

# Desarrollo de Aplicaciones Telemáticas (2014-15)

## Grado en Ingeniería Tecnologías de Telecomunicación (URJC)

Jesús M. González Barahona, Gregorio Robles Martínez

<http://cursosweb.github.io>  
GSyC, Universidad Rey Juan Carlos

4 de abril de 2015



©2002-2015 Jesús M. González Barahona, Gregorio Robles.  
Algunos derechos reservados. Este artículo se distribuye bajo la licencia  
"Reconocimiento-CompartirIgual 3.0 España" de Creative Commons, disponible en  
<http://creativecommons.org/licenses/by-sa/3.0/es/deed.es>  
Este documento (o uno muy similar) está disponible en  
<http://cursosweb.github.io>

Las transparencias del tema “CSS - Hojas de estilo”  
están basadas en el libro de librosweb.es  
disponible en <http://www.librosweb.es/css/>  
Se ha pedido explícitamente autorización al autor original  
para realizar esta obra derivada con fines educativos.  
©Javier Eguiluz - Librosweb.es

- 1 Presentación de la asignatura
- 2 HTML: HyperText Markup Language
- 3 Hojas de estilo CSS
- 4 Hojas de estilo CSS3
- 5 Bootstrap
- 6 HTML5
- 7 5 cosas sobre HTML5
- 8 Algunos elementos HTML5
- 9 El canvas
- 10 Local Storage
- 11 Navegación Off-line
- 12 Detección de funcionalidad HTML5
- 13 Geolocalización
- 14 Web Workers
- 15 WebSocket
- 16 History API
- 17 La API de los servicios de Google
- 18 La API de Google+
- 19 Firefox OS

# Presentación de la asignatura

# Datos, datos, datos

- Profesores:
  - ① Jesús M. González Barahona ([jgb @ gsyc.urjc.es](mailto:jgb@gsyc.urjc.es))
  - ② Gregorio Robles ([grex @ gsyc.urjc.es](mailto:grex@gsyc.urjc.es))
- Grupo de Sistemas y Comunicaciones (GSyC)
- Despachos: 003 Biblioteca y 110 Departamental III
- Tutoría: X de 16:00 a 19:00 (en los propios Laboratorios)
- Horario: L (13:00-15:00) y M (13:00-15:00)
- Laboratorio 209 Laboratorios III (habitualmente)

Campus virtual: <http://campusonline.urjc.es/>  
Acortador: <http://pili.la/dat15>

# ¿De qué va todo esto?



*El viejo Internet ya no nos vale*



# En concreto...

- HTML, HTML5
- CSS, CSS3, Bootstrap
- JavaScript
- Canvas, WebWorkers, WebSockets...
- Video, Audio
- Geolocalización
- APIs
- ...

# Fundamentos filosofales de la asignatura

A photograph showing the back of a person with short brown hair, wearing a blue t-shirt, sitting at a desk and working on a laptop. The laptop screen displays several lines of programming code in a dark-themed editor. In the background, another computer monitor is visible, though its screen content is out of focus.

La programación es el lenguaje de  
la tecnología

# Lenguaje de Programación: JavaScript



Primer Mandamiento:  
Amarás JavaScript por encima de (casi) todo.

# Ejemplos

- Utilizando Google Maps para ver las antípodas  
<http://www.antipodemap.com/>
- Pintando en el Canvas  
<http://www.williammalone.com/articles/create-html5-canvas-javascript-drawing-app/>
- Transiciones CSS3  
<http://css3.bradshawenterprises.com/transitions/>
- Creando un Comecocos  
<http://www.canvasdemos.com/2010/07/30/pacman/>
- Creando webs modernas y ubicuas  
<http://gsyc.es/~grex/leonardo/>
- ...

# Metodología

- Objetivo principal: conceptos básicos de construcción de aplicaciones HTML5 portables
- Clases de teoría y de prácticas, pero...
- Teoría en prácticas, prácticas en teoría
- Uso de resolución de problemas para aprender
- Fundamentalmente, entender lo fundamental

# Fundamentos filosofales de la asignatura



Aprender no puede ser aburrido

# Las Clases

- Empezamos a las 13:00 en punto
- 10 minutos con un tema motivacional
  - Gadgets tecnológicos
  - Aplicaciones
  - Cuestiones interesantes
  - ...
- Generalmente, explicación de los conceptos más importantes y luego realización de ejercicios
- No hay descanso
- Ejercicios para hacer fuera de clase (y entregar)

# Fundamentos filosofales de la asignatura



El estudiante es el centro del aprendizaje

# Evaluación

- Teoría (obligatorio): nota de 0 a 4.
- Práctica final (obligatorio): nota de 0 a 2.
- Opciones y mejoras práctica final: nota de 0 a 3
- Prácticas incrementales: 0 a 1
- Ejercicios en foro/GitHub: nota de 0 a 2
- Nota final: Suma de notas, moderada por la interpretación del profesor
- Mínimo para aprobar:
  - Aprobado en teoría (2) y práctica final (1), y
  - 5 puntos de nota final en total

# Evaluación (2)

- Evaluación teoría: prueba escrita
- Evaluación prácticas incrementales (evaluación continua):
  - entre 0 y 1 (sobre todo las extensiones)
  - es muy recomendable hacerlas
- Evaluación práctica final
  - posibilidad de examen presencial para práctica final
  - ¡tiene que funcionar en el laboratorio!
  - enunciado mínimo obligatorio supone 1, se llega a 2 sólo con calidad y cuidado en los detalles
- Opciones y mejoras práctica final:
  - permiten subir la nota mucho
- Evaluación ejercicios (evaluación continua):
  - preguntas y ejercicios en foro/GitHub
- Evaluación extraordinaria:
  - prueba escrita (si no se aprobó la ordinaria)
  - nueva práctica final (si no se aprobó la ordinaria)

# Ejemplos de prácticas finales de otros años

- Alejandro García - Gascó Pérez:

<https://www.youtube.com/watch?v=iDS1l1ZE5ak>

- Alejandro Campos:

<https://www.youtube.com/watch?v=rzoK09BjSvI>

- Jesús Alonso: <https://www.youtube.com/watch?v=3QNGv0rA2cM>

- Sandra Álvarez:

<https://www.youtube.com/watch?v=-6j7FbR-mLc>

(puedes buscar en YouTube por muchos más ejemplos)

¡Ánimo!

Aquí se enseñan cómo son las cosas  
que se usan en el mundo real

Las buenas noticias son...  
que no son tan difíciles

# HTML: HyperText Markup Language

# Basic structure of a document

```
<!DOCTYPE html>
<html>
  <head>
    <title> Tittle </title>
  </head>
  <body>
    <p>This is a paragraph</p>
  </body>
</html>
```

**Ejercicio:** Escribe una página HTML simple, y mírala en el navegador.

# Basic structure of a document (2)

[Nota: en general, se describirá HTML5]

- <!DOCTYPE html> es la marca de HTML5
- En HTML 4.x era más complicado...
- En general, cada elemento se abre y se cierra
- La estructura de un documento es un árbol  
(cada elemento va dentro de otro)
- Elemento raíz: HTML
- Elementos bajo HTML: HEAD (indicaciones) y BODY (contenidos)

# Sintaxis básica

Sintaxis similar a la de XML:

- Etiquetas (elementos, marcas):

```
<p> ... </p>  
<p/>
```

- Atributos:

```
<p id="abstract">
```

- Las etiquetas han de estar “encapsuladas”:

se cierran en orden inverso al que se abrieron

(esto es, van siempre “unas dentro de otras”, son contenedores)

- Las etiquetas son nodos en el árbol,  
los atributos anotaciones de los nodos

- Caracteres “escapados”: &lt; (<) &gt; (>)

**Ejercicio:** Escribe una página HTML con cabecera (que tenga al menos un título) y cuerpo, con las dos sintaxis de etiquetas, y con elementos que tengan atributos.

# Elemento HTML

- Sintaxis básica:

```
<html lang="es">
```

- “lang” es el idioma (primario) del texto
- Contiene un elemento HEAD y un elemento BODY

Language tags in HTML and XML:

<http://www.w3.org/International/articles/language-tags/>

# Elemento HEAD

Información para el navegador y para bots.

Ejemplo:

```
<head>
  <meta charset="utf-8" />
  <title>El titulo</title>
  <link rel="stylesheet" type="text/css" href="main.css" />
  <link rel="alternate" type="application/atom+xml"
        title="Canal RSS"
        href="canal.rss" />
  <link rel="shortcut icon" href="/favicon.ico" />
</head>
```

# Elemento HEAD (2)

- META:
  - juego de caracteres  
(simplifica versiones pre-HTML5, http-equiv)
  - description
- LINK puede apuntar a complementos para la página
  - rel="stylesheet": hoja de estilo CSS
- LINK puede apuntar a otros recursos relacionados
  - rel="alternate": contenido equivalente de otros tipos (type)  
o en otros idiomas (hreflang)
  - rel="author": autor
  - rel="next", rel="prev": anterior, posterior
  - rel="shortcut icon": ícono de la página
  - Otros: license,nofollow, search, tag, etc.
- Otros: STYLE, SCRIPT (CSS o JavaScript embebidos)

# Elementos en BODY

- H1 - H6: cabeceras (headings)
- P: párrafos de texto
- A: ancla (anchor) (absoluto a url, absoluto a recurso, relativo)

```
<a href="http://linkedsite/url.html">Documento</a>
<a href="/url.html">Documento en mismo sitio</a>
<a href="url.html">Documento en mismo sitio y "dir"</a>
```

- UL, OL, DL: listas (sin ordenar, ordenadas, de definiciones)

```
<ul>
    <li>Un elemento</li>
    <li>Otro elemento</li>
</ul>
```

# Elementos en BODY (2)

- Tabla

```
<table>
  <tr>
    <td>Primera fila, primera columna</td>
    <td>Primera fila, segunda columna</td>
  </tr>
  <tr>
    <td>Segunda fila, primera columna</td>
    <td>Segunda fila, segunda columna</td>
  </tr>
</table>
```

# Elementos en BODY (3)

- DIV: contenedor genérico (referencias para CSS)
- HEADER, FOOTER, NAV, ARTICLE, SECTION, ASIDE: elementos de sección
- IMG: imágenes

```

```

```

```

- MAP, AREA: mapa de imagen en el lado del cliente

# Elementos en BODY (4)

- FORM, FIELDSET, LABEL, INPUT

```
<form action="recurso" method="post">
  <fieldset>
    <legend>Formulario</legend>

    <label>Nombre</label>
    <input type="text" name="nombre"><br />

    <label>Apellido</label>
    <input type="text" name="apellido"><br />

    <input type="submit" name="persona">
  </fieldset>
</form>
```

# Material complementario

- HyperText Markup Language (Wikibook):  
[http://en.wikibooks.org/wiki/HTML\\_Programming](http://en.wikibooks.org/wiki/HTML_Programming)
- HTML5: A tutorial for beginners:  
<http://www.html-5-tutorial.com/>
- Dive into HTML5:  
<http://diveintohtml5.info>
- HTML5 (Wikipedia):  
<http://en.wikipedia.org/wiki/HTML5>
- Web Fundamentals (Code Academy):  
<http://www.codecademy.com/tracks/web>

# Hojas de estilo CSS

# ¿Qué es CSS?

- Es un lenguaje de hojas de estilos creado para **controlar el aspecto** o presentación de los documentos electrónicos definidos con HTML
- Es la mejor forma de **separar los contenidos y su presentación** y es imprescindible para crear páginas web complejas
  - Obliga a crear documentos HTML bien definidos y con significado completo (también llamados *documentos semánticos*)
  - Mejora la accesibilidad del documento
  - Reduce la complejidad de su mantenimiento
  - Permite visualizar el mismo documento en infinidad de dispositivos diferentes

# Antes del CSS

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<title>Ejemplo de estilos sin CSS</title>
</head>

<body>
<h1><font color="red" face="Arial" size="5">
    Titular de la página
</font></h1>
<p><font color="gray" face="Verdana" size="2">
    Un párrafo de texto no muy largo.
</font></p>
</body>
</html>
```

# Con CSS

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<title>Ejemplo de estilos con CSS</title>
<style type="text/css">
  h1 { color: red; font-family: Arial; font-size: large; }
  p { color: gray; font-family: Verdana; font-size: medium; }
</style>
</head>

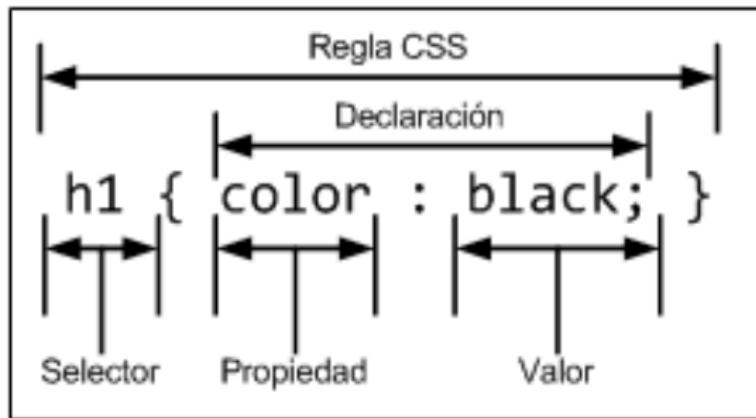
<body>
  <h1>Titular de la página</h1>
  <p>Un párrafo de texto no muy largo.</p>
</body>
</html>
```

# CSS en un documento HTML

Se pueden integrar instrucciones CSS de varias maneras en un documento HTML:

- ① Incluir CSS en el mismo documento HTML
- ② Definir CSS en un archivo externo
- ③ Incluir CSS en los elementos HTML

# Glosario Básico (I)



- **Regla:** cada uno de los estilos que componen una hoja de estilos CSS. Cada regla está compuesta de una parte de “selectores”, un símbolo de “llave de apertura” ({}), otra parte denominada “declaración” y por último, un símbolo de “llave de cierre” (}).
- **Selector:** indica el elemento o elementos HTML a los que se aplica la regla CSS.

# Glosario Básico (y II)



- **Declaración:** especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- **Propiedad:** característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
- **Valor:** establece el nuevo valor de la característica modificada en el elemento.

CSS 2.1 define 115 propiedades, mientras que CSS 3 define 239 propiedades.

# Selectores

- A un mismo elemento HTML se le pueden aplicar varias reglas
- Cada regla puede aplicarse a un número ilimitado de elementos
- Cuando el selector de dos o más reglas CSS es idéntico, se pueden agrupar las declaraciones de las reglas para hacer las hojas de estilos más eficientes
- Cuando se establece el valor de una propiedad CSS en un elemento, sus elementos descendientes heredan de forma automática el valor de esa propiedad

CSS 2.1 incluye una docena de tipos diferentes de selectores, que permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web.

# Selectores básicos

- ① Selector universal
- ② Selector de tipo o etiqueta
- ③ Selector descendente
- ④ Selector de clase
- ⑤ Selector de identidad

# Selector Universal

- No se utiliza habitualmente
- Generalmente es equivalente para poner estilo a `<body>`
- Se suele combinar con otros selectores y además, forma parte de algunos hacks muy utilizados

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

# Selector de tipo o etiqueta

- Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector
- Se pueden agrupar todas las reglas individuales en una sola regla con un selector múltiple.
- Buena práctica: agrupar las propiedades comunes de varios elementos en una única regla CSS y posteriormente definir las propiedades específicas de esos mismos elementos

```
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; }
```

# Selector descendente

- Selecciona los elementos que se encuentran dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento.

```
p span { color: red; }  
[...]  
<p>  
  ...  
  <span>texto1</span>  
  ...  
  <a href="">...<span>texto2</span></a>  
  ...  
</p>
```

Al resto de elementos `<span>` de la página que no están dentro de un elemento `<p>`, no se les aplica la regla CSS anterior.

# Ejercicio

¿Qué elementos se seleccionarían con estos tipos de selectores?

- p a span em { text-decoration: underline; }
- p, a, span, em { text-decoration: underline; }
- p a { color: red; }
- p \* a { color: red; }

# Selector de clase

- Se utiliza el atributo class de HTML sobre ese elemento para indicar directamente la regla CSS que se le debe aplicar
- Se crea en el archivo CSS una nueva regla llamada destacado con todos los estilos que se van a aplicar al elemento
- Se prefija el valor del atributo class con un punto (.)

```
.destacado { color: red; }
[...]
<p class="destacado">Lorem ipsum dolor sit amet...</p>
<p>Nunc sed lacus et
<a href="#" class="destacado">est adipiscing</a>
  accumsan...</p>
  <p>Class aptent taciti <em class="destacado">sociosqu
ad</em> litora...</p>
```

# Selector de clase más específico

- Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico.

```
p.destacado { color: red }
[...]
<p class="destacado">Lorem ipsum dolor sit amet...</p>
<p>Nunc sed lacus et <a href="#" class="destacado">est
adipiscing</a> accumsan...</p>
<p>Class aptent taciti <em class="destacado">sociosqu
ad</em> litora...</p>
```

# Ejercicio

¿Qué elementos se seleccionarían con estos tipos de selectores?

- p.aviso { ... }
- p .aviso { ... }
- p, .aviso { ... }
- \*.aviso { ... }

# Selectores de identificador

- Aplica estilos CSS a un único elemento de la página
- El valor del atributo id no se puede repetir en dos elementos diferentes de una misma página

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

```
<p>Primer párrafo</p>
<p id="destacado">Segundo párrafo</p>
<p>Tercer párrafo</p>
```

# Ejercicio

¿Qué elementos se seleccionarían con estos tipos de selectores?

- p#aviso { ... }
- p #aviso { ... }
- p, #aviso { ... }
- \*#aviso { ... }

¿Tienen sentido estos selectores? ¿Cuándo?

# Ejercicio: Combinación de selectores

¿Qué elementos se seleccionarían con estos tipos de selectores?

- `.aviso .especial { ... }`
- `div.aviso span.especial { ... }`
- `ul#menuPrincipal li.destacado a#inicio { ... }`

# Colisión de estilos (simplificado)

- ① Cuanto más específico sea un selector, más importancia tiene su regla asociada.
- ② A igual especificidad, se considera la última regla indicada.

# Unidades de medida

- Unidades absolutas
  - in, cm, mm, pt, pc
- Unidades relativas
  - em, ex, px
- Porcentajes

En general, se recomienda el uso de unidades relativas siempre que sea posible

Normalmente se utilizan píxel y porcentajes para definir el layout del documento (básicamente, la anchura de las columnas y de los elementos de las páginas) y em y porcentajes para el tamaño de letra de los textos.

# Colores

- Palabras clave
  - aqua, black, blue, fuchsia, gray...
- RGB decimal
- RGB porcentual
- RGB hexadecimal

# Tipos de elementos (I)

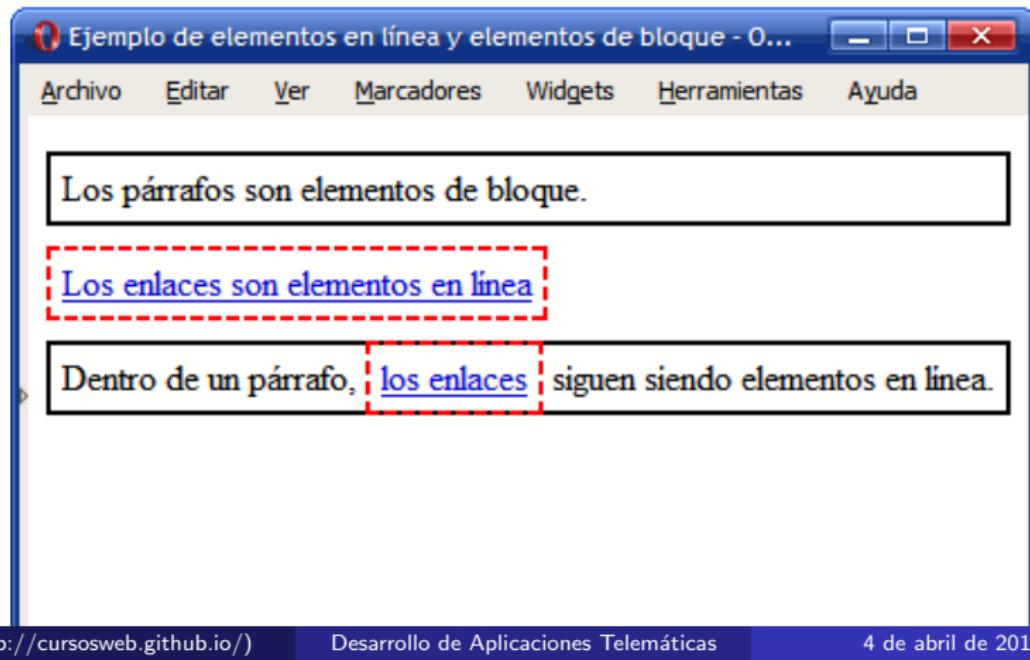
El estándar HTML clasifica a todos sus elementos en dos grandes grupos:  
Elementos de línea:

- Los elementos en línea (“inline elements” en inglés) no empiezan necesariamente en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

# Tipos de elementos (II)

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

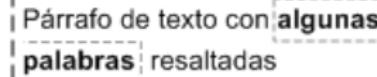


# El modelo de cajas

- Es el comportamiento de CSS que hace que todos los elementos de las páginas se representen mediante cajas rectangulares
- Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento

```
<p>Párrafo de texto con <strong>algunas  
palabras</strong> resaltadas</p>
```

```
<p>Otro párrafo</p>
```



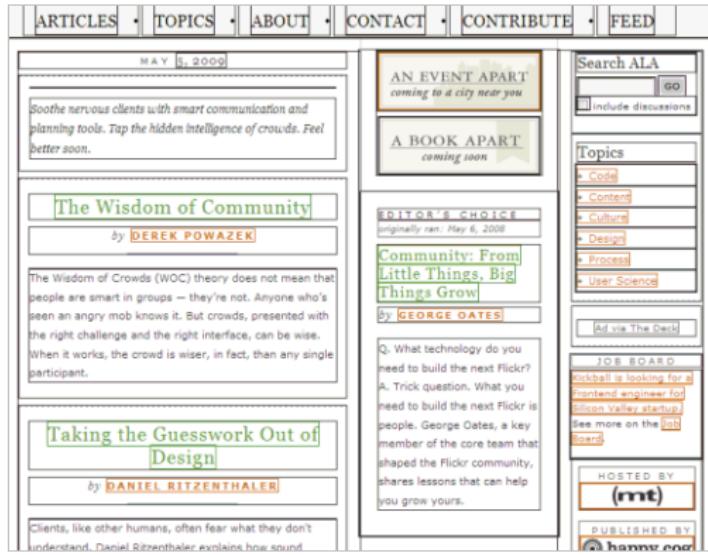
Párrafo de texto con **algunas**  
**palabras** resaltadas



Otro párrafo

# El modelo de cajas (II)

- No son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde

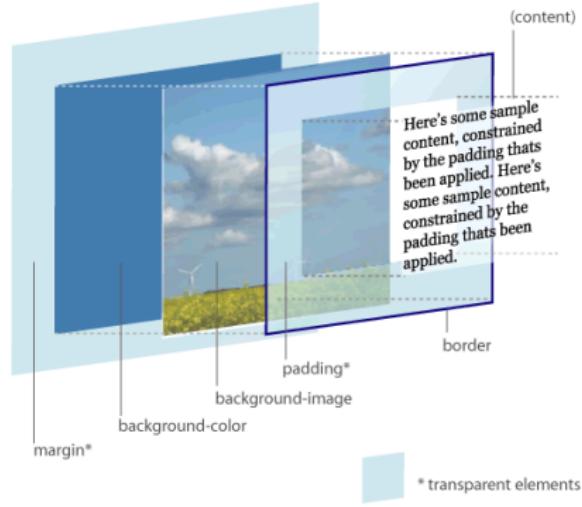


Ejemplo de <http://www.alistapart.com/> después de forzar a que todas las cajas muestren su borde

# El modelo de cajas (III)

- 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



Representación tridimensional del box model de CSS (Esquema utilizado con permiso de <http://www.hicksdesign.co.uk/boxmodel/>)

# Partes que componen cada caja

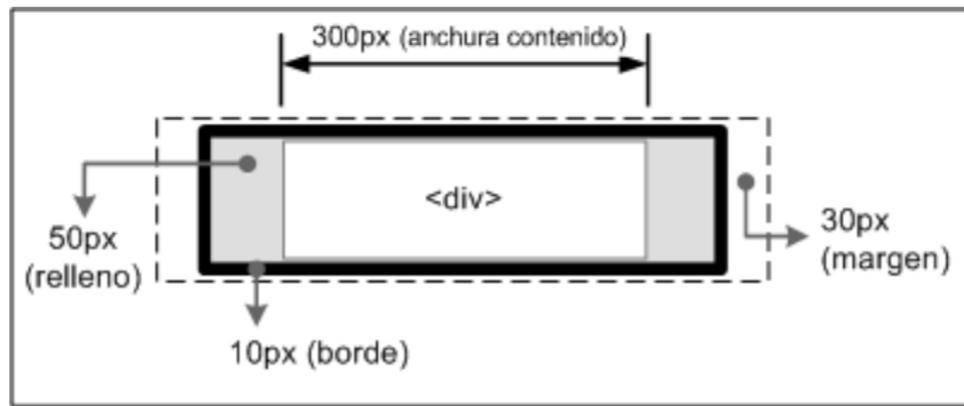
- **Contenido** (content): se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- **Relleno** (padding): espacio libre opcional existente entre el contenido y el borde.
- **Borde** (border): línea que encierra completamente el contenido y su relleno.
- **Imagen de fondo** (background image): imagen que se muestra por detrás del contenido y el espacio de relleno.
- **Color de fondo** (background color): color que se muestra por detrás del contenido y el espacio de relleno.
- **Margen** (margin): separación opcional existente entre la caja y el resto de cajas adyacentes.

# Margen, relleno, bordes y modelo de cajas (I)

- El margen, el relleno y los bordes establecidos a un elemento determinan la anchura y altura final del elemento

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

# Margen, relleno, bordes y modelo de cajas (y II)

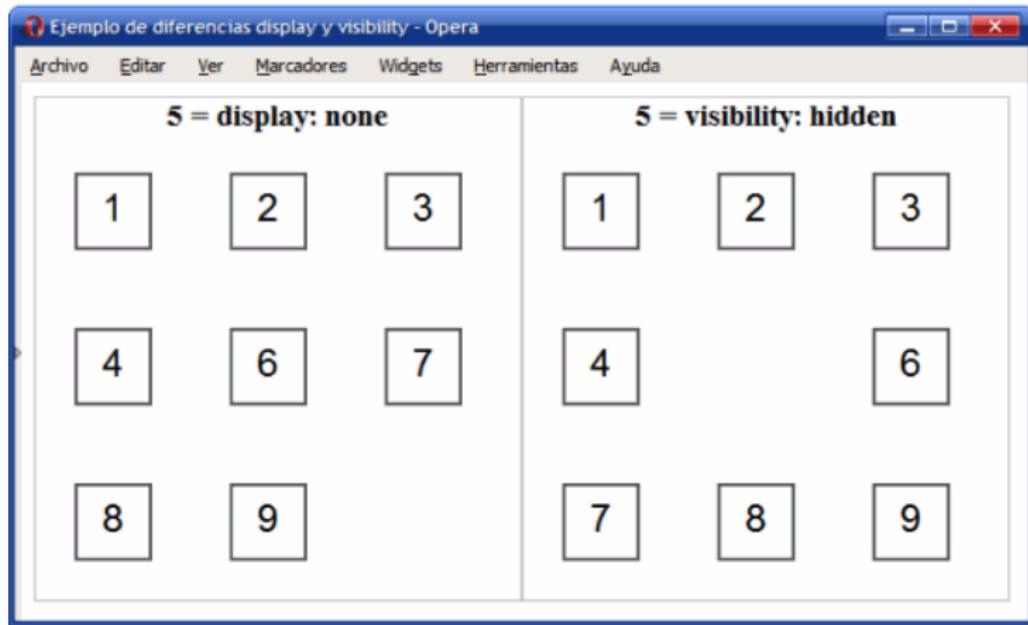


De esta forma, la anchura del elemento en pantalla sería igual a la suma de la anchura original, los márgenes, los bordes y los rellenos:  
 $30\text{px} + 10\text{px} + 50\text{px} + 300\text{px} + 50\text{px} + 10\text{px} + 30\text{px} = 480 \text{ píxel}$

# Visualización

- CSS define otras cuatro propiedades para controlar su visualización: display, visibility, overflow y z-index.
- 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.
- La propiedad display también permite modificar el comportamiento de un elemento a bloque (block) o en línea (inline).
- 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.

# Diferencias entre display y visibility



# Propiedades margin

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

Valores	$< medida >$ — $< porcentaje >$ — 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

**Cuadro:** Definición de la propiedad *margin-top*, *margin-right*, *margin-bottom*, *margin-left* de CSS

# Propiedad margin (propiedad *shorthand*)

Propiedad	<b>margin</b>
Valores	( <i>&lt; medida &gt;</i> — <i>&lt; porcentaje &gt;</i> — auto ) 1, 4 — inherit
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

Cuadro: Definición de la propiedad margin de CSS

# Propiedad margin (propiedad *shorthand*) (y II)

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.

# CSS: Consideraciones adicionales

# CSS Consideraciones Adicionales (transparencias de referencia)

# Orden de visualización

- El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos)
- En el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos)
- Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos)
- Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza
- Si la imagen de fondo no cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo.

# Propiedad width

Propiedad	<b>width</b>
Valores	$<\text{medida}>$ — $<\text{porcentaje}>$ — 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

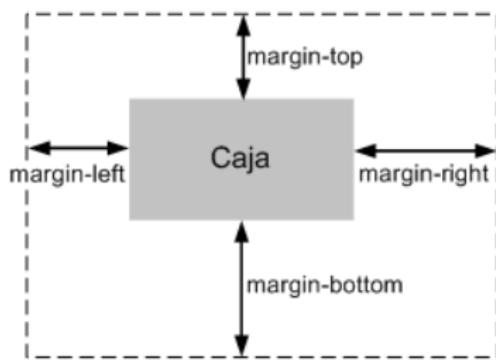
Cuadro: Definición de la propiedad *width* de CSS

# Propiedad height

Propiedad	<b>height</b>
Valores	$<\text{medida}>$ — $<\text{porcentaje}>$ — auto — inherit
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

Cuadro: Definición de la propiedad *height* de CSS

# Márgenes



- En vez de utilizar la etiqueta `<blockquote>` de HTML, debería utilizarse la propiedad `margin-left` de CS
- 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

# El margen vertical

Es algo peculiar:

- 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.
- 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
- 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.

# Relleno

Propiedades **padding-top, padding-right, padding-bottom, padding-left**

Valores	$<medida>$ — $<porcentaje>$ — 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

**Cuadro:** Definición de la propiedad padding-top, padding-right, padding-bottom, padding-left de CSS

Propiedad	<b>padding</b>
Valores	( <i>&lt; medida &gt;</i> — <i>&lt; porcentaje &gt;</i> ) 1, 4 — inherit
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

Cuadro: Definición de la propiedad padding de CSS

# Anchura de los bordes

Propiedades	<b>border-top-width, border-right-width, border-bottom-width, border-left-width</b>
Valores	( <i>&lt; medida &gt;</i> — thin — medium — thick ) — inherit
Se aplica a	Todos los elementos
Valor inicial	Medium
Descripción	Establece la anchura de cada uno de los cuatro bordes de los elementos

**Cuadro:** Definición de la propiedad border-top-width, border-right-width, border-bottom-width, border-left-width de CSS

# Anchura de los bordes (shorthand)

Propiedad	<b>border-width</b>
Valores	( <i>&lt; medida &gt;</i> — 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

**Cuadro:** Definición de la propiedad border-width de CSS

# Color de los bordes

Propiedades **border-top-color, border-right-color, border-bottom-color, border-left-color**

---

Valores < color > — transparent — inherit

---

Se aplica a Todos los elementos

---

Valor -  
initial

---

Descripción Establece el color de cada uno de los cuatro bordes de los elementos

**Cuadro:** Definición de la propiedad border-top-color, border-right-color, border-bottom-color, border-left-color de CSS

# Color de los bordes (shorthand)

Propiedad	<b>border-color</b>
Valores	( <i>&lt; color &gt;</i> — transparent ) 1, 4 — inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de todos los bordes del elemento

**Cuadro:** Definición de la propiedad border-color de CSS

# Estilo de los bordes

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
Valor inicial	none
Descripción	Establece el estilo de cada uno de los cuatro bordes de los elementos

**Cuadro:** Definición de la propiedad border-top-style, border-right-style, border-bottom-style, border-left-style de CSS

# Estilo de los bordes *shorthand*

Propiedad	<b>border-style</b>
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

**Cuadro:** Definición de la propiedad border-style de CSS

# Propiedades *shorthand* para bordes

Propiedades **border-top**, **border-right**, **border-bottom**, **border-left**

---

Valores (*< 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

**Cuadro:** Definición de la propiedad border-top, border-right, border-bottom, border-left de CSS

# Propiedad *shorthand* para borde (global)

Propiedad	<b>border</b>
Valores	( < 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

Cuadro: Definición de la propiedad border de CSS

# Más sobre bordes

- 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
- 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;  
}
```

# Fondos

- Puede ser un color simple o una imagen.
- 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
- Se puede 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.

# Propiedad background-color

Propiedad	<b>background-color</b>
Valores	<i>color</i> > — transparent — inherit
Se aplica a	Todos los elementos
Valor inicial	transparent
Descripción	Establece un color de fondo para los elementos

Cuadro: Definición de la propiedad background-color de CSS

# Propiedad background-image

Propiedad	<b>background-image</b>
Valores	<i>url</i> > — none — inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece una imagen como fondo para los elementos

Cuadro: Definición de la propiedad background-image de CSS

# Propiedad background-repeat

Propiedad	<b>background-repeat</b>
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

Cuadro: Definición de la propiedad background-repeat de CSS

# Propiedad background-position

Propiedad	<b>background-position</b>
Valores	( ( < porcentaje > — < medida > — left — center — right ) ( < porcentaje > — < medida > — top — center — bottom )? ) — ( ( left — 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

Cuadro: Definición de la propiedad background-position de CSS

# Propiedad background-attachment

Propiedad	<b>background-attachment</b>
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 junto con la ventana

**Cuadro:** Definición de la propiedad background-attachment de CSS

# Propiedad *shorthand* background

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

Cuadro: Definición de la propiedad background de CSS

# Posicionamiento y visualización

- Los navegadores crean y posicionan de forma automática todas las cajas que forman cada página HTML
- El diseñador puede modificar la posición en la que se muestra cada caja.
- Existen **cinco tipos de posicionamiento** definidos para las cajas

# Tipos de posicionamiento

- ① **Normal o estático:** posicionamientos si no se indica lo contrario.
- ② **Relativo:** consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.
- ③ **Absoluto:** la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.
- ④ **Fijo:** variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.
- ⑤ **Flotante:** desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

# Propiedad position

Propiedad	<b>position</b>
Valores	static — relative — absolute — fixed — inherit
Se aplica a	Todos los elementos
Valor inicial	static
Descripción	Selecciona el posicionamiento con el que se mostrará el elemento

Cuadro: Definición de la propiedad position de CSS

# Significados propiedad position

- static: corresponde al posicionamiento normal o estático. Si se utiliza este valor, se ignoran los valores de las propiedades top, right, bottom y left que se verán a continuación.
- relative: corresponde al posicionamiento relativo. El desplazamiento de la caja se controla con las propiedades top, right, bottom y left.
- absolute: corresponde al posicionamiento absoluto. El desplazamiento de la caja también se controla con las propiedades top, right, bottom y left, pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor.
- fixed: corresponde al posicionamiento fijo. El desplazamiento se establece de la misma forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

# Pseudo-clases

Como con los atributos id o class no es posible aplicar diferentes estilos a un mismo elemento en función de su estado, CSS introduce un nuevo concepto llamado pseudo-clases. Por ejemplo, en enlaces:

- **:link**: enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- **:visited**: enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario. El historial de enlaces visitados se borra automáticamente cada cierto tiempo y el usuario también puede borrarlo manualmente.
- **:hover**: enlace sobre el que el usuario ha posicionado el puntero del ratón.
- **:active**: enlace que está pinchando el usuario. Los estilos sólo se aplican desde que el usuario pincha el botón del ratón hasta que lo suelta.

# Propiedades top, right, bottom, left

## Propiedades **top, right, bottom, left**

---

Valores       $< medida >$  —  $< porcentaje >$  — auto — inherit

---

Se aplica a    Todos los elementos posicionados

---

Valor inicial    auto

---

Descripción    Indican el desplazamiento horizontal y vertical del elemento respecto de su posición original

---

**Cuadro:** Definición de la propiedad top, right, bottom, left de CSS

# Posicionamiento normal (o estático)

- Utilizado por defecto por los navegadores
- Sólo se tiene en cuenta si el elemento es de bloque o en línea, sus propiedades width y height y su contenido.
- 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.
- 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.



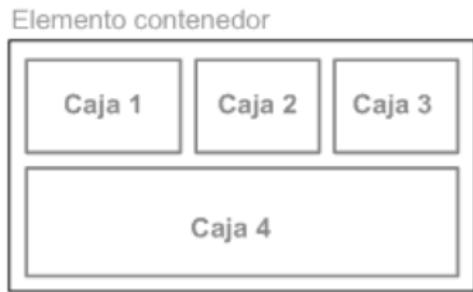
# Posicionamiento normal (o estático) (y II)

- Los elementos en línea forman los “contextos de formato en línea”. Las cajas se muestran una detrás de otra de forma horizontal comenzando desde la posición más a la izquierda de su elemento contenedor.
- 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

- 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.
- 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.



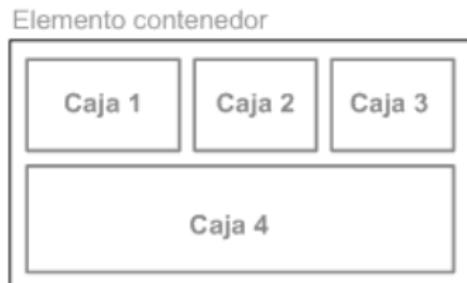
Posicionamiento normal



Posicionamiento relativo de la caja 2

# Posicionamiento absoluto

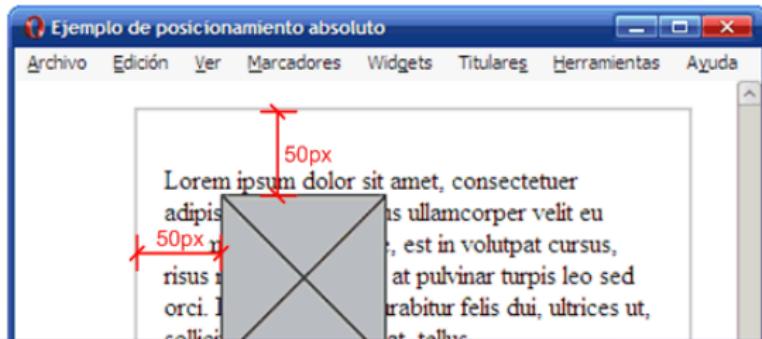
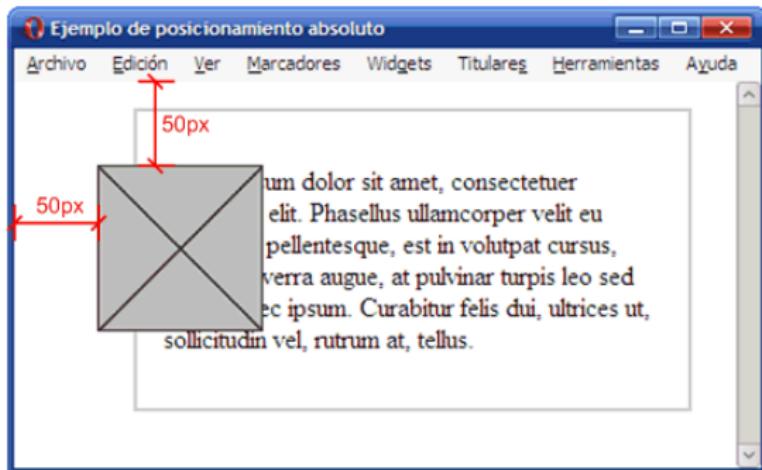
- Se emplea para establecer de forma exacta la posición en la que se muestra la caja de un elemento.
- Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página se ven afectados y modifican su posición.



# Posicionamiento absoluto

- 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:
  - Top: desplazamiento desde el borde superior del elemento contenedor que se utiliza como referencia.
  - Right: ídem pero desde el borde derecho al borde derecho.
  - Left: ídem pero desde el borde izquierdo al borde izquierdo.
  - Bottom: ídem pero desde el borde inferior al borde inferior.

# Diferencias entre posicionamiento absoluto y relativo



# Posicionamiento fijo

- Es un caso particular del posicionamiento absoluto, ya que sólo se diferencian en el comportamiento de las cajas posicionadas.
- 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.

# Posicionamiento flotante

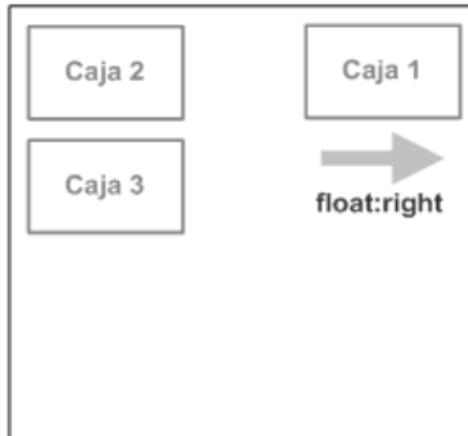
- Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una caja flotante, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.

Elemento contenedor



Posicionamiento normal

Elemento contenedor



Posicionamiento float de la caja 1

# Posicionamiento flotante (y II)

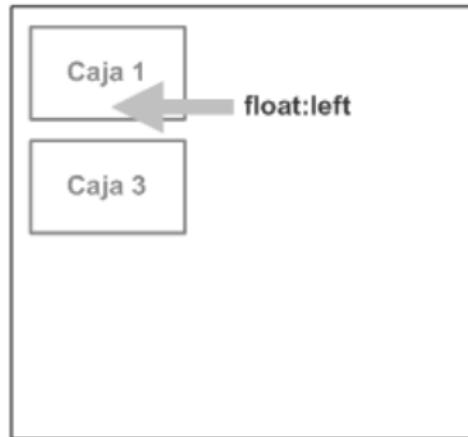
- Cuando se posiciona una caja de forma flotante:
  - La caja deja de pertenecer al flujo normal de la página, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante.
  - La caja flotante se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la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

Elemento contenedor



Posicionamiento normal

Elemento contenedor

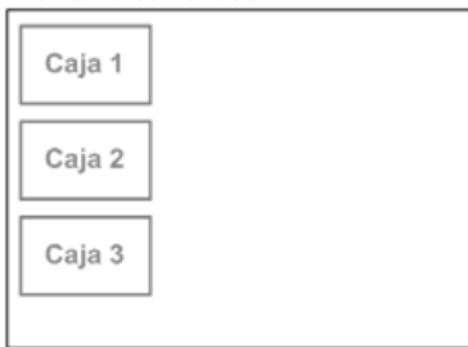


Posicionamiento float de la caja 1

# Posicionamiento flotante (y III)

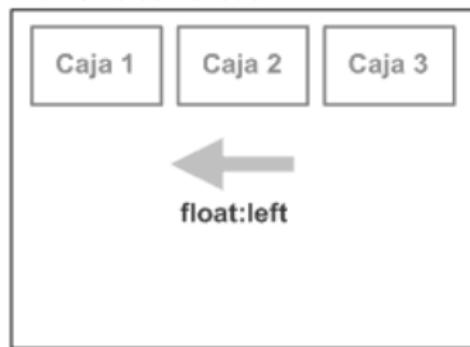
- Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible.
- Si no existiera sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea

Elemento contenedor



Posicionamiento normal

Elemento contenedor



Posicionamiento float de las 3 cajas

# Propiedad float

Propiedad	<b>float</b>
Valores	left — right — none — inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el tipo de posicionamiento flotante del elemento

Cuadro: Definición de la propiedad float de CSS

# Posicionamiento flotante (y IV)

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

Elemento contenedor



Posicionamiento normal

Elemento contenedor



Posicionamiento float de la imagen

# Propiedad clear

- 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”.

# Propiedad clear

Propiedad	<b>clear</b>
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

Cuadro: Definición de la propiedad clear de CSS

# Propiedad display

Propiedad	<b>display</b>
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

Cuadro: Definición de la propiedad display de CSS

# Propiedad visibility

Propiedad	<b>visibility</b>
Valores	visible — hidden — collapse — inherit
Se aplica a	Todos los elementos
Valor inicial	visible
Descripción	Permite hacer visibles e invisibles a los elementos

**Cuadro:** Definición de la propiedad visibility de CSS

# Propiedad overflow

- 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.
- 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.

# Propiedad overflow (y II)



# Propiedad overflow (y III)

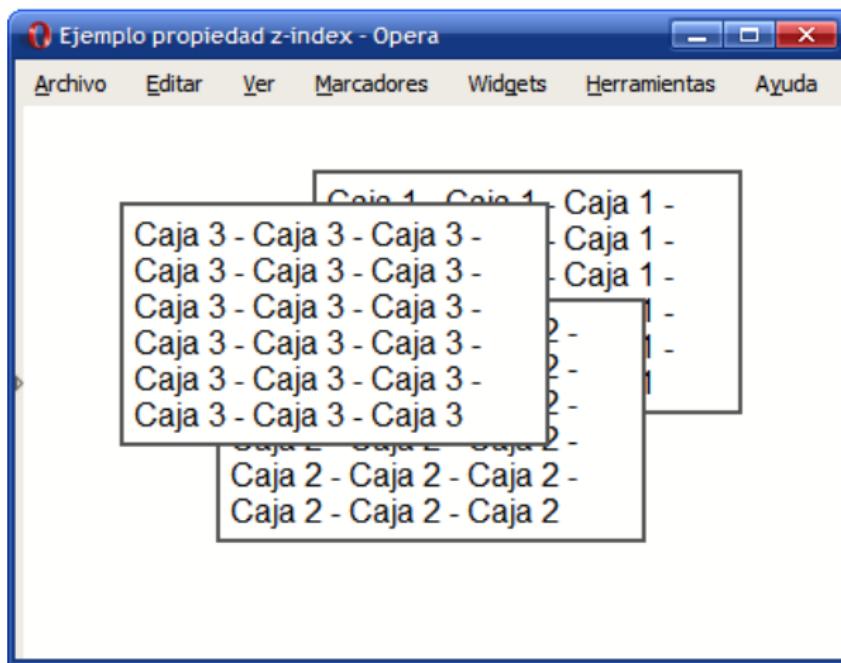
Propiedad	<b>overflow</b>
Valores	visible — hidden — scroll — auto — inherit
Se aplica a	Elementos de bloque y celdas de tablas
Valor inicial	visible
Descripción	Permite controlar los contenidos sobrantes de un elemento

**Cuadro:** Definición de la propiedad overflow de CSS

# Propiedad z-index

- CSS permite controlar la posición tridimensional de las cajas posicionadas
- Es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.
- Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja.

## Propiedad z-index (II)



# Propiedad z-index (y III)

Propiedad	<b>z-index</b>
Valores	auto — < <i>numero</i> > — inherit
Se aplica a	Elementos que han sido posicionados explícitamente
Valor inicial	auto
Descripción	Establece el nivel tridimensional en el que se muestra el elemento

**Cuadro:** Definición de la propiedad z-index de CSS

# Selectores avanzados

## ① Selector de hijos

```
p > span { color: blue; }
```

## ② Selector adyacente

```
p + p { text-indent: 1.5em; }
```

## ③ Selector de atributos

```
a[class="externo"] { color: blue; }  
a[class~= "externo"] { color: blue; }  
*[lang=en] { ... }  
*[lang|= "es"] { color : red }
```

# Colisión de estilos

El método seguido por CSS para resolver las colisiones de estilos se muestra a continuación:

- ① Determinar todas las declaraciones que se aplican al elemento para el medio CSS seleccionado.
- ② Ordenar las declaraciones según su origen (CSS de navegador, de usuario o de diseñador) y su prioridad (palabra clave !important).
- ③ Ordenar las declaraciones según lo específico que sea el selector. Cuanto más genérico es un selector, menos importancia tienen sus declaraciones.
- ④ Si después de aplicar las normas anteriores existen dos o más reglas con la misma prioridad, se aplica la que se indicó en último lugar.

# Medios CSS

- Permiten definir diferentes estilos para diferentes medios o dispositivos: pantallas, impresoras, móviles, proyectores, etc.
- Define algunas propiedades específicamente para determinados medios: la paginación y los saltos de página para los medios impresos o el volumen y tipo de voz para los medios de audio.
- Ejemplos:
  - screen: Pantallas de ordenador
  - print: Impresoras y navegadores en el modo “Vista Previa para Imprimir”
  - handheld: Dispositivos de mano: móviles, PDA, etc.

# Formas de indicar el medio

## ① Reglas de tipo @media

```
@media print {  
    body { font-size: 10pt }  
}  
  
@media screen {  
    body { font-size: 13px }  
}
```

## ② Reglas de tipo @import

```
@import url("estilos_basicos.css") screen;  
@import url("estilos_impresora.css") print;
```

## ③ Medios definidos con la etiqueta

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css"  
<link rel="stylesheet" type="text/css" media="print, handheld" href="e
```

## ④ Medios definidos mezclando varios métodos

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css"  
@import url("estilos_seccion.css") screen;  
@media print {
```

# Comentarios

- El comienzo de un comentario se indica mediante los caracteres /\* y el final del comentario se indica mediante \*/

```
/* Este es un comentario en CSS */
```
- Pueden ocupar tantas líneas como sea necesario, pero no se puede incluir un comentario dentro de otro comentario  

```
/* Este es un
comentario CSS de varias
lineas */
```

# Tipografía

- CSS define numerosas propiedades para modificar la apariencia del texto
- color se utiliza para establecer el color de la letra
- Como el valor de la propiedad color se hereda, normalmente se establece la propiedad color en el elemento body para establecer el color de letra de todos los elementos de la página
- font-family se utiliza para indicar el tipo de letra con el que se muestra el texto
- Suele definirse como una lista de tipos de letra alternativos separados por comas. El último valor de la lista es el nombre de la familia tipográfica genérica que más se parece al tipo de letra que se quiere utilizar.

# Propiedad color

Propiedad	<b>color</b>
Valores	$<color>$ — inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el color de letra utilizado para el texto

Cuadro: Definición de la propiedad color de CSS

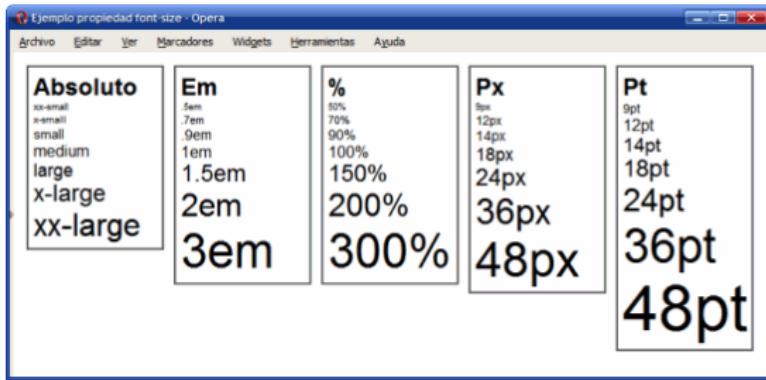
# Propiedad font-family

Propiedad	<b>font-family</b>
Valores	(( < nombre_familia > — < familia_generica > ) (,nombre_familia > — < familia_generica)* ) — inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el tipo de letra utilizado para el texto

**Cuadro:** Definición de la propiedad font-family de CSS

# Propiedad font-size (I)

- Además de medida relativas, absolutas y de porcentajes, CSS permite utilizar una serie de palabras clave para indicar el tamaño de letra del texto:
  - tamaño\_absoluto: indica el tamaño de letra de forma absoluta mediante alguna de las siguientes palabras clave: xx-small, x-small, small, medium, large, x-large, xx-large.
  - tamaño\_relativo: indica de forma relativa el tamaño de letra del texto mediante dos palabras clave (larger, smaller) que toman como referencia el tamaño de letra del elemento padre.



# Propiedad font-size (y II)

Propiedad	<b>font-size</b>
Valores	$< \text{tamano\_absoluto} >$ — $< \text{tamano\_relativo} >$ — $< \text{medida} >$ — $< \text{porcentaje} >$ — inherit
Se aplica a	Todos los elementos
Valor inicial	medium
Descripción	Establece el tamaño de letra utilizado para el texto

**Cuadro:** Definición de la propiedad font-size de CSS

# Propiedad font-weight

Propiedad	<b>font-weight</b>
Valores	normal — bold — bolder — lighter — 100 — 200 — 300 — 400 — 500 — 600 — 700 — 800 — 900 — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece la anchura de la letra utilizada para el texto

**Cuadro:** Definición de la propiedad font-weight de CSS

# Propiedad font-style

Propiedad	<b>font-style</b>
Valores	normal — italic — oblique — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo de la letra utilizada para el texto

**Cuadro:** Definición de la propiedad font-style de CSS

# Propiedad font-variant

Propiedad	<b>font-variant</b>
Valores	normal — small-caps — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo alternativo de la letra utilizada para el texto

**Cuadro:** Definición de la propiedad font-variant de CSS

# Propiedad *short-hand font*

Propiedad	<b>font</b>
Valores	( ( < font – style > — — < font – variant > — — < font – weight > )? < font – size > ( / < line – height > )? < font – family > ) — caption — icon — menu — message-box — small-caption — status-bar — inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Permite indicar de forma directa todas las propiedades de la tipografía de un texto

Cuadro: Definición de la propiedad font de CSS

# Propiedad *short-hand font* (y II)

- El orden en el que se deben indicar las propiedades del texto es el siguiente:
  - En primer lugar y de forma opcional se indican el font-style, font-variant y font-weight en cualquier orden.
  - A continuación, se indica obligatoriamente el valor de font-size seguido opcionalmente por el valor de line-height.
  - Por último, se indica obligatoriamente el tipo de letra a utilizar.

```
font: bold 1em "Trebuchet MS", Arial, Sans-Serif;
```

```
font: normal 0.9em "Lucida Grande", Verdana, Arial, Helvetica, sans-serif;
```

```
font: normal 1.2em/1em helvetica, arial, sans-serif;
```

# Texto

- Además de las propiedades relativas a la tipografía del texto, CSS define numerosas propiedades que determinan la apariencia del texto en su conjunto.
- Estas propiedades adicionales permiten controlar
  - la alineación del texto,
  - el interlineado,
  - la separación entre palabras,
  - etc.

# Propiedad text-align

Propiedad	<b>text-align</b>
Valores	left — right — center — justify — inherit
Se aplica a	Elementos de bloque y celdas de tabla
Valor inicial	left
Descripción	Establece la alineación del contenido del elemento

**Cuadro:** Definición de la propiedad text-align de CSS

# Propiedad line-height

Propiedad	<b>line-height</b>
Valores	normal — $< \text{numero} >$ — $< \text{medida} >$ — $< \text{porcentaje} >$ — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer la altura de línea de los elementos

**Cuadro:** Definición de la propiedad line-height de CSS

# Propiedad text-decoration

Propiedad	<b>text-decoration</b>
Valores	none — ( underline — overline — line-through — blink ) — inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece la decoración del texto (subrayado, tachado, parpadeante, etc.)

Cuadro: Definición de la propiedad text-decoration de CSS

# Propiedad text-transform

Propiedad	<b>text-transform</b>
Valores	capitalize — uppercase — lowercase — none — inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Transforma el texto original (lo transforma a mayúsculas, a minúsculas, etc.)

Cuadro: Definición de la propiedad text-transform de CSS

# Propiedad vertical-align

Propiedad	<b>vertical-align</b>
Valores	baseline — sub — super — top — text-top — middle — bottom — text-bottom — < <i>porcentaje</i> > — < <i>medida</i> > — inherit
Se aplica a	Elementos en línea y celdas de tabla
Valor inicial	baseline
Descripción	Determina la alineación vertical de los contenidos de un elemento

**Cuadro:** Definición de la propiedad vertical-align de CSS

# Propiedad text-indent

Propiedad	<b>text-indent</b>
Valores	<i>&lt; medida &gt;</i> — <i>&lt; porcentaje &gt;</i> — inherit
Se aplica a	Los elementos de bloque y las celdas de tabla
Valor inicial	0
Descripción	Tabula desde la izquierda la primera línea del texto original

**Cuadro:** Definición de la propiedad text-indent de CSS

# Propiedad letter-spacing

Propiedad	<b>letter-spacing</b>
Valores	normal — <i>medida</i> — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las letras que forman las palabras del texto

**Cuadro:** Definición de la propiedad letter-spacing de CSS

# Propiedad word-spacing

Propiedad	<b>word-spacing</b>
Valores	normal — <i>medida</i> — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las palabras que forman el texto

**Cuadro:** Definición de la propiedad word-spacing de CSS

# Propiedad white-space

Propiedad	<b>white-space</b>
Valores	normal — pre — nowrap — pre-wrap — pre-line — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el tratamiento de los espacios en blanco del texto

**Cuadro:** Definición de la propiedad white-space de CSS

# Propiedad white-space (y II)

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

- normal: comportamiento por defecto de HTML.
- pre: se respetan los espacios en blanco y las nuevas líneas (exactamente igual que la etiqueta `<pre>`). Si la línea es muy larga, se sale del espacio asignado para ese contenido.
- nowrap: elimina los espacios en blanco y las nuevas líneas. Si la línea es muy larga, se sale del espacio asignado para ese contenido.
- pre-wrap: se respetan los espacios en blanco y las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.
- pre-line: elimina los espacios en blanco y respeta las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.

# Propiedad list-style-type

Propiedad	<b>list-style-type</b>
Valores	disc — circle — square — decimal — decimal-leading-zero — lower-roman — upper-roman — lower-greek — lower-latin — upper-latin — armenian — georgian — lower-alpha — upper-alpha — none — inherit
Se aplica a	Elementos de una lista
Valor inicial	disc
Descripción	Permite establecer el tipo de viñeta mostrada para una lista

**Cuadro:** Definición de la propiedad list-style-type de CSS

# Propiedad list-style-position

Propiedad	<b>list-style-position</b>
Valores	inside — outside — inherit
Se aplica a	Elementos de una lista
Valor inicial	outside
Descripción	Permite establecer la posición de la viñeta de cada elemento de una lista

**Cuadro:** Definición de la propiedad list-style-type de CSS

# Propiedad list-style-image

Propiedad	<b>list-style-image</b>
Valores	$<url>$ — none — inherit
Se aplica a	Elementos de una lista
Valor inicial	none
Descripción	Permite reemplazar las viñetas automáticas por una imagen personalizada

**Cuadro:** Definición de la propiedad list-style-image de CSS

# Propiedad *shorthand* list-style

Propiedad	<b>list-style</b>
Valores	( <i>&lt;list-style-type&gt;</i> — <i>&lt;list-style-position&gt;</i> — <i>&lt;list-style-image&gt;</i> ) — inherit
Se aplica a	Elementos de una lista
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultánea todas las opciones de una lista

**Cuadro:** Definición de la propiedad list-style de CSS

# Imágenes según el estilo del enlace

- En ocasiones, puede resultar útil incluir un pequeño ícono al lado de un enlace para indicar el tipo de contenido que enlaza.
- Este tipo de imágenes son puramente decorativas en vez de imágenes de contenido, por lo que se deberían añadir con CSS y no con elementos de tipo `<img>`. Utilizando la propiedad `background` (y `background-image`) se puede personalizar el aspecto de los enlaces para que incluyan un pequeño ícono a su lado.
- La técnica consiste en mostrar una imagen de fondo sin repetición en el enlace y añadir el padding necesario al texto del enlace para que no se solape con la imagen de fondo.

# Imágenes según el estilo del enlace (II)

```
a { margin: 1em 0; float: left; clear: left; }

.rss {
    color: #E37529;
    padding: 0 0 0 18px;
    background: #FFF url(imagenes/rss.gif) no-repeat left center;
}

.pdf {
    padding: 0 0 0 22px;
    background: #FFF url(imagenes/pdf.png) no-repeat left center;
}

<a href="#">Enlace con el estilo por defecto</a>
<a class="rss" href="#">Enlace a un archivo RSS</a>
<a class="pdf" href="#">Enlace a un documento PDF</a>
```

# Imágenes según el estilo del enlace (y III)



# Mostrar los enlaces como si fueran botones

```
a { margin: 1em 0; float: left; clear: left; }
.a.boton {
    text-decoration: none;
    background: #EEE;
    color: #222;
    border: 1px outset #CCC;
    padding: .1em .5em;
}
.a.boton:hover {
    background: #CCB;
}
.a.boton:active {
    border: 1px inset #000;
}
<a class="boton" href="#">Guardar</a>
<a class="boton" href="#">Enviar</a>
```

# Crear un menú vertical

- ① Definir anchura del menú
- ② Eliminar las viñetas automáticas y todos los márgenes y espaciados aplicados por defecto
- ③ Añadir un borde al menú de navegación y establecer el color de fondo y los bordes de cada elemento del menú
- ④ Aplicar estilos a los enlaces: mostrarlos como un elemento de bloque para que ocupen todo el espacio de cada `<li>` del menú, añadir un espacio de relleno y modificar los colores y la decoración por defecto



# Menú horizontal básico

- ① Aplicar los estilos CSS básicos para establecer el estilo del menú (similares a los del menú vertical)
- ② Establecer la anchura de los elementos del menú. Si el menú es de anchura variable y contiene cinco elementos, se asigna una anchura del 20 % a cada elemento
- ③ Establecer los bordes de los elementos que forman el menú
- ④ Se elimina el borde derecho del último elemento de la lista, para evitar el borde duplicado

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5
------------	------------	------------	------------	------------

# Hojas de estilo CSS3

# ¿Qué es CSS3?

- CSS3 ofrece una gran variedad de maneras nuevas para el diseño de hojas de estilo
- Al ser tan amplio el desarrollo de CSS3, se ha dividido en módulos:
  - El modelo de caja
  - Listas
  - Presentación de hipervínculos
  - Voz
  - Fondos y bordes
  - Efectos de texto
  - ...

# CSS3 todavía en desarrollo

- Hay módulos, la mayoría, cuya especificación no está terminada
- Hay módulos terminados, pero los navegadores no los implementan
- Es muy importante conocer el estado de la implementación en navegadores
- Hay navegadores que tienen reglas específicas (temporales)
  - Comprobar si está implementado en <http://caniuse.com/>

# ¿Qué hay de nuevo en CSS3?

- Las buenas noticias son que CSS3
  - es **compatible hacia atrás**
  - Mantiene la **misma filosofía**
- Básicamente, CSS3 añade nuevos selectores y propiedades
- Algunas son funcionalidades nuevas (animaciones o gradientes), y otras permiten diseños más sencillos (p.ej. el uso de columnas)

# Especificidades de navegadores

P.ej. border-radius:

```
.box {  
    -moz-border-radius: 10px;  
    -webkit-border-radius: 10px;  
    border-radius: 10px;  
}
```

(¡imaginaos lo que era antes conseguir bordes redondeados!)

# Módulo Selectores

- first-child: primer elemento de la etiqueta padre
- last-child: último elemento de la etiqueta padre
- nth-child: selecciona múltiples elementos según su posición en el árbol

```
p:first-child {  
    background-color: yellow;  
}  
  
box:last-child {  
    padding: 12px;  
}
```

# Módulo Colores

- opacity: indica la opacidad de un elemento
- rgba: indica la opacidad de un color con el “alpha”

```
/* red with opacity */
.box1 {background-color: rgba(255,0,0,0.3);}
/* green with opacity */
.box2 {background-color: rgba(0,255,0,0.3);}
/* blue with opacity */
.box3 {background-color: rgba(0,0,255,0.3);}
```

# Módulo Fuentes

- @font-face: permite utilizar fuentes no instaladas en el cliente

```
@font-face {  
    font-family: "Essays 1743";  
    font-weight: bold;  
    src: url("f/Essays1743-bold.woff") format("woff");  
}
```

# Módulo Fondos

- Se puede añadir más de una imagen como fondo
- border-radius: se pueden añadir bordes redondeados
- box-shadow: incluye sombras en cajas

# Bootstrap

# ¿Qué es Bootstrap?

- Bootstrap es un framework libre para desarrollo web
- Realizado por ingenieros de Twitter
- Incluye plantillas HTML y CSS con tipografías, formas, botones, cuadros, barras de navegación, carruseles de imágenes y muchas otras
- También existe la posibilidad de utilizar plugins de JavaScript
- Aunque su preferencia es *mobile first*, permite crear diseños que se ven bien en múltiples dispositivos (*responsive design*)

# Ventajas de Bootstrap (para ingenieros)

- Es sencillo y rápido
- Se adapta a distintos dispositivos (*responsive design*)
- Proporciona un diseño consistente
- Es compatible con los navegadores modernos
- Es software libre

# Ficheros de Bootstrap

```
bootstrap/
|--- css/
|   |--- bootstrap.css
|   |--- bootstrap.css.map
|   |--- bootstrap.min.css
|   |--- bootstrap-theme.css
|   |--- bootstrap-theme.css.map
|   |--- bootstrap-theme.min.css
|--- js/
|   |--- bootstrap.js
|   |--- bootstrap.min.js
|--- fonts/
    |--- glyphicons-halflings-regular.eot
    |--- glyphicons-halflings-regular.svg
    |--- glyphicons-halflings-regular.ttf
    |--- glyphicons-halflings-regular.woff
    |--- glyphicons-halflings-regular.woff2
```

# La plantilla de Bootstrap

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Basic Bootstrap Template</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
</head>
<body>
  <h1>Hello, world!</h1>
  <script src="http://code.jquery.com/jquery.min.js"></script>
  <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

# Bootstrap en CDN

- Con un CDN (Content Delivery Network) no hace falta tener Bootstrap en nuestros archivos. Además, si un usuario ya ha descargado esas URLs, probablemente las tenga ya en la caché del navegador (con el consiguiente ahorro de tiempo).

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
      href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css"

<!-- jQuery library -->
<script
      src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js">
</script>

<!-- Latest compiled JavaScript -->
<script
      src="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js">
</script>
```

# Mobile first

- En los navegadores, el *viewport* es la parte visible de un documento
- Si el documento es mayor que el área de visualización, el usuario puede cambiar la vista desplazándose
- Con una propiedad de etiqueta meta, podemos indicar la escala inicial del *viewport*

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- Se puede inhabilitar el zoom en dispositivos móviles con `user-scalable=no`
- Los usuarios sólo podrán hacer *scroll* y tendrá una apariencia nativa.
- Usar con precaución. No vale para todas las aplicaciones

```
<meta name="viewport" content="width=device-width, initial-scale=1,  
maximum-scale=1, user-scalable=no">
```

# Contenedores

- Bootstrap requiere tener un elemento contenedor que cubra contenidos y el sistema de rejilla.
- Para un contenedor responsive de tamaño fijo, usa .container

```
<div class="container">  
  ...  
</div>
```

- Si se desea un contenedor con el ancho total (del *viewport*), se ha de usar .container-fluid

```
<div class="container-fluid">  
  ...  
</div>
```

# El sistema de rejilla (I)

El diseño de páginas basado en rejilla se realiza mediante filas y columnas donde se colocan los contenidos. Así funciona la rejilla de Bootstrap:

- Filas, dentro un contenedor, agrupan horizontalmente varias columnas, que son las que tienen contenido
- La pantalla se divide en 12 columnas
- Las columnas definen su anchura especificando cuántas columnas de la fila ocupan
- Hay clases CSS (como por ejemplo .row y .col-xs-4) para crear rejillas rápidamente
- Hay padding entre columnas. En la primera y última columnas, las filas (elementos .row) aplican márgenes negativos

# El sistema de rejilla (y II)

## Three equal columns

Get three equal-width columns **starting at desktops and scaling to large desktops**. On mobile devices, tablets and below, the columns will automatically stack.

.col-md-4	.col-md-4	.col-md-4
-----------	-----------	-----------

## Three unequal columns

Get three columns **starting at desktops and scaling to large desktops** of various widths. Remember, grid columns should add up to twelve for a single horizontal block. More than that, and columns start stacking no matter the viewport.

.col-md-3	.col-md-6	.col-md-3
-----------	-----------	-----------

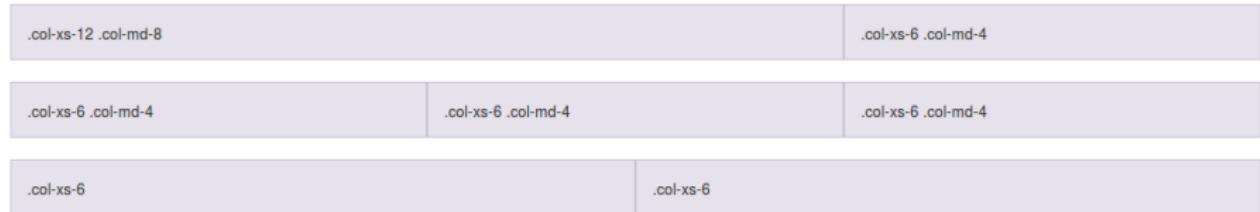
## Two columns

Get two columns **starting at desktops and scaling to large desktops**.

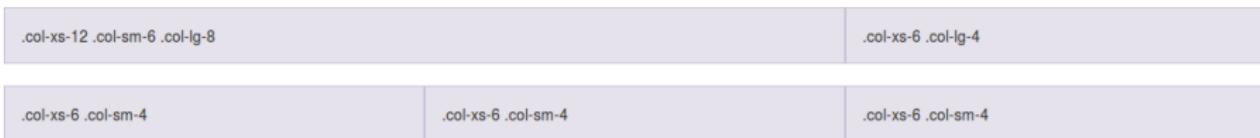
.col-md-8	.col-md-4
-----------	-----------

# Responsive design

Mobile and desktop:



Mobile, tablet, and desktop:



# Media queries

Bootstrap utiliza *media queries* para establecer puntos de ruptura en los que la rejilla se transforma para adaptarse a cada dispositivo.

```
/* Extra small devices (phones, less than 768px) */  
/* No media query since this is the default in Bootstrap */  
  
/* Small devices (tablets, 768px and up) */  
@media (min-width: @screen-sm-min) { ... }  
/* Medium devices (desktops, 992px and up) */  
@media (min-width: @screen-md-min) { ... }  
/* Large devices (large desktops, 1200px and up) */  
@media (min-width: @screen-lg-min) { ... }
```

Se pueden utilizar también *media queries* que definen la propiedad max-width y restringen los dispositivos a los que se aplican los estilos:

```
@media (max-width: @screen-xs-max) { ... }  
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }  
@media (min-width: @screen-md-min) and (max-width: @screen-md-max) { ... }  
@media (min-width: @screen-lg-min) { ... }
```

# Otras clases

Bootstrap viene con una serie de estilos (generalmente en formato de clase CSS) por defecto, entre otros:

- `table`
- `form`
- `buttons`
- `img-responsive`
- `helper classes`
- y otras utilidades responsivas

# JavaScript

Con Bootstrap vienen una serie de *plug-ins* de JavaScript, como:

- transitions.js: efectos de transición
- modal.js: diálogos simples y flexibles
- dropdown.js: menús desplegables
- scrollspy.js: según vamos bajando en la página, actualiza la *navbar*
- tab.js: pestañas
- tooltip.js y popover.js: cajas con consejos o más información
- alert.js: alertas
- button.js: botones
- collapse.js: permite mostrar/ocultar elementos
- carousel.js: el (ya famoso carrusel)
- affix.js: Menú de navegación lateral

# Bootlint

- Herramienta que detecta algunos errores comunes el HTML de diseños Bootstrap
- Comprueba que las instancias de componentes Bootstrap han sido correctamente estructurados
- Analiza también la inclusión de ciertas etiquetas <*meta*>, la declaración DOCTYPE HTML5, etc.
- Página web: <https://github.com/twbs/bootlint>

# Otros sitios de interés

- Listado de recursos sobre Bootstrap  
<http://bootsnipp.com/resources>
- Bootsnipp: Cientos de componentes adicionales  
<http://www.bootsnipp.com>
- Startbootstrap: Plantillas y temas Bootstrap (gratis)  
<http://startbootstrap.com/>
- WrapBootstrap: Plantillas y temas Boostrap (de pago)  
<https://wrapbootstrap.com/>
- BootTheme: Generador (no libre) de diseños Bootstrap  
<http://www.boottheme.com/>

# HTML5

©Mark Pilgrim  
©Gregorio Robles - Universidad Rey Juan Carlos

Algunos derechos reservados. Este artículo se distribuye bajo la licencia "Reconocimiento 3.0 España" de Creative Commons, disponible en <http://creativecommons.org/licenses/by/3.0/es/deed.es>

Este documento se basa en el libro "Dive into HTML5"  
disponible en <http://diveintohtml5.info/>

# 5 cosas sobre HTML5

# 1. No es una única cosa grande

- HTML5 es una colección de funcionalidades
- No hace falta buscar soporte completo, pero sí de las funcionalidades concretas
- HTML5 especifica también cómo interactuar con JavaScript y DOM

## 2. No hace falta tirar nada

- Se basa en HTML4
- Si una web funcionaba con HTML4, funcionará con HTML5
- HTML5 extiende HTML4
- Los navegadores antiguos tratarán muchos elementos HTML5 como si no existieran

### 3. Es fácil empezar

- Cambiar a HTML5 es sencillo
- En realidad, sólo hay que cambiar el DOCTYPE:  
`<!DOCTYPE html>`
- Los elementos obsoletos de HTML4 todavía se verán en HTML5

## 4. Ya funciona (en líneas generales)

- Los navegadores están haciendo un gran esfuerzo por incluir HTML5
- Mientras tanto, tendremos que estar atentos a la compatibilidad de los mismos
- Todavía no está totalmente estandarizado
- Hasta 2020 no se espera soporte completo

## 5. Está aquí para quedarse

- Reemplaza otras tecnologías: Flash
- Sigue la tendencia de que todo va a la nube
- Sigue la tendencia de la ubicuidad

# Algunos elementos HTML5

# DOCTYPE

- Hasta ahora había diferentes modos:
  - Quirks Mode
  - Standards Mode
  - Almost Standards Mode
- En HTML5, sólo hace falta:

```
<!DOCTYPE html>
```

# El Elemento raíz

- Antes:

```
<html xmlns="http://www.w3.org/1999/xhtml"  
      lang="en"  
      xml:lang="en">
```

- Ahora:

```
<html lang="en">
```

# Codificación de caracteres

- Si podemos, envíamos cabecera:

Content-Type: text/html; charset="utf-8"

- Si no se puede, en HTML5s:

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

# Relaciones

- Hoja de estilo: rel="stylesheet"
- Feed: rel="alternate"
- Emoticono: rel="shortcut icon"
- Serie de páginas: rel="start", rel="prev", rel="next"
- ...

# Elementos semánticos

- < *section* >
- < *nav* >
- < *article* >
- < *aside* >
- < *hgroup* >
- < *header* >
- < *footer* >
- < *time* >
- < *mark* >

# ¿Cómo muestran los navegadores elementos desconocidos?

- Las etiquetas desconocidas se muestran como si fueran "inline"
- Regla CSS para cambiar el comportamiento:

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

Las versiones antiguas de Internet Explorer son especialmente "particulares".

# El canvas

# La etiqueta canvas

- Elemento canvas
- Se ha de especificar un tamaño (width y height)
- Requiere un id para poder manipularla
- El canvas siempre empieza vacío

```
<canvas id="a" width="300" height="225"></canvas>
```

# Pintemos algo

```
function draw_b() {  
    var b_canvas = document.getElementById("b");  
    var b_context = b_canvas.getContext("2d");  
    b_context.fillRect(50, 25, 150, 100);  
}
```

# Algunas cuestiones a la hora de pintar

- La propiedad *fillStyle* puede ser cualquier color CSS (también un patrón o un gradiente). Por defecto, es negro.
- `fillRect(x, y, width, height)` pinta un rectángulo relleno (con el color de *fillStyle*).
- La propiedad *strokeStyle* es como *fillStyle* pero para líneas.
- `strokeRect(x, y, width, height)` pinta un rectángulo con el estilo de *strokeStyle*.
- `clearRect(x, y, width, height)` borra los píxeles del rectángulo especificado.

# Sobre las coordenadas

- La coordenada  $(0,0)$  está en la esquina superior izquierda del canvas
- El eje X crece hacia la derecha
- El eje Y crece hacia abajo (!!)

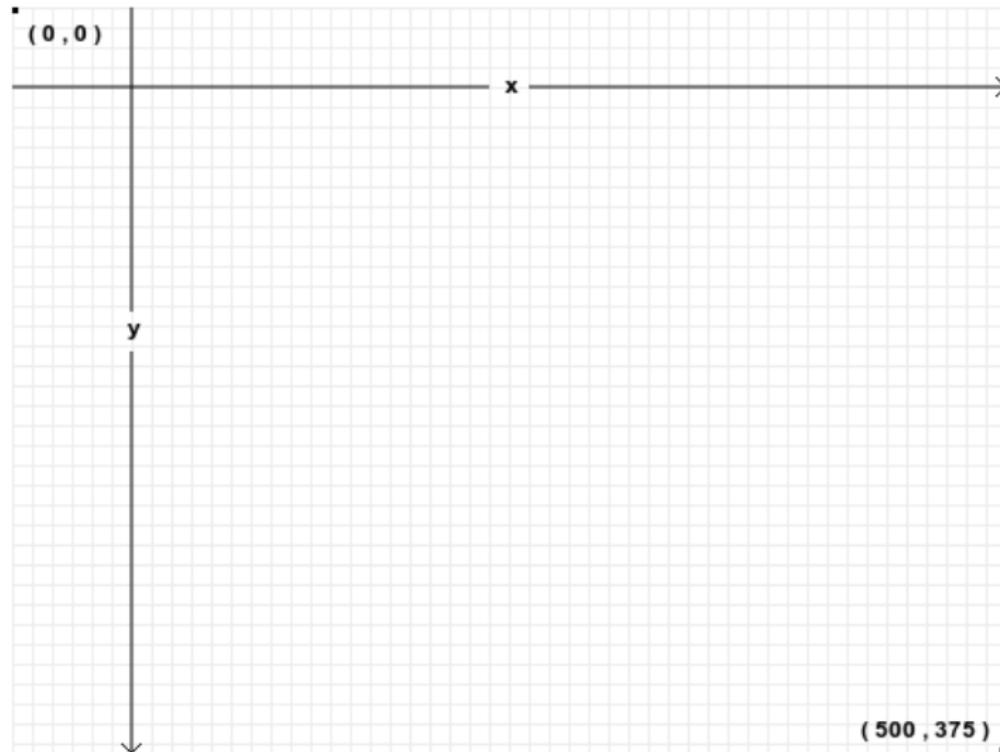
# Caminos

- beginPath(): empieza un nuevo camino
- moveTo(x, y): sitúa el lápiz en la coordenada especificada.
- lineTo(x, y): pinta una línea hasta la coordenada especificada.
- stroke(): dibuja (con el estilo de *strokeStyle*).
- arc(x, y, radio, angulo\_comienzo, angulo\_final, direccion);
  - angulo\_comienzo y angulo\_final pueden ser p.ej. `Math.PI * 2`
  - direccion: false si en el sentido del reloj

# Texto

- font: como *font* de CSS.
- textAlign: como el *text-align* de CSS.
- textBaseline: top, hanging, middle, alphabetic, ideographic, o bottom.
- fillText("texto", x, y): introduce "texto" en la posición (x, y).

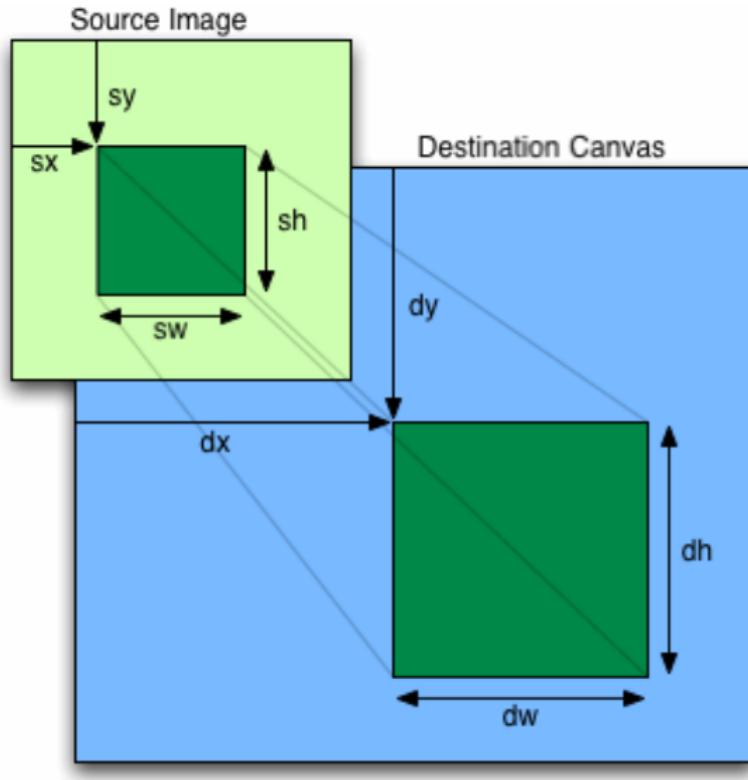
# Ejercicio: Pintemos un diagrama de coordenadas



# Imágenes (I)

- `drawImage(image, dx, dy)`: toma la imagen "image" y la pinta en el canvas. Las coordenadas dx y dy dan la esquina superior izquierda de la imagen.
- `drawImage(image, dx, dy, dw, dh)`: toma la imagen "image" y la pinta en el canvas. Las coordenadas dx y dy dan la esquina superior izquierda de la imagen escalándola a una anchura dw y a una altura dh.
- `drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)`: toma la imagen "image" y la introduce en el rectángulo (sx, sy, sw, sh), escalándola a las dimensiones (dw, dh), y la pinta en las coordenadas (dx, dy).  
Véase la siguiente transparencia.

# Imágenes (y II)



# Local Storage

# Buscando un espacio para guardar en local

¿Y por qué no las cookies?

- ① Las cookies se incluyen en cada petición HTTP, ralentizando una aplicación al enviarse una y otra vez
- ② Las cookies se incluyen en cada petición HTTP, enviando datos de manera no cifrada (a menos de que toda la aplicación se sirva sobre SSL)
- ③ Las cookies están limitadas a 4KB de datos - suficiente para ralentizar la conexión (véase punto 1), pero no lo suficiente para ser muy útiles

# HTML5 storage (Web storage)

- Es una manera de guardar parejas llave/valor de manera local, en el navegador cliente
- Estos datos persisten aún cuando cerramos el navegador
- Estos datos no se transmiten al servidor, a menos de que se pidan de manera explícita
- Es un mecanismo implementado de manera nativa en navegadores web, evitando tener que utilizar plug-ins y otras extensiones/trucos.

# Accediendo al almacenamiento local

La interfaz:

```
interface Storage {  
    getter any getItem(in DOMString key);  
    setter creator void.setItem(in DOMString key, in any data);  
};
```

Ejemplo de uso:

```
var foo = localStorage.getItem("bar");  
// ...  
localStorage.setItem("bar", foo);
```

También valdría:

```
var foo = localStorage["bar"];  
// ...  
localStorage["bar"] = foo;
```

# Borrando e iterando el almacenamiento local

Interfaz de borrado:

```
interface Storage {  
    deleter void removeItem(in DOMString key);  
    void clear();  
};
```

Interfaz de iteración:

```
interface Storage {  
    readonly attribute unsigned long length;  
    getter DOMString key(in unsigned long index);  
};
```

# Navegación Off-line

# Funcionamiento básico

- La página web de una aplicación off-line indicará una lista (llamada **manifest**) donde se indicarán los archivos a descargarse
- Un navegador que implemente aplicaciones offline HTML5 leerá la lista del manifiesto y descargará los archivos.
- El navegador se encargará de tenerlos en caché y actualizarlos si hay cambios
- Cuando se intente acceder a la aplicación web sin conexión, entonces el navegador web automáticamente utilizará las copias locales.

(hay un flag en el DOM que indica si estamos on-line u off-line)

# El manifiesto

¿Cómo se indica el manifiesto en una página web?

```
<!DOCTYPE html>
<html manifest="/cache.manifest">
<body>
...
</body>
</html>
```

El content-type del manifiesto ha de ser: text/cache-manifest .manifest

# Un manifiesto sencillo

Si tenemos un único archivo HTML, no hace falta indicarlo en el manifiesto. Si son varios, entonces sí (además de indicar el manifiesto en cada una de las páginas).

CACHE MANIFEST

/clock.css  
/clock.js  
/clock-face.jpg

Todos estos elementos están en la sección *explícita*: se descargarán y guardarán en local.

# La sección Network

CACHE MANIFEST

NETWORK:

/tracking.cgi

CACHE:

/clock.css

/clock.js

/clock-face.jpg

Elementos bajo la sección *NETWORK* no se almacenan en local. La sección *explícita* pasa a indicarse como *CACHE*.

# La sección fallback

Permite definir recursos que sustituyen a elementos en línea que por la razón que sea no pueden ser almacenados en local (o no se almacenaron con éxito):

CACHE MANIFEST

FALLBACK:

/ /offline.html

NETWORK:

\*

# Detección de funcionalidad HTML5

# Técnicas de detección de características HTML5

- ① Comprobar si existe una propiedad en un objeto global (como window o navigator). P.ej. geolocalización.
- ② Crear un elemento y entonces comprobar si existe una propiedad específica en ese elemento. P.ej. soporte para canvas.
- ③ Crear un elemento, comprobar si existe un método específico en ese elemento y entonces llamar al método y comprobar el valor de retorno. P.ej. comprobar formatos de vídeo soportados
- ④ Crear un elemento, poner un valor a una propiedad, y comprobar si la propiedad ha retenido ese valor. P.ej. comprobar los tipos de <*input*> soportados
- ⑤ ... o utilizar la biblioteca JavaScript Modernizr

# Modernizr

```
if (Modernizr.ELEMENT) {  
    // let's go!  
} else {  
    // no native ELEMENT support available :(.  
    // Maybe try a fallback  
}
```

donde *ELEMENT* puede ser: canvas, canvastext, video, video.webm, video.ogg, video.h264, localstorage, webworkers, applicationcache, geolocation, input.autofocus, history...

# Geolocalización

# Un ejemplo de geolocalización

```
function get_location() {  
    if (Modernizr.geolocation) {  
        navigator.geolocation.getCurrentPosition(show_map);  
    } else {  
        // no native support; maybe try a fallback?  
    }  
}
```

Geolocalización es opt-in para el usuario. Este código ofrece un menú donde puede compartir su geolocalización o no.

# Sigamos con el ejemplo

```
function show_map(position) {  
    var latitude = position.coords.latitude;  
    var longitude = position.coords.longitude;  
    // let's show a map or do something interesting!  
}
```

La función *callback* se llamará con un único parámetro: un objeto con dos propiedades: *coords* y *timestamp*

# El objeto posición

Propiedad	Tipo	Nota
coords.latitude	double	decimal degrees
coords.longitude	double	decimal degrees
coords.altitude	double or null	meters above reference ellipsoid
coords.accuracy	double	meters
coords.altitudeAccuracy	double or null	meters
coords.heading	double or null	degrees clockwise from true north
coords.speed	double or null	meters/second
timestamp	DOMTimeStamp	like a Date() object

# Web Workers

# ¿Qué son?

- Scripts en JavaScript que se ejecutan en el *background*
- Por tanto, no bloquean la página (los navegadores suelen ser de un único hilo)
- Son parecidos a los *threads* (¡pero no comparten memoria!)
- Son relativamente pesados... sólo deberían utilizarse si van a realizar tareas pesadas que hagan que el coste de iniciar uno (en memoria y en procesamiento) valga la pena
- Los Web Workers no tienen acceso al DOM, se comunican con el programa principal mediante envío de mensajes.

# ¿Cómo se lanzan?

- ① Se instancia un objeto Worker, cuyo parámetro es una URL con el código JavaScript La URL tiene las mismas limitaciones de seguridad que JavaScript. Se puede empotrar en código JavaScript en la misma página HTML, pero entonces hay que crear un objeto URL.

```
var worker = new Worker("worker.js");
```

- ② Se pasa un mensaje al Worker. Normalmente este mensaje estará en formato JSON.

```
worker.postMessage("Hello World!");
```

# ¿Cómo se lanzan? (y II)

- ③ En caso de recibir un mensaje de respuesta, se captura el evento y se hace lo que queramos con los datos que recibidos (nota: generalmente recibiremos un JSON):

```
worker.onmessage = function(event) {  
    console.log("Hemos recibido " + event.data);  
    hacemosAlgo();  
}
```

- ④ Se da por cerrado el Worker:

```
worker.terminate();
```

# ¿Cómo se ejecutan?)

- El código del Worker estará en un fichero JavaScript separado (p.ej., `worker.js`)
- Utiliza la misma interfaz de comunicación, con la salvedad de que en vez de realizarse sobre el objeto tipo Worker lo hace sobre sí mismo, con `self`.
- Recuerda: no puede acceder al DOM
- Puede enviar varios mensajes, no tiene por qué ser un único

```
self.onmessage = function(event) {  
    console.log("He recibido " + event.data);  
    salida = hacemosAlgo();  
    self.postMessage("Te devuelvo " + salida)  
}
```

# WebSocket

# ¿Qué son los WebSockets?

- Protocolo que permite tener comunicación full-duplex sobre TCP (RFC 6455)
- Usa el puerto 80 (bueno para evitar firewalls)
- Es independiente de HTTP - el handshake es interpretado por los servidores como una petición *Upgrade*
- Permite que el servidor envíe contenido al navegador sin que éste la haya solicitado
- Antes de la estandarización por el W3C, la solución pasaba por tecnologías como Comet

# El *handshake* con WebSockets

## Petición del cliente:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

## Respuesta del servidor:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sM1YUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

# Utilizando WebSockets

- ① Instanciando un objeto WebSocket (nótese que la URL empieza por 'ws'; también existe conexión segura mediante el uso de 'wss'):

```
var connection = new WebSocket('ws://gsyc.es/echo');
```

- ② Enviamos datos al servidor (lo suyo sería enviar JSON, pero podemos enviar también otras cosas, como Blob o ArrayBuffer):

```
connection.onopen = function () {  
    connection.send('Ping');  
};
```

# Utilizando WebSockets (y II)

- ③ Podemos mirar si hay errores:

```
connection.onerror = function (error) {  
    console.log('Error en el WebSocket: ' + error);  
};
```

- ④ Recibimos del servidor (generalmente serán JSON, pero se pueden recibir otras cosas, como Blob o ArrayBuffer):

```
connection.onmessage = function (e) {  
    console.log('Server: ' + e.data);  
};
```

# Implementaciones en el servidor

- Node.js
  - Socket.IO
  - WebSocket-Node
  - ws
- Python
  - pywebsocket
  - Tornado
- Java
  - Jetty
- Ruby
  - EventMachine

# History API

# ¿Qué es la History API?

- El objeto `window` del DOM proporciona acceso al historial del browser a través del objeto `history`
- Tendremos métodos y propiedades que nos permitan avanzar y retroceder a través del historial (esto lo hemos podido hacer desde (casi) siempre)
- Podemos manipular el contenido del historial (¡nuevo en HTML5!)

# Moviéndonos en el historial

- Hacia atrás: `window.history.back();`
- Hacia adelante: `window.history.forward();`
- A un punto específico del historial: `window.history.go(-2);`
- Longitud de la pila: `window.history.length;`

# Manipulando el historial

- Añadir entradas en el historial: history.pushState();

```
var stateObj = { foo: "bar" }; // puede ser cualquier cosa  
history.pushState(stateObj, "titulo pagina", "introducida.html")
```

- Modificar entradas en el historial: history.replaceState();

```
var stateObj = { foo: "bar" };  
history.replaceState(stateObj, "titulo pagina", "nueva.html")
```

- Evento popstate: se lanza cada vez que la entrada al historial cambia. La propiedad state del evento popstate contiene una copia del historial de entradas del objeto estado.

# La API de los servicios de Google

# La(s) API(s) de los servicios de Google

- Google ofrece una API para acceder a muchos servicios
- Algunas APIs son de pago
- Otras tienen limitaciones de uso gratuito
- Hay peticiones que requieren autenticación y autorización

# El API Explorer (I)

Google ofrece un servicio para explorar sus APIs, el API Explorer:

- Permite conocer los métodos de las APIs
- Permite probarlos mediante un formulario web
- A veces requieren autorización/autenticación
- Incluyen un enlace a la documentación completa

APIs Explorer: <https://developers.google.com/apis-explorer/#p/>

# El API Explorer (II)

The screenshot shows the Google APIs Explorer interface. At the top, there is a search bar with placeholder text "Search for services, methods, and recent requests..." and a magnifying glass icon. Below the search bar, the title "APIs Explorer" is displayed, followed by a "Services" link and a gear icon for settings.

The main content area shows the "Services > URL Shortener API v1 > urlshortener.url.get" path. To the right of the path, there is an "Authorize requests using OAuth 2.0:" button set to "OFF".

The method parameters are listed below:

- shortUrl**: A text input field with the description "The short URL, including the protocol. (string)".
- projection**: A text input field with the description "Additional information to return. (string)".
- fields**: A text input field with the description "Selector specifying which fields to include in a partial response.  
[Use fields editor](#)".

At the bottom left, there is a note: "**bold red** = required". On the right side, there is a large blue "Execute" button.

# Ejemplo de uso del API Explorer

Utilicemos el API Explorer para el servicio de acortadores de URLs de Google.

- En el campo *shortUrl*, introduce *http://goo.gl/JCHYE*
- Pulsa el botón Execute
- Observa la petición: es una petición HTTP REST
- Observa la respuesta: es en formato JSON

# La Google API Console

Es una interfaz web para gestionar el uso de la API de Google en tu proyecto. Permite:

- Activar APIs
- Obtener la llave (*project key*) que se envía al utilizar la API, gestionar autenticación y autorización
- Obtener información de tráfico (y ver las cuotas de uso)
- Gestionar cobros
- Gestionar miembros del equipo

# Ejercicio: Google API Console

Crea un proyecto para utilizar la Google API Console.

- Selecciona los servicios de Google+ API y URL Shortener API
- Obtén tu *project key*
- Añade como *referer* `http://watson.gsync.es/`
- Incluye información de autorización vía OAuth2.0, también para `http://watson.gsync.es/`
- Échale un ojo a los menús de *Reports* y *Quotas*

Nota: Para realizar este ejercicio (y los siguientes), necesitarás tener una cuenta en Google.

# La API de Google+

# La API de Google+

- Google+ ofrece una API REST para acceder a los contenidos de la red social
- Hay varios lenguajes soportados, entre ellos Javascript
- Los navegadores modernos lo soportan:
  - Chrome 8+
  - Firefox 3.5+
  - MSIE 8+
  - Safari 4+

# Referencia de la API de Google+

Hay cuatro tipo de recursos, que vienen representados con una estructura de datos JSON:

- People
- Activities
- Comments
- Moments

Más información: <https://developers.google.com/+/api/latest/>

# Recursos

Cada recurso cuenta con una serie de métodos. Por ejemplo, el recurso People tiene:

- get
- search
- listByActivity
- list

Más información:

<https://developers.google.com/+/api/latest/people#resource>

# Estructura de un script Javascript

- ① Se carga la biblioteca JavaScript
- ② Se indican los credenciales de acceso
- ③ Se carga la API del servicio con el que queremos trabajar
- ④ Se inicializa un objeto que encapsula la petición
- ⑤ Se ejecuta el objeto petición
- ⑥ Se procesa el resultado

# 1. Cargando la biblioteca JavaScript

```
<script  
  src="https://apis.google.com/js/client.js?onload=OnLoadCallback">  
</script>
```

- ① Al producirse el evento, llama a la función *callback* que se indica

## 2. Indicando credenciales (I)

Hay dos tipos de acceso:

① Simple:

- Llamadas que no acceden a datos privados
- Hace falta enviar la *API key*

② Autorizada:

- Llamadas que leen o escriben datos privados o de la propia aplicación
- Hace falta un credencial OAuth 2.0

## 2. Indicando credenciales (I)

Ejemplo de acceso simple:

```
var apiKey = 'AIzaSyDy.....VbeXghdvuHDI8A'; // From API Console  
gapi.client.setApiKey(apiKey);
```

Ejemplo de acceso autorizado:

```
// From the API Console  
var clientId = '4764....7773';  
// Services to be used  
var scopes = 'https://www.googleapis.com/auth/plus.me';  
  
gapi.auth.authorize({client_id: clientId, scope: scopes, immediate: true},  
    callback_function);
```

`callback_function` es la función que se ejecutará una vez se haya obtenido la autorización.

### 3. Cargar API del servicio

```
gapi.client.load(API_NAME, API_VERSION, CALLBACK);
```

donde:

- API\_NAME es el nombre de la API
- API\_VERSION es la versión de la API
- CALLBACK es una función opcional a ejecutar cuando se haya cargado

Para cargar la versión 1 de la API de Google+:

```
gapi.client.load('plus', 'v1', function() {
  console.log('loaded.');
});
```

## 4. Objeto que encapsula petición

```
var ApiRequest = gapi.client.METHOD_NAME(PARAMETERS_OBJECT);
```

donde:

- METHOD\_NAME es un método de la API del servicio
- PARAMETERS\_OBJECT es un objeto con parámetros que dependerán del método utilizado

Por ejemplo, la siguiente llamada busca actividades cuyo título sea Google+:

```
var request = gapi.client.plus.activities.search(  
  {'query': 'Google+', 'orderBy': 'best'}  
) ;
```

## 5. Ejecución del objeto petición

```
ApiRequest.execute(callback);
```

donde:

- ApiRequest es un objeto que encapsula la petición (ver paso 4.)
- callback es la función que tratará la respuesta

Por ejemplo:

```
request.execute(function(resp) { console.log(resp); });
```

## 6. Procesar el resultado

- La API devuelve dos objetos a la función callback:
  - ① jsonResp: un objeto JSON
  - ② rawResp: un string con la respuesta HTTP

Cuando la respuesta no pueda ofrecerse como JSON, el valor de jsonResp será *false*; pero rawResp seguirá ofreciendo la respuesta HTTP completa como string.

# Firefox OS

# ¿Qué es Firefox OS?

- Sistema operativo basado en Linux para *smartphones* y tabletas
- Promovido por Mozilla (y otros *partners* industriales)
- Basado completamente en estándares abiertos: HTML5, CSS3 y JavaScript
- Incluye un modelo de privilegios y una API web abierta para comunicarse con el hardware del dispositivo