



SKANDA
Technology Advisors

FUNDAMENTOS DEL DISEÑO WEB



Contenido

Fundamentos del front-end.....	4
Que es front-end	4
Frameworks, plugins, librerías, widgets	4
jquery,bootstrap,foundation.. etc.....	4
Jquery	4
Bootstrap.....	5
Foundation	6
Herramientas de creación de código	7
sublime text (emmet), atom, Intel® XDK	7
SublimeText.....	7
Atom.....	8
Plugin Emmet	8
Instalación de un servidor local y servicios disponibles en la nube para tu web	11
Instalación de un Wamp y opciones que existen en la nube de servidores web	11
¿Qué es responsive design?	11
Principios del responsive y para qué sirve	11
Tecnologías implicadas en el desarrollo front-end	12
Html5	12
Etiquetas	12
Estructura de una página web con html.....	12
Html5 vs html	13
Definiendo cabeceras en HTML5.....	16
Seccionado implícito.....	17
CSS3	19
Fundamentos de CSS	19
Selectores	20
Diseño web aplicando estilos con css.....	20
Fundamentos de la maquetación	25
Prototipado	25

Introducción al prototipado	25
Manejo de ninjamock.....	28
Modelos de caja	28
Posicionamiento	30
Posicionamiento en CSS	30
Dimensiones	30
Los modos de compatibilidad de Internet Explorer 8	53
Posicionamiento y visualización.....	66
Tipos de elementos.....	66
Posicionamiento normal	68
Relación entre display, float y position	91
EJERCICIO.....	96
Ejemplo práctico de una maqueta web.....	99
Contenido y diseño de la estructura web.....	99
Cabecera y menú.....	99
Pie de página de la web	101
Ejercicio practico	101
Aplicación de todo lo aprendido de maqueta web.....	101
Facilitando el diseño web.....	102
Frameworks para maqueta web	102
Foundation, Bootstrap, Boilerplate.....	102
Bootstrap.....	104
Boilerplate.....	104
Introducción a bootstrap	105
Estructura, conceptos e implementación	105
Diseñando para bootstrap	109
Crear una página web con bootstrap	109
Bibliografía	109
Cibergrafía	110

Fundamentos del Diseño Web

Fundamentos del front-end

Que es front-end

Los frontends tienden a ser programadores, pero hay diseñadores genios que también hacen frontend. Son los encargados de maquetar la estructura semántica del contenido (HTML), codificar el diseño en hojas de estilo (CSS) y agregar la interacción con el usuario (Javascript).

En la época actual los frontends tienen HTML5 y CSS3. Con HTML5, desde el frontend, es posible hacer geolocalización, dibujo vectorial, guardar datos en el disco del usuario, insertar audio y video, entre otras cosas.

Con CSS3, se pueden crear diseños altamente complejos sin la necesidad de imágenes cortadas, sólo usando código. Bordes redondeados, sombras, degradados, fondos múltiples, entre otros.

Por último, Javascript y sus frameworks añaden el componente de interactividad y conexión al servidor. Es posible comunicarse con el backend y la base de datos sin recargar la página usando AJAX o WebSockets, recibir esos datos y cambiar el diseño entero del sitio. jQuery hace todo esto fácil pero no es el único framework de Javascript.

Frameworks, plugins, librerías, widgets

jquery,bootstrap,foundation.. etc

Jquery

jQuery es uno de los complementos más esenciales para el desarrollo web, usado en millones de sitios en toda la web, ya que nos facilita mucho el desarrollo de aplicaciones enriquecidas del lado del cliente, en Javascript, compatibles con todos los navegadores.

Para los que se inician, conviene aclarar que jQuery no es un lenguaje, sino una serie de funciones y métodos de Javascript. Por tanto, Javascript es el lenguaje y jQuery es una librería que podemos usar opcionalmente si queremos facilitar nuestra vida cuando programamos en Javascript. A veces nos podemos referir a jQuery como framework o incluso como un API de funciones, útiles en la mayoría de proyectos web.

Antes de llegar jQuery los desarrolladores estábamos obligados a discriminar entre los diversos navegadores, para ejecutar aquel código Javascript que funcionaba en cada browser. Con la llegada de jQuery la principal ventaja es que ya no necesitamos preocuparnos sobre si el navegador del usuario es Explorer, Chrome, Firefox, etc. sino que la propia librería hará el trabajo "sucio" por nosotros y ejecutará el código que sea compatible con el software del cliente que está accediendo a nuestra web. Para ello usaremos las funciones que jQuery nos proporciona, dentro de un grandísimo abanico de funcionalidades que además se extiende por medio de miles de plugins que ofrece la comunidad para implementar cualquier tipo de comportamiento.

Para aprender jQuery necesitas saber Javascript. No requiere ser un gran maestro en el lenguaje, pero al menos sí trabajar con él con cierta soltura. Date cuenta que cuando programas con jQuery en realidad estás programando con Javascript, por ello es importante que no intentes empezar la casa por el tejado y primero aprendas el lenguaje "padre". En DesarrolloWeb.com encontrarás varios manuales de Javascript.

Bootstrap

Bootstrap, es un framework originalmente creado por Twitter, que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como "responsive design" o diseño adaptativo.

El beneficio de usar responsive design en un sitio web, es principalmente que el sitio web se adapta automáticamente al dispositivo desde donde se acceda. Lo que se usa con más frecuencia, y que a mi opinión personal me gusta más, es el uso de media queries, que es un módulo de CSS3 que permite la representación de contenido para adaptarse a condiciones como la resolución de la pantalla y si trabajás las dimensiones de tu contenido en porcentajes, puedes tener una web muy fluida capaz de adaptarse a casi cualquier tamaño de forma automática.

Pero si no quieres nada que ver con los media queries, otra muy buena opción es el uso del framework de Bootstrap, que como te dije te ayudará a desarrollar tus sitios adaptativos.

Aun ofreciendo todas las posibilidades que ofrece Bootstrap a la hora de crear interfaces web, los diseños creados con Bootstrap son simples, limpios e intuitivos, esto

les da agilidad a la hora de cargar y al adaptarse a otros dispositivos. El Framework trae varios elementos con estilos predefinidos fáciles de configurar: Botones, Menús desplegables, Formularios incluyendo todos sus elementos e integración jQuery para ofrecer ventanas y tooltips dinámicos.

Bootstrap tiene un soporte relativamente incompleto para HTML5 y CSS 3, pero es compatible con la mayoría de los navegadores web. La información básica de compatibilidad de sitios web o aplicaciones está disponible para todos los dispositivos y navegadores. Existe un concepto de compatibilidad parcial que hace disponible la información básica de un sitio web para todos los dispositivos y navegadores. Por ejemplo, las propiedades introducidas en CSS3 para las esquinas redondeadas, gradientes y sombras son usadas por Bootstrap a pesar de la falta de soporte de navegadores antiguos. Esto extiende la funcionalidad de la herramienta, pero no es requerida para su uso.

Foundation

Foundation surge como un proyecto de ZURB para desarrollar código de interfaz de usuario más rápido y mejor. En octubre 2011, ZURB liberó Foundation 2.0 como código abierto bajo licencia de MIT License.¹ En junio 2012 ZURB liberó una actualización importante, Foundation 3.0.2 En febrero 2013 ZURB liberó otra actualización importante, Foundation 4.0.3 En noviembre 2013 ZURB liberó otra actualización importante, Foundation 5.0. El equipo está trabajando en la próxima versión de Foundation para Sitios la cual tendría que ser liberado en la primavera del 2015.

Foundation para Email fue liberado en septiembre del 2013

Foundation para Aplicaciones fue liberado en diciembre del 2014

Foundation fue diseñado para y probado en numerosos dispositivos y navegadores. Es el primer framework mobile first responsive construido con Sass/SCSS dando buenas prácticas a diseñadores para el desarrollo rápido. El framework incluye los patrones necesarios más comunes para rápidamente maquetar un sitio responsive. A través del uso de Sass mixins, los componentes de Foundation son fácilmente estilizados y sencillos de extender.

Desde la versión 2.0 también soporta el diseño responsive.⁴ Esto significa el diseño gráfico de las páginas webs se ajusta dinámicamente, teniendo en cuenta las características del dispositivo utilizado (PC, tableta, teléfono celular). Además, desde 4.0 ha tomado un enfoque mobile-first, diseñando y desarrollando para dispositivos móviles primero, y mejorando las páginas web y aplicaciones para pantallas más grandes.⁵

Foundation es código abierto y está disponible en Github. Los desarrolladores están animados para participar en el proyecto y hacer sus contribuciones propias a la plataforma.

Herramientas de creación de código

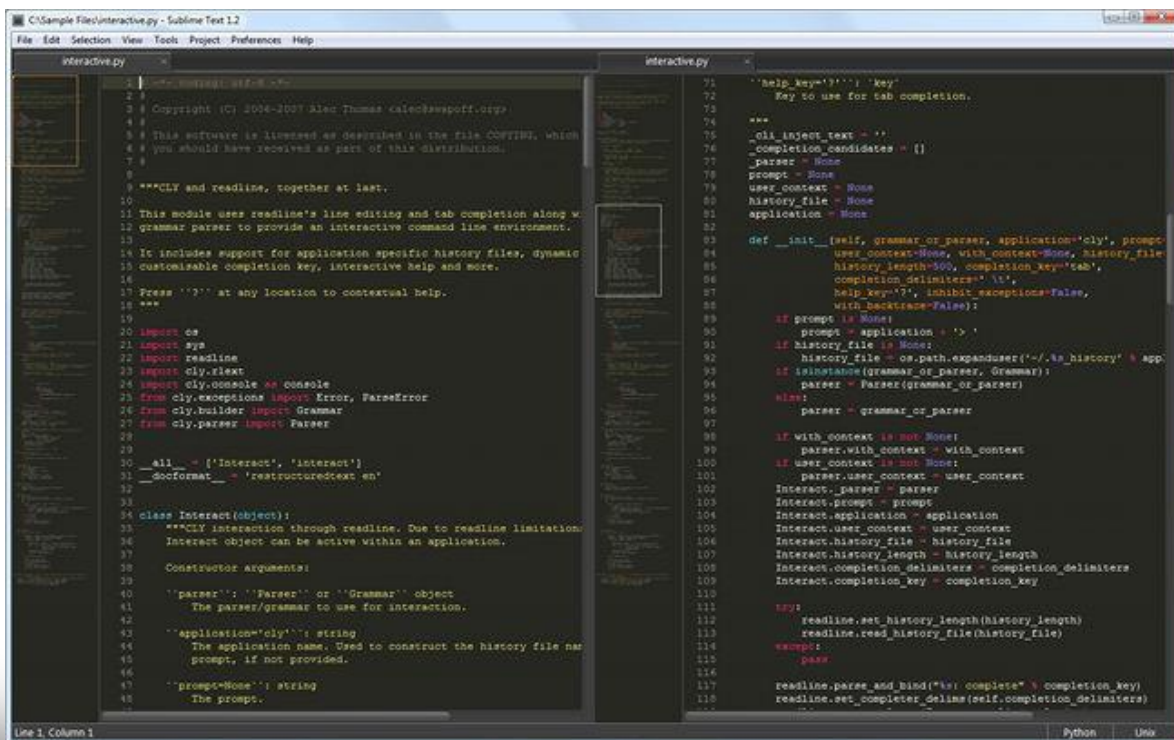
sublime text (emmet), atom, Intel® XDK

SublimeText

Sublime Text es un editor de código multiplataforma, ligero y con pocas concesiones a las florituras. Es una herramienta concebida para programar sin distracciones. Su interfaz de color oscuro y la riqueza de coloreado de la sintaxis, centra nuestra atención completamente.

Sublime Text permite tener varios documentos abiertos mediante pestañas, e incluso emplear varios paneles para aquellos que utilicen más de un monitor. Dispone de modo de pantalla completa, para aprovechar al máximo el espacio visual disponible de la pantalla.

El programa cuenta "de serie" con 22 combinaciones de color posibles, aunque se pueden conseguir más. Para navegar por el código cuenta con Minimap, un panel que permite moverse por el código de forma rápida.



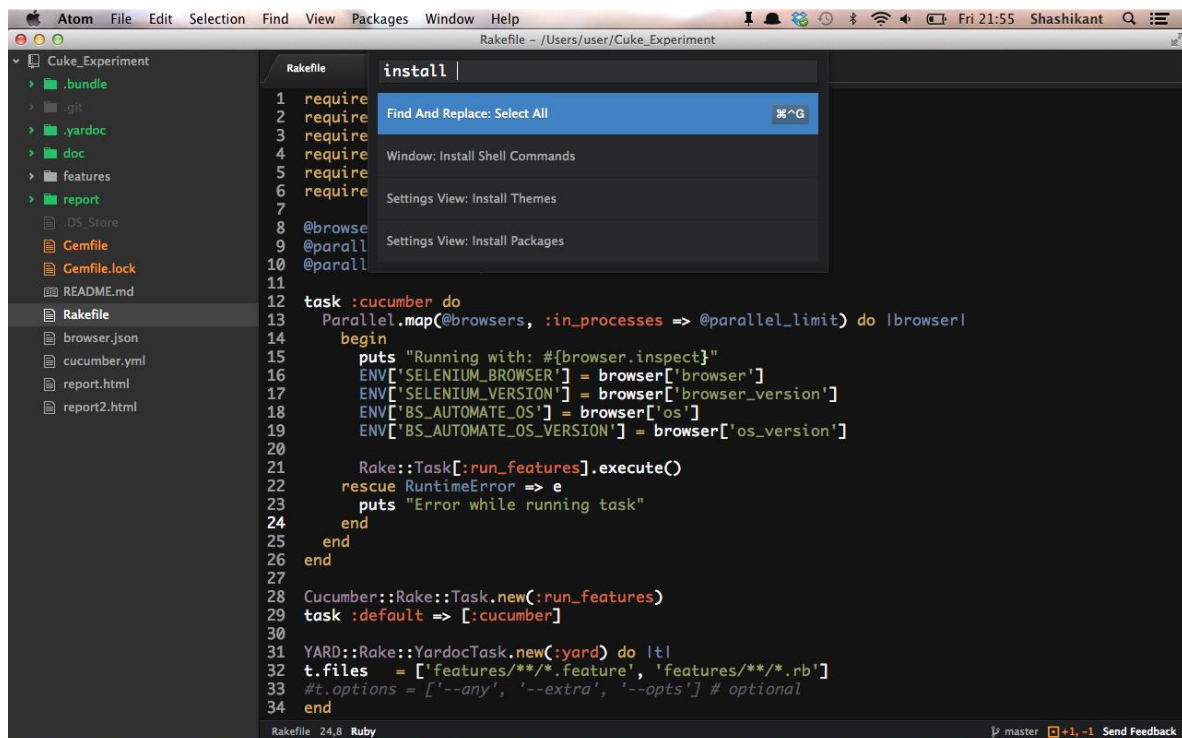
```
1 #!/usr/bin/env python
2
3 Copyright (C) 2006-2007 Alex Thomas <alex@awspoff.org>
4
5 This software is licensed as described in the file COPYING, which
6 you should have received as part of this distribution.
7
8
9 """CLI and readline, together at last.
10
11 This module uses readline's line editing and tab completion along w
12 grammar parser to provide an interactive command line environment.
13
14 It includes support for application specific history files, dynamic
15 customisable completion key, interactive help and more.
16
17 Press '?' at any location to contextual help.
18 """
19
20 import os
21 import sys
22 import readline
23 import cly.rex
24 import cly.console as console
25 from cly.exceptions import Error, ParseError
26 from cly.builder import Grammar
27 from cly.parser import Parser
28
29
30 all_ = ['Interact', 'Interact']
31 __docformat__ = 'restructuredtext en'
32
33 class Interact(object):
34     """CLI interaction through readline. Due to readline limitations
35     Interact object can be active within an application.
36
37     Constructor arguments:
38     'parser': 'Parser' or 'Grammar' object
39         The parser/grammar to use for interaction.
40     'application': string
41         The application name. Used to construct the history file na
42         prompt, if not provided.
43     'prompt': string
44         The prompt.
45
46     """
47     def __init__(self, grammar_or_parser, application='cli', prompt
48                 user_context=None, with_context=None, history_file
49                 history_length=500, completion_key='tab',
50                 completion_delimiters=' \t',
51                 help_key='?', inhibit_exceptions=False,
52                 with_backspace=False):
53         if prompt is None:
54             prompt = application + ' '
55         if history_file is None:
56             history_file = os.path.expanduser('~/.%s_history' % app
57         if isinstance(grammar_or_parser, Grammar):
58             parser = Parser(grammar_or_parser)
59         else:
60             parser = grammar_or_parser
61
62         self.parser = parser
63         self.with_context = with_context
64         self.parser.with_context = with_context
65         self.user_context = user_context
66         self.parser.user_context = user_context
67         self.interact_parser = parser
68         self.interact_prompt = prompt
69         self.interact_application = application
70         self.interact_user_context = user_context
71         self.interact_history_file = history_file
72         self.interact_history_length = history_length
73         self.interact_completion_delimiters = completion_delimiters
74         self.interact_completion_key = completion_key
75
76     def __call__(self):
77         readline.set_history_length(self.interact_history_length)
78         readline.read_history_file(self.interact_history_file)
79         example:
80         pass
81
82         readline.parse_and_bind("?: complete" % self.interact_completion_key)
83         readline.set_completer_delimiter(self.interact_completion_delimiters)
```

Atom

Atom, el editor de código desarrollado por la gente de Github. No es el más ambicioso ni futurista, pero pinta bien... dentro de lo poco que hemos podido ver hasta el momento.

La principal característica de Atom es, según sus creadores, que es totalmente "hackeable", es decir, que se puede toquetear y modificar hasta el mismo core. Esto es posible gracias que, a pesar de ser una aplicación de escritorio (lo que antiguamente se llamaba programa), está realizada enteramente con tecnologías web. Además, estará perfectamente integrado con Node.js lo que permitirá el uso de los miles de paquetes y librerías que componen su repositorio además de los 50 paquetes que contendrá por defecto su core.

A todo esto, habrá que añadir navegación por el sistema de ficheros, buscar y reemplazar a través de todo el proyecto activo, múltiples ventanas, cursores y selectores, snippets y muchas más... aunque tendremos que esperar todavía para ver todas estas características porque Atom se encuentra en fase de beta privada. Las invitaciones se pueden pedir en la misma página del proyecto.



Plugin Emmet

Para escribir código basta con el editor de texto más sencillo de tu sistema operativo llámese Bloc de Notas, Gedit, Kate o TextEdit. Pero todos los que trabajamos

desarrollando código sabemos que escribir en un editor de texto plano equivale a darle a la cajera del supermercado una libreta y un boli (Con un poco de piedad una calculadora también). Una tarea posible en la teoría, pero imposible en la práctica. Cada vez tenemos editores más complejos y con más funcionalidades que ya no solo se limitan a colorear el código sino potentes herramientas de autocompletado, detección de errores en tiempo real, refactorización, etc.

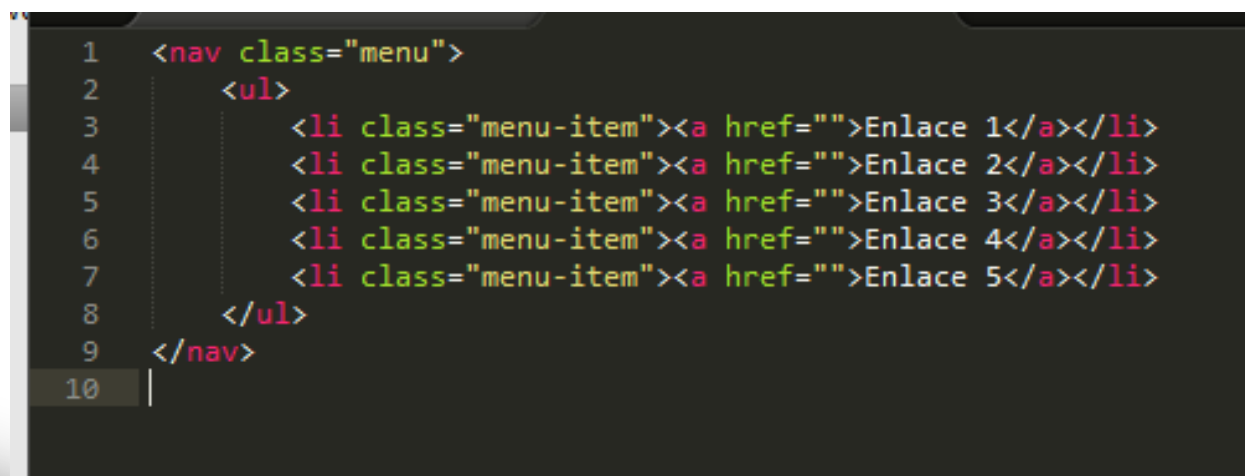
Si desarrollas para la web, en concreto para el frontend, Emmet da una vuelta de tuerca más para hacerte altamente productivo, se trata de un plugin que a base de escribir pequeñas abreviaciones genera grandes estructuras de código HTML y CSS.

Aunque parezca una contradicción estaréis de acuerdo conmigo en que a los programadores no nos gusta escribir código. Nos gusta resolver problemas, plasmarlo en código es un daño colateral para la mayoría. Siempre vamos a estar buscando la manera de escribiendo lo menos posible hacer más cosas, es la esencia de la programación.

El lenguaje HTML está muy bien, un lenguaje de marcas basado en XML que se basa en el uso de etiquetas y atributos. Fácil de procesar para la máquina y fácil de leer para el ser humano, aunque el que escribe el código no está tan de acuerdo con esto ya que es tedioso de escribir por repetitivo. Con Emmet se acabó para siempre lo tedioso, por fin nos podremos centrar en lo que de verdad importa, entregarlo a tiempo haciendo lo mínimo.

Como decíamos Emmet se base en pequeñas abreviaciones que generan grandes estructuras de código. Lo mejor es verlo con algunos ejemplos. Por ejemplo, la siguiente línea:

nav.menu>ul>li.menu-item*5>a{Enlace \$}



```
1  <nav class="menu">
2    <ul>
3      <li class="menu-item"><a href="">Enlace 1</a></li>
4      <li class="menu-item"><a href="">Enlace 2</a></li>
5      <li class="menu-item"><a href="">Enlace 3</a></li>
6      <li class="menu-item"><a href="">Enlace 4</a></li>
7      <li class="menu-item"><a href="">Enlace 5</a></li>
8    </ul>
9  </nav>
10 |
```

ul.generic-list>lorem10.item-\$*4

```
1 <ul class="generic-list">
2   <li class="item-1">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ea, voluptatibus!</li>
3   <li class="item-2">Eaque, impedit est tempora doloreque praesentium voluptatum quas nostrum consectetur.</li>
4   <li class="item-3">Illum, reprehenderit minus ex soluta adipisci aperiam explicabo modi sapiente.</li>
5   <li class="item-4">Nihil ratione voluptates iure cum eligendi dolores deleniti quos nostrum.</li>
6 </ul>
```

p*3>lorem15

```
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Officia, doloribus cum animi perferendis  
fugit tenetur.</p>
<p>Dolorum, rem labore voluptatem iusto officia corporis molestias distinctio. Voluptas corrupti  
consectetur molestiae sed libero!</p>
<p>Corporis, reiciendis quos cumque nihil harum eaque libero officiis enim amet eum odio a veniam.</p>
```

Como hemos dicho Emmet es un plugin, pero ¿Para qué editor? Pues para ¡Muchos! Emmet está disponible de manera oficial para:

Eclipse/Aptana

TextMate

Coda

Espresso

Chocolat

Komodo Edit

Notepad++

PSPad

textarea

CodeMirror

Brackets

NetBeans

Adobe Dreamweaver

Instalación de un servidor local y servicios disponibles en la nube para tu web

Instalación de un Wamp y opciones que existen en la nube de servidores web

WampServer es un entorno de desarrollo web para Windows con el que podrás crear aplicaciones web con Apache, PHP y bases de datos MySQL database. También incluye PHPMyAdmin y SQLiteManager para manejar tus bases de datos

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa.

¿Qué es responsive design?

Principios del responsive y para qué sirve

El diseño web adaptable, adaptativo o responsivo, conocido por las siglas RWD del inglés Responsive Web Design, es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarla. Hoy día las páginas web se visualizan en multitud de tipos de dispositivos como tabletas, teléfonos inteligentes, libros electrónicos, portátiles, PC, etcétera. Además, aún dentro de cada tipo, cada dispositivo tiene sus características concretas: tamaño de pantalla, resolución, potencia de CPU, capacidad de memoria, entre otras. Esta tecnología pretende que, con un solo diseño web, se tenga una visualización adecuada en cualquier dispositivo.

El diseñador y autor norteamericano Ethan Marcotte creó y difundió esta técnica a partir de una serie de artículos en A List Apart,¹ una publicación en línea especializada en diseño y desarrollo web, idea que luego extendería en su libro Responsive Web Design.

Tecnologías implicadas en el desarrollo front-end

Html5

HTML5 es una colección de estándares para el diseño y desarrollo de páginas web. Esta colección representa la manera en que se presenta la información en el explorador de internet y la manera de interactuar con ella.

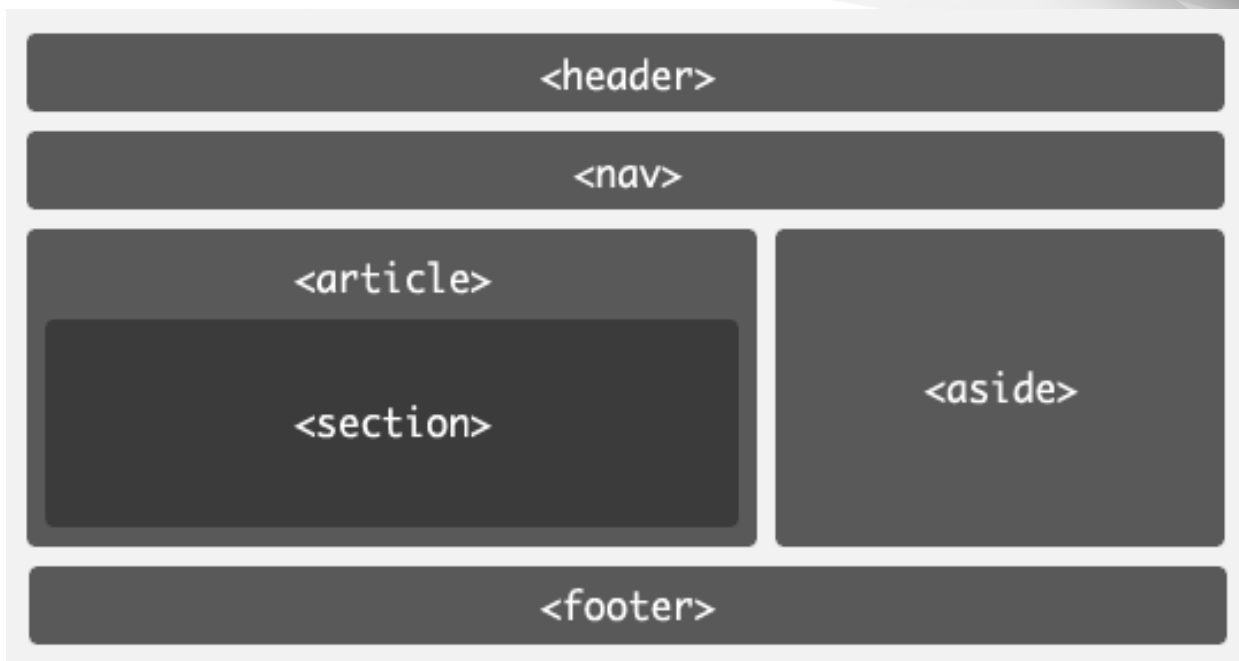
HTML5 está siendo desarrollado por Ian Hickson de Google Inc. y David Hyatt de Apple Inc. junto con todas las personas que participan en Web Hypertext Application Technology Working Group.

Etiquetas

```
<!DOCTYPE html>
<html>
<head>...</head>
<body>
<header><!-- El encabezado a utilizar --></header>
<nav><!-- Nuestro menú de navegación --></nav>
<article>
  <section>...</section>
</article>
<footer>...</footer>
</body>
</html>
```

Estructura de una página web con html

HTML5 nos permite una mayor interacción entre nuestras páginas web y contenido media (video, audio, entre otros) así como una mayor facilidad a la hora de codificar nuestro diseño básico.



Esta nueva versión se basó en el diseño más común de las páginas web alrededor del mundo para llegar a un estándar de etiquetas que realicen las mismas tareas de manera más rápida y eficiente, he aquí algunos ejemplos:

Un nuevo diseño para páginas web, reflejado en las etiquetas `<header>`, `<footer>`, `<nav>`, `<section>`, `<article>` las cuales están destinadas a remplazar la necesidad de tener una `<div>` para cada parte de la página, y en cambio, tener etiquetas específicas para ello.

La nueva etiqueta `<video>` para insertar un reproductor de video, mejorando el reproductor antiguo utilizado por la etiqueta `<embed>` y evitándonos la pena de insertar el código de `<object>`, así como eliminar la necesidad del Flash Player para reproducir videos (lo que nos lleva a un ahorro en la cantidad de memoria utilizada).

Una nueva tag `<audio>` para insertar audio en nuestro sitio web, remplazando la vieja etiqueta `<embed>` con las mismas cualidades de la etiqueta anterior.

Una etiqueta `<canvas>` para manejo de gráficos en internet, sea para dibujar vectores o hacer animaciones.

Html5 vs html

Solo algunas de las diferencias más notables son las siguientes:

✓ No es necesario el cierre de las etiquetas `img`, `br`, `hr`, `input`, etc.
Por ejemplo:

 ahora se sustituye por:

<hr /> por: <hr>
 por:
<input/> por: <input>

- ✓ Se puede usar tanto minúsculas como mayúsculas en el código a diferencia del XHTML que solo admite minúsculas.
- ✓ El Doctype o declaración del documento está completamente minimizado, en HTML5 solo es necesario emplear:
<!DOCTYPE html>
- ✓ Introduce nuevos marcadores para sumarlos a los existentes <div> en usos específicos, por ejemplo: <nav>, <footer>, <audio>, <video>, etc.

Todo el contenido incluido dentro del elemento <body> es parte de una sección. Las secciones en HTML5 pueden ser anidadas. Además de la sección principal, definida por el elemento <body>, los límites de la sección son definidos explícita o implícitamente. Las secciones definidas explícitamente son el contenido definido en las etiquetas <body>, <section>, <article>, <aside>, <footer>, <header>, y <nav>.

Ejemplo:

```
<section>
  <h1>Forest elephants</h1>
  <section>
    <h1>Introduction</h1>
    <p>In this section, we discuss the lesser known forest elepha
  </section>
  <section>
    <h1>Habitat</h1>
    <p>Forest elephants do not live in trees but among them.
  </section>
  <aside>
    <p>advertising block
  </aside>
</section>
<footer>
  <p>(c) 2010 The Example company
</footer>
```

El bloque de HTML define dos secciones de alto nivel:

```
<section>
  <h1>Forest elephants</h1>
  <section>
    <h1>Introduction</h1>
    <p>In this section, we discuss the lesser known forest elephants.
  </section>
  <section>
    <h1>Habitat</h1>
    <p>Forest elephants do not live in trees but among them.
  </section>
  <aside>
    <p>advertising block
  </aside>
</section>

<footer>
  <p>(c) 2010 The Example company
</footer>
```

La primera sección tiene tres subsecciones:

```
<section>
  <h1>Forest elephants</h1>

  <section>
    <h1>Introduction</h1>
    <p>In this section, we discuss the lesser known forest elephants.
  </section>

  <section>
    <h1>Habitat</h1>
    <p>Forest elephants do not live in trees but among them.
  </section>

  <aside>
    <p>advertising block
  </aside>
</section>

<footer>
  <p>(c) 2010 The Example company
</footer>
```

Esto genera la siguiente estructura:

- 1. Forest elephants
 - 1.1 Introduction
 - 1.2 Habitat
 - 1.3 Section (aside)

Definiendo cabeceras en HTML5

Mientras que los elementos de seccionado en HTML definen la estructura de un documento, un perfil también necesita cabeceras para ser útiles. La regla básica es simple: el primer elemento de cabecera HTML (uno de los `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`) define la cabecera de la sección actual.

Los elementos de cabecera tienen un *rango* dado por el número del nombre del elemento, donde `<h1>` tiene el rango *más alto*, y `<h6>` tiene el rango *más bajo*. Los rangos relativos importan sólo en una sección; las estructuras de las secciones determinan el perfil, no el rango de cabeceras de las secciones. Por ejemplo, este código:

```
<section>
  <h1>Forest elephants</h1>
  <p>In this section, we discuss the lesser known forest elephants.
  ...this section continues...
  <section>
    <h2>Habitat</h2>
    <p>Forest elephants do not live in trees but among them.
    ...this subsection continues...
  </section>
</section>
<section>
  <h3>Mongolian gerbils</h3>
  <p>In this section, we discuss the famous mongolian gerbils.
  ...this section continues...
</section>
```

Genera el siguiente perfil:

- 1. Forest elephants
 - 1.1 Habitat
- 2. Mongolian gerbils

Nótese que el rango del elemento de cabecera (en el ejemplo, `<h1>` para la primera sección de más alto nivel, `<h2>` para la subsección y `<h3>` para la segunda sección de alto nivel) no es importante. (Cualquier rango puede ser usado como la cabecera de una sección explícitamente definida, aunque esta práctica no es recomendada).

Seccionado implícito

Debido a que los elementos de seccionado HTML5 no son obligatorios para definir un perfil, para mantener la compatibilidad con la web existente dominada por HTML4, hay una forma de definir secciones sin ellas. Esto es llamado *seccionado implícito*.

Los elementos de cabecera (<h1> a <h6>) definen un nuevo seccionado implícito cuando ellos no son el primer encabezado de sus secciones padre. La forma en que esta sección implícita es posicionada en el perfil es definida por su rango relativo con la cabecera anterior en su sección padre. Si es de un rango más bajo que la cabecera anterior, abre una sub-sección implícita de la sección. Este código:

```
<section>
  <h1>Forest elephants</h1>
  <p>In this section, we discuss the lesser known forest elephants.
  ...this section continues...
  <h3 class="implicit subsection">Habitat</h3>
  <p>Forest elephants do not live in trees but among them.
  ...this subsection continues...
</section>
```

Genera el siguiente perfil:

- 1. Forest elephants
 - 1.1 Habitat (implicitly defined by the h3 element)

Si es del mismo rango que la cabecera anterior, cierra la sección previa (¡que puede haber sido explícita!) y abre una nueva sección implícita del mismo nivel:

```
<section>
  <h1>Forest elephants</h1>
  <p>In this section, we discuss the lesser known forest elephants.
  ...this section continues...
  <h1 class="implicit section">Mongolian gerbils</h1>
  <p>Mongolian gerbils are cute little mammals.
  ...this section continues...
</section>
```

genera el siguiente perfil:

- 1. Forest elephants
- 2. Mongolian gerbils (implicitly defined by the h1 element, which closed the previous section at the same time)

Si es de un rango más alto que el encabezamiento anterior, cierra la sección anterior y abre una nueva sección implícita al más alto nivel:

```
<body>
  <h1>Mammals</h1>
  <h2>Whales</h2>
  <p>In this section, we discuss the swimming whales.
    ...this section continues...
  <section>
    <h3>Forest elephants</h3>
    <p>In this section, we discuss the lesser known forest elephants.
      ...this section continues...
    <h3>Mongolian gerbils</h3>
    <p>Hordes of gerbils have spread their range far beyond Mongolia.
      ...this subsection continues...
    <h2>Reptiles</h2>
    <p>Reptiles are animals with cold blood.
      ...this subsection continues...
  </section>
</body>
```

generando el siguiente perfil:

1. Mammals
 - 1.1 Whales (implicitly defined by the h2 element)
 - 1.2 Forest elephants (explicitly defined by the section element)
 - 1.3 Mongolian gerbils (implicitly defined by the h3 element, which closes the previous section at the same time)
2. Reptiles (implicitly defined by the h2 element, which closes the previous section at the same time)

Este no es el perfil que uno podría esperar por mirar rápidamente las etiquetas de encabezamiento. Para hacer que tu marcado sea entendible por humanos, es una buena práctica usar etiquetas explícitas para abrir y cerrar secciones, y emparejar el rango de cabecera al nivel anidado de la sección deseada. Sin embargo, esto no es requerido por la especificación HTML5. Si encuentra que los navegadores están generando el perfil de su documento en formas inesperadas, verifique si tiene secciones que están cerradas implícitamente por elementos de cabecera.

Una excepción a la regla de que ese rango de cabecera debe emparejar el nivel anidado de sección es para secciones que pueden ser reusadas en múltiples documentos. Por ejemplo, una sección puede ser almacenada en un sistema de gestión de contenidos y ensamblada en documentos en tiempo de ejecución. En este caso, una buena práctica es comenzar en `<h1>` para el nivel de cabecera

más alto de la sección reusable. El nivel anidado de la sección reusable será determinado por la jerarquía de la sección del documento en el que aparece. Las etiquetas de secciones aún son útiles en este caso.

CSS3

Fundamentos de CSS

CSS es un lenguaje utilizado en la presentación de documentos HTML. Un documento HTML viene siendo coloquialmente "una página web". Entonces podemos decir que el lenguaje CSS sirve para organizar la presentación y aspecto de una página web. Este lenguaje es principalmente utilizado por parte de los navegadores web de internet y por los programadores web informáticos para elegir multitud de opciones de presentación como colores, tipos y tamaños de letra, etc.



La filosofía de CSS se basa en intentar separar lo que es la estructura del documento HTML de su presentación. Por decirlo de alguna manera: la página web sería lo que hay debajo (el contenido) y CSS sería un cristal de color que hace que el contenido se vea de una forma u otra. Usando esta filosofía, resulta muy fácil cambiarle el aspecto a una página web: basta con cambiar "el cristal" que tiene delante. Piensa por ejemplo qué ocurre si tienes un libro de papel y lo miras a través de un cristal de color azul: que ves el libro azul. En cambio, si lo miras a través de un cristal amarillo, verás el libro amarillo. El libro (el contenido) es el mismo, pero lo puedes ver de distintas maneras.

Algunas opciones básicas del lenguaje CSS por ejemplo pueden ser el poder cambiar el color de algunas típicas etiquetas HTML como <H1> (h1 es una etiqueta en el lenguaje HTML destinada a mostrar un texto como encabezado, en tamaño grande). Pero también hay funciones algo más complejas, como introducir espaciado entre elementos <DIV> (div es una etiqueta HTML para identificar una determinada región o división de contenido dentro de una página web) o establecer imágenes de fondo.

CSS es muy intuitivo y sencillo una vez se llega a aprender, ya que para su definición siempre se hace uso de un identificador de etiqueta HTML (como por ejemplo <H1>), y luego indicamos con qué aspecto queremos que se muestren todas las etiquetas <H1> que aparezcan en un documento. Al igual que con <H1> podemos definir cómo queremos que se muestren las distintas partes del

documento HTML, pudiendo en cada caso definir sus propiedades (color, tipo de fuente, tamaño, espacio, imagen) con algún determinado valor deseado.

Vamos a partir de un ejemplo muy sencillo, que tratará de una página web o archivo HTML donde tan solo tendremos un párrafo de texto y sobre cuya etiqueta <p> iremos realizando cambios e iremos aplicando los conocimientos que vamos a ir adquiriendo y posteriormente veremos los resultados que obtenemos.

Nuestro documento html contendrá el siguiente texto de partida (en este caso se ha llamado ejemplo.html):

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo aplicación CSS - aprenderaprogramar.com </title>
  </head>
  <body>
    <p>Texto de ejemplo para visualizar resultados </p>
  </body>
</html>
```

Una vez hemos visualizado este ejemplo tenemos una página web que tan solo tiene un párrafo ("Texto de ejemplo para visualizar resultados"). Este aparece en color negro por defecto y nosotros, para ver la utilidad de CSS deseamos mostrar el texto en rojo. En realidad, con CSS podemos hacer cosas mucho más complejas, pero solo queremos poner un ejemplo para mostrar la utilidad de CSS.

Selectores

Para ello vamos a crear nuestro archivo de hojas de estilo CSS que llamaremos estilos.css, que crearemos en el mismo directorio donde tengamos el archivo ejemplo.html y que contendrá lo siguiente:

```
p {color:red;}
```

```
.azul{color:blue;}
.roja{color:red;}
#postdata{color:green;font-size:10px;}
```

Diseño web aplicando estilos con css

Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. El siguiente ejemplo selecciona todos los párrafos de la página:

```
p {  
  
...  
}
```

Para utilizar este selector, solamente es necesario indicar el nombre de una etiqueta HTML (sin los caracteres `<` y `>`) correspondiente a los elementos que se quieren seleccionar.

El siguiente ejemplo aplica diferentes estilos a los titulares y a los párrafos de una página HTML:

```
h1 {  
  color: red;  
}  
  
h2 {  
  color: blue;  
}  
  
p {  
  color: black;  
}
```

Si se quiere aplicar los mismos estilos a dos etiquetas diferentes, se pueden encadenar los selectores. En el siguiente ejemplo, los títulos de sección h1, h2 y h3 comparten los mismos estilos:

```
h1 {  
  color: #8A8E27;  
  font-weight: normal;  
  font-family: Arial, Helvetica, sans-serif;  
}  
h2 {  
  color: #8A8E27;  
  font-weight: normal;  
  font-family: Arial, Helvetica, sans-serif;  
}  
h3 {  
  color: #8A8E27;  
  font-weight: normal;  
  font-family: Arial, Helvetica, sans-serif;  
}
```

En este caso, CSS permite agrupar todas las reglas individuales en una sola regla con un selector múltiple. Para ello, se incluyen todos los selectores separados por una coma (,) y el resultado es que la siguiente regla CSS es equivalente a las tres reglas anteriores:

```
h1, h2, h3 {  
  color: #8A8E27;  
  font-weight: normal;  
  font-family: Arial, Helvetica, sans-serif;  
}
```

En las hojas de estilo complejas, es habitual agrupar las propiedades comunes de varios elementos en una única regla CSS y posteriormente definir las propiedades específicas de esos mismos elementos. El siguiente ejemplo establece en primer lugar las propiedades comunes de los títulos de sección (color y tipo de letra) y a continuación, establece el tamaño de letra de cada uno de ellos:

```
h1, h2, h3 {  
  color: #8A8E27;  
  font-weight: normal;  
  font-family: Arial, Helvetica, sans-serif;  
}  
  
h1 { font-size: 2em; }  
h2 { font-size: 1.5em; }  
h3 { font-size: 1.2em; }
```

Veremos un diseño practico



Este es el código


```
<html>
<head>
  <title>Ejemplo pagina web</title>
</head>
<body>
  <div id="cont">
    <div id="encabezado">
      <h1>Salamanca Fotografias</h1>
      <h2>Portafolio Personal</h2>
    </div>
    <div id="contenido">
      <h2>BIENVENIDOS <span>A Mi Sitio Web</span></h2>
      <p>Soy un amante de la fotografia, en mi tiempo libre me gusta salir con mi camara a capturar esos momentos e imagenes que quiero que queden en mis recuerdos.</p>
      <p>A lo largo de este sitio encontraras las mejores fotografias que he tomado en los ultimos años, podras ver fotos de Paisajes, Animales y Gente. Te invito a que recorras mi sitio web y disfrutes tanto como yo del arte de la fotografia</p>
    </div>
    <div id="menu">
      <h3>Menu</h3>
      <ul>
        <li><a>Home</a></li>
        <li><a>Paisajes</a></li>
        <li><a>Animales</a></li>
        <li><a>Gente</a></li>
        <li><a>Contacto</a></li>
      </ul>
    </div>
    <div id="pie">
      <h2>ULTIMAS <span>fotografias tomadas</span></h2>
      <div id="fotos">
        <div id="foto1">Fotografia 1</div>
        <div id="foto2">Fotografia 2</div>
        <div id="foto3">Fotografia 3</div>
        <div id="foto4">Fotografia 4</div>
        <div id="foto5">Fotografia 5</div>
        <div id="foto6">Fotografia 6</div>
      </div>
      <p>Todos los derechos reservados // <a>Curso de HTML / CSS</a> <a href="http://www.guachipedia.com">guachipedia.com</a></p>
    </div>
  </div>
</body>
</html>
```

Ahora tendremos el código CSS

```
a, Arial, Geneva, sans-serif; color:#797979;}
#cont{ width: 693px; margin: 0 auto 0 auto; text-align: left;}
#encabezado{ height: 32px; margin-bottom: 40px; width: 693px;}
/*los textos del encabezado lo dejamos para mas adelante
porque tenemos que hacer reemplazo de texto por imagen*/
#contenido{ width: 500px; float: left; margin-bottom: 70px;}

#contenido h2{ font-size: 18px; color: #464545; font-weight: normal }
#contenido h2 span{ color: #a7a7a7; } /*esto nos sirve para cambiar
el color a gris en los titulos cuando colocamos la etiqueta span!*/
#contenido p{font-size: 13px;}

#menu{ width: 180px; float: left; text-transform: uppercase}
#pie{ clear: both; height: 296px;}
#pie h2{ font-size: 18px; color: #464545; font-weight: normal }
#pie h2 span{ color: #a7a7a7; }
#pie p{ text-align: center; font-size: 13px; padding-top: 10px; clear: both;}
```

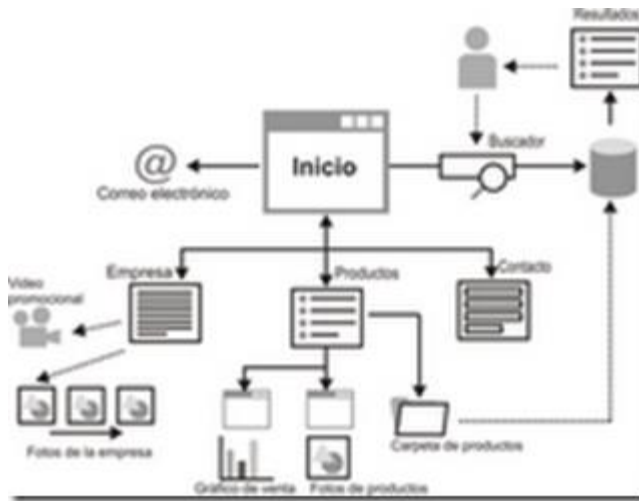
Fundamentos de la maquetación

Prototipado

Introducción al prototipado

El Prototipado consiste en planos y maquetas antes de construir el sitio web.

a) Los planos



Los planos se denominan igualmente Blueprint, diagramas de contenido, diagramas de flujo o mapa web.

Lo más importante es que sea comprensible (estructura, flujo de navegación, relación entre los elementos). Además, conviene incluir una leyenda para mejorar la comprensión.

b) Las maquetas

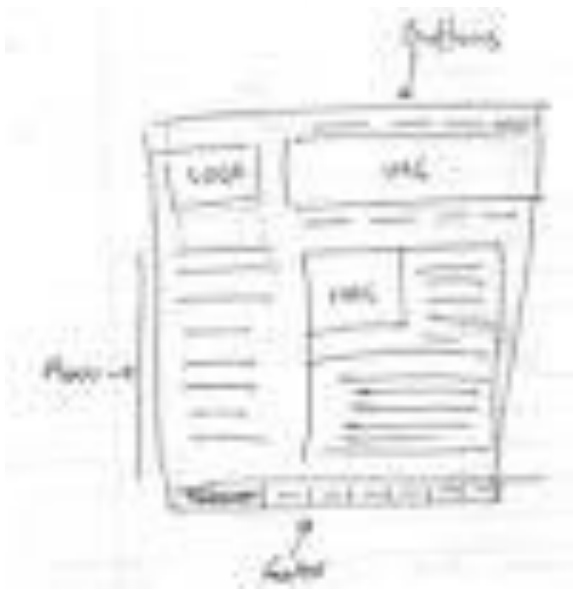
Las maquetas son diagramas de presentación para crear una referencia visual de la estructura, organización e interacción a nivel de página.

No se incluye diseño gráfico en el prototipo.

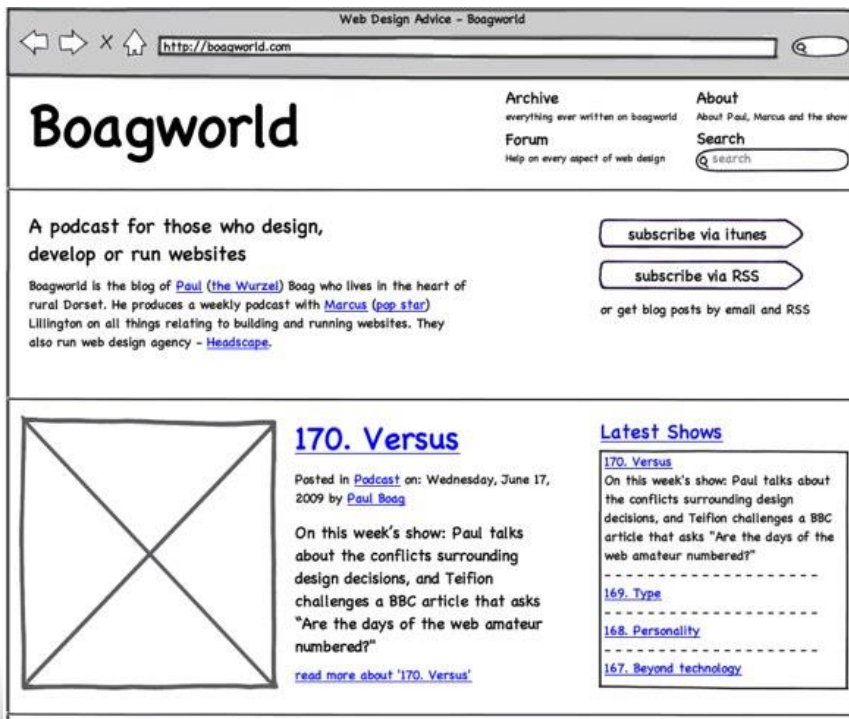
- Los prototipos de baja fidelidad

Son dibujos estáticos.

→ el Sketching (bocetos informales)



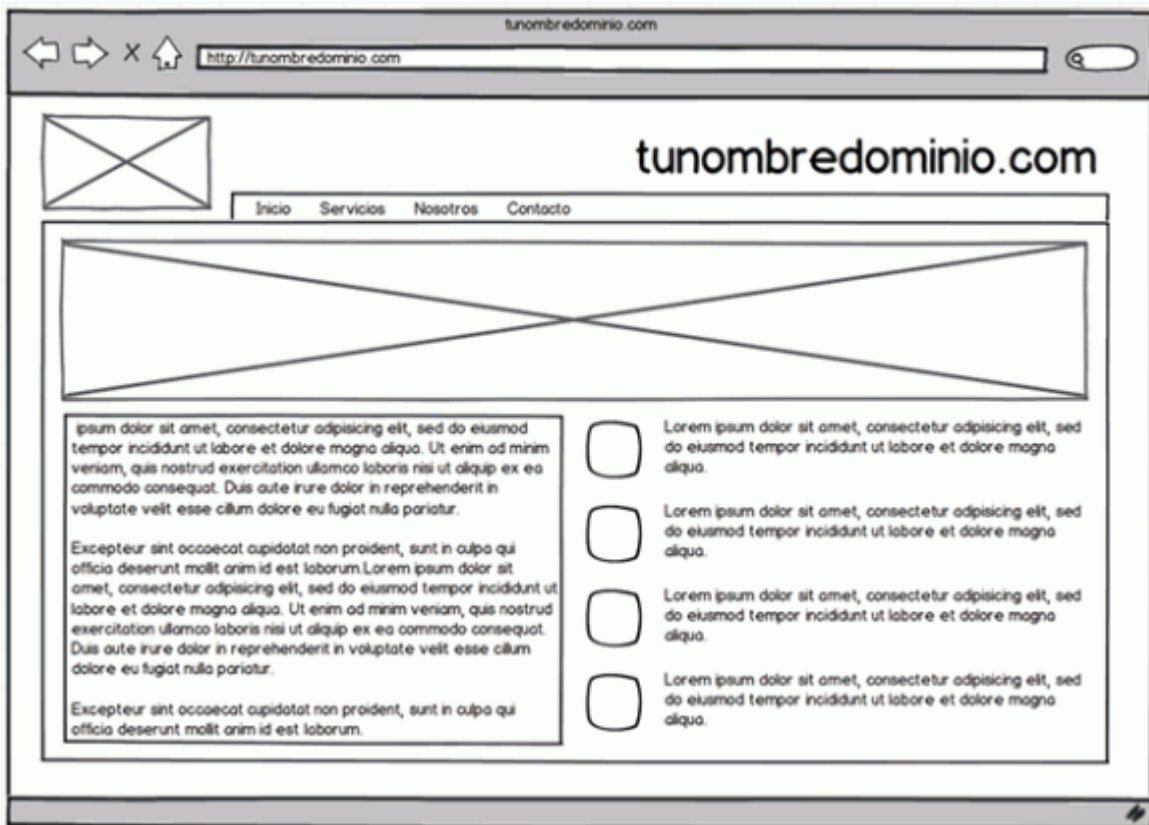
- ➔ el Wireframe es más elaborado e incluye el inventariado de contenido, es decir el contenido presente en cada página, los elementos de la página (cabeceras, enlaces, listas, formularios, etc.), el etiquetado de los vínculos o de los títulos, el layout (colocación, ubicación, agrupación de los elementos de la página, la estrategia de navegación y la priorización de contenidos)



➔ **Storyboards.** Cuando tenemos una secuencia Wireframe

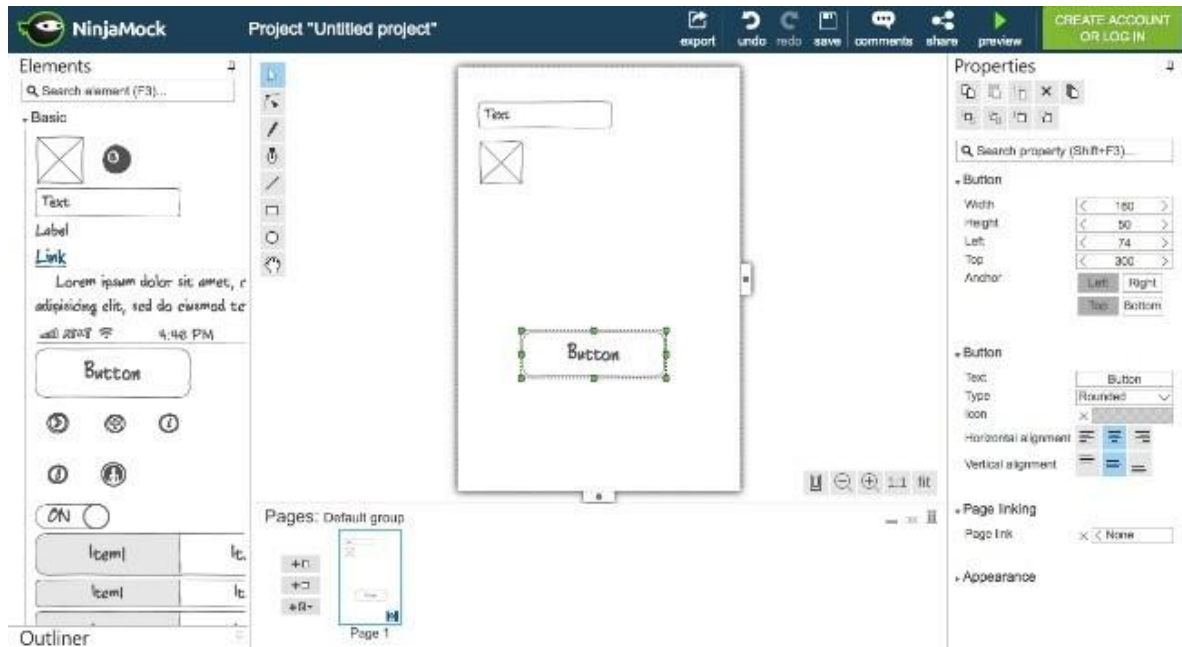


➔ **MockUps**



Manejo de ninjamock

herramienta online donde los elementos que incorporamos ofrecen un acabo de dibujo a mano alzada. Su principal ventaja es que es completamente gratuita. La interacción de sus prototipos no va más allá de enlaces entre pantallas y pantallas, pero bueno. Para ideas básicas, para un primer esbozo resulta eficiente.



Modelos de caja

El modelo de cajas o "box model" es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El modelo de cajas es el comportamiento de CSS que hace que todos los elementos de las páginas se representen mediante cajas rectangulares.

Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento. La siguiente imagen muestra las tres cajas rectangulares que crean las tres etiquetas HTML que incluye la página:

Las cajas se crean automáticamente al definir cada elemento HTML

<code><p>Párrafo de texto con algunas palabras resaltadas</p></code>	Párrafo de texto con algunas palabras resaltadas
<code><p>Otro párrafo</p></code>	Otro párrafo

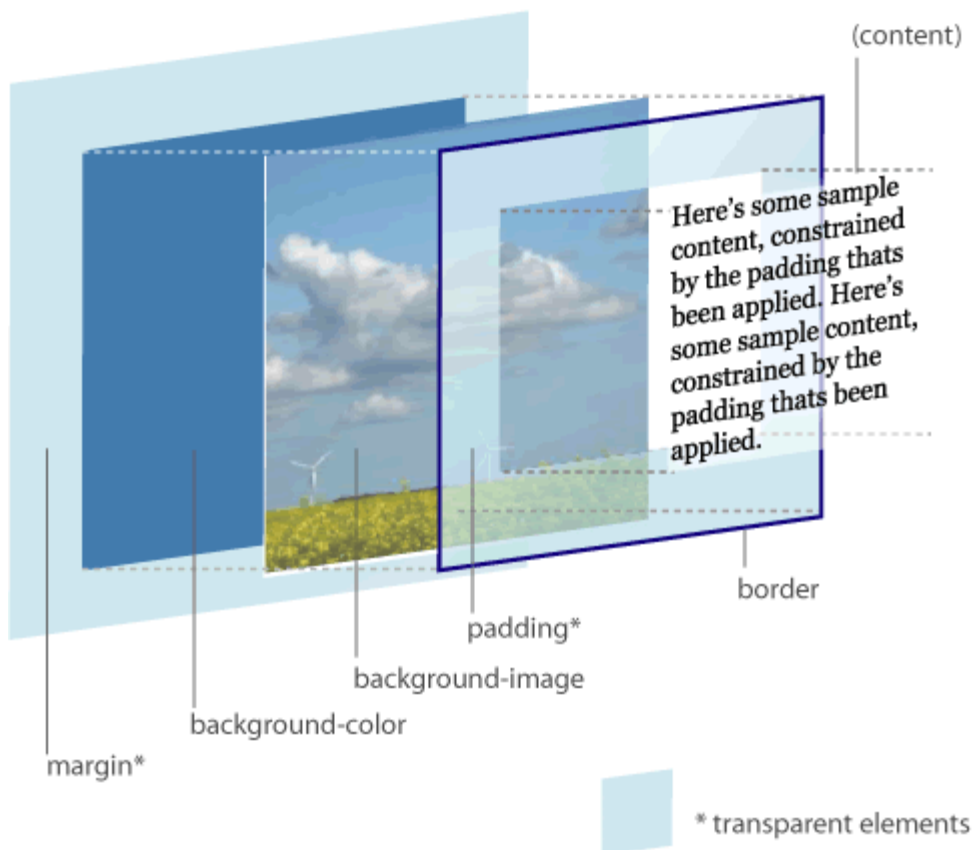
Las cajas de las páginas no son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde. La siguiente imagen muestra las cajas que forman la página web de <http://www.alistapart.com/> después de forzar a que todas las cajas muestren su borde:



Cajas que forman la página alistapart.comn al modelo de caja

Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características. Cada una de las cajas está formada por seis partes, tal y como muestra la siguiente imagen:

THE CSS BOX MODEL HIERARCHY



Posicionamiento

Posicionamiento en CSS

Posicionamiento Float

Dimensiones

Uso de las dimensiones en css

Anchura

La propiedad CSS que controla la anchura de la caja de los elementos se denomina width.

Propiedad	width
-----------	-------

Propiedad	width
Valores	unidad de medida porcentaje auto inherit
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las filas de tabla y los grupos de filas de tabla
Valor inicial	auto
Descripción	Establece la anchura de un elemento

La propiedad width no admite valores negativos y los valores en porcentaje se calculan a partir de la anchura de su elemento padre. El valor inherit indica que la anchura del elemento se hereda de su elemento padre. El valor auto, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la anchura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

El siguiente ejemplo establece el valor de la anchura del elemento <div> lateral:

```
#lateral { width: 200px;

<div id="lateral">
  ...
</div>
```

CSS define otras dos propiedades relacionadas con la anchura de los elementos: min-width y max-width, que se verán más adelante.

Altura

La propiedad CSS que controla la altura de los elementos se denomina height.

Propiedad	height
Valores	unidad de medida porcentaje auto inherit

Propiedad	height
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las columnas de tabla y los grupos de columnas de tabla
Valor inicial	auto
Descripción	Establece la altura de un elemento

Al igual que sucede con width, la propiedad height no admite valores negativos. Si se indica un porcentaje, se toma como referencia la altura del elemento padre. Si el elemento padre no tiene una altura definida explícitamente, se asigna el valor auto a la altura.

El valor inherit indica que la altura del elemento se hereda de su elemento padre. El valor auto, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la altura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

El siguiente ejemplo establece el valor de la altura del elemento <div> de cabecera:

```
#cabecera { height: 60px; }
```

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

```
...
```

```
</div>
```

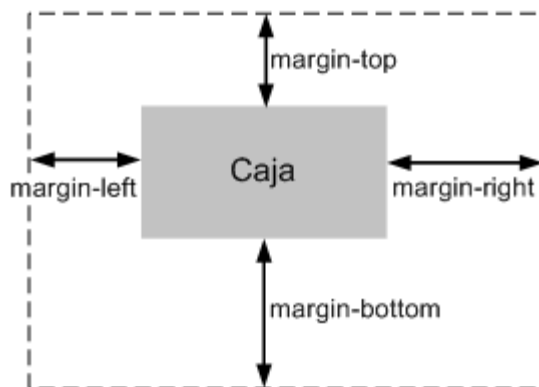
Margen

CSS define cuatro propiedades para controlar cada uno de los márgenes horizontales y verticales de un elemento.

Propiedades	margin-top, margin-right, margin-bottom, margin-left
-------------	---

Propiedades	margin-top, margin-right, margin-bottom, margin-left
Valores	unidad de medida porcentaje auto inherit
Se aplica a	Todos los elementos, salvo margin-top y margin-bottom que sólo se aplican a los elementos de bloque y a las imágenes
Valor inicial	0
Descripción	Establece cada uno de los márgenes horizontales y verticales de un elemento

Cada una de las propiedades establece la separación entre el borde lateral de la caja y el resto de cajas adyacentes:



Las cuatro propiedades relacionadas con los márgenes

Las unidades más utilizadas para indicar los márgenes de un elemento son los píxeles (cuando se requiere una precisión total), los em (para hacer diseños que mantengan las proporciones) y los porcentajes (para hacer diseños líquidos o fluidos).

El siguiente ejemplo añade un margen izquierdo al segundo párrafo:

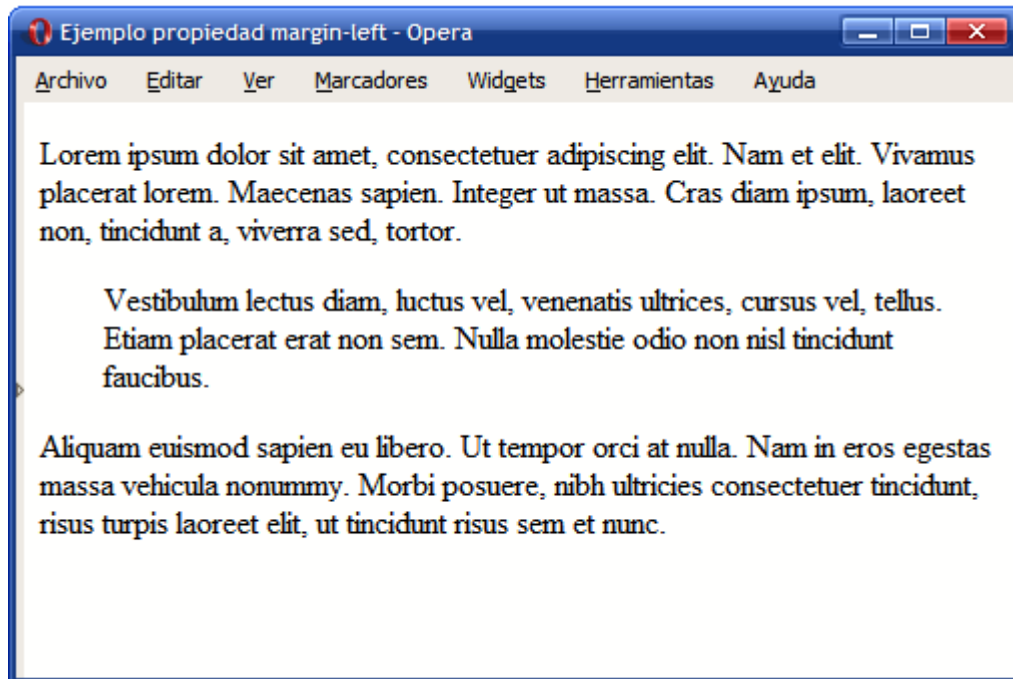
```
.destacado {
  margin-left: 2em;
}

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam et elit. Vivamus placerat lorem. Maecenas sapien. Integer ut massa. Cras diam ipsum, laoreet non, tincidunt a, viverra sed, tortor.</p>

<p class="destacado">Vestibulum lectus diam, luctus vel, venenatis ultrices, cursus vel, tellus. Etiam placerat erat non sem. Nulla molestie odio non nisl tincidunt faucibus.</p>

<p>Aliquam euismod sapien eu libero. Ut tempor orci at nulla. Nam in eros egestas massa vehicula nonummy. Morbi posuere, nibh ultricies consectetur tincidunt, risus turpis laoreet elit, ut tincidunt risus sem et nunc.</p>
```

A continuación, se muestra el aspecto del ejemplo anterior en cualquier navegador:

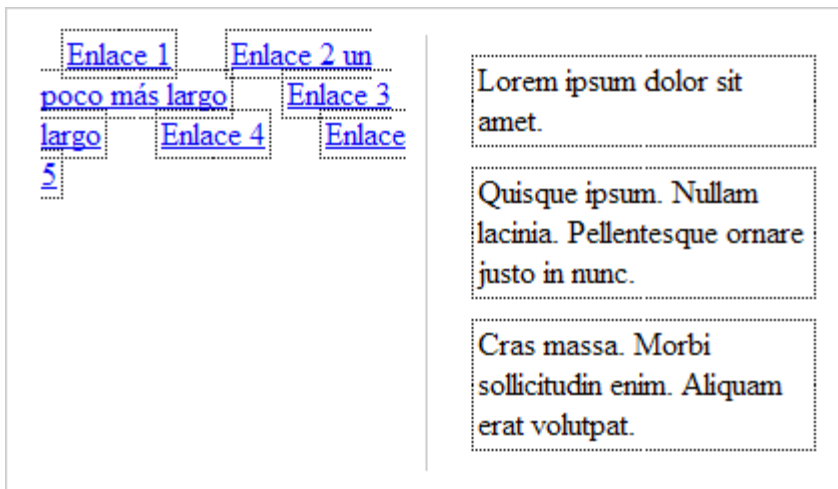


Ejemplo de propiedad margin-left

Algunos diseñadores web utilizan la etiqueta `<blockquote>` para tabular los contenidos de los párrafos. Se trata de un error grave porque HTML no debe utilizarse para controlar el aspecto de los elementos. CSS es el único responsable de establecer el estilo de los elementos, por lo que en vez de utilizar la etiqueta `<blockquote>` de HTML, debería utilizarse la propiedad `margin-left` de CSS.

Los márgenes verticales (`margin-top` y `margin-bottom`) sólo se pueden aplicar a los elementos de bloque y las imágenes, mientras que los márgenes laterales (`margin-`

left y margin-right) se pueden aplicar a cualquier elemento, tal y como muestra la siguiente imagen:



Los márgenes verticales sólo se aplican a los elementos de bloque e imágenes

La imagen anterior muestra el resultado de aplicar los mismos márgenes a varios enlaces (elementos en línea) y varios párrafos (elementos de bloque). En los elementos en línea los márgenes verticales no tienen ningún efecto, por lo que los enlaces no muestran ninguna separación vertical, al contrario de lo que sucede con los párrafos. Sin embargo, los márgenes laterales funcionan sobre cualquier tipo de elemento, por lo que los enlaces se muestran separados entre sí y los párrafos aumentan su separación con los bordes laterales de su elemento contenedor.

Además de las cuatro propiedades que controlan cada uno de los márgenes del elemento, CSS define una propiedad especial que permite establecer los cuatro márgenes de forma simultánea. Estas propiedades especiales se denominan "*propiedades shorthand*" y CSS define varias propiedades de este tipo, como se verá más adelante.

La propiedad que permite definir de forma simultánea los cuatro márgenes se denomina margin.

Propiedad	margin
Valores	(unidad de medida porcentaje auto) {1, 4} inherit

Propiedad	margin
Se aplica a	Todos los elementos salvo algunos casos especiales de elementos mostrados como tablas
Valor inicial	-
Descripción	Establece de forma directa todos los márgenes de un elemento

La notación {1, 4} de la definición anterior significa que la propiedad `margin` admite entre uno y cuatro valores, con el siguiente significado:

Si solo se indica un valor, todos los márgenes tienen ese valor.

Si se indican dos valores, el primero se asigna al margen superior e inferior y el segundo se asigna a los márgenes izquierdo y derecho.

Si se indican tres valores, el primero se asigna al margen superior, el tercero se asigna al margen inferior y el segundo valor se asigna los márgenes izquierdo y derecho.

Si se indican los cuatro valores, el orden de asignación es: margen superior, margen derecho, margen inferior y margen izquierdo.

El ejemplo anterior de márgenes se puede reescribir utilizando la propiedad `margin`:

Código CSS original:

```
div img {
  margin-top: .5em;
  margin-bottom: .5em;
  margin-left: 1em;
  margin-right: .5em;
}
```

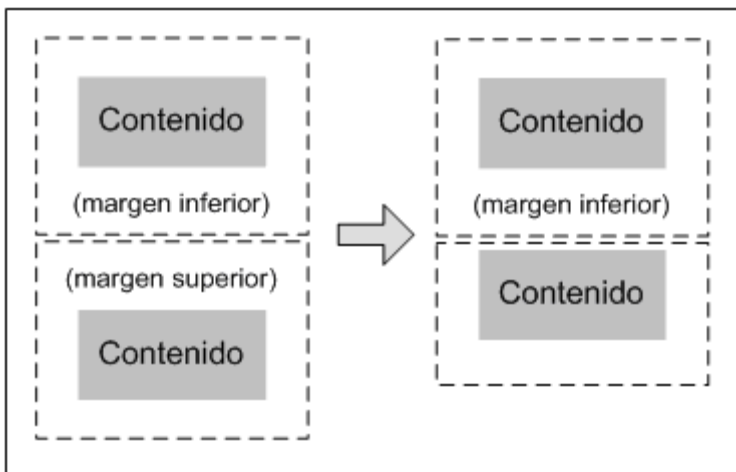
Alternativa directa:

```
div img {  
  margin: .5em .5em .5m 1em;  
}
```

Otra alternativa:

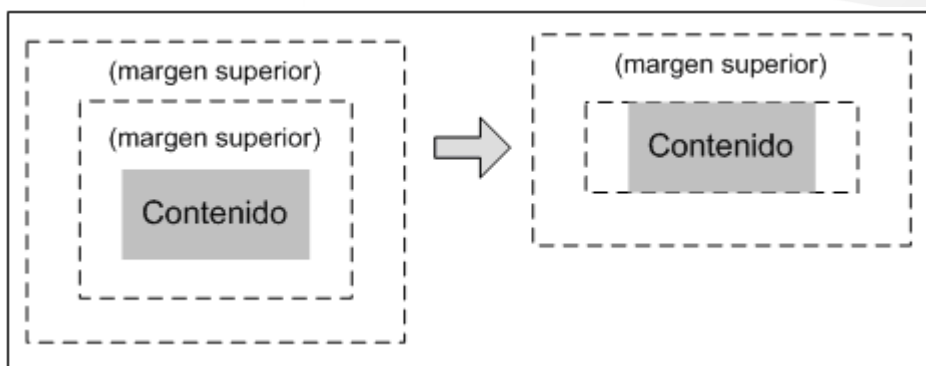
```
div img {  
  margin: .5em;  
  margin-left: 1em;  
}
```

El comportamiento de los márgenes verticales es más complejo de lo que se puede imaginar. Cuando se juntan dos o más márgenes verticales, se fusionan de forma automática y la altura del nuevo margen será igual a la altura del margen más alto de los que se han fusionado.



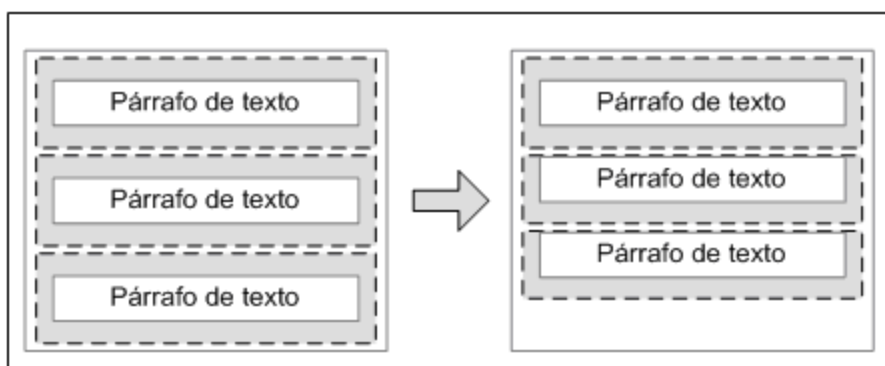
Fusión automática de los márgenes verticales

De la misma forma, si un elemento está contenido dentro de otro elemento, sus márgenes verticales se fusionan y resultan en un nuevo margen de la misma altura que el mayor margen de los que se han fusionado:



Fusión de los márgenes de los elementos interiores

Aunque en principio puede parecer un comportamiento extraño, la razón por la que se propuso este mecanismo de fusión automática de márgenes verticales es el de dar uniformidad a las páginas web habituales. En una página con varios párrafos, si no se diera este comportamiento y se estableciera un determinado margen a todos los párrafos, el primer párrafo no mostraría un aspecto homogéneo respecto de los demás.



Motivo por el que se fusionan automáticamente los márgenes verticales

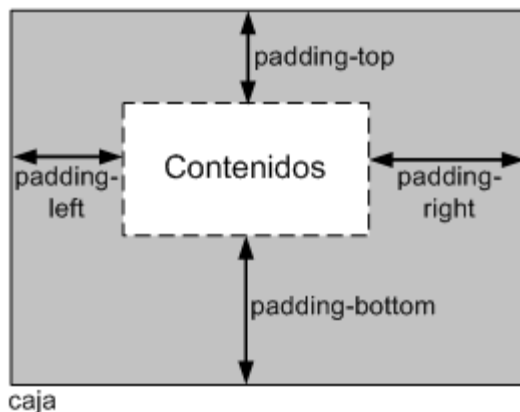
En el caso de un elemento que se encuentra en el interior de otro y sus márgenes se fusionan de forma automática, se puede evitar este comportamiento añadiendo un pequeño relleno (`padding: 1px`) o un borde (`border: 1px solid transparent`) al elemento contenedor.

Relleno

CSS define cuatro propiedades para controlar cada uno de los espacios de relleno horizontales y verticales de un elemento.

Propiedades	padding-top, padding-right, padding-bottom, padding-left
Valores	unidad de medida porcentaje inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	0
Descripción	Establece cada uno de los rellenos horizontales y verticales de un elemento

Cada una de estas propiedades establece la separación entre el contenido y los bordes laterales de la caja del elemento:



Las cuatro propiedades relacionadas con los rellenos

Como sucede con los márgenes, CSS también define una propiedad de tipo "shorthand" llamada **padding** para establecer los cuatro rellenos de un elemento de forma simultánea.

Propiedad	padding
Valores	(unidad de medida porcentaje) {1, 4} inherit

Propiedad	padding
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	-
Descripción	Establece de forma directa todos los rellenos de los elementos

La notación {1, 4} de la definición anterior significa que la propiedad padding admite entre uno y cuatro valores, con el mismo significado que el de la propiedad margin. Ejemplo:

```
body {padding: 2em} /* Todos los rellenos valen 2em */
body {padding: 1em 2em} /* Superior e inferior = 1em, Izquierdo y derecho = 2em */
body {padding: 1em 2em 3em} /* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 2em */
body {padding: 1em 2em 3em 4em} /* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 4em */
```

Bordes

CSS permite modificar el aspecto de cada uno de los cuatro bordes de la caja de un elemento. Para cada borde se puede establecer su anchura o grosor, su color y su estilo, por lo que en total CSS define 20 propiedades relacionadas con los bordes.

Anchura

La anchura de los bordes se controla con las cuatro propiedades siguientes:

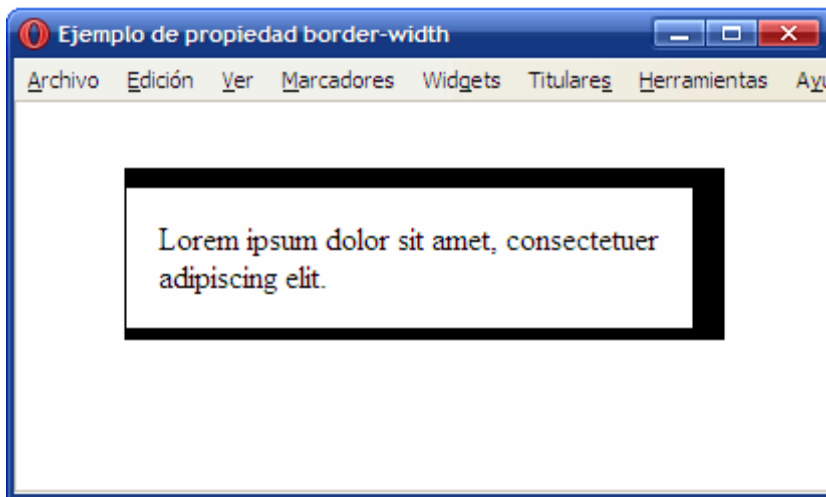
Propiedades	border-top-width, border-right-width, border-bottom-width, border-left-width
Valores	(unidad de medida thin medium thick) inherit
Se aplica a	Todos los elementos
Valor inicial	Medium

Propiedades	border-top-width, border-right-width, border-bottom-width, border-left-width
Descripción	Establece la anchura de cada uno de los cuatro bordes de los elementos

La anchura de los bordes se indica mediante una medida (en cualquier unidad de medida absoluta o relativa) o mediante las palabras clave thin (borde delgado), medium (borde normal) y thick (borde ancho).

La unidad de medida más habitual para establecer el grosor de los bordes es el píxel, ya que es la que permite un control más preciso sobre el grosor. Las palabras clave apenas se utilizan, ya que el estándar CSS no indica explícitamente el grosor al que equivale cada palabra clave, por lo que pueden producirse diferencias visuales entre navegadores. Así, por ejemplo, el grosor medium equivale a 4px en algunas versiones de Internet Explorer y a 3px en el resto de navegadores.

El siguiente ejemplo muestra un elemento con cuatro anchuras diferentes de borde:



Ejemplo de propiedad border-width

Las reglas CSS utilizadas se muestran a continuación:

```
div {
  border-top-width: 10px;
  border-right-width: 1em;
  border-bottom-width: thick;
  border-left-width: thin;
}
```

Si se quiere establecer de forma simultánea la anchura de todos los bordes de una caja, es necesario utilizar una propiedad "*shorthand*" llamada `border-width`:

Propiedad	border-width
Valores	(unidad de medida thin medium thick) {1, 4} inherit
Se aplica a	Todos los elementos
Valor inicial	Medium
Descripción	Establece la anchura de todos los bordes del elemento

La propiedad `border-width` permite indicar entre uno y cuatro valores. El significado de cada caso es el habitual de las propiedades "*shorthand*":

```
p { border-width: thin } /* thin thin thin thin */
p { border-width: thin thick } /* thin thick thin thick */
p { border-width: thin thick medium } /* thin thick medium thick */
p { border-width: thin thick medium thin } /* thin thick medium thin */
```

Si se indica un solo valor, se aplica a los cuatro bordes. Si se indican dos valores, el primero se aplica al borde superior e inferior y el segundo valor se aplica al borde izquierdo y derecho.

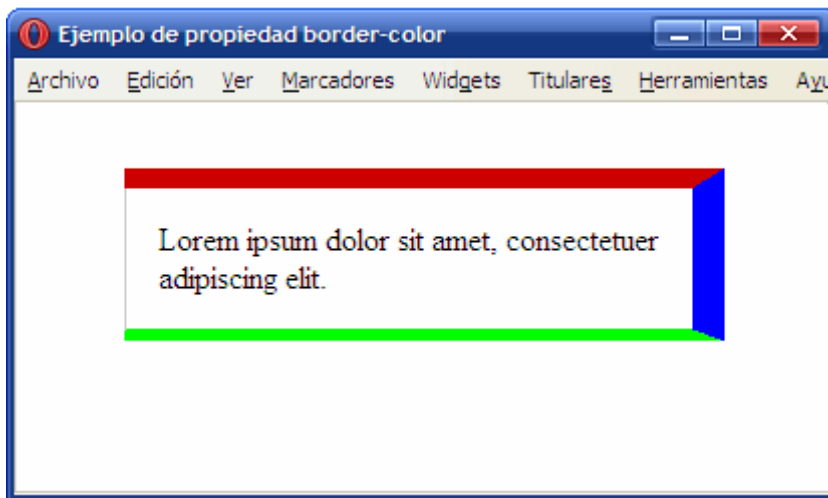
Si se indican tres valores, el primero se aplica al borde superior, el segundo se aplica al borde izquierdo y derecho y el tercer valor se aplica al borde inferior. Si se indican los cuatro valores, el orden de aplicación es superior, derecho, inferior e izquierdo.

Color

El color de los bordes se controla con las cuatro propiedades siguientes:

Propiedades	border-top-color, border-right-color, border-bottom-color, border-left-color
Valores	color transparent inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de cada uno de los cuatro bordes de los elementos

El ejemplo anterior se puede modificar para mostrar cada uno de los bordes de un color diferente:



Ejemplo de propiedad border-color

Las reglas CSS necesarias para mostrar los colores anteriores son las siguientes:

```
div {
  border-top-color: #CC0000;
  border-right-color: blue;
  border-bottom-color: #00FF00;
  border-left-color: #CCC;
}
```

CSS incluye una propiedad "*shorthand*" llamada `border-color` para establecer de forma simultánea el color de todos los bordes de una caja:

Propiedad	border-color
Valores	(color transparent) {1, 4} inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de todos los bordes del elemento

En este caso, al igual que sucede con la propiedad `border-width`, es posible indicar de uno a cuatro valores y las reglas de aplicación son idénticas a las de la propiedad `border-width`.

Estilo

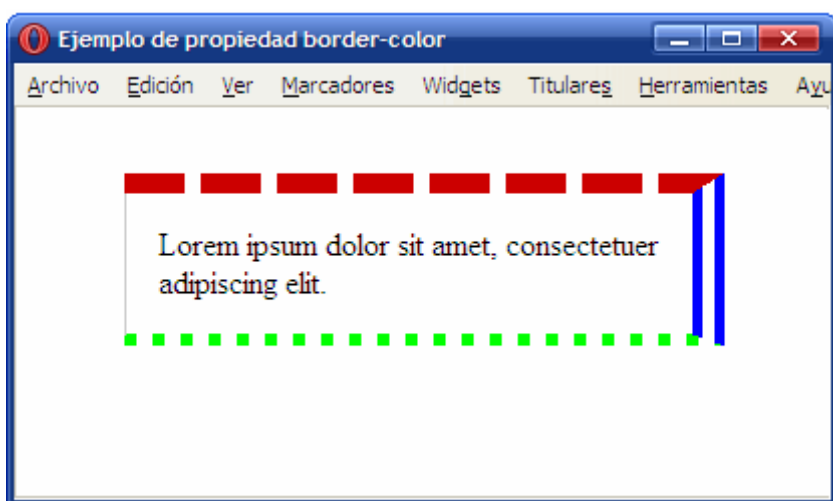
Por último, CSS permite establecer el estilo de cada uno de los bordes mediante las siguientes propiedades:

Propiedades	border-top-style, border-right-style, border-bottom-style, border-left-style
Valores	none hidden dotted dashed solid double groove ridge inset outset inherit
Se aplica a	Todos los elementos

Propiedades	border-top-style, border-right-style, border-bottom-style, border-left-style
Valor inicial	none
Descripción	Establece el estilo de cada uno de los cuatro bordes de los elementos

El estilo de los bordes sólo se puede indicar mediante alguna de las palabras reservadas definidas por CSS. Como el valor por defecto de esta propiedad es none, los elementos no muestran ningún borde visible a menos que se establezca explícitamente un estilo de borde.

Siguiendo el ejemplo anterior, se puede modificar el estilo de cada uno de los bordes:

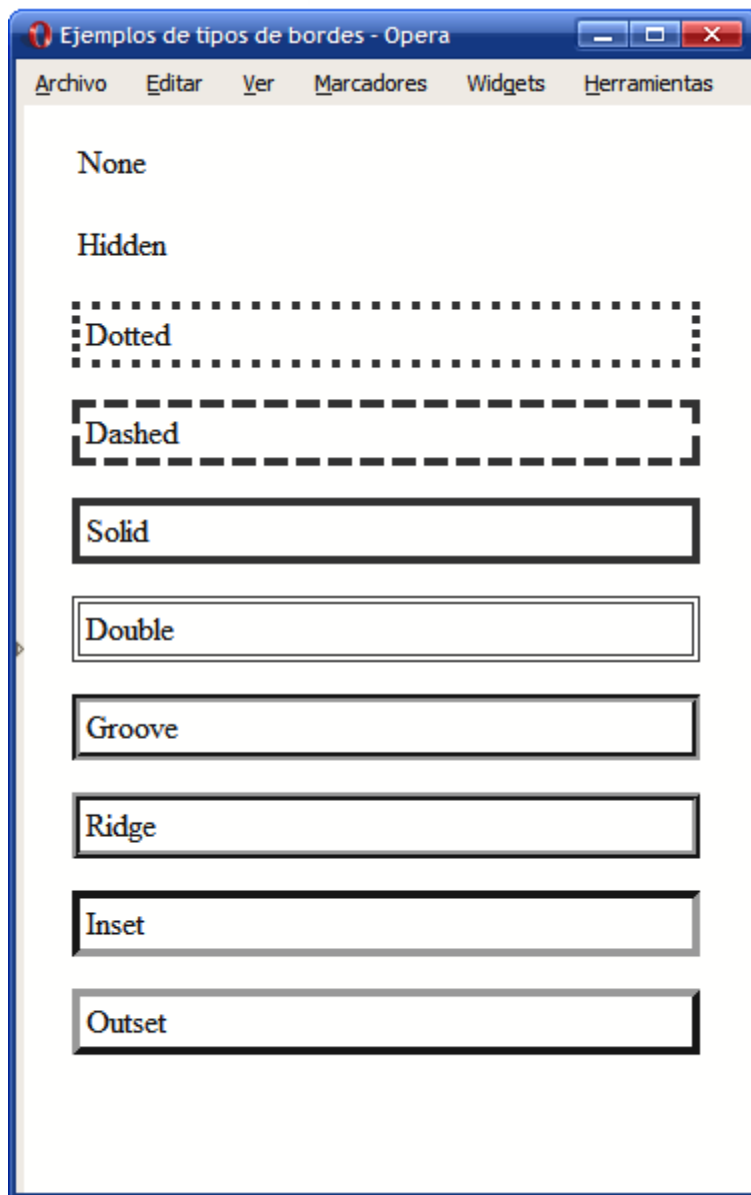


Ejemplo de propiedad border-style

Las reglas CSS necesarias para mostrar los estilos anteriores son las siguientes:

```
div {
  border-top-style: dashed;
  border-right-style: double;
  border-bottom-style: dotted;
  border-left-style: solid;
}
```

El aspecto con el que los navegadores muestran los diferentes tipos de borde se muestra a continuación:



Tipos de bordes definidos por CSS

Los bordes más utilizados son `solid` y `dashed`, seguidos de `double` y `dotted`. Los estilos `none` y `hidden` son idénticos visualmente, pero se diferencian en la forma que los navegadores resuelven los conflictos entre los bordes de las celdas adyacentes en las tablas.

Para establecer de forma simultánea los estilos de todos los bordes de una caja, es necesario utilizar la propiedad "*shorthand*" llamada `border-style`:

Propiedad	border-style
Valores	(none hidden dotted dashed solid double groove ridge inset outset) {1, 4} inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo de todos los bordes del elemento

Como es habitual, la propiedad permite indicar de uno a cuatro valores diferentes y las reglas de aplicación son las habituales de las propiedades *"shorthand"*.

Propiedades shorthand

Como sucede con los márgenes y los rellenos, CSS define una serie de propiedades de tipo *"shorthand"* que permiten establecer todos los atributos de los bordes de forma simultánea. CSS incluye una propiedad *"shorthand"* para cada uno de los cuatro bordes y una propiedad *"shorthand"* global.

Propiedades	border-top, border-right, border-bottom, border-left
Valores	(unidad de medida_borde color_borde estilo_borde) inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de cada uno de los cuatro bordes de los elementos

El significado de cada uno de los valores especiales es el siguiente:

<medida_borde>: una medida CSS o alguna de las siguientes palabras clave: thin, medium, thick.

<color_borde>: un color de CSS o la palabra clave transparent

<estilo_borde>: una de las siguientes palabras clave: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset.

Las propiedades "*shorthand*" permiten establecer alguno o todos los atributos de cada borde. El siguiente ejemplo establece el color y el tipo del borde inferior, pero no su anchura:

```
h1 {
  border-bottom: solid red;
}
```

En el ejemplo anterior, la anchura del borde será la correspondiente al valor por defecto (medium). Este otro ejemplo muestra la forma habitual utilizada para establecer el estilo de cada borde:

```
div {
  border-top: 1px solid #369;
  border-bottom: 3px double #369;
}
```

Por último, CSS define una propiedad de tipo "*shorthand*" global para establecer el valor de todos los atributos de todos los bordes de forma directa:

Propiedad	border
Valores	(unidad de medida_borde color_borde estilo_borde) inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de todos los bordes de los elementos

Las siguientes reglas CSS son equivalentes:

```
div {  
  border-top: 1px solid red;  
  border-right: 1px solid red;  
  border-bottom: 1px solid red;  
  border-left: 1px solid red;  
}  
  
div { border: 1px solid red; }
```

Como el valor por defecto de la propiedad `border-style` es `none`, si una propiedad *shorthand* no establece explícitamente el estilo de un borde, el elemento no muestra ese borde:

```
/* Sólo se establece el color, por lo que el estilo es  
   "none" y el borde no se muestra */  
div { border: red; }  
  
/* Se establece el grosor y el color del borde, pero no  
   su estilo, por lo que es "none" y el borde no se muestra */  
div { border-bottom: 5px blue; }
```

Cuando los cuatro bordes no son idénticos, pero sí muy parecidos, se puede utilizar la propiedad `border` para establecer de forma directa los atributos comunes de todos los bordes y posteriormente especificar para cada uno de los cuatro bordes sus propiedades particulares:

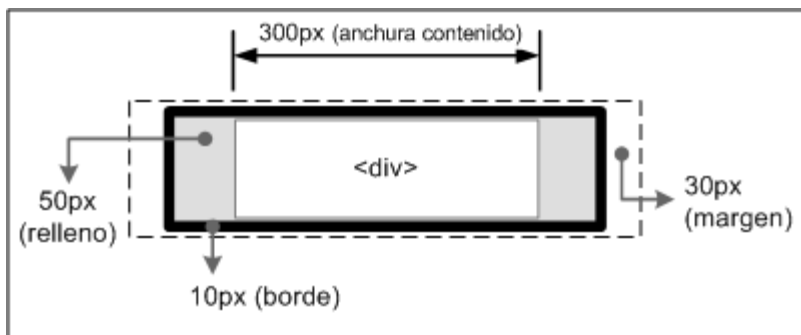
```
h1 {  
  border: solid #000;  
  border-top-width: 6px;  
  border-left-width: 8px;  
}
```

Margen, relleno, bordes y modelo de cajas

La anchura y altura de un elemento no solamente se calculan teniendo en cuenta sus propiedades `width` y `height`. El margen, el relleno y los bordes establecidos a un elemento determinan la anchura y altura final del elemento. En el siguiente ejemplo se muestran los estilos CSS de un elemento:

```
div {
  width: 300px;
  padding-left: 50px;
  padding-right: 50px;
  margin-left: 30px;
  margin-right: 30px;
  border: 10px solid black;
}
```

La anchura total con la que se muestra el elemento no son los 300 píxel indicados en la propiedad `width`, sino que también se añaden todos sus márgenes, rellenos y bordes:



La anchura total de un elemento tiene en cuenta los márgenes, rellenos y bordes

De esta forma, la anchura del elemento en pantalla sería igual a la suma de la anchura original, los márgenes, los bordes y los rellenos:

$$30\text{px} + 10\text{px} + 50\text{px} + 300\text{px} + 50\text{px} + 10\text{px} + 30\text{px} = 480 \text{ píxel}$$

Así, la anchura/altura establecida con CSS siempre hace referencia a la anchura/altura del contenido. La anchura/altura total del elemento debe tener en cuenta además los valores del resto de partes que componen la caja del *box model*.

Por otra parte, la guerra de navegadores que se produjo en los años 90 provocó que cada fabricante (Microsoft y Netscape) añadiera sus propias extensiones y mejoras en sus productos. Posteriormente, aparecieron los estándares publicados por el W3C y los fabricantes se encontraron con el problema de la incompatibilidad entre sus implementaciones anteriores de HTML y CSS y las implementaciones que requerían los estándares.

La solución que adoptaron fue la de incluir en el navegador dos modos diferentes de funcionamiento: modo compatible con las páginas antiguas (denominado "*modo quirks*" y que se podría traducir como "*modo raro*") y modo compatible con los nuevos estándares (denominado "*modo estándar*"). El modo *quirks* es equivalente a la forma en la que se visualizaban las páginas en los navegadores Internet Explorer 4 y Netscape Navigator 4.

La diferencia más notable entre los dos modos es el tratamiento del "*box model*", lo que puede afectar gravemente al diseño de las páginas HTML. Los navegadores seleccionan automáticamente el modo en el que muestran las páginas en función del DOCTYPE definido por el documento. En general, los siguientes tipos de DOCTYPE activan el modo *quirks* en los navegadores:

No utilizar ningún DOCTYPE

DOCTYPE anterior a HTML 4.0 (<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">)

DOCTYPE de HTML 4.01 sin URL (<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">)

En el caso concreto de Internet Explorer, también activan el modo quirks los modos XHTML 1.0 que incluyen la declaración de XML (por ejemplo <?xml version="1.0" encoding="UTF-8"?>) al principio de la página web:

```
<?xml version="1.0" encoding="UTF-8"?>
```

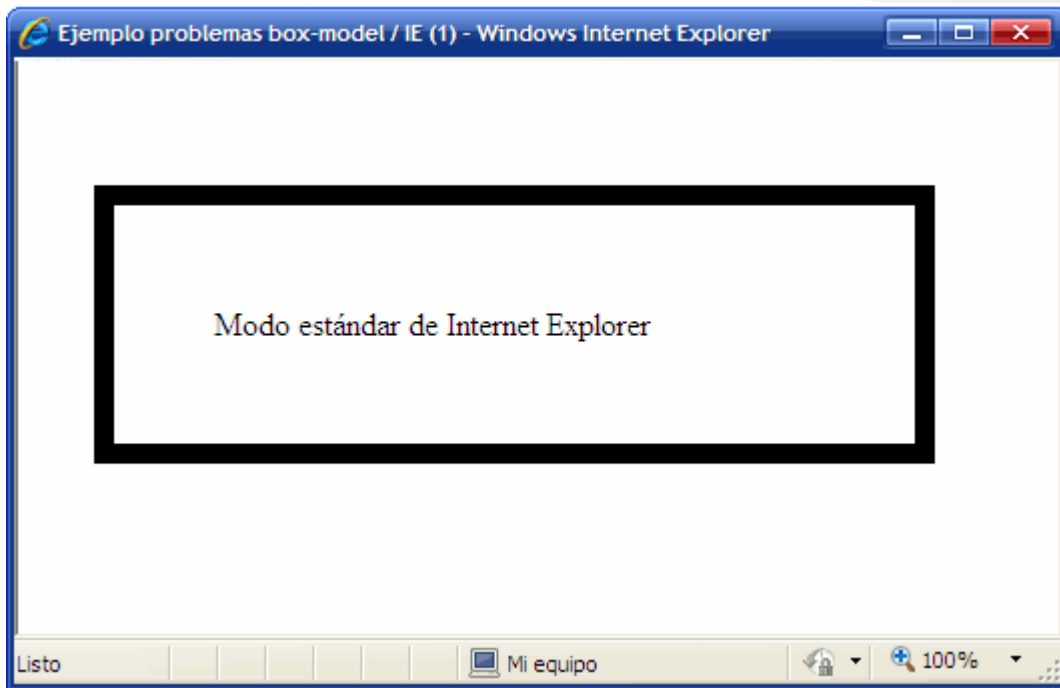
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Se pueden consultar todos los casos concretos que activan el modo *quirks* para cada navegador en la página <http://hsivonen.iki.fi/doctype/>

La versión 5.5 y anteriores de Internet Explorer y las versiones 6 y 7 en modo *quirks* siguen su propio modelo de cálculo de anchuras y alturas que es muy diferente al método definido por el estándar.

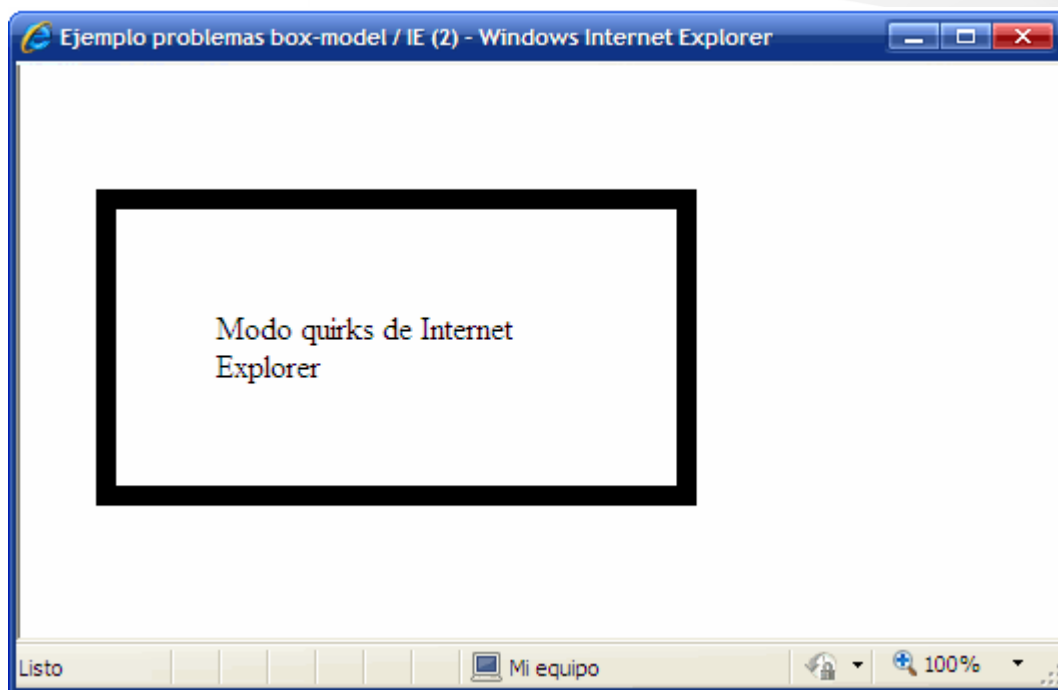
La siguiente imagen muestra el elemento del ejemplo anterior en la versión 6 de Internet Explorer en modo estándar:



Internet Explorer 6 en modo estándar

La anchura del elemento es la que se obtiene de sumar la anchura de su contenido (300), sus bordes (2×10) y sus rellenos (2×50). Por lo tanto, la anchura del elemento son 420 píxel, a los que se suman los 30 píxel de margen lateral a cada lado.

Sin embargo, el mismo ejemplo en el modo *quirks* de la versión 6 de Internet Explorer muestra el siguiente aspecto:



Internet Explorer 6 en modo quirks

Las versiones anteriores de Internet Explorer y las versiones 6 y 7 en modo *quirks* consideran que la anchura establecida por CSS no sólo es la anchura del contenido, sino que también incluye los bordes y el relleno.

Por lo tanto, en este caso la anchura total del elemento (sin contar los márgenes laterales) es de 300píxel, el mismo valor que se indica en la propiedad `width`. El espacio ocupado por los bordes del elemento (2 x 10) y sus rellenos (2 x 50) se resta de la anchura de su contenido.

Para evitar este problema y crear diseños con el mismo aspecto en cualquier navegador, es necesario evitar el modo *quirks* de Internet Explorer. Por tanto, todas las páginas deben incluir la declaración apropiada de DOCTYPE.

Los modos de compatibilidad de Internet Explorer 8

El navegador Internet Explorer 8 introduce el concepto de "*compatibilidad de la página*" para asegurar que todas las páginas HTML se vean correctamente en cualquier versión de ese navegador. En realidad, esta nueva característica es una mejora del *modo quirks* explicado anteriormente.

Internet Explorer 8, a diferencia de sus versiones anteriores, soporta completamente el estándar CSS 2.1. Sin embargo, muchos sitios web se diseñaron

para Internet Explorer 6 y 7, por lo que incluyen trucos, *hacks* y filtros que arreglan los errores y carencias de esas versiones del navegador.

Para evitar que las páginas diseñadas para navegadores anteriores se vean mal en esta nueva versión, Internet Explorer 8 incluye la opción de "compatibilidad de la página", que permite indicar la versión de Internet Explorer para la que la página ha sido diseñada.

De esta forma, si la página no se visualiza correctamente en Internet Explorer 8, se puede indicar al navegador que la muestre como si fuera Internet Explorer 6 o 7. En realidad, Internet Explorer 8 incluye seis modos de funcionamiento:

Modo IE5: la página se muestra según el modo *quirks* de Internet Explorer 7, que es casi idéntico a como se veían las páginas en el navegador Internet Explorer 5.

Modo IE7: la página se muestra en el modo estándar de Internet Explorer 7, sin importar si la página contiene o no la directiva `<!DOCTYPE>`.

Modo IE8: los contenidos se muestran en el modo estándar de Internet Explorer 8, que es el más parecido al del resto de navegadores que soportan los estándares (Firefox, Opera, Safari y Google Chrome).

Emular el modo IE7: el navegador decide cómo mostrar los contenidos a partir de la directiva `<!DOCTYPE>` de la página. Si esa directiva es una de las que activan el modo estándar, la página se muestra en el modo estándar de Internet Explorer 7. En otro caso, se muestra en el modo *quirks* de Internet Explorer 5. Este modo es el más útil para la mayoría de sitios web.

Emular el modo IE8: el navegador decide cómo mostrar los contenidos a partir de la directiva `<!DOCTYPE>` de la página. Si esa directiva es una de las que activan el modo estándar, la página se muestra en el modo estándar de Internet Explorer 8. En otro caso, se muestra en el modo *quirks* de Internet Explorer 5.

Modo límite ("edge mode"): indica a Internet Explorer que los contenidos se deben mostrar en el modo de compatibilidad más avanzado disponible. Actualmente, este modo es equivalente al modo IE8. Si las futuras versiones Internet Explorer 9 y 10 incluyeran mejor compatibilidad, las páginas se visualizarían en ese modo avanzado de compatibilidad.

El modo de compatibilidad de la página se indica mediante una nueva etiqueta `<meta>` con la propiedad `X-UA-Compatible` y cuyo valor es el que utiliza Internet Explorer 8 para determinar el modo que se utiliza:

```
<!-- Modo IE5 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=5" />
  ...
</head>

<!-- Modo IE7 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=7" />
  ...
</head>

<!-- Modo IE8 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=8" />
  ...
</head>

<!-- Emular el modo IE7 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
  ...
</head>

<!-- Emular el modo IE8 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8" />
  ...
</head>

<!-- Modo límite -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  ...
</head>
```

No obstante, esta opción de compatibilidad de la página debe entenderse como una solución temporal que evita que los sitios web se vean mal en Internet Explorer 8. La única solución correcta a largo plazo consiste en actualizar las páginas para que sus diseños sigan los estándares web.

Fondos

El último elemento que forma el *box model* es el fondo de la caja del elemento. El fondo puede ser un color simple o una imagen. El fondo solamente se visualiza en el área ocupada por el contenido y su relleno, ya que el color de los bordes se controla directamente desde los bordes y las zonas de los márgenes siempre son transparentes.

Para establecer un color o imagen de fondo en la página entera, se debe establecer un fondo al elemento `<body>`. Si se establece un fondo a la página, como el valor inicial del fondo de los elementos es transparente, todos los elementos de la página se visualizan con el mismo fondo a menos que algún elemento especifique su propio fondo.

CSS define cinco propiedades para establecer el fondo de cada elemento (`background-color`, `background-image`, `background-repeat`, `background-attachment`, `background-position`) y otra propiedad de tipo "shorthand" (`background`).

La propiedad `background-color` permite mostrar un color de fondo sólido en la caja de un elemento. Esta propiedad no permite crear degradados ni ningún otro efecto avanzado.

Propiedad	background-color
Valores	color transparent inherit
Se aplica a	Todos los elementos
Valor inicial	transparent
Descripción	Establece un color de fondo para los elementos

El siguiente ejemplo muestra una página web con un color gris claro de fondo:

```

1  body {
2      background-color: #F5F5F5;
3  }
4  |

```

Para crear efectos gráficos avanzados, es necesario utilizar la propiedad `background-image`, que permite mostrar una imagen como fondo de la caja de cualquier elemento:

Propiedad	background-image
Valores	url none inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece una imagen como fondo para los elementos

CSS permite establecer de forma simultánea un color y una imagen de fondo. En este caso, la imagen se muestra delante del color, por lo que solamente si la imagen contiene zonas transparentes es posible ver el color de fondo.

El siguiente ejemplo muestra una imagen como fondo de toda la página:

```
body { background-image: url("imagenes/fondo.png") }
```

Las imágenes de fondo se indican a través de su URL, que puede ser absoluta o relativa. Suele ser recomendable crear una carpeta de imágenes que se encuentre en el mismo directorio que los archivos CSS y que almacene todas las imágenes utilizadas en el diseño de las páginas.

Así, las imágenes correspondientes al diseño de la página se mantienen separadas del resto de imágenes del sitio y el código CSS es más sencillo (por utilizar URL relativas) y más fácil de mantener (por no tener que actualizar URL absolutas en caso de que se cambie la estructura del sitio web).

Por otra parte, suele ser habitual indicar un color de fondo siempre que se muestra una imagen de fondo. En caso de que la imagen no se pueda mostrar o contenga errores, el navegador mostrará el color indicado (que debería ser, en lo posible, similar a la imagen) y la página no parecerá que contiene errores.

Si la imagen que se quiere mostrar es demasiado grande para el fondo del elemento, solamente se muestra la parte de imagen comprendida en el tamaño del elemento. Si la imagen es más pequeña que el elemento, CSS la repite horizontal y verticalmente hasta llenar el fondo del elemento.

Este comportamiento es útil para establecer un fondo complejo a una página web entera. El siguiente ejemplo utiliza una imagen muy pequeña para establecer un fondo complejo a toda una página:

Imagen original

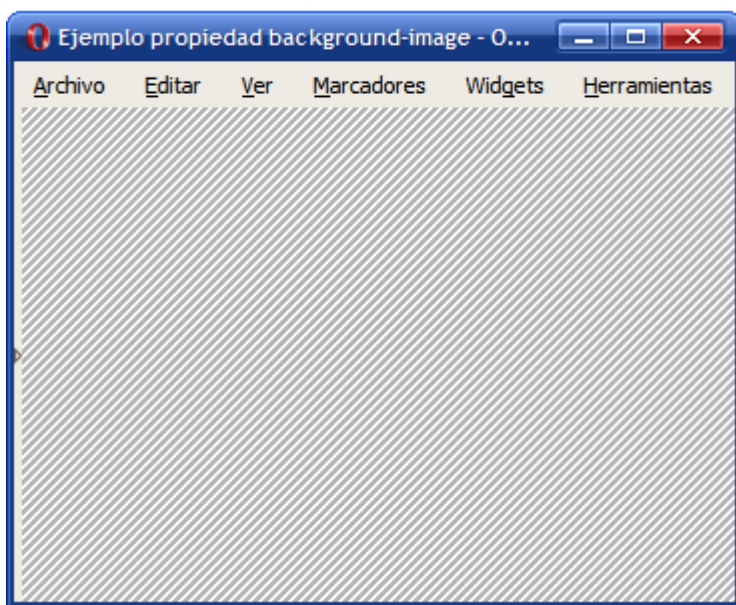


Imagen original utilizada para el fondo de la página

Reglas CSS

```
body {  
    background-image:url(imagenes/fondo.gif);  
}
```

Resultado



Página con una imagen de fondo

Con una imagen muy pequeña (y que, por tanto, se puede descargar en muy poco tiempo) se consigue cubrir completamente el fondo de la página, con lo que se consigue un gran ahorro de ancho de banda.

En ocasiones, no es conveniente que la imagen de fondo se repita horizontal y verticalmente. Para ello, CSS introduce la propiedad `background-repeat` que permite controlar la forma de repetición de las imágenes de fondo.

Propiedad	background-repeat
Valores	repeat repeat-x repeat-y no-repeat inherit
Se aplica a	Todos los elementos
Valor inicial	repeat
Descripción	Controla la forma en la que se repiten las imágenes de fondo

El valor `repeat` indica que la imagen se debe repetir en todas direcciones y por tanto, es el comportamiento por defecto. El valor `no-repeat` muestra una sola vez la imagen y no se repite en ninguna dirección. El valor `repeat-x` repite la imagen

sólo horizontalmente y el valor repeat-y repite la imagen solamente de forma vertical.

El sitio web <http://www.kottke.org/> utiliza el valor repeat-x para mostrar una imagen de fondo en la cabecera de la página:

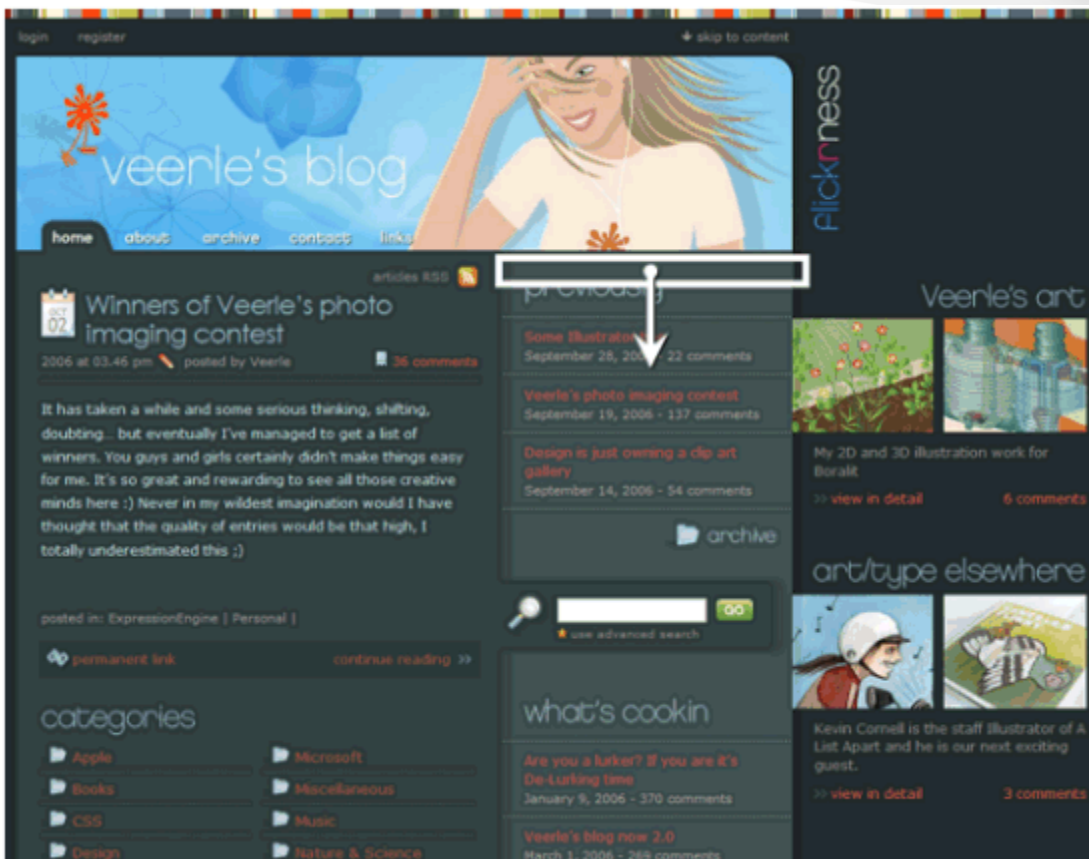


Uso de repeat-x en la página de Kottke.org

Las reglas CSS definidas para la cabecera son:

```
#hdr {
  background: url("/images/ds.gif") repeat-x;
  width: 100%;
  text-align: center;
}
```

Por otra parte, el sitio web <http://veerle.duoh.com/> utiliza el valor repeat-y para mostrar el fondo de una columna de contenidos:



Uso de repeat-y en la página de Veerle.duoh.com

Las reglas CSS definidas para esa columna de contenidos son:

```
.wide #content-secondary {
    width: 272px;
    margin: 13px 0 0 0;
    position: relative;
    margin-left: -8px;
    background: url("../graphics/wide/bg-content-secondary.gif") repeat-y;
}
```

Además de seleccionar el tipo de repetición de las imágenes de fondo, CSS permite controlar la posición de la imagen dentro del fondo del elemento mediante la propiedad `background-position`.

Propiedad	background-position
Valores	((porcentaje unidad de medida left center right) (porcentaje unidad de medida top center bottom) ?) ((left

Propiedad	background-position
	center right) (top center bottom)) inherit
Se aplica a	Todos los elementos
Valor inicial	0% 0%
Descripción	Controla la posición en la que se muestra la imagen en el fondo del elemento

La propiedad `background-position` permite indicar la distancia que se desplaza la imagen de fondo respecto de su posición original situada en la esquina superior izquierda.

Si se indican dos porcentajes o dos medidas, el primero indica el desplazamiento horizontal y el segundo el desplazamiento vertical respecto del origen (situado en la esquina superior izquierda). Si solamente se indica un porcentaje o una medida, se considera que es el desplazamiento horizontal y al desplazamiento vertical se le asigna automáticamente el valor de 50%.

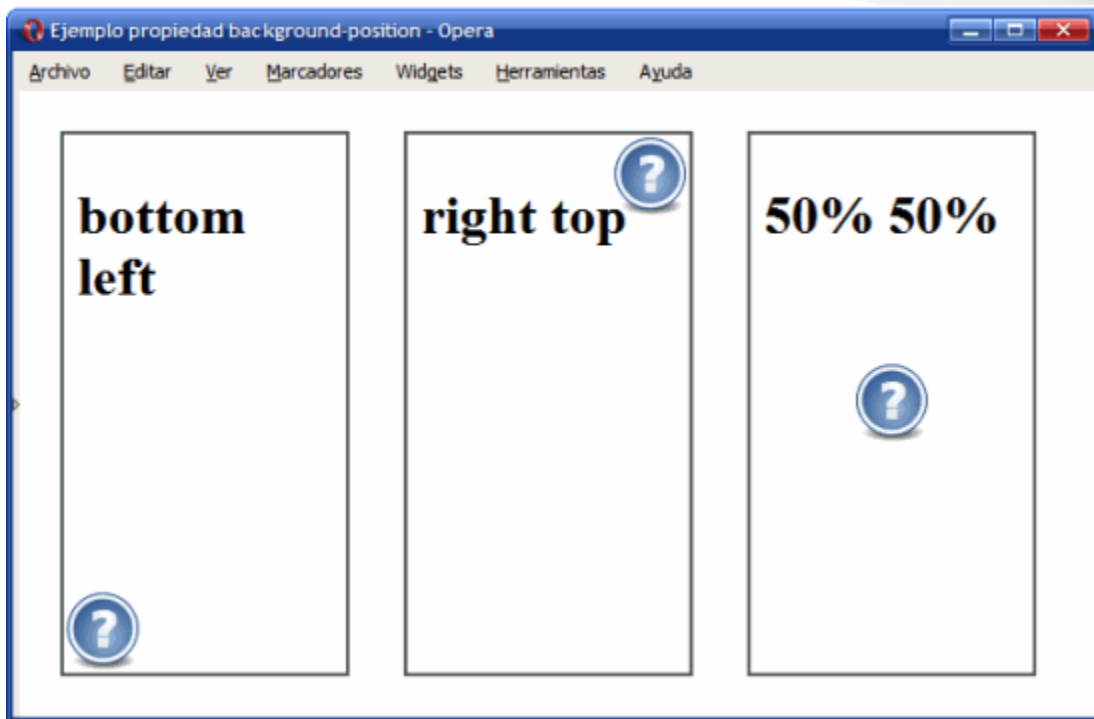
Cuando se utilizan porcentajes, su interpretación no es intuitiva. Si el valor de la propiedad `background-position` se indica mediante dos porcentajes `x% y%`, el navegador coloca el punto `(x%, y%)` de la imagen de fondo en el punto `(x%, y%)` del elemento.

Las palabras clave permitidas son equivalentes a algunos porcentajes significativos: `top` = 0%, `left` = 0%, `center` = 50%, `bottom` = 100%, `right` = 100%.

CSS permite mezclar porcentajes y palabras clave, como por ejemplo `50% 2cm`, `center 2cm`, `center 10%`.

Si se utilizan solamente palabras clave, el orden es indiferente y por tanto, es equivalente indicar `top left` y `left top`.

El siguiente ejemplo muestra una misma imagen de fondo posicionada de tres formas diferentes:



Ejemplo de propiedad background-position

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
#caja1 {
  background-image: url("images/help.png");
  background-repeat: no-repeat;
  background-position: bottom left;
}
#caja2 {
  background-image: url("images/help.png");
  background-repeat: no-repeat;
  background-position: right top;
}
#caja3 {
  background-image: url("images/help.png");
  background-repeat: no-repeat;
  background-position: 50% 50%;
}

<div id="caja1"><h1>bottom left</h1></div>
<div id="caja2"><h1>right top</h1></div>
<div id="caja3"><h1>50% 50%</h1></div>
```

Opcionalmente, se puede indicar que el fondo permanezca fijo cuando la ventana del navegador se desplaza mediante las barras de *scroll*. Se trata de un comportamiento que en general no es deseable y que algunos navegadores no soportan correctamente. La propiedad que controla este comportamiento es `background-attachment`.

Propiedad	background-attachment
Valores	scroll fixed inherit
Se aplica a	Todos los elementos
Valor inicial	scroll
Descripción	Controla la forma en la que se visualiza la imagen de fondo: permanece fija cuando se hace scroll en la ventana del navegador o se desplaza

Propiedad	background-attachment
	junto con la ventana

Para hacer que una imagen de fondo se muestre fija al desplazar la ventana del navegador, se debe añadir la propiedad `background-attachment: fixed`.

Por último, CSS define una propiedad de tipo "shorthand" para indicar todas las propiedades de los colores e imágenes de fondo de forma directa. La propiedad se denomina `background` y es la que generalmente se utiliza para establecer las propiedades del fondo de los elementos.

Propiedad	background
Valores	(background-color background-image background-repeat background-attachment background-position) inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece todas las propiedades del fondo de un elemento

El orden en el que se indican las propiedades es indiferente, aunque en general se sigue el formato indicado de color, url de imagen, repetición y posición.

El siguiente ejemplo muestra la ventaja de utilizar la propiedad `background`:

```
/* Color e imagen de fondo de la página mediante una propiedad shorthand */
body { background: #222d2d url(./graphics/colorstrip.gif) repeat-x 0 0; }

/* La propiedad shorthand anterior es equivalente a las siguientes propiedades */
body {
  background-color: #222d2d;
  background-image: url("./graphics/colorstrip.gif");
  background-repeat: repeat-x;
  background-position: 0 0;
}
```

La propiedad `background` permite asignar todos o sólo algunos de todos los valores que se pueden definir para los fondos de los elementos:

```
background: url("../graphics/wide/bg-content-secondary.gif") repeat-y;
background: url("../graphics/wide/footer-content-secondary.gif") no-repeat bottom left;
background: transparent url("../graphics/navigation.gif") no-repeat 0 -27px;
background: none;
background: #293838 url("../graphics/icons/icon-permalink-big.gif") no-repeat center left;
```

Posicionamiento y visualización

Cuando los navegadores descargan el contenido HTML y CSS de las páginas web, aplican un procesamiento muy complejo antes de mostrar las páginas en la pantalla del usuario.

Para cumplir con el modelo de cajas presentado en el capítulo anterior, los navegadores crean una caja para representar a cada elemento de la página HTML. Los factores que se tienen en cuenta para generar cada caja son:

Las propiedades `width` y `height` de la caja (si están establecidas).

El tipo de cada elemento HTML (elemento de bloque o elemento en línea).

Posicionamiento de la caja (normal, relativo, absoluto, fijo o flotante).

Las relaciones entre elementos (dónde se encuentra cada elemento, elementos descendientes, etc.)

Otro tipo de información, como por ejemplo el tamaño de las imágenes y el tamaño de la ventana del navegador.

En este capítulo se muestran los cinco tipos de posicionamientos definidos para las cajas y se presentan otras propiedades que afectan a la forma en la que se visualizan las cajas.

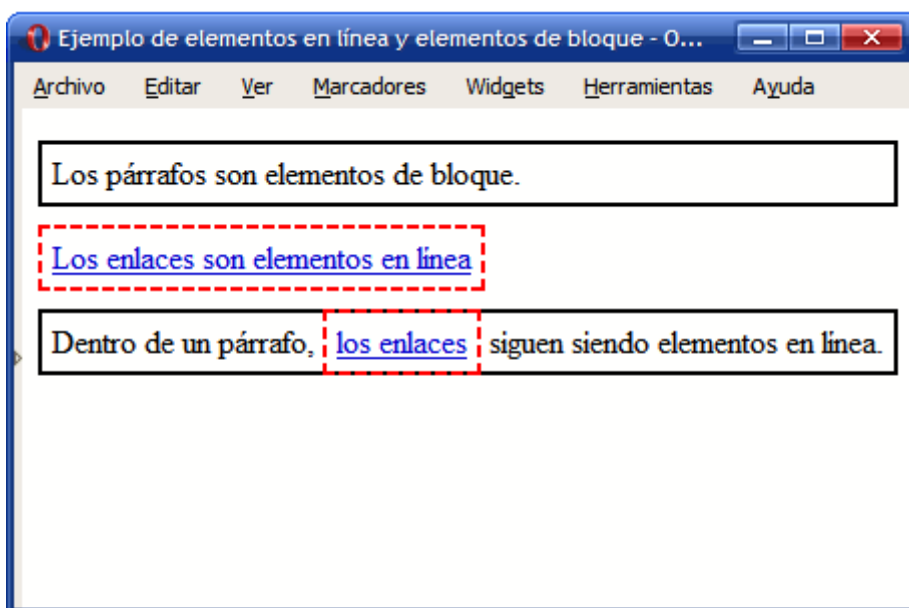
Tipos de elementos

El estándar HTML clasifica a todos sus elementos en dos grandes grupos: elementos en línea y elementos de bloque.

Los elementos de bloque ("*block elements*" en inglés) siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea. Por su parte, los elementos en línea ("*inline elements*" en inglés) no empiezan

necesariamente en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

Debido a este comportamiento, el tipo de un elemento influye de forma decisiva en la caja que el navegador crea para mostrarlo. La siguiente imagen muestra las cajas que crea el navegador para representar los diferentes elementos que forman una página HTML:



Cajas creadas por los elementos de línea y los elementos de bloque

El primer elemento de la página anterior es un párrafo. Los párrafos son elementos de bloque y por ese motivo su caja empieza en una nueva línea y llega hasta el final de esa misma línea. Aunque los contenidos de texto del párrafo no son suficientes para ocupar toda la línea, el navegador reserva todo el espacio disponible en la primera línea.

El segundo elemento de la página es un enlace. Los enlaces son elementos en línea, por lo que su caja sólo ocupa el espacio necesario para mostrar sus contenidos. Si después de este elemento se incluye otro elemento en línea (por ejemplo, otro enlace o una imagen) el navegador mostraría los dos elementos en la misma línea, ya que existe espacio suficiente.

Por último, el tercer elemento de la página es un párrafo que se comporta de la misma forma que el primer párrafo. En su interior, se encuentra un enlace que también se comporta de la misma forma que el enlace anterior. Así, el segundo párrafo ocupa toda una línea y el segundo enlace sólo ocupa el espacio necesario para mostrar sus contenidos.

Por sus características, los elementos de bloque no pueden insertarse dentro de elementos en línea y tan sólo pueden aparecer dentro de otros elementos de bloque. En cambio, un elemento en línea puede aparecer tanto dentro de un elemento de bloque como dentro de otro elemento en línea.

Los elementos en línea definidos por HTML son: a, abbr, acronym, b, basefont, bdo, big, br, cite, code, dfn, em, font, i, img, input, kbd, label, q, s, samp, select, small, span, strike, strong, sub, sup, textarea, tt, u, var.

Los elementos de bloque definidos por HTML son: address, blockquote, center, dir, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, isindex, menu, noframes, noscript, ol, p, pre, table, ul.

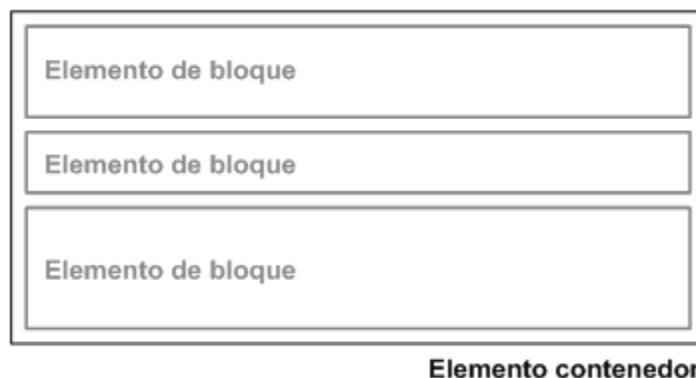
Los siguientes elementos también se considera que son de bloque: dd, dt, frameset, li, tbody, td, tfoot, th, thead, tr.

Los siguientes elementos pueden ser en línea y de bloque según las circunstancias: button, del, iframe, ins, map, object, script.

Posicionamiento normal

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, sólo se tiene en cuenta si el elemento es de bloque o en línea, sus propiedades `width` y `height` y su contenido.

Los elementos de bloque forman lo que CSS denomina "*contextos de formato de bloque*". En este tipo de contextos, las cajas se muestran una debajo de otra comenzando desde el principio del elemento contenedor. La distancia entre las cajas se controla mediante los márgenes verticales.

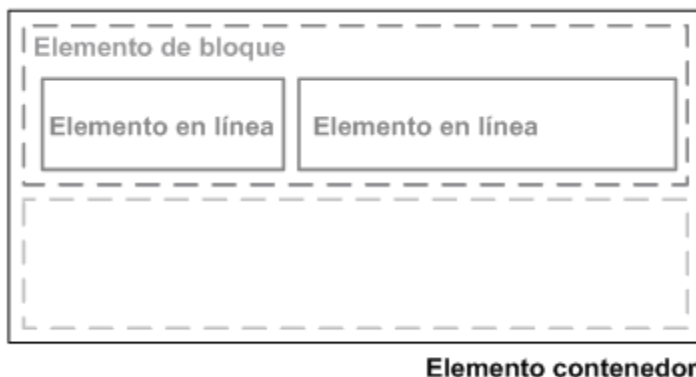


Posicionamiento normal de los elementos de bloque

Si un elemento se encuentra dentro de otro, el elemento padre se llama "*elemento contenedor*" y determina tanto la posición como el tamaño de todas sus cajas interiores.

Si un elemento no se encuentra dentro de un elemento contenedor, entonces su elemento contenedor es el elemento `<body>` de la página. Normalmente, la anchura de los elementos de bloque está limitada a la anchura de su elemento contenedor, aunque en algunos casos sus contenidos pueden desbordar el espacio disponible.

Los elementos en línea forman los "*contextos de formato en línea*". En este tipo de contextos, las cajas se muestran una detrás de otra de forma horizontal comenzando desde la posición más a la izquierda de su elemento contenedor. La distancia entre las cajas se controla mediante los márgenes laterales.



Posicionamiento normal de los elementos en línea

Si las cajas en línea ocupan más espacio del disponible en su propia línea, el resto de cajas se muestran en las líneas inferiores. Si las cajas en línea ocupan un espacio menor que su propia línea, se puede controlar la distribución de las cajas mediante la propiedad `text-align` para centrarlas, alinearlas a la derecha o justificarlas.

Posicionamiento relativo

El estándar CSS considera que el posicionamiento relativo es un caso particular del posicionamiento normal, aunque en la práctica presenta muchas diferencias.

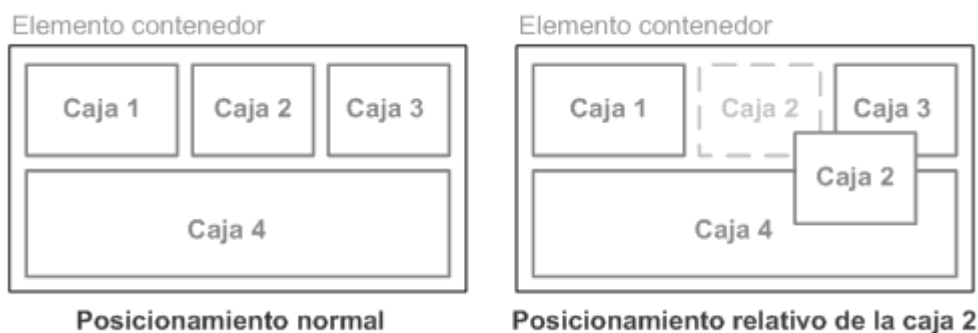
El posicionamiento relativo desplaza una caja respecto de su posición original establecida mediante el posicionamiento normal. El desplazamiento de la caja se controla con las propiedades `top`, `right`, `bottom` y `left`.

El valor de la propiedad `top` se interpreta como el desplazamiento entre el borde superior de la caja en su posición final y el borde superior de la misma caja en su posición original.

De la misma forma, el valor de las propiedades `left`, `right` y `bottom` indica respectivamente el desplazamiento entre el borde izquierdo/derecho/inferior de la caja en su posición final y el borde izquierdo/derecho/inferior de la caja original.

Por tanto, la propiedad `top` se emplea para mover las cajas de forma descendente, la propiedad `bottom` mueve las cajas de forma ascendente, la propiedad `left` se utiliza para desplazar las cajas hacia la derecha y la propiedad `right` mueve las cajas hacia la izquierda. Este comportamiento parece poco intuitivo y es causa de errores cuando se empiezan a diseñar páginas con CSS. Si se utilizan valores negativos en las propiedades `top`, `right`, `bottom` y `left`, su efecto es justamente el inverso.

El desplazamiento relativo de una caja no afecta al resto de cajas adyacentes, que se muestran en la misma posición que si la caja desplazada no se hubiera movido de su posición original.



Ejemplo de posicionamiento relativo de un elemento

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.

Las cajas desplazadas de forma relativa no modifican su tamaño, por lo que los valores de las propiedades `left` y `right` siempre cumplen que `left = -right`.

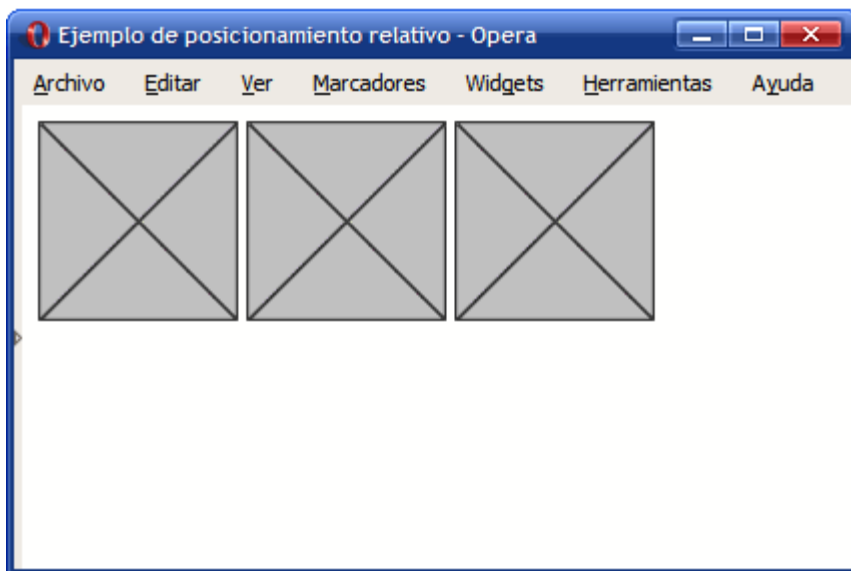
Si tanto `left` como `right` tienen un valor de `auto` (que es su valor por defecto) la caja no se mueve de su posición original. Si sólo el valor de `left` es `auto`, su valor real es `-right`. Igualmente, si sólo el valor de `right` es `auto`, su valor real es `-left`.

Si tanto `left` como `right` tienen valores distintos de `auto`, uno de los dos valores se tiene que ignorar porque son mutuamente excluyentes. Para determinar la propiedad que se tiene en cuenta, se considera el valor de la propiedad `direction`.

La propiedad `direction` permite establecer la dirección del texto de un contenido. Si el valor de `direction` es `ltr`, el texto se muestra de izquierda a derecha, que es el método de escritura habitual en la mayoría de países. Si el valor de `direction` es `rtl`, el método de escritura es de derecha a izquierda, como el utilizado por los idiomas árabe y hebreo.

Si el valor de `direction` es `ltr`, y las propiedades `left` y `right` tienen valores distintos de `auto`, se ignora la propiedad `right` y sólo se tiene en cuenta el valor de la propiedad `left`. De la misma forma, si el valor de `direction` es `rtl`, se ignora el valor de `left` y sólo se tiene en cuenta el valor de `right`.

El siguiente ejemplo muestra tres imágenes posicionadas de forma normal:



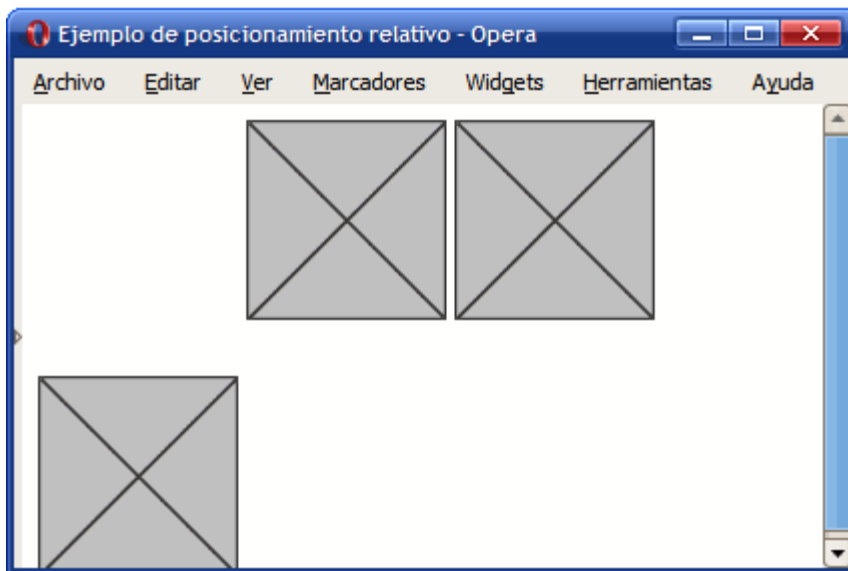
Elementos posicionados de forma normal

Aplicando el posicionamiento relativo, se desplaza la primera imagen de forma descendente:


```
img.desplazada {  
  position: relative;  
  top: 8em;  
}  
  
  
  

```

El aspecto que muestran ahora las imágenes es el siguiente:



Elemento posicionado de forma relativa

El resto de imágenes no varían su posición y por tanto no ocupan el hueco dejado por la primera imagen, ya que el posicionamiento relativo no influye en el resto de elementos de la página. El principal problema de posicionar elementos de forma relativa es que se pueden producir solapamientos con otros elementos de la página.

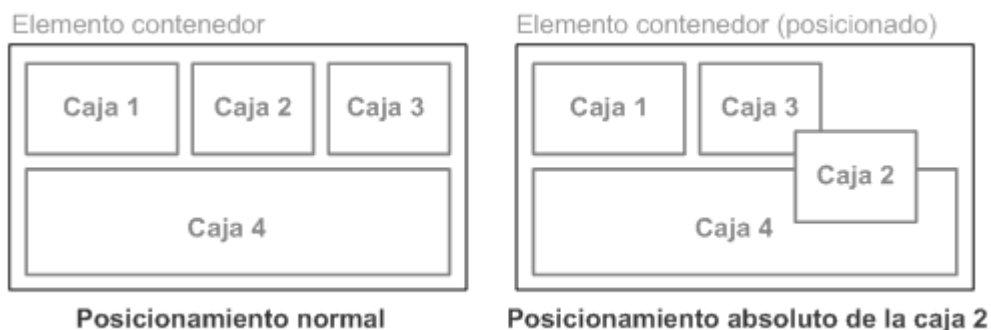
Posicionamiento absoluto

El posicionamiento absoluto se emplea para establecer de forma exacta la posición en la que se muestra la caja de un elemento. La nueva posición de la caja se indica mediante las propiedades `top`, `right`, `bottom` y `left`. La interpretación de los valores de estas propiedades es mucho más compleja que en el posicionamiento relativo, ya que en este caso dependen del posicionamiento del elemento contenedor.

Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página se ven afectados y modifican su posición. Al igual que en el

posicionamiento relativo, cuando se posiciona de forma absoluta una caja es probable que se produzcan solapamientos con otras cajas.

En el siguiente ejemplo, se posiciona de forma absoluta la caja 2:



Ejemplo de posicionamiento absoluto de un elemento

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.

El estándar de CSS 2.1 indica que las cajas posicionadas de forma absoluta "*salen del flujo normal de la página*", lo que provoca que el resto de elementos de la página se muevan y en ocasiones, ocupen la posición original en la que se encontraba la caja.

Por otra parte, el desplazamiento de una caja posicionada de forma absoluta se controla mediante las propiedades `top`, `right`, `bottom` y `left`. A diferencia del posicionamiento relativo, la interpretación de los valores de estas propiedades depende del elemento contenedor de la caja posicionada.

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:

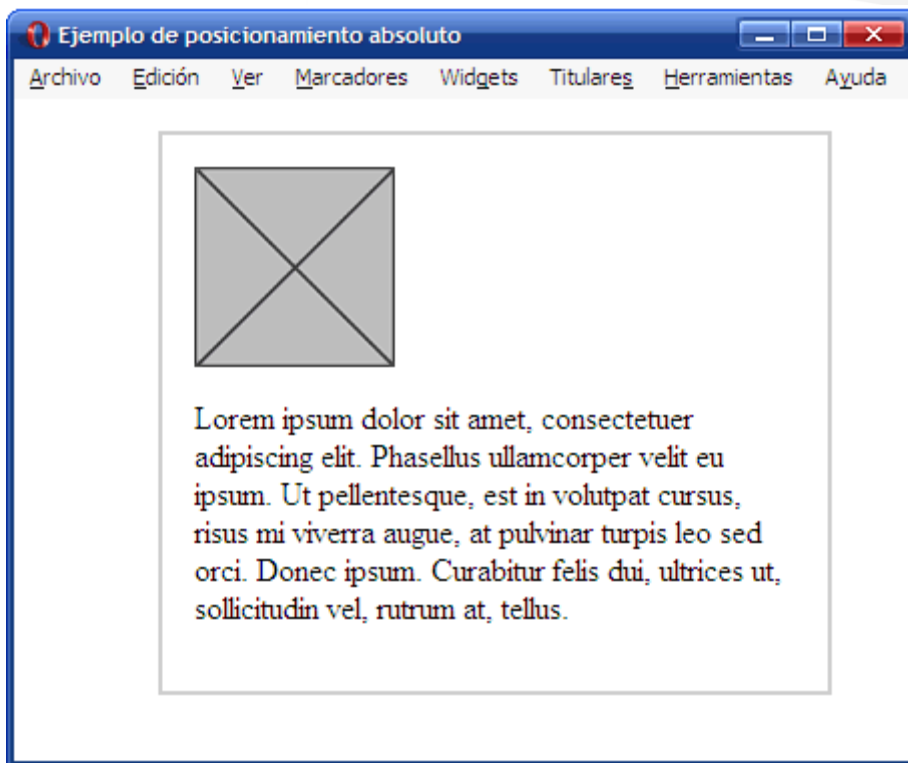
El valor de la propiedad `top` indica el desplazamiento desde el borde superior de la caja hasta el borde superior del elemento contenedor que se utiliza como referencia.

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:

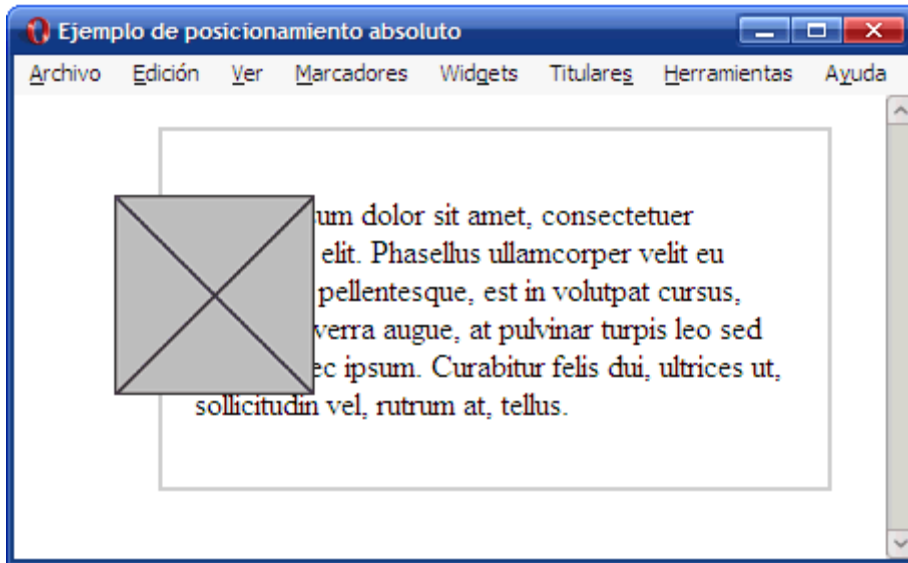
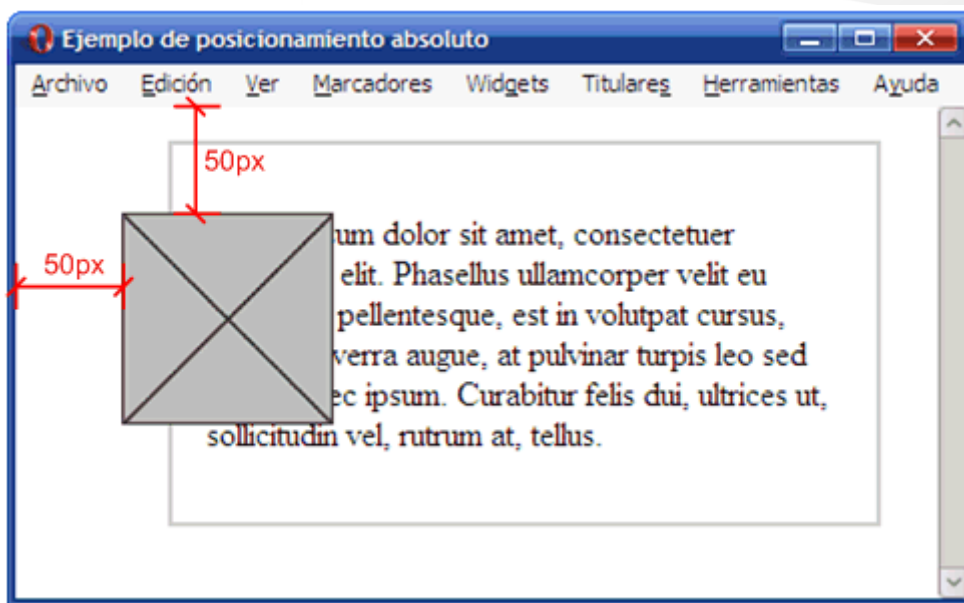


Imagen posicionada de forma absoluta

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:

Obtiene la lista de elementos contenedores de la imagen: `<div>` y `<body>`.

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.

El posicionamiento de todos los elementos contenedores es el normal o estático, ya que ni siquiera tienen establecida la propiedad `position`

Como ningún elemento contenedor está posicionado, la referencia es la ventana del navegador.

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:

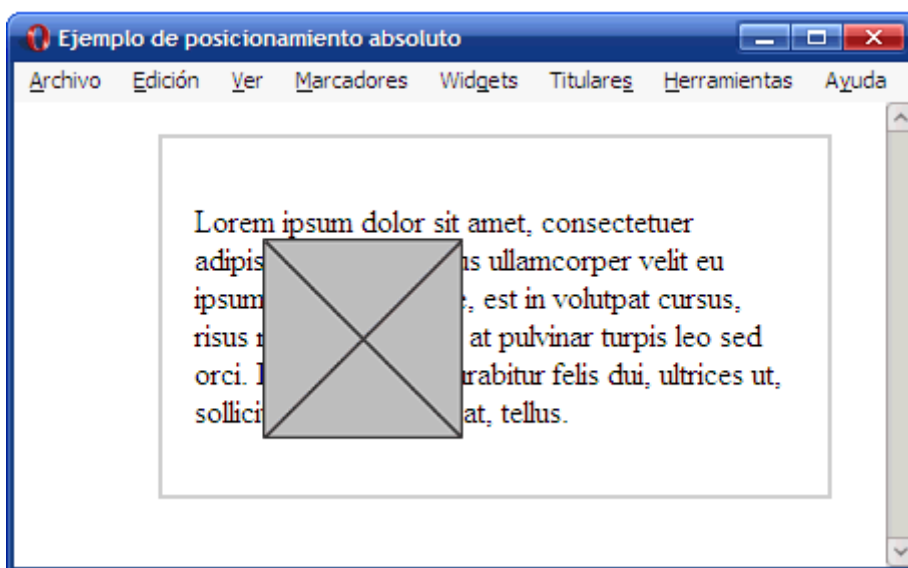
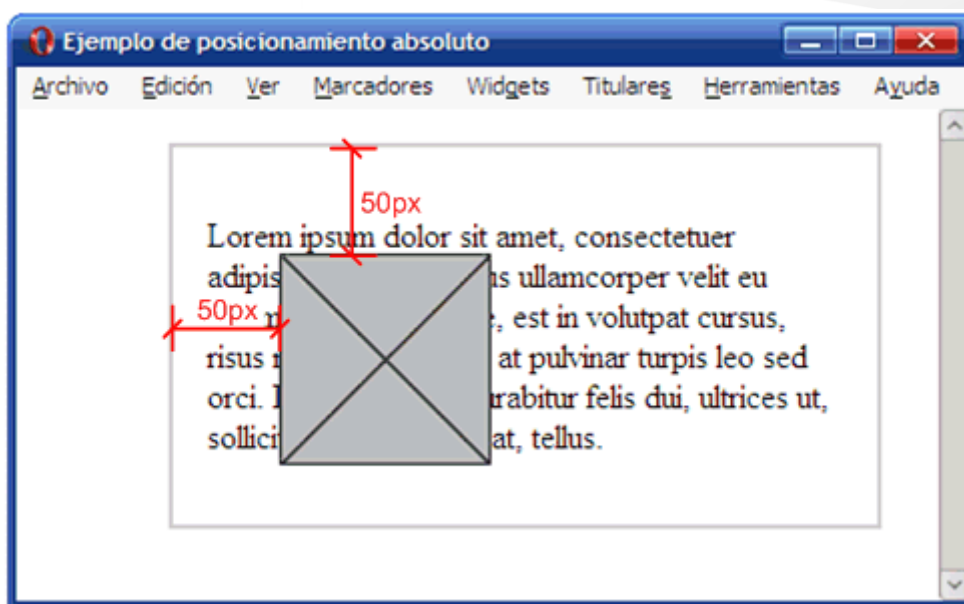


Imagen posicionada de forma absoluta

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.

Posicionamiento fijo

El estándar CSS considera que el posicionamiento fijo es un caso particular del posicionamiento absoluto, ya que sólo se diferencian en el comportamiento de las cajas posicionadas.

Cuando una caja se posiciona de forma fija, la forma de obtener el origen de coordenadas para interpretar su desplazamiento es idéntica al posicionamiento absoluto. De hecho, si el usuario no mueve la página HTML en la ventana del navegador, no existe ninguna diferencia entre estos dos modelos de posicionamiento.

La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador. El posicionamiento fijo hace que las cajas no modifiquen su posición ni, aunque el usuario suba o baje la página en la ventana de su navegador.

Si la página se visualiza en un medio paginado (por ejemplo, en una impresora) las cajas posicionadas de forma fija se repiten en todas las páginas. Esta característica puede ser útil para crear encabezados o pies de página en páginas HTML preparadas para imprimir.

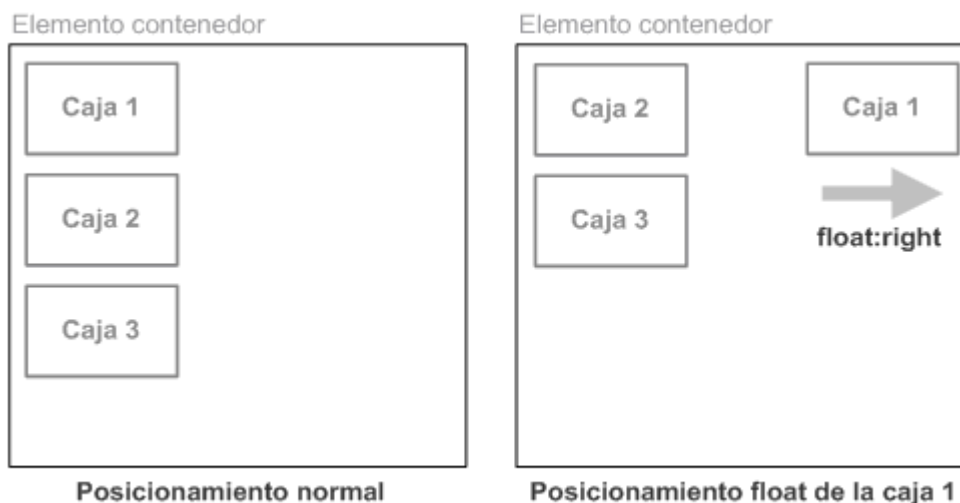
El posicionamiento fijo apenas se ha utilizado en el diseño de páginas web hasta hace poco tiempo porque el navegador Internet Explorer 6 y las versiones anteriores no lo soportan.

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, como se verá más adelante.

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.

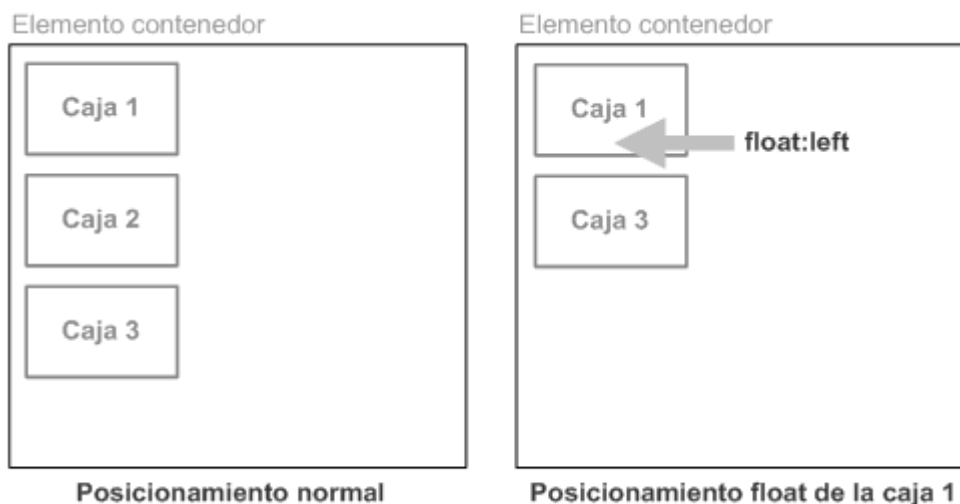
La siguiente imagen muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1:



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:

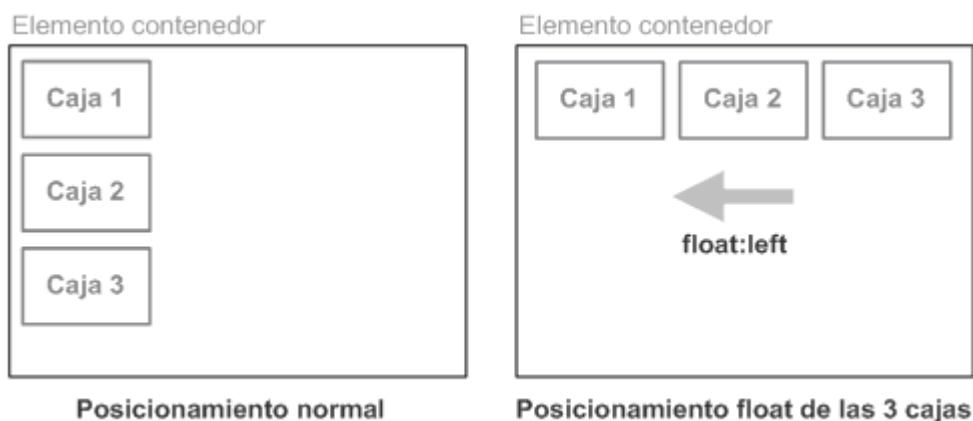


Ejemplo de posicionamiento float de una caja

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 dónde estaba la caja 1 y la caja 3 se muestra dónde 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.

