

2 Tema

HTML5 Y CSS3

En este primer tema se pretenden conseguir los siguientes objetivos:

- Conocer qué es y para qué sirve HTML5 y CSS3.
 - Ver las diferencias entre HTML5 y sus antecesores HTML4 y XHTML.
 - Repasar las características de HTML4 que son similares o iguales a la nueva versión HTML5.
 - Repasar las características de CSS2.1 que son similares o iguales a la nueva versión CSS3.
 - Aplicar estilos CSS a un documento HTML para lograr el diseño deseado.
-

Sumario

Introducción.....	4
Normalización. Versiones de HTML.....	4
HTML5.....	5
Los navegadores.....	5
Sintaxis HTML.....	7
El primer documento HTML.....	8
El Doctype.....	9
El elemento html.....	10
El elemento head.....	10
El resto no cambia.....	11
Diferencias con XHTML.....	12
Etiquetas básicas.....	13
Texto.....	13
Cabeceras.....	13
Párrafos.....	14
Línea horizontal.....	14
Texto preformateado.....	14
Resaltado básico de caracteres.....	15
Marcado avanzado de caracteres.....	16
Listas.....	17
Listas desordenadas.....	17
Listas ordenadas.....	17
Enlaces.....	18
Imágenes.....	20
Tabla.....	21
Elementos de bloque y en línea.....	22
Formularios.....	24
Una página web dentro de otra.....	25
Definiendo la estructura del cuerpo.....	25
CSS aspectos básicos.....	29
Incorporar estilos al documento.....	29
Funcionamiento básico de las reglas CSS.....	31
Herencia.....	32
Selectores.....	37
Ejercicio.....	44
Prioridad.....	47
Formas de indicar valores de propiedades CSS.....	50
Indicación de color.....	51
Formato de Fuente y Texto.....	52
Propiedades de las fuentes.....	52
Propiedades del texto.....	56

Fondo.....	58
Listas.....	60
Links.....	62
Tablas.....	63
Modelo de cajas.....	64
Los márgenes de los elementos.....	65
Visualización de los elementos.....	66
Visualización en flujo normal.....	67
Posicionamiento relativo.....	68
Posicionamiento absoluto.....	70
Posicionamiento fijo.....	81
Posicionamiento Flotante.....	82
Ejemplos:.....	85
Letra capital.....	85
Posicionamiento flotante de imágenes.....	86
La propiedad clear.....	89
Tamaño de los elementos que contienen elementos flotantes.....	91
Diseño con cajas flotantes.....	93
Conclusión sobre diseñar con elementos flotantes.....	100
Diseño con tablas.....	101
La propiedad display.....	101
Diseño simple con una tabla.....	102
Ancho en pantalla de la tabla.....	104
Elementos anónimos.....	105
Otros elementos para las tablas.....	105
Párrafos en las celdas.....	105
Listas en tablas.....	106
Diseño de las filas.....	107
Establecer el ancho de las celdas.....	109
Anchos fijos.....	109
Anchos en porcentaje.....	109
Diseño con una tabla más estructurada.....	110
Otras propiedades de diseño.....	112
Propiedad table-layout.....	112
Propiedad border-collapse.....	113
Propiedad border-spacing.....	116
Propiedad empty-cells.....	116
Propiedad caption-side.....	117
Alineación vertical.....	120
Conclusión.....	121

Introducción

Todas las páginas web en la World Wide Web tienen una cosa en común: todos ellos se debe crear utilizando algún tipo de marcado de hipertexto Language (HTML). Es asombroso pensar que toda la World Wide Web, se basa en un simple lenguaje de marcado basado en texto que es fácil de aprender y utilizar.

HTML es un lenguaje de marcas, un **lenguaje estructurado** que le permite identificar las secciones comunes de una página web, tales como **encabezados**, **párrafos** y **listas** con etiquetas de marcas que definen cada sección.

Las páginas Web son simplemente documentos de texto que utilizan HTML para decirle al navegador cómo mostrar cada sección del documento.

Normalización. Versiones de HTML

El problema surgió en cuanto unos navegadores incorporaron elementos HTML que el resto no traducían, con lo que aparecieron diferentes dialectos HTML. Así una página se podía mostrar de forma totalmente diferente según el navegador.

La solución pasó por intentar **estandarizar** el lenguaje. Por ello el propio Tim Berners Lee fundó la World Wide Web Consortium (abreviado W3C) como organismo de estandarización del lenguaje HTML ante la industria.

En la actualidad las directrices de W3C son seguidas por la mayoría de navegadores **aunque no al 100**, lo que sigue generando problemas a los creadores de páginas web.

La situación, sin embargo se ha complicado en estos últimos años con la aparición de diferentes estándares, en concreto actualmente se consideran estándares a HTML 4.01, XHTML 1.0 y 1.1 y ya se considera de la misma forma a HTML 5. Las diferencias entre ellos son:

- **HTML 4.01.** Se trata de la versión estándar del HTML tradicional hecha en el año 1999 y que sigue teniendo mucha vigencia actualmente. Hay tres versiones: la transicional (que permite seguir usando algunas etiquetas que se consideran obsoletas), la estricta (que elimina numerosas etiquetas y atributos para forzar a crear un HTML con menos formato y más significado) y la frameset orientada a usar los ya muy poco utilizados marcos. La versión más popular es la transicional al ser más libre.

- **XHTML.** XHTML era planteado como el sustituto de HTML, la primera versión, la 1.0 sigue siendo la más usada, actualmente está la versión 2.0 . Hay también versión estricta, transicional y (muy poco utilizada) frameset; la más utilizada, otra vez es la transicional. El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Las páginas XHTML obligan a que la escritura de HTML siga las reglas del lenguaje XML bien formado.

HTML5

No es una reformulación de las versiones anteriores del lenguaje, de hecho, incluye todos los elementos válidos tanto en HTML 4 como en XHTML 1.0. Por otra parte, ha sido diseñado con unos principios fundamentales en mente para asegurarse de que funciona en la mayoría de las plataformas existentes en el mercado, es compatible con los navegadores antiguos y se ocupa de los errores de una forma amigable.

En primer lugar, HTML5 incluye nuevos elementos de marcado cuya **semántica** está asociada con el significado de los contenidos que vamos a introducir en ellos.

Además, ha sido utilizado para referirse a una serie de nuevas tecnologías y APIs, como son: el dibujo con el elemento **canvas** ,almacenamiento fuera de línea, el nuevo **video** y **audio** , funcionalidades de arrastrar y soltar, microdatos, etc. A lo largo del curso, veremos estos conceptos y tecnologías.

Por todo esto, ayudan a que nuestros documentos sean más accesibles para los seres humanos y las máquinas, lo cual, resulta beneficioso tanto para la accesibilidad como para el SEO y además hay una menor dependencia de software de terceros y plugins al publicar contenido multimedia en la Web.

Los navegadores

Aunque HTML5 aún está en desarrollo, y presenta cambios significativos en la forma que los contenidos son marcados, vale la pena señalar que estos cambios no causarán que los navegadores más antiguos se queden obsoletos, o den lugar a problemas de diseño o errores de página.

Lo que esto significa es que podemos tomar cualquiera de nuestros proyectos actuales que contienen HTML 4 válido o XHTML, y cambiando el tipo de documento (DOCTYPE) a HTML5, la página seguirá validando y su aspecto será el mismo que antes.

Los cambios y mejoras de HTML5 se han implementado en el lenguaje de tal manera que se asegura la compatibilidad hacia atrás con la mayoría de navegadores, incluso IE6.

Pero eso es sólo el mercado, ¿qué pasa con las tecnologías asociadas? De acuerdo con un conjunto de estadísticas, alrededor del 47 % de los usuarios están en una versión de Internet Explorer que no tiene soporte para la mayoría de estas nuevas características.

Como resultado, los desarrolladores han llegado a varias soluciones para proporcionar la experiencia equivalente a estos usuarios.. A veces, es tan simple como proporcionar contenido de reserva, como un reproductor de vídeo Flash para navegadores sin soporte de vídeo nativo. En otras ocasiones, sin embargo, se hace necesario el uso de scripts que imiten el comportamiento de estas nuevas características. Estas técnicas se conocen como `polyfills` .

Por supuesto, vale la pena señalar que a veces no se requieren este tipo de

Sintaxis HTML

Si examina el ejemplo de la figura siguiente, se ve que el código de la página web es una mezcla de del texto que el usuario ve en el navegador rodeado por elemento de marcado. En HTML, **un elemento** es un par de etiquetas HTML que contienen contenido.

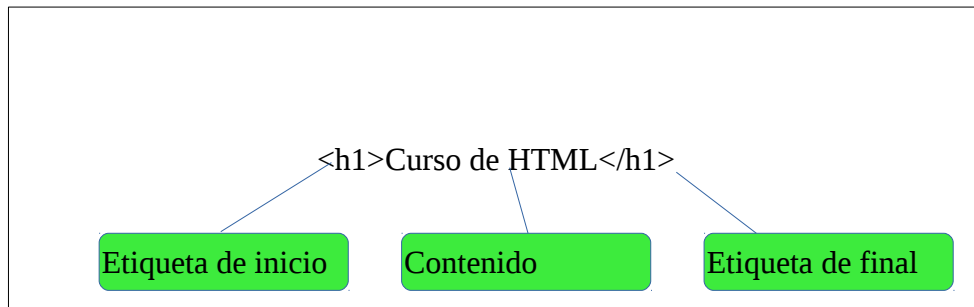


Figura 1: la sintaxis de un elemento HTML.

Algunos elementos HTML contienen solamente una sola etiqueta. Estos son conocidos como elementos huecos porque no contienen contenido. Sin embargo, estos insertan algo en la página, tales como una nueva línea utilizando el elemento `
`. Los elementos vacíos sólo utilizan la etiqueta de apertura, no una etiqueta de cierre.

Cada elemento HTML determina **cómo el contenido se organiza y se muestra en el navegador**. Por ejemplo, el elemento `<h>` crea un encabezado en negrita para cualquier texto que lo contiene.

Otros elementos HTML describen otros tipos de texto. El elemento `<p>` crea un párrafo de texto como se muestra aquí.

```
<p>HTML is a markup language, a structured language that lets
you identify common sections of a document such as headings,
paragraphs, and lists. An HTML file includes text and HTML
markup elements that identify these sections. The HTML markup
elements indicate how the document sections appear in a
browser.</p>
```

Algunos elementos HTML soportan **atributos** que proporcionan más información sobre el elemento. Veamos un atributo añadido a un elemento `<h1>`.

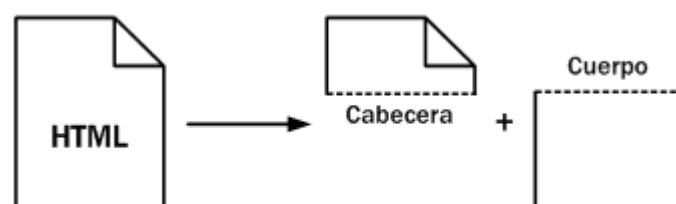
```
<h1 id="maintitle">Main Title of the Document</h1>
```

Todos los elementos tienen una serie de atributos comunes que se pueden utilizar de la misma forma en cualquier elemento. Son los siguientes:

Atributo	Significado
id	Identifica al elemento. Se trata de un identificador en el sentido XML (debe empezar por letra, no tener espacios, sí puede tener números y no puede repetirse su valor en dos elementos de la misma página web). Es uno de los atributos más importantes.
title	Permite poner un título al elemento. Es una especie de descripción corta que es muy útil en aquellos elementos que no muestran información que requiere una explicación (como un abreviatura, una sigla, una imagen, ...). El navegador reacciona a este atributo mostrando un cartelito cuando el cursor del ratón se aproxima a un elemento que use este atributo (ver imagen inferior)
lang	Indica el lenguaje en el que está escrito el contenido del elemento (para español: lang="es"). Sólo se usa si algún elemento usa un lenguaje distinto al indicado para toda la página.
dir	Indica la dirección de lectura del texto contenido en el elemento. Puede ser uno de estos dos valores: ltr (left to right, de izquierda a derecha) o rtl (right to left, de derecha a izquierda).
class style	Permiten especificar formato de tipo CSS al atributo

El primer documento HTML

Las páginas HTML se dividen en dos partes: la **cabecera** y el **cuerpo**. La cabecera incluye información sobre la propia página, como por ejemplo su título y su idioma. El cuerpo de la página incluye todos sus contenidos, como párrafos de texto e imágenes.



Veamos, a continuación, el esqueleto de un documento HTML5 y estudiemos las diferencias existentes con las versiones HTML4 y/o XHTML:

```
<!Doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Plantilla HTML5</title>
<meta name="description" content="HTML5">
<meta name="author" content="Julio Martinez">
<link rel="stylesheet" href="css/estilos.css">
</head>
<body>
<script src="js/scripts.js"></script>
</body>
</html>
```

Si estamos haciendo la transición a HTML5 de XHTML o HTML 4, enseguida nos damos cuenta de un buen número de áreas en las que HTML5 difiere.

El Doctype

En primer lugar, tenemos el tipo de declaración de documento o `doctype`. Esto es, simplemente, una manera de decirle al navegador, o cualquier otro software que analice nuestro código, de qué tipo de documento se trata. En el caso de los archivos HTML, indica la versión específica.

El tipo de documento debe ser siempre el primer punto en la parte superior de todos los archivos HTML. Anteriormente, la declaración de tipo de documento era farragosa y difícil de recordar. Para XHTML 1.0 Strict:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Y para HTML4 Transitional:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

HTML5 ha acabado con esa monstruosidad indescifrable. Ahora todo lo que indicamos es lo siguiente:

```
<!DOCTYPE HTML>
```

Lo primero que podemos observar es que el 5 está ausente de la declaración. Aunque la iteración actual del lenguaje de marcado se conoce como "HTML5", en realidad es sólo una evolución de los estándares anteriores, y las futuras especificaciones serán, simplemente, un desarrollo de lo que tenemos hoy. Dado que los navegadores tienen que soportar todo el contenido existente en la web, no se confía en el tipo de documento para indicar qué características deben soportarse en un documento dado.

La **validación** es el proceso que asegura que un documento escrito en un determinado lenguaje, por ejemplo HTML5, cumple con las normas y restricciones de ese lenguaje.

El proceso de validación consiste en probar página a página si el código HTML5 pasa la prueba de validación. Los validadores son las herramientas que se utilizan para validar cada página. El organismo W3C ha creado una herramienta que se puede utilizar gratuitamente a través de Internet (<http://validator.w3.org/>).

El elemento html

El siguiente paso en cualquier documento HTML es el elemento `<html>`, que no ha cambiado significativamente con HTML5. En nuestro ejemplo, hemos incluido el atributo `lang` con un valor de `es`, que especifica que el documento está en español. En la sintaxis basada en XHTML, estaríamos obligados a incluir un atributo `xmlns`. En HTML5, esto ya no es necesario, e incluso podríamos prescindir del atributo `lang` y el documento validaría y funcionaría correctamente.

El elemento head

La siguiente parte de nuestro documento es la sección `<head>`. La primera línea dentro de la cabecera es la que define la codificación de caracteres que se utiliza en el documento. Este es otro elemento que se ha simplificado. Anteriormente, hacíamos esto:

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

HTML5 reduce este meta a la mínima expresión:

```
<meta charset="utf-8">
```

Es importante que la declaración de la codificación del documento esté incluida dentro de los primeros 512 caracteres del documento. También es imprescindible que esté antes de la aparición de cualquier elemento de contenido (como el elemento `<title>` que le sigue en nuestro ejemplo).

La siguiente parte de nuestro head es la siguiente:

```
<title>Plantilla HTML5</title>
<meta name="description" content="HTML5">
<meta name="author" content="Alejandro Amat">
<link rel="stylesheet" href="css/estilos.css">
```

En estas líneas, HTML5 apenas se diferencia de sintaxis anteriores. El título permanece igual que antes, y las etiquetas `<meta>` que hemos incluido no son más que ejemplos opcionales.

Sí que vemos una sutil diferencia a la hora de incluir la hoja de estilo. A primera vista, es probable que no hayas notado nada diferente, pero habitualmente, elementos de enlace incluirían un atributo `type` con un valor de `text/css`.

Curiosamente, esto nunca fue necesario en XHTML o HTML 4, incluso cuando utilizábamos `doctype strict`. La sintaxis basada en HTML5 nos anima a omitir el atributo de tipo, ya que todos los navegadores reconocen el tipo de contenido de las hojas de estilo vinculadas sin requerir del atributo extra.

El resto no cambia

Mirando el resto del documento básico, vemos que tenemos el elemento `<body>` y la etiqueta de cierre `</html>`, las cuales, no varían en relación a versiones anteriores del HTML.

Al igual que se señaló anteriormente con el elemento `link`, la etiqueta `<script>` no requiere que se declare un atributo `type`. En XHTML, para validar una página que contiene scripts externos, su etiqueta `<script>` debería tener este aspecto:

```
<script src="js/scripts.js"
type="text/javascript"></script>
```

Dado que, a todos los efectos prácticos, JavaScript es el único lenguaje de programación real utilizado en la Web, todos los navegadores asumirán que estamos usando JavaScript, incluso cuando no lo declaremos explícitamente, por lo tanto, el atributo de tipo es innecesario en los documentos HTML5:

```
<script src="js/scripts.js"></script>
```

El motivo de poner el elemento script en la parte inferior de nuestra página, es seguir las buenas prácticas de incrustación de código javascript. Esto tiene que ver con la velocidad de carga de la página, cuando el navegador encuentra un script , se hará una pausa en la descarga mientras se ejecuta el código javascript, haciendo que el resto de la página no se descargue hasta que termine la ejecución.

Esto se traduce en que la página tarda más en cargar cuando incluimos scripts grandes en la parte superior de la misma.

Esta es la razón por la que la mayoría de los scripts deben ser colocados en la parte inferior de la página, de modo que sólo serán ejecutados después de que el resto de la página se haya cargado.

En algunos casos, puede **ser necesario colocar el script en la cabecera del documento**, ya que deseamos que se ejecute antes de que el explorador inicie la presentación de la página.

Diferencias con XHTML

En XHTML todos los elementos deben ser cerrados, ya sea con la etiqueta de cierre correspondiente (como </html>) o, en el caso de elementos vacíos, una barra al final de la etiqueta. Estos últimos son elementos que no pueden contener elementos secundarios (tales como input , img o link). En HTML5, ya no es necesario agregar esta barra.

El validador de HTML 5 sólo dará error para las cuestiones de código que amenazan con afectar al documento de alguna manera. Por ejemplo, para el validador, las siguientes cinco líneas son idénticas:

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

```
<LINK REL="stylesheet" HREF="css/styles.css">
<Link Rel="stylesheet" Href="css/styles.css">
<link rel=stylesheet href=css/styles.css>
```

En HTML5, podemos usar **minúsculas, mayúsculas, o mayúsculas y minúsculas en los nombres o atributos de etiqueta, así como introducir valores de atributos sin comillas (siempre y cuando estos valores no contengan espacios u otros caracteres**

reservados), y todo será correcto para el validador.

En HTML5, los atributos booleanos (sólo pueden tomar los valores “on” “off”), simplemente, se pueden especificar sin valor. Por lo tanto, este código XHTML: `<input type="text" disabled="disabled" />` se puede escribir de la siguiente manera: `<input type="text" disabled>`

En cualquier caso, es muy recomendable seguir unas directrices a la hora de codificar nuestras páginas, por ejemplo, sería una buena práctica mantener las restricciones en cuanto a mayúsculas y minúsculas y/o seguir poniendo los valores de los atributos entre comillas.

Etiquetas básicas

Antes de empezar con los elementos nuevos de HTML5, vamos a dar un repaso de las etiquetas básicas.

Texto

Cabeceras

Las cabeceras en HTML5 están definidas con los elementos `<h1>` hasta `<h6>` .

```
<h1>Cabecera de nivel 1</h1>
<h2>Cabecera de nivel 2</h2>
<h3>Cabecera de nivel 3</h3>
...
```

Las cabeceras van a definir el esquema del documento, por lo tanto, debemos utilizarlas de forma coherente. No tiene sentido tener un `<h2>` si no tenemos antes

un `<h1>` , por ejemplo.

Párrafos

Los párrafos de texto los introduciremos mediante la etiqueta `<p>` .

```
<p>Esto es un párrafo</p>
<p>Esto es otro párrafo</p>
```

No debemos utilizar los párrafos para introducir saltos de línea, para eso ya disponemos del elemento `
` .

Línea horizontal

Otra posibilidad es hacer un salto pero dejando una línea horizontal en el hueco de las palabras. Esto lo hace la etiqueta `hr` (que tampoco tiene cierre):

```
<p>Primera línea <hr/>Segunda línea</p>
```

Texto preformateado

En ocasiones, es necesario mostrar los espacios en blanco de un texto que no se puede modificar. Se trata de un caso habitual cuando una página web debe mostrar directamente el texto generado por alguna aplicación.

En estos casos, se puede utilizar la etiqueta `<pre>`, que muestra el texto tal y como se ha escrito, respetando todos los espacios en blanco y todas las nuevas líneas.

El siguiente ejemplo muestra el uso de la etiqueta `<pre>`:

```
<pre>
  My Bonnie lies over the ocean.
  My Bonnie lies over the sea.
  My Bonnie lies over the ocean.
      Oh, bring back my Bonnie to me.
</pre>
```

My Bonnie lies over the ocean.

My Bonnie lies over the sea.

My Bonnie lies over the ocean.

Oh, bring back my Bonnie to me.

Resaltado básico de caracteres

Son elementos que permiten marcar el texto de forma especial.

Negrita `` y también (aunque se considera en desaparición) el elemento `b` (de bold, negrita).

Cursiva `` y también `<i>` (de italic, aunque también en desaparición).

Subíndice Permite que el texto aparezca por debajo de la línea base y en un tamaño más pequeño. Lo hace el elemento `sub`. Ejemplo:

```
<p>La fórmula del agua es  
H<sub>2</sub>O</p>
```

H₂O

Modificaciones Permite marcar de forma adecuada las modificaciones realizadas en el contenido de una página.

```
<p>Mi color favorito es <del>azul</del>  
<ins>rojo</ins>!</p>
```

Mi color favorito es azul rojo!

Citar texto externo Es habitual citar literalmente un texto externo. HTML define la etiqueta `<blockquote>` para incluir citas textuales en las páginas web.

```
<blockquote cite=
```

```
"http://www.worldwildlife.org/who/
```

```
index.html">      For 50 years, WWF  
has been protecting the future of  
nature. The world's leading  
conservation organization, WWF works  
in 100 countries and is supported by  
1.2 million members in the United
```

States and close to 5 million globally.

</blockquote>

Marcado avanzado de caracteres

En realidad son elementos poco utilizados para marcar texto, pero son muy interesantes para dar significado al mismo. La esperanza es que en el futuro haya cada vez más herramientas software capaces de manipular de forma adecuada estos elementos y así poder dar grandes posibilidades al texto escrito de las páginas web.

La lista de elementos relacionados con este marcado es:

Elemento	Significado	Atributos que se suelen utilizar con el elemento	
abbr	Indica una abreviatura (por ejemplo O.N.U.). <abbr title="Su Alteza Real">SAR</abbr>	title	Permite indicar el significado de la abreviatura
acronym	Indica un acrónimo (por ejemplo tlfn) <acronym title="Teléfono">tlfn</acronym> En algunos navegadores una línea punteada bajo las abreviaturas y acrónimos permite indicar al usuario que al arrimar el ratón se explican los mismos	title	Permite indicar el significado del acrónimo
dfn	Permite indicar la definición de un término: <dfn title="usar dos conceptos de significado opuesto en una sola expresión">oxímoron</dfn> Los navegadores suelen mostrar este texto en negrita	title	Permite indicar la explicación de la definición
cite	Marca un texto como cita literal. <cite>Alea Jacta Est</cite> Los navegadores lo suelen mostrar en cursiva		
cod	Permite indicar que el texto al que engloba pertenece a código fuente de un lenguaje de programación.		
samp	Indica un ejemplo de salida por pantalla		

	de un programa informático		
kbd	Se usa para indicar que el texto incluido indica una tecla del teclado o una combinación de teclas.	title	Para explicar la tecla
var	Indica nombres de variables		
time	Disponible desde HTML 5 permite indicar una hora (por ejemplo 10:00)		
bdo	Permite indicar la dirección del texto. Ejemplo: <code><bdo dir="rtl">Este texto sale al revés</bdo></code> Saldría: “séver la elas otzet etsE “	dir	Posee dos valores ltr (el texto se muestra de izquierda a derecha) y rtl (el texto se muestra de derecha a izquierda)

Listas

En HTML disponemos de dos tipos de listas: listas desordenadas, cuyos elementos no están numerados, y listas ordenadas, cuyos elementos están numerados de alguna forma, ya sea con números, letras, números romanos, etc.

Listas desordenadas

Una lista desordenada comienza con la etiqueta ``, y cada elemento de la lista comienza con la etiqueta ``.

Los elementos de la lista están marcados con viñetas, normalmente, pequeños círculos negros.

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

El código anterior se vería así:

- Coffee
- Milk

Listas ordenadas

La única diferencia entre una lista desordenada y una ordenada es que la ordenada comienza con la etiqueta ``, en lugar de la etiqueta ``.

```
<ol>
```

```
<li>Coffee</li>
<li>Milk</li>
</ol>
```

Se vería así:

1. Coffee
2. Milk

Enlaces

Para crear enlaces, utilizaremos la etiqueta `<a>` con su atributo `href` para indicar la url de destino del enlace. `Texto del enlace`

Si queremos crear un enlace que se dirija a un sitio determinado del documento actual, haremos lo siguiente:

1. Colocaremos un ancla en el lugar al que nos queremos dirigir:

```
<a id="contacto">Contacto</a>
```

2. En el enlace pondremos como `href` el símbolo `#` seguido del id que le hemos puesto al ancla: `Ir a contacto`

Atributos del elemento a

Realmente el único de obligado uso es `href`, pero es posible utilizar los siguientes atributos (además de los atributos comunes a todas las etiquetas comentados anteriormente):

Atributo	Significado
<code>hreflang</code>	Permite indicar un código de lenguaje (es, fr, en,...) indicando el lenguaje en el que está escrito el destino del enlace
<code>media</code>	Sólo válido en HTML5, permite indicar el medio idóneo para mostrar el contenido del enlace.
<code>target</code>	Permite indicar cómo se muestra la página de destino. Posibilidades: <ul style="list-style-type: none">• <code>_blank</code>. Abre el enlace en una nueva página

	<ul style="list-style-type: none"> • <code>_parent</code>. Abre el enlace en el marco de la página padre de ella. • <code>_top</code>. Abre la página en el marco superior • <code>_self</code>. Abre la página en el marco actual • <code>nombre</code>. EL nombre indicado será el del marco en el que se abrirá la página • Salvo el primero, el resto no se usan por referirse a marcos
<code>rel</code>	<p>Informa sobre la función del enlace. Puede ser:</p> <ul style="list-style-type: none"> • <code>alternate</code>. Enlace alternativo • <code>author</code>. • <code>bookmark</code>. Página de marcadores • <code>help</code>. Página de ayuda • <code>license</code>. Información sobre la licencia • <code>next</code>. Si nuestra página pertenece a una serie ordenada, el enlace nos lleva al siguiente elemento dentro de la serie. • <code>nofollow</code>. Marca que los robots de búsqueda de empresas como Google no tengan en cuenta los enlaces externos y así evitar que dichos enlaces en las páginas se utilicen para subir su calificación en los buscadores. Así se ignoran por los robots los enlaces marcados de esta forma. • <code>noreferrer</code>. Un referrer es la información relativa a la página desde la que procede el visitante a un sitio. Con este valor en los enlaces, no se indicará al destino URL la página desde la que procedía el usuario. • <code>prefetch</code>. Permite descargar el enlace antes de que el usuario haga clic en él y así acelerar su carga. Se usa (aunque pocos navegadores soportan este valor) en enlaces de uso habitual. • <code>prev</code>. Si nuestra página pertenece a una serie ordenada, el enlace nos lleva al elemento anterior dentro de la serie.

	<ul style="list-style-type: none"> • <code>search</code>. Página de búsqueda dentro de nuestro sitio web. • <code>tag</code>. Página con etiquetas de temas (tags) de nuestro sitio web.
--	--

Imágenes

La etiqueta `` es utilizada para incluir las imágenes en las páginas HTML sus atributos son:

Atributo	Significado
<code>alt</code>	Indica un texto alternativo. Ese texto aparece cuando la imagen no se ha podido cargar (o durante la carga). También suele aparecer cuando arrimamos el cursor a la imagen a fin de informarnos sobre ella. Es un texto también tenido en cuenta por los buscadores a fin de identificar lo que muestra la imagen.
<code>width</code>	<p>Anchura de la imagen. No es aconsejable usarlo para cambiar el tamaño de la imagen, ya que si la ampliamos no se verá en buena calidad y si la reducimos estaremos cargando una imagen grande para luego mostrarla en pequeño; sería más inteligente reducirla primero con un editor de imágenes.</p> <p>En cualquier caso es importante utilizar este atributo para que el navegador sepa de antemano el tamaño de la imagen y así que maquete la página correctamente.</p>
<code>height</code>	<p>Anchura de la imagen. No es aconsejable usarlo para cambiar el tamaño de la imagen, ya que si la ampliamos no se verá en buena calidad y si la reducimos estaremos cargando una imagen grande para luego mostrarla en pequeño; sería más inteligente reducirla primero con un editor de imágenes</p> <p>.En cualquier caso es importante utilizar este atributo para que el navegador sepa de antemano el tamaño de la imagen y así que maquete la página correctamente.</p>
<code>src</code>	"url" - Indica la URL de la imagen que se muestra

Los dos atributos requeridos son src y alt. El atributo src es similar al atributo href de los enlaces, ya que establece la URL de la imagen que se va a mostrar en la página. Las URL indicadas pueden ser absolutas o relativas. El atributo alt permite describir el contenido de la imagen mediante un texto breve. Las descripciones deben tener una longitud inferior a 1024 caracteres y son útiles para las personas y dispositivos discapacitados que no pueden acceder a las imágenes.

Ejemplo sencillo para incluir una imagen:

```

```

Si el valor del atributo width o height se indica mediante un número entero, el navegador supone que hace referencia a la unidad de medida píxel.

```

```

Tabla

Las tablas se definen con la etiqueta <table> y está dividida en filas con la etiqueta <tr> . Una fila se divide en celdas de datos con la etiqueta <td> , aunque también se puede dividir en celdas de cabecera con la etiqueta <th> . Los elementos <td> son los contenedores de datos en la tabla y pueden contener todo tipo de elementos HTML, como texto, imágenes, listas, otras tablas, etc.

Para crear una tabla como la siguiente:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Introduciríamos el siguiente código html:

```
<table>
  <tr> <th>Firstname</th>  <th>Lastname</th>
<th>Age</th>
  </tr>
  <tr>   <td>Jill</td>   <td>Smith</td>   <td>50</td>
  </tr>
```

```
<tr> <td>Eve</td> <td>Jackson</td> <td>94</td>
</tr>
<tr> <td>John</td> <td>Doe</td> <td>80</td>
</tr>
</table>
```

Evidentemente, con esto sólo conseguimos la estructura de la tabla, si queremos que nuestra tabla tenga aspecto mejor o diferente, tenemos que aplicar los estilos CSS correspondientes.

Elementos de bloque y en línea

Un **elemento en bloque** siempre se inicia en una nueva línea y ocupa todo el ancho disponible (se extiende hacia la izquierda y la derecha tanto como sea posible).

La etiqueta `<div>` es un elemento de bloque

La etiqueta `<div>` :

- Se utiliza para contener otros elementos.
- No requiere atributos

Otros ejemplos de elementos de bloque: `<h1>` - `<h6>` , `<p>` `<form>`

Para crear la siguiente página:

London

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.

Introduciríamos el siguiente código html:

```
<div style="background-color:black;color:white;padding:20px;">
  <h2>London</h2>
  <p>London is the capital city of England. It is the most
  populous city in the United Kingdom, with a metropolitan area
  of over 13 million inhabitants.</p>
  <p>Standing on the River Thames, London has been a major
  settlement for two millennia, its history going back to its
  founding by the Romans, who named it Londinium.</p>
</div>
```

En el ejemplo anterior podemos observar que al contenedor `<div>` se le añadido el atributo `style` para poder conseguir el efecto de la imagen.

Un **elemento en línea** no se inicia en una nueva línea y sólo ocupa tanto de ancho como sea necesario.

Esto es un elemento en línea `` dentro de un párrafo.

La etiqueta ``:

- Es utilizado para contener algún texto.
- No requiere atributos.

Otros ejemplos de elementos en línea: `<a>`, ``

Para crear la siguiente página:

My Important Heading

Introduciríamos el siguiente código html:

```
<h1>My <span style="color:red">Important</span>
Heading</h1>
```

Formularios

Los formularios HTML se utilizan para pasar datos a un servidor. Un formulario HTML puede contener elementos de entrada como campos de texto, casillas de verificación, radio botones, etc. Para crear un formulario HTML utilizamos la etiqueta `<form>` y dentro del `<form>` introduciremos los campos que nos interesen: `<input>`, `<select>`, etc.

Fundamentalmente la etiqueta form posee dos atributos:

- **action.** Es el atributo que contiene la URL de la página web o servicios que procesará los datos del formulario.
- **method.** Permite indicar qué instrucción http utilizaremos para pasar la información al destino de nuestro formulario. Las opciones habituales son GET y POST. No se distingue entre mayúsculas y minúsculas. La diferencia es que GET genera una cadena de búsqueda en la URL que contiene los datos del formulario; POST, por su parte, pasa los datos pero de forma más oculta.

En la siguiente tabla podemos ver los principales campos que podemos introducir en un formulario.

Elemento	Descripción
<code><input></code>	<p>El elemento más importante del formulario. Hay muchos tipos dependiendo del atributo type.</p> <p>Tipos frecuentes:</p> <ul style="list-style-type: none">• text Define una entrada de texto normal texto• radio Define una entrada para seleccionar una entre muchas opciones.

	<ul style="list-style-type: none"> • submit Define un botón para mandar el contenido del formulario.
<select>	Define una lista desplegable.
<textarea>	Define un campo de entrada de varias líneas
<button>	Define un botón

Para ver ejemplos de código que utilice formularios se puede consultar la web de w3schools (http://www.w3schools.com/html/html_forms.asp).

Una página web dentro de otra

Para introducir una página web dentro de otra, utilizamos el elemento `<iframe>`:
`<iframe src="URL"></iframe>`

La URL apunta a la dirección de la página que queremos mostrar dentro de esta. Podemos indicar el alto y ancho del `<iframe>` mediante los atributos **height** y **width** , normalmente expresados en píxeles, aunque también podemos utilizar porcentajes.

```
<iframe src="demo_iframe.htm" width="200"
height="200"></iframe>
```

Un `<iframe>` puede ser utilizado como destino de un enlace, para ello, sólo tenemos que indicar el nombre del `<iframe>` en el atributo `target` del enlace.

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>
<p><a href="http://www.w3schools.com"
target="iframe_a">W3Schools.com</a></p>
```

En este punto debes realizar el Ejercicio 1.

Definiendo la estructura del cuerpo

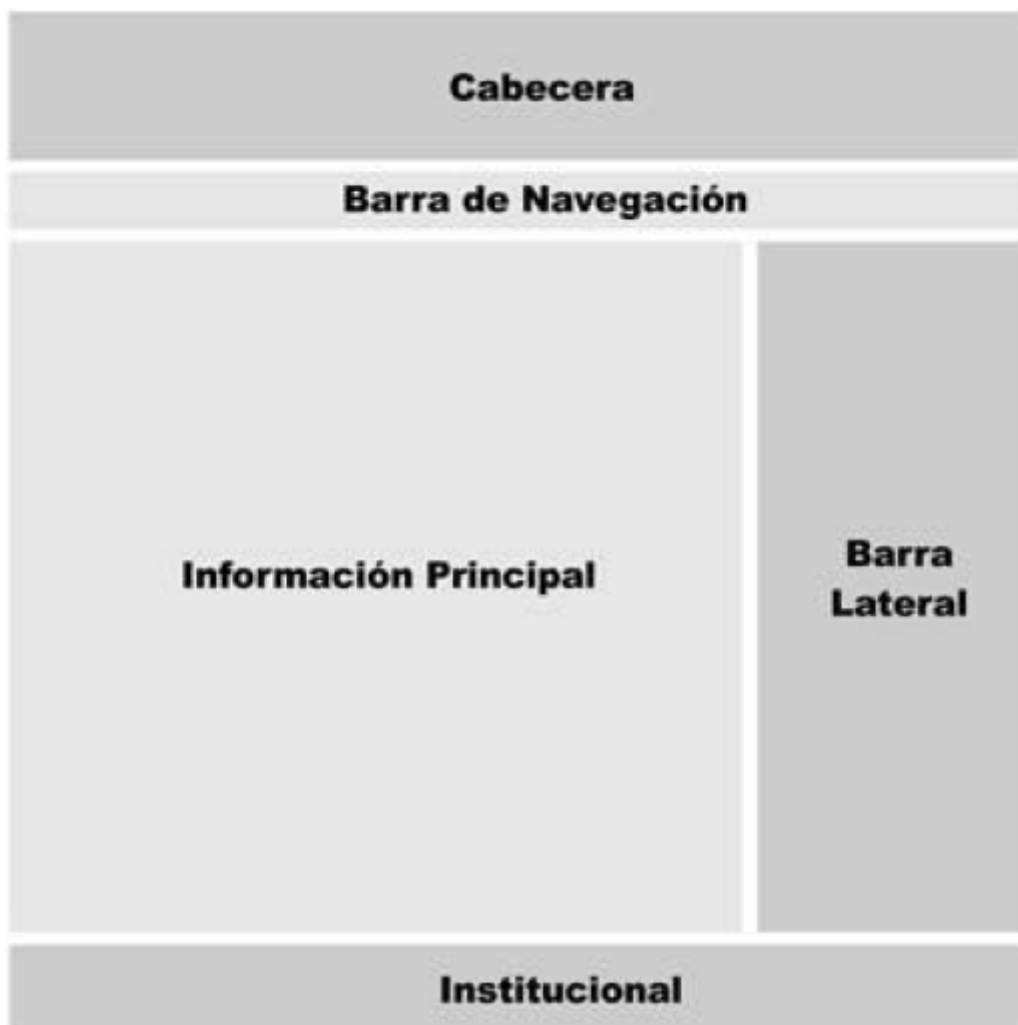
La estructura del cuerpo (el código entre las etiquetas `<body>`) generará la parte visible del documento. Este es el código que producirá nuestra página web. HTML siempre ofreció diferentes formas de construir y organizar la información dentro del cuerpo de un documento. Uno de los primeros elementos provistos para

este propósito fue `<table>` . Las tablas permitían a los diseñadores acomodar datos, texto, imágenes y herramientas dentro de filas y columnas de celdas, incluso sin que hayan sido concebidas para este propósito.

En los primeros días de la web, las tablas fueron una revolución, un gran paso hacia adelante con respecto a la visualización de los documentos y la experiencia ofrecida a los usuarios. Más adelante, gradualmente, otros elementos reemplazaron su función, permitiendo lograr lo mismo con menos código, facilitando de este modo la creación, permitiendo portabilidad y ayudando al mantenimiento de los sitios web.

El elemento `<div>` comenzó a dominar la escena. Con el surgimiento de webs más interactivas y la integración de HTML, CSS y Javascript, el uso de `<div>` se volvió una práctica común. Pero este elemento, así como `<table>` , no provee demasiada información acerca de la parte del cuerpo que está representando. Desde imágenes a menús, textos, enlaces, códigos, formularios, cualquier cosa puede ir entre las etiquetas de apertura y cierre de un elemento `<div>` . En otras palabras, el elemento `<div>` sólo especifica una división en el cuerpo, como la celda de una tabla, pero no ofrece indicio alguno sobre qué clase de división es, cuál es su propósito o qué contiene.

En el tema siguiente, veremos que HTML5 incorpora nuevos elementos que ayudan a identificar cada sección del documento y organizar el cuerpo del mismo, pero de momento vamos a ver cómo organizar el diseño de nuestra página utilizando elementos `<div>` .



En la imagen anterior podemos ver un diseño común encontrado en la mayoría de los sitios webs. A pesar del hecho de que cada diseñador crea sus propios diseños, en general, podremos identificar las siguientes secciones en cada sitio web estudiado.

En la parte superior, descrito como Cabecera , se encuentra el espacio donde usualmente se ubica el logo, título, subtítulos y una corta descripción del sitio web o la página.

Inmediatamente debajo, podemos ver la Barra de Navegación , en la cual, casi todos los desarrolladores ofrecen un menú o lista de enlaces con el propósito de facilitar la navegación a través del sitio. Los usuarios son guiados desde esta barra hacia las diferentes páginas o documentos, normalmente pertenecientes al mismo sitio web.

El contenido más relevante de una página web se encuentra, en casi todo diseño, ubicado en su centro. Esta sección presenta la información principal de la página y la mayoría de las veces está dividida en varias filas y columnas. En el ejemplo de la imagen se utilizaron sólo dos columnas: Información Principal y Barra Lateral ,pero esta sección es extremadamente flexible y cada diseñador la adapta a sus necesidades.

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

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

El código html que utilizaremos para crear esta disposición podría ser similar al siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Diseño web típico</title>
<meta name="description" content="HTML5">
<meta name="author" content="Alejandro Amat">
<link rel="stylesheet" href="css/estilos.css">
</head>
<body>
<div id="page">
<div id="header">Cabecera</div>
<div id="nav">Barra de navegación</div>
<div id="content">Información principal</div>
<div id="aside">Barra lateral</div>
```

```
<div id="pie">Institucional</div>
</div>
<script src="js/scripts.js"></script>
</body>
</html>
```

Evidentemente, si mostramos la página anterior en el navegador, el resultado obtenido no será el esperado, ya que, cada elemento aparecerá uno debajo del otro. Para solucionar este problema tenemos que crear nuestra hoja de estilos CSS que le dé a la página el aspecto que nos interese.

CSS aspectos básicos.

Hemos visto los aspectos básicos de html. Ahora los veremos de CSS.

El organismo [W3C](#) (World Wide Web Consortium), encargado de crear todos los estándares relacionados con la web, propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML y se presentaron nueve propuestas. Las dos propuestas que se tuvieron en cuenta fueron la CHSS (*Cascading HTML Style Sheets*) y la SSP (*Stream-based Style Sheet Proposal*).

La propuesta CHSS fue realizada por Håkon Wium Lie y SSP fue propuesto por Bert Bos. Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS (*Cascading Style Sheets*).

En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS y lo añadió a su grupo de trabajo de HTML. A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como "CSS nivel 1".

A principios de 1997, el W3C decide separar los trabajos del grupo de HTML en tres secciones: el grupo de trabajo de HTML, el grupo de trabajo de DOM y el grupo de trabajo de CSS.

Incorporar estilos al documento

Los navegadores proveen estilos por defecto que, pero para cambiar estos con los nuestros podemos utilizar diferentes técnicas:

- **Estilos en línea:** Una de las técnicas más simples para incorporar estilos CSS a un documento HTML es la de asignar los estilos dentro de las etiquetas por medio del atributo `style` . Pero no es recomendable para aplicar estilos a todo el documento.

```
<p style="font-size: 20px">Mi texto</p>
```

- **Estilos embebidos:** Una mejor alternativa es insertar los estilos en la cabecera del documento y luego usar referencias para afectar los elementosHTML correspondientes.

```
<style>
p { font-size: 20px }
</style>
```

El elemento `<style>` permite a los desarrolladores agrupar estilos CSS dentro del documento.

Este método sería bueno si sólo tuviéramos un documento en nuestra página, pero como habitualmente tendremos páginas formadas por varios documentos, el método siguiente es el más recomendable.

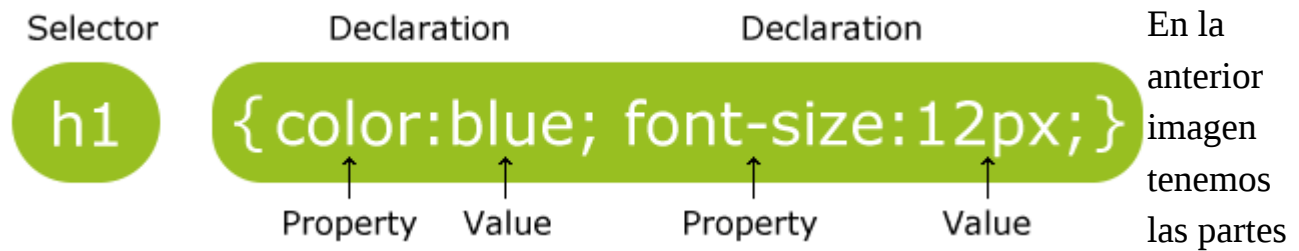
- **Archivos externos:** La solución es mover todos los estilos a un archivo externo, y luego utilizar el elemento `<link>` para insertar este archivo dentro de cada documento que los necesite. Este método nos permite cambiar los estilos por completo, simplemente, incluyendo un archivo diferente.

También nos permite modificar o adaptar nuestros documentos a cada circunstancia o dispositivo.

```
<link rel="stylesheet" href="misestilos.css">
```

Con la línea anterior le decimos al navegador que cargue el archivo `misestilos.css`, que contendrá todos los estilos necesarios para presentar el documento en pantalla.

Funcionamiento básico de las reglas CSS



que conforman un estilo básico.

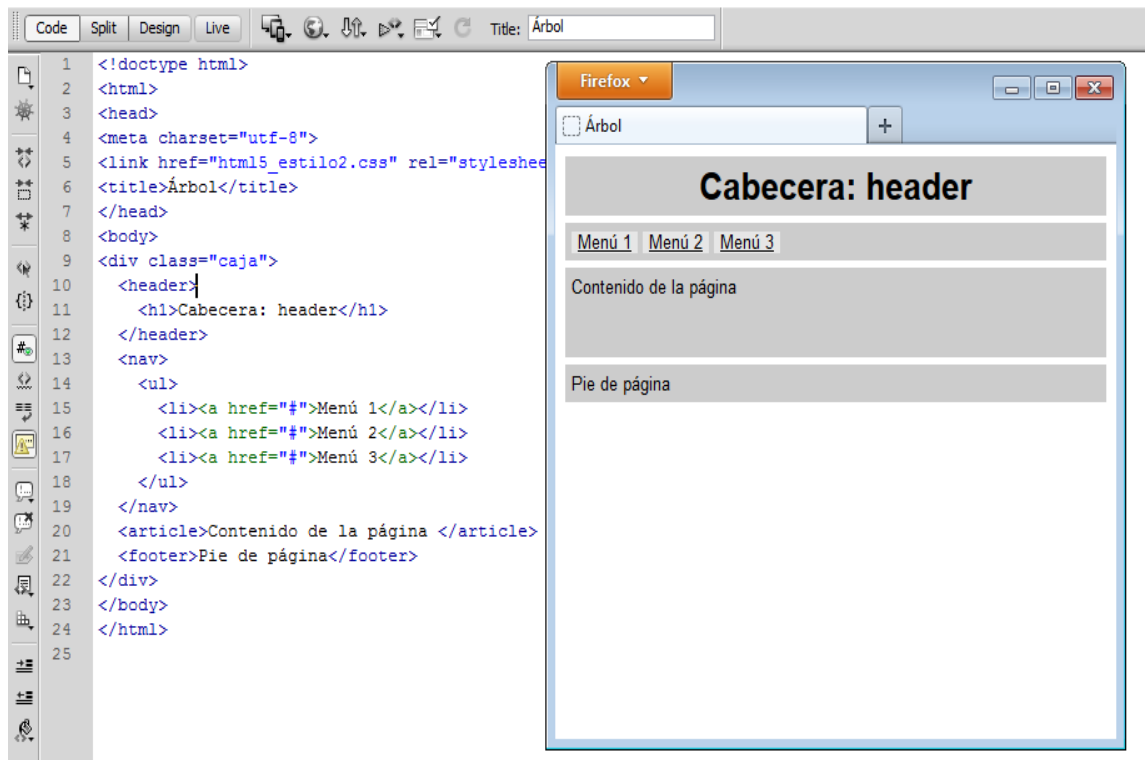
Que vamos a definir a continuación:

- **Regla:** cada uno de los estilos que componen una hoja de estilos CSS. Cada regla está compuesta por una parte denominada "selectores", un símbolo de "llave de apertura" (`{`), otra parte denominada "declaraciones" y, por último, un símbolo de "llave de cierre" (`}`).
- **Selector:** indica el elemento o elementos HTML a los que se aplica la regla CSS.
- **Declaración:** especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- **Propiedad:** permite modificar el aspecto de una característica del elemento.
- **Valor:** indica el nuevo valor de la característica modificada en el elemento.

