

# 3

## Tema

# HTML5 & CSS3 AVANZADO

---

En este segundo tema se pretenden conseguir los siguientes objetivos:

- Entender la importancia de la semántica de los elemento HTML5.
  - Conocer las nuevas etiquetas de HTML5.
  - Utilizar correctamente las nuevas etiquetas estructurales en función de su semántica.
  - Conocer los nuevos selectores de CSS3.
  - Crear diseños web vistosos utilizando degradados, sombras, transparencias, bordes redondeados y/o bordes con imágenes.
-

## Sumario

Introducción.....	3
HTML5.....	3
La semántica de los elementos.....	4
Nuevos elementos estructurales.....	5
Cabecera.....	6
El elemento section.....	7
El elemento article.....	8
El elemento nav.....	9
El elemento aside.....	9
El elemento footer.....	10
El elemento main.....	10
Organizando el diseño de la página con elementos HTML5.....	12
El esquema del documento.....	12
Otros elementos semánticos.....	14
Los elementos figure y figcaption.....	14
El elemento mark.....	14
El elemento time.....	15
Elementos con la semántica redefinida.....	17
El elemento small.....	17
El elemento cite.....	17
El elemento address.....	17
Categorías de elementos HTML5.....	18
Navegadores antiguos.....	19
CSS3.....	21
Los módulos CSS3.....	21
Etapas en la concepción de las CSS3.....	22
Los prefijos de los navegadores.....	23
El sitio Can I use.....	24
Algunos otros selectores más.....	26
Selector general de elementos hermanos.....	26
Nuevos selectores de atributo.....	27
Las pseudoclases de primer y último hijo.....	27
La pseudoclase de primeros hijos.....	28
La pseudoclase de últimos hijos.....	31
Las pseudoclases del primer y del último hijo de un tipo.....	31
Las pseudoclases de los primeros y los últimos hijos de un tipo.....	31
La pseudoclase de los elementos sin hermanos.....	33
La pseudoclase de los elementos sin hermanos de un tipo.....	34
La pseudoclase de los elementos vacíos.....	34
La pseudoclase de negación.....	36
Nuevas propiedades CSS3.....	37

Fuentes tipográficas.....	37
Las fuentes incrustadas.....	37
Los formatos de fuente.....	37
Las fuentes en línea.....	40
Las Google Fonts.....	40
Ajustar el tamaño de los caracteres.....	41
Otras propiedades tipográficas.....	43
El contenido generado.....	43
El contenido textual antes y después.....	43
Los contadores.....	45
El desborde de texto.....	47
Los colores.....	49
Los formularios.....	49
Dar formato.....	49
Redimensionar un campo.....	49
Pseudoclasas para formularios.....	51
Los campos requeridos y los opcionales.....	54
Dar formato al «foco».....	56
Validar la información introducida.....	57
Los contornos dinámicos.....	61
Los cursores.....	62
Bordes.....	65
Los bordes de fantasía.....	67
Las esquinas redondeadas.....	69
Esquinas con elipses.....	70
Especificar el cálculo de la anchura.....	71
El desborde de contenido.....	73
Las sombras paralelas.....	75

## Introducción

En el anterior tema hemos visto una introducción y en algunos casos ya hemos visto algunas etiquetas y estilos propios de hml5 y css3.

En este tema vamos a estudiar en profundidad las nueva características de ambas tecnologías. Y los motivos que han llevado a introducir estos nuevos elementos.

## 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>`.

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

## La semántica de los elementos

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.

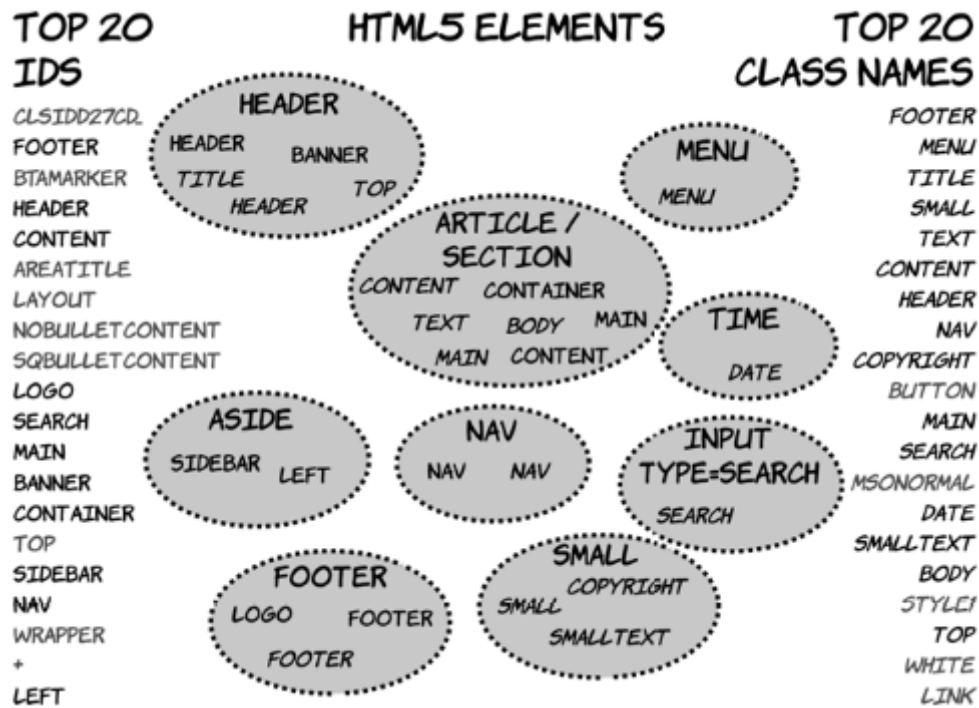
El hecho de que seamos capaces de describir la estructura del documento de esta manera es importante, porque, de esta forma, **podremos separar los contenidos que mostramos de la forma en que los mostramos**. El resultado es que una página web, si está bien estructurada, podrá leerse fácilmente en un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

Con este objetivo se observó que 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.

Con esto en mente se realizó un estudio , 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.



## Nuevos elementos estructurales

Ya vimos en el tema anterior 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

## Cabecera

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.

`<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.

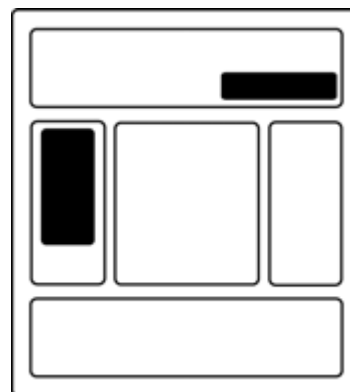


- Artículos de revistas o periódicos.
- Las entradas de un blog.
- los comentarios enviados por los usuarios, etc.

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).

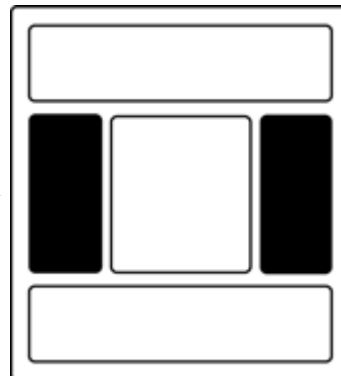
### El elemento `aside`

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.

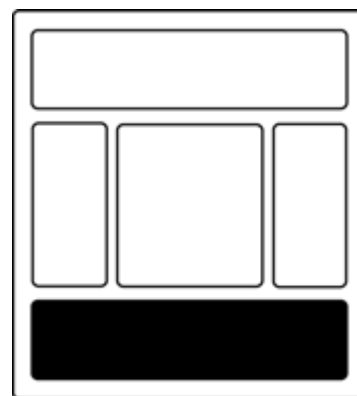
Algunos usos posibles para un `<aside>` serían: una barra lateral, una lista secundaria de enlaces, o un espacio para la publicidad. También hay que señalar que el elemento `<aside>` (como en el caso del `<header>`) no se define por su posición en la página. Podría ser en un "lado", o podría estar en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

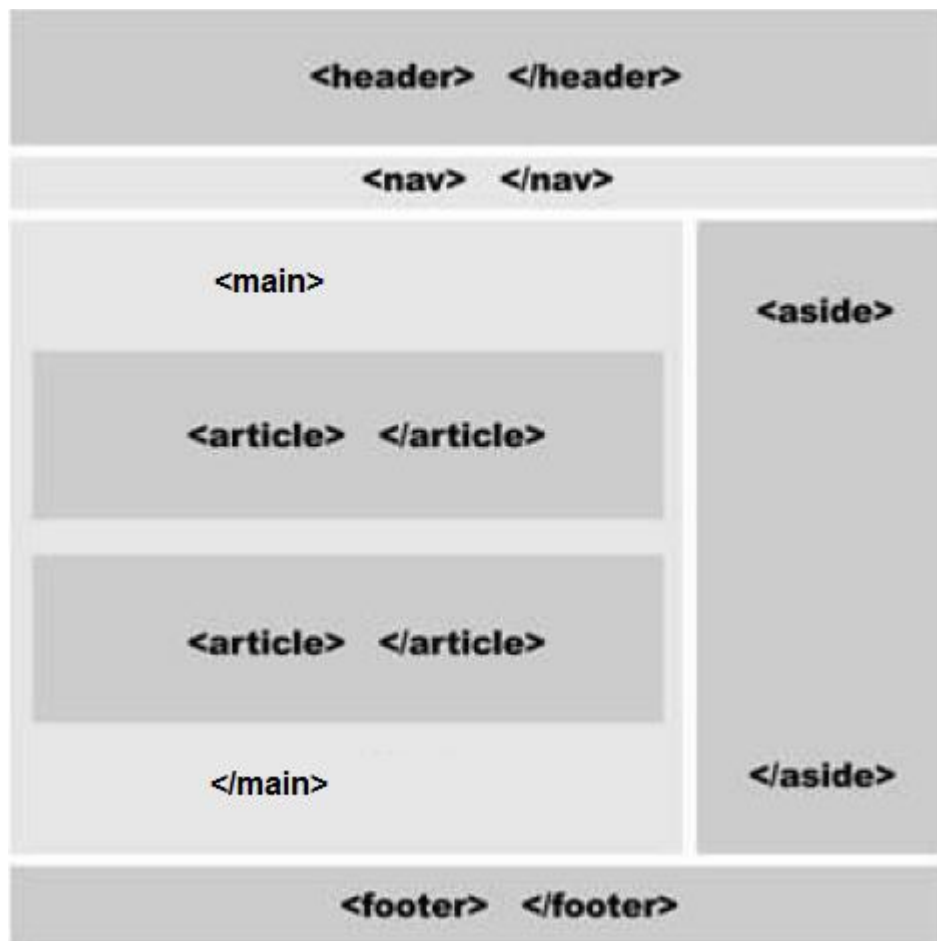
- Representa el contenido principal del cuerpo (`<body>`) de una web o aplicación.

- 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



### Imagen : típico diseño utilizando elementos HTML5

En esta imagen se muestra el típico diseño presentado en el tema anterior, 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>` Título `</h1>`

<h2>SubTitulo</h2>

<h3>Otro nivel más</h3>

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

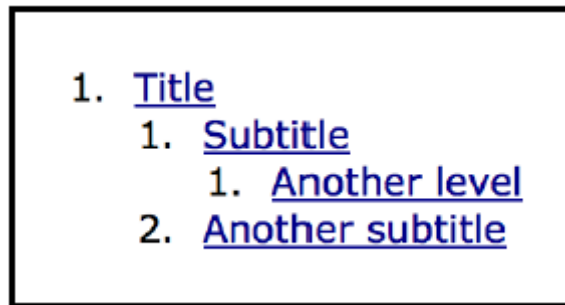


Imagen: Esquema de documento

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" 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.

## 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>`

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; 2015 Julio Martinez</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, los elementos de una página HTML pertenecen 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 `<em>` , `<strong>` , `<cite>` y similares.
- **Contenido incrustado (embedded content):** Elementos que son incrustados en una página, como `<img>` , `<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.

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](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;
}
```

# CSS3

## Los módulos CSS3

Las CSS tienen como objetivo separar la estructura de la página (con su contenido, creado con HTML) del diseño y el formato de la propia página. Las **Cascading Style Sheets Level 1** se publicaron como **Recommendation** (documento finalizado por el W3C) el 17 de diciembre de 1996 y se revisaron el 11 abril de 2008:

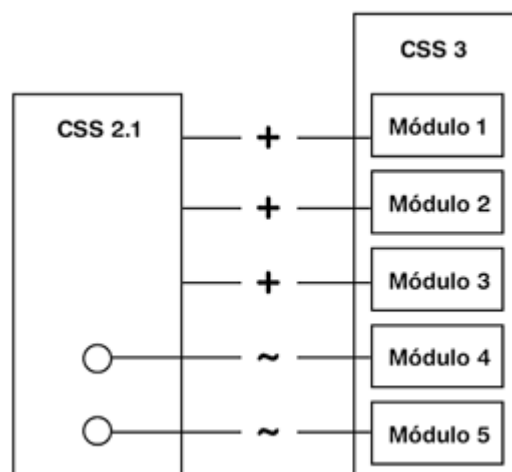
<http://www.w3.org/TR/CSS1/>. Las CSS 2.1 se han publicado como

**Recommendation** el 7 de junio de 2011: <http://www.w3.org/TR/CSS21/>.

Cada recomendación de CSS (1, 2 y 2.1) implicó un enorme trabajo, desde los borradores (**Working Draft**) hasta el documento finalizado (**Recommendation**). La recomendación final es un documento único con un número impresionante de páginas.

Para la versión siguiente de las CSS, la versión 3, el W3C decidió cambiar su modo de trabajo.

- **Primer punto importante:** en lugar de hacer una recomendación «enorme», monolítica y de difícil mantenimiento, el W3C ha trabajado de manera modular. Las propiedades se han reagrupado en módulos funcionales independientes que evolucionan a su propio ritmo. De este modo, los navegadores han podido probar e implementar las nuevas propiedades de forma progresiva, módulo a módulo.
- **Segundo punto,** cuya comprensión resulta muy importante: las CSS3 no reemplazan las CSS 2.1. Las CSS3 son una continuación, un complemento, una prolongación de las CSS 2.1. Las CSS 2.1 se mantienen plenamente vigentes y usables. Lo único que hacen las CSS3 es añadir nuevas propiedades y redefinir algunas de las que ya existían desde las CSS 2.1.



## Etapas en la concepción de las CSS3

Cada módulo pasa por una serie de etapas perfectamente estandarizadas por el W3C.

La primera etapa se centra en el borrador de trabajo, el **Working Draft (WD)**. En este documento se recogen las propuestas de propiedades para que toda la comunidad (público, empresas, navegadores...) puedan probar su implementación.

A continuación, se entra en la etapa **Last Call (LC)**. El W3C considera que su trabajo es técnicamente satisfactorio, anuncia la fecha de finalización de las pruebas y toma en cuenta los comentarios de los grupos de trabajo.

Después viene la **Candidate Recommendation (CR)**. El W3C ha revisado prolongadamente su copia, de acuerdo con los grupos de trabajo y con las pruebas de puesta en práctica. El documento casi está finalizado.

La última etapa es la de **Recommendation (REC)**. El W3C publica el documento, que desempeña el papel de estándar y al cual deben referirse todos los navegadores si quieren respetar estos estándares.

Además, es preciso entender que las CSS3 están «vivas», evolucionan, pasan del Estado **Working Draft** a la **Recommendation**. Cada módulo evolucionará a su propia velocidad. Por lo tanto, conviene referirse con frecuencia a la página de la evolución de las CSS3 en el sitio del W3C, la página **Current Work**:

<http://www.w3.org/Style/CSS/current-work>

En la zona **Table of specifications**, puede observar la evolución de los diversos módulos, organizados en varias categorías. Este es el estado que presentaban en mayo de 2015:

- **Completed:** módulos terminados, en **Recommendation**.
- **Testing:** módulos en estado **Candidate Recommendation**.
- **Refining:** módulos que se están redefiniendo.
- **Revising:** módulos que se están reestructurando.
- **Exploring:** módulos que aún están en desarrollo.
- **Rewriting:** módulos que se están reescribiendo por completo.

- **Abandoned:** módulos que sencillamente se han abandonado.

Esta es la razón de que sea imprescindible consultar de forma regular este sitio, para estar al corriente de la evolución de los módulos CSS3.

## Los prefijos de los navegadores

La concepción de las CSS3 en módulos independientes supone para los navegadores un trabajo considerable de implementación de las nuevas propiedades en los motores de renderizado. Y, además, mientras los módulos no pertenezcan a la categoría **Candidate Recommendation**, las propiedades pueden cambiar, lo que complica todavía más el trabajo de los navegadores. Con objeto de trabajar a mayor velocidad, el W3C propone que los navegadores utilicen un prefijo específico para cada uno de ellos, con la finalidad de probar la implementación de las nuevas propiedades. En cuanto la especificación llegue al nivel **CR, Candidate Recommendation**, los prefijos resultaran inútiles.

He aquí estos prefijos (designación según proveedor):

- -moz-: para el motor de renderizado Gecko que utiliza Mozilla Firefox.
- -webkit-: para el motor de renderizado WebKit que utilizan Apple Safari y Google Chrome, con sus bifurcaciones.
- -o-: para el motor de renderizado de Opera.
- -ms-: para el motor de renderizado de Microsoft Internet Explorer.
- -khtml-: para el motor de renderizado KHTML que usan varios navegadores bajo Linux.

Tomemos un ejemplo preciso. Supongamos que queremos utilizar la propiedad border-radius, que permite crear ángulos redondeados en las etiquetas <div>, por ejemplo.

He aquí la sintaxis que se debe utilizar para que la reconozcan todos los navegadores:

```
header {  
    -moz-border-radius: 10px;  
    -webkit-border-radius: 10px;
```

```
-o-border-radius: 10px;  
-ms-border-radius: 10px;  
-khtml-border-radius: 10px;  
border-radius: 10px;  
}
```

Las primeras líneas se han escrito respectivamente para cada motor de renderizado que hemos mencionado en la lista previa. La última línea corresponde al uso de la propiedad estándar para todos los navegadores que reconocerán más adelante esta propiedad, en el momento en que el W3C la considere finalizada.

**El orden de las líneas es importante. Al ubicar la propiedad «estándar» en la última posición, se asegura de que esta se impone a las líneas precedentes.** Es decir, primero debe indicar las propiedades con los prefijos (las cuales pueden evolucionar) y después finalizar con la propiedad «oficial».

Esta sintaxis puede parecer pesada, pero facilita la mejora de la evolución y la portabilidad de las propiedades que aún no son oficiales.

Para ayudarle a gestionar fácilmente los prefijos, existen numerosos servicios en línea:

- **Net Tuts Prefixr:** <http://prefixr.com/>
- **prefixMyCSS:** <http://prefixmycss.com/>
- **Online CSS Prefixer:** <http://www.css-prefix.com>

También puede utilizar un prefijador en JavaScript:

<http://leaverou.github.io/prefixfree/>

Pero, aun en este caso, debe realizar un trabajo de verificación. Le corresponde a usted comprobar si los prefijos deben usarse con una propiedad en concreto. Para ello, consulte el sitio **Current Work** del W3C, <http://www.w3.org/Style/CSS/current-work#CSS3>, y el sitio **Can I use**, que vamos a ver a continuación.

## El sitio Can I use

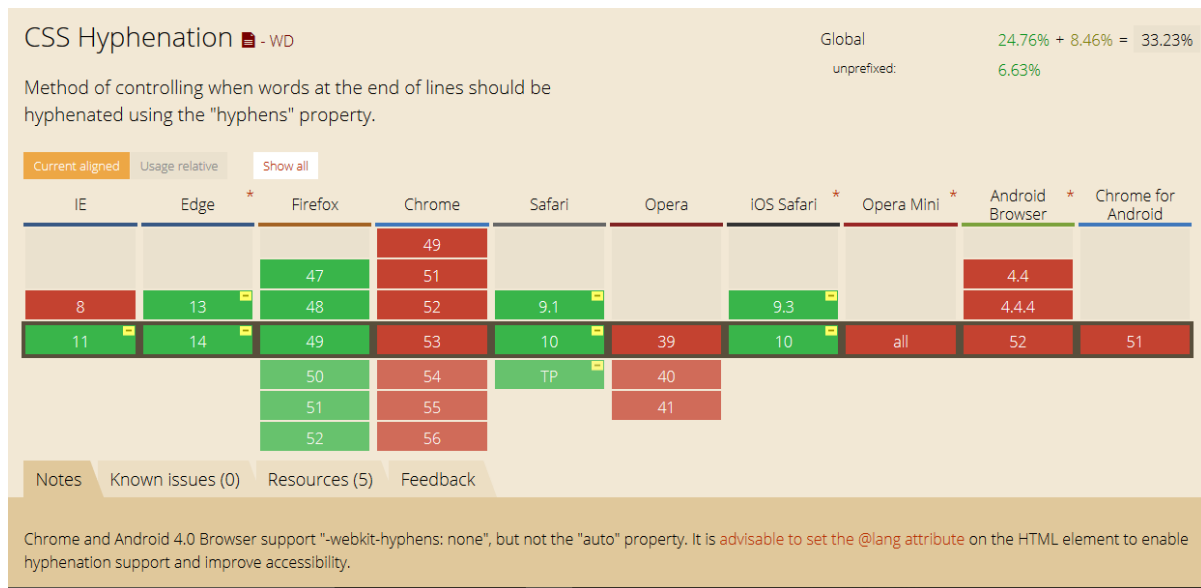
Como acabamos de ver, el sitio **Current Work** del W3C permite seguir la evolución de los módulos que componen las CSS3. Sin embargo, no puede visualizar la implementación de las propiedades en los navegadores, es decir, no visualiza las



propiedades que pueden usarse en el instante «t». Para ello, debe consultar, también regularmente, el sitio **Can I use**: <http://caniuse.com>

Este sitio muestra las compatibilidades entre los navegadores y los lenguajes HTML5 y CSS3 (y también SVG, JavaScript API y otros estándares del Web).

En el campo **Can I use**, introduzca la propiedad CSS3 o el elemento HTML5 que desee utilizar. En este ejemplo, hemos probado con la propiedad hyphens.



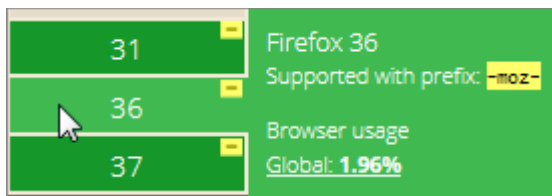
Aquí dispone de la tabla de compatibilidad entre la propiedad introducida y las principales versiones de los navegadores actuales. La tabla emplea cuatro colores:

- **Verde:** Supported.
- **Rojo:** Not supported.
- **Verde flojo:** Partial support.
- **Gris:** Support unknown.

A continuación, dispone de cuatro pestañas que son también muy importantes para obtener toda la información técnica útil: **Notes**, **Known issues**, **Resources** y **Feedback**.

Finalmente, se indica, para cada navegador, si es necesario el uso de prefijos.

En este ejemplo (propiedad hyphens), el uso de prefijos es necesario para los navegadores que reconocen esta propiedad. Si pasa el puntero del ratón por la casilla, se le indicará el prefijo de estos navegadores.



## Algunos otros selectores más

La nueva versión de CSS incorpora algunos nuevos selectores que pueden sernos útiles a la hora de crear nuestros diseños. En el anterior tema hemos visto algunos pero vamos a ver algunos más.

### 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 `<h2>` de la página, ya que es el único que cumple que es hermano de `<h1>` y se encuentra justo detrás en el código HTML. Por su parte, el segundo selector (`h1 ~ h2`) selecciona todos los elementos `<h2>` de la página salvo el segundo. Aunque el segundo `<h2>` se encuentra

detrás de <h1> 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 ^= será asignada a todo elemento <p> 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 <p> que contenga un atributo name con un valor que acabe en "mi" (por ejemplo "textomi", "casami").
- La regla con el selector \*= será asignada a todo elemento <p> 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 <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.

### Las pseudoclasas de primer y último hijo

La pseudoclase :first-child permite apuntar al primer hijo de un elemento padre.

He aquí un ejemplo sencillo:

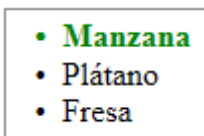
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las pseudoclasas de primer y último
hijo</title>
  <meta charset="UTF-8" />
  <style>
```

```

        li:first-child {
            font-weight: bold;
            color: green;
        }
    </style>
</head>
<body>
<ul>
    <li>Manzana</li>
    <li>Plátano</li>
    <li>Fresa</li>
</ul>
</body>
</html>

```

Cuando se visualice este código, el primer ítem <li> se mostrará en verde y negrita.



Siguiendo exactamente el mismo principio, la pseudoclase :last-child permite apuntar al último hijo de un elemento padre.

### La pseudoclase de primeros hijos

La pseudoclase :nth-child(x) permite apuntar a los x primeros hijos de un elemento padre. El número de hijos viene dado por el argumento x entre paréntesis.

El valor puede ser un número fijo: 2 o 5, por ejemplo.

Ejemplo:

```

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

```

```
<title>La pseudoclase de primeros hijos</title>
<meta charset="UTF-8" />
<style>
    li:nth-child(2) {
        font-weight: bold;
        background-color: gold;
    }
</style>
</head>
<body>
<ul>
    <li>Manzana</li>
    <li>Plátano</li>
    <li>Fresa</li>
    <li>Cereza</li>
    <li>Pera</li>
    <li>Kiwi</li>
    <li>Mango</li>
</ul>
</body>
</html>
```

Resultado que se muestra:

- Manzana
- **Plátano**
- Fresa
- Cereza
- Pera
- Kiwi
- Mango

El valor puede referirse a los hijos pares (even) o a los impares (odd).

Veamos el ejemplo anterior con `li:nth-child(odd)`.

- **Manzana**
- Plátano
- **Fresa**
- Cereza
- **Pera**
- Kiwi
- **Mango**

El valor  $x$  también puede ser un cálculo en formato:  $an+b$ .  $n$  representa un valor que comienza por 0 y que se incrementa en 1 con cada paso, y puede ser positivo o negativo.

Primer ejemplo :

`:nth-child(n+3)` permite dejar al margen los dos primeros hijos.

Este es el cálculo:

$0+3=3$ , el tercer hijo,

$1+3=4$ , el cuarto hijo,

$2+3=5$ , el quinto hijo...

Esto es lo que se muestra:

- Manzana
- Plátano
- **Fresa**
- **Cereza**
- **Pera**
- **Kiwi**
- **Mango**

Segundo ejemplo (**02\_14.html**):

`:nth-child(3n+1)` permite apuntar a un hijo de cada 3 comenzando por el primero.

Este es el cálculo:

$(3 \times 0)+1=1$ , el primer hijo,

$(3 \times 1)+1=4$ , el cuarto hijo,

$(3 \times 2)+1=7$ , el séptimo hijo,

$(3 \times 3) + 1 = 10$ , el décimo hijo...

Esto es lo que se muestra:

- Manzana
- Plátano
- Fresa
- Cereza
- Pera
- Kiwi
- Mango

### La pseudoclase de últimos hijos

La pseudoclase `:nth-last-child()` permite apuntar a los  $x$  últimos hijos de un elemento padre. Funciona del mismo modo que `:nth-child()`.

### Las pseudoclases del primer y del último hijo de un tipo

Las nuevas pseudoclases `:first-of-type` y `:last-of-type` permiten apuntar al primer y al último elemento de un tipo especificado.

He aquí un ejemplo de la lista `<ul>` anterior, tomando como elementos de destino el primer y el último ítem `<li>`.

```
li:first-of-type, li:last-of-type {  
    background-color: gold;  
}
```

Se muestra lo siguiente:

- Manzana
- Plátano
- Fresa
- Cereza
- Pera
- Kiwi
- Mango

### Las pseudoclases de los primeros y los últimos hijos de un tipo

La pseudoclase `:nth-of-type()` permite apuntar al  $n$ ésimo elemento de un tipo especificado. Los argumentos posibles son idénticos a los que hemos visto con la pseudoclase `:nth-child()`.

He aquí un ejemplo, muy sencillo, con la aplicación de formato (<strong> y <em>) a determinadas palabras en un párrafo (<p>):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las pseudoclases</title>
  <meta charset="UTF-8" />
  <style>
    strong:nth-of-type(2) {
      background-color: gold;
    }
  </style>
</head>
<body>
<p>Donec ullamcorper <strong>nulla</strong> non metus
auctor <em>fringilla</em>. Fusce dapibus, tellus ac
cursus commodo, <strong>tortor</strong> mauris
condimentum nibh,
ut fermentum massa justo sit amet risus. Cum sociis
natoque <strong>penatibus</strong> y magnis dis
parturient montes.</p>
</body>
</html>
```

El elemento <p> tiene como primer elemento hijo un <strong>, como segundo un <em>, como tercero y cuarto otro <strong>. Deseamos apuntar al segundo <strong>, y no al segundo elemento hijo.

La clase :nth-last-of-type() permite apuntar al enésimo último elemento de un tipo dado.



## La pseudoclase de los elementos sin hermanos

La pseudoclase `:only-child` apunta a los elementos que no tienen hermanos.

Tomemos el ejemplo de un texto (`<p>`) en el que resaltamos algunos términos con `<strong>`. Deseamos destacar los párrafos que incluyen un único resalte `<strong>`.

Este es el código de este ejemplo .

El segundo párrafo `<p>` incluye dos `<strong>`; el primero, uno solo.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>La pseudoclase de los elementos sin
hermanos</title>
  <meta charset="UTF-8" />
  <style>
    strong:only-child {
      background-color: gold;
    }
  </style>
</head>
<body>
<p>Donec sed odio dui. Aenean <strong>lacinia
bibendum</strong> nulla sed consectetur.Donec id elit non
mi porta gravida at eget metus. Nullam id dolor id nibh
ultricies vehicula ut id elit.</p>
<p>Nullam id dolor id <strong>nibh</strong> ultricies
vehicula ut id elit. Cum sociis natoque penatibus et
magnis dis parturient montes, <strong>nascetur</strong>
ridiculus mus.</p>
</body>
```

```
</html>
```

## La pseudoclase de los elementos sin hermanos de un tipo

La pseudoclase `:only-of-type` apunta a los elementos de un tipo dado que no tienen hermanos.

Retomemos el anterior ejemplo el principio precedente añadiendo un resalte `<em>` en el primer párrafo.

```
<p>Donec sed odio dui. Aenean <strong>lacinia  
bibendum</strong> nulla sed consectetur.Donec id elit non  
mi porta gravida at eget metus. Nullam id dolor id  
<em>nibh ultricies</em> vehicula ut id elit.</p>  
<p>Nullam id dolor id <strong>nibh</strong> ultricies  
vehicula ut id elit. Cum sociis natoque penatibus et  
magnis dis parturient montes, <strong>nascetur</strong>  
ridiculus mus.</p>
```

El estilo modificado:

```
strong:only-of-type {  
    background-color: gold;  
}
```

Con el estilo CSS anterior, el `<strong>` del primer párrafo ya no aparece con un fondo de color, ya que este párrafo contiene ahora dos resaltes. El `<strong>` no está solo; también hay un `<em>`. Con la pseudoclase `:only-of-type` ahora podemos especificar el tipo de elemento de forma aislada.

## La pseudoclase de los elementos vacíos

La **pseudoclase `:empty`** apunta a elementos que están vacíos, que no contienen ningún elemento. Esto puede resultar práctico si trabaja en una página web que se construye dinámicamente; puede incluir elementos (`<li>`, `<td>`, `<strong>`...) que están vacíos. Con esta pseudoclase es posible resaltarlas.

Veamos el código de este ejemplo:

```
<!DOCTYPE html>
```

```

<html lang="es">
<head>
  <title>La pseudoclase de los elementos vacíos</title>
  <meta charset="UTF-8" />
  <style>
    li:empty {
      background-color: gold;
    }
  </style>
</head>
<body>
<ul>
  <li>Manzana</li>
  <li>Plátano</li>
  <li></li>
  <li>Cereza</li>
  <li></li>
  <li>Kiwi</li>
  <li>Mango</li>
</ul>
</body>
</html>

```

Esto es lo que se muestra:

- Manzana
- Plátano
- 
- Cereza
- 
- Kiwi
- Mango

## La pseudoclase de negación

La pseudoclase de negación `:not` permite apuntar a todos los elementos HTML que no son el especificado.

Veamos el código de este ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las pseudoclases</title>
  <meta charset="UTF-8" />
  <style>
    p:not(u) {
      background-color: gold;
    }
  </style>
</head>
<body>
<p>Donec sed odio dui. Aenean <strong>lacinia
bibendum</strong> nulla sed <u>consectetur</u>. Donec id
elit non mi porta gravida at eget metus. Nullam id dolor
id <em>nibh ultricies</em> vehicula ut id elit. Nullam id
dolor id <strong>nibh</strong> ultricies vehicula ut id
elit. Cum sociis natoque <u>penatibus</u> et magnis dis
parturient montes, <em>nascetur</em> ridiculus mus.</p>
</body>
</html>
```

En el párrafo `<p>`, hay tres tipos de formato: negrita (`<strong>`), cursiva (`<em>`) y subrayado (`<u>`). El selector va a apuntar, en el párrafo `<p>`, a todos los elementos hijos que no están subrayados `<u>`: `p :not(u)`. Todos estos elementos aparecerán con un fondo dorado.

Todos los elementos hijos tienen un fondo dorado, excepto los subrayados.

## Nuevas propiedades CSS3

### Fuentes tipográficas

Vamos a ver algunas propiedades más, relativas a las fuentes tipográficas.

Para mostrar una fuente tipográfica en pantalla, el navegador empieza por identificar el origen de la fuente que se le pide. Este origen puede estar definido en el equipo del internauta, incrustado en la página o hallarse en un servidor en el Web.

Es perfectamente posible definir un conjunto de diversas fuentes tipográficas, e incluso resulta aconsejable. Si la fuente solicitada está disponible en el sistema operativo en el que se halla el navegador, se utiliza. Si no la encuentra allí, el navegador prueba con la segunda fuente de la lista, y así hasta el final. Y si el navegador no encuentra la fuente solicitada en el equipo del internauta, usa la fuente predeterminada que esté definida en las preferencias.

Una vez que el navegador ha identificado la fuente que debe usar, intenta aplicarle las demás propiedades: estilos, caja, grosor y tamaño.

### Las fuentes incrustadas

#### Los formatos de fuente

La regla `@font-face` de las CSS3 permite «incrustar» fuentes tipográficas en las páginas web. De este modo, los diseñadores no quedan limitados al uso de fuentes genéricas. No obstante, tenga siempre en cuenta estos tres grandes principios:

- La mayoría de las fuentes «profesionales» están sujetas a derechos de uso y difusión.
- Cuando «incrusta» una fuente tipográfica, se incorpora al archivo todo el juego de caracteres, lo que puede hacer que sus páginas pesen considerablemente más.
- Por lo general, el antialiasing (un método que evita que los caracteres se vean pixelados) no se aplica a las páginas web.

Ahora hemos de enfrentarnos al problema de la compatibilidad de los formatos de fuentes tipográficas con las distintas versiones de los navegadores.

He aquí los formatos de fuentes reconocidos por los navegadores:

- **TrueType:** extensión .ttf.

- **OpenType**: extensión .otf.
- **Web Open Font**: extensión .woff.
- **SVG Font**: extensión .svg y .svgz.
- **Embed OpenType**: extensión .eot. Pero cuidado: es un formato propietario de Microsoft y, por lo tanto, solo compatible con Internet Explorer.

Estas son las compatibilidades con los formatos **.ttf** y **.otf** (<http://caniuse.com/#feat=ttf>) según el sitio **Can I use**.

Y estas son las compatibilidades con el formato **.woff** (<http://caniuse.com/#feat=woff>):

Esta es la sintaxis de la regla @font-face:

```
@font-face {
    font-family: "Skia";
    src: url('Skia Regular.ttf');
}
```

Una vez se ha hecho la declaración de la regla @font-face, puede indicar el nombre que quiera para la fuente que se ha de incrustar mediante la propiedad font-family. A continuación, con la propiedad src, especificará la ruta de acceso al archivo de la fuente.

Para aplicarla, basta con indicar al selector deseado el nombre de la fuente mediante la propiedad font-family:

```
h1, h2 {
    font-family: Skia;
}
```

La propiedad font-family es arbitrario; no tiene por qué corresponder estrictamente al nombre del archivo de la fuente. Se trata de una etiqueta que usted da a esta fuente. Lo que es importante es el nombre del archivo de origen de la fuente en src.

He aquí una sintaxis correcta que funciona sin ningún problema:

```
@font-face {
```

```

    font-family: "Fuente titulo";
    src: url('Skia.ttf');
}
h1, h2 {
    font-family: 'Fuente titulo', Arial, Helvetica, sans-
    serif;
}

```

Podemos suponer que algunos internautas dispondrán de esta fuente de forma local, en su ordenador. Vamos a pedir que se utilice la fuente local si dicha fuente se halla en el equipo, especificando local en src.

```

@font-face {
    font-family: "Skia";
    src: local("Skia"), url('Skia.ttf');
}

```

El nombre de la fuente indicado en local debe ser el nombre del archivo en el sistema.

También puede indicar varios formatos de fuentes (si existen) para que la compatibilidad sea más elevada:

```

@font-face {
    font-family: "Skia";
    src: url('Skia.ttf')format("truetype"),
        url('Skia.woff')format("woff");
}

```

Por supuesto, puede indicar otras fuentes para tener mayor compatibilidad con navegadores antiguos:

```

h1, h2 {
    font-family: Skia, Arial, Helvetica, sans-serif;
}

```

## Las fuentes en línea

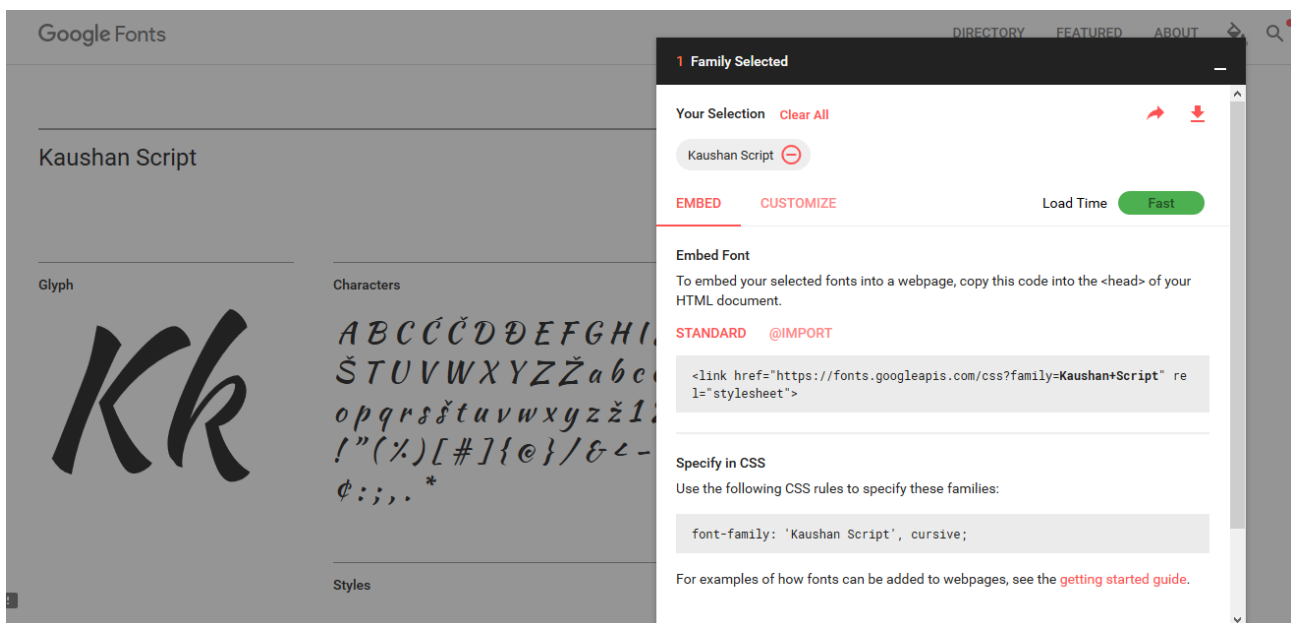
### Las Google Fonts

Existen numerosos servicios en línea que ofrecen la posibilidad de utilizar fuentes tipográficas para el Web. El servicio **Google Fonts** es uno de los más conocidos: <http://www.google.com/fonts/>.

Puede utilizar los filtros que aparecen a la derecha de la pantalla para elegir una fuente.

Seleccionamos una fuente y nos aparecerá las especificaciones de esta, en una nueva pantalla. Que nos indica :

- la sintaxis que se ha de utilizar para enlazar esta fuente tipográfica a las páginas web.
- la sintaxis que se ha de usar en las reglas CSS
- El impacto de carga de la página.



Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las fuentes de Google</title>
```



```

    <meta charset="UTF-8" />
    <link href='http://fonts.googleapis.com/css?
family=Pacifico'
rel='stylesheet' type='text/css'>
    <style>
        h1, h2 {
            font-family: 'Pacifico', cursive;
        }
    </style>
</head>
<body>
    <h1>Título de mi página</h1>
    <h2>Subtítulo</h2>
    <p>Donec ullamcorper nulla...</p>
</body>
</html>

```

### ***Probarlo.***

#### ***Ajustar el tamaño de los caracteres***

En los programas de tratamiento de texto, de autoedición y para el Web, el tamaño de las fuentes se basa en la altura de los ascendentes (parte vertical de las letras l, d, b...) y los descendentes (parte vertical de las letras p, q...). Visualmente, algunas fuentes presentan alturas distintas para las minúsculas. La fuente Verdana (creada especialmente para mejorar la legibilidad en el Web) tiene una minúscula más alta que una Arial o una Helvética, o que una Futura o una Bodoni,

La propiedad CSS3 **font-size-adjust** permite ajustar el tamaño de los caracteres basándose en la altura de las minúsculas, y no en la altura tradicional de los caracteres (ascendentes y descendentes). El valor que se atribuye a esta propiedad es un múltiplo del tamaño usual.