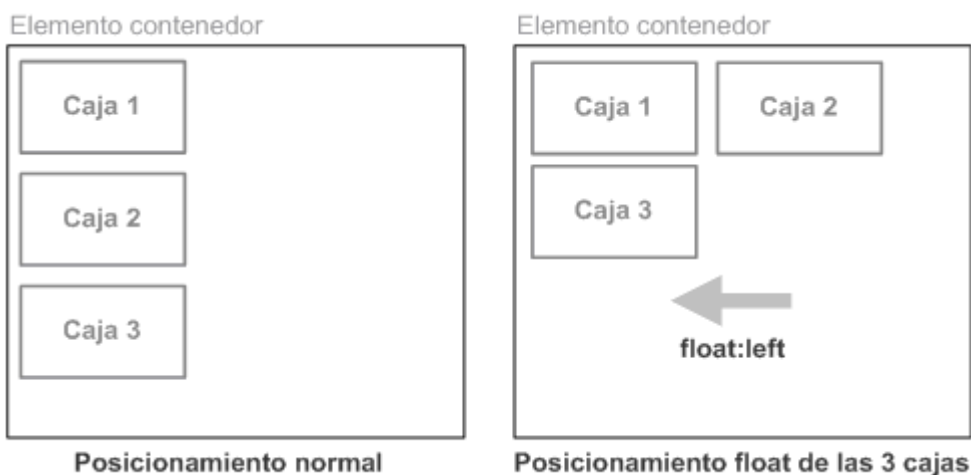
Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible. En el siguiente ejemplo se posicionan de forma flotante hacia la izquierda las tres cajas:



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.

La propiedad CSS que permite posicionar de forma flotante una caja se denomina float:

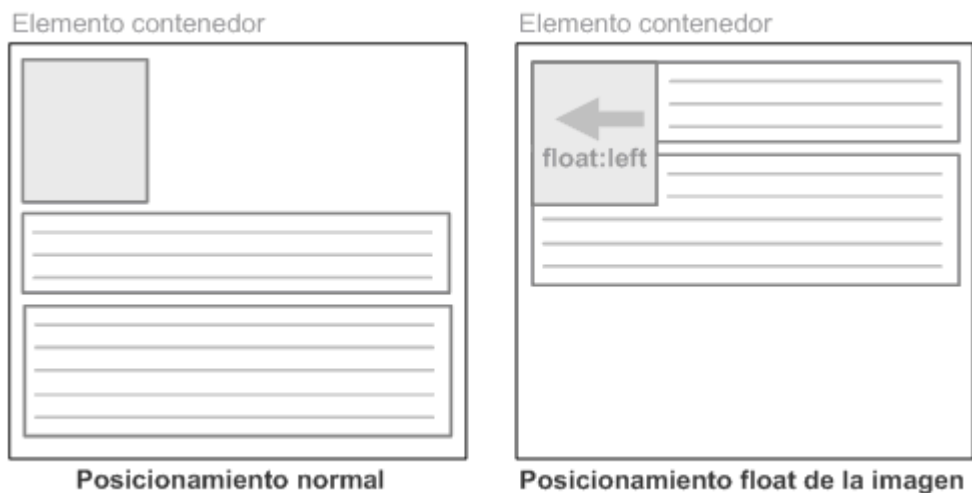
Propiedad	float
Valores	left right none inherit

Propiedad	float
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el tipo de posicionamiento flotante del elemento

Si se indica un valor `left`, la caja se desplaza hasta el punto más a la izquierda posible en esa misma línea (si no existe sitio en esa línea, la caja baja una línea y se muestra lo más a la izquierda posible en esa nueva línea). El resto de elementos adyacentes se adaptan y *fluyen* alrededor de la caja flotante.

El valor `right` tiene un funcionamiento idéntico, salvo que, en este caso, la caja se desplaza hacia la derecha. El valor `none` permite anular el posicionamiento flotante de forma que el elemento se muestre en su posición original.

Los elementos que se encuentran alrededor de una caja flotante adaptan sus contenidos para que fluyan alrededor del elemento posicionado:



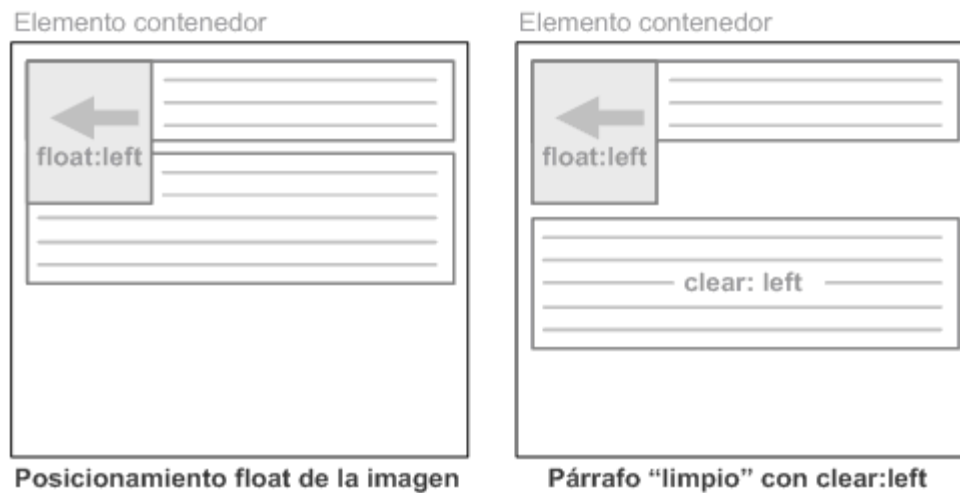
Elementos que fluyen alrededor de un elemento posicionado mediante float

La regla CSS que se aplica en la imagen del ejemplo anterior es:

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

Uno de los principales motivos para la creación del posicionamiento float fue precisamente la posibilidad de colocar imágenes alrededor de las cuales fluye el texto.

CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante float. De hecho, en muchas ocasiones es admisible que algunos contenidos fluyan alrededor de una imagen, pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen:



Forzando a que un elemento no fluya alrededor de otro elemento posicionado mediante float

La propiedad clear permite modificar el comportamiento por defecto del posicionamiento flotante para forzar a un elemento a mostrarse debajo de cualquier caja flotante. La regla CSS que se aplica al segundo párrafo del ejemplo anterior es la siguiente:

```
<p style="clear: left;">...</p>
```

La definición formal de la propiedad clear se muestra a continuación:

Propiedad	clear
-----------	--------------

Propiedad	clear
Valores	none left right both inherit
Se aplica a	Todos los elementos de bloque
Valor inicial	none
Descripción	Indica el lado del elemento que no debe ser adyacente a ninguna caja flotante

La propiedad `clear` indica el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante. Si se indica el valor `left`, el elemento se desplaza de forma descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante en el lado izquierdo.

La especificación oficial de CSS explica este comportamiento como *"un desplazamiento descendente hasta que el borde superior del elemento esté por debajo del borde inferior de cualquier elemento flotante hacia la izquierda"*.

Si se indica el valor `right`, el comportamiento es análogo, salvo que en este caso se tienen en cuenta los elementos desplazados hacia la derecha.

El valor `both` despeja los lados izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante hacia la izquierda o hacia la derecha.

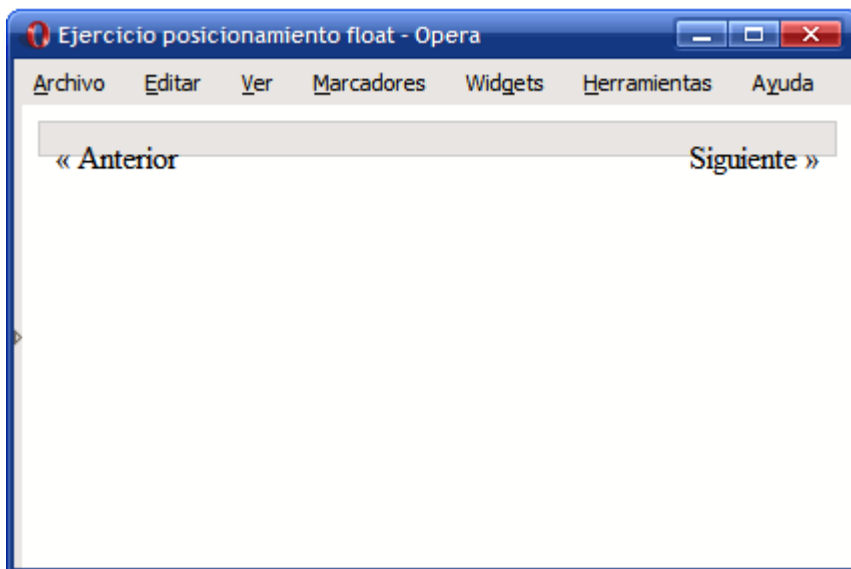
Como se verá más adelante, la propiedad `clear` es imprescindible cuando se crean las estructuras de las páginas web complejas.

Si se considera el siguiente código CSS y HTML:


```
#paginacion {
  border: 1px solid #CCC;
  background-color: #E0E0E0;
  padding: .5em;
}

.derecha { float: right; }
.izquierda { float: left; }
<div id="paginacion">
  <span class="izquierda">&laquo; Anterior</span>
  <span class="derecha">Siguiete &raquo;</span>
</div>
```

Si se visualiza la página anterior en cualquier navegador, el resultado es el que muestra la siguiente imagen:



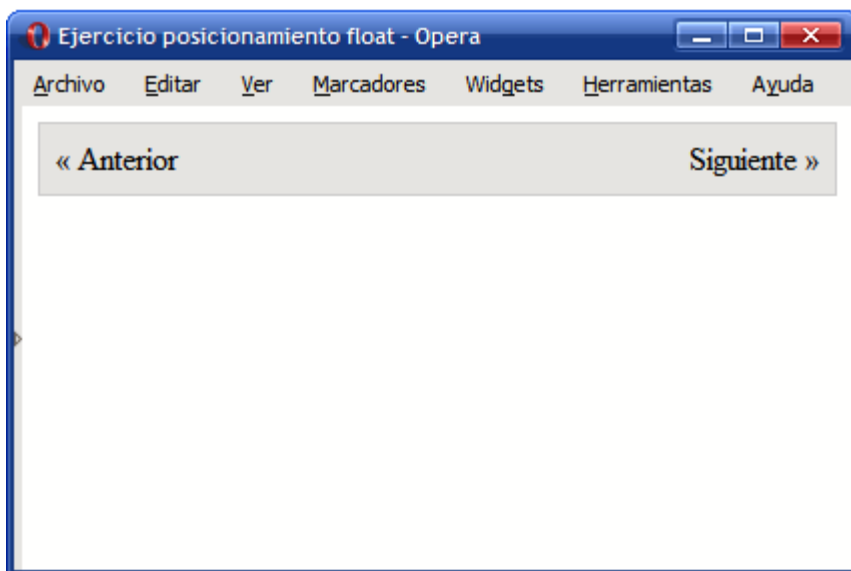
Visualización incorrecta de dos elementos posicionados mediante float

Los elementos Anterior y Siguiete se salen de su elemento contenedor y el resultado es visualmente incorrecto. El motivo de este comportamiento es que un elemento posicionado de forma flotante ya no pertenece al flujo normal de la página HTML. Por tanto, el elemento `<div id="paginacion">` en realidad no encierra ningún contenido y por eso se visualiza incorrectamente.

La solución consiste en utilizar la propiedad `overflow` (que se explica más adelante) sobre el elemento contenedor:

```
#paginacion {  
  border: 1px solid #CCC;  
  background-color: #E0E0E0;  
  padding: .5em;  
  overflow: hidden;  
}  
  
.derecha { float: right; }  
.izquierda { float: left; }
```

Si se visualiza de nuevo la página anterior en cualquier navegador, el resultado ahora sí que es el esperado:



Visualización correcta de dos elementos posicionados mediante float

Visualización

Además de las propiedades que controlan el posicionamiento de los elementos, CSS define otras cuatro propiedades para controlar su visualización: `display`, `visibility`, `overflow` y `z-index`.

Utilizando algunas de estas propiedades es posible ocultar y/o hacer invisibles las cajas de los elementos, por lo que son imprescindibles para realizar efectos avanzados y animaciones.

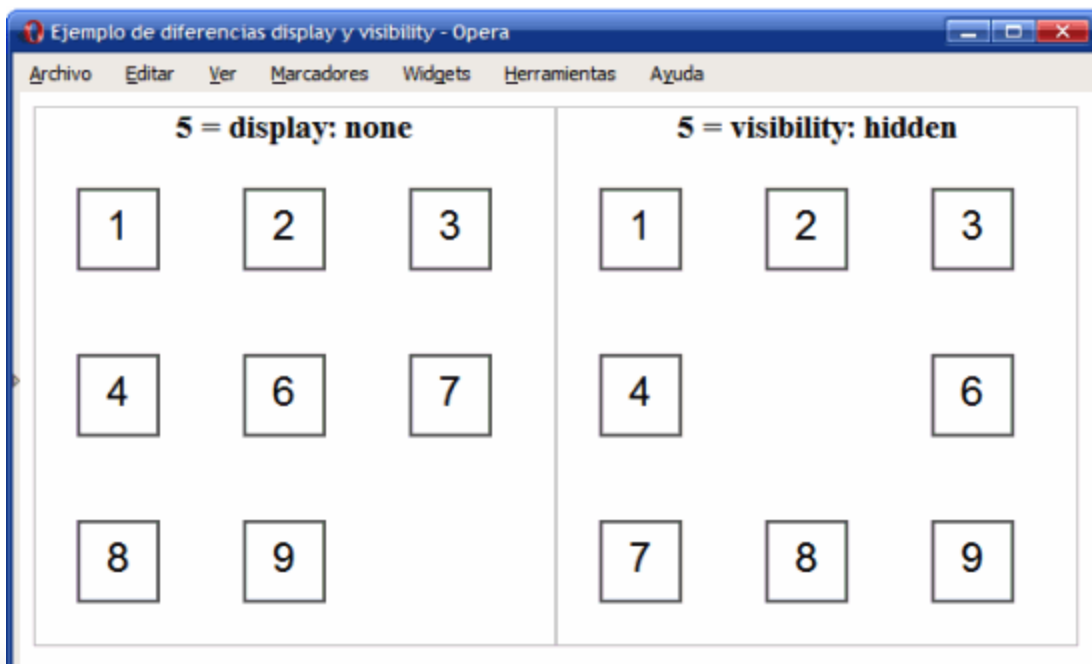
Propiedades display y visibility

Las propiedades `display` y `visibility` controlan la visualización de los elementos. Las dos propiedades permiten ocultar cualquier elemento de la página. Habitualmente se utilizan junto con JavaScript para crear efectos dinámicos como mostrar y ocultar determinados textos o imágenes cuando el usuario pincha sobre ellos.

La propiedad `display` permite ocultar completamente un elemento haciendo que desaparezca de la página. Como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.

Por otra parte, la propiedad `visibility` permite hacer invisible un elemento, lo que significa que el navegador crea la caja del elemento, pero no la muestra. En este caso, el resto de elementos de la página no modifican su posición, ya que, aunque la caja no se ve, sigue ocupando sitio.

La siguiente imagen muestra la diferencia entre ocultar la caja número 5 mediante la propiedad `display` o hacerla invisible mediante la propiedad `visibility`:



Diferencias visuales entre las propiedades display y visibility

En general, cuando se oculta un elemento no es deseable que siga ocupando sitio en la página, por lo que la propiedad `display` se utiliza mucho más que la propiedad `visibility`.

A continuación, se muestra la definición completa de la propiedad `display`:

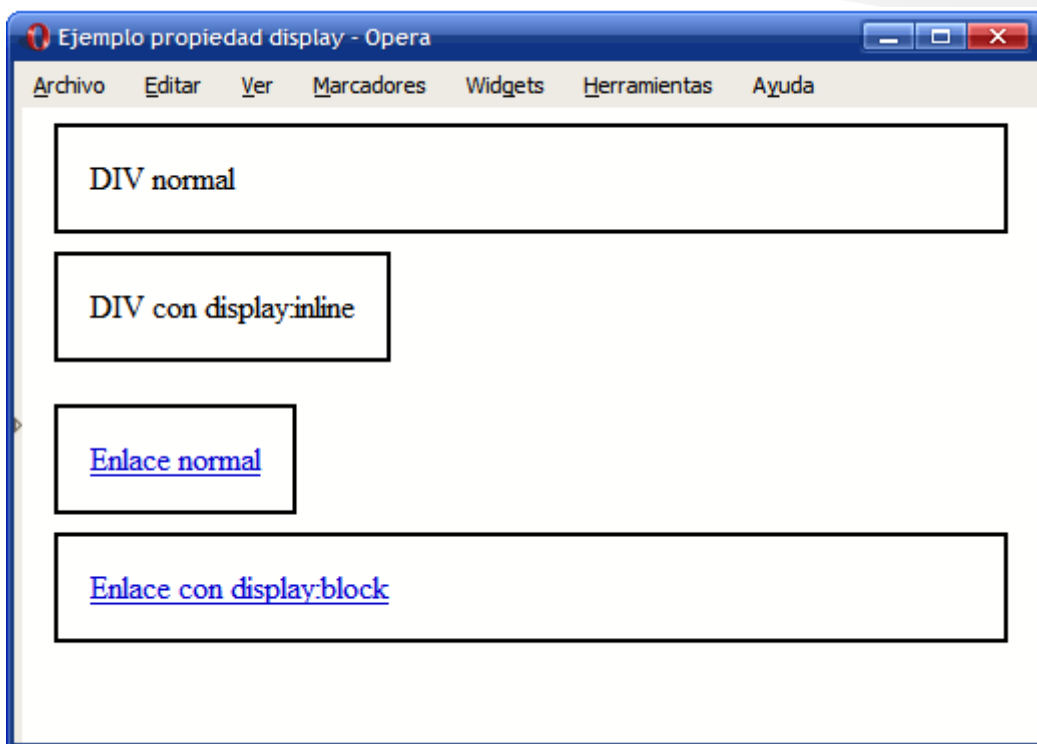
Propiedad	display
Valores	inline block none list-item run-in inline-block table inline-table table-row-group table-header-group table-footer-group table-row table-column-group table-column table-cell table-caption inherit
Se aplica a	Todos los elementos
Valor inicial	inline
Descripción	Permite controlar la forma de visualizar un elemento e incluso ocultarlo

Las posibilidades de la propiedad `display` son mucho más avanzadas que simplemente ocultar elementos. En realidad, la propiedad `display` modifica la forma en la que se visualiza un elemento.

Los valores más utilizados son `inline`, `block` y `none`. El valor `block` muestra un elemento como si fuera un elemento de bloque, independientemente del tipo de elemento que se trate. El valor `inline` visualiza un elemento en forma de elemento en línea, independientemente del tipo de elemento que se trate.

El valor `none` oculta un elemento y hace que desaparezca de la página. El resto de elementos de la página se visualizan como si no existiera el elemento oculto, es decir, pueden ocupar el espacio en el que se debería visualizar el elemento.

El siguiente ejemplo muestra el uso de la propiedad `display` para mostrar un elemento de bloque como si fuera un elemento en línea y para mostrar un elemento en línea como si fuera un elemento de bloque:



Ejemplo de propiedad display

Las reglas CSS del ejemplo anterior son las siguientes:

```
<div>DIV normal</div>
<div style="display:inline">DIV con display:inline</div>

<a href="#">Enlace normal</a>
<a href="#" style="display:block">Enlace con display:block</a>
```

Como se verá más adelante, la propiedad `display: inline` se puede utilizar en las listas (``, ``) que se quieren mostrar horizontalmente y la propiedad `display: block` se emplea frecuentemente para los enlaces que forman el menú de navegación.

Por su parte, la definición completa de la propiedad `visibility` es mucho más sencilla:

Propiedad	visibility
Valores	visible hidden collapse inherit

Propiedad	visibility
Se aplica a	Todos los elementos
Valor inicial	visible
Descripción	Permite hacer visibles e invisibles a los elementos

Las posibilidades de la propiedad `visibility` son mucho más limitadas que las de la propiedad `display`, ya que sólo permite hacer visibles o invisibles a los elementos de la página.

Inicialmente todas las cajas que componen la página son visibles. Empleando el valor `hidden` es posible convertir una caja en invisible para que no muestre sus contenidos. El resto de elementos de la página se muestran como si la caja todavía fuera visible, por lo que en el lugar donde originalmente se mostraba la caja invisible, ahora se muestra un hueco vacío.

Por último, el valor `collapse` de la propiedad `visibility` sólo se puede utilizar en las filas, grupos de filas, columnas y grupos de columnas de una tabla. Su efecto es similar al de la propiedad `display`, ya que oculta completamente la fila y/o columna y se pueden mostrar otros contenidos en ese lugar. Si se utiliza el valor `collapse` sobre cualquier otro tipo de elemento, su efecto es idéntico al valor `hidden`.

Relación entre `display`, `float` y `position`

Cuando se establecen las propiedades `display`, `float` y `position` sobre una misma caja, su interpretación es la siguiente:

Si `display` vale `none`, se ignoran las propiedades `float` y `position` y la caja no se muestra en la página.

Si `position` vale `absolute` o `fixed`, la caja se posiciona de forma absoluta, se considera que `float` vale `none` y la propiedad `display` vale `block` tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades `top`, `right`, `bottom` y `left`.

En cualquier otro caso, si `float` tiene un valor distinto de `none`, la caja se posiciona de forma flotante y la propiedad `display` vale `block` tanto para los elementos en línea como para los elementos de bloque.

Propiedad `overflow`

Normalmente, los contenidos de un elemento se pueden mostrar en el espacio reservado para ese elemento. Sin embargo, en algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda.

La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad `width` y/o `height`. Otra situación habitual es la de las líneas muy largas contenidas dentro de un elemento `<pre>`, que hacen que la página entera sea demasiado ancha.

CSS define la propiedad `overflow` para controlar la forma en la que se visualizan los contenidos que sobresalen de sus elementos.

Propiedad	<code>overflow</code>
Valores	<code>visible</code> <code>hidden</code> <code>scroll</code> <code>auto</code> <code>inherit</code>
Se aplica a	Elementos de bloque y celdas de tablas
Valor inicial	<code>visible</code>
Descripción	Permite controlar los contenidos sobrantes de un elemento

Los valores de la propiedad `overflow` tienen el siguiente significado:

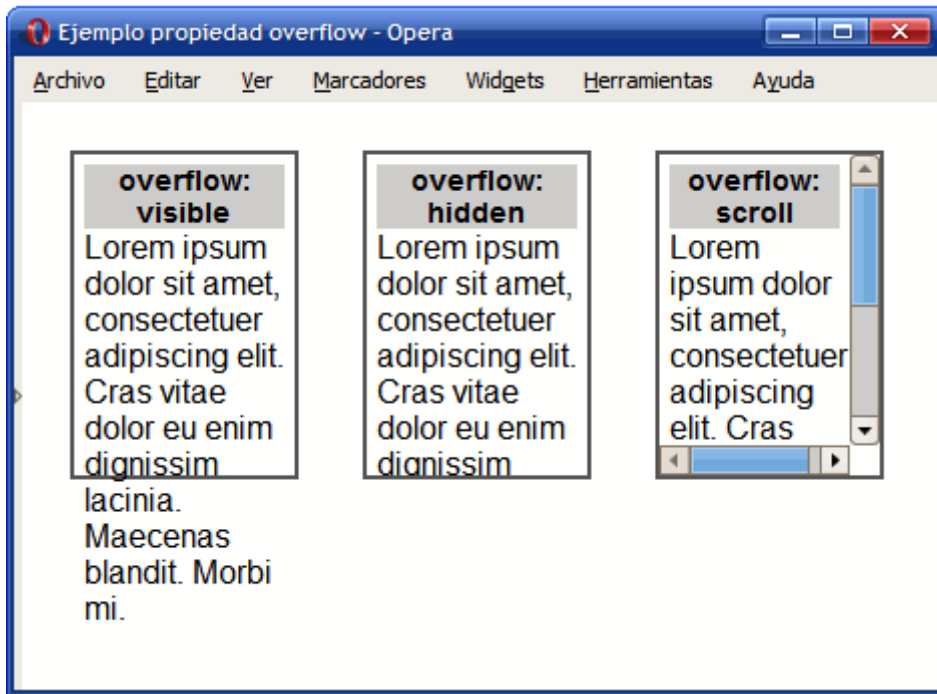
`visible`: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.

`hidden`: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.

`scroll`: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de *scroll* que permiten visualizar el resto del contenido.

auto: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad scroll.

La siguiente imagen muestra un ejemplo de los tres valores típicos de la propiedad overflow:



Ejemplo de propiedad overflow

El código HTML y CSS del ejemplo anterior se muestran a continuación:

```
div {
  display: inline;
  float: left;
  margin: 1em;
  padding: .3em;
  border: 2px solid #555;
  width: 100px;
  height: 150px;
  font: 1em Arial, Helvetica, sans-serif;
}

<div><h1>overflow: visible</h1> Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Cras vitae dolor eu enim dignissim lacinia. Maecenas
blandit. Morbi mi.</div>

<div style="overflow:hidden"><h1>overflow: hidden</h1> Lorem ipsum dolor
sit amet, consectetur adipiscing elit. Cras vitae dolor eu enim dignissim
lacinia. Maecenas blandit. Morbi mi.</div>

<div style="overflow:scroll"><h1>overflow: scroll</h1> Lorem ipsum dolor sit
amet, consectetur adipiscing elit. Cras vitae dolor eu enim dignissim lacinia.
Maecenas blandit. Morbi mi.</div>
```

Propiedad z-index

Además de posicionar una caja de forma horizontal y vertical, CSS permite controlar la posición tridimensional de las cajas posicionadas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional de un elemento se establece sobre un tercer eje llamado Z y se controla mediante la propiedad `z-index`. Utilizando esta propiedad es posible crear páginas complejas con varios niveles o capas.

A continuación, se muestra la definición formal de la propiedad `z-index`:

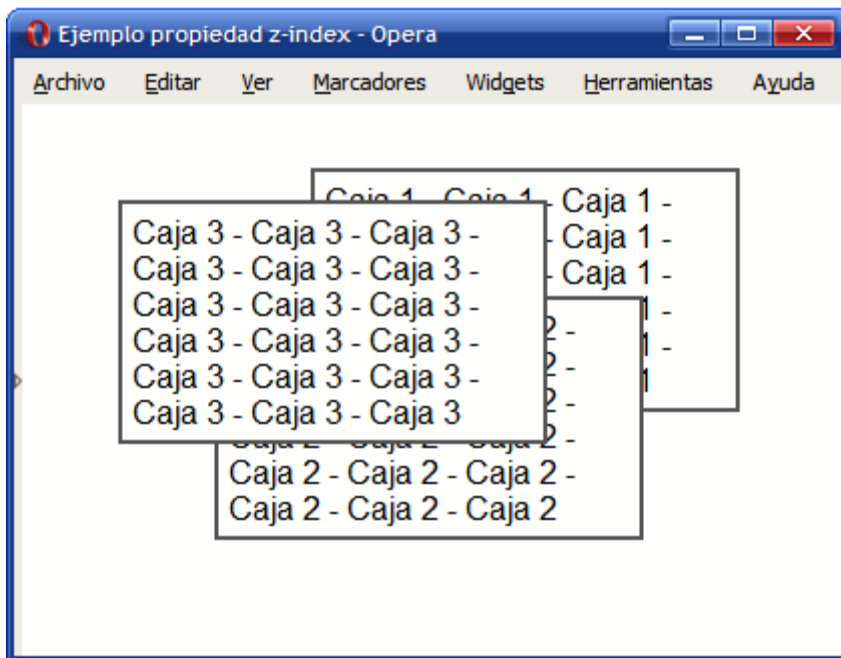
Propiedad	z-index
Valores	auto numero inherit
Se aplica a	Elementos que han sido posicionados explícitamente
Valor inicial	auto

Propiedad	z-index
Descripción	Establece el nivel tridimensional en el que se muestra el elemento

El valor más común de la propiedad z-index es un número entero. Aunque la especificación oficial permite los números negativos, en general se considera el número 0 como el nivel más bajo.

Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja. Un elemento con z-index: 10 se muestra por encima de los elementos con z-index: 8 o z-index: 9, pero por debajo de elementos con z-index: 20 o z-index: 50.

La siguiente imagen muestra un ejemplo de uso de la propiedad z-index:



Ejemplo de propiedad z-index

El código HTML y CSS del ejemplo anterior es el siguiente:

```
div { position: absolute; }
#caja1 { z-index: 5; top: 1em; left: 8em;}
#caja2 { z-index: 15; top: 5em; left: 5em;}
#caja3 { z-index: 25; top: 2em; left: 2em;}

<div id="caja1">Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 -
Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 -
Caja 1 - Caja 1 - Caja 1 - Caja 1</div>

<div id="caja2">Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 -
Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 -
Caja 2 - Caja 2 - Caja 2 - Caja 2</div>

<div id="caja3">Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 -
Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 -
Caja 3 - Caja 3 - Caja 3 - Caja 3</div>
```

La propiedad z-index sólo tiene efecto en los elementos posicionados, por lo que es obligatorio que la propiedad z-index vaya acompañada de la propiedad position. Si debes posicionar un elemento pero no quieres moverlo de su posición original ni afectar al resto de elementos de la página, puedes utilizar el posicionamiento relativo (position: relative).

EJERCICIO

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir los siguientes márgenes y rellenos:



Página original

El elemento #cabecera debe tener un relleno de 1em en todos los lados.

El elemento #menu debe tener un relleno de 0.5em en todos los lados y un margen inferior de 0.5em.

El resto de elementos (#noticias, #publicidad, #principal, #secundario) deben tener 0.5em de relleno en todos sus lados, salvo el elemento #pie, que sólo debe tener relleno en la zona superior e inferior.

Los elementos .articulo deben mostrar una separación entre ellos de 1em.

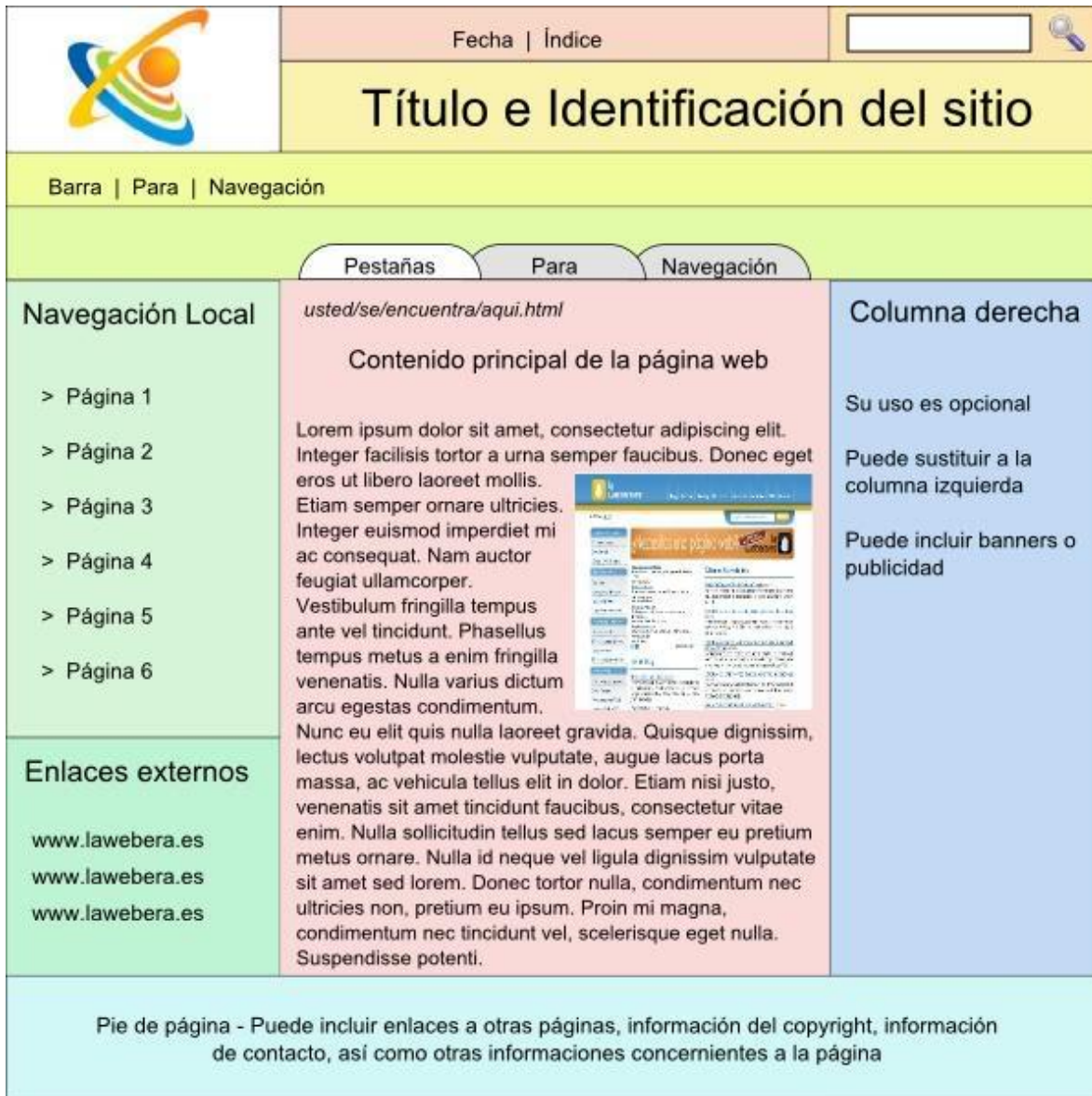
Las imágenes de los artículos muestran un margen de 0.5em en todos sus lados.

El elemento #publicidad está separado 1em de su elemento superior.

El elemento #pie debe tener un margen superior de 1em.

Ejemplo práctico de una maquetación web

Contenido y diseño de la estructura web



Cabecera y menú

De todas las partes de la estructura del **diseño web**, el encabezado es la que mayor tendencia tiene a repetirse entre páginas de un mismo sitio. Aun cuando el sitio tenga grandes diferencias en estructura y contenido entre sus páginas, si mantiene una unidad en su encabezado los usuarios percibirán la sensación de unidad que el sitio requiere.

El **encabezado** es una faja horizontal que ocupa todo el ancho de la página en la parte superior de la misma. A la izquierda del encabezado suele ubicarse el **logotipo** del sitio, que ocupa un área importante dentro del mismo, aunque suelen haber diferencias importantes de un sitio a otro. Esta imagen debe tener un enlace a la página principal del sitio, ya que es uno de los estándares más respetados por los diseñadores y una de las formas que tienen los usuarios de volver al inicio.

Desde el final del **logotipo**, ocupando el centro y la zona derecha del **encabezado**, frecuentemente se sitúan una serie de franjas de diferentes grosores. En la faja superior, se encuentran algunos **enlaces** generales de la página, como acceso a usuarios registrados, registros de usuarios nuevos, acceso a perfiles, salidas, etc., así como ayudas y **buscadores** internos. La zona más ancha generalmente se encuentra ocupada por el título de la página, nombre de empresa, slogan, etc.

La zona inferior del encabezado se emplea para ubicar los **links** de navegación y las **etiquetas de navegación**, uno de los elementos más importantes a la hora de ayudar a los usuarios a encontrar las páginas componentes del sitio web. Dependiendo de la complejidad del sitio, puede encontrarse solo uno de estos sistemas de navegación o los dos; por ejemplo, cuando el sitio está dividido en secciones, uno de ellos está destinado a estas divisiones, mientras que el otro sirve para navegar dentro de cada sección, aunque pueden emplearse otros métodos de clasificación de los contenidos y por consiguiente, otros métodos de organizar la navegación interna.

Otra forma de colaborar con la **exploración del sitio** por parte de los usuarios es la instalación de un **buscador interno**. También es aquí donde los usuarios buscan este tipo de elementos, ya que convencionalmente es aquí donde la mayoría de los diseñadores los ubican, aunque en algunos casos, puede encontrarse en la parte superior de alguna de las columnas laterales. A continuación, mostramos un ejemplo de la aplicación de un buscador interno en el encabezado.



Esta zona del diseño de la página, es la que menos reglas tiene respecto de su diseño, ya que el hecho de que en ella se encuentren los contenidos principales de la página, el diseño depende en gran medida de los mismos, sobre todo en lo que respecta al tipo de contenidos, la cantidad, la disposición que se desea emplear, etc. Sin embargo, hay que considerar algunas reglas importantes y que, salvo muy raras ocasiones, deben ser seguidas al pie de la letra.

En la parte superior de esta área se debe emplazar el título entre etiquetas <h1>, ya que este título tiene las palabras clave que los usuarios luego emplearán en sus búsquedas y será de gran importancia para que los buscadores indexen la página en forma correcta. Además, los usuarios buscarán en este punto el título de lo que se expone, siendo de extrema importancia que el mismo sepa a qué contenidos está accediendo.

En caso de que los contenidos sean muchos y haya que hacer desplazamientos importantes hacia abajo, es de mucha utilidad colocar al final del área algún tipo de salto al inicio, para evitar que el usuario deba realizar grandes desplazamientos. En el caso de que se empleen varias páginas para albergar contenidos relacionados, tanto al inicio de la página como al final de la misma es de gran utilidad colocar un navegador de páginas (del tipo Página anterior 1 – 2 – 3 ... Página siguiente) para que la navegabilidad entre ellas sea sencilla.

Pie de página de la web

Finalmente, debajo de todas las secciones, se coloca una faja horizontal en cuyo interior se colocan algunos elementos importantes, como el autor, copyright, acceso a diversas políticas del sitio y asuntos legales, datos de contacto, enlaces a sitios relacionados, enlaces internos, etc.

Ejercicio practico

****Crear una página con los elementos de contenido requeridos**

Aplicación de todo lo aprendido de maquetación

****Un ejemplo de los temas vistos**

Facilitando el diseño web

Frameworks para maquetación

Foundation, Bootstrap, Boilerplate

Foundation Zurb

A la hora de afrontar proyectos de diseño web responsivo, adaptables a cualquier pantalla, los diseñadores y programadores web contamos con multitud de herramientas que nos facilitan el trabajo, especialmente cuando se trata de resolver construcciones repetitivas y trilladas; es en estas circunstancias donde los *frameworks* CSS (Foundation, Bootstrap, Pure,...) y librerías de mixins y utilidades (Compass, Bourbon...) exhiben todo su potencial.

Foundation permite centrarnos en lo que realmente da **valor al proyecto**

Hay que tener en cuenta que los *frameworks* CSS se van a encargar de homogeneizar la experiencia de uso a lo largo de los diferentes navegadores disponibles en el mercado (browser compatibility), de forma que, como creadores, nos olvidaremos de las particularidades de cada uno de ellos.

Gracias a la inestimable ayuda que nos proporcionan, ahorramos tiempo y podemos centrarnos en el desarrollo de aquellas particularidades que hacen único a cada proyecto, *site*, o página. Dicho de otra forma, los diseñadores hemos logrado quitarnos de encima una considerable carga de trabajo realizada artesanalmente para poder dedicar ese tiempo a **aportar auténtico valor** (tanto técnico como creativo) en nuestros proyectos.



Miembros del equipo técnico de Cycle-IT durante la ponencia sobre Foundation 5

Como no podría ser de otra forma, en Cycle-IT contamos con **Bootstrap 3** para el desarrollo de aplicaciones web; un *framework* CSS indiscutible por su solidez y versatilidad. Pero no nos quedamos solo con Bootstrap, ya que también necesitábamos un *framework* ligero que nos facilitara la creación base, con características *Responsive Web Design* (RWB), de sitios web modernos y fácilmente mantenibles. Sitios web donde el amplio soporte para elementos UI de Bootstrap resulta irrelevante y en los que se requiere un control potente y flexible del layout así como un peso controlado de las librerías (CSS y JS) acopladas al proyecto. *Foundation fue el primer framework CSS en incorporar el **diseño web responsivo** (RWD)*

Para decantarnos por el más conveniente evaluamos los principales *frameworks* disponibles: Bootstrap 3 (de nuevo, ahora focalizado en este área), Foundation 5, Pure, Flat UI y YUI. Tras analizarlos en base a diferentes variables (funcionalidades aportadas, dependencias, peso, calidad de la documentación, fiabilidad y soporte y como no, popularidad) llegamos a la conclusión de que Foundation 5, un *framework* con varias versiones a sus espaldas, cubre plenamente nuestras necesidades.

Veamos a continuación algunas de sus características principales:

¿Qué diferencia Foundation 5 de otros *frameworks* CSS? Durante los últimos años han surgido incontables *frameworks* de desarrollo CSS, entre los cuales Foundation brilla con luz propia. Entonces... ¿qué hace tan popular a este *framework*? Para empezar fue el primero (en su versión 2) en incorporar el **diseño web responsivo** (RWD), lo que le avala como una sólida herramienta en esta materia. Hoy en día no se concibe diseñar un site únicamente para escritorio, hay que ofrecer una experiencia perfecta sobre cualquier pantalla, desde monitores de escritorio de gran formato hasta un pequeño *smartphone*, pasando por un ecosistema cada vez más fragmentado de dispositivos: *smartphones*, *tablets*, *phablets* (a medio camino de los anteriores) e incluso smartTVs (poco usadas en España para ver *websites*, pero chico, ahí están).

Bootstrap

Bootstrap es un framework (librerías de CSS) que nos facilita y estandariza el desarrollo de sitios web.

A partir de la versión 3.x ha sido implementado pensando que se adapte tanto a las pantallas de equipos de escritorio como a móviles y tablets.

Bootstrap ha sido desarrollada y es mantenida por la empresa Twitter y la ha liberado como un producto Open Source.

Tiene una filosofía muy intuitiva para el maquetado de sitios web que puede ser rápidamente aprendida por desarrolladores que no vienen del mundo del diseño web.

El corazón de este framework es un archivo CSS que lo podemos descargar del sitio <http://getbootstrap.com>.

Luego debemos enlazar a nuestra página con el archivo bootstrap.min.css y ya podemos utilizar esta librería.

La primera clase que podemos probar es `class="container"` (esta muestra todo el contenido del div centrado), además el framework ya define estilos por defecto para una gran cantidad de elementos HTML de la página, por ejemplo, define la tipografía de tipo Helvética.

Boilerplate

HTML5 Boilerplate es una poderosa plantilla maestra con la vamos a poder poner en marcha un proyecto en la nueva versión del lenguaje básico de la web.

Existen numerosas características que forman parte de HTML5 Boilerplate (ver siguiente párrafo) entre todas ellas destaca la idea de que las páginas web creadas con este recurso van a poder ser visualizadas de forma correcta incluso en navegadores que no dan soporte a HTML5 y CSS3.

Además de lo anterior HTML5 Boilerplate nos permite:

Especificar con que características descargarlo: Si lo queremos con Initializr, con script de Analytics, etc...

Incluye los condicionales adecuados para compatibilizarlo con versiones anteriores de IE sin agregar otro archivo CSS.

Tiene detalles interesantes para diseños adaptables: evita que el texto cambie cuando cambiamos la orientación en iOS, sin tener que quitar el Zoom.

Está muy detallado y organizado, lo que facilita acostumbrarse a su uso.

HTML5 Boilerplate se ofrece en tres "sabores" distintos: una versión comentada y explicada, una versión sin comentarios y una versión con mayores opciones de personalización.

Además, es interesante señalar que ya tenemos disponible la página de este proyecto en español con lo que el idioma no será una barrera para no aprovechar sus posibilidades. No obstante, lo anterior hay que señalar que en ocasiones el contenido en la versión española de HTML5 Boilerplate está algo desactualizado.

Introducción a bootstrap

Estructura, conceptos e implementación

¿Qué es bootstrap?

Bootstrap no es más que **un framework de css y js** que contiene las clases más utilizadas a la hora de maquetar <http://getbootstrap.com/>

Bondades

Entre sus bondades es que ya no debes preocuparte por el tamaño de la pantalla a la hora de crear webs responsivas.

Ya no debes preocuparte del diseño gráfico, ya que contiene muchos templates en base a este framework libres en la web.

Su línea de aprendizaje es muy alta, por lo que es muy sencillo de aprender.

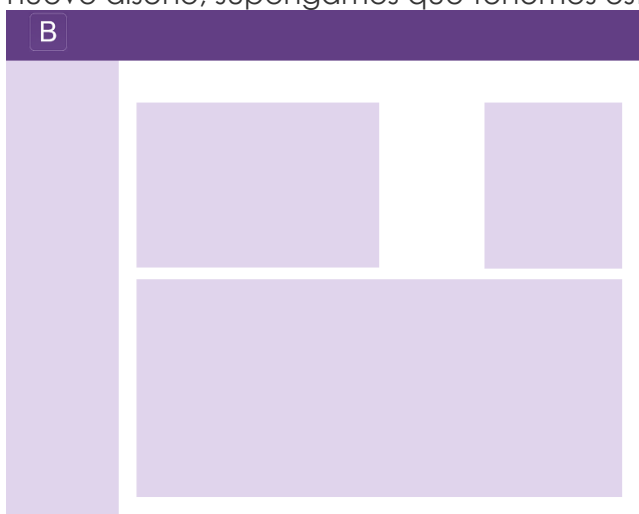
Contras

Si estas aprendiendo es necesario tener la web de bootstrap abierta para poder usar de manera correcta las etiquetas, un punto desfavorable si no posees internet.

Cuando ya dominas por completo el framework te das cuenta que hay muchas cosas que se le podrían agregar.

Sistema de grilla

Lo más importante a la hora de maquetar es ver cuantas secciones cuenta el nuevo diseño, supongamos que tenemos este prototipo:



Como podemos ver en lo que deseamos hacer tenemos que:
Tenemos un **header** en el cual se encuentra **el logo**.

Un **aside** en el que se encontrará el **menú**.

Un **container** en el que se encontrarán **secciones adicionales**.

Dentro del container un **div con 2 columnas** y un **div 100%**.

¿Cómo empezar?

Para comenzar se colocan las etiquetas html correspondientes al formato html estandar.


```
<!doctype html>
<html lang="es">
  <head>
    <title>Mi primera web con bootstrap</title>
    <meta charset="utf8">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css"> <!-- Importación del framework css -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap-theme.min.css"> <!-- importación del tema css (Esto es opcional) -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script> <!-- Importación del framework en js -->
  </head>
  <body>
  </body>
</html>
```

Ya tenemos la estructura base, con esto ya tenemos el bootstrap importado a nuestro código HTML y podemos comenzar a maquetar.

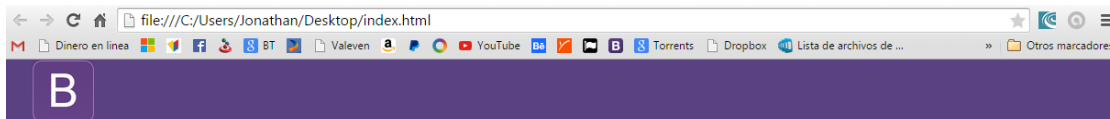
Paso 1

Vemos que tenemos un **header** y dentro una **imagen de bootstrap**, el mismo tiene un color hex **#5a4181**, (Le recomiendo usar el gotero de photoshop o gimp para saber el hex de el color).

Nos vamos al **Body** y creamos nuestro **header**

```
<header class="container-fluid" style="background:#5a4181">
  <div class="col-xs-1">
    
  </div>
</header>
```

El **resultado** sería algo como:



Aquí vemos una clase muy interesante **"col-xs-1"** Esta clase es la encargada del seccionamiento, supongamos que tenemos un div que **mide el 100%** de su espacio. Ese espacio se puede sectorizar o dividir por columnas colocando una clase. Que se descompone de la siguiente manera:

COL-XS-1
┌────────┴────────┐
Columna Tamaño Ancho

La parte más importante de la clase se descompone en el tamaño que es el tamaño de la pantalla a la que se va **renderizar**. Aquí entra el **responsive**.

Tu puedes indicar que un **div para los móviles** ocupa todo el ancho y que para **las tablets y desktops** ocupa solo el 20% del tamaño,

El **máximo valor** de ancho para una columna **es de 12**. que es igual que decir que ocupa el **100%** y **6 ocupa el 50%**.

Para indicar a qué pantalla se hace referencia se puede utilizar **xs=moviles**, **sm=tablets**, **lg=Desktops**.

Paso 2

Continuamos maquetando nuestro template de bootstrap, al navegar vemos que tenemos un **aside y un container**, si lo vemos por filas, vemos que el template se compone por 2 filas, **un header** y un **body**

En este caso vamos a crear la **fila número 2** que compone un **aside y un container**

```
<section class="container-fluid">  
<aside class="col-xs-12 col-sm-3" style="background:#dbd2ea"> Menu </aside>  
<article class="col-xs-12 col-sm-9"> Contenido </article>  
</section>
```



Ya nuestro template está **tomando forma**, ya hemos hecho el espacio del menú y falta crear la griya del contenido.

Para finalizar creamos los **divs dentro del container** y nos quedaria algo como:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Mi primera web con bootstrap</title>
  <meta charset="utf8">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css"> <!-- Importación del framework css -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap-theme.min.css"> <!-- importación del tema css (Esto es opcional) -->
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script> <!-- Importación del framework en js -->
</head>
<body>
  <header class="container-fluid" style="background:#5a4181">
    <div class="col-xs-1">
      
    </div>
  </header>
  <section class="container-fluid">
    <aside class="col-xs-12 col-sm-3" style="background:#dbd2ea; height:300px;">
      Menu
    </aside>
    <article class="col-xs-12 col-sm-9" style="height:300px;">
      <div class="col-xs-6" style="background:#dbd2ea">
        Primera columna
      </div>
      <div class="col-xs-3 col-xs-offset-3" style="background:#dbd2ea">
        Segunda columna con margen
      </div>
      <div class="col-xs-12" style="background:#dbd2ea">
        Div con 100%
      </div>
    </article>
  </section>
</body>
</html>
```

Diseñando para bootstrap

Crear una página web con bootstrap

**Crear un ejercicio con Bootstrap

Bibliografía

Autor:	Jon Duckett(author)
Título:	HTML and CSS: Design and Build Websites
País:	USA
Editorial:	Wiley; 1 edition
Año de publicación:	2011

Autor:	Mark Myers
Título:	A Smarter Way to Learn HTML & CSS: Learn it faster. Remember it longer. (Volume 2)

País:	USA
Editorial:	CreateSpace Independent Publishing Platform; 1 edition
Año de publicación:	2015

Cibergrafía

Autor:	Jorge González Villanueva
Título:	Fundamentos del diseño web front-end
Vínculo:	https://www.video2brain.com/mx/cursos/fundamentos-del-diseno-web-front-end
Editor:	Jorge González Villanueva
Año de publicación:	30/04/2015

Autor:	Marcos González Sancho
Título:	Fundamentos de la maquetación web para diseñadores
Vínculo:	https://www.video2brain.com/mx/cursos/fundamentos-de-la-maquetacion-web-para-disenadores
Editor:	Marcos González Sancho
Año de publicación:	18/12/2014