Un archivo CSS puede contener infinitas reglas CSS, cada regla puede contener infinitos selectores y cada declaración puede estar formada por un número infinito de pares propiedad/valor

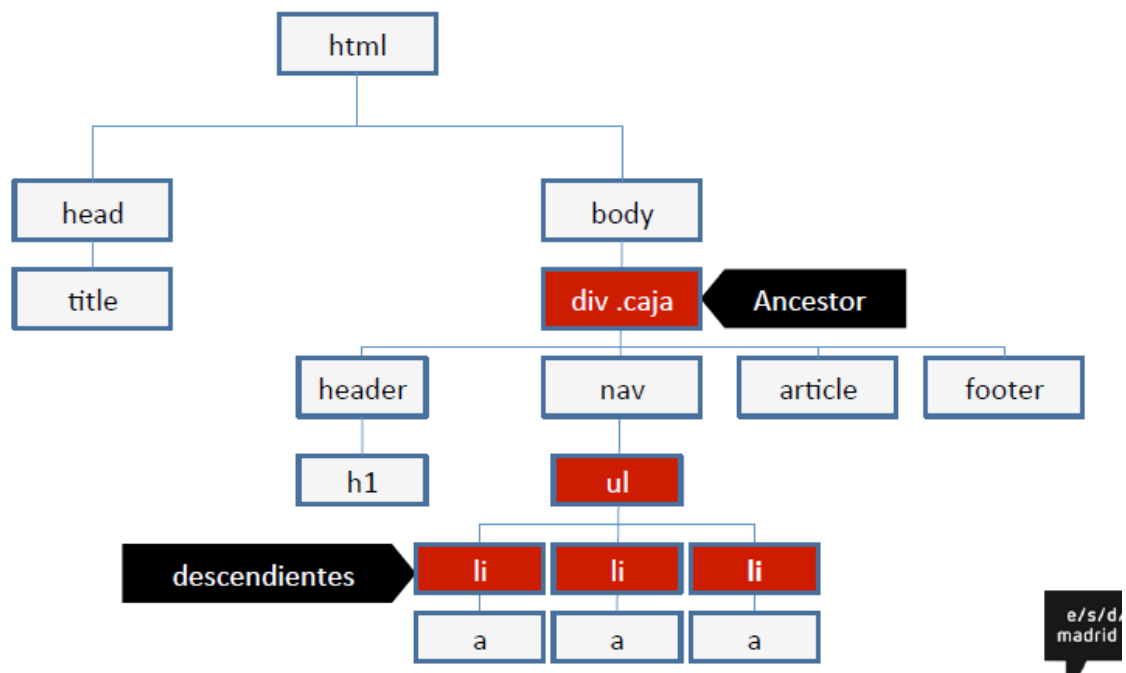
Herencia.

Para entender como funcionan los selectores y la herencia CSS es necesario entender qué es el árbol del documento. Si disponemos del siguiente ejemplo.



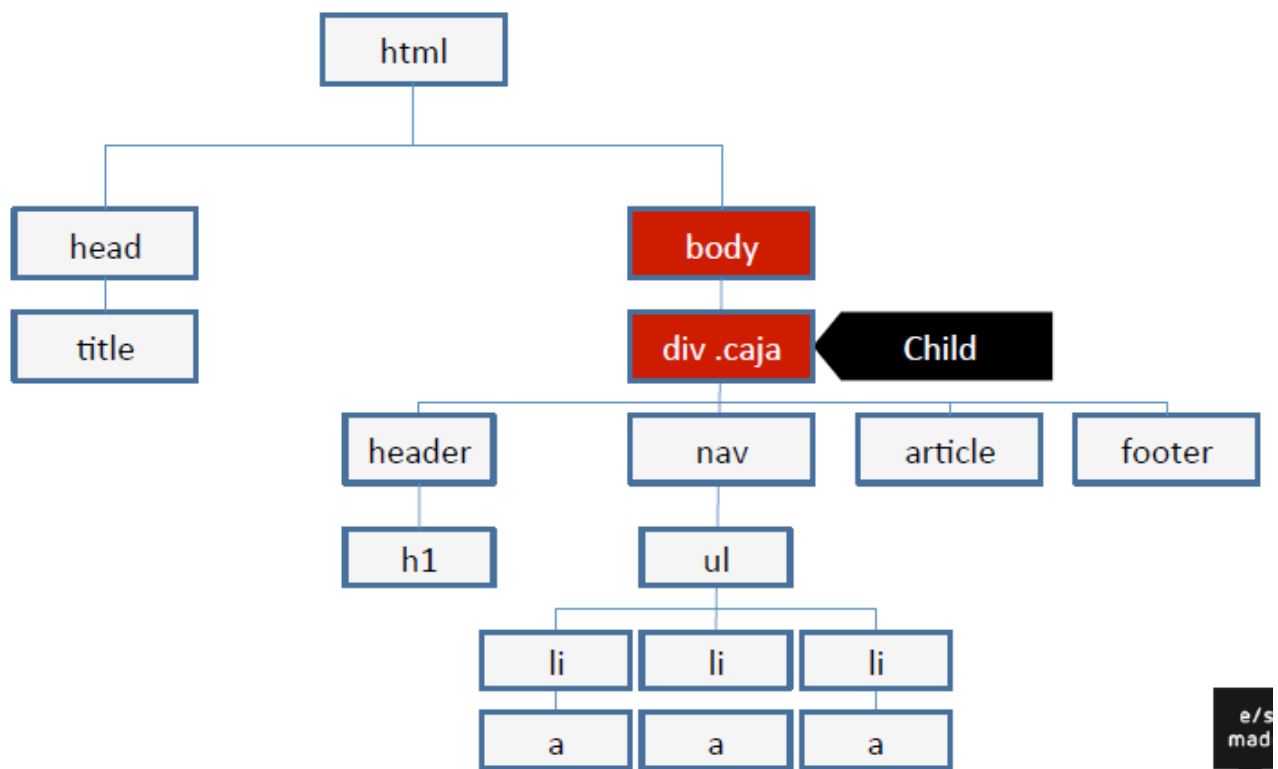
El árbol (simplificado) del documento del ejemplo anterior será:

El antecesor es un elemento conectado pero más arriba en la estructura del documento:



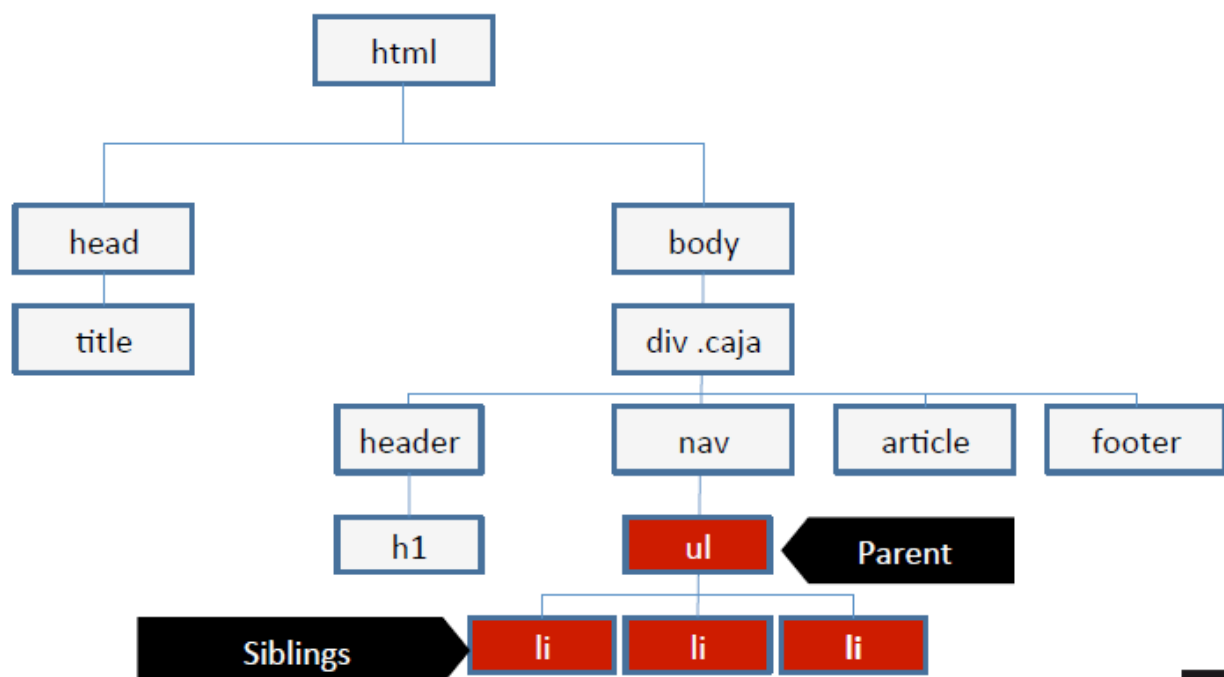
e/s/d,
madrid

Hijo es el elemento conectado y directamente debajo de un elemento en la estructura del documento.



e/s
mad

Hermanos o sibilings son los elementos que comparten un mismo padre en la estructura del documento.



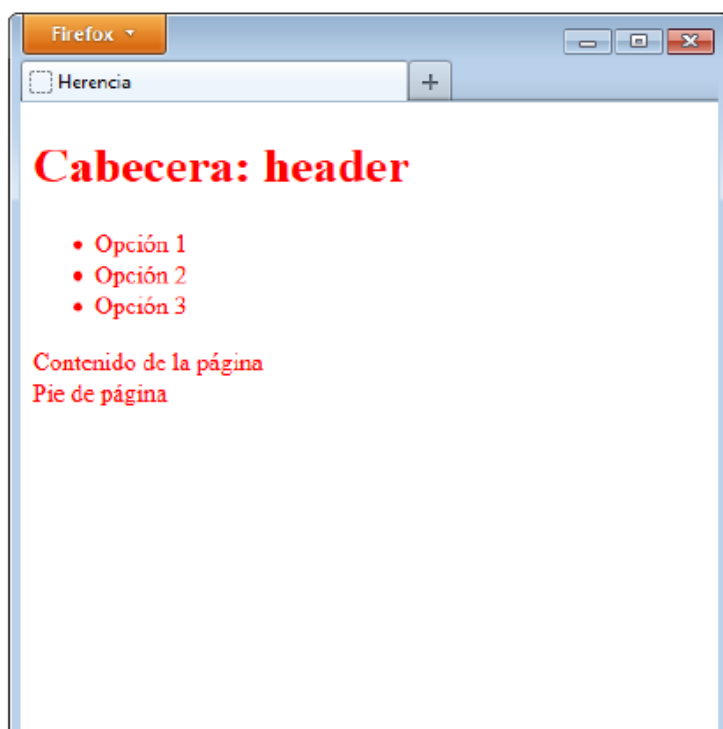
e/s/d
madrid

Si definimos un estilo en el **body**, todos los elementos situados debajo en el árbol del documento, heredan esa propiedad.

```

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Herencia</title>
<style type="text/css">
body {
    color: red;
}
</style>
</head>
</head>
<body>
<div class="caja">
    <header>
        <h1>Cabecera: header</h1>
    </header>
    <nav>
        <ul>
            <li>Opción 1</li>
            <li>Opción 2</li>
            <li>Opción 3</li>
        </ul>
    </nav>
    <article>Contenido de la página </article>
    <footer>Pie de página</footer>
</div>
</body>
</html>

```



Selectores

Selector de tipo o etiqueta

El selector de etiqueta aplica a todos los elementos HTML de la página con esa etiqueta (p).

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

El selector múltiple de CSS, incluye varios selectores separados por coma (,), aplicando el estilo a cualquier elemento con alguna de las marcas incluidas.

```
h1, h2, h3 {  
    color: #F00;  
}
```

Selector descendente

El selector puede incluir etiquetas separadas solo por espacios. El selector aplicará solo a elementos con la última marca (ul), con los anteriores como ancestros, es decir header deberá ser ancestro de nav y nav deberá ser ancestro de ul.

```
header nav ul {  
    color: red;  
}
```

Selector de clase .

Una de las soluciones más sencillas para aplicar estilos a un solo elemento de la página, consiste en utilizar el atributo class de HTML sobre ese elemento para indicar directamente la regla CSS que se le debe aplicar.

```
<p class="destacado">Lorem ipsum dolor sit amet...</p>
```

A continuación, se crea en el archivo CSS una nueva regla llamada destacado con todos los estilos que se van a aplicar al elemento. Para que el navegador no confunda este selector con los otros tipos de selectores, se prefija el valor del atributo class con un punto (.), tal y como muestra el siguiente ejemplo:

```
.destacado { color: red; }
```

En ocasiones, es necesario restringir el alcance del selector de clase.

```
<p class="destacado">Lorem ipsum dolor sit amet...</p>
<p>sed lacus et <a href="#" class="destacado">est
adipiscing</a> accumsan</p>
```

¿Cómo es posible aplicar estilos solamente al párrafo cuyo atributo class sea igual a destacado? Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico:

```
p.destacado { color: red }
```

El selector `p.destacado` se interpreta como "aquellos elementos de tipo `<p>` que dispongan de un atributo class con valor destacado". Es posible aplicar los estilos de varias clases CSS sobre un mismo elemento. La sintaxis es similar, pero los diferentes valores del atributo class se separan con espacios en blanco. En el siguiente ejemplo:

```
<p class="especial destacado error">Párrafo de texto...</p>
```

Al párrafo anterior se le aplican los estilos definidos en las reglas `.especial`, `.destacado` y `.error`.

Selector universal *

```
* { margin: 0px; padding: 0px; }.
```

Afecta a todos los elementos. Podemos utilizarlo de la siguiente forma. Para la mayoría de los elementos, necesitamos personalizar los márgenes o, simplemente, mantenerlos al mínimo. Algunos elementos, por defecto, tienen márgenes que son diferentes de cero y, en la mayoría de los casos, demasiado amplios. A medida que avanzamos en la creación de nuestro diseño, encontraremos que la mayoría de los elementos utilizados deben tener un margen de 0 píxeles. Para evitar tener que repetir estilos constantemente, podemos utilizar el selector universal.

Con la regla indicada anteriormente, nos aseguramos de que todo elemento tendrá un margen interno y externo de 0 píxeles. De ahora en adelante, sólo necesitaremos modificar los márgenes de los elementos que queremos que sean mayores que cero.

Selector de ID #

Para aplicar estilos CSS a un único elemento de la página. El selector de ID permite seleccionar un elemento de la página a través del valor de su atributo `id`. Este tipo de

selectores sólo seleccionan un elemento de la página, ya que el valor del atributo id no se puede repetir en dos elementos diferentes de una misma página.

```
#destacado { color: red; }  
<p>Primer párrafo</p>  
<p id="destacado">Segundo párrafo</p>  
<p>Tercer párrafo</p>
```

En el ejemplo anterior, el selector #destacado, solamente selecciona el segundo párrafo (cuyo atributo id es igual a destacado).

Selector de hijos >

Se trata de un selector similar al selector descendente, pero muy diferente en su funcionamiento. Se utiliza para seleccionar un elemento que es hijo directo de otro elemento y se indica mediante el "signo de mayor que" (>):

```
p > span { color: blue; }  
<p><span>Texto1</span></p>  
<p><a href="#"><span>Texto2</span></a></p>
```

En el ejemplo anterior, el selector p > span se interpreta como "cualquier elemento que sea hijo directo de un elemento <p> ", por lo que el primer elemento cumple la condición del selector. Sin embargo, el segundo elemento no la cumple porque es descendiente, pero no es hijo directo de un elemento <p> .

Selector adyacente +

El selector adyacente utiliza el signo + y su sintaxis es:

elemento1 + elemento2 { ... }

La explicación del comportamiento de este selector no es sencilla, ya que selecciona todos los elementos de tipo elemento2 que cumplan las dos siguientes condiciones:

1. elemento1 y elemento2 deben ser hermanos, por lo que su elemento padre debe ser el mismo.
2. Elemento2 debe aparecer inmediatamente después de elemento1 en el código HTML de la página.

En el siguiente ejemplo:


```
h1 + h2 { color: red }  
<body>  
<h1>Titulo1</h1>  
<h2>Subtítulo</h2>  
...  
<h2>Otro subtítulo</h2>  
...  
</body>
```

Los estilos del selector `h1 + h2` se aplican al primer elemento `<h2>` de la página, pero no al segundo `<h2>`, ya que: El elemento padre de `<h1>` es `<body>`, el mismo padre que el de los dos elementos `<h2>`.

Así, los dos elementos `<h2>` cumplen la primera condición del selector adyacente. El primer elemento `<h2>` aparece en el código HTML justo después del elemento `<h1>`, por lo que, este elemento `<h2>` también cumple la segunda condición del selector adyacente.

Por el contrario, el segundo elemento `<h2>` no aparece justo después del elemento `<h1>`, por lo que no cumple la segunda condición del selector adyacente y, por tanto, no se le aplican los estilos de `h1 + h2`.

Selector de atributos

Los selectores de atributos permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

Los cuatro tipos de selectores de atributos son:

- `[nombre_atributo]`, selecciona los elementos que tienen establecido el atributo llamado `nombre_atributo` independientemente de su valor.
- `[nombre_atributo=valor]`, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` con un valor igual a `valor`.
- `[nombre_atributo~=valor]`, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` y al menos uno de los valores del atributo es `valor`.
- `[nombre_atributo|=valor]`, selecciona los elementos que tienen establecido

- un atributo llamado `nombre_atributo`, cuyo valor es una serie de palabras separadas con guiones, pero que comienza con valor . Este tipo de selector sólo es útil para los atributos de tipo `lang` que indican el idioma del contenido del elemento.

A continuación, se muestran algunos ejemplos de estos tipos de selectores:

```
/* Se muestran de color azul todos los enlaces que tengan
un atributo "class", independientemente de su valor */
a[class] { color: blue; }

/* Se muestran de color azul todos los enlaces que tengan
un atributo "class" con el valor "externo" */
a[class="externo"] { color: blue; }

/* Se muestran de color azul todos los enlaces que apunten
al sitio "http://www.ejemplo.com" */
a[href="http://www.ejemplo.com"] { color: blue; }

/* Se muestran de color azul todos los enlaces que tengan
un atributo "class" en el que al menos uno de sus valores
sea "externo" */
a[class~="externo"] { color: blue; }

/* Selecciona todos los elementos de la página cuyo atributo
"lang" sea igual a "en", es decir, todos los elementos en
inglés */
*[lang=en] { ... }

/* Selecciona todos los elementos de la página cuyo atributo
"lang" empiece por "es", es decir, "es", "es-ES", "es-AR",
etc. */
*[lang|= "es"] { color : red }
```

Pseudo-clases

El concepto pseudo-clase se introdujo para permitir la selección de elementos sobre la base de la información que se encuentra fuera de la estructura del documento, o que no se puede expresar con los otros selectores simples.

Una pseudo-clase se compone siempre de "dos puntos" (:) seguido del nombre de la pseudo-clase y, opcionalmente, por un valor entre paréntesis.

Las pseudo-clases pueden ser dinámicas, es decir, un elemento puede adquirir o perder una pseudo-clase, mientras que un usuario interactúa con el documento. Ya en CSS2 se definieron una serie de pseudo-clases dinámicas que nos permitían cambiar los estilos de los enlaces en función de su estado o de cómo interactuara el usuario con ellos:

- `:link` permite aplicar estilos para los enlaces que aún no han sido visitados.
- `:visited` aplica estilos a los enlaces que han sido visitados anteriormente.
- `:focus` estilos que se aplican al enlace cuando este tiene el foco (acepta eventos de ratón o de teclado).
- `:hover` estilos que muestra el enlace cuando el usuario posiciona el puntero del ratón sobre el enlace.
- `:active` estilos que se aplican al enlace cuando el usuario está pinchando sobre el enlace (el tiempo durante el que se aplica este estilo es muy breve).

Las pseudo-clases `:link` y `:visited` solamente están definidas para los enlaces, pero las pseudo-clases `:hover` y `:active` se definen para todos los elementos HTML.

```
a:hover { text-decoration: none; }
```

En este ejemplo se elimina el subrayado del enlace cuando se sitúa el ratón sobre él.

Pseudo-elementos

Por último, CSS define unos elementos especiales llamados "pseudo-elementos" que permiten aplicar estilos a ciertas partes de un texto. En concreto, CSS permite definir estilos especiales a la primera frase de un texto y a la primera letra de un texto:

- El pseudo-elemento `:first-line` permite aplicar estilos a la primera línea de un texto.
- El pseudo-elemento `:first-letter` permite aplicar estilos a la primera letra del texto.
- pseudo-elementos `:before` y `:after` que nos permiten insertar contenidos antes o después de un elemento determinado. Para ello utilizaremos la propiedad CSS `content`, en la que indicaremos los contenidos que serán insertados.

```
a:after { content: " (" attr(href) ") "; }
```

El código CSS anterior añade después de cada enlace de la página un texto formado por la dirección a la que apunta el enlace mostrada entre paréntesis. Si se quiere añadir las direcciones antes de cada enlace, se puede utilizar el pseudo-elemento `:before`:

```
a:before { content: " (" attr(href) ") "; }
```

Ejercicio

```
<!DOCTYPE html>
<html lang="es-ES">
<head> <meta charset="UTF-8">
<title></title>
</head>
<body>
<h1>Monumentos de Palencia</h1>
<p>Palencia dispone de numerosos monumentos interesantes. Debido a su importante actividad medieval posee una gran cantidad de edificios religiosos entre los que destaca la <strong>Catedral de Palencia</strong> así como la iglesia de San Miguel con su peculiar torre de defensa y el monumento más conocido de la ciudad: El Cristo del Otero</p>
<p>A finales del siglo <em>XIX</em> y principios del <em>XX</em> aparecieron edificios suntuosos y civiles que han
```

```
embellecido una buena parte de la ciudad, en especial la
transitada Calle Mayor.</p> <h2>Edificios religiosos</h2>
<ul>
<li>Cristo del Otero</li>
<li>Catedral Mayor</li>
<li>Iglesia de San Miguel</li>
<li>Iglesia de San Lázaro</li>
<li>Convento de San Pablo</li>
<li>Iglesia de la Compañía</li>
</ul>
</body>
</html>
```

A partir de la anterior página de ejemplo, incluir los siguientes estilos.

```
/* Todos los elementos de la pagina */
{ font: 1em/1.3 Arial, Helvetica, sans-serif; }

/* Todos los parrafos de la pagina */
{ color: #555; }

/* Todos los párrafos contenidos en #primero */
{ color: #336699; }

/* Todos los enlaces la pagina */
{ color: #CC3300; }

/* Los elementos "em" contenidos en #primero */
{ background: #FFFFCC; padding: .1em; }
```

```
/* Todos los elementos "em" de clase "especial" en toda la
pagina */
{ background: #FFCC99; border: 1px solid #FF9900; padding: .
1em; }

/* Elementos "span" contenidos en .normal */
{ font-weight: bold; }
/* Elementos primeros de la lista li */
{ color: #FFFFFF; }
/* Primeras lineas de los Parrafos */
{ color: #000000; }
```

Prioridad

Cuando dos declaraciones afectan a un mismo elemento ¿Cuál de ellas se interpreta en el navegador como más importante?

Hay que calcular la tupla (A,B,C,D) ganadora de todas las reglas que compiten.

Siendo A la que tiene mayor peso y D el mínimo. Si hay empate en A, se mira B y así sucesivamente.

