



pue

IMPULSANDO EL CONOCIMIENTO
TIC CUALIFICADO

**Desarrollo de
aplicaciones HTML5
para móviles**

Modulo 1: introducción histórica y galería de aplicaciones

pue

Que es HTML (Hyper Text Mark Language)

- Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.
- Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web.
- Define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos.

Historia del HTML

- 1995: Se formaliza el HTML 2.0: sintaxis y la mayoría de las reglas actualmente implementadas
- 1997: HTML 3.2 durante mucho tiempo ignorado por las empresas que proveen navegadores, las cuales implementan sus propias etiquetas
- 1998: se otorga peso a las recomendaciones del W3C y se promocionan navegadores basados en dichos estándar
- 1999: Se estabiliza la sintaxis y la estructura del HTML 4.0, convirtiéndose en el estándar para la web
- 2000: Nace el XHTML 1.0 diseñado para adaptar el HTML a XML. Uso de DTD para renderizar como HTML

Historia del HTML

- 2000-2004: Incremento de las conexiones produciéndose una demanda en el campo del desarrollo de las aplicaciones y la multimedia, donde tecnologías como Flash y Ajax hacen que se trabaje en la especificación XHTML 2.0
- 2004: Apple, Mozilla y Opera proponen evolucionar el estándar HTML 4.0. Aunque son rechazados, forman el WHATWG (Web Hypertext Application Technology Working Group)
- 2005: Se publica el borrador de trabajo Web Applications 1.0

Historia del HTML

- 2007: El W3C adopta el trabajo de WHATWG publicando lo que sería el borrador de trabajo de HTML 5
- 2009: Última llamada expedida para el proyecto de trabajo de HTML 5. El W3C no renueva XTM 2.0
- 2010: Borrador del W3C para HTML5
- Se prevé que HTML5 finalice las pruebas finales en 2020, y alcance la categoría de recomendación de pleno derecho en 2022.

Historia del HTML

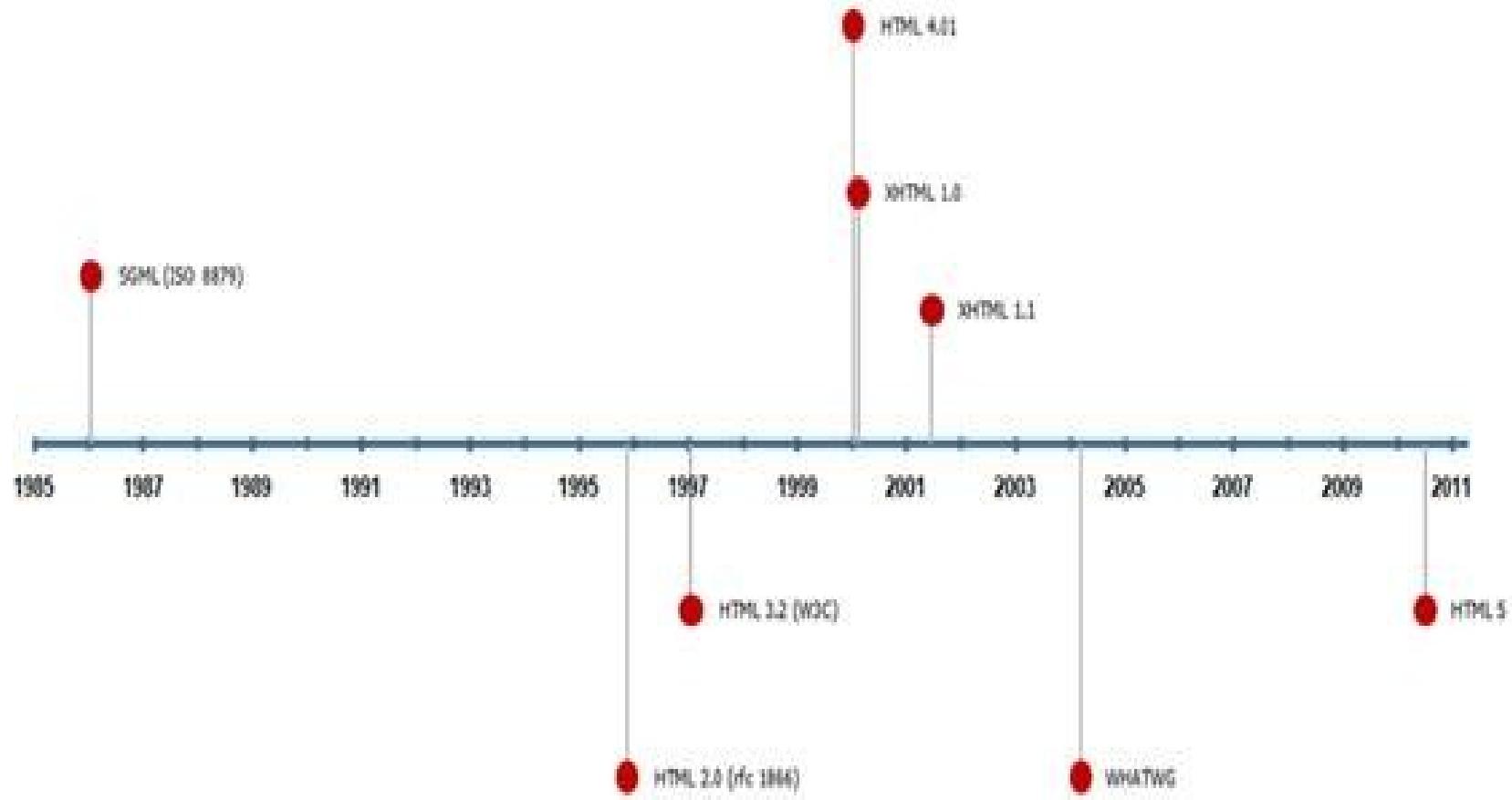


Imagen 1: Evolución del lenguaje de marcado HTML

HTML 5

- Unifica dos elementos tecnológicos.
 - 1.- Evolución por un lado de lo que ha sido hasta ahora el lenguaje de marcado HTML4
 - 2.- Evolución de la API Document Object Model 2 (DOM2).
- A través de HTML5 vamos a disponer de nuevas API's que van a ayudar a los desarrolladores a generar aplicaciones web mucho más dinámicas y ricas.

HTML 5: Novedades

- No se va a quedar solamente en una simple redefinición de etiquetas de marcado sino que va mucho más allá:

Nuevas etiquetas semánticas:

Google debe entender los contenidos

Video, audio y animación sin plugins

Acceso universal: cualquier dispositivo, cualquier navegador, cualquier versión.

Nuevos elementos de formularios

Más usabilidad y menor uso de JavaScript

Qué es CSS (cascading style sheets)

- Lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML.
- La mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Qué es CSS (cascading style sheets)

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas:

- Obliga a crear "documentos semánticos" → HTML/XHTML bien definidos y con significado completo.
- Mejora la accesibilidad
- Reduce la complejidad en su mantenimiento
- Visualizar en infinidad de dispositivos diferentes.

Historia del CSS

- W3C(World Wide Web Consortium) propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML y se presentaron nueve propuestas.
- Las dos propuestas que se tuvieron en cuenta fueron la CHSS (Cascading HTML Style Sheets) y la SSP (Stream-based Style Sheet Proposal).
- Entre finales de 1994 y 1995 se unieron para tomar lo mejor de cada propuesta y lo llamaron CSS (Cascading Style Sheets).
- En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS y lo añadió a su grupo de trabajo de HTML.

Historia del CSS

- A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como "CSS nivel 1".
- En 1998 se publica su segunda recomendación "CSS nivel 2".
- En 2004 se inicia la versión de CSS que utilizan todos los navegadores de hoy en día es CSS 2.1, una revisión de CSS 2 que aún se está elaborando
- En 2011 se publica la siguiente recomendación de "CSS nivel 3".
- En 2019 se estima la siguiente recomendación de "CSS nivel 4".
- Por el momento, ningún navegador tiene soporte completo de CSS 2.1.

Nuevo en CSS3

- Usar cualquier tipografía
- Efectos decorativos: sombras, RGBa, bordes redondeados, gradientes
 - menor uso de imágenes
- Movimientos: transformaciones, transiciones, animaciones
 - menor uso de JavaScript
- Diseño adaptable
- Selectores más precisos
 - menor uso de marcado innecesario

Qué es JavaScript

- A principios de los años 90 con aplicaciones web cada vez más complejas y una velocidad de navegación lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario.
- Si el usuario no llenaba correctamente un formulario, no esperaba a que el servidor mostrara los errores.
- Brendan Eich, programador de Netscape, adaptó tecnologías existentes (como ScriptEase) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995 y lo denominó LiveScript.
- Netscape firmó una alianza con Sun Microsystems y cambiaron el nombre por el de JavaScript (marketing, Java era la palabra de moda)

Qué es JavaScript

- Netscape Navigator 3.0 ya incorporaba la siguiente versión del lenguaje, la versión 1.1.
- Al mismo tiempo, Microsoft lanzó, en su navegador Internet Explorer 3, JScript que era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales.
- Para evitar una guerra de tecnologías, Netscape decidió que lo mejor sería estandarizar el lenguaje JavaScript.
- De esta forma, en 1997 se envió la especificación JavaScript 1.1 al organismo ECMA European Computer Manufacturers Association).
- La organización internacional para la estandarización (ISO) adoptó el estándar ECMA-262 dando lugar al estándar ISO/IEC-16262.

Qué es JavaScript

- En Junio de 1998 se realizaron pequeñas modificaciones para adaptarlo al estandar ISO/IEC-16262 y se creó la segunda edición.
- La tercera edición del estándar ECMA-262 publicada en Diciembre de 1999, es la versión que utilizan los navegadores actuales
- Actualmente se encuentra en desarrollo la cuarta versión de ECMA-262, que podría incluir novedades como paquetes, namespaces, definición explícita de clases, etc.

Modulo2: Experiencia de usuario sobre dispositivos móviles y sus limitaciones asociadas al hardware

pue

La pantalla como limitación: Diseño web adaptativo

- El diseño web adaptativo surgió como respuesta al problema del diseño web convencional de mostrar sitios web correctamente en todos los dispositivos, **sobre todo en dispositivos móviles.**
- El diseño web adaptativo no requiere de varias versiones de una misma página web para mostrarse correctamente.
- Cubre todos los dispositivos y las resoluciones de pantalla que cada uno traiga, es decir, el sitio web estará optimizado de manera nativa para todo tipo de dispositivos: PCs, tabletas, teléfonos móviles ...

HTML 5 y Diseño web adaptativo

- Incorpora nuevos elementos respecto del HTML anterior y reemplaza otros ya obsoletos.
- Es más eficiente y ocupa menos recursos en la computadora del visitante.
- El usuario no necesita ni descargar ni tener instalado ningún plugin adicional como ocurre por ejemplo con sitios que utilizan animaciones en flash.
- El diseño web adaptativo utiliza HTML 5 y CSS 3 puro y se complementa con JavaScript para diferentes funcionalidades y animaciones.

HTML 5 y Diseño web adaptativo

- El reajuste de elementos se realiza sin perder tamaño.
- Un párrafo o una imagen se adaptan a la nueva resolución, pero no se achica o se hace ilegible
- Re-adapta el menú horizontal y lo convertiría en menú vertical, con iconos y textos grandes, para que el usuario pueda ver y seleccionar el contenido correctamente.

pue

Modulo3:

Herramientas de

desarrollo y depuracion

en el navegador

pue

Herramientas de Desarrollo

- Una deficiencia que tiene el desarrollo web es la falta de un IDE que facilite y agilice el desarrollo de nuestro código.
- Hay muchas alternativas tanto de pago como libres para desarrollar nuestro código pero una de las más populares y versátiles es Sublime Text

pue

Herramientas de Desarrollo: Instalar Sublime Text Editor

- Descargar de
- <http://www.sublimetext.com/2>

Home

Download

Buy

Blog

Forum

Support

Sublime Text 2

Download

The current version of Sublime Text 2 is 2.0.2.

[Sublime Text 3](#) is currently in beta, and contains many improvements over Sublime Text 2.

- [OS X](#) (OS X 10.6 or later is required)
- [Windows](#) - also available as a [portable version](#)
- [Windows 64 bit](#) - also available as a [portable version](#)
- [Linux 32 bit](#)
- [Linux 64 bit](#)

pue

Herramientas de Desarrollo:

Instalar Sublime Text Editor

- Descomprimir

```
# cp /home/jasanchez/Descargas/Sublime\ Text\ 2.0.2\ x64.tar.bz2 /usr/local/  
# cd /usr/local/  
# bunzip2 Sublime\ Text\ 2.0.2\ x64.tar.bz2  
# tar xf Sublime\ Text\ 2.0.2\ x64.tar
```

Herramientas de Desarrollo:

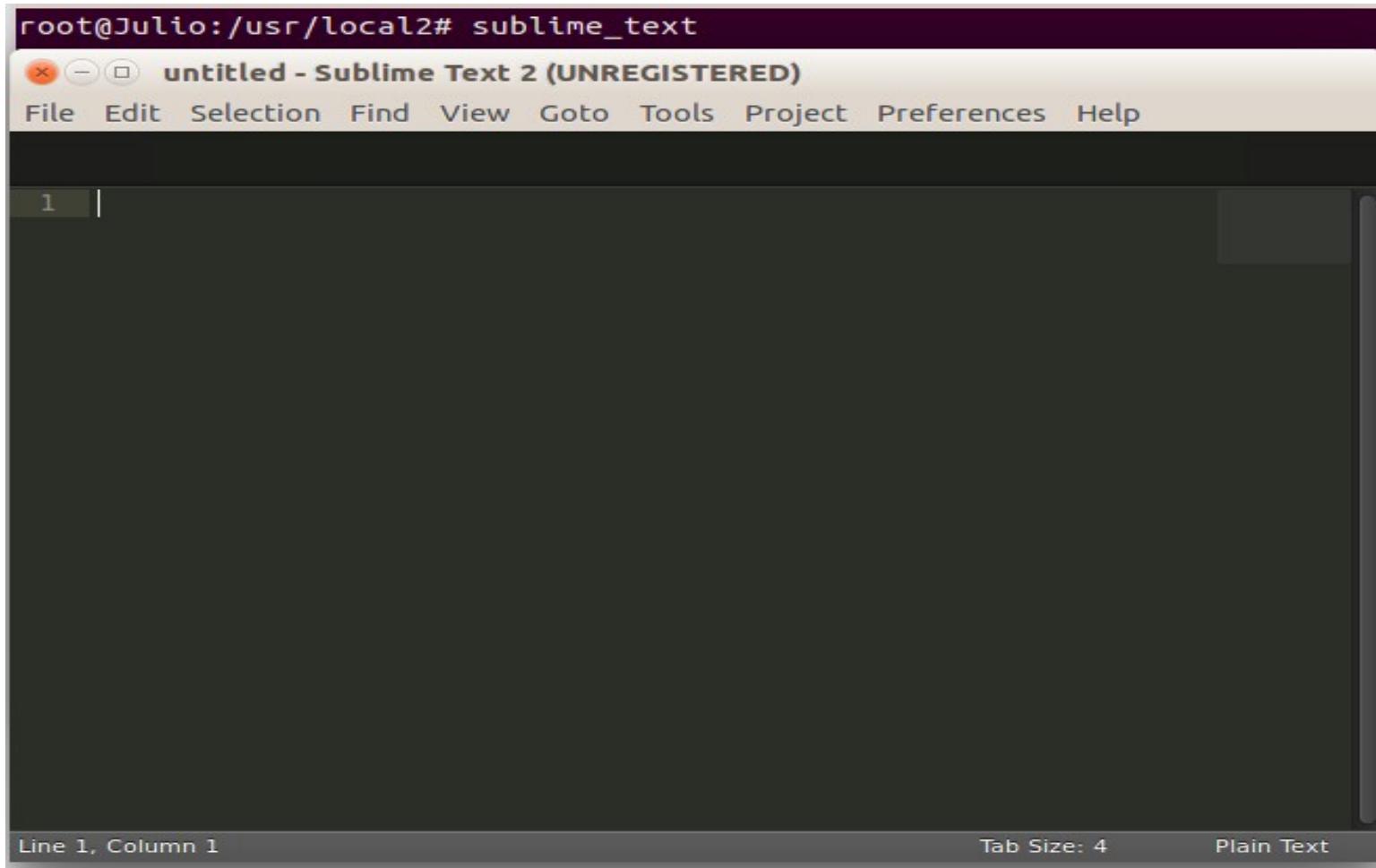
Instalar Sublime Text Editor

- Preparar para ejecutar

```
# chmod -R 777 /usr/local/Sublime\ Text\ 2
# echo 'PATH=$PATH:/usr/local/SublimeText2' >> /home/jasanchez/.bashrc
# echo 'export PATH' >> /home/jasanchez/.bashrc
```

Herramientas de Desarrollo: Ejecutar y Configurar Sublime Text

- Ejecutar



Herramientas de Desarrollo: Ejecutar y Configurar Sublime Text

- Instalar Package Control:
 - Nos permite gestionar fácilmente los plugins añadidos a sublime
 - <https://packagecontrol.io/installation#st2>
 - Seguir las instrucciones:
 - 1- Abrir consola de sublime
 - 2- Pegar el código de Phyton
 - 3- Reiniciar Sublime

Herramientas de Desarrollo: Ejecutar y Configurar Sublime Text

Package Control

INSTALLATION

Simple

The simplest method of installation is through the Sublime Text console. The console is accessed via the `ctrl+`` shortcut or the `View > Show Console` menu. Once open, paste the appropriate Python code for your version of Sublime Text into the console.

SUBLIME TEXT 3

SUBLIME TEXT 2

```
import urllib2,os,hashlib; h =
'2915d1851351e5ee549c20394736b442' +
'8bc59f460fa1548d1514676163dafc88'; pf = 'Package
Control.sublime-package'; ipp =
sublime.installed_packages_path(); os.makedirs( ipp ) if not
os.path.exists(ipp) else None; urllib2.install_opener(
urllib2.build_opener( urllib2.ProxyHandler() ) ); by =
urllib2.urlopen( 'http://packagecontrol.io/' + pf.replace(
'', '%20')).read(); dh = hashlib.sha256(by).hexdigest();
open( os.path.join( ipp, pf ), 'wb' ).write(by) if dh == h
else None; print('Error validating download (got %s instead
of %s), please try manual install' % (dh, h) if dh != h else
'Please restart Sublime Text to finish installation')
```

Search

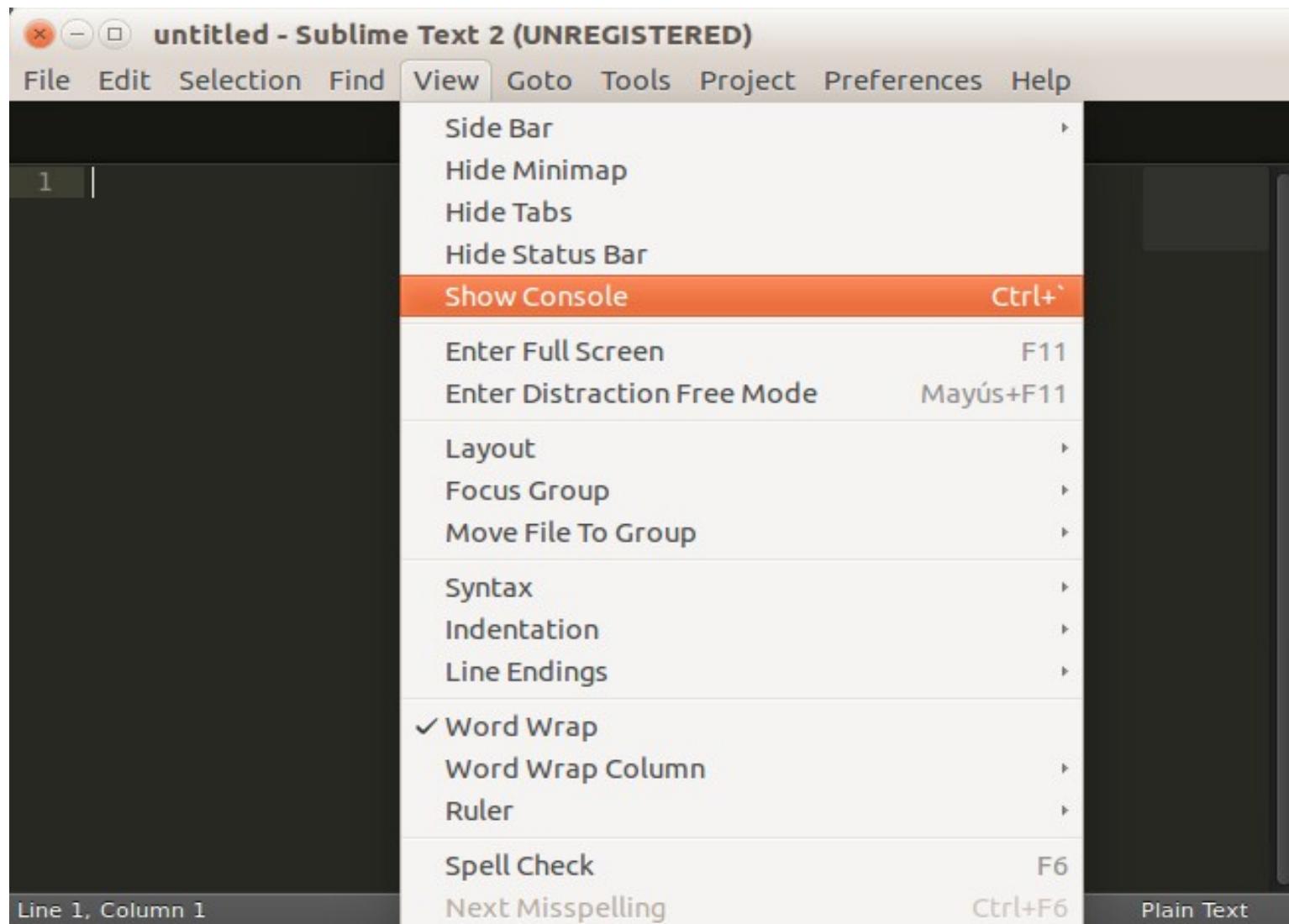
Manual

If for some
having a p
Package C

1. Click t
2. Brows
3. Downl
- Inst
4. Resta

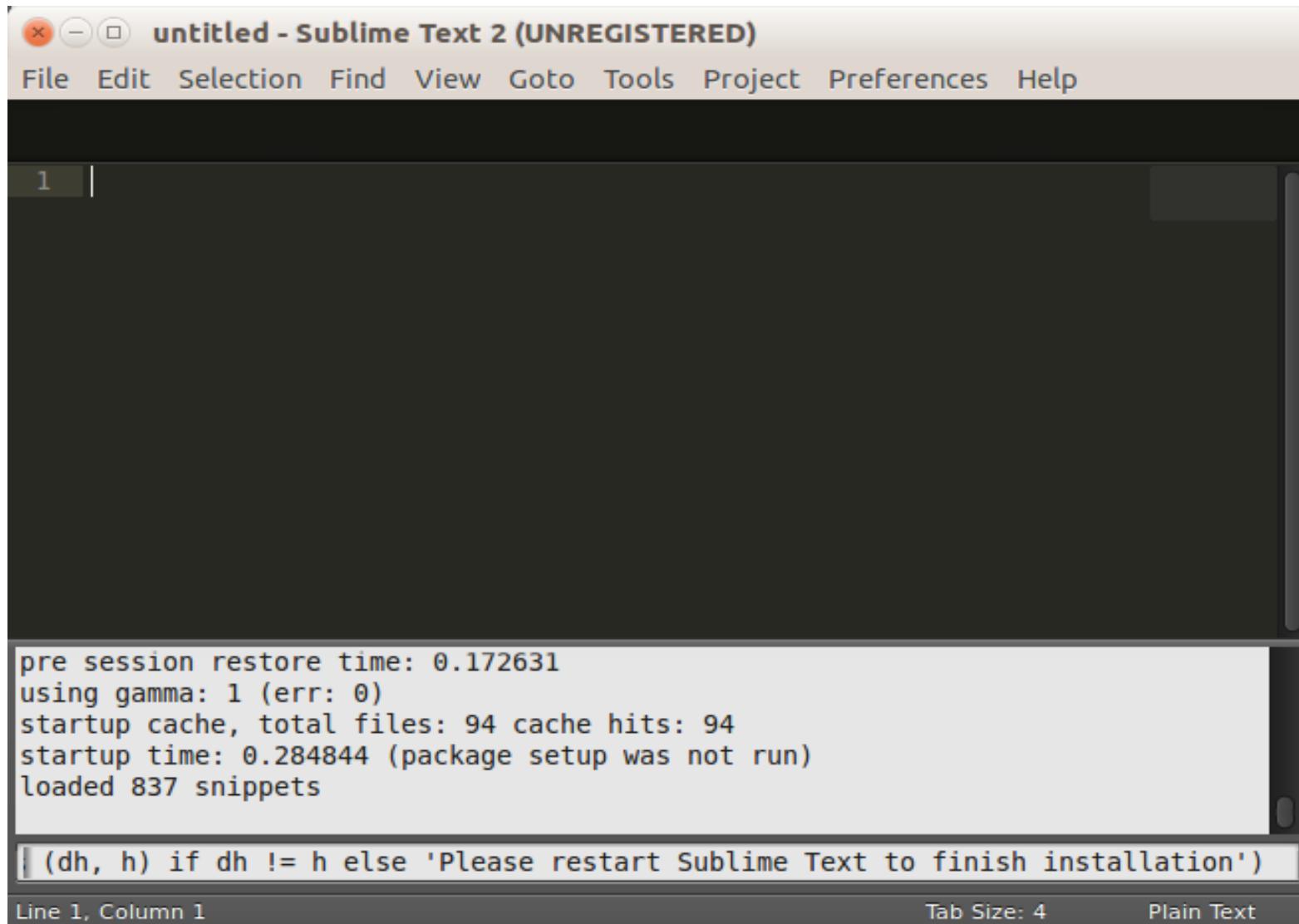
pue

Herramientas de Desarrollo: Ejecutar y Configurar Sublime Text



pue

Herramientas de Desarrollo: Ejecutar y Configurar Sublime Text

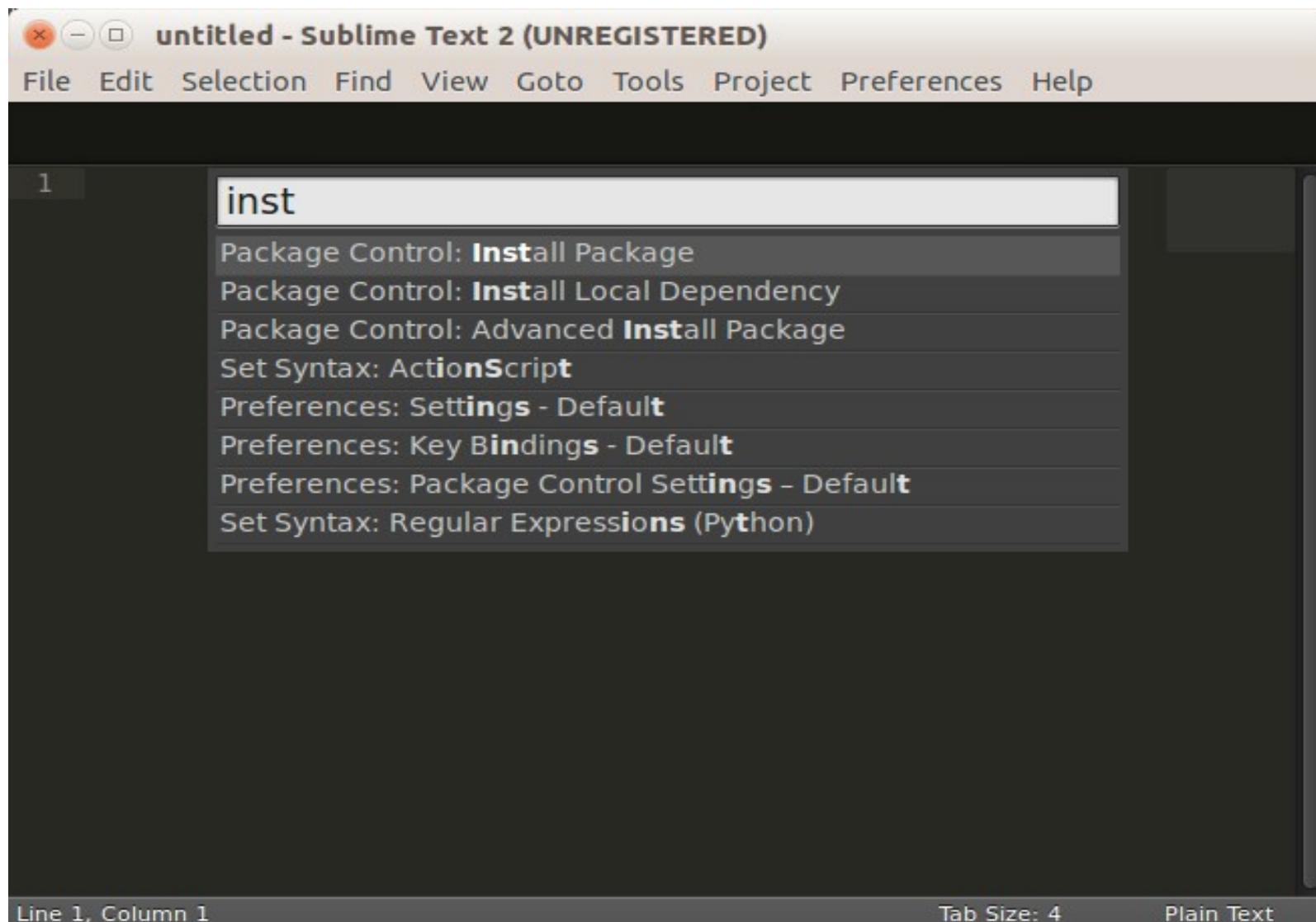


Herramientas de Desarrollo:

Instalar plugin Linter

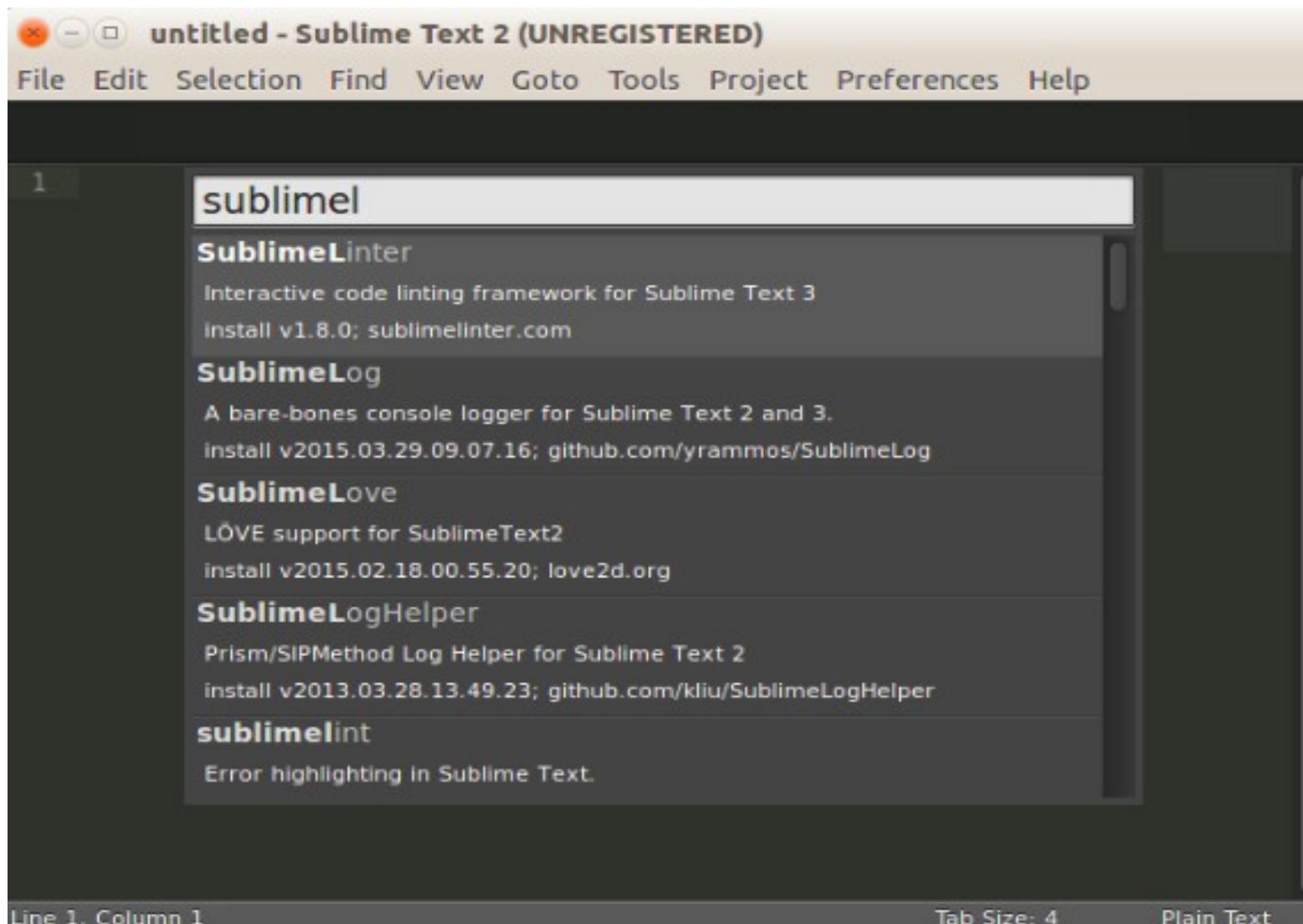
- Instalar linter:
 - Revisa nuestro código para alertarnos de los posibles fallos de sintaxis
 - Soporta multiples lenguajes: JavaScript,css, php,....
- Dos Opciones para abrir Package Control:
 - 1) preferences > Packet Control
 - 2) Ctrl + shift + p

Herramientas de Desarrollo: Instalar plugin Linter



pue

Herramientas de Desarrollo: Instalar plugin Linter

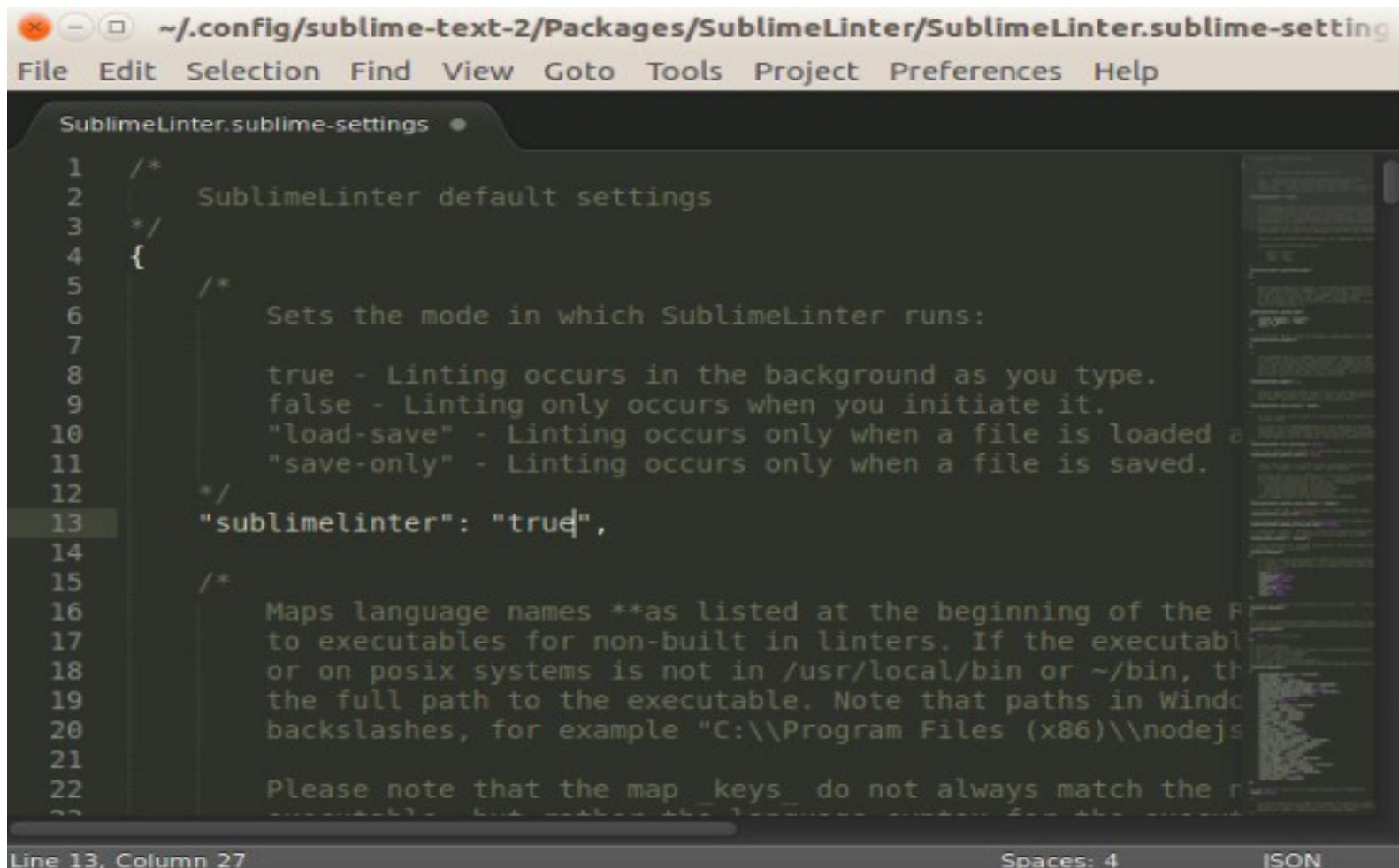


pue

Herramientas de Desarrollo:

Instalar plugin Linter

- Configurar Linter para que corrija la sintaxis según vamos escribiendo



A screenshot of the Sublime Text 2 interface. The title bar shows the path `~/.config/sublime-text-2/Packages/SublimeLinter/SublimeLinter.sublime-settings`. The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. A tab labeled "SublimeLinter.sublime-settings" is open. The code editor displays the following JSON configuration for SublimeLinter:

```
1  /*
2   * SublimeLinter default settings
3   */
4  {
5      /*
6       * Sets the mode in which SublimeLinter runs:
7       *
8       *     true - Linting occurs in the background as you type.
9       *     false - Linting only occurs when you initiate it.
10      *    "load-save" - Linting occurs only when a file is loaded or saved.
11      *    "save-only" - Linting occurs only when a file is saved.
12      */
13     "sublimelinter": "true",
14
15     /*
16      * Maps language names **as listed at the beginning of the file** to executables for non-built in linters. If the executable is not found in /usr/local/bin or ~/bin, then provide the full path to the executable. Note that paths in Windows use backslashes, for example "C:\\Program Files (x86)\\nodejs\\node".
17      *
18      * Please note that the map_keys do not always match the language names.
19      */
20
21
22
```

The status bar at the bottom indicates "Line 13, Column 27", "Spaces: 4", and "JSON".

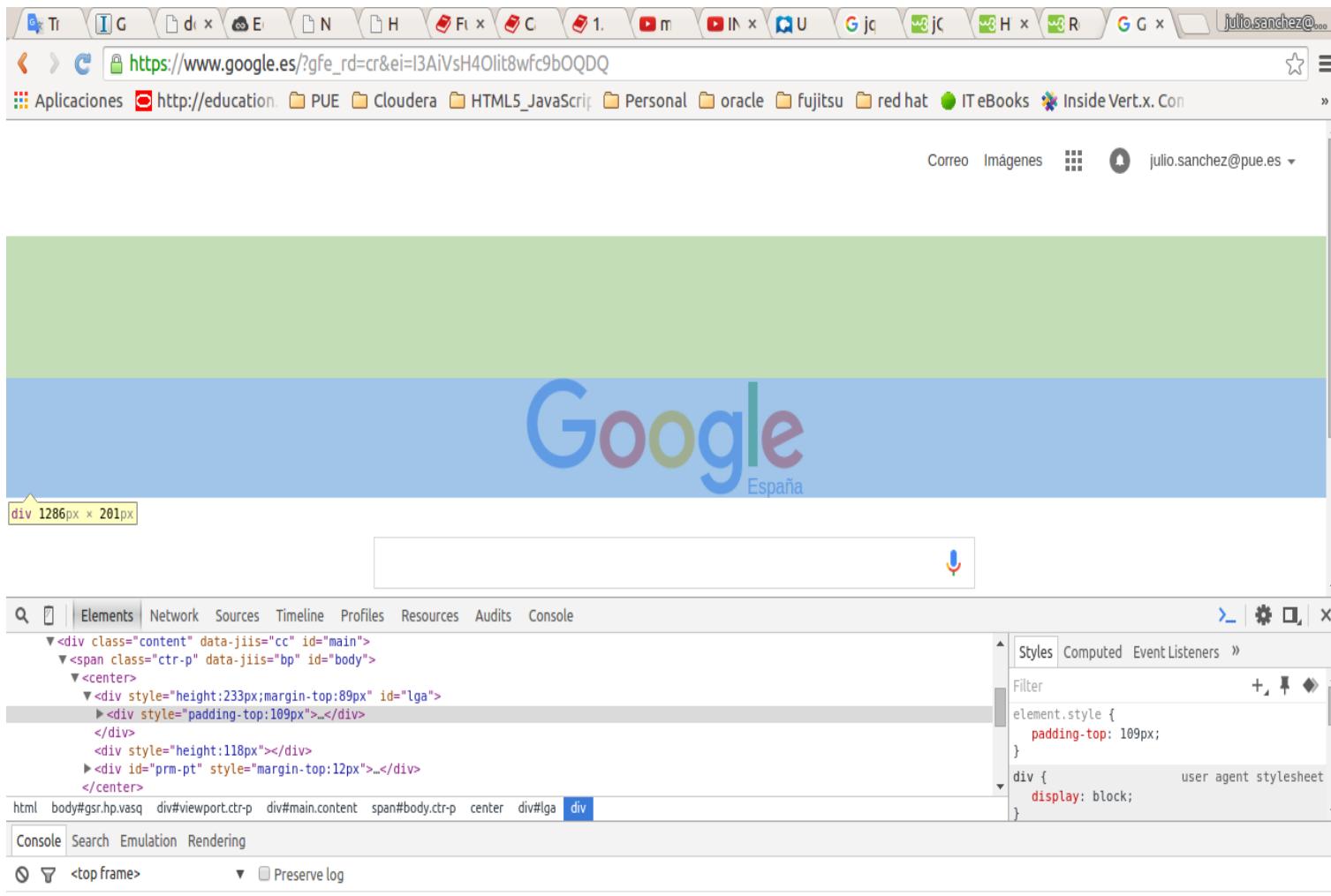
pue

Herramientas de Desarrollo: Depuración desde el navegador

- Podemos utilizar cualquier navegador.
- El más popular google chrome.



Herramientas de Desarrollo: Depuración desde el navegador



pue

Módulo4:

Estructura de HTML y CSS

pue

Estructura básica antes de HTML 5

Estructura básica

Lenguaje html

```
<html>
  <head>
    Información técnica
    para el navegador
  </head>
  <body>
    Contenido que aparecerá
    en la página web
  </body>
</html>
```

Estructura más avanzada antes de HTML 5

```
<div id="header">
```

```
<div id="nav">
```

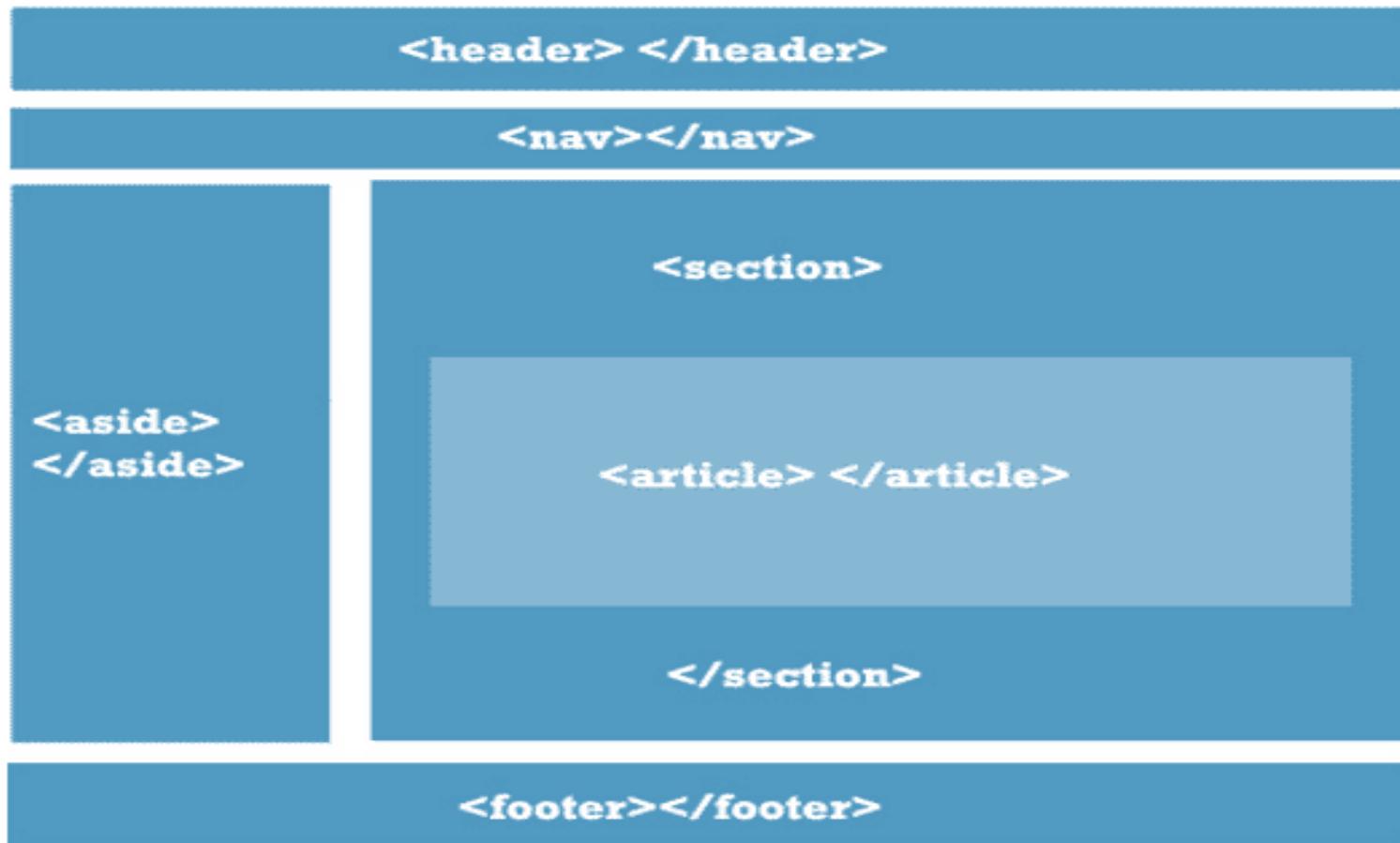
```
<div id="article">
```

```
<div id="section">
```

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

```
<div id="footer">
```

Estructura básica con HTML 5



pue

Programando en HTML 5 y CSS 3

Estructura básica

- Creación de la estructura básica con HTML 5:

```
1  <!DOCTYPE html>
2  <html en='es'>
3      <head> ...
8      </head>
9      <body>
10         <header> ...
16         </header>
17         <nav> ...
24         </nav>
25         <section>
26             <h3><p>sección principal 1</p></h3>
27             <article>
28                 <table> ...
33                 </table>
34             </article>
35             <article> ...
38             </article>
39             <article> ...
42             </article>
43         </section>
44         <aside> ...
47         </aside>
48         <footer> ...
51         </footer>
52     </body>
53 </html>
```

Programando en HTML 5 y CSS 3

Estructura básica

- **DOCTYPE**: recoge el conjunto de normas y restricciones que debe cumplir el documento.
Simplificado para HTML 5
- **HTML**: Contenedor de nuestro código con idioma en español.
- **Head**: Información del documento principalmente usada por navegadores y buscadores. No aparece como contenido de nuestra página.
- **Meta viewport**: Diseño adaptativo al tamaño de pantalla adaptativo
- **Meta keywords/description**: Info. Para posicionamiento web y buscadores
- **Title**: Info. A mostrar en la pestaña del navegador. Por los buscadores

pue

Programando en HTML 5 y CSS 3

Estructura básica paso a paso

```
1 <!DOCTYPE html>
2 <html en='es'>
3   <head>
4     <title>Info en pestaña del navegador</title>
5     <meta name="keywords" content="para, posicionamiento, en, buscadores"/>
6     <meta name="description" content="para posicionamiento en buscadores"/>
7     <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=5.
8       0">
9   </head>
```

Programando en HTML 5 y CSS 3

Estructura básica paso a paso

- **Body:** Contenedor de nuestro contenido para mostrar en la página
- **Header:** Cabecera, Su información puede ser usada por los buscadores.
- **Hgroup:** Agrupacion de etiquetas <H*> para su información la tengan en cuenta los buscadores.
- **H1-9:** Información con diferente tamaño de letra.

pue

Programando en HTML 5 y CSS 3

Estructura básica paso a paso

```
8   </head>
9   <body>
10  <header>
11    <hgroup>
12      Que los buscadores puedan usar las dos cabeceras
13      <h1>Cabecera principal del documento</h1>
14      <h2>Cabecera secundaria del documento</h2>
15    </hgroup>
16  </header>
```

Programando en HTML 5 y CSS 3

Estructura básica paso a paso

- **nav**: Contenedor de nuestra lista de navegación, que puede contener más información como cabecera, imágenes, ...
- **ul**: Lista de navegación
- **li**: Elementos de la lista
- **Image**: Imagen a mostrar
- **Figcaption**: Información a mostrar al pie de la foto

pue

Programando en HTML 5 y CSS 3

Estructura básica

```
16      </header>
17      <nav>
18          <h1><p>para navegar</p></h1>
19          <ul>
20              <li>elemento 1</li>
21              <li>elemento 2</li>
22              <li>elemento 3</li>
23          </ul>
24          
25          <figcaption>
26              Mi imagen de un cielo
27          </figcaption>
28      </nav>
```

Programando en HTML 5 y CSS 3

Estructura básica paso a paso

- **Section**: Contenedor para organizar nuestra información principal a mostrar en la página.
- **Article**: Nos permite organizar información más concreta de nuestra página
- **Table**: Mostrar datos en una tabla
- **Th**: Cabecera a mostrar de la tabla
- **Tr**: Las filas de la tabla
- **Td**: Los campos de la tabla

Programando en HTML 5 y CSS 3

Estructura básica paso a paso

```
28      </nav>
29      <section>
30          <h3><p>seccion principal 1</p></h3>
31          <article>
32              <table>
33                  <th>cabecera de la tabla</th>
34                  <tr><td>campol</td><td>campo2</td></tr>
35                  <tr><td>campol</td><td>campo2</td></tr>
36                  <tr><td>campol</td><td>campo2</td></tr>
37          </table>
38      </article>
39      <carticle>
40          <p>segundo articulo</p>
41      </article>
42      <article>
43          <p>tercer articulo</p>
44      </article>
45  </section>
```

Programando en HTML 5 y CSS 3

Estructura básica paso a paso

- **Aside:** Contenedor para organizar información externa o complementaria como enlaces a otras páginas o publicidad.
- Su colocación suele ser a la derecha o izquierda de la página.
- **Footer:** Contenedor para organizar información secundaria o complementaria (Copyright, trabaja con nosotros, quienes somos, ...)
- Su colocación será al final de la página.

Programando en HTML 5 y CSS 3

Estructura básica paso a paso

```
45      </section>
46      <aside>
47          <a href="http://www.google.es">enlace1</a>
48          <a href="http://www.w3c.es">enlace2</a>
49      </aside>
50      <footer>
51          <h4><p>copyright, trabaja con nosotros</p></h4>
52      </footer>
53  </body>
54 </html>
```

Programando en HTML 5 y CSS 3

Dando formato paso a paso

- La estructura visual es simple, colocando los elementos uno debajo de otro sin definir tamaños, tipos de letra, colores, etc...
- Para personalizar la estructura visual hacemos uso de CSS 3.
- Hay 3 formas de poder aplicar dicho formato:

pue

Programando en HTML 5 y CSS 3

Dando formato paso a paso: EN LINEA

- Sobre el elemento/etiqueta añadimos la propiedad style.
- **Inconveniente:** Muy difícil de mantener.
 - Si hay que modificar un estilo establecido a 50 elementos,
 - tendremos que modificar 50 líneas de nuestro código.

Programando en HTML 5 y CSS 3

Estructura básica paso a paso: EN LINEA

```
1  <!DOCTYPE html>
2  <html en='es'>
3      <head> ...
4      </head>
5      <body style="background: GhostWhite;">
6          <header style="border: 1px solid DarkOrange"> ...
7          </header>
8          <nav style="border: 1px solid DarkOrange">
9              <h1><p>para navegar</p></h1>
10             <ul> ...
11             </ul>
12             
13             <figcaption style="padding: 20px;">
14                 Mi imagen de un cielo
15             </figcaption>
16         </nav>
17         <section style="border: 1px solid DarkOrange">
18             <h3><p>seccion principal 1</p></h3>
19             <article style="border: 1px solid DarkOrange"> ...
20             </article>
21             <article style="border: 1px solid DarkOrange">
22                 <p>segundo articulo</p>
23             </article>
24             <article style="border: 1px solid DarkOrange">
25                 <p>tercer articulo</p>
26             </article>
27         </section>
28         <aside style="border: 1px solid DarkOrange"> ...
29         </aside>
30         <footer style="border: 1px solid DarkOrange">
31             <h4><p>copyright, trabaja con nosotros</p></h4>
32         </footer>
33     </body>
34 </html>
```

pue

Programando en HTML 5 y CSS 3

Dando formato paso a paso: EMBEBIDO

- Etiqueta `<style></style>` dentro de la etiqueta `<head>`
- **Inconveniente:** Difícil de mantener.
 - Si hay que modificar un estilo establecido a 50 páginas,
 - tendremos que modificar 50 ficheros.

Programando en HTML 5 y CSS 3

Estructura básica paso a paso: EMBEBIDO

```
1  <!DOCTYPE html>
2  <html en='es'>
3      <head>
4          <title>Info en pestaña del navegador</title>
5          <meta name="keywords" content="para, posicionamiento, en, buscadores"/>
6          <meta name="description" content="para posicionamiento en buscadores"/>
7          <meta name="viewport" content="width=device-width, initial-scale=1.0,
8              scale=5.0">
9          <style>
10             body {
11                 background: GhostWhite;
12             }
13
14             header,nav,section,article,aside,footer {
15                 border: 1px solid DarkOrange;
16             }
17         </style>
18     </head>
19     <body>
20         <header>...
21         </header>
22         <nav>...
23         </nav>
24         <section>...
25         </section>
26         <aside>
27             <a href="http://www.google.es">enlace1</a>
28             <a href="http://www.w3c.es">enlace2</a>
29         </aside>
30         <footer>
31             <h4><p>copyright, trabaja con nosotros</p></h4>
32         </footer>
33     </body>
34 </html>
```

pue

Programando en HTML 5 y CSS 3

Dando formato paso a paso: FICHERO

- Etiqueta `<link></link>` dentro de la etiqueta `<head>`
- apuntando a nuestro código
- **Ventajas:** Facilita el mantenimiento y estructura bien definida.
 - Puede ser compartido por varias páginas de nuestra aplicación y realizamos los cambios en un único punto.

Programando en HTML 5 y CSS 3

Estructura básica paso a paso: FICHERO

```
1 <!DOCTYPE html>
2 <html en='es'>
3     <head>
4         <title>Info en pestaña del navegador</title>
5         <meta name="keywords" content="para, posicionamiento, en, buscadores"/>
6         <meta name="description" content="para posicionamiento en buscadores"/>
7         <meta name="viewport" content="width=device-width, initial-scale=1.0,
scale=5.0">
8         <link rel="stylesheet" href="css3/4-estructuraFormato.css">
9     </head>
10    <body> ...
55    </body>
56 </html>
```

```
1 body {
2     background: GhostWhite;
3 }
4 header,nav,section,article,aside,footer {
5     border: 1px solid DarkOrange;
6 }
```

pue

Programando en HTML 5 y CSS 3

Dando formato: Organización

```
1  /* Selector global. todos los marcadores */
2  * {
3      margin: 0%;
4      padding: 0%;
5      /* padding y border se aplicara dentro de las medidas
6      width y height
7      asi dos elementos con mismo ancho y alto pero diferentes
8      separaciones seran del mismo tamaño
9      */
10     -moz-box-sizing: border-box;
11     -webkit-box-sizing: border-box;
12     box-sizing: border-box;
13 }
```

pue

Programando en HTML 5 y CSS 3

Dando formato: Organización

```
15  body {  
16      background: GhostWhite;  
17  }  
18  /* borde para identificar elementos */  
19  header,nav,section,article,aside,footer {  
20      border: 1px solid DarkOrange;  
21  }
```

```
23  /* Que su informacion este centrada en la pantalla */  
24  header,nav,footer{  
25      text-align: center;  
26  }  
27  /* como situo a los elementos de la lista */  
28  li {  
29      text-align: inherit;  
30      /*display: inline-block; /* unos al lado de otros, sobreescribe los siguientes */  
31      list-style: square; /* delante un cuadrado */  
32      list-style-position: inside; /* sino se dibuja fuera del border */  
33  }
```

Programando en HTML 5 y CSS 3

Dando formato: Organización

```
34 /* colocacion adaptativa */
35 body {
36     /* distribucion en cajas
37     body ocupa el 100% del dispositivo
38     */
39     display: inline-block;
40     width: 100%;
41 }
42 header,nav{
43     /* ocupan el 100% del dispositivo, una debajo de otra */
44     width: 100%;
45 }
46 /* entre la seccion y el aside me aseguro que no superen el 100%
47     en la estructura de cajas la seccion a la izquierda y el aside a la derecha
48 */
49 section {
50     width: 70%;
51     float: left;
52 }
53 aside {
54     width: 30%;
55     float: right;
56 }
```

Programando en HTML 5 y CSS 3

Dando formato: Organización

```
58 section > article {  
59     /* hay 3 articulos dentro de la seccion  
60      su ancho esta limitado al section  
61      entre los 3 me aseguro que no superen el 100% del section  
62      se van colocando en orden hacia la izq. */  
63     width: 33%;  
64     float: left;  
65 }  
66  
67 footer{  
68     /* evitar que el footer suba y descoloque la composicion de cajas */  
69     clear: both;  
70 }
```

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- Permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web.
- Utilizando solamente los cinco selectores básicos de CSS 2.1 (universal, de tipo, de clase, de id y descendente) es posible diseñar cualquier página web.
- Pero los selectores avanzados de CSS 2.1 permiten simplificar las reglas CSS y también el código HTML.
- Hoy en día, todos los navegadores más utilizados soportan los selectores avanzados de CSS 3.

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- **Selector universal:** Para seleccionar todos los elementos de la página.
`* { 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.
`h1 { color: #8A8E27; } →` Selecciona todos los `<h1>` de la página.

`h1,h2 { color: #8A8E27; } →` Selecciona todos los `<h1>` y `<h2>` de la página.

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- **Selector de clase:** Para seleccionar elementos que tenga el atributo class con un valor específico.

`p.destacado { color: red }` → Aquellos elementos de tipo `<p>` que dispongan de un atributo **class** con valor destacado

`.destacado { color: red }` → Cualquier elemento que dispongan de un atributo **class** con valor destacado.

`.error.destacado { color: blue; }` → Elementos de la página que dispongan de un atributo **class** con al menos los valores error y destacado

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- **Selector de ID:** Para seleccionar elementos que tenga el atributo id con un valor específico.

`p#destacado { color: red }` → Aquellos elementos de tipo `<p>` que dispongan de un atributo id con valor destacado

`#destacado { color: red }` → Cualquier elemento que dispongan de un atributo class con valor destacado.

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- **Selector de atributos (1/2)**: Para seleccionar elementos que tenga el atributo específico.

[name] → Aquellos elementos que dispongan de un atributo name.

p[name] → Aquellos elementos de tipo <p> que dispongan de un atributo name y su valor empieza por inicio.

p[name="tres"] → Aquellos elementos de tipo <p> que dispongan de un atributo name con valor tres.

p[name^="inicio"] → Aquellos elementos de tipo <p> que dispongan de un atributo name y su valor empieza por inicio.

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- **Selector de atributos (2/2):** Para seleccionar elementos que tenga el atributo específico.

`p[name^="inicio"]` → Aquellos elementos de tipo `<p>` que dispongan de un atributo name y su valor empieza por inicio

`p[name$="fin"]` → Aquellos elementos de tipo `<p>` que dispongan de un atributo name y su valor acabe con fin.

`p[name*="contiene"]` → Aquellos elementos de tipo `<p>` que dispongan de un atributo name cuyo valor contenga la cadena contiene.

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- **Selector descendente:** Seleccionar todos los elementos que esté dentro de otro, independientemente de lo profundo que se encuentre

`p a { color: red; }` → Se aplica a todos los elementos `<a>` que se encuentran dentro de elementos `<p>`.
- **Selector hijos:** Seleccionar todos los elementos que este dentro de otro, pero deben ser hijos directos (no nieto ni posteriores)

`p > a { color: red; }` → obliga a que los elementos `<a>` sean hijos directo de un elemento `<p>`.

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- **Selector Adyacente:** Seleccionar el primer elemento que sea hijo directo de otro.

`div.articulo2 + p` → Se aplica al primer elemento `<p>` que sea hijo directo de un elemento `<div>` con atributo `class` con valor `articulo2`.

`div.articulo2 ~ p` → Se aplica a todos los elementos `<p>` hermanos que sean hijos directos de un elemento `<div>` con atributo `class` con valor `articulo2`.

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- **pseudo-clase :first-child:** Para seleccionar al primer elemento que sea hijo de otro elemento.

`div p:first-child { color: red }` → Al primer elemento de tipo `<p>` que sea hijo de primer nivel de div

- **pseudo-clase :last-child:** Para seleccionar al último elemento que sea hijo de otro elemento.

`div p:last-child { color: red }` → Al último elemento de tipo `<p>` que sea hijo de primer nivel de div.

- **pseudo-clase :first-child(n):** Para seleccionar al elemento enesimo que sea hijo de otro elemento.

`div p:nth-child(2) { color: red }` → Al segundo elemento (empezando por 1) de tipo `<p>` que sea hijo de primer nivel de div

Programando en HTML 5 y CSS 3

Dando formato: Selectores

- **pseudo-clase :invalid:** Para seleccionar al todos los elementos cuyo formato introducido sea incorrecto.
`input:invalid{ border: 1px solid red}` → Todos los input que su valor sea incorrecto al no cumplir patrón o no tener valor
- **pseudo-clase :valid:** Para seleccionar al todos los elementos cuyo formato introducido sea correcto.
`input:valid{ border: 1px solid blue}` → Todos los input que su valor introducido sea correcto o no sea obligatorio tener valor.

Programando en HTML 5 y CSS 3

Ejercicio

- Crear una página HTML con CSS para que sea similar a:

CABEZERA		
sección principal	sección secundaria	enlace1
Parrafo 1	Parrafo 1	enlace2
Parrafo 2	Parrafo 2	enlace3
Parrafo 3	Parrafo 3	enlace4
Parrafo 1	Parrafo 1	
Parrafo 2	Parrafo 2	
Parrafo 3	Parrafo 3	
Parrafo 1	Parrafo 1	
Parrafo 2	Parrafo 2	
Parrafo 3	Parrafo 3	

pue

Programando en HTML 5 y CSS 3

Formularios.

- Elemento para crear aplicaciones web.
- Permite que los usuarios interactúen con ella.
- Un formulario es una sección del documento destinada a recoger información del usuario introduciendo datos que serán enviados al servidor para tratarlos.

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- Etiqueta `<form>` → que encierra todos los contenidos del formulario.
- Atributo `action = "url"` → Indica la URL **del servidor** que se encarga de procesar los datos del formulario.
- Atributo `method = "POST o GET"` - Método HTTP empleado al enviar el formulario.

MÉTODO	CONCEPTO	OBSERVACIONES
GET	GET lleva los datos de forma "visible" al cliente (navegador web). El medio de envío es la URL. Los datos los puede ver cualquiera.	Los datos son visibles por la URL, por ejemplo: www.aprenderaprogramar.com/ action.php?nombre=pedro&apellidos1=gomez
POST	POST consiste en datos "ocultos" (porque el cliente no los ve) enviados por un formulario cuyo método de envío es post. Es adecuado para formularios. Los datos no son visibles.	La ventaja de usar POST es que estos datos no son visibles al usuario de la web. En el caso de usar get, el propio usuario podría modificar la URL escribiendo diferentes parámetros a los reales en su navegador, dando lugar a que la información tratada no sea la prevista.

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- Etiqueta `<input>` → Contiene los controles para interactuar con el usuario.
- Atributo `type` → Permite elegir un control concreto. Sus valores pueden ser:
`Text` → muestra un cuadro de texto vacío en el que el usuario puede escribir cualquier texto

Nombre

pue

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- **textarea** → Cuadro de texto para escribir varias líneas.
 - Muy útil si necesitamos que el usuario deje comentarios, observaciones o anotaciones
 - Para definir el tamaño inicial podemos usar los atributos **rows** y **cols**



pue

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- **Password** → Cuadro de texto que cuando el usuario escribe no se ve en la pantalla.
- En su lugar, los navegadores ocultan el texto utilizando asteriscos o círculos.
- Ideal para ocultar datos privados como las contraseñas.



pue

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- **checkbox** → Permiten al usuario seleccionar y de-seleccionar opciones individualmente.
- Aunque se muestran varios juntos, cada uno de ellos es completamente independiente del resto.
- Para poder seleccionar opciones relacionadas pero no excluyentes.

Puestos de trabajo buscados

- Dirección
- Técnico
- Empleado

pue

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- **radio** → Similares a los checkbox, pero presentan una diferencia muy importante: son mutuamente excluyentes.
- Se utilizan cuando el usuario solamente puede escoger una opción , cada vez que se selecciona una opción, automáticamente se de-selecciona la otra opción que estaba seleccionaba.

Sexo
 Hombre
 Mujer

pue

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- **submit** → Para enviar al servidor los datos introducidos por el usuario
- **Reset** → Para borrar los datos rellenados por el usuario volviendo al punto de partida.
- **Button** → Para agregar otras funcionalidades a las 2 básicas anteriores (mostrar alerta, modificar el DOM, ...)

Guardar Cambios

pue

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- etiqueta `<select>` → Permite mostrar una lista desplegable con las diferentes opciones que podemos elegir.
- Por defecto solo se podrá elegir un elemento de la lista, pero podremos modificar ese comportamiento añadiendo el atributo `multiple`



pue

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- etiqueta `<label>` → No todos los controles disponen de la opción de establecer el texto que se muestra junto al control.
- En esos casos se incluye la etiqueta `<label>` para establecer el título de cada campo del formulario.

Programando en HTML 5 y CSS 3

Formularios. Componentes:

- etiqueta `<fieldset>` → Agrupa campos del formulario.
- etiqueta `<legend>` → Se usa junto a la etiqueta `fieldset`.
- Asigna un nombre a la Agrupación.

Datos personales

Nombre

Apellidos

DNI

Datos de conexión

Nombre de usuario

Contraseña

Repite la contraseña

pue

Programando en HTML 5 y CSS 3

Ejemplo Formulario básico.

```
1  <!DOCTYPE html>
2  <html lang='es'>
3      <body>
4          <form action="datos_enviados.html" method="post">
5              <label id="text">nombre usuario</label> <br/>
6              <input type="text" value="">
7              <br/><br/>
8              <label id="password">contrase&ntilde;a</label><br/>
9              <input type="password" value="">
10             <br/><br/>
11             <label id="checkbox">Gustos </label><br/>
12             <input type="checkbox" value="deporte"> deporte<br/>
13             <input type="checkbox" value="ciencia"> ciencia<br/>
14             <input type="checkbox" value="libros"> libros<br/>
15             <br/>
16             <label id="radio">Sexo</label> <br/>
17             <input type="radio" value="hombre"> <label>hombre</label>
18             <br/>
19             <input type="radio" value="mujer" checked="checked"> <label>mujer</label>
20             <br/><br/>
21             <label id="select">Estudios </label><br/>
22             <select name="elige una opcion">
23                 <option value="Sin Estudios">Volvo</option>
24                 <option value="Primarios">Saab</option>
25                 <option value="Universitarios">Mercedes</option>
26             </select>
27             <br/><br/>
28             <label id="textarea">Comentarios </label><br/>
29             <textarea rows="10" cols="20"> Dejar comentarios </textarea>
30             <br/><br/>
31             <input type="submit" value="submit">
32             <input type="reset" value="limpiar datos">
33
34     </form>
35 </body>
</html>
```

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- HTML5 nos permite definir patrones de validación para los diferentes campos de un formulario sin utilizar código de javascript.
- **Ventajas:**
 - Ahorrar recursos al servidor
 - Mejorar la experiencia del usuario.
- HTML5 ofrece diferentes métodos de validación **<input>**.
- Los más destacados son **required**, **pattern** y **type**.

Programando en HTML 5 y CSS 3

Validación de Formularios HTML5: REQUIRED

- Tal y como su nombre indica, este atributo se utiliza para decir al navegador los campos del formulario que son requeridos de forma obligatoria, es decir, que no pueden quedar vacíos.
- Basta con poner **required** o **required="required"**.
- Por ejemplo:

```
<input type="text" required>
```

Programando en HTML 5 y CSS 3

Validación de Formularios HTML5: PATTERN

- Nos permite definir patrones de validación personalizados sin prácticamente límite.
- El valor de este atributo ha de ser un formato con notación de expresión regular javascript.
- Por ejemplo, en España un número de teléfono fijo comienza por 9 o por 8 y un número móvil comienza por 6 o por 7 y en ambos casos le sigue un número de 8 dígitos.

```
<input type="text" pattern="^[9|8|7|6]\d{8}$">
```

Programando en HTML 5 y CSS 3

Validación de Formularios HTML5: TYPE

- Como ya hemos visto, con el atributo `<type>` indicamos al navegador el control que queremos utilizar sabiendo que a partir de HTML5 podemos utilizar una amplia gama con sus propias características de validación:

Programando en HTML 5 y CSS 3

Validación de Formularios HTML5: TYPE

- **tel** → Para números de teléfono.
- Tanto las letras y los números son validos porque la mayoría de los países tienen diferentes tipos variando longitudes.

pue

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- url → Para almacenar una URL.
- Muy útil si necesitamos guardar sitios web relacionados con el usuario.
- Debe empezar por http a menos que pongamos otro patrón con pattern



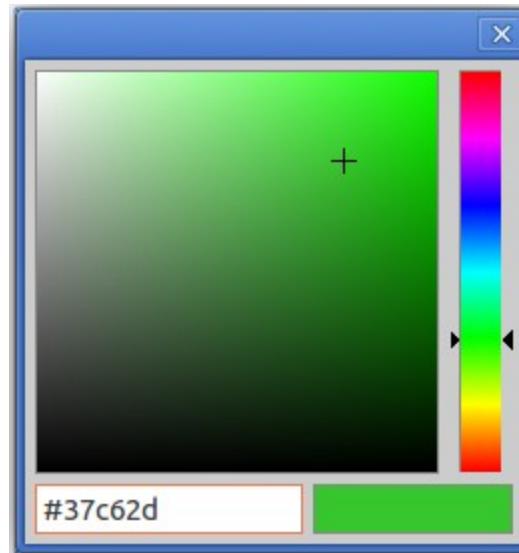
pue

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- **color** → Para permitir al usuario elegir un color y poder auto personalizar parte de la página.
- Aparece un cuadro para poder abrir la paleta y elegir un color.

color

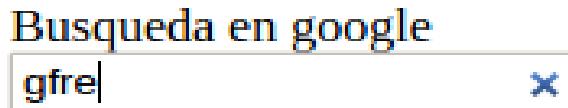


pue

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- **search** → Para campo de búsqueda
- Muy útil si necesitamos que se pueda buscar información en la web.
- Lleva una X incrustada para resetear el campo

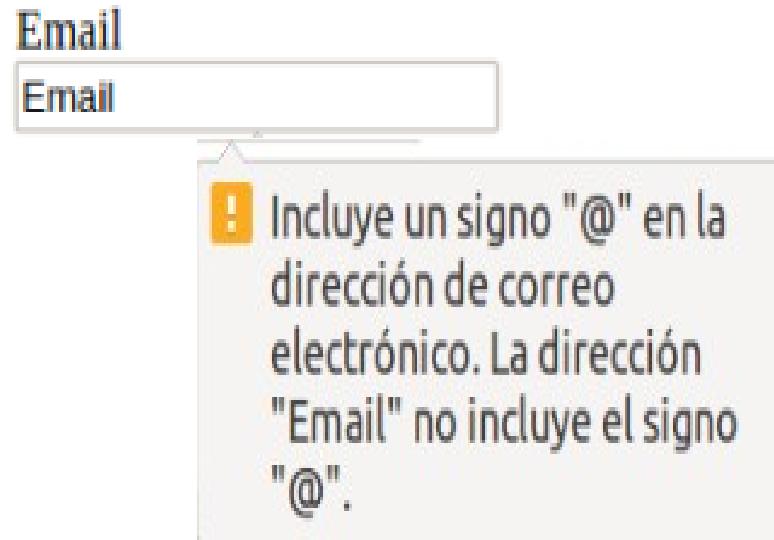


pue

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- **email** → Para almacenar una dirección de correo electrónico
- Verifica que hay @ en la dirección.
- En los dispositivos móviles, aparece el @ en la pantalla principal.



pue

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- **number** → Para validar un campo numérico.
- Con los atributos **min** y **max** podremos establecer un rango.
- Aparecen flechas en la parte derecha de la caja para subir/bajar el valor



pue

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- **range** → Para establecer un numérico.
- Con los atributos **min** y **max** podremos establecer un rango.
- En lugar de un cuadro de texto aparecerá una barra de desplazamiento horizontal.



pue

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- **date** → Para establecer la fecha.
- Para establecer la fecha tenemos varias formas:
 - Cuadro de texto donde escribir
 - Flechas para subir o bajar valor
 - Calendario



pue

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- **time** → Para establecer la hora.
- Disponemos de un cuadro de texto con formato de hora
- También están presentes las flechas en la parte derecha



pue

Programando en HTML 5 y CSS 3

Formularios. Nuevos componentes html5:

- **month** → Para establecer el mes.
- **Week** → Para establecer la semana.
- **datetime** → Para establecer fecha y hora simultaneamente.
- **Problema:** chrome, firefox e Internet Explorer no lo soportan

pue

Programando en HTML 5 y CSS 3

Formularios. Ejemplo con html5:

```
4      <form action="datos_enviados.html" method="post">
5          <fieldset>
6              <legend>Nuevos en HTML5</legend>
7              <label id="tel">Telefono Fijo </label><br/>
8              <input type='tel' pattern="^[89]\d{8}$" required>
9              <br/><br/>
10             <label id="url">URL personal</label><br/>
11             <input type="url" placeholder="tu url personal" required="required"><br/>
12             <br/>
13             <label id="search">Busqueda en google</label><br/>
14             <input type="search" name="buscar en google"><br/>
15             <br/>
16             <label id="email">Email</label> <br/>
17             <input type="email" value="Email">
18             <br/><br/>
19             <label id="number">Edad</label>
20             <input type="number" min="0" max="110">
21             <br/><br/>
22             <label id="range">Puntuacion</label>
23             <input type="range" min="0" max="10">
24             <br/><br/>
25             <label id="color">color</label>
26             <input type="color" min="0" max="10">
27             <br/><br/>
28             <label id="date">Fecha de Nacimiento</label>
29             <input type="date">
30             <br/><br/>
31             <label id="time">Hora de Nacimiento</label>
32             <input type="time">
33             <br/><br/>
34         </fieldset>
35         <br/>
36         <input type="submit" value="submit">
37         <input type="reset" value="limpiar datos">
38     </form>
```

pue

Programando en HTML 5 y CSS 3

Ejercicio

- Crear una página HTML con un formulario que pida los siguientes datos:
 - Nombre: debe empezar por una letra
 - Password: mínimo 8 caracteres
 - Edad: entre 0 y 110
 - fecha de nacimiento, email, sexo
- TODAS ELLAS OBLIGATORIAS
 - Teléfono fijo: empieza por 8 o 9
 - Teléfono móvil: empieza por 6 o 7
 - comentario
- Si hay datos incorrectos que su borde sea rojo

Programando en HTML 5 y CSS 3

Ejercicio

- Salida similar a esta:

— Datos conexión —

Nombre:

Contraseña:

— Datos personales —

Edad:

Fecha de Nacimiento:

Email:

Fecha de Nacimiento:
 hombre
 mujer

Teléfono fijo:

Teléfono móvil:

Comentarios:

pue

Módulo 5:

CSS 3 y especificidades

propietarias

pue

CSS3 - Estilos Avanzados

- El reajuste de elementos se realiza sin perder tamaño.
- Un párrafo o una imagen se adaptan a la nueva resolución, pero no se achica o se hace ilegible
- Re-adapta el menú horizontal y lo convertiría en menú vertical, con iconos y textos grandes, para que el usuario pueda ver y seleccionar el contenido correctamente.

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- HTML con la plantilla sencilla sobre la que operar:

```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <title>Estilos avanzados con CSS3</title>
5  </head>
6  <body>
7      <header id="principal1">
8          <span id="titulo1"> plantilla con estilos avanzados con css3 </span>
9      </header>
10
11     <section id="principal2">
12         <span id="titulo2"> plantilla con estilos avanzados con css3 </span>
13     </section>
14     <section id="principal3">
15         <span id="titulo3"> plantilla con estilos avanzados con css3 </span>
16     </section>
17     <section id="principal4">
18         <span id="titulo4"> plantilla con estilos avanzados con css3 </span>
19     </section>
20     </body>
21 </html>
```

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Aplicamos estilos de organización ya vistos

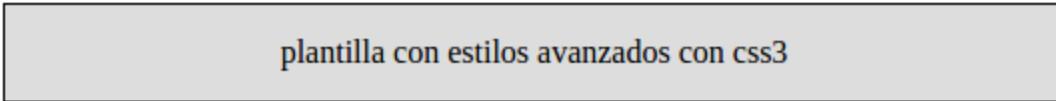
```
1  body{  
2      text-align: center;  
3  }  
4  section{  
5      display: inline-block;  
6      border: 1px solid black;  
7      padding: 2%;  
8      width: 30%;  
9  }  
10 #principal1{  
11     display: block;  
12     width: 500px;  
13     margin: 50px auto;  
14     padding: 15px;  
15     text-align: center;  
16     border: 1px solid black;  
17     background: #DDDDDD;  
18  }  
19 #principal2{  
20     text-align: center;  
21     background: #DDDDDD;  
22  }  
23 #principal3{  
24     text-align: center;  
25     background: #AAAAAA;  
26  }
```

pue

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- La salida sería:

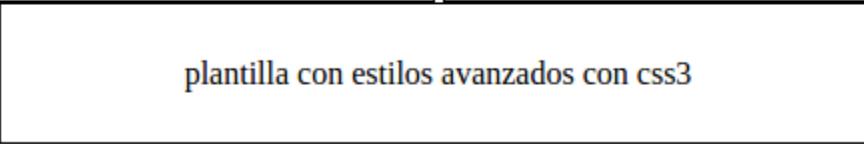


plantilla con estilos avanzados con css3



plantilla con estilos avanzados con css3

plantilla con estilos avanzados con css3



plantilla con estilos avanzados con css3

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Propiedad **border-radius** → Permite estilizar los bordes del elemento redondeándolo a mayor o menor tamaño.
- Ejemplos de Sintaxis:
border-radius: 25px;
→ Para todos los bordes el mismo tamaño.
border-radius: 1px 5px 10px 20px;
→ arriba-izq arriba-dcha abajo-izq abajo-dcha
border-radius: 10px 20px;
→ arriba-izq/abajo-dcha arriba-dcha/abajo-izq
border-radius: 40% / 25%;
→ elipses ancho/alto

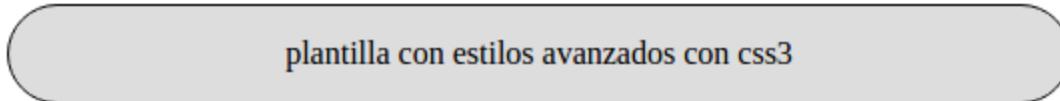
CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Con este código:

```
28 #principal1{  
29     border-radius: 25px;  
30 }  
31 #principal2{  
32     border-radius: 40% / 25%;  
33 }
```

- Este resultado:



plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

pue

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Función `rgba()` → Permite establecer:
 - 1.- color: Valores HEXADECIMAL al rojo verde y azul
 - 2.- opacidad: Valor entre 0 (transparente) al 1(opaco).
- Sintaxis:
`rgba(rojo,verde,azul,opacidad)`
- Ejemplos de Sintaxis:
`rgba(80,80,80,0.9)`

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Función **hsla()** → Permite establecer:
 - 1.- color: en grados. rojo (0°) a verde (120°) a azul (240°)
 - 2.- saturación: en por ciento. 0% gris – 100% color
 - 3.- luminosidad: en por ciento. 0% negro – 100% blanco
 - 4.- transparencia: de 0 (transparente) a 1 (opaco)
- Ejemplos de Sintaxis:
hsla(150,100%, 50%,0.5)

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Propiedad **box-shadow** → Permite establecer una sombra sobre el borde del elemento.
- Por defecto proyectada hacia fuera.
- Para proyectarla hacia dentro **inset**
- Sintaxis:
- **box-shadow: color sombra-horizontal sombra-vertical dureza-sombra [inset]**
- Ejemplos de Sintaxis:
`box-shadow: black 3px 5px 10px;`
`box-shadow: rgba(80,80,80,0.9) 3px 5px 10px;`
`box-shadow: black 3px 5px 10px inset;`

pue

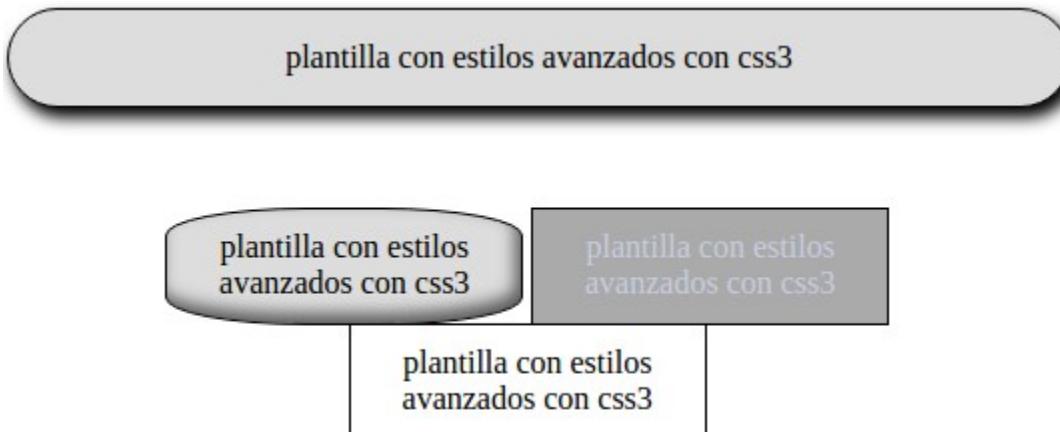
CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Con este código:

```
35 #principal1{  
36   box-shadow: black 3px 5px 10px;  
37 }  
38 #principal2{  
39   box-shadow: rgba(80,80,80,0.9) -3px -5px 10px inset;  
40 }  
41 #principal3{  
42   color: hsla(220, 75%, 90%, 0.7);  
43 }
```

- Este resultado:



plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

pue

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Propiedad **text-shadow** → Permite establecer una sombra sobre el texto del elemento.
- Sintaxis:
`text-shadow: color sombra-horizontal sombra-vertical
dureza-sombra`
- Ejemplos de Sintaxis:
`text-shadow: black 3px 5px 10px;`
`text-shadow: rgba(80,80,80,0.9) 3px 5px 10px;`

CSS3 - Estilos Avanzados

Ejemplo paso a paso

```
44 #principal{  
45   text-shadow: rgba(90,90,90,0.9) 3px 5px 10px;  
46 }  
47 #principal2{  
48   text-shadow: rgba(100,100,100,0.9) -3px -5px 10px;  
49 }
```

- Con este código:

- Este resultado:

plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

pue

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Función **linear-gradient()** → Permite establecer El degradado de colores.
- Sintaxis:
`linear-gradient(Inicio y fin del gradiente,Color de inicio, color de fin)`
- El inicio y fin del gradiente podemos especificar:
`top|bottom - left|right`
- Ejemplos de Sintaxis:
`background: -webkit-linear-gradient(top left,DarkRed ,DarkOrange);`
`background: -ms-linear-gradient(top,DarkRed ,DarkOrange);`

CSS3 - Estilos Avanzados

Ejemplo paso a paso

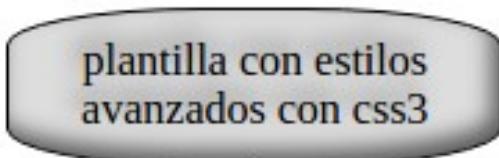
- Con este código:

```
50 #principal{  
51 |   background: -webkit-linear-gradient(top left,DarkRed ,DarkOrange );  
52 | }
```

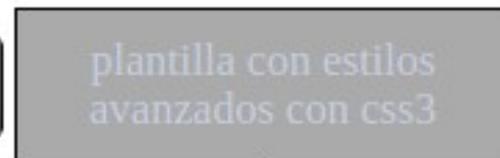
- Este resultado:



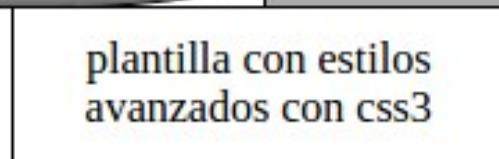
plantilla con estilos avanzados con css3



plantilla con estilos
avanzados con css3



plantilla con estilos
avanzados con css3



plantilla con estilos
avanzados con css3

pue

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Función **radial-gradient()** → forman diseños circulares, con una distribución radial uniforme, o degradados en elipse, con una distribución radial variable.
- Sintaxis:
`radial-gradient(Inicio del gradiente,circle|ellipse,Color de inicio y porcentaje, color de fin y porcentaje)`
- El inicio y fin del gradiente podemos especificar:
`center|top|bottom|left|right`
- Ejemplos de Sintaxis:
`background: -webkit-radial-gradient(center,ellipse,White 7%,DarkOrange 50%);`
`background: -o-radial-gradient(left,circle,White 7%,DarkOrange 50%);`

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Con este código:

```
54 #principal2{  
55   background: -webkit-radial-gradient(center,ellipse,White 7%,DarkOrange 50%);  
56 }
```

- Este resultado:



plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

pue

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Función **border-image()** → Permite utilizar una imagen para definir los bordes de los elementos.
- Esto hace que dibujarlos sea más simple y elimina la necesidad de utilizar muchas cajas en algunos casos.
- Sintaxis:
`Border-image(url(path_imagen),tamañoEnEsquinas,deformacionParteCentral)`
- En la deformación de la parte central los más comunes son:
`stretch|round`
- Ejemplos de Sintaxis:
`border-image: url("../fotos/iconos/flower.gif") 25% round;`

CSS3 - Estilos Avanzados

Ejemplo paso a paso

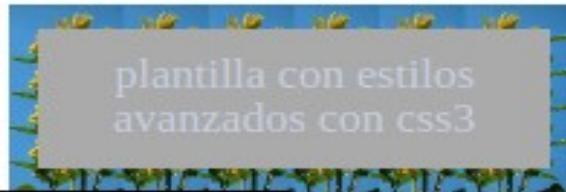
```
57 #principal3{  
58     border: 10px solid transparent;  
59     border-image: url("../fotos/iconos/flower.gif") 25% round;  
60 }
```

- Con este código:

- Este resultado:

plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3



plantilla con estilos
avanzados con css3

pue

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Propiedad **outline** → Permite un segundo borde al elemento.
- Propiedad **outline-offset** → Permite especificar la separación con el primer borde.
- Ejemplos de Sintaxis:
`outline: 2px dashed DarkBlue;`
`outline-offset: 1px;`

CSS3 - Estilos Avanzados

Ejemplo paso a paso

- Con este código:

```
61 #principal3{  
62   outline: 2px dashed DarkBlue;  
63   outline-offset: 1px;  
64 }
```

- Este resultado:

plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

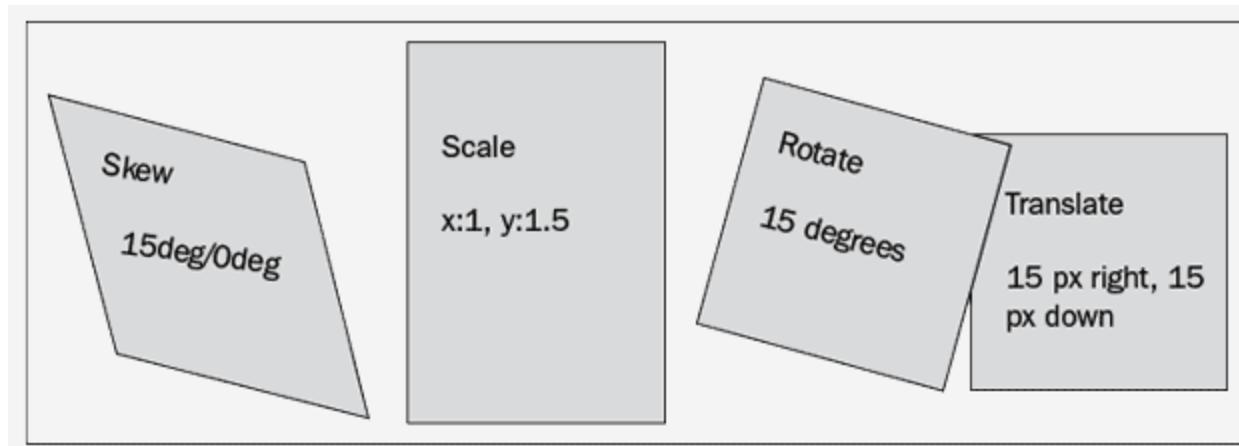
plantilla con estilos
avanzados con css3

pue

CSS3 - Efectos especiales y Animaciones

TRANSFORMACIONES

- las transformaciones consisten rotar, escalar y distorsionar y trasladar un elemento dentro de nuestra página.



pue

CSS3 - Efectos especiales y Animaciones

TRANSFORMACIONES: rotate

Rotate(grados):

- Permite rotar un elemento dándole un ángulo de giro en grados en dirección a las agujas del reloj.
- Con valores negativos en dirección contraria a las agujas del reloj.

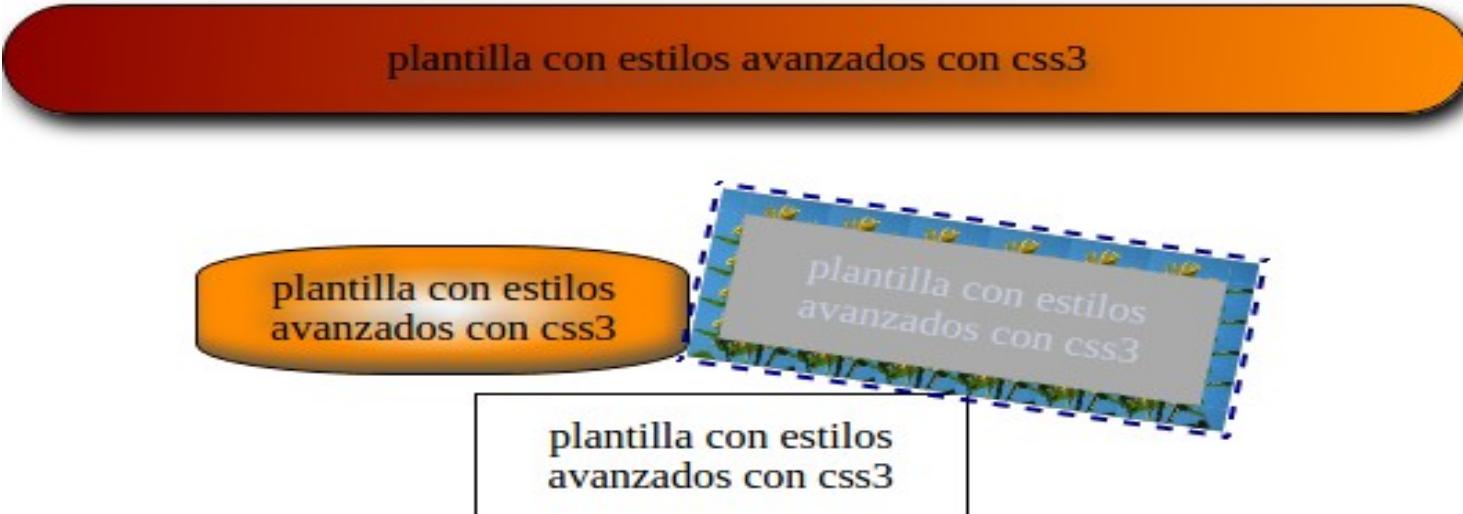
CSS3 - Estilos Avanzados

Ejemplo paso a paso

```
65 #principal3{  
66     -webkit-transform: rotate(10deg);  
67 }
```

- Con este código:

- Este resultado:



plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

pue

CSS3 - Efectos especiales y Animaciones

TRANSFORMACIONES: scale

Scale(ejeX,ejeY):

- Permite escalar un elemento, toma valores positivos y negativos y se le pueden poner decimales.
- Con valores negativos en dirección contraria a las agujas del reloj.

CSS3 - Estilos Avanzados

TRANSFORMACIONES: scale

- Con este código:

```
65 #principal3{  
66   -webkit-transform: scale(-0.9,-0.5);  
67 }
```

- Este resultado:

plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3



plantilla con estilos
avanzados con css3

pue

CSS3 - Efectos especiales y Animaciones

TRANSFORMACIONES: translate

Translate(ejeXfin,ejeYfin):

- Permite trasladar un elemento a la vez en el eje de las X y de las Y, dándole las coordenadas finales.
- Con valores negativos a izquierda y arriba.

CSS3 - Estilos Avanzados

TRANSFORMACIONES: translate

- Con este código:

```
65 #principal3{  
66   -webkit-transform: translate(-17px, 15px);  
67 }
```

- Este resultado:

plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

pue

CSS3 - Efectos especiales y Animaciones

TRANSFORMACIONES: skew

skew(grados):

- Permite cambiar la diferencia entre la parte superior y la inferior
- Con valores negativos la parte de abajo mas a la derecha

CSS3 - Estilos Avanzados

TRANSFORMACIONES: skew

- Con este código:

```
65 #principal3{  
66   -webkit-transform: skew(-10deg);  
67 }
```

- Este resultado:

plantilla con estilos avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

plantilla con estilos
avanzados con css3

pue

CSS3 - Efectos especiales y Animaciones

TRANSFORMACIONES: Múltiples

- Para aplicar varias transformaciones sobre el mismo elemento podemos especificarlas todas en la misma propiedad, separadas por espacio.

pue

CSS3 - Estilos Avanzados

TRANSFORMACIONES: Múltiples

- Con este código:

```
65 #principal3{  
66   -webkit-transform: scale(-0.9,-0.5) rotate(10deg) skew(-10deg) translate(-7px,5px);  
67 }
```

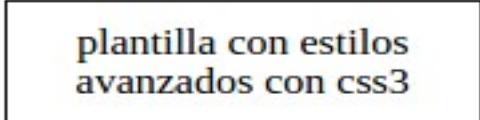
- Este resultado:



plantilla con estilos avanzados con css3



plantilla con estilos
avanzados con css3



plantilla con estilos
avanzados con css3

pue

CSS3 - Efectos especiales y Animaciones

TRANSICIONES

- Cuando cambiamos el aspecto de los elementos con CSS, es un cambio estático, es decir, sucede de manera instantánea
- Para aplicar cambios en un elemento de manera gradual podemos usar las transiciones
- Para activar una transición es necesario que se detecte un evento.
 - `:hover` -> cuando el cursor del ratón se sitúa sobre el elemento (equivale al evento `onmouseover`).
 - `:active` -> cuando se pulsa el ratón sobre el elemento (equivale al evento `onclick`).
 - `:focus` -> cuando el elemento recibe el foco, por ejemplo, al situar el cursor en el campo de un formulario (equivale al evento `onfocus`).

CSS3 - Efectos especiales y Animaciones

TRANSICIONES

- Las transiciones de CSS3 pueden tomar estos 4 valores:

transition-property:

- Que cambio haremos en el elemento
- Por ejemplo, cambiar el color o aplicar una transformación

pue

CSS3 - Efectos especiales y Animaciones

TRANSICIONES

- Las transiciones de CSS3 pueden tomar estos 4 valores:

transition-duration:

→ Especificamos el tiempo que va a tardar la transición expresado en segundos

pue

CSS3 - Efectos especiales y Animaciones

TRANSICIONES

- Las transiciones de CSS3 pueden tomar estos 4 valores:

transition-delay:

→ Especificamos un tiempo de espera antes de empezar a realizar la transición.

pue

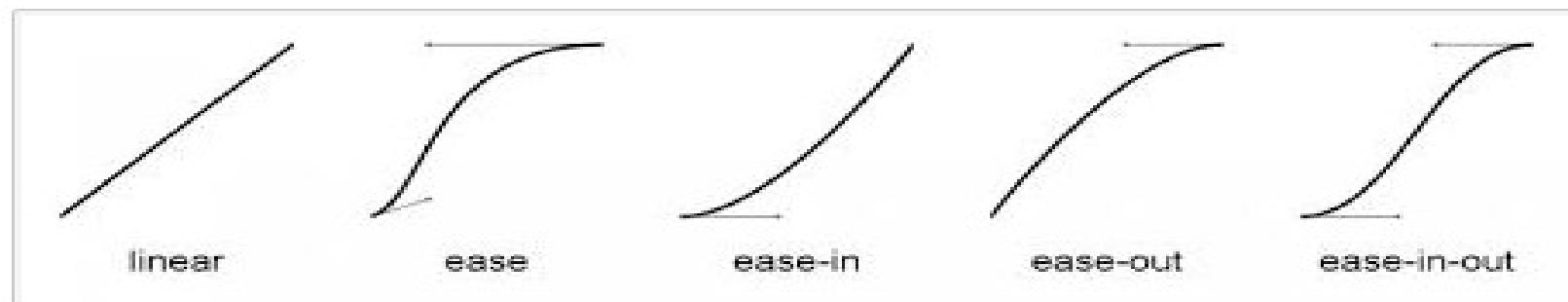
CSS3 - Efectos especiales y Animaciones

TRANSICIONES

- Las transiciones de CSS3 pueden tomar estos 4 valores:

transition-timing-function:

- Especifica la velocidad en que se desarrollará la transición, utilizando las llamadas curvas de Bezier.
- Así, podemos programar la transición para que empiece a una velocidad, luego frene y después se acelere
- Los posibles valores son:



pue

CSS3 - Estilos Avanzados

TRANSICIONES

- Con este código:

```
95 #principal4{  
96     background-color: #900;  
97     -webkit-transition-property: background-color, -webkit-transform;  
98     -webkit-transition-duration: 10s;  
99     -webkit-transition-delay: 1s;  
100    -webkit-transition-timing-function: ease-in-out;  
101 }  
102 #principal4:hover{  
103     background-color: #090;  
104     -webkit-transform: scale(-0.9,-0.5) rotate(10deg) skew(-10deg) translate(-7px,5px);  
105 }
```

- Resultado:
 - Partimos de rojo y elemento estático y gradualmente pasamos a verde y “transitamos”.
 - Ejecutar en navegador!!!

CSS3 - Efectos especiales y Animaciones

FUENTES

- Uno de los mayores inconvenientes con los que se encuentra un diseñador web es la utilización de tipos de letra especiales.
- Hasta hace poco había que limitarse a las denominadas “fuentes seguras” → Casi con total seguridad el usuario tuviera instaladas en su ordenador.
- De lo contrario, los textos se mostrarían con otras tipografías, distintas a las que habíamos elegido.

CSS3 - Efectos especiales y Animaciones

FUENTES

- La regla de CSS3 `@font-face` soluciona este problema mediante la incrustación de fuentes.
- El usuario se descargará las fuentes utilizadas en la página web de manera similar a lo que sucede con las imágenes.
- Podemos crear nuestras propias fuentes o descargarlas de alguna página web
- <http://www.fontsquirrel.com>

CSS3 - Efectos especiales y Animaciones

FUENTES: paso a paso

- Elegir y descargar la familia elegida de:
- http://www.fontsquirrel.com/fonts/list/find_fonts

The screenshot shows the FontSquirrel homepage with the navigation bar: FIND FONTS, HOT, RECENT, ALMOST FREE, SHOP, GENERATOR, MATCHERATOR. Below the navigation is a large heading 'FIND COMMERCIAL FREE FONTS'. A navigation bar at the bottom shows '1942 report' by 'Johan Holmdahl' with 1 style available. There are download links for 'DOWNLOAD TTF' and 'DOWNLOAD OTF'. The font '1942 report' is displayed in various styles (normal, italic, bold) along with its character set. The word 'pue' is visible in the bottom right corner.

CSS3 - Efectos especiales y Animaciones

FUENTES: paso a paso

- Preparar las fuentes en nuestro Sistema de Ficheros

```
jasanchez@Julio:~$ mkdir Documentos/MisCursos/HTML5_CSS_JavaScript-moviles/curso/fuentes/HOLMDAHL
jasanchez@Julio:~$ mv Descargas/1942-report.zip Documentos/MisCursos/HTML5_CSS_JavaScript-moviles/curso/fuentes/HOLMDAHL
jasanchez@Julio:~$ cd Documentos/MisCursos/HTML5_CSS_JavaScript-moviles/curso/fuentes/HOLMDAHL
jasanchez@Julio:~/Documentos/MisCursos/HTML5_CSS_JavaScript-moviles/curso/fuentes/HOLMDAHL$ unzip 1942-report.zip
Archive: 1942-report.zip
  inflating: 1942.ttf
  inflating: Freeware License.txt
```

CSS3 - Efectos especiales y Animaciones

FUENTES: paso a paso

- Subir fichero **ttf** a:
- <http://www.fontsquirrel.com/tools/webfont-generator>
- Prepara las fuentes y todos sus ficheros para usarlos.

The screenshot shows the Fontsquirrel WebFont Generator interface. At the top, there's a navigation bar with links like 'FIND FONTS', 'HOT', 'RECENT', 'ALMOST FREE', 'SHOP', 'GENERATOR', and 'M...'. Below the navigation is a large title 'WEBFONT GENERATOR'. A usage instruction says: 'Usage: Click the "Upload Fonts" button, check the agreement and download your fonts. If you need more fine control, choose the Expert option.' A purple button labeled 'UPLOAD FONTS' with an upward arrow is prominent. Below it, a message says 'You currently have no fonts uploaded.' There are three radio button options: 'BASIC' (selected), 'OPTIMAL', and 'EXPERT...'. Under 'Agreement:', a checkbox is checked with the text 'Yes, the fonts I'm uploading are legally eligible for web embedding.' A file selection dialog box is overlaid on the page, titled 'Abrir archivos'. It shows a list of files, with 'A 1942.ttf' selected. The dialog includes buttons for 'Cancelar' (Cancel) and 'Abrir' (Open). The word 'pue' is visible in the bottom right corner of the slide.

CSS3 - Efectos especiales y Animaciones

FUENTES: paso a paso

- Descargar las fuentes preparadas por FontSquirrel

UPLOAD FONTS ↑

1942 report	ttf	95 glyphs	51 KB	
<input type="radio"/> BASIC Straight conversion with minimal processing.	<input checked="" type="radio"/> OPTIMAL Recommended settings for performance and speed.	<input type="radio"/> EXPERT... You decide how best to optimize your fonts.		
Agreement:	<input checked="" type="checkbox"/> Yes, the fonts I'm uploading are legally eligible for web embedding. Font Squirrel offers this service in good faith. Please honor the EULAs of your fonts.			
DOWNLOAD YOUR KIT				

pue

CSS3 - Efectos especiales y Animaciones

FUENTES: paso a paso

- Preparar las fuentes en nuestro Sistema de Ficheros

```
jasanchez@Julio:~$ mv Descargas/webfontkit-20151023-074628.zip Documentos/MisCursos/HTML5_CSS_JavaScript-moviles/curso/fuentes/HOLMDAHL
jasanchez@Julio:~$ cd Documentos/MisCursos/HTML5_CSS_JavaScript-moviles/curso/fuentes/HOLMDAHL
jasanchez@Julio:~/Documentos/MisCursos/HTML5_CSS_JavaScript-moviles/curso/fuentes/HOLMDAHL$ unzip webfontkit-20151023-074628.zip
Archive: webfontkit-20151023-074628.zip
  inflating: 1942-webfont.eot
  inflating: 1942-webfont.svg
  inflating: 1942-webfont.ttf
  inflating: 1942-webfont.woff
  inflating: 1942-webfont.woff2
  inflating: 1942-demo.html
  inflating: stylesheet.css
  inflating: generator_config.txt
  inflating: specimen_files/specimen_stylesheet.css
  inflating: specimen_files/grid_12-825-55-15.css
```

CSS3 - Efectos especiales y Animaciones

FUENTES: paso a paso

- Ya podemos codificar nuestras nuevas fuentes, copiando a nuestro fichero css el contenido que hay en el fichero stylesheet.css que acabamos de descargar.

CSS3 - Efectos especiales y Animaciones

FUENTES: paso a paso

- Fichero plantilla de html:

```
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <title>Estilos avanzados con CSS3</title>
5          <link rel="stylesheet" href="css3/11-fuentes.css">
6      </head>
7      <body>
8          <header id="principal1">
9              <span id="titulo1"> plantilla con estilos avanzados con css3 </span>
10         </header>
11         <section id="principal2">
12             <span id="titulo2"> plantilla con estilos avanzados con css3 </span>
13         </section>
14         <section id="principal3">
15             <span id="titulo3"> plantilla con estilos avanzados con css3 </span>
16         </section>
17     </body>
18 </html>
```

CSS3 - Efectos especiales y Animaciones

FUENTES: paso a paso

- Fichero CSS:

```
@font-face { /* Copiar/Pegar del fichero stylesheet.css que nos genera www.fontsquirrel.com */
    font-family: 'kaushan_scriptregular';
    src: url('../fuentes/IMPALLARI/kaushanscript-regular-webfont.eot');
    src: url('../fuentes/IMPALLARI/kaushanscript-regular-webfont.eot?#iefix') format('embedded-opentype'),
         url('../fuentes/IMPALLARI/kaushanscript-regular-webfont.woff2') format('woff2'),
         url('../fuentes/IMPALLARI/kaushanscript-regular-webfont.woff') format('woff'),
         url('../fuentes/IMPALLARI/kaushanscript-regular-webfont.ttf') format('truetype'),
         url('../fuentes/IMPALLARI/kaushanscript-regular-webfont.svg#kaushan_scriptregular') format('svg');
    font-weight: normal;
    font-style: normal;

}
span#titulol1{
    font: bold 36px "kaushan_scriptregular";
}
```

CSS3 - Efectos especiales y Animaciones

FUENTES: paso a paso

- Salida:

plantilla con estilos avanzados con css3

plantilla con estilos avanzados con css3

plantilla con estilos avanzados con css3

pue

Modulo 6: **JavaScript**

pue

Introducción a JavaScript:

- JavaScript proporciona funcionalidad y dinamismo a la página a través de los eventos y acciones de usuario
- Transforma el código JavaScript en código máquina para superar antiguas limitaciones.
- Antes había que usar otros elementos como java o flash pero en las nuevas implementaciones podemos hacer uso de librerías como jQuery quitando la necesidad de otros lenguajes.

Normas básicas

- No se tienen en cuenta los espacios en blanco y las nuevas líneas
- Se distinguen las mayúsculas y minúsculas
- No se define el tipo de las variables. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del Script.
- No es necesario terminar cada sentencia con el carácter de punto y coma aunque como una buena práctica.
- Se pueden incluir comentarios
 - // de una linea
 - /*Multiples linea */

pue

Programación básica:

Variables

- Una variable es un elemento que se emplea para almacenar y hacer referencia a otro valor.
- Las variables en JavaScript se crean mediante la palabra reservada **var** que solamente se debe indicar al definir por primera vez, lo que se denomina **declarar** una variable.
- Cuando se utilizan las variables en el resto de instrucciones del Script, solamente es necesario indicar su nombre.
- Si cuando se declara se le asigna también un valor, se dice que la variable ha sido **inicializada**.

Programación básica:

Tipos de variables

- **Numéricas:** para almacenar valores numéricos enteros o decimales
- **Cadenas:** Se utilizan para almacenar caracteres, palabras o frases.
 - Para asignar el valor a la variable, se encierra el valor entre comillas dobles o simples
- **Booleanos:** tipo especial de valor que solamente puede tomar dos valores: true (verdadero) o false (falso).
- **Arrays:** colección de variables, que pueden ser todas del mismo tipo o cada una de un tipo diferente.
 - La variable tiene un identificador único y los diferentes valores se diferencian por un índice comenzando en 0.

Programación básica:

Tipos de variables. Ejemplos

```
var iva = 16;          // variable tipo entero  
var total = 234.65;   // variable tipo decimal  
var mensaje = "Hola Mundo!!";  
var conlva = false;  
  
var dias = ["Lunes", "Martes", "Miércoles", "Jueves",  
"Viernes", "Sábado", "Domingo"];  
var hoy = dias[0]; // hoy= "Lunes"
```

Programación básica:

Operadores

Operador	Uso	Descripción
+	op1 + op2	Suma op1 y op2 (*)
-	op1 - op2	Resta op2 de op1
*	op1 * op2	Multiplica op1 y op2
/	op1 / op2	Divide op1 por op2
%	op1 % op2	Obtiene el resto de dividir op1 por op2
++	op ++	Incrementa op en 1; evalúa el valor antes de incrementar
++	++ op	Incrementa op en 1; evalúa el valor después de incrementar
--	op --	Decrementa op en 1; evalúa el valor antes de decrementar
--	-- op	Decrementa op en 1; evalúa el valor después de decrementar

Operadores Relacionales	
&&	And
	Or
!	Not

Programación básica: Operadores

Ejemplo	Nombre	Resultado
<code>\$a == \$b</code>	Igual	TRUE si <code>\$a</code> es igual a <code>\$b</code> después de la manipulación de tipos.
<code>\$a === \$b</code>	Idéntico	TRUE si <code>\$a</code> es igual a <code>\$b</code> , y son del mismo tipo.
<code>\$a != \$b</code>	Diferente	TRUE si <code>\$a</code> no es igual a <code>\$b</code> después de la manipulación de tipos.
<code>\$a <> \$b</code>	Diferente	TRUE si <code>\$a</code> no es igual a <code>\$b</code> después de la manipulación de tipos.
<code>\$a !== \$b</code>	No idéntico	TRUE si <code>\$a</code> no es igual a <code>\$b</code> , o si no son del mismo tipo.
<code>\$a < \$b</code>	Menor que	TRUE si <code>\$a</code> es estrictamente menor que <code>\$b</code> .
<code>\$a > \$b</code>	Mayor que	TRUE si <code>\$a</code> es estrictamente mayor que <code>\$b</code> .
<code>\$a <= \$b</code>	Menor o igual que	TRUE si <code>\$a</code> es menor o igual que <code>\$b</code> .
<code>\$a >= \$b</code>	Mayor o igual que	TRUE si <code>\$a</code> es mayor o igual que <code>\$b</code> .

pue

Programación básica: Estructuras de control de flujo

Estructura if:

```
if(mostrarMensaje == true) {  
    alert("Hola Mundo");  
}
```

Estructura if...else:

```
if(edad >= 18) {  
    alert("Eres mayor de edad");  
}  
else {  
    alert("Todavía eres menor de edad");  
}
```

Programación básica: Estructuras de control de flujo

Estructura if encadenados:

```
if(edad < 12) {  
    alert("Todavía eres muy pequeño");  
}  
else if(edad < 19) {  
    alert("Eres un adolescente");  
}  
else if(edad < 35) {  
    alert("Aun sigues siendo joven");  
}  
else {  
    alert("Piensa en cuidarte un poco más");  
}
```

Programación básica: Estructuras de control de flujo

Estructura for:

```
var mensaje = "Hola, estoy dentro de un bucle";  
  
for(var i = 0; i < 5; i++) {  
    alert(mensaje);  
}  
  
var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];  
  
for(var i=0; i<7; i++) {  
    alert(dias[i]);  
}
```

Programación básica: Estructuras de control de flujo

Estructura **foreach**:

```
var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];

for(i in dias) {
    alert(dias[i]);
}
```

pue

Programación básica: Estructuras de control de flujo

Estructura while:

```
var resultado = 0;  
var numero = 100;  
var i = 0;  
  
while(i <= numero) {  
    resultado += i;  
    i++;  
}
```

Programación básica:

Sentencia Break

- Las sentencias break permiten manipular el comportamiento normal de los bucles para detener el bucle.

```
var cadena = "En un lugar de la Mancha de cuyo nombre no quiero acordarme...";  
var letras = cadena.split("");  
var resultado = "";  
  
for(i in letras) {  
    if(letras[i] == 'a') {  
        break;  
    }  
    else {  
        resultado += letras[i];  
    }  
}  
alert(resultado);  
// muestra "En un lug"
```

pue

Programación básica: Sentencia Continue

- Las sentencia continue permiten manipular el comportamiento normal de los bucles para saltarse algunas repeticiones.

```
var cadena = "En un lugar de la Mancha de cuyo nombre no quiero acordarme...";  
var letras = cadena.split("");  
var resultado = "";  
  
for(i in letras) {  
    if(letras[i] == 'a') {  
        continue;  
    }  
    else {  
        resultado += letras[i];  
    }  
}  
alert(resultado);  
// muestra "En un lugr de l Mnch de cuyo nombre no quiero cordrme..."
```

pue

Programación básica: Funciones de Cadena

Length: calcula la longitud de una cadena de texto

```
var mensaje = "Hola Mundo";  
var numeroLetras = mensaje.length; // numeroLetras = 10
```

+: Concatena varias cadenas

```
var mensaje1 = "Hola";  
var mensaje2 = " Mundo";  
var mensaje = mensaje1 + mensaje2; // mensaje = "Hola Mundo"
```

Programación básica: Funciones de Cadena

ToUpperCase(): transforma todos los caracteres de la cadena a mayúsculas:

```
var mensaje1 = "Hola";  
var mensaje2 = mensaje1.toUpperCase(); // mensaje2 = "HOLA"
```

ToLowerCase(): transforma todos los caracteres de la cadena a minúsculas:

```
var mensaje1 = "HolA";  
var mensaje2 = mensaje1.toLowerCase(); // mensaje2 = "hola"
```

Programación básica: Funciones de Cadena

substring(inicio, final): Extrae una porción de una cadena de texto.

→ El segundo parámetro es opcional y si no se indica devuelve hasta el final:

```
var mensaje = "Hola Mundo";  
var porcion = mensaje.substring(2); // porcion = "la Mundo"
```

Programación básica: Funciones para Arrays

Length: calcula el número de elementos de un array

```
var vocales = ["a", "e", "i", "o", "u"];
var numeroVocales = vocales.length; // numeroVocales = 5
```

Programación básica: Funciones para Arrays

Pop(): Elimina el último elemento del array y lo devuelve.

→ El array original se modifica y su longitud disminuye en 1 elemento.

```
var array = [1, 2, 3];
var ultimo = array.pop();
// ahora array = [1, 2], ultimo = 3
```

Push(): Añade un elemento al final del array.

→ El array original se modifica y aumenta su longitud en 1 elemento.

```
var array = [1, 2, 3];
array.push(4);
// ahora array = [1, 2, 3, 4]
```

pue

Programación básica: Funciones para Arrays

Shift() : Elimina el primer elemento del array y lo devuelve.

→ El array modifica su longitud disminuida en 1 elemento.

```
var array = [1, 2, 3];
var primero = array.shift();
// ahora array = [2, 3], primero = 1
```

Unshift(): añade un elemento al principio del array.

→ El array original se modifica y aumenta su longitud en 1 elemento.

```
var array = [1, 2, 3];
array.unshift(0);
// ahora array = [0, 1, 2, 3]
```

pue

Programación básica:

Funciones: Creación

- Cuando se desarrolla es muy habitual utilizar una y otra vez las mismas instrucciones.
- **Inconvenientes:**
 - El código más largo .
 - Si se quiere modificar alguna de las instrucciones repetidas, se deben hacer tantas modificaciones como veces se haya escrito esa instrucción.
- Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.

Programación básica:

Funciones: Argumentos y Valores de Retorno

- Las variables que necesitan las funciones se llaman argumentos.
- la función debe indicar cuántos argumentos necesita y cuál es el nombre de cada argumento.
- Al invocar la función, se deben incluir los valores que se le van a pasar a la función.
- Los argumentos se indican dentro de los paréntesis que van detrás del nombre de la función y se separan con una coma (,).
- Para que la función devuelva un valor, solamente es necesario que la última sentencia sea **return** junto con el nombre de la variable que se quiere devolver.

Programación básica: Funciones: Ejemplos

```
function sumar(n1,n2){  
    var suma = n1 + n2;  
    console.log("suma: " + suma);  
    alert("suma: " + suma);  
};  
sumar(3,5); // suma: 8  
  
function sumar(n1,n2){  
    var suma = n1 + n2;  
    return suma;  
};  
  
var resultado = sumar(3,5);  
console.log("suma: " + resultado); // suma: 8  
alert("suma: " + resultado);
```

Programación básica: Funciones: Ejemplos

```
var sumar = function (n1,n2){  
    var suma = n1 + n2;  
    return suma;  
};  
console.log("suma: " + sumar(2,4)); // suma: 8  
var resultado = sumar(3,5);  
console.log("suma: " + resultado); // suma: 8
```

Programación básica:

Funciones Anónimas

- Útil cuando:
 - 1) Queremos pasar como parámetro una función muy sencilla y definirla a parte sería cuanto menos innecesario.
 - 2) Intentamos evitar el uso de variables globales.

Programación básica: Funciones: Ejemplos

```
(function(n1,n2){  
    var suma = n1 + n2;  
    return console.log("suma: " + suma);  
})(1,4); // suma: 8
```

Programación básica: Ámbito de las variables (**CLOSURE**)

- El ámbito de una variable ("scope" o "CLOSURE") es la zona del programa en la que se define la variable.
- Determina en qué lugares de nuestro programa se puede referenciar.
- JavaScript define dos ámbitos para las variables:
 - **global**
 - **local**.

pue

Programación básica: Ámbito de las variables. LOCAL

- En el siguiente ejemplo, se muestra algún mensaje?

```
function creaMensaje() {  
    var mensaje = "Mensaje de prueba";  
}  
creaMensaje();  
alert(mensaje);
```

- La variable mensaje se ha definido dentro de la función **creaMensaje()**
→ Es una variable local a la función.
- NO se muestra mensaje!

Programación básica: Ámbito de las variables. LOCAL

- En el siguiente ejemplo, se muestra algún mensaje?

```
var mensaje = "Mensaje de prueba";  
  
function muestraMensaje() {  
    alert(mensaje);  
}
```

- La variable mensaje se ha definido fuera de la función `creaMensaje()`
→ Es una variable local al programa.
- SI se muestra el mensaje!

Programación básica: Ámbito de las variables. GLOBAL

- ¿Qué sucede si una función define una variable local con el mismo nombre que una variable global que ya existe?

```
var mensaje = "gana la de fuera";  
  
function muestraMensaje() {  
    var mensaje = "gana la de dentro";  
    alert(mensaje);  
}  
  
alert(mensaje);      gana la de fuera  
muestraMensaje();  gana la de dentro  
alert(mensaje);      gana la de fuera
```

- las variables **locales prevalecen** sobre las globales, **pero sólo en su ámbito!**

Programación básica: Ámbito de las variables. GLOBAL

- ¿Qué sucede si dentro de una función se define una variable global con el mismo nombre que otra variable global que ya existe?

```
var mensaje = "gana la de fuera";
function muestraMensaje() {
    mensaje = "gana la de dentro";
    alert(mensaje);
}

alert(mensaje);      gana la de fuera
muestraMensaje();   gana la de dentro
alert(mensaje);      gana la de dentro
```

- la **variable dentro de la función** simplemente **modifica el valor** de la variable **global** definida anteriormente

pue

Programación básica:

Objetos

- Los objetos son elementos que representan un objeto del mundo real.
- Los objetos definen sus propiedades a través de pares **clave : valor**
- Las claves pueden ser cualquier palabra o número válido.
- El valor puede ser cualquier tipo de valor:
→ Número,Cadena,Array,Función, incluso otro objeto.
- Cuando un valor del objeto es una función se denomina **método**. De lo contrario, se denomina **propiedad**.

pue

Programación básica: Objetos. Creación

- Creación de un objeto (en formato JSON) que:
 - 1) No se puede inicializar
 - 2) No tiene ocultas sus propiedades

```
var punto = {
    x: 10,
    mover: function(val){
        this.x = this.x + val;
    },
    print: function(){
        return console.log('x: ' + this.x);
    }
};
console.log('x: ' + punto.x); // x: 10
punto.mover(2); // x: 12
punto.x = punto.x + 2; // x: 14
punto.print(); // x: 14
```

pue

Programación básica: Objetos. Creación

- Creación de un objeto (en formato JSON) que:
 - 1) Si se puede inicializar
 - 2) No tiene ocultas sus propiedades

```
function puntoGenerico(inicio){  
    var pg = {  
        x: inicio,  
        mover: function(val){  
            this.x = this.x + val;  
        },  
        printar: function(){  
            return console.log('x: ' + this.x);  
        }  
    }  
    return pg;  
};  
  
var punto1 = new puntoGenerico(10); // x: 10  
console.log("x: " + punto1.x); // x: 10  
punto1.x = punto1.x + 2; // x: 12  
punto1.mover(2); // x: 14  
punto1.printar(); // x: 14
```

pue

Programación básica: Objetos. Creación

- Creación de un objeto que:
 - 1) Si se puede inicializar
 - 2) No tiene ocultas sus propiedades

```
function punto2(inicio){  
    this.x = inicio;  
    this.mover = function(val){  
        this.x = this.x + val;  
    };  
    this.printar = function(){  
        return console.log('X: ' + this.x);  
    };  
}  
var p2 = new punto2(10); // X: 10  
console.log('X: ' + p2.x); // X: 10  
p2.mover(2); // X: 12  
p2.x = p2.x + 2; // X: 14  
p2.printar(); // X: 14
```

pue

Programación básica: Objetos. Creación

- Creación de un objeto que:
 - 1) Si se puede inicializar
 - 2) Si tiene ocultas sus propiedades

```
function punto3(inicio){  
    var x = inicio;  
    this.mover = function(val){  
        x = x + val;  
    };  
    this.printar = function(){  
        return console.log('X: ' + x);  
    };  
}  
  
var p3 = new punto3(10); // X: 10  
console.log('X: ' + p3.x); // propiedad del objeto oculta (x), propiedad de la instancia  
(x) no existe  
p3.mover(2); // X: 12  
p3.x = p3.x + 2; // Creacion de la propiedad de la instancia x  
p3.printar(); // X: 12
```

pue

Programación básica:

Práctica

- Crear un hotel con las siguientes propiedades:
 - Nombre
 - Número total de habitaciones
 - Número total de reservas
 - Método verDisponibilidad()
 - Que muestre si hay habitaciones libres
 - Método reservar()
 - Que reserve si hay habitaciones libres o muestre un error en consola

Programación básica:

Práctica: Solución

```
function hotel(n,h,r){  
    var nombre = n;  
    var habitaciones = h;  
    var reservas = r;  
    this.reservar = function(x){  
        var todos = true;  
        for (var i=1; i<=x; i++){  
            if ( reservas + i  <= habitaciones ){  
                continue;  
            } else {  
                todos = false;  
                break;  
            }  
        }  
        if (todos === true){  
            reservas = reservas + x;  
        } else {  
            console.log("No hay habitaciones suficientes");  
        }  
    };  
    this.disponibilidad = function(){  
        var quedan = habitaciones - reservas;  
        return quedan;  
    };  
}  
  
var hotel1 = new hotel("sol",10,7);  
console.log("Habitaciones disponibles: " + hotel1.disponibilidad());  
hotel1.reservar(2);  
console.log("Habitaciones disponibles: " + hotel1.disponibilidad());  
hotel1.reservar(2);
```

pue

Document Object Model (DOM)

Introducción:

- La creación del Document Object Model(DOM) es una de las innovaciones que más ha influido en el desarrollo de las páginas web.
- Nos permite acceder y manipular las páginas HTML como si fueran documentos XML.
- De hecho, DOM se diseñó originalmente para manipular de forma sencilla los documentos XML.
- A pesar de sus orígenes, DOM se ha convertido en una utilidad disponible para la mayoría de lenguajes de programación (Java, PHP, JavaScript).

Document Object Model (DOM)

Árbol de nodos:

- Una de las tareas habituales con JavaScript consiste en la manipulación de las páginas web.
- Es habitual obtener el valor almacenado por algún elementos y crear otro elemento de forma dinámica y para añadirlo a la página.
- Para poder utilizar las utilidades de DOM, es necesario "transformar" la página original.

Document Object Model (DOM)

Árbol de nodos:

- Una página HTML normal no es más que una sucesión de caracteres, por lo que es un formato muy difícil de manipular.
- Por ello, los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular.
- DOM transforma todos los documentos en elementos llamados nodos, que están interconectados y las relaciones entre ellos.
- Por su aspecto, la unión de todos los nodos se llama "árbol de nodos".

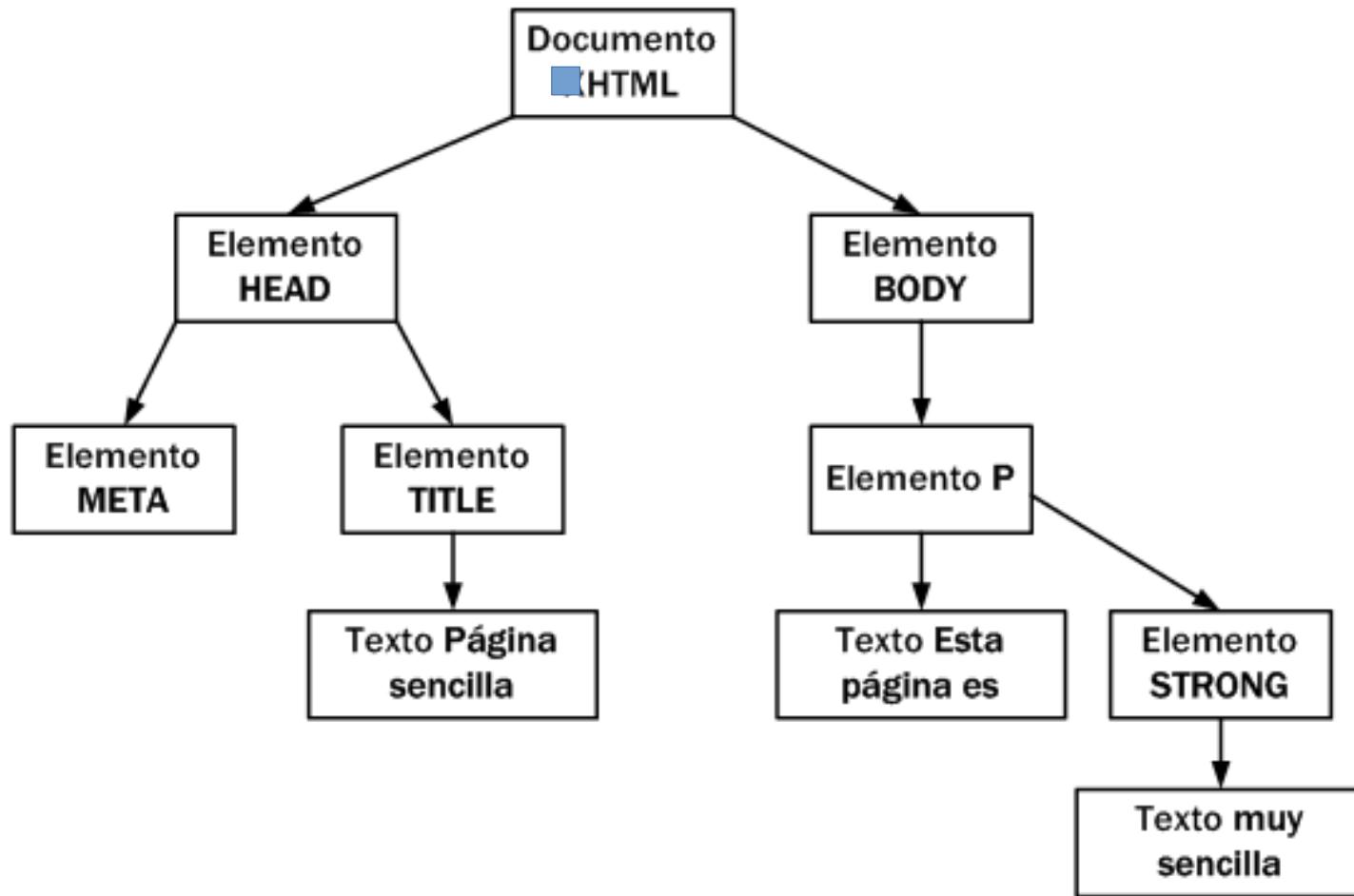
Document Object Model (DOM)

Árbol de nodos:

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Página sencilla</title>
  </head>
  <body>
    <p>Esta página es <strong>muy sencilla</strong></p>
  </body>
</html>
```

Document Object Model (DOM)

Árbol de nodos:



pue

Document Object Model (DOM)

Tipos de nodos.

- La especificación completa de DOM define 12 tipos de nodos, aunque habitualmente se pueden manipular manejando solamente cuatro tipos de nodos:
 - 1) **Document**: Padre de los demás nodos del árbol.
 - 2) **Element**: Representa cada una de las etiquetas. Se trata del único nodo que:
 - Puede contener atributos
 - Del que pueden derivar otros nodos.
 - 3) **Attr**: Representar cada uno de los atributos de las etiquetas. Uno por cada par atributo=valor.
 - 4) **Text**: Contiene el texto de una etiqueta.

Document Object Model (DOM)

Acceso directo a los nodos.

getElementsByName(“etiqueta”)

→ Obtiene TODOS LOS ELEMENTOS con esa etiqueta.

```
// obtener todos los párrafos  
var parrafos = document.getElementsByTagName("p");
```

```
// obtener el primer párrafo  
var primerParrafo = parrafos[0];  
console.log("Contenido Parrafo 0 " + primerParrafo.innerText);
```

```
// recorrer todos los párrafos de la página  
for(var i=0; i<parrafos.length; i++) {  
    console.log("Contenido Parrafo " + i + " " + parrafos[i].innerText);  
}
```

Document Object Model (DOM)

Acceso directo a los nodos.

getElementsByName(“*nombre*”)

→ Obtiene TODOS LOS ELEMENTOS cuyo valor en el atributo **Name** cumpla con ese nombre.

```
var parrafoEspecial = document.getElementsByName("especial");
```

```
<p name="prueba">...</p>
```

```
<p name="especial">...</p>
```

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

Document Object Model (DOM)

Acceso directo a los nodos.

`getElementsByClassName("clase")`

→ Obtiene TODOS LOS ELEMENTOS cuyo valor en el atributo `Class` cumpla con esa clase.

```
var parrafoClase1| = document.getElementsByClassName("clase1");  </script>
```

```
<p class="prueba">...</p>
<p class="clase1">...</p>
<p>...</p>
```

Document Object Model (DOM)

Acceso directo a los nodos.

`getElementById("id")`

→ Obtiene SOLO EL PRIMER ELEMENTO cuyo valor en el atributo `Id` cumpla con ese id.

```
var cabecera = document.getElementById("cabecera");
```

```
<div id="cabecera">  
  <a href="/" id="logo">...</a>  
</div>
```

Document Object Model (DOM)

Acceso a través de selectores CSS.

- También disponemos de métodos que nos permiten acceder a cualquier nodo identificado por un selector CSS específico.
- Cuando la página tiene una estructura más compleja, los métodos anteriores no nos permiten un acceso sencillo
- Los métodos `querySelector` y `querySelectorAll` podemos hacer uso de los selectores CSS para simplificar nuestro código con acceso a los elementos más granular.

Document Object Model (DOM)

Acceso a través de selectores CSS.

querySelector(*selector*)

→ Obtiene SOLO EL PRIMER ELEMENTO cuyo valor cumpla con el selector.

querySelectorAll(*selector*)

→ Obtiene TODOS LOS ELEMENTOS cuyo valor cumpla con el selector.

Document Object Model (DOM)

Acceso a través de selectores CSS.

```
// para el primer elemento <input> hijo de otro elemento con id="principal"  
document.querySelector('#principal > input:first-child').onclick=mostrarAlClick;
```

```
// para el segundo elemento <input>, EMPEZANDO POR 1, hijo de otro elemento con id="principal"  
document.querySelector('#principal > input:nth-child(2)').onmouseover=mostrarAlPasar;
```

```
// Para todos los elementos <input> hijos de otro elemento con id="secundario"  
var inputSecundario = document.querySelectorAll('#secundario > input');  
inputSecundario[0].onclick=mostrarAlClick;  
inputSecundario[1].onmouseover=mostrarAlPasar;
```

Document Object Model (DOM)

Modificación del DOM.

- Algunas operaciones habituales son las de crear, modificar y eliminar nodos del árbol DOM, es decir, crear, modificar y eliminar "trozos" de la página web.
- Los principales métodos del DOM que nos permiten hacer esto son:

Document Object Model (DOM)

Modificación del DOM.

`document.createElement("etiqueta")`

→ Crea elemento del tipo indicado.

`document.createTextNode("texto")`

→ Crea un nodo de tipo texto

`nodoPadre.appendChild(nodoHijo)`

→ Añade nodoHijo como contenido de nodoPadre.

`document.body.appendChild(nodoHijo);`

→ Añade nodoHijo como último elemento del <body>

`nodo.parentNode`

→ Propiedad que almacena quién es su nodo Padre

Document Object Model (DOM)

Modificación del DOM.

`nodoPadre.insertBefore(nodoHijoInsertar,nodoHijoExistente)`

→ Inserta sobre nodoPadre el nodoHijoInsertar justo antes de nodoHijoExistente.

`nodoPadre.replaceChild(nodoHijoCreado,nodoHijoBorrado);`

→ Sustituye de nodoPadre a nodoHijoBorrado por nodoHijoCreado.

`nodoPadre.removeChild(nodo);`

→ Elimina nodo del nodoPadre.

`nodo.innerHTML;`

→ Alamacena todo el contenido del nodo, incluidos todos los nodos hijo, nietos, ...

Document Object Model (DOM)

Modificación del DOM.

`nodo.hasAttribute("atributo");`

→ Devuelve true si el nodo tiene definido dicho atributo

`nodo.getAttribute("atributo");`

→ Devuelve el valor de dicho atributo para ese nodo

`nodo.setAttribute("atributo","valor")`

→ Establece el valor de dicho atributo para ese nodo.

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

Documento HTML de plantilla:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Página sencilla</title>
  </head>

  <body>
    <p>Esta página es <strong>muy sencilla</strong></p>
  </body>
</html>
```

Cuyo resultado es:

Esta página es **muy sencilla**

pue

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

- Primera modificación con JavaScript:

```
// añadimos nuevo parrafo a la pagina
// PRIMERO: creamos elemento <p>
var miParrafo = document.createElement("p");
// SEGUNDO: añadimos atributo. <p name="mip">
miParrafo.setAttribute("name", "mip");
// TERCERO: creamos elemento texto
var contenido = document.createTextNode("hola mundo");
// CUARTO: añadimos texto al parrafo
miParrafo.appendChild(contenido);
// QUINTO: anadimos parrafo al final del <body>
document.body.appendChild(miParrafo);
```

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

- Resultado:

Esta página es muy sencilla

hola mundo

pue

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

- Segunda modificación con JavaScript:

```
// añadimos boton entre los 2 parrafos
// PRIMERO: quien es el padre de mi parrafo
var padreMiParrafo = miParrafo.parentNode;
// SEGUNDO: crear boton
var boton = document.createElement("input");
boton.setAttribute("type", "button");
boton.setAttribute("name", "boton");
boton.setAttribute("value", "boton");
// TERCERO: insertarlo
padreMiParrafo.insertBefore(boton, miParrafo);
```

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

- Resultado:

Esta página es **muy sencilla**

boton

hola mundo

pue

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

- Tercera modificación con JavaScript:

```
// modificamos el texto del parrafo insertado
// PRIMERO: crear nuevo parrafo
var nuevoParrafo = document.createElement("p");
nuevoParrafo.setAttribute("Name", "nuevoP");
nuevoParrafo.setAttribute("style", "color: Orange");
var nuevoTexto = document.createTextNode("nuevo
Texto");
nuevoParrafo.appendChild(nuevoTexto);
// SEGUNDO: sustituir desde el padre
padreMiParrafo.replaceChild(nuevoParrafo, miParrafo);
```

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

- Resultado:

Esta página es **muy sencilla**

boton

nuevo Texto

pue

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

- Cuarta modificación con JavaScript:

```
// quitamos el Parrafo inicial  
var parrafo = document.getElementsByTagName("p");  
parrafo[0].parentNode.removeChild(parrafo[0]);
```

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

- Resultado:

boton

nuevo Texto

pue

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

- Quinta modificación con JavaScript:

```
// Sobre el Parrafo:  
// PRIMERO: Le incluimos una Cabecera <h1>  
var queES = nuevoParrafo.innerHTML;  
nuevoParrafo.innerHTML = '<h1>' + queES + '</h1>';  
// SEGUNDO: Actualizamos el estilo del parrafo  
var miParrafoMod = document.getElementsByName("nuevoP")[0];  
if (miParrafoMod.hasAttribute("style")){  
    var valorAttrActual = miParrafoMod.getAttribute("style");  
    var valorAttrIncluir = " ;background: black; ";  
    var valorAttr = valorAttrActual + valorAttrIncluir;  
    miParrafoMod.setAttribute("style",valorAttr );  
}
```

Document Object Model (DOM)

Modificación del DOM: Ejemplo paso a paso

· Resultado:

boton

nuevo Texto

pue

Eventos - Introducción

- Hasta ahora, todos los scripts creados se ejecutan desde la primera instrucción hasta la última de forma secuencial.
- Gracias a las estructuras de control de flujo (if, for, while) es posible modificar ligeramente este comportamiento y repetir algunos trozos del script y saltarse otros trozos en función de algunas condiciones.

Eventos - Introducción

- Este tipo de aplicaciones son poco útiles, ya que no interactúan con los usuarios.
- Afortunadamente, las aplicaciones web creadas con el lenguaje JavaScript pueden utilizar el modelo de programación basada en eventos.
- Los scripts se dedican a esperar a que el usuario "haga algo" (que pulse una tecla, que mueva el ratón, que cierre la ventana del navegador) y responde a la acción del usuario procesando esa información y generando un resultado.

Eventos - Tipos de eventos

- Cada elemento o etiqueta HTML define su propia lista de posibles eventos que se le pueden asignar.
- Un evento puede estar definido para varios elementos HTML diferentes.
- Un elemento HTML puede tener asociados varios eventos diferentes.
- El nombre de cada evento se construye mediante:
- → prefijo **on**,
→ seguido del nombre en inglés de la acción asociada.
- Por ejemplo:
→ El evento de pinchar un elemento con el ratón se denomina **onclick**.

Eventos - Tipos de eventos

- Los eventos más utilizados en las aplicaciones web tradicionales son:
 - **onload** para esperar a que se cargue la página por completo.
 - **onclick**, **onmouseover**, **onmouseout** para controlar el ratón.
 - **onsubmit** para controlar el envío de los formularios.
- Hay acciones típicas que realiza un usuario que pueden dar lugar a una sucesión de eventos.
 - Al pulsar un botón se desencadenan los eventos **onmousedown**, **onclick**, **onmouseup** y **onsubmit** de forma consecutiva.

Eventos - Tipos de eventos

Evento	Descripción	Elementos para los que está definido
onblur	Deseleccionar el elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onchange	Deseleccionar un elemento que se ha modificado	<input>, <select>, <textarea>
onclick	Pinchar y soltar el ratón	Todos los elementos
ondblclick	Pinchar dos veces seguidas con el ratón	Todos los elementos
onfocus	Seleccionar un elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onkeydown	Pulsar una tecla (sin soltar)	Elementos de formulario y <body>
onkeypress	Pulsar una tecla	Elementos de formulario y <body>
onkeyup	Soltar una tecla pulsada	Elementos de formulario y <body>
onload	La página se ha cargado completamente	<body>

Eventos - Tipos de eventos

Evento	Descripción	Elementos para los que está definido
onmousedown	Pulsar (sin soltar) un botón del ratón	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón "sale" del elemento (pasa por encima de otro elemento)	Todos los elementos
onmouseover	El ratón "entra" en el elemento (pasa por encima del elemento)	Todos los elementos
onmouseup	Soltar el botón que estaba pulsado en el ratón	Todos los elementos

pue

Eventos - Tipos de eventos

Evento	Descripción	Elementos para los que está definido
onreset	Inicializar el formulario (borrar todos sus datos)	<form>
onresize	Se ha modificado el tamaño de la ventana del navegador	<body>
onselect	Seleccionar un texto	<input>, <textarea>
onsubmit	Enviar el formulario	<form>
onunload	Se abandona la página (por ejemplo al cerrar el navegador)	<body>

pue

Eventos - Tipos de eventos

Evento	Descripción	Elementos para los que está definido
onmousedown	Pulsar (sin soltar) un botón del ratón	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón "sale" del elemento (pasa por encima de otro elemento)	Todos los elementos
onmouseover	El ratón "entra" en el elemento (pasa por encima del elemento)	Todos los elementos
onmouseup	Soltar el botón que estaba pulsado en el ratón	Todos los elementos

pue

Eventos - Manejadores de eventos

- Un evento de JavaScript por sí mismo, normalmente, carece de utilidad.
- Para que resulten útiles, se deben asociar funciones o código JavaScript para que, cuando se produce el evento se ejecute el código
- Las funciones o código JavaScript que se definen para cada evento se denominan "manejador de eventos".
- Existen varias formas diferentes de indicar los manejadores:
 - En línea
 - Embebido.
 - Fichero.

Programando en HTML 5 y JavaScript

Ubicación del código: EN LINEA

- Asociamos sobre el Evento del elemento/etiqueta una función anónima auto ejecutable.
- **Inconveniente:** Muy difícil de mantener.
 - Si hay que modificar el código establecido a 50 elementos,
 - tendremos que modificar 50 lugares de nuestro código

pue

Programando en HTML 5 y CSS 3

Ubicación del código: EN LINEA

```
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <title>
5              Ejemplo de uso con JavaScript EN LINEA
6          </title>
7      </head>
8      <body>
9          <div id="principal">
10         <input type="button" value="saludar" onclick="
11             function(){
12                 var miP = document.createElement('p');
13                 var textoP = document.createTextNode('Hola Mundo!!');
14                 miP.appendChild(textoP);
15                 document.getElementsByTagName('input')[0].parentNode.appendChild(miP);
16             }
17         }()
18     </div>
19 </body>
20 </html>
```

pue

Programando en HTML 5 y CSS 3

Ubicación del código: EMBEBIDO

- Etiqueta `<script></script>` dentro de la etiqueta `<head>`
- **CUIDADO!!**
 - Cuando el navegador lee la etiqueta `<script>` todavía no existe el elemento al que hemos asociado nuestro código a través de un evento.
 - Encerraremos todo nuestro código en una función que se ejecutará cuando se haya cargado toda la página (**evento load** sobre el **objeto window**).
- **Inconveniente:** Difícil de mantener.
 - Si hay que modificar el código establecido a 50 páginas,
 - tendremos que modificar 50 ficheros.

Programando en HTML 5 y CSS 3

Ubicación del código: EMBEBIDO

```
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <title>
5              Ejemplo de HTML5 con JavaScript EMBEBIDO
6          </title>
7          <script type="text/javascript">
8              function inicio(){
9                  var boton = document.getElementsByTagName("input");
10                 boton[0].onclick=function(){
11                     var miP = document.createElement("p");
12                     var textoP = document.createTextNode("hola Mundo!!");
13                     miP.appendChild(textoP);
14                     boton[0].parentNode.appendChild(miP);
15                 };
16             }
17             window.onload=inicio;
18         </script>
19     </head>
20     <body>
21         <div id="principal">
22             <input type="button" value="saludar">
23         </div>
24     </body>
25 </html>
```

pue

Programando en HTML 5 y CSS 3

Ubicación del código: FICHERO

- Etiqueta **atributo src** que referencia al fichero de la etiqueta `<script></script>` dentro de la etiqueta `<head>` apuntando a nuestro código
- También debemos esperar a que el navegador haya cargado toda la página (`window.load`).
- **Ventajas:** Facilita el mantenimiento, código bien estructurado.
→ Puede ser compartido por varias páginas de nuestra aplicación y realizamos los cambios en un único punto.

Programando en HTML 5 y CSS 3

Ubicación del código: FICHERO

```
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>
5       Ejemplo de HTML5 con JavaScript EMBEBIDO
6     </title>
7     <script type="text/javascript" src="javascript/14-jsFichero.js"></script>
8   </head>
9   <body>
10    <div id="principal">
11      <input type="button" value="saludar">
12    </div>
13  </body>
14 </html>
```

```
1 function inicio(){
2   var boton = document.getElementsByName("input");
3   boton[0].onclick=function(){
4     var miP = document.createElement("p");
5     var textoP = document.createTextNode("hola Mundo!!!");
6     miP.appendChild(textoP);
7     boton[0].parentNode.appendChild(miP);
8   };
9 }
10 window.onload=inicio;
```

pue

Programando en HTML 5 y CSS 3

Ubicación del código: FICHERO y Funciones Anónimas

```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <title> Ejemplo de HTML5 con JavaScript Fichero | </title>
5      <script type="text/javascript" src="javascript/14-jsFichero.js"></script>
6  </head>
7  <body>
8      <div id="principal">
9          <input type="button" value="saludar">
10     </div>
11  </body>
12 </html>
```

```
1 window.onload = function() {
2     var boton = document.getElementsByTagName("input");
3     boton[0].onclick=function(){
4         var miP = document.createElement("p");
5         var textoP = document.createTextNode("hola Mundo!!");
6         miP.appendChild(textoP);
7         miP.setAttribute("style","color: Orange; background-color: black");
8         boton[0].parentNode.appendChild(miP);
9     };
10};
```

pue

Document Object Model (DOM)

Ejercicios

- Sobre la página web Obtener la siguiente información:
 - Numero de enlaces de la pagina
 - Dirección del penúltimo enlace
 - Numero de enlaces del tercer párrafo
- ([15-EjercicioDOM-ContarEnlaces.html](#))

Document Object Model (DOM)

Ejercicios

- Página que muestre inicialmente la frase:
En un lugar de la mancha ...
- Cuando haga click en el texto debe completarse la frase con:
De cuyo nombre no quiero acordarme ...
- Y al volver a pinchar, debe desaparecer.
- ([16-EjercicioDOM-MostrarFrase.html](#))

Información del evento (objeto event)

- Los manejadores de eventos pueden requerir información adicional para procesar sus tareas.
- Por ejemplo, en los eventos asociados al teclado, puede ser importante conocer la tecla que se ha pulsado.
- JavaScript permite obtener información sobre el ratón y el teclado mediante un objeto especial llamado **event**.
- Desafortunadamente, los navegadores presentan diferencias en el tratamiento de la información sobre los eventos.

Información del evento (objeto event)

- La principal diferencia reside en cómo obtener el objeto **event**.
- **Internet Explorer** considera que este objeto forma parte del **objeto window**
- El **resto de navegadores** lo consideran como el único argumento que tienen las funciones manejadoras de eventos y no es necesario incluirlo en la llamada.
- Para utilizar este argumento, sólo es necesario asignarle un nombre, y los navegadores lo crean automáticamente.

Información del evento (objeto event)

- En Internet Explorer:
`var evento = window.event;`
- En el resto:
`function manejadorEventos(elEvento) {
 var evento = elEvento;
}`
- Para que funcione correctamente en todos los navegadores:
`function manejadorEventos(elEvento) {
 var evento = elEvento || window.event;
}`

pue

Información del evento (objeto event)

Sobre eventos de teclado

- Cuando un usuario pulsa una tecla normal, se producen tres eventos seguidos y en este orden:
 - **onkeydown** : pulsar una tecla y no soltarla.
 - **onkeypress** : la propia pulsación de la tecla.
 - **onkeyup** : soltar tecla pulsada.
- **La forma más sencilla** de obtener la información sobre la **tecla que se ha pulsado** es mediante el evento **onkeypress**.
- La información que proporcionan los eventos **onkeydown** y **onkeyup** **devuelven el código interno** de cada tecla y **no el carácter** que se ha pulsado.

Información del evento (objeto event)

Sobre eventos de teclado

- Para conocer la tecla pulsada con evento Keypress:
- **Internet Explorer** → Propiedad keyCode.
- **Resto de navegadores** → Propiedad charCode

```
1 window.onload = function(){
2     //ERES internet explorer??
3     var es_ie = navigator.userAgent.toLowerCase().indexOf('msie') != -1;
4     document.onkeypress = function(e) {
5         var evento = window.event || e;
6         var tecla;
7         if (es_ie){
8             tecla = String.fromCharCode(evento.keyCode);
9         }
10        else{
11            tecla = String.fromCharCode(evento.charCode);
12        }
13        console.log("Se ha pulsado: " + tecla);
14    };
15 }
```

pue

Información del evento (objeto event)

Sobre eventos de Ratón

- La información más relevante sobre los eventos relacionados con el ratón son las coordenadas de la posición del puntero del ratón.
- Es posible obtener la posición del ratón respecto a:
→ la pantalla del ordenador: **screenX** y **screenY**

Información del evento (objeto event)

Sobre eventos de Ratón

- Es posible obtener la posición del ratón respecto a:
→ la ventana del navegador: `clientX` y `clientY`

Información del evento (objeto event)

Sobre eventos de Ratón

- Es posible obtener la posición del ratón respecto a:
→ la propia página HTML (que se utiliza cuando el usuario ha hecho scroll sobre la página):
- **Internet Explorer:**
`evento.clientX + document.body.scrollLeft` y
`evento.clientY + document.body.scrollTop`
- **Resto navegadores:**
`PageX` y `pageY`

Información del evento (objeto event)

Sobre eventos de Ratón

```
1 window.onload = function(){
2     //ERES internet explorer??
3     var es_ie = navigator.userAgent.toLowerCase().indexOf('msie') != -1;
4     document.onclick = function(e) {
5         var evento = window.event || e;
6         var posicionPantallaX = evento.screenX;
7         var posicionPantallaY = evento.screenY;
8         var posicionNavegadorX = evento.clientX;
9         var posicionNavegadorY = evento.clientY;
10        var posicionPagina;
11        if (es_ie){
12            posicionPaginaX = evento.clientX + document.body.scrollLeft;
13            posicionPaginaY = evento.clientY + document.body.scrollTop;
14        }
15        else{
16            posicionPaginaX = evento.pageX;
17            posicionPaginaY = evento.pageY;
18        }
19        console.log("coordenadas en LA PANTALLA: (" + posicionPantallaX + "," + posicionPantallaY + ")");
20        console.log("coordenadas en EL NAVEGADOR: (" + posicionNavegadorX + "," + posicionNavegadorY + ")");
21        console.log("coordenadas en LA PAGINA: (" + posicionPaginaX + "," + posicionPaginaY + ")");
22    };
23}
```

Formularios - Introducción

- Una de las razones por las que se inventó JavaScript fue la necesidad de validar los datos de los formularios directamente en el navegador del usuario, evitando recargar la página cuando el usuario cometía errores al llenar los formularios.
- No obstante, la aparición de las aplicaciones AJAX y , sobre todo, las novedades con HTML5 ha relevado al tratamiento de formularios
- Ahora, el principal uso de JavaScript es el de las comunicaciones asíncronas con los servidores y el de la manipulación dinámica de las aplicaciones.

Formularios - Propiedades básicas

- Cuando se carga una página, el navegador crea automáticamente un **array llamado forms** y que contiene la referencia a todos los formularios de la página.
- Para acceder al array forms, se utiliza el **objeto document**. Por lo que `document.forms` es el array que contiene todos los formularios de la página.
- Además el navegador crea automáticamente un **array llamado elements** con la referencia a todos los elementos del formulario (cuadros de texto, botones, listas desplegables, etc...).

Formularios - Propiedades básicas

Utilizando la sintaxis de los arrays:

→ Obtener el primer elemento del primer formulario:

`document.forms[0].elements[0];`

→ Obtener último elemento del primer formulario:

`document.forms[0].elements[document.forms[0].elements.length-1];`

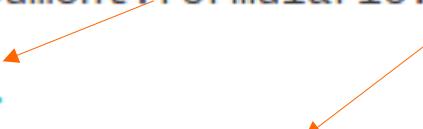
- **Inconveniente:** ¿Qué sucede si accediendo al DOM se modifican los formularios o sus elementos?
 - Por este motivo, siempre debería evitarse el acceso a los formularios mediante el **array `document.forms`**.

Formularios - Propiedades básicas

- Solución:

- 1) Acceder a través de su **atributo name**

```
var formularioPrincipal = document.formulario;  
var primerElemento = document.formulario.elemento;  
  
<form name="formulario">  
  <input type="text" name="elemento" />  
</form>
```



- 2) Acceder a través de su **atributo id**.

```
var formularioPrincipal = document.getElementById("formulario");  
var primerElemento = document.getElementById("elemento");  
  
<form name="formulario" id="formulario" >  
  <input type="text" name="elemento" id="elemento" />  
</form>
```

Formularios - Valor de los campos

Cuadro de texto y textarea:

- El valor del texto mostrado se obtiene y establece mediante la propiedad value.

```
<input type="text" id="texto" />  
var valor = document.getElementById("texto").value;  
  
<textarea id="parrafo"></textarea>  
var valor = document.getElementById("parrafo").value;
```

Formularios - Valor de los campos

Radiobutton

- Para conocer cuál se ha seleccionado tenemos la propiedad **checked** (true/false).

```
<input type="radio" value="si" name="pregunta" id="pregunta_si"/> SI  
<input type="radio" value="no" name="pregunta" id="pregunta_no"/> NO  
<input type="radio" value="nsnc" name="pregunta" id="pregunta_nsnc"/> NS/NC  
  
var elementos = document.getElementsByName("pregunta");  
  
for(var i=0; i<elementos.length; i++) {  
    alert(" Elemento: " + elementos[i].value + "\n Seleccionado: " + elementos[i].checked);  
}
```

Formularios - Valor de los campos

Checkbox

- Para conocer cuál se ha seleccionado tenemos la propiedad **checked** (true/false).

```
<input type="checkbox" value="condiciones" id="condiciones"/> hombre  
<input type="checkbox" value="privacidad" id="privacidad"/> mujer  
  
var elemento = document.getElementById("condiciones");  
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);  
  
elemento = document.getElementById("privacidad");  
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);
```

Formularios - Valor de los campos

Select

- Obtener el valor del atributo **value** de la etiqueta **<option>**.

Options:

Es un array con las todas las opciones de esa lista.

SelectedIndex:

índice del elemento seleccionado del array options cuando el usuario selecciona una opción.

Formularios - Valor de los campos

```
<select id="opciones" name="opciones">
    <option value="1">Primer valor</option>
    <option value="2">Segundo valor</option>
    <option value="3">Tercer valor</option>
    <option value="4">Cuarto valor</option>
</select>

// Obtener la referencia a la lista
var lista = document.getElementById("opciones");

// Obtener el índice de la opción que se ha seleccionado
var indiceSeleccionado = lista.selectedIndex;
// Con el índice y el array "options", obtener la opción seleccionada
var opcionSeleccionada = lista.options[indiceSeleccionado];

// Obtener el valor y el texto de la opción seleccionada
var textoSeleccionado = opcionSeleccionada.text;
var valorSeleccionado = opcionSeleccionada.value;
```

Modulo 7:

Introducción a

JQuery

pue

Introducción

- Jquery es una biblioteca que se encuentra escrita en JavaScript, creada inicialmente por John Resig.
- Permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
- jQuery es ligero con la filosofía, "**escribe menos, haz más**".
- Es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Introducción

- JQuery selecciona un gran conjunto de tareas comunes que requieren de varias líneas de código en JavaScript y las “envuelve” en métodos que se pueden llamar con una sola línea.
- Algunas de las características más comunes ofrecidas por la librería de jQuery son:
 - Manipulación de HTML/CSS/DOM
 - Gestión de eventos HTML
 - Efectos y animaciones
 - AJAX

Conceptos Básicos - Sintaxis Básica

- La sintaxis básica de uso de jQuery es:

`$(selector).metodo()`

- `$`: Define que se usa jQuery
- `(selector)`: Define como encontrar los elementos HTML
- `metodo()`: Acción a realizar sobre los elementos

Conceptos Básicos - Selectores

- El concepto más básico de jQuery es el de "seleccionar algunos elementos y realizar acciones con ellos".
- La biblioteca soporta gran parte de los selectores CSS3 y varios no estandarizados.
- Ejemplos de algunas técnicas comunes:

Conceptos Básicos - Selectores

Ejemplos

Selección de elementos en base a su `id`

```
$('#myId'); // notar que los IDs deben ser únicos por página
```

Selección de elementos en base al nombre de clase

```
$('.myClass'); // si se especifica la etiqueta se mejora el rendimiento
```

Selección de elementos por su atributo

```
 $('[name=first_name]'); // tenga cuidado, que puede ser muy lento
```

Selección de elementos en forma de selector CSS

```
 $('#contents ul.people li');
```

pue

Conceptos Básicos - Selectores

Ejemplos

Pseudo-selectores

```
// selecciona el primer elemento <a> con la clase 'external'  
$('a.external:first');
```

```
// selecciona todos los elementos <tr> impares de una tabla  
$('tr:odd');
```

```
// selecciona todos los elementos del tipo input dentro del formulario #myForm  
$('#myForm :input');
```

```
// selecciona todos los divs visibles  
$('div:visible');
```

Conceptos Básicos - Métodos

- El núcleo de jQuery son sus métodos/funciones, creadas con JavaScript, que permiten determinar que hacer y como hacerlo.
- Simplifican y facilitan la tarea, evitando escribir varias líneas de código de JavaScript.
- La mayoría de las funciones jQuery recibirán argumentos para especificar qué hacer o donde hacerlo.

Conceptos Básicos - Métodos

- En muchas ocasiones, estos argumentos pasados a la función serán a su vez otras funciones.
- En estas funciones es donde nosotros escribiremos nuestro código para personalizar el comportamiento de nuestra página web.
- Es decir, cómo argumento a una función definida en jQuery definiremos nuestra función.

Conceptos Básicos - Métodos

Ejemplo: Esperar cargar página

- No es posible interactuar de forma segura con el contenido de una página hasta que el documento no se encuentre preparado para su manipulación.
- jQuery permite detectar dicho estado a través de la declaración `$(document).ready()` de forma tal que el bloque se ejecutará sólo una vez que la página este disponible.

```
$(document).ready(function() {  
    console.log('el documento está preparado');  
});
```

Conceptos Básicos - Selectores

Funciones de Refinamiento de selecciones

- A veces, el selector contiene más de lo que necesita y es necesario refinar dicha selección.
- jQuery ofrece varios métodos para poder obtener exactamente lo que desea.

```
$('div.foo').has('p');           // el elemento div.foo contiene elementos <p>  
$('h1').not('.bar');             // el elemento h1 no posse la clase 'bar'  
$('ul li').filter('.current');   // un item de una lista desordenada  
                                // que posse la clase 'current'  
$('ul li').first();              // el primer item de una lista desordenada  
$('ul li').eq(5);                // el sexto item de una lista desordenada
```

Conceptos Básicos - Selectores

Encadenamiento

- Consiste en utilizar varias funciones de jQuery juntas para realizar acciones sobre un elemento concreto.

```
$('#content').find('h3').eq(2).html('nuevo texto para el tercer elemento h3');
```

- El encadenamiento es muy poderoso ya que no existe una regla que indique el límite.
- Por ello, debe ser utilizado con cuidado porque puede hacer un código extremadamente difícil de modificar y depurar.
- si se está escribiendo un encadenamiento de métodos que incluyen muchos pasos, es posible escribirlos línea por línea, haciendo que el código luzca más agradable para leer.

Conceptos Básicos - Selectores

Encadenamiento

- Si se está escribiendo un encadenamiento de métodos que incluyen muchos pasos, es posible escribirlos línea por línea, para facilitar su lectura.
- Así, este ejemplo:

```
$( '#content' ).find( 'h3' ).eq(2).html('nuevo texto para el tercer elemento h3');
```

- Podríamos escribirlo así:

```
$( '#content' )
    .find( 'h3' )
    .eq(2)
    .html('nuevo texto para el tercer elemento h3');
```

Conceptos Básicos - Selectores

Encadenamiento

- Después de navegar por los elementos a través del encadenamiento y realizar alguna acción sobre un elemento podemos volver al inicio.
 - Método `.end()`
 - Esto nos permite navegar y realizar otra acción sobre otro elemento sin necesidad de volver a seleccionarlo.
- ```
• $('#content')
 .find('h3')
 .eq(2)
 .html('nuevo texto para el tercer elemento h3')
 .end() // reestablece la selección a todos los elementos h3 en #content
 .eq(0)
 .html('nuevo texto para el primer elemento h3');
```

# Conceptos Básicos - Getters And Setters

---

- jQuery "sobrecarga" sus métodos permitiendo usar el mismo tanto para establecer un valor como para obtener un valor.
- Cuando un método es utilizado para establecer un valor, es llamado método “establecedor” (en inglés setter).
- Cuando un método es utilizado para obtener (o leer) un valor, es llamado “obtenedor” (en inglés getter).

```
El método .html() utilizado como SETTER
$('h1').html('hello world');
```

```
El método html utilizado como GETTER
var obtener = $('h1').html();
```

# Estilos

---

- jQuery incluye una manera útil de obtener y establecer propiedades CSS a los elementos: Método `.css()`.

## Obtener propiedades CSS

```
$('h1').css('fontSize'); // devuelve una cadena de caracteres como "19px"
$('h1').css('font-size'); // también funciona
```

## Establecer propiedades CSS

```
// establece una propiedad individual CSS
$('h1').css('fontSize', '100px');
```

- Cuando necesitamos establecer mas de un estilo, deberemos pasar un objeto JSON:

```
// establece múltiples propiedades CSS
$('h1').css({ 'fontSize' : '100px', 'color' : 'red' });
```

pue

# Estilos - Uso de Clases

---

- `.css()` es muy útil para obtener los estilos aplicados, pero, para establecer estilos se debe evitar.
- En su lugar deberíamos escribir reglas CSS aplicadas a clases que describan los diferentes estados visuales.
- Posteriormente cambiar la clase del elemento para aplicar el estilo.
- Los métodos usados son:

`.addClass(clase)`

→ Añade la clase

`.removeClass(clase)`

→ Quita la clase

`.toggleClass(clase)`

→ Si está la Quita

→ Si no está la Añade

```
var $h1 = $('h1');
```

```
$h1.addClass('big');
```

```
$h1.removeClass('big');
```

```
$h1.toggleClass('big');
```

pue

# Estilos - Dimensiones

---

- Métodos para obtener o establecer la dimensión de un elemento:

```
$('.h1').width('50px'); // establece el ancho de todos los elementos H1
```

```
$('.h1').width(); // obtiene el ancho del primer elemento H1
```

```
$('.h1').height('50px'); // establece el alto de todos los elementos H1
```

```
$('.h1').height(); // obtiene el alto del primer elemento H1
```

# Atributos

---

- jQuery incluye una manera útil de obtener y establecer atributos a los elementos.
- Métodos `.attr()` y `.removeAttr('atributo')`.  
`.attr()` permite tanto establecer como obtener el valor.

```
($('a').attr('href', 'allMyHrefsAreTheSameNow.html');
var enlace = $('a').attr('href');
```

- Al igual que el método `.css()`, cuando necesitamos establecer varios atributos usaremos un objeto JSON:

```
($('a').attr({
 'title' : 'all titles are the same too',
 'href' : 'somethingNew.html'
});
```

# Recorrer el DOM

---

- Una vez obtenida la selección, es posible encontrar otros elementos a partir de esa misma selección.
- Hay que ser cuidadoso en recorrer largas distancias pues obliga a que la estructura del documento sea siempre la misma, algo que es difícil de garantizar.
- Uno o dos pasos para el recorrido debería ser el límite.

```
// seleccionar el inmediato y próximo elemento <p> con respecto a H1
$('h1').next('p');
```

```
// seleccionar el elemento contenedor a un div visible
$('div:visible').parent();
```

```
// seleccionar todos los elementos hijos de #myList
$('#myList').children();
```

```
// seleccionar todos los items hermanos del elemento
$('li.selected').siblings();
```

# Manipulación del DOM

---

- Algunas operaciones habituales son las de crear, modificar y eliminar nodos del árbol DOM, es decir, crear, modificar y eliminar "trozos" de la página web.
- Los principales métodos del DOM que nos permiten hacer esto son:

# Modificación del DOM.

---

- Cambiar el HTML interno:

```
$('#myDiv p:first')
 .html('Nuevo primer párrafo');
```

# Modificación del DOM.

---

- Para Mover, Copiar y Eliminar elementos podemos usar 2 enfoques:
  - Tomar como referencia el elemento seleccionado

```
$("p").insertAfter("#foo");
```

- Tomar de referencia el elemento a Modificar

```
$("#foo").after("p");
```

- En ambos casos se inserta párrafo después de elementos con clase foo.

# Modificación del DOM.

---

- Las funciones que se pueden utilizar con ambos enfoques son:

.InsertBefore() / .before()

→ Inserta ... antes de ...

.insertAfter() / .after()

→ Inserta ... después de ...

.appendTo() / .append()

→ Inserta ... al final de ...

.prependTo() / .prepend()

→ Inserta ... al principio de ...

# Modificación del DOM.

---

## .remove()

→ Elimina elemento y todos sus descendientes.

## .detach()

→ Elimina elemento y todos sus descendientes.

→ Permite almacenar el elemento en una variable para su posterior inserción.

## .empty()

→ Elimina el contenido del elemento junto a todos sus descendientes

→ El propio elemento permanece.

# Modificación del DOM.

---

- Para Crear nuevos elementos jQuery provee una forma fácil y elegante a través del mismo método `$()` que se utiliza para realizar selecciones.
- Crear elemento nuevo:

```
var e1 = $('- item de la lista');

```

- Crear un nuevo elemento con atributos:

```
var e2 = $('- ', {
 html : 'item de la lista,'
 'class' : 'nuevo',
 href : 'foo.html'
});

```

Entre " por ser palabra reservada

# **Document Object Model (DOM)**

---

## **Modificación del DOM.**

- Crear un nuevo elemento en la página:

```
var e3 = $('- item de la lista');
e4.appendTo('ul');

```

- Crear y, a la vez, añadir un elemento a la página:

```
($('ul').append('item de la lista');
```

# Modificación del DOM: Ejemplo paso a paso

---

- Documento HTML de plantilla:

```
<!DOCTYPE html>
<html>
 <head>
 <title>Página sencilla</title>
 </head>

 <body>
 <p>Esta página es muy sencilla</p>
 </body>
</html>
```

- Cuyo resultado es:

Esta página es **muy sencilla**

pue

# Modificación del DOM: Ejemplo paso a paso

---

- Primera modificación con jQuery:

```
1 $(document).ready(function(){
2 // Añadimos nuevo parrafo a la pagina
3 // PRIMERO: Creo elemento <p>
4 var nuevoP = $("<p>hola mundo</p>");
5 // SEGUNDO: Añadimos atributo <p name='miP'>
6 nuevoP.attr('name','miP');
7 // TERCERO: Añadimos parrafo al final del <body>
8 $('body').append(nuevoP);
9 });
```

- La misma modificación pero en una sola línea

```
1 $(document).ready(function(){
2 // Lo mismo pero en una linea
3 $('body').append($("<p>hola mundo</p>").attr({
4 'name': 'miP',
5 'id': 'p'
6 }));
7});
```

pue

# **Document Object Model (DOM)**

---

## **Modificación del DOM: Ejemplo paso a paso**

- Resultado:

Esta página es muy sencilla

hola mundo

pue

# Modificación del DOM: Ejemplo paso a paso

---

- Segunda modificación con jQuery:

```
15 // Añadimos boton entre los 2 párrafos
16 // PRIMERO: Creo <input>
17 // SEGUNDO: le doy atributos para ser botón
18 // TERCERO: lo inserto antes de mi Parrafo
19 $('<input>').attr({
20 'type': 'button',
21 'name': 'boton',
22 'value': 'boton',
23 }).insertBefore('p[name=miP]');
24
25
26 });
```

# **Document Object Model (DOM)**

---

## **Modificación del DOM: Ejemplo paso a paso**

- Resultado:

Esta página es **muy sencilla**

**boton**

hola mundo

pue

# Modificación del DOM: Ejemplo paso a paso

---

- Tercera modificación con jQuery:

```
24 // Modificamos el parrafo insertado
25 // PRIMERO: cambiamos el texto
26 // SEGUNDO: modificamos estilo para ponerle color naranja a la letra
27 $('p[name=miP]').html("nuevo texto").css('color','DarkOrange');
```

# **Document Object Model (DOM)**

---

## **Modificación del DOM: Ejemplo paso a paso**

- Resultado:

Esta página es **muy sencilla**

boton

nuevo Texto

pue

# Modificación del DOM: Ejemplo paso a paso

---

- Cuarta modificación con jQuery:

```
28 //Quitamos el parrafo inicial
29 $('body > p:first').remove();
```

# **Document Object Model (DOM)**

---

## **Modificación del DOM: Ejemplo paso a paso**

- Resultado:

boton

nuevo Texto

pue

# Modificación del DOM: Ejemplo paso a paso

---

- Quinta modificación con jQuery:

```
30 //Sobre el parrafo
31 // PRIMERO: le incluimos una cabecera <h1>
32 // lo desattachamos para añadirlo despues dentro de <h1>
33 var antiguoP = $('p[name=miP]').detach();
34 $('body').append(
35 "<h1>").append(antiguoP)
36);
37 // SEGUNDO: Actualizamos el estilo del parrafo
38 // poniendo de color de fondo el negro
39 $('p[name=miP]').css({
40 'color': 'DarkOrange',
41 'background-color': 'Black',
42 });
```

# Modificación del DOM: Ejemplo paso a paso

---

- Resultado:

boton

nuevo Texto

pue

# Eventos

---

- Desde la versión 1.7 de jQuery tenemos el método **on()**, que nos ofrece toda la funcionalidad necesaria para gestionar eventos.
  - Gracias a este método ya no necesitamos los antiguos:
    - \* **bind()** → Para asociar un manejador a los elementos existentes al cargar la pagina.
    - \* **live()** o **delegate()** → Para asociar un manejador a los elementos que pudieran crearse posteriormente al cargar la pagina.
- Tampoco es necesario acceder directamente a los métodos asociados a los eventos como:
- \* **blur()**, **focus()**, **click()**, **hover()** entre otros.

# Eventos - Eventos más importantes

---

## Eventos del ratón

.mousedown() : presionar botón.

```
// al pulsar el botón, pongo texto de <a> de color Gris
$('input#boton1').on("mousedown",function(){
 $('a#enlace1').css({'color':'DarkGrey'});
});
```

.mouseup() : soltar botón pulsado.

```
// al soltar el botón, pongo texto de <a> de color naranja
$('input#boton1').on("mouseup",function(){
 $('a#enlace1').css({'color':'DarkOrange'});
});
```

# Eventos - Eventos más importantes

---

## Eventos del ratón

`.click()` : pulsar una vez el ratón sobre un elemento y soltarlo:

```
// al pulsar boton, pongo texto de <a> de color rojo
// al soltar el boton, deja el texto de <a> de color rojo
// tendrá prioridad sobre evento mouseup
$('input#boton1').on("click",function(){
 $('a#enlace1').css({'color':'Red'});
});
```

`.dblclick()` : pulsar el ratón dos veces seguidas sobre un elemento:

```
// al pulsar 2 veces, pongo texto de <a> de color verde
$('input#boton1').on("dblclick",function(){
 $('a#enlace1').css({'color':'Green'});
});
```

pue

# Eventos - Eventos más importantes

---

## Eventos del ratón

- En ocasiones es interesante que varios eventos sobre un mismo elemento que ejecuten la misma función.
- Por ejemplo, para **mouseup** junto a **click**

```
// El texto de <a> lo pongo rojo tanto si se lanza
// click como mouseup
$('input#boton1').on("click mouseup",function(){
 $('a#enlace1').css({'color':'Red'});
});
```

# Eventos - Eventos más importantes

---

## Eventos del ratón

.mouseenter() : El ratón se sitúa encima de un elemento.

**Para volver a dispararlo hay que salir:**

.mouseover() : El ratón se sitúa encima de un elemento.

**Para volver a dispararlo NO hay que salir:**

```
// al poner el raton encima de <a> ampliamos letra del texto
$('a#enlace1').on('mouseenter',function(){
 $(this).css({
 'color':$(this).css('color'),
 'font-size': '26px'
 });
});
```

.mouseleave() : El ratón, que estaba situado encima de un elemento, sale de él:

```
// al salir el raton de encima de <a> reducimos letra del texto
$('a#enlace1').on('mouseleave',function(){
 $(this).css({
 'color':$(this).css('color'),
 'font-size': '16px'
 });
});
```

pue

# Eventos - Eventos más importantes

---

## Eventos del ratón

- En ocasiones necesitamos ejecutar tareas diferentes para diferentes eventos, **compartiendo selector!**
- Pasamos a **.on()**, en formato JSON, los múltiples eventos(CLAVE) con sus funciones(VALOR).
- Un ejemplo: simular función **hover()** con los eventos mouseenter y mouseleave.

```
// Simular hover, haciendo lo mismo con el tamaño de letra
$('a#enlace1').on({
 'mouseenter':function(){
 $(this).css({
 'color':$(this).css('color'),
 'font-size': '26px'
 });
 }

 'mouseleave':function(){
 $(this).css({
 'color':$(this).css('color'),
 'font-size': '16px'
 });
 }
});
```

pue

# **Eventos - Eventos más importantes**

---

## **Eventos de Teclado**

- Se suelen aplicar al documento (document), y no a un elemento concreto y son:

**.keydown()** : Al presionar una tecla, liberada la presión o no.

**.keypress()** : Al tener pulsada una tecla.

**.keyup()** : Al dejar de presionar una tecla que teníamos pulsada.

# Eventos - Eventos más importantes

## Eventos combinados teclado-ratón

- Son los eventos que controlan el "foco" o elemento seleccionado.
  - Podemos llevar el foco a un elemento mediante:
    - Ratón: haciendo click
    - teclado: pulsando tabulador.
- .focus() : El elemento adquiere el foco.  
.blur() : El elemento pierde el foco.

```
// al coger foco añadir clase foco
$('a').on('focus',function(){
 $(this).addClass('foco');
});
// al perder el foco quitar la clase foco
$('a').on('blur',function(){
 $(this).removeClass('foco');
})
```

```
/* con clase foco poner un borde naranja */
.foco{
 border: 3px solid DarkOrange;
};
```

pue

# Eventos - El objeto del evento

---

- La función controladora del evento puede recibir un argumento donde almacenara el objeto del evento.
- El objeto del evento contiene métodos y propiedades:
  - pageX, pageY :**
    - posición del puntero relativo a la página
    - en el momento que el evento ocurrió.
  - Type :**
    - El tipo de evento (por ejemplo click).
  - Which :**
    - El botón o tecla presionada.
  - Target :**
    - El elemento DOM que inicializó el evento.
  - preventDefault()**
    - Cancela la acción predeterminada del evento
    - por ejemplo: seguir un enlace.

pue

# AJAX - Introducción

---

- Ajax (Asynchronous JavaScript and XML)
  - Permite a los navegadores **comunicarse con el servidor sin la necesidad de recargar TODA la página.**
  - Este método permite la creación de aplicaciones ricas en interactividad.
  - Las peticiones Ajax son ejecutadas por JavaScript:
    - Envía una petición a una URL
    - Recibe una respuesta y una función de devolución se ejecuta.
- La respuesta es asíncrona, por tanto, el resto del código de la aplicación continua ejecutándose.

# AJAX - Introducción

---

- A través de varios métodos nos abstraemos de las diferencias que pueden existir entre navegadores.
- Aunque la definición de Ajax posee la palabra XML, se suele utilizar el formato JSON (JavaScript Object Notation).

# AJAX - Conceptos clave

---

## GET y POST

- El método **GET** debe ser utilizado para operaciones donde se está obteniendo datos del servidor, pero no modificando.
- Por ejemplo, Pedir los datos de un usuario.
- El método **POST** debe ser utilizado para operaciones donde se está guardando información en el servidor.
- Por ejemplo, Crear un usuario nuevo.

# AJAX - Métodos

---

- Se facilita cargar los datos estáticos o dinámicos utilizando jQuery AJAX. JQuery proporciona método load () para hacer el trabajo

Sintaxis:

[Selector] .load (URL, [los datos], [de devolución de llamada]);

# AJAX - Métodos

---

**URL** - La URL del recurso del lado del servidor al que se envía la solicitud. Podría ser un CGI, ASP, JSP, PHP o script que genera datos de forma dinámica o fuera de una base de datos.

**datos** - Parámetro opcional que representa un objeto cuyas propiedades serán serializadas. la solicitud se realiza mediante el método POST. Si se omite, se utilizará GET.

**Función** - Invocada después de que los datos de respuesta se han cargado. El primer parámetro es la respuesta recibida desde el servidor y segundo parámetro es el código de estado.

# AJAX - Métodos

## Ejemplo

---

```
<html>
 <head>
 <title>Ejemplo de jQuery con AJAX</title>
 <script type="text/javascript"
 src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
 <script type="text/javascript" language="javascript">
 $(document).ready(function() {
 $("#driver").click(function(event){
 $('#stage').load('/jquery/result.html');
 });
 });
 </script>
 </head>
 <body>
 <p>Click on the button to load /jquery/result.html file -</p>
 <div id="stage" style="background-color:cc0;">
 STAGE
 </div>
 <input type="button" id="driver" value="Load Data" />
 </body>
</html>
```

pue

# AJAX - Métodos

---

## Ejemplo

- load() inicia una petición Ajax al archivo URL /jquery/result.html especificado.
- Después de cargar este archivo, todo el contenido se rellena el interior de la etiqueta <div>.
- Suponiendo, nuestro archivo /jquery/result.html tiene una sola línea de código HTML:

```
<h1> ESTO ES RESULTADO ... </ h1>
```

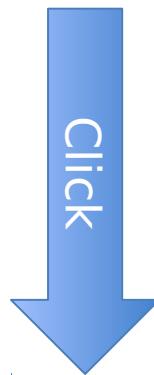
- Al hacer clic en el botón se carga el archivo result.html.

# Ajax - Métodos ejemplo

---

Click on the button to load result.html file –

STAGE



Click on the button to load result.html file –

**THIS IS RESULT...**

pue

# AJAX - Formato JSON

---

- Habrá situaciones en la que el servidor devolverá cadena JSON a la petición.
- La función `getJSON()` parsea la cadena JSON y la pone a disposición de la función como su primer parámetro.

sintaxis

`[Selector] .getJSON (URL, [datos], [funcion]);`

# AJAX - Formato JSON

---

**URL** - La URL del recurso del lado del servidor en contacto a través del método GET.

**datos** - Un objeto con los pares **clave / valor** que se utilizan para la construcción de la cadena de consulta que se usa en la dirección URL.

**Función** – Invocada después de que los datos de respuesta se han cargado. Recibe como parámetro el objeto JSON con los datos devueltos por el servidor.

# AJAX - Formato JSON

---

## Ejemplo

```
<html>
 <head>
 <title>The jquery Example</title>
 <script type="text/javascript"
 src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
 <script type="text/javascript" language="javascript">
 $(document).ready(function() {
 $("#driver").click(function(event){
 $.getJSON('/jquery/result.json', function(jd) {
 $('#stage').html('<p> Name: ' + jd.name + '</p>');
 $('#stage').append('<p>Age : ' + jd.age+ '</p>');
 $('#stage').append('<p> Sex: ' + jd.sex+ '</p>');
 });
 });
 });
 </script>
 </head>
 <body>
 <p>Click on the button to load result.json file -</p>
 <div id="stage" style="background-color:#eee;">
 STAGE
 </div>
 <input type="button" id="driver" value="Load Data" />
 </body>
</html>
```

# AJAX - Formato JSON

---

## Ejemplo

- `getJSON()` inicia una petición Ajax al archivo URL `result.json` especificado.
- Después de cargar este archivo, todo el contenido se pasa a la función que finalmente se rellena en la etiqueta `<div>`.
- Suponiendo, que el archivo `result.json` ha devuelto el siguiente contenido JSON:

```
{
 "name": "Zara Ali",
 "edad": "67",
 "sexo": "hembra"
}
```

# AJAX - Formato JSON

## Ejemplo

Cuando pulsamos el botón se genera el contenido

Click on the button to load result.html file –

STAGE

Load Data



Click on the button to load result.html file –

Name: Zara Ali

Age : 67

pue

# AJAX - Analizando datos del servidor

---

- Muchas veces recogemos datos introducidos por el usuario y se los pasamos al servidor para que los procese y almacene.
- Con jQuery y AJAX se simplifica este procedimiento simplemente pasándole como argumento los datos en formato JSON
- En el siguiente ejemplo muestra cómo pasar los datos de entrada del usuario a un script de servidor web:

# AJAX - Analizando datos del servidor

## Ejemplo

---

```
<html>
 <head>
 <title>The jQuery Example</title>
 <script type="text/javascript"
 src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
 <script type="text/javascript" language="javascript">
 $(document).ready(function() {
 $("#driver").click(function(event){
 var name = $("#name").val();
 $("#stage").load('/jquery/result.php', {"name":name});
 });
 });
 </script>
 </head>
 <body>
 <p>Enter your name and click on the button:</p>
 <input type="input" id="name" size="40" />

 <div id="stage" style="background-color:cc0;">
 STAGE
 </div>
 <input type="button" id="driver" value="Show Result" />
 </body>
</html>
```

pue

# AJAX - Analizando datos del servidor

---

## Ejemplo

Enter your name and click on the button –

STAGE

Show Result



Enter your name and click on the button –

Welcome Julio

Show Result

pue

# AJAX - Listado de otros métodos

---

- Una vez vistas las bases de jQuery con AJAX en la siguiente tabla tenemos otros métodos que pueden resultar útiles:

## Methods & Description

1 `jQuery.ajax( options )`

Load a remote page using an HTTP request.

2 `jQuery.ajaxSetup( options )`

Setup global settings for AJAX requests.

# AJAX - Listado de otros métodos

---

- 3 `jQuery.get( url, [data], [callback], [type] )`  
Load a remote page using an HTTP GET request.
- 4 `jQuerygetJSON( url, [data], [callback] )`  
Load JSON data using an HTTP GET request.
- 5 `jQuery.getScript( url, [callback] )`  
Loads and executes a JavaScript file using an HTTP GET request.
- 6 `jQuery.post( url, [data], [callback], [type] )`  
Load a remote page using an HTTP POST request.
- 7 `load( url, [data], [callback] )`  
Load HTML from a remote file and inject it into the DOM.
- 8 `serialize( )`  
Serializes a set of input elements into a string of data.
- 9 `serializeArray( )`  
Serializes all forms and form elements like the `.serialize()` method but returns a JSON data structure for you to work with.

# Efectos y Animaciones - Introducción

---

- Con jQuery, agregar efectos o animaciones a una página es muy fácil.
- Los efectos poseen una configuración predeterminada pero también es posible proveerles parámetros personalizados.
- Además es posible crear animaciones particulares estableciendo valores de propiedades CSS.

# Efectos y Animaciones

---

## Algunos Efectos de la biblioteca

- Los efectos más utilizados ya vienen incorporados dentro de la biblioteca en forma de métodos:

- `$.fn.show` Muestra el elemento seleccionado.
- `$.fn.hide` Oculta el elemento seleccionado.
- `$.fn.fadeIn` De forma animada, muestra y cambia la opacidad del elemento al 100 %.
- `$.fn.fadeOut` De forma animada, oculta y cambia la opacidad del elemento al 0%
- `$.fn.slideDown` Muestra elemento con un movimiento de deslizamiento vertical.
- `$.fn.slideUp` Oculta elemento con un movimiento de deslizamiento vertical.
- `$.fn.slideToggle` Muestra o oculta el elemento con un movimiento de deslizamiento vertical, dependiendo si actualmente el está visible o no.

```
$('h1') . show();
```

# Efectos y Animaciones

---

## Cambiar la duración de los efectos

- Con la excepción de `$.fn.show` y `$.fn.hide`, todos los métodos tienen una **duración predeterminada** de la animación en **400 milisegundos**.
- Este valor es posible cambiarlo.

```
$('h1').fadeIn(300); // desvanecimiento en 300ms
$('h1').fadeOut('slow'); // utilizar una definición de velocidad interna
```

# Efectos y Animaciones

## Cambiar la duración de los efectos

- jQuery tambien proveé de un objeto que permite personalizar valores para la velocidad:
  - velocidad **predeterminada** valores para las
  - velocidad **slow**
  - velocidad **fast**

```
jQuery.fx.speeds = {
 slow: 600,
 fast: 200,
 // velocidad predeterminada
 _default: 400
}
```

```
if ($(e.target).hasClass('h1')){
 $('ul#v1').slideDown('slow');
};
if ($(e.target).hasClass('h2')){
 $('ul#v2').slideDown('slow');
};
```

pue

# Efectos y Animaciones

---

## Cambiar la duración de los efectos

- Además es posible ejecutar una función cuando termine el efecto.
- La **función a ejecutar** al finalizar el efecto **se pasa** como **segundo argumento** a la función que realiza el efecto

```
if ($(e.target).hasClass('h2')){
 $('ul#v2').slideDown('slow',function(){
 console.log('animacion terminada');
 });
};
```

# Efectos y Animaciones

## Efectos personalizados

---

- Es posible realizar animaciones en propiedades CSS utilizando el método `$.fn.animate`.
- Establece o cambia valores a propiedades CSS
- Solo se pueden modificar propiedades con valores numéricos.

```
$(this).animate({opacity : 1.0, width: "110%"});
```

# Efectos y Animaciones

## Efectos personalizados

- El método `$.fn.animate` permite 2 argumentos más:
  - Tiempo en realizar el evento
  - Función a realizar al terminar el evento.

```
$(this).animate({opacity : 1.0, width: "110%"},'slow',function(){
 console.log("animacion terminada");
});
```

# Efectos y Animaciones

## Control de los efectos

---

- jQuery provee otras herramientas para el manejo de animaciones.
- `$.fn.stop` Detiene las animaciones que se están ejecutando en el elemento seleccionado.
- `$.fn.delay` Espera un tiempo determinado antes de ejecutar la próxima animación.

```
$('h1').show(300).delay(1000).hide(300);
```

# Ejercicio

---

- Crear un menú con las siguientes características:
  - Menú principal horizontal con 4 elementos
  - De cada uno debe salir un submenu vertical
  - El submenu debe estar oculto y solo debe aparecer al pasar el ratón encima de su menu.
  - Realizar algún efecto que permita identificar visualmente por donde vamos dentro de los menus y submenus

pue

# Eventos - El objeto del evento

- Solución similar a este:

## Realizar eventos y animaciones con jQuery

menu1

menu2

## Realizar eventos y animaciones con jQuery

menu1

menu2

submenu1\_1

submenu1\_2

submenu1\_3

submenu1\_4

pue

---

# **Módulo 8:**

# **Aprender a usar Bootstrap**

pue

# Introducción

---

- Haciendo un resumen, con la aparición de la web 2.0 Internet ha cambiado y se ha transformado para dar acogida a todas las necesidades de sus usuarios, y por esa razón los sitios web también ha tenido que cambiar mucho.
- Desde aproximadamente 2011 se empezó a hablar de los sitios web responsive o adaptables a todo tipo de pantallas y dispositivos fuese cual fuese su tamaño.
- Esta capacidad de adaptación de los sitios web se consiguió utilizando técnicas CSS avanzadas para su desarrollo o utilizando frameworks CSS, siendo uno de los más populares **Bootstrap**.

# Introducción - Historia

---

- Bootstrap es un framework CSS desarrollado inicialmente (en el año 2011) por Twitter
- Permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web.
- Aunque el desarrollo del framework Bootstrap fue iniciado por Twitter, fue liberado bajo licencia MIT en el año 2011 y su desarrollo continua en un repositorio de GitHub.

# Introducción - Qué es Bootstrap

---

- Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño.
- Además, **Bootstrap** ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías.
- Desde la aparición de la **versión 3** el framework se ha vuelto bastante más compatible con desarrollo web responsive, entre otras características se han reforzado las siguientes:

# Introducción - Qué es Bootstrap

---

- **Sistema GRID** – Permite diseñar usando un GRID de 12 columnas donde se debe plasmar el contenido de forma mucho más fácil e intuitiva.
- **Media Queries** – Establece Media Queries para 4 tamaños de dispositivos diferentes variando dependiendo del tamaño de su pantalla de forma mucho más fácil.
- **Imagenes responsive** – Con solo insertar la imagen con la clase “img-responsive” las imágenes se adaptaran al tamaño del contenedor.

# Introducción - Compatibilidad

---

- Boostrap es compatible con la mayoría de navegadores web del mercado, y más desde la versión 3:
  - Google Chrome (en todas las plataformas).
  - Safari (tanto en iOS como en Mac).
  - Mozilla Firefox (en Mac y en Windows).
  - Internet Explorer (en Windows y Windows Phone).
  - Opera (en Windows y Mac).

# **Primeros pasos - Usando Bootstrap**

---

- Existen varias formas diferentes de empezar con Bootstrap, cada una orientada a un tipo de público en función de su nivel técnico:

# Primeros pasos - Usando Bootstrap

---

- **Descargar el código CSS y JavaScript compilado -**  
La forma más sencilla de empezar.  
→ DESVENTAJA: No incluye ni archivos originales ni documentación.  
→ Descarga: [getbootstrap.com](http://getbootstrap.com)

# Primeros pasos - Usando Bootstrap

---

- **Descargar el código fuente** - contiene todos los archivos originales de Bootstrap.
  - DESVENTAJA: requiere compilador Less y configuración.
  - Descarga: [github.com/twbs/bootstrap/releases](https://github.com/twbs/bootstrap/releases).

# Primeros pasos - Usando Bootstrap

---

- **Usando CDN (Content Delivery Network) -**
- La empresa NetDNA aloja de forma gratuita una copia de los archivos CSS y JavaScript de Bootstrap.
  - Para utilizar estos archivos, debemos incluir en el `<head>` los siguientes enlaces:

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css"
integrity="sha512-dTfge/zgoMYpP7QbHy4gWMEGsbsdZeCXz7irItjcC3sPUFtf0kuFbDz/ixG7ArTxmDjLXDmezHubeNikyKGVyQ==" crossorigin="anonymous">

<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css" integrity="sha384-aUGj/X2zp5rLCbBxumKTCw2Z50WgIr1vs/PFN4pra0TvYXW1Vyh2UtNUU0KAUhAX" crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"
integrity="sha512-K1qjQ+NcF2TY0/eI3M6v8EiNYZfA95pQumfvcVrTHtwQVDG+aHRqLi/ETn2uB+1JqwYqVG3LIvdm9lJ6imS/pQ==" crossorigin="anonymous"></script>
```

# Uso con CDN - Ejemplo

```
<!DOCTYPE html>
<html lang="en">
 <head>
 <meta charset="utf-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <title>Plantilla básica de Bootstrap</title>

 <!-- CSS de Bootstrap-->
 <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
 <!-- librerías opcionales que activan el soporte de HTML5 para IE8 -->
 <!--[if lt IE 9]>
 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
 <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
 <![endif]-->
 </head>
 <body>
 <h1>¡Hola mundo!</h1>

 <!-- Librería jQuery requerida por los plugins de JavaScript -->
 <script src="https://code.jquery.com/jquery.js"></script>

 <!-- Todos los plugins JavaScript de Bootstrap (también puedes
 incluir archivos JavaScript individuales de los únicos
 plugins que utilices)-->
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
 </body>
</html>
```

pue

# Diseñando con rejilla

---

- Se requiere el doctype de HTML5
- Bootstrap utiliza algunos elementos HTML y algunas propiedades CSS que requieren el uso del doctype de HTML5.
- No olvides incluir este doctype en todas tus páginas con el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
 ...
</html>
```

# Diseñando con rejilla

---

- El móvil es importante
- Añadir la siguiente etiqueta dentro de la cabecera <head> de las páginas:

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

- Si quieres deshabilitar el zoom en tus páginas, añade la propiedad user-scalable=no a la etiqueta anterior:

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

# Diseñando con rejilla

---

- Imágenes responsive, para que las imágenes se adapten automáticamente a la pantalla del dispositivo.
- Debemos añadir la clase .img-responsive a cada imagen que incluye max-width: 100%; y height: auto;

```

```

# Diseñando con rejilla

---

- Si quieres centrar una página respecto a la ventana del navegador, encierra sus contenidos dentro de un elemento (normalmente <div>) y aplícale la clase .container:

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

# Diseñando con rejilla - Tipos de rejillas

---

- Bootstrap incluye una rejila o retícula fluída
- Pensada para móviles y que cumple con el diseño web responsive.
- Esta rejilla crece hasta 12 columnas a medida que crece el tamaño de la pantalla del dispositivo.
- Bootstrap incluye clases CSS para utilizar la rejilla directamente en tus diseños:

# Diseñando con rejilla - Tipos de rejillas

---

## Introducción

- El diseño de páginas basado en rejilla se realiza mediante filas y columnas donde se colocan los contenidos.
- Así funciona la rejilla de Bootstrap:
  - 1) Las filas siempre se definen en un contenedor de:  
→ **anchura fija (.container)**  
→ **anchura variable,todo el ancho del dispositivo (.container-fluid).**
  - 2) Las filas se utilizan para agrupar horizontalmente a varias columnas.

# Diseñando con rejilla - Tipos de rejillas

---

## Introducción

- 3) El contenido siempre se coloca dentro de las columnas
- 4) Bootstrap define muchas clases CSS (como por ejemplo `.row` , `.col-xs-4` que) para crear rejillas rápidamente.
- 5) La separación entre columnas se realiza aplicando padding.  
→ Para contrarrestar sus efectos en la primera y última columnas, las filas (elementos `.row`) aplican márgenes negativos.

# Diseñando con rejilla - Tipos de rejillas

---

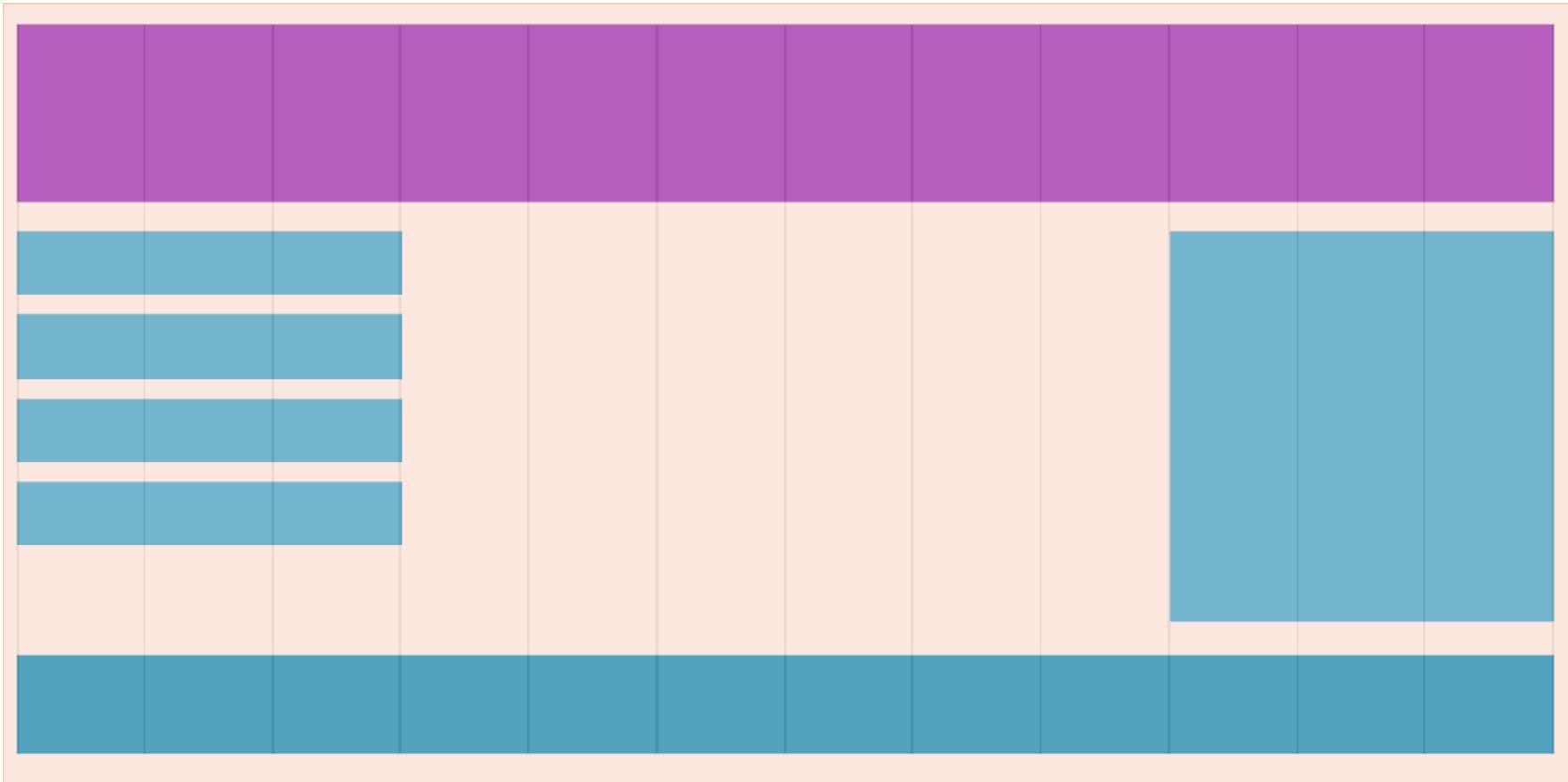
## Introducción

- 6) Las columnas de la rejilla definen su anchura especificando cuántas de las 12 columnas de la fila ocupan.
- Si por ejemplo quieres dividir una fila en tres columnas iguales, podemos utilizar la clase `.col-xs-4`
- **4 columnas por elemento / 12 columnas = 3 elementos en cada fila.**

pue

# Diseñando con rejilla - Tipos de rejillas

## Introducción



pue

# Diseñando con rejilla - Tipos de rejillas

## Media queries

- Establecer los diferentes puntos de ruptura en los que la rejilla se transforma para adaptarse a cada dispositivo.

```
/* Dispositivos muy pequeños (teléfonos de hasta 768px de anchura) */
/* No se define ninguna media query porque este es el estilo por
 defecto utilizado por Bootstrap 3 */

/* Dispositivos pequeños (tablets, anchura mayor o igual a 768px) */
@media (min-width: @screen-sm-min) { ... }

/* Dispositivos medianos (ordenadores, anchura mayor o igual a 992px) */
@media (min-width: @screen-md-min) { ... }

/* Dispositivos grandes (ordenadores, anchura mayor o igual a 1200px) */
@media (min-width: @screen-lg-min) { ... }
```

# Diseñando con rejilla - Tipos de rejillas

---

## Media queries

- En ocasiones, se define la propiedad max-width y min-width para ampliar el rango de dispositivos a los que se aplican los estilos CSS:

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

# Diseñando con rejilla - Tipos de rejillas

## Características de cada rejilla

	Dispositivos muy pequeños Teléfonos (<768px)	Dispositivos pequeños Tablets (≥768px)	Dispositivos medianos Ordenadores (≥992px)	Dispositivos grandes Ordenadores (≥1200px)
Comportamiento	Las columnas se muestran siempre horizontalmente.	Si se estrecha el navegador, las columnas se muestran verticalmente. A medida que aumenta su anchura, la rejilla muestra su aspecto horizontal normal.		
Anchura máxima del contenedor	Ninguna ( <code>auto</code> )	728px	940px	1170px
Prefijo de las clases CSS	<code>.col-xs-</code>	<code>.col-sm-</code>	<code>.col-md-</code>	<code>.col-lg-</code>
Número de columnas	12			
Anchura máxima de columna	<code>auto</code>	60px	78px	95px

pue

# Diseñando con rejilla - Tipos de rejillas

## Ejemplo de rejilla creada con Bootstrap

```
<!DOCTYPE html>
<html lang="en">
 <head>
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <title>Rejilla de Bootstrap</title>
 <!-- CSS de Bootstrap-->
 <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
 <style type="text/css">
 div.caja {
 border: 1px solid DarkOrange;
 }
 </style>
 <!-- Librería jQuery requerida por los plugins de JavaScript -->
 <script src="//code.jquery.com/jquery.js"></script>
 <!-- JavaScript de Bootstrap -->
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
 </head>
```

# Diseñando con rejilla - Tipos de rejillas

## Ejemplo de rejilla creada con Bootstrap

```
<body>
 <!-- Contenedor de tamaño variable
 <div class="container-fluid">
 -->
 <div class="container">
 <div class="row">
 <div class="caja col-md-1">col-md-1</div>
 <div class="caja col-md-1">col-md-1</div>
 </div>
 <div class="row">
 <div class="caja col-md-4">col-md-4</div>
 <div class="caja col-md-4">col-md-4</div>
 <div class="caja col-md-4">col-md-4</div>
 </div>
 <div class="row">
 <div class="caja col-md-8">col-md-4</div>
 <div class="caja col-md-4">col-md-4</div>
 </div>
 </div>
</body>
```

pue

# Diseñando con rejilla - Tipos de rejillas

## Ejemplo de rejilla creada con Bootstrap

col-md-1											
col-md-4				col-md-4				col-md-4			
col-md-4				col-md-4				col-md-4			

pue

# Diseñando con rejilla - Tipos de rejillas

---

## rejilla adaptada al tamaño de pantalla

- Si no quieres que las columnas de la rejilla se muestren verticalmente en los dispositivos pequeños
- utiliza a la vez las clases .col-xs-\* y .col-md-\*
- Podemos añadir las clases .col-sm-\* pensadas para tablets
- Dependiendo del tamaño de la pantalla usara la medida que mejor se ajusta.

```
<div class="row">
 <div class="col-xs-12 col-md-8">.col-xs-12 col-md-8</div>
 <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>
```

# Diseñando con rejilla - Tipos de rejillas

---

## rejilla adaptada al tamaño de pantalla

- Debido a los cuatro niveles de cuadrículas disponibles en Bootstrap, pueden darse problemas en ciertos puntos de corte o ruptura, ya que unas columnas son más altas que otras.
- Para solucionarlo, utilizamos la clase .clearfix y eliminamos una de las clases para que no se mezclen

```
<!-- Código opcional para limpiar las columnas XS en caso de que el
 contenido de todas las columnas no coincida en altura -->
<div class="clearfix visible-xs"></div>
<div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
```

# Diseñando con rejilla - Tipos de rejillas

## Desplazando columnas

---

- Es muy sencillo organizar la posición de los elementos (derecha/izquierda/centrado) poniendo un desplazamiento, dejando blanco a la derecha

```
<div class="row">
 <div class="caja col-md-2 col-md-offset-2">col-md-2 offset-2</div>
 <div class="caja col-md-2 col-md-offset-4">col-md-2 offset-4</div>
</div>
<div class="row">
 <div class="caja col-md-4">col-md4</div>
 <div class="caja col-md-4">col-md4</div>
 <div class="caja col-md-4">col-md4</div>
</div>
<div class="row">
 <div class="caja col-md-offset-2 col-md-2">col-md-2 offset-2</div>
 <div class="caja col-md-offset-4 col-md-4">col-md-4 offset-4</div>
</div>
```

# Diseñando con rejilla - Tipos de rejillas

---

## Desplazando columnas



pue

# Diseñando con rejilla - Tipos de rejillas

## Reordenando columnas

- Con las clases `.pull` y `.push` lo que hacemos es empujar y tirar “Significado literal”.
- Con **push** empujamos el elemento a la derecha el numero de columna que especifiquemos.
- Con **pull** tiramos el elemento a su izquierda el numero de columnas que especifiquemos.

```
<div class="row">
 <div class="caja col-md-6 col-md-push-6">Soy el primero</div>
 <div class="caja col-md-6 col-md-pull-6">Soy el segundo</div>
</div>
```

Soy el segundo

Soy el primero

pue

# Tipografía - Cabeceras

---

- Los estilos relacionados con la tipografía y el texto de los contenidos son esenciales.
- Bootstrap 3 define estilos por defecto para todos los niveles de las cabeceras y los elementos <small> incluidos dentro.

**h1. Bootstrap heading** Secondary text

**h2. Bootstrap heading** Secondary text

**h3. Bootstrap heading** Secondary text

# Tipografía - Texto

---

- El tamaño de letra (`font-size`) por defecto es 14px
- El interlineado (`line-height`) es 1.428.
- Se aplican tanto al `<body>` como a todos los `<p>`.
- Los `<p>` también incluyen un margen inferior de 10px

```
<p>Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.</p>
```

```
<p>Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec ullamcorper nulla non metus auctor fringilla. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.</p>
```

- Se vería así:

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec ullamcorper nulla non metus auctor fringilla. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.

pue

# Tipografía - Alineamiento del Texto

---

- Permite situar el texto fácilmente simplemente añadiendo una clase.

```
<p class="text-left">Left aligned text.</p>
<p class="text-center">Center aligned text.</p>
<p class="text-right">Right aligned text.</p>
```

Left aligned text.

Center aligned text.

Right aligned text.

# Tipografía - Transformación del Texto

---

- Permite intercambiar mayúsculas/minúsculas el texto fácilmente simplemente añadiendo una clase.

```
<p class="text-lowercase">Lowercased text.</p>
<p class="text-uppercase">Uppercased text.</p>
<p class="text-capitalize">Capitalized text.</p>
```

lowercased text.

UPPERCASED TEXT.

Capitalized Text.

# **Tipografía - Tipo de contenido a través del color del texto.**

```
<p class="text-muted">Fusce dapibus, tellus ac cursus commodo, tortor mauris nibh.</p>
<p class="text-primary">Nullam id dolor id nibh ultricies vehicula ut id elit.</p>
<p class="text-success">Duis mollis, est non commodo luctus, nisi erat porttitor ligula.</p>
<p class="text-info">Maecenas sed diam eget risus varius blandit sit amet non magna.</p>
<p class="text-warning">Etiam porta sem malesuada magna mollis euismod.</p>
<p class="text-danger">Donec ullamcorper nulla non metus auctor fringilla.</p>
```

Fusce dapibus, tellus ac cursus commodo, tortor mauris nibh.

Nullam id dolor id nibh ultricies vehicula ut id elit.

Duis mollis, est non commodo luctus, nisi erat porttitor ligula.

Maecenas sed diam eget risus varius blandit sit amet non magna.

Etiam porta sem malesuada magna mollis euismod.

Donec ullamcorper nulla non metus auctor fringilla.

# Tipografía - Listas

---

## Navegación sin estilo

- Como resulta muy habitual mostrar las listas sin viñetas y sin margen izquierdo se incluye una clase CSS llamada `.list-unstyled`.

```
<ul class="list-unstyled">
 Facilisis in pretium nisl aliquet
 Nulla volutpat aliquam velit

 Phasellus iaculis neque
 Purus sodales ultricies
 Vestibulum laoreet porttitor sem
 Ac tristique libero volutpat at


```

Facilisis in pretium nisl aliquet  
Nulla volutpat aliquam velit

- Phasellus iaculis neque
- Purus sodales ultricies
- Vestibulum laoreet porttitor sem
- Ac tristique libero volutpat at

# Tipografía - Listas

## Navegación horizontales

- También resulta habitual mostrar los elementos horizontalmente, como en el menú principal.
- Para ello, Bootstrap 3 define la clase CSS `.inline-block`.

```
<ul class="list-inline">
 Lorem ipsum
 Phasellus iaculis
 Nulla volutpat

```

Lorem ipsum Phasellus iaculis Nulla volutpat

# Tipografía - Listas

---

## Navegación con pestañas

- Aplica la clase `.nav` para crear un elemento de navegación.
- Después aplica la clase `.nav-tabs` para mostrar sus enlaces en forma de pestaña.

```
<ul class="nav nav-tabs">
 <li class="active">Inicio
 Perfil
 Mensajes

```



# Tipografía - Listas

---

## Navegación con botones

- Aplica la clase `.nav` para crear un elemento de navegación.
- Después aplica la clase `.nav-pills`

```
<ul class="nav nav-pills">
 <li class="active">Inicio
 Perfil
 Mensajes

```



# Tipografía - Listas

---

## Navegación con menús desplegables

- Aplica la clase `.nav` y `.nav-tabs` para crear el elemento de navegación padre `<ul>`.
  - Los elementos `<li>` del padre deben tener la clase `.dropdown`
  - Los anchors `<a>` debe llevar:
    - clase `.dropdown-toggle`
    - atributo `data-toggle="dropdown"`Aplica la clase `.dropdown-menu` para crear los elementos de navegación de los hijos`<ul>`.
- Mejor con un ejemplo:

# Tipografía - Listas

## Navegación con menús desplegables

```
<ul class="nav nav-tabs">
 <li class="dropdown">

 Menu desplegable 1

 <ul class="dropdown-menu">
 op1
 op2

 <li class="dropdown">

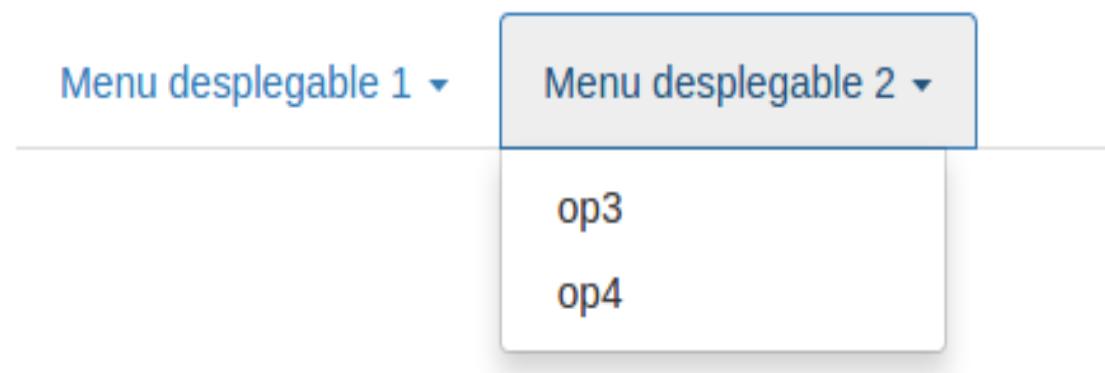
 Menu desplegable 2

 <ul class="dropdown-menu">
 op3
 op4


```

# Tipografía - Listas

## Navegación con menús desplegables



pue

# Tipografía - Listas

## Paginadores

---

```
<ul class="pager">
 Anterior
 Siguiente

```

[Anterior](#)[Siguiente](#)

```
<ul class="pagination">
 «
 1
 2
 3
 4
 5
 »

```

«	1	2	3	4	5	»
---	---	---	---	---	---	---

# Tablas - Tablas básicas

---

- Añade la clase `.table` a cualquier elemento `<table>` para aplicar los estilos básicos de Bootstrap.
- El resultado es una tabla con un padding muy sutil y con líneas de separación solamente en las filas.

```
<table class="table">
 ...
</table>
```

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

# Tablas - Tablas “cebreadas”

---

- Aquellas cuyas filas alternan su color de fondo para mejorar la legibilidad de los contenidos.

```
<table class="table table-striped">
 ...
</table>
```

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

# Tablas - Tablas con bordes

---

- Si prefieres utilizar el estilo tradicional de las tablas con los cuatro bordes en todas las celdas y en la propia tabla, añade la clase `.table-bordered`.

```
<table class="table table-bordered">
 ...
</table>
```

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

# Tablas - Tablas resaltadas

---

- Los contenidos de la tabla todavía más fáciles de entender añadiendo la clase `.table-hover`
- Modificar ligeramente el aspecto de las filas cuando pasa el ratón por encima

```
<table class="table table-hover">
 ...
</table>
```

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

pue

# Tablas - Tablas “responsive”

---

- La solución que propone Bootstrap 3 en dispositivos pequeños consiste en **añadir un scroll horizontal**.
- se ven normal cuando la anchura es superior a 768px.
- Para ello, encierra cualquier tabla con la clase **.table** dentro de un elemento con la **clase .table-responsive**.

```
<div class="table-responsive">
 <table class="table">
 ...
 </table>
</div>
```

#	Cabecera de tabla	Cabecera de tabla	C
1	Celda de tabla	Celda de tabla	C
2	Celda de tabla	Celda de tabla	C
3	Celda de tabla	Celda de tabla	C

pue

# Utilidades de posicionamiento

---

## Elementos flotantes

- Flotar un elemento a la derecha o a la izquierda es muy habitual.
- Bootstrap define dos clases CSS - .pull-left / .pull-right - que puedes aplicar sobre cualquier elemento:

```
<div class="pull-left">...</div>
<div class="pull-right">...</div>
<div class="center-block">...</div>
```

# Utilidades de posicionamiento

---

## Limpiando floats

- Cuando un diseño utiliza muchos elementos flotantes, es común tener que limpiar un elemento para que no le afecten otros elementos flotantes.

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

## Ocultando y mostrando elementos

```
<div class="show">...</div>
<div class="hide">...</div>
```

pue

## Formularios - Formulario básico

---

- Bootstrap aplica por defecto algunos estilos a todos los componentes de los formularios.
- Si además añades la clase `.form-control` a los elementos `<input>`, `<textarea>` y `<select>`, su anchura se establece a `width: 100%`.
- Para optimizar el espaciado, utiliza la clase `.form-group` para encerrar cada campo de formulario con su `<label>`.

# Formularios - Formulario básico

---

```
<form role="form">
 <div class="form-group">
 <label for="ejemplo_email_1">Email</label>
 <input type="email" class="form-control" placeholder="Introduce tu email">
 </div>
 <div class="form-group">
 <label for="ejemplo_password_1">Contraseña</label>
 <input type="password" class="form-control" placeholder="Contraseña">
 </div>
 <button type="submit" class="btn btn-default">Enviar</button>
</form>
```

---

Email

Introduce tu email

Contraseña

Contraseña

Enviar

pue

# Formularios - Formulario en línea

---

- Para que el formulario ocupe el menor espacio posible, añade la clase `.form-inline`.
- Las etiquetas `<label>` se muestren a la izquierda de cada campo del formulario.

```
<form class="form-inline" role="form">
 <div class="form-group">
 <label for="ejemplo_email_1">Email</label>
 <input type="email" class="form-control" placeholder="Introduce tu email">
 </div>
 <div class="form-group">
 <label for="ejemplo_password_1">Contrasena</label>
 <input type="password" class="form-control" placeholder="Contrasena">
 </div>
 <button type="submit" class="btn btn-default">Enviar</button>
</form>
```

The screenshot shows a web form with a light gray background and a thin orange border. It contains two groups of inputs. The first group, labeled "Email", has a blue placeholder "Introduce tu email". The second group, labeled "Contraseña", also has a blue placeholder "Contraseña". To the right of these groups is a blue "Enviar" button. The entire form is centered on the page.

pue

## Formularios - Formularios horizontales

---

- Bootstrap también permite, además de alinear los elementos `<label>` y sus campos , definir que los elementos se comporten como las rejillas.
- Para ello, añade la clase `.form-horizontal` al formulario.
- Esta clase modifica la clase `.form-group` para que se comporte como la fila de una rejilla, así puedes añadir elementos de la clase `.row` para definir cuanto ocupan.

# Formularios - Formularios horizontales

```
<form class="form-horizontal" role="form">
 <div class="form-group">
 <label class="col-md-1 col-md-offset-4" for="ejemplo_email_1">Email</label>
 <input class="col-md-2" type="email" class="form-control" placeholder="Introduce tu email">
 </div>
 <div class="form-group">
 <label class="col-md-1 col-md-offset-4" for="ejemplo_password_1">Contrasena</label>
 <input class="col-md-2" type="password" class="form-control" placeholder="Contrasena">
 </div>
 <button class="col-md-1 col-md-offset-4 btn btn-default" type="submit">Enviar</button>
</form>
```

A screenshot of a web form with a light orange background. It contains two input fields: one for 'Email' with placeholder 'Introduce tu email' and another for 'Contraseña' with placeholder 'Contraseña'. Below the inputs is a large, rounded rectangular button labeled 'Enviar'.

pue

# Formularios - Campos de formulario

---

- Bootstrap define estilos adecuados para todos y cada uno de los campos de formulario existentes.
- Solamente se aplican los estilos a los campos `<input>` que definen explícitamente su tipo mediante el **atributo type**.

# Formularios - Campos de formulario

---

## Checkboxes y radio buttons en línea

- para que ocupen menos espacio, utiliza las clases CSS `.checkbox-inline` o `.radio-inline`.

```
<label class="checkbox-inline">
 <input type="checkbox" id="checkboxEnLinea1" value="opcion_1"> 1
</label>
<label class="checkbox-inline">
 <input type="checkbox" id="checkboxEnLinea2" value="opcion_2"> 2
</label>
```



# Formularios - Campos de formulario

---

## Listas desplegables

- Para mostrar una lista desplegada, añade el atributo multiple.

```
<select class="form-control">
 <option>1</option>
 <option>2</option>
 <option>3</option>
 <option>4</option>
 <option>5</option>
</select>
```

```
<select multiple class="form-control">
 <option>1</option>
 <option>2</option>
 <option>3</option>
 <option>4</option>
 <option>5</option>
</select>
```



1



1  
2  
3  
4

pue

# **Formularios - Campos de formulario**

---

## **Estados de formulario**

- Modificar el estado de los controles del formulario o de sus elementos <label> es una de las mejores formas de proporcionar información adicional a los usuarios.

# Formularios - Campos de formulario

---

## Estados de formulario

### Campos seleccionados

- Bootstrap aplica una sombra al **aplicar la clase form-control**, a los campos seleccionados mediante la propiedad **box-shadow** de CSS aplicada a la pseudo-clase `:focus` del elemento.

```
<input class="form-control" type="text" value="Este campo está seleccionado...">
```

Este campo está seleccionado...

# Formularios - Campos de formulario

---

## Estados de formulario

### Campos deshabilitados

- Añadiendo el atributo `disabled` a campos de texto, además de evitar que el usuario pueda introducir información, Bootstrap lo muestra con un aspecto muy diferente.

```
|<input class="form-control" type="text" disabled placeholder="Este campo está deshabilitado...">
```

Este campo está deshabilitado...

# **Formularios - Campos de formulario**

---

## **Estados de formulario**

### **Estados de validación**

- Bootstrap define varios estilos para indicar el estado de la validación de cada campo del formulario:
  - `.has-warning` : para las advertencias,
  - `.has-error` :para los errores
  - `.has-success` : para cuando el valor es correcto.
- Lo mejor es que estas clases se pueden aplicar a cualquier elemento que contenga una de las tres siguientes **clases**: `.control-label`, `.form-control` y `.help-block`.

# Formularios - Campos de formulario

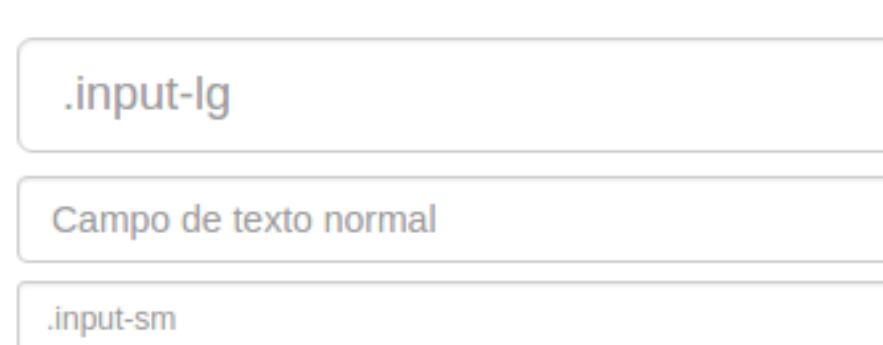
## Redimensionando campos

---

### Cambiando la altura

- Podemos modificar la altura de los elementos utilizando las clases .input-XX

```
<input class="form-control input-lg" type="text" placeholder=".input-lg">
<input class="form-control" type="text" placeholder="Campo de texto">
<input class="form-control input-sm" type="text" placeholder=".input-sm">
```



The image shows three input fields side-by-side. The top field is labeled '.input-lg' and has a larger height than the others. The middle field is labeled 'Campo de texto normal' and is of standard height. The bottom field is labeled '.input-sm' and has a smaller height than the others.

# Formularios - Campos de formulario

## Redimensionando campos

### Cambiando la anchura

- La forma más sencilla consiste en encerrarlos en otros elementos y aplicar las clases ya vistas para rejillas.

```
<div class="row">
 <div class="col-xs-3">
 <input type="text" class="form-control" placeholder=".col-xs-3">
 </div>
 <div class="col-xs-4">
 <input type="text" class="form-control" placeholder=".col-xs-4">
 </div>
 <div class="col-xs-5">
 <input type="text" class="form-control" placeholder=".col-xs-5">
 </div>
</div>
```

.col-xs-3

.col-xs-4

.col-xs-5

pue

# Formularios - Botones

## Diferentes maneras de crearlos

- Gracias a los estilos de Bootstrap, puedes utilizar cualquiera de las siguientes etiquetas para mostrar botones:

```
Enlace
<button class="btn btn-default" type="submit">Botón</button>
<input class="btn btn-default" type="button" value="Campo input">
<input class="btn btn-default" type="submit" value="Enviar">
```

Enlace

Botón

Campo input

Enviar

pue

# Formularios - Botones

---

## Cambiar el color

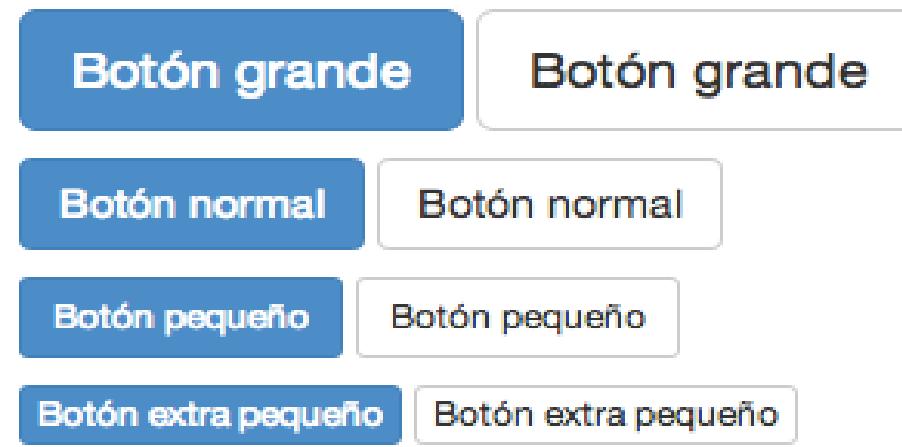
```
<button type="button" class="btn btn-default">Normal</button>
<button type="button" class="btn btn-primary">Destacado</button>
<button type="button" class="btn btn-success">Éxito</button>
<button type="button" class="btn btn-info">Información</button>
<button type="button" class="btn btn-warning">Advertencia</button>
<button type="button" class="btn btn-danger">Peligro</button>
<button type="button" class="btn btn-link">Enlace</button>
```



# Formularios - Botones

## Cambiar el tamaño

```
<button type="button" class="btn btn-primary btn-lg">Botón grande</button>
<button type="button" class="btn btn-default btn-lg">Botón grande</button>
<button type="button" class="btn btn-primary">Botón normal</button>
<button type="button" class="btn btn-default">Botón normal</button>
<button type="button" class="btn btn-primary btn-sm">Botón pequeño</button>
<button type="button" class="btn btn-default btn-sm">Botón pequeño</button>
<button type="button" class="btn btn-primary btn-xs">Botón extra pequeño</button>
<button type="button" class="btn btn-default btn-xs">Botón extra pequeño</button>
```



# Formularios - Botones

---

## Grupos de botones

- Para agrupar varios botones relacionados entre sí para mostrarlos en una única línea.

```
<div class="btn-group">
 <button type="button" class="btn btn-default">Izquierdo</button>
 <button type="button" class="btn btn-default">Central</button>
 <button type="button" class="btn btn-default">Derecho</button>
</div>
```



# **Formularios - Botones**

---

## **Botones desplegables**

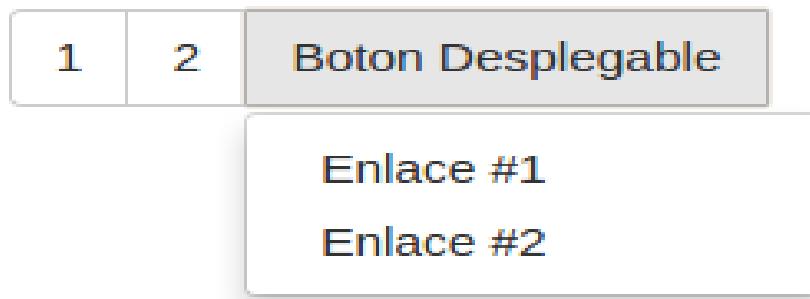
- También se permite mostrar menús desplegables dentro de los grupos de botones.
- Para ello:
  - Incluir un elemento con la clase `.btn-group` dentro de otro elemento con la clase `.btn-group`
  - Añadir propiedad `data-toggle="dropdown"`
  - La lista debe ser: `<ul class="dropdown-menu">`

# Formularios - Botones

## Botones desplegables

```
<div class="btn-group">
 <button type="button" class="btn btn-default">1</button>
 <button type="button" class="btn btn-default">2</button>
 <div class="btn-group">
 <button type="button" class="btn btn-default" data-toggle="dropdown">
 Boton Desplegable
 </button>
 <ul class="dropdown-menu">
 Enlace #1
 Enlace #2

 </div>
```

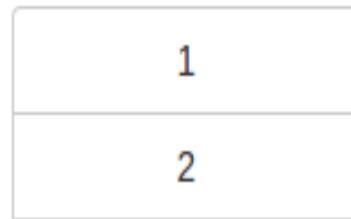


# Formularios - Botones

## Grupos de botones verticales

- Aplica la clase `.btn-group-vertical` sobre un grupo de botones para mostrarlos verticalmente en vez de con su estilo horizontal habitual

```
<div class="btn-group-vertical">
 <button type="button" class="btn btn-default">1</button>
 <button type="button" class="btn btn-default">2</button>
</div>
```



## Formularios - Grupos de campos tipo texto

---

- Los campos de tipo texto se pueden modificar para que muestren un texto o un botón delante, detrás o a ambos lados.
- Para ello, se agrupa el campo dentro de un elemento con la clase `.input-group` y se marca el elemento que va delante o detrás con la clase `.input-group-addon`.

```
<div class="input-group">
 $
 <input type="text" class="form-control">
 .00
</div>
```



pue

## Iconos (glyphicon)

---

- Bootstrap incluye 180 iconos creados mediante una fuente especial llamada Glyphicon Halflings.
- Aunque esta fuente normalmente no es gratuita, su creador permite utilizar estos iconos gratuitamente dentro de Bootstrap.
- Solo piden que, si es posible, incluyas en tu sitio un enlace al proyecto Glyphicons.
- El listado completo de estos Iconos esta en:  
<http://getbootstrap.com/components/>

## Iconos (glyphicon)

---

- Para usarlos se requiere de una clase CSS común para todos y de una clase CSS específica para cada uno.

```

```

- Como los iconos son vectoriales, quedan bien en cualquier elemento y a cualquier tamaño. Utilízalos en botones, barras de navegación o incluso en campos de formulario.

```
<button type="button" class="btn btn-default btn-lg">
 Star
</button>
```



```
<button type="button" class="btn btn-default">
 Star
</button>
```

pue

# Etiquetas

---

```
Por defecto
Principal
Éxito
Información
Aviso
Peligro
```

Por defecto

Principal

Éxito

Información

Aviso

Peligro

pue

# Ejercicio

---

- Crear una página que contenga la estructura básica de HTML5 con los siguientes componentes:
  - 1) Cabecera con nombre de EMPRESA
  - 2) Barra de navegación Horizontal con 2 menus desplegables
    - el primero con 2 enlaces, el primero a un formulario y el segundo a una calculadora
    - el segundo con 1 enlace, a una página que tenga 2 frases incompletas, que se completan al pinchar
  - 3) Sección principal con un articulo relacionado con la actividad de la empresa y otro articulo a su derecha con una foto
  - 4) Acabar con un footer con el copyright

# **Programando en HTML 5 y CSS 3**

---

## **Ejercicio → formulario**

El formulario debe:

- Seguir mostrando la barra de navegación.
- Situado en el centro de la página
- que pida los siguientes datos:
  - Nombre: debe empezar por una letra
  - Password: mínimo 8 caracteres
  - Edad:entre 0 y 110
  - fecha de nacimiento, email, sexo
- TODAS ELLAS OBLIGATORIAS
  - Teléfono fijo: empieza por 8 o 9
  - Teléfono móvil: empieza por 6 o 7
  - comenario
- Un botón para limpiar los campos
- otro para enviar que nos envie a una pagina que ponga “enviado correctamente”

---

# **Módulo 9:**

## **Conociendo más**

## **herramientas útiles**

pue

# API history - Introducción

---

- El API history de HTML es la manera estándar de manipular el historial de navegación a través de JavaScript.
- HTML5 incluye una nueva manera de añadir entradas al historial de navegación, modificando la URL pero sin actualizar la página actual
- Eventos que se disparan cuando el usuario vuelve a través del historial de navegador.
- La barra de direcciones sigue funcionando de la misma manera, cacheando los recursos de manera única, aún cuando las aplicaciones hacen un uso intensivo de JavaScript sin recargar la página.

# API history - Introducción

---

- La URL realmente importa.
- Representa un recurso único. Podemos enlazarlo directamente, almacenarlo como favorito, los motores de búsqueda pueden analizar su contenido, podemos copiarlo y enviarlo por email ....
- Así pretendemos es que contenidos únicos dispongan de una URL única.

## API history - Navegadores

---

- Hasta ahora, el comportamiento normal de los navegadores es recargar la página si modificábamos la URL.
- Realizar una nueva petición y obtener de nuevo todos los recursos del servidor.
- Hasta ahora no había manera de decir al navegador que cambiase la URL pero descargase únicamente la parte de la página.

# API history - Navegadores

---

- El API history de HTML5 permite precisamente esto. En lugar de solicitar la carga de toda la página, podemos utilizar JavaScript para cargar únicamente los contenidos que deseemos.
- Imaginemos que tenemos una página A y otra página B, que comparten el 90% de su contenido.
- Cuando desde la página A, se quiere navegar a la B, en lugar de realizar una petición al servidor, interrumpimos esta navegación y realizamos los siguientes pasos de manera manual:

## API history - Navegadores

---

- 1) Cargar el 10% de contenido diferente de la página B, a través de algún método como AJAX.
- 2) Cambiar el contenido, utilizando innerHTML u otros métodos del DOM.
- 3) Actualizar la URL del navegador, indicando la dirección de la página B, utilizando el API history de HTML5.

Tras realizar estos pasos, disponemos de un DOM exacto al de la página B, como si hubiésemos navegado hasta ella, pero sin realizar una petición completa.

# API history - Métodos

## Moverse por el historial

---

- Para moverte hacia atrás, como si se hiciera clic en el botón "atrás" del browser.  
`window.history.back();`
- De manera similar, podemos movernos hacia adelante:  
`window.history.forward();`
- También podemos movernos a un lugar concreto del historial tanto para delante (valor positivo) como para atrás (valor negativo):

```
window.history.go(1);
```

```
window.history.go(-1);
```

pue

# API history - Métodos

## Número de elementos del historial

- Para saber cuantas páginas hay en el historial de manera muy sencilla:

```
var numberOfEntries = window.history.length;
```

# API history - Métodos pushState() y replaceState()

---

## pushState():

- Modifica el histórico añadiendo una nueva URL pero:
  - **Modifica URL, sin recargar página.**
  - Se puede guardar información de ESTADO que será usado por el navegador al usar el histórico HACIA ADELANTE
  - Si, después de navegar por otros sitios, volvemos hacia atrás, el navegador puede necesitar cargar la página.

## replaceState():

- Igual que pushState excepto que sustituye en el histórico la nueva URL por la antigua, aunque el contenido mostrado es el de la página antigua

pue

# API history - Métodos pushState() y replaceState()

- Sintaxis:

```
var stateObj = { sms: "historico actualizado con pushState" };
window.history.pushState(stateObj, "hago que voy a pagina 3", "35-historyAPI_3.html");
```

```
var stateObj = { sms: "historico actualizado con replaceState" };
window.history.replaceState(stateObj, "hago que voy a pagina 3", "35-historyAPI_3.html");
```

Argumento1 → objeto con información disponible por el navegador al acceder de nuevo a la página a través del histórico

Argumento2 → futura implementación como título de la ventana

Argumento3 → URL a incluir en el histórico

pue

# API history - Ejemplo paso a paso

Código HTML PAGINA 1:

```
<body>
 <div class="container">
 <div class="row">
 <div class="col-md-6">
 <a href="//localhost:8000/Documentos/MisCursos/HTML5_CSS_JavaScript-moviles/curso/35-historyAPI_2.html"
 >hola mundo Pagina 1, si me haces click voy a pagina2
 </div>
 <div class="col-md-12">
 <p> </p>
 </div>
 <div class="col-md-12">
 <input type="button" class="btn btn-primary" id="atras" value="«">
 <input type="button" class="btn btn-primary" id="alante" value="»">
 </div>
 <div class="col-md-12">
 <p> </p>
 </div>
 <div class="col-md-12">
 <input type="button" class="btn btn-warning" id="pushState" value="Pagina 3???">
 <input type="button" class="btn btn-danger" id="replaceState" value="Pagina 3???">
 </div>
 <div class="col-md-12">
 <p> </p>
 </div>
 </div>
 </div>
</body>
```

# API history - Ejemplo paso a paso

---

Código HTML PAGINA 2:

```
<body>
<div class="container">
 <div class="row">
 <div class="col-md-6">
 <a href="//localhost:8000/Documentos/MisCursos/HTML5_CSS_JavaScript-moviles/curso/35-historyAPI_1.html"
 >hola mundo Pagina 2, si me hacer click voy a pagina 1
 </div>
 <div class="col-md-12">
 <p> </p>
 </div>
 <div class="col-md-12">
 <input type="button" class="btn btn-primary" id="atras" value="&lquo;">
 <input type="button" class="btn btn-primary" id="alante" value="&raqo;">
 </div>
 <div class="col-md-12">
 <p> </p>
 </div>
 <div class="col-md-12">
 <input type="button" class="btn btn-warning" id="pushState" value="Pagina 3???">
 <input type="button" class="btn btn-danger" id="replaceState" value="Pagina 3???">
 </div>
 <div class="col-md-12">
 <p> </p>
 </div>
 </div>
</div>
</body>
```

# API history - Ejemplo paso a paso

Código HTML PAGINA 3:

```
<body>
 <div class="container">
 <div class="row">
 <div class="col-md-6">
 PAGINA 3, de aqui no me muevo !!!
 </div>
 <div class="col-md-12">
 <p> </p>
 </div>
 <div class="col-md-12">
 <input type="button" class="btn btn-primary" id="atras" value="&lquo;">
 <input type="button" class="btn btn-primary" id="alante" value="»">
 </div>
 <div class="col-md-12">
 <p> </p>
 </div>
 <div class="col-md-12">
 <input type="button" class="btn btn-warning" id="pushState" value="Pagina 3???">
 <input type="button" class="btn btn-danger" id="replaceState" value="Pagina 3???">
 </div>
 <div class="col-md-12">
 <p> </p>
 </div>
 <div class="col-md-12">
 <p> hola a todos, estamos realmente en la pagina 3 </p>
 </div>
 </div>
 </div>
</body>
```

# API history - Ejemplo paso a paso

Código JavaScript:

```
$(document).ready(function(){
 // Cada vez que cargamos la página
 console.log("Pagina Recargada");
 console.log("historico actual guardado: " + window.history.length++);
 // Evento que se ejecuta al incluir una nueva página al historico
 // Guarda objeto --> StateObj con el contenido de la página
 window.onpopstate = function(event) {
 $('body').append($("<p class='col-md-12 text-warning bg-danger'>" + event.state.sms + "</p>"));
 };
 // movernos por el historico
 $('input#atras').on('click',function(){
 window.history.back();
 });
 $('input#alante').on('click',function(){
 window.history.forward();
 });
 // modificar el historico
 $('input#pushState').on('click',function(){
 var stateObj = { sms: "historico actualizado con pushState" };
 window.history.pushState(stateObj, "hago que voy a pagina 3", "35-historyAPI_3.html");
 });
 $('input#replaceState').on('click',function(){
 var stateObj = { sms: "historico actualizado con replaceState" };
 window.history.replaceState(stateObj, "hago que voy a pagina 3", "35-historyAPI_3.html");
 });
});
```

pue

# API history - Ejemplo paso a paso

URL inicio:

localalhost:8000/Documentos/MisCursos/HTML5\_CSS\_JavaScript-moviles/curso/**35-historyAPI\_1.html**

hola mundo Pagina 1, si me haces click voy a pagina2



Pagina 3??

Pagina 3??

🚫 ⚏ <top frame> ▾  Preserve log

Pagina Recargada

historico actual guardado: 1



- Pinchamos para ir a la página 2

pue

# API history - Ejemplo paso a paso

URL inicio:

localhost:8000/Documentos/MisCursos/HTML5\_CSS\_JavaScript-moviles/curso/**35-historyAPI\_2.html**

hola mundo Pagina 2, si me hacer click voy a pagina 1



Pagina 3??      Página 3??



- Pinchamos para ir a la página 3?? NARANJA

pue

# API history - Ejemplo paso a paso

URL inicio:

localhost:8000/Documentos/MisCursos/HTML5\_CSS\_JavaScript-moviles/curso/**35-historyAPI\_3.html**

hola mundo Pagina 2, si me hacer click voy a pagina 1



Pagina 3?? Página 3??

Preserve log

Página Recargada

histórico actual guardado: 2



- Solo ha cambiado la URL, pero mantenemos el contenido de página 2
- No se recarga la página, en consola no vemos que hay 3 en el histórico
- Pinchamos en « para volver en el histórico al principio y comprobar que están las 3

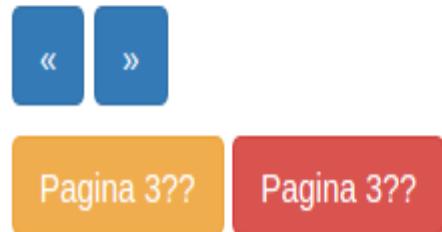
pue

# API history - Ejemplo paso a paso

- URL inicio:

localhost:8000/Documentos/MisCursos/HTML5\_CSS\_JavaScript-moviles/curso/**35-historyAPI\_1.html**

hola mundo Pagina 1, si me haces click voy a pagina2



- Vemos en consola que están las 3 en el histórico.
- Pinchamos en para avanzar en el histórico al hasta la supuesta “página 3” .

pue

# API history - Ejemplo paso a paso

URL inicio:

localhost:8000/Documentos/MisCursos/HTML5\_CSS\_JavaScript-moviles/curso/35-historyAPI\_3.html

- Podemos ver El mensaje de estado guardado al actualizar el histórico.

The screenshot shows a web page with the following content:

hola mundo Pagina 2, si me hacer click voy a pagina 1

« »

Pagina 3?? Página 3??

historico actualizado con pushState

Historical log (top frame):

- Página Recargada
- histórico actual guardado: 3

- Pinchamos para ir a la página 1 y la página 2

pue

# API history - Ejemplo paso a paso

- URL inicio:

localhost:8000/Documentos/MisCursos/HTML5\_CSS\_JavaScript-moviles/curso/35-historyAPI\_2.html

hola mundo Pagina 2, si me hacer click voy a pagina 1



- Pinchamos para ir a la pagina 3?? ROJO

pue

# API history - Ejemplo paso a paso

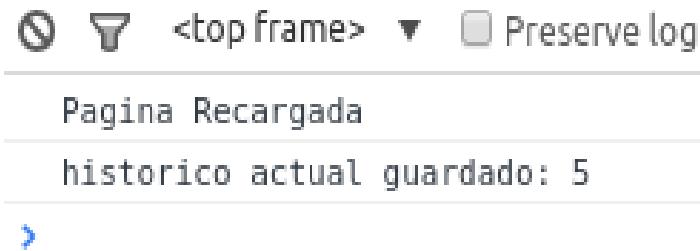
URL inicio:

localhost:8000/Documentos/MisCursos/HTML5\_CSS\_JavaScript-moviles/curso/**35-historyAPI\_3.html**

hola mundo Pagina 2, si me hacer click voy a pagina 1



Pagina 3??    Página 3??



- Solo ha cambiado la URL, pero mantenemos el contenido de página 2
- En el histórico se ha sustituido la página 2 por la página 3, por lo tanto sigue habiendo 5.

pue

## **API history - Ejemplo paso a paso**

---

- Pinchamos de nuevo para ir a la página 1 y 2 de nuevo.
- Tenemos 7 páginas en el histórico
- Si vamos para atrás en el histórico se recargará la página 3
- Si vamos para delante vemos que se carga el objeto estado pero no la página.

# API Canvas - Introducción

---

- Canvas (lienzo en inglés) es un elemento nuevo en HTML5 que permite la generación de gráficos por medio del scripting.
- Permite independizarse de flash o java applets para poder realizar tareas como:
  - > dibujar gráficos
  - > animaciones
  - > procesar texto e imágenes
  - > desarrollar videojuegos

# API Canvas - Etiqueta HTML5

---

- Divide el lienzo en pixeles en eje X e Y empezando en (0,0)
- Cualquier texto dentro de este elemento se mostrará solo en los navegadores que no soportan el canvas.
- Tiene 2 propiedades fundamentales, ancho y largo que determinan los límites del lienzo

```
<section id="cajalienzol" style="display:inline-block">
 <canvas id="lienzo1" weight="300" height="250" style="border: 1px solid black">
 <p>Aqui esta nuestro lienzo</p>
 </canvas>
</section>
```

# API Canvas - Contexto

---

- Primero debemos referenciar el elemento canvas y adquirir su contexto.
- Una vez adquirimos el contexto, podemos empezar a dibujar en la superficie trazos, rellenos, gradientes, sombras, formas y curvas Bézier usando métodos de JavaScript.
- Para obtener el contexto :

```
var elemento = document.getElementById('lienzo1');

miLienzo = elemento.getContext('2d');
```

# API Canvas - Figuras

---

- Para dibujar figuras poseemos varios métodos.
- Los argumentos que necesitan son:
  - ( `inicio_pos_x, inicio_pos_y, cuanto_pos_x, cuanto_pos_y` )
- Para dibujar un cuadrado relleno:

```
miLienzo.fillRect(1,1,80,40);
```

- Para dibujar un cuadrado vacío:

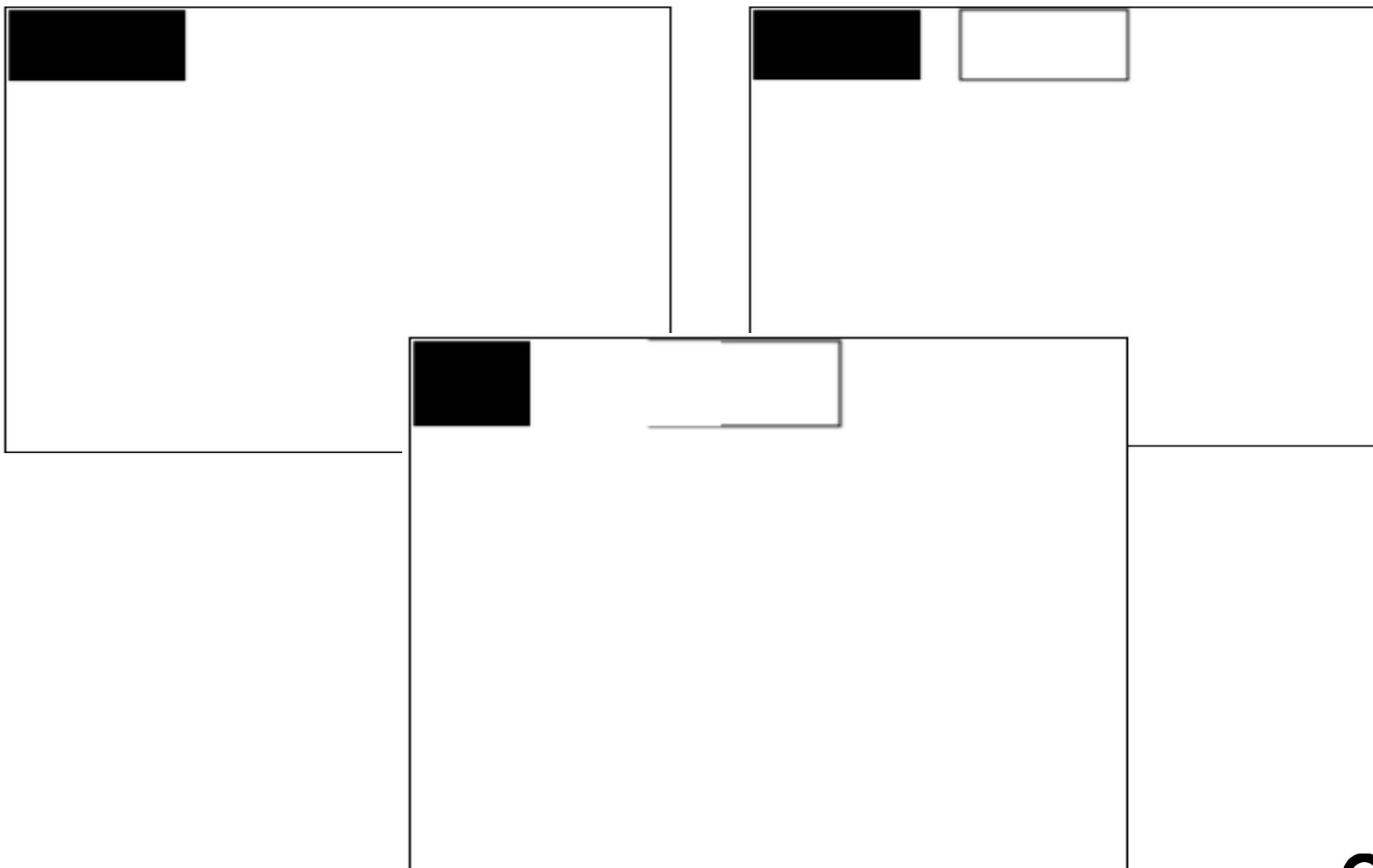
```
miLienzo.strokeRect(100,1,80,40);
```

- También disponemos de un borrador:

```
miLienzo.clearRect(50,1,80,40);
```

# API Canvas - Figuras

---



pue

# API Canvas - Estilos

---

## Colores y transparencia

- Para cambiar el color negro de por defecto en los cuadrados rellenos:

```
|miLienzo2.strokeStyle="Red";
```

- Para cambiar el color gris por defecto en los cuadrados sin relleno:

```
miLienzo2.fillStyle="Blue";
```

- Para modificar la transparencia aplicada a los elementos dibujados en el lienzo:

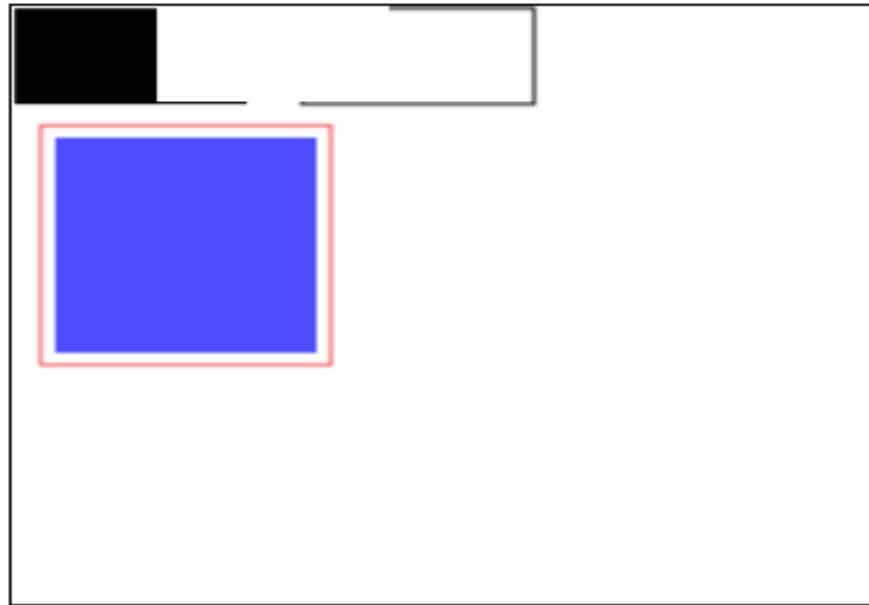
```
miLienzo2.globalAlpha="0.7";
```

# API Canvas - Estilos

---

## Colores y transparencia

```
miLienzo.globalAlpha="0.7";
miLienzo.strokeStyle="Red";
miLienzo.strokeRect(10,50,100,100);
miLienzo.fillStyle="Blue";
miLienzo.fillRect(15,55,90,90);
```



# API Canvas - Estilos

---

## Gradientes

- Podemos establecer un **gradiente lineal**, a través de un objeto que establezca:
  - las coordenadas de inicio de los 2 colores que forman el gradiente:

```
var gradienteLineal = miLienzo2.createLinearGradient(0,10,20,50);
```

- El final de los colores del gradiente, junto con los colores que lo forman:

```
gradienteLineal.addColorStop(0.5, "orange");
gradienteLineal.addColorStop(1, "red");
```

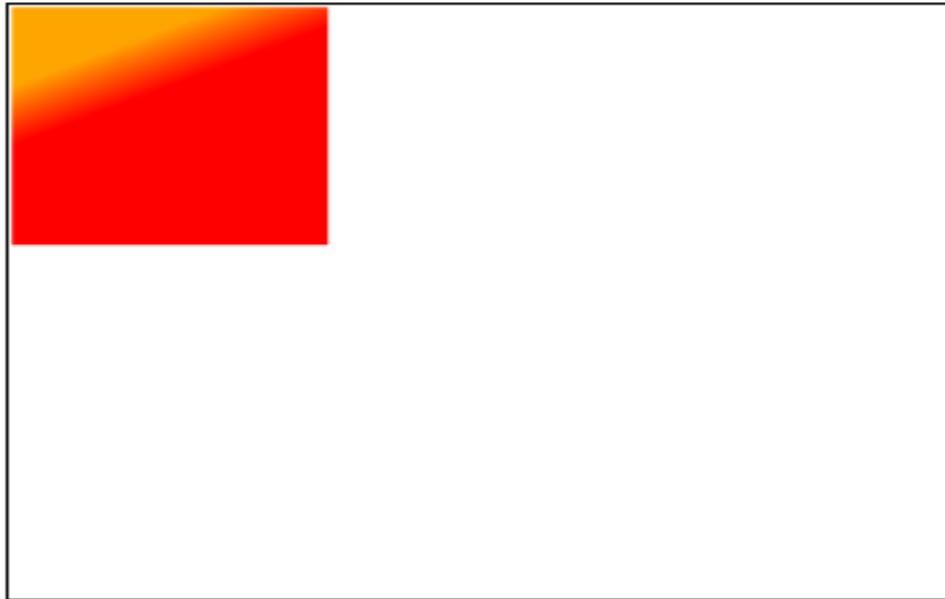
- Para finalizar, establecemos el estilo al objeto y pintamos el objeto:

```
miLienzo2.fillStyle = gradienteLineal;
miLienzo2.fillRect(1,1,100,100);
```

# API Canvas - Estilos

---

## Gradientes



pue

# API Canvas - Estilos

---

## Gradientes

- De forma similar podemos establecer un **gradiente Radial**, a través de un objeto que establezca:
  - las coordenadas de inicio y Radio de los 2 colores que forman el gradiente:

```
var gradienteRadial = miLienzo2.createRadialGradient(0,10,10,10,20,100);
```

- El final de los colores del gradiente, junto con los colores que lo forman:

```
gradienteRadial.addColorStop(0.5,"orange");
gradienteRadial.addColorStop(1,"red");
```

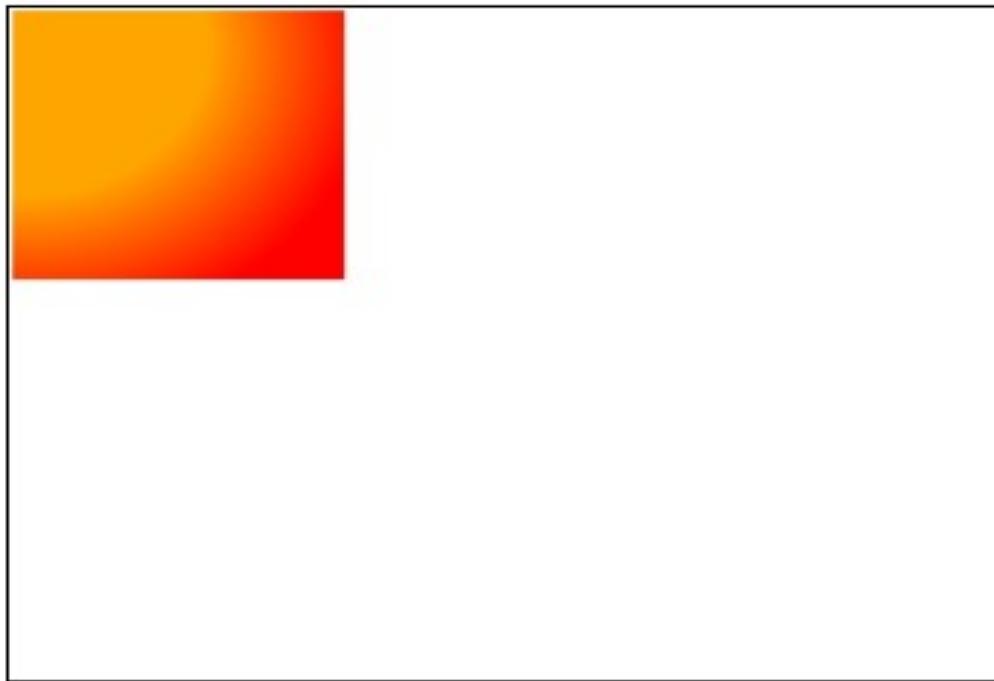
- Para finalizar, establecemos el estilo al objeto y pintamos el objeto:

```
miLienzo2.fillStyle = gradienteRadial;
miLienzo2.fillRect(1,1,100,100);
```

# API Canvas - Estilos

---

## Gradientes



pue

## **API Canvas - Dibujar con “lápiz”**

---

- Disponemos de diferentes métodos que nos permitirán situarnos y dibujar simulando el movimiento de un lápiz.
- Técnica muy potente para crear multitud de figuras y formas, si bien toda esta potencia depende de nuestra imaginación imaginación.
- Los principales métodos son:

# API Canvas - Dibujar con “lápiz”

---

`beginPath();`

- Situa el lápiz en el lienzo
- Marca el inicio de la pintura “con el lápiz”

`lineTo(X_fin,Y_fin);`

- Dibuja una línea desde donde esta el lápiz hasta donde indiquemos con la función.

`moveTo(X,Y);`

- Para levantar el “lápiz” y moverlo a otra posición.

`rect(X,Y,ancho,alto)`

- Para crear un rectangulo.

pue

# API Canvas - Dibujar con “lápiz”

---

`arc(X,Y,radio,angulo_inicio,angulo_fin,direccion)`

- Permite dibujar curvas o círculos

`closePath();`

- Finaliza el dibujo del “lápiz” con linea desde donde acabo ultimo trazo al inicio

`Fill();`

- Permite, si en los trazos hemos creado figuras cerradas, llenar el dibujo.

`stroke();`

- Permite determinar la anchura del trazo realizado

# API Canvas - Dibujar con “lápiz”

---

## Ejemplos paso a paso

### Dibujar un **cuadrado**

```
miLienzo2.beginPath();
miLienzo2.moveTo(150,50);
miLienzo2.lineTo(150,100);
miLienzo2.lineTo(200,100);
miLienzo2.lineTo(200,50);
miLienzo2.closePath();
miLienzo2.fill();
```

### Dibujar un **trinagulo**

```
miLienzo2.beginPath();
miLienzo2.moveTo(150,100);
miLienzo2.lineTo(150,150);
miLienzo2.lineTo(200,150);
miLienzo2.closePath();
miLienzo2.stroke();
```

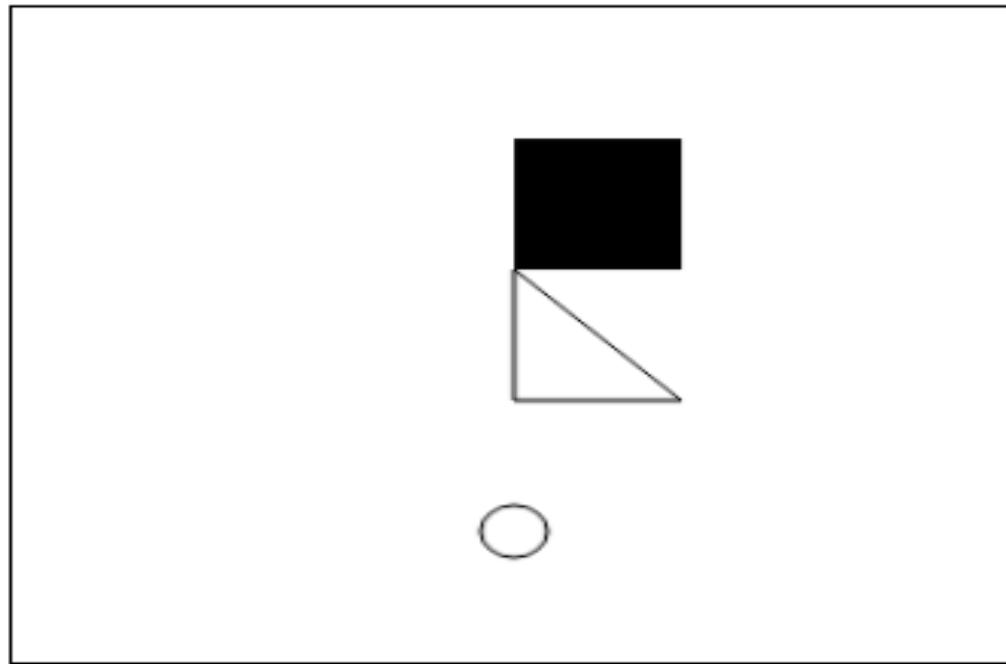
### Dibujar un **ciruclo**

```
miLienzo2.beginPath();
miLienzo2.arc(150,200,10,0,360);
miLienzo2.closePath();
miLienzo2.stroke();
```

# API Canvas - Dibujar con “lápiz”

---

## Ejemplos paso a paso



pue

# API Canvas - Dibujar con “lápiz”

---

## Ejemplos paso a paso

Dibujar un tablero de tres en raya

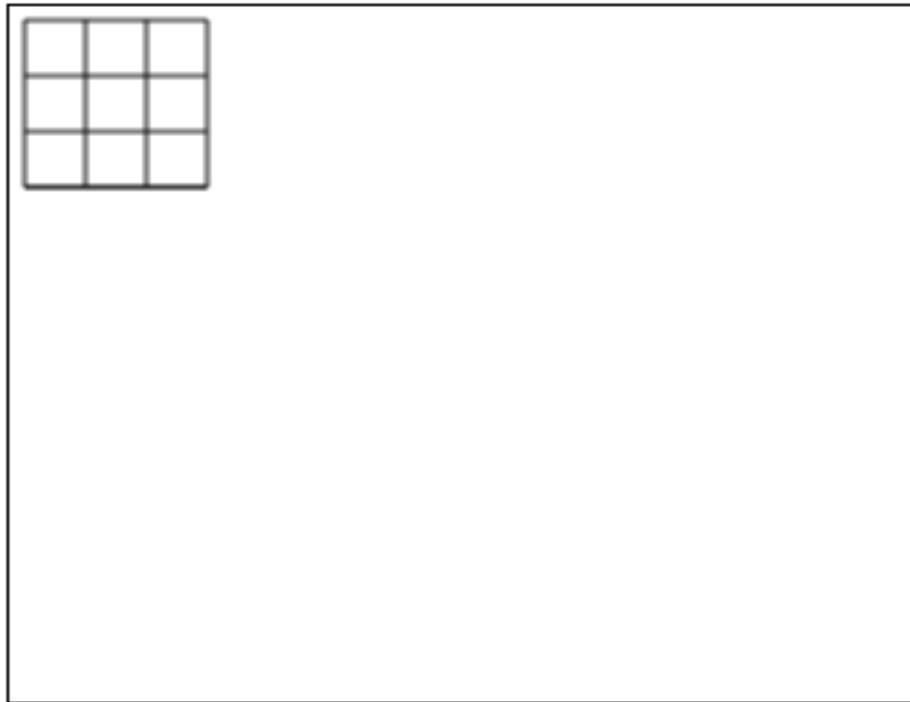
```
// Iniciamos el lápiz
elemento3.beginPath();
// líneas horizontales moviendo y pintando
for (x=5; x<=65; x=x+20){
 elemento3.moveTo(x,5);
 elemento3.lineTo(x,65);
}
// líneas verticales moviendo y pintando
for (y=5; y<=65; y=y+20){
 elemento3.moveTo(5,y);
 elemento3.lineTo(65,y);
}
// Finalizar el tablero
elemento3.closePath();
elemento3.stroke();
```

pue

# API Canvas - Dibujar con “lápiz”

---

## Ejemplos paso a paso



pue

## API Canvas - Estilos del “lápiz”

---

- Disponemos de diferentes propiedades que nos permitirán personalizar diferentes aspectos de las líneas creadas con el lápiz:

`strokeStyle="DarkOrange"` → Para establecer el color

`lineWidth=20` → Para establecer el ancho

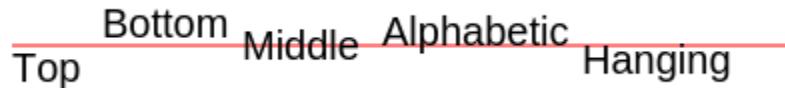
`lineCap="round"` → Para redondear los extremos

# API Canvas - Insertar Texto

---

- También podemos incluir texto en nuestro lienzo.
- Empezaremos alineando el texto con las siguientes propiedades:

**textBaseline** → alineación horizontal



**textAling** → alineación vertical

A vertical red line with five labels to its right: "textAlign=start", "textAlign=end", "textAlign=left", "textAlign=center", and "textAlign=right".

pue

## API Canvas - Insertar Texto

---

- Continuando por establecer la fuente y el tamaño:  
`font="15px Verdana";`
- Finalizando por la propia escritura, con el método  
`fillText`, que recibe como parámetros, el Texto y las  
coordenadas de inicio:  
`fillText("mi texto",25,105);`

# API Canvas - Insertar Texto

## Ejemplo Escribir un texto y subrayarlo

- Comenzamos dibujando una línea que actuará de subrayado modificando su estilo:
  - anchura
  - color
  - Redondeo de extremos

```
// iniciamos lapiz
elemento3.beginPath();
// movemos
elemento3.moveTo(25,100);
// cambiamos color
elemento3.strokeStyle="darkorange";
// cambiamos ancho
elemento3.lineWidth=20;
// linea con redondeo en extremos
elemento3.lineCap="round";
// pintamos linea
elemento3.lineTo(75,100);
// finalizar linea
elemento3.closePath();
elemento3.stroke();
```

# API Canvas - Insertar Texto

---

## Ejemplo Escribir un texto y subrayarlo

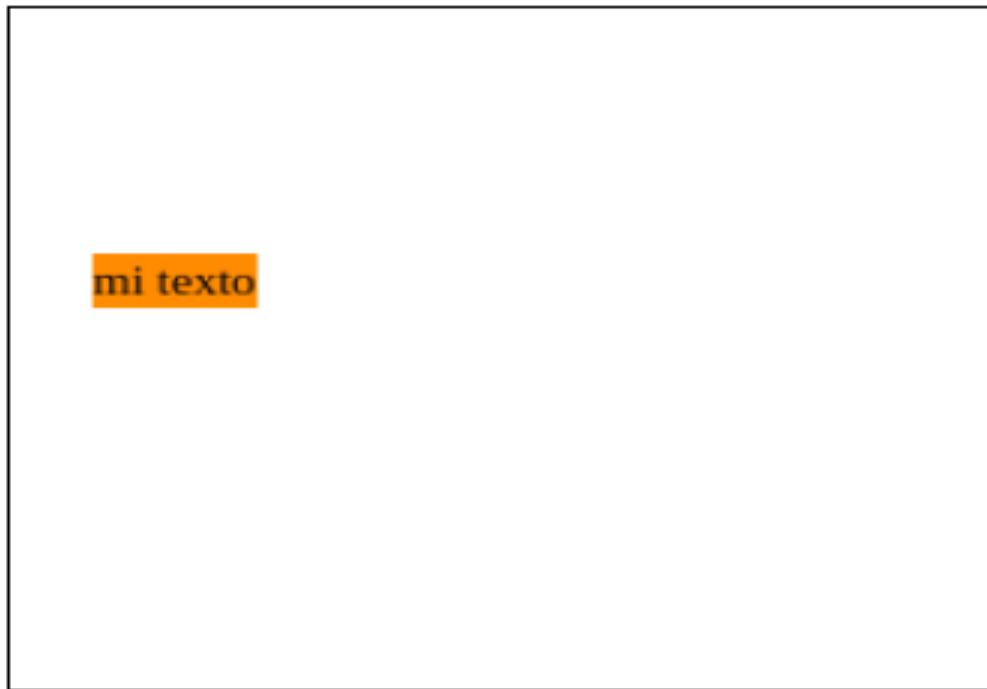
- Terminamos con la propia escritura donde estableceremos:
  - Fuente
  - Alineación

```
// Fuente y tamaño
elemento3.font="15px Verdana";
// alineacion
elemento3.textAlign="start";
elemento3.textBaseline="center";
// El propio texto
elemento3.fillText("mi texto",25,105);
```

# API Canvas - Insertar Texto

---

## Ejemplo Escribir un texto y subrayarlo



mi texto

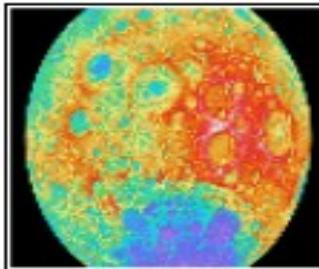
pue

# API Canvas - Insertar Imagenes

---

- Para insertar una imagen que ya existe es necesario:
- Instanciar un **objeto de la clase Image**
- Cuando se cargue la imagen llamar al método **drawImage()** para situarla en el lienzo

```
var imagen = new Image();
imagen.src='fotos/luna_de_colores.jpg';
imagen.onload = function(){
 //empieza en 1,1, acaba en 100,100
 miLienzo4.drawImage(imagen,1,1,100,100);
};
```



pue

# API Canvas - Práctica

---

- Dibujar un tablero de ajedrez con:
  - Posiciones numeradas
  - Posiciones de colores blanco y marrón.

# Caché Offline - Introducción

---

- HTML5 ofrece un mecanismo de almacenamiento en caché de las aplicaciones que permite que se ejecuten aunque no estés conectado a la red.
- Los desarrolladores pueden utilizar la interfaz de aplicaciones Caché (AppCache) para especificar los recursos que el navegador debe cachear y poner a disposición de los usuarios sin conexión.
- Las aplicaciones que se almacenan en caché se cargan y funcionan correctamente, incluso si los usuarios hacen clic en el botón Actualizar cuando están descobectados.

# Caché Offline - Habilitar la AppCache

---

- Se debe incluir el atributo manifest en el elemento `<html>` en las páginas de la aplicación, que es un archivo de texto que enumera los recursos (archivos) que el navegador debe almacenar en caché para su aplicación, como se muestra en el siguiente ejemplo:

```
<html manifest="example.appcache">
 ...
</html>
```

- No es necesario enumerar todas las páginas que desea en caché en el archivo de manifiesto.
- El navegador añade implícitamente todas las páginas que el usuario visita y que tiene el atributo manifest establecido en la caché de la aplicación.

# Caché Offline - Habilitar la AppCache

---

- El uso de AppCache modifica el proceso normal de carga de un documento:
  - Si existe una caché de la aplicación, el navegador carga el documento y sus recursos desde la caché
  - Esto acelera el tiempo de carga de documentos.
- Antes, El navegador comprueba si el manifiesto de caché se ha actualizado en el servidor.
  - Para descargar una nueva versión del manifiesto y los recursos que aparecen en el manifiesto.
  - Esto se hace en segundo plano y no afecta al rendimiento significativamente.

# Caché Offline - Fichero manifest

---

- Un manifiesto puede tener tres secciones distintas:

## 1) CACHE:

Esta es la sección predeterminada.

Los archivos listados se almacenan en caché de forma explícita después de ser descargado por primera vez.

## 2) LA RED:

Para los archivos que requieren una conexión con el servidor.

## 3) FALLBACK:

Especifica dos URI

→ PRIMERO: posible recurso inaccesible

→ SEGUNDO: página a usar si la primera no está disponible.

Ambos URI deben ser relativas y desde el mismo origen que el archivo de manifiesto. pue

# Caché Offline - Compatibilidad con navegadores

Mobile						
Feature	Android	Firefox Mobile (Gecko)	Firefox OS	IE Mobile	Opera Mobile	Safari Mobile
Basic support	2.1	(Yes)	1.0.1	11.0	11.0	3.2
Desktop						
Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari	
Basic support	4.0	3.5	10.0	10.6	4.0	

pue

# Caché Offline - Ejemplos de Fichero

---

```
CACHE MANIFEST
v1 - 2011-08-13
Esto es un comentario.
http://www.ejemplo.com/index.html
http://www.ejemplo.com/encabezado.png
http://www.ejemplo.com/blah/blah
```

- Los navegadores solamente actualizan el caché de aplicación cuando el archivo de manifest cambia byte por byte.
- Se puede hacer cualquier cambio al archivo de manifest, pero cambiar el número de versión es una práctica recomendada.

# Caché Offline - Ejemplos de Fichero

---

```
CACHE MANIFEST
v1 2011-08-14
Este es otro comentario
index.html
cache.html
style.css
image1.png

Usar desde la red si está disponible
NETWORK:
network.html

Contenido de fallback
FALLBACK:
/fallback.html
```

- La página network.html que deberá ser recuperada desde la red
- La página fallback.html se usará en caso que una conexión al servidor no pueda establecerse.

# API Geolocalización - Métodos

---

- Nos permite conocer la ubicación geográfica del usuario, siempre y cuando esté usando un navegador que la tenga implementada y que el usuario dé su permiso.
  - Aunque la primera impresión sea que sólo será útil para usuarios de navegadores móviles, la realidad es que éste utiliza otros medios además del GPS para calcular la ubicación del usuario, como por ejemplo a través de su dirección IP.

# API Geolocalización - Introducción

---

- Nos permite conocer la ubicación geográfica del usuario, siempre y cuando esté usando un navegador que la tenga implementada y que el usuario dé su permiso.
- Aunque la primera impresión sea que sólo será útil para usuarios de navegadores móviles, la realidad es que éste utiliza otros medios además del GPS para calcular la ubicación del usuario, como por ejemplo a través de su dirección IP.

# API Geolocalización - Métodos

---

- El objeto **navegator** implementa el objeto **geolocation** que aporta la funcionalidad.
- Los 2 métodos, pertenecientes al objeto geolocation, que permiten obtener la posición son:
  - **getCurrentPosition()** : Obtiene los datos de la posición **SOLO en el momento de su invocación** .
  - **watchPosition()** : Obtiene y **mantiene actualizados** los datos de posición en el momento de su invocación.
- Ambos métodos comparte:
  - 1) Argumentos
  - 2) Como devuelven la posición

# API Geolocalización - Argumentos

---

- Primer Argumento:
  - Función que recibe la posición en el `objeto coords`.
- Segundo Argumento:
  - Función que recibe el error en caso de haberse producido.
- Tercer Argumento:
  - Objeto para personalizar la obtención de la posición con las siguiente propiedades:
    - `enableHighAccuracy`: true // lo mas preciso posible
    - `timeout`: 1000, // tiempo maximo para obtener localizacion
    - `maximumAge`: 600 // cada cuanto tiempo pide la nueva localizacion(`watchPosition`)

# API Geolocalización - Objeto coords

---

- Almacena la posición obtenida en las siguientes propiedades:
  - **latitude** : Latitud en grados
  - **longitude** : Longitud en grados
  - **accuracy** : Exactitud en Metros

# API Geolocalización - Mostrar mapa

---

## Ejemplo paso a paso

- A partir del siguiente código HTML donde tenemos un botón, que cuando lo pinchemos mostrara el mapa en la sección

```
<body>
 <div class="container">
 <div class="row">

 </div>
 <div class="row">
 <div class="col-md-4 col-md-offset-4">
 <input type="button" id="obtener" class="btn btn-info" value="Obtener Ubicacion">
 </div>
 </div>
 <div class="row">

 </div>
 <div class="row">
 <div class="col-md-4 col-md-offset-4">
 <section id="mapa">
 </div>
 </div>
 </div>
 </div>
</body>
```

pue

# API Geolocalización - Mostrar mapa

---

## Ejemplo paso a paso

- Esperamos a cargar la página.
- Al hacer click en el botón:
  - 1) Creamos objeto que configura la obtención de la posición

```
$(document).ready(function(){
 $('input#obtener').on('click',function(){
 var tuneo_posicion = {
 // lo mas preciso posible
 enableHighAccuracy: true,
 // tiempo maximo para obtener localizacion
 timeout: 1000,
 // tiempo que dura cache con localizaciones anteriores(getCurrentPosition)
 maximumAge: 600
 };
 });
});
```

# API Geolocalización - Mostrar mapa

---

## Ejemplo paso a paso

- 2) Obtenemos la posición con `getCurrentPosition()`
  - Definimos segundo argumento: Mostrar error si lo hay
  - pasamos el objeto tuneo como tercer argumento

```
function(elerror){
 // mostrar error si lo hay
 alert("CodigoError: " + elerror.code + " MensajeDeError: " + elerror.message);
},
tuneo);
});
});
```

# API Geolocalización - Mostrar mapa

## Ejemplo paso a paso

Antes de hacer click

Obtener Ubicacion

Después de hacer click

Obtener Ubicacion



pue

# API Geolocalización - Para geolocalización

---

- Cuando usamos el método `watchPosition()`, la posición se esta actualizando periódicamente.

```
geoId = navigator.geolocation.watchPosition(mostrarMapa,siError,tuneo);
```

- Para detener esa actualización tenemos el método `clearWatch()` que recibe como parámetro el objeto devuelto por `watchPosition()`

```
navigator.geolocation.clearWatch(geoId);
```

# **API Geolocalización - API de Google Maps**

---

- Google tiene disponible una API para poder enlazar google maps en nuestras páginas web.
- Algunas características que podemos utilizar son:
  - Usar mapa de satélite o carretera
  - Usar el zoom
  - Incluir el icono de google para reconocer la ubicación

# API Geolocalización - API de Google Maps

---

- Los requisitos para poder usar la API de Google son:
  - 1) incluir la API en nuestro código

```
<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
```

# API Geolocalización - API de Google Maps

---

2) usar Objeto de google map

```
var mapa = new google.maps.Map(lugarMapa, opcionesGoogleMaps);
```

que necesita 2 parámetros:

→ Coordenadas con objeto de Google

```
var lat = laloc.coords.latitude;
var lon = laloc.coords.longitude;
var posicionGoogle = new google.maps.LatLng(lat,lon);
```

→ Opciones de configuración del Mapa de Google

```
opcionesGoogleMaps={
 // Que localizacion
 center: posicionGoogle,
 // zoom
 zoom: 15,
 // el mapa de carretera, el de por defecto en googlemaps
 mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

pue

# API Geolocalización - API de Google Maps

---

3) Poner el Marcador al mapa

→ Necesita haber creado el mapa anteriormente

`new google.maps.Map()` `new google.maps.LatLng()`

```
var marker = new google.maps.Marker({
 position: posicionGoogle,
 map: mapa,
 title: "estas aqui",
 animation: google.maps.Animation.BOUNCE
});
```

# **API Geolocalización**

---

## **Ejercicio**

- Crear una Página con 2 botones:
  - 1) Para Mostrar el mapa de la ubicación actual
  - 2) Para Borrar el mapa de la ubicación actual

# API WebStorage - Introducción

---

- En muchas aplicaciones, para mostrar la información al usuario, el servidor debe procesar dicha información mediante petición/respuesta.
- Debido a la posibilidad de falta de conexión, por ejemplo en lugares sin cobertura, surge la necesidad de:
  - cliente almacene información sin comunicarse con el servidor.

## **Primera solución: cookies**

→ cadenas de texto almacenadas en variables

## **Segunda solución: API WebStorage**

→ Pares de clave/valor

# API WebStorage - Alternativas

---

## Diferencias

- Disponemos de 2 alternativas, sessionStorage y localStorage.
- Ambas comparten el funcionamiento y los métodos pero tienen una diferencia fundamental, el tiempo que dura la información almacenada en el navegador:
- **sessionStorage** : La información queda almacenada hasta que cerremos la pestaña del navegador
- **localStorage** : La información queda almacenada hasta que el usuario decida borrar el historial

# API WebStorage - Métodos

---

## .setItem(clave,valor)

- guarda ese par clave/valor O
- cambia valor para esa clave

```
sessionStorage.setItem(clave.value,valor.value);
```

## .getItem(clave)

- devuelve el valor para la clave conocida

```
v = sessionStorage.getItem(k);
```

## .removeItem(clave)

- borra ese par clave/valor

```
sessionStorage.removeItem(clave.value);
```

# API WebStorage - Métodos

---

- Para recorrer todos los pares clave/valor hay que tratarlo como un Array:

```
for (i=0; i<sessionStorage.length; i++){
 key = sessionStorage.key(i)
 value = sessionStorage[k]
}
```

# API WebStorage

## Ejemplo paso a paso

- A partir de este HTML:
  - 1) Formulario para **incluir** clave/valor

```
<form class="form-inline">
 <div class="form-group">
 <label>Introduce Clave</label>
 <input type="text" class="form-control" id="clave" placeholder="Pon clave">
 </div>
 <div class="form-group">
 <label>Introduce valor</label>
 <input type="text" class="form-control" id="valor" placeholder="Pon valor">
 </div>
 <input type="button" id="guardar" class="btn btn-default" value="Guardar">
</form>
```

# API WebStorage

## Ejemplo paso a paso

- A partir de este HTML:
  - 2) Formulario para **buscar** clave/valor

```
<form class="form-inline">
 <div class="form-group">
 <label>Clave a buscar</label>
 <input type="text" class="form-control" id="claveBuscar">
 </div>
 <div class="form-group">
 <label>Valor encontrado</label>
 <input type="text" class="form-control" id="valorBuscar" disabled>
 </div>
 <input type="button" id="buscar" class="btn btn-default" value="Buscar">
</form>
```

# API WebStorage

## Ejemplo paso a paso

- A partir de este HTML:
  - 3) Formulario para **borrar** clave/valor

```
<form class="form-inline">
 <div class="form-group">
 <label>Clave a borrar</label>
 <input type="text" class="form-control" id="claveBorrar">
 </div>
 <input type="button" id="borrar" class="btn btn-default" value="Borrar">
</form>
```

# API WebStorage

## Ejemplo paso a paso

Obtenemos página similar a esta:

<b>Introduce Clave</b>	<input type="text" value="Pon clave"/>	<b>Introduce valor</b>	<input type="text" value="Pon valor"/>	<b>Guardar</b>
<b>Clave a buscar</b>	<input type="text"/>	<b>Valor encontrado</b>	<input type="text"/>	<b>Buscar</b>
<b>Clave a borrar</b>	<input type="text" value="a"/>	<b>Borrar</b>		

pue

# API WebStorage

## Ejemplo paso a paso

En el depurador partimos de:

The screenshot shows the Chrome DevTools interface with the "Resources" tab selected. On the left, a sidebar lists storage types: Frames, Web SQL, IndexedDB, Local Storage, Session Storage, Cookies, Application Cache, and Cache Storage. Under "Session Storage", there is one entry for the URL "http://localhost:8000". The main area displays a table with two columns: "Key" and "Value". Both columns are currently empty, indicating no data has been stored in session storage for this specific URL.

Key	Value

pue

# API WebStorage

## Ejemplo paso a paso

- Código para **insertar** clave/valor al hacer click en Guardar

```
$('input#guardar').on('click',function(){
 var c = $('input#clave').val();
 var v = $('input#valor').val();
 sessionStorage.setItem(c,v);
});
```

Introduce Clave

Introduce valor

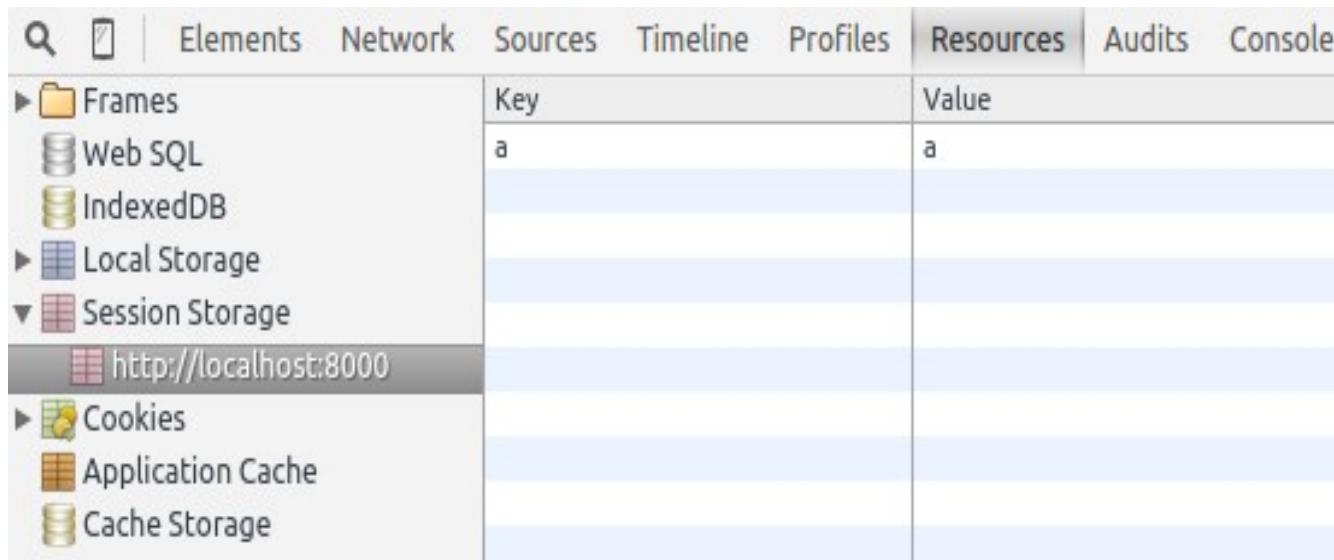
Guardar

pue

# API WebStorage

## Ejemplo paso a paso

El depurador web nos muestra:



The screenshot shows the Chrome DevTools Resources panel. The left sidebar lists storage types: Frames, Web SQL, IndexedDB, Local Storage, Session Storage, Cookies, Application Cache, and Cache Storage. The Session Storage section is expanded, showing a list item for 'http://localhost:8000'. The main table has two columns: 'Key' and 'Value'. A single row is present with 'a' in the Key column and 'a' in the Value column.

	Key	Value
http://localhost:8000	a	a

pue

# API WebStorage

## Ejemplo paso a paso

- Código para **buscar** clave/valor al hacer click en Buscar

```
$('input#buscar').on('click',function(){
 var cbus = $('input#claveBuscar').val();
 var vbus = sessionStorage.getItem(cbus);
 $('input#valorBuscar').attr('value',vbus);
});
```

Clave a buscar

a

Valor encontrado

a

Buscar

pue

# API WebStorage

## Ejemplo paso a paso

- Código para **borrar** clave/valor al hacer click en Borrar

```
$('input#borrar').on('click',function(){
 var cbor = $('input#claveBorrar').val();
 sessionStorage.removeItem(cbor);
 alert("borrado: " + cbor);
});
```

Clave a borrar a|

Borrar

pue

# API WebStorage

## Ejemplo paso a paso

El depurador web nos muestra:

The screenshot shows the Chrome DevTools Resources panel. The left sidebar lists storage types: Frames, Web SQL, IndexedDB, Local Storage, Session Storage, Cookies, Application Cache, and Cache Storage. The Session Storage section is expanded, showing a table for the domain `http://localhost:8000`. The table has two columns: Key and Value. There are four rows in the table, each with a light blue background.

	Key	Value
1		
2		
3		
4		

pue

# **API WebStorage**

## **Ejercicio**

---

- Crear página web con 3 formularios:
  - 1) Para insertar clave/valor
    - Cajas de texto para poner clave/valor
    - Botón para guardar solo activo si la clave y valor tienen valores
  - 2) Para mostrar todas las pares Clave/valor existentes
    - Texto para mostrarlas
    - Botón para buscarlas
  - 3) Para buscar el valor de una clave
    - Texto para insertar clave a buscar
    - Texto para mostrar el valor
    - Botón para buscar clave, solo activo si se introduce clave para buscar

# API Multimedia - Introducción

---

- Hasta hace no mucho tiempo, la tecnología Flash era el dominador indiscutible en el campo multimedia de la web.
- Gracias a esta tecnología es relativamente sencillo transmitir audio y vídeo a través de la red, y realizar animaciones.
- Prácticamente todos los navegadores tienen incorporado un plugin para la reproducción de archivos flash, por lo que la elección era clara.

# API Multimedia - Introducción

---

- Hasta ahora, para incluir un elemento multimedia en un documento, se hacía uso del elemento `<object>`, cuya función es incluir un elemento externo genérico.
- Debido a la incompatibilidad entre navegadores, se hacía también necesario el uso del elemento `<embed>` y duplicar una serie de parámetros.
- Dejando de lado que el código es poco amigable, en este caso tenemos un problema:
- → El navegador tiene que transmitir el vídeo a un plugin instalado en el navegador
- → El usuario, entre otros requisitos, tiene que tener instalada la versión correcta y tenga permisos para hacerlo.

# API Multimedia - Introducción

---

- Los plugins pueden causar que el navegador o el sistema se comporte de manera inestable,
- O incluso podemos crear una inseguridad a usuarios sin conocimientos técnicos, a los que se pide que descarguen e instalen un nuevo software.

# API Multimedia - Vídeo

---

- Una de las mayores ventajas del elemento `<video>` de HTML5 es que está totalmente integrados en la web.
- Ya no es necesario depender de software de terceros.
- El elemento `<video>` pueden personalizarse a través de estilos CSS y **manipularlos con total libertad**.
- Para hacer funcionar el vídeo en HTML, es suficiente con incluir el siguiente marcado:

```
<video src="movie.webm"> </video>
```

- Sin embargo, esto lo único que se muestra es el primer fotograma de la película.

# API Multimedia - Vídeo

---

## AUTOPLAY

- Podemos indicar al navegador que reproduzca el vídeo de manera automática una vez se haya cargado la página.
- No es una buena práctica, ya que a muchos usuarios les parecerá una práctica muy intrusiva (no querrán que el vídeo se reproduzca sin su autorización).

```
<video src="movie.webm" autoplay>
 <!-- Your fallback content here -->
</video>
```

# API Multimedia - Vídeo

## CONTROLES

---

- Proporcionar controles es mejor visto por los usuarios que reproducir el vídeo de manera automática.
- Cada navegador los controla de manera diferente, pero todos coinciden en los controles a mostrar:
  - Reproducir/Pausa,
  - Barra de progreso
  - Control de volumen.
- Normalmente, los navegadores esconden los controles, y solamente aparecen al mover el ratón sobre el vídeo.

```
<video src="movie.webm" controls>
 <!-- Your fallback content here -->
</video>
```

# API Multimedia - Vídeo

## CONTROLES

---

- Ejemplo de como suele mostrarlo Google Chrome



pue

# API Multimedia - Vídeo

---

## POSTER

- Indica la imagen que el navegador debe mostrar mientras:
  - El vídeo se está descargando O
  - Hasta que el usuario reproduce el vídeo.
- Esto elimina la necesidad de mostrar una imagen externa que después hay que eliminar con JavaScript.
- Si no se indica este atributo, el navegador muestra el primer fotograma del vídeo.

```
<video src="movie.webm" poster="poster.jpg">
 <!-- Your fallback content here -->
</video>
```

# API Multimedia - Vídeo

---

## MUTED y LOOP

- El atributo **muted**, permite que el elemento multimedia se reproduzca inicialmente sin sonido, lo que requiere una acción por parte del usuario para recuperar el volumen.
- El atributo **loop** indica que el vídeo se reproduce de nuevo una vez que ha finalizado su reproducción.

```
<video src="movie.webm" controls loop muted>
 <!-- Your fallback content here -->
</video>
```

# **API Multimedia - Vídeo**

---

## **HEIGHT, WIDTH**

- Indican al navegador el tamaño del vídeo en pixels.
- Si no se indican estas medidas el navegador utilizará:
  - 1) Medidas definidas en el vídeo de origen
  - 2) Si no, medidas definidas en el fotograma poster
  - 3) Si no el ancho por defecto es de 300 pixels.
- Si únicamente se especifica una de las dos medidas, el navegador automáticamente ajusta la medida de la dimensión no proporcionada, conservando la proporción del vídeo.

# API Multimedia - Vídeo

---

## PRELOAD

- Sugerencia al navegador de cuando se empieza la descarga del vídeo.
- Si no indicamos uno en concreto, es el propio navegador el que decide qué hacer.
- Existen tres valores definidos para preload:  
`preload=auto`: comience la descarga automáticamente.  
`preload=none`: no comience la descarga hasta que lo indique el usuario.  
`preload=metadata`: carga de los metadatos (dimensiones, fotogramas, duración...), pero no el propio vídeo hasta que el usuario lo indique.

# API Multimedia - Vídeo

## Donde está el vídeo y formatos

- A través de la etiqueta `<source src="">` se indica la localización del recurso a reproducir si el navegador soporta el codec o formato específico.
- Utilizar **una sola etiqueta** `<source src="">` **no es viable** porque no todos los navegadores pueden reproducir los mismos formatos:

	WEBM	MP4	OGV
Opera	Sí	No	Sí
Firefox	Sí	Sí	Sí
Chrome	Sí	No	Sí
IE9+	No	Sí	No
Safari	No	Sí	No

pue

# API Multimedia - Vídeo

## Donde está el vídeo y formatos

```
<video controls>
 <source src="cocinando.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
 <source src="cocinando.webm" type='video/webm; codecs="vp8, vorbis"'>
 <p>Your browser doesn't support video.</p>
</video>
```

# API Multimedia - Vídeo

## Media Query para vídeo

---

- Los ficheros de vídeo tienden a ser pesados, y enviar un vídeo en alta calidad a un dispositivo con un tamaño de pantalla reducido es algo ineficiente.
- HTML5 permite utilizar el atributo media en el elemento `<source>` y consultar al navegador por el ancho de la pantalla y definir el tamaño del vídeo según el dispositivo.

```
<video controls>
 <source src="hi-res.mp4" media="(min-device-width: 800px)">
 <source src="lo-res.mp4">
</video>
```

# API Multimedia - Vídeo

---

## FullScreen

- Flash ha ofrecido un modo de pantalla completa durante muchos años a la que muchos navegadores se han resistido **por seguridad**.
- Si se fuerza a una aplicación o web a funcionar a pantalla completa, el usuario pierde el control del propio navegador y controles estándar del sistema operativo.
- Puede que el **usuario no sepa después volver del modo de pantalla completa** o
- Puede haber **problemas de seguridad** relacionados, como la simulación del sistema operativo, petición de password, etc.

pue

# API Multimedia - Vídeo

---

## FullScreen

- Debido a ello, la propiedad controls no incluye el acceso a la pantalla completa.
- Pero el API establece un método JavaScript para activar esta característica:
  - `.requestFullScreen()`
  - `.mozRequestFullScreen()`
  - `.webkitRequestFullscreen()`
- Una vez que un elemento pasa a pantalla completa, aparece un mensaje de forma temporal para informar al usuario de que puede presionar la tecla ESC en cualquier momento para volver a la ventana anterior.

# API Multimedia - Vídeo

---

## Ejemplo Paso a Paso

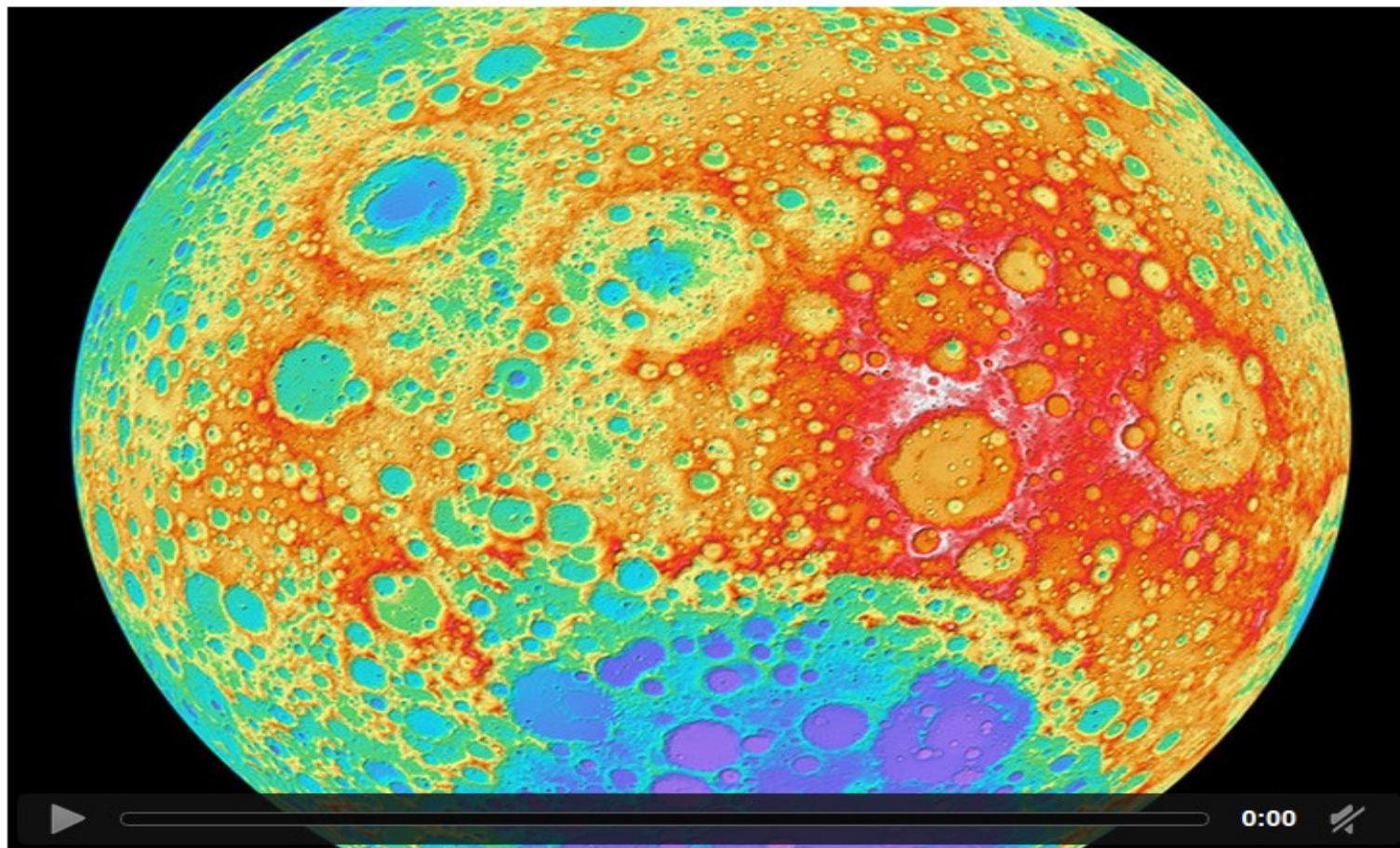
### Código HTML

```
<body>
 <div class="container">
 <div class="row">
 <div class="col-md-4 col-md-offset-4">
 <video id="v" poster="fotos/luna_de_colores.jpg" controls="controls"></video>
 </div>
 </div>
 <div class="row">
 <div class="col-md-4 col-md-offset-4">
 <button id="full"></button>
 </div>
 </div>
 </div>
</body>
```

# API Multimedia - Vídeo

## Ejemplo Paso a Paso

Página HTML



pue

# API Multimedia - Vídeo

## Ejemplo Paso a Paso

Código JavaScript para implementar el fullScreen:

```
$($document).ready(function(){
 $('button#full').on('click', function(){
 console.log("pulsado");
 var video = $('video#v');

 if (video[0].requestFullscreen) {
 console.log("normal");
 video[0].requestFullscreen();
 }
 if (video[0].mozRequestFullScreen) {
 console.log("moz");
 video[0].mozRequestFullScreen();
 }
 if (video[0].webkitRequestFullscreen) {
 console.log("webkit");
 video[0].webkitRequestFullscreen();
 }
 });
});
```

# API Multimedia - Audio

---

- Para hacer funcionar el audio en HTML, al igual que con el vídeo es suficiente con incluir lo siguiente:

```
<audio src="audio.mp3">
</audio>
```

# API Multimedia - Audio

---

- Para hacer funcionar el audio en HTML, al igual que con el vídeo es suficiente con incluir lo siguiente:

```
<audio src="audio.mp3">
</audio>
```

- Al igual que con vídeo, podemos incluir los controles para gestionarlo muy fácilmente con el atributo **controls**:

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

# API Multimedia - Audio

## Formatos

---

- La mejor solución en estos momentos es ofrecer tanto el formato libre OGG, como el propietario MP3
- Los formatos soportados por los navegadores son los siguientes:

	MP3	MP4	WAV	OGG
Opera	No	No	Sí	Sí
Firefox	No	No	Sí	Sí
Chrome	Sí	Sí	Sí	Sí
IE9+	Sí	Sí	No	No
Safari	Sí	Sí	Sí	No

# API Multimedia - Acceder a la cámara

---

- Gracias al proyecto llamado WebRTC (comunicaciones web en tiempo real)
- Supervisado por W3C
- Los navegadores están trabajando actualmente para incorporar implementaciones en sus navegadores.
- Aunque no todos los navegadores lo soportan.
- `navigator.getUserMedia()` es la puerta de entrada al microfono y cámara pidiendo permiso al usuario.
  - Proporciona la forma de acceder al flujo de datos de la cámara y del micrófono del usuario.
  - Sin tener que instalar nada!!

# API Multimedia - Acceder a la cámara

---

- Como preguntar si el navegador soporta dicha API:

```
function has GetUserMedia() {
 // Note: Opera builds are unprefixed.
 return !(navigator.getUserMedia || navigator.webkit GetUserMedia ||
 navigator.mozGetUserMedia || navigator.msGetUserMedia);
}

if (has GetUserMedia()) {
 // Good to go!
} else {
 alert('getUserMedia() is not supported in your browser');
}
```

# API Multimedia - Acceder a la cámara

---

## Acceso al dispositivo

- Para utilizar una cámara web o un micrófono, necesitamos solicitar permiso.
- **El primer parámetro de `getUserMedia()`** es para especificar el tipo de medio al que se quiere acceder.
- Si quieres solicitar la cámara web, el primer parámetro debe ser video.
- Para usar tanto el micrófono como la cámara, utiliza video, audio:

# API Multimedia - Acceder a la cámara

---

## Acceso al dispositivo

- **El segundo parámetro de `getUserMedia()`** es la función que realiza las acciones si hemos tenido acceso.
- Recibe como argumento el flujo de bits generados por la cámara y el micrófono.
- **El tercer parámetro de `getUserMedia()`** es la función que realiza las acciones si hemos tenido errores al acceder a la cámara o micrófono.

# **API Multimedia - Acceder a la cámara**

---

## **Ejemplo paso a paso**

- En este ejemplo permitiremos activar y desactivar la cámara y el micrófono de nuestro dispositivo además de sacar una fotografía
- Para ello usaremos en nuestro código de HTML5 varios elementos:
  - `<video>` donde definiremos `autoplay` para que el video no pare en el primer fotograma
  - `<canvas>` para poder pintar la foto del contenido del video
  - `<button>` para:
    - Grabar
    - Parar Grabación
    - Sacar foto

# API Multimedia - Acceder a la cámara

## Ejemplo paso a paso

- Código HTML

```
<body>
 <div class="container">
 <div class="row">
 <div class="col-md-4">

 </div>
 </div>
 <div class="row">
 <div class="col-md-5">
 <video id="camara" controls="controls" width="400" height="250" autoplay></video>
 </div>
 <div class="col-md-5">
 <canvas style="border: 1px solid black" id="canvas" width="400" height="250"></canvas>
 </div>
 </div>
 <div class="row">
 <div class="col-md-8">
 <div class="btn-group">
 <button id="inicio" class="btn btn-warning">Grabar</button>
 <button id="fin" class="btn btn-primary">Parar</button>
 <button id="foto" class="btn btn-info">Foto</button>
 </div>
 </div>
 </div>
 </div>
</body>
```

# API Multimedia - Acceder a la cámara

## Ejemplo paso a paso

- Código JavaScript:
  - Comprobar si el navegador soporta getUserMedia()

```
$(document).ready(function(){

 // comprobar si el navegador soporta getUserMedia()
 function has GetUserMedia() {
 return !(navigator.getUserMedia || navigator.webkit GetUserMedia ||
 navigator.mozGetUserMedia || navigator.msGetUserMedia);
 }
 if (has GetUserMedia()) {
 console.log(navigator.getUserMedia || navigator.webkit GetUserMedia ||
 navigator.mozGetUserMedia || navigator.msGetUserMedia);
 } else {
 alert('getUserMedia() is not supported in your browser');
 }
})
```

# API Multimedia - Acceder a la cámara

## Ejemplo paso a paso

- Código JavaScript:
  - Comprobar si el navegador soporta getUserMedia()

```
$(document).ready(function(){

 // comprobar si el navegador soporta getUserMedia()
 function has GetUserMedia() {
 return !(navigator.getUserMedia || navigator.webkit GetUserMedia ||
 || navigator.mozGetUserMedia || navigator.msGetUserMedia);
 }
 if (has GetUserMedia()) {
 console.log(navigator.getUserMedia || navigator.webkit GetUserMedia ||
 || navigator.mozGetUserMedia || navigator.msGetUserMedia);
 } else {
 alert('getUserMedia() is not supported in your browser');
 }
})
```

# API Multimedia - Acceder a la cámara

---

## Ejemplo paso a paso

- Código JavaScript:
  - Crear objeto que guarda info. de los datos emitidos por cámara y micrófono

```
// creamos un objeto video en la ventana que contiene los 2 datos que necesitamos
// el flujo de bits que genera la camara
// la url creada para acceder al flujo de la camara
// asi podemos grabar y parar la grabacion
window.datosVideo = {
 'flujo' : null ,
 'url' : null
};
```

# API Multimedia - Acceder a la cámara

## Ejemplo paso a paso

- Código JavaScript:
  - Que hacer al iniciar la grabación

```
// si pinchamos en inicio
$('button#inicio').on('click',function(){
 // si soporta getUserMedia para chrome
 if (navigator.webkitGetUserMedia){
 // PARAMETRO 1: pedimos video y audio
 navigator.webkitGetUserMedia({video: true, audio: true},
 // PARAMETRO 2: funcion de ok que recibe el flujo de camara y microfono
 function(flujoCamara) {
 // le damos los datos al objeto del window
 // para poder parar la grabacion
 datosVideo.flujo = flujoCamara;
 datosVideo.url = window.URL.createObjectURL(flujoCamara);
 // modificamos la propiedad src de la etiqueta video para:
 // darle la url con el flujo generado por camara y microfono
 $('#video#camara').attr({
 "src" : datosVideo.url,
 });

 }, // PARAMETRO 3: funcion de error
 function(e){
 console.log("error al intentar grabar");
 });
 });
});
```

pue

# API Multimedia - Acceder a la cámara

## Ejemplo paso a paso

- Código JavaScript:
  - Que hacer al parar la grabación

```
// si hacemos click en para
$('button#fin').on('click', function(){
 // si soporta getUserMedia para chrome
 if (navigator.webkitGetUserMedia){
 // Si hay datos recibidos de la cámara
 if (datosVideo.flujo !== null){
 // del objeto de la ventana
 // paramos el flujo
 datosVideo.flujo.stop()
 // paramos el flujo de la cámara
 window.URL.revokeObjectURL(datosVideo.url)
 // quitamos propiedad src a la etiqueta video
 $('video#camara').attr({
 "src" : null
 });
 }
 }
});
```

pue

# API Multimedia - Acceder a la cámara

## Ejemplo paso a paso

- Código JavaScript:
  - Dibujar en canvas la foto con lo que hay en el vídeo

```
// Para hacer una foto
// obtenemos lo que hay actualmente en la etiqueta video
// y lo dibujamos con canvas
$('button#foto').on('click', function(e){
 // la camara
 var oCamara = $('video#camara');
 // el lienzo de canvas
 var oFoto = $('canvas#canvas');
 // creamos contexto del lienzo para empezar a pintar
 var oContexto = oFoto[0].getContext('2d');
 // dibujamos la imagen de la camara:
 // desde las coordenadas (0,0)
 // hasta donde acabe la imagen (,)
 oContexto.drawImage(oCamara[0], 0, 0, oCamara.width(), oCamara.height());
});

});
```

# API Web Sockets - Introducción

---

- En las aplicaciones web con cliente y servidor se utiliza el paradigma solicitud/respuesta de HTTP.
- Un cliente carga una página web del servidor, se cierra la conexión y no ocurre nada hasta que el usuario hace clic en un enlace o envía un formulario.
- En este tipo de comunicación:
  - 1) Es el cliente el que debe iniciar la comunicación.
  - 2) Es uni-direccional, el cliente envía o recibe.

# API Web Sockets - Introducción

---

- Los WebSockets nos ofrecen una conexión bidireccional entre el servidor y el navegador que se mantiene permanentemente abierta hasta que se cierre de manera explícita.
- Ahora cuando el servidor quiere enviar datos, el mensaje se traslada inmediatamente.

## API Web Sockets - Beneficios

---

- Si disponemos de un socket abierto, el servidor puede enviar datos a todos los clientes conectados a ese socket, sin tener que estar constantemente procesando peticiones de Ajax.
- La latencia en las comunicaciones es mínima ya que los datos son enviados inmediatamente desde el servidor al navegador.
- Los datos a transmitir se reducen también de manera drástica, pasando de un mínimo de 200-300 bytes en peticiones Ajax, a 10-20 bytes utilizando websockets.

# API Web Sockets - Crear un Web Socket

---

- Para abrir una conexión WebSocket con el servidor, sólo tenemos que ejecutar el constructor WebSocket, que toma como parámetro la URL del socket a abrir en el servidor.
- Hay que tener en cuenta que el protocolo a utilizar es ws://

```
var socket = new WebSocket('ws://html5rocks.websocket.org/tweets');
```

- También existe wss:// para conexiones seguras.

# API Web Sockets - Comunicación con el Servidor

---

- Cuando se establece una conexión con el servidor, se puede empezar a enviar datos al servidor con el método send a través del socket creado.

```
socket.send("Hey there, I'm using WebSockets");
```

# API Web Sockets - Comunicación con el Servidor

---

- De la misma forma, el servidor puede enviarnos mensajes en cualquier momento.
- Cada vez que esto ocurra, se activa el **evento onmessage**.
- Los datos enviados por el servidor se encuentran en la propiedad data del objeto event.

```
socket.onmessage = function(event) {
 var data = JSON.parse(event.data);
 if (data.action == 'joined') {
 initialiseChat();
 } else {
 showNewMessage(data.who, data.text);
 }
};
```

pue

# API Web Workers - Introducción

---

- Los navegadores ejecutan las aplicaciones en un único thread, lo que significa que si JavaScript está ejecutando una tarea muy complicada el rendimiento del navegador se ve afectado.
- Los Web Workers permite crear un entorno en el que un bloque de código JavaScript puede ejecutarse de manera paralela sin afectar al thread principal del navegador.
- Los Web Workers utilizan un protocolo de paso de mensajes similar a los utilizados en programación paralela.

# API Web Workers - Introducción

---

- Los Web Workers se ejecutan en un subprocesso aislado y se debe encontrar en un archivo independiente.
- Pero lo primero que se tiene que hacer es crear un nuevo objeto Worker en la página principal:

```
var worker = new Worker('miWorker.js');
```

- Si el archivo existe, el navegador generará el subprocesso descargando el archivo de forma asíncrona que no comenzará a ejecutarse hasta que este descargado completamente.
- Si hay errores al crear el Worker se lanza el **evento error**.

```
worker.onerror = function(e){
 console.log("ERROR del WORKER");
};
```

pue

# API Web Workers - Comunicación

---

- Para utilizar los Workers, es necesario conocer el protocolo de paso de mensajes,
  - También utilizado en otras APIs como WebSocket.
- 
- **Tanto Worker como Main** se escuchan, con el evento **message**
  - En el argumento pasado al evento message tenemos la propiedad **data** para conocer el contenido de la comunicación.
  - La comunicación entre un Worker y su página principal se realiza mediante el método **postMessage()** que acepta como parámetro una cadena o un objeto JSON.

# API Web Workers - Comunicación

---

## 1) Cadena, envío y recepción:

```
var sms = 'hola worker';
worker.postMessage(sms);
```

```
self.onmessage = function(sms){
 var sms_recibido = sms.data;
```

## 2) Objeto JSON, envío y recepción::

```
var sms_json = {
 'clave1' : 'valor1',
 'clave2' : 'valor2'
};
worker.postMessage(sms_json);
```

```
self.onmessage = function(sms){
 var sms_recibido = sms.data.clave1 + " " + sms.data.clave2;
```

pue

# **API Web Workers - Funciones disponibles para Workers**

---

- Tienen restringido el conjunto de funciones JavaScript:  
Objetos navigator y location (de solo lectura)  
XMLHttpRequest  
setTimeout()/clearTimeout() y  
setInterval()/clearInterval()  
Caché de la aplicación  
Generación de otros Web Workers
- Y NO pueden acceder a las siguientes funciones:  
DOM (no es seguro para el subproceso)  
Objetos window, document ni parent

# API Web Workers - Subworkers

---

- Los Workers tienen la capacidad de generar Workers secundarios.
- Esto es algo fantástico para dividir aún más las tareas intensivas en el tiempo de ejecución.
- Sin embargo, los navegadores generan procesos independientes para cada Worker.
- Antes de generar un conjunto de Workers, comprueba que no utilizaran demasiados recursos del sistema del usuario.

# API Web Workers - Cuando usarlos

---

- *Algunos ejemplos de cuando usarlos son:*
- Obtención previa y/o almacenamiento en caché de datos para un uso futuro
- Análisis de datos de vídeo o audio
- Solicitud de servicios web (AJAX)
- Procesamiento de conjuntos o respuestas JSON de gran tamaño
- Actualización de varias filas de una base de datos web local

# **API Web Workers**

---

## **Ejemplo Cadena Paso a Paso**

- Ejemplo sobre cómo utilizar una cadena para transferir a un Worker.
- El Worker simplemente devuelve el mensaje completado con un saludo.

# API Web Workers

## Ejemplo Cadena Paso a Paso

- HTML con cajas de texto y botón para comunicarnos con el Worker.

```
<div class="form-group">
 <label for="main">main</label>
 <input type="text" id='main' class="form-control" placeholder="quien eres??">
</div>
<button type="submit" id="saludar" class="btn btn-default">Saludar con worker</button>
<div class="form-group">
 <label for="worker">main</label>
 <input type="text" id='worker' class="form-control" disabled>
</div>
```

# API Web Workers

## Ejemplo Cadena Paso a Paso

- Salida HTML:

main

quién eres??

Saludar con worker

main

pue

# API Web Workers

## Ejemplo Cadena Paso a Paso

- Código del Main:

```
$(document).ready(function(){
 // instanciamos worker
 var worker = new Worker('miWorker.js');
 // Ver por consola si hay errores
 worker.onerror = function(e){
 console.log("ERROR del WORKER");
 };
 // Leemos lo introducido por el usuario en la caja de texto y
 // al hacer click en el boton y se lo enviamos mensajes al worker
 $('button#saludar').on('click',function(){
 var sms = $('input#main').val();
 worker.postMessage(sms);
 });
 // Escuchamos lo que nos devuelve el worker
 worker.onmessage =function(e) {
 $('input#worker').val(e.data);
 };
});
```

# API Web Workers

## Ejemplo Cadena Paso a Paso

- Código del Worker:

```
// El worker espera recibir un mensaje para hacer su trabajo
// en este caso:
// Recoger la cadena pasada, añadir palabra worker y devolverlo
self.onmessage = function(sms){
| self.postMessage("hola " + sms.data);
};
```

# API Web Workers

## Ejemplo Cadena Paso a Paso

- Ejecución:

main

julio

Saludar con worker

main

hola julio

pue

# **API Web Workers**

## **Ejercicio**

---

- Ejemplo sobre cómo utilizar una cadena para transferir a un Worker.
- El Worker simplemente devuelve el mensaje completado con un saludo.

---

# **Módulo 10:**

# **Eventos Touch con jQuery**

pue

# Introducción

---

- Con la llegada de HTML5, la forma de crear aplicaciones web ha cambiado
- JAVASCRIPT no es HTML, pero sí es un elemento de gran importancia para desarrollar con HTML5 donde los desarrolladores tienen a mano nuevos elementos.
- Estos son soportados por nuevas o mejoradas APIS JAVASCRIPT.
- Sin olvidar a CSS3 que ha contribuido en una nueva forma **versátil y adaptable** de visualizar aplicaciones web con una sutileza semántica.

# Introducción

---

- Los dispositivos móviles como los SmartPhones o Tablets suelen tener una pantalla capacitiva sensible al tacto.
- Reconocen las interacciones de los dedos de los usuarios con la pantalla y se hace necesario que los desarrolladores web dispongan de un método de gestión de estos eventos.
- Cuando desarrollamos para dispositivos móviles, principalmente táctiles, tenemos que tener en cuenta que no disponemos de un ratón.
- A priori, no nos es posible interactuar con eventos del estilo `onMouseMove`, `onMouseOver`, etc.

# Eventos touch

---

- Por suerte para los desarrolladores, disponemos de la opción de capturar las acciones del Touch en estos dispositivos .
- Los navegadores de los dispositivos táctiles hacen que los eventos de ratón como el clic, puedan convertirse en un evento Touch, por eso, al hacer una aplicación web, ésta funciona aunque sea procesando eventos del ratón.
- Ahora se han agregado a JAVASCRIPT una interesante gama de eventos Touch, que a pesar de ser solo de tres tipos, podemos hacer una combinación de los mismos, y así se obtienen mejores resultados.

# Eventos touch

---

**touchstart**: Este se genera al hacer cualquier toque a la pantalla, sin importar su duración o movimientos realizados.

**touchend**: Este se ejecuta cuando se deja de tocar la pantalla o el objeto que tiene asignado el manejador de eventos.

**touchmove**: Este es ejecutado una vez se desplaza el dedo, por encima de la pantalla u objeto que está siendo controlado a través del manejador de eventos.

# Parámetros devueltos por Eventos touch

---

- Cada evento Touch posee una lista de parámetros en común que complementan el manejo y procesamientos tras la interacción de un usuario a través de un dispositivo táctil.
- Son:
  - touches:** Array de todos los dedos con los que se está tocando CUALQUIER LUGAR DE LA PANTALLA.
  - targetTouches:** Array de todos los dedos con los que se está tocando EL MISMO ELEMENTO.
  - changedTouches:** Array con los dedos que causaron la última acción.

# Parámetros devueltos por Eventos touch

---

- Ejemplos del funcionamiento de las listas, para entender mejor su utilidad:
- **Poner un solo dedo sobre la pantalla:**
  - Las tres listas contendrán la misma información.
  - La relativa a dicho dedo.
- **Poner un segundo dedo sobre la pantalla.**
  - `touches` tendrá dos elementos, uno para cada dedo.
  - `targetTouches` contendrá dos elementos si el segundo dedo se posiciona sobre el mismo elemento que el primero
  - `changedTouches` solo contendrá información sobre el segundo dedo.

# Parámetros devueltos por Eventos touch

---

- **Mover dedo por la pantalla:**
  - la única lista que registrara cambios será `changedTouches`.
  - Tendrá información sobre el dedo.
- **Levantar un dedo:**
  - Su información es eliminada de `touches` y de `targetTouches`,
  - Però aparecerá en `changedTouches`
- **Retirar el último dedo:**
  - `touches` y `targetTouches` se vacian.
  - `changedTouches` mantiene la información del último dedo.

# Información sobre el dedo

---

**target**: Nodo sobre el que se origino el evento.

**pageX**: coordenada X relativa a la página.  
(incluye scroll)

**pageY**: coordenada Y relativa a la página  
(incluye scroll)

**clientX**: coordenada X del touch relativa al viewport.  
(excluye el offset del scroll)

**clientY**: coordenada Y del touch relativa al viewport.  
(excluye el offset del scroll)

**screenX**: coordenada X relativa a la pantalla.

**screenY**: coordenada Y relativa a la pantalla.

**identifier**: Identificador numérico, único para cada evento.

## Ejemplo Paso a Paso

---

- Ejemplo donde vemos como, pulsar el ratón nos dice en que posición está de la ventana
- Al moverlo nos indica si nos movemos arriba,abajo, izquierda y derecha respecto a esa posición inicial
  - Al dejar de pulsar nos indica donde se quedo el ratón.

# Ejemplo Paso a Paso

---

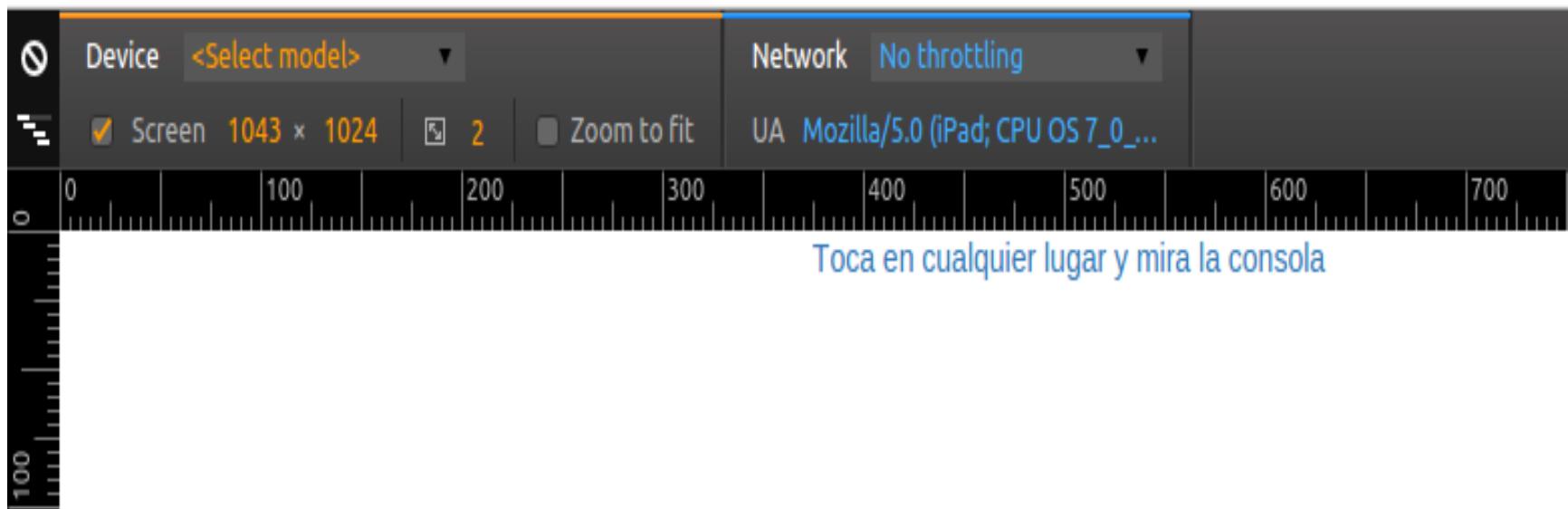
- Código HTML

```
</head>
<body>
 <div class="container">
 <div class="row">
 <div class="col-md-4 col-md-offset-4">
 <a>Toca en cualquier lugar y mira la consola
 </div>
 </div>
 </div>
</body>
```

# Ejemplo Paso a Paso

---

- Salida HTML



pue

# Ejemplo Paso a Paso

---

- Código JavaScript → Inicio

```
$($.document).ready(function(){
 // inicializamos coordenadas de referencia
 var x = 0;
 var y = 0;
```

# Ejemplo Paso a Paso

---

- Código JavaScript → Tocar pantalla

```
// evento lanzado al tocar la pantalla
$(document).on('touchstart', function(dedos){
 // si hay algun dedo
 if (dedos.originalEvent.touches.length > 0){
 // miramos al primer dedo
 // Almacenamos coordenadas de referencia
 var dedo = dedos.originalEvent.touches[0];
 x = dedo.clientX;
 y = dedo.clientY;
 // Mostramos las coordenadas de referencia
 console.log("Empiezo en: " + x + ", " + y + "\n");
 }
});
```

# Ejemplo Paso a Paso

---

- Código JavaScript → Mover dedo por la pantalla

```
// evento lanzado mover dedo por la pantalla
$(document).on('touchmove', function(dedos){
 // inicializamos variables que contendran
 // las coordenadas al movernos
 var movX = "";
 var movY = "";
 // si hay algun dedo todavia
 if (dedos.originalEvent.targetTouches.length > 0){
 // miramos al primer dedo
 // calculamos hacia donde se mueve
 var dedo = dedos.originalEvent.targetTouches[0];
 var nuevaX = x - dedo.clientX;
 var nuevaY = y - dedo.clientY;
```

# Ejemplo Paso a Paso

---

- Código JavaScript → Mover dedo por la pantalla

```
// hacia donde se movio en horizontal
// sensibilidad de 10px
if (nuevaX > 10){
 movX = "izquierda";
}else{
 if (nuevaX < 10){
 movX = "derecha";
 }else{
 movX = "centro";
 }
}
// hacia donde se movio en vertical
// sensibilidad de 10px
if (nuevaY > 10){
 movY = "arriba";
}else{
 if (nuevaY < 10){
 movY = "abajo";
 }else{
 movY = "centro";
 }
}
console.log("Me he movido hacia: " + movX + " y " + movY);
});
```

pue

# Ejemplo Paso a Paso

---

- Código JavaScript → levantar dedo de la pantalla

```
// evento lanzado dejar de tocar la pantalla
$(document).on('touchend', function(dedos){
 // Del dedo que soltó, donde se quedó
 if (dedos.originalEvent.changedTouches.length > 0){
 var dedo = dedos.originalEvent.changedTouches[0];
 x = dedo.clientX;
 y = dedo.clientY;
 console.log("Acabo en: \\" + x + "," + y + "\")");
 }
});
```

# Ejemplo Paso a Paso

---

- Después de “tocar, mover y soltar dedo” la salida por consola es:

```
✖️ ⏴ <top frame> ✎ Preserve log
Empiezo en: (256,65)
71 Me he movido hacia: derecha y abajo
 Me he movido hacia: centro y abajo
83 Me he movido hacia: izquierda y abajo
 Me he movido hacia: izquierda y centro
77 Me he movido hacia: izquierda y arriba
 Me he movido hacia: derecha y arriba
 Me he movido hacia: derecha y centro
2 Me he movido hacia: derecha y abajo
 Acabo en: (333,57)
> |
```

## Ejercicio 1

---

A partir de un html con un elemento <div>, permitir su movimiento por el documento

## Ejercicio 2

---

A partir de un html con un elemento canvas, permitir dibujar simulando que el “dedo” es un lapiz

---

# **Módulo 11:**

# **Mimificadores**

pue

# Introducción

---

- HTML, CSS y JavaScript son los lenguajes utilizados para desarrollar páginas web.
- Las páginas web pueden estar formadas por pocos archivos o por muchos, si es el caso de estar formada por muchos archivos puede que la web se vuelva lenta y tarde mucho en cargar.
- Una web lenta nos perjudica ya que a los buscadores como Google no les gustan y penalizan por ello bajando posiciones en las búsquedas.
- Además los usuarios optarán por páginas más rápidas que la nuestra.
- Para evitar estos problemas y conseguir rapidez en la web se usa la técnica de minificar.

# Que es mimificar el código

---

- Minificar es conseguir que un fichero sea menos pesado modificando ciertos elementos:
  - espacios innecesarios
  - tabulaciones,
  - comentarios,
  - Cambiar nombres de variables por otros más cortos,
  - etc ...

# Que es Mimificar el código

---

- Para que esta técnica sea más efectiva es recomendable juntar los archivos CSS en uno solo para que:
  - El cliente haga menos llamadas al servidor
  - Aunque el fichero es más grande se evita enviar mucha información adicional de cada comunicación.
- También se deben juntar los archivos de JavaScript pero estos con más cuidado dado que a veces no funcionan algunas secuencias y puede ocasionar errores en la web.
- Para evitar estos errores se pueden ir juntando los archivos poco a poco y comprobar que archivos no pueden ser combinados.

## Mimificar CSS3 - clean-css

---

- El proceso de mimificación típico sólo elimina espacios en blanco y comentarios de tu CSS.
- El paquete clean-css por otro lado puede también hacer cosas como:
  - Combinar nombres de selector duplicados
  - Combinar propiedades duplicadas dentro del mismo estilo
  - Redondear números con muchos decimales
  - Remover puntos y comas y espacios al final de los estilos del selector

# Mimificar CSS3 - clean-css

---

- Para instalar clean-css: `$ npm install clean-css -g`
- Para mimificar varios ficheros css en uno único:

```
$ cleancss -o style-min.css *.css
```

→ Busca todos los ficheros acabado en .css del directorio actual, los junta y los mimifica en el fichero de **salida style-min.css**

# Mimificar JavaScript - uglyfi-js

---

- Es común tener múltiples archivos JavaScript como jQuery y nuestros propios códigos.
- También queremos minimizar el número de peticiones http que hace nuestro sitio y que el tiempo de carga de los archivos Javascript sea lo más rápido.
- La mejor forma de manejar ambos asuntos es concatenar simultáneamente nuestros archivos Javascript en un sólo archivo.
- Eso significará que sólo necesitaremos una petición http para cargar, y del menor tamaño posible.

# Minificación de Javascript de terceros

---

- Si estás combinando y/o minificando scripts de terceros como jQuery asegúrate de usar las versiones extendidas de esos archivos.
- Es generalmente una mala idea volver a minificar archivos que ya han pasado por éste proceso ya que puedes afectar su funcionalidad.
- Puedes identificar archivos que ya han sido minificados por su nombre, que contendrá la cadena **min**, por ejemplo:
  - **Sin mimificar** : "nombre.js"
  - **Mimificado** : "nombre.min.js",

# Mimificar JavaScript - uglyfi-js

---

- Para instalar uglify-js: `$ npm install uglify-js -g`
- Para la unificación y mimificación tenemos varias opciones:
  - 1) mimificación mínima:

```
$ uglifyjs file1.js file2.js --output file.min.js
```

- 2) Modificar también parte del código, reduciendo variables, funciones, etc a un solo carácter:

```
$ uglifyjs file1.js file2.js --output file.min.js --mangle
```

---

# **Módulo 12:**

# **Introducción a phonegapp**

pue

# Que es phonegap

---

- Framework creado por **Nitobi**, el cual **ahora** es **propiedad de Adobe**.
- Permite desarrollar aplicaciones multiplataforma o híbridas de manera utilizando las tecnologías web como Html5, Css3 y JavaScript.
- También nos permite integrarlo con Jquery y otras librerías que se requiera integrar a un proyecto.
- Una **ventaja sobre** las **nativas** es que pueden **usarse** en **cualquier dispositivo**, aunque posean diferentes sistemas operativos.
- Es una **alternativa** a desarrollar tu aplicación en **nativo para cada** uno de los **sistemas operativos**, con el consiguiente trabajo que ello conlleva.

pue

# Que es phonegap

---

- Dentro de las principales características de Phonegap, nos encontramos una serie de Apis para controlar los diferentes recursos del dispositivo como lo son:
  - Cámara
  - Acelerómetro
  - GPS
  - Notificaciones
  - Almacenamiento
  - ....

# **Compatibilidad con Sistemas Operativos**

---

- En la actualidad Phonegap permite desarrollar aplicaciones para los diferentes tipos de sistemas operativos para móviles:
  - Android
  - iOS
  - BlackBerry OS
  - Windows Phone
  - Web OS
  - Symbian
  - Bada

# PhoneGap y Apache Cordova

---

- PhoneGap es una distribución de Apache Cordova.
- El proyecto originalmente nació con el nombre de PhoneGap, y luego se cedió a la fundación Apache como software libre.
- La comunidad del proyecto, Apache, decidió en 2012 cambiarle el nombre, entre otros motivos para diferenciarlo de la marca PhoneGap, que continúa en poder de Adobe.
- No hay grandes diferencias:
  - **Apache Cordova** suele actualizarse más frecuentemente
  - **PhoneGap** incluye algunas librerías adicionales para integrarse con diversos productos de Adobe.

# Cuando usar phonegap

---

- PhoneGap es un excelente camino para resolver necesidades de creación de aplicaciones de una manera única y compatible con todos los dispositivos.
- **Desventaja** : El rendimiento y posibilidades no llegan a la altura del desarrollo nativo.
- Por tanto, escoger o no PhoneGap para el desarrollo de un nuevo producto es una decisión que hay que tomar con cuidado.

# **Cuando usar phonegap**

---

## **Usa PhoneGap si:**

- Tu aplicación no requiere exprimir el rendimiento del dispositivo, donde el contenido es parecido al que encontrarías en una web.
- Tienes prisa en lanzar el desarrollo y necesitas un alcance global en todos los dispositivos.
- No requieres usar intensivamente mucha variedad de sensores y periféricos.

pue

# **Cuando usar phonegap**

---

## **No usar PhoneGap si:**

- No importa aprender varios lenguajes para realizar el desarrollo nativo en cada sistema operativo.
- Tu aplicación va a requerir mucha cantidad de procesamiento (Los juegos son un claro ejemplo)
- Solo quieres desarrollar para un único sistema operativo.
- Quieres hacer uso de muchos sensores o periféricos específicos de cada dispositivo.

# Cómo funciona PhoneGap

---

- En vez de aprender las librerías propias de cada sistema con sus lenguajes, haces uso de aquellas que te proporciona el framework, usando un único lenguaje de programación, Javascript
- Por ejemplo:
  - En lugar de comunicar directamente con la cámara,
  - Comunicas con las librerías de PhoneGap,
  - Éstas, por medio de una interfaz, te permiten interactuar con la cámara.
- Quien habla en el idioma que el sistema operativo del dispositivo requiere son las librerías de phonegap.

# Cómo funciona PhoneGap

---

- Pero, aunque se programe únicamente en Javascript, necesitas instalar todo el conjunto de librerías del sistema donde quieras publicar tu proyecto.
- Por ejemplo, si estás programando para iOS, necesitas:
  - Entorno de desarrollo en un Mac, comúnmente el IDE Xcode,
  - SDK para desarrollo iOS,
  - Plugins adicionales y
  - La licencia de desarrollador de Apple.
- Para comenzar a trabajar con PhoneGap, es recomendable realizar diversas aproximaciones para cada sistema operativo por separado.

# Instalación de phonegap en linux

---

- Software necesario para desarrollar sobre un proyecto de phonegap o cordova mientras se prueba en el emulador como Ripple de Apache
- Esta instalación no incluye ningún SDK específico ya que no construiremos paquetes instalables (como el APK de Android).

# Instalación de phonegap en linux

---

- Instalar NodeJS y su administrador de paquetes

```
$ sudo apt-get install nodejs npm
```

- Instalar el paquete de Phonegap o cordova:

```
$ npm install -g cordova
```

- Instalar el paquete de Ant, necesario para construir y ejecutar las aplicaciones desarrolladas con Phonegap.

```
$ npm install -g ant
```

# Creación de un proyecto en phonegap

---

- Crear un directorio para organizar los proyectos que realizaremos con phonegap:

```
$ mkdir proyectosPhoneGap
$ cd proyectosPhoneGap/
```

- Crear el proyecto a partir de una plantilla.

```
$ cordova create MiProyecto
```

- Esto crea un directorio con el nombre del proyecto y un conjunto de ficheros necesarios para phonegap

```
$ ls MiProyecto
config.xml hooks platforms plugins www
```

# Creación de un proyecto en phonegap

---

- Cabe destacar el subdirectorio www/ que contiene la estructura básica de nuestros proyectos en HTML5 CSS3 y JavaScript

```
$ ls MiProyecto/www/
css img index.html js
.
```

# **Emulando dispositivos móviles con Ripple**

---

- Ripple es un entorno de emulación multiplataforma que permite crear dentro del navegador web Google Chrome un ambiente similar al de un dispositivo móvil.
- También emula importantes funciones como pueden ser la geolocalización o la brújula disponibles en los smartphones.
- El principal enfoque de Ripple está en testear aplicaciones web móviles desarrolladas en la plataforma PhoneGAP, WebWorks y aplicaciones web basadas en HTML5.

# Trabajando con Ripple

---

- Para instalar ripple: `$ npm install -g ripple-emulator`
- Para Ejecutar el emulador, nos movemos al directorio www/ de nuestro proyecto y lo ejecutamos:

```
$ cd MiProyecto/www/
$ ripple emulate
INFO: Server instance running on: http://localhost:4400
INFO: CORS XHR proxy service on: http://localhost:4400/ripple/xhr_proxy
INFO: JSONP XHR proxy service on: http://localhost:4400/ripple/jsonp_xhr_proxy
INFO: Could not find cordova as a local module. Expecting to find it installed globally.
INFO: Using Browser User Agent (String)
INFO: Using Browser User Agent (String)
```

# Trabajando con Ripple

---

- Una vez ejecutado el emulador abre un servidor local en el puerto 4400.
- Con un panel central donde se visualiza el contenido
- 2 paneles laterales donde podemos simular cambios en nuestro dispositivo como:
  - orientación
  - acelerómetro

# Trabajando con Ripple

The screenshot shows the Ripple emulator interface. On the left, there's a sidebar with various settings and information panels:

- Devices:** Generic - WVGA (480x800)
- Orientation:** Landscape (indicated by icons of a phone and a tablet)
- Platforms:** Information panel showing:
  - Platform: Apache Cordova / PhoneGap
  - Device: Generic - WVGA (480x800)
  - OS: Generic Generic
  - Manufacturer: Generic
  - Screen: 480x800
  - Density: 96 PPI
  - User Agent: undefined
- Accelerometer:** A 3D icon representing an accelerometer.

A central message at the bottom of the sidebar says: "Click and drag with the mouse to manipulate the device. Hold the Shift key to modify 'alpha'". Below this, it shows "xAxis: -7.03m / s^2".

The main area of the interface is a large window titled "beta Ripple". It displays a dark gray screen with a white Apache Cordova logo in the center. The logo is a stylized "A" shape with blue glowing elements. Below the logo, the text "APACHE CORDOVA" is written in capital letters.

To the right of the Ripple window, there's a vertical sidebar with the following sections:

- Settings:** Device & Network Settings, Geo Location, Config
- Events:** A panel with the text "Select the event that you want to fire." containing two buttons: "deviceready" and "Fire Event".

pue

---

# Módulo 12: Publicar aplicaciones

# Publicar en Google Play

---

- Lo primero que tenemos que hacer es convertir nuestra cuenta de Google en una cuenta de desarrolladores.
- Para ello, accederemos a Google Play Developer Console por primera vez , haciendo login con una cuenta de gmail desde:

<https://play.google.com/apps/publish/>

# Publicar en Google Play

Google Play Developer Console

Inicia sesión con tu cuenta de Google    Acepta el Acuerdo para desarrolladores    Paga la cuota de registro    Rellena la información de tu cuenta

HAS INICIADO SESIÓN COMO...

julio.sanchez@pue.es

Esta es la cuenta de Google que se asociará a tu consola para desarrolladores.

Si quieras utilizar otra cuenta, puedes seleccionarla en las opciones que aparecen a continuación. Si eres una empresa, considera la posibilidad de registrar una nueva cuenta de Google en lugar de utilizar una cuenta personal.

[Iniciar sesión con otra cuenta](#)    [Crear una cuenta nueva de Google](#)

ANTES DE CONTINUAR...

Consulta y acepta el [Acuerdo de distribución para desarrolladores de Google Play](#).

Acepto las condiciones y quiero asociar el registro de la cuenta con el Acuerdo de distribución para desarrolladores de Google Play.

Consulta los países de distribución en los que puedes vender y distribuir aplicaciones.

Si piensas vender aplicaciones o productos integrados en aplicaciones, comprueba si tienes una cuenta de comerciante en tu país.

\$25

Asegúrate de tener tu tarjeta de crédito preparada para pagar la cuota de registro (25 USD) en el siguiente paso.

[Continuar para completar el pago](#)

pue

# Publicar en Google Play

---

- Allí deberemos hacer un pago de 25 dólares, una única vez, con lo que ya podremos ser desarrolladores que distribuyen apps en Google Play.
- A partir de aquí, podemos entrar en Google Play Developer Console por primera vez, nuestro centro de gestión e información como desarrolladores.
- En esta web, podremos ver:
  - Listado de nuestras aplicaciones
  - Servicios para Google Play Games
  - Informes de nuestros beneficios
  - Configuración
  - Anuncios
  - Alertas

# Publicar en Google Play

The screenshot shows the Google Play Developer Console interface. At the top left is the Google Play logo and the text "Google play | Developer Console". To the right is a search bar with a magnifying glass icon. On the far right of the header is a "Logout" button.

The main area is titled "ALL APPLICATIONS" in large capital letters. Below it is a teal button with a white plus sign and the text "Add new application".

To the left of the main content is a sidebar with the following items:

- All applications
- Game services
- Financial reports
- Settings
- Alerts
- Announcements

Below the sidebar is a "Filter" button with a downward arrow icon.

The main table has three columns: "APP NAME", "PRICE", and "CURRENT / TOTAL INSTALLS". There are several rows of blurred application entries.

pue

# **Publicar en Google Play**

---

- Una vez aquí, se nos abrirá toda la información sobre la aplicación. Publicar es realmente sencillo, pues basta con ir siguiendo los pasos que nos encontramos a la izquierda:

pue

# Publicar en AppleStore

---

## Registrarse como Apple Developer

- El primer paso es registrarse como desarrollador de Apple desde <http://developer.apple.com/devcenter/ios/index.action> pudiendo enlazar nuestro Apple ID existente a una cuenta de desarrollador o crear una nueva.
- Una vez llenados nuestros datos y verificada la cuenta, deberemos entrar dentro de ella.

# Publicar en AppleStore

---

- Durante este proceso deberemos prestar atención a varios puntos:
- En el registro ningún dato deberá contener tildes.
- Deberemos pagar la cuota anual de desarrollador de Apple.
- Existen varios planes según si por ejemplo queremos que nuestras apps se distribuyan sólo a través del Apple Store o también queremos que puedan ser instaladas directamente en el dispositivo sin pasar por él.
- La cuota básica que ronda unos 99\$.

# Publicar en AppleStore

---

- Deberemos aportar los datos acreditativos de empresa en el proceso o como individual, esto varía según el país.
- Tras realizar el pago, el proceso de registro quedará en espera para que Apple valide el nuevo usuario.
- Esta espera puede ser de unos pocos días, durante los cuales Apple puede ponerse en contacto con nosotros por teléfono para verificar o pedir aclaraciones sobre algún dato.

# Publicar en AppleStore

---

## iOS Provisioning portal: certificados

- Cuando Apple nos verifique, volveremos a entrar en el portal Apple Developer.
- El área iOS Provisioning portal nos permite correr apps en entorno de pruebas que todavía no han sido publicadas a través de certificados digitales, que existen de dos tipos:
  - **Development profiles:** sólo válidos para desarrollo
  - **Distribution profiles:** para distribución
- Para poder empezar, entraremos en iOS Provisioning portal / Certificates donde están las pestañas de Development y Distribution.
- Haremos una petición de certificado mediante el botón de Request Certificate.

# Publicar en AppleStore

---

- Volveremos al navegador web y subiremos el nuevo archivo generado.
- Nos devolverá a la página principal de certificados que el estado de nuestro certificado puede estar pendiente de aprobación.
- Esta petición se resuelve automáticamente en poco tiempo (máximo en algunos minutos), y cuando esté lista nos aparecerá un botón que nos permitirá la descarga del certificado.
- Una vez descargado, haremos doble click sobre el archivo y se importará a nuestro Keychain.

# Publicar en AppleStore

---

## iOS Provisioning portal: App ID y Provisioning

- Por cada app, deberemos crear un App ID que identificará exclusivamente a dicha app en iOS Provisioning portal -> Devices -> App ID
- En Provisioning haremos click en New profile donde crearemos un perfil para nuestro App ID.
- Entre las opciones deberemos prestar atención al Distribution Method: App Store o Ad-Hoc.
- Nos devolverá a la pantalla principal de Provisioning y veremos un botón de descarga del profile.
- Lo descargaremos e instalaremos en nuestro PC, abriremos Xcode y en Organizer, veremos que en sus Provisioning profiles tenemos ya preparado el perfil para nuestra App.

---

# Módulo 13: Proyecto web

pue

# Práctica / Proyecto

---

- Crear una aplicación web relacionada con nuestra tienda.
- La tienda y sus productos quedan a vuestra elección.
- En la pagina principal debemos tener:
  - Lista de navegación para ver los productos por categorías y el carrito de la compra
  - Una zona central donde :
    - 1)enseñar ofertas del mes
    - 2)describiremos nuestra tienda con texto, fotos, vídeo ...
    - 3) Enlace para ir a otra pagina donde se muestre un mapa con la ubicación de la tienda
  - un pie de página con copiright e info. De contacto

# Práctica / Proyecto

---

- Navegando por las categorías de nuestra tienda deben aparecer los productos con el precio y la opción de añadirlo al carrito de la compra
- La lista de navegación y carrito debe mostrarse en todas las páginas.
- Para realizar la compra:
  - 1) pinchar en el carrito, donde aparezca la lista de productos con su precio individual y su precio total
  - 2) El coste total de la compra
  - 3) La posibilidad de eliminar productos del carrito
  - 4) Botón para comprar

# Práctica / Proyecto

---

- Para finalizar la compra mostramos un formulario donde pedimos:
  - Dirección de entrega
  - email
  - Teléfono de contacto
  - Persona de contacto
  - botón de compra que muestre una alerta con la compra enviada, vacie el carrito y lleve a una pagina que muestre los datos del pedido



# pue

IMPULSANDO EL CONOCIMIENTO  
TIC CUALIFICADO