Ejemplo sin ajuste:

```
<!DOCTYPE html>
```

```

<html lang="es">
<head>
  <title>Ajustar tamaños</title>
  <meta charset="UTF-8" />
  <style>
    .helvetica {
      font-family: Helvetica;
    }
    .arial {
      font-family: Arial;
    }
  </style>
</head>
<body>
  <h1 class="helvetica">Título en Helvética</h1>
  <h1 class="arial">Título en Arial</h1>
</body>
</html>

```

Y esto es lo que se obtiene con Firefox, el único navegador que reconoce esta propiedad:

**Título en Helvética**

**Título en Arial**

Ahora, agreguemos la propiedad `font-size-adjust: .56;` al selector `.arial`.

```

.arial {
  font-family: Arial;
  font-size-adjust: .56;
}

```

}

Obtenemos lo siguiente:

**Título en Helvética**

**Título en Arial**

Se nota claramente la atenuación de la diferencia entre ambas fuentes. Le corresponde a usted determinar con precisión el valor que se debe emplear.

### ***Otras propiedades tipográficas.***

La propiedad **font-kerning** permitirá administrar el interletraje, es decir, el espacio entre los caracteres, los glifos.

La propiedad **font-variant-ligatures** permitirá administrar las ligaduras de los caracteres. Las ligaduras son los elementos de fusión entre dos caracteres, como la ese con la te, la efe con la i o con la ele...

La propiedad **font-variant-position** permitirá administrar mejor los caracteres en subíndice y superíndice, con criterios más tipográficos que los que aplica la propiedad vertical-align de la que hemos hablado con anterioridad.

La propiedad **font-variant-caps** permitirá gestionar mejor las versalitas, con criterios más tipográficos que los que aplica la propiedad font-variant de la que hemos hablado con anterioridad.

La propiedad **font-variant-numeric** permitirá gestionar mejor las variantes numéricas, como las fracciones, por ejemplo.

La propiedad **font-variant-alternates** permitirá gestionar los caracteres alternativos, como las letras de tipo antiguo o aquellas con un diseño específico para el final de las palabras.

La propiedad **font-variant-east-asian** permitirá gestionar los caracteres asiáticos.

### **El contenido generado**

#### ***El contenido textual antes y después***

El «contenido generado» permite a los diseñadores generar automáticamente un contenido antes o después de un elemento HTML especificado. Para ello, usamos los

dos pseudoelementos CSS3 `::before` y `::after`. Veamos los detalles de la propiedad `CSS content`.

Estos son los valores que se pueden usar:

- `none`: sin contenido generado.
- `normal`: sin contenido generado para los pseudoelementos.
- `string`: cadena de caracteres entre comillas.
- `uri`: enlace al recurso de imagen que se va a mostrar.
- `counter`: permite mostrar los valores de incremento.
- `attr`: identificador para los contadores.
- `open-quote`, `close-quote`, `no-open-quote`, `no-close-quote`: gestión de las comillas.

He aquí un ejemplo sencillo de uso de contenido generado en una lista. En él suprimimos las viñetas estándares de lista `<ul>`, para añadir delante una doble flecha y después una raya «eme».

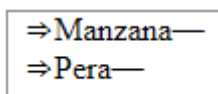
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Los pseudoelementos de contenido</title>
  <meta charset="UTF-8" />
  <style>
    li {
      list-style: none;
    }
    li::before {
      content: "\21d2";
    }
    li::after {
      content: "\2014";
    }
  </style>
</head>
<body>
  <ul>
    <li>Primer elemento de la lista</li>
    <li>Segundo elemento de la lista</li>
    <li>Tercer elemento de la lista</li>
  </ul>
</body>
</html>
```

```

        }
    </style>
</head>
<body>
<ul>
    <li>Manzana</li>
    <li>Pera</li>
</ul>
</body>
</html>

```

Esto es lo que obtenemos:



## Los contadores

Las CSS 2.1 ofrecen la posibilidad de insertar un contador con el fin de poder numerar automáticamente elementos HTML de una página. Para ello, debemos utilizar dos propiedades: counter-increment y counter-reset y la función counter().

La propiedad counter-increment permite establecer el valor del incremento de la numeración automática.

La propiedad counter-reset permite definir una nueva numeración (para contadores anidados).

La función counter() permite definir el contador.

Veamos un ejemplo con una numeración anidada.

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>Los contadores</title>
    <meta charset="UTF-8" />
    <style>

```

```

        body {
            counter-reset: parte;
        }
        h1:before {
            content: "Parte " counter(parte) " - ";
            counter-increment: parte;
        }
        h1 {
            counter-reset: seccion;
        }
        h2:before {
            content: counter(parte) "."
counter(seccion) " ";
            counter-increment: seccion;
        }
    </style>
</head>
<body>
    <h1>Introducción</h1>
    <h2>2014</h2>
    <p>Vestibulum id ligula porta...</p>
    <h2>2015</h2>
    <p>Maecenas sed diam eget risus...</p>
    <h1>Resultado</h1>
    <h2>Europa</h2>
    <p>Praesent commodo cursus magna...</p>
    <h2>Asia</h2>
    <p>Morbi leo risus, porta ac consectetur...</p>

```

```
<h1>Conclusión</h1>
  <p>Maecenas sed diam eget risus...</p>
</body>
</html>
```

El elemento <h1> determina el primer nivel de la numeración.

El elemento <h2> determina el segundo nivel de la numeración.

En el elemento <body>, la propiedad counter-reset define el nivel de partida del contador con la numeración denominada parte.

Antes del elemento <h1>, definimos un contenido generado que incluye:

el texto fijo Parte, a continuación, la numeración automática denominada parte, y el texto fijo que comporta un espacio, un guion y un espacio.

Siempre antes del elemento <h1>, se define por defecto en 1 el incremento del contador denominado parte, si no se indica ningún valor.

En el elemento <h1>, definimos el reinicio de una nueva numeración denominada seccion.

Antes del elemento <h2>, definimos un contenido generado que incluye:

**la numeración automática denominada parte, a continuación, un texto fijo con el carácter punto (.), luego, la numeración automática denominada seccion,**

Siempre antes del elemento <h2>, se define por defecto en 1 el incremento del contador denominado seccion, si no se indica ningún valor.

### **El desborde de texto**

Esta propiedad permite especificar el comportamiento de un texto en la línea cuando excede del ancho definido para su contenedor. Se utiliza con la propiedad overflow sin el valor visible (o sea, con los valores hidden o auto) y con la propiedad white-space: nowrap.

Estos son los valores que se pueden usar:

- clip: el texto se corta cuando llega al límite del bloque.
- ellipsis: el texto se corta y muestra puntos suspensivos (...).

- string: el texto se corta y muestra un carácter determinado por el diseñador de la página.

Veamos un ejemplo sencillo :

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Desborde de texto</title>
  <meta charset="UTF-8" />
  <style>
    p {
      white-space: nowrap;
      overflow: hidden;
      width: 500px;
      background-color: lightyellow;
      border: 1px solid #333;
    }
    .cortar {
      text-overflow: clip;
    }
    .suspender {
      text-overflow: ellipsis;
    }
  </style>
</head>
<body>
  <p class="cortar">Aenean lacinia bibendum nulla sed
consectetur. Integer posuere erat a ante venenatis dapibus
posuere velit aliquet.</p>
```



```
<p class="suspender">Donec ullamcorper nulla no metus  
auctor fringilla. Duis mollis, est no commodo luctus,  
nisi erat porttitor ligula.</p>  
</body>  
</html>
```

## Los colores

Otra posibilidad es indicar un color especificando los valores de los tres componentes HSL: matiz (hue), saturación (saturation) y brillo (lightness). El componente del matiz va de 0 (rojo), a 60 (amarillo), a 120 (verde), a 180 (turquesa), a 240 (azul), a 300 (magenta). La unidad es un grado en la rueda cromática. El componente de saturación va de 0 (color desaturado) a 100 (color saturado). El componente del brillo va de 0 (negro) a 100 (blanco). Las unidades para la saturación y el brillo son porcentajes.

Ejemplos:

- `hsl(100,75,90)` es un verde claro.
- `hsl(328,81,51)` es un violeta oscuro.

Como en el caso anterior, es posible añadir un grado de transparencia con la propiedad `hsla(328,81,51,0.3)`.

## Los formularios

### ***Dar formato***

Para dar formato a los formularios, puede utilizar las propiedades CSS aplicables al texto, así como las propiedades de los contenedores (ancho, contorno, color de fondo...). No existe ninguna propiedad ni ningún módulo específicamente dedicado a los formularios.

Vamos a utilizar, sencillamente, selectores, propiedades y atributos que se adaptan bien a los formularios.

### ***Redimensionar un campo***

Se utiliza la propiedad `resize`:

Los valores que se pueden utilizar:

- `none`: sin posibilidad de redimensionamiento.

- both: redimensionable horizontal y verticalmente.
- horizontal: redimensionable horizontalmente.
- vertical: redimensionable verticalmente.

He aquí un ejemplo sencillo.

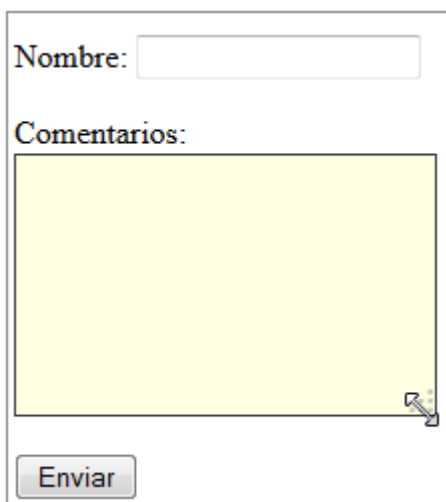
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Redimensionar un campo</title>
  <meta charset="UTF-8" />
  <style>
    #comentario {
      width: 200px;
      height: 90px;
      border: 1px solid #333;
      background-color: lightyellow;
      resize: both;
    }
  </style>
</head>
<body>
<form id="test" method="#" action="#">
  <p>
    <label for="nombre">Nombre: </label>
    <input type="text" id="nombre" />
  </p>
  <p>
    <label for="comentario">Comentarios: </label>
    <br />
  </p>
</form>
```

```

        <textarea name="comentario"
id="comentario"></textarea>
    </p>
    <p>
        <input type="submit" name="envio" id="envio"
value="Enviar" />
    </p>
</form>
</body>
</html>

```

Esto es lo que obtenemos: el usuario puede cambiar las dimensiones del campo **Comentarios**.



The screenshot shows a web form with a light gray border. At the top, there is a label 'Nombre:' followed by a small white text input field. Below this is a label 'Comentarios:' followed by a large yellow text area. The text area has a thin black border and a small handle in the bottom right corner with a blue arrow, indicating it is resizable. At the bottom of the form is a gray button with the text 'Enviar'.

### ***Pseudoclases para formularios***

Las CSS3 proponen una serie de pseudoclases muy útiles para los objetos de formulario; permiten resaltar los objetos activos, desactivados y marcados en un formulario. De este modo, el usuario puede saber qué ha hecho él, qué está activo y qué inactivo.

Estas nuevas pseudoclases son: `:enabled`, `:disabled` y `:checked`.

En este ejemplo, resaltamos el uso de diferentes campos de formulario.

El primer selector se dirige a la etiqueta (label) de la casilla de verificación (#acuerdo) para saber si está marcada (checked). Lo que queremos resaltar es,

efectivamente, la etiqueta de la casilla de verificación, y no la casilla de verificación propiamente dicha; de ahí el uso de un selector adyacente.

El segundo selector se dirige al botón de opción marcado, también en este caso para detectar que lo está.

Los otros dos estilos simplemente sirven para atribuir un fondo de color según el estado del campo.

Este es el código:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Pseudoclases para los formularios</title>
  <meta charset="UTF-8" />
  <style>
    #acuerdo:checked+label {
      background-color: gold;
      font-weight: bold;
    }
    #hombre:checked+label, #mujer:checked+label {
      font-style: italic;
    }
    :enabled {
      background-color: lightgreen;
    }
    :disabled {
      background-color: lightcoral;
    }
  </style>
</head>
```

```

<body>
<form id="form1" method="#" action="#">
<p>
    <label for="nombre">Su nombre: </label>
    <input type="text" id="nombre" />
</p>
<p>
    <label for="edad">Su edad: </label>
    <input type="text" id="edad" disabled="disabled"/>
</p>
<p>
    <input type="radio" name="sexo" id="hombre"
value="hombre" />
    <label for="hombre">Hombre</label><br/>
    <input type="radio" name="sexo" id="mujer" value="mujer"
/>
    <label for="hombre">Mujer</label>
</p>
<p>
    <input type="checkbox" id="acuerdo" />
    <label for="acuerdo">Estoy de acuerdo con el
reglamento</label>
</p>
</form>
</body>
</html>

```

Esto es lo que aparece cuando se carga la página:

Su nombre:

Su edad:

☐ Hombre

☐ Mujer

☐ Estoy de acuerdo con el reglamento

Esto es lo que se ve cuando el usuario marca el botón de opción **Hombre**.

Su nombre:

Su edad:

☒ *Hombre*

☐ Mujer

☐ Estoy de acuerdo con el reglamento

Y esto es lo que se obtiene cuando el usuario marca la casilla de verificación.

Su nombre:

Su edad:

☒ *Hombre*

☐ Mujer

☒ **Estoy de acuerdo con el reglamento**

Aún existe otro estado para los botones de opción y las casillas de verificación: se trata del estado «indeterminado»: :indeterminate, que indica que el objeto no está ni marcado ni desmarcado.

### ***Los campos requeridos y los opcionales***

Si lo desea, puede resaltar los campos que sean obligatorios u opcionales con las pseudoclases :required y :optional.

Los campos que son obligatorios, es decir, aquellos en los que se exige introducir o marcar algo, deben disponer del atributo booleano required. Los que no sean obligatorios serán opcionales de forma predeterminada, sin que resulte necesario especificarlo.

He aquí un ejemplo:

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>Los campos requeridos y los
opcionales</title>
    <meta charset="UTF-8" />
    <style>
        input:required {
            background-color: lightcoral;
        }
        input:optional {
            background-color: lightgoldenrodyellow;
        }
        input#envio {
            background: none;
        }
    </style>
</head>
<body>
<form id="form1" method="#" action="#">
    <p>
        <label for="nombre">Su nombre: </label>
        <input type="text" name="nombre" id="nombre"
required />
    </p>
    <p>
        <label for="edad">Su edad: </label>

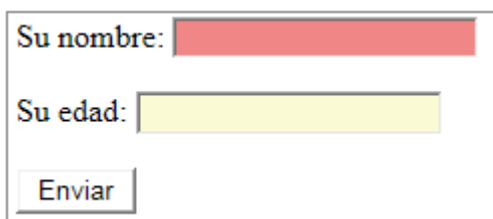
```

```

        <input type="text" name="edad" id="edad" />
    </p>
    <p>
        <input type="submit" id="envio"
value="Enviar" />
    </p>
</form>
</body>
</html>

```

Esto es lo que se obtiene:



The screenshot shows a web form with two input fields and a submit button. The first field is labeled 'Su nombre:' and has a red background. The second field is labeled 'Su edad:' and has a yellow background. Below these fields is a button labeled 'Enviar'.

### ***Dar formato al «foco»***

La pseudoclase :focus sirve para resaltar el campo que se está usando mediante la adición de un contorno suplementario que proporciona la propiedad outline. Las CSS3 añaden una nueva propiedad: outline-offset, que define la distancia entre el límite de la caja y el contorno de resalte.

He aquí un ejemplo muy sencillo:

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>El foco</title>
    <meta charset="UTF-8" />
    <style>
        input:focus {
            outline: solid 3px green;
            outline-offset: 2px;
        }
        input#envio {
            background: none;
        }
    </style>

```

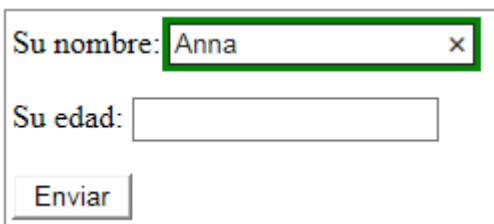


```

        </style>
</head>
<body>
<form id="form1" method="#" action="#">
    <p>
        <label for="nombre">Su nombre: </label>
        <input type="text" name="nombre" id="nombre"
required />
    </p>
    <p>
        <label for="edad">Su edad: </label>
        <input type="text" name="edad" id="edad" />
    </p>
    <p>
        <input type="submit" id="envio"
value="Enviar" />
    </p>
</form>
</body>
</html>

```

He aquí lo que aparece cuando el usuario hace clic en un campo:



### ***Validar la información introducida***

Se trata de resaltar la validación de los campos en un formulario. En este ejemplo , los dos primeros campos son obligatorios debido al uso del atributo required. El tercer campo incorpora un patrón que deben seguir los datos que se introduzcan (pattern). En estos tres campos se mostrará un círculo rojo cuando la información introducida no sea válida, y un círculo verde cuando sí lo sea.

Veamos el código de este ejemplo:

```

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

```

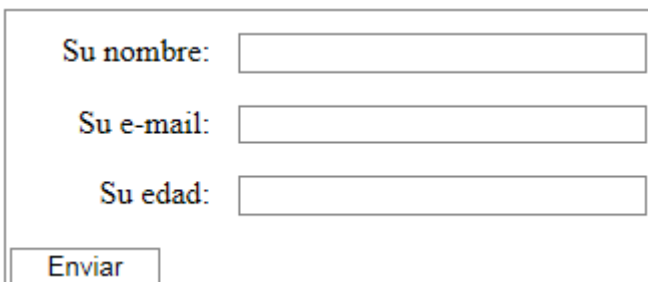
```

<title>Validación de un formulario</title>
<meta charset="UTF-8" />
<style>
    input {
        border: solid 1px gray;
        width: 200px;
    }
    label, input{
        display: inline-block;
    }
    label {
        width: 100px;
        text-align: right;
        margin-right: 10px;
    }
    input[type=submit]{
        width: 75px;
        background: none;
    }
    input:focus:invalid {
        background: url(rojo.png) no-repeat 98%
center;
    }
    input:focus:valid {
        background: url(verde.png) no-repeat 98%
center;
    }
</style>
</head>

```

```
<body>
<form id="inscription" method="#" action="#">
  <p>
    <label for="nombre">Su nombre: </label>
    <input type="text" id="nombre" required />
  </p>
  <p>
    <label for="email">Su e-mail: </label>
    <input type="email" id="email" required />
  </p>
  <p>
    <label for="edad">Su edad: </label>
    <input type="text" id="edad" pattern="\d{2}" />
  </p>
  <p>
    <input type="submit" id="envio" value="Enviar" />
  </p>
</form>
</body>
</html>
```

Esto es lo que aparece cuando se carga la página:



The screenshot shows a web form with three input fields and a submit button. The first field is labeled "Su nombre:" and is a text input. The second field is labeled "Su e-mail:" and is an email input. The third field is labeled "Su edad:" and is a text input with a pattern constraint. Below these fields is a submit button labeled "Enviar".

Esto es lo que se muestra cuando el usuario hace clic en el primer campo, **Su nombre:**

Su nombre:  ●

Su e-mail:

Su edad:

El campo no es válido, ya que no se ha introducido nada y es obligatorio rellenar este campo.

Veamos lo que aparece cuando el usuario escribe algo en el primer campo, **Su nombre:**

Su nombre:  ●

Su e-mail:

Su edad:

El campo tiene contenido, por lo que la validación es correcta.

Esto es lo que aparece cuando el usuario únicamente introduce una cifra en el campo **Su edad:**

Su nombre:

Su e-mail:

Su edad:  ●

El dato no es válido, ya que tiene únicamente una cifra.

Y esto es lo que aparece cuando el usuario introduce dos cifras en el campo **Su edad:**

Su nombre:

Su e-mail:

Su edad:  ●

El dato es válido, ya que tiene dos cifras.

## Los contornos dinámicos

El contorno dinámico sirve para resaltar un elemento de la página, principalmente en el momento en que el usuario realiza una acción. El ejemplo más común es su utilización en los campos de un formulario. En el apartado dedicado a los formularios del capítulo Tablas y formularios, hallará un ejemplo de este uso.

La propiedad outline es la sintaxis corta de estas propiedades individuales: **outline-color**, **outline-style** y **outline-width**.

Veamos un ejemplo muy sencillo de contorno dinámico en un párrafo en el que el usuario ha hecho clic.

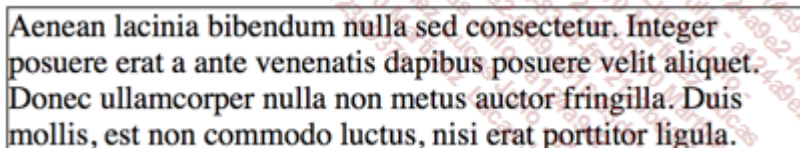
He aquí el código:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Párrafo resaltado</title>
  <meta charset="UTF-8" />
  <style>
    body {
      margin: 30px;
    }
    .cuadro {
      width: 400px;
      border: 1px solid #333;
    }
    .cuadro:active {
      outline: fuchsia solid 10px;
      outline-offset: 2px;
    }
  </style>
</head>
```

```
<body>
    <p class="cuadro">Aenean lacinia bibendum nulla sed
consectetur. Integer posuere erat a ante venenatis dapibus
posuere velit aliquet. Donec ullamcorper nulla non metus
auctor fringilla. Duis mollis, est non commodo luctus,
nisi erat porttitor ligula.</p>
</body>
</html>
```

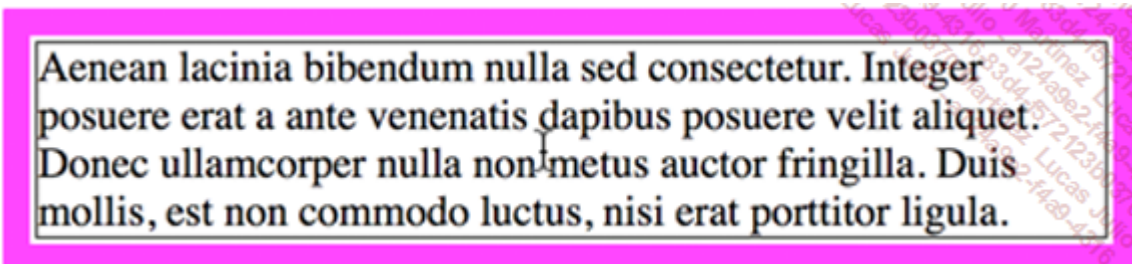
Cuando el usuario haga clic en el párrafo (evento :active), el párrafo mostrará el cuadro que se ha definido.

Este es el aspecto que presenta la página cuando se carga:



Aenean lacinia bibendum nulla sed consectetur. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Donec ullamcorper nulla non metus auctor fringilla. Duis mollis, est non commodo luctus, nisi erat porttitor ligula.

Y esto es lo que se ve cuando el usuario hace clic en el párrafo:



Aenean lacinia bibendum nulla sed consectetur. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Donec ullamcorper nulla non metus auctor fringilla. Duis mollis, est non commodo luctus, nisi erat porttitor ligula.

## Los cursores

La propiedad cursor permite cambiar la apariencia del cursor en un contexto determinado de la página.

**Propiedad:** cursor

He aquí los valores que pueden usarse:

Cursores generales:

- url: indica la ruta de acceso al archivo de imagen que reemplaza al cursor.
- auto: el navegador escoge la apariencia del cursor.

- default: cursor por defecto de la plataforma utilizada (con frecuencia es una flecha).
- none: sin cursor.

#### Cursores de vínculo y estado:

- content-menu: para mostrar el cursor por encima de un elemento.
- help: el cursor puede tomar el aspecto de un signo de interrogación o un globo para indicar que el elemento señalado dispone de ayuda.
- pointer: el cursor toma la apariencia de un dedo extendido.
- progress: el cursor toma la apariencia de una rueda o de una flecha con un reloj para indicar que se está llevando a cabo un proceso.
- wait: el cursor toma la apariencia de un reloj (que puede ser de arena) para indicar que hay que esperar.

#### Cursores para seleccionar:

- cell: para seleccionar celdas.
- crosshair: cursor en forma de cruz.
- text: indica que puede seleccionarse texto. El cursor toma la apariencia de una I.
- vertical-text: indica que puede seleccionarse un texto vertical. El cursor toma la apariencia de una I horizontal.

#### Cursores para hacer clic y arrastrar-soltar:

- alias: indica el alias de un elemento. Usualmente se trata de una flecha con una curva.
- copy: indica que se ha copiado un elemento. Usualmente se trata de una flecha con un signo +.
- move: indica que un objeto puede moverse.
- no-drop: indica que un elemento no puede soltarse. Usualmente se trata de una flecha o una mano con un signo de prohibición.

- not-allowed: indica una acción no autorizada. Usualmente se trata de un signo de prohibición.
- grab: indica que algo puede «agarrarse». Se representa con un dorso de mano abierta.
- grabbing: indica que se tiene algo «agarrado» y que se está moviendo. Se representa con un puño.

#### Cursores de cambio de tamaño y desplazamiento:

- e-resize, n-resize, ne-resize, nw-resize, s-resize, se-resize, sw-resize, w-resize: indica que el borde de un objeto puede desplazarse. Los valores son coordenadas «geográficas»: ne para nordeste, sw para sudoeste...
- ew-resize, ns-resize, nesw-resize, nwse-resize: cursores bidireccionales.
- col-resize: indica que puede cambiarse el tamaño de una columna.
- row-resize: indica que puede cambiarse el tamaño de una fila.
- all-resize: indica que un objeto puede redimensionarse en todas las direcciones.

#### Cursores de zoom:

- zoom-in, zoom-out: indica la posibilidad de aplicar un zoom. Usualmente se trata de una lupa con los signos + y -.

He aquí un ejemplo sencillo, de uso de dos cursores: ayuda (help) y espera (wait):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Cursores</title>
  <meta charset="UTF-8" />
  <style>
    .ayuda {
      cursor: help;
    }
    .espera {
```

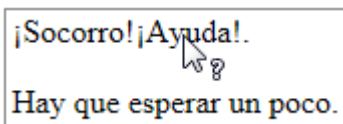


```

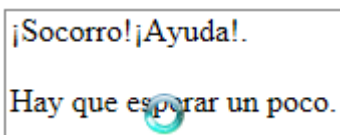
        cursor: wait;
    }
</style>
</head>
<body>
    <p>¡Socorro!<span class="ayuda"> ¡Ayuda!</span>.</p>
    <p>Hay que <span class="espera">esperar</span> un
poco.</p>
</body>
</html>

```

Este es el aspecto del cursor de ayuda:



Y este, el del cursor de espera:



## Bordes

Las cajas pueden enriquecerse con bordes que se colocan en los límites de la caja. Es posible definir un borde idéntico para todos los lados de la caja o bien bordes individuales para cada lado. Para ello, utilice estas propiedades:

- **border-style:** para gestionar el estilo del borde. Estos son los valores posibles:
  - none: sin borde.
  - dotted: trazo de puntos.
  - dashed: trazo de guiones.
  - solid: borde de trazo ininterrumpido.
  - double: doble trazo ininterrumpido.

- groove: trazo de cruces.
- ridge: trazo en relieve.
- inset: da a la caja un aspecto de bajorrelieve.
- outset: da a la caja un aspecto de relieve.
- **border-width:** para gestionar el grosor del borde. Estos son los valores que pueden usarse:
  - longitud: permite indicar el valor del grosor del borde y su unidad. No se admiten valores negativos.
  - thin, medium y thick son valores preestablecidos para grosores finos, medios y anchos, respectivamente. Cada navegador atribuye su propio grosor a estos valores preestablecidos.
- **border-color:** para gestionar el color del borde.
- **Border (propiedad generiaca): Valor:** border-width | border-style | border-color

He aquí un ejemplo sencillo de cajas con bordes: la primera caja posee un borde idéntico para todos los lados, mientras que la segunda incluye bordes diferentes.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Los bordes de las cajas</title>
  <meta charset="UTF-8" />
  <style>
    .caja-1 {
      width: 300px;
      padding: 20px;
      border: 3px double #333;
    }
    .caja-2 {
```

```

        width: 300px;
        padding: 20px;
        border-top: 5px dotted red;
        border-right: 3px solid green;
        border-bottom: 10px dashed blue;
        border-left: medium outset purple;
    }
</style>
</head>
<body>
<div class="caja-1">Praesent commodo...</div>
<p>&nbsp;</p>
<div class="caja-2">Aenean lacinia bibendum...</div>
</div>
</body>
</html>

```

## pruébalo

### ***Los bordes de fantasía***

Las CSS3 permiten crear bordes de fantasía en los que se incluyen imágenes; la imagen se repetirá en el borde siguiendo el motivo definido por el diseñador.

En este ejemplo, tenemos una imagen de 60 píxeles de lado. Rematamos cada ángulo con un círculo, mientras que los cuatro lados están compuestos por triángulos orientados convenientemente. Cada «motivo» mide, por tanto, 20 píxeles de lado.

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>Los bordes de fantasía</title>
    <meta charset="UTF-8" />
    <style>

```

```

        .borde {
            width: 300px;
            padding: 10px;
            border-width: 20px;
            border-image: url(motivo.png) 20 round;
        }
    </style>
</head>
<body>
<div class="borde">Aenean lacinia bibendum...</div>
</div>
</body>
</html>

```

Observe la aparición de la propiedad estándar `border-width`, que en este ejemplo es igual a las dimensiones de cada motivo que se ha de situar en el borde: 20 px.

Los argumentos de la propiedad (aquí con su sintaxis corta) son:

- `url(motivo.png)`: ruta de acceso a la imagen.
- `20`: porción en píxeles de la imagen que se ha de utilizar para cada motivo. Disponemos de 20 píxeles de la imagen para cada motivo. Si el valor es el mismo para cada lado, entonces basta con un solo valor, que será equivalente a: `20 20 20 20`. Si los valores son diferentes para cada lado, entonces indíquelos en este orden: superior, derecho, inferior e izquierdo.
- `round`: permite repetir la imagen y cambiar sus dimensiones para que ocupe la totalidad del borde sin que quede truncada (los otros dos valores son `stretch` para extender el motivo a toda la dimensión del lado y `repeat` para repetir el motivo, que podrá truncarse).

Aquí hemos usado la sintaxis corta: `border-image`. Pero es posible individualizar cada propiedad:

- `border-image-source`: URL de la fuente de la imagen.

- border-image-slice: los valores de recorte de la imagen para obtener los motivos.
- border-image-width: grosor del borde.
- border-image-outset: desplazamiento de la imagen en relación con el borde.
- border-image-repeat: tipo de repetición de la imagen.

### ***Las esquinas redondeadas***

Puede obtener esquinas redondeadas idénticas con la propiedad border-radius, o bien esquinas diferentes con las propiedades border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius.

Veamos un ejemplo , con una primera caja que usa esquinas redondeadas idénticas y una segunda caja que usa esquinas redondeadas diferentes.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las esquinas redondeadas</title>
  <meta charset="UTF-8" />
  <style>
    .esquinas-iguales {
      width: 300px;
      padding: 10px;
      border: 1px solid #333;
      border-radius: 10px;
    }
    .esquinas-diferentes {
      width: 300px;
      padding: 10px;
      border: 1px solid #333;
      border-radius: 10px 20px 20px 40px;
```

```

    }
</style>
</head>
<body>
<div class="esquinas-iguales">Praesent commodo...</div>
<p>&nbsp;</p>
<div class="esquinas-diferentes">Aenean lacinia
bibendum...</div>
</div>
</body>
</html>

```

Esto es lo que se obtiene:

Praesent commodo cursus magna, vel scelerisque nisl consectetur et. Etiam porta sem malesuada magna mollis euismod. Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Maecenas sed diam eget risus varius blandit sit amet non magna.

Aenean lacinia bibendum nulla sed consectetur. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Vestibulum id ligula porta felis euismod semper. Nulla vitae elit libero, a pharetra augue. Cras justo odio, dapibus ac facilisis in, egestas eget quam.

## ***Esquinas con elipses***

También es posible aplicar elipses en vez de círculos a las esquinas de las cajas. Basta con indicar valores separados por una barra: /. El primer valor corresponde al radio horizontal, y el segundo, al radio vertical.

Veamos un ejemplo sencillo (**08\_10.html**):

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>Las esquinas redondeadas</title>

```

```

<meta charset="UTF-8" />
<style>
    .ellipse {
        width: 300px;
        padding: 10px;
        border: 1px solid #333;
        border-radius: 40px/20px;
    }
</style>
</head>
<body>
<div class="ellipse">Aenean lacinia bibendum nulla sed
consectetur. Integer posuere erat a ante venenatis dapibus
posuere velit aliquet. Vestibulum id ligula porta felis
eismod semper. Nulla vitae elit libero, a pharetra augue.
Cras justo odio, dapibus ac facilisis in, egestas eget
quam.</div>
</div>
</body>
</html>

```

Esto es lo que se obtiene:

Aenean lacinia bibendum nulla sed  
consectetur. Integer posuere erat a ante  
venenatis dapibus posuere velit aliquet.  
Vestibulum id ligula porta felis eismod  
semper. Nulla vitae elit libero, a pharetra  
augue. Cras justo odio, dapibus ac facilisis in,  
egestas eget quam.

### ***Especificar el cálculo de la anchura***

Las CSS3 permiten elegir cómo se calcula la anchura de las cajas, con la propiedad **box-sizing**. (no se hereda)

Puede indicar que los valores de border y de padding se incluyan en el ancho width.

La propiedad acepta tres valores:

- **content-box:** la anchura de la caja se calcula sumando la anchura del contenido, el grosor del borde y la anchura del relleno, como en CSS 2.1. Es el valor por defecto.
- **border-box:** la anchura de la caja es igual a la anchura del contenido. Las anchuras del borde y del relleno están incluidas en la anchura del contenido.

He aquí un ejemplo sencillo con dos cajas que tienen anchuras de contenido (width), rellenos (padding) y bordes idénticos (border). La caja `<div class="mi-caja-3">` incluye también la propiedad box-sizing: border-box.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las anchuras de las cajas</title>
  <meta charset="UTF-8" />
  <style>
    .mi-caja-1 {
      width: 300px;
      padding: 10px;
      border: 10px solid #333;
    }
    .mi-caja-2 {
      width: 300px;
      padding: 10px;
      border: 10px solid #333;
    }
    .mi-caja-3 {
      width: 300px;
```



```

padding: 10px;
border: 10px solid #333;
box-sizing: border-box;
    }
</style>
</head>
<body>
<div class="mi-caja-1">Praesent commodo...</div>
<p>&nbsp;</p>
<div class="mi-caja-2">Aenean lacinia bibendum...</div>
</div>
<p>&nbsp;</p>
<div class="mi-caja-3">Donec sed odio dui...</div>
</body>
</html>

```

## Probarlo.

### El desborde de contenido

Las CSS 2.1 permiten utilizar la propiedad `overflow` para gestionar el contenido cuando este es mayor que el contenedor, algo que se conoce como desborde de contenido.

Las CSS3 proponen dos nuevas propiedades para gestionar el desborde:

- **overflow-x**: para gestionar el desborde horizontal.
- **overflow-y**: para gestionar el desborde vertical.

Los valores de base son los siguientes:

- **visible**: el contenido siempre es visible.
- **hidden**: el contenido está oculto, aunque se puede hacer que se muestre con métodos programáticos.
- **clip**: el contenido está siempre oculto.

- **scroll**: el contenido puede verse usando una barra de desplazamiento que siempre está visible.
- **auto**: el contenido puede verse usando una barra de desplazamiento que aparece si es necesaria.

En el ejemplo tenemos dos cajas; la primera (**#caja-1**) utiliza la propiedad **overflow: auto**; la segunda (**#caja-2**), usa la propiedad **overflow-x: visible** y la propiedad **white-space: nowrap** indispensable. Para que no haya ni blancos ni saltos de páginas que no se hayan especificado.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Desbordamiento de contenido</title>
  <meta charset="UTF-8" />
  <style>
    #caja-1 {
      width: 350px;
      height: 200px;
      border: 1px solid #333;
      padding: 0;
      overflow: auto;
      margin-bottom: 30px;
    }
    #caja-2 {
      width: 350px;
      height: 200px;
      border: 1px solid #333;
      padding: 0;
      overflow-x: scroll;
      white-space: nowrap;
```

```

        }
        .sin-margen {
            margin: 0;
        }
    </style>
</head>
<body>
<div id="caja-1" class="sin-margen">Aenean lacinia
bibendum nulla sed consectetur...</div>
<div id="caja-2" class="sin-margen">Aenean lacinia
bibendum nulla sed consectetur....</div>
</body>
</html>

```

Este es el aspecto que presenta la primera caja, con la barra de desplazamiento vertical en acción.

Y la segunda caja, con la barra de desplazamiento horizontal en acción.

## Las sombras paralelas

La propiedad **box-shadow** permite aplicar sombras paralelas a las cajas.

Los parámetros que pueden usarse con shadow son:

- La distancia horizontal de la sombra paralela.
- La distancia vertical de la sombra paralela.
- El tamaño de desvanecimiento de la sombra paralela. Este atributo es optativo; si se omite, su valor es 0.
- El tamaño de extensión de la sombra paralela. Este permite colocar el «punto mediano» del degradado, punto a partir del cual la sombra se difumina. Un valor positivo extiende la sombra, mientras que un valor negativo la contrae. Este atributo es optativo; si se omite, su valor es 0.
- El color de la sombra paralela.

- La posición de la sombra paralela (inset: en el interior, outset en el exterior, que es el valor por defecto).

Puede aplicar perfectamente varias sombras paralelas a la misma caja.

En este ejemplo, tenemos cuatro muestras de sombras paralelas diferentes:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Las sombras paralelas</title>
    <meta charset="UTF-8" />
    <style>
        .sombra1 {
            width: 350px;
            border: 1px solid #333;
            padding: 10px;
            box-shadow: 10px 15px 8px 2px rgb(12,34,56);
        }
        .sombra2 {
            width: 350px;
            border: 1px solid #333;
            padding: 10px;
            box-shadow: 10px 15px 0 0 grey;
        }
        .sombra3 {
            width: 350px;
            border: 1px solid #333;
            padding: 10px;
            box-shadow: 10px 10px 5px 5px grey;
        }
    </style>

```

```
        .sombra4 {
            width: 350px;
            border: 1px solid #333;
            padding: 10px;
            box-shadow: 10px 10px 5px -5px grey;
        }
    </style>
</head>
<body>
<div class="sombra1">Aenean lacinia bibendum...</div>
<p>&nbsp;</p>
<div class="sombra2">Cras justo odio...</div>
<p>&nbsp;</p>
<div class="sombra3">Aenean lacinia bibendum...</div>
<p>&nbsp;</p>
<div class="sombra4">Curabitur blandit...</div>
</body>
</html>
```

Esto es lo que se obtiene:

Aenean lacinia bibendum nulla sed consectetur.  
Integer posuere erat a ante venenatis dapibus posuere  
velit aliquet. Vestibulum id ligula porta felis euismod  
semper. Nulla vitae elit libero, a pharetra augue.

Cras justo odio, dapibus ac facilisis in, egestas eget  
quam. Praesent commodo cursus magna, vel  
scelerisque nisl consectetur et. Vestibulum id ligula  
porta. Lorem ipsum dolor sit amet, consectetur  
adipiscing elit.

Aenean lacinia bibendum nulla sed consectetur.  
Integer posuere erat a ante venenatis dapibus posuere  
velit aliquet. Vestibulum id ligula porta felis euismod  
semper. Nulla vitae elit libero, a pharetra augue.

Curabitur blandit tempus porttitor. Vivamus sagittis  
lacus vel augue laoreet rutrum faucibus dolor auctor.  
Cras justo odio, dapibus ac facilisis in, egestas eget  
quam. Nullam id dolor id nibh ultricies vehicula ut id  
elit. Maecenas faucibus mollis interdum. Duis mollis,  
est non commodo luctus.