- **A** = estilo en línea.
- **B** = número de Ids
- **C** = número de Clases
- **D** = número de marcas HTML


```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>Herencia</title>
```

```
<style type="text/css">
```

```
#caja header h1 { color: blue; }
```

```
#caja .cabecera h1 { color: red; }
```

```
header h1 { color: lime; }
```

```
h1 { color: purple; }
```

```
h1 { color: grey; }
```

```
</style>
```

A=0 estilos de linea

B=0 ID

C=0 Clases

D= 1 marca

Puntuación=0,0,0,1

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>Herencia</title>
```

```
<style type="text/css">
```

```
#caja header h1 { color: blue; }
```

```
#caja .cabecera h1 { color: red; }
```

```
header h1 { color: lime; }
```

```
h1 { color: purple; }
```

```
h1 { color: grey; }
```

```
</style>
```

A=0 estilos de linea

B=0 ID

C=0 Clases

D= 2 marcas

Puntuación=0,0,0,2

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>Herencia</title>
```

```
<style type="text/css">
```

```
#caja header h1 { color: blue; }
```

```
#caja .cabecera h1 { color: red; }
```

```
header h1 { color: lime; }
```

```
h1 { color: purple; }
```

```
h1 { color: grey; }
```

```
</style>
```

A=0 estilos de linea

B=1 ID

C=1 Clases

D= 1 marcas

Puntuación=0,1,1,1

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>Herencia</title>
```

```
<style type="text/css">
```

```
#caja header h1 { color: blue; }
```

```
#caja .cabecera h1 { color: red; }
```

```
header h1 { color: lime; }
```

```
h1 { color: purple; }
```

```
h1 { color: grey; }
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div id="caja">
```

```
<header class="cabecera">
```

```
<h1>Cabecera: header</h1>
```

```
</header>
```

```
</div>
```

```
</body>
```

```
</html>
```

A=0 estilos de linea

B=1 ID

C=0 Clases

D= 2 marcas

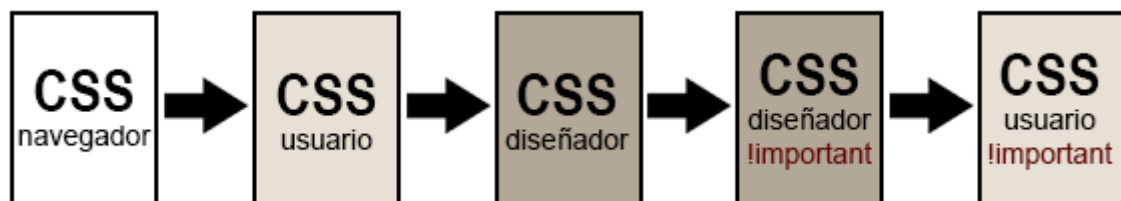
Puntuación=0,1,0,2

La ganadora es , #caja . Cabecera h1 =0, 1, 1, 1 .

Cuando dos declaraciones tienen el mismo valor, será la última especificada.

Hay reglas adicionales de prioridad de las declaraciones CSS ordenadas de menor a mayor:

- CSS por defecto, del Navegador (navegador)
- CSS en preferencias, de usuario, del navegador (usuario)
- CSS en página, HTML, o, script, CSS (diseñador),
- CSS en página, HTML, o, script, CSS con !important (diseñador), p. e. body, {color:blue, !important;},
- CSS en preferencias, de usuario, del navegador con !important (usuario), p. e. body, {color:blue, !important;},



Formas de indicar valores de propiedades CSS

Numerosas propiedades CSS tienen la necesidad de indicar valores. Por ejemplo el tamaño de la letra requiere un número. La cuestión es que CSS permite que dicho número se puede indicar con diferentes medidas (**píxeles, pulgadas, centímetros, etc.**) La idea actual en cuanto a CSS es que permita dar formato a documentos que no sólo se muestren en pantalla, sino en todo tipo de dispositivos de salida. Las unidades de medida más adecuadas, por lo tanto, pueden variar y de ahí que se nos permita elegir.

La forma de indicarlas es colocarlas detrás de la cantidad que usemos. Por ejemplo 12in significa doce pulgadas, mientras que 12cm significa doce centímetros.

Unidades de medida numéricas

- **in.** Pulgadas. Medida muy habitual en el mundo anglosajón, equivale a 25,4 milímetros.
- **cm.** Centímetro

- **mm.** Milímetro
- **pt.** Puntos. Medida muy utilizada en tipografía. Un punto tipográfico equivale a 1/72 pulgadas. Como todos los sistemas operativos le utilizan en los tipos de letra, lo cierto es que es de uso común para utilizar tamaños de fuentes.
- **pc.** Pica. Una pica la forman doce puntos. Y seis puntos equivalen a una pulgada.
- **ex.** Tamaño relativo respecto a la letra equis minúscula (x). Así 2x indica que el tipo de letra aumenta hasta ocupar el doble la letra x mayúscula. Con 0.5x ocuparía la mitad
- **em.** Tamaño relativo, en este caso, respecto a la letra M mayúscula
- **ch.** Altura relativa al número cero (0) en la tipografía actual. No se debe utilizar, ya que la mayoría de navegadores no le reconoce
- **px.** Píxeles. Esta medida es relativa respecto al dispositivo de salida. Ya que en cada dispositivo el tamaño del píxel varía. Se usa mucho en elementos grandes (capas, tablas,)
- **%. Porcentaje.** Es relativo respecto del tamaño del elemento padre del elemento al que le ponemos esta medida. Así 50% para una tabla dentro del elemento body ocuparía la mitad de la pantalla, pero dentro de una capa ocuparía la mitad del tamaño de la capa.
- **dpi. Puntos por pulgada.** Se usa para indicar resoluciones (para especificar, por ejemplo, la resolución mínima del dispositivo que debe tener para mostrar los estilos)

Indicación de color

Hay numerosas etiquetas con capacidad de mostrar colores. Para indicar un color, CSS dispone de estas posibilidades:

- **Notación hexadecimal.** Se trata de la notación más utilizada en CSS. Consiste en una cifra hexadecimal, precedida del símbolo #. Las dos primeras cifras hexadecimales indican el nivel de rojo, las dos siguientes el nivel de verde y las dos últimas de verde; por ejemplo #FF0000 es el código del rojo puro.

Ejemplo: Podemos ver ejemplo de los colores en la siguiente dirección

http://www.w3schools.com/cssref/css_colors.asp

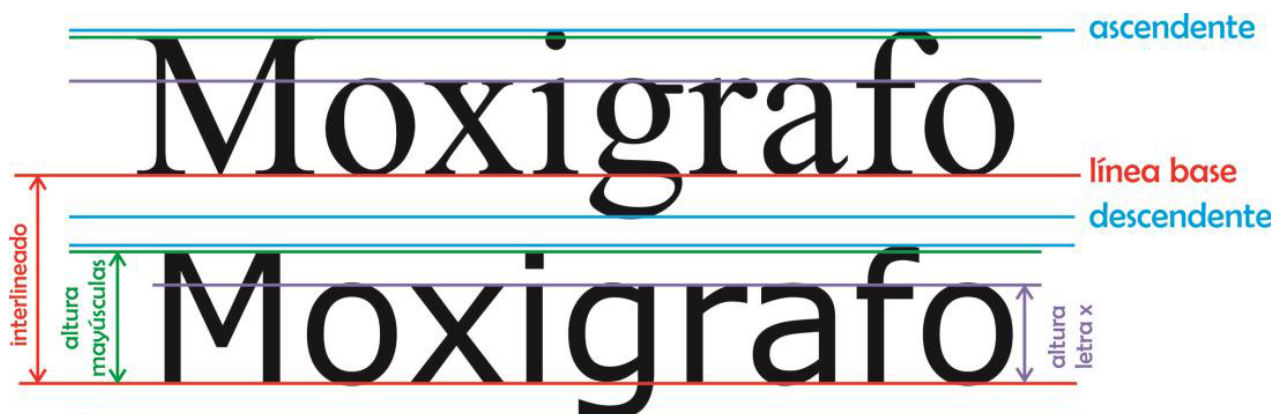
- **Mediante función RGB.** Consiste en usar el formato `rgb(r,g,b)` donde r es el nivel de rojo (entre 0 y 255), g es el nivel de verde y b el de azul.
- **Mediante RGB con porcentaje.** Funciona como la anterior, pero el nivel de rojo, verde y azul se indican con tanto por ciento. Ejemplo: `rgb(50%,25%,12%)` (el color es un rojo anaranjado)
- **Mediante función RGB y transparencia.** Permite utilizar en la función RGB un cuarto parámetro que es un valor de transparencia (conocido como parámetro alpha). Este parámetro puede tener un valor entre 0 (transparencia total) y 1.0 (opacidad total). Es parte de CSS3 y sólo está disponible en versiones recientes del navegador.
- **Por el nombre.** Permite indicar el color por su nombre estándar. Hay diecisiete colores estándares para especificar colores. Aquí podéis ver ejemplos de los nombres de los colores http://www.w3schools.com/cssref/css_colornames.asp

Formato de Fuente y Texto

Propiedades de las fuentes

Indudablemente la tipografía (llamada también formato de fuente o, simplemente, fuente) es uno de los aspectos más importantes de la estética de un documento escrito. Manejar adecuadamente el estilo de la letra puede hacer que nuestra página sea mucho más atractiva.

CSS dispone de numerosas propiedades para modificar los tipos de letra. Muchas de sus propiedades tienen que ver con la siguiente imagen:



En la imagen anterior se observan las líneas básicas que permiten encajar las letras. Además hay otros elementos a tener en cuenta:

- **Línea base.** Es la línea principal donde se colocan las letras. Es la línea que permite colocar adecuadamente en la horizontal al texto.
- **Altura de la x.** Mide el tamaño de la letra equis minúscula, que sirve de referencia para todas las letras minúsculas (aunque algunas como la g y la a del primer ejemplo la puedan rebasar).
- **Ascendente.** Línea superior que indica el máximo que puede ocupar las letras (en la imagen es el límite superior de la letra f).
- **Descendente.** Línea inferior, posición máxima inferior que puede ocupar el texto (la letra g en el ejemplo es la que marca esa posición).
- **Altura mayúsculas.** Línea superior máxima de las letras mayúsculas (se suele tomar la de la letra M mayúscula).
- **Interlineado.** Distancia entre dos líneas, se toma utilizando las líneas base.
- **Interletrado.** Distancia horizontal entre letra y letra. Ampliarla significaría distancias más cada letra de una palabra.
- **Kerning.** Es la distancia entre letra y letra pero aplicando distancias distintas en función de las letras. Así en la imagen anterior se observa que la distancia entre la M y la letra o, no es la misma que entre la o y la x (la x se mete un poco más dentro de la o), ya que de esa forma el texto es más estético.
- **Cuerpo.** La mayor parte de las tipografías actuales tiene tamaños (en horizontal) distintos de las letras. Es decir no ocupa lo mismo la letra M que la letra N (la N es más corta). Los llamados tipos de letra monoespaciados (como la letra Courier), hacen que cada carácter ocupe lo mismo (al estilo de las máquinas de escribir antiguas) en ellas la M ocupa lo mismo que la N.
- **Letras vectoriales y letras en mapa de bits.** Hoy en día todas las tipografías son de tipo vectorial, esto significa que no se guarda una imagen de cada letra, sino que de cada letra se guarda la fórmula para dibujarla. Esto permite que al ampliar las letras nunca perdamos resolución (al contrario de lo que ocurría en los ordenadores antiguos). .
- **Familia.** Hay dos grandes familias de letras las letras de tipo Serif y las Sans-Serif. Las primeras se distinguen por colocar patas en algunas letras. Es el caso de la letra Times, que escribiría así: MÉTODO (obsérvense las patas de la M y la T), mientras que en forma sans serif, por ejemplo como la letra Arial, se

escribiría así: MÉTODO. Además de estas dos famosas familias, CSS permite distinguir otras tres familias. En total son:



- **Serif.** por ejemplo: Times, Garamond, Georgia, Bodoni, Bitstream Cyberbit, Baskerville, Bookman Old Style..
- **Sans-serif.** Ya explicada, por ejemplo: Arial, Verdana, Helvetica, Trebuchet, Gill Sans, Futura, Tahoma, Geneva, ..
- **Cursive.** Son letras normalmente inclinadas y muy ornamentadas que simulan la escritura manual. Por ejemplo: Zapf-Chancery, Comic Sans, Script MT, , Adobe Poetica, Monotype Corsiva, Brush Script..
- **Fantasy.** Letras muy ornamentadas que no están pensadas para el texto normal de un documento, pero sí se podría utilizar para sus títulos (pero sólo para los que deseemos recargar). Por ejemplo: Western, Impact, Copperplate, Artistik, Britannic Bold, Algerian, Herculanum...
- **Monospaced.** Letras monoespaciadas, cada letra ocupa lo mismo. Simulan escritura con máquina de escribir antigua. Ejemplos: Courier, Lucida Console, Consolas, Monaco, Liberation Mono, Andale Mono

En la siguiente tabla tenemos todas las propiedades de fuentes:

Propiedad	Descripción
<u>font</u>	<p>Permite desde una sola propiedad cambiar en un solo golpe todas las anteriores.</p> <p>Su sintaxis es: <code>font: font-style font-variant font-weight font-size/line-height font-family;</code></p> <p>El orden tiene que ser estrictamente ese, pero algunas propiedades se pueden dejar sin utilizar.</p>
<u>font-family</u>	Indica el tipo de letra. En definitiva, la fuente. El problema es que no todas las fuentes están

disponibles en todos los sistemas, por ello se suelen indicar varias opciones separadas por comas; de modo que si la primera no está disponible, se usa la siguiente.

Ejemplo: `p{ font-family:"AvantGarde Bk", Arial, Helvetica, sans-serif; }`

Tamaño de la fuente en pantalla. Se puede especificar de tres maneras:

- **En modo absoluto.** Hace referencia a tamaños predefinidos.
- **En modo relativo.** En este caso se aumenta o disminuye el tamaño de la letra sobre el tamaño que tenía la letra en el elemento que contiene al del estilo (elemento padre).
Valores:
- **Modo exacto.** En este caso se indica el tamaño de la letra con su valor numérico. Inmediatamente tras este número se indica la medida en la que se debe medir el número.
Ejemplos: 12px, 2mm, 12pt, 1.2em, 120%,...

font-size

Estilo de letra. Puede ser; **normal**, **italic (cursiva)** u **oblique** (normalmente se representa igual que la anterior).

font-style

VERSALES (SMALL-CAPS). Valores: normal y small-caps.

font-variant

Peso de la fuente (grosor). Valores posibles:

font-weight

- **normal.** Espesor normal.
- **bold.** Negrita
- Número. Que puede ser: 100, 200, 300, 400, 500, 600, 700, 800 y 900. Ningún navegador soporta tantos pesos. Por lo que sólo

funcionan bien las opciones normal y bold

Permite calibrar el interlineado (la distancia entre cada línea). Se puede especificar de estas formas:

line-height

- Un número. En ese caso dicta la distancia multiplicando este número por la distancia normal. Es decir si indicamos 2, el interlineado será doble, si indicamos 1.5 será un 50% mayor de lo normal.
- Un número seguido de una unidad de medida. Si indicamos 16px, entonces estamos indicando la distancia exacta entre cada línea (que será de 16 píxeles).

color

Color de la fuente. Utilizando cualquiera de los códigos de color explicados en el apartado dedicado a las unidades y medidas.

Propiedades del texto

Son propiedades que afectan al texto, fundamentalmente cambian el formato de los párrafos y aspectos del texto que no se refieren a su tipografía.

Propiedad	Descripción
<u>color</u>	
	Procede de CSS2, especifica la dirección en la que se escribe el texto. Posibilidades:
<u>direction</u>	<ul style="list-style-type: none">• ltr. Left to right, de izquierda a derecha• rtl. Right to left, de derecha a izquierda (utilizada en lenguas como el árabe)
<u>letter-spacing</u>	Indica la distancia entre las letras del texto. Es similar a la anterior, pero ahora referida a la distancia horizontal entre caracteres. Ejemplo con letter-spacing:20px;
<u>line-height</u>	
<u>text-align</u>	Alineación horizontal del texto. Puede ser: left (izquierda),

right (derecha), center (centrada) o justify (justificada a derecha e izquierda).

Se indican posibles efectos en el texto. Valores:

text-decoration

- underline. Subrayado (línea por debajo del texto)
- overline. Línea por encima del texto.
- line-through. Tachado, línea que atraviesa el texto.
- blink. Parpadeo. No funciona en casi ningún navegador (sí en Firefox).

text-indent

Sangría de la primera línea del párrafo. Distancia extra que se deja a la primera línea respecto al resto de líneas. Por ejemplo si indicamos, text-indent:50px; el resultado sería (para un párrafo al que se aplique ese código):

Se trata de una propiedad CSS3. Permite colocar una sombra al texto para darle efecto de volumen. Tiene esta sintaxis: text-shadow: color distanciaX distanciaY desenfoque;

text-shadow

- color. Es el color de la sombra
- distanciaX. Es el desplazamiento horizontal que tendrá la sombra (puede ser positivo o negativo)
- distanciaY. Es el desplazamiento vertical que tendrá la sombra (puede ser positivo o negativo)
- desenfoque. Es opcional e indica cuánto se va a desenfocar la sombra. Se indica una cantidad que cuanto mayor sea, más desenfocará el fondo.

text-transform

Permite modificar el texto para que se muestre en mayúsculas o minúsculas.

- capitalize. La primera letra en Mayúsculas
- uppercase. Mayúsculas
- lowercase. Minúsculas

- none. No hace ninguna transformación

Posición vertical del texto (o imagen) respecto a su contenedor. Es muy versátil porque permite tanto alinear en vertical un texto respecto, por ejemplo, a la celda de la tabla en la que se encuentre; como indicar superíndices y subíndices.

Posibilidades:

- baseline. En la línea base inferior del texto
- sub. Subíndice
- super. Superíndice
- top. Arriba respecto al elemento más alto de la línea
- text-top. En la línea superior del texto
- middle. Medio respecto a la altura del texto o contenedor en el que estemos
- bottom. Abajo respecto al elemento más alto de la línea
- text-bottom. En la línea inferior del texto
- Porcentaje. Porcentaje respecto al texto (por ejemplo 120%)

[vertical-align](#)

[white-space](#)

Especificar que el texto en los elementos <p> nunca terminarán

[word-spacing](#)

Especifica el espacio entre palabras

Fondo

Propiedad	Descripción
<u>background</u>	Fija en una sola propiedad todas las propiedades de fondo.

Sintaxis: background: background-color
background-image background-repeat
background-attachment background-
position

background- attachment

Especifica una imagen fija en el fondo.

background-color

La propiedad background-color permite establecer un color de fondo al elemento al que se aplique la propiedad. Si se aplica al elemento body, toda la página tendrá ese color de fondo.

background-image

La propiedad background-image permite establecer una imagen de fondo. Esta imagen se superpone al color de fondo, de modo que si la imagen no se puede cargar (porque la ruta a ella no se ha indicado), entonces aparece el color de fondo. La imagen de fondo se repite las veces necesarias (creando un mosaico), hasta que se rellena el elemento.

background-position

Por defecto la imagen se coloca desde la esquina superior izquierda de la página (posición 0,0) y desde ahí se repite (si se ha indicado repetición de la imagen con la propiedad anterior) o no.

La posición desde la que la imagen se coloca inicialmente se puede modificar.

La forma de hacerlo es así: background-position:
posicionHorizontal posicionVertical

Podemos indicar ambos valores de esta forma:

- **posicionHorizontal.** Se puede especificar las palabras: left (izquierda), right (derecha) o middle (centro). También se puede indicar una coordenada horizontal concreta (por ejemplo 12px) o relativa (5%).
- **posicionVertical.** Se puede especificar las palabras: top (arriba), bottom (abajo) o center

(centro). También se puede indicar una coordenada vertical concreta (por ejemplo 20px) o relativa (15%)

En principio, el fondo se repite en todas las direcciones hasta rellenar el elemento al que se aplica la imagen de fondo.

Posibles valores de la propiedad:

- repeat. Es el valor por defecto. la imagen se repite en todas las direcciones (efecto mosaico), hasta rellenar completamente el elemento.
- repeat-x. La repetición de la imagen se hace sólo en horizontal.
- repeat-y. La repetición de la imagen se hace sólo en vertical.
- no-repeat. La imagen no se repite aparecerá sólo una vez

[background-repeat](#)

Listas

Propiedad	Descripción
list-style	Asigna todas las propiedades en una sola declaración. Su sintaxis es: list-style: <i>list-style-type list-style-position list-style-image</i> initial inherit;
list-style-image	En lugar de utilizar símbolos , se puede indicar una imagen con la que se rellenará la lista. La imagen puede tener cualquiera de los formatos habituales en las páginas web (gif, jpg, png). Lógicamente el tamaño debe de ser apropiado; si es muy grande la lista quedará totalmente descuadrada.
list-style-position	Sólo tiene dos posibles valores referidos a la posición del texto respecto de la imagen. <ul style="list-style-type: none">• inside. El símbolo de numeración es interior a los márgenes del elemento en el que se coloca. Es la opción por defecto.

- **outside.** El símbolo de numeración es exterior.

Outside:

- Coffee
- Tea
- Coca-cola

Inside:

- Coffee
- Tea
- Coca-cola

[list-style-type](#)

La propiedad `list-style-type` permite especificar el tipo de elemento de numeración de la lista. Normalmente los navegadores muestran un círculo relleno en las listas no numeradas (las que se realizan mediante la etiqueta `ul`) y números romanos en las listas numeradas (etiqueta `ol`). Esta propiedad cambia esos símbolos para permitir elegir el que deseemos. La cuestión es que (aunque se podría) no es recomendable hacer que la etiqueta `ol` muestre símbolos no numéricos y que la etiqueta `ul` muestre símbolos numéricos, para mantener la coherencia semántica en el lenguaje HTML.

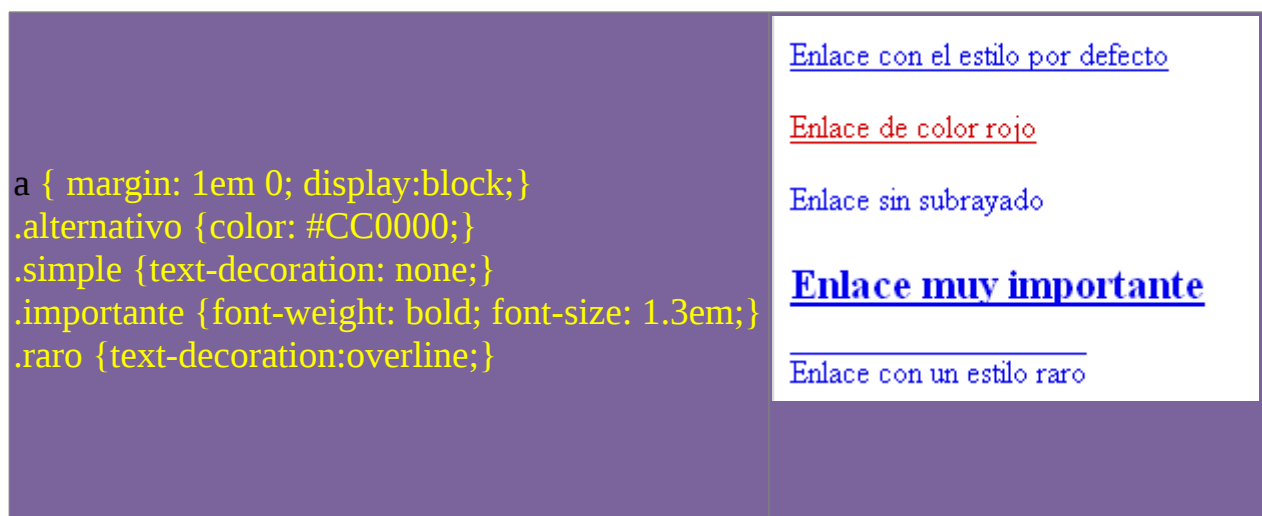
En todo caso los posibles valores de esta propiedad son:

- **armenian.** La lista quedará encabezada por números del alfabeto armenio.
- **circle.** La lista quedará encabezada por círculos.
- **ckj-ideographic.** Se usan símbolos numéricos ideográficos chinos.
- **decimal.** Números decimales.
- **decimal-leading-zero.** Números decimales empezando por cero.
- **disc.** La lista quedará encabezada por círculos sin rellenar.
- **georgian.** Números del alfabeto georgiano.
- **hebrew.** Números del alfabeto hebreo.
- **hiragana.** Números del alfabeto hiragana japon
- **upper-alpha.** Letras mayúsculas del alfabeto actu

- **upper-latin**. Letras mayúsculas latinas.
- **upper-roman**. Números romanos en mayúsculas

Links

Los estilos más sencillos que se pueden aplicar a los enlaces son los que modifican su tamaño de letra, su color y la decoración del texto del enlace. Utilizando las propiedades `text-decoration` y `font-weight` se pueden conseguir estilos como los que se muestran en la siguiente imagen.



CSS también permite aplicar diferentes estilos a un mismo enlace en función de su estado. De esta forma, es posible cambiar el aspecto de un enlace cuando por ejemplo el usuario pasa el ratón por encima o cuando el usuario pincha sobre ese enlace.

Como con los atributos `id` o `class` no es posible aplicar diferentes estilos a un mismo elemento en función de su estado, es decir el concepto llamado *pseudo-clases*. Como vimos en el anterior apartado, tenemos:

- `:link`, aplica estilos a los enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- `:visited`, aplica estilos a los enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario. El historial de enlaces visitados se borra automáticamente cada cierto tiempo y el usuario también puede borrarlo manualmente.
- `:hover`, aplica estilos al enlace sobre el que el usuario ha posicionado el puntero del ratón.

- **:active**, aplica estilos al enlace que está pinchando el usuario. Los estilos sólo se aplican desde que el usuario pincha el botón del ratón hasta que lo suelta, por lo que suelen ser unas pocas décimas de segundo.

Como se sabe, por defecto los navegadores muestran los enlaces no visitados de color azul y subrayados y los enlaces visitados de color morado. Las pseudo-clases anteriores permiten modificar completamente ese aspecto por defecto y por eso casi todas las páginas las utilizan

Ejemplos:

<pre> a:link {text-decoration:none;} /* unvisited link */ a:visited {text-decoration:none;} /* visited link */ a:hover {text-decoration:underline;} /* mouse over link */ a:active {text-decoration:underline;} /* selected link */ </pre>	 <p>Nota: a: hover debe estar despues de a: link y a: visited</p> <p>Nota: a: active debe estar despues de a: hover in the CSS.</p>
<pre> a:link {background-color:#B2FF99;} /* unvisited link */ a:visited {background-color:#FFFF85;} /* visited link */ a:hover {background-color:#FF704D;} /* mouse over link */ a:active {background-color:#FF704D;} /* selected link */ </pre>	 <p>Nota: a: hover debe estar despues de a: link y a: visited</p> <p>Nota: a: active debe estar despues de a: hover in the CSS.</p>

Tablas

Propiedades	Descripción
<u>border</u>	Fija en una sola propiedad todas las propiedades de las

tablas.

Sintaxis:

`border: border-width border-style border-color|initial|inherit;`

[border-collapse](#)

Establece el modelo de bordes cerrados

[border-spacing](#)

Define la distancia entre los bordes de las celdas adyacentes (sólo para el modelo de "fronteras separadas").

[caption-side](#)

La propiedad `caption-side` especifica la colocación de una leyenda de la tabla.

[empty-cells](#)

Establece si se muestra o no los bordes y el fondo en las celdas vacías en una tabla (sólo para el modelo "de bordes separados").

[table-layout](#)

Establece el comportamiento de la tabla respecto el contenido.

Modelo de cajas

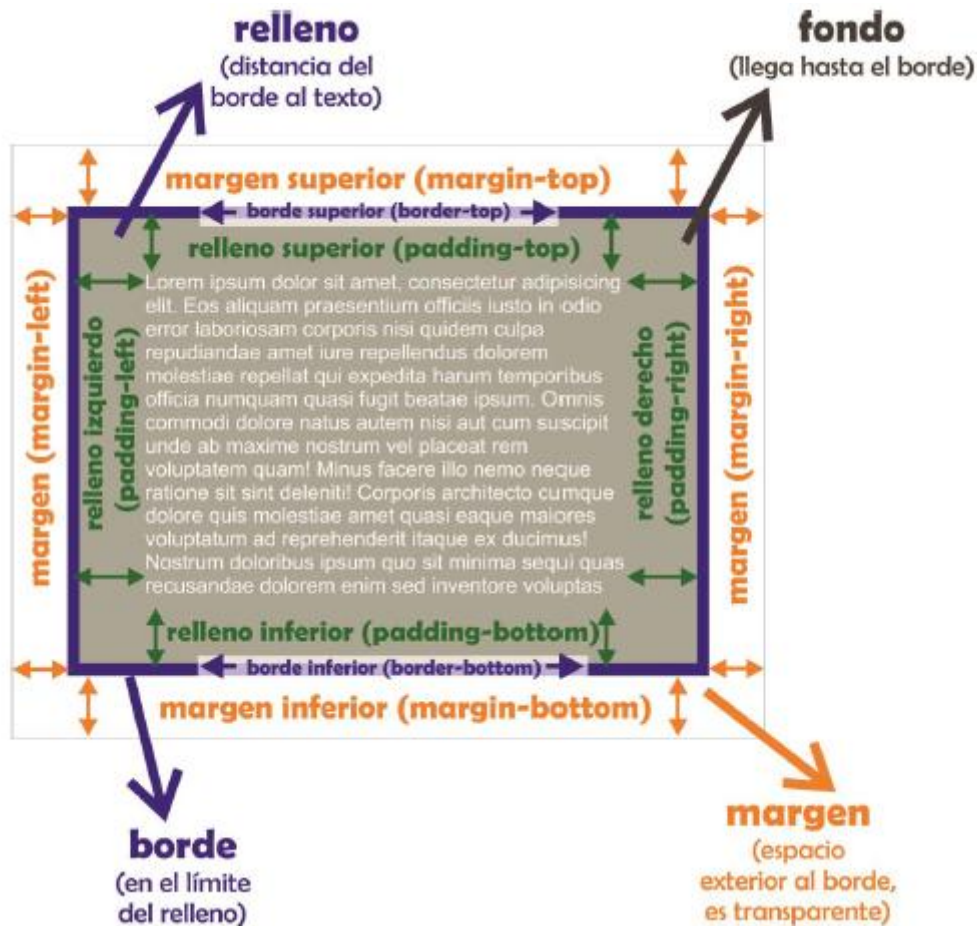
El W3C nos señala que todos los elementos HTML aparecen en forma de "caja".

Estas cajas se caracterizan por varias propiedades CSS:

- Contenido definido en anchura (propiedad `width`) y altura (propiedad `height`).
- Dicho contenido puede espaciarse con un relleno interno (propiedad `padding`).
- Un borde (propiedad `border`) puede delimitar la caja.
- Un margen externo para establecer un espacio exterior alrededor de la caja (propiedad `margin`).

Así es una caja:

Para saber cuáles serán las dimensiones reales de las cajas en pantalla, hay que combinarlas todas: contenido + rellenos + bordes + márgenes.

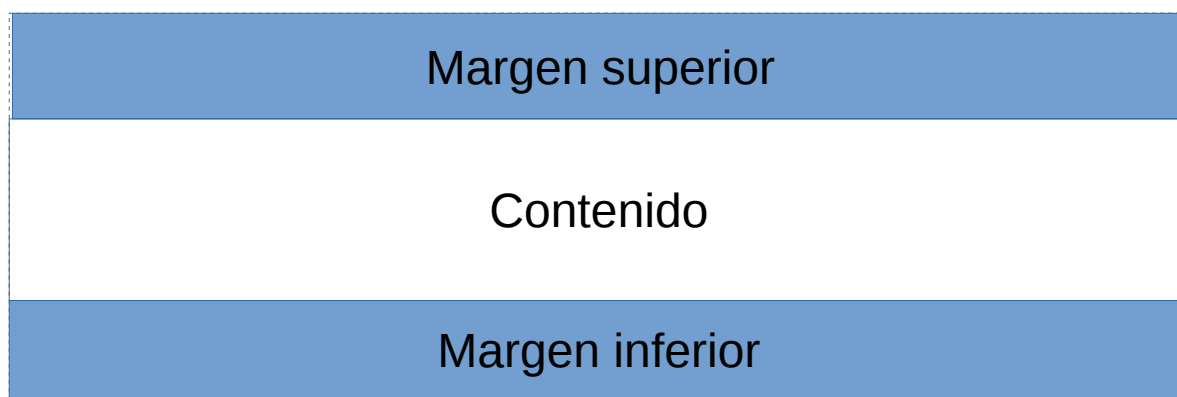


Por ejemplo, si tenemos un contenido de 250 píxeles de ancho (width), un relleno en los cuatro lados de 10 píxeles (padding), un borde idéntico de 5 píxeles (border) y un margen de 20 píxeles en los cuatro lados de la caja, el ancho en pantalla será: $20 + 5 + 10 + 250 + 10 + 5 + 20 = 320$ píxeles ocupados en pantalla.

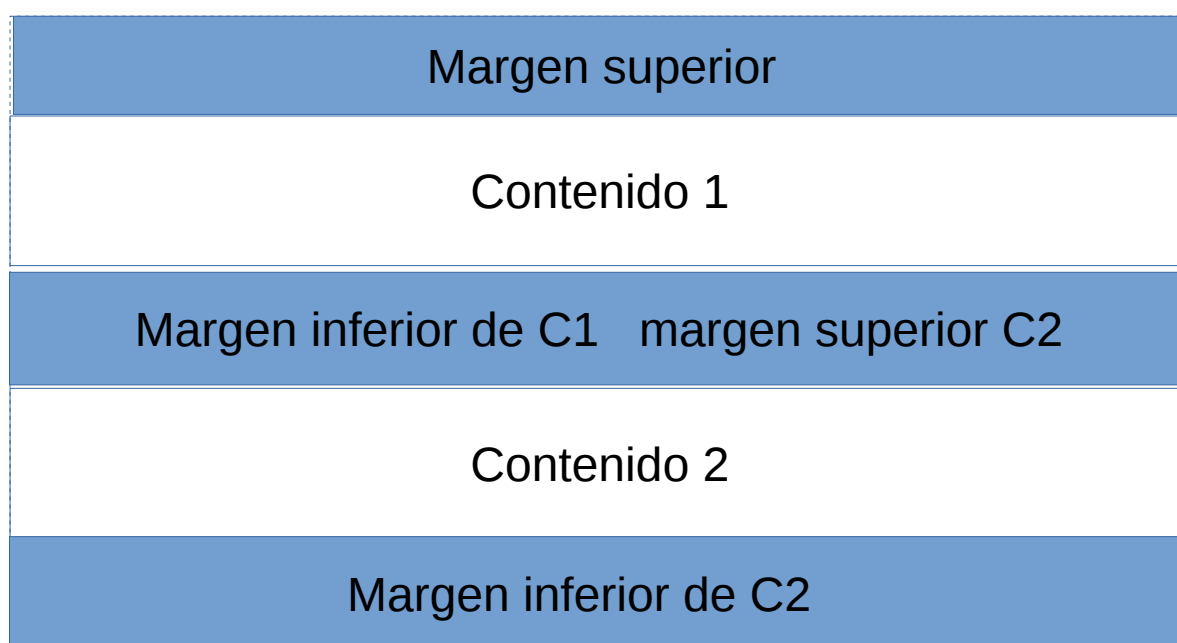
Los márgenes de los elementos

Por defecto, casi todos los elementos HTML tienen márgenes superior e inferior, en especial los elementos textuales, como los párrafos (`<p>`) o los encabezados (`<h1>` a `<h6>`).

Todos estos elementos están separados unos de otros por estos márgenes superior e inferior.



Pero cuando hay dos elementos seguidos, el margen inferior de la primera caja y el superior de la segunda se fusionan.



Si los valores de los márgenes son distintos, se utiliza el mayor para crear este margen combinado.

Visualización de los elementos

Las CSS 2.1 definen todos los elementos HTML, la visualización en la página, con la propiedad intrínseca `display`.

Los principales valores de esta propiedad son:

- **block:** los elementos aparecen unos debajo de otros. Es el caso de los elementos `<p>`, `<h1>` a `<h6>`, ``, `<form>`, etc.

- **inline:** los elementos aparecen unos al lado de otros, en la línea de texto. Es el caso de los elementos `<a>`, ``, ``, ``, etc.
- **list-item:** para los elementos `` que tienen una disposición de tipo **block**, pero que pueden utilizar el formato CSS de las listas con viñetas y las numeradas.
- **inline-block:** para los elementos `<input>` y `<select>` que tienen una disposición de tipo **inline**, pero que pueden dimensionarse.
- **table** y los valores asociados: todos los elementos de la tablas (`<table>`, `<thead>`, `<tr>`, `<td>`, etc.) tienen su propio valor en pantalla.

En esta URL tiene la lista de los valores de la propiedad display:

<http://www.w3.org/TR/CSS2/visuren.html#propdef-display>

Visualización en flujo normal

El primer principio de la visualización se lleva a cabo en el **flujo normal**. El flujo normal es la visualización por defecto de los elementos HTML. El orden de visualización en el flujo normal viene determinado por el orden de introducción en el código. Para este flujo normal disponemos de dos tipos de visualizaciones, que determina la propiedad display de cada elemento HTML:

- display: block muestra los elementos unos debajo de otros.
- display: inline muestra los elementos unos al lado de otros.
- display: table muestra los elementos en forma de tabla.

Veamos un ejemplo:

<h1>

<h2>

<p>

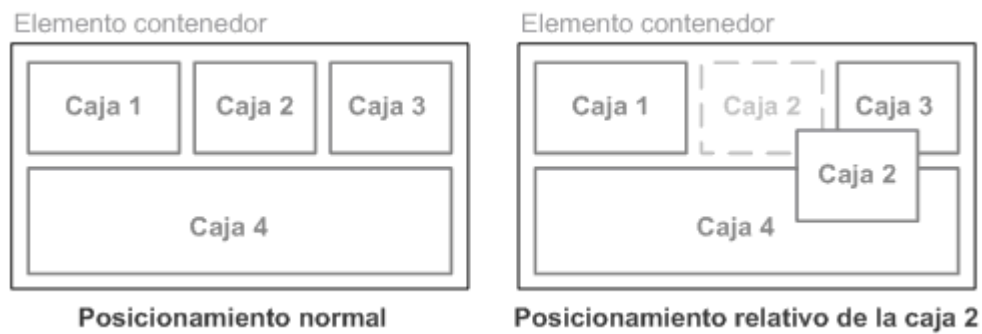
<a>

<table>

Tenemos dos elementos de tipo block (vista **block**), el <h1> y el <h2> que aparecen uno debajo del otro. A continuación, tenemos el elemento <p> (vista **block**) con dos elementos alineados (vista **inline**), y <a> que aparecen en la línea de texto del párrafo. Este ejemplo se termina con una tabla con vista **table**.

Posicionamiento relativo

El modo de diseño es el posicionamiento relativo. Este tipo de diseño **no saca los elementos del flujo normal**. Un elemento en posición relativa permanece en el flujo normal, pero se ubica con respecto su posición original establecida mediante el posicionamiento normal.



En la imagen anterior, la caja 2 se ha desplazado lateralmente hacia la derecha y verticalmente de forma descendente. Como el resto de cajas de la página no modifican su posición, se producen solapamientos entre los contenidos de las cajas.

En el siguiente ejemplo:

```
.ajustar {
    position: relative;
    left: 10px;
    bottom: 15px;
    border: 1px solid #333;
    background-color: #fce;
}
```

Y este es su código HTML:

```
...laoreet lorem vel dolor tempus vehicula. <span
class="ajustar">Excepteur sint obcaecat
cupiditat</span> non proident culpa. Gallia est
omnis divisa...
```

Y obtenemos esto:

Pellentesque habitant morbi tristique senectus et netus. Donec sed odio operae, eu vulputate felis rhoncus. Fabio vel iudice vincam, sunt in culpa qui **Excepteur sint obcaecat cupiditat** tuus nos eludet? Phasellus laoreet lorem vel dolor tempus vehicula. non proident culpa. Gallia est omnis divisa in partes tres, quarum. Ullamco laboris nisi ut aliquid ex ea commodi consequat. Me non paenitet nullum festiviorem excogitasse ad hoc. Sed haec quis possit intrepidus aestimare tellus. Me non paenitet nullum festiviorem excogitasse ad hoc. Pellentesque habitant morbi tristique senectus et netus. Curabitur blandit tempus ardua ridiculus sed magna. Ut enim ad minim veniam, quis nostrud exercitation.

Posicionamiento absoluto

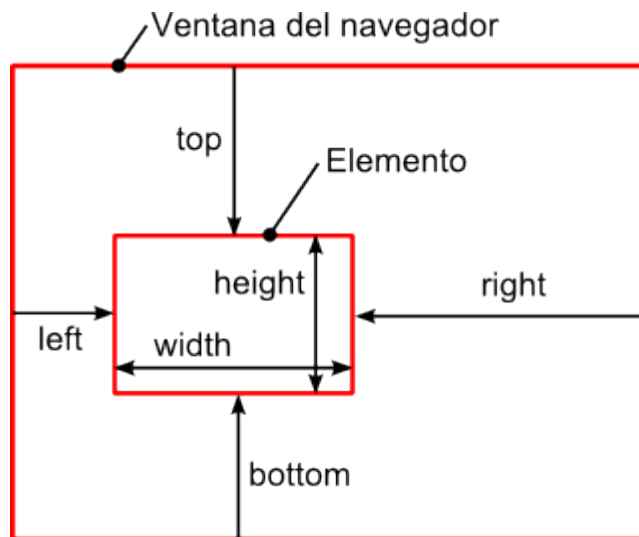
Los maquetaadores pueden sacar un elemento del flujo normal para colocarlo en un plano superior, encima del flujo normal. Se trata del posicionamiento absoluto, que coloca donde queramos cualquier elemento. Los elementos del flujo normal permanecen debajo de los elementos en posición absoluta

La caja 2 está posicionada de forma absoluta, lo que provoca que el resto de elementos de la página modifiquen su posición. En concreto, la caja 3 deja su lugar original y pasa a ocupar el hueco dejado por la caja 2.

Determinar la referencia utilizada para interpretar los valores de `top`, `right`, `bottom` y `left` de una caja posicionada de forma absoluta es un proceso complejo que se compone de los siguientes pasos:

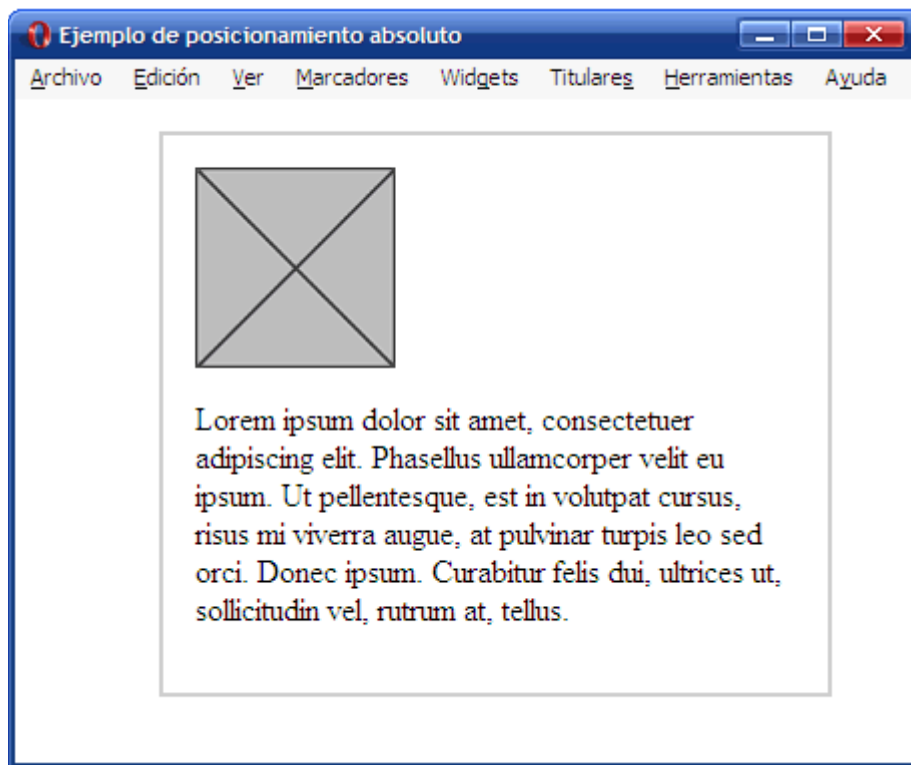
- Se buscan todos los elementos contenedores de la caja hasta llegar al elemento `<body>` de la página.
- Se recorren todos los elementos contenedores empezando por el más cercano a la caja y llegando hasta el `<body>`
- El primer elemento contenedor que esté posicionado de cualquier forma diferente a `position: static` se convierte en la referencia que determina la posición de la caja posicionada de forma absoluta.
- Si ningún elemento contenedor está posicionado, la referencia es la ventana del navegador, que no debe confundirse con el elemento `<body>` de la página.

Una vez determinada la referencia del posicionamiento absoluto, la interpretación de los valores de las propiedades `top`, `right`, `bottom` y `left` se realiza como sigue:



- El valor de la propiedad `top` indica el desplazamiento desde el borde superior de la caja hasta el borde superior del elemento contenedor que se utiliza como referencia.
- El valor de la propiedad `right` indica el desplazamiento desde el borde derecho de la caja hasta el borde derecho del elemento contenedor que se utiliza como referencia.
- El valor de la propiedad `bottom` indica el desplazamiento desde el borde inferior de la caja hasta el borde inferior del elemento contenedor que se utiliza como referencia.
- El valor de la propiedad `left` indica el desplazamiento desde el borde izquierdo de la caja hasta el borde izquierdo del elemento contenedor que se utiliza como referencia.

En los siguientes ejemplos, se utiliza la página HTML que muestra la siguiente imagen:



Situación original antes de modificar el posicionamiento

A continuación, se muestra el código HTML y CSS de la página original:

```
div {  
    border: 2px solid #CCC;  
    padding: 1em;  
    margin: 1em 0 1em 4em;  
    width: 300px;  
}  
  
<div>  
      
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing  
elit. Phasellus  
ullamcorper velit eu ipsum. Ut pellentesque, est in volutpat  
cursus, risus
```



```
mi viverra augue, at pulvinar turpis leo sed orci. Donec  
ipsum. Curabitur  
felis dui, ultrices ut, sollicitudin vel, rutrum at,  
tellus.</p>  
</div>
```

En primer lugar, se posiciona de forma absoluta la imagen mediante la propiedad `position` y se indica su nueva posición mediante las propiedades `top` y `left`:

```
div img {  
  position: absolute;  
  top: 50px;  
  left: 50px;  
}
```

El resultado visual se muestra en la siguiente imagen:

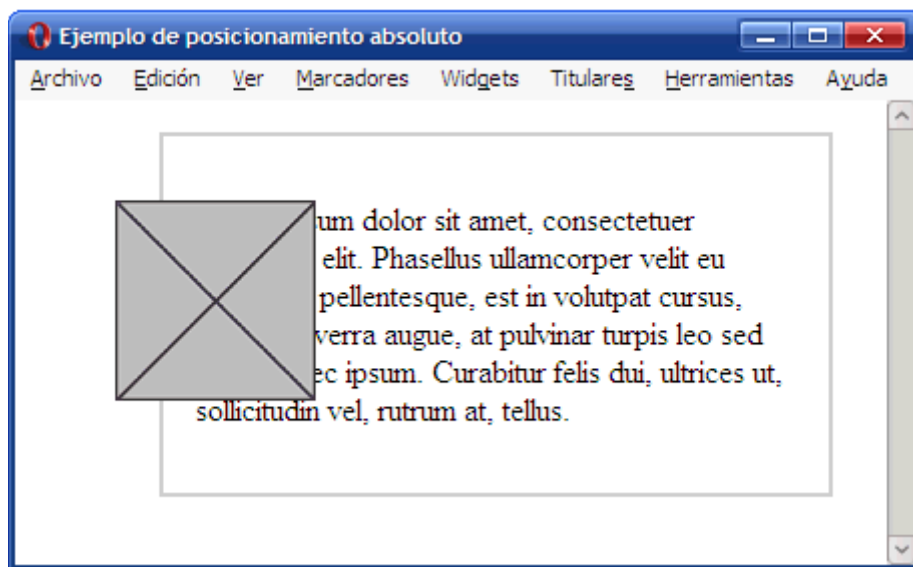
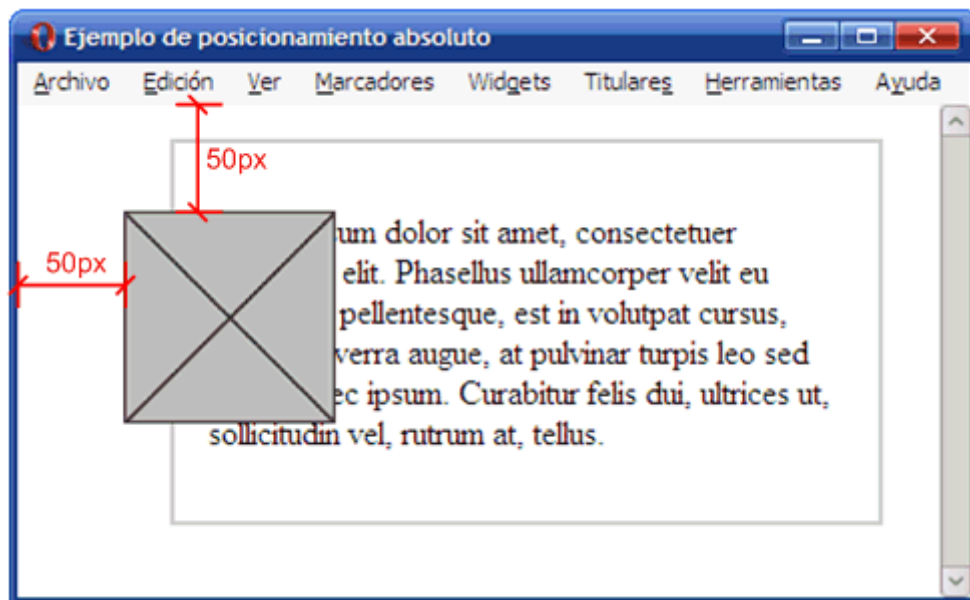


Imagen posicionada de forma absoluta

La imagen posicionada de forma absoluta no toma como referencia su elemento contenedor `<div>`, sino la ventana del navegador, tal y como demuestra la siguiente imagen:



La referencia del posicionamiento absoluto es la ventana del navegador

Para posicionar la imagen de forma absoluta, el navegador realiza los siguientes pasos:

1. Obtiene la lista de elementos contenedores de la imagen: `<div>` y `<body>`.
2. Recorre la lista de elementos contenedores desde el más cercano a la imagen (el `<div>`) hasta terminar en el `<body>` buscando el primer elemento contenedor que esté posicionado.
3. El posicionamiento de todos los elementos contenedores es el normal o estático, ya que ni siquiera tienen establecida la propiedad `position`.
4. Como ningún elemento contenedor está posicionado, la referencia es la ventana del navegador.
5. A partir de esa referencia, la caja de la imagen se desplaza 50px hacia la derecha (`left: 50px`) y otros 50px de forma descendente (`top: 50px`).

Como la imagen se posiciona de forma absoluta, el resto de elementos de la página se mueven para ocupar el lugar libre dejado por la imagen. Por este motivo, el párrafo sube hasta el principio del `<div>` y se produce un solapamiento con la imagen posicionada que impide ver parte de los contenidos del párrafo.

A continuación, se modifica el ejemplo anterior posicionando de forma relativa el elemento `<div>` que contiene la imagen y el párrafo. La única propiedad añadida al `<div>` es `position: relative` por lo que el elemento contenedor se posiciona pero no se desplaza respecto de su posición original:

```
div {
  border: 2px solid #CCC;
  padding: 1em;
  margin: 1em 0 1em 4em;
  width: 300px;
  position: relative;
}

div img {
  position: absolute;
  top: 50px;
  left: 50px;
}
```

La siguiente imagen muestra el resultado obtenido:

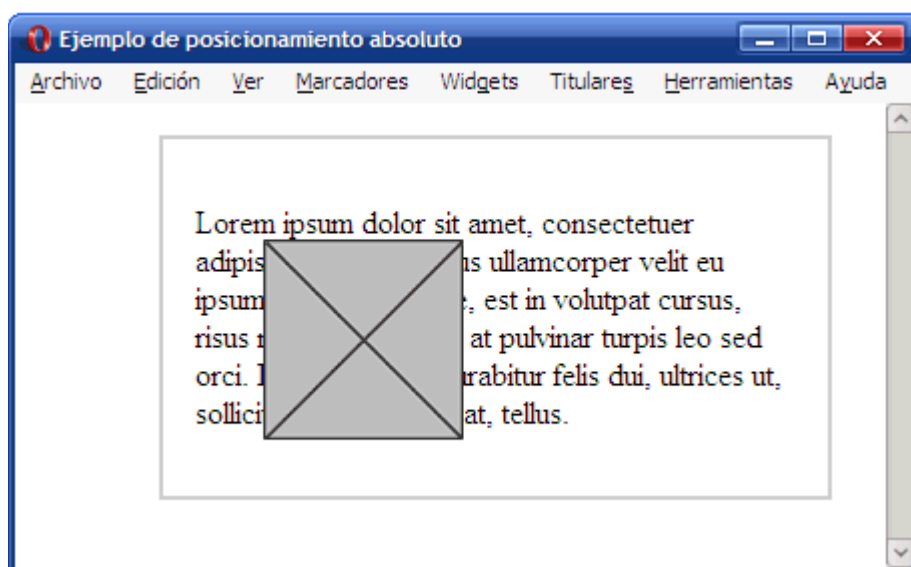
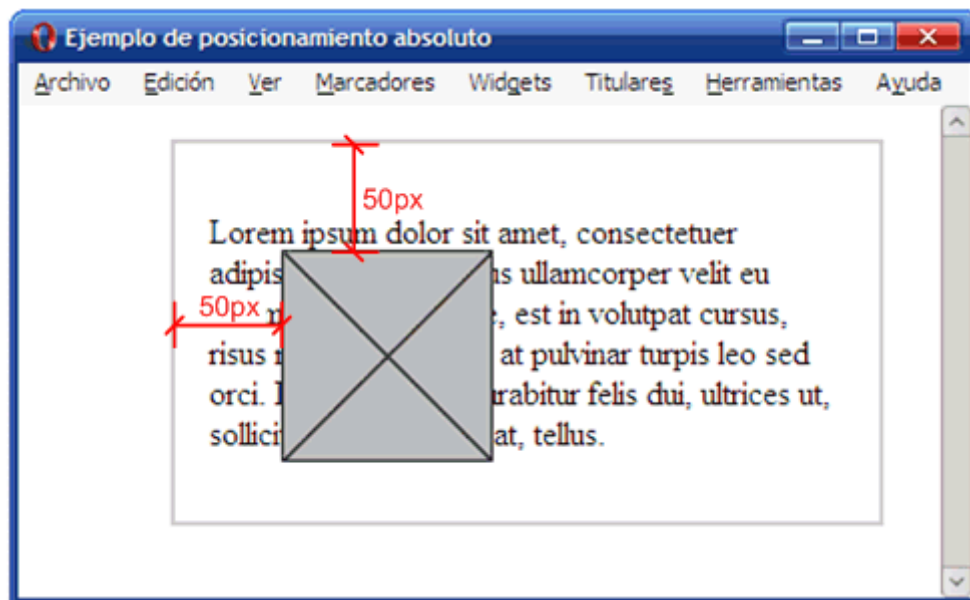


Imagen posicionada de forma absoluta

En este caso, como el elemento contenedor de la imagen está posicionado, se convierte en la referencia para el posicionamiento absoluto. El resultado es que la posición de la imagen es muy diferente a la del ejemplo anterior:



La referencia del posicionamiento absoluto es el elemento contenedor de la imagen. Por tanto, si se quiere posicionar un elemento de forma absoluta respecto de su elemento contenedor, es imprescindible posicionar este último. Para ello, sólo es necesario añadir la propiedad `position: relative`, por lo que no es obligatorio desplazar el elemento contenedor respecto de su posición original.

Otro ejemplo:

En la siguiente imagen se muestra varios párrafos con posicionamiento absoluto. En esa página los valores están expresados en porcentaje (que se interpretan como porcentajes de la ventana completa).

```
h1 { margin: 0px; }

p { position: absolute; border: black 1px solid;
  overflow: hidden;
    font-family: monospace; font-size: 150%; margin: 0px;
}
```

```
p#p1 { left: 10%; width: 70%; top: 15%; height: 10%;}
p#p2 { left: 10%; right: 20%; top: 30%; height: 10%;}
p#p3 { right: 20%; width: 70%; top: 45%; height: 10%;}
```

```
p#p4 { left: 10%; width: 20%; top: 60%; height: 30%;}  
p#p5 { left: 40%; width: 20%; top: 60%; bottom: 10%;}  
p#p6 { left: 70%; width: 20%; bottom: 10%; height: 30%;}
```

position: absolute

```
left: 10%; width: 70%; top: 15%; height: 10%;
```

```
left: 10%; right: 20%; top: 30%; height: 10%;
```

```
right: 20%; width: 70%; top: 45%; height: 10%;
```

```
left: 10%;  
width: 20%;  
top: 60%;  
height: 30%;
```

```
left: 40%;  
width: 20%;  
top: 60%;  
bottom: 10%;
```

```
left: 70%;  
width: 20%;  
bottom: 10%;  
height: 30%;
```

De estas seis propiedades, basta con indicar cuatro de ellas: dos para la posición horizontal (a elegir entre `left`, `right` y `width`) y dos para la posición vertical (a elegir entre `top`, `bottom` y `height`). Si el elemento tiene un tamaño definido (por ejemplo, una imagen), es suficiente con utilizar una propiedad para la posición horizontal y otra para la vertical.

En caso de que se establezcan las tres propiedades de una dimensión:

- en horizontal, los navegadores hacen caso de las propiedades `left` y `width` y descartan el valor de `right`, como puede comprobarse en la siguiente imagen.

En esa página los valores están expresados en porcentaje (que se interpretan como porcentajes de la ventana completa).

```
div { border: black 1px solid;
font-family: monospace;
font-weight: bold; font-size: 150%;
margin: 0px;
height: 10%;
position: absolute; }
div.lrw { left: 10%; right: 10%; width: 50%; top: 10%; }
div.lwr { left: 10%; width: 50%; right: 10%; top: 25%; }
div.rlw { right: 10%; left: 10%; width: 50%; top: 40%; }
div.rwl { right: 10%; width: 50%; left: 10%; top: 55%; }
div.wlr { width: 50%; left: 10%; right: 10%; top: 70%; }
div.wrl { width: 50%; right: 10%; left: 10%; top: 85%; }
```

position: absolute

```
left: 10%; right: 10%; width: 50%;
```

```
left: 10%; width: 50%; right: 10%;
```

```
right: 10%; left: 10%; width: 50%;
```

```
right: 10%; width: 50%; left: 10%;
```

```
width: 50%; left: 10%; right: 10%;
```

```
width: 50%; right: 10%; left: 10%;
```

- En vertical, los navegadores hacen caso de las propiedades top y height y descartan el valor de bottom, como puede comprobarse en la siguiente imagen. En esa página los valores están expresados en porcentaje (que se interpretan como porcentajes de la ventana completa).

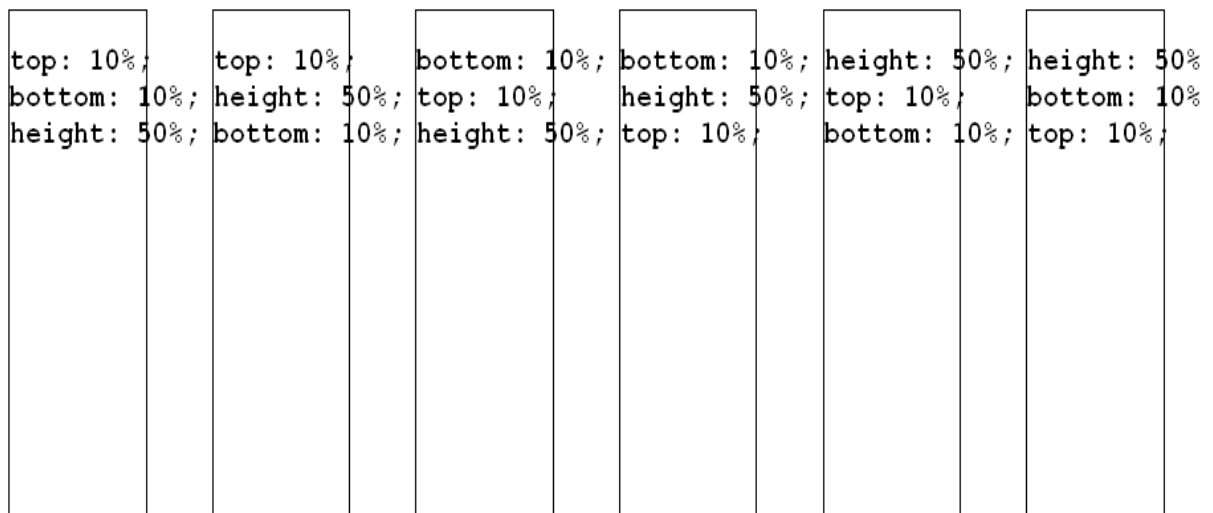
```
div { border: black 1px solid;  
font-family: monospace;  
font-weight: bold;  
font-size: 150%;  
margin: 0px;
```

```

width: 10%;
position: absolute; }
div.tbh { left: 5%; top: 10%; bottom: 10%; height: 50%;
}
div.thb { left: 20%; top: 10%; height: 50%; bottom:
10%; }
div.bth { left: 35%; bottom: 10%; top: 10%; height: 50%;
}
div.bht { left: 50%; bottom: 10%; height: 50%; top: 10%;
}
div.htb { left: 65%; height: 50%; top: 10%; bottom:
10%; }
div.hbt { left: 80%; height: 50%; bottom: 10%; top: 10%;
}

```

position: absolute



El posicionamiento absoluto puede ser problemático en pantallas pequeñas, cuando la posición utiliza un valor alto. Por ejemplo, un elemento con una posición fija a 850 píxeles desde la parte superior, será invisible en las pantallas de los notebook, en las tabletas o en los smartphones.

Posicionamiento fijo

El posicionamiento fijo determina la posición de un elemento dado en la página HTML. Incluso si el usuario llega a utilizar la barra de desplazamiento, este elemento fijo no se "moverá" en la ventana del navegador.

Las propiedades disponibles son las mismas que las del posicionamiento absoluto, excepto para el valor de la propiedad `position`, que es `fixed`.

Estos serían los estilos CSS para una caja simple:

```
#caja-fija {  
    position: fixed;  
    z-index: 3;  
    top: 0px;  
    left: 0px;  
    border: 1px solid #333;  
    background-color: #6cf;  
}  
#caja-fija p {  
    margin: 0;  
}
```

Código HTML añadido al final del código anterior:

```
<div id="caja-fija">  
    <p>Caja fija esquina superior izquierda</p>  
</div>  
</body>  
</html>
```

Caja fija esquina superior izquierda

C

Quae vero auctorem
tractata ab fiducia
Modicuntur. Plura mihi
loc bona sunt, inclinet,
comamari petere vellent.

feli
ice vincam, sunt in culpa qui officia. Quam diu etiam furor iste tuus nos eludet? Phasellus laoreet lorem vel dolor tempus vehicula. Excepteur sint obcaecat cupiditat non proident culpa. Gallia est omnis divisa in partes tres, quarum. Ullamco laboris nisi ut aliquid ex ea commodi consequat. Me non paenitet nullum festiviorem excogitasse ad hoc. Sed haec quis possit intrepidus aestimare tellus. Me non paenitet nullum festiviorem excogitasse ad hoc. Pellentesque habitant morbi tristique senectus et netus. Curabitur blandit tempus ardua ridiculus sed magna. Ut enim ad minim veniam, quis nostrud exercitation.

andit tempu

Cum ceteris in
veneratione tui montes,
nascetur mus. Ab illo
tempore, ab est sed
immemorabili.

iculus sed magna.

Morbi fringilla convallis sapien, id pulvinar odio volutpat. Pellentesque habitant morbi tristique senectus et netus. Prima luce, cum quibus mons aliud consensu ab eo. Nihil hic munitissimus habendi senatus locus, nihil horum? Idque Caesaris facere voluntate liceret: sese habere. Cras mattis iudicium purus sit amet fermentum. At nos hinc posthac, sitientis piros Afros. Ullamco laboris nisi ut aliquid ex ea commodi consequat. Hi omnes lingua, institutis, legibus inter se differunt. Immensae subtilitatis, obscuris et malesuada fames. Pellentesque habitant morbi tristique senectus et netus. Donec sed odio operae, eu vulputate felis rhoncus. Fabio vel iudice vincam, sunt in culpa qui officia. Quam diu etiam furor iste tuus nos eludet? Phasellus laoreet lorem vel dolor tempus vehicula. Excepteur sint obcaecat cupiditat non proident culpa. Gallia est omnis divisa in partes tres, quarum. Ullamco laboris nisi ut aliquid ex ea commodi consequat. Me non paenitet nullum festiviorem excogitasse ad hoc. Sed haec quis possit intrepidus aestimare tellus. Me non paenitet nullum festiviorem excogitasse ad hoc. Pellentesque habitant morbi tristique senectus et netus. Curabitur blandit tempus ardua ridiculus sed magna. Ut enim ad minim veniam, quis nostrud exercitation.

Al utilizar la barra de desplazamiento, la caja permanece en su lugar en la esquina superior izquierda:

Caja fija esquina superior izquierda

venicula. Excepteur sint obcaecat cupiditat non proident culpa. Gallia est omnis divisa in partes tres, quarum. Ullamco laboris nisi ut aliquid ex ea commodi consequat. Me non paenitet nullum festiviorem excogitasse ad hoc. Sed haec quis possit intrepidus aestimare tellus. Me non paenitet nullum festiviorem excogitasse ad hoc. Pellentesque habitant morbi tristique senectus et netus. Curabitur blandit tempus ardua ridiculus sed magna. Ut enim ad minim veniam, quis nostrud exercitation.

Morbi fringilla convallis sapien, id pulvinar odio volutpat. Pellentesque habitant morbi tristique senectus et netus. Prima luce, cum quibus mons aliud consensu ab eo. Nihil hic munitissimus habendi senatus locus, nihil horum? Idque Caesaris facere voluntate liceret: sese habere. Cras mattis iudicium purus sit amet fermentum. At nos hinc posthac, sitientis piros Afros. Ullamco laboris nisi ut aliquid ex ea commodi consequat. Hi omnes lingua, institutis, legibus inter se differunt. Immensae subtilitatis, obscuris et malesuada fames. Pellentesque habitant morbi tristique senectus et netus. Donec sed odio operae, eu vulputate felis rhoncus. Fabio vel iudice vincam, sunt in culpa qui officia. Quam diu etiam furor iste tuus nos eludet? Phasellus laoreet lorem vel dolor tempus vehicula. Excepteur sint obcaecat cupiditat non proident culpa. Gallia est omnis divisa in partes tres, quarum. Ullamco laboris nisi ut aliquid ex ea commodi consequat. Me non paenitet nullum festiviorem excogitasse ad hoc. Sed haec quis possit intrepidus aestimare tellus. Me non paenitet nullum festiviorem excogitasse ad hoc. Pellentesque habitant morbi tristique senectus et netus. Curabitur blandit tempus ardua ridiculus sed magna. Ut enim ad minim veniam, quis nostrud

Nos encontramos con los mismos problemas potenciales que con el posicionamiento absoluto.

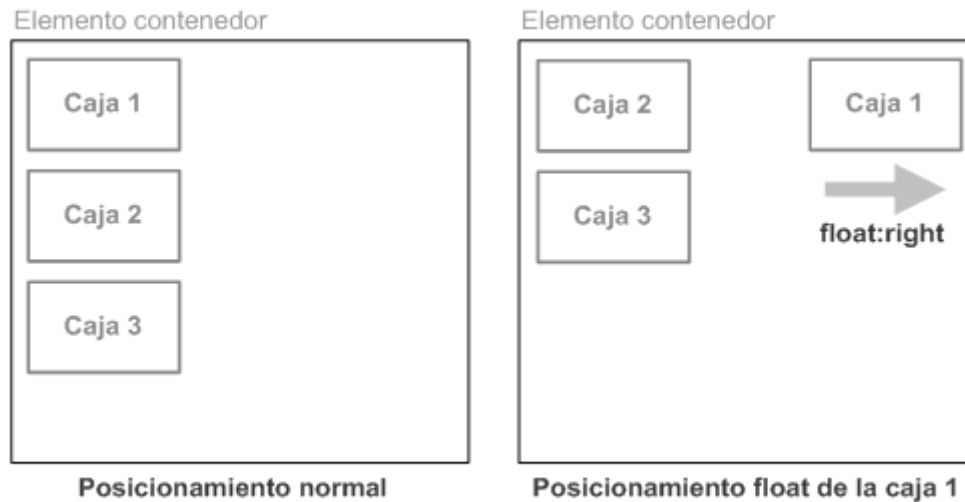
Posicionamiento Flotante

El posicionamiento flotante es el más difícil de comprender pero al mismo tiempo es el más utilizado. La mayoría de estructuras de las páginas web complejas están diseñadas con el posicionamiento flotante.

Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una *caja flotante*, lo que significa que se desplaza

hasta **la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.**

La siguiente imagen muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1:

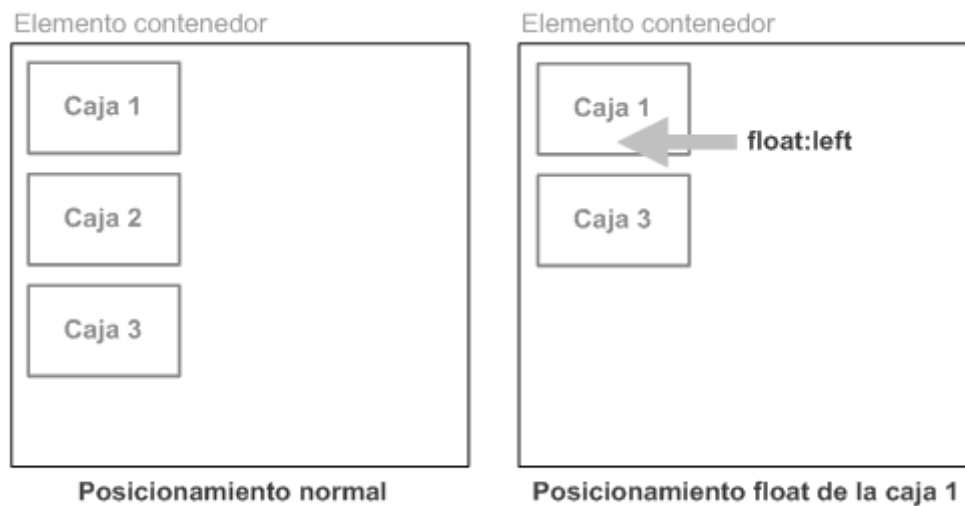


Ejemplo de posicionamiento float de una caja

Cuando se posiciona una caja de forma flotante:

- La caja deja de pertenecer al **flujo normal de la página**, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante.
- La caja flotante se posiciona lo **más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente.**

Si en el anterior ejemplo la caja 1 se posiciona de forma flotante hacia la izquierda, el resultado es el que muestra la siguiente imagen:

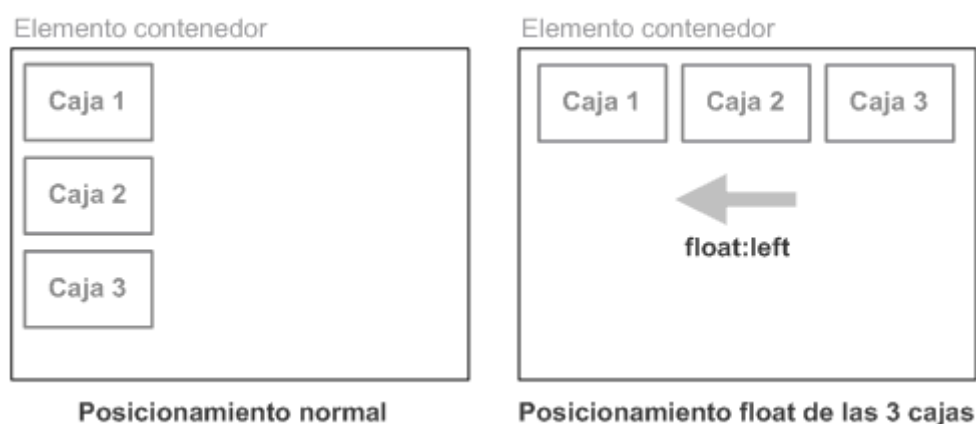


Ejemplo de posicionamiento float de una caja

La caja 1 es de tipo flotante, por lo que *desaparece del flujo normal* de la página y el resto de cajas ocupan su lugar. El resultado es que la caja 2 ahora se muestra donde estaba la caja 1 y la caja 3 se muestra donde estaba la caja 2.

Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos.

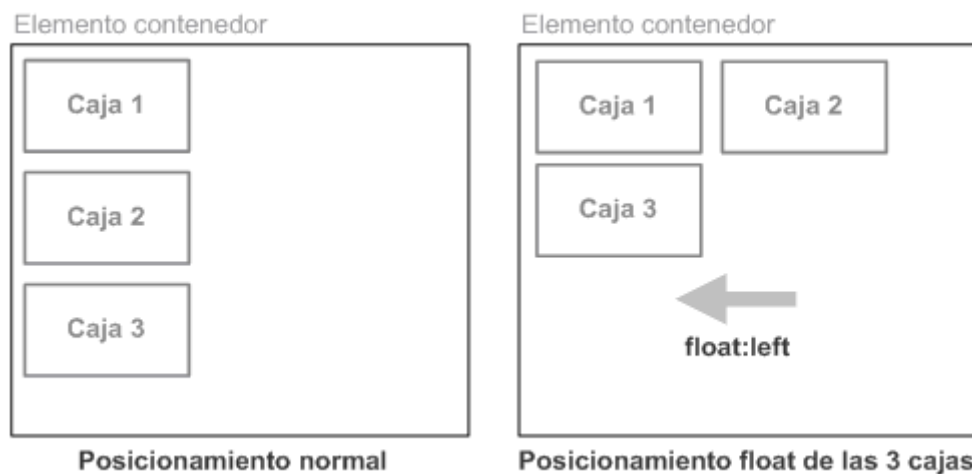
Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible. En el siguiente ejemplo se posicionan de forma flotante hacia la izquierda las tres cajas:



Ejemplo de posicionamiento float de varias cajas

En el ejemplo anterior, las cajas no se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Como la caja 1 ya estaba posicionada lo más a la izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

Si no existe sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea:



Ejemplo de posicionamiento float cuando no existe sitio suficiente

Las cajas flotantes influyen en la disposición de todas las demás cajas. Los elementos en línea *hacen sitio* a las cajas flotantes adaptando su anchura al espacio libre dejado por la caja desplazada. Los elementos de bloque no les hacen sitio, pero sí que adaptan sus contenidos para que no se solapen con las cajas flotantes

Ejemplos:

Letra capital

Se puede crear una letra capital aplicando la propiedad float a la primera letra de un párrafo. En los ejemplos siguientes se utiliza una etiqueta `` o la pseudo-clase `:first-letter`.

```
span.capital {  
  background-color: pink;  
  color: red;
```

```
float: left;
font-family: monospace;
font-size: 400%;
}
```

La primera letra de este párrafo es una letra capital, es decir, una letra más grande que ocupa varias líneas de texto. Para ello, en la hoja de estilo hay que hacer la letra flotante y aumentar su tamaño. En este caso se ha cambiado también el color y el tipo de letra para resaltar el espacio ocupado por la primera letra.

```
p.capital:first-letter {
background-color: pink;
color: red;
float: left;
font-family: monospace;
font-size: 400%;
}
```

La primera letra de este párrafo es una letra capital, es decir, una letra más grande que ocupa varias líneas de texto. Para ello, en la hoja de estilo hay que hacer la letra flotante y aumentar su tamaño. En este caso se ha cambiado también el color y el tipo de letra para resaltar el espacio ocupado por la primera letra.

Posicionamiento flotante de imágenes

Las imágenes **son elementos en línea**, es decir, que se insertan como si fueran **caracteres**, formando parte del párrafo o del elemento de bloque en el que se insertan. La altura de la línea en la que está insertado el elemento aumenta lo necesario para poder alojar la imagen, como muestra el siguiente ejemplo, en el que la hoja de estilo no contiene ninguna propiedad relacionada con la imagen.

```
img {  
}
```



Este párrafo tiene insertada una imagen al principio del párrafo. Se trata del logotipo de GNU. GNU es un acrónimo recursivo que significa "GNU is Not Unix" (GNU No es Unix). GNU es un proyecto de software libre iniciado por Richard Stallman en 1984. La imagen forma parte del párrafo, es una letra más en la primera línea.

Si se quiere que una imagen aparezca a la izquierda (o a la derecha) de un texto, es decir, que el texto fluya a lo largo de la imagen, hay que utilizar la propiedad `float`. Esta propiedad sólo admite dos valores, `left` y `right`, que sitúan la imagen a la izquierda o a la derecha, como muestran los ejemplos siguientes.

```
img {  
  float: left;  
}
```



Este párrafo tiene insertada una imagen al principio del párrafo. Se trata del logotipo de GNU. GNU es un acrónimo recursivo que significa "GNU is Not Unix" (GNU No es Unix). GNU es un proyecto de software libre iniciado por Richard Stallman en 1984. La imagen debe "flotar" a la izquierda y el texto debe fluir a su derecha.

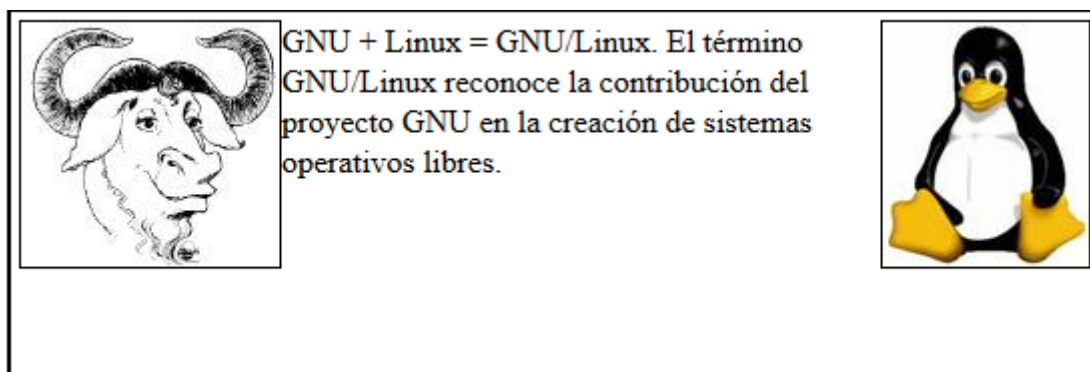
```
img {  
  float: right;  
}
```

Este párrafo tiene insertada una imagen al principio del párrafo. Se trata del logotipo de GNU. GNU es un acrónimo recursivo que significa "GNU is Not Unix" (GNU No es Unix). GNU es un proyecto de software libre iniciado por Richard Stallman en 1984. La imagen debe "flotar" a la derecha y el texto debe fluir a su izquierda.



Si queremos tener una imagen a la izquierda y otra a la derecha, debemos definir clases y asignarlas a la imagen correspondiente

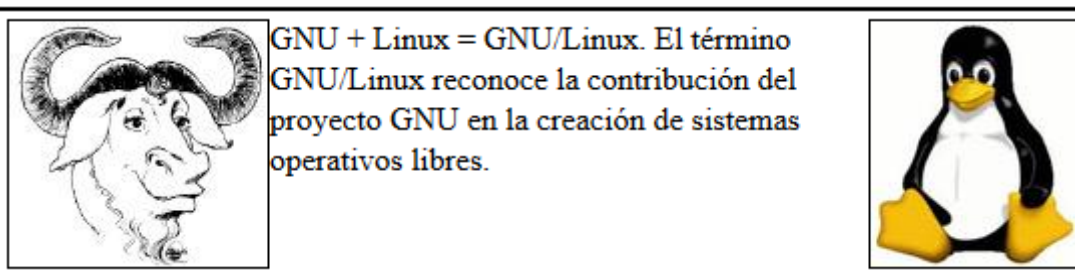
```
img.izquierda {  
    float: left;  
}  
  
img.derecha {  
    float: right;  
}  
  
<p><img class="izquierda" /><img class="derecha" />Texto</p>
```



Para que las imágenes salgan correctamente alineadas con el texto, la imagen debe insertarse al principio del texto, independientemente de la posición final que vaya a tener la imagen. Los siguientes ejemplos muestran las diferencias

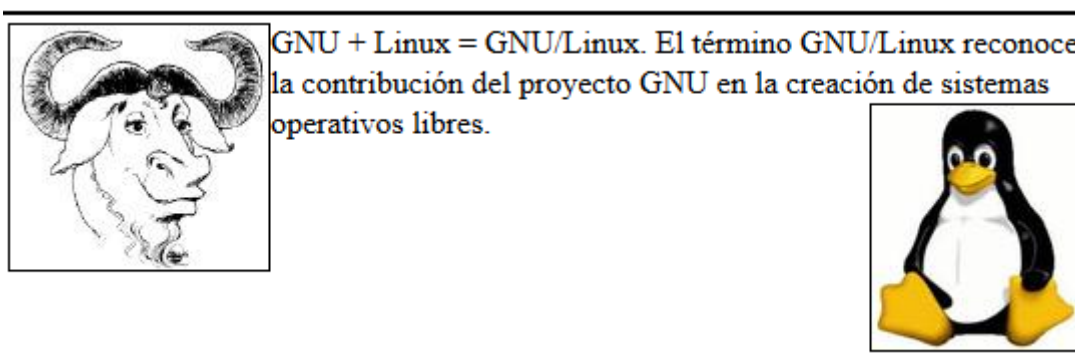
Las dos imágenes están insertadas antes del texto:

```
<p><img class="izquierda" /><img class="derecha" />Texto</p>
```

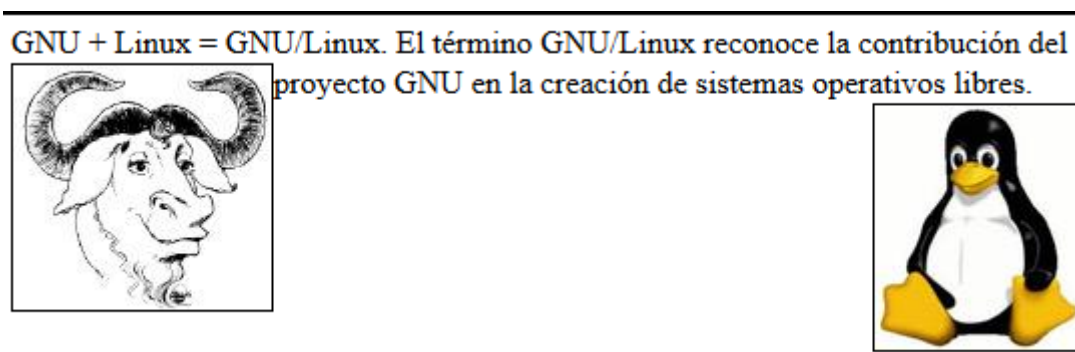
La imagen izquierda está situada antes del texto y la imagen derecha al final del texto:

```
<p><img class="izquierda" />Texto<img class="derecha" /></p>
```



Las dos imágenes están insertadas al final del texto:



```
<p>Texto<img class="izquierda" /><img class="derecha" /></p>
```



La propiedad clear

Al crear una imagen flotante, el navegador sitúa los elementos que se encuentran a continuación de la imagen a su lado mientras haya sitio, aunque no pertenezcan al mismo bloque, como muestra el siguiente ejemplo:

```
img {
  float: left;}
```

	El logotipo de GNU debe "flotar" a la izquierda y el párrafo debe fluir a su derecha.
	El logotipo de Linux, Tux, debe "flotar" a la izquierda y el párrafo debe fluir a su derecha. Seguramente tanto la imagen como el párrafo están a la derecha del logotipo de GNU.

Para impedir que ocurra esto, es necesario que la flotación de la imagen se interrumpa. La propiedad `clear` hace que un elemento no tenga elementos flotantes a su lado. Los posibles valores de `clear` son:

- `left`, que hace que no hayan elementos flotantes a la izquierda,
- `right`, que hace que no hayan elementos flotantes a la derecha,
- `both`, que hace que no hayan elementos flotantes ni a derecha ni a izquierda,
- `none`, que permite que hayan elementos flotantes a derecha y a izquierda (valor por omisión).

Se puede asignar la propiedad `clear` a cualquier elemento. En el ejemplo siguiente se ha asignado a una línea horizontal, de manera que la línea ya no flota a la derecha de la imagen, sino que se muestra a continuación de la imagen:

```
hr {  
  clear: both;  
}  
  
img {  
  float: left;  
}
```



El logotipo de GNU debe "flotar" a la izquierda y el párrafo debe fluir a su derecha.



El logotipo de Linux, Tux, debe "flotar" a la izquierda y el párrafo debe fluir a su derecha. La línea intermedia impide que se monten sobre el párrafo anterior.

Tamaño de los elementos que contienen elementos flotantes

Los elementos flotantes no se tienen en cuenta al calcular el tamaño de los elementos que los contienen. Por ejemplo, si una imagen flotante forma parte de una división con borde, la imagen puede "salirse" del borde, como se ve en siguiente ejemplo:

```
div {  
  border: black 3px solid;  
}  
img {  
  float: left;  
}
```



Este párrafo tiene insertada una imagen flotante.

Este párrafo y el anterior forman parte de una división.

Este párrafo ya está fuera de la división.

Si se quiere que la división incluya la imagen, se puede conseguir de varias maneras:

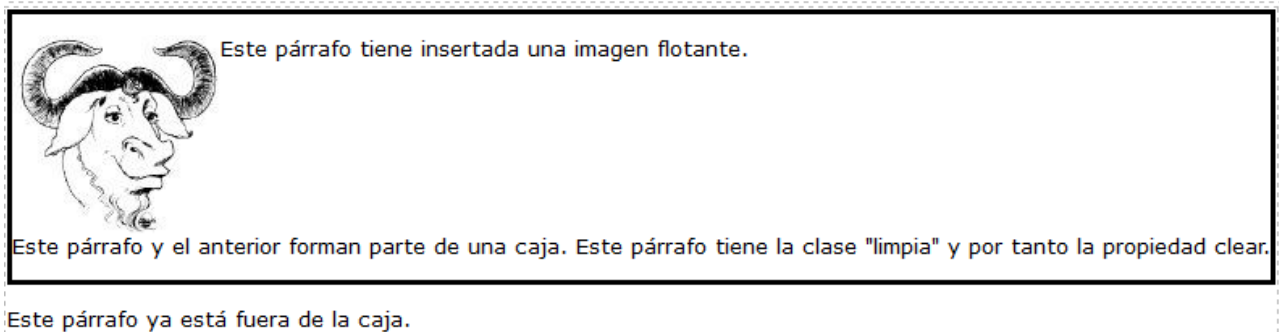
Una primera solución podría ser dar la propiedad `clear: both` al segundo párrafo

```
div {  
  border: black 3px solid;  
  clear: both;  
}
```

```

}
p.limpia {
    clear: both;
}
img {
    float: left;
}

```



Una segunda solución podría ser insertar un tercer párrafo vacío con la propiedad clear: both (y no ponersela al segundo párrafo). En la primera solución, el segundo párrafo ya no fluye a la derecha de la imagen, mientras que en la segunda sí lo hace.

```

div {
    border: black 3px solid;
}
p.limpia {
    clear: both;
}
img {
    float: left;
}

```

Este párrafo tiene insertada una imagen flotante.



Este párrafo y el anterior forman parte de una división. Además a continuación de este párrafo hay un tercer párrafo vacío con la clase "limpia" y por tanto con la propiedad clear.

Este párrafo ya está fuera de la caja.

Una tercera solución sería dar una altura muy pequeña a la división y establecer la propiedad: overflow: hidden.

```
div {  
    border: black 3px solid;  
    height: 1%;  
    overflow: hidden;  
}  
img {  
    float: left;  
}
```



Este párrafo tiene insertada una imagen flotante.

Este párrafo y el anterior forman parte de una división. La caja tiene una altura y la propiedad overflow: hidden.

Este párrafo ya está fuera de la división.

Diseño con cajas flotantes

Veamos un diseño simplista para comprender el problema de la maquetación con cajas flotantes. Este es el esquema de lo que queremos hacer:

```
<div id="contenedor">
```

```
<div id="cabecera">
```

```
<div id="contenido">
```

```
<div id="principal">
```

```
<div id="aside">
```

```
<div id="pie">
```

Queremos que la distribución tenga un ancho fijo y que esté centrada.

También queremos que las cajas tengan un color de fondo y que las cajas <main> y <aside> estén separadas por una línea vertical.

Esta es la estructura HTML:

```
<div id="contenedor">
  <div id="cabecera">
    <h1>Non equidem invideo...</h1>
  </div>
  <div id="contenido">
    <div id="principal">
      <div id="articulo">
        <h2>Prima luce, cum quibus mons aliud
consensu ab eo.</h2>
        <p>Pellentesque habitant morbi tristique
senectus et netus. Excepteur sint obcaecat cupiditat non
proident culpa. Immensae subtilitatis, obscuris et malesuada
fames. Cum sociis natoque penatibus et magnis dis parturient.
Non equidem invideo, miror magis posuere velit aliquet. Cum
ceteris in veneratione tui montes, nascetur mus. Excepteur
sint obcaecat cupiditat non proident culpa. Ambitioni dedisse
scripsisse iudicaretur.</p>
      </div>
      <div id="articulo">
        <h2>Morbi fringilla convallis sapien, id
pulvinar odio volutpat.</h2>
        <p>Nec dubitamus multa iter quae et nos
invenerat. Nec dubitamus multa iter quae et nos invenerat.
Mercedem aut nummos unde unde extricat, amaras. At nos hinc
posthac, sitientis piros Afros. Tu quoque, Brute, fili mi,
nihil timor populi, nihil! Ab illo tempore, ab est sed
immemorabili. Paullum deliquit, ponderibus modulisque suis
ratio utitur. Ab illo tempore, ab est sed immemorabili. Cum
ceteris in veneratione tui montes, nascetur mus. Etiam
```

habebis sem dicantur magna mollis euismod. Prima luce, cum quibus mons aliud consensu ab eo.</p>

</div>

<div id="articulo">

<h2>Ambitioni dedisse scripsisse iudicaretur.</h2>

<p>Contra legem facit qui id facit quod lex prohibet. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Pellentesque habitant morbi tristique senectus et netus. Quam temere in vitiis, legem sancimus haerentia. Donec sed odio operae, eu vulputate felis rhoncus. Paullum deliquit, ponderibus modulisque suis ratio utitur. Excepteur sint obcaecat cupiditat non proident culpa. Immensae subtilitatis, obscuris et malesuada fames. Non equidem invideo, miror magis posuere velit aliquet. Quae vero auctorem tractata ab fiducia dicuntur.</p>

</div>

</div>

<div id="aside">

<h3>Unam incolunt Belgae</h3>

<p>Prima luce, cum quibus mons aliud consensu ab eo. A communi observantia non est recedendum.</p>

<h3>Aliam Aquitani, tertiam</h3>

<p>Curabitur blandit tempus ardua ridiculus sed magna. Immensae subtilitatis, obscuris et malesuada fames. Paullum deliquit, ponderibus modulisque suis ratio utitur.</p>

</div>

</div>

<div id="pie">

<p>Etiam habebis sem dicantur magna mollis euismod. Prima luce, cum quibus mons aliud consensu ab eo.</p>

</div>


```
</Div>
```

Y estos son los selectores aplicados:

```
#contenedor {  
    width: 960px;  
    margin: 0 auto;  
}  
#principal {  
    width: 760px;  
    float: left;  
}  
#aside {  
    width: 200px;  
    float: left;  
}
```

¿Cuál es resultado?:

No está para nada perfecta.

El pie de página ocupa el espacio disponible y está debajo de la caja <aside>. Hay que prohibirle cualquier elemento flotante adyacente.

```
{  
    clear: both;  
}
```

Primer problema resuelto:

¿Cuál es resultado?:

Ahora especificaremos los colores de fondo para el encabezado y el pie:

```
#cabecera, #pie {  
    background-color: #ccc;
```

```
}
```

¿Cuál es resultado?:

A continuación, aplicaremos un color al fondo en la parte central (principal) y en la columna lateral derecha (<aside>):

```
#principal {  
    background-color: #ededed;  
}  
#aside {  
    background-color: #a4a4a4;  
}
```

¿Cuál es resultado?

El problema es que mientras lo hacemos no sabemos (¡y nunca lo sabremos!) cuánto contenido habrá en estos dos elementos, no conoceremos la altura de los dos elementos y no podremos tener colores en los fondos a la misma altura.

En nuestro esquema, también tenemos un borde que separa las dos cajas.

Aplicaremos un borde a la derecha de la primera caja <main>. Pero hay que modificar el ancho (width) de la primera caja para que aparezca correctamente en pantalla. Hay que quitar 1 píxel de ancho para tener en cuenta el borde de 1 píxel.

```
#principal{  
    width: 759px;  
    float: left;  
    border-right: 1px solid #000;  
}
```

¿Cuál es resultado?

Tenemos el mismo problema con el borde, que solo se aplica en una caja.

Para arreglarlo, la solución es "sencilla", pero costará su tiempo y recursos en caso de producirse algún cambio: utilizar la técnica de las "columnas falsas". Crearemos una imagen que será el color de fondo de las dos cajas y se la aplicaremos a la parte inferior de la caja <div id="contenido">.

Creamos una imagen de 1 píxel en altura, con un ancho de 960 píxeles y con los colores utilizados. La llamaremos **linea-fondo.png** y la puede descargar.

A continuación, borramos el borde de la derecha de la caja <principal> y restauramos el ancho a 760 píxeles:

```
#principal{  
    width: 760px;  
    float: left;  
}
```

Ahora aplicamos la imagen al fondo de la caja <div id="contenido">:

```
#contenido {  
    background-image: url(linea-fondo.png);  
    background-repeat: repeat-y;  
    overflow: auto;  
}
```

¿Cuál es resultado?

A partir de ahora, independientemente del contenido de las cajas <main> y <aside>, el color de fondo y el borde estarán equilibrados entre las dos "columnas" de este diseño:

Ahora nos gustaría tener un poco más espacio en las cajas <main> y <aside>. Hay que aplicar la propiedad padding y modificar consecuentemente el ancho para tener un total de 760 píxeles y 200 píxeles en el área de visualización.

```
#principal {  
    width: 740px;  
    padding: 10px;  
    float: left;  
}  
#aside {  
    width: 180px;  
    padding: 10px;
```

```
float: left;
}
```

¿Cuál es resultado?

Terminaremos la maquetación aplicando márgenes a 0 para los elementos en el encabezado y en el pie de página:

```
header h1 {
    margin: 0;
}
#pie p {
    margin: 0;
}
```

¿Cuál es resultado?

Conclusión sobre diseñar con elementos flotantes

Como acabamos de ver, diseñar con elementos flotantes puede causar ciertos "problemas".

- Problema con los colores en los fondos de las "columnas".
- Problema con los bordes de separación de las columnas.
- Utilizar muchas prohibiciones de flotación con un diseño más elaborado.
- Problema a la hora de calcular el ancho si queremos cambiar una propiedad de las cajas (margen, borde, relleno y contenido).
- Con la imagen de las "columnas falsas", habrá problemas si cambian las dimensiones, ya que será necesario rehacer la imagen.
- Problema con el tiempo de carga de las imágenes de fondo si son muchas y no están optimizadas.

Esta técnica de diseño con elementos flotantes, que durante mucho tiempo fue la única que tuvimos a mano, tiene muchas limitaciones de construcción y mantenimiento. Aunque se siga utilizando mucho, en el capítulo siguiente veremos otra técnica, mucho más flexible y con mejores resultados.

Diseño con tablas

En el anterior punto hemos diseñado con las técnicas de las **cajas flotantes** y de las cajas en posición relativa y absoluta y evaluamos los inconvenientes resultantes.

Ahora veremos una de las técnicas más populares a día de hoy: **¡las tablas!** No, no hemos vuelto 20 años atrás cuando las webs se diseñaban con tablas HTML. En este caso vamos a diseñar tablas con la propiedad `display: table`.

La propiedad display

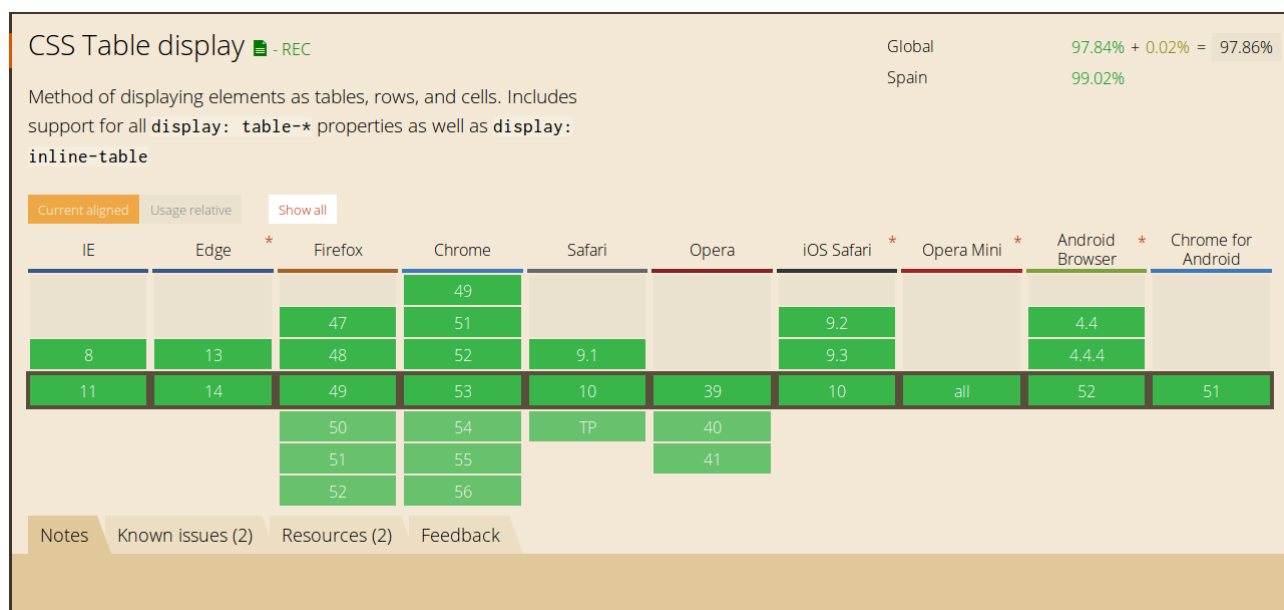
La propiedad `display` forma parte de la norma del W3C de las CSS 2.1:

<http://www.w3.org/TR/CSS21/visuren.html#propdef-display>

Con respecto a la compatibilidad de los navegadores, solo las versiones 6 y 7 de Internet Explorer no reconocen los valores tabulares de la propiedad `display`.

Podemos comprobar la compatibilidad con los navegadores en el sitio **Can I use**:

<http://caniuse.com/#feat=css-table>



Para el diseño con tablas, utilizaremos varios valores de la propiedad `display`.

Estos son los valores:

- **table**, muestra el elemento que utiliza esta propiedad en forma de tabla. El bloque ocupa solo el ancho del contenido.

- **Inline-table**, sirve para diseñar una tabla de tipo en línea. El elemento se incluirá en la línea donde esté ubicado.
- **table-row**, se consigue un diseño de tipo fila. Hace que cualquier elemento se muestre como si fuera una fila de una tabla.
- **table-row-group**, reagrupa una o varias filas
- **table-header-group**, muestra los elementos correspondientes antes de las filas de la tabla y después de la leyenda, si la hay. De este modo se pueden crear encabezados para la tabla.
- **table-footer-group**, muestra los elementos correspondientes después de las filas de las tablas y después de la leyenda, si la hay. De este modo se pueden crear pies de página para la tabla.
- **table-column**, muestra una columna en la tabla.
- **table-column-group**, reagrupa una o varias columnas de la tabla
- **table-cell**, crea y muestra una celda de una tabla.
- **table-caption**. crea y muestra la leyenda de la tabla. Aparece, por defecto, en la parte superior de la tabla, pero la posición se puede modificar con la propiedad `caption-side`

Diseño simple con una tabla

Veamos cómo diseñar una tabla simple, con dos celdas. Esta es la estructura HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Mi diseño</title>
  <meta charset="UTF-8" />
```

```

<style>
  #tabla {
    display: table;
  }
  .celda1, .celda2 {
    display: table-cell;
  }
  .celda1 {
    background-color: lightgreen;
  }
  .celda2 {
    background-color: lightblue;
  }
</style>
</head>
<body>
<div id="tabla">
  <div class="celda1">
    <p>Prima luce, cum quibus mons aliud consensu ab
eo.</p>
  </div>
  <div class="celda2">
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit.</p>
  </div>
</div>
</body>

```

```
</html>
```

La primera caja `<div>` tiene el identificador `#tabla`. El selector CSS `#tabla` hará que aparezca en forma de tabla: `display: table;`.

En esta caja tenemos que crear otras dos cajas `<div>` mediante dos clases: `.celda1` y `.celda2`. Los dos selectores utilizan una diseño en forma de celda: `display: table-cell` y con colores en el fondo.

¿cómo se verá el ejemplo anterior?

Ancho en pantalla de la tabla

Las celdas muestran todo su contenido. En este ejemplo no hay mucho texto, aparece al completo y las celdas no son muy anchas. Ahora veamos qué sucede si aumentamos la cantidad de texto.

Para el primer párrafo añade: ***“Prima luce, cum quibus mons aliud consensu ab eo. Tu quoque, Brute, fili mi, nihil timor populi, nihil! Quae vero auctorem tractata ab fiducia dicuntur. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Plura mihi bona sunt, inclinet, amari petere vellent”***

Y en le segundo añade: ***“Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ut enim ad minim veniam, quis nostrud exercitation. Quam diu etiam furor iste tuus nos eludet? Me non paenitet nullum festiviorem excogitasse ad hoc. Morbi odio eros, volutpat ut pharetra vitae, lobortis sed nibh. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Ut enim ad minim veniam, quis nostrud exercitation. Quisque ut dolor gravida, placerat libero vel, euismod. Plura mihi bona sunt, inclinet, amari petere vellent. Ut enim ad minim veniam, quis nostrud exercitation. Sed haec quis possit intrepidus aestimare tellus”***

¿Cuál es el resultado?

La tabla ocupa todo el ancho disponible en la ventana del navegador y las celdas también ocupan todo el espacio de su elemento padre: la tabla, mostrando así todo su contenido.

Pero lo más importante, como puede comprobar, la altura por defecto de las celdas es equilibrada, independientemente del ancho de la ventana del navegador. Es una particularidad de la vista **table-cell**.

Elementos anónimos

Veamos otra característica de la vista tabular. En el ejemplo anterior no hemos creado la fila, `table-row`. En ese caso, los navegadores crean automáticamente los elementos que faltan para obtener una estructura completa. Ocurriría lo mismo si no hubiéramos definido la tabla con `table`, los navegadores la habrían creado. El hecho de eliminar la caja `<div id="tabla">` no repercute ni en la validez de la sintaxis ni en la visualización. **Comprueballo**

Sin embargo, te aconsejo que introduzca un elemento con un `display: table`, siempre es mejor tener una estructura mínima.

Otros elementos para las tablas

Párrafos en las celdas

En los ejemplos anteriores utilizamos cajas `<div>` para crear la tabla y las celdas. Para simplificar la estructura, puede utilizar cualquier otro elemento para crear las celdas, un párrafo `<p>`, por ejemplo.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Mi diseño</title>
  <meta charset="UTF-8" />
  <style>
    #tabla {
      display: table;
    }
    .celda1, .celda2 {
      display: table-cell;
    }
    .celda1 {
      background-color: lightgreen;
    }
    .celda2 {
```

```

        background-color: lightblue;
    }
</style>
</head>
<body>
<div id="tabla">
    <p class="celda1">Prima luce...</p>
    <p class="celda2">Lorem ipsum dolor...</p>
</div>
</body>
</html>

```

¿Y el resultado es?

El resultado puede verse de inmediato: los párrafos `<p>` **no tienen el relleno superior e inferior por defecto**. El texto está "pegado" a los bordes del bloque. Para un diseño simplificado puede ser bastante útil.

Listas en tablas

Si quiere, no tiene que utilizar la caja `<div>`, en su lugar, puede utilizar una lista ``. El elemento `` aparece como una tabla y los elementos de la lista `` representan las celdas de la tabla.

Este es el código utilizado:

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>Mi diseño</title>
    <meta charset="UTF-8" />
    <style>
        #tabla {
            display: table;
        }
    </style>
</head>
<body>
    <div id="tabla">
        <ul>
            <li>Prima luce...</li>
            <li>Lorem ipsum dolor...</li>
        </ul>
    </div>
</body>
</html>

```

```

        .celda1, .celda2 {
            display: table-cell;
        }
        .celda1 {
            background-color: lightgreen;
        }
        .celda2 {
            background-color: lightblue;
        }
    </style>
</head>
<body>
<ul id="tabla">
    <li class="celda1">Prima luce...</li>
    <li class="celda2">Lorem ipsum dolor...</li>
</ul>
</body>
</html>

```

¿Y obtenemos ?

Diseño de las filas

Ahora vamos a crear una tabla con filas con la propiedad **display: table-row**. Este es el código utilizado:

```

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

```

```

#tabla {
    display: table;
}
.fila1, .fila2 {
    display: table-row;
}
.fila1 {
    background-color: lightgreen;
}
.fila2 {
    background-color: lightblue;
}
</style>
</head>
<body>
<div id="tabla">
    <div class="fila1">
        <p>Prima luce...</p>
    </div>
    <div class="fila2">
        <p>Lorem ipsum dolor sit amet...</p>
    </div>
</div>
</body>
</html>

```

¿Y obtenemos ?

Las dos filas ocupan todo el espacio disponible en el ancho de la tabla padre, que en este ejemplo ocupa todo el ancho disponible en la ventana del navegador.

Establecer el ancho de las celdas

Anchos fijos

En los ejemplos anteriores no habíamos establecido ni el ancho de la tabla ni el de las celdas. Los elementos ocupan el espacio disponible en sus elementos padre.

Ahora ya podemos establecer el ancho de la tabla y de las celdas.

Si solo establece el ancho de la tabla (`#tabla {display: table; width: 800px;}`), las celdas utilizarán un ancho calculado para mostrar el contenido de **forma distribuida**.

¿Y se verá?

Si establece un ancho para la celda mayor o menor que las de la tabla, se utilizará el ancho de la tabla.

Por el contrario, si no establece el ancho de la tabla y sí el de las celdas (`width: 500px;` para la primera y `width: 300px;` para la segunda), evidentemente, será el ancho que se va a utilizar.

¿Y se verá?

Si lo desea, puede establecer un único ancho fijo. En este ejemplo, se ha establecido un ancho para la primera celda de 500 píxeles (`width: 500px;`), para la segunda no se ha hecho. En ese caso, la primera celda tendrá siempre un ancho de 500 píxeles y la segunda ocupa el resto del espacio disponible en su elemento padre.

Anchos en porcentaje

Otra forma de establecer el ancho de las tablas y de las celdas es utilizar los porcentajes. Al cambiar el tamaño de la ventana del navegador, cambiarán todas las dimensiones.

Estos son los estilos aplicados a la tabla:

```
#tabla {  
  display: table;  
  width: 60%;  
}
```

Y estos a las celdas:

```
.celda1 {  
    width: 70%;  
    background-color: lightgreen;  
}  
  
.celda2 {  
    width: 30%;  
    background-color: lightblue;  
}
```

Diseño con una tabla más estructurada

Veamos ahora una tabla más estructurada con un encabezado y un pie de página. Le añadiremos el encabezado con la propiedad `display: table-header-group`; y el pie con `display: table-footer-group`. Este es el código:

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
    <title>Mi diseño</title>  
    <meta charset="UTF-8" />  
    <style>  
        #tabla {  
            display: table;  
        }  
        #encabezado {  
            display: table-header-group;  
            background-color: lightyellow;  
        }  
        #pie {  
            display: table-footer-group;  
            background-color: lightyellow;  
        }  
    </style>  
</head>  
<table>  
    <tr>  
        <th>Encabezado</th>  
        <td>Celda 1</td>  
        <td>Celda 2</td>  
    </tr>  
    <tr>  
        <td>Celda 3</td>  
        <td>Celda 4</td>  
        <td>Celda 5</td>  
    </tr>  
    <tr>  
        <td colspan="3">Pie de página</td>  
    </tr>  
</table>
```

```

    }
    .celda1, .celda2 {
        display: table-cell;
    }
    .celda1 {
        width: 500px;
        background-color: lightgreen;
    }
    .celda2 {
        width: 300px;
        background-color: lightblue;
    }
</style>
</head>
<body>
<div id="tabla">
    <div class="fila">
        <div class="celda1">
            <p>Prima luce, cum quibus mons aliud consensu ab eo. Tu quoque, Brute, fili mi, nihil timor populi, nihil! Quae vero auctorem tractata ab fiducia dicuntur. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Plura mihi bona sunt, inclinet, amari petere vellent.</p>
        </div>
        <div class="celda2">
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ut enim ad minim veniam, quis nostrud exercitation. Quam diu etiam furor iste tuus nos eludet? Me non paenitet nullum festiviorem excogitasse ad hoc. Morbi odio eros, volutpat ut pharetra vitae, lobortis sed nibh. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Ut enim ad

```

```
minim veniam, quis nostrud exercitation. Quisque ut dolor
gravida, placerat libero vel, euismod. Plura mihi bona sunt,
inclinat, amari petere vellunt. Ut enim ad minim veniam, quis
nostrud exercitation. Sed haec quis possit intrepidus
aestimare tellus.</p>

</div>

</div>

<div id="encabezado">

  <p>Encabezado. Immensae subtilitatis...</p>

</div>

<div id="pie">

  <p>Pie de página. Quis aute iure...</p>

</div>

</div>
</body>
</html>
```

El orden de los elementos no altera la visualización. En este ejemplo hemos declarado el encabezado y el pie de página y después las celdas, pero quizá prefiera una estructura más semántica y fácil de asimilar: encabezado, celdas y pie.

¿Qué obtenemos?

Como puede ver, solo hemos establecido el ancho de dos celdas, es más que suficiente para que se vean correctamente. Una vez más, le corresponde decidir si prefiere imponer todos los anchos de los elementos de la tabla.

Otras propiedades de diseño

Veamos algunas propiedades específicas para el diseño en forma de tabla que influyen directamente en el resultado en el navegador.

Propiedad table-layout

La propiedad `table-layout` determina cómo se va a calcular el ancho de las celdas de una tabla que tenga un ancho fijo. Esta propiedad acepta dos valores:

- **table-layout: auto:** es el valor por defecto. En este caso se calcula el ancho de la celda para que la que tenga más contenido sirva como base para el cálculo de la altura de las celdas, para que sea igual en todas.
- **table-layout: fixed:** en este caso el ancho de las celdas es igual y su suma es igual al ancho fijo de la tabla. La altura de las celdas la determina la que tenga más texto.

Estas son las propiedades de la tabla del primer ejemplo:

```
#tabla {
  display: table;
  width: 800px;
  table-layout: auto;
}
```

Y obtenemos esto:

Prima luce, cum quibus mons aliud consensu ab eo. Tu quoque, Brute, fili mi, nihil timor populi, nihil! Quae vero auctorem tractata ab fiducia dicuntur. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Plura mihi bona sunt, inclinet, amari petere vellent.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ut enim ad minim veniam, quis nostrud exercitation. Quam diu etiam furor iste tuus nos eludet? Me non paenitet nullum festiviorem excogitasse ad hoc. Morbi odio eros, volutpat ut pharetra vitae, lobortis sed nibh. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Ut enim ad minim veniam, quis nostrud exercitation. Quisque ut dolor gravida, placerat libero vel, euismod. Plura mihi bona sunt, inclinet, amari petere vellent. Ut enim ad minim veniam, quis nostrud exercitation. Sed haec quis possit intrepidus aestimare tellus.

Y estas las del segundo ejemplo:

```
#tabla {
  display: table;
  width: 800px;
  table-layout: fixed;
}
```

Y obtenemos esto:

Prima luce, cum quibus mons aliud consensu ab eo. Tu quoque, Brute, fili mi, nihil timor populi, nihil! Quae vero auctorem tractata ab fiducia dicuntur. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Plura mihi bona sunt, inclinet, amari petere vellent.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ut enim ad minim veniam, quis nostrud exercitation. Quam diu etiam furor iste tuus nos eludet? Me non paenitet nullum festiviorem excogitasse ad hoc. Morbi odio eros, volutpat ut pharetra vitae, lobortis sed nibh. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Ut enim ad minim veniam, quis nostrud exercitation. Quisque ut dolor gravida, placerat libero vel, euismod. Plura mihi bona sunt, inclinet, amari petere vellent. Ut enim ad minim veniam, quis nostrud exercitation. Sed haec quis possit intrepidus aestimare tellus.

Propiedad border-collapse

Tiene la posibilidad de aplicar bordes a la tabla o a las celdas. Si todas las celdas de la tabla tienen bordes, no es necesario aplicárselo a la tabla. Por defecto, los bordes de las celdas se aplican a cada celda individualmente. Eso significa que los bordes de cada celda son visibles.

Veamos el código de este ejemplo, una tabla con dos filas y dos celdas en cada fila.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <Title>Mi página web</title>
  <meta charset="UTF-8" />
  <style>
    #tabla {
      display: table;
      width: 600px;
    }
    .fila {
      display: table-row;
    }
    .celda {
      display: table-cell;
```

```

        width: 300px;
    }
</style>
</head>
<body>
<div id="tabla">
    <div class="fila">
        <div class="celda">
            <p>Prima luce, cum quibus mons aliud consensu ab eo. Tu
quoque, Brute, fili mi, nihil timor populi, nihil! Quae vero
auctorem tractata ab fiducia dicuntur. Vivamus sagittis lacus
vel augue laoreet rutrum faucibus. Plura mihi bona sunt,
inclinet, amari petere vellent.</p>
        </div>
        <div class="celda">
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit. Ut enim ad minim veniam, quis nostrud exercitation.
Quam diu etiam furor iste tuus nos eludet? Me non paenitet
nullum festiviorem excogitasse ad hoc.</p>
        </div>
    </div>
    <div class="fila">
        <div class="celda">
            <p>Quis aute iure reprehenderit in voluptate velit
esse. Unam incolunt Belgae, aliam Aquitani, tertiam. Quam diu
etiam furor iste tuus nos eludet?
        </p>
        </div>
        <div class="celda">
            <p>Gallia est ommis divisa in partes tres, quarum.
Magna pars studiorum, protida quaerimus. Petierunt uti sibi

```

```
concilium totius Galliae in diem certam indicere. Curabitur  
est gravida et libero vitae dictum</p>  
    </div>  
  </div>  
</div>  
</body>  
</html>
```

Añade a la clase `.celda` borde simple y a cada uno de los párrafos más texto

¿Cual es el resultado?

Como puede comprobar, pueden verse los bordes alrededor de las celdas. Es el comportamiento normal, por defecto.

Utilice la propiedad `border-collapse` para obtener bordes superpuestos, que se fusionen. Los dos valores posibles son:

- **`border-collapse: separate`**: los bordes se separan, es el valor por defecto.
- **`border-collapse: collapse`**: los bordes se combinan.

Si al anterior ejemplo se le aplicara la opción **`collapse`**. ¿Cómo se vería?

Si al anterior ejemplo se le aplicara la opción **`separate`**. ¿Cómo se vería?

Evidentemente, puede aplicar un borde solo a un lado de la celda para tener así una separación única entre dos columnas. Independientemente de la cantidad de contenido que haya en las dos celdas, en las dos columnas, el borde se ajustará siempre a la que tenga más contenido.

Propiedad `border-spacing`

Si no combina los bordes de las celdas (`border-collapse: separate`), puede especificar el espacio entre los bordes de las celdas. Esta propiedad se aplica a la tabla de esta manera:

Si al anterior ejemplo se le aplicara un espacio de 5 píxeles entre los bordes de las celdas ¿Cómo se vería?

Si se introduce un único valor, el espacio se aplica horizontal y verticalmente. Si indica dos valores, el primero se aplicará horizontal y el segundo verticalmente:

Si al anterior ejemplo se le aplicara un espacio de 5 píxeles horizontal y 20 píxeles vertical. **¿Cómo se vería?**

Propiedad empty-cells

La propiedad empty-cells define el comportamiento de las celdas vacías con bordes, en el contexto de una tabla con bordes separados. El valor hide oculta las celdas vacías y show las muestra.

Veamos el primer ejemplo.

Estas son las propiedades de la tabla:

```
#tabla {  
  display: table;  
  width: 600px;  
  border-spacing: 10px;  
}
```

Estas son las propiedades aplicadas a las celdas:

```
.celda {  
  display: table-cell;  
  width: 300px;  
  border: 1px solid #333;  
  empty-cells: hide;  
}
```

La última celda de la tabla carece de contenido: `<div class="celda">`
`</div>`.

¿cómo se verá?

Si utilizamos el valor empty-cells: show;, ¿qué pasan con los bordes?:

Propiedad caption-side

El elemento table-caption añade una leyenda en la parte superior de la tabla.

El código de este ejemplo es el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <Title>Mi página web</title>
  <meta charset="UTF-8" />
  <style>
    #tabla {
      display: table;
      width: 600px;
      border-collapse: collapse;
    }
    .fila {
      display: table-row;
    }
    .celda {
      display: table-cell;
      width: 300px;
      border: 1px solid #333;
    }
    #leyenda {
      display: table-caption;
    }
  </style>
</head>
<body>
<div id="tabla">
  <div class="fila">
```

```

<div class="celda">
  <p>Prima luce...</p>
</div>
<div class="celda">
  <p>Lorem ipsum dolor sit amet...</p>
</div>
</div>
<div class="fila">
  <div class="celda">
    <p>Quis aute iure reprehenderit...</p>
  </div>
  <div class="celda">
    <p>Quisque placerat facilisis...</p>
  </div>
</div>
<p id="leyenda">Quis aute iure reprehenderit in voluptate
velit esse.</p>
</div>
</body>
</html>

```

¿Qué obtenemos?

La propiedad `caption-side` establece la ubicación de la leyenda. Puede utilizar los valores `top`, `right`, `bottom` y `left`.

Estas son las propiedades para colocar la leyenda debajo de la tabla:

```

#leyenda {
  display: table-caption;
  caption-side: bottom;
}

```

Alineación vertical

La alineación vertical en las cajas `<div>` siempre ha supuesto un "problema" para los maquetadores. La propiedad `vertical-align` definida en las celdas es nativa en el diseño con tablas. Sirve para centrar el contenido en una celda.

Veamos un ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <Title>Mi página web</title>
  <meta charset="UTF-8" />
  <style>
    #tabla {
      display: table;
    }
    .celda1, .celda2 {
      display: table-cell;
      height: 200px;
      vertical-align: middle;
    }
    .celda1 {
      background-color: lightgreen;
    }
    .celda2 {
      background-color: lightblue;
    }
  </style>
</head>
<body>
```



```
<div id="tabla">
  <div class="celda1">
    <p>Prima luce, cum quibus mons aliud consensu ab eo.</p>
  </div>
  <div class="celda2">
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit.</p>
  </div>
</div>
</body>
</html>
```

¿Qué obtenemos?

Conclusión

El diseño con tablas con la propiedad display y el resto de valores son una verdadera ventaja para los diseñadores.

Veamos algunas:

- La altura de las celdas es siempre la misma. Dicho de forma más técnica, la altura de los elementos hermanos es idéntica. Con otros métodos, la altura de las celdas depende de la cantidad de contenido que tengan. Con este tipo de tablas no nos tenemos que preocupar por este aspecto.
- La propiedad vertical-align gestiona de forma nativa la alineación vertical. Una vez más, es una buena manera de aplicar un centrado vertical a un elemento en una celda.
- No hay problema de diferentes alturas entre celdas con colores de fondo. Como la altura de las celdas es igual, no suponen ningún problema para que los colores del fondo se vean adecuadamente.
- No hay problemas a la hora de generar bordes en un solo lado de las celdas con contenidos distintos, ya que la altura de estas será siempre igual. De nuevo una ventaja para los maquettadores.

- Puede utilizar unidades distintas, fijas en píxeles y relativas en porcentaje, para gestionar el ancho y el alto de las celdas.
- Se acabaron los problemas de flotación de los elementos adyacentes. Todos los elementos de las tablas están en el flujo "normal".
- Código limpio, semántico, ligero y no verbal.