

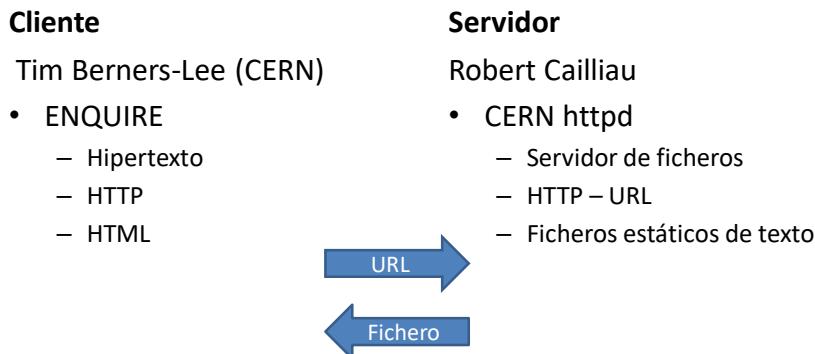
# HTML & CSS

© JMA 2015. All rights reserved

## EVOLUCIÓN DEL DESARROLLO WEB

© JMA 2015. All rights reserved

# 1990



© JMA 2015. All rights reserved

## Origen

- La Web no fue concebida para el desarrollo de aplicaciones. El problema que se pretendía resolver su inventor, Tim Berners-Lee, era el cómo organizar información a través de enlaces.
- De hecho la Web nació en el laboratorio de partículas CERN básicamente para agrupar un conjunto muy grande de información y datos del acelerador de partículas que se encontraba muy dispersa y aislada.
- Mediante un protocolo muy simple (HTTP), un sistema de localización de recursos (URL) y un lenguaje de marcas (HTML) se podía poner a disposición de todo científico en el mundo la información existente en el CERN de tal forma que mediante enlaces se pudiese acceder a información relacionada con la consultada.

© JMA 2015. All rights reserved

# HTML

## Cliente

- Navegadores
  - HTML

## Servidor

- Servidor Web
  - Servidor de ficheros
  - Ficheros estáticos
    - Texto
    - HTML
  - Ficheros dinámicos
    - CGI (C, Perl)
    - ...
    - Servlet



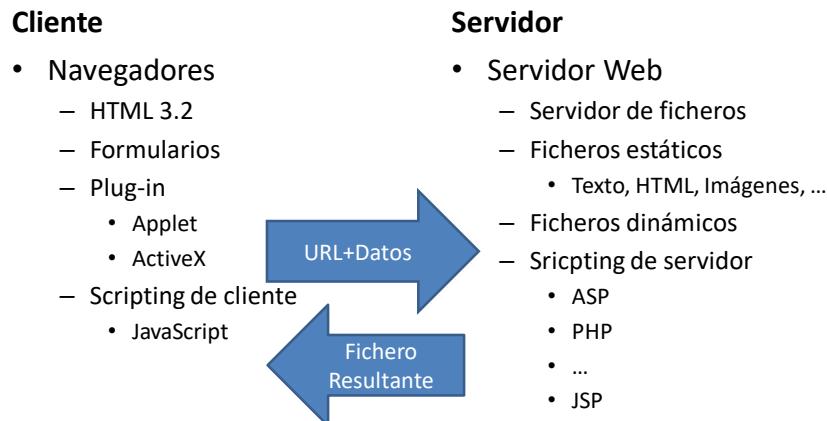
© JMA 2015. All rights reserved

# CGI

- Por la necesidad que el servidor Web pudiese devolver páginas Web dinámicas y no únicamente contenido estático residente en ficheros HTML se desarrolló la tecnología CGI (Common Gateway Interface) donde el servidor Web invocaba un programa el cual se ejecutaba, devolvía la página Web y el servidor Web remitía este flujo de datos al navegador.
- Un programa CGI podía ser cualquier programa que la máquina pudiese ejecutar: un programa en C, o en Visual Basic o en Perl. Normalmente se elegía este último por ser un lenguaje de script el cual podía ser traslado con facilidad de una arquitectura a otra. CGI era únicamente una pasarela que comunicaba el servidor Web con el ejecutable que devolvía la página Web.

© JMA 2015. All rights reserved

# Formularios



© JMA 2015. All rights reserved

# Scripting

- CGI era una solución cómoda de realizar páginas Web dinámicas pero tenía un grave problema de rendimiento que lo hizo insostenible en cuanto la demanda de la Web comenzó a disparar las peticiones de los servidores Web.
- Para agilizar esto, los principales servidores Web del momento (Netscape e IIS) desarrollaron un sistema para la ejecución dinámica de aplicaciones usando el propio contexto del servidor Web. En el caso de Netscape se le denominó NSAPI (Netscape Server Application Program Interface) y en el caso de IIS se le llamó ISAPI.

© JMA 2015. All rights reserved

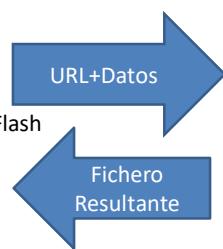
# Web 2.0

## Cliente

- Navegadores
  - HTML 4
  - CSS
  - DOM
  - AJAX
  - RIA
    - Shockwave Flash
    - Silverlight
  - JS Framework

## Servidor

- Servidor Web
  - Servidor de ficheros
  - Ficheros estáticos
  - Ficheros dinámicos
  - Sricpting de servidor
  - Servidor de aplicaciones
    - ASP.NET
    - J2EE
- Web Services
  - WS XML
  - RestFul



© JMA 2015. All rights reserved

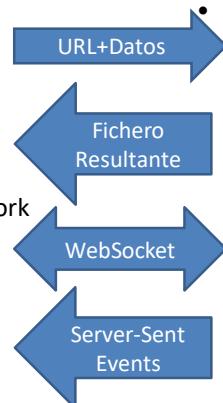
# Actualidad

## Cliente

- Navegadores
  - HTML 5
  - CSS 3
  - RIA
  - Móviles
  - JS RIA Framework
  - EcmaScript 6

## Servidor

- Servidor Web
  - Servidor de ficheros
  - Ficheros estáticos
  - Ficheros dinámicos
  - Sricpting de servidor
  - Servidor de aplicaciones
    - NodeJS
- Web Services
- Server-Sent Events
  - Notificaciones PUSH



© JMA 2015. All rights reserved

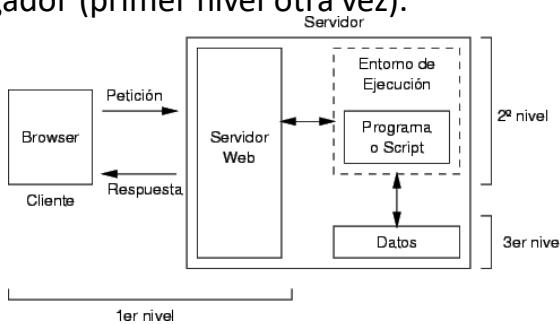
## Aplicación Web

- Conjunto de páginas que residen en un directorio web y sus subdirectorios.
- Cualquier página puede ser el punto de entrada de la aplicación, no se debe confundir con el concepto de página principal con el de página por defecto.
- Externamente, no existe el concepto de aplicación como entidad única.
- Internamente, dependiendo de la tecnología utilizada una aplicación puede ser una entidad única o una colección de objetos independientes.

© JMA 2015. All rights reserved

## Arquitectura

- Una aplicación Web típica recogerá datos del usuario (primer nivel), los enviará al servidor, que ejecutará un programa (segundo y tercer nivel) y cuyo resultado será formateado y presentado al usuario en el navegador (primer nivel otra vez).



© JMA 2015. All rights reserved

# Ventajas e inconvenientes

## Ventajas

- Inmediatez y accesibilidad
- No ocupan espacio local
- Actualizaciones inmediatas
- No hay problemas de compatibilidad
- Multiplataforma
- Consumo de recursos bajo
- Portables
- Alta escalabilidad y disponibilidad

## Inconvenientes

- Interfaz de interacción con el usuario muy limitada
- Base tecnológica inadecuada
- Incompatibilidades entre navegadores
- Bajo rendimiento al ser interpretado
- Cesión tecnológica
- Seguridad menos robusta

© JMA 2015. All rights reserved

# Single-page application (SPA)

- Un single-page application (SPA), o aplicación de página única es una aplicación web o es un sitio web que utiliza una sola página con el propósito de dar una experiencia más fluida a los usuarios como una aplicación de escritorio.
- En un SPA todo el códigos de HTML, JavaScript y CSS se carga de una sola vez o los recursos necesarios se cargan dinámicamente cuando lo requiera la página y se van agregando, normalmente como respuesta de las acciones del usuario.
- La página no se tiene que cargar otra vez en ningún punto del proceso, tampoco se transfiere a otra página, aunque las tecnologías modernas (como el pushState() API del HTML5) pueden permitir la navegabilidad en páginas lógicas dentro de la aplicación.
- La interacción con las aplicaciones de página única pueden involucrar comunicaciones dinámicas con el servidor web que está por detrás, habitualmente utilizando AJAX o WebSocket (HTML5).

© JMA 2015. All rights reserved

## Estado actual de la adopción de los nuevos estándares

- <http://caniuse.com/>
- HTML 5
  - <http://html5test.com>
  - <http://html5demos.com>
- CSS
  - <http://css3test.com/>
- EcmaScript
  - <https://kangax.github.io/compat-table/es6/>
- Navegadores mas utilizados
  - <http://www.netmarketshare.com/>

© JMA 2015. All rights reserved

Hyper Text Markup Language

**HTML**

© JMA 2015. All rights reserved

## ¿Qué es el HTML?

- Para publicar información y distribuirla globalmente, se necesita un lenguaje entendido universalmente, una especie de lengua franca de publicación que todas las computadoras puedan comprender potencialmente.
- El HTML (acrónimo de HyperText Markup Language, Lenguaje para el Formato de Documentos de Hipertexto) es un vocabulario de marcas textuales que permite dar el formato de presentación a los documentos de texto.
- Es el lenguaje de publicación usado por la World Wide Web.

© JMA 2015. All rights reserved

## Objetivos

- El HTML da a los autores las herramientas para:
  - Publicar documentos en línea con encabezados, textos, tablas, listas, fotos, etc.
  - Obtener información en línea a través de vínculos de hipertexto, haciendo clic con el botón de un ratón.
  - Diseñar formularios para realizar transacciones con servicios remotos, para buscar información, hacer reservas, pedir productos, etc.
  - Incluir hojas de cálculo, videoclips, sonidos, y otras aplicaciones directamente en sus documentos.

© JMA 2015. All rights reserved

## Historia

- El HTML fue desarrollado originalmente por Tim Berners-Lee mientras estaba en el CERN, y fue popularizado por el navegador Mosaic desarrollado en el NCSA.
- Durante los años 90 fue proliferado con el crecimiento explosivo de la Web.
- Durante este tiempo, el HTML se ha desarrollado de diferentes maneras.
- La Web depende de que los autores de páginas Web y las compañías compartan las mismas convenciones de HTML.
- Esto ha motivado el trabajo colectivo en las especificaciones del HTML.
- La estandarización recayó inicialmente en la Internet Engineering Task Force (IETF) para pasar posteriormente a manos del Grupo de Trabajo HTML del World Wide Web Consortium (W3C).

© JMA 2015. All rights reserved

## Especificaciones

- 1991 – HTML (primera mención)
- 1993 – HTML (primera versión publica - IETF)
- 1995 – HTML 2 – W3C
- 1997 – HTML 3.2
- 1997 – HTML 4 – CSS
- 1999 – HTML 4.01
- 2001 – XHTML
- 2014 – HTML5

© JMA 2015. All rights reserved

# Documentos HTML

- Un documento HTML es un archivo de texto que contienen etiquetas de marcado.
- Las etiquetas de marcado indican al navegador como mostrar la página.
- Los archivos HTML deben tener la extensión htm o html
  - .html es la preferida
  - .htm es solo para los viejos sistemas operativos que sólo puede utilizar nombres de "8 + 3" (ocho caracteres, punto, tres caracteres)
- Los archivos HTML se pueden crear con cualquier editor de texto simple.
  - Los procesadores de texto, como el Microsoft Word's, introducen caracteres especiales (formato) por lo que no pueden ser usados para crear ficheros HTML (salvo exportación).
- Para los contenidos no textuales, el HTML dispone de etiquetas que referencian dichos contenidos como ficheros externos.

# Etiquetas

```
<NombreDeEtiqueta atributos>
... Contenido ...
</NombreDeEtiqueta>
```

- Se utilizan para marcar los elementos HTML y se encierran entre paréntesis angulares
- La mayoría de las etiquetas HTML vienen en pares (etiqueta inicial y etiqueta final)
- El texto entre las etiquetas de inicio y fin es el contenido del elemento
- Las etiquetas actúan como contenedores (que contienen el contenido del elemento) y deben estar correctamente anidados
- Los nombres de las etiquetas y atributos pueden ir indistintamente en mayúsculas y minúsculas, aunque por norma de estilo se recomienda que se utilicen las minúsculas.

## Observaciones

- En determinados casos la aparición de una etiqueta cierra a la etiqueta anterior.
- El formato del contenido se ignora (excepto en la etiqueta <PRE>): no interpreta los tabuladores ni los saltos de línea, son sustituidos por un espacio y si aparecen varios espacios en blanco consecutivos se interpretan como si fueran uno solo.
- Los nombres de las etiquetas y sus atributos vienen definidos en la versión de HTML.
- Las etiquetas desconocidas por el navegador son ignoradas, mostrándose directamente el contenido.

© JMA 2015. All rights reserved

## Atributos

- Los atributos personalizan las etiquetas y son opcionales.
- Los atributos deben ir dentro de la etiqueta de comienzo, a continuación del nombre del elemento y separados por espacios en blanco.
- Los atributos cuentan con dos formatos:
  - NombreDeAtributo (aparece solo e indica que la etiqueta cuenta con dicho atributo, atributos booleanos)
  - NombreDeAtributo=Valor (el atributo toma el valor asignado).
- El valor debe encontrarse delimitado por apostrofes ('') o comillas (""), permitiendo el uso del otro delimitador dentro del valor.  
`<input type="checkbox" name="chkAcepto" checked>`

© JMA 2015. All rights reserved

# Entidades HTML

- En el código fuente no se puede utilizar determinados caracteres que tienen significado para el HTML o que no pertenecen al juego de caracteres utilizado por el inglés.
- No es posible utilizar el signo menor que (<) o mayor que (>) en el texto, ya que el navegador los asocian con las etiquetas.
- Para que el navegador muestre el carácter correspondiente se debe usar las entidades de caracteres en el código fuente HTML.
- Formatos:
  - &nombre\_entidad;
  - &#valor\_numérico\_del\_carácter;
  - &letra\_o\_vocal\_mnemotécnico;
- Mnemotécnicos:
  - acute (á), grave (è), circ (í), uml (Ü), tilde (Ñ), ring (å), ...

© JMA 2015. All rights reserved

## Nombres de Entidades

Resultado	Descripción	Nombre Entidad	Número Entidad
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
€	euro	&euro;	&#8364;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
™	trademark	&trade;	&#8482;

© JMA 2015. All rights reserved

## Entidades habituales

&aacute;	á	&Aacute;	Á
&eacute;	é	&Eacute;	É
&iacute;	í	&Iacute;	Í
&oacute;	ó	&Oacute;	Ó
&uacute;	ú	&Uacute;	Ú
&uuml;	ü	&Uuml;	Ü
&ntilde;	ñ	&Ntilde;	Ñ
&#191;	Símbolo ¿	&#161;	Símbolo ¡

© JMA 2015. All rights reserved

## Espacio en blanco

- En HTML el espacio en blanco es cualquier carácter no imprimible (espacio, tabulador, salto de línea y algunos otros).
- HTML trata a todos los espacios en blanco como separadores de palabras y hace fluir automáticamente el texto de una línea a la siguiente, dependiendo del ancho de la página.
- Si aparecen varios espacios en blanco consecutivos se interpretan como si fueran uno solo.
- Para grupos de texto en párrafos, con salto de línea entre párrafos, se encierra cada párrafo entre etiquetas `<p>` y `</p>`
- Para forzar HTML ha utilizar espacios en blanco exactamente como se escribió en el documento se encierra cada el texto entre etiquetas `<pre>` y `</pre>` etiquetas ("pre" significa "con formato previo")
- Para forzar el salto de línea se utiliza la etiqueta `<br>`
- Cuando se quieren conservar los espacios se sustituyen por

© JMA 2015. All rights reserved

## Comentarios

- Los comentarios son ignorados por el navegador.
- Los comentarios empiezan por <!-- y terminan con -->
- Pueden contener parte de una línea, una línea o varias líneas.
- Pueden ir en cualquier parte del documento salvo en mitad de una etiqueta de comienzo o de final.

```
<!-- Mi comentario -->
...
<!--
<table class="newstable">
...
-->
```

© JMA 2015. All rights reserved

## Codificación

- El HTML es sólo la estructura, no la apariencia, que depende del navegador.
- El código fuente HTML debe estar formateado para aumentar la legibilidad y facilitar la depuración.
- Cada elemento de bloque debe comenzar en una nueva línea.
- Cada elemento (bloque) anidado debe ir correctamente indentado.
- Los navegadores, al interpretar el código fuente de la página, ignoran espacios en blanco múltiples por lo que el formato es inofensivo.
- Solo por motivos de rendimiento el formato debe ser sacrificado

© JMA 2015. All rights reserved

## Diferencias del XHTML con HTML

- Los nombres de elementos y atributos deben escribirse en minúsculas.
- Todos los valores de los atributos deben ir entrecomillados.
- Todos los elementos "no vacíos" deben ir entre la etiqueta de principio y la etiqueta de final.
- Todos los elementos deben estar anidados ordenadamente.
- Minimización de los atributos: atributo="atributo".
- Los elementos "vacíos" deben llevar terminación: <hr/> <br/>.

© JMA 2015. All rights reserved

## ESTRUCTURA

© JMA 2015. All rights reserved

## Estructura de documento

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN">  
<HTML>  
  <HEAD>  
    ... Etiquetas de cabecera ...  
  </HEAD>  
  <BODY>  
    ... Cuerpo del documento ...  
  </BODY>  
</HTML>
```

© JMA 2015. All rights reserved

## Información sobre la versión de HTML

- Los documentos HTML deben comenzar con una definición de tipo de documento (DTD)
  - Informa a los navegadores sobre la versión de HTML utilizada en el documento
  - Posibles versiones: HTML 4.01, XHTML 1.0 (Transitional or Strict), XHTML 1.1, HTML 5

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

© JMA 2015. All rights reserved

## Etiqueta HTML

- Marca el comienzo y final del documento.
- Atributos opcionales:
  - LANG: idioma del documento (Ej. LANG = "es")
  - DIR: Dirección del texto. Posibles valores:
  - LTR: De izquierda a derecha (por defecto).
  - RTL: De derecha a izquierda.

© JMA 2015. All rights reserved

## Sección de cabecera

- Contiene información adicional que no aparece directamente en la página visible destinada a los navegadores, buscadores y otras herramientas.
  - TITLE, META, BASE, LINK, BASEFONT, STYLE, SCRIPT
- La etiqueta HEAD marca el comienzo y final de la cabecera, el resto de las etiquetas de la cabecera deben aparecer entre las marcas.
- Atributos opcionales: LANG y DIR.

```
<HEAD>
... Etiquetas de la cabecera ...
</HEAD>
```

© JMA 2015. All rights reserved

## Etiqueta TITLE

- Titulo de la página que identifica la página y normalmente aparece en la barra de título del navegador o de la solapa.
- Aunque opcional, es conveniente que todas las páginas tengan título.
- Atributos opcionales: LANG y DIR.

```
<title>El titulo de mi pagina</title>
```

© JMA 2015. All rights reserved

## Etiqueta BASE

- Especifica la URL de partida para las referencias relativas y el destino.
- Target: \_blank, \_parent, \_self, \_top, *framename*

```
<base href="http://www.mydomain.com/images/" />
<base target="_blank" />
```

© JMA 2015. All rights reserved

## Etiquetas META

- Permite al autor definir datos sobre el documento (author, copyright, keywords, date, review, security), dichos datos podrán ser tratados electrónicamente por motores de páginas WEB (buscadores, indexadores, servidores, ...). El documento puede tener tantas entradas META como se desee.
- Define pares de nombre y descripción, y opcionalmente, el idioma usado con motores de búsqueda.  
`<META NAME="Identificador" LANG="es" CONTENT="Valor o descripcion">`
- El atributo HTTP-EQUIV puede utilizarse en sustitución de NAME que tiene un significado especial cuando los documentos se transmiten vía HTTP. Los servidores HTTP utilizarán este atributo para generar una cabecera de estilo RFC-822 en la respuesta HTTP. Por ejemplo:  
`<META HTTP-EQUIV="Expires" CONTENT="Tue, 20 Aug 1996 14:25:27 GMT">`
- y como resultado generará la cabecera:  
`Expires: Tue, 20 Aug 1996 14:25:27 GMT`

© JMA 2015. All rights reserved

## Etiquetas LINK

- Permite establecer relaciones con otras páginas, posicionando a la página dentro de un documento extenso.
- Los atributos son:
  - REL: Tipo de relación con el documento.
  - HREF: Especifica la página enlazada.
  - HREFLANG: Idioma de la página relacionada. (opcional)
  - TYPE: Tipo MIME de la página, necesario cuando no es "text/HTML".
  - REV: Especifica el tipo de relación inversa (procedencia REL de otras páginas). (opcional)
  - TITLE: Título del enlace. (opcional)
  - MEDIA: Indica el medio para el que está definido el estilo del enlace. Por defecto es "screen".

© JMA 2015. All rights reserved

## Etiquetas LINK

- Los tipos de relación son:
  - alternate, appendix, bookmark, chapter, contents, copyright, glossary, help, home, index, next, prev, section, start, **stylesheet**, subsection.
- Los tipos de medio son:
  - **screen**, tty, tv, projection, handheld, **print**, braille, **aural**, *all*

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

© JMA 2015. All rights reserved

## Etiqueta BASEFONT

- Define las características de la fuente utilizada por defecto en el cuerpo del documento.
- 
- Los atributos son:
  - SIZE= Tamaño de la fuente.
  - COLOR= Color de la fuente.
  - FACE= Familias de fuentes opcionales separadas por comas.
- Permite definir estilos para las etiquetas de la página. Ver Hojas de Estilo en Cascada (CSS).
- Etiquetas SCRIPT
- Permite incluir instrucciones ejecutables en la página. Ver Lenguajes de Comandos.

© JMA 2015. All rights reserved

## Etiquetas STYLE

- Permite definir estilos para las etiquetas de la página, utilizando Hojas de Estilo en Cascada (CSS).

```
<html>
  <head>
    <style type="text/css">
      p { font-size: 12pt; line-height: 12pt; }
      p:first-letter { font-size: 200%; }
      span { text-transform: uppercase; }
    </style>
  </head>
  <body>
    <p>Styles demo.<br />
      <span>Test uppercase</span>.
    </p>
  </body>
</html>
```

© JMA 2015. All rights reserved

## Etiquetas SCRIPT

- Permite incluir instrucciones en la página. Ver Lenguajes de Comandos.
- La etiqueta `<script>` se utiliza para incrustar secuencias de comandos ejecutables en un documento HTML
- Los scripts se ejecutan en el navegador del cliente
- Los scripts se pueden incluir tanto en la sección `<head>` como en el `<body>`.
- Los lenguajes compatibles de scripting del lado del cliente son:
  - JavaScript (no es Java!) - Estándar de facto
  - VBScript (obsoleto)
  - JScript (obsoleto)

© JMA 2015. All rights reserved

## Estructura con marcos

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
  <HEAD>
    ... Etiquetas de cabecera ...
  </HEAD>
  <FRAMESET>
    ... Conjunto de marcos ...
  </ FRAMESET >
</HTML>
```

© JMA 2015. All rights reserved

## HTML FRAMES

- Con los marcos se puede mostrar más de un documento HTML en la misma ventana del navegador.
- En cada marco se carga un documento HTML y cada marco es independiente de los otros.
- Las desventajas del uso de marcos son:
  - Dificultan el acceso a la información.
  - Producen documentos parciales.
  - Están marcados como obsoletos y desaparecerán en futuras versiones de HTML

```
<html>
  <head>
    <title>My Title</title>
  </head>
  <frameset cols="25%,75%">
    <frame src="frame_a.htm" />
    <frame src="frame_b.htm" />
  <noframes>
    ...
  </noframes>
</frameset>
</html>
```

© JMA 2015. All rights reserved

## CUERPO DEL DOCUMENTO

© JMA 2015. All rights reserved

### Sección del cuerpo

- El cuerpo contiene la información que el navegador muestra al usuario.
- Es un conjunto de etiquetas que formatean el contenido y la estructura del documento.
- Aunque muchas etiquetas disponen de atributos de representación visual no se recomienda su uso siendo preferible el uso de las hojas de estilos.
- Atributos comunes a todas las etiquetas.
  - ID= Identificador único de la etiqueta. El nombre debe ser único para el documento.
  - CLASS= Clase del estilo al que pertenece la etiqueta. Es el encargado de realizar el enlace entre las etiquetas y las hojas de estilos.
  - STYLE= Definición de estilo.
  - TITLE= Título o descripción de la etiqueta. Normalmente los navegadores lo muestran brevemente como un mensaje corto que aparece cuando el dispositivo señalador se detiene sobre la etiqueta.

© JMA 2015. All rights reserved

## Etiqueta BODY

- Engloba al cuerpo del documento.
- Atributos:
  - BACKGROUND = Dirección URL de la imagen de fondo.
  - BGCOLOR= Color de fondo del documento.
  - TEXT = Color del texto del documento.
  - LINK = Color de los enlaces no visitados.
  - VLINK = Color de los enlaces ya visitados.
  - ALINK = Color de resalte cuando el usuario se sitúa sobre el.

```
<html>
  <head><title>Test page</title></head>
  <body>
    <!-- This is the Web page body -->
  </body>
</html>
```

© JMA 2015. All rights reserved

## Encabezados y Párrafos

- H1, H2, H3, H4, H5, H6: Encabezados de mayor a menor nivel
- P: Párrafo
- PRE: Párrafo pre formateado
- BLOCKQUOTE Párrafo con cita literal
- Q: Párrafo con cita literal, entrecomillado
- ADDRESS: Información del autor

© JMA 2015. All rights reserved

# Listas

- UL: Lista no ordenada
- OL: Lista ordenada
- LI: Elemento de la lista

```
<ul>
<li>Azucar</li>
<li>Patatas</li>
<li>Galletas</li>
<li>Chocolate</li>
</ul>
```

- Listas de definiciones:
- DL: Lista de definiciones
  - DT: Término
  - DD: Definición

```
<DL>
<DT>Hacker
<DD>un programador
inteligente
<DT>Nerd
<DD>persona técnicamente
brillante pero socialmente inepto
</DL>
```

© JMA 2015. All rights reserved

# Organización del documento.

## Grupos de elementos

- DIV: Crea una división o capa en el documento (capítulo, sección, ...).
- SPAN: Identifica un segmento de información en una línea.
- CENTER: Equivale a <DIV ALIGN=“CENTER”>
- DEL : Texto borrado (no se visualiza).
- INS: Texto añadido.
- <HR> separador con una línea horizontal.
- <BR> Retorno de carro

## Tablas

- TABLE: contenedor
- CAPTION: título
- COLGROUP: Grupo de columnas
  - COL: columna
- THEAD: Cabecera
- TFOOT: Pie
- TBODY: Cuerpo
- TR: fila
  - TH: celda de cabecera
  - TD: celda de datos

© JMA 2015. All rights reserved

# Formatos de caracteres

## Estilos lógicos:

- DFM: Palabra a definir (itálica).
- EM: Palabra a enfatizar (itálica).
- CITE: Citas textuales (itálica).
- CODE: Fragmentos de código de programas (ancho fijo).
- KBD: Entrada por teclado (ancho fijo).
- SAMP: Ejemplo (ancho fijo).
- STRONG: Gran énfasis (negrita).
- VAR: Para una variable (itálica).
- ABBR: Abreviaturas.
- ACRONYM: Siglas

## Estilos físicos:

- B: Negrita.
- I: Itálica.
- TT: Ancho Fijo.
- U: Subrayado.
- STRIKE: Tachado.
- S: Tachado.
- BIG: Grande.
- SMALL: Pequeño.
- SUB: Subíndice.
- SUP: Superíndices.

© JMA 2015. All rights reserved

# Formatos de caracteres

## • Fuentes de letras:

- FONT: Define el tipo de fuente utilizado.
  - SIZE= Tamaño de la fuente.
  - COLOR= Color de la fuente.
  - FACE= Familias de fuentes opcionales separadas por comas.

© JMA 2015. All rights reserved

# Enlaces

- Permite saltar mediante un vínculo hipertexto a otro punto de la página o a otra página.
- <A HREF="...página...">...Texto o imagen del enlace...</A>
- El texto del enlace aparece automáticamente subrayado de azul (o de morado si se visitó recientemente)
- Los atributos son:
  - NAME: Nombre de etiqueta (para referencias dentro de la página #etiqueta).
  - HREF: Especifica la página o etiqueta enlazada.
  - HREFLANG: Idioma de la página relacionada.
  - TARGET: Marco de destino de la página solicitada.
  - TYPE: Tipo MIME de la página, necesario cuando no es “text/HTML”.
  - REL: Tipo de relación con el documento.
  - REV: Especifica el tipo de relación inversa (procedencia REL de otras páginas).

© JMA 2015. All rights reserved

# Enlaces

- Para enlazar a otra parte de la misma página:
  - Esto es un <a href = "http://www.examples.org/ejenlaces.html">ejemplo</a> de enlaces.
- Para enlazar a otra parte de la misma página:
  - Inserte un enlace con nombre: <a name="refs"> Referencias</a>
  - Y enlace a la misma con: <a href="#refs"> Mis referencias </a>
- Para enlazar a un anclaje con nombre de una página diferente:
  - <a href="PageURL#refs"> Mis referencias </a>
- Para enviar un correo electrónico:
  - <A HREF="mailto:direccion@email">...Texto para enviar mensaje...</A>
- Para recibir un fichero:
  - <A HREF="ftp:... URI ...">...Texto indicando la descarga...</A>

© JMA 2015. All rights reserved

# Imágenes

- Las imágenes (fotografías) no son parte de una página HTML, el código HTML sólo le dice dónde encontrar la imagen.
- Para añadir una imagen a una página se utiliza
  - <IMG SRC="URL" ALT="descripción de texto" WIDTH="150" HEIGHT="100">
- El atributo SRC es obligatorio, el resto son opcionales
- Los atributos pueden estar en cualquier orden
- La URL puede referirse a cualquier archivo .gif, .jpg o .png o
- Otros formatos gráficos pueden no ser reconocidos
- El atributo ALT proporciona una representación de texto de la imagen por si la imagen no se descarga o no se puede visualizar
- Los atributos de altura y anchura, si se incluye, mejorarán la representación en pantalla cuando se está descargando la página
- Si la altura o anchura es incorrecta, la imagen se distorsionará
- No hay ninguna etiqueta </img> final, porque <img> no es un contenedor

© JMA 2015. All rights reserved

# Enlaces en imágenes

- Los mapas permiten asignar vínculos hipertexto a determinadas áreas de la imagen. Cuando el usuario pulsa sobre la región se ejecuta el enlace.
- Un mapa de imagen se crea por la asociación de un objeto con una especificación de las áreas geométricas sensibles en el objeto.

```
<img src = "planets.gif" width="145" height="126" alt="Planets"
usemap = "#planetmap" />

<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun" />
  <area shape="circle" coords="9,5,3" href="mercur.htm" alt="Mercury" />
  <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus" />
</map>
```

© JMA 2015. All rights reserved

# Formularios

- Los formularios HTML se utilizan para crear interfaces que interactúen (GUIs muy básicos) en las páginas Web
  - Por lo general, el propósito es pedir al usuario información
  - La información se envía de vuelta al servidor
- Un formulario es un área que puede contener elementos de formulario
  - <FORM attributes> ...form elements... </FORM>
  - Los elementos de formulario son: botones, casillas de verificación, campos de texto, botones de radio, menús desplegables, etc.
  - Otros tipos de etiquetas HTML se pueden mezclar con los elementos del formulario
  - Un formulario por lo general contiene un botón Enviar (Submit) para reunir y enviar la información de forma automática al servidor.
  - Los parámetros del formulario indican cómo enviar la información al servidor (hay dos maneras diferentes que podría ser enviada)
  - Los formularios se pueden usar para otras cosas, como una interfaz gráfica de usuario para programas sencillos

© JMA 2015. All rights reserved

# Formularios y JavaScript

- El lenguaje JavaScript se puede utilizar para hacer que las páginas que "haga algo".
  - Se puede usar JavaScript para escribir programas completos, pero ...
  - Por lo general, sólo se tienen que utilizar fragmentos de JavaScript aquí y allá a través de la página Web
- Los fragmentos de código JavaScript pueden interactuar con los elementos del formulario
  - Por ejemplo, es posible comprobar que el campo de código postal contiene un número entero de 5 dígitos antes de enviar la información al servidor
- Los formularios HTML se puede utilizar sin JavaScript y el JavaScript puede ser utilizado sin formularios HTML, pero trabajan bien juntos.

© JMA 2015. All rights reserved

## Etiqueta FORM

- <FORM atributos> ... </FORM>
  - ACTION = Dirección URI del recurso que va a procesar la información enviada en el formulario.
  - METHOD = Método de envío del formulario.
    - GET: Monta una nueva URL que partiendo del valor de ACTION le añade un interrogante (?) y a continuación va añadiendo los datos del formulario (en pares de Nombre del control = Valor) y separados por ampersand (&). Envía los datos en la cabecera de la petición HTTP (el tamaño máximo de la cabecera HTTP es 1Kb).
    - POST: Monta una estructura con los datos del formulario y la envía en el cuerpo de la petición HTTP, no tiene restricción de tamaño.
  - ENCTYPE = Formato en que se codifican los datos para su envío (por defecto: "application/x-www-form-urlencoded"). Para enviar ficheros se utiliza "multipart/form-data".
  - NAME = Nombre del formulario.
  - TARGET = Destino de la respuesta del formulario.

© JMA 2015. All rights reserved

## Elementos de formularios

- La mayoría de los controles de formulario están encapsulados en la etiqueta <INPUT> que toma diferentes formas.
  - NAME = Nombre del control.
  - ALT= Texto alternativo para el control
  - TYPE = Tipo de interfaz de entrada
  - SIZE = Tamaño en caracteres de visualización de la caja.
  - MAXLENGTH = Número máximo de caracteres que acepta la caja.
  - CHECKED : Indica que el control está seleccionado (CHECKBOX y RADIO).
  - READONLY : El valor del control no puede ser modificado.
  - DISABLED : El control se encuentra deshabilitado, no se puede modificar y no recibe el foco.
  - TABINDEX= Entero que indica el orden de elemento en la tabulación.
  - ACCESSKEY= Tecla de acceso rápido.
  - VALUE = Valor de la cadena.

© JMA 2015. All rights reserved

# Tipos de INPUT

- TYPE = Tipo de interfaz de entrada:
  - TEXT : Caja de texto de una sola línea.
  - PASSWORD : Caja de texto de una sola línea don los caracteres se visualizan como asteriscos (\*).
  - FILE : Caja de texto acompañada de un botón para la selección de un fichero local.
  - CHECKBOX : Caja de chequeo.
  - RADIO : Botón de radio, permite seleccionar una opción entre varias. Están agrupados todos los que tienen el mismo NAME.
  - SUBMIT : Botón que desencadena el envío del formulario.
  - RESET : Botón que reinitializa el formulario.
  - IMAGE : Similar a SUMIT pero sustituye la estética de botón por una imagen.
  - BUTTON : Botón sin funcionalidad asociada.
  - HIDDEN : Oculto, el valor no se muestra.

© JMA 2015. All rights reserved

## Entradas de texto

Campos de texto:

```
<input type="text" name="textfield" value="with an initial value">
A text field: 
```

Campos de contraseña:

```
<input type="password" name="textfield3" value="secret">
A password field: 
```

Campos de texto con múltiples líneas:

```
<textarea name="textarea" cols="24" rows="2">Hello</textarea>
A multi-line text field 
```

© JMA 2015. All rights reserved

## Botones

- Botón de enviar:  
`<input type="submit" name="btn1" value="Submit">`
  - Botón de reinicializar:  
`<input type="reset" name="btn2" value="Reset">`
  - Botón plano:  
`<input type="button" name="btn3" value="Push Me">`
  - submit: envía los datos
  - reset: restaura todos los elementos del formulario a su estado inicial
  - button: la acción se especifica mediante JavaScript
- A submit button:
- A reset button:
- A plain button:

© JMA 2015. All rights reserved

## Checkboxes

- A checkbox `<input type="checkbox" name="checkbox" value="checkbox" checked>`
- A checkbox:
- type: "checkbox"
  - value: Valor asociado al control cuando se encuentre seleccionado, solo se envía al servidor si está seleccionado.
  - El control es solo la caja de verificación sin texto asociado por lo que hay que envolverlo con otras etiquetas HTML.

© JMA 2015. All rights reserved

## Botones de radio

Radio buttons:

```
<input type="radio" name="radiobutton"
value="myValue1">male<br>
<input type="radio" name="radiobutton"
value="myValue2" checked>female<br>
```

Radio buttons:  
 male  
 female

- Si uno o mas botones de radio tienen el mismo NAME (grupo), el usuario solo puede seleccionar uno de ellos, desmarcándose automáticamente el resto.
- Al servidor se envía el NAME común asociado al VALUE del INPUT seleccionado.
- Al igual que los checkboxes, los radio buttons no contienen texto indicativo.

© JMA 2015. All rights reserved

## Menús desplegables

- Un menú o lista:

```
<select name="select">
  <option value="red">red</option>
  <option value="green">green</option>
  <option value="BLUE">blue</option>
</select>
```

A menu or list: 

- Atributos adicionales:

- size: el número de elementos visibles de la lista (por defecto es "1")
  - 1 → Combobox
  - > 1 → Listbox
- multiple: Si es "true", se pueden seleccionar varios elementos (por defecto es "false")

© JMA 2015. All rights reserved

## Campos ocultos

A hidden field: <input type="hidden" name="hiddenField" value="nyah">&lt;-- right there, don't you see it?

A hidden field: <-- right there, don't you see it?

- Utilidad:
  - El navegador no muestra los campos ocultos al pintar la página pero se pueden ver al mostrar el código de la página.
  - Todos los campos del formulario se envían de vuelta al servidor, incluyendo los campos ocultos.
  - Permiten a los servidores enviar información para que luego les sea devuelta sin modificar y sin mostrarla.

© JMA 2015. All rights reserved

## Ejemplo completo

- <html>
 <head>
 <title>Get Identity</title>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
 </head>
 <body>
 <p><b>Who are you?</b></p>
 <form method="post" action="">
 <p>Name:<br/>
 <input type="text" name="textfield">
 </p>
 <p>Gender:<br/>
 <input type="radio" name="gender" value="m">Male
 <input type="radio" name="gender" value="f">Female</p>
 </form>
 </body>
 </html>

Who are you?

Name:

Gender:  Male  Female

© JMA 2015. All rights reserved

## Otros controles de formularios

- <INPUT TYPE="FILE" > Define el mecanismo de seleccionar y enviar ficheros al servidor
- <INPUT TYPE="IMAGE" > Define una imagen como si fuera un botón SUBMIT.
- <BUTTON> Otra forma de definir los botones.
- <OPTGROUP> Define un grupo de opciones relacionadas en un listado del <SELECT>
- <LABEL> Etiqueta de campo. Asocia un literal a un control de entrada de datos.
- <FIELDSET> Agrupa a un conjunto de elementos del formulario, enmarcándolos en un recuadro.
- <LEGEND> Permite que aparezca un literal en el marco del FIELDSET.

© JMA 2015. All rights reserved

## HTML Helpers (Plug-Ins)

- Un documento puede ampliar la funcionalidad del navegador con la inclusión de aplicaciones Java y objetos.
- Dichas aplicaciones de ayuda se inician usando la etiquetas <object>.
- Etiquetas:
  - OBJECT: Objetos genéricos embebidos.
    - PARAM: Valor de personalización
  - APPLET: Java applet (obsoleta)  
<OBJECT codetype="application/java" classid="MyApplet" ...>

© JMA 2015. All rights reserved



© JMA 2015. All rights reserved

## Estado actual de la adopción de los nuevos estándares

- 
- <http://caniuse.com/>
  - HTML 5
    - <http://html5test.com>
    - <http://html5demos.com>
  - CSS
    - <http://css3test.com/>
  - EcmaScript
    - <https://kangax.github.io/compat-table/es6/>
  - Navegadores mas utilizados
    - <http://www.netmarketshare.com/>

---

© JMA 2015. All rights reserved

# HTML5

© JMA 2015. All rights reserved

## Objetivos

- La última versión es HTML5
  - Dirigido a tener todo el poder de las aplicaciones nativas
  - Ejecutable en cualquier plataforma (Windows, Linux, iPhone, Android, etc.)
- Las nuevas características se deben basar en HTML, CSS, DOM y JavaScript
- Reducir la necesidad de plugins externos
- Mejor manejo de errores
- Más marcado para reemplazar el scripting

© JMA 2015. All rights reserved

## Enlaces de interés

- [http://www.w3.org/TR/#tr\\_HTML](http://www.w3.org/TR/#tr_HTML)
- <http://html5test.com>
- <http://caniuse.com/>
- <http://html5demos.com>
- <https://validator.w3.org/>

© JMA 2015. All rights reserved

## Cambios en las etiquetas

- Etiqueta Doctype:

```
<!DOCTYPE html>
```

- Etiqueta HTML:

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

- Etiqueta Meta:

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- Etiqueta Link:

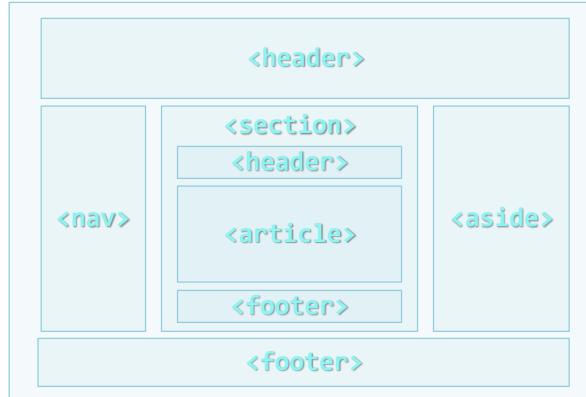
```
<link rel="stylesheet" href="style-original.css">
```

© JMA 2015. All rights reserved

## Nueva organización

Nuevas etiquetas que permiten estructurar el documento:

- <section>
- <header>
- <nav>
- <article>
- <aside>
- <footer>



© JMA 2015. All rights reserved

## Nueva organización

- Elementos como encabezado y pie de página no están destinados a ser únicamente la parte superior e inferior de la página
  - Puede haber encabezado y pie de página de cada sección del documento
- No son muy diferentes de la etiqueta <DIV> pero permiten definir semánticamente mejor la estructura del documento
- Adecuan la estructura a las tendencias actuales de los documentos.

© JMA 2015. All rights reserved

# Nuevas etiquetas

- <article>: Para el contenido externo, como el texto de un artículo de noticias, blog, foro o cualquier otra fuente externa
- <aside>: Para otros contenidos separados pero relacionados con el contenido principal
- <summary>: Un subtítulo o un resumen, dentro del elemento detalles
- <nav>: Para una sección de la navegación
- <section>: Para una sección en un documento (por ejemplo, capítulos, encabezados, pies de página)
- <main> Representa el contenido principal del cuerpo de un documento o aplicación.
- <details>: Para la descripción de los detalles de un documento o partes de un documento
- <figure> Para representar un cierto contenido adicional de flujo principal del documento, opcionalmente con una leyenda <figcaption> auto-contenida (como una oración completa)
- <wbr>: Punto de ruptura de palabra. Para la definición de un lugar apropiado para partir una palabra larga o sentencia al cambiar de línea

© JMA 2015. All rights reserved

## Etiqueta HGROUP

- La etiqueta <hgroup> es usada para agrupar un conjunto de elementos h1–h6, por ejemplo, cuando tenemos un título y a continuación una pequeña descripción o subtítulo.

```
<section>
  <hgroup>
    <h1>Título de la sección</h1>
    <h2>Subtítulo</h2>
  </hgroup>
  <p>Contenido de la sección</p>
</section>
```

© JMA 2015. All rights reserved

## Doble formato

- Para proporcionar tanto un valor legible por máquina para los fines de procesadores de datos como su valor legible por humanos para los fines de representación en un navegador Web aparecen dos nuevas etiquetas:

```
<DATA VALUE='1'>Primero</DATA>
<TIME DATETIME= "2007-08-02T23: 30Z">
Vie, 03 de agosto 2007 a las 09:30
</ Time>
```

© JMA 2015. All rights reserved

## Etiqueta MARK

- Representa el resultado de texto en un documento marcado con fines de referencia, debido a su relevancia en otro contexto.
- Agrega impacto en el texto resaltándolo con un color brillante.
- Permite destacar el resultado de las búsquedas, los puntos de interés o resaltados.

© JMA 2015. All rights reserved

## Etiqueta MARK

The highlighted part below is where the error lies:

```
var i: Integer;
begin
    i := 1.1;
end.
```

<p>The highlighted part below is where the error lies:</p>

```
<pre><code>var i: Integer;
begin
    i := <mark>1.1</mark>;
end.</code></pre>
```

© JMA 2015. All rights reserved

## Etiqueta RUBY

- Permite que uno o más tramos de contenido estático a estar marcados con anotaciones de Ruby.
- Las anotaciones de Ruby son tiras cortas de texto que se presentan junto texto base y se utiliza principalmente en la tipografía de Asia Oriental como una guía para la pronunciación o incluir otras anotaciones.
- En japonés, esta forma de la tipografía también se conoce como furigana.
- El texto Ruby puede aparecer en cualquiera de los lados y, a veces, en ambos lados del texto base

にほんご　か　さくぶん  
日本語で書いた作文です。

きょう
今日
group

© JMA 2015. All rights reserved

# Formularios

- Los controles en HTML4 son muy limitados
- Nuevos controles
  - Nuevos tipos del INPUT
  - Nuevas etiquetas
- Nuevos atributos de validación
- Nuevos APIs
  - APIs for the text field selections
  - Constraint validation API

© JMA 2015. All rights reserved

# Formularios

- Existen nuevos atributos para el elemento <form>:
  - autocomplete: Este es un viejo atributo que se ha vuelto estándar en esta especificación. Puede tomar dos valores: on y off. El valor por defecto es on. Puede ser implementado en el elemento <form> o en cualquier elemento <input> independientemente.
  - novalidate Los formularios son automáticamente validados. Para evitar este comportamiento, podemos usar el atributo novalidate. Para lograr lo mismo para elementos <input> específicos, existe otro atributo llamado formnovalidate.
- Se han ampliado los tipos del elemento <input> (atributo type)

© JMA 2015. All rights reserved

# eMail y URLs

`lachlan.hunt@lachy.id.au`

`http://lachy.id.au`

`http://lachy.id.au/`

`http://lachy.id.au/log/`

Lachlan Hunt: Web Development Guru

Lachy's Log

- `<input type="email">`
- `<input type="url">`

© JMA 2015. All rights reserved

# Fecha y hora

2007-03-07	13:15	UTC					
March	2007						
Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
9	26	27	28	1	2	3	4
10	5	6	7	8	9	10	11
11	12	13	14	15	16	17	18
12	19	20	21	22	23	24	25
13	26	27	28	29	30	31	1
14	2	3	4	5	6	7	8
<input type="button" value="Today"/>				<input type="button" value="None"/>			

- `date`: para introducir una fecha (mes, día y año).
- `time`: para introducir una hora en el formato de 24 horas (hora, minutos y segundos) y con información de zona horaria.
- `datetime`: para introducir una fecha y una hora.
- `datetime-local`: igual que el anterior pero sin la posibilidad de indicar una zona horaria, ya que se entiende que es la zona local.
- `month`: para introducir un mes de un año.
- `week`: para introducir una determinada semana en un año.

© JMA 2015. All rights reserved

## Números



- `<input type="number">`
- `<input type="range">`
- Algunos atributos nuevos que pueden ser útiles para este campo:
  - min El valor de este atributo determina el mínimo valor aceptado para el campo.
  - max El valor de este atributo determina el máximo valor aceptado para el campo.
  - step El valor de este atributo determina el tamaño en el que el valor será incrementado o disminuido en cada paso.

© JMA 2015. All rights reserved

## Etiqueta METER

- Permite la representación gráfica de las mediciones escalares o valores fraccionarios

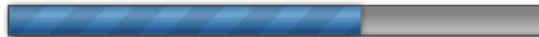
Rating:

- `<meter value="0.6">60%</meter>`
- `<meter value="3" min="0" max="5">3/5</meter>`
- `<meter value="6" min="1" max="10">6 blocks used (out of 10 total)</meter>`
- `<meter value="0.6">Medium</meter>`

© JMA 2015. All rights reserved

## Etiqueta PROGRESS

- Para mostrar progreso realización de una tarea
- Las barras de progreso son ampliamente utilizados en otras aplicaciones
- Funciona con aplicaciones de secuencias de comandos



- <progress value="22" max="100"></progress>
- <progress value="3" max="6">Step 3 of 6</progress>
- <progress>Loading ...</progress>
- <progress value="0.5">Half way!</progress>

© JMA 2015. All rights reserved

## Otras etiquetas

- Otros tipos del INPUT
  - Search state (type=search)
  - Telephone state (type=tel)
  - Color state (type=color)
- <keygen> representa un control generador de pares de claves criptográficas, cuando se envía el formulario con el control, la clave privada se almacena en el almacén de claves local y la clave pública se empaqueta y se envía al servidor.
- <output> representa un punto donde dejar el resultado de una acción de cálculo o de usuario.

© JMA 2015. All rights reserved

## Nuevos atributos de INPUT

- placeholder: permite introducir texto de ejemplo o de consejo en los cuadros de texto.
- form: el atributo form es una adición útil que nos permite declarar elementos para un formulario fuera del ámbito de las etiquetas <form>.
- autofocus: enfocará la página web sobre el elemento seleccionado pero considerando la situación actual.
- disabled: deshabilita el control de formulario.
- list: permite mostrar una lista asociada a un cuadro de texto. El valor del atributo list es el identificador de una lista que seguidamente se define con la etiqueta <datalist>, que también es nueva en HTML5.

© JMA 2015. All rights reserved

## Listas de datos

Title: Dr

Mr  
Mrs  
Miss  
Ms

- `<input list="title-list">`  
`<datalist id="title-list">`  
`<option label="..." value="...">>`  
`</datalist>`

© JMA 2015. All rights reserved

## Validación restringida

- El HTML5 brinda sintaxis y elementos de API para posibilitar la validación de formularios del lado del cliente.
- Aunque esta funcionalidad no reemplaza la validación del lado del servidor, que todavía es necesaria por seguridad e integridad de la información, la validación del lado del cliente puede brindar una experiencia de usuario mejor al darle al usuario una respuesta inmediata acerca de la información ingresada.
- Se puede evitar la validación restringida especificando el atributo novalidate en el elemento <form>, o el atributo formnovalidate en el elemento <button> y en el elemento <input> (cuando type es submit o image). Estos atributos indican que el formulario no será validado cuando se envíe.

© JMA 2015. All rights reserved

## Validaciones

- Se han incorporado una serie de atributos a la etiqueta INPUT que permiten validar automáticamente el formulario antes de enviarlo.
  - Required: es obligatorio dar valor.
  - Multiple: indica si al usuario se le permite especificar más de un valor (email, url, file, ...).
  - Pattern: especifica una expresión regular que debe cumplir el valor del control (o los valores si múltiple está activado)
  - Min y Max: indican el rango permitido de valores para el elemento.
  - Step: indica la granularidad que se espera (y requiere) del valor, mediante la limitación de los valores permitidos.

© JMA 2015. All rights reserved

## API de validación restringida

- En objetos `HTMLFormElement` el método `checkValidity()`, que devuelve verdadero si todos los elementos asociados del formulario que necesitan validación satisfacen las restricciones.
- En elementos asociados al formulario:
  - la propiedad `willValidate`, es falso si el elemento no satisface las restricciones.
  - la propiedad `validity`, es un objeto `ValidityState` que representa los estados de validación en que está el elemento (p. ej., condiciones de restricción que han fallado o exitosas).
  - la propiedad `validationMessage`, es un mensaje que contiene todas las fallas o errores en las restricciones que pertenecen a ese elemento.
  - el método `checkValidity()`, devuelve falso si el elemento no logra satisfacer alguna de las restricciones, o verdadero si pasa lo contrario.
  - el método `setCustomValidity()`, establece un mensaje de validación personalizado, permitiendo imponer y validar restricciones más allá de las que están predefinidas.

© JMA 2015. All rights reserved

## Propiedad validity

```
interface ValidityState {  
    readonly attribute boolean valueMissing;  
    readonly attribute boolean typeMismatch;  
    readonly attribute boolean patternMismatch;  
    readonly attribute boolean tooLong;  
    readonly attribute boolean tooShort;  
    readonly attribute boolean rangeUnderflow;  
    readonly attribute boolean rangeOverflow;  
    readonly attribute boolean stepMismatch;  
    readonly attribute boolean badInput;  
    readonly attribute boolean customError;  
    readonly attribute boolean valid;  
};
```

© JMA 2015. All rights reserved

## Compatibilidad

- En navegadores que no entienden las nuevas características de los formularios de HTML5 debemos comprobar si el navegador admite un elemento o atributo HTML5 y en caso negativo, ejecutar el código adicional JavaScript similar a:

```
var nombre =document.getElementById("txtNombre");
var email = document.getElementById("txtEmail");
var tel = document.getElementById("txtTelefono");
window.onload = function() {
    if (!elementSupportsAttribute("input", "placeholder")){
        nombre.setAttribute("style", "color: gray;");
        email.setAttribute("style", "color: gray;");
        tel.setAttribute("style", "color: gray;");
        nombre.value = nombre.getAttribute("placeholder");
        email.value = email.getAttribute("placeholder");
        tel.value = tel.getAttribute("placeholder");
    }
}
```

© JMA 2015. All rights reserved

## Multimedia

- HTML5 incorpora nuevos elementos para reproducir vídeo y audio como parte integrante del lenguaje
  - ya no es necesario depender de software de terceros (Ej: componente Flash Player)
  - En navegadores antiguos también se debe incorporar una versión adicional en formato Flash
- Actualmente conviven distintos formatos y códigos y, lo que es peor los mismos
  - necesitamos disponer de más de una versión del mismo archivo de vídeo o de audio.

© JMA 2015. All rights reserved

## Etiquetas multimedia

- <audio> Permite reproducir audio
  - Atributos: autoplay, controls, loop, src
- <video> Permite reproducir video
  - Atributos: autoplay, controls, loop, height, width, src
- <track> Permite a especificar pistas de texto explícitos externos cronometradas para elementos multimedia (subtítulos).
- <source> Permite especificar múltiples recursos alternativos de los medios de elementos multimedia.

```
<audio width="360" height="240" controls= "controls" >
  <source src="someSong.mp3" type="audio/mp3">
  </source>
  Audio tag is not supported
</audio>
```

© JMA 2015. All rights reserved

## Etiqueta VIDEO

- **autoplay**: indica al navegador que reproduzca el vídeo de manera automática una vez se haya cargado la página
- **controls**: permite que se muestre controles para la reproducción y pausa del vídeo.
- **poster**: indica la imagen que el navegador debe mostrar mientras el vídeo se está descargando, o hasta que el usuario reproduce el vídeo.
- **muted**: permite que el elemento multimedia se reproduzca inicialmente sin sonido, lo que requiere una acción por parte del usuario para recuperar el volumen.
- **width** y **height**: para establecer las dimensiones del vídeo. Si no coinciden con las dimensiones originales, el navegador puede ajustar el tamaño estirando o reduciendo el vídeo.

© JMA 2015. All rights reserved

## Etiqueta VIDEO

- **loop**: indica que el vídeo se reproduce de nuevo una vez que ha finalizado su reproducción.
- **preload**: indica al navegador que comience la descarga del vídeo antes de que el usuario inicie su reproducción:
  - auto (o simplemente preload): sugiere que el navegador debe empezar a descargar el vídeo inmediatamente.
  - none: sugiere que el navegador no debería descargar el vídeo hasta que el usuario lo solicite.
  - metadata: sugiere que el navegador debería empezar a descargar únicamente información acerca del vídeo (duración, dimensiones, etc.). Las distintas pistas de audio y vídeo que visualiza el usuario solo deberían descargarse cuando este "le dé al play".
- **src** : indica la localización del recurso, que el navegador debe reproducir si el navegador soporta el codec o formato específico.

© JMA 2015. All rights reserved

## Códecs de video

- El formato de un archivo de vídeo es el contenedor de la información, por lo que establece cómo se almacena el contenido del vídeo (MPEG4, Ogg, webM, Flash)
- Los códecs se encargan de comprimir y descomprimir la información del vídeo para reducir su tamaño.
- Debemos disponer de más de una versión del mismo archivo de vídeo. Al menos con un códec libre, como Oggheora o VP8 y un códec propietario, como H.264.

© JMA 2015. All rights reserved

## Códecs de video

	H.264	Ogg Theora	VP8 (WebM)
	native	with install	with installs
	native for now; with install from Microsoft	native	native
	native	with install	no
	with install from Microsoft	native	native
	no	native	native

© JMA 2015. All rights reserved

## Etiqueta SOURCE

- Dentro de la etiqueta <video> o <audio> se deben incluir etiquetas adicionales <source> indicando las ubicaciones alternativas del archivo en diferentes formatos. El atributo type indica la naturaleza del mismo.

```
<video width="350" height="280" controls
      poster="media/poster.jpg">
  <source src="media/peli.mp4" type="video/mp4">
  <source src="media/peli.ogv" type="video/ogg">
  <source src="media/peli.webm" type="video/webm">
  Descárguese el vídeo desde <a
      href="media/peli.mp4">aquí</a>.
</video>
```

© JMA 2015. All rights reserved

## Etiqueta TRACK

- **Src:** Apunta al archivo VTT que contiene los subtítulos u otros.
- **Default:** La pista que lo contiene será mostrada por defecto.
- **Label:** El título que identificará la pista de texto, por ejemplo en el menú de selección de subtítulos.
- **Kind:** Identifica el tipo de archivo VTT de que se trata:
  - **captions:** Títulos o leyendas, pueden contener además de textos, efectos de sonido y audio.
  - **chapters:** Capítulos, indica que contiene información para navegación en el video.
  - **descriptions:** Descripciones, indica que contiene texto que será mostrado cuando no esté disponible el componente de video.
  - **metadata:** Metadatos de diferente tipo, contenidos en src.
  - **subtitles:** Subtítulos de texto.
- **Scrlang:** Se trata del código del lenguaje del texto de los subtítulos. Sólo se requiere si el archivo es de tipo subtitles.

© JMA 2015. All rights reserved

## Etiqueta TRACK

- El formato del archivo de pista de texto –subtítulos en formato VTT-, debe cumplir las siguientes condiciones:
  - El texto debe ser codificado como UTF-8
  - Una cabecera con el texto *WEBVTT*
  - Seguida de una línea en blanco
  - Un indicador de la pista –una especie de etiqueta-. Es opcional y será ignorado
  - Seguida de la referencia de tiempo inicial y final en el formato:  
[hh:]mm:ss.sss --> [hh:]mm:ss.sss
  - Inmediatamente después vendría la línea con el texto

© JMA 2015. All rights reserved

## Etiqueta TRACK

WEBVTT

*<espacio en blanco>*

[Indicador o etiqueta]

[hh:]mm:sssss --> [hh:]mm:sssss

Texto del subtítulo

```
<video controls>
<source src="blackhole.mp4" type="video/mp4">
<track label="Subtítulos en español" kind="subtitles" srclang="es"
src="subtitles-es.vtt" default>
<track label="English subtitles" kind="subtitles" srclang="en"
src="subtitles-en.vtt">
<track label="Français sous-titres" kind="subtitles" srclang="fr"
src="subtitles-fr.vtt">
</video>
```

© JMA 2015. All rights reserved

## Navegadores antiguos

- En navegadores antiguos tendremos que añadir el vídeo utilizando **Flash**.
- El código lo incluiremos dentro de la etiqueta **<video>**, como si fuera otra etiqueta **<source>**

```
<object type="application/x-shockwave-flash"
data="http://releases.flowplayer.org/swf/flowplayer-
3.2.1.swf" width="640" height="360">
  <param name="movie"
value="http://releases.flowplayer.org/swf/flowplayer-
3.2.1.swf" />
  <param name="allowFullScreen" value="true">
  <param name="wmode" value="transparent">
  <param name="flashVars"
value="config={'playlist':[{'url':'media%2Fcoches.mp4',
autoPlay':false}]}">
</object>
```

© JMA 2015. All rights reserved

## Etiqueta AUDIO

- Los atributos de la etiqueta `<audio>` que pueden utilizarse son: `autoplay`, `controls`, `loop`, `preload` y `src`. Tienen el mismo significado que para la etiqueta `<video>`.
  - Lo habitual es disponer de un archivo en formato propietario, como MP3; y otro en formato libre, como Ogg. Para compatibilidad con navegadores antiguos, volveremos a utilizar Flash
- ```
<audio controls>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mpeg">
</audio>
```

© JMA 2015. All rights reserved

## API Multimedia

- HTML5 también facilita una API para manejar esos elementos mediante código.
- Los eventos y métodos de los elementos de audio y vídeo son exactamente los mismos, su única diferencia se da en los atributos.

© JMA 2015. All rights reserved

## API Multimedia: Métodos

- Algunos métodos:
  - **play**: para reproducir el archivo.
  - **pause**: para detener la reproducción, pero el marcador de posición se queda en la situación actual. No hay un método "stop" para detener la reproducción y volver al principio del archivo.
  - **load**: para empezar la carga del archivo multimedia.
  - **canPlayType**: para comprobar si el navegador puede reproducir el archivo multimedia (es decir, si es compatible con el formato y códecs utilizados).

© JMA 2015. All rights reserved

## API Multimedia: Eventos

- Los eventos más relevantes para esta API son:
  - **progress**: es disparado periódicamente para informar el progreso en la descarga del medio.
  - **canplaythrough**: es disparado cuando el medio completo puede ser reproducido sin interrupción.
  - **canplay**: es disparado cuando el medio puede ser reproducido. A diferencia del evento previo, éste es disparado cuando solo parte del archivo fue descargado
  - **ended**: es disparado cuando la reproducción llega al final del medio.

© JMA 2015. All rights reserved

## API Multimedia: Eventos

- Los eventos más relevantes para esta API son:
  - **pause**: es disparado cuando la reproducción es pausada.
  - **play**: es disparado cuando el medio comienza a ser reproducido.
  - **error**: es disparado cuando ocurre un error. El evento es despachado desde el elemento `<source>` (si se encuentra presente) correspondiente a la fuente del medio que produjo el error.

© JMA 2015. All rights reserved

## API Multimedia: Propiedades

- Las propiedades más comunes de esta API son:
  - **paused**: retorna true (verdadero) si la reproducción del medio se encuentra pausada o no ha comenzado.
  - **ended**: retorna true (verdadero) si la reproducción llegó al final del medio.
  - **duration**: retorna la duración del medio en segundos.
  - **currentTime**: puede retornar o recibir un valor para informar la posición en la cual el medio se encuentra reproduciendo o establecer una nueva posición donde comenzar a reproducir.

© JMA 2015. All rights reserved

## API Multimedia: Propiedades

- Las propiedades más comunes de esta API son:
  - **error**: el valor del error cuando un error ocurre.
  - **buffered**: ofrece información sobre la cantidad del archivo que fue descargado e introducido en el buffer.
    - Retorna un array contenido datos sobre cada porción del medio que ha sido descargada.
    - Si el usuario salta a otra parte del medio que no ha sido aún descargada, el navegador comenzará a descargar el medio desde ese punto, generando una nueva porción en el buffer.
    - Los elementos del array son accesibles por medio de los atributos end() y start().

© JMA 2015. All rights reserved

## Etiqueta CANVAS

- Ofrece scripts con un lienzo (mapa de bits) que se puede utilizar para la prestación de gráficos, gráficos de juego, arte u otras imágenes visuales generadas sobre la marcha.
- No se deben utilizar el elemento CANVAS en un documento cuando este disponible otro elemento más adecuado.
- Es indicado para cuando se requiera una imagen personalizada creada de forma dinámica.
- Se ha definido un API de dibujo en 2D (HTML Canvas 2D Context):
  - <https://www.w3.org/TR/2dcontext/>

© JMA 2015. All rights reserved

## Gráficos

- La API canvas nos permite dibujar, presentar gráficos en pantalla, animar y procesar imágenes y texto. Es adecuado para el diseño de graficas, juegos, animaciones, composiciones de fotografías, etc.
- El elemento <canvas> genera un espacio rectangular vacío en la página web (lienzo) en el cual serán mostrados los resultados de ejecutar los métodos provistos por la API.
- Los atributos width (ancho) y height (alto) declaran el tamaño del lienzo en pixeles. Todo lo que sea dibujado sobre el elemento tendrá esos valores como referencia.

© JMA 2015. All rights reserved

## Gráficos

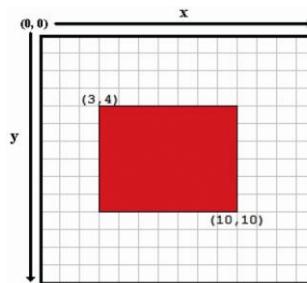
- El método getContext() genera un contexto de dibujo que será asignado al lienzo. A través de la referencia que retorna podremos aplicar el resto de la API.
 

```
function iniciar(){
    var elemento=document.getElementById('lienzo');
    lienzo=elemento.getContext('2d');
}
window.addEventListener("load", iniciar, false);
```
- Una vez obtenido el contexto donde dibujar, podemos hacerlo con una serie de métodos.
- Se puede usar 'webgl' en vez de '2d' pero es no normativo por lo que depende de la implementación de la navegador de WebGL.

© JMA 2015. All rights reserved

## Gráficos

- La superficie de dibujo del contexto 2d se entiende como aquella en la que el origen de coordenadas o punto (0, 0) coincide con el borde superior izquierdo del elemento canvas, como se muestra en esta figura.



© JMA 2015. All rights reserved

## Dibujar rectángulos

- Se utilizan tres métodos del contexto 2d:
  - fillRect: dibuja un rectángulo relleno.
  - strokeRect: dibuja solo el borde del rectángulo.
  - clearRect: limpia el área correspondiente al rectángulo, haciéndola transparente.

```
function iniciar(){
    var elemento=document.getElementById('lienzo');
    contextoDibujo = elemento.getContext('2d');

    contextoDibujo.strokeRect(100,100,120,120);
    contextoDibujo.fillRect(110,110,100,100);
    contextoDibujo.clearRect(120,120,80,80);
}
window.addEventListener("load", iniciar, false);
```

© JMA 2015. All rights reserved

## Dibujar trazados

1. Utilizar el método **beginPath** para iniciar el trazado.
2. Utilizar los métodos de dibujo, como **lineTo** para dibujar líneas o **arc** para dibujar arcos.
3. Utilizar opcionalmente el método **closePath** para cerrar el trazado. Este método cierra el trazado mediante una línea que une el último punto dibujado con el inicial, si es necesario.
4. Finalmente, utilizar el método **stroke** o **fill** para dibujar realmente en la superficie del canvas. El método **stroke** solo incluye el borde de las figuras, mientras que **fill** las dibuja con relleno.

© JMA 2015. All rights reserved

## Métodos de dibujo

- **lineTo(x, y)**: dibuja una línea. Los parámetros x e y representan las coordenadas del punto final de la línea, siendo el punto inicial la posición actual.
- **arc(x, y, radio, ánguloInicial, ánguloFinal, sentido)**: para dibujar círculos o arcos. Este método emplea cinco parámetros:
  - x e y representan las coordenadas del centro del círculo
  - radio es la longitud (en píxeles) del radio del círculo
  - ánguloInicial y ánguloFinal definen el punto inicial y final del círculo. Su valor se establece en radianes
  - sentido: indica si se utiliza el sentido de las agujas del reloj (valor false) o el contrario (valor true)

© JMA 2015. All rights reserved

# Métodos de dibujo

- Más métodos de dibujo:
  - quadraticCurveTo(cx1, cy1, x, y): para dibujar curvas Bézier con un único punto de control.
  - bezierCurveTo(cx1, cy1, cx2, cy2, x, y): para dibujar curvas Bézier con dos puntos de control.
  - rect(x, y, anchura, altura): para dibujar rectángulos que forman parte de un trazado. En este caso actúa como el resto de los métodos anteriores, es decir, que no se dibujará el rectángulo hasta que no utilicemos el método stroke o fill correspondiente
- Colores de trazo y relleno:
  - strokeStyle: declara el color para el contorno o trazo de la figura.
  - fillStyle: declara el color para el interior de la figura o relleno.
  - globalAlpha: especifica la transparencia para todas las figuras dibujadas en el lienzo.

© JMA 2015. All rights reserved

# Estilos de líneas

- Estilos de líneas. Existen cuatro propiedades específicas para este propósito:
  - lineWidth Esta propiedad determina el grosor de la línea. Por defecto el valor es 1.0 unidades.
  - lineCap Esta propiedad determina la forma de la terminación de la línea. Puede recibir uno de estos tres valores: butt, round y square.
  - lineJoin Esta propiedad determina la forma de la conexión entre dos líneas. Los valores posibles son: round, bevel y miter.
  - miterLimit Trabajando en conjunto con lineJoin, esta propiedad determina cuánto la conexión de dos líneas será extendida cuando la propiedad lineJoin es declarada con el valor miter.
- Las propiedades afectarán el trazado completo. Cada vez que tenemos que cambiar las características de las líneas debemos crear un nuevo trazado.

© JMA 2015. All rights reserved

## Gradientes

- Gradientes: para utilizar un gradiente a la hora de trazar o de llenar una figura, primero debes crearlo y después establecerlo como el valor de la propiedad `strokeStyle` o `fillStyle`.
  - `createLinearGradient(x1, y1, x2, y2)` Este método crea un objeto que luego será usado para aplicar un gradiente lineal al lienzo.
  - `createRadialGradient(x1, y1, r1, x2, y2, r2)` Este método crea un objeto que luego será usado para aplicar un gradiente circular o radial al lienzo usando dos círculos. Los valores representan la posición del centro de cada círculo y sus radios.
  - `addColorStop(posición, color)` Este método especifica los colores a ser usados por el gradiente. El atributo `posición` es un valor entre 0.0 y 1.0 que determina dónde la degradación comenzará para ese color en particular.

© JMA 2015. All rights reserved

## Patrones

- `createPattern(imagen, tipo)` El atributo `imagen` es una referencia a la imagen que vamos a usar como patrón, y `tipo` configura el patrón por medio de cuatro valores: `repeat`, `repeat-x`, `repeat-y` y `no-repeat`.

```
<script type="text/javascript">
    window.onload = function() {
        var lienzo = document.getElementById("lienzo1");
        var contexto = lienzo.getContext("2d");
        var imagen = document.createElement("img");
        imagen.src = "images/stripes.jpg";
        imagen.onload = function() {
            var pattern = contexto.createPattern(this, "repeat");
            contexto.fillStyle = pattern;
            contexto.fillRect(0, 0, lienzo.width, lienzo.height);
        }
    }
</script>
```

© JMA 2015. All rights reserved

## Procesando imágenes

- El método `drawImage()` es el único a cargo de dibujar una imagen en el lienzo. Puede recibir un número de valores que producen diferentes resultados:
  - `drawImage(imagen, x, y)` Esta sintaxis es para dibujar una imagen en el lienzo en la posición declarada por x e y. El primer valor es una referencia a la imagen que será dibujada.
  - `drawImage(imagen, x, y, ancho, alto)` Esta sintaxis nos permite escalar la imagen antes de dibujarla en el lienzo, cambiando su tamaño con los valores de los atributos
  - `drawImage(imagen, x1, y1, ancho1, alto1, x2, y2, ancho2, alto2)` Esta es la sintaxis más compleja. Hay dos valores para cada parámetro. El propósito es cortar partes de la imagen y luego dibujarlas en el lienzo con un tamaño y una posición específica.

© JMA 2015. All rights reserved

## Procesando imágenes

```
function iniciar(){
    var elemento=document.getElementById('lienzo');
    lienzo=elemento.getContext('2d');
    var imagen=new Image();
    imagen.src="http://www.minkbooks.com/content/snow.jpg";
    imagen.addEventListener("load", function(){
        lienzo.drawImage(imagen,135,30,50,50,0,0,200,200)
    }, false);
}
window.addEventListener("load", iniciar, false);
```

© JMA 2015. All rights reserved

## Dibujar texto

- Las tres propiedades son ofrecidas para configurar texto:
  - font: Esta propiedad tiene una sintaxis similar a la propiedad font de CSS, y acepta los mismos valores.
  - textAlign: Esta propiedad alinea el texto. Existen varios valores posibles: start (comienzo), end (final), left (izquierda), right (derecha) y center (centro).
  - textBaseline: Esta propiedad es para alineamiento vertical. Establece diferentes posiciones para el texto (incluyendo texto Unicode). Los posibles valores son: top, hanging, middle, alphabetic, ideographic y bottom.

© JMA 2015. All rights reserved

## Dibujar texto

- strokeText(texto, x, y): Similar al método stroke() para el trazado, este método dibujará el texto especificado en la posición x,y como una figura vacía (solo los contornos). Puede también incluir un cuarto valor para declarar el tamaño máximo.
- fillText(texto, x, y): Este método es similar al método anterior excepto que esta vez el texto dibujado será sólido (igual que la función para el trazado).
- measureText(): Este método retorna información sobre el tamaño de un texto específico.

© JMA 2015. All rights reserved

## Sombras

- Podemos generar sombras para cada trazado e incluso textos.
- La API provee cuatro propiedades para hacerlo:
  - **shadowOffsetY**: Esta propiedad recibe un número para determinar qué tan lejos la sombra estará ubicada del objeto (dirección vertical).
  - **shadowBlur**: Esta propiedad produce un efecto de difuminado para la sombra.
  - **shadowColor**: Esta propiedad declara el color de la sombra usando sintaxis CSS.
  - **shadowOffsetX**: Esta propiedad recibe un número para determinar qué tan lejos la sombra estará ubicada del objeto (dirección horizontal).

© JMA 2015. All rights reserved

## Transformaciones

- **translate(x, y)** Este método de transformación es usado para mover el origen del lienzo. Cada lienzo comienza en el punto 0,0 localizado en la esquina superior izquierda, y los valores se incrementan en cualquier dirección dentro del lienzo. Valores negativos caen fuera del lienzo. A veces es bueno poder usar valores negativos para crear figuras complejas.

© JMA 2015. All rights reserved

## Transformaciones

- El método `translate()` nos permite mover el punto 0,0 a una posición específica para usar el origen como referencia para nuestros dibujos o para aplicar otras transformaciones.
- `rotate(ángulo)` Este método de transformación rotará el lienzo alrededor del origen tantos ángulos como sean especificados.

© JMA 2015. All rights reserved

## Transformaciones

- `scale(x, y)` Este método de transformación incrementa o disminuye las unidades de la grilla para reducir o ampliar todo lo que esté dibujado en el lienzo. La escala puede ser cambiada independientemente para el valor horizontal o vertical usando los atributos x e y. Los valores pueden ser negativos, produciendo un efecto de espejo. Por defecto los valores son iguales a 1.0.

© JMA 2015. All rights reserved

## Transformaciones

- `transform(m1, m2, m3, m4, dx, dy)` El lienzo contiene una matriz de valores que especifican sus propiedades. El método `transform()` aplica una nueva matriz sobre la actual para modificar el lienzo.
- `setTransform(m1, m2, m3, m4, dx, dy)` Este método reinicializa la actual matriz de transformación y establece una nueva desde los valores provistos en sus atributos.

© JMA 2015. All rights reserved

## Nuevos APIs DOM

- Geolocation
- LocalStorage
- Native Drag and Drop events
- Multitasking (Worker processes)
- Application Cache (offline access)
- Sockets (real-time server communication: chat, games, etc.)
- Server-Sent Events

© JMA 2015. All rights reserved

# Geolocalización

- Nos permite averiguar la posición geográfica del usuario (lat, lon)
  - Hay métodos más precisos (GPS) y menos (a partir de la dirección IP o usando la red GSM)
  - El método exacto por el que se está calculando la posición es transparente al desarrollador Javascript
  - Lo único que nos da el API son las coordenadas. Necesitaremos algún servicio adicional si queremos dibujar un mapa con la posición, etc. (p.ej. Google Maps)
- Este API no funciona en Explorer 8 y anteriores. Se pueden usar librerías alternativas, como Google Gears (funciona, pero el API es distinto)

© JMA 2015. All rights reserved

# Geolocalización

```
<script>
var x = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation){
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    x.innerHTML= "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
```

© JMA 2015. All rights reserved

## data-\*

- Un atributo de datos personalizado es un atributo que no está en ningún espacio de nombres, cuyo nombre comienza con la cadena "data-", tiene al menos un carácter después del guion, es compatible con XML y **no contiene letras ASCII en mayúsculas**.
 

```
<div class="spaceship" data-ship-id="92432"
    data-weapons="laser 2" data-shields="50%"
    data-x="30" data-y="10" data-z="90">
```
- Los atributos de datos personalizados están destinados a almacenar datos personalizados privados en la página o aplicación, para la cual no hay atributos o elementos más apropiados.
- La propiedad dataset de las etiquetas expone los datos personalizados como propiedades, cuyo nombre es el nombre del atributo sin el prefijo "data-" y, en caso de contener -, se elimina y el siguiente carácter se pasa a mayúscula:
 

```
var x = miDiv.dataset.shipId;
```

© JMA 2015. All rights reserved

## Persistencia de estado

### Servidor

- Opciones:
  - Memoria (Session)
  - Disco (BBDD)
- Problemas:
  - Escalabilidad
  - Temporalidad

### Cliente

- Opciones:
  - Cookies
  - <input type="hidden" ...>
- Problemas:
  - Seguridad
  - Sobrecoste
- Nueva opción HTML5
  - Almacenamiento Local
  - Reduce el sobrecoste

© JMA 2015. All rights reserved

# Bases de Datos

- Web Storage (<http://www.w3.org/TR/webstorage/>): Es el sistema de almacenamiento más simple, ya que los datos se almacenan en parejas de clave/valor. Ampliamente soportado por todos los navegadores.
- Web SQL Database (<http://www.w3.org/TR/webdatabase/>): Sistema de almacenamiento basado en SQL. La especificación indica que no va a ser mantenido en el futuro, pero actualmente su uso está muy extendido y es soportado por Chrome, Safari y Opera.
- IndexedDB (<http://www.w3.org/TR/Indexeddb/>): Sistema de almacenamiento basado en objetos. Actualmente soportado por Chrome, Firefox e Internet Explorer.

© JMA 2015. All rights reserved

## Web Storage

- El almacenamiento local es una muy buena forma de guardar datos en el cliente sin tener que utilizar cookies.
- A diferencia de las cookies, el límite de almacenamiento es mucho mayor (al menos 5 MB) y la información nunca se transfiere al servidor.
- En localStorage, los datos que se guardan son de tipo/valor (key/value), así que si se desea guardar datos más complejos, podemos hacerlo guardando JSON en forma de string (JSON.stringify(obj)) que con JSON.parse(str) recuperaremos la estructura guardada.
- El almacenamiento local está vinculado al origen (por dominio y protocolo). Todas las páginas, de un mismo origen, se pueden almacenar y acceder a los mismos datos.
- HTML define dos objetos (arrays asociativos de JavaScript) para almacenar datos en el cliente:
  - window.localStorage: almacena de datos sin fecha de caducidad
  - window.sessionStorage: almacena los datos para una sola sesión (los datos se pierden cuando la pestaña del navegador se cierra)

© JMA 2015. All rights reserved

## Web Storage

- Para comprobar el soporte del navegador:

```
if (typeof(Storage) !== "undefined") {  
    // Con soporte para almacenamiento.  
} else {  
    // Sin soporte para almacenamiento.  
}
```
- Con localStorage.setItem("clave", "valor") si no existe previamente la clave: se crea la clave y se le asigna el valor, o si existe: se actualiza el valor.
- Con localStorage.getItem("clave") se recupera el valor asociado a la clave, si existe, "undefined" o si no existe.
- Con localStorage.removeItem("clave") se elimina del almacenamiento de forma permanente.

© JMA 2015. All rights reserved

## Drag and Drop

- Arrastrar y soltar es una característica muy común. Que es cuando se "agarra" un objeto y se arrastra a una ubicación diferente.
- En HTML5, arrastrar y soltar es parte de la norma: Cualquier elemento puede ser arrastrable.
- Con el atributo draggable="true" se indica las etiquetas que permitimos arrastrar.
- Se utilizan una serie de eventos que se ejecutan durante las diversas etapas de la operación de arrastre y colocación.

© JMA 2015. All rights reserved

## Drag and Drop

- ondragstart: especifica lo que debe suceder cuando se arrastra el elemento, cachea en el argumento del evento la información a arrastrar:
  - ev.dataTransfer.setData("text", ev.target.id);
- ondragover: cada elemento debe especificar a través del evento si permite recibir el arrastre (soltar) y en casos. Para autorizar el soltar:
  - ev.preventDefault();
- ondrop: define las operaciones a realizar cuando se suelta dentro del elemento, recupera la información arrastrada del argumento del evento, que debe ser del mismo tipo que la caheada:
  - var data = ev.dataTransfer.getData("text");

© JMA 2015. All rights reserved

## Web Worker

- Los navegadores ejecutan las aplicaciones en un único thread, lo que significa que si JavaScript está ejecutando una tarea muy complicada, que se traduce en tiempo de procesado, el rendimiento del navegador se ve afectado.
- Los Web workers se introdujeron con la idea de simplificar la ejecución de threads en el navegador.
- Un worker 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.
- El usuario puede seguir haciendo lo que quiere: hacer clic, la selección de las cosas, etc., mientras que el trabajador web se ejecuta en segundo plano.
- Dado que los trabajadores web están en archivos externos, no tienen acceso a los siguientes objetos JavaScript (hilo principal del JavaScript): DOM, window, document y parent.
- Esta limitación se puede solventar mediante el paso de mensajes con el método postMessage y con el evento onmessage del objeto trabajador web, los datos del trabajador web se traspasan en el event.data.

© JMA 2015. All rights reserved

## Crear un Web Worker

- Se crea un fichero .js con la implementación del Worker, pero a diferencia de la ejecución un script en el documento principal, la visibilidad de un Worker es mucho más reducida.
- La palabra reservada this no hace referencia al objeto Global o Window, sino a la instancia del Worker que se está ejecutando.
- Para enviar mensajes al hilo principal:  
`this.postMessage(datos);`
- Para recibir mensajes del hilo principal:  
`this.addEventListener('message', function(e) {  
 var datos = e.data;  
 // ...  
}, false);`
- El Worker termina cuando acaba el código JavaScript o cuando se invoca a `this.close()`.

© JMA 2015. All rights reserved

## Usar un Web Worker

- Comprobar la disponibilidad de la característica:  
`if(typeof(Worker) !== "undefined") {`
- Crear el Worker y asociar las funciones que procesan los mensajes recibidos y los errores.  
`var worker = new Worker('workerWithError.js');  
worker.addEventListener('message', function(e) {  
 var datos = e.data;  
 // ...  
}, false);`
- Para enviar mensajes al Worker:  
`worker.postMessage(datos);`

© JMA 2015. All rights reserved

# Application Cache

- Cada vez es más importante poder acceder a las aplicaciones web sin conexión.
- HTML5 permite resolver algunas de las molestias asociadas al trabajo sin conexión mediante la interfaz ApplicationCache.
- Las tres ventajas que conlleva el uso de la interfaz de caché para una aplicación.
  - Navegación sin conexión: los usuarios pueden explorar todo el sitio web sin conexión.
  - Velocidad: los recursos almacenados en caché son locales y, por tanto, se cargan más rápido.
  - Reducción de carga del servidor: el navegador solo descarga recursos del servidor que han cambiado.
- La caché de aplicación (o AppCache) permite que el desarrollador especifique los archivos que el navegador debe almacenar en caché y poner a disposición de los usuarios que trabajen sin conexión.
- La aplicación se cargará y funcionará correctamente, incluso si el usuario pulsa el botón de actualización mientras trabaja sin conexión.

© JMA 2015. All rights reserved

## Manifiesto de caché

- El archivo de manifiesto de caché es un sencillo archivo de texto que contiene los recursos que debe almacenar en caché el navegador para el acceso sin conexión.
 

```
<html manifest="http://www.example.com/example.mf">
```
- El atributo manifest debe estar incluido en todas las páginas de la aplicación web que se quiera almacenar en caché, el navegador no almacenará en caché ninguna página que no contenga el atributo manifest (a menos que esa página aparezca explícitamente en el propio archivo de manifiesto).
- Un archivo de manifiesto puede incluir tres secciones:
  - CACHE: Los archivos incluidos en esta sección (o inmediatamente después de CACHE MANIFEST) se almacenarán en caché explícitamente después de descargarse por primera vez (sección predeterminada).
  - NETWORK: Los archivos incluidos en esta sección no se cachean, siempre se solicitan al servidor incluso si el usuario está trabajando sin conexión.
  - Fallback: se especifican páginas alternativas en caso de no poder acceder a un recurso. La primera URI corresponde al recurso y la segunda, a la página alternativa.

© JMA 2015. All rights reserved

# Manifiesto de caché

## CACHE MANIFEST

/favicon.ico

### CACHE:

index.html  
stylesheet.css  
images/logo.png  
scripts/main.js

### NETWORK:

login.php

### FALLBACK:

\*.html /offline.html

© JMA 2015. All rights reserved

# API Application Cache

- El objeto `window.applicationCache` permite acceder (mediante programación) a la caché de aplicación del navegador.
- Su propiedad `status` permite comprobar el estado de la memoria caché.
- Para actualizar la caché mediante programación, primero se debe hacer una llamada a `applicationCache.update()`, cuando el estado de `applicationCache.status` sea `UPDATEREADY`, al llamar a `applicationCache.swapCache()`, se sustituirá la antigua caché por la nueva.
- El navegador activa eventos para una serie de acciones como el progreso de las descargas, la actualización de la caché de las aplicaciones y los estados de error.
- Los navegadores suelen limitar a 5 MB de datos almacenados en caché por cada sitio.

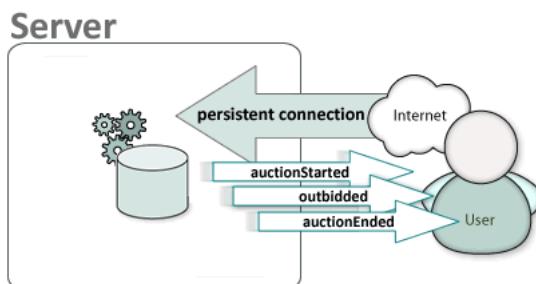
© JMA 2015. All rights reserved

## Server-Sent Events

- La notificación del servidor al cliente es inmediata.
- Evita sondeos reiterados e innecesarios al servidor.
- Envía datos arbitrarios desde el servidor para el cliente, destinados a ser procesado por un script.
- Permite la actualización del contenido sin recargar la página.
- Útil para:
  - Chat en tiempo real o de juego
  - Correo electrónico
  - Actualizaciones en vivo

© JMA 2015. All rights reserved

## Server-Sent Events



© JMA 2015. All rights reserved

## Server-Sent Events

```
var sse = new EventSource('/sse/service');
sse.onopen = function () {
    initData();
};
sse.onmessage = function (event) {
    var data = JSON.parse(event.data);
    recibeData(data);
};
sse.onerror= function (event) {
    diplayError(event);
};
```

© JMA 2015. All rights reserved

## WebSockets

- La especificación WebSocket define un API que establece conexiones "socket" entre un navegador web y un servidor.
- Permite una conexión persistente entre el cliente y el servidor, y ambas partes pueden empezar a enviar datos en cualquier momento.
  - var connection = new WebSocket('ws://echo.websocket.org/');
- Cuando se establezca una conexión con el servidor (cuando el evento open se active), se puede empezar a enviar datos al servidor con el método send('your message') en el objeto de conexión.
- El servidor puede enviar mensajes en cualquier momento, que activa el evento onmessage que recibe un objeto "event" para acceder al mensaje actual mediante la propiedad data.
- WebSocket sigue siendo una tecnología joven y no está implementada completamente en todos los navegadores.
- Presentan problemas de compatibilidad con los servidores proxy que median las conexiones HTTP en la mayoría de las redes corporativas, con la petición HTTP de cambio de HTTP a WebSocket (normalmente utilizado para HTTP/SSL).
- <http://websocket.org/>

© JMA 2015. All rights reserved

Cascading Style Sheets

## CSS

© JMA 2015. All rights reserved

## Qué es el CSS

- CSS significa Cascading Style Sheets (Hojas de estilo en cascada)
- No es un lenguaje de programación, un lenguaje de marcas
- Los estilos definen la estética para mostrar los elementos HTML
- CSS le dice al navegador cómo es el estilo de la página
- Se añadieron Estilos a HTML 4.0 para resolver un problema:
  - Separa el contenido de la estética, esto hace que sea más fácil mantener múltiples estilos para una página
- Las hojas de estilo externas pueden ahorrar mucho trabajo
- Las hojas de estilo externas se almacenan en archivos CSS

© JMA 2015. All rights reserved

## Ventajas

- Centralizar en un único punto el diseño estético de todo un sitio web
- Cambiar el estilo de un sitio web implica hacer los cambios en un único lugar del código
- Reutilizar el estilo en cualquier página web nueva que se cree.
- Aportar coherencia al sitio web, todas las páginas tienen la misma estética
- Redefinir etiquetas HTML
- Desarrollar diseños más avanzados que de otro modo sería imposible o ineficaz con sólo el formato HTML
- Desventajas:
  - 5% de los navegadores aún no reconocen CSS, alrededor del 20% de los navegadores están utilizando modelos de cajas amortizadas

© JMA 2015. All rights reserved

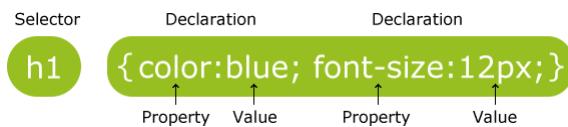
## Definición

- Las recomendaciones actuales establecen que la definición estética de los componentes de una página debe ser establecida mediante estilos.
- Para lo cual se establecen una serie de propiedades con los correspondientes valores opcionales.
- Determinadas propiedades pueden agrupar la definición de varias propiedades simples, en cuyo caso el valor de la propiedad será el conjunto de los valores de las propiedades simples.

© JMA 2015. All rights reserved

## Sintaxis

- Una regla CSS consta de dos partes principales: un selector y una o más declaraciones:



- El selector es normalmente el elemento HTML al que se desea aplicar el estilo.
- Cada declaración consiste en una propiedad y un valor.
- La propiedad es el atributo de estilo que se desea cambiar. Cada propiedad tiene un valor.

© JMA 2015. All rights reserved

## Sintaxis

- Los grupos de declaraciones se encierran entre llaves
- Una declaración siempre termina con un punto y coma
- Para hacer la CSS sea más legible, se recomienda cada declaración en una línea
- Un comentario en CSS comienza con "/ \*" y termina con "\* /"
- Tipos de selectores:
  - Selector de etiqueta HTML
  - Selector de clase
  - Selector de identificación
- Los grupos de selectores están separados por comas

© JMA 2015. All rights reserved

## Selector de Identificación

- El selector de Identificación se utiliza para especificar un estilo para un elemento único.
- NO se debe iniciar un nombre de identificación con un número. No funcionará en todos los navegadores.
- El selector de Identificación utiliza el atributo id del elemento HTML y se define con una "#".
- La siguiente regla de estilo se aplicará al elemento con id = "para1":  
`#para1 {...}`
- Sólo etiquetas con un identificador particular:  
`etiqueta#myId {...}`

© JMA 2015. All rights reserved

## Selector de clase

- Las etiquetas HTML se pueden clasificar introduciendo un nombre arbitrario en el atributo CLASS de la etiqueta.
- El selector de clase se utiliza para especificar un estilo para un grupo de elementos, todos aquellos cuyo atributo CLASS tenga el nombre indicado.
- Diferentes tipos de etiquetas pueden tener el mismo atributo CLASS.
- NO se debe iniciar un nombre de clase con un número. No funcionará en todos los navegadores.
- Esto le permite establecer un estilo particular para muchas etiquetas HTML con la misma clase.
- Se define con un ".":  
`.mi clase {...}`  
`*.mi clase {...}`
- También se puede especificar que sólo se aplique a un determinado tipo de etiquetas de una clase.  
`tag.myClass {...}`

© JMA 2015. All rights reserved

## Combinación de selectores

- Es posible aplicar un estilo para un selector dentro de un selector.
- El anidamiento ayuda a minimizar la codificación lo que conlleva que las páginas se carguen más rápidamente.
- Se especifica el estilo sólo para elementos de tag2 que se encuentren dentro de elementos tag1 :
 

```
tag1 tag2 {...}
```
- Se especifica el estilo solamente para los elementos de tag2 dentro de elementos con class="miClase":
 

```
.myClass tag2 {...}
```
- Se especifica el estilo sólo para los elementos de tag2 dentro de elementos tag1 con class = "miClase":
 

```
tag1.myClass tag2 {...}
```

© JMA 2015. All rights reserved

## Combinación de selectores

- Selectores de hijos:
  - Un selector de hijo coincide cuando un elemento es el hijo de algún elemento.
  - Un selector hijo se compone de dos o más selectores separados por ">".
 

```
body> P {...}
```
- Selectores de hermanos adyacentes
  - Selectores de hermanos adyacentes tienen la siguiente sintaxis: E1 + E2, donde E2 es el sujeto del selector.
  - El selector se aplica a E2 si E1 y E2 comparten el mismo parente en la estructura del documento y E1 precede inmediatamente a E2, haciendo caso omiso de lo que no sean etiquetas (como los nodos de texto y comentarios).
 

```
E1 + E2 {...}
```

© JMA 2015. All rights reserved

## Selectores de atributos

- CSS 2.1 permite especificar reglas donde las etiquetas contengan ciertos atributos coincidentes definidos en el documento de origen.
- La sintaxis de los selectores de atributos:  
 etiqueta [coincidencia] {...}
- Selectores de atributos pueden coincidir en cuatro formas:
  - [att] Cuando la etiqueta establece el atributo "att", cualquiera que sea el valor del atributo.
  - [att = val] Cuando el valor del atributo "att" de la etiqueta es igual a "val".
  - [att ~ = val] El atributo att cuyo valor es una lista separada por espacios en blanco de las palabras y una de las cuales es igual a "val".
  - [att | = val] Cuando el valor del atributo "att" de la etiqueta es igual a "val" o que comienza con "val" inmediatamente seguido por "-" (U + 002D). Esto está pensado principalmente para permitir coincidencias con subcódigo idioma.

© JMA 2015. All rights reserved

## Pseudoclases o Pseudoelementos

- Las pseudoclases o pseudoelementos se utilizan para añadir efectos especiales a algunos selectores.
- Los pseudoelementos crean abstracciones acerca de la estructura del documento más allá de los especificados por el lenguaje del documento.
- Las pseudoclases clasifican las etiquetas sobre características diferentes a su nombre, atributos o contenido.
- La sintaxis de pseudoclases o pseudoelementos:  
 selector:pseudo { ... }  
 selector.class:pseudo { ... }

© JMA 2015. All rights reserved

# Pseudoclases o Pseudoelementos

Selector	Ejemplo	Descripción del ejemplo
:link	a:link	Selecciona todos los enlaces no visitados
:visited	a:visited	Selecciona todos los enlaces no visitados
:active	a:active	Selecciona el enlaces activo
:hover	a:hover	Selecciona el enlace cuando pasa el ratón por encima
:focus	input:focus	Selecciona cuando el control tiene el foco del navegador
:first-letter	p:first-letter	Selecciona la primera letra del texto del párrafo
:first-line	p:first-line	Selecciona la primera línea de texto del párrafo
:first-child	p:first-child	Selecciona la primera etiqueta contenida en el párrafo
:before	p:before	Inserta el valor de la propiedad content delante de cualquier párrafo
:after	p:after	Inserta el valor de la propiedad content después de cualquier párrafo
:lang(lg)	p:lang(it)	Selecciona los párrafos en italiano (atributo lang="it")

© JMA 2015. All rights reserved

# Declaraciones

- Sintaxis:  
propiedad : valor;
- Si el valor es múltiple se separa por comas.
- Valores especiales:
  - Numéricos, pueden indicar las unidades:
    - %: porcentajes
    - in: pulgadas (2,54cm), cm: centímetros, mm: milímetros
    - pt: puntos (1/72in=0,35mm), pc: picas (12pt), px: pixel (0,75pt)
  - URLs y URIs (las comillas son opcionales):
    - url("...")
  - Colors:
    - #RGB o #RRGGBB
    - rgb(0..255, 0..255, 0..255) o rgb(0..100%, 0%..100%, 0%..100%)
- Se puede marcar el valor como importante (aumenta la prioridad):  
valor!important

© JMA 2015. All rights reserved

# Reglas especiales

- La regla @import permite a importar reglas de estilo de otras hojas de estilo.  

```
@import "mystyle.css";
@import url("mystyle.css") media;
```
- Una regla @media especifica los tipos de dispositivos de destino (separadas por comas) de un conjunto de reglas (delimitado por llaves).  

```
@media print {
    ... rules ...
}
```
- Se pueden especificar los márgenes del un cuadro de la página con una regla @page. La regla consta de la palabra clave "@page", seguida de un pseudo-selector de página opcional (selección de izquierda, derecha y primeras páginas), seguido de un bloque que contiene las declaraciones y reglas.  

```
@page {
    ... margin declarations ...
}
```

© JMA 2015. All rights reserved

## Different Media Types

Media Type	Description
all	Used for all media type devices
aural	Used for speech and sound synthesizers
braille	Used for braille tactile feedback devices
embossed	Used for paged braille printers
handheld	Used for small or handheld devices
print	Used for printers
projection	Used for projected presentations, like slides
screen	Used for computer screens
tty	Used for media using a fixed-pitch character grid, like teletypes and terminals
tv	Used for television-type devices

© JMA 2015. All rights reserved

# Implementación

- En la etiqueta, con el atributo STYLE  
... STYLE="Propiedad : Valor; Propiedad : Valor; "
- En la cabecera de la página, con la etiqueta STYLE
 

```
<HEAD>
<STYLE TYPE="text/css">
  Selector { /* Comentarios */
    Propiedad : Valor;
    ...
  }
  ...
</STYLE>
</HEAD>
```
- En ficheros independientes (Hojas de Estilo en Cascada) con extensión .css y se referencian en la cabecera de la página con la etiqueta:  
`<LINK REL="stylesheet" TYPE="text/css" HREF="... URI ...">`

© JMA 2015. All rights reserved

# Prioridad

- Los navegadores deben asignar el valor especificado a cada propiedad en base a los siguientes mecanismos (en orden de prioridad):
  1. Si evaluar la cascada produce un valor, lo utilizan.
  2. De lo contrario, si la propiedad se hereda y el elemento no es la raíz del árbol del documento, utilice el valor calculado para el elemento padre.
  3. De lo contrario, utilice el valor inicial de la propiedad. El valor inicial de cada propiedad se indica en la definición de la propiedad.
- Orden de evaluación
  - Orden en cascada de las definiciones !important
  - Orden en cascada de las demás definiciones
- Orden en cascada:
  - declaraciones inline (etiqueta)
  - declaraciones internas (cabeceras)
  - declaraciones externos (ficheros)
  - Valores por defecto del navegador

© JMA 2015. All rights reserved

# PROPIEDADES

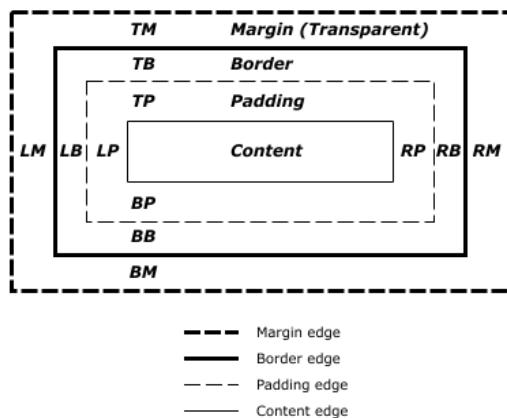
© JMA 2015. All rights reserved

## Backgrounds

- Propiedades relevantes
  - background-color
    - {color-name, color-rgb, color-hex, transparent,inherit}
      - name - a color name, like "red"
      - RGB - an RGB value, like "rgb(255,0,0)"
      - Hex - a hex value, like "#ff000000"
  - background-image
    - {url(URL), none, inherit}
  - background-repeat
    - {repeat-x, repeat-y, no-repeat, inherit}
  - background-attachment
    - {scroll, fixed, inherit}
  - background-position
    - {x y}
      - Puede ser declarada mediante píxeles, porcentaje del ancho total de la página, o un par de {left, top, center, bottom, right}

© JMA 2015. All rights reserved

# Modelo de caja



© JMA 2015. All rights reserved

## Propiedades del modelo

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

© JMA 2015. All rights reserved

## Tamaños

- Propiedades relevantes
  - width y height
    - {unidad de medida, porcentaje, auto}
  - min-width y max-width
    - {unidad de medida, porcentaje, auto}
  - min-height y max-height
    - {unidad de medida, porcentaje, auto}
  - border-width
    - border-top-width, border-right-width, border-bottom-width, border-left-width
    - {unidad de medida, thin, medium, thick }

© JMA 2015. All rights reserved

## Posicionamiento y visualización

- position
  - {static, relative, absolute, fixed}
- top, right, bottom, left
  - {unidad de medida, porcentaje, auto}
- display
  - {inline, block, none, list-item, run-in, inline-block, table, inline-table, table-row-group, table-header-group, table-footer-group, table-row, table-column-group, table-column, table-cell, table-caption}
- visibility
  - {visible, hidden, collapse}
- overflow
  - {visible, hidden, scroll, auto}
- z-index
  - {auto, numero}

© JMA 2015. All rights reserved

# Texto

- Propiedades relevantes
  - color
    - {color-name, color-rgb, color-hex, transparent, inherit}
  - direction
    - {ltr, rtl}
  - text-align
    - {left, right, center, justify}
  - text-decoration
    - {none, underline, overline, line-through, blink}
  - text-transform
    - {none, capitalize, uppercase, lowercase}
  - text-indent
    - {length}
      - La longitud puede expresarse en píxeles o porcentajes
  - vertical-align (use sparingly)
    - {baseline, sub, super, top, text-top, middle, bottom, text-bottom, length}

© JMA 2015. All rights reserved

# Font

- Propiedades relevantes
  - font-family
    - {family}
      - Hay que tener en cuenta que las fuentes deben estar disponibles en el navegador de destino, por lo que la familia debe terminar con fuentes genericas
  - font-style
    - {normal, italic}
  - font-variant
    - {normal, small-caps}
  - font-size
    - Por defecto: 16 pixels = 1 em, Conversión (pixels/16 = em)
      - » Ser capaz de gestionar el tamaño del texto es importante en el diseño web. Sin embargo, no se deben utilizar los ajustes de tamaño de la letra para que los párrafos se parezcan a los títulos o los títulos se parezcan a párrafos. Hay que utilizar siempre las etiquetas HTML adecuadas, como <h1> - <h6> para los títulos y <p> para párrafos.
    - Usar pixels para tamaño absoluto y em para tamaño relativo
  - font-weight
    - {normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900}

© JMA 2015. All rights reserved

# Enlaces

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

© JMA 2015. All rights reserved

# Listas

- Propiedades relevantes
  - list-style-type
    - {disc, circle, square, decimal, decimal-leading-zero, lower-roman, upper-roman, lower-greek, lower-latin, upper-latin, armenian, georgian, lower-alpha, upper-alpha, none}
  - list-style-position
    - {inside, outside}
  - list-style-image
    - {url, none}

© JMA 2015. All rights reserved

# Tablas

- Propiedades relevantes
  - border
    - {size style color}
  - border-collapse
    - {collapse}
  - border-spacing
  - empty-cells
    - {show, hide}
  - caption-side
    - {top, bottom}
  - width, height, text-align, vertical-align
  - padding
    - {top, right, bottom, left}
      - Se establece en sentido horario (padding-top, padding-right, padding-bottom, padding-left).
  - color, background-color
- Nota, no hay ninguna propiedad "cellspacing".

© JMA 2015. All rights reserved

# CSS3

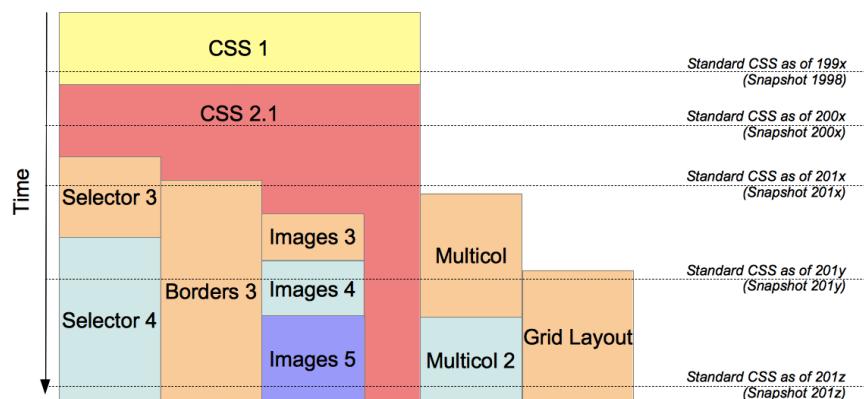
© JMA 2015. All rights reserved

# Especificación

- A diferencia de CSS2, que fue una única especificación que definía varias funcionalidades, CSS3 está dividida en varios documentos separados, llamados "módulos".
- Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad.
- Especificación actualmente en desarrollo, no soportada completamente por la mayoría de los navegadores

© JMA 2015. All rights reserved

# Especificación



© JMA 2015. All rights reserved

## Características

- Mejora los selectores para determinar sobre qué contenido tiene efecto
  - (buen soporte en todos los navegadores modernos)
- Soporte de fuentes exportables (TTF, OTF)
- Mejora del diseño del texto con soporte de columnas
- Esquinas redondeadas, Reflexiones (WebKit)
- Múltiples orígenes, transformaciones (rotación, escala, etc), animaciones (WebKit)

© JMA 2015. All rights reserved

## Prefijos (vendor prefixes)

- Algunas de las propiedades no están definidas por completo o son sólo borradores, por lo que algunos navegadores no las implementan siguiendo la especificación al pie de la letra, sólo funcionan parcialmente o simplemente funcionan de forma diferente dependiendo del navegador que se utilice.
- Para evitar el problema, se idearon una serie de prefijos por navegador (vendor prefixes), que facilitan la segmentación de funcionalidades (nombre entre guiones):
  - Internet Explorer -ms-
  - Firefox -moz-
  - Chrome -chrome- y -webkit-
  - Safari -webkit-
  - Opera -o-

© JMA 2015. All rights reserved

## Prefijos (vendor prefixes)

- Es importante escribir en la última posición el nombre de la propiedad "estándar", es decir, la que aparece en la especificación sin el prefijo: cuando los navegadores que utilizan prefijos especiales implementen dicha propiedad tal como aparece en la especificación, esta es la que finalmente se aplicará.

```
div {
    -webkit-transform: ... /* Chrome, Safari, Opera... (Motor WebKit) */
    -moz-transform: ... /* Mozilla Firefox */
    -ms-transform: ... /* Microsoft Internet Explorer */
    -o-transform: ... /* Antiguo opera */
    transform: ... /* Navegador con especificación oficial */
}
```

© JMA 2015. All rights reserved

## Nuevos selectores de CSS3:

- Selectores de atributos:
  - elemento[atributo^="valor"], selecciona todos los elementos que disponen de ese atributo y cuyo valor comienza exactamente por la cadena de texto indicada.
  - elemento[atributo\$="valor"], selecciona todos los elementos que disponen de ese atributo y cuyo valor termina exactamente por la cadena de texto indicada.
  - elemento[atributo\*="valor"], selecciona todos los elementos que disponen de ese atributo y cuyo valor contiene la cadena de texto indicada.
- Selector general de hermanos:
  - elemento1 ~ elemento2: selecciona cualquier elemento2 que es hermano de elemento1 en el árbol del documento y que aparece detrás en el código HTML aunque no necesariamente de forma inmediata.

© JMA 2015. All rights reserved

## Nuevos selectores de CSS3:

- Nuevos pseudo-elementos: cambian la sintaxis y ahora se utiliza :: en lugar de :
  - ::first-line selecciona la primera línea del texto de un elemento.
  - ::first-letter selecciona la primera letra del texto de un elemento.
  - ::before selecciona la parte anterior al contenido de un elemento para insertar nuevo contenido.
  - ::after selecciona la parte posterior al contenido de un elemento para insertar nuevo contenido.
  - ::selection selecciona el texto que ha seleccionado un usuario con su ratón o teclado.

© JMA 2015. All rights reserved

## Nuevos selectores de CSS3:

- Nuevas pseudoclases:
  - elemento:nth-child(numero) selecciona el elemento indicado pero con la condición de que sea el hijo enésimo de su padre. Por ejemplo, para seleccionar un determinado párrafo o elemento de una lista.
  - elemento:nth-last-child(numero) igual que el anterior pero el número indicado se empieza a contar desde el último hijo.
  - elemento:empty selecciona el elemento indicado pero con la condición de que no tenga ningún hijo (ni siquiera nodos de texto).
  - elemento:first-child y elemento:last-child seleccionan los elementos indicados con la condición de que sean los primeros o últimos hijos de su elemento padre, respectivamente.

© JMA 2015. All rights reserved

# Nuevos selectores de CSS3:

- Nuevas pseudo-clases(2):
  - elemento:nth-of-type(numero) selecciona el elemento indicado con la condición de que sea el enésimo elemento hermano de ese tipo.
  - elemento:nth-last-of-type(numero) igual que el anterior pero el número indicado se empieza a contar desde el último hijo.
  - :not() para seleccionar los elementos que no cumplen con una condición. Por ejemplo, para seleccionar todos los elementos de un documento que no sean enlaces, se podría utilizar ::not(a)
- Algunas pseudo-clases como :nth-child(numero) permiten el uso de expresiones complejas para realizar selecciones avanzadas:
  - li:nth-child(2n+1) { ... } /\* selecciona todos los elementos impares de una lista \*/
  - li:nth-child(2n) { ... } /\* selecciona todos los elementos pares de una lista \*/

© JMA 2015. All rights reserved

# Fuentes

- Las versiones previas de CSS tenían una gran limitación: la imposibilidad de utilizar fuentes personalizadas. En CSS3 se especifica una nueva sintaxis para importar fuentes no disponibles en el navegador y que se descargan de un servidor web.
- Para importar una fuente utilizaremos la regla @font-face: indicamos el nombre de la fuente a importar y la dirección web de donde la debe descargar el navegador.
 

```
@font-face{
    font-family: [nombre de la fuente];
    src: local(""),
          url("nombreadrchivo.woff") format("woff"),
          url("nombreadrchivo.otf") format("opentype"),
          url("nombreadrchivo.svg#nombre de la fuente") format("svg");
}
```
- No todos los navegadores y dispositivos admiten los mismos tipos de fuentes, es necesario proporcionar la misma fuente en distintos formatos.

© JMA 2015. All rights reserved

# Diversos formatos de fuentes

- TrueType Fonts (TTF): TrueType es una fuente estándar desarrollado a finales de 1980, por Apple y Microsoft. TrueType es el formato de fuente más común para los sistemas operativos Mac OS y Microsoft Windows.
- OpenType Fonts (OTF): OpenType es un formato para fuentes de la computadora escalables. Fue construido en TrueType, y es una marca registrada de Microsoft. Las fuentes OpenType se utilizan comúnmente en la actualidad en las principales plataformas informáticas.
- Web Open Font Format (WOFF): WOFF es un formato de fuente para su uso en páginas web. Fue desarrollado en 2009, y ahora es una Recomendación del W3C. WOFF es esencialmente OpenType o TrueType con compresión y metadatos adicionales. El objetivo es soportar la distribución de la fuente del servidor al cliente en una red con restricciones de ancho de banda.
- Web Open Font Format (WOFF 2.0): OpenType/TrueType que proporciona mejor compresión que WOFF 1.0.
- SVG Fonts/Shapes: permiten utilizar glifos SVG para mostrar texto. La especificación SVG 1.1 define un módulo de fuentes que permite la creación de fuentes dentro de un documento SVG.
- Embedded OpenType Fonts (EOT): son una forma compacta de fuentes OpenType diseñado por Microsoft para su uso como fuentes incrustadas en las páginas web.

© JMA 2015. All rights reserved

## ¿Dónde encontrar fuentes?

- **Font Squirrel**([www.fontsquirrel.com](http://www.fontsquirrel.com))
  - normalmente en formato OTF o TTF.
  - convertirlas a los otros formatos:
  - @font-face Generator
  - descargar completos kits de fuentes:
  - @font-face Kits
- **Google Font Directory** <https://fonts.googleapis.com/>
  - Podemos descargar el archivo o archivos correspondientes o utilizar los servidores de Google añadiendo un enlace en la sección de cabecera
  - <link href="https://fonts.googleapis.com/css?family=Indie+Flower" rel="stylesheet">



© JMA 2015. All rights reserved

## Textos

- Ya se pueden crear una estructura de varias columnas (no olvidar prefijos de navegador) :
  - column-count. Indica el número de columnas que queremos tener.
  - column-width. Define el ancho de cada columna.
  - column-gap. Define la separación entre columnas.
  - column-rule. Crea una línea de separación entre las columnas.



© JMA 2015. All rights reserved

## Ajuste de texto

- Word-wrap: permite cortar las palabras que sean demasiado largas para que quepan en el ancho disponible de la caja.

**Sin word-wrap**



**Con word-wrap**



- word-wrap: break-word; /\*Rompe las palabras\*/
- word-wrap: normal; /\*Se porta de la forma habitual\*/

© JMA 2015. All rights reserved

## Desbordamiento de texto

- Con la propiedad CSS3 text-overflow se especifica cómo el contenido desbordado (que no se muestra) debe ser indicado al usuario.

```
p.test1{
    white-space: nowrap;
    width: 200px;
    border: 1px solid #000000;
    overflow: hidden;
    text-overflow: clip;
}
```

Esto es un texto largo qu...

```
p.test2{
    white-space: nowrap;
    width: 200px;
    border: 1px solid #000000;
    overflow: hidden;
    text-overflow: ellipsis;
}
```

Esto es un texto largo q...

© JMA 2015. All rights reserved

## Colores

- Colores RGBA: permite añadir un canal alfa (0 máxima transparencia y 1 máxima opacidad) al sistema de color RGB.
  - backgroundcolor:rgba(100,20,40,0.5);
- Colores HSL y HSLA: tono, saturación y brillo (con o sin transparencia). Se pueden modificar los colores de forma más intuitiva que con RGB.
  - background-color: hsl(360,100%,20%);
  - background-color: hsla(300,130%,65%,10%);
- Propiedad opacity: podemos conseguir que cualquier elemento tenga también un cierto grado de transparencia.
  - #capa1 {opacity: 0.5}

© JMA 2015. All rights reserved

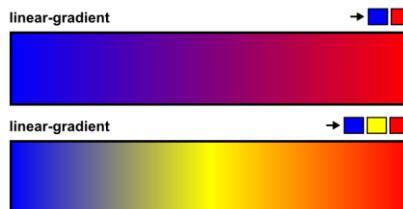
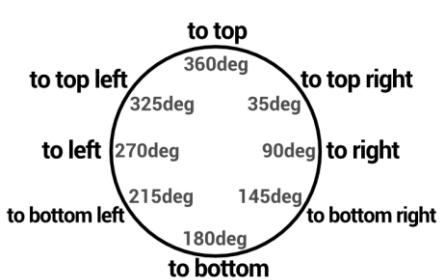
## Gradientes de color

- Podemos utilizar gradientes de color como una forma distinta de establecer una imagen con las propiedades background-image, list-style-image y border-image.
- Para Webkit y Mozilla tendremos que conocer la sintaxis particular de cada caso o acudir a un generador de gradientes. Por ejemplo : <http://westciv.com/tools/gradients>
- Internet Explorer y Opera se supone que admitirán la estandarizada en la especificación CSS3.

© JMA 2015. All rights reserved

## Gradientes lineales

- Permiten crear fondos con los colores gradientes indicados y una cierta dirección definida:
  - background-image: linear-gradient([dirección], [color1], [color2], ...)



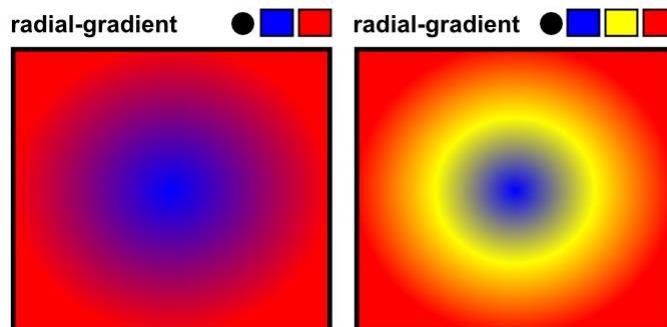
© JMA 2015. All rights reserved

## Gradientes radiales

- Se pueden crear gradientes con formas circulares:
  - `background-image: radial-gradient([forma] [tamaño] at [ubicación], [color1], [color2], ...);`
    - forma: ellipse | circle
    - tamaño: farthest-corner | closest-corner | farthest-side | closest-side | [tamaño]
    - Ubicación: center | top | left | right | bottom | top left | top right | bottom left | bottom right

© JMA 2015. All rights reserved

## Gradientes radiales



© JMA 2015. All rights reserved

## Gradientes recursivos

- Añadiendo el prefijo repeating- a las expresiones anteriores podemos conseguir que el efecto del gradiente, en lugar de adaptarse a la región completa, realice una repetición constante. Comprueba los siguientes ejemplos individuales:
  - background: repeating-linear-gradient(circle, blue, yellow, red);
- Generadores y editores de gradientes on-line
  - <http://www.cssmatic.com/gradient-generator>
  - <http://www.colorzilla.com/gradient-editor/>

© JMA 2015. All rights reserved

## Fondos

- background-size: especifica el tamaño de la imagen de fondo.
 

```
div{
  background: url(img_flwr.gif);
  -moz-background-size: 80px 60px; /* Firefox 3.6 */
  background-size: 80px 60px;
  background-repeat: no-repeat;
}
```
- background-origin: permite definir el origen del fondo de una caja ajustándola al borde (border-box, por defecto), al padding (padding-box) o al contenido (content-box).
- background-clip: permite recortar el área de fondo de una caja ajustándola al borde (border-box , por defecto), al padding (padding-box) o al contenido (content-box)

© JMA 2015. All rights reserved

# Fondos

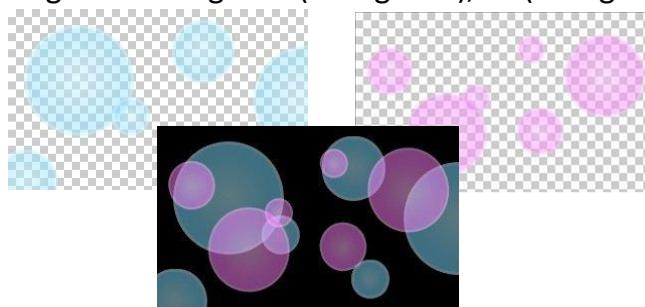


© JMA 2015. All rights reserved

# Fondos

- **background-image:** CSS3 permite asignar a un mismo elemento varios fondos mediante este atributo

`background-image: url("imagen1"), url("imagen2");`



© JMA 2015. All rights reserved

# Filtros (imagenes)

- Los filtros CSS son una característica relativamente nueva de CSS que permite aplicar ciertos efectos de imagen propios de aplicaciones de retoque fotográfico como sepia o variación de contraste al vuelo, sin hacer cambios permanentes sobre una imagen.
- Dichos filtros funcionan a través de la propiedad filter, a la cuál hay que especificarle una función concreta:

**grayscale**: Escala de blanco y negro  
**blur**: Grado de difuminado  
**sepia**: Grado de color sepia  
**saturate**: Grado de saturación  
**opacity**: Grado de transparencia

**brightness**: Brillo  
**contrast**: Contraste  
**hue-rotate**: Rotación de color (matiz)  
**invert**: Invertir  
**drop-shadow**: Sombra idéntica

- Mediante la propiedad mix-blend-mode también se pueden utilizar los denominados modos de fusión, muy comunes en programas profesionales de retoque fotográfico como Photoshop, para aplicar sobre elementos de contenido, como imágenes o similares.

© JMA 2015. All rights reserved

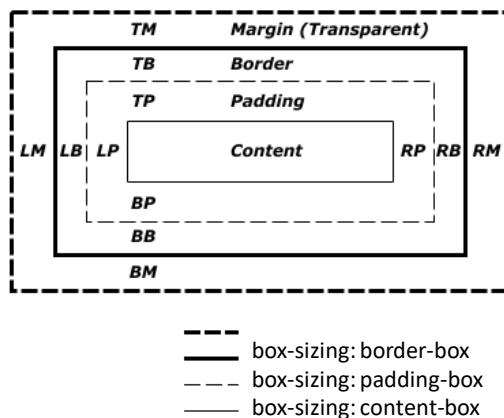
# Modelo de caja

- Por defecto, las propiedades width y height son las medidas solo para el contenido, pero no incluyen el padding, el border o el margin.
- Para calcular el tamaño final es necesario sumar al espacio del contenido el espacio destinado al padding y al border.
- La propiedad box-sizing simplifica los cálculos indicando si las medidas expresadas en las propiedades width y height incluyen:
  - content-box**: solo en el contenido (por defecto), no incluyen el padding, el border o el margin.
  - padding-box**: incluyen el padding pero no incluyen el border y el margin.
  - border-box**: incluyen el padding y el border, pero no el margin.
- Estos dos div ocupan lo mismo, independientemente del padding, al indicarse box-sizing: border-box:
 

```
.div1
{width:300px; height:100px; border:1px; box-sizing: border-box;}
.div2
{width:300px; height:100px; border:1px; box-sizing: border-box; padding:50px;}
```

© JMA 2015. All rights reserved

## Modelo de caja



© JMA 2015. All rights reserved

## Modelo de caja flexible (Flexbox)

- Tradicionalmente, en CSS se ha utilizado el posicionamiento (static, relative, absolute...), los elementos en línea o en bloque (y derivados) o los float, lo que a grandes rasgos no dejaba de ser un sistema de creación de diseños bastante tradicional que no encaja con los retos que tenemos hoy en día (sistemas de escritorio, dispositivos móviles, múltiples resoluciones, etc...).
- Flexbox es un sistema de elementos flexibles que llega con la idea de olvidar estos mecanismos y acostumbrarnos a una mecánica más potente, limpia y personalizable, en la que los elementos HTML se adaptan y colocan automáticamente y es más fácil personalizar los diseños.
- Flexbox permite la definición del estilo de los elementos que ocupan el espacio horizontal o vertical de una página.

© JMA 2015. All rights reserved

# Modelo de caja flexible (Flexbox)

- En CSS3, además de las formas tradicionales de diseñar la página (propiedades position o float) hay una nueva forma de diseñar la página, mediante "cajas flexibles" o "FlexBox".
- Formas de crear una caja en Flexbox:
  - box: habilita este elemento y los secundarios directos dentro del modelo Flexbox. El modelo Flexbox solo funciona con elementos secundarios directos.
  - box-orient (Vertical | Horizontal): indica cómo se alinean los elementos secundarios dentro de box.
  - box-pack (start | end | center | justify ): alineación de box a lo largo de su propio eje (horizontal o vertical).
  - box-align (start | end | center | baseline | stretch): alineación de los elementos secundarios dentro de box.
  - box-ordinal-group (entero): indica el orden en el que se colocan los elementos dentro de la caja.
  - box-flex (0 ... número entero): es el índice de flexibilidad del elemento secundario. Si el valor de un elemento secundario es 1 y el de otro elemento secundario es 2, este último consumirá el doble de espacio adicional de la caja principal. El valor predeterminado es 0, lo que indica que no es flexible.

© JMA 2015. All rights reserved

# Esquinas redondeadas

- CSS3 añade interesantes características en materia de bordes, como la posibilidad de crear bordes con esquinas redondeadas, característica que en versiones anteriores de CSS era muy complicado de lograr (necesitando recurrir al apoyo de imágenes gráficas), siendo en CSS3 realmente sencillo.
- Basta con utilizar la propiedad border-radius, con la cual podrás especificar un radio para el borde de las esquinas.
- Con border-top-left-radius, border-top-right-radius, border-bottom-left-radius y border-bottom-right-radius se personaliza cada una de las esquinas.
- Es posible diferenciar el radio horizontal del radio vertical (separándolos con la / ) de una esquina determinada, creando una esquina redondeada irregular:
  - radio horizontal /radio vertical
  - border-radius: 5px 50px / 50px 15px;

© JMA 2015. All rights reserved

## Bordes con imágenes

- Otra de las novedades que ofrece CSS3 es la de utilizar una imagen como borde.
- Sin embargo, su implementación no es tan sencilla como utilizar una imagen de fondo, ya que el borde debería definir claramente las zonas de la imagen para tenerlas en cuenta a la hora de ampliar, reducir o estirar el elemento y sus bordes.
  - border-image-width indica el tamaño de ancho que tendrá el borde de la imagen. No olvides que hay que indicar también un border-width y un border-style para que el borde CSS esté definido y se pueda visualizar.
  - border-image-source establece, mediante la expresión url(), la imagen que vamos a utilizar para crear nuestro borde con imágenes.
  - border-image-slice define la posición de las líneas divisorias de la imagen, por lo que podemos utilizar esta propiedad para mover dichas líneas a nuestro gusto. Por defecto, el valor es de 100% (límite de la imagen). Se recomienda utilizar porcentajes o píxeles sin especificar el texto "px". Se puede utilizar, como ya estamos acostumbrados, el formato de 1, 2, 3 o 4 parámetros.
  - border-image-outset establece el desplazamiento hacia fuera de la imagen. Muy útil para compensar la imagen si se extiende hasta el contenido. Por defecto no tiene desplazamiento.
  - border-image-repeat establece como deben comportarse los fragmentos del borde y el tipo de repetición que deben efectuar. Es importante recalcar los dos últimos valores (round y space) los cuales actúan igual que repeat, pero con un comportamiento ligeramente diferente que nos puede interesar en el caso de que la zona repetida quede descompensada.

© JMA 2015. All rights reserved

## Efectos de sombra

- Las propiedades text-shadow y box-shadow permiten aplicar sombras en el texto y en la caja delimitadora de un elemento de bloque respectivamente.
- Los valores que tenemos que establecer son:
  - Desplazamiento horizontal de la sombra respecto del texto. Si es un valor positivo, el sentido de la sombra es hacia la derecha; si es negativo, el sentido es hacia la izquierda.
  - Desplazamiento vertical de la sombra respecto del texto. Si es un valor positivo, el sentido de la sombra es hacia abajo; si es negativo, es hacia arriba.
  - Valor de difuminado (desenfoque).
  - El color de la sombra.

```
h1 {
  color: white;
  text-shadow: 2px 2px 4px #000000;
}
```
- Para agregar más de una sombra al texto, puede agregar una lista separada por comas de sombras.
 

```
h1 {
  color: white;
  text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;
}
```

© JMA 2015. All rights reserved

# Interacciones

- Contornos (outline)
  - La familia de propiedades outline-\* nos permiten modificar el comportamiento del contorno de los elementos: una línea divisoria que rodea el contenido externo del propio elemento. A diferencia de los bordes, esta línea divisoria, por defecto no ocupa espacio y no tiene porque tener una forma rectangular.
  - Es fácil de observar esta línea divisoria en los navegadores, pulsando TAB y moviéndonos por los diferentes enlaces de la página. Generalmente, aparece como una línea punteada y es muy similar al funcionamiento de los bordes.
- Cambio de tamaño (resize )
  - La propiedad resize especifica si el usuario puede cambiar el tamaño de un elemento. Los valores que toma pueden ser: horizontal, vertical, both (ambos) y none (ninguno).
  - Si queremos que redimensione sólo hasta cierto tamaño y no más, podríamos usar las propiedades max-width y max-height, indicando en ellas, los valores correspondientes, lo mismo que los mínimos con las propiedades min-width y min-height.

© JMA 2015. All rights reserved

## Cursos CSS3

CSS2		CSS3	
↳ default	↔ e-resize	none	↔ ew-resize
⊕ crosshair	↖ ne-resize	↗ context-menu	↓ ns-resize
↳ help	↖ nw-resize	⊕ cell	↗ nesw-resize
❖ move	↑ n-resize	→ vertical-text	↖ nwse-resize
↳ pointer	↖ se-resize	↖ alias	↔ col-resize
⌚ progress	↖ sw-resize	↖ copy	↑ row-resize
_  text	↑ s-resize	↖ url(images/nyan.png)	❖ all-scroll
⌚ wait	↔ w-resize	🚫 not-allowed	🚫 no-drop
		✋ grab	✋ grabbing
		⊗ zoom-in	⊗ zoom-out

© JMA 2015. All rights reserved

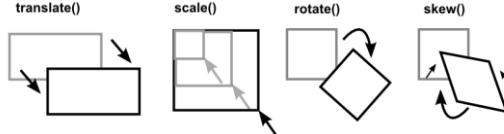
# Transformaciones

- Las transformaciones CSS3 permiten trasladar, rotar, escalar y sesgar elementos.
- Una transformación es un efecto que permite un cambio de elemento de forma, tamaño y posición.
- CSS3 admite transformaciones 2D y 3D.
- La propiedad `transform` permite establecer la función o funciones que se deben aplicar.
- Las propiedades `transform-origin`, `transform-style`, `perspective`, `perspective-origin` y `backface-visibility` aportan definiciones adicionales y comunes a las funciones de transformación.

© JMA 2015. All rights reserved

## Transformaciones 2D

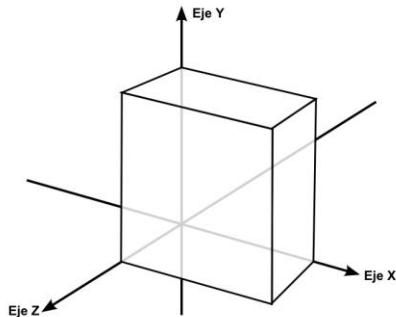
- `translate()`: mueve un elemento desde su posición actual (de acuerdo con los parámetros dados para el eje X y el eje Y). Métodos para un solo eje: `translateX()` y `translateY()`.
- `rotate()`: hace girar un elemento sobre un punto de acuerdo con un determinado grado.
- `scale()`: aumenta o disminuye el tamaño de un elemento (de acuerdo con los parámetros dados para la anchura y altura). Métodos para un solo eje: `scaleX()` y `scaleY()`.
- `skew()`: sesga (deforma) un elemento a lo largo del X y del eje Y por los ángulos dados. Métodos para un solo eje: `skewX()` y `skewY()`.
- `matrix()`: combina todos métodos 2D de transformación en uno solo.
  - Toma seis parámetros, que contiene funciones matemáticas, que le permite rotar, escalar, mover y sesgar elementos: `matrix (scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())`



© JMA 2015. All rights reserved

## Transformaciones 3D

- `rotateX()`: hace girar un elemento alrededor de su eje X en un grado dado (reflejo horizontal).
- `rotateY()`: hace girar un elemento alrededor de su eje Y en un grado dado (reflejo vertical).
- `rotateZ()`: hace girar un elemento alrededor de su eje Z en un grado dado.



© JMA 2015. All rights reserved

## Transiciones

- Las transiciones se basan en un principio muy básico, conseguir un efecto suavizado entre un estado inicial y un estado final.
- Las transiciones CSS3 permiten el cambio de los valores de una o varias propiedades no se realice directamente, sino que pase por los valores intermedios entre el valor inicial al valor final durante un periodo de tiempo determinado.
- `transition-timing-function` especifica la curva de velocidad del efecto de transición:
  - `ease`: Especifica un efecto de transición con un comienzo lento, luego rápido, y luego terminar lentamente (por defecto)
  - `linear`: Especifica un efecto de transición con la misma velocidad de principio a fin
  - `ease-in`: Especifica un efecto de transición con un comienzo lento
  - `ease-out`: Especifica un efecto de transición con un final lento
  - `ease-in-out`: Especifica un efecto de transición con un lento comienzo y el final
  - `cubic-bezier(n,n,n,n)`: Le permite definir sus propios valores de una función cúbica-Bezier
- `transition-delay` especifica un retraso (en segundos) para el efecto de transición.

© JMA 2015. All rights reserved

# Animaciones

- La animaciones CSS3 amplían el concepto de transiciones convirtiéndolo en algo mucho más flexible y potente.
- La idea de las animaciones CSS es permitir la adición de más estados, pudiendo realizar cambios desde un estado inicial, a un estado posterior, y luego a otro estado posterior, y así sucesivamente.
- La creación de animaciones esta compuesta:
  - Regla @keyframes, que incluye los fotogramas de la animación.
  - Propiedades CSS de las animaciones, que definen el comportamiento de la misma.

© JMA 2015. All rights reserved

## Regla @keyframes

```
@keyframes nombre{
  selectorTemporal { ... asignación de valores a propiedades ... }
  ... Definición de otros selectores temporales ...
}
```

- Todas las reglas tienen un nombre para la animación para hacer referencia a la animación en la propiedad animation o animation-name.
- El selector temporal, tantos como sean necesarios, se establece como el porcentaje de avance de la animación.

```
@keyframes example {
  0% {background-color: red;}
  50% {background-color: blue;}
  100% {background-color: green;}
}
```

- Las palabras clave "from" y "to" (que representa el 0% (inicio) y 100% (completa)).

```
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

© JMA 2015. All rights reserved

# Propiedades de las animaciones

- **animation-name:** permite especificar el nombre del fotograma a utilizar, mientras que las propiedades `animation-duration`, `animation-timing-function` y `animation-delay` funcionan exactamente igual que en el tema anterior de transiciones.
- **animation-iteration-count:** permite indicar el número de veces que se repite la animación, pudiendo establecer un número concreto de repeticiones o indicando `infinite` para que se repita continuamente.
- **animation-direction:** indica el orden en el que se reproducirán los fotogramas, pudiendo escoger un valor de los siguientes:
  - `normal`: Los fotogramas se reproducen desde el principio al final (por defecto).
  - `reverse`: Los fotogramas se reproducen desde el final al principio.
  - `alternate`: En iteraciones par, de forma normal. Las impares, a la inversa.
  - `alternate-reverse`: En iteraciones impares, de forma normal. Las pares, inversa.
- **animation-fill-mode:** indica que debe mostrar la animación cuando ha finalizado y ya no se está reproduciendo; si mostrar el estado inicial (`backwards`, por defecto), el estado final (`forwards`) o una combinación de ambas (`both`).
- **animation-play-state:** permite establecer la animación a estado de reproducción (`running`) o pausarla (`paused`).

© JMA 2015. All rights reserved

RWD – Responsive Web Design

## DISEÑO WEB ADAPTATIVO

© JMA 2015. All rights reserved

## Diseño Adaptativo

- Es un enfoque de diseño destinado a la elaboración de sitios/aplicaciones para proporcionar un entorno óptimo de:
  - Lectura Fácil
  - Navegación correcta con un número mínimo de cambio de tamaño
  - Planificaciones y desplazamientos
- Con la irrupción multitud de nuevos dispositivos y el que el acceso a internet se realiza ya mayoritariamente desde dispositivos diferentes a los tradicionales ordenadores ha obligado a seguir dicho enfoque en las aplicaciones WEB.
- Contempla la definición de múltiples elementos antes de la realización de la programación real.
  - Elementos de página en las unidades de medidas correctas
  - Imágenes flexibles
  - Utilización de CSS dependiendo de la aplicación

© JMA 2015. All rights reserved

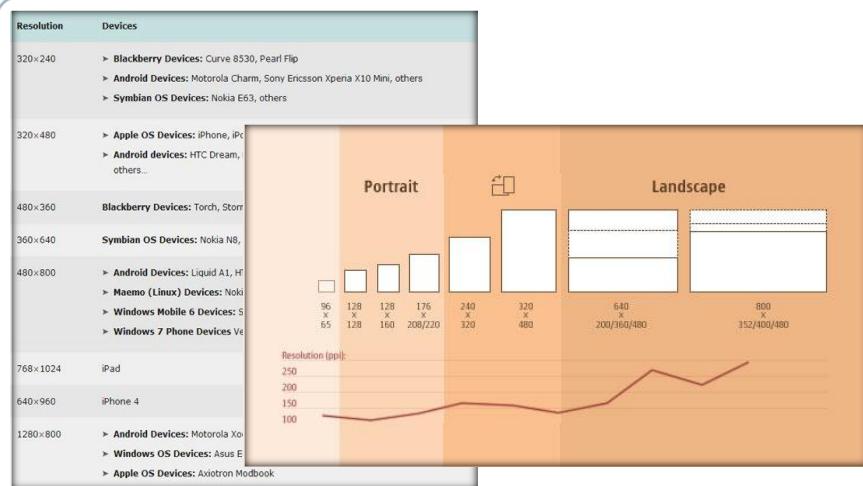
## Resolución

- Los dispositivos móviles tienen una característica distintiva, y es su resolución de pantalla.
- Es necesario conocer cuales son las resoluciones más comunes en este tipo de dispositivos móviles, de los gadgets más utilizados, etc.
- Las resoluciones van cambiando de forma muy rápida y en dispositivos nuevos



© JMA 2015. All rights reserved

# Resolución



© JMA 2015. All rights reserved

# Resolución

- También deberemos tener en cuenta la resoluciones de otros dispositivos como:
  - Tablets
  - TV SmartTV
  - Pizarras electrónicas, etc



**Pizarras** 10.1" y 11.6" (2560x1440, 1920x1080, 1366x768), 17" (1920x1080)

**PC** 12" (1280x800), 14" (1920x1080, 1366x768), 15.6" (1920x1080)

**Family hub** 23" (1920x1080), 27" (2560x1440)

© JMA 2015. All rights reserved

# Orientación de Página

- La orientación del papel es la forma en la que una página rectangular está orientada y es visualizada.
- Los dos tipos más comunes son:
  - Landscape (Horizontal)
  - Portrait (Vertical)



© JMA 2015. All rights reserved

## Recomendaciones de Diseño

1. Utilizar porcentajes y "ems" como unidad de medida en lugar de utilizar los valores determinados como definición de pixel.

Las ems son unidades relativas, así que más exactamente 1 em equivale al cien por cien del tamaño inicial de fuente.

© JMA 2015. All rights reserved

## Recomendaciones de Diseño

2. Determinar el tamaño y definición de las imágenes a utilizar
  - Recortar, Ajustar, etc



© JMA 2015. All rights reserved

## Recomendaciones de Diseño

3. El contenido y funcionalidad BÁSICA debe de ser accesible por todos los navegadores



© JMA 2015. All rights reserved

## Recomendaciones de Diseño

### 4. Definición correcta de contenidos en función del dispositivo



© JMA 2015. All rights reserved

## Ventajas

- Soporte de dispositivos móviles.
- Con una sola versión en HTML y CSS se cubren todas las resoluciones de pantalla.
- Mejora la experiencia de usuario.
- Se reducen los costos de creación y mantenimiento cuando el diseño de las pantallas es similar entre dispositivos de distintos tamaños.
- Evita tener que desarrollar aplicaciones específicas para cada sistema operativo móvil.
- Facilita la referenciación y posicionamiento en buscadores, versión única contenido/página.

© JMA 2015. All rights reserved

# viewport

- Hace referencia a la región visible del navegador, o sea, la parte de la página que está visualizándose actualmente en el navegador.
- Podemos redimensionar la ventana del navegador para reducir el tamaño del viewport y simular que se trata de una pantalla y dispositivo más pequeño.
 

```
<meta name="viewport" content="initial-scale=1, width=device-width">
```
- Los parámetros de comportamiento para el viewport son:
  - width: Indica un ancho para el viewport.
  - height: Indica un alto para el viewport.
  - initial-scale: Escala inicial con la que se visualiza la página web.
  - minimum-scale: Escala mínima a la que se puede reducir al hacer zoom.
  - maximum-scale: Escala máxima a la que se puede aumentar al hacer zoom.
  - user-scalable: Posibilidad de hacer zoom en la página web.

© JMA 2015. All rights reserved

# Consultas de medios

- Las consultas de medios en CSS3 extiende la idea de CSS2: en lugar de buscar un tipo de dispositivo, se mira la capacidad del dispositivo.
- Las consultas de medios se pueden utilizar para comprobar muchas cosas, tales como:
  - anchura y la altura de la ventana gráfica
  - anchura y la altura del dispositivo
  - orientación (es la tableta / teléfono en modo horizontal o vertical)
  - resolución

```
@media not|only mediatype and (expressions) {
  CSS-Code;
}
```

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">
```
- Las consultas de medios son una técnica popular para la entrega de una hoja de estilo a medida para tabletas, iPhone y Androids.
 

```
@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
```

© JMA 2015. All rights reserved

## LESS

© JMA 2015. All rights reserved

# Introducción

- A medida que la web avanza y la estética gana importancia hasta ser imprescindible, mas las recomendación de separar la estética de la estructura, ha provocado que el CSS se vuelva inmanejable pasando de unos cientos de líneas a miles de ellas.
- Esto ha propiciado la aparición de alternativas que simplifiquen la definición del estilo y facilite la mantenibilidad evitando errores por lo que redundan en un aumento de la calidad.
- Los preprocesadores de CSS son herramientas que permiten escribir pseudo-código alternativo al CSS que luego será convertido a CSS estándar y utilizarse donde pueda usarse el CSS.
- Los mas conocidos son Sass, Less y Stylus.

© JMA 2015. All rights reserved

## Less

- Less es un metalenguaje dinámico de hojas de estilo, basado en la sintaxis de CSS, a la que ha incorporado variables, anidamiento, operadores, mixins y funciones, lo que es válido en CSS es válido en Less con la misma semántica.
- Diseñado por Alexis Sellier, está influenciado por Sass y ha influido, a su vez, en la nueva sintaxis "SCSS" de Sass.
- LESS es de código abierto. Su primera versión fue escrita en Ruby, sin embargo, en las versiones posteriores, se abandonó el uso de Ruby y se lo sustituyó por JavaScript.
- La principal diferencia entre Less y otros precompiladores CSS es que Less permite la compilación por el navegador en tiempo real vía less.js. LESS se puede ejecutar en el lado del cliente y del lado del servidor, o se puede compilar en CSS estático.
- La extensión de los ficheros de código precompilable es .less

© JMA 2015. All rights reserved

## Uso

- Instalación:
  - npm install -g less
- Compilación en línea de comandos:
  - lessc styles.less styles.css
- Hay disponibles extensiones para los principales editores y entornos de desarrollo que generan automáticamente la versión .css al guardar los ficheros .less
- Para la compilación dinámica en el navegador:
 

```
<link rel="stylesheet/less" type="text/css" href="styles.less" />
<script
src="//cdnjs.cloudflare.com/ajax/libs/less.js/2.7.2/less.min.js"></script>
```

© JMA 2015. All rights reserved

## Variables

- Permiten asociar nombres a valores y utilizar los nombres en sustitución de los valores reales en las reglas.
- Durante la traducción, los nombres se sustituyen por valores reales en el documento CSS de salida.

```
@colorLetra: #3d3d3d;
@tamanioLetra: 12pt;
```

```
#titulo {
    color: @colorLetra;
    font-size:@tamanioLetra;
}
h2 {
    color: @colorLetra;
}
```

© JMA 2015. All rights reserved

## Anidamiento

- LESS permite anidar los selectores dentro de otros selectores generando el anidamiento lógico del CSS. Esto hace la herencia clara y las hojas de estilo más cortas.

```
.menu {
    font-size:@tamanioLetra;
    a { ... }
    a:hover { ... }
}
```

- Genera:
- ```
.menu { font-size:@tamanioLetra; }
.menu a { ... }
.menu a:hover { ... }
```

© JMA 2015. All rights reserved

# Mixin

- Mixins es una forma de definir e incluir ("mezclar") un conjunto de propiedades en otras reglas, evitando la repetición de las mismas y acortando el código.
- Los mixins se pueden definir utilizando la sintaxis de las clases o de los identificadores.

```
.bordered {
  border-top: dotted 1px black;
  border-bottom: solid 2px black;
}
#menu a {
  color: @colorLetra;
  .bordered;
}
.post a {
  color: red;
  .bordered;
}
```

© JMA 2015. All rights reserved

# Mixin !important

- Si se usa la palabra clave !important después de la llamada al mixin se marcan todas las propiedades heredadas como !important:

```
.foo { background: #f5f5f5; }
.unimportant{
  .foo;
}
.important{
  .foo !important;
}

• Genera:
.unimportant{
  background: #f5f5f5;
}
.important{
  background: #f5f5f5 !important;
}
```

© JMA 2015. All rights reserved

## Mixin parametrizados

- Los Mixin también puede tomar argumentos, que son variables pasadas al mixin cuando se utilizan en el bloque de la regla. Los parámetros pueden contar con un valor por defecto, en cuyo caso los argumentos son opcionales:

```
.border-radius(@radius: 5px) {
  -webkit-border-radius: @radius;
  -moz-border-radius: @radius;
  border-radius: @radius;
}
#header {
  .border-radius;
}
.button {
  .border-radius(6px);
}
```

© JMA 2015. All rights reserved

## Selectores principales

- El operador & representa los selectores principales de una regla anidada y se usa con más frecuencia cuando se aplica una clase o pseudoclase modificadora a un selector existente:

```
.especial {
  color: blue;
  &:hover { color: green; }
}
a { .especial; }
button { .especial; }
.button {
  &-ok { ... }
  &-cancel { ... }
}
```

- Genera:

```
a { color: blue; }
a:hover { color: green; }
button { color: blue; }
button:hover { color: green; }
.button-ok { ... }
.button-cancel { ... }
```

© JMA 2015. All rights reserved

# Escape e Interpolación

- Cualquier cosa dentro ~"anything" o ~'anything' se usa tal cual, sin cambios en la compilación, excepto la interpolación.
- Si la variable contiene el nombre de una clase, etiqueta, propiedad o parte de una cadena debe ente {} para que sustituya el valor.
 

```
@my-selector: banner;
@images: "../img";
@property: color;
.{@{my-selector}}{
  background: url("@{images}/white-sand.png");
  @{property}: #0ee;
}
```
- Genera:
 

```
.banner {
  background: url("../img/white-sand.png");
  color: #0ee;
}
```

© JMA 2015. All rights reserved

# Operaciones

- Less permite expresiones aritméticas con +, -, \* y / que pueden operar con cualquier número, color o variable.
- Si es posible, las operaciones toman en cuenta las unidades y convierten los números antes de sumarlos, restarlos o compararlos. El resultado se deja en el tipo de la primera unidad (mas a la izquierda) que se indique explícitamente. Si la conversión es imposible o no significativa, las unidades son ignoradas.
- Los colores se dividen en sus dimensiones roja, verde, azul y alfa. La operación se aplica a cada dimensión de color por separado.

```
@conversion-1: 5cm + 10mm; // result is 6cm
@conversion-2: 10mm + 5cm; // result is 60mm
@incompatible-units: 2 + 5px - 3cm; // impossible: result is 4px
@color: #224488 / 2; //results in #112244
@baseLetra: 10pt;
h1 { font-size: @baseLetra * 4; } // result is 40pt
h2 { font-size: @baseLetra * 3; } // result is 30pt
```

© JMA 2015. All rights reserved

## Funciones

- Less proporciona una gran variedad de funciones que transforman colores, manipulan cadenas y hacen cálculos matemáticos.
- Usarlos es bastante sencillo. El siguiente ejemplo usa funciones para convertir 0.5 a 50%, aumentar la saturación de un color base en un 5% y luego establecer el color de fondo en uno que se aclara un 25% y se gira en 8 grados:

```
@base: #f04615;
@width: 0.5;
```

```
.class {
  width: percentage(@width); // returns '50%'
  color: saturate(@base, 5%);
  background-color: spin(lighten(@base, 25%), 8);
}
```

© JMA 2015. All rights reserved

## Alcance

- El alcance de los nombres es muy similar al de los lenguajes de programación. Las variables y mixins se buscan primero localmente, y si no se encuentran, el compilador buscará en el ámbito superior, y así sucesivamente.

```
@colorLetra: red;
```

```
#page {
  @colorLetra: white;
  #header {
    color: @colorLetra; // white
  }
}
```

© JMA 2015. All rights reserved

## Espacios de nombres

- A veces, es necesario agrupar los mixins con fines de organización o de encapsulación.

```
#bundle {
  .button { ... }
  .tab { ... }
  .citation { ... }
}
#other {
  .button { ... }
}

#header a {
  color: orange;
  #bundle > .button;
}
#footer { #other > .button; }
```

© JMA 2015. All rights reserved

## Comentarios e Importaciones

- Se pueden usar comentarios en bloque y añade los comentarios en línea:  
`// Hasta el fin de la línea`
- La importación funciona de forma similar al CSS. Se puede importar un archivo .less y todas las variables y mixin estarán disponibles. La extensión es opcional para los archivos .less.  
`@import "library"; // library.less`

© JMA 2015. All rights reserved

## Mixin Condicionales

- Se pueden definir varias versiones del mismo mixin parametrizado que dependan del cumplimiento de una condición que se resolverá en función al valor pasado en las reglas que lo utilicen:
 

```
. mixin (@a) when (lightness(@a) >= 50%) {
    background-color: black;
}
.mixin (@a) when (lightness(@a) < 50%) {
    background-color: white;
}
.mixin (@a) { color: @a; }
```
- Genera:
 

```
.class1 { .mixin(#ddd) }
.class2 { .mixin(#555) }
```

© JMA 2015. All rights reserved

## Mixin recursivos

- Un mixin puede llamarse a sí mismo y si se combinan con las condiciones se pueden usar para crear varias estructuras iterativas.
 

```
.generate-columns(@n, @i: 1) when (@i <= @n) {
  .column-@{i} {
    width: (@i * 100% / @n);
  }
  .generate-columns(@n, (@i + 1));
}
.generate-columns(4);
```
- Genera:
 

```
.column-1 { width: 25%; }
.column-2 { width: 50%; }
.column-3 { width: 75%; }
.column-4 { width: 100%; }
```

© JMA 2015. All rights reserved

# Accesibilidad

© JMA 2015. All rights reserved

## Definiciones

- La accesibilidad Web significa que personas con algún tipo de discapacidad van a poder hacer uso de la Web. En concreto, al hablar de accesibilidad Web se está haciendo referencia a un diseño Web que va a permitir que estas personas puedan percibir, entender, navegar e interactuar con la Web, aportando a su vez contenidos. La accesibilidad Web también beneficia a otras personas, incluyendo personas de edad avanzada que han visto mermadas sus habilidad a consecuencia de la edad.
- La accesibilidad web o de la interfaz, indica la capacidad de acceso a la Web y a sus contenidos por todas las personas, independientemente de la discapacidad (física, intelectual o técnica) que presenten o de las que se deriven del contexto de uso (tecnológicas o ambientales). Esta cualidad está íntimamente relacionada con la usabilidad.
- Un sitio web es accesible si las personas con discapacidad lo pueden utilizar con la misma efectividad, seguridad y protección que las personas sin discapacidad.

© JMA 2015. All rights reserved

# Limitaciones

- Las limitaciones en la accesibilidad de los sitios Web pueden ser:
  - **Visuales:** En sus distintos grados, desde la baja visión a la ceguera total, además de problemas para distinguir colores (Daltonismo).
  - **Motrices:** Dificultad o la imposibilidad de usar las manos, incluidos temblores, lentitud muscular, etc, debido a enfermedades como el Parkinson, distrofia muscular, parálisis cerebral, amputaciones, entre otras.
  - **Auditivas:** Sordera o deficiencias auditivas.
  - **Cognitivas:** Dificultades de aprendizaje (dislexia, discalculia, etc) o discapacidades cognitivas que afecten a la memoria, la atención, las habilidades lógicas, etc.
- A las personas con discapacidad podemos añadir el conjunto de personas de la "tercera edad", ya que las carencias y problemas de los medios físicos, así como muchas veces el contenido, hacen que estas personas se encuentren también en riesgo de infoexclusión.

© JMA 2015. All rights reserved

## Problemas de accesibilidad

- **Manejo de terminales:** Los teléfonos, ordenadores, cajeros automáticos y televisión digital la mayoría de las veces no están diseñados y colocados, en el caso de los cajeros, prestando atención a las necesidades de las personas con discapacidad. La variedad de terminales es muy grande, lo que se debe buscar es seguir la tendencia a reducirlos y acceder a todos los servicios a través de unos pocos.
- **Interacción con las interfaces:** Los menús, barras de navegación y botones no suelen ser accesibles desde una variedad de terminales adaptados.
- **Acceso a los contenidos:** Los contenidos a los que se tiene acceso desde un mismo dispositivo son cada vez mayores y, este rápido crecimiento no suele atender las necesidades específicas de la discapacidad.

© JMA 2015. All rights reserved

## Características de un sitio accesible

- **Transformable:** La información y los servicios deben ser accesibles para todos y deben poder ser utilizados con todos los dispositivos de navegación.
- **Comprendible:** Contenidos claros y simples.
- **Navegable:** Mecanismos sencillos de navegación.

© JMA 2015. All rights reserved

## Ayudas técnicas

- Las ayudas técnicas, también llamadas tecnologías de apoyo, son los dispositivos empleados por las personas con discapacidad para prevenir, compensar, mitigar o neutralizar la discapacidad que poseen.
- Las siguientes son algunas de las tecnologías de apoyo que usan los usuarios discapacitados para navegar de la web:
  - Un programa lector de pantalla, que puede leer usando síntesis de voz, los elementos que se muestran en el monitor (de gran ayuda para los usuarios con dificultades de aprendizaje o lectura), o que puede leer todo lo que está pasando en el PC (utilizado por los usuarios ciegos y de visión reducida).
  - Líneas Braille, que consiste en dispositivo hardware que convierte el texto en caracteres Braille.
  - Un programa magnificador de pantalla que amplía lo que se muestra en el monitor de la computadora, haciéndolo más fácil de leer para los usuarios de visión reducida.

© JMA 2015. All rights reserved

## Legislación

- Desde el año 2002, en España<sup>4</sup> se han desarrollado varias leyes que definen los niveles de accesibilidad y fechas de cumplimiento:
  - Ley 34/2002 del 11 de julio, de servicios de la sociedad de la información y de comercio electrónico.
  - Ley 51/2003 del 2 de diciembre de Igualdad de Oportunidades, No Discriminación y Accesibilidad Universal con discapacidad (LIONDAU).
  - Real Decreto 366/2007 del 16 de marzo, de accesibilidad y no discriminación de las personas con discapacidad en sus relaciones con la Administración General del Estado.
  - Ley 27/2007, del 23 de octubre, por la que se reconocen las lenguas de signos españolas y se regulan los medios de apoyo a la comunicación oral de las personas sordas, con discapacidad auditiva y sordociegas.
  - Real Decreto 1494/2007, del 12 de noviembre, por el que se aprueba el Reglamento sobre las condiciones básicas para el acceso de las personas con discapacidad a la sociedad de la información.
  - Ley 49/2007, del 26 de diciembre, por la que se establece el régimen de infracciones y sanciones en materia de igualdad de oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad.
  - Ley 56/2007, del 28 de diciembre, de Medidas de Impulso de la Sociedad de la Información que modifica la redacción de la disposición adicional quinta de la Ley 34/2002, del 11 de julio.
- Normativa UNE 139803:2012
  - En 2012 se actualizó para adoptar las WCAG 2.0 como base.

© JMA 2015. All rights reserved

## Están obligadas

- Las Administraciones públicas
- Las empresas que presten servicios al público en general de especial trascendencia económica, las que agrupen a más de cien trabajadores o su volumen anual de operaciones, calculado conforme a lo establecido en la normativa del Impuesto sobre el Valor Añadido, excede de 6.010.121,04 euros y que, en ambos casos, operen en los siguientes sectores económicos:
  - Servicios de comunicaciones electrónicas a consumidores
  - Servicios financieros destinados a consumidores (Servicios bancarios, de crédito o de pago, de inversión, de seguros privados, de corredor de seguros, Planes de pensiones)
  - Servicios de suministro de agua a consumidores
  - Servicios de suministro de gas al por menor
  - Servicios de suministro eléctrico a consumidores finales
  - Servicios de agencia de viajes
  - Servicios de transporte de viajeros por carretera, ferrocarril, por vía marítima, o por vía aérea
  - Actividades de comercio al por menor

© JMA 2015. All rights reserved

## Beneficios

- Uno de los visitantes más fieles de nuestro sitio web es un usuario con discapacidad. Es ciego (no ve las imágenes, al menos de momento, ni los vídeos, ni admira la belleza de nuestras animaciones), es sordo, navega sin plugins instalados, sin applets y sin JavaScript activo, por lo tanto le es imposible seguir los enlaces que dependen de JavaScript.
- Jakob Nielsen le llama el “usuario ciego más rico del mundo” [Nielsen, 2012]. Este usuario es Googlebot, el robot de búsqueda de Google.
- La mitad de las visitas a tu sitio vienen de Google, y Google sólo ve lo que un ciego puede ver. Si tu sitio no es accesible, tendrás menos visitas. Fin de la historia.
- Esta es la razón por la cual muchas de las técnicas que aplicamos para hacer nuestra web más accesible repercuten directa y positivamente en su indexación y posicionamiento en los buscadores (SEO).

© JMA 2015. All rights reserved

## Beneficios

- Aumenta el número de potenciales visitantes de la página web: esta es una razón muy importante para una empresa que pretenda captar nuevos clientes. Cuando una página web es accesible no presenta barreras que dificulten su acceso, independientemente de las condiciones del usuario y es más probable que se visualice correctamente en cualquier dispositivo con cualquier navegador.
- Disminuye los costes de desarrollo y mantenimiento: aunque inicialmente aprender a hacer una página web accesible supone un coste, una vez se tienen los conocimientos, el coste de desarrollar y mantener una página web accesible es menor que frente a una no accesible, dado que es menos propensa a contener errores y más sencilla de actualizar.
- Reduce el tiempo de carga de las páginas web y la carga del servidor web: al separar el contenido de la información sobre la presentación de una página web mediante CSS se logra reducir el tamaño de las páginas web y, por tanto, se reduce el tiempo de carga de las páginas web.
- Aumenta la usabilidad de la página web: esto también implica indirectamente, que la página podrá ser visualizada desde cualquier navegador.
- Demostramos que nos implicamos socialmente.
- Aumenta el capital humano de las comunidades de aprendizaje potenciando la inteligencia colectiva..

© JMA 2015. All rights reserved

Web Accessibility Initiative

**WAI**

© JMA 2015. All rights reserved

## Pautas de accesibilidad Web

- El máximo organismo dentro de la jerarquía de Internet que se encarga de promover la accesibilidad es el World Wide Web Consortium (W3C), en especial su grupo de trabajo Web Accessibility Initiative (WAI). En 1999 el WAI publicó la versión 1.0 de sus pautas de accesibilidad Web. Con el paso del tiempo se han convertido en un referente internacionalmente aceptado. En diciembre del 2008 las WCAG 2.0 fueron aprobadas como recomendación oficial.
- Estas pautas se dividen en tres bloques:
  - Pautas de Accesibilidad al Contenido en la Web (WCAG): Están dirigidas a los webmasters e indican cómo hacer que los contenidos del sitio web sean accesibles.
  - Pautas de Accesibilidad para Herramientas de Autor (ATAG): Están dirigidas a los desarrolladores del software que usan los webmasters, para que estos programas faciliten la creación de sitios accesibles.
  - Pautas de Accesibilidad para Agentes de Usuario (UAAG): Están dirigidas a los desarrolladores de Agentes de usuario (navegadores y similares), para que estos programas faciliten a todos los usuarios el acceso a los sitios Web.

© JMA 2015. All rights reserved

# Guía breve para crear sitios webs accesibles

- Imágenes y animaciones: Use el atributo alt para describir la función de cada elemento visual.
- Mapas de imagen: Use el elemento map y texto para las zonas activas.
- Multimedia: Proporcione subtítulos y transcripción del sonido, y descripción del vídeo.
- Enlaces de hipertexto: Use texto que tenga sentido leído fuera de contexto. Por ejemplo, evite "pincha aquí".
- Organización de las páginas: Use encabezados, listas y estructura consistente. Use CSS para la maquetación donde sea posible.
- Figuras y diagramas: Describalos brevemente en la pagina o use el atributo longdesc.
- Scripts, applets y plug-ins: Ofrezca contenido alternativo si las funciones nuevas no son accesibles.
- Marcos: Use el elemento noframes y títulos con sentido.
- Tablas: Facilite la lectura línea a línea. Resuma.
- Revise su trabajo: Verifique.

© JMA 2015. All rights reserved

## Pautas de Accesibilidad del Contenido en la Web (WCAG 1.0)

1. Proporcione alternativas equivalentes para el contenido visual y auditivo
2. No se base sólo en el color
3. Utilice marcadores y hojas de estilo y hágalo apropiadamente
4. Identifique el idioma usado
5. Cree tablas que se transformen correctamente
6. Asegúrese de que las páginas que incorporen nuevas tecnologías se transformen correctamente
7. Asegure al usuario el control sobre los cambios de los contenidos tempo-dependientes
8. Asegure la accesibilidad directa de las interfaces incrustadas
9. Diseñe para la independencia del dispositivo
10. Utilice soluciones provisionales
11. Utilice las tecnologías y pautas W3C
12. Proporcione información de contexto y orientación
13. Proporcione mecanismos claros de navegación
14. Asegúrese de que los documentos sean claros y simples

[http://www.saregune.net/ikasi/hezigune/accesibilidad/documentacion/WAI-WEBCONTENT-19990505\\_es.html](http://www.saregune.net/ikasi/hezigune/accesibilidad/documentacion/WAI-WEBCONTENT-19990505_es.html)

© JMA 2015. All rights reserved

## Orientación de las WCAG 2.0

- Principios - En el nivel más alto se sitúan los cuatro principios que proporcionan los fundamentos de la accesibilidad web: perceptible, operable, comprensible y robusto.
- Pautas - Por debajo de los principios están las pautas. Las doce pautas proporcionan los objetivos básicos que los autores deben lograr con el fin de crear un contenido más accesible para los usuarios con distintas discapacidad. Estas pautas no son verificables, pero proporcionan el marco y los objetivos generales que ayudan a los autores a comprender los criterios de conformidad y a implementar mejor las técnicas.
- Criterios de Conformidad - Para cada pauta se proporcionan los criterios de conformidad verificables que permiten emplear las WCAG 2.0 en aquellas situaciones en las que existan requisitos y necesidad de evaluación de conformidad como: especificaciones de diseño, compras, regulación o acuerdos contractuales. Con el fin de cumplir con las necesidades de los diferentes grupos y situaciones, se definen tres niveles de conformidad: A (el más bajo), AA y AAA (el más alto).

© JMA 2015. All rights reserved

## Orientación de las WCAG 2.0

- Técnicas suficientes y recomendables - Para cada una de las pautas y criterios de conformidad del propio documento de las WCAG 2.0, el grupo de trabajo ha documentado también una amplia variedad de técnicas. Las técnicas son informativas y se agrupan en dos categorías: aquellas que son suficientes para satisfacer los criterios de conformidad, y aquellas que son recomendables. Las técnicas recomendables van más allá de los requisitos de cada criterio de conformidad individual y permiten a los autores afrontar mejor las pautas. Algunas de las técnicas recomendables tratan sobre barreras de accesibilidad que no han sido cubiertas por los criterios de conformidad verificables. También se han documentado los errores frecuentes que son conocidos.

© JMA 2015. All rights reserved

# Principios y pautas

<http://www.sidar.org/traducciones/wcag20/es/>

- 1 Perceptible
  - 1.1 Proporcionar alternativas textuales para todo contenido no textual de modo que se pueda convertir a otros formatos que las personas necesiten, tales como textos ampliados, braille, voz, símbolos o en un lenguaje más simple.
  - 1.2 Medios tempodependientes: proporcionar alternativas para los medios tempodependientes.
  - 1.3 Crear contenido que pueda presentarse de diferentes formas (por ejemplo, con una disposición más simple) sin perder información o estructura.
  - 1.4 Facilitar a los usuarios ver y oír el contenido, incluyendo la separación entre el primer plano y el fondo.

© JMA 2015. All rights reserved

# Principios y pautas

- 2 Operable
  - 2.1 Proporcionar acceso a toda la funcionalidad mediante el teclado.
  - 2.2 Proporcionar a los usuarios el tiempo suficiente para leer y usar el contenido.
  - 2.3 No diseñar contenido de un modo que se sepa podría provocar ataques, espasmos o convulsiones.
  - 2.4 Proporcionar medios para ayudar a los usuarios a navegar, encontrar contenido y determinar dónde se encuentran.
- 3 Comprensible
  - 3.1 Hacer que los contenidos textuales resulten legibles y comprensibles.
  - 3.2 Hacer que las páginas web aparezcan y operen de manera predecible.
  - 3.3 Ayudar a los usuarios a evitar y corregir los errores.
- 4 Robusto
  - 4.1 Maximizar la compatibilidad con las aplicaciones de usuario actuales y futuras, incluyendo las ayudas técnicas.

© JMA 2015. All rights reserved

# Conformidad

- Nivel de conformidad: Uno de los siguientes niveles de conformidad se satisface por completo.
  - Nivel A (el mínimo): la página web satisface todos los Criterios de Conformidad del Nivel A, o proporciona una versión alternativa conforme.
  - Nivel AA: la página web satisface todos los Criterios de Conformidad de los Niveles A y AA, o se proporciona una versión alternativa conforme al Nivel AA.
  - Nivel AAA: la página web satisface todos los Criterios de Conformidad de los Niveles A, AA y AAA, o proporciona una versión alternativa conforme al Nivel AAA.
- Páginas completas: La conformidad (y el nivel de conformidad) se aplica a páginas web completas, y no se puede alcanzar si se excluye una parte de la página.
- Procesos completos: Cuando una página web es parte de una serie de páginas web que presentan un proceso (es decir, una secuencia de pasos que es necesario completar para realizar una actividad), todas las páginas en ese proceso deben ser conformes con el nivel especificado o uno superior.

© JMA 2015. All rights reserved

# Conformidad

- Uso de tecnologías exclusivamente según métodos que sean compatibles con la accesibilidad: Para satisfacer los criterios de conformidad sólo se depende de aquellos usos de las tecnologías que sean compatibles con la accesibilidad. Toda información o funcionalidad que se proporcione de una forma que no sea compatible con la accesibilidad debe estar disponible de una forma que sí sea compatible con la accesibilidad.
- Sin interferencia: Si las tecnologías se usan de una forma que no es compatible con la accesibilidad, o está usada de una forma que no cumple los requisitos de conformidad, no debe impedir a los usuarios acceder al contenido del resto de la página. Además, es necesario que la página web como un todo siga cumpliendo con los requisitos de conformidad en las siguientes circunstancias:
  - cuando cualquier tecnología de la que no se depende está activada en una aplicación de usuario,
  - cuando cualquier tecnología de la que no se depende está desactivada en una aplicación de usuario, y
  - cuando cualquier tecnología de la que no se depende no es soportada por una aplicación de usuario

© JMA 2015. All rights reserved

## Conformidad

- Además, los siguientes criterios de conformidad se aplican a todo el contenido de la página, incluyendo el contenido del que, de todos modos, no se depende para alcanzar la conformidad, ya que su incumplimiento puede interferir con el uso de la página:
  - 1.4.2 - Control del audio,
  - 2.1.2 - Sin trampas para el foco del teclado,
  - 2.3.1 - Umbral de tres destellos o menos, y
  - 2.2.2 - Poner en pausa, detener, ocultar.

© JMA 2015. All rights reserved

## Declaraciones de conformidad

- La conformidad se aplica sólo a las páginas web. Sin embargo, la declaración de conformidad puede cubrir una sola página, una serie de páginas o múltiples páginas web relacionadas.
- Las declaraciones de conformidad no son obligatorias. Los autores pueden cumplir con los requisitos de las WCAG 2.0 sin realizar la declaración. Sin embargo, si se realiza la declaración, ésta debe contener la siguiente información:
  - **Fecha** de la declaración
  - **Título de las pautas**, versión y URI "Web Content Accessibility Guidelines 2.0 en <http://www.w3.org/TR/2008/REC-WCAG20-20081211/>"
  - **Nivel de conformidad satisfecho:** (Nivel A, AA o AAA)
  - Una **breve descripción de las páginas web**, como por ejemplo una lista de sus URI para las que se hace la declaración, incluyendo si los subdominios están incluidos en la declaración.
  - Una **lista de las tecnologías** de contenido web de las que se depende.

© JMA 2015. All rights reserved

Accessible Rich Internet Applications  
<https://www.w3.org/TR/wai-aria-1.1/>

## WAI-ARIA

© JMA 2015. All rights reserved

# Introducción

- El W3C define ARIA como:  
*La forma para crear contenido Web y aplicaciones Web que sean accesibles para las personas con discapacidades.*
- Una de las problemáticas más recientes de accesibilidad en la Web surgió con la llegada de Ajax, momento en el que los desarrolladores se "animaron" a usar esta tecnología motivados por sus posibilidades para la actualización dinámica del contenido y la **creación de diferentes controles a medida** (widgets). Todo esto con el fin de simular una GUI más parecida a las de las aplicaciones de escritorio, obteniendo como resultado aplicaciones web más ricas e interactivas, pero menos accesibles.
- La creación de widgets que combinan etiquetas HTML, CSS y JavaScript para definir nuevos comportamientos separan los elementos originales del estándar por lo que presentan problemas con las ayudas técnicas o tecnologías de apoyo como el lector de pantalla.

© JMA 2015. All rights reserved

# ARIA

- ARIA proporciona un marco de trabajo complementario:
  - Estructuras más semánticas para las zonas funcionales.
  - Mejora de la navegación mediante el teclado.
  - Controles complejos (widgets) más accesibles.
  - Accesibilidad para el contenido actualizado de forma dinámica.
- Para ello ARIA cuenta con:
  - Roles: su misión es definir el papel que juegan los elementos dentro del documento web.
  - Estados y propiedades: determinan las características y los valores de cada elemento.
- Por tanto, ARIA no funciona como una tecnología restrictiva o exclusiva, sino que se trata de un complemento con el que podemos hacer accesibles las aplicaciones web enriquecidas.

© JMA 2015. All rights reserved

## Estructura semántica de un documento

- Con ARIA podemos especificar los roles de las diferentes zonas funcionales de un documento web, haciendo que sea más semántico y accesible.
- Por ejemplo, actualmente estamos acostumbrados a usar un enlace para saltar al contenido principal (skip to content), mientras que ARIA nos permite, a través de los Document Landmarks, crear una estructura más semántica y accesible para que, entre otras cosas, las tecnologías asistivas puedan discernir donde se encuentra el contenido principal sin falta de recurrir a un enlace.
- Para crear una estructura semántica accesible con ARIA únicamente tenemos que especificar los roles de cada zona funcional por medio de la propiedad role:
  - application, banner, complementary, contentinfo, form, main, navigation, search

```
<div id="navigation" role="navigation">
```

© JMA 2015. All rights reserved

## Navegación mediante el teclado

- Por medio del teclado y a través del atributo tabindex podemos navegar entre los diferentes elementos de una web, pero sólo algunos elementos soportan este atributo y son capaces de recibir el foco: A, AREA, BUTTON, INPUT, OBJECT, SELECT, y TEXTAREA. Por tanto, un simple elemento DIV no puede ser accedido a través del teclado.
- Es justo aquí donde hace acto de presencia ARIA:
  - Haciendo que el atributo tabindex sea soportado por todos los elementos visibles de una web.
  - Permitiendo el valor “-1” en el atributo tabindex, lo que posibilita sacar un elemento del orden natural de navegación y del orden expresado por el índice de tabulación. Un elemento con el atributo tabindex="-1" únicamente podrá recibir el foco por medio de JavaScript (con el método focus()).

© JMA 2015. All rights reserved

## Accesibilidad en los controles (widgets)

- HTML fue concebido para compartir documentos en la web y no para crear aplicaciones, razón por la cual no ofrece controles (widgets) avanzados.
- El diseño y desarrollo de un control de este tipo puede tener el siguiente aspecto:
 

```
<div id="slider-bg" title="level">
  <div id="slider-handler"></div>
</div>
```


- Problema de la accesibilidad resuelto con ARIA:
 

```
<p id="slider-description">Puede usar las teclas derecha/izquierda para cambiar el nivel.</p>
<span id="slider-label">Nivel:</span>
<div id="slider-rail">
  <button id="slider-handler" role="slider" aria-labelledby="slider-label" aria-describedby="slider-description" aria-valuemin="1" aria-valuemax="3" aria-valuenow="2"></button>
</div>
```

© JMA 2015. All rights reserved

## Accesibilidad en actualizaciones dinámicas de contenido

- Otro de los grandes problemas de la accesibilidad en las aplicaciones web enriquecidas, recae sobre la actualización dinámica del contenido (o parte del contenido) que se realiza por medio de Ajax y en “segundo plano”.
- ARIA denomina “regiones activas” a los elementos/zonas que pueden presentar estos cambios, y cuenta con la propiedad aria-live con la que indicar el valor de “intrusismo” (off, polite, assertive o rude) sobre la actividad actual del usuario.

```
<label for="user-name">
    Nombre (requerido)
    <input name="user-name" id="user-name" type="text" size="20"
           maxlength="20" aria-required="true"/>
    <span id="count" aria-live="polite"></span> pendientes.
</label>
```

© JMA 2015. All rights reserved

## Implementando ARIA

- La dificultad de implementar ARIA puede ser considerada proporcional al grado de complejidad de la aplicación web, aunque al final se trata siempre de gestionar roles, estados y propiedades por medio de JavaScript.
- Inyectando ARIA con jQuery:

```
$(document).ready(function() {
    $('#logo').attr('role', 'banner');
    $('#nav').attr('role', 'navigation');
    $('#searchform').attr('role', 'search');
    $('#main').attr('role', 'content');
    $('#footer').attr('role', 'contentinfo');
    $('.required').attr('aria-required', 'true');
});
```

© JMA 2015. All rights reserved

# JavaScript

ECMA-262

ISO/IEC-16262

© JMA 2015. All rights reserved

## INTRODUCCIÓN

© JMA 2015. All rights reserved

# Introducción

- El JavaScript es un lenguaje de secuencias de comandos multiplataforma e independiente de cualquier empresa, heredero de la sintaxis del C, que originalmente fue diseñado para ir inmerso dentro de las páginas web, ser interpretado por el navegador y hacer programable el HTML.
- Creado por Brendan Eich para el navegador Netscape Navigator 2.0, que iba a lanzarse en 1995. Inicialmente, se denominó LiveScript.
- Justo antes del lanzamiento, tras la alianza de Netscape con Sun Microsystems, se decidió cambiar el nombre por el de JavaScript. La razón del cambio de nombre fue exclusivamente por marketing, ya que Java era la palabra de moda en el mundo informático y de Internet de la época, no existe ninguna relación con el lenguaje Java de SUN salvo la común herencia de la sintaxis con el C.
- Al mismo tiempo, Microsoft lanzó JScript con su navegador Internet Explorer 3. JScript era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales.
- En 1997, Netscape decidió liberar el lenguaje y lo estandarizó a través del ECMA. El primer estándar se denominó ECMA-262, en el que se definió por primera vez el lenguaje ECMAScript, quedando el nombre de JavaScript como la implementación que realizó la empresa Netscape del estándar ECMAScript. La ISO adoptó el estándar ECMA-262 dando lugar al estándar ISO/IEC-16262.

© JMA 2015. All rights reserved

# Especificaciones oficiales

- ECMA ha publicado varios estándares relacionados con ECMAScript.
- En Junio de 1997 se publicó la primera edición del estándar ECMA-262.
- Un año después, en Junio de 1998 se realizaron pequeñas modificaciones para adaptarlo al estándar 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 soportan todos los navegadores actuales.
- La cuarta versión de ECMA-262 no llegó a publicarse.
- En junio de 2011, se publicó la edición 5.1 que solo soporta las versiones más modernas de los navegadores.
- Entre las nuevas características se incluyen: descriptor de acceso de propiedades, reflexión para la creación e inspección de objetos, el control por código de los atributos de propiedad, funciones adicionales de manipulación de matriz, soporte para el formato de codificación de objetos JSON y un modo estricto que proporciona una mayor comprobación de errores y seguridad del código.
- Se puede consultar la especificación completa en:
  - <http://www.ecma-international.org/publications/standards/Ecma-262.htm>

© JMA 2015. All rights reserved

## Añadir JavaScript a una página

- Existen dos formas de insertar código JavaScript dentro de una página: escribiendo código en la misma (en inglés inline) o a través de un archivo externo utilizando la etiqueta script.
- El orden en el cual se incluye el código es importante:
  - Un código que depende de otro debe ser incluido después del que referencia.
  - El código es interpretado secuencialmente desde el inicio de la página.
  - Para mejorar el rendimiento de la página, el código JavaScript debe ser incluido al final del BODY.
  - Es conveniente, cuando se trabaja en un ambiente de producción con múltiples archivos JavaScript, que éstos sean combinados en un solo archivo minimizado.

© JMA 2015. All rights reserved

## Etiquetas

- Interno:  

```
<script type="text/javascript">
  ... Código JavaScript ...
</script>
```
- Externo:  

```
<script type="text/javascript" src="/js/codigo.js"></script>
```
- En línea:  

```
<eti evento="javascript: ... Código JavaScript ..." ...>
```
- Para los navegadores que no disponen de soporte completo de JavaScript o en los que ha sido bloqueado o inhabilitado por el usuario.  

```
<noscript>
  ... Aviso en HTML ...
</noscript>
```

© JMA 2015. All rights reserved

# TIPOS, VARIABLES Y EXPRESIONES

© JMA 2015. All rights reserved

## Sintaxis

- Sensible a mayúsculas y minúsculas.
- Sentencias separadas por punto y coma (;), aunque opcional en algunas ocasiones es recomendable utilizarlo siempre.
- Formato libre en las sentencias, una sentencia puede utilizar varias líneas y una línea puede incluir varias sentencias.
- Los espacios en blanco se compactan a uno solo.
- Bloques marcados por llaves: { <sentencias> }
- Comentarios:
  - // hasta el final de la línea
  - /\* ... \*/ una o varias líneas

© JMA 2015. All rights reserved

## Separadores

- ( ) para contener listas de parámetros en los métodos o funciones, expresiones de control de flujo y establecer precedencias en las expresiones.
- { } para definir bloques de código e inicializar objetos.
- [ ] para declarar y referencias tablas.
- ; separar instrucciones.
- , separar identificadores en su declaración y concatenar sentencias dentro del for.
- . separador decimal y para hacer referencia a los atributos y métodos.

© JMA 2015. All rights reserved

## Identificadores

- Compuestos por letras, números, \_, \$.
- Sin límite en el número de caracteres.
- NO pueden comenzar con números.
- NO debe tener el mismo nombre que otro elemento del mismo ámbito.
- NO pueden ser palabras reservadas ni el nombre de un valor con nombre (true / false / null / undefined).
- Palabras reservadas:
  - break, case, catch, continue, debugger, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with
- Palabras reservadas para uso futuro:
  - class, const, enum, export, extends, import, super, implements, interface, let, package, private, protected, public, static, yield

© JMA 2015. All rights reserved

# Tipos de datos

- JavaScript divide los distintos tipos de datos en dos grupos:
  - Tipos del lenguaje:
    - Tipos primitivos: undefined, null, boolean, string, number
    - Tipos de referencia: Undefined, Null, Boolean, String, Number, Object.
  - Tipos de especificación (meta-valores): Reference, List, Completion, Property Descriptor, Property Identifier, Lexical Environment, Environment Record
- JavaScript define algunos objetos de forma nativa, por lo que pueden ser utilizados directamente por las aplicaciones sin tener que declararlos:
  - Global, Function, Array, Math, Date, Error, JSON y RegExp

© JMA 2015. All rights reserved

# Tipos de datos

- Tipo Undefined
  - Tipo con un solo valor llamado “undefined”.
  - Valor de las variables que no han sido aun declaradas o que no se les ha asignado un valor.
- Tipo Null
  - Tipo con un solo valor llamado “null”.
- Tipo Boolean
  - Sólo puede almacenar uno de los dos valores lógicos: “true” o “false”.
  - Los valores false, 0, null y undefined son considerados como false en las operaciones lógicas y el resto de los valores como true.

© JMA 2015. All rights reserved

## Tipo numérico

- Acepta Valores aritméticos enteros y reales. Si el número es real, se debe utilizar el punto (.) para separar la parte entera de la decimal.
- Formatos:
  - Decimal
  - octal (0????)
  - hexadecimal (0x????)
  - coma flotante (?.**??e??**).
- JavaScript define tres valores especiales muy útiles cuando se trabaja con números.
  - Se definen los valores Infinity y -Infinity para representar números demasiado grandes (positivos y negativos) y con los que JavaScript no puede trabajar.
  - NaN (Not a Number), resultado de las operaciones matemáticas con variables no numéricas. isNaN() informa de que el valor no puede ser usado como numérico.
- JavaScript define constantes y operaciones matemáticas adicionales a través del objeto Math.

© JMA 2015. All rights reserved

## Tipo cadena

- Conjunto de caracteres encerrados entre comillas simples (apóstrofes) o dobles.
- Permite el anidamiento de una cadena entre comillas dobles dentro de una cadena entre comillas simples y viceversa.
- Las constantes de tipo cadena no pueden ocupar mas de una línea, para utilizar varias líneas requiere concatenación.
- El formato de las cadenas es UTF8 y pueden contener secuencias de escape:
  - \n, \r, \\, \", \', \t, \v, \e, \f
  - \[0-9]{1,3}
  - \x[0-9A-Fa-f]{2}
  - \u[0-9A-Fa-f]{4}

© JMA 2015. All rights reserved

# Conversión entre tipos

- JavaScript es un lenguaje "no tipado", lo que significa que una misma variable puede guardar diferentes tipos de datos a lo largo de la ejecución de la aplicación.
- En JavaScript las conversiones entre tipos numéricos y cadenas se realiza de forma implícita.
- Los valores aritméticos se convierten automáticamente en cadenas en las operaciones de concatenación.
- En caso de ser necesario, para realizar las conversiones de forma explícita se utiliza:
  - El método `toString()` que permite convertir valores de cualquier tipo a valores de tipo cadena.
  - Las funciones `parseInt()` y `parseFloat()` convierten la cadena en un número entero o decimal respectivamente.
    - Si la cadena no contiene un valor aritmético o no empieza por un dígito devuelve el valor `Nan`.
    - La conversión numérica de una cadena se realiza carácter a carácter empezando por el de la primera posición hasta que encuentra un carácter que no es un dígito o termina la cadena.

© JMA 2015. All rights reserved

# Operadores

- Asignación  
Simple: <Destino> = <Expresión>
- Aritméticos
  - + : Suma, si el algún operando es una cadena se concatena.
  - : Resta, cambio de signo cuando es unario.
  - \* : Producto
  - / : División, el resultado es siempre real.
  - % : Resto de la división entera
- Acumulativos: `+=`, `-=`, `*=`, `/=`, `%=`
- Operadores relacionales
  - > : mayor
  - `>=` : mayor o igual
  - < : menor
  - `<=` : menor o igual
  - `==` : igual
  - `!=` : distinto
  - `==` : identidad (coincide el tipo)
  - `!=` : no identidad

© JMA 2015. All rights reserved

# Operadores

- Operadores lógicos (los operandos son booleanos)
 

&&	: AND
	: OR
!	: NOT
- Binarios
 

>>	: Desplazamiento Derecha
<<	: Desplazamiento Izquierda
>>>	: Desplazamiento Derecha sin signo
&	: AND binario
	: OR binario
^	: XOR binario
~	: complemento
- Acumulativos: >>=, <<=, >>>=, &=, |=, ^=, ~=
- Incremento/decuento:
 

++<Variable>	: Se incrementa en 1 y se consulta.
--<Variable>	: Se decrementa en 1 y se consulta.
<Variable>++	: Se consulta y se incrementa en 1.
<Variable>--	: Se consulta y se decrementa en 1.

© JMA 2015. All rights reserved

# Operadores

- Operador unario: **new**
  - Crea un nuevo objeto.
- Operador unario: **delete**
  - Elimina una propiedad de un objeto o quita un elemento de una matriz.
- Operador unario: **void**
  - Descarta el operando y devuelve no definido (undefined).
- Operador unario de tipo: **typeof**
  - Devuelve una cadena con el tipo de la variable, los valores devueltos son: "number", "string", "boolean", "object", "function" y "undefined".
- Operador binario de pertenencia a tipo: **instanceof**
  - Devuelve una true si es del tipo indicado.
- Operador binario de pertenencia: **in**
  - Comprueba la existencia de una propiedad en un objeto: property in object
- Operador condicional ternario:
  - <condición>?<expresión cuando verdadero>:<expresión cuando falso>

© JMA 2015. All rights reserved

# Precedencia de operadores

Operador	Descripción
. [] ()	Acceso a campos, indexación de matrices y llamadas a funciones
++ -- ~ ! typeof new void delete	Operadores unarios, tipos de datos devueltos, creación de objetos, valores no definidos
* / %	Multiplicación, división, división módulo
+ - +	Adición, sustracción, concatenación de cadenas
<< >> >>	Desplazamiento de bits
< <= > >=	Menor que, menor que o igual a, mayor que, mayor que o igual a
== != === !==	Igualdad, desigualdad, identidad, no identidad
&	AND de bits
^	XOR de bits
	OR de bits
&&	AND lógico
	OR lógico
?:	Condicional
= OP=	Asignación, asignación con operación
,	Evaluación múltiple

© JMA 2015. All rights reserved

# Variables

- No es necesario declararlas de forma explícita, aunque sí recomendable.
- Se declaran sin tipo, van asumiendo el tipo del valor contenido.
- Declaración:  
`var <Identificador> [= expresión], <Otro> [= expresión], ...;`
- Ámbito:
  - Local:
    - Función en la que está declarada
    - Accesible desde el propio bloque y todos sus bloques anidados.
  - Global:
    - Declarada fuera de cualquier bloque.
    - Las variables no declaradas son siempre globales.
    - Accesibles desde cualquier punto.
  - Las variables locales prevalecen sobre las globales

© JMA 2015. All rights reserved

# Array

- Un array en JavaScript es una colección indexada numéricamente que permite almacenar múltiples valores.
- Cada valor del array puede ser de cualquier tipo.
- Los valores de índice siempre son numéricos, empezando en 0 hasta n, sin tamaño predefinido.
  - Si se accede a un elemento que no existe se obtiene undefined.
  - Si se asigna un elemento que no existe se crea automáticamente.
- Los corchetes [] actúan como selectores y delimitadores de los arrays.
- Declaración (vacío, sin dimensión):
 

```
var <Identificador> = new Array();
var <Identificador> = [];
```

© JMA 2015. All rights reserved

# Array

- Declaración (initializado, con dimensión):
 

```
var <Identificador> = new Array(valor1, valor2, valor3);
var <Identificador> = [valor1, valor2, valor3];
```
- Recuperación de elementos:
 

```
var destino = tabla[indice];
```
- Creación o modificación de elementos:
 

```
tabla[indice] = valor;
```
- No se pueden asignar arrays completos, si se asigna un array completo a otro se obtiene una doble referencia sobre el array asignado.
- Los elementos pueden ser, a su vez, otros arrays con lo que se pueden crear arrays multidimensionales.
- En realidad, en JavaScript, los array son objetos con sus correspondientes propiedades y métodos, que conservan la notación [] por semejanza con otros lenguajes.

© JMA 2015. All rights reserved

## INSTRUCCIONES DE CONTROL

© JMA 2015. All rights reserved

### Bifurcación simple

```
if(<Condición>
    <Instrucción>; o {<Bloque de instrucciones>}
    [
    else
        <Instrucción>; o {<Bloque de instrucciones>}
    ]
```

- Sin límite en los anidamientos.
- La parte else es opcional. Para evitar confusión es correcto crear bloques cuando existan anidamientos.

© JMA 2015. All rights reserved

# Bifurcación múltiple

```
switch(<Expresión>) {
    case <Valor 1> :
        <Bloque de instrucciones>
        [break;]
    case <Valor 2> :
        <Bloque de instrucciones>
        [break;]

    ...
    [default:
        <Bloque de instrucciones>
    ]
}
```

- Se ejecutan las instrucciones en cascada hasta el final o hasta que encuentra un **break**.
- Sin límite en los anidamientos.
- La parte **default** es opcional, se ejecuta cuando no se cumplen las condiciones anteriores.

© JMA 2015. All rights reserved

# Bucles

```
for([var] <Inicio>; <Final>; <Iteración>)
    ; o <Instrucción>; o {<Bloque de instrucciones>}
```

- Apartados (todos son opcionales):
  - <Inicio> : Se ejecutan antes de comenzar. Lista de expresiones separadas por comas. La palabra reservada **var** es opcional.
  - <Final> : Expresión condicional de finalización, se ejecuta mientras se cumpla la condición.
  - <Iteración> : Se ejecutan en cada iteración. Lista de expresiones separadas por comas.

```
for([var] <Variable> in <Objeto>)
    <Instrucción>; o {<Bloque de instrucciones>}
```

- A <Variable> se le asignan uno a uno todos los valores contenido en el <Objeto>.

© JMA 2015. All rights reserved

# Bucles

```
while(<Condición>
      ; o <Instrucción>; o {<Bloque de instrucciones>})
```

```
do
    <Instrucción>; o <Bloque de instrucciones>
while (<Condición>);
```

- **while:** Se ejecuta de 0 a n veces.
- **do while:** Se ejecuta de 1 a n veces.

© JMA 2015. All rights reserved

# Control de flujo

- **break;**
  - Termina la instrucción asociada y pasa a la siguiente.
- **continue;**
  - Detiene el ciclo actual de un bucle y salta a evaluar la condición.
- **return [<Expresión>];**
  - Termina la ejecución de la función y, opcionalmente, devuelve el resultado de la misma;

© JMA 2015. All rights reserved

# Tratamiento de excepciones

```
try
    {<Bloque de instrucciones>}
  catch (<Identificador>)
    {<Bloque de instrucciones>}
  finally
    {<Bloque de instrucciones>}
```

- La sección catch solo se ejecuta en caso de error. El <Identificador> contiene un objeto de tipo Error con la información adicional sobre la excepción.
- La sección finally se ejecuta siempre, independientemente de si se produce o no la excepción.
- Las secciones catch y finally son opcionales pero al menos debe aparecer una.
- Se pueden anidar instrucciones try.
- El código de las secciones catch y finally no se encuentra vigilado por lo que es necesario envolverlo en una instrucción try si es susceptible de producir excepciones.
- Para lanzar excepciones se utiliza:  
`throw new Error(<Mensaje>);`

© JMA 2015. All rights reserved

# Otras instrucciones

- **with (<objeto>)**
  - {<Bloque de instrucciones>}
  - Selector de objeto: No es necesario utilizar el punto para la selección de sus métodos y atributos.
- **debugger;**
  - Provoca un punto de interrupción cuando se ejecuta en un depurador, si un depurador no está presente o activo esta declaración no tiene ningún efecto observable.

© JMA 2015. All rights reserved

# FUNCIONES

© JMA 2015. All rights reserved

## Sintaxis

```
function Identificador ([<Lista de parámetros>]) {  
    ...  
    [return [<Expresión>];]  
}
```

- Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente, permitiendo modularizar el código.
- Una función puede devolver un valor y participar en expresiones.
- No es obligatorio que las funciones tengan una instrucción de tipo return para devolver valores. Cuando una función no devuelve ningún valor o cuando en la instrucción return no se indica ningún valor, automáticamente se devuelve el valor undefined.
- Una función puede contener en su interior otras funciones anidadas.

© JMA 2015. All rights reserved

## Lista de parámetros

- Una función puede recibir uno o varios valores para realizar sus operaciones.
  - La lista de parámetros es una lista de identificadores separados por comas.
  - Al ser el JavaScript un lenguaje "no tipado", no es posible asegurar que los parámetros que se pasan a una función sean del tipo adecuado para las operaciones que realiza la función.
  - El número de argumentos que se pasa a una función debería ser el mismo que el número de argumentos que ha indicado la función. No obstante, JavaScript no muestra ningún error si se pasan más o menos argumentos de los necesarios.
  - Si se pasan menos parámetros que los definidos en la función, al resto de parámetros hasta completar el número correcto se les asigna el valor `undefined`.
  - Si a una función se le pasan más parámetros que los que ha definido, los parámetros sobrantes se ignoran.
  - El orden de los argumentos es fundamental, ya que el primer dato que se indica en la llamada, será el primer valor que espera la función; el segundo valor indicado en la llamada, es el segundo valor que espera la función y así sucesivamente.
  - Se puede utilizar un número ilimitado de argumentos.
- ```
function fn(p1, p2, p3) { ... }
rslt = fn('algo', var1, 3*pi);
```

© JMA 2015. All rights reserved

## Lista de parámetros variables

- Una función se puede definir sin parámetros pero utilizar los parámetros pasados.
- La propiedad de la función `arguments` es un objeto de tipo `Arguments` que se puede tratar como si fuera un `Array`.
 

```
function avg() {
    var rslt= 0;
    for(var param in arguments) {
        rslt += param;
    }
    return arguments.length ? (rslt / arguments.length) : 0;
}

var variable1 = avg(1, 3, 5, 8);
var variable2 = avg(4, 6, 8, 1, 2, 3, 4, 5);
```
- La propiedad `callee` hace referencia a la función que se está ejecutando, accediendo a `arguments.callee.length` se puede obtener el número de parámetros con los que se ha definido la función.

© JMA 2015. All rights reserved

# Tipo Function

- Las funciones son objetos de tipo Function y como tal se pueden almacenar en variables.
- ```
function suma(a, b) {
    return a + b;
}
var miFuncion = suma;
rslt = miFuncion(2, 2);
```
- La asignación se debe realizar solo con el identificador sin paréntesis para evitar que se evalúe.
  - De igual forma se pueden pasar una función como valor de un argumento de otra función.

```
function calcula(fn, a, b) {
    return fn(a, b);
}

rslt = calcula(suma, 2, 2);
```

© JMA 2015. All rights reserved

# Funciones anónimas

- La asignación permite definir una función con una expresión en la que el nombre de la función es opcional, se conocen como funciones anónimas.

```
var miFuncion = function(a, b) { return a + b; }
```

- De igual forma, una función puede devolver otra función:
- ```
return function() { return a + b; }
```
- Una función anónima autoejecutable consiste en crear una expresión de función e inmediatamente ejecutarla. Es muy útil para casos en que no se desea intervenir espacios de nombres globales, debido a que ninguna variable declarada dentro de la función es visible desde afuera.

```
(function(){
    var local= 'Valor interno';
})();
```

© JMA 2015. All rights reserved

# OBJETOS

© JMA 2015. All rights reserved

## Introducción

- El JavaScript utiliza objetos pero no es un lenguaje orientado a objetos, esta orientado a prototipos.
- No soporta clases, herencia, sobrecarga, ...
- Técnicamente, un objeto de JavaScript es un array asociativo formado por las propiedades y los métodos del objeto.
- Un array asociativo es aquel en el que cada elemento no está asociado a su posición numérica dentro del array, sino que está asociado a otro valor específico, pares nombre/valor. En los arrays normales, los valores se asocian a índices que siempre son numéricos, mientras que en los arrays asociativos se asocian a claves que siempre son cadenas de texto.
- Las propiedades son variables globales del objeto y los métodos son variables globales del objeto pero de tipo Function.
- Los miembros son accesibles externamente mediante la notación punto (objeto.miembro) o mediante la notación array (objeto["miembro"]).
- Los miembros son accesibles internamente mediante la notación punto y la palabra clave this (this.miembro).

© JMA 2015. All rights reserved

# Creación de Objetos

- Implementación directa

```
var elObjeto = new Object();
elObjeto.id = "99";
elObjeto.nombre = "Objeto de prueba";
elObjeto.muestraId = function() {
    alert("El ID del objeto es " + this.id);
}
elObjeto.ponNombre = function(nom) {
    this.nombre=nom.toUpperCase();
}
```
- Notación JSON

```
var elObjeto = {
    id : "99",
    nombre : "Objeto de prueba",
    muestraId : function() {
        alert("El ID del objeto es " + this.id);
    },
    ponNombre : function(nom) {
        this.nombre=nom.toUpperCase();
    }
}
```

© JMA 2015. All rights reserved

# Funciones constructoras

- Al contrario que en los lenguajes orientados a objetos, en JavaScript no existe el concepto de constructor. Por lo tanto, al definir un objeto no se incluyen uno o varios constructores. En realidad, JavaScript emula el funcionamiento de los constructores mediante el uso de funciones constructoras.

```
function MiClase(elId, elNombre) {
    this.id = elId;
    this.nombre = elNombre;
    this.muestraId = function() {
        alert("El ID del objeto es " + this.id);
    }
    this.ponNombre = function(nom) {
        this.nombre=nom.toUpperCase();
    }
}
var elObjeto = new MiClase("99", "Objeto de prueba");
```

© JMA 2015. All rights reserved

# Prototype

- Cada vez que se instancia un objeto con la función constructora, se definen tantas nuevas funciones como métodos incluya la función constructora.
- La penalización en el rendimiento y el consumo excesivo de recursos de esta técnica puede suponer un inconveniente en las aplicaciones profesionales realizadas con JavaScript.
- Todos los objetos de JavaScript incluyen una referencia interna a otro objeto llamado prototype o "prototipo". Cualquier propiedad o método que contenga el objeto prototipo, está presente de forma automática en el objeto original.
- Es como si cualquier objeto de JavaScript heredara de forma automática todas las propiedades y métodos de otro objeto llamado prototype. Cada tipo de objeto diferente hereda de un objeto prototype diferente.
- Dado que el prototype es el molde con el que se fabrica cada objeto de ese tipo. Si se modifica el molde o se le añaden nuevas características, todos los objetos fabricados con ese molde tendrán esas características.

© JMA 2015. All rights reserved

# Prototype

- En el prototype de un objeto sólo se deben añadir aquellos elementos comunes para todos los objetos. Normalmente se añaden los métodos y las constantes (propiedades cuyo valor no varía durante la ejecución de la aplicación). Las propiedades del objeto permanecen en la función constructora para que cada objeto diferente pueda tener un valor distinto en esas propiedades.

```
function MiClase(elId, elNombre) {
    this.id = elId;
    this.nombre = elNombre;
}
MiClase.prototype.muestraId = function() {
    alert("El ID del objeto es " + this.id);
}
MiClase.prototype.ponNombre = function(nom) {
    this.nombre=nom.toUpperCase();
}
```

© JMA 2015. All rights reserved

## Prototype

- La propiedad `prototype` también permite añadir y/o modificar las propiedades y métodos de los objetos predefinidos por JavaScript.
- Por lo tanto, es posible redefinir el comportamiento habitual de algunos métodos de los objetos nativos de JavaScript.
- Además, se pueden añadir propiedades o métodos completamente nuevos.
- Existen librerías de JavaScript formadas por un conjunto de utilidades que facilitan la programación de las aplicaciones y una de sus características habituales es el uso de la propiedad `prototype` para mejorar las funcionalidades básicas de JavaScript.

© JMA 2015. All rights reserved

## Palabra clave `this`

- En JavaScript, así como en la mayoría de los lenguajes de programación orientados a objetos, `this` es una palabra clave especial que hace referencia al objeto en donde el método está siendo invocado.
- El valor de `this` se determina utilizando una serie de simples reglas:
  1. Si la función es invocada utilizando `Function.call` o `Function.apply`, `this` tendrá el valor del primer argumento pasado al método. Si el argumento es nulo (`null`) o indefinido (`undefined`), `this` hará referencia al objeto global (el objeto `window`);
  2. Si la función a invocar es creada utilizando `Function.bind`, `this` será el primer argumento que es pasado a la función en el momento en que se la crea;
  3. Si la función es invocada como un método de un objeto, `this` referenciará a dicho objeto;
  4. De lo contrario, si la función es invocada como una función independiente, no unida a algún objeto, `this` referenciará al objeto global.

© JMA 2015. All rights reserved

## Métodos apply() y call()

- Los métodos del objeto Funciton apply() y call() permiten ejecutar una función como si fuera un método de otro objeto. La única diferencia entre los dos métodos es la forma en la que se pasan los argumentos a la función.
- El primer parámetro del método call() es el objeto sobre el que se va a ejecutar la función. El resto de parámetros del método call() son los parámetros que se pasan a la función.

```
function miFuncion(x) {
    return this.numero + x;
}
var elObjeto = new Object();
elObjeto.numero = 5;

var resultado = miFuncion.call(elObjeto, 4);
alert(resultado);
```

- El método apply() es idéntico al método call(), salvo que en este caso los parámetros se pasan como un array:

```
var resultado = miFuncion.apply(elObjeto, [4]);
alert(resultado);
```

© JMA 2015. All rights reserved

<http://ecma-international.org/ecma-262/5.1/#sec-15>

## STANDARD BUILT-IN ECMASCIPT OBJECTS

© JMA 2015. All rights reserved

# Objeto Global

- El objeto Global es parte del entorno léxico del programa en ejecución.
- Se encarga de encapsular los elementos no sintácticos del lenguaje:
  - contantes globales (propiedades valor)
    - NaN, Infinity, undefined
  - funciones globales (propiedades función)
    - eval(x), parseInt(string , radix), parseFloat(string), isNaN(number), isFinite(number)
    - decodeURI(uri), decodeURIComponent(uri), encodeURI(uri), encodeURIComponent(uri)
  - objetos globales (propiedades referencia)
    - Math, JSON
  - tipos predefinidos (propiedades funciones constructoras).
    - Object, Function, Array, String, Boolean, Number, Date, RegExp
    - Error, EvalError, RangeError, ReferenceError, SyntaxError, TypeError, URIError

© JMA 2015. All rights reserved

# Object

- **Object.create (Función):** Crea un objeto que tiene un prototipo especificado y contiene opcionalmente propiedades especificadas.
- **Object.defineProperties (Función):** Agrega una o varias propiedades a un objeto, o modifica atributos de propiedades existentes.
- **Object.defineProperty (Función):** Agrega una propiedad a un objeto o modifica atributos de una propiedad existente.
- **Object.seal (Función):** Impide la modificación de atributos de propiedades existentes e impide agregar nuevas propiedades.
- **Object.freeze (Función):** Impide la modificación de atributos y valores de propiedad existentes e impide agregar nuevas propiedades.
- **Object.preventExtensions (Función):** Impide la adición de nuevas propiedades a un objeto.
- **Object.isExtensible (Función):** Devuelve un valor que indica si se pueden agregar nuevas propiedades a un objeto.
- **Object.isFrozen (Función):** Devuelve true si no se pueden modificar atributos y valores de propiedad existentes en un objeto y no se pueden agregar nuevas propiedades al objeto.

© JMA 2015. All rights reserved

# Object

- **Object.isSealed** (Función): Devuelve true si no se pueden modificar atributos de propiedad existentes en un objeto y no se pueden agregar nuevas propiedades al objeto.
- **Object.keys** (Función): Devuelve los nombres de las propiedades y los métodos enumerables de un objeto.
- **Object.getPrototypeOf** (Función): Devuelve el prototipo de un objeto.
- **Object.getOwnPropertyDescriptor** (Función): Devuelve la definición de una propiedad de datos o de una propiedad de descriptor de acceso.
- **Object.getOwnPropertyNames** (Función): Devuelve los nombres de las propiedades y métodos de un objeto.
- **prototype** (Propiedad): Devuelve una referencia al prototipo correspondiente a una clase de objetos.
- **toLocaleString** (Método): Devuelve un objeto convertido en una cadena basándose en la configuración regional actual.
- **toString** (Método): Devuelve una representación en forma de cadena de un objeto.
- **valueOf** (Método): Devuelve el valor primitivo del objeto especificado.

© JMA 2015. All rights reserved

# Function

- **arguments** (Propiedad): Obtiene los argumentos del objeto Function que se está ejecutando actualmente.
- **caller** (Propiedad): Obtiene la función invocada por la función actual.
- **length** (Propiedad): Obtiene el número de argumentos definidos para una función.
- **apply** (Método): Llama a la función, sustituyendo el objeto especificado por el valor de this de la función, y la matriz especificada por los argumentos de la función.
- **bind** (Método): Para una función determinada, crea una función enlazada con el mismo cuerpo que la función original. En la función enlazada, el objeto this se resuelve en el objeto pasado. La función enlazada tiene los parámetros iniciales especificados.
- **call** (Método): Llama a un método de un objeto y sustituye el objeto actual por otro objeto.

© JMA 2015. All rights reserved

# Error

- **name** (Propiedad): Devuelve el nombre de un error.
- **message** (Propiedad): Devuelve una cadena con un mensaje de error.
  - *RangeError*: Este error se produce cuando se proporciona a una función un argumento que ha superado su intervalo permitido.
  - *ReferenceError*: Este error tiene lugar cuando se detecta una referencia no válida.
  - *SyntaxError*: Este error se produce cuando se analiza el texto de origen y su sintaxis no es correcta.
  - *TypeError*: Este error se produce cuando el tipo real de un operando no coincide con el tipo esperado.
  - *URIError*: Este error tiene lugar cuando se detecta un identificador uniforme de recursos (URI) no válido.

© JMA 2015. All rights reserved

# Array

- **length** (Propiedad): Devuelve un valor entero que supera en uno al elemento mayor definido en una matriz.
- **Array.isArray** (Función): Devuelve un valor de tipo booleano que indica si un objeto es una matriz.
- **concat** (Método): Devuelve una matriz nueva que se compone de una combinación de dos matrices.
- **join** (Método): Devuelve un objeto String formado por todos los elementos de una matriz concatenados.
- **pop** (Método): Quita el último elemento de una matriz y lo devuelve.
- **push** (Método): Anexa nuevos elementos a una matriz y devuelve la nueva longitud de la matriz.
- **reverse** (Método): Devuelve un objeto Array con los elementos invertidos.
- **shift** (Método): Quita el primer elemento de una matriz y lo devuelve.
- **slice** (Método): Devuelve una sección de una matriz.
- **sort** (Método): Devuelve un objeto Array con los elementos ordenados.

© JMA 2015. All rights reserved

# Array

- **splice** (Método): Quita elementos de una matriz, inserta nuevos elementos en su lugar si procede y devuelve los elementos eliminados.
- **unshift** (Método): Inserta nuevos elementos al principio de una matriz.
- **indexOf** (Método): Devuelve el índice de la primera aparición de un valor de una matriz.
- **lastIndexOf** (Método): Devuelve el índice de la última aparición de un valor especificado de una matriz.
- **every** (Método): Comprueba si una función de devolución de llamada definida devuelve true para todos los elementos de una matriz.
- **some** (Método): Comprueba si una función de devolución de llamada definida devuelve true para cualquier elemento de una matriz.
- **forEach** (Método): Llama a una función de devolución de llamada definida para cada elemento de una matriz.
- **map** (Método): Llama a una función de devolución de llamada definida para cada elemento de una matriz y devuelve una matriz que contiene los resultados.

© JMA 2015. All rights reserved

# Array

- **filter** (Método): Llama a una función de devolución de llamada definida para cada elemento de una matriz y devuelve una matriz de aquellos valores para los que esa función devuelve true.
- **reduce** (Método): Acumula un solo resultado mediante la llamada a una función de devolución de llamada definida para todos los elementos de una matriz. El valor devuelto de la función de devolución de llamada es el resultado acumulado, y se proporciona como un argumento en la siguiente llamada a dicha función.
- **reduceRight** (Método): Acumula un solo resultado mediante la llamada a una función de devolución de llamada definida para todos los elementos de una matriz, en orden descendente. El valor devuelto de la función de devolución de llamada es el resultado acumulado, y se proporciona como un argumento en la siguiente llamada a dicha función.
- **toLocaleString** (Método): Devuelve una cadena con la configuración regional actual.
- **toString** (Método): Devuelve una representación de cadena de una matriz.
- **valueOf** (Método): Obtiene una referencia a la matriz.

© JMA 2015. All rights reserved

# Date

- **Date.now** (Función): Devuelve el número de milisegundos que hay entre el 1 de enero de 1970 y la fecha y hora actuales.
- **Date.parse** (Función): Analiza una cadena que contiene una fecha y devuelve el número de milisegundos transcurridos entre esa fecha y la medianoche del 1 de enero de 1970.
- **Date.UTC** (Función): Devuelve el número de milisegundos transcurrido entre la medianoche del 1 de enero de 1970 en el horario universal coordinado (UTC) (o GMT) y la fecha proporcionada.
- **toString** (Método): Devuelve una representación en forma de cadena de un objeto.
- **toDateString** (Método): Devuelve una fecha como un valor de cadena.
- **toTimeString** (Método): Devuelve una hora como un valor de cadena.
- **toLocaleString** (Método): Devuelve un objeto convertido en cadena usando la configuración regional actual.
- **toLocaleDateString** (Método): Devuelve una fecha como un valor de cadena apropiado para la configuración regional actual del entorno host.
- **toLocaleTimeString** (Método): Devuelve una hora como un valor de cadena apropiado para la configuración regional actual del entorno host.
- **toISOString** (método): Devuelve una fecha como un valor alfanumérico en formato ISO.
- **toJSON** (método): Se utiliza para transformar datos de un tipo de objeto antes de la serialización JSON.
- **valueOf** (Método): Devuelve el valor primitivo del objeto especificado.

© JMA 2015. All rights reserved

# Date

- **getTime** (Método): Devuelve el valor de tiempo en un objeto Date en milisegundos desde la medianoche del 1 de enero de 1970.
- **getFullYear, getUTCFullYear** (Método): Devuelve el valor de año usando la hora local o UTC.
- **getMonth, getUTCMonth** (Método): Devuelve el valor de mes usando la hora local o UTC.
- **getDate, getUTCDate** (Método): Devuelve el valor de día del mes usando la hora local o UTC.
- **getDay, getUTCDay** (Método): Devuelve el valor de día de la semana usando la hora local o UTC.
- **getHours, getUTCHours** (Método): Devuelve el valor de horas usando la hora local o UTC.
- **getMinutes, getUTCMinutes** (Método): Devuelve el valor de minutos usando la hora local o UTC.
- **getSeconds, getUTCSeconds** (Método): Devuelve el valor de segundos usando la hora local o UTC.
- **getMilliseconds, getUTCMilliseconds** (Método): Devuelve el valor de milisegundos usando la hora local o UTC.
- **getTimezoneOffset** (Método): Devuelve la diferencia en minutos entre la hora del equipo host y la hora universal coordinada (UTC).

© JMA 2015. All rights reserved

# Date

- **setTime**  (Método): Establece el valor de fecha y hora en el objeto Date.
- **setMilliseconds**  (Método): Establece el valor de milisegundos usando la hora local.
- **setUTCMilliseconds**  (Método): Establece el valor de milisegundos usando la hora UTC.
- **setSeconds**  (Método): Establece el valor de segundos usando la hora local.
- **setUTCSeconds**  (Método): Establece el valor de segundos usando la hora UTC.
- **setMinutes**  (Método): Establece el valor de minutos usando la hora local.
- **setUTCMinutes**  (Método): Establece el valor de minutos usando la hora UTC.
- **setHours**  (Método): Establece el valor de horas usando la hora local.
- **setUTCHours**  (Método): Establece el valor de horas usando la hora UTC.
- **setDate**  (Método): Establece el día del mes numérico usando la hora local.
- **setUTCDate**  (Método): Establece el día numérico del mes usando la hora UTC.
- **setMonth**  (Método): Establece el valor de mes usando la hora local.
- **setUTCMonth**  (Método): Establece el valor de mes usando la hora UTC.
- **setFullYear**  (Método): Establece el valor de año usando la hora local.
- **setUTCFullYear**  (Método): Establece el valor de año usando la hora UTC.

© JMA 2015. All rights reserved

# Number

- **Number.MAX\_VALUE** : El número más grande que se puede representar en JavaScript. Igual a aproximadamente 1,79E+308.
- **Number.MIN\_VALUE** : El número más cercano a cero que se puede representar en JavaScript. Igual a aproximadamente 5,00E-324.
- **Number.NaN** : Un valor que no es un número.
- **Number.NEGATIVE\_INFINITY** : Un valor inferior al número negativo más grande que se puede representar en JavaScript.
- **Number.POSITIVE\_INFINITY** : Un valor superior al número más grande que se puede representar en JavaScript.
- **toExponential**  (Método): Devuelve una cadena que contiene un número representado en notación exponencial.
- **toFixed**  (Método): Devuelve una cadena que representa un número en notación de punto fijo.
- **toLocaleString**  (Método): Devuelve un objeto convertido en una cadena basándose en la configuración regional actual.
- **toPrecision**  (Método): Devuelve una cadena que contiene un número representado en notación exponencial o de punto fijo y que tiene un número especificado de dígitos.
- **toString**  (Método): Devuelve una representación en forma de cadena de un objeto.
- **valueOf**  (Método): Devuelve el valor primitivo del objeto especificado.

© JMA 2015. All rights reserved

# String

- **length** (Propiedad): Devuelve la longitud de un objeto String.
- **String.fromCharCode** (Función): Devuelve una cadena a partir de varios valores de caracteres Unicode.
- **charAt** (Método): Devuelve el carácter que se encuentra en el índice especificado.
- **charCodeAt** (Método): Devuelve la codificación Unicode del carácter que se especifique.
- **concat** (Método): Devuelve una cadena que contiene la concatenación de las dos cadenas proporcionadas.
- **indexOf** (Método): Devuelve la posición del carácter donde tiene lugar la primera repetición de una subcadena dentro de una cadena.
- **lastIndexOf** (Método): Devuelve la última repetición de una subcadena dentro de una cadena.
- **localeCompare** (Método): Devuelve un valor que indica si dos cadenas son equivalentes en la configuración regional actual.
- **match** (Método): Busca una cadena mediante un objeto Regular Expression proporcionado y devuelve los resultados como una matriz.
- **replace** (Método): Usa una expresión regular para reemplazar texto en una cadena y devuelve el resultado.
- **search** (Método): Devuelve la posición de la primera coincidencia de subcadena en una búsqueda de expresión regular.
- **slice** (Método): Devuelve una sección de una cadena.

© JMA 2015. All rights reserved

# String

- **split** (Método): Devuelve la matriz de cadenas resultante de la separación de una cadena en subcadenas.
- **substring** (Método): Devuelve la subcadena en la ubicación especificada dentro de un objeto String.
- **toLocaleLowerCase** (Método): Devuelve una cadena en la que todos los caracteres alfabéticos se convierten a minúsculas, según la configuración regional actual del entorno de host.
- **toLocaleString** (Método): Devuelve un objeto convertido en cadena usando la configuración regional actual.
- **toLocaleUpperCase** (Método): Devuelve una cadena en la que todos los caracteres alfabéticos se convierten a mayúsculas, según la configuración regional actual del entorno de host.
- **toLowerCase** (Método): Devuelve una cadena en la que todos los caracteres alfabéticos se convierten a minúsculas.
- **toString** (Método): Devuelve la cadena.
- **toUpperCase** (Método): Devuelve una cadena en la que todos los caracteres alfabéticos se convierten a mayúsculas.
- **trim** (Método): Devuelve una cadena donde se han quitado los caracteres de espacio en blanco iniciales y finales y los caracteres de terminador de línea.
- **valueOf** (Método): Devuelve la cadena.

© JMA 2015. All rights reserved

# JSON

- **JSON.parse** (Función): Convierte una cadena de la notación de objetos de JavaScript (JSON) en un objeto.
- **JSON.stringify** (Función): Convierte un valor de JavaScript en una cadena de la notación de objetos JavaScript (JSON).

© JMA 2015. All rights reserved

# Math

- **Math.E**: Constante matemática e. Es el número de Euler, base de los logaritmos naturales.
- **Math.LN2**: Logaritmo natural de 2.
- **Math.LN10**: Logaritmo natural de 10.
- **Math.LOG2E**: Logaritmo de base 2 de e.
- **Math.LOG10E**: Logaritmo de base 10 de e.
- **Math.PI**: Pi. Es la proporción entre la circunferencia de un círculo y su diámetro.
- **Math.SQRT1\_2**: Raíz cuadrada de 0,5, o, de forma equivalente, uno dividido por la raíz cuadrada de 2.
- **Math.SQRT2**: Raíz cuadrada de 2.
- **Math.abs** (Función): Devuelve el valor absoluto de un número.
- **Math.acos** (Función): Devuelve el arco coseno de un número.
- **Math.asin** (Función): Devuelve el arcoseno de un número.
- **Math.atan** (Función): Devuelve el arco tangente de un número.
- **Math.atan2** (Función): Devuelve el ángulo, en radianes, desde el eje X a un punto representado por las coordenadas x e y proporcionadas.
- **Math.cos** (Función): Devuelve el coseno de un número.

© JMA 2015. All rights reserved

# Math

- **Math.ceil** (Función): Devuelve el entero más pequeño que sea mayor o igual que la expresión numérica proporcionada.
- **Math.exp** (Función): Devuelve  $e$  (base de los logaritmos naturales) elevado a una potencia.
- **Math.floor** (Función): Devuelve el entero más grande que sea menor o igual que la expresión numérica proporcionada.
- **Math.log** (Función): Devuelve el logaritmo natural de un número.
- **Math.max** (Función): Devuelve la mayor de dos expresiones numéricas proporcionadas.
- **Math.min** (Función): Devuelve el menor de dos números proporcionados.
- **Math.pow** (Función): Devuelve el valor de una expresión base elevada a una potencia especificada.
- **Math.random** (Función): Devuelve un número pseudoaleatorio entre 0 y 1.
- **Math.round** (Función): Devuelve una expresión numérica especificada redondeada al entero más cercano.
- **Math.sin** (Función): Devuelve el seno de un número.
- **Math.sqrt** (Función): Devuelve la raíz cuadrada de un número.
- **Math.tan** (Función): Devuelve la tangente de un número.

© JMA 2015. All rights reserved

# RegEx

- Crea un objeto 'expresión regular' para encontrar texto de acuerdo a un patrón.
- Las cadenas delimitadas por / generan un objeto RegEx con el patrón contenido en la cadena:  

```
var re = new RegExp("^\\d{1,8}[A-Z]$");
var re = /^\\d{1,8}[A-Z]$/;
```
- Permiten los modificadores:
  - g: búsqueda global (no para tras la primera coincidencia)
  - i: ignorar mayúsculas o minúsculas
  - m: tratar caracteres de inicio y fin (^ y \$) como múltiples líneas de texto
- Las expresiones regulares se utilizan con los métodos de RegExp (test y exec) y con los métodos de String (match, replace, search y split).

© JMA 2015. All rights reserved

# RegEx: Escapes de carácter

| Carácter de escape | Descripción                                                                                                                            | Modelo                     | Coincidencias                                            |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------|----------------------------|----------------------------------------------------------|
| \a                 | Coincide con un carácter de campana, \u0007.                                                                                           | \a                         | "\u0007" en "Error!" + "\u0007"                          |
| \b                 | En una clase de caracteres, coincide con un retroceso, \u0008.                                                                         | \b\b\b\b                   | "(b\b)b\b" en "(b\b)b\b"                                 |
| \t                 | Coincide con una tabulación, \u0009.                                                                                                   | (\w+)\t                    | "artículo1\t", "artículo2\t" en "artículo1\tartículo2\t" |
| \r                 | Coincide con un retorno de carro, \u000D. (\r no es equivalente al carácter de nueva línea, \n.)                                       | \r\n(\w+)                  | "\r\nEstas" en "\r\nEstas son\rndos líneas."             |
| \v                 | Coincide con una tabulación vertical, \u000B.                                                                                          | \v\v                       | "\v\v\v" en "\v\v\v"                                     |
| \f                 | Coincide con un avance de página, \u000C.                                                                                              | \f\f\f                     | "\f\f\f" en "\f\f\f"                                     |
| \n                 | Coincide con una nueva línea, \u000A.                                                                                                  | \r\n(\w+)                  | "\r\nEstas" en "\r\nEstas son\rndos líneas."             |
| \e                 | Coincide con un escape, \u001B.                                                                                                        | \e                         | "\x001B" en "\x001B"                                     |
| \nnn               | Usa la representación octal para especificar un carácter ( nn consta de tres dígitos como máximo).                                     | \w\040\w                   | "a\b", "c\d" en "a bc d"                                 |
| \nnn               | Usa la representación hexadecimal para especificar un carácter ( nn consta de exactamente dos dígitos).                                | \w\x20\w                   | "a b", "c d" en "a bc d"                                 |
| \cX                | Se corresponde con el carácter de control ASCII especificado por X, donde X es la letra del carácter de control.                       | \cC                        | "\x0003" en "\x0003" (Ctrl-C)                            |
| \unnn              | Se corresponde con un carácter Unicode usando la representación hexadecimal (exactamente cuatro dígitos, representados aquí por nnnn). | \w\u0020\w                 | "a b", "c d" en "a bc d"                                 |
| \                  | Cuando va seguido de un carácter sin significado, coincide con ese carácter                                                            | (d+(\ +x\*) d+d+\ \-\ +\ + | "2+2" y "3*9" en "(2>2) * 3*9"                           |

© JMA 2015. All rights reserved

# RegEx: Clases de carácter

| Clase de carácter   | Descripción                                                                                                                                                                                 | Modelo               | Coincidencias                                   |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-------------------------------------------------|
| [grupo_caracteres]  | Coincide con cualquier carácter único de grupo_caracteres. De forma predeterminada, la coincidencia distingue entre mayúsculas y minúsculas.                                                | [aeiou]              | "a" en "casa"<br>"a", "e" en "ave"              |
| [^grupo_caracteres] | Negación: coincide con cualquier carácter individual que no esté en grupo_caracteres. De forma predeterminada, los caracteres de grupo_caracteres distinguen entre mayúsculas y minúsculas. | [^aei]               | "r", "n", "o" en "reino"                        |
| [primero-último]    | Intervalo de caracteres: coincide con cualquier carácter individual en el intervalo de primero a último.                                                                                    | [A-Z]                | "A", "B" en "AB123"                             |
| .                   | Carácter comodín: coincide con cualquier carácter excepto con \n.                                                                                                                           | a.e                  | "ave" en "llave"<br>"ate" en "yate"             |
| \p{name}            | Coincide con cualquier carácter único que pertenezca a la categoría general Unicode o al bloque con nombre especificado por nombre.                                                         | \p{Lu}\p{LsCyrillic} | "C", "L" en "City Lights"<br>"Д", "Ж" en "Джем" |
| \P{name}            | Coincide con cualquier carácter único que no pertenezca a la categoría general Unicode o al bloque con nombre especificado por nombre.                                                      | \P{Lu}\P{LsCyrillic} | "I", "t", "у" en "City"<br>"е", "м" en "Джем"   |
| \w                  | Coincide con cualquier carácter de una palabra.                                                                                                                                             | \w                   | "I", "D", "A", "1", "3" en "ID A1.3"            |
| \W                  | Coincide con cualquier carácter que no pertenezca a una palabra.                                                                                                                            | \W                   | " ", " " en "ID A1.3"                           |
| \s                  | Coincide con cualquier carácter que sea un espacio en blanco.                                                                                                                               | \w\s                 | "D" en "ID A1.3"                                |
| \S                  | Coincide con cualquier carácter que no sea un espacio en blanco.                                                                                                                            | \s\S                 | "_" en "int __ctr"                              |
| \d                  | Coincide con cualquier dígito decimal.                                                                                                                                                      | \d                   | "4" en "4 = IV"                                 |
| \D                  | Coincide con cualquier dígito que no sea decimal.                                                                                                                                           | \D                   | " ", "-", "+", "/", "/" en "4 = IV"             |

© JMA 2015. All rights reserved

# RegEx: Delimitadores

| Aserción | Descripción                                                                                                           | Modelo         | Coincidencias                                                    |
|----------|-----------------------------------------------------------------------------------------------------------------------|----------------|------------------------------------------------------------------|
| ^        | La coincidencia debe comenzar al principio de la cadena o de la línea.                                                | ^\d{3}         | "901-" en<br>"901-333-"                                          |
| \$       | La coincidencia se debe producir al final de la cadena o antes de \n al final de la línea o de la cadena.             | -\d{3}\\$      | ".333" en<br>"-901-333"                                          |
| \A       | La coincidencia se debe producir al principio de la cadena.                                                           | \A\d{3}        | "901-" en<br>"901-333-"                                          |
| \z       | La coincidencia se debe producir al final de la cadena o antes de \n al final de la cadena.                           | -\d{3}\Z       | ".333" en<br>"-901-333"                                          |
| \z       | La coincidencia se debe producir al final de la cadena.                                                               | -\d{3}\z       | ".333" en<br>"-901-333"                                          |
| \G       | La coincidencia se debe producir en el punto en el que finalizó la coincidencia anterior.                             | \G\(\d\)       | "(1)" , "(3)" , "(5" en<br>"(1)(3)(5)[7](9)"                     |
| \b       | La coincidencia se debe producir en un límite entre un carácter \w (alfanumérico) y un carácter \W (no alfanumérico). | \b\w+\\$ \w+\b | "ellos ellos" en "ellos tema<br>ellos ellos"                     |
| \B       | La coincidencia no se debe producir en un límite \b.                                                                  | \Bend\w*\b     | "fin" , "final" en "finalizar<br>finalista finalizador finalizó" |

© JMA 2015. All rights reserved

# RegEx: Construcciones de agrupamiento

| Construcción de agrupamiento     | Descripción                                                                     | Modelo                                                                  | Coincidencias                                                                    |
|----------------------------------|---------------------------------------------------------------------------------|-------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| (subexpresión)                   | Captura la subexpresión coincidente y le asigna un número ordinal de base cero. | (\w)\1                                                                  | "aa" en "aarón"                                                                  |
| (?:name>subexpresión)            | Captura la subexpresión coincidente en un grupo con nombre.                     | (?:double>\w)\k<double>                                                 | "aa" en "aarón"                                                                  |
| (?:nombre1-nombre2>subexpresión) | Define una definición de grupo de equilibrio.                                   | ((?:Open'\ [^\\\] \\')*+(?:Close-Open'\ [^\\\] \\')*+(?:Open(?:?!) \$)) | "((1-3)*(3-1))" en "3+2*((1-3)*(3-1))"                                           |
| (?:subexpresión)                 | Define un grupo sin captura.                                                    | Write(?:Line)?                                                          | "WriteLine" en "Console.WriteLine()"                                             |
| (?:imnsx-imnsx:subexpresión)     | Aplica o deshabilita las opciones especificadas dentro de subexpresión.         | A \d{2}?:\w+\b                                                          | "A12xI" , "A12XL" en "A12xI A12XL a12xI"                                         |
| (?:subexpresión)                 | Aserción de búsqueda anticipada positiva de ancho cero.                         | \w+(?=.)                                                                | "es" , "corría" y "hermoso" en "Él es. El perro corría.<br>El sol está hermoso." |
| (?:!subexpresión)                | Aserción de búsqueda anticipada negativa de ancho cero.                         | \b(?:!un)\w+\b                                                          | "seguro" , "usado" en "aseguro seguro unidad usado"                              |
| (?:<subexpresión>)               | Aserción de búsqueda tardía positiva de ancho cero.                             | (?:<19 \d{2}\b                                                          | "99" , "50" , "05" en "1851 1999 1950 1905 2003"                                 |
| (?:<!subexpresión>)              | Aserción de búsqueda tardía negativa de ancho cero.                             | (?:<19 \d{2}\b                                                          | "51" , "03" en "1851 1999 1950 1905 2003"                                        |
| (?:subexpresión)                 | Subexpresión sin retroceso (o "expansiva").                                     | [13579](?:A+B+)                                                         | "1ABB" , "3ABB" y "5AB" en "1ABB 3ABC 5AB SAC"                                   |

## RegEx: Quantificadores

| Cuantificador | Descripción                                                                                        | Modelo                                                           | Coincidencias                                                    |
|---------------|----------------------------------------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|
| *             | Coincide con el elemento anterior cero o más veces.                                                | \d*\.\d                                                          | ".0", "19.9", "219.9"                                            |
| +             | Coincide con el elemento anterior una o más veces.                                                 | "be+"                                                            | "caí" en "caída", "be" en "bebé"                                 |
| ?             | Coincide con el elemento anterior cero veces o una vez.                                            | "rai?n"                                                          | "ata", "racilia"                                                 |
| {n}           | Coincide con el elemento anterior exactamente n veces.                                             | ".043" en "1,043.6", ".876", ".543", y ".210" en "9,876,543,210" |                                                                  |
| {n,}          | Coincide con el elemento anterior al menos n veces.                                                | "166", "29", "1930"                                              |                                                                  |
| {n,m}         | Coincide con el elemento anterior al menos n veces, pero no más de m veces.                        | \d{3,5}"                                                         | "166", "17668", "19302" en "193024"                              |
| *?            | Coincide con el elemento anterior cero o más veces, pero el menor número de veces que sea posible. | \d*?.\d                                                          | ".0", "19.9", "219.9"                                            |
| +?            | Coincide con el elemento anterior una o más veces, pero el menor número de veces que sea posible.  | "be+?"                                                           | "caí" en "caída", "be" en "bebé"                                 |
| ??            | Coincide con el elemento anterior cero o una vez, pero el menor número de veces que sea posible.   | "rai??n"                                                         | "ata", "racilia"                                                 |
| {n}?          | Coincide con el elemento anterior exactamente n veces.                                             | \d{3}?"                                                          | ".043" en "1,043.6", ".876", ".543", y ".210" en "9,876,543,210" |
| {n,}?         | Coincide con el elemento anterior al menos n veces, pero el menor número de veces posible.         | \d{2,}?"                                                         | "166", "29", "1930"                                              |
| {n,m}?        | Coincide con el elemento anterior entre n y m veces, pero el menor número de veces posible.        | \d{3,5}?"                                                        | "166", "17668", "19302" en "193024"                              |

© JMA 2015. All rights reserved

## RegEx: Construcciones de alternancia

| Construcciones de alternancia | Descripción                                                                                                                                         | Modelo                                | Coincidencias                                                   |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|-----------------------------------------------------------------|
|                               | Coincide con cualquier elemento separado por el carácter de barra vertical ( ).                                                                     | th(e is at)                           | "el", "este" en "este es el día."                               |
| (?expresión)sí no)            | Coincide con sí si expresión coincide; de lo contrario, coincide con la parte opcional no. expresión se interpreta como una aserción de ancho cero. | (?A)A\d{2}\b \b\d{3}\b                | "A10", "910" en "A10 C103 910"                                  |
| (?name)sí no)                 | Coincide con sí si la captura con nombre nombre tiene una coincidencia; de lo contrario, coincide con la parte opcional no.                         | (?<quoted>)?(? (quoted).+?" \\$ +\\$) | Dogs.jpg, "Yiska playing.jpg" en "Dogs.jpg "Yiska playing.jpg"" |

© JMA 2015. All rights reserved

## RegEx: Sustituciones

| Carácter | Descripción                                                                 | Modelo                             | Modelo de reemplazo | Cadena de entrada | Cadena de resultado |
|----------|-----------------------------------------------------------------------------|------------------------------------|---------------------|-------------------|---------------------|
| \$number | Sustituye la subcadena que coincide con el grupo número.                    | /b/(w+)(/s)(/w+)/b                 | \$3\$2\$1           | "one two"         | "two one"           |
| \${name} | Sustituye la subcadena que coincide con el grupo nombre.                    | \b(?<word1>\w+)(\s)(?<word2>\w+)\b | \${word2} \${word1} | "one two"         | "two one"           |
| \$\$     | Sustituye un "\$" literal.                                                  | \b(\d+)\\$?USD                     | \$\$1               | "103 USD"         | "\$103"             |
| \$&      | Sustituye una copia de toda la coincidencia.                                | (\\$*(\d*(\.\+\d+)?){1})           | **\$&               | "\$1.30"          | "**\$1.30**"        |
| \$`      | Sustituye todo el texto de la cadena de entrada delante de la coincidencia. | B+                                 | \$`                 | "AABBCC"          | "AAAACC"            |
| \$'      | Sustituye todo el texto de la cadena de entrada detrás de la coincidencia.  | B+                                 | \$'                 | "AABBCC"          | "AACCCC"            |
| \$+      | Sustituye el último grupo capturado.                                        | B+(C+)                             | \$+                 | "AABBCCDD"        | AACCCD              |
| \$_      | Sustituye toda la cadena de entrada.                                        | B+                                 | \$_                 | "AABBCC"          | "AAAABBCCCC"        |

© JMA 2015. All rights reserved

## DETECCIÓN Y CORRECCIÓN DE ERRORES

© JMA 2015. All rights reserved

# Depuración

- La mayoría de los navegadores modernos ofrecen utilidades de desarrollo y depuración.
- Cada depurador ofrece:
  - Un editor multi-linea para experimentar con JavaScript;
  - Un inspector para revisar el código generado en la página;
  - Un visualizador de red o recursos, para examinar las peticiones que se realizan.
- Suelen suministrar un objeto console con los siguientes métodos:
  - console.log() para enviar y registrar mensajes generales.
  - console.dir() para registrar un objeto y visualizar sus propiedades.
  - console.warn() para registrar mensajes de alerta.
  - console.error() para registrar mensajes de error.
- Existen otros métodos para utilizar desde la consola, pero estos pueden variar según el navegador. La consola además provee la posibilidad de establecer puntos de interrupción y observar expresiones en el código con el fin de facilitar su depuración.

© JMA 2015. All rights reserved

JavaScript 6: <http://www.ecma-international.org/ecma-262/6.0/>

## ECMASCRIPT 2015

© JMA 2015. All rights reserved

# Introducción

- En Junio de 2015 aparece la 6<sup>a</sup> edición del estándar ECMAScript (ES6), cuyo nombre oficial es ECMAScript 2015.
- Se está extendiendo progresivamente el soporte a ES6, pero aún queda trabajo por hacer.
- Las alternativas para empezar a utilizarlo son:
  - Transpiladores ("Transpiler": "Translator" y "Compiler"): Traducen o compilan un lenguaje de alto nivel a otro lenguaje de alto nivel, en este caso código ES6 a ES5. Los más conocidos y usados son Babel, TypeScript, Google Traceur, CoffeeScript.
  - Polyfill: Un trozo de código o un plugin que permite tener las nuevas funcionalidades de HTML5 o ES6 en aquellos navegadores que nativamente no lo soportan.

© JMA 2015. All rights reserved

## EcmaScript 6

- <http://www.ecma-international.org/ecma-262/6.0/>
- <http://es6-features.org/>
- <https://kangax.github.io/compat-table/es6/>
- <https://babeljs.io/>
- <https://github.com/google/traceur-compiler>
- <https://www.typescriptlang.org/>
- <https://github.com/zloirock/core-js>

© JMA 2015. All rights reserved

# Declaración de variables

- let: ámbito de bloque, solo accesible en el bloque donde está declarada.
 

```
(function() {
    if(true) {
        let x = "variable local";
    }
    console.log(x); // error, "x" definida dentro del "if"
})();
```
- const: constante, se asina valor al declararla y ya no puede cambiar de valor.
 

```
const PI = 3.15;
PI = 3.14159; // error, es de sólo-lectura
```
- Las constantes numérica ahora se pueden expresar en binario y octal.
 

```
0b111110111 === 503
0o767 === 503
```

© JMA 2015. All rights reserved

# Template Strings

- Interpolación: sustituye dentro de la cadena las variable por su valor:
 

```
let var1 = "JavaScript";
let var2 = "Templates";
console.log(`El ${var1} ya tiene ${var2}.`);
// El JavaScript ya tiene Templates.
```
- Las constantes cadenas pueden ser multilínea sin necesidad de concatenarlos con +.
 

```
let var1 = " El JavaScript
ya tiene
Templates ";
```
- Incorpora soporte extendido para el uso de Unicode en cadenas y expresiones regulares (las cadenas pueden contener caracteres en hebreo, árabe, cirílico, ...).

© JMA 2015. All rights reserved

## Destructuring

- Asignar (repartir) los valores de un objeto o array en varias variables:

```
var tab = ["hola", "adiós"];
var [a, b] = tab;
console.log(a); // "hola"
console.log(b); // "adiós"
[ b, a ] = [ a, b ]
```

```
var obj = { nombre: "Pepito", apellido: "Grillo" };
var { nombre, apellido } = obj;
console.log(nombre); // "Pepito"
```

© JMA 2015. All rights reserved

## Parámetros de funciones

- Valores por defecto: Se pueden definir valores por defecto a los parámetros en las funciones.  
`function(valor = "foo") {...};`
- Resto de los parámetros: Convierte una lista de parámetros en un array.  
`function f (x, y, ...a) {
 return (x + y) * a.length
}
f(1, 2, "hello", true, 7) === 9`
- Operador de propagación: Convierte un array o cadena en una lista de parámetros.  
`var str = "foo"
var chars = [ ...str ] // [ "f", "o", "o" ]`

© JMA 2015. All rights reserved

# Función Arrow

- Funciones anónimas.

```
data.forEach(elem => {
    console.log(elem);
    // ...
});
var fn = (num1, num2) => num1 + num2;
pairs = evens.map(v => ({ even: v, odd: v + 1 }));
fn = () => {console.log("Error");};
```
- this: dentro de una función Arrow hace referencia al contenedor y no al contexto de la propia función.

```
bar : function() {
    document.addEventListener("click", (e) => this.foo());
}
```
- equivale a (ES5):

```
bar : function() {
    document.addEventListener("click", function(e) {
        this.foo();
    }.bind(this));
}
```

© JMA 2015. All rights reserved

# Propiedades de objetos

## ES 2015

```
obj = { x, y }
obj = {
  foo (a, b) {
    ...
  },
  bar (x, y) {
    ...
  },
  *iter (x, y) {
    ...
  }
}
```

## Anteriormente:

```
obj = { x: x, y: y };
obj = {
  foo: function (a, b) {
    ...
  },
  bar: function (x, y) {
    ...
  },
  // iter: sin equivalencia
  ...
};

};
```

© JMA 2015. All rights reserved

# Clases

- Ahora JavaScript tendrá clases, muy parecidas las funciones constructoras de objetos que realizábamos en el estándar anterior, pero ahora bajo el paradigma de clases, con todo lo que eso conlleva, como por ejemplo, herencia.

```
class LibroTecnico extends Libro {
    constructor(tematica, paginas) {
        super(tematica, paginas);
        this.capitulos = [];
        this.precio = "";
        // ...
    }
    metodo() {
        // ...
    }
}
```

© JMA 2015. All rights reserved

# Static Members

```
class Rectangle extends Shape {
    ...
    static defaultRectangle () {
        return new Rectangle("default", 0, 0, 100, 100)
    }
}
class Circle extends Shape {
    ...
    static defaultCircle () {
        return new Circle("default", 0, 0, 100)
    }
}
var defRectangle = Rectangle.defaultRectangle()
var defCircle = Circle.defaultCircle()
```

© JMA 2015. All rights reserved

# Getter/Setter

```
class Rectangle {
    constructor (width, height) {
        this._width = width;
        this._height = height;
    }
    set width (width) { this._width = width; }
    get width () { return this._width; }
    set height (height) { this._height = height; }
    get height () { return this._height; }
    get area () { return this._width * this._height; }
}
var r = new Rectangle(50, 20)
if(r.area === 1000) r.width = 10;
```

© JMA 2015. All rights reserved

# mixin

- Soporte para la herencia de estilo mixin mediante la ampliación de las expresiones que producen objetos de función.

```
var aggregation = (baseClass, ...mixins) => {
    let base = class _Combined extends baseClass {
        constructor(...args) {
            super(...args);
            mixins.forEach((mixin) => {
                mixin.prototype.initializer.call(this);
            })
        }
    }
    let copyProps = (target, source) => {
        Object.getOwnPropertyNames(source)
            .concat(Object.getOwnPropertySymbols(source))
            .forEach((prop) => {
                if (prop.match(/^constructor|prototype|arguments|caller|name|bind|call|apply|toString|length$/))
                    return;
                Object.defineProperty(target, prop, Object.getOwnPropertyDescriptor(source, prop));
            })
    }
    mixins.forEach((mixin) => {
        copyProps(base.prototype, mixin.prototype);
        copyProps(base, mixin);
    })
    return base;
}
```

© JMA 2015. All rights reserved

# Módulos

- Estructura el código en módulos similares a los espacios de nombres
- Cada fichero se comporta como un módulo.
- Solo las partes marcadas como export pueden ser importadas en otros ficheros.
- Se puede llamar a las funciones desde los propios Scripts, sin tener que importarlos en el HTML, si usamos JavaScript en el navegador.

```
//File: lib/person.js
    export function hello(nombre) {
        return nombre;
    }
Y para importar en otro fichero:
//File: app.js
import { hello } from "lib/person";
var app = {
    foo: function() {
        hello("Carlos");
    }
}
export app;
```

© JMA 2015. All rights reserved

# Iteradores y Generadores

- El patrón Iterador permite trabajar con colecciones por medio de abstracciones de alto nivel
- Un Iterador es un objeto que sabe como acceder a los elementos de una secuencia, uno cada vez, mientras que mantiene la referencia a su posición actual en la secuencia.
- En ES2015 las colecciones (arrays, maps, sets) son objetos iteradores.
- Los generadores permiten la implementación del patrón Iterador.
- Los Generadores son funciones que pueden ser detenidas y reanudadas en otro momento.
- Estas pausas en realidad ceden la ejecución al resto del programa, es decir no bloquean la ejecución.
- Los Generadores devuelven (generan) un objeto "Iterator" (iterador)

© JMA 2015. All rights reserved

# Generadores

- Para crear una función generadora
 

```
function* myGenerator(){
  // ...
  yield value;
  // ...
}
```
- La instrucción `yield` devuelve el valor y queda a la espera de continuar cuando se solicite el siguiente valor.
- El método `next()` ejecuta el generador hasta el siguiente `yield` dentro del mismo y devuelve un objeto con el valor.
 

```
var iter = myGenerator();
// ...
rslt = iter.next();
// ...
```
- La nueva sintaxis del `for` permite recorrer el iterador completo:
 

```
for (let value of iter)
```

© JMA 2015. All rights reserved

# Nuevos Objetos

- Map: Lista de pares clave-valor.
- Set: Colección de valores únicos que pueden ser de cualquier tipo.
- WeakMap: Colección de pares clave-valor en los que cada clave es una referencia de objeto.
- WeakSet: Colección de objetos únicos.
- Promise: Proporciona un mecanismo para programar el trabajo de modo que se lleve a cabo en un valor que todavía no se calculó.
- Proxy: Habilita el comportamiento personalizado de un objeto.
- Reflect: Proporciona métodos para su uso en las operaciones que se interceptan.
- Symbol: Permite crear un identificador único.
- Intl.Collator: Proporciona comparaciones de cadenas de configuración regional.
- Intl.DateTimeFormat: Proporciona formato de fecha y hora específico de la configuración regional.
- Intl.NumberFormat: Proporciona formato de número específico de la configuración regional.

© JMA 2015. All rights reserved

# Nuevos Objetos

- ArrayBuffer: Representa un búfer sin formato de datos binarios, que se usa para almacenar datos de las diferentes matrices con tipo. No se puede leer directamente de ArrayBuffer ni escribir directamente en ArrayBuffer, pero se puede pasar a una matriz con tipo o un objeto DataView para interpretar el búfer sin formato según sea necesario.
- DataView: Se usa para leer y escribir diferentes tipos de datos binarios en cualquier ubicación de ArrayBuffer.
- Float32Array: Matriz con tipo de valores flotantes de 32 bits.
- Float64Array: Matriz con tipo de valores flotantes de 64 bits.
- Int8Array: Matriz con tipo de valores enteros de 8 bits.
- Int16Array: Matriz con tipo de valores enteros de 16 bits.
- Int32Array: Matriz con tipo de valores enteros de 32 bits.
- Uint8Array: Matriz con tipo de valores enteros sin signo de 8 bits.
- Uint8ClampedArray: Matriz con tipo de enteros sin signo de 8 bits con valores fijos.
- Uint16Array: Matriz con tipo de valores enteros sin signo de 16 bits.
- Uint32Array: Matriz con tipo de valores enteros sin signo de 32 bits.

© JMA 2015. All rights reserved

# Promise Pattern

- El Promise Pattern es un patrón de organización de código que permite encadenar llamadas a métodos que se ejecutarán a la conclusión del anterior (flujos).
- Simplifica y soluciona los problemas comunes con el patrón Callback:
  - Llamadas anidadas
  - Complejidad de código
$$o.m(1, 2, f(m1(3, f1(4, 5, ff(8)))) \rightarrow o.m(1, 2).f().m1(3).f1(4, 5).ff(8)$$
- Aunque se utiliza extensamente para las operaciones asíncronas, no es exclusivo de las mismas.
- El servicio \$q es un servicio de AngularJS que contiene toda la funcionalidad de las promesas (está basado en la implementación Q de Kris Kowal).
- La librería JQuery incluye el objeto \$.Deferred desde la versión 1.5.
- Las promesas se han incorporado a los objetos estándar de JavaScript en la versión 6.

© JMA 2015. All rights reserved

# Objeto Promise

- Una “promesa” es un objeto que actúa como proxy en los casos en los que no se puede utilizar el verdadero valor porque aún no se conoce (no se ha generado, llegado, ...) pero se debe continuar sin esperar a que este disponible (no se puede bloquear la función esperando a su obtención).
- Una “promesa” puede tener los siguientes estados:
  - Pendiente: Aún no se sabe si se podrá o no obtener el resultado.
  - Resuelta: Se ha podido obtener el resultado (`Promise.resolve()`)
  - Rechazada: Ha habido algún tipo de error y no se ha podido obtener el resultado (`Promise.reject()`)
- Los métodos del objeto promesa devuelven al propio objeto para permitir apilar llamadas sucesivas.
- Como objeto, la promesa se puede almacenar en una variable, pasar como parámetro o devolver desde una función, lo que permite aplicar los métodos en distintos puntos del código.

© JMA 2015. All rights reserved

# Crear promesas

- El objeto Promise gestiona la creación de la promesa y los cambios de estados de la misma.

```
list() {
  return new Promise((resolve, reject) => {
    this.http.get(this.baseUrl).subscribe(
      data => resolve(data),
      err => reject(err)
    )
  });
}
```

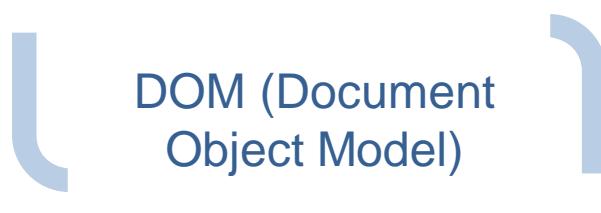
- Para crear promesas ya concluidas:
  - `Promise.reject`: Crea una promesa nueva como rechazada cuyo resultado es igual que el argumento pasado.
  - `Promise.resolve`: Crea una promesa nueva como resuelta cuyo resultado es igual que su argumento.

© JMA 2015. All rights reserved

## Invocar promesas

- El objeto Promise creado expone los métodos:
  - then(fnResuelta, fnRechazada): Recibe como parámetro la función a ejecutar cuando termine la anterior y, opcionalmente, la función a ejecutar en caso de que falle la anterior.
  - catch(fnError): Recibe como parámetro la función a ejecutar en caso de que falle.  
list().then(calcular, ponError).then(guardar)
- Otras formas de crear e invocar promesas son:
  - Promise.all: Combina dos o más promesas y realiza la devolución solo cuando todas las promesas especificadas se completan o alguna se rechaza.
  - Promise.race: Crea una nueva promesa que resolverá o rechazará con el mismo valor de resultado que la primera promesa que se va resolver o rechazar entre los argumentos pasados.

© JMA 2015. All rights reserved



© JMA 2015. All rights reserved

# DOM

© JMA 2015. All rights reserved

## Introducción a DOM

- El Modelo de Objetos del Documento es una interfaz independiente de la plataforma y del lenguaje que permite a programas y scripts acceder y actualizar dinámicamente los contenidos, la estructura y el estilo de los documentos.
- El Nivel 1 del Modelo de Objetos del Documento proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar de cómo pueden combinarse estos objetos y una interfaz estándar para acceder a ellos y manipularlos.
- El Modelo de Objetos del Documento Nivel 2 Hojas de Estilo y Hojas de Estilo en Cascada (CSS) define una interfaz neutral para la plataforma y el lenguaje que permite a los programas y scripts acceder dinámicamente y actualizar el contenido de documentos de hojas de estilo.
- El Modelo de Objetos del Documento de nivel 3 especifica una plataforma y un interfaz de lenguaje neutral que permiten a programas y scripts acceder dinámicamente y actualizar el contenido, estructura y estilos del documento.
- Se encuentra normalizado a través de la recomendación del W3C:
  - [http://www.w3.org/TR/tr\\_DOM](http://www.w3.org/TR/tr_DOM)
- Es responsabilidad de los navegadores dar soporte a los diferentes niveles de las recomendaciones.

© JMA 2015. All rights reserved

## Soporte del DOM

- La primera especificación del DOM (Document Object Model Level 1) se definió en 1998 y permitió homogeneizar la implementación del DHTML o HTML dinámico en los diferentes navegadores, ya que permitía modificar el contenido de las páginas web sin necesidad de recargar la página entera.
- Desafortunadamente, las posibilidades teóricas de DOM son mucho más avanzadas de las que se pueden utilizar en la práctica para desarrollar aplicaciones web.
- El motivo es que el uso de DOM siempre está limitado por las posibilidades que ofrece cada navegador.
- Mientras que algunos navegadores como Firefox y Safari implementan DOM de nivel 1 y 2 (y parte del 3), otros navegadores como Internet Explorer (versión 7 y anteriores) ni siquiera son capaces de ofrecer una implementación completa de DOM nivel 1.
- Con la aparición del HTML 5, esta situación se corregirá al unificar las recomendaciones del DOM con las del propio HTML, pero persistirá en las implementaciones anteriores de los navegadores.

© JMA 2015. All rights reserved

## Árbol de nodos

- Antes de poder utilizar sus funciones, DOM transforma internamente el archivo XML o HTML original en una estructura más fácil de manejar formada por una jerarquía de nodos.
- DOM transforma el código XML en una serie de nodos interconectados en forma de árbol.
- El árbol generado no sólo representa los contenidos del archivo original (mediante los nodos del árbol) sino que también representa sus relaciones (mediante las ramas del árbol que conectan los nodos).
- Aunque en ocasiones DOM se asocia con la programación web y con JavaScript, la API de DOM es independiente de cualquier lenguaje de programación.
- DOM está disponible en la mayoría de lenguajes de programación comúnmente empleados.

© JMA 2015. All rights reserved

# Árbol de nodos

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <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>
```

```

graph TD
    DX[Documento XHTML] --> EH[Elemento HEAD]
    DX --> EB[Elemento BODY]
    EH --> EM[Elemento META]
    EH --> ET[Elemento TITLE]
    ET --> TP[Texto Página sencilla]
    EB --> EP[Elemento P]
    EP --> TE[Texto Esta página es]
    TE --> ES[Elemento STRONG]
    ES --> TM[Texto muy sencilla]
  
```

© JMA 2015. All rights reserved

# Tipos de nodos

- Los documentos XML y HTML tratados por DOM se convierten en una jerarquía de nodos.
- Los nodos que representan los documentos pueden ser de diferentes tipos.
- Los tipos más importantes son:
  - Document: es el nodo raíz de todos los documentos HTML y XML. Todos los demás nodos derivan de él.
  - DocumentType: es el nodo que contiene la representación del DTD empleado en la página (indicado mediante el DOCTYPE).
  - Element: representa el contenido definido por un par de etiquetas de apertura y cierre (<etiqueta>...</etiqueta>) o de una etiqueta abreviada que se abre y se cierra a la vez (<etiqueta/>). Es el único nodo que puede tener tanto nodos hijos como atributos.
  - Attr: representa el par nombre-de-atributo/valor.
  - Text: almacena el contenido del texto que se encuentra entre una etiqueta de apertura y una de cierre. También almacena el contenido de una sección de tipo CDATA.
  - CDataSection: es el nodo que representa una sección de tipo <![CDATA[ ]]>.
  - Comment: representa un comentario de XML.
- Los otros tipos de nodos son: DocumentFragment, Entity, EntityReference, ProcessingInstruction y Notation.

© JMA 2015. All rights reserved

## Interfaz Node: Constantes

- Node.ELEMENT\_NODE = 1
- Node.ATTRIBUTE\_NODE = 2
- Node.TEXT\_NODE = 3
- Node.CDATA\_SECTION\_NODE = 4
- Node.ENTITY\_REFERENCE\_NODE = 5
- Node.ENTITY\_NODE = 6
- Node.PROCESSING\_INSTRUCTION\_NODE = 7
- Node.COMMENT\_NODE = 8
- Node.DOCUMENT\_NODE = 9
- Node.DOCUMENT\_TYPE\_NODE = 10
- Node.DOCUMENT\_FRAGMENT\_NODE = 11
- Node.NOTATION\_NODE = 12

© JMA 2015. All rights reserved

## Interfaz Node: Propiedades

Propiedad	Valor devuelto	Descripción
nodeName	String	El nombre del nodo (no definido en todos tipos de nodo)
nodeValue	String	El valor del nodo (no definido en todos los tipos de nodo)
nodeType	Number	Una de las 12 constantes definidas anteriormente
ownerDocument	Document	Referencia del documento al que pertenece el nodo
firstChild	Node	Referencia del primer nodo de la lista childNodes
lastChild	Node	Referencia del último nodo de la lista childNodes
childNodes	NodeList	Lista de todos los nodos hijo del nodo actual
previousSibling	Node	Referencia del nodo hermano anterior o null si este nodo es el primer hermano
nextSibling	Node	Referencia del nodo hermano siguiente o null si este nodo es el último hermano
attributes	NamedNodeMap	Se emplea con nodos de tipo Element. Contiene objetos de tipo Attr que definen todos los atributos del elemento

© JMA 2015. All rights reserved

# Interfaz Node: Métodos

Método	Valor devuelto	Descripción
hasChildNodes()	Boolean	Devuelve true si el nodo actual tiene uno o más nodos hijo
appendChild(nodo)	Node	Añade un nuevo nodo al final de la lista childNodes
insertBefore(nuevoNodo, anteriorNodo)	Node	Inserta el nodo nuevoNodo antes que la posición del nodo anteriorNodo dentro de la lista childNodes
removeChild(nodo)	Node	Elimina un nodo de la lista childNodes
replaceChild(nuevoNodo, anteriorNodo)	Node	Reemplaza el nodo anteriorNodo por el nodo nuevoNodo

© JMA 2015. All rights reserved

## Acceso a los nodos

- DOM proporciona dos métodos alternativos para acceder a un nodo específico: acceso a través de sus nodos padre y acceso directo.
- Los propiedades del interfaz Node permiten acceder al nodo raíz de la página y navegar a través de la jerarquía de los nodos padre a sus nodos hijos hasta alcanzar el nodo buscado.
- DOM proporciona una serie de métodos para acceder de forma directa a los nodos deseados:
  - **getElementsByName(eti):** obtiene todos los elementos de la página cuya etiqueta sea igual que el parámetro que se le pasa a la función.
  - **getElementsByName(name):** obtiene todos los elementos de la página cuyo atributo name coincide con el parámetro que se le pasa a la función.
  - **getElementById(id):** obtiene el elemento de la página cuyo atributo id coincide con el parámetro que se le pasa a la función. Se trata de la función preferida para acceder directamente a un nodo.

© JMA 2015. All rights reserved

# Atributos

- Los nodos de tipo Element contienen la propiedad attributes, que permite acceder a todos los atributos de cada elemento.
- DOM proporciona diversos métodos para tratar con los atributos:
  - getItem(nombre), devuelve el nodo cuya propiedad nodeName contenga el valor nombre.
  - setNamedItem(nodo), añade el nodo a la lista attributes, indexándolo según su propiedad nodeName.
  - removeNamedItem(nombre), elimina el nodo cuya propiedad nodeName coincide con el valor nombre.
  - item(posicion), devuelve el nodo que se encuentra en la posición indicada por el valor numérico del parámetro.
- Estos métodos devuelven un nodo de tipo Attr, para acceder a su valor para consultarlo o modificarlo es necesario utilizar su propiedad nodeValue.
- DOM proporciona métodos de acceso directo:
  - getAttribute(nombre), es equivalente a attributes.getNamedItem(nombre).
  - setAttribute(nombre, valor) equivalente a attributes.getNamedItem(nombre).value = valor.
  - removeAttribute(nombre), equivalente a attributes.removeNamedItem(nombre).

© JMA 2015. All rights reserved

# Creación y eliminación de nodos

- Los métodos DOM disponibles para el mantenimiento del árbol de nodos son los siguientes:
  - createElement(eti): Crea un elemento del tipo indicado en el parámetro.
  - createAttribute(atrib): Crea un nodo de tipo atributo con el nombre indicado.
  - createTextNode(texto): Crea un nodo de tipo texto con el valor indicado como parámetro.
  - appendChild(nodo): Añade un nodo al final de la lista childNodes de otro nodo. Se debe invocar sobre el nodo que va a ser nodo padre del nodo añadido.
  - replaceChild(new, old): intercambia un nodo por otro. Se debe invocar sobre el nodo padre que contiene el nodo que se va a cambiar.
  - removeChild(nodo): elimina un nodo. Se debe invocar sobre el nodo padre del nodo que se va a eliminar.

© JMA 2015. All rights reserved

## Pasos para crear elementos

1. Crear un nodo de tipo elemento
2. Cualificar el nuevo elemento con atributos. Por cada atributo:
  1. Crear un nodo de tipo atributo
  2. Asociar el nodo de atributo al elemento
3. Dar contenido al nuevo elemento.
  1. Crear un nodo de tipo texto
  2. Asociar el nodo de texto al elemento
4. Modificar al elemento padre que lo va a contener en la página original.
  - Añadir el nodo del nuevo elemento.
  - Buscar el nodo original y sustituirlo por el nodo del nuevo elemento.

© JMA 2015. All rights reserved

## HTML DOM

- El Modelo de Objetos del Documento HTML amplia el interfaz Node exponiendo ciertos métodos y propiedades de conveniencia que son consistentes con los modelos existentes y que son más apropiados para los autores de scripts.
- Incluye las siguientes especializaciones para HTML:
  - Una Interfaz HTMLDocument, derivada de la interfaz Document del núcleo. HTMLDocument especifica las operaciones y consultas que pueden realizarse en un documento HTML.
  - Una Interfaz HTMLElement, derivada de la interfaz Element del núcleo. HTMLElement especifica las operaciones y consultas que pueden realizarse en cualquier elemento HTML. Entre los métodos de HTMLElement se incluyen aquellos que permiten leer y modificar los atributos que se aplican a todos los elementos HTML.
  - Especializaciones para todos los elementos HTML que tengan atributos que vayan más allá de los especificados en la Interfaz HTMLElement. La interfaz derivada para el elemento contiene métodos explícitos para establecer y obtener los valores para todos estos atributos . Así mismo expone los diferentes eventos soportados por cada elemento.

© JMA 2015. All rights reserved

# Interfaz HTMLDocument

- Un HTMLDocument es la raíz de la jerarquía HTML y almacena todos los contenidos. Además de proporcionar acceso a la jerarquía, también proporciona algunos métodos de conveniencia para acceder a ciertos conjuntos de información del documento.
- Métodos:
  - open: Abre un flujo de documento para escribir.
  - close: Cierra un flujo de documento abierto por open() y fuerza la representación.
  - write: Escribe una cadena de texto en un flujo de documento abierto por open().
  - writeln: Escribe una cadena de texto seguida por un carácter de nueva línea en un flujo de documento abierto por open().
  - getElementById: Devuelve el elemento cuyo id está dado por elementId.
  - getElementsByName: Devuelve el conjunto (posiblemente vacío) de elementos cuyo valor name está dado por elementName.
  - querySelectorAll: Devuelve todos los elementos secundarios que coinciden con un selector CSS.

© JMA 2015. All rights reserved

# HTMLDocument

- Atributos:
  - title: El título de un documento según se especifica en el elemento TITLE de la cabecera del documento.
  - referrer: Devuelve el URI de la página que referenció a esta página. Si el usuario navegó hasta esta página directamente el valor es una cadena vacía.
  - domain: El nombre de dominio del servidor que sirvió el documento, o una cadena vacía si el servidor no puede ser identificado por un nombre de dominio.
  - URL: El URI completo del documento.
  - body: El elemento que contiene el contenido del documento.
  - images: Un conjunto de todos los elementos IMG de un documento. Por motivos de compatibilidad, el comportamiento se limita a elementos IMG.
  - applets: Un conjunto de todos los elementos OBJECT que incluyan aplicaciones y elementos APPLET (desaprobados) de un documento.
  - links: Un conjunto de todos los elementos AREA y elementos ancla (A) de un documento con un valor para el atributo href.
  - forms: Un conjunto de todos los formularios de un documento.
  - anchors: Un conjunto de todos los elementos ancla (A) de un documento con un valor para el atributo name.
  - cookie: Las cookies asociadas a este documento.

© JMA 2015. All rights reserved

# Interfaz HTMLElement

- Todas las interfaces de elementos HTML derivan de esta interfaz.
- Atributos:
  - id: El identificador del elemento.
  - title: El título consultivo del elemento.
  - lang: Código de idioma definido en RFC 1766.
  - dir: Especifica la dirección base de la direccionalidad del texto neutral y la de direccionalidad de tablas.
  - className: El atributo de class del elemento.
  - innerHTML: Contenido de un elemento.
  - textContent: Contenido textual de un nodo y sus descendientes
  - style: Estilo de un elemento

© JMA 2015. All rights reserved

## Métodos HTMLElement

Método	Descripción
click()	Simula un clic del ratón sobre un elemento
focus(), blur()	Da y quita el foco a un elemento
addEventListener()	Conecta un controlador de eventos para el elemento especificado
removeEventListener()	Elimina un controlador de eventos que se han asociado con addEventListener()
getElementsByClassName(), getElementsByTagName()	Devuelve una colección de todos los elementos secundarios con el nombre de clase especificado o el nombre de la etiqueta especificada
querySelector()	Devuelve el primer elemento hijo que coincide con un selector CSS
querySelectorAll()	Devuelve todos los elementos secundarios que coinciden con un selector CSS
setAttribute()	Establece o cambia el atributo especificado con el valor especificado
removeAttribute()	Elimina un atributo especificado de un elemento
appendChild()	Añade un nuevo nodo hijo a un elemento como el último nodo hijo
replaceChild()	Reemplaza un nodo secundario en un elemento
removeChild()	Elimina un nodo hijo de un elemento
cloneNode()	Clones un elemento

© JMA 2015. All rights reserved

# Interfaces HTMLElement

HTMLHtmlElement	HTMLInputElement	HTMLDivElement	HTMLParamElement
HTMLHeadElement	HTMLTextAreaElement	HTMLParagraphElement	HTMLAppletElement
HTMLLinkElement	HTMLButtonElement	HTMLHeadingElement	HTMLMapElement
HTMLTitleElement	HTMLLabelElement	HTMLQuoteElement	HTMLAreaElement
HTMLMetaElement	HTMLFieldSetElement	HTMLPreElement	HTMLScriptElement
HTMLBaseElement	HTMLLegendElement	HTMLBRElement	HTMLTableElement
HTMLIndexElement	HTMLULListElement	HTMLBaseFontElement	HTMLTableCaptionElement
HTMLStyleElement	HTMLOLListElement	HTMLFontElement	HTMLTableColElement
HTMLBodyElement	HTMLDLListElement	HTMLHRElement	HTMLTableSectionElement
HTMLFormElement	HTMLDirectoryElement	HTMLModElement	HTMLTableRowElement
HTMLSelectElement	HTMLMenuElement	HTMLAnchorElement	HTMLTableCellElement
HTMLOptGroupElement	HTMLLIElement	HTMLImageElement	HTMLFrameSetElement
HTMLOptionElement	HTMLBlockquoteElement	HTMLObjectElement	HTMLFrameElement
			HTMLIFrameElement

© JMA 2015. All rights reserved

# Programación basada en eventos

- Frente a la programación tradicional, donde las aplicaciones se ejecutaban secuencialmente de principio a fin, en la actualidad el modelo predominante es el de la programación asíncrona basada en eventos.
- Los scripts y programas inicializan el entorno o página y quedan a la espera, sin realizar ninguna tarea, hasta que se produzca un evento, situación ante la cual les interesa reaccionar.
- Una vez producido, para realizar el tratamiento del evento, ejecutan alguna tarea asociada a la aparición de ese evento. Al concluir el tratamiento, el script o programa vuelve al estado de espera del siguiente evento.
- El tratamiento del evento suele encapsularse en una función, conocidas como “controladores de eventos” o “manejadores de eventos”.
- JavaScript permite asignar una función a cada uno de los eventos. De esta forma, cuando se produce cualquier evento, JavaScript ejecuta su función asociada.
- Si la función devuelve el valor false se cancela la ejecución del evento.

© JMA 2015. All rights reserved

# Eventos en DOM

- El nivel 1 de DOM no incluye especificaciones relativas a los eventos JavaScript. El nivel 2 de DOM incluye ciertos aspectos relacionados con los eventos y el nivel 3 de DOM incluye la especificación completa de los eventos de JavaScript.
- La incompatibilidad más importante entre navegadores y versiones de los mismos se da en el modelo de eventos del navegador.
- Así, existen hasta tres modelos diferentes para manejar los eventos dependiendo del navegador en el que se ejecute la aplicación:
  - **Modelo básico de eventos:** Este modelo simple de eventos se introdujo para la versión 4 del estándar HTML y se considera parte del nivel más básico de DOM. Aunque sus características son limitadas, es el único modelo que es compatible en todos los navegadores y por tanto, el único que permite crear aplicaciones que funcionan de la misma manera en todos los navegadores.
  - **Modelo de eventos estándar:** Las versiones más avanzadas del estándar DOM (DOM nivel 2) definen un modelo de eventos completamente nuevo y mucho más poderoso que el original. Todos los navegadores modernos lo incluyen, salvo Internet Explorer.
  - **Modelo de eventos de Internet Explorer:** Internet Explorer utiliza su propio modelo de eventos, que es similar pero incompatible con el modelo estándar. Creado para Internet Explorer 4, Microsoft decidió seguir utilizándolo en el resto de versiones, a pesar de haber participado en la creación del estándar de DOM que define el modelo de eventos estándar.

© JMA 2015. All rights reserved

## Modelo básico de eventos

- Cada elemento o etiqueta define su propia lista de posibles eventos que se le pueden asignar.
- Un mismo tipo de evento puede estar definido para varios elementos diferentes y un elemento puede tener asociados varios eventos diferentes.
- El nombre de cada evento se construye mediante el prefijo on, seguido del nombre en inglés de la acción asociada al evento.
- Algunos de los eventos mas comunes son:
  - onclick: Pinchar y soltar el ratón
  - onchange: Deseleccionar un elemento que se ha modificado
  - onfocus: Seleccionar un elemento
  - onblur: Deseleccionar el elemento
  - onselect: Seleccionar un texto
  - onsubmit: Enviar el formulario
  - onload: La página se ha cargado completamente
  - onunload: Se abandona la página

© JMA 2015. All rights reserved

## Manejadores de eventos

- El HTML expone, en su recomendación, la lista de posibles eventos de una etiqueta como si fueran atributos de la etiqueta.
- Existen varias formas para enlazar el código de control del evento al evento:
  - Manejadores como atributos de los elementos.
    - En la etiqueta, el valor del atributo evento es el código JavaScript del controlador.
  - Manejadores como funciones externas.
    - En la etiqueta, el valor del atributo evento es la invocación de una función reutilizable de JavaScript que es el controlador.
  - Manejadores “semánticos” o “no intrusivos”.
    - Para separar los contenidos del comportamiento, una vez definido el contenido en HTML, mediante un script se localizan las etiquetas y se les asocia a sus eventos los controladores de eventos.

© JMA 2015. All rights reserved

## Modelo de eventos estándar

- La especificación DOM define otros dos métodos:
  - addEventListener(): asocia manejadores de eventos a eventos.
  - removeEventListener(): desasociar manejadores de eventos previamente asociados al evento.
- Ambos métodos requieren los tres mismos parámetros:
  - el nombre del “event listener”: el nombre del evento sin el prefijo on.
  - una referencia a la función encargada de procesar el evento
  - el tipo de flujo de eventos al que se aplica: true cuando el manejador se emplea en la fase de capture y false cuando el manejador se asocia a la fase de bubbling.
- El modelo DOM permiten asociar mas de un manejador de evento al mismo evento de un objeto, por lo que para desasociar el evento los parámetros deben ser exactamente los mismos que en la asociación.

© JMA 2015. All rights reserved

# El flujo de eventos

- El flujo de eventos permite que varios elementos diferentes puedan responder a un mismo evento, propagación del evento siguiendo las relaciones de contenidos y contenidos.
- El orden en el que se ejecutan los eventos asignados a cada elemento de la página es lo que constituye el flujo de eventos, pero existen muchas diferencias en el flujo de eventos de cada navegador:
  - Event bubbling: En este modelo, el orden que se sigue es ascendente desde el elemento más específico (contenido) hasta el elemento menos específico (continente).
  - Event capturing: el flujo de eventos se define descendente desde el elemento menos específico hasta el elemento más específico.
  - Eventos DOM: El flujo de eventos definido en la especificación DOM soporta tanto el bubbling como el capturing, pero el "event capturing" se ejecuta en primer lugar. Los dos flujos de eventos recorren todos los objetos DOM desde el objeto document hasta el elemento más específico y viceversa. Además, la mayoría de navegadores que implementan los estándares, continúan el flujo hasta el objeto window.

© JMA 2015. All rights reserved

# El objeto event

- Cuando se produce un evento, no es suficiente con asignarle una función responsable de procesar ese evento.
- El objeto event es el mecanismo definido por los navegadores para proporcionar toda esa información.
- Se trata de un objeto que se crea automáticamente cuando se produce un evento y que se destruye de forma automática cuando se han ejecutado todas las funciones asignadas al evento.
- El estándar DOM especifica que el objeto event es el único parámetro que se debe pasar a las funciones encargadas de procesar los eventos.
  - var ev= arguments[0];
- Internet Explorer permite el acceso al objeto event a través del objeto window.
  - var ev= window.event;
- El objeto event presenta unas propiedades y métodos muy diferentes en función del tipo de navegador en el que se ejecuta la aplicación JavaScript.

© JMA 2015. All rights reserved

## event (DOM)

Propiedad/Método	Devuelve	Descripción
altKey	Boolean	Devuelve true si se ha pulsado la tecla ALT y false en otro caso
bubbles	Boolean	Indica si el evento pertenece al flujo de eventos de bubbling
button	Entero	El botón del ratón que ha sido pulsado.
cancelable	Boolean	Indica si el evento se puede cancelar
cancelBubble	Boolean	Indica si se ha detenido el flujo de eventos de tipo bubbling
charCode	Entero	El código unicode del carácter correspondiente a la tecla pulsada
clientX	Entero	Coordenada X de la posición del ratón respecto del área visible de la ventana
clientY	Entero	Coordenada Y de la posición del ratón respecto del área visible de la ventana
ctrlKey	Boolean	Devuelve true si se ha pulsado la tecla CTRL y false en otro caso

© JMA 2015. All rights reserved

## event (DOM)

Propiedad/Método	Devuelve	Descripción
currentTarget	Element	El elemento que es el objetivo del evento
detail	Entero	El número de veces que se han pulsado los botones del ratón
eventPhase	Entero	La fase a la que pertenece el evento: 0 – Fase capturing 1 – En el elemento destino 2 – Fase bubbling
isChar	Boolean	Indica si la tecla pulsada corresponde a un carácter
keyCode	Entero	Indica el código numérico de la tecla pulsada
metaKey	Entero	Devuelve true si se ha pulsado la tecla META y false en otro caso
pageX	Entero	Coordenada X de la posición del ratón respecto de la página
pageY	Entero	Coordenada Y de la posición del ratón respecto de la página
preventDefault()	Función	Se emplea para cancelar la acción predefinida del evento

© JMA 2015. All rights reserved

## event (DOM)

Propiedad/ Método	Devuelve	Descripción
relatedTarget	Element	El elemento que es el objetivo secundario del evento (relacionado con los eventos de ratón)
screenX	Entero	Coordenada X de la posición del ratón respecto de la pantalla completa
screenY	Entero	Coordenada Y de la posición del ratón respecto de la pantalla completa
shiftKey	Boolean	Devuelve true si se ha pulsado la tecla SHIFT y false en otro caso
stopPropagation()	Función	Se emplea para detener el flujo de eventos de tipo bubbling
target	Element	El elemento que origina el evento
timeStamp	Número	La fecha y hora en la que se ha producido el evento
type	Cadena	El nombre del evento

© JMA 2015. All rights reserved

## event (Internet Explorer)

Propiedad/ Método	Devuelve	Descripción
altKey	Boolean	Devuelve true si se ha pulsado la tecla ALT y false en otro caso
button	Entero	El botón del ratón que ha sido pulsado.
cancelBubble	Boolean	Si se establece un valor true, se detiene el flujo de eventos de tipo bubbling
clientX	Entero	Coordenada X de la posición del ratón respecto del área visible de la ventana
clientY	Entero	Coordenada Y de la posición del ratón respecto del área visible de la ventana
ctrlKey	Boolean	Devuelve true si se ha pulsado la tecla CTRL y false en otro caso
fromElement	Element	El elemento del que sale el ratón (para ciertos eventos de ratón)
keyCode	Entero	En el evento keypress, indica el carácter de la tecla pulsada.
offsetX	Entero	Coordenada X del ratón respecto del elemento que origina el evento
offsetY	Entero	Coordenada Y del ratón respecto del elemento que origina el evento

© JMA 2015. All rights reserved

# event (Internet Explorer)

Propiedad/ Método	Devuelve	Descripción
repeat	Boolean	Devuelve true si se está produciendo el evento keydown de forma continua y false en otro caso
returnValue	Boolean	Se emplea para cancelar la acción predefinida del evento
screenX	Entero	Coordenada X de la posición del ratón respecto de la pantalla completa
screenY	Entero	Coordenada Y de la posición del ratón respecto de la pantalla completa
shiftKey	Boolean	Devuelve true si se ha pulsado la tecla SHIFT y false en otro caso
srcElement	Element	El elemento que origina el evento
toElement	Element	El elemento al que entra el ratón (para ciertos eventos de ratón)
type	Cadena	El nombre del evento
x	Entero	Coordenada X del ratón respecto del elemento padre del elemento que origina el evento
y	Entero	Coordenada Y del ratón respecto del elemento padre del elemento que origina el evento

© JMA 2015. All rights reserved

## Asociar eventos

- Controlador de eventos:  

```
function controlador(e) {
    e.preventDefault(); // Cancela propagación
    // ...
}
```
- Asociar declarativamente:  

```
<input type="button" id="btn" value="Enviar" onclick="controlador();" />
```
- Asociar controlador único:  

```
window.onload = controlador;
```
- Asociar un listener:  

```
window.addEventListener("load", controlador, false);
```
- Desasociar el controlador:  

```
window.onload = null;
window.removeEventListener("load", controlador, false);
```

© JMA 2015. All rights reserved

# JavaScript no intrusivo

- JavaScript no intrusivo (*unobtrusive javascript*) es un paradigma en el uso del lenguaje de programación JavaScript que promueve:
  - Separación de la funcionalidad JavaScript (la "capa del comportamiento") de las capas de estructura/contenido y de presentación de una página.
  - Uso de buenas prácticas a fin de evitar los problemas de incompatibilidad de la programación tradicional en JavaScript (tales como inconsistencias entre navegadores y falta de escalabilidad).
  - Cumplimiento de las normas de accesibilidad.
- **Intrusivo:**  
`<input type="button" id="btn" value="pulsa aquí" onclick="helloWorld();"/>`
- **No intrusivo:**  
`<input type="button" id="btn" value="pulsa aquí" />`

```
window.onload = function() {
  document.getElementById("btn").onclick = helloWorld;
}
// HTML5
document.addEventListener("DOMContentLoaded", function() {
  document.getElementById("btn").addEventListener("click", helloWorld);
});
```

© JMA 2015. All rights reserved

# Formularios

- Los formularios permiten las entradas de usuario, una de las principales razones por las que se inventó el lenguaje de programación JavaScript fue la necesidad de validar los datos de los formularios directamente en el navegador del usuario.
- `document.forms` es el array que contiene todos los formularios de la página.
- Se crea automáticamente un array llamado `elements` por cada uno de los formularios de la página que contiene la referencia a todos los elementos (cuadros de texto, botones, listas desplegables, etc.) del formulario.
- Cada elemento cuenta con las siguientes propiedades:
  - `type`: indica el tipo de elemento que se trata. Para los elementos de tipo `<input>` (text, button, checkbox, etc.) coincide con el valor de su atributo `type`. Para las listas desplegables normales (`<select>`) su valor es `select-one`, lo que permite diferenciarlas de las listas que permiten seleccionar varios elementos a la vez y cuyo tipo es `select-multiple`. Por último, en los elementos de tipo `<textarea>`, el valor de `type` es `textarea`.
  - `form`: es una referencia directa al formulario al que pertenece el elemento.
  - `name`: obtiene el valor del atributo `name` de XHTML.
  - `value`: permite leer y modificar el valor del atributo `value`. Para los campos de texto (`<input type='text'>` y `<textarea>`) obtiene el texto que ha escrito el usuario. Para los botones obtiene el texto que se muestra en el botón.

© JMA 2015. All rights reserved

# Formularios

- Los eventos más utilizados en el manejo de los formularios son los siguientes:
  - onclick: evento que se produce cuando se pincha con el ratón sobre un elemento. Normalmente se utiliza con cualquiera de los tipos de botones (`<input type="button">`, `<input type="submit">`, `<input type="image">`).
  - onchange: evento que se produce cuando el usuario cambia el valor de un elemento de texto (`<input type="text">` o `<textarea>`). También se produce cuando el usuario selecciona una opción en una lista desplegable (`<select>`). Sin embargo, el evento sólo se produce si después de realizar el cambio, el usuario pasa al siguiente campo del formulario, lo que técnicamente se conoce como que "el otro campo de formulario ha perdido el foco".
  - onfocus: evento que se produce cuando el usuario selecciona un elemento del formulario.
  - onblur: evento complementario de onfocus, ya que se produce cuando el usuario ha deseleccionado un elemento por haber seleccionado otro elemento del formulario. Técnicamente, se dice que el elemento anterior "ha perdido el foco".

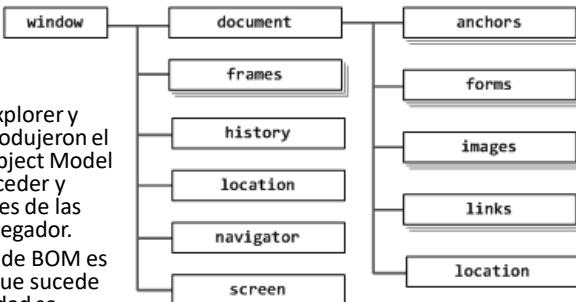
© JMA 2015. All rights reserved

# Validación

- La validación de formularios en cliente evita la sobrecarga que supone las innecesarias idas y venidas al servidor, pero no sustituye la validación de servidor, con lo que mejorar la experiencia de usuario y ayuda a reducir el consumo de red y la carga de procesamiento en el servidor.
- Normalmente, la validación de un formulario consiste en llamar a una función de validación cuando el usuario pulsa sobre el botón de envío del formulario.
- El formulario dispone del evento onsubmit que se dispara justo antes de enviar el formulario. La función de validación se puede asociar a dicho evento.
- Si el evento onsubmit devuelve el valor true, el formulario se envía normalmente pero, si es false, se cancelara en envío del formulario.
- El formulario se envía cuando el usuario pulsa sobre el botón `<input type="submit">` o cuando se invoca el método `form.submit()`.

© JMA 2015. All rights reserved

# BOM (Browser Object Model)



- Las versiones 3.0 de los navegadores Internet Explorer y Netscape Navigator introdujeron el concepto de Browser Object Model o BOM, que permite acceder y modificar las propiedades de las ventanas del propio navegador.
- El mayor inconveniente de BOM es que, al contrario de lo que sucede con DOM, ninguna entidad se encarga de estandarizarlo o definir unos mínimos de interoperabilidad entre navegadores.

© JMA 2015. All rights reserved

## window

- El objeto window representa la ventana completa del navegador. Mediante este objeto, es posible mover, redimensionar y manipular la ventana actual del navegador. Incluso es posible abrir y cerrar nuevas ventanas de navegador.
- La propiedades mas habituales de la ventana son:
  - defaultStatus: Establece o devuelve el texto por defecto en la barra de estado de una ventana
  - name: Establece o devuelve el nombre de una ventana
  - status: Establece o devuelve el texto en la barra de estado de una ventana
  - opener: Devuelve una referencia a la ventana que creó la ventana
  - parent: Devuelve la ventana principal de la ventana actual
  - closed: Devuelve un valor booleano que indica si una ventana se ha cerrado o no
  - innerHeight: Devuelve la altura interior del área de contenido de una ventana
  - innerWidth: Devuelve la anchura interior del área de contenido de una ventana
  - document: Devuelve el objeto de documento de la ventana (Ver objeto Document)
  - history: Devuelve el objeto Historia de la ventana (Ver objeto History)
  - location: Devuelve el objeto de localización de la ventana (Ver objeto Location)
  - navigator: Devuelve el objeto Navigator para la ventana (Ver objeto Navigator)

© JMA 2015. All rights reserved

# window

Método	Descripción
alert()	Muestra un cuadro de alerta con un mensaje y un botón Aceptar
prompt()	Muestra un cuadro de diálogo que solicita una entrada de texto
confirm()	Muestra un cuadro de diálogo con un mensaje con Aceptar/Cancelar
stop()	Detiene la ventana de carga
open()	Abre una nueva ventana del navegador
close()	Cierra la ventana actual
createPopup()	Crea una ventana emergente
print()	Imprime el contenido de la ventana actual
focus()	Pone el foco en la ventana actual
blur()	Quita el foco de la ventana actual
atob()	Decodifica una cadena codificada en base 64
btoa()	Codifica una cadena en base 64

© JMA 2015. All rights reserved

# window

Método	Descripción
setTimeout()	Llama a una función o evalúa una expresión después de un número especificado de milisegundos
clearTimeout()	Borra un temporizador programado con setTimeout ()
setInterval()	Llama a una función o una expresión evalúa a intervalos especificados (en milisegundos)
clearInterval()	Borra un temporizador programado con setInterval ()
moveTo()	Mueve una ventana a la posición especificada
moveBy()	Mueve una ventana con relación a su posición actual
resizeBy()	Cambia el tamaño de la ventana por los píxeles especificados
resizeTo()	Cambia el tamaño de la ventana a la anchura y altura especificadas
scrollBy()	Desplaza el documento por el número de píxeles especificado
scrollTo()	Desplaza el documento a las coordenadas especificadas

© JMA 2015. All rights reserved

## location

- El objeto location es uno de los objetos más útiles del BOM. Debido a la falta de estandarización, location es una propiedad tanto del objeto window como del objeto document.
- El objeto location representa la URL de la página HTML que se muestra en la ventana del navegador y proporciona varias propiedades y métodos útiles para el manejo de la URL:
  - hash: El contenido de la URL que se encuentra después del signo # (ancla)
  - href: La URL completa de la página actual
  - search: Todo el contenido que se encuentra tras el símbolo ?, es decir, la consulta o "query string"
  - assign(newURL): Equivalente a location.href = newURL
  - replace(newURL): Similar a assign(), salvo que se borra la página actual del array history del navegador
  - reload(sr): Recarga la página. Si el argumento es true, se carga la página desde el servidor, o cuando es false desde la cache del navegador.

© JMA 2015. All rights reserved

## history

- El objeto history proporciona acceso al historial del navegador. Esto expone métodos útiles y propiedades permiten avanzar y retroceder a través del historial del usuario:
  - length: Para obtener el número de páginas en el historial de la pila.
  - back(): Para moverse hacia atrás, página anterior, actuará exactamente como si el usuario hiciera clic en el botón atrás en la barra de herramientas del navegador.
  - forward(): Para moverse hacia adelante, página siguiente, actuará exactamente como si el usuario hiciera clic en el botón siguiente en la barra de herramientas del navegador.
  - go(np): para cargar una página desde el historial de la sesión, identificada por su posición relativa a la página actual (Iniciando con la página actual, relativa al índice 0, los valores negativos representan páginas anteriores y los positivos páginas posteriores).

© JMA 2015. All rights reserved

## navigator

- El objeto navigator es uno de los primeros objetos que incluyó el BOM y permite obtener información sobre el propio navegador.
- Aunque es uno de los objetos menos estandarizados, algunas de sus propiedades son comunes en casi todos los navegadores:
  - appName: Cadena que representa el nombre oficial del navegador
  - appVersion: Cadena que representa la versión del navegador
  - browserLanguage: Cadena que representa el idioma del navegador
  - cookieEnabled: Boolean que indica si las cookies están habilitadas
  - javaEnabled: Boolean que indica si Java está habilitado
  - language: Cadena que representa el idioma del navegador
  - platform: Cadena que representa la plataforma sobre la que se ejecuta el navegador
  - plugins: Array con la lista de plugins instalados en el navegador
  - userAgent: Cadena que representa la cadena que el navegador emplea para identificarse en los servidores

© JMA 2015. All rights reserved

## screen

- El objeto screen se utiliza para obtener información sobre la pantalla del usuario.
- Uno de los datos más importantes que proporciona el objeto screen es la resolución del monitor en el que se están visualizando las páginas.
- Los diseñadores de páginas web necesitan conocer las resoluciones más utilizadas por los usuarios para adaptar sus diseños a esas resoluciones.
- Las siguientes propiedades están disponibles en el objeto screen:
  - availHeight: Altura de pantalla disponible para las ventanas
  - availWidth: Anchura de pantalla disponible para las ventanas
  - colorDepth: Profundidad de color de la pantalla (32 bits normalmente)
  - height: Altura total de la pantalla en píxel
  - width: Anchura total de la pantalla en píxel

© JMA 2015. All rights reserved

# JavaScript Framework

- **MooTools** (My object oriented tools) es un Framework web orientado a objetos para JavaScript, de código abierto, compacto y modular. Aporta una manera de desarrollar JavaScript sin importar en qué navegador se ejecute de una manera elegante.
- **Prototype** es un framework escrito en JavaScript que se orienta al desarrollo sencillo y dinámico de aplicaciones web. Es una herramienta que implementa las técnicas AJAX y su potencial es aprovechado al máximo cuando se desarrolla con Ruby On Rails.
- **jQuery** es una biblioteca de JavaScript, 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 UI** es una biblioteca de componentes para el framework jQuery que le añaden un conjunto de plug-ins, widgets y efectos visuales para la creación de aplicaciones web. Cada componente o módulo se desarrolla de acuerdo a la filosofía de jQuery5 (find something, manipulate it: encuentra algo, manipúlalo).
- **Dojo** es un framework que contiene APIs y widgets (controles) para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, los efectos de UI, drag and drop APIs, widget APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX.

© JMA 2015. All rights reserved

# JavaScript Framework

- **Twitter Bootstrap** es un framework enfocado en el diseño adaptativo que permite construir sitios web rápidamente con estilos, plantillas y funciones predefinidas puestas a disposición del desarrollador.
- **AngularJS** es un framework de JavaScript de código abierto, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.
- **Backbone** es una herramienta de desarrollo/API para el lenguaje de programación Javascript con un interfaz RESTful por JSON , basada en el paradigma de diseño de aplicaciones Modelo Vista Controlador. Está diseñada para desarrollar aplicaciones de una única página y para mantener las diferentes partes de las aplicaciones web sincronizadas.
- **Ext JS** es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM.
- **Node.js** es un entorno de programación en la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.

© JMA 2015. All rights reserved



JavaScript Object Notation

<http://tools.ietf.org/html/rfc4627>

© JMA 2015. All rights reserved

## Introducción

- JSON (JavaScript Object Notation) es un formato sencillo para el intercambio de información.
- El formato JSON permite representar estructuras de datos (arrays) y objetos (arrays asociativos) en forma de texto.
- La notación de objetos mediante JSON es una de las características principales de JavaScript y es un mecanismo definido en los fundamentos básicos del lenguaje.
- En los últimos años, JSON se ha convertido en una alternativa al formato XML, ya que es más fácil de leer y escribir, además de ser mucho más conciso.
- No obstante, XML es superior técnicamente porque es un lenguaje de marcado, mientras que JSON es simplemente un formato para intercambiar datos.
- La especificación completa del JSON es la RFC 4627, su tipo MIME oficial es application/json y la extensión recomendada es .json.

© JMA 2015. All rights reserved

# Estructuras

- JSON está constituido por dos estructuras:
  - Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
  - Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.
- Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

© JMA 2015. All rights reserved

# Sintaxis

- Un array es un conjunto de valores separados por comas (,) que se encierran entre corchetes [ ... ]
- Un objeto es un conjunto de pares nombre:valor separados por comas (,) que se acotan entre llaves { ... }
- Los nombre son cadenas, entre comillas dobles (").
- El separador entre el nombre y el valor son los dos puntos (:)
- El valor debe ser un objeto, un array, un número, una cadena o uno de los tres nombres literales siguientes (en minúsculas):
  - true, false o null
- Se codifica en Unicode, la codificación predeterminada es UTF-8.

© JMA 2015. All rights reserved

## Valores numéricos

- La representación de números es similar a la utilizada en la mayoría de los lenguajes de programación.
- Un número contiene una parte entera que puede ser prefijada con un signo menos opcional, que puede ser seguida por una parte fraccionaria y / o una parte exponencial.
- La parte fraccionaria comienza con un punto (como separador decimal) seguido de uno o más dígitos.
- La parte exponencial comienza con la letra E en mayúsculas o minúsculas, lo que puede ser seguido por un signo más o menos, y son seguidas por uno o más dígitos.
- Los formatos octales y hexadecimales no están permitidos. Los ceros iniciales no están permitidos.
- No se permiten valores numéricos que no se puedan representar como secuencias de dígitos (como infinito y NaN).

© JMA 2015. All rights reserved

## Valores cadena

- La representación de las cadenas es similar a las convenciones utilizadas en la familia C de lenguajes de programación.
- Una cadena comienza y termina con comillas ("").
- Se pueden utilizar todos los caracteres Unicode dentro de las comillas con excepción de los caracteres que se deben escapar: los caracteres de control (U + 0000 a U + 001F) y los caracteres con significado.
- Cuando un carácter se encuentra fuera del plano multilingüe básico (U + 0000 a U + FFFF), puede ser representado por su correspondiente valor hexadecimal. Las letras hexadecimales A-F puede ir en mayúsculas o en minúsculas.
- Secuencias de escape:
  - \\, \/, \" , \n, \r, \b, \f, \t
  - \u[0-9A-Fa-f]{4}

© JMA 2015. All rights reserved

## Objeto con anidamientos

```
{
  "Image": {
    "Width": 800,
    "Height": 600,
    "Title": "View from 15th Floor",
    "Thumbnail": {
      "Url": "/image/481989943",
      "Height": 125,
      "Width": "100"
    },
    "IDs": [116, 943, 234, 38793]
  }
}
```

© JMA 2015. All rights reserved

## Array de objetos

```
[
  {
    "precision": "zip",
    "Latitude": 37.7668,
    "Longitude": -122.3959,
    "City": "SAN FRANCISCO",
    "State": "CA",
    "Zip": "94107"
  },
  {
    "precision": "zip",
    "Latitude": 37.371991,
    "Longitude": -122.026020,
    "City": "SUNNYVALE",
    "State": "CA",
    "Zip": "94085"
  }
]
```

© JMA 2015. All rights reserved



© JMA 2015. All rights reserved

## Introducción

- AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.
- Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. Ajax no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.
- JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.
- Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

© JMA 2015. All rights reserved

# XMLHttpRequest

- XMLHttpRequest (XHR), también conocido como XMLHTTP (Extensible Markup Language / Hypertext Transfer Protocol), es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores Web.
- Para los datos transferidos se usa cualquier codificación basada en texto, incluyendo: texto plano, XML, JSON, HTML y codificaciones particulares específicas.
- El navegador implementa la interfaz como una clase de la que una aplicación cliente puede generar tantas instancias como necesite y permita el navegador para manejar el diálogo con el servidor.
- La primera versión de la interfaz XMLHttpRequest fue desarrollada por Microsoft que la introdujo en la versión 5.0 de Internet Explorer utilizando un objeto ActiveX. A partir de la versión 7 la interfaz se ofrece de manera integrada.
- El proyecto Mozilla incorporó la primera implementación integrada en la versión 1.0 de la Suite Mozilla en 2002. Esta implementación sería seguida por Apple a partir de Safari 1.2, Opera Software a partir del Opera 8.0 e iCab desde la versión 3.0b352.
- El World Wide Web Consortium presentó el 27 de septiembre de 2006 el primer borrador de una especificación estándar de la interfaz.

© JMA 2015. All rights reserved

## Propiedades

Propiedad	Descripción	
readyState	0	No inicializado (objeto creado, pero no se ha invocado el método open)
	1	Cargando (objeto creado, pero no se ha invocado el método send)
	2	Cargado (se ha invocado el método send, pero el servidor aún no ha respondido)
	3	Interactivo (descargando, se han recibido algunos datos, aunque no se puede emplear la propiedad responseText)
	4	Completo (se han recibido todos los datos de la respuesta del servidor)
responseText	El contenido de la respuesta del servidor en forma de cadena de texto	
responseXML	El contenido de la respuesta del servidor en formato XML. El objeto devuelto se puede procesar como un objeto DOM	
status	El código de estado HTTP devuelto por el servidor (200 para una respuesta correcta, 404 para "No encontrado", 500 para un error de servidor, etc.)	
statusText	El código de estado HTTP devuelto por el servidor en forma de cadena de texto: "OK", "Not Found", "Internal Server Error", etc.	

© JMA 2015. All rights reserved

# Métodos y eventos

Método	Descripción
abort()	Detiene la petición actual.
getAllResponseHeaders()	Devuelve una cadena de texto con todas las cabeceras de la respuesta del servidor.
getResponseHeader("cabecera")	Devuelve una cadena de texto con el contenido de la cabecera solicitada.
open("metodo", "url")	Establece los parámetros de la petición que se realiza al servidor. Los parámetros necesarios son el método HTTP empleado y la URL destino.
send(contenido)	Realiza la petición HTTP al servidor
setRequestHeader("cabecera", "valor")	Permite establecer cabeceras personalizadas en la petición HTTP. Se debe invocar entre el open() y el send().
onreadystatechange	<b>Evento.</b> Se invoca cada vez que se produce un cambio en el estado de la petición HTTP.

© JMA 2015. All rights reserved

# Códigos HTTP (status)

status	statusText	Descripción
100	Continue	Una parte de la petición (normalmente la primera) se ha recibido sin problemas y se puede enviar el resto de la petición
101	Switching protocols	El servidor va a cambiar el protocolo con el que se envía la información de la respuesta. En la cabecera Upgrade indica el nuevo protocolo
200	OK	<b>La petición se ha recibido correctamente y se está enviando la respuesta. Este código es con mucha diferencia el que mas devuelven los servidores</b>
201	Created	Se ha creado un nuevo recurso (por ejemplo una página web o un archivo) como parte de la respuesta
202	Accepted	La petición se ha recibido correctamente y se va a responder, pero no de forma inmediata
203	Non-Authoritative Information	La respuesta que se envía la ha generado un servidor externo. A efectos prácticos, es muy parecido al código 200
204	No Content	La petición se ha recibido de forma correcta pero no es necesaria una respuesta
205	Reset Content	El servidor solicita al navegador que inicialice el documento desde el que se realizó la petición, como por ejemplo un formulario
206	Partial Content	La respuesta contiene sólo la parte concreta del documento que se ha solicitado en la petición

© JMA 2015. All rights reserved

# Códigos de redirección

status	statusText	Descripción
300	Multiple Choices	El contenido original ha cambiado de sitio y se devuelve una lista con varias direcciones alternativas en las que se puede encontrar el contenido
301	Moved Permanently	El contenido original ha cambiado de sitio y el servidor devuelve la nueva URL del contenido. La próxima vez que solicite el contenido, el navegador utiliza la nueva URL
302	Found	El contenido original ha cambiado de sitio de forma temporal. El servidor devuelve la nueva URL, pero el navegador debe seguir utilizando la URL original en las próximas peticiones
303	See Other	El contenido solicitado se puede obtener en la URL alternativa devuelta por el servidor. Este código no implica que el contenido original ha cambiado de sitio
304	Not Modified	Normalmente, el navegador guarda en su caché los contenidos accedidos frecuentemente. Cuando el navegador solicita esos contenidos, incluye la condición de que no hayan cambiado desde la última vez que los recibió. Si el contenido no ha cambiado, el servidor devuelve este código para indicar que la respuesta sería la misma que la última vez
305	Use Proxy	El recurso solicitado sólo se puede obtener a través de un proxy, cuyos datos se incluyen en la respuesta
307	Temporary Redirect	Se trata de un código muy similar al 302, ya que indica que el recurso solicitado se encuentra de forma temporal en otra URL

© JMA 2015. All rights reserved

# Códigos de error del navegador

status	statusText	Descripción
400	Bad Request	El servidor no entiende la petición porque no ha sido creada de forma correcta
401	Unauthorized	El recurso solicitado requiere autorización previa
402	Payment Required	Código reservado para su uso futuro
403	Forbidden	No se puede acceder al recurso solicitado por falta de permisos o porque el usuario y contraseña indicados no son correctos
404	Not Found	El recurso solicitado no se encuentra en la URL indicada. Se trata de uno de los códigos más utilizados y responsable de los típicos errores de <i>Página no encontrada</i>
405	Method Not Allowed	El servidor no permite el uso del método utilizado por la petición, por ejemplo por utilizar el método GET cuando el servidor sólo permite el método POST
406	Not Acceptable	El tipo de contenido solicitado por el navegador no se encuentra entre la lista de tipos de contenidos que admite, por lo que no se envía en la respuesta
407	Proxy Authentication Required	Similar al código 401, indica que el navegador debe obtener autorización del proxy antes de que se le pueda enviar el contenido solicitado
408	Request Timeout	El navegador ha tardado demasiado tiempo en realizar la petición, por lo que el servidor la descarta

© JMA 2015. All rights reserved

## Códigos de error del navegador

status	statusText	Descripción
409	Conflict	El navegador no puede procesar la petición, ya que implica realizar una operación no permitida (como por ejemplo crear, modificar o borrar un archivo)
410	Gone	Similar al código 404. Indica que el recurso solicitado ha cambiado para siempre su localización, pero no se proporciona su nueva URL
411	Length Required	El servidor no procesa la petición porque no se ha indicado de forma explícita el tamaño del contenido de la petición
412	Precondition Failed	No se cumple una de las condiciones bajo las que se realizó la petición
413	Request Entity Too Large	La petición incluye más datos de los que el servidor es capaz de procesar. Normalmente este error se produce cuando se adjunta en la petición un archivo con un tamaño demasiado grande
414	Request-URI Too Long	La URL de la petición es demasiado grande, como cuando se incluyen más de 512 bytes en una petición realizada con el método GET
415	Unsupported Media Type	Al menos una parte de la petición incluye un formato que el servidor no es capaz de procesar
416	Requested Range Not Suitable	El trozo de documento solicitado no está disponible, como por ejemplo cuando se solicitan bytes que están por encima del tamaño total del contenido
417	Expectation Failed	El servidor no puede procesar la petición porque al menos uno de los valores incluidos en la cabecera Expect no se pueden cumplir

© JMA 2015. All rights reserved

## Códigos de error del servidor

status	statusText	Descripción
500	Internal Server Error	Se ha producido algún error en el servidor que impide procesar la petición
501	Not Implemented	Procesar la respuesta requiere ciertas características no soportadas por el servidor
502	Bad Gateway	El servidor está actuando de proxy entre el navegador y un servidor externo del que ha obtenido una respuesta no válida
503	Service Unavailable	El servidor está sobrecargado de peticiones y no puede procesar la petición realizada
504	Gateway Timeout	El servidor está actuando de proxy entre el navegador y un servidor externo que ha tardado demasiado tiempo en responder
505	HTTP Version Not Supported	El servidor no es capaz de procesar la versión HTTP utilizada en la petición. La respuesta indica las versiones de HTTP que soporta el servidor

© JMA 2015. All rights reserved

## Pasos a seguir

1. Obtener XMLHttpRequest
2. Crear y asignar el controlador del evento onreadystatechange.
3. Abrir conexión: open(method, url, *async*):
4. Opcional. Añadir cabeceras.
5. Opcional. Serializar datos a enviar vía POST.
6. Enviar petición: send()

© JMA 2015. All rights reserved

## Obtener XMLHttpRequest

- Los navegadores que siguen los estándares (Firefox, Safari, Opera, Internet Explorer 7+) implementan el objeto XMLHttpRequest de forma nativa, por lo que se puede obtener a través del objeto window. Los navegadores obsoletos (Internet Explorer 5 y 6) implementan el objeto XMLHttpRequest como un objeto de tipo ActiveX.

```
var xmlhttp;
if (window.XMLHttpRequest) {
    //El explorador implementa la interfaz de forma nativa
    xmlhttp = new XMLHttpRequest();
} else if (window.ActiveXObject) {
    //El explorador permite crear objetos ActiveX
    try {
        xmlhttp = new ActiveXObject("MSXML2.XMLHTTP");
    } catch (e) {
        try {
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {}
    }
}
if (!xmlhttp) {
    alert("No ha sido posible crear una instancia de XMLHttpRequest");
}
```

© JMA 2015. All rights reserved

## Controlador del evento onreadystatechange

- readyState: cuando vale 4 (Completo: se han recibido todos los datos de la respuesta del servidor)
- status: cuando vale 200 (OK: La petición se ha recibido correctamente y se está enviando la respuesta)
- responseText: El contenido de la respuesta del servidor en forma de cadena de texto
- responseXML: El contenido de la respuesta del servidor en formato XML.

```
xmlhttp.onreadystatechange = function () {
    if (xmlhttp.readyState == 4)
        if (xmlhttp.status == 200) {
            xmlDoc = xmlhttp.responseXML;
            // Tratamiento de los datos recibidos
        } else {
            // Tratamiento de excepción
        }
};
```

© JMA 2015. All rights reserved

## Enviar petición

- Abrir conexión: open(method, url, async):
  - method: Verbo HTTP (GET, POST, ...)
  - async: Opcional, marcar con false para comportamiento síncrono

```
xmlhttp.open("GET", "demo_get.asp?nocache=" + Math.random());
xmlhttp.open("POST", "demo_post.asp");
```
- Opcional. Añadir cabeceras.
 

```
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
```
- Opcional. Serializar datos a enviar vía POST.
  - Reunir los datos a enviar en una cadena formada por pares “nombre=valor” concatenados con ampersand (&).

```
var nombre= document.getElementById("fldNombre");
var datos="nombre="+encodeURIComponent(nombre)
+"&apellidos="+encodeURIComponent(apellidos);
```
- Enviar petición: send()
 

```
xmlhttp.send(datos);
```

© JMA 2015. All rights reserved

## Texto plano, HTML y JavaScript

- Texto plano

```
var rs1t = http_request.responseText;
document.getElementById("myDiv").innerHTML=rs1t;
```

- HTML

```
var rs1t = http_request.responseText;
document.getElementById("myDiv").innerHTML=rs1t;
```

- JavaScript

```
var respuesta = http_request.responseText;
eval(respuesta);
```

© JMA 2015. All rights reserved

## XML

- Serializar:

```
var datos=<Datos>;
datos+="<>Nombre>" + nombre + "</Nombre>";
datos+="<>Apellidos>" + apellidos + "</Apellidos>";
datos+="</Datos>;
```

- Recibir:

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("Provincias");
for (i=0;i<x.length;i++) {
    txt=txt + x[i].childNodes[0].nodeValue + "<br>";
}
document.getElementById("myDiv").innerHTML=txt;
```

© JMA 2015. All rights reserved

# JSON

- Serializar:

```
var datos=JSON.stringify(objeto_json);
var datos='{"Datos": {';
datos+="Nombre":'+nombre+'';
datos+=",Apellidos":'+apellidos+'';
datos+='}}';
```

- Recibir:

```
var respuesta = http_request.responseText;
var objeto_json= JSON.parse(respuesta).Provincias;
//var objeto_json = eval("( "+respuesta+" )").Provincias;
txt="";
for (i=0;i<objeto_json.length;i++) {
    txt=txt + objeto_json[i].Nombre + "<br>";
}
document.getElementById("myDiv").innerHTML=txt;
```

© JMA 2015. All rights reserved

## Detener las peticiones

```
...
// Fijar un temporizador que aborte la petición
var temporizador = setTimeout(function() {
    xmlhttp.abort();
    alert(...);
}, 18000); // 2 minutos
xmlhttp.onreadystatechange = function () {
    if (xmlhttp.readyState == 4) {
        clearTimeout(temporizador);
        if (xmlhttp.status == 200) {
            // Eliminar el temporizador, innecesario
    ...
}
```

© JMA 2015. All rights reserved

# Indicador de descarga

```

...
var temporizador = setTimeout(function() {
    document.getElementById("trabajandoAJAX").style.display="none";
    xmlhttp.abort();
    alert(...);
}, 18000); // 30 segundos
// Muestra el indicador hasta ahora oculto
document.getElementById("trabajandoAJAX").style.display = "block";
xmlhttp.onreadystatechange = function () {
    if (xmlhttp.readyState == 4)
        try {
            if (xmlhttp.status == 200) {
                ...
            } finally {
                // Oculta el indicador
                document.getElementById("trabajandoAJAX").style.display="none";
            }
        ...
}
...

```

© JMA 2015. All rights reserved

# Seguridad

- La ejecución de aplicaciones JavaScript puede suponer un riesgo para el usuario que permite su ejecución.
- Por este motivo, los navegadores restringen la ejecución de todo código JavaScript a un entorno de ejecución limitado.
- Las aplicaciones JavaScript no pueden establecer conexiones de red con dominios distintos al dominio en el que se aloja la aplicación JavaScript.
- Los navegadores emplean un método estricto para diferenciar entre dos dominios ya que no permiten ni subdominios ni otros protocolos ni otros puertos.
- Si el código JavaScript se descarga desde la siguiente URL:  
<http://www.ejemplo.com>
- Las funciones y métodos incluidos en ese código no pueden acceder a:
  - <https://www.ejemplo.com/scripts/codigo2.js>
  - <http://www.ejemplo.com:8080/scripts/codigo2.js>
  - <http://scripts.ejemplo.com/codigo2.js>
  - <http://192.168.0.1/scripts/codigo2.js>
- La propiedad `document.domain` se emplea para permitir el acceso entre subdominios del dominio principal de la aplicación.

© JMA 2015. All rights reserved

## JSONP (JSON con padding)

- JSONP es una técnica de comunicación utilizada en los programas JavaScript para realizar llamadas asíncronas a dominios diferentes. JSONP es un método concebido para superar la limitación de AJAX entre dominios por razones de seguridad. Esta restricción no se aplica a la etiqueta <script> de HTML, para la cual se puede especificar en su atributo src la URL de un script alojado en un servidor remoto.
- En esta técnica se devuelve un objeto JSON envuelto en la llamada de una función (debe ser código JavaScript válido), la función ya debe estar definida en el entorno de JavaScript y se encarga de manipular los datos JSON.
- Por convención, el nombre de la función de retorno se suele especificar mediante un parámetro de la consulta, normalmente, utilizando jsonp o callback como nombre del campo en la solicitud al servidor.

```
<script type="text/javascript"
src="http://otrodominio.com/datos.json?callback=
miJsonCallback"></script>
```

© JMA 2015. All rights reserved

## Insertar elemento script

- El uso de JSONP sólo tiene sentido si se quiere realizar una llamada asíncrona a otro dominio, por ello es necesario manipular el DOM para insertar un elemento <script> en la cabecera de la página, ya que una vez cargado el documento, no es posible escribir en él.

```
function loadScript (id, src, callback) {
    // Crear elemento
    var script = document.createElement("script");
    // Atributos del script
    script.setAttribute("type", "text/javascript");
    script.setAttribute("src", src + "?callback=" +
        callback);
    script.setAttribute("id", id);
    // Insertar script en la cabecera
    document.getElementsByTagName("head")[0]
        .appendChild(script);
}
```

© JMA 2015. All rights reserved

## REST (REpresentational State Transfer)

- Un **estilo de arquitectura** para desarrollar aplicaciones web distribuidas que se basa en el uso del protocolo HTTP e Hypermedia.
- Definido en el 2000 por Roy Fielding, para no reinventar la rueda, se basa en aprovechar lo que ya estaba definido en el HTTP pero que no se utilizaba.
- El HTTP ya define 8 métodos (algunas veces referido como "verbos") que indica la acción que desea que se efectúe sobre el recurso identificado:
  - HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT
- El HTTP permite en el encabezado transmitir la información de comportamiento:
  - Accept, Content-type, Response (códigos de estado), Authorization, Cache-control, ...

© JMA 2015. All rights reserved

## Uso de la cabecera

- Request: Método /uri?parámetros
  - GET: Recupera el recurso
    - Todos: Sin parámetros
    - Uno: Con parámetros
  - POST: Crea un nuevo recurso
  - PUT: Edita el recurso
  - DELETE: Elimina el recurso
- Accept: Indica al servidor el formato o posibles formatos esperados, utilizando MIME.
- Content-type: Indica en que formato está codificado el cuerpo, utilizando MIME
- Response: Código de estado con el que el servidor informa del resultado de la petición.

© JMA 2015. All rights reserved

# Peticiones

```

Request: GET /users
Response: 200
content-type:application/json
Request: GET /users/11
Response: 200
content-type:application/json
Request: POST /users
Response: 201 Created
content-type:application/json
body
Request: PUT /users/11
Response: 200
content-type:application/json
body
Request: DELETE /users/11
Response: 204 no content

```

© JMA 2015. All rights reserved

# CORS

- Un recurso hace una solicitud HTTP de origen cruzado cuando solicita otro recurso de un dominio distinto al que pertenece y, por razones de seguridad, los exploradores restringen las solicitudes HTTP de origen cruzado iniciadas dentro de un script.
- XMLHttpRequest sigue la política de mismo-origen, por lo que, una aplicación usando XMLHttpRequest solo puede hacer solicitudes HTTP a su propio dominio. Para mejorar las aplicaciones web, los desarrolladores pidieron a los proveedores de navegadores que permitieran a XMLHttpRequest realizar solicitudes de dominio cruzado.
- El Grupo de Trabajo de Aplicaciones Web del W3C recomienda el nuevo mecanismo de Intercambio de Recursos de Origen Cruzado (CORS, Cross-origin resource sharing). Los servidores deben indicar al navegador mediante cabeceras si aceptan peticiones cruzadas y con qué características:
  - "Access-Control-Allow-Origin", "\*"
  - "Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept"
  - "Access-Control-Allow-Methods", "GET, POST, PUT, DELETE"
- Soporte: Chrome 3+ Firefox 3.5+ Opera 12+ Safari 4+ Internet Explorer 8+

© JMA 2015. All rights reserved

# JWT: JSON Web Tokens

<https://jwt.io>

- JSON Web Token (JWT) es un estándar abierto (RFC-7519) basado en JSON para crear un token que sirva para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros.
- El caso más común de uso de los JWT es para manejar la autenticación en aplicaciones móviles o web. Para esto cuando el usuario se quiere autenticar manda sus datos de inicio del sesión al servidor, este genera el JWT y se lo manda a la aplicación cliente, posteriormente en cada petición el cliente envía este token que el servidor usa para verificar que el usuario esté correctamente autenticado y saber quién es.
- También es posible usarlo para transferir cualquier datos entre servicios de nuestra aplicación y asegurarnos de que sean siempre válido. Por ejemplo si tenemos un servicio de envío de email otro servicio podría enviar una petición con un JWT junto al contenido del mail o cualquier otro dato necesario y que estemos seguros que esos datos no fueron alterados de ninguna forma.

© JMA 2015. All rights reserved

# XMLHttpRequest Level 2

- En las nuevas API definidas con HTML5 se encuentra el API Communication con la especificación del XMLHttpRequest Level 2.
- El nivel 2 de XMLHttpRequest incorpora nuevas características como comunicación con múltiples orígenes, nuevos eventos para controlar la evolución de las solicitudes, nuevos tipos de respuestas, subidas de ficheros, ...
- Estas mejoras simplifican códigos y ofrecen nuevas opciones, como interactuar con diferentes servidores desde la misma aplicación o trabajar con pequeñas trozos de datos en lugar de archivos enteros.

© JMA 2015. All rights reserved

## Nuevos eventos

Evento	Descripción
loadstart	Este evento es disparado cuando la solicitud comienza.
progress	Este evento es disparado periódicamente mientras se envían o descargan datos.
abort	Este evento es disparado cuando la solicitud es abortada.
error	Este evento es disparado cuando un error ocurre durante el procesamiento de la solicitud.
load	Este evento es disparado cuando la solicitud ha sido completada.
timeout	Si un valor para timeout ha sido especificado, este evento será disparado cuando la solicitud no pueda ser completada en el período de tiempo determinado.
loadend	Este evento es disparado cuando la solicitud ha sido completada (sin considerar si el proceso fue exitoso o no).

© JMA 2015. All rights reserved

## Especificación de un formato de respuesta

- **xhr.responseType**
  - Antes de enviar una solicitud, se establece xhr.responseType en "arraybuffer", "blob", "document", "json" o "text", en función de los datos que se necesiten.
  - Si se establece xhr.responseType = "" o si se omite, se utilizará la respuesta predeterminada "text".
- **xhr.response**
  - Después de una solicitud correcta, la propiedad response de xhr contendrá los datos solicitados como DOMString, ArrayBuffer, Blob o Document (en función del valor establecido para responseType).

© JMA 2015. All rights reserved

# Novedades en la petición

- Nuevos atributos:
  - timeout: Se puede establecer en un tiempo en milisegundos. Cuando se establece en un valor distinto de cero provocará que la recuperación finalice después de que haya pasado el tiempo dado.
  - withCredentials: A true cuando las credenciales del usuario deben incluirse en una solicitud de origen cruzado. A false (por defecto) cuando deben excluirse en una solicitud de origen cruzado y cuando se deben ignorar las cookies en su respuesta.
  - upload: Devuelve el XMLHttpRequestUpload objeto asociado . Se puede usar para recopilar información de transmisión cuando los datos se transfieren a un servidor.
- Nuevos formatos de datos para el método send:
  - ArrayBufferView, Blob, Document, ScalarValueString o FormData

© JMA 2015. All rights reserved

## FormData

- Representa la colección de pares nombre/valor. Esta interface sencilla tiene solo un constructor y un método con el que obtener y trabajar sobre objetos FormData.
- **FormData(*form*)**
  - Este constructor retorna una objeto FormData usado luego por el método send() para enviar información.
  - Si se le pasa un formulario como parámetro rellena la colección con los datos del formulario, si no devuelve la colección vacía.
- **append(nombre, valor)**
  - Este método agrega datos al objeto FormData. Toma un par clave/valor como parámetros. El parámetro valor puede ser una cadena de texto o un blob (son objetos inmutables que representan los datos en bruto).

© JMA 2015. All rights reserved



<http://jquery.com/>

© JMA 2015. All rights reserved

## INTRODUCCIÓN

© JMA 2015. All rights reserved

# Introducción

- jQuery es una biblioteca de JavaScript que 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 a través de una multitud de navegadores..
- Creada inicialmente por John Resig, fue presentada el 14 de enero de 2006 en el BarCamp NYC y se ha convertido en la biblioteca de JavaScript más utilizada.
- jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados.
- jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.
- Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+.

© JMA 2015. All rights reserved

# Características

- Selección de elementos DOM.
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3 y un plugin básico de XPath.
- Manipulación de la hoja de estilos CSS.
- Eventos.
- Efectos y animaciones.
- Animaciones personalizadas.
- AJAX.
- Soporta extensiones.
- Utilidades varias para obtener información del navegador, operar con objetos y vectores, funciones para rutinas comunes, etc.

© JMA 2015. All rights reserved

# Añadir jQuery a una página

- Se pueden utilizar dos estrategias diferentes:
  - Incluir ficheros locales.
  - Incluir ficheros compartidos en una CDN (Content Delivery Network).
- Se pueden descargar los ficheros locales desde <http://jquery.com/download/>:
  - Versión de producción: para servidores web se encuentra minimizada y compactada para reducir al máximo su tamaño.
  - Versión de desarrollo: para desarrollo y pruebas sin comprimir y depurable.

```
<head>
<script type="text/javascript" src="/js/jquery-1.11.2.min.js"></script>
</head>
```
- Las CDN permiten compartir contenidos comunes entre diferentes sitios y evitar descargas al aprovechar la cache de los navegadores:
 

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js">
</script>
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-
1.11.2.min.js"> </script>
```
- Es conveniente incluirlos después de las hojas de estilos para asegurar la importación de todos los estilos.

© JMA 2015. All rights reserved

## Sintaxis

- El acceso a la funcionalidad del jQuery se realiza a través de la función `jQuery()` o su acceso directo `$()`.
- La sintaxis de jQuery está hecha a medida para la selección de los elementos HTML y realizar alguna acción sobre dichos elementos.
- Con jQuery se pueden encadenar acciones/métodos, permitiendo ejecutar múltiples métodos de jQuery (en el mismo elemento) en una sola sentencia.
- La sintaxis básica es:
 

```
$(selector).acción().otra().otraMas()
```
- El método `ready` permite asignar la función que se quiere ejecutar cuando el documento haya terminado de cargarse y esté listo para trabajar con él. Esto permite tener el código JavaScript antes que el cuerpo del documento, en la sección de la cabeza.
 

```
$(document).ready(function(){ ... });
$(function () { ... });
```

© JMA 2015. All rights reserved

# Resolución de conflictos

- Hay muchos otros frameworks populares en JavaScript como: Angular, Backbone, Ember, Knockout, ...
- Si dos marcos diferentes están utilizando el mismo método abreviado \$, uno de ellos puede dejar de funcionar.
- El método noConflict () libera el identificador de acceso directo \$, por lo que otros scripts pueden usarlo.
- El método noConflict () devuelve una referencia a jQuery, que se puede guardar en una variable para su uso posterior como atajo propio.

```
var jq = $.noConflict();
jq(document).ready(function(){
```
- Si se tiene un bloque de código jQuery que utiliza los atajos \$ y no se quiere cambiarlo todo, el signo \$ se puede pasar como parámetro al método. Esto permite acceder a jQuery usando \$ dentro de la función aunque fuera de ella se tendrá que usar "jQuery":

```
$.noConflict();
jQuery(document).ready(function($){
  $("button").click(function(){
```

© JMA 2015. All rights reserved

# SELECCIÓN DE ELEMENTOS

© JMA 2015. All rights reserved

# Introducción

- Los selectores de jQuery se utilizan para "encontrar" (o seleccionar) elementos HTML en función a identificadores, clases, tipos, atributos, valores de atributos y mucho más.
- La biblioteca soporta gran parte de los selectores CSS3 y varios selectores personalizados propios. En <http://api.jquery.com/category/selectors> se puede encontrar una completa referencia sobre los selectores de la biblioteca.
- Es importante tener en cuenta que el resultado de la selección siempre es múltiple, devuelve todos los elementos que cumplan la condición y las acciones se aplican cada uno de los elementos devueltos.
- Como el número de elementos devueltos puede ser: 0 .. N, si es necesario se puede consultar con la propiedad length.  

```
if ($('#div').length) { ... }
```
- El proceso de selección es costoso y puede requerir la ejecución de gran cantidad de código, por lo que es conveniente encadenar acciones o almacenar el resultado en variables cuando se vayan a realizar varias operaciones sobre el resultado.  

```
var $divs = $('#div');
```

© JMA 2015. All rights reserved

# Selectores

- Selección de todos los elementos  
`'*'`
- Selección de elementos en base al tipo de elemento  
`'etiqueta'`
- Selección de elementos en base al nombre de clase  
`'.miClass'`  
`'etiqueta.miClass'`
- Selección de elementos en base a su id  
`'#valorId';`
- Selección de elemento actual  
`this`
- Selección múltiple  
`'selector1, selector2, selectorN';`

© JMA 2015. All rights reserved

## Selectores jerárquicos

- Selecciona todos los elementos hijos directos de un elemento padre.  
`$( 'padre > hijo' );`
- Selecciona todos los elementos descendientes de un elemento antecesor.  
`$( 'antecesor descendant' );`
- Selecciona todos los elementos que están inmediatamente después del elemento dado (solo el primero).  
`$( 'elemento + siguiente' );`
- Selecciona todos los hermanos que están inmediatamente después del elemento dado (solo el primero).  
`$( 'elemento ~ siguiente' );`
- Selección de elementos con selectores CSS  
`$( '#contents ul.people li' );`

© JMA 2015. All rights reserved

## Selectores de atributos

- Elementos que tienen un atributo  
`$( '[atributo]' );`  
`$( 'etiqueta[atributo]' );`
- Seleccionar por el valor del atributo (secuencias de escape: \' \")  
`$( '[atributo="valor"]' );`
- Operadores de comparación (es sensible a mayúsculas y minúsculas):
  - = El valor es un valor determinado.
  - != El valor no es un valor determinado o no tiene el atributo.
  - \*= Un valor que contiene una subcadena determinada.
  - ~= Un valor que contiene una palabra dada, delimitada por espacios.
  - ^= Un valor que comienza con una subcadena determinada.
  - |= Un valor ya sea igual a la cadena indicada o comