y ese significado es lo que conocemos como semántica del elemento.

Por ejemplo, cuando introducimos un elemento `<h1>` estamos indicando que contiene un título de nivel 1, cuando introducimos un elemento `<p>` estamos indicando que contiene un párrafo de texto, etc.

El hecho de que seamos capaces de describir la estructura del documento de esta manera es importante, porque, de esta forma, podremos separar los contenidos que mostramos de la forma en que los mostramos. El resultado es que una página web, si está bien estructurada, podrá leerse fácilmente en un ordenador de escritorio, en un teléfono móvil o en cualquier otro dispositivo no visual, por ejemplo, un conversor de texto a voz.

Podemos comparar esto con un formato de documento como PDF, donde el diseño y el contenido están profundamente relacionados entre sí, ya que la finalidad de la salida impresa es el objetivo principal. Esto hace que sea difícil leer un PDF con tamaño A4 en un móvil, porque no está pensado para visualizarlo en este tamaño.

HTML4 dispone de dos atributos que le permiten ampliar la semántica de los elementos: `id` y `class`.

El atributo `id` es un identificador único, pero, en lugar de una cadena aleatoria, el identificador suele ser una palabra significativa, puede tener valor semántico. Por el contrario, el atributo `class` no es único, distintos elementos pueden tener el mismo valor en este atributo, además, un elemento puede tener aplicadas múltiples clases. A continuación, veremos algunos ejemplos:

Etiqueta	Significado
<code><p></code>	Un párrafo
<code><p id="autor"></code>	Un párrafo que representa un autor particular
<code><p class="biografia"></code>	Un párrafo que representa una biografía
<code><p class="autor biografia"></code>	Un párrafo que representa la biografía de un autor

Tabla 1: Usos de los atributos `id` y `class`

No hay un estándar que diga qué significado tienen los valores que utilizamos, por lo que un sitio podría utilizar `escritor` haciendo referencia a lo mismo que en otro sitio están llamando `autor`, o dos sitios podría utilizar `autor` para hacer referencia a algo completamente diferente.

Esto no es un gran problema, ya que HTML no tiene la intención de describir las cosas del mundo real como autores, coches, etc. El significado de esos valores será específico del sitio. Pero los atributos `id` y `class` también pueden ser usados para describir las características del documento, por ejemplo, el valor `nav` para el atributo `class`, probablemente, indicará un elemento que contiene la navegación de la página. Por este motivo, si buscas ideas para añadir nuevos elementos a HTML y mejorar su capacidad para describir los documentos, un estudio de los valores que se utilizan en los atributos `id` y `class` sería un buen lugar para empezar.

Con esto en mente, en 2005 se realizaron varios estudios que analizaban cómo los autores estaban usando los valores `id` y `class` en la web. Dos de ellos son de particular interés:

- 🚦 En noviembre de 2005, un estudio de 1.315 sitios web contaba con qué frecuencia se utilizaron diferentes valores para el atributo `id`.
- 🚦 En diciembre de 2005, un estudio de millones de páginas web analizó, entre otras cosas, con qué frecuencia aparecían determinados nombres en el atributo `class`.

El diagrama que sigue muestra los 20 resultados que más veces aparecían en cada categoría y los nuevos elementos de HTML5 correspondientes.

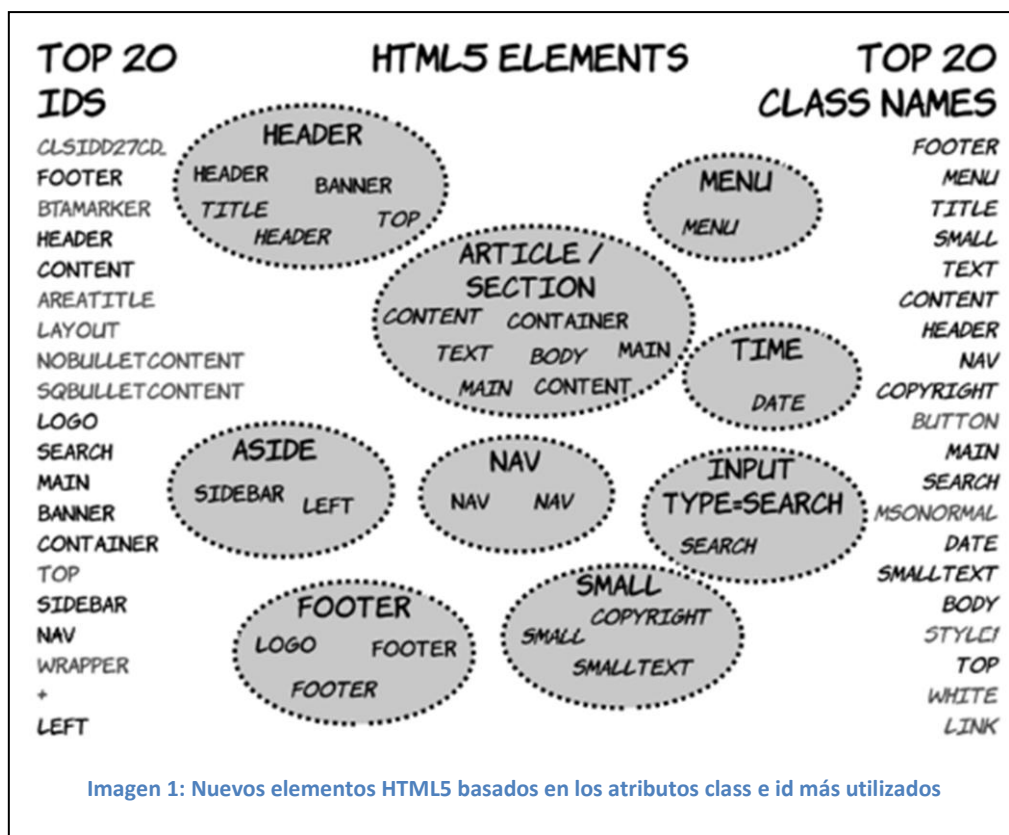


Imagen 1: Nuevos elementos HTML5 basados en los atributos class e id más utilizados

Muchos de los identificadores más utilizados, como `btamarker` y `nobulletcontent`, son generados automáticamente por alguna herramienta software como Microsoft FrontPage y otras aplicaciones. Por ello, su popularidad es más una indicación de la penetración en el mercado de los productos que de las intenciones del autor. En los siguientes puntos, veremos algunos de los nuevos elementos que han sido añadidos a HTML5 como resultado de estos estudios.

Nuevos elementos estructurales

Ya vimos en el tema 1 que la mayoría de los sitios web tienen un diseño similar. La estructura de las páginas web suele dividirse en diferentes secciones donde suelen aparecer una cabecera, una barra de navegación, una zona de contenidos, etc. En la siguiente imagen, observaremos este diseño web típico aplicado a la página web de un blog.



En este ejemplo, se puede identificar claramente 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

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.

Para alcanzar este diseño, en primer lugar, consideraremos cuáles son los elementos que queremos usar en la construcción del diseño general de nuestro sitio. Dado que HTML5 incluye una gama más amplia de elementos semánticos que sus antecesores, puede que necesitemos un poco más de tiempo para pensar acerca de la estructura y el significado de los contenidos de nuestra página de lo

que empleábamos en el pasado con HTML4 o XHTML, pero como contrapartida obtendremos páginas mucho más coherentes en lo que se refiere a la semántica de las etiquetas utilizadas.

A continuación, veremos algunos de los nuevos elementos semánticos HTML5 que podrían utilizarse para ayudar a dividir nuestra página y añadir más significado a la estructura de nuestro documento.

El elemento header

El primer elemento que veremos es el elemento de encabezado.



La especificación WHATWG lo describe como *"a group of introductory or navigational aids"*, que viene a ser "un grupo de ayudas introductorias o de navegación". En esencia, esto significa que todo el contenido que acostumbrábamos a incluir dentro de un elemento `<div id="header">`, ahora se incluiría en un `<header>`. Pero hay un aspecto que diferencia el `<header>` del `<div id="header">`, mientras que sólo podemos tener un elemento `<div id="header">` en toda la página, no existe esta restricción para el elemento `<header>`, se puede incluir un nuevo elemento de encabezado para introducir cada sección del sitio. Podemos interpretar una sección como cualquier parte de contenido que pueda necesitar su propio encabezado.

Un elemento `<header>` puede ser utilizado para incluir contenido introductorio o ayudas a la navegación que sean específicas de una sola sección de la página, que se aplican a la totalidad de la página, o ambas cosas.

Normalmente, el `<header>` se colocará en la parte superior de una página o sección, pero su definición es independiente de su posición. Es decir, podemos tener un `<header>` que se encuentre a la izquierda, a la derecha, o incluso debajo del contenido que describe.

El elemento section

El siguiente elemento con el que debemos familiarizarnos es el elemento `<section>` de HTML5.

La especificación WHATWG lo define de la siguiente manera:

"El elemento `section` representa una sección genérica de un documento o aplicación. Una sección, en este contexto, es una agrupación temática de los contenidos, por lo general, con un título".

Se explica, además, que una sección no debe ser utilizada como un contenedor genérico que existe sólo con fines de estilo o scripting, en esos casos utilizaremos el elemento `<div>` que es para lo que se creó.

Volviendo a la definición de las especificaciones, el contenido del elemento de sección debe ser "temático", por lo que sería incorrecto utilizarlo de una manera genérica para envolver piezas sin relación de contenido.

Algunos ejemplos de usos aceptables para elementos `<section>` incluyen:

- ✚ Secciones individuales de una interfaz con pestañas.
- ✚ Segmentos de una página "Acerca de", por ejemplo, la página "Acerca de" de una empresa podría incluir secciones sobre la historia de la empresa, su misión y su equipo.
- ✚ Diferentes partes de los "términos de servicio" de la página.
- ✚ distintas secciones de un sitio de noticias en línea, por ejemplo, los artículos se podrían agrupar en secciones que cubren los deportes, los asuntos mundiales, y las noticias económicas.

Incluso Bruce Lawson, una autoridad muy respetada en HTML5, ha admitido el uso de la sección de forma incorrecta en el pasado. Bruce posteó en su blog (<http://html5doctor.com/the-section-element/>) una entrada explicando su error, y bien vale la pena leerla. En resumen:

- ✚ `<section>` es genérico, por lo que, si un elemento semántico más específico es apropiado (como `<article>`, `<aside>`, o `<nav>`), lo utilizaremos en su lugar.
- ✚ `<section>` tiene significado semántico, esto implica que la información que contiene está relacionada de alguna manera. Si no somos capaces de describir sucintamente todo el contenido que estamos tratando de poner en una sección utilizando sólo unas pocas palabras, lo más probable es que necesitemos un contenedor semánticamente neutral (`<div>`) en su lugar.

También debemos tener en cuenta que, si es apropiado, es correcto utilizar elementos `<section>` anidados dentro de otros elementos `<section>` existentes. Por

ejemplo, para un sitio web de noticias en línea, la sección de noticias del mundo podría subdividirse en una sección para cada región global importante.





El elemento `article`

El elemento `<article>` es similar al elemento `<section>`, pero hay algunas diferencias notables. He aquí la definición según WHATWG:

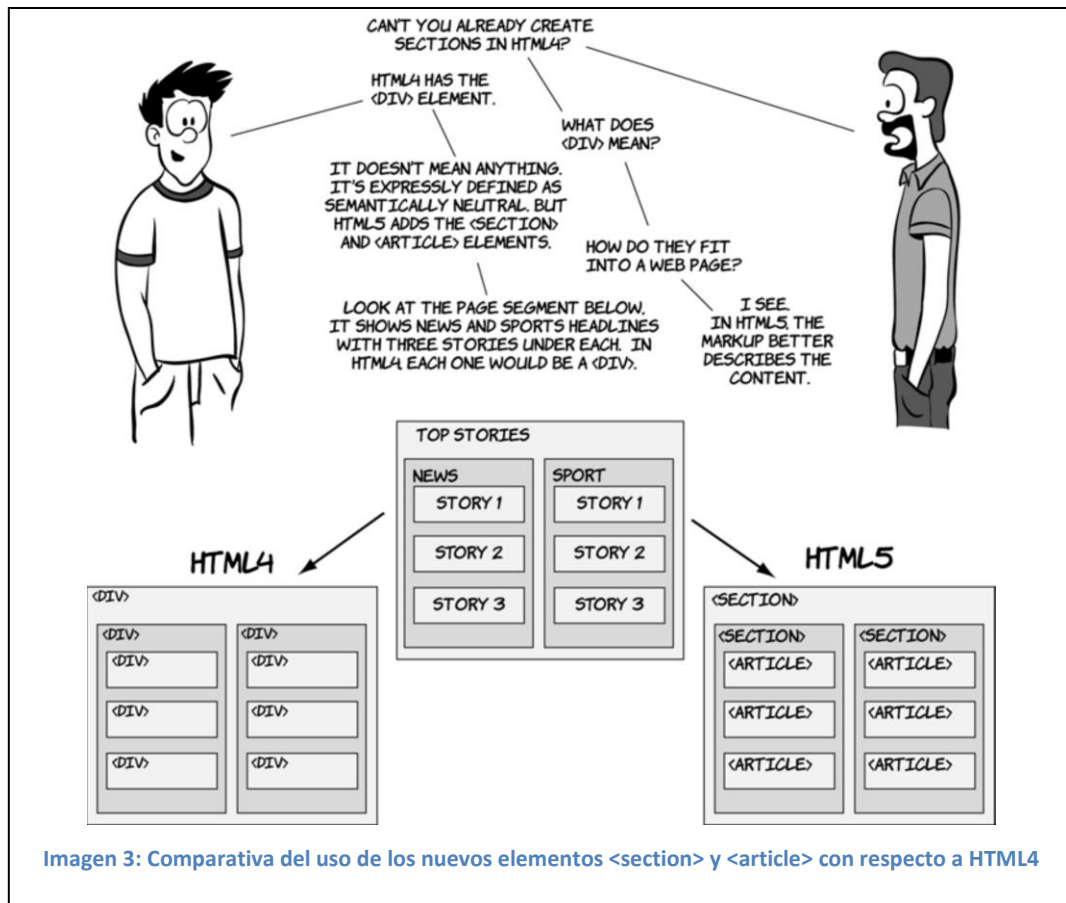
"El elemento `<article>` representa una composición auto-contenida en un documento, página, aplicación o sitio, y que es, en principio, distribuible de forma independiente o reutilizable, por ejemplo, en la sindicación".

Los términos claves en esta definición son la composición autónoma e independientemente distribuible. Mientras que `<section>` puede contener cualquier contenido que se puede agrupar temáticamente, `<article>` debe ser una sola pieza de contenido que puede valerse por sí misma. En caso de duda, haremos la prueba de sindicación: si una parte del contenido se puede publicar en otro sitio sin ser modificada o enviada como una actualización a través de RSS o en las redes sociales (sitios como Twitter o Facebook), tiene los ingredientes para ser un `<article>`.

Aquí hay algunas sugerencias de uso de `<article>`:

-  mensajes en el foro.
-  Artículos de revistas o periódicos.
-  Las entradas de un blog.
-  los comentarios enviados por los usuarios, etc.

Por último, al igual que los elementos `<section>`, los elementos `<article>` se pueden anidar. También podemos anidar una sección dentro de un artículo, y viceversa.

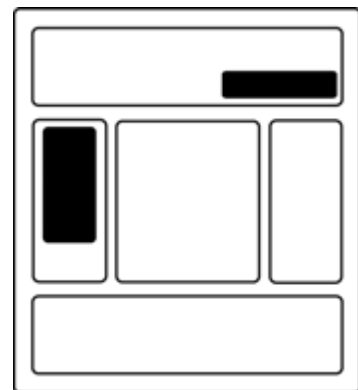


El elemento `nav`

Podemos asumir que este elemento aparecerá en, prácticamente, todos los proyectos. `<nav>` representa un grupo de vínculos de navegación. Lo más habitual será que contenga una lista desordenada de enlaces, aunque hay otras opciones. En cualquier caso, el `<nav>` se debe reservar para la navegación que es de primordial importancia. Será adecuado un elemento `<nav>` en cualquiera de los siguientes casos:

- ✚ Si tenemos una barra de navegación principal del sitio web.
- ✚ Si tenemos un conjunto secundario de enlaces que apuntan a diferentes partes de la página actual (mediante anclajes).
- ✚ Para un formulario de búsqueda que constituye el principal medio de navegación de un sitio (como es el caso de Google).

No será adecuado un elemento `<nav>`, por ejemplo, en los enlaces que aparecen en el pie de página.

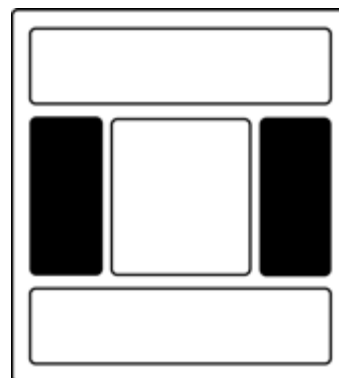


El elemento `aside`

Este elemento representa una parte de la página que está *"tangencialmente relacionado con el contenido que se encuentra alrededor, y que podría considerarse separado de ese contenido"*.

El elemento `<aside>` podría ser utilizado para envolver una porción de contenido que es tangencial a:

- ✚ una pieza independiente de contenido específico (por ejemplo, un artículo o una sección)
- ✚ una página entera o documento, como se ha hecho habitualmente cuando añadimos una "barra lateral" (*sidebar*) a una página o sitio web.



El elemento `<aside>` nunca debe ser usado para envolver las secciones de la página que son parte del contenido principal. El contenido de un `<aside>` puede valerse por sí mismo, pero aún debe ser parte de un todo más grande.

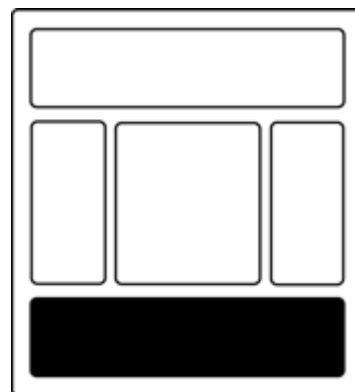
Algunos usos posibles para un `<aside>` serían: una barra lateral, una lista secundaria de enlaces, o un espacio para la publicidad. También hay que señalar que el elemento `<aside>` (como en el caso del `<header>`) no se define por su posición en la página. Podría ser en un "lado", o podría estar en otro lugar. Es el contenido en sí, y su relación con otros elementos, lo que lo define.

El elemento `footer`

Al igual que con el `<header>`, podemos tener varios `<footer>` en una sola página.

Un elemento `<footer>`, de acuerdo a la especificación, *"representa un pie de página de la sección de contenido que es su ancestro más cercano"*. La "sección" de contenido podría ser todo el documento, o podría ser un `<section>`, un `<article>` o un `<aside>`.

A menudo, un pie de página contendrá información sobre el copyright, las listas de enlaces, información del autor, y contenido similar que, normalmente, consideramos como el final de un bloque de contenido. Sin embargo, al igual que `<aside>` y `<header>`, un elemento `<footer>` no se define en términos de su posición en la página, por lo que no tiene porqué aparecer al final de una sección, o en la parte inferior de una página. Lo más probable es que sí, pero esto no es necesario.



Por ejemplo, la información sobre el autor de un blog es posible que aparezca por encima del mensaje en lugar de debajo de ella, y todavía será considerada como información de pie de página.

El elemento `main`

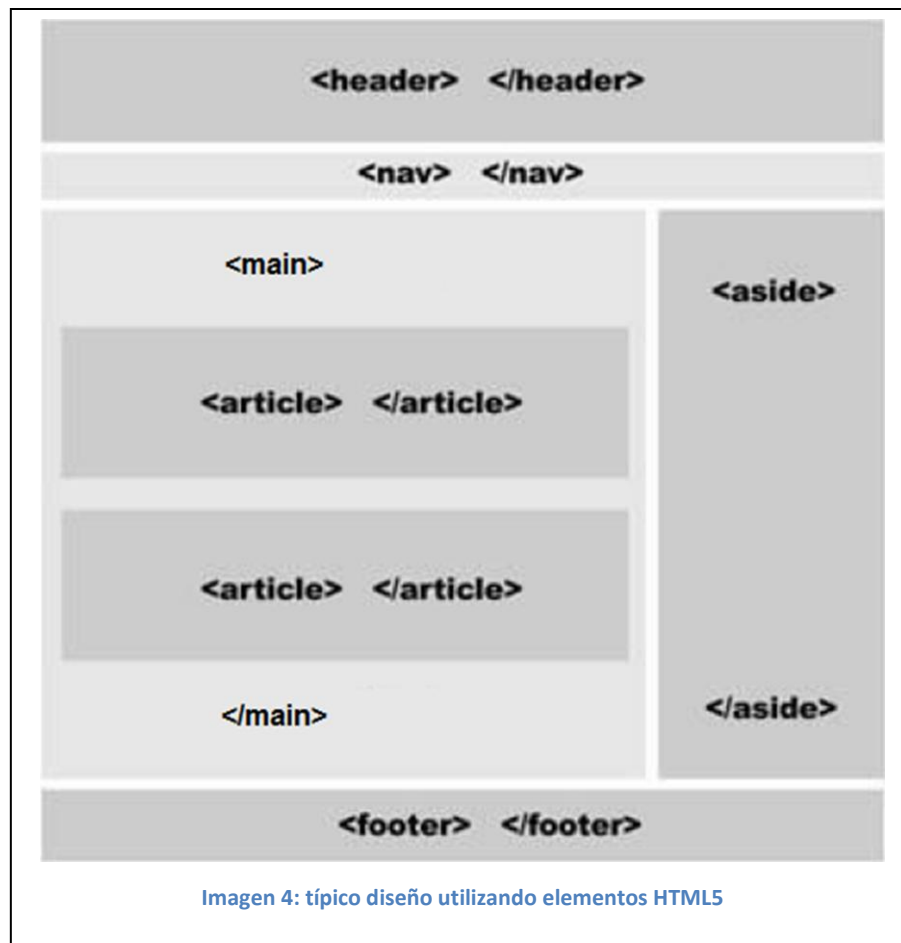
El elemento `<main>` especifica el contenido principal del documento. Las características principales de este elemento según la especificación son:

- ✚ Representa el contenido principal del cuerpo (`<body>`) de una web o aplicación.
- ✚ Incluye el contenido que es único en la página, y excluye contenido que se repite en todas las páginas de la web (menú, pie de página, barra lateral, etc.).
- ✚ No debe incluirse más de un elemento `<main>` por página.
- ✚ No debe incluirse el elemento `<main>` dentro de elementos como `<article>`, `<aside>`, `<footer>`, `<header>` o `<nav>`.

La forma más fácil de ver cuando debemos usar el elemento `<main>` es sustituyendo los `<div>` que tienen como id principal (`main`) o contenidos (`content`).

```
<body>
  <header role="banner">
    [...]
  </header>
  <main>
    [...]
  </main>
  <footer>
    [...]
  </footer>
</body>
```

Organizando el diseño de la página con elementos HTML5



En esta imagen se muestra el típico diseño presentado en el tema 1, pero esta vez con los correspondientes elementos HTML5 (incluyendo etiquetas de apertura y cierre). Si dentro del contenido principal tuviéramos artículos agrupados por diferentes temáticas, cada grupo estaría dentro de un elemento `<section>`.

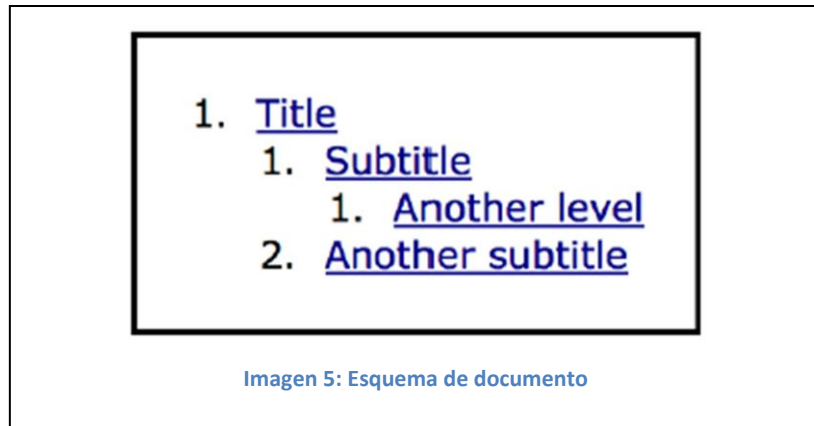
El esquema del documento

En versiones anteriores de HTML, podemos elaborar un esquema de cualquier documento mirando los diferentes niveles de encabezados (`<h1>` hasta `<h6>`) contenidos en la página.

Por ejemplo, estas etiquetas:

```
<h1> Title </ h1>
:
<h2> Subtitle </ h2>
:
<h3> Another level </ h3>
:
<h2> Another subtitle </ h2>
```

Producirían el esquema de documento que se muestra en la siguiente imagen:



Es preferible que cada página tenga un solo elemento `<h1>`, seguido de otros epígrafes de forma secuencial.

Con el fin de hacer que el contenido sea más fácil de syndicar y más portable, la especificación HTML5 cambia un poco esta filosofía, de forma que, cada elemento que entra en la categoría de "contenido de seccionamiento"¹ crea un nuevo nodo en el esquema del documento. Veamos un ejemplo:

```
<section>
  <h1>Title</h1>
  :
  <article>
    <h1>Article Title</h1>
    :
    <h2>Article Subtitle</h2>
    :
  </article>
  <article>
    <h1>Another subtitle</h1>
    :
  </article>
</section>
```

Cada pieza de contenido de seccionamiento (los elementos `<article>` en este ejemplo) crea una nueva rama en el árbol de documentos, por lo que puede tener su propio `<h1>`. De esta manera, cada sección tiene su propio esquema de documento.

La ventaja de esto, es que podemos mover una sección entera a un documento totalmente diferente conservando el mismo esquema de documento.

¹ En HTML5, esto incluye `article`, `aside`, `nav` y `section`

Otros elementos semánticos

Además de los elementos estructurales vistos anteriormente, HTML5 introduce otros elementos con contenido semántico. A continuación, veremos los más importantes.

Los elementos `figure` y `figcaption`

El elemento `<figure>` se explica en las especificaciones de la siguiente forma:

"El elemento se utilizará para anotar ilustraciones, diagramas, fotos, listados de código, etc, a los cuales se refiera el contenido principal del documento, pero que podrían, sin afectar al flujo del documento, ser movidos lejos de ese contenido primario, por ejemplo, a un lado de la página, a las páginas dedicadas, o a un apéndice".

El elemento `<figcaption>` es, simplemente, una manera de marcar un título para una parte del contenido que aparece en el interior de una figura.

Cuando utilizamos el elemento `<figure>`, el contenido que coloquemos en su interior debe tener alguna relación con el contenido principal en el que aparece. Si se puede eliminar por completo de un documento, y el contenido del documento no pierde sentido, probablemente no deberíamos utilizar este elemento.

```
<figure>
  <figcaption>Screen Reader Support for WAI-ARIA</figcaption>
  
</figure>
```

El elemento `mark`

El elemento `<mark>` *"indica una parte del documento que ha sido remarcada por la importancia que tiene para la actividad actual del usuario"*. Es cierto que podemos imaginar muy pocos usos para este elemento, posiblemente, el más habitual sea en el contexto de una búsqueda, donde queremos destacar las palabras clave que se han buscado dentro de los resultados.

Por ejemplo, si un usuario ha llegado a un artículo en nuestro sitio utilizando una búsqueda en Google de la palabra "HTML5", podríamos resaltar esta palabra en el artículo usando el elemento `<mark>`, así:

```
<h1>Sí, usted puede utilizar <mark> HTML5 </mark> Hoy!</h1>
```

El elemento `<mark>` se puede agregar al documento, ya sea usando código del lado del servidor, o JavaScript una vez que la página se haya cargado.

El elemento `time`

Las fechas y horarios son partes fundamentales de los contenidos de las páginas web. Los motores de búsqueda son capaces de filtrar los resultados basándose en el tiempo y, en algunos casos, un resultado de búsqueda específico puede recibir más o menos peso en función de cuando fue publicado por primera vez.

El elemento `<time>` ha sido diseñado específicamente para tratar con el problema de la lectura de las fechas y horas. Veamos el siguiente ejemplo:

```
<p>La siguiente conferencia de HTML5 será el próximo 12 de Octubre.</p>
```

Aunque cuando una persona lee el párrafo anterior tiene claro cuándo se va a producir el evento, si es una máquina la que está analizando la información, probablemente, encuentre problemas para deducirlo. A continuación, veremos el mismo párrafo, pero introduciendo el elemento `<time>`:

```
<p>La siguiente conferencia de HTML5 será el próximo <time  
datetime="2014-10-12">12 de Octubre</time>.</p>
```

El elemento `<time>` se encuentra representado en un formato de 24 horas, o en una fecha precisa en el calendario Gregoriano utilizando, opcionalmente, el horario y zona horaria. Veamos algunos ejemplos:

- ✚ Sin el atributo `datetime` el contenido debe ser válido:

```
<time>2009-11-13</time>
```

- ✚ Con el atributo `datetime` el contenido puede ser cualquier cosa:

```
<time datetime="2009-11-13">13 Noviembre</time>
```

```
<time datetime="20:00">Comienza a las 8pm</time>
```

- ✚ Utilizando la zona horaria:

```
<time datetime="2009-11-13T20:00+00:00">Comienza a las 8pm</time>
```

- ✚ utilizando la zona horaria "Z" (La zona "Z" la utilizamos para representar Universal Coordinated Time (UTC)):

```
<time datetime="2009-11-13T20:00Z">Comienza a las 8pm</time>
```

El único problema con el elemento `<time>` es que debe contener una fecha positiva en el calendario Gregoriano, es decir, no se puede establecer una fecha menor a la Era Cristiana.

Por otro lado, también podemos escribir fechas que no se encuentren completas:

- ✚ `<time datetime="1905">` significa el año 1905

- ✚ `<time datetime="1905-11">` significa Noviembre 1905

- ✚ `<time datetime="11-13">` significa el 13 de Noviembre (cualquier año)

- ✚ `<time datetime="1905-W21">` significa semana 21 de 1905

También podemos indicar duraciones utilizando el prefijo P para períodos, D para días, H para horas, M para minutos, y S para segundos.

`<time datetime="P4D">` es una duración de 4 días.

`<time datetime="PT23H9M30S">` es una duración de 23 horas, 9 minutos y 30 segundos.

```
<article>
  <header>
    <h1>Evento HTML5 en Madrid</h1>
    <p>Próximo <time datetime="2014-05-15">15 de Mayo</time></p>
  </header>
  <p>El 15 de mayo...</p>
</article>
```

Elementos con la semántica redefinida

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 no solo están relacionadas con nuevos elementos, también con cómo usamos los ya existentes.

El elemento `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. Esto hizo que dicho elemento dejara de usarse, ya que, con la aparición de las páginas CSS todo lo que estuviera relacionado con la presentación de los contenidos debía indicarse mediante CSS. En HTML5, el nuevo propósito del elemento `<small>` es presentar la llamada letra pequeña como impresiones legales, descargos, etc...

```
<small>Derechos Reservados &copy; 2014 Alejandro Amat</small>
```

El elemento `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>Me encanta la película <cite>Gladiator</cite></span>
```

El elemento `address`

El elemento `<address>` es un viejo elemento convertido en un elemento estructural. 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.

Por ejemplo, podríamos incluirlo dentro de un `<footer>`, como en el siguiente ejemplo:

```
<article>
  <header>
    <h1>Título del mensaje </h1>
  </header>
  <p>Este es el texto del mensaje</p>
  <footer>
    <address>
      Escrito por: <a href="http://www.mario.com">Mario</a>
    </address>
  </footer>
</article>
```

Categorías de elementos HTML5

Como vimos en el tema anterior, por motivos de diseño y estilo, los desarrolladores se han acostumbrado a pensar en los elementos de una página HTML como pertenecientes a una de dos categorías: en bloque (incluyen un salto de línea antes y después del elemento) y en línea (no incluyen salto de línea).

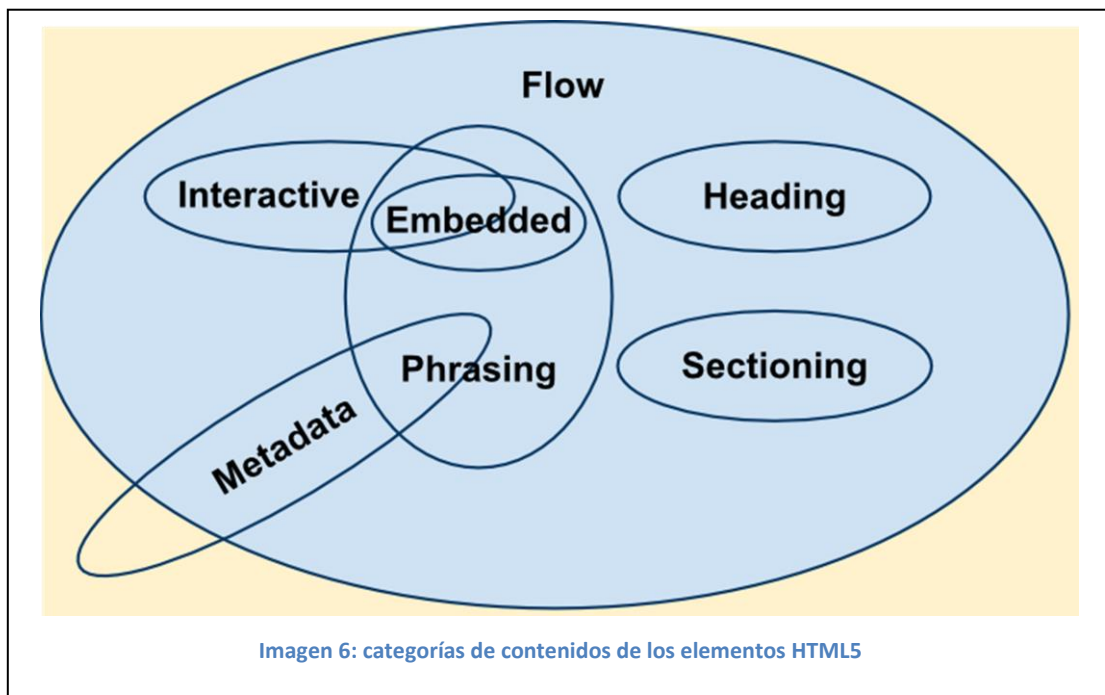
Aunque los navegadores todavía presentan los elementos como elementos en bloque o en línea, la especificación HTML5 va un paso más allá a la hora de clasificarlos, y los divide en las siguientes categorías:

- ✚ Metadatos (metadata content): los datos que no están presentes en la propia página, pero afectan a la presentación de la misma o incluyen otra información sobre ella. Esto incluye elementos como `<title>`, `<link>`, `<meta>` y `<style>`.
- ✚ Elementos que afectan al flujo de contenidos de la página (flow content): incluye casi todos los elementos que se utilizan en el cuerpo (`<body>`) de un documento HTML, por ejemplo `<header>`, `<footer>`, e incluso `<p>`.
- ✚ Elementos de seccionamiento (sectioning content): Hemos hablado en puntos anteriores del término genérico "sección" para referirnos a un bloque de contenido que pudiera contener un `<header>`, `<footer>` o `<aside>`.

En HTML5, esto incluye `<article>`, `<aside>`, `<nav>`, y `<section>`.

- ✚ Encabezados (heading content): Este tipo de contenido define el encabezado de una sección determinada, e incluye los elementos de encabezado (<h1>, <h2>, etc.).
- ✚ Contenido de frase (phrasing content): Esta categoría incluye los elementos en línea, como por ejemplo , , <cite> y similares.
- ✚ Contenido incrustado (embedded content): Elementos que son incrustados en una página, como , <object>, <embed>, <video>, <canvas> y otros.
- ✚ Contenido interactivo (interactive content): Esta categoría incluye cualquier contenido con el que los usuarios pueden interactuar. Se compone principalmente de elementos de formulario, así como enlaces y otros elementos que son interactivos sólo cuando ciertos atributos están presentes.

Como se puede deducir de la lista anterior, algunos elementos pueden pertenecer a más de una categoría.



En la imagen se puede apreciar, como las distintas categorías están solapadas, ya que, hay elementos que pertenecen a más de una categoría (Para más información sobre la clasificación de los elementos de contenido, podéis consultar la siguiente URL:

https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Content_categories).

Navegadores antiguos

Como hemos visto, HTML5 incluye una serie de nuevos elementos, tales como `article`, `section`, etc. Se podría pensar que esto es un gran problema para los navegadores más antiguos, pero no es así en la mayoría de los casos. En realidad, no importa los tags que utilicemos. Si tuviéramos un documento HTML con una etiqueta `<recipiente>` en él y, mediante CSS, asociáramos algunos estilos a este elemento, casi todos los navegadores procederían como si esto fuera totalmente normal y aplicarían los estilos sin problema.

Por supuesto, este documento hipotético no validaría, pero funcionaría bien en casi todos los navegadores, con la excepción de Internet Explorer. Antes de la versión 9, IE impedía aplicar estilos a elementos no reconocidos, y esta característica afecta a los nuevos elementos de HTML5.

Afortunadamente, hay una solución: una pieza muy simple de JavaScript conocida como "HTML5 shiv" o "HTML5 shim", y desarrollada originalmente por John Resig, puede hacer que los nuevos elementos de HTML5 sean visibles para versiones antiguas de IE.

Debemos incluir este javascript en nuestros documentos rodeado de comentarios condicionales. Los comentarios condicionales son una característica propia implementada por Microsoft en Internet Explorer que nos proporciona la posibilidad de aplicar estilos o enlazar scripts sólo para versiones específicas de este navegador.

Este comentario condicional está diciendo al navegador que el contenido encerrado sólo debe aparecer para los usuarios que visualizan la página con versiones de Internet Explorer anteriores a la versión 9:

```
<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

Debe tenerse en cuenta que si utilizamos una biblioteca de JavaScript que se ocupa de las características de HTML5 o las nuevas API, es posible que ya tenga implementada esta funcionalidad, por lo que se puede eliminar la referencia a este script. Un ejemplo de esto sería `Modernizr`, una biblioteca JavaScript que detecta características nuevas de HTML y CSS. `Modernizr` incluye código que permite utilizar los elementos de HTML5 en versiones antiguas de IE, por lo que el script anterior sería redundante.

Evidentemente, los usuarios que tengan el javascript deshabilitado en su navegador y utilicen una versión de IE inferior a la 9, no podrán beneficiarse de las nuevas características de HTML5, esto es algo que debemos asumir, pero está demostrado que el porcentaje de usuarios que trabaja con esta configuración es ínfimo.

Otra cosa que debemos tener en cuenta es que algunos navegadores no reconocerán los nuevos elementos y, aunque esto no sea demasiado problema, debemos asegurarnos de que el navegador ubica los elementos estructurales como elementos en bloque. Para ello, bastará con establecer la propiedad `display: block;` en la hoja de estilos.

Siempre deberíamos insertar la siguiente regla css en nuestra hoja de estilos:

```
article, section, aside, nav, header, footer, figure, figcaption, main
{
    display: block;
}
```

3. CSS3

Como aclaramos anteriormente, la nueva especificación de HTML (HTML5) no describe sólo los nuevos elementos HTML o el lenguaje mismo. La web demanda diseño y funcionalidad, no sólo organización estructural o definición de secciones.

La versión 2.0, asignada a la web para describir un nuevo nivel de desarrollo (web 2.0), 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.




La falta de soporte por parte de los navegadores era evidente, pero la organización responsable de los estándares web no tomó muy en serio las nuevas tendencias e intentó seguir su propio camino. Afortunadamente, algunas mentes brillantes siguieron desarrollando nuevos estándares en paralelo y pronto HTML5 nació. Posteriormente, la integración entre HTML, CSS y Javascript, bajo la tutela de HTML5, se impuso al resto de alternativas. En este nuevo paradigma, HTML se presenta junto con CSS y Javascript como un único instrumento integrado.




En este punto estudiaremos las contribuciones hechas por CSS3 a HTML5, y algunas de las propiedades que simplifican la vida de diseñadores y programadores.

Una nota sobre prefijos de proveedores

Hoy en día, si queremos utilizar muchas de las nuevas características de CSS3, estamos obligados a incluir un buen número de líneas adicionales de código. Esto se debe a que los fabricantes de navegadores han implementado muchas de las nuevas características de CSS3 utilizando sus propias versiones "prefijadas" de una propiedad.

Por ejemplo, para transformar un elemento en Firefox, es necesario utilizar la propiedad `-moz-transform`; para hacer lo mismo en los navegadores basados en WebKit, como Safari y Google Chrome, se utiliza la propiedad `-webkit-transform`. En algunos casos, tendremos hasta cuatro líneas de código para una única propiedad CSS. 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.

Puede parecer que esto elimina algunos de los beneficios de CSS3, pero los fabricantes de navegadores hacen esto por una buena razón: cuando las nuevas especificaciones están por definir las primeras implementaciones tienden a tener errores, así que, se proporcionan valores a las implementaciones actuales utilizando los prefijos de proveedores, y también se proporciona una versión “perenne” de cada propiedad a través de una declaración sin prefijo. Con el tiempo, cuando las especificaciones están perfectamente definidas y las propiedades refinadas, se eliminarán las propiedades con prefijo, propias de navegadores individuales.

Si quieres ver los navegadores que soportan una determinada propiedad CSS o un elemento HTML5, puedes consultar la página <http://caniuse.com>

Nuevos selectores CSS3

La nueva versión de CSS incorpora algunos nuevos selectores que pueden sernos útiles a la hora de crear nuestros diseños.

Selector general de elementos hermanos

Este nuevo selector generaliza el selector adyacente de CSS 2.1. Su sintaxis es `elemento1 ~ elemento2` y selecciona el `elemento2` que es hermano de `elemento1` y se encuentra detrás en el código HTML. En el selector adyacente la condición adicional era que los dos elementos debían estar uno junto al otro en el código HTML sin ningún otro elemento entre ambos, mientras que ahora la única condición es que uno esté detrás de otro, aunque hayan elementos entre ambos.

Si se considera el siguiente ejemplo:

```
h1 + h2 { ... } /* selector adyacente */  
h1 ~ h2 { ... } /* selector general de hermanos */
```

```
<h1>...</h1>  
<h2>...</h2>  
<p>...</p>  
<div>  
  <h2>...</h2>  
</div>  
<h2>...</h2>
```


El primer selector ($h1 + h2$) sólo selecciona el primer elemento $\langle h2 \rangle$ de la página, ya que es el único que cumple que es hermano de $\langle h1 \rangle$ y se encuentra justo detrás en el código HTML. Por su parte, el segundo selector ($h1 \sim h2$) selecciona todos los elementos $\langle h2 \rangle$ de la página salvo el segundo. Aunque el segundo $\langle h2 \rangle$ se encuentra detrás de $\langle h1 \rangle$ en el código HTML no son elementos hermanos, ya que no tienen el mismo elemento padre.

Nuevos selectores de atributo

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 }
```

- La regla con el selector $\wedge=$ será asignada a todo elemento $\langle p \rangle$ que contenga un atributo `name` con un valor que comience en "mi" (por ejemplo, "mitexto", "micasa").
- La regla con el selector $\$=$ será asignada a todo elemento $\langle p \rangle$ que contenga un atributo `name` con un valor que acabe en "mi" (por ejemplo "textomi", "casami").
- La regla con el selector $\ast=$ será asignada a todo elemento $\langle p \rangle$ que contenga un atributo `name` con un valor que incluya el texto "mi" (en este caso, el texto podría también encontrarse en el medio, como en "textomicasa").

En estos ejemplos usamos el elemento $\langle p \rangle$, 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.

Pseudo-elementos

Los pseudo-elementos de CSS 2.1 se mantienen en CSS 3, pero cambia su sintaxis y ahora se utilizan $::$ en vez de $:$ delante del nombre de cada pseudo-elemento:

- $::\text{first-line}$, selecciona la primera línea del texto de un elemento.
- $::\text{first-letter}$, selecciona la primera letra del texto de un elemento.
- $::\text{before}$, selecciona la parte anterior al contenido de un elemento para insertar nuevo contenido generado.
- $::\text{after}$, selecciona la parte posterior al contenido de un elemento para insertar nuevo contenido generado.

CSS 3 añade, además, un nuevo pseudo-elemento:

✚ ::selection, selecciona el texto que ha seleccionado un usuario con su ratón o teclado. Sólo unas pocas propiedades css pueden ser aplicadas a este selector, en concreto: color, background, cursor y outline.

```
::selection {color:red;background:yellow;}
```

El selector anterior pondrá la fuente de color rojo y el fondo amarillo para los elementos que el usuario seleccione con el ratón. Para que este selector funcione en Firefox, debemos añadir el prefijo -moz-, por lo tanto, para tener la funcionalidad disponible en todos los navegadores, haremos lo siguiente:

```
::selection {color:red;background:yellow;}  
::-moz-selection {color:red;background:yellow;}
```

Nuevas pseudo-clases

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

```
<div>  
  <p>Mi texto1</p>  
  <p>Mi texto2</p>  
  <p>Mi texto3</p>  
  <p>Mi texto4</p>  
</div>
```

El código anterior contiene cinco elementos <p>, uno que es hijo del documento y cuatro que 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(1) { background: #999999; }
```

La pseudo-clase nth-child() nos permite encontrar un hijo específico. El número entre paréntesis será el número de la posición del hijo, o índice. En el ejemplo se seleccionarán todos aquellos elementos <p> que sean los primeros hijos de su padre. En este caso, se seleccionará el párrafo de fuera del <div> y el primero de dentro. Si quisiéramos restringir la selección sólo a los hijos del <div>, haríamos lo siguiente:

```
div p:nth-child(1) { background: #999999; }
```

También se pueden utilizar las palabras clave even y odd como índice para seleccionar los hijos pares o impares respectivamente:

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

En este caso, se seleccionarán los párrafos 2 y 4.

Existen otras importantes pseudo-clases relacionadas con esta última, las veremos todas en la siguiente tabla:

Pseudo-clase	descripción
:nth-child(n)	Selecciona el enésimo hijo de su padre
:nth-last-child(n)	Selecciona el enésimo hijo de su padre contando desde el último
:nth-of-type(n)	Selecciona el enésimo hermano de su tipo
:nth-last-of-type(n)	Selecciona el enésimo hermano de su tipo comenzando desde el último
:first-child	Selecciona el primer hijo de su padre
:last-child	Selecciona el último hijo de su padre
:first-of-type	Selecciona el primer hermano de su tipo
:last-of-type	Selecciona el último hermano de su tipo
:only-child	Selecciona los que sean hijos únicos
:only-of-type	Selecciona los que sean los únicos hermanos de su tipo
:empty	Selecciona los elementos que no tienen hijos. Si un elemento contiene sólo texto no se considera vacío.

Tabla 2: Pseudo-clases para la selección de hijos o hermanos

Además de estas, CSS3 incorpora otras pseudo-clases que también son importantes:

- ✚ :enabled, :disabled Selecciona elementos de interfaz de usuario (formularios) según estén activados (:enabled) o desactivados (:disabled).
- ✚ :checked Selecciona elementos de interfaz de usuario que están en estado checked, por ejemplo, radio botones o casillas de verificación.
- ✚ La pseudo-clase :not() se utiliza realizar una negación:

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

Este selector asignará un margen de 0 píxeles a todos los elementos del documento excepto los elementos <p>. A diferencia del selector universal, la pseudo-clase :not() nos permite declarar una excepción.

Para indicar los elementos que no queremos seleccionar podemos utilizar cualquier selector válido:

```
/* Todos excepto los que tengan el class mitexto */
:not(.mitexto) { margin: 0px; }
/* Todos excepto el id menu */
:not(#menu)
```

Nuevas propiedades CSS3

Para facilitar el aprendizaje de las nuevas propiedades CSS3 vamos a ir aplicándolas sobre la misma plantilla o documento de ejemplo. Por este motivo, comenzaremos por crear un documento HTML sencillo con algunos estilos básicos.

Plantilla.html:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Curso HTML5 y CSS3</title>
  <link rel="stylesheet" href="estiloscss3.css">
  <!--[if lt IE 9]>
    <script
      src="http://html5shiv.googlecode.com/svn/trunk/html5.js"
    >
    </script>
  <![endif]-->
</head>
<body>
  <header id="principal">
    <h1>Curso HTML5 y CSS3</h1>
  </header>
</body>
</html>
```

Estiloscss3.css:

```
body { text-align: center; }

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

#titulo { font: bold 36px verdana, sans-serif; }
```

No hay nada nuevo en estas reglas, sólo los estilos necesarios para dar forma a la plantilla y crear una caja de 500 píxeles de ancho posicionada en el centro de la ventana, con un fondo gris, un borde y un texto grande en su interior que dice "Curso HTML5 y CSS3".

Vamos a destacar el uso de la propiedad `display: block` para asegurarnos de que el navegador interpreta el elemento `<header>` como un elemento en bloque. Para ver más usos de esta propiedad se recomienda consultar la siguiente URL: http://librosweb.es/css_avanzado/capitulo_4/propiedad_display.html.

Si visualizamos la página se verá así:



Curso HTML5 y CSS3

Bordes con esquinas redondeadas (border-radius)

Durante muchos años los diseñadores han sufrido intentando lograr el efecto de esquinas redondeadas en las cajas de sus páginas web, eran esa clase de cosas que nos hacía pensar: “debería ser fácil hacerlo”.

Gracias a la propiedad `border-radius` de CSS3, ahora podemos hacerlo de forma muy sencilla (si queremos asegurarnos la compatibilidad con el mayor número de navegadores posible, debemos usar los prefijos `-moz-` y `-webkit-` para que funcione 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. Las siguientes propiedades las añadiremos a la regla `#principal` de nuestro ejemplo:

```
-moz-border-radius: 20px;  
-webkit-border-radius: 20px;  
border-radius: 20px;
```

Curso HTML5 y CSS3

Sin embargo, como ocurre con las propiedades `margin` y `padding`, podemos también declarar un valor diferente para cada esquina. Los valores se aplicarán en el siguiente orden: esquina superior izquierda, esquina superior derecha, esquina inferior derecha y esquina inferior izquierda:

```
-moz-border-radius: 20px 10px 30px 50px;  
-webkit-border-radius: 20px 10px 30px 50px;  
border-radius: 20px 10px 30px 50px;
```

Curso HTML5 y CSS3

Al igual que con `margin` o `padding`, `border-radius` también puede trabajar sólo 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):

```
-moz-border-radius: 20px 50px;  
-webkit-border-radius: 20px 50px;  
border-radius: 20px 50px;
```

Curso HTML5 y CSS3

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 elipse:

```
-moz-border-radius: 50px / 20px;  
-webkit-border-radius: 50px / 20px;  
border-radius: 50px / 20px;
```

Curso HTML5 y CSS3

Para probar de una forma sencilla cómo quedaría esta propiedad puedes utilizar la herramienta disponible en la url: <http://www.cssmatic.com/border-radius>

Sombras en las cajas (box-shadow)

Otro muy buen efecto que había sido complicado de lograr hasta este momento, son las sombras. Tradicionalmente, los diseñadores combinaban imágenes, elementos y algunas propiedades CSS para generar sombras. Gracias a CSS3 y a la nueva propiedad `box-shadow` podemos aplicar sombras a nuestras cajas con una simple línea de código.

La propiedad `box-shadow` necesita al menos tres valores: Los dos primeros valores, expresados en píxeles, establecen el desplazamiento de la sombra y el tercero indicará el color de la misma.

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 píxeles o nulos posicionarán la sombra, exactamente, detrás del elemento, permitiendo la posibilidad de crear un efecto difuminado a su alrededor.

Veamos cómo quedaría nuestra caja, después de aplicarle los siguientes estilos:

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

Curso HTML5 y CSS3

Como se puede observar, al igual que ocurría con la propiedad `border-radius`, debemos incluir los prefijos oportunos para que funcione en todos los navegadores.

La sombra que hemos obtenido es sólida, sin gradientes o transparencias (no es, realmente, como una sombra suele aparecer). Existen algunos parámetros que nos van a permitir mejorar la apariencia de la sombra.

Otro valor que se puede agregar a la propiedad ya estudiada antes de indicar el color es la distancia de difuminación. Este será un valor expresado en píxeles que indicará como se va a difuminar la sombra:

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

Curso HTML5 y CSS3

Si añadimos un cuarto valor en píxeles antes del color, veremos cómo aumenta el tamaño de la sombra en todas las direcciones. Este efecto cambia un poco la naturaleza de la sombra expandiendo el área que cubre:

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

Curso HTML5 y CSS3

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

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

Curso HTML5 y CSS3

Es importante tener en cuenta que las sombras no expanden el elemento o incrementan su tamaño, por lo que, siempre tendremos que controlar cuidadosamente que el espacio disponible es suficiente para que la sombra sea expuesta y correctamente dibujada en la pantalla sin superponerse con otros elementos cercanos.

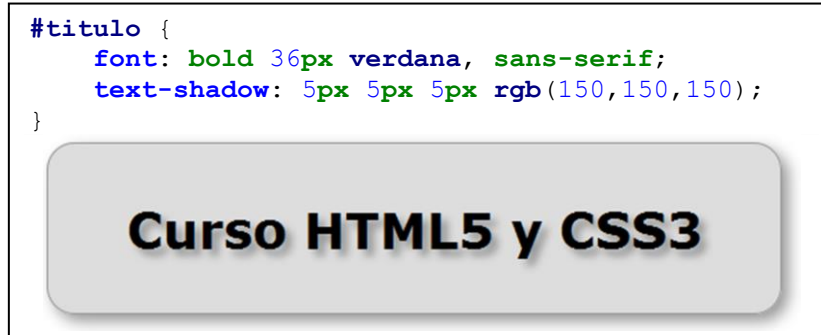
Para probar de una forma sencilla cómo quedaría esta propiedad puedes utilizar la herramienta disponible en la url: <http://www.cssmatic.com/box-shadow>

Sombras en los textos (`text-shadow`)

La propiedad `box-shadow` fue diseñada especialmente para ser aplicada en cajas. Si intentamos aplicar este efecto a un elemento `<h1>`, 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`.

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

Veamos cómo queda nuestro título al aplicarle una sombra:



Para probar de una forma sencilla cómo quedaría esta propiedad puedes utilizar la herramienta disponible en la url: <http://css3gen.com/text-shadow/>

Fondos con gradiente

Los gradientes son uno de los efectos más utilizados en la web. Anteriormente, era necesario el uso de imágenes para lograr este efecto, pero ahora es realmente fácil de hacer usando CSS.

Los gradientes son configurados como fondos, por lo que podemos usar las propiedades `background` o `background-image` para declararlos.

Tenemos dos tipos de gradientes: gradiente lineal y gradiente radial.

Gradiente lineal

Los gradientes lineales los indicamos con la función `linear-gradient`, que recibe los siguientes parámetros:

- ✚ Punto de comienzo: indica el punto donde comenzará el gradiente. Puede ser especificado en píxeles, porcentaje o usando las palabras clave `top`, `bottom`, `left` y `right`. También puede ser reemplazado por un ángulo para declarar una dirección específica del gradiente.
- ✚ Color inicial: indica con que color comenzará el gradiente.
- ✚ Color final: indica con que color terminará el gradiente.

Con el parámetro punto de comienzo, debemos tener en cuenta que hay una pequeña diferencia entre la función estándar y las específicas de los navegadores: en la función estándar debemos indicar la palabra `to` y la dirección del gradiente, mientras que en las específicas de los navegadores indicamos la posición de comienzo del gradiente.

Al igual que las propiedades anteriores cada navegador tendrá sus prefijos para trabajar con esta función.

A continuación, veremos cómo queda nuestra plantilla de ejemplo aplicando un gradiente a la regla #principal:

```
background: -webkit-linear-gradient(top, #FFFFFF, #006699);  
background: -moz-linear-gradient(top, #FFFFFF, #006699);  
background: linear-gradient(to bottom, #FFFFFF, #006699);
```

Curso HTML5 y CSS3

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

Curso HTML5 y CSS3

También podemos declarar los puntos de terminación para cada color de la siguiente forma:

```
background: -webkit-linear-gradient(30deg, #FFFFFF 50%, #006699 90%);  
background: -moz-linear-gradient(30deg, #FFFFFF 50%, #006699 90%);  
background: linear-gradient(30deg, #FFFFFF 50%, #006699 90%);
```

Curso HTML5 y CSS3

Gradiente radial

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

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

Curso HTML5 y CSS3

Existen dos posibles valores para la forma: `circle` y `ellipse` (el valor por defecto será `ellipse`). Además, como puedes observar en el ejemplo, para cada color se debe indicar la posición donde las transiciones comienzan (en el ejemplo, el gradiente comenzaría en la posición 0% y terminaría en la posición 200%).

Para practicar con gradientes puedes utilizar la herramienta disponible en la url: <http://css3gen.com/gradient-generator/>

Transparencias

Tenemos dos formas de definir las transparencias de un elemento: expresando el color del elemento con transparencia o indicando un valor que indicará la opacidad del elemento.

RGBA

Hasta ahora, hemos declarado los colores como sólidos utilizando valores hexadecimales o la función `rgb()`. CSS3 ha agregado una nueva función llamada `rgba()` que simplifica la asignación de colores y transparencias.

La función `rgba()` tiene cuatro atributos. Los tres primeros 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.

Por ejemplo, podemos mejorar la sombra del texto de nuestro título de ejemplo utilizando esta función de la siguiente forma:

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

Curso HTML5 y CSS3

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.

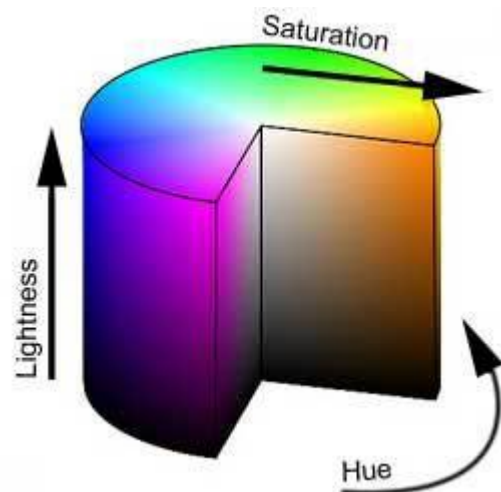
Esta función, además, resuelve un problema previo provocado por la propiedad `opacity`. En versiones previas de CSS, teníamos que usar diferentes técnicas en diferentes navegadores para hacer un elemento transparente. Todas presentaban el mismo problema: el valor de opacidad de un elemento era heredado por sus hijos. Ese problema fue resuelto por `rgba()`, y ahora podemos asignar un valor de opacidad al fondo de una caja sin afectar a su contenido.

HSLA

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

La función `hsla()` es, simplemente, otra función para generar colores que, en lugar de utilizar distintos valores para los colores rojo, verde y azul, utiliza distintos valores de tono, saturación y luminosidad.

La sintaxis es la siguiente: `hsla(tono, saturación, luminosidad, opacidad)`.



- tono representa el color extraído de un círculo de color imaginario 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.

- saturación es representado en porcentaje, desde 0% (escala de grises) a 100% (todo color o completamente saturado).

✚ luminosidad es también un valor en porcentaje desde 0% (completamente oscuro) a 100% (completamente iluminado). El valor 50% representa luminosidad normal o promedio.

✚ opacidad funciona igual que en la función `rgba()`.

Si utilizamos esta función para cambiar el color de la fuente de nuestro título, obtendremos el siguiente resultado:

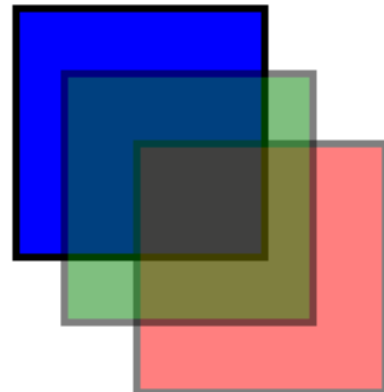
```
color: hsla(120, 100%, 50%, 0.5);
```



Propiedad opacity

El valor de la propiedad `opacity` se establece mediante un número comprendido entre 0.0 y 1.0 (al igual que antes, 0.0 será transparente y 1.0 totalmente opaco). En el siguiente ejemplo, se establece la propiedad `opacity` con un valor de 0.5 para conseguir una transparencia del 50% sobre dos de los elementos `<div>`:

```
#segundo, #tercero { opacity: 0.5; }  
#primero { background-color: blue; }  
#segundo { background-color: red; }  
#tercero { background-color: green; }
```

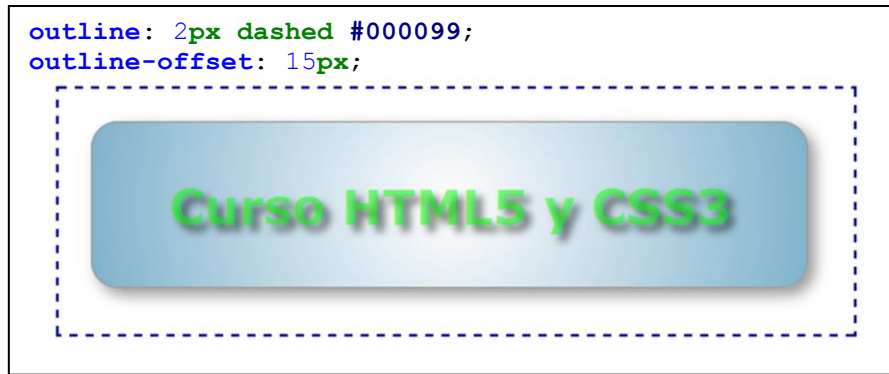


Outline

La propiedad `outline` es una vieja propiedad CSS que ha sido expandida en CSS3 para incluir un valor de desplazamiento.

Esta propiedad se usaba para crear un segundo borde, y ahora ese borde puede ser mostrado alejado del borde real del elemento.

Por ejemplo, podemos añadir un segundo borde de 2 píxeles y un desplazamiento de 15 píxeles a la caja de nuestro título, si ponemos las siguientes propiedades en la regla `#principal`:



La propiedad `outline` tiene similares características y usa los mismos parámetros que `border`. La propiedad `outline-offset` sólo necesita un valor en píxeles.

Border-image

Los posibles efectos logrados por las propiedades `border` y `outline` están limitados a líneas simples y sólo 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.

Al igual que otras propiedades `css`, la propiedad `border-image` es una propiedad compuesta por varios valores, y estos valores simples podemos indicarlos mediante otras propiedades más simples que veremos a continuación.

Imagen de borde: `border-image-source`

La propiedad `border-image-source` establece la imagen a utilizar como imagen de borde. El elemento tiene que tener definido un borde² y la imagen se amplía o reduce para mostrarse completa en función del ancho que le hayamos puesto a la propiedad `border`. Si sólo se utiliza esta propiedad, la imagen se muestra en las cuatro esquinas del elemento.

² Es importante que tenga también definido un estilo de borde (propiedad `border-style`), ya que, si no, no se mostrará correctamente en Firefox

```
#principal {  
  display: block;  
  width: 500px;  
  margin: 50px auto;  
  padding: 15px;  
  text-align: center;  
  border: 50px;  
  border-style: solid;  
  border-image-source: url("fo_bola_100.png");  
}
```

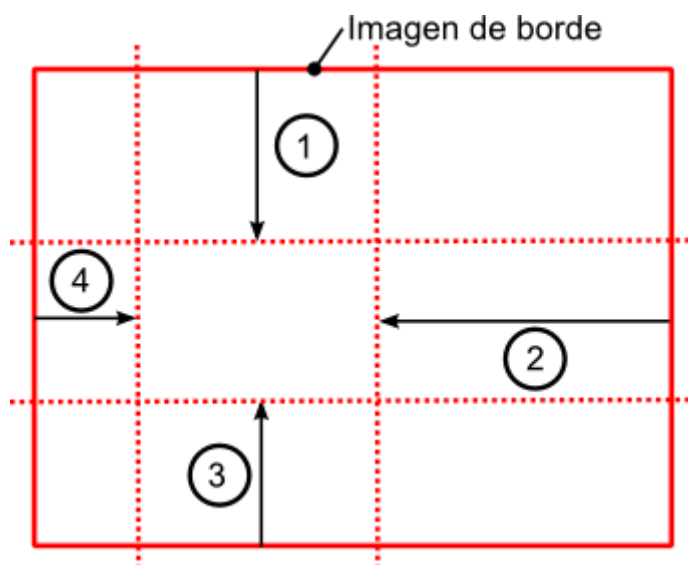


Troceado de la imagen de borde: border-image-slice

La propiedad `border-image-slice` permite trocear la imagen de borde, de manera que cada trozo se coloque en un lado del borde.

Los posibles valores de la propiedad `border-image-slice` son:

- ✚ De uno a cuatro valores en píxeles. Estos valores indican a qué distancia de los bordes se recorta la imagen, como muestra la imagen siguiente:



Cada uno de los ocho trozos (descartando el trozo central) se utiliza en cada una de las ocho zonas correspondientes del borde. Al igual que en otras propiedades como `margin` o `padding`, podemos indicar sólo dos valores, o incluso, un único valor:

```
border: 27px;  
border-style:solid;  
border-image-source: url("fo_pastilla_60.png");  
border-image-slice: 27;
```



```
border-width: 50px 80px 20px 80px;  
border-style:solid;  
border-image-source: url("fo_ventana.png");  
border-image-slice: 50 80 20 80;
```



fill: este valor hace que la parte central de la imagen se utilice para rellenar el elemento.

```
border: 27px;  
border-style:solid;  
border-image-source: url("fo_pastilla_60.png");  
border-image-slice: 27 fill;
```



Repetición de la imagen de borde: border-image-repeat

La propiedad border-image-repeat indica de qué forma se realizará el relleno de los bordes del elemento. Los posibles valores son:

- stretch: la imagen se estira o encoge para ocupar todo el espacio necesario

```
border: 40px;  
border-style:solid;  
border-image-source: url ("fo_hormigas_210.png");  
border-image-slice: 70;  
border-image-repeat: stretch;
```



repeat: la imagen se repite para ocupar todo el espacio necesario



round: la imagen se repite el máximo número entero de veces posible hasta rellenar el espacio, la imagen será reescalada para que ninguna de las repeticiones quede cortada.



Si queréis reproducir estos ejemplos, podéis encontrar las imágenes utilizadas en el archivo recursos.zip del aula virtual.