



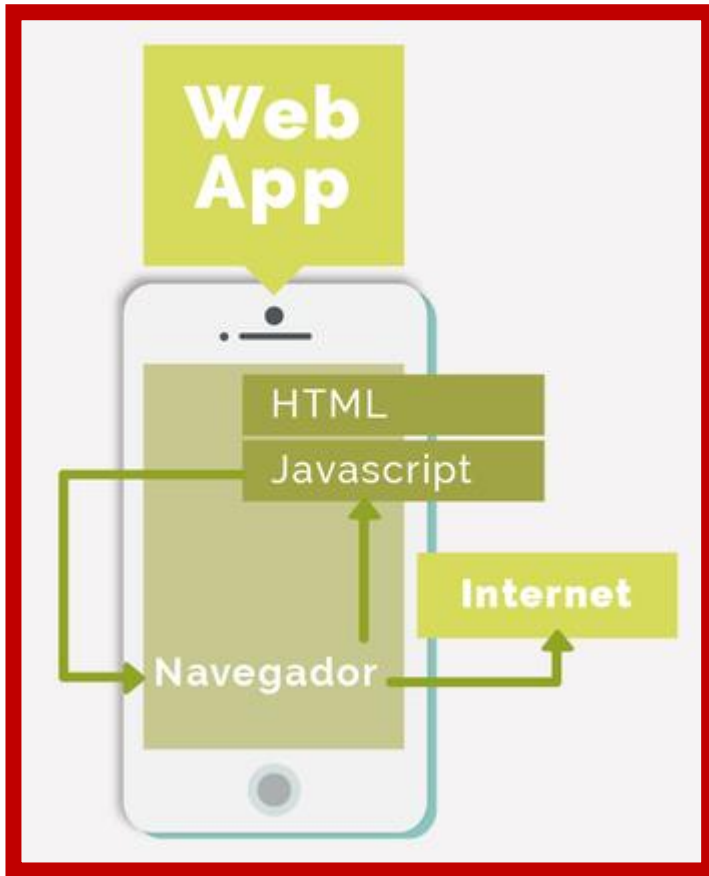
# Desarrollo de apps con tecnologías web

Programación de aplicaciones para dispositivos móviles

Curso 2024-25



# Apps nativas vs web apps vs apps híbridas



# Tecnologías web

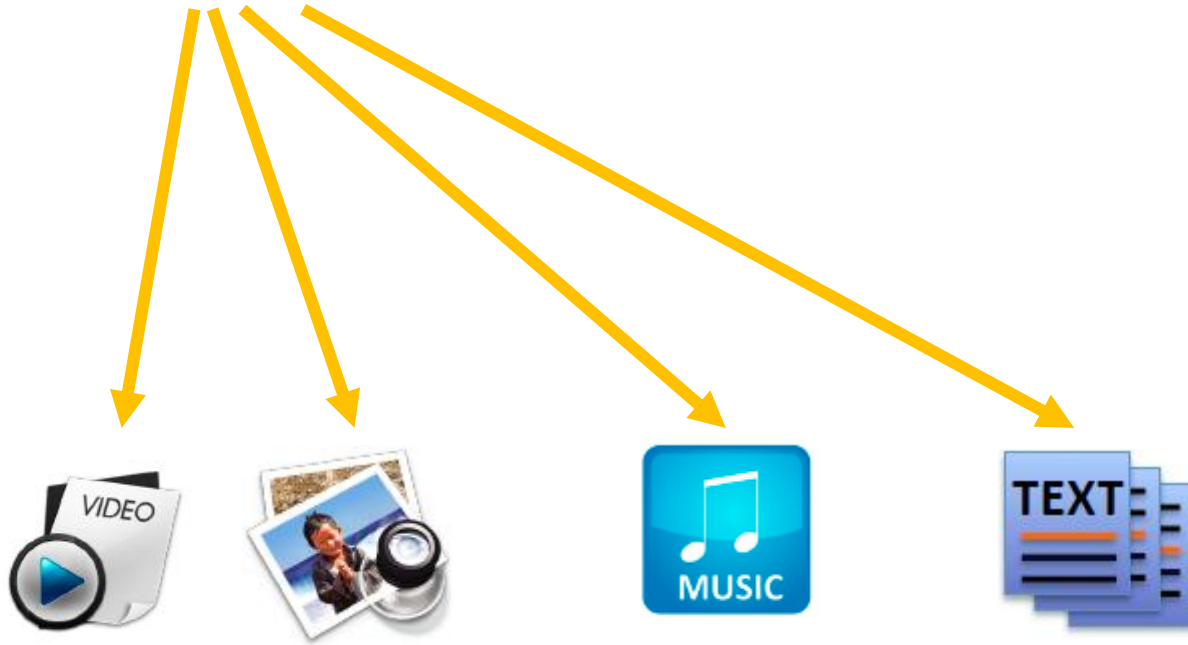
- Las aplicaciones móviles se podrán ejecutar en distintos navegadores independientemente del dispositivo móvil desde el que se inicie.
- La aplicación móvil en este caso deberá estar contenida al menos en una página web (**HTML**), presentará una cierta apariencia (**CSS**) y podrá ser dinámica (**JavaScript**).



```
<!DOCTYPE  
<HTML>  
<HEAD>  
<TITLE>RA  
<LINK REV  
<META NAM
```

Estructura:  
HTML  
(Hypertext  
Markup  
Language)

# Hypertext Markup Language



# Hypertext Markup Language



```
<!doctype html>
<html>
<head>
  <title>Tecnologías Web</title>
</head>
<body>
...
</body>
</html>
```

# Hypertext Markup Language

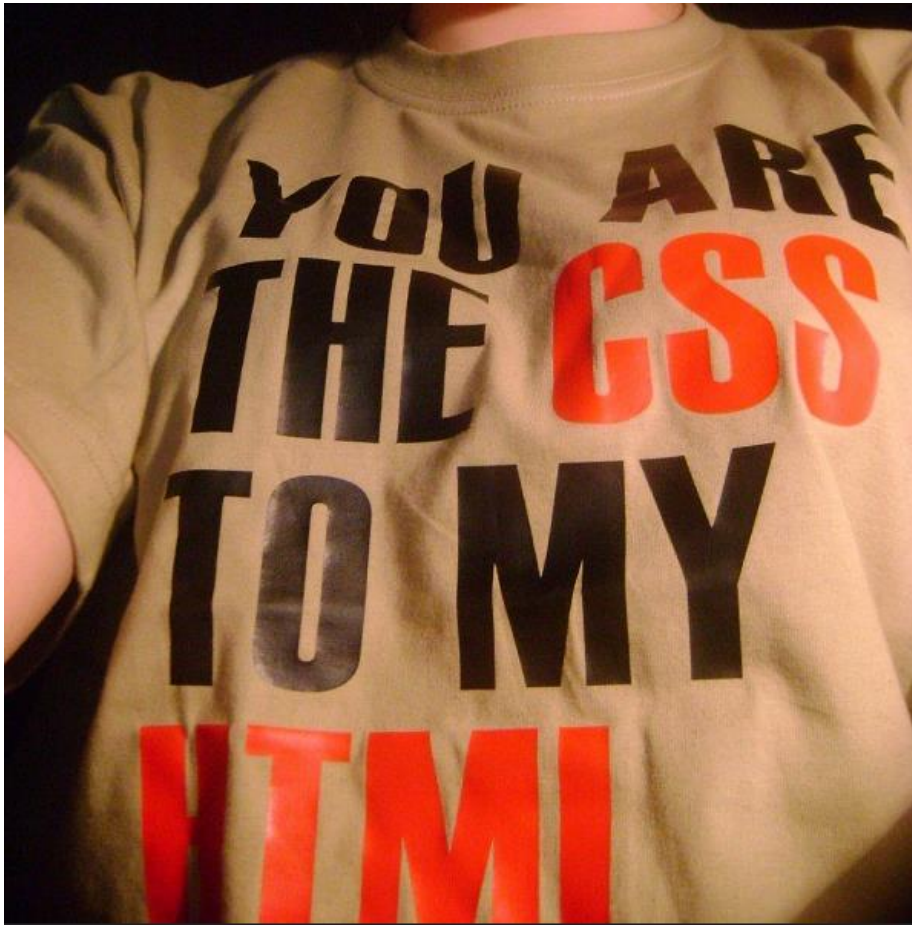


```
<h1>  
  <div>¡Hola mundo!</h1>  
</div>
```



```
<h1>  
  <div>¡Hola mundo!</div>  
</h1>
```

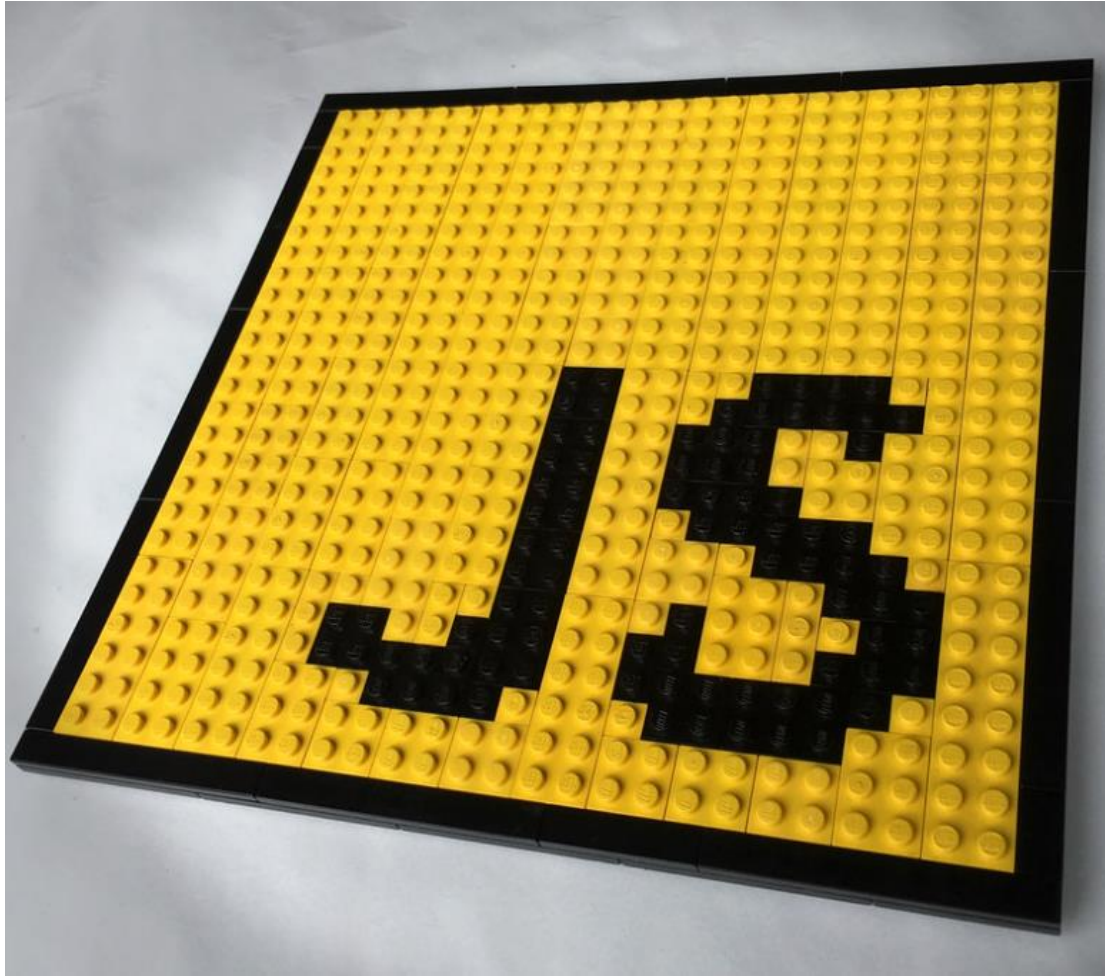




```
1
2  »  »  »  #content .post .post_conte
3  »  »  »  »  margin:0.0.20px.0;1
4  »  »  »  }1
5
6  »  »  »  #content .post .post_conte
7  »  »  »  »  margin:0.0.20px.0;1
8  »  »  »  »  padding-left:15px;1
9  »  »  »  }1
10
```

Estilos: CSS  
(Cascading  
Style Sheets)





```
...tion = columns[currentColumn]+rows[currentRow];  
my = "<img src='pac-maze-ghost.jpg'>"  
myColumn = Math.floor(Math.random()*8);  
myRow = Math.floor(Math.random()*9);  
myPosition = columns[enemyColumn]+rows[enemyRow];  
  
tPosition = "a5";  
= "<img src='Pac-Maze-Man.jpg'>";  
  
start() {  
  getElementById(currentPosition).innerHTML = user;  
  getElementById(enemyPosition).innerHTML = enemy;  
}
```

Lógica:  
JavaScript (JS)

# Lenguajes de la Web

- HTML: Documento y estructura.
- CSS: Apariencia.
- JavaScript: Lógica.



# Navegadores (1/3)

- Programas que permiten acceder a la Web.
- Los navegadores interpretan los documentos HTML.
- Diferencias importantes entre los distintos navegadores:
  - Motor de renderizado: Gecko (Firefox), Blink (Chrome, Opera), WebKit (Chrome iOS, Safari), Trident (Internet Explorer).
  - VM de JavaScript: SpiderMonkey (Firefox), V8 (Chrome), JavaScript Core/Nitro (Safari), Chakra (versiones antiguas de Microsoft Edge).
  - Consideraciones técnicas: tanto a nivel de interfaz como de seguridad.
  - Herramientas de desarrollo: depuradores, *profilers*, *network requests*, etc.

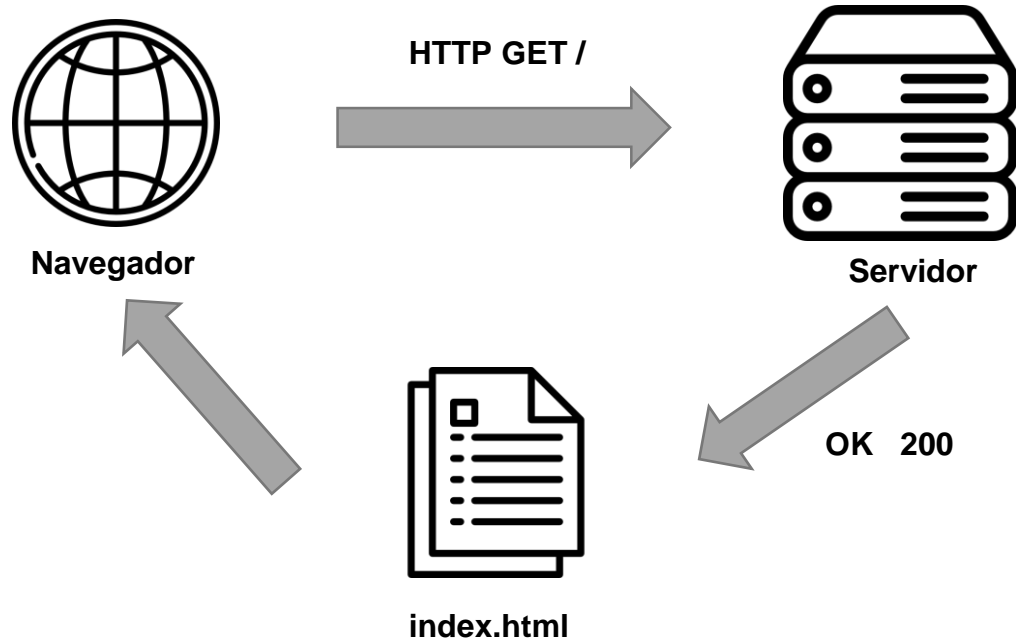
## Navegadores (2/3)

- Para detectar qué soporta cada navegador:
  - [Caniuse.com](https://caniuse.com)
  - [MDN](#): Proporciona detalles concretos sobre la implementación y diferencia entre navegadores.

## Navegadores (3/3)

- Una página web no tiene por qué estar en Internet para que un navegador los muestre: se pueden crear y abrir ficheros en local de documentos HTML sin conexión a Internet.
- El navegador es como si fuera un visor de tipo pdf, pero que entiende HTML, CSS y JavaScript.
- El navegador además es un visor que se conecta: puede acceder y recuperar HTMLs, CSSs y código JavaScript que están en Internet.
  - Se conecta a otro ordenador, le pide archivos, se los descarga y los muestra.
  - Se los puede descargar a un directorio local.

# Peticiones Web



# HTML: Introducción

- **HTML** (*HyperText Markup Language*): Lenguaje para describir páginas web.
- Es un **lenguaje de marcado**, ~~no de programación~~, basado en **etiquetas**.
- Las **etiquetas** van a **describir los contenidos del documento**.  
Ejemplo: `<h1>Acantilado rojo</h1>`
- Una página HTML contiene la descripción de un documento en base a etiquetas HTML y texto en claro (del inglés *plain text*).
- Los documentos HTML se denominan también páginas web.

# HTML: Ejemplo de documento

```
<!DOCTYPE html> <!-- Establece el tipo de HTML -->
<html lang="es"> <!-- Abre la página -->
<head> <!-- `head` son los datos del documento -->
    <meta charset="UTF-8">
    <title>Películas</title>
</head>
<body> <!-- Cuerpo del documento, lo que se ve -->
    <h1>Acantilado rojo</h1>
    <p>John Woo</p>
</body>
</html> <!-- Cierra la página -->
```



# HTML: Etiquetas (1/3)

- Las etiquetas HTML son palabras clave encerradas entre ángulos: **<etiqueta>**.
- Normalmente las etiquetas aparecen por pares, una para su **apertura** y otra para su **cierre**. Ejemplo:

`<strong>Texto a mostrar con mucho énfasis</strong>`

- La etiqueta de cierre se escribe como la de apertura, pero con / antes del nombre de la etiqueta.
- El fragmento de documento que comienza en una etiqueta de apertura y termina en una de cierre se denomina **elemento**.

## HTML: Etiquetas (2/3)

- Los elementos vacíos se cierran en la etiqueta de apertura. Ejemplo:  
`<br />`
- Aunque en muchas versiones de HTML hay elementos cuyas etiquetas de cierre son opcionales (versiones derivadas del metalenguaje **SGML**, definido por la norma ISO 8879 en 1986).

Ejemplo: `<input type="text" id="fname" name="fname"><br><br>`

- Como esto no ocurre con XHTML (versión derivada del metalenguaje **XML**), es buena práctica cerrar siempre de manera explícita los elementos.

## HTML: Etiquetas (3/3)

- Muchas versiones de HTML (las derivadas de SGML) no distinguen entre mayúsculas y minúsculas en los nombres de etiquetas. Dado que XHTML sí que realiza esa distinción, y utiliza siempre nombres en minúsculas, es recomendable utilizar siempre nombres de etiquetas en minúscula.
- Las etiquetas se pueden anidar. Ejemplo:  

```
<p>  
  <em>El dinero</em> es importante, pero <strong>la salud</strong> lo es más.  
</p>
```

*El dinero es importante pero **la salud** lo es más.*

# HTML: Comentarios

- Se encierran entre `<!-- ... -->`

```
<!-- Ejemplo de comentario -->
```

## HTML: Atributos (1/2)

- La mayoría de los elementos HTML pueden tener atributos.
- Los atributos proporcionan información adicional sobre un elemento.
- Se colocan en la etiqueta de apertura del elemento.
- Se especifican mediante pares del tipo **nombre="valor"**. Ejemplo:

```
<a href="http://www.ucm.es">UCM</a>
```

- Aunque hay versiones de HTML que permiten omitir las comillas en el valor (e incluso, como en el caso de los atributos booleanos en HTML5, que permiten omitir el valor en sí), es recomendable seguir el convenio estándar de entrecomillar los valores de los atributos.

## HTML: Atributos (2/2)

- Aunque los nombres y valores de atributos en HTML no distinguen entre mayúsculas y minúsculas, se recomienda utilizar siempre minúsculas.
- Atributos que pueden asociarse con cualquier elemento HTML:
  - **class** : Permite asociar una clase o categoría con el elemento (útil para la asociación de reglas de estilo *Cascading Style Sheets*, CSS).
  - **id** : Permite asociar un identificador único al elemento dentro del documento.
  - **style**: Permite asociar reglas de estilo CSS específicas para el elemento.
  - **title**: Permite asociar un título con el elemento (el navegador puede mostrarlo como un “*tooltip*”).

# HTML: Elementos estructurales básicos

- `<html>`: Define un documento HTML.
- `<h1> ... <h6>`: Define distintos niveles de títulos (encabezado).
- `<body>`: Define el cuerpo del documento.
- `<p>`: Define un párrafo.
- `<hr>`: Marca una línea horizontal en el documento.
- `<br>`: Marca un salto de línea.
- **NOTA:** El contenido es formateado por el navegador (de esta forma, el texto en un párrafo es formateado eliminando blancos superfluos, saltos de línea, etc.; Por tanto, si se desean distintos párrafos, deben marcarse explícitamente).

# HTML: Etiquetas de formato (1/2)

## ■ Formato explícito (no es una buena práctica):

- `<b>`: Texto en negrita
- `<i>`: Texto en cursiva
- `<bdo>`: Permite definir la dirección del texto. Ejemplo:  
`<bdo dir="rtl">hola</bdo> => aloh`

## ■ Formato “semántico” (recomendado):

- `<em>`: Texto enfatizado (normalmente, pero no necesariamente, se visualiza en cursiva).
- `<strong>`: Texto con fuerte énfasis (normalmente visualizado en negrita).
- `<mark>`: Texto resaltado.
- `<ins>`: Texto subrayado.
- `<del>`: Texto tachado.
- `<sup>`: Texto en superíndice.
- `<sub>`: Texto en subíndice.



## HTML: Etiquetas de formato (2/2)

### ■ Marcado de texto de ordenador:

- `<code>`: Marcado de código.
- `<kbd>`: Marcado de texto tecleado.
- `<samp>`: Marcado de código de ejemplo.
- `<var>`: Marcado de una variable.
- `<pre>`: Marcado de texto preformateado.

### ■ Referencias, citas y definiciones:

- `<abbr>`: Marca una abreviatura o acrónimo (con `title` se introduce el término completo).
- `<address>`: Marca la información de contacto de una persona (ej., autor del documento).
- `<blockquote>`: Marca un párrafo citado de otra fuente (con `ref` se enlaza a la fuente).
- `<q>`: Marca una cita corta (le pone comillas dobles).
- `<cite>`: Marca una referencia bibliográfica.
- `<dfn>`: Marca un término definido.

# HTML: Navegadores y declaración *DOCTYPE*

- Los navegadores no muestran directamente las etiquetas, sino que las utilizan para determinar cómo presentar/visualizar los contenidos.
- Dado que existen distintas versiones del lenguaje (HTML, HTML+, HTML 2.0, HTML 4.0.1, XHTML 1.0, HTML5... ), se debe informar al navegador de la versión utilizada.
- Esto se lleva a cabo mediante la declaración DOCTYPE al comienzo del documento. Ejemplos:
  - HTML5: `<!DOCTYPE html>`
  - XHTML 1.0: `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`
- Más información sobre declaraciones HTML:  
[http://www.w3schools.com/tags/tag\\_doctype.asp](http://www.w3schools.com/tags/tag_doctype.asp)

# HTML: Enlaces (1/2)

- HTML permite describir documentos **hipertextuales**: pueden contener enlaces a otros recursos, o a partes del mismo documento.
- La definición de hiperenlaces se lleva a cabo mediante la etiqueta [<a>](#).
- La etiqueta permite enlazar un recurso, especificando su URL como parte del atributo href:

```
<a href="url">Texto del enlace</a>
```

- Dicha URL puede ser absoluta o relativa (en este caso, el navegador la completa con la base del documento donde se sitúa el enlace):

```
<a href="http://www.foo.es/">Enlace</a>
```

```
<a href="http://www.foo.es/index.html">Enlace</a>
```

```
<a href="index.html">Enlace</a>
```

## HTML: Enlaces (2/2)

- La etiqueta permite definir también **anclas** (lugares en un documento a los que puede enlazarse). Para ello debe especificarse un identificador único para el ancla:

```
<a id="comienzo">Comienzo</a>
```

- Las referencias a anclas se realizan anexando `#identificador` al `href` correspondiente:

```
<a href="#comienzo">Ir a comienzo</a>
```

- El comportamiento que resulta de pulsar un enlace puede controlarse mediante el atributo [target](#) de `<a>`. Ejemplos:
  - `_blank`: El recurso se muestra en una nueva ventana o pestaña.
  - `_self`: El recurso se muestra en la ventana o pestaña actual (valor por defecto).

# HTML: Encabezado

- El elemento `<head>` debajo de `<html>` permite especificar información de encabezado del documento:

- `<title>`: Título del documento.
- `<base>`: Base para todas las URLs relativas en el documento.
- `<link>`: Relación con otros recursos. Ejemplo:

```
<link rel="stylesheet" type="text/css" href="estilo.css" />
```

- `<style>`: Permite incluir reglas de estilo en el propio documento.
- `<meta>`: Permite incluir metainformación.

- `<meta name="tipo" content="valores" />`

- Ejemplos:

```
<meta name="keywords" content="mobile,jquery,HTML,CSS,JavaScript" />
```

```
<meta name="author" content="Steven Gordon" />
```

```
<meta name="description" content="Introduction to mobile device apps" />
```

# HTML: Imágenes (1/2)

- **Elemento <img>:** ``. **Ejemplo:**

```

```

- **Es posible especificar el ancho y el alto de la imagen a través de los atributos `width` and `height`.** **Ejemplo:**

```

```

## HTML: Imágenes (2/2)

- También es posible asociar enlaces a regiones de imágenes con el elemento `<map>`. Ejemplo:

```

```

```
<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" href="sol.htm" alt="Sol">
  <area shape="circle" coords="90,58,3" href="mercurio.htm"
alt="Mercurio">
  <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus">
</map>
```

# HTML: Tablas

- Elemento `<table>`
- Cada fila se define mediante `<tr>`
- Cada celda mediante `<td>`
- También pueden definirse encabezados de columna mediante `<th>`

```
<table>
```

```
  <tr><th>Producto</th><th>Valor</th></tr>
```

```
  <tr><td>Bicicleta</td><td>300</td></tr>
```

```
  <tr><td>Balón</td><td>90</td></tr>
```

```
  <tr><td>Triciclo</td><td>100</td></tr>
```

```
</table>
```



**Producto Valor**

Bicicleta 300

Balón 90

Triciclo 100

- **NOTA:** Las tablas deben utilizarse para representar información tabular, pero no para maquetar documentos (para realizar el maquetado están las reglas de estilo).



# HTML: Listas (1/2)

- Listas no ordenadas: `<ul>`
- Listas ordenadas: `<ol>`
- Cada ítem en ambos tipos de lista se describe mediante `<li>`

```
<ul>
```

```
<li>Sevilla</li>
```

```
<li>Málaga</li>
```

```
<li>Almería</li>
```

```
</ul>
```



- Sevilla
- Málaga
- Almería

```
<ol>
```

```
<li>Sevilla</li>
```

```
<li>Málaga</li>
```

```
<li>Almería</li>
```

```
</ol>
```



1. Sevilla
2. Málaga
3. Almería

## HTML: Listas (2/2)

- Listas de definiciones: `<dl>`
- El término definido se marca mediante `<dt>`.
- La definición se marca mediante `<dd>`

```
<dl>  
  <dt>Café</dt>  
  <dd>- Bebida negra caliente</dd>  
  <dt>Leche</dt>  
  <dd>- Bebida blanca fría</dd>  
</dl>
```



Café  
- Bebida negra caliente  
Leche  
- Bebida blanca fría

# HTML: Marcado de secciones

- `<div>`: Permite marcar un bloque en el documento.
- `<span>`: Permite marcar un segmento de línea.
- Como elemento de bloque, `<div>` provoca que su contenido se muestre en una nueva línea.
- Fuera de este efecto, ni `<div>` ni `<span>` tienen otro significado aparte del de delimitar secciones en el documento.
- Sin embargo, pueden contener atributos genéricos, como `id` o `class`, lo que permite asociar a las secciones que marcan información de estilo mediante reglas de estilo.
- En HTML5 la presencia de nuevos elementos semánticos minimiza la necesidad de estos elementos estructurales.

# HTML: Formularios (1/2)

- Los formularios permiten definir controles para introducir información.
- Se marcan mediante elementos de tipo `<form>`
- Los controles en el formulario se definen mediante elementos de tipo `<input>`
- Diferentes tipos de controles:
  - Campos para introducir texto: `<input type="text" name="nombreusuario" />`
  - Campos para introducir claves: `<input type="password" name="clave" />`
  - Grupo de opciones exclusivas (radio buttons):  
`<input type="radio" name="modalidad" value="presencial" />`  
`<input type="radio" name="modalidad" value="online" />`
  - Grupo de opciones no exclusivas (checkboxes):  
`<input type="checkbox" name="hobby" value="lectura" />`  
`<input type="checkbox" name="hobby" value="cine" />`
  - Botón de envío:  
`<input type="submit" name="envioDatos" value="Enviar">`

## HTML: Formularios (2/2)

```
<!DOCTYPE html>
<html>
  <body>
    <form>
      Nombre: <input type="text" name="nombreusuario" /><br /><br />
      Contraseña: <input type="password" name="clave" /><br /><br />
      Modalidad<br />
      • <input type="radio" name="modalidad" value="presencial" /> Presencial
      • <input type="radio" name="modalidad" value="online" /> Online
      Aficiones<br />
      • <input type="checkbox" name="hobby" value="cine" /> Cine
      • <input type="checkbox" name="hobby" value="lectura" /> Lectura
      <input type="submit" name="envioDatos" value="Enviar" />
    </form>
  </body>
</html>
```

Nombre:

Contraseña:

Modalidad

- ☐ Presencial
- ☐ Online

Aficiones

- ☐ Cine
- ☐ Lectura



# HTML: Entidades

- Para utilizar caracteres reservados en HTML en el contenido es necesario utilizar **entidades**.
  - `&nombre;`
  - `&#código;`
- Ejemplos de entidades:
  - `&lt;` (equivalente a `&#60;`;) equivale a `<`
  - `&nbsp;` equivale a espacio en blanco
  - `&euro;` (equivalente a `&#8364;` o a `&#x20AC;`) equivale a €
- Ver [referencias](#) del lenguaje para un listado completo de entidades.

# HTML: Juego de caracteres

- En HTML4:

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

- En HTML5:

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

# HTML5: Introducción

- HTML5 es el último estándar para HTML.
- La versión anterior, HTML 4.01, se remonta a 1999.
- HTML5 se ha diseñado para sustituir tanto a HTML4, como a XHTML y al nivel 2 de DOM (*Document Object Model*) para HTML.
- HTML5 permite mostrar contenido muy variado sin necesidad de utilizar *plug-ins* externos: animaciones, gráficos, música y películas.
- HTML5 resulta una buena opción para construir aplicaciones web sofisticadas.
- HTML5 es multi-plataforma. Se ha diseñado para funcionar, por ejemplo, con PCs, dispositivos móviles y TV interactivas.



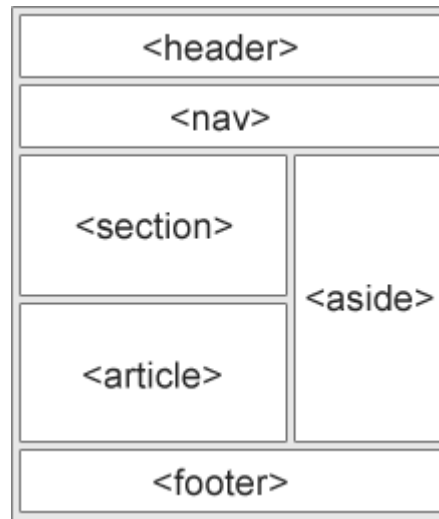
## HTML5: Elementos semánticos (1/2)

- Los elementos semánticos determinan el significado de su contenido, por ejemplo, `<table>`. Se contraponen a los elementos estructurales, como serían `<div>` o `<span>`.
- HTML5 introduce muchos elementos semánticos que sustituyen la necesidad de utilizar `<div>` o `<span>`.
- De esta manera, en lugar de utilizar `<div class="header">` puede utilizarse el elemento `<header>` en HTML5.

# HTML5: Elementos semánticos (2/2)

## ■ Elementos semánticos en HTML5:

- `<header>`
- `<nav>`
- `<section>`
- `<article>`
- `<aside>`
- `<figure>`
- `<figcaption>`
- `<footer>`
- `<details>`
- `<summary>`
- `<mark>`
- `<time>`



## HTML5: Nuevos elementos

- `<canvas>`: Define una sección en la que se puede dibujar utilizando lenguajes de script, por ejemplo, JavaScript.
- Elementos multimedia para audio y vídeo.
- Nuevos elementos para formularios.
- Nuevos elementos semánticos y estructurales.
- Aparte de estas extensiones, HTML5 también ha eliminado elementos en HTML4 obsoletos o poco utilizados.

# HTML5: Canvas

- `<canvas>`: Se utiliza como contexto en el que dibujar gráficos mediante scripts.
- Se define únicamente el panel para gráficos y ofrece métodos que pueden ser invocados a través de scripts para dibujar líneas, cajas, círculos, texto, añadir imágenes...

## HTML5: SVG

- Otra forma de crear gráficos en HTML5 es a través de [<svg>](#).
- `<svg>` permite insertar gráficos vectoriales 2D descritos mediante el lenguaje de marcado SVG (*Scalable Vector Graphics*).
- La diferencia con `<canvas>` es que en `<canvas>` los gráficos los crean scripts invocando a los métodos de `<canvas>`.
- Con `<svg>` es posible describir los gráficos declarativamente. Dichas descripciones se integran en el DOM resultante, y pueden ser manipuladas a través de scripts, como cualquier otro elemento.

# HTML5: Multimedia

- HTML5 incluye elementos para insertar vídeo (<video>) y audio (<audio>) en las páginas sin necesidad de emplear plug-ins externos.

```
<video width="320" height="240" controls>  
  <source src="movie.mp4" type="video/mp4">  
</video>
```

```
<video width="320" height="240" controls>  
  <source src="movie.mp4" type="video/mp4">  
</video>
```

```
<audio controls>  
  <source src="horse.mp3" type="audio/mpeg">  
</audio>
```

```
<audio controls>  
  <source src="horse.ogg" type="audio/ogg">  
</audio>
```

# HTML5: Formularios – Nuevos tipos de entrada

- HTML5 extiende los posibles valores para `type` en `<input>`:

- [color](#)
- [date](#)
- [datetime](#) (UTC)
- [datetime-local](#)
- [email](#)
- [month](#)
- [number](#)
- [range](#)
- [search](#)
- [tel](#)
- [time](#)
- [url](#)
- [week](#)

# HTML5: Formularios – Nuevos controles

- <datalist>: permite definir una lista predefinida de valores para un elemento `<input>`

```
<input list="browsers" name="browsers">  
  <datalist id="browsers">  
    <option value="Internet Explorer">  
    <option value="Firefox">  
    <option value="Chrome">  
    <option value="Opera">  
    <option value="Safari">  
  </datalist>
```

- <output>: representa el resultado de un cálculo.



# HTML5: Formularios – Nuevos atributos

## ■ Nuevos atributos en <form>

- autocomplete
- novalidate

## ■ Nuevos atributos en <input>

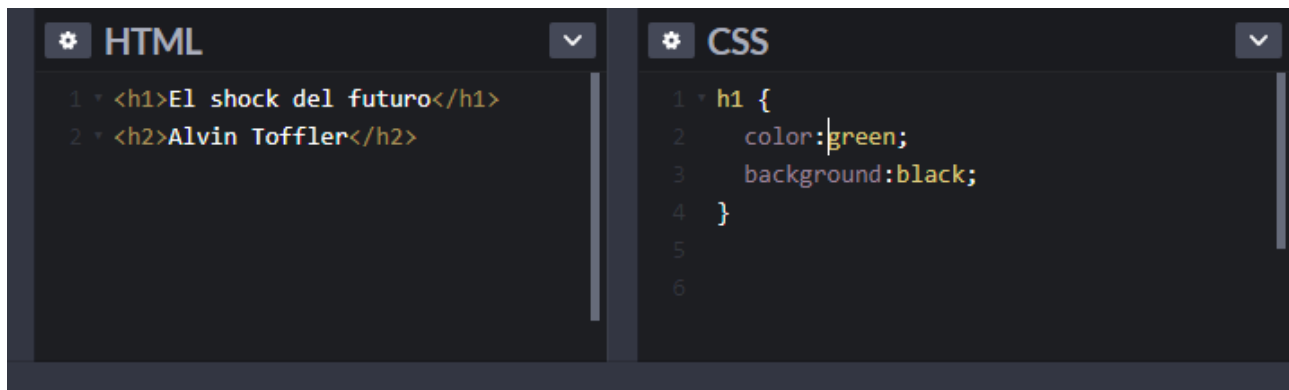
- |                  |                    |
|------------------|--------------------|
| • autocomplete   | • list             |
| • autofocus      | • min y max        |
| • form           | • multiple         |
| • formaction     | • pattern (regexp) |
| • formenctype    | • placeholder      |
| • formmethod     | • required         |
| • formnovalidate | • step             |
| • formtarget     |                    |
| • height y width |                    |

## CSS: Introducción (1/2)

- **CSS** (*Cascading Style Sheets*): Permite establecer la apariencia de los elementos a través de propiedades específicas (implica aplicar los estilos a cada uno de los elementos HTML de manera individual).
- CSS permite definir **reglas de estilo** para controlar la forma en la que se muestran los elementos HTML.
- A través de **hojas de estilo**, CSS permite separar la estructura del documento (descrito mediante HTML) de los detalles de presentación de una manera unificada (por ejemplo, tipos de letra, colores, tamaños y posición).

## CSS: Introducción (2/2)

- Una **hoja de estilo** es un archivo de texto que contiene la apariencia asignada para cada tipo de elemento.
- De esta manera, un mismo documento puede presentarse de múltiples maneras simplemente cambiando la hoja de estilo asociada al mismo.



The image shows a code editor with two panels. The left panel is titled 'HTML' and contains two lines of code: `<h1>El shock del futuro</h1>` and `<h2>Alvin Toffler</h2>`. The right panel is titled 'CSS' and contains a CSS rule for `h1` with `color: green;` and `background: black;`.

```
HTML
1 <h1>El shock del futuro</h1>
2 <h2>Alvin Toffler</h2>

CSS
1 h1 {
2   color: green;
3   background: black;
4 }
5
6
```

**El shock del futuro**

Alvin Toffler

# CSS: Sintaxis

- Una hoja de estilo consta de una **secuencia de reglas de estilo**.
- Cada regla de estilo es de la forma:  

```
selector {propiedad: valor; ...; propiedad: valor }
```
- El **selector** es un patrón que selecciona un conjunto de elementos del documento.
- Los pares `propiedad:valor` definen **atributos de presentación** que se añaden como pares `atributo-valor` adicionales a los elementos seleccionados. Ejemplo:

```
p /* regla aplicable a elementos <p> */  
{  
    color:red;           /* color del texto */  
    text-align:center;   /* alineado del texto */  
}
```

# CSS: Selectores

- Los selectores son expresiones que seleccionan conjuntos de elementos en el documento:
  - Nombres de etiqueta (por ejemplo, `p` y `table`).
  - Identificadores de elementos (por ejemplo, `#para1`): para seleccionar elementos individuales en virtud del valor del atributo `id`.
  - Clases de elementos (por ejemplo, `.class1`): para seleccionar elementos en virtud del valor del atributo `class`.
  - Clases de elementos de un determinado tipo (por ejemplo, `p.class1`).
- Es posible indicar varios selectores para una misma regla. Ejemplo:

```
h1,h2,p
{
    color:red;
    text-align:center;
}
```

# CSS: Asociación de CSS con documentos HTML

- Para añadir una hoja de estilo a un documento, utilizar `<link>` en el `<head>` del documento.

```
<link rel="stylesheet" type="text/css" href="mystyle.css">
```

- Es posible incorporar las reglas de estilo en el propio documento, como contenido de `<style>` en `<head>`. Ejemplo:

```
<style>
  hr {color:sienna;}
  p {margin-left:20px;}
  body {background-image:url("images/background.gif");}
</style>
```

- Por último, es posible indicar el estilo de elementos individuales como valor del atributo `style`.

```
<p style="color:sienna;margin-left:20px;">Un párrafo</p>
```

# CSS: Aplicación de las reglas de estilo

- Un mismo documento puede incluir:
  - Varias CSS externas.
  - Reglas de estilo internas.
  - Estilos específicos en atributos `style`.
- También puede asumirse la existencia de una CSS por defecto (navegador).
- Los valores de los atributos de estilo se determinan mediante la unión de los cuerpos de todas las reglas aplicables. En caso de conflicto para una propiedad (es decir, tiene distintos valores en distintas reglas aplicables) se aplicarán en base a su prioridad:
  - Primero, el valor definido en el atributo `style`.
  - Segundo, el valor definido en las CSS introducidas por `<style>` y por `<link>`, en sentido inverso de aparición.
  - Por último, el valor definido en la CSS por defecto del navegador.

# CSS: Estilo de fondo (1/2)

## ■ Propiedades CSS utilizadas para los estilos de fondo:

- `background-color`
- `background-image`
- `background-repeat` (si hay imagen, determina cómo se repite)
- `background-attachment` (si hay imagen, determina si está sujeta a desplazamiento)
- `background-position` (si hay imagen, determina su posición)

## ■ Ejemplos:

```
.class1{background-color:#b0c4de;}  
body  
{  
    background-image:url("img_tree.png");  
    background-repeat:no-repeat;  
    background-position:right top;  
}
```



## CSS: Estilo de fondo (2/2)

- Es posible resumir toda la información de estilo de fondo en una única propiedad `background` (se ponen, en secuencia, los valores para `background-color`, `background-image`, `background-repeat`, `background-attachment` y `background-position`). Ejemplo:

```
body {background:#ffffff url("img_tree.png") no-repeat right top;}
```

# CSS: Formato de texto

■ `color`: Color del texto.

■ Codificación de colores en CSS:

- Valor hexadecimal (RGB). Ejemplo: `#ff0000`
- Valor RGB. Ejemplo: `rgb(255, 0, 0)`
- En caso de que exista, un nombre para el color. Ejemplo: `red`

■ `text-align`: Alineado del texto (`center`, `right`, `justify`)

■ `text-decoration`: Decoración del texto. Ejemplos:

```
a {text-decoration:none}
h1 {text-decoration:overline;}
h2 {text-decoration:line-through;}
h3 {text-decoration:underline;}
```

■ `text-transform`: Procesamiento de texto (`uppercase`, `lowercase`, `capitalize`)

■ `text-indent`: Indentado de la primera línea. Ejemplo: `p {text-indent:50px;}`

■ Otras propiedades: `vertical-align`, `white-space`, `word-space`, etc.

# CSS: Fuentes

- La especificación de una fuente está gobernada por su familia, su estilo y su tamaño.
- **Familia:** `font-family` (genéricas: `serif`, `sans-serif`, `monospace`; específicas: `Times`, `"Times New Roman"`). Si se dan varios valores, se aplica el primero que exista en el sistema. Ejemplo: `p{font-family:"Times New Roman", Times, serif;}`
- **Estilo:** `font-style` (`normal`, `italic`, `oblique`). **Negrita utilizando** `font-weight`.
- **Tamaño:** `font-size`.
- **Tamaños en CSS:**
  - *Absolutos:* `10cm`, `10mm`, `10i` (pulgadas), `10pt` (1 punto = 1/72 pulgadas), `10pc` (picas, 1 pica = 12 puntos).
  - *Relativos:* `10px` (píxeles), `10em` (10 veces el tamaño por defecto de la letra del elemento), `10ex` (10 veces la altura por defecto de la letra del elemento), `80%` (80% del tamaño del elemento padre).
- **Otras propiedades:** `font-variant`, `font` (permite especificar todas las propiedades en una única).

# CSS: Estilo de listas

- `list-style-type`: Establece el estilo de la decoración los ítems de la lista (viñetas y numeración). Ejemplos:

```
ul.a {list-style-type: circle;}
```

```
ul.b {list-style-type: square;}
```

```
ol.c {list-style-type: upper-roman;} => números romanos en mayúscula (I, II, III, etc.).
```

```
ol.d {list-style-type: lower-alpha;} => letras ASCII en minúscula (a, b, c, ... z).
```

- `list-style-image`: Establece una imagen como decoración de los ítems. Ejemplo:

```
ul{list-style-image: url('sqpurple.gif');}
```

- Otras propiedades: `list-style-position`, `list-style`.

# CSS: Estilo de enlaces

- Los enlaces pueden tener diferente estilo dependiendo de su estado.
- Para ello, el estado puede especificarse en el selector:

`a:link` => Un enlace no visitado.

`a:visited` => Un enlace ya visitado.

`a:hover` => Un enlace sobre el que está pasando el ratón u otro elemento de selección.

`a:active` => Un enlace que está seleccionado.

- Ejemplos:

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

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

```
a:hover {text-decoration:underline;}
```

```
a:active {text-decoration:underline;}
```

## CSS: Estilo de tablas

- **border:** Estilo del borde. Ejemplo:

```
table, th, td {border: 1px solid black;}
```

- **border-collapse:** Permite especificar que los bordes adyacentes se fusionen en uno. Ejemplo: `table {border-collapse: collapse;}`

- **width y height:** Anchura y altura. Ejemplos:

```
table {width:100%;vc}  
th{height:50px;}
```

- **padding:** Espacio entre el borde y el contenido. Ejemplo:

```
td{padding:15px;}
```

- Son aplicables también las propiedades sobre alineado de texto (`text-align`), así como sobre color de fondo (`background-color`) y color de texto (`color`), tanto sobre la tabla total como sobre sus subelementos.

## CSS: Modelo de caja (1/2)



- `width` y `height`: Anchura y altura del contenido.
- `padding`, `border` y `margin`: Tamaño de la distancia del borde al contenido, características del borde en sí, y tamaño del margen que separa el documento de sus alrededores.

## CSS: Modelo de caja (2/2)

- `border` resume tres propiedades:
  - `border-width`: Anchura del borde.
  - `border-style`: Estilo del borde (`none`, `solid`, `double`...).
  - `border-color`: Color del borde.
- También se pueden fijar diferentes tipos de borde en cada lado. Ejemplo:  
`border-top-style`, `border-left-color`.
- Es posible especificar márgenes asimétricos (`margin-left`, `margin-top`...), y también paddings asimétricos (`padding-left`, `padding-bottom`...).
- Las descripciones pueden realizarse también especificando una secuencia de valores para `margin` / `padding`.
- Aparte de la anchura y altura exacta de un elemento (`width`, `height`), también es posible especificar cotas inferiores y superiores a dichos valores (`max-width`, `min-height`...).



# CSS: Posicionamiento (1/2)

- CSS introduce propiedades que permite el posicionamiento de los elementos del documento.
- Propiedades para posicionar elementos: `top`, `bottom`, `left` y `right`. E
- Resulta necesario fijar antes el valor de la propiedad `position`. El efecto dependerá del tipo de posicionamiento elegido:
  - Posicionamiento estático (por omisión). Los elementos se posicionan de acuerdo con el orden normal del documento. Las propiedades de posicionamiento no tienen ningún efecto.
  - Posicionamiento fijo (`position: fixed`). Permiten situar el elemento en una posición fija de la página. Dicha posición no se verá alterada, aunque se haga scroll.
  - Posicionamiento relativo (`position: relative`). La posición es relativa a su ubicación de acuerdo con el flujo normal de elementos.
  - Posicionamiento absoluto (`position: absolute`). La posición es absoluta con respecto al primer ancestro no posicionado de forma estática (o bien, respecto a `<html>`).

## CSS: Posicionamiento (2/2)

- Cuando los elementos se solapan, es posible especificar el orden de visualización mediante `z-index`.
- Mediante `overflow` es posible controlar qué ocurre cuando el tamaño del contenido de un elemento excede a la anchura del mismo. Ejemplo: `overflow:scroll`.
- Mediante `float` es posible alternar el flujo normal de elementos (uno debajo de otro) situando un elemento a la izquierda (`float:left`), o a la derecha (`float:right`), y permitiendo que los que vienen después se distribuyan alrededor del mismo (a no ser que se dicho comportamiento se inhabilite con `clear`. Ejemplo: `clear:both`)
- Otras propiedades: [cursor](#)...

# CSS3: Introducción

- **CSS3** es la versión más reciente de CSS.
- Está dividido en diferentes módulos.
- Cubre todos los aspectos de las versiones anteriores (distribuidos en distintos módulos), y añade nuevos módulos.
- Algunos de los módulos más importantes de CSS3:
  - Selectores.
  - Modelo de caja.
  - Fondos y bordes.
  - Valores de imágenes y contenido de reemplazamiento.
  - Efectos de texto.
  - Transformaciones 2D/3D.
  - Animaciones.
- Mayor soporte para disposición multi-columna.
- Mayor soporte para interfaces de usuario.

## CSS3: Bordos

- Aparte de las propiedades básicas de bordes en CSS, en CSS3 es posible definir bordes redondeados, añadir sombras a las cajas, y usar una imagen en los bordes.
- `border-radius`: Permite definir bordes redondeados, especificando el radio de las esquinas. Ejemplo: `div {border:2px solid; border-radius:25px;}`
- `box-shadow`: Permite añadir un efecto de sombra al borde. Ejemplo: `div{box-shadow: 10px 10px 5px #888888;}`
- `border-image`: Permite utilizar una imagen para definir el borde. Ejemplo: `div{border-image:url(border.png) 30 30 round;}`
- Existen propiedades que permiten especificar los valores individualmente: , `border-image-source`, `boder-image-slice`...
- No está soportado en todas las versiones de navegadores (en algunos casos hay que indicar prefijo, por ejemplo, `-webkit-border-image`).

## CSS3: Fondos

- CSS3 permite controlar el tamaño de la imagen de fondo (`background-size`).
- También permite posicionar la imagen en la caja de borde, la caja de relleno, o la caja de contenido (`background-origin`).
- También permite, como valor de `background`, especificar varias imágenes para componer el fondo.

## CSS3: Degradados

- CSS3 permite definir degradados: Transiciones suaves entre dos o más colores.
- Esto permite prescindir de imágenes para obtener estos efectos (como en versiones anteriores): Menor tiempo de carga, diseños más robustos al escalado.
- Dos tipos de degradados: Lineales (sobre una línea que comienza desde arriba, desde abajo, desde la derecha, desde la izquierda, o diagonal), y radiales (comienzan en el centro). Ejemplos:

```
#grad{background: linear-gradient(red, blue); }  
#grad{background: radial-gradient(red, green, blue); }
```
- En versiones antiguas de navegadores puede ser necesario utilizar prefijos (por ejemplo, `-webkit-linear-gradient`, `-webkit-radial-gradient`).

## CSS3: Efectos de texto

- CSS3 soporta nuevos efectos para la presentación de texto

- Algunos de estos efectos:

- `text-shadow`: Añade sombra a un texto. Ejemplo:

- ```
h1{text-shadow: 5px 5px 5px #FF0000;}
```

- `text-wrap`: Permite que el texto se divida cuando excede la caja de contenido, aunque implique partir una palabra. Ejemplo:

- ```
p {word-wrap:break-word;}
```

## CSS3: Carga remota de fuentes

- CSS3 permite cargar remotamente fuentes: de esta manera se pueden usar fuentes que no están definidas en la máquina del cliente.

```
@font-face
{
    font-family: myFirstFont;
    src: url(sansation_light.woff);
}
```



# CSS3: Transformaciones

- CSS3 permite aplicar transformaciones geométricas a elementos (por ejemplo, translación, rotación, escalado, inclinación, o una transformación más general, definida por su matriz de transformación).

- Ejemplos de transformaciones 2D:

```
div{transform: rotate(30deg);}  
div{transform: translate(50px,100px);}  
div{transform: rotate(30deg);}  
div{transform: scale(2,4);}  
div{transform: skew(30deg,20deg);}  
div{transform:matrix(0.866,0.5,-0.5,0.866,0,0);}
```

- Ejemplos de transformaciones 3D:

```
div{transform: rotateX(120deg);}  
div{transform: translate3D(50px,50px,50px);}
```

- En navegadores antiguos, puede ser necesario usar prefijos.

## CSS3: Transiciones

- CSS3 permite especificar que se produzca un efecto de transición cuando se cambia el valor de una propiedad de estilo (por ejemplo, a través de JavaScript). Ejemplos:

```
div{transition: width 2s, height 2s, transform 2s;}  
div{transition: width 1s linear 2s;}
```

- En navegadores antiguos, puede ser necesario usar prefijos.

# CSS3: Animaciones

- CSS3 permite especificar animaciones como secuencias de transiciones. Ejemplos:

```
div{animation: animacion1 5s;}
@keyframes animacion1 {
  from {background: red;}
  to {background: yellow;}
}
#p56{animation: animacion2 5s;}
@keyframes animacion2
{0%    {background: red;}
 25%   {background: yellow;}
 50%   {background: blue;}
100%   {background: green;}}
```

- En navegadores antiguos, puede ser necesario usar prefijos.

## CSS3: Disposición multi-columna

- CSS3 facilita el formato de texto a varias columnas. Ejemplo:

```
#p3 {  
  column-count:3;  
  column-gap:40px;  
  column-rule:3px outset #ff00ff;  
}
```

- En navegadores antiguos, puede ser necesario usar prefijos.

## CSS3: Opciones de interfaz de usuario

- Presenta varias opciones para mejorar la interfaz con el documento:  
`appearance, box-sizing, icon, resize...`
- Una de las más interesantes es `resize`, que permite especificar si el usuario puede ajustar o no el tamaño de un elemento.

# CSS: Enlaces de interés

- [Hover.css](#)
- [Magic Animations](#)
- [Animista](#)
- [Animate.css](#)
- [CSSHAKE](#)
- [Wicked CSS](#)
- [50 Projects 50 days](#)
- [Font Awesome](#)
- Vídeo Codemotion: [How CSS came about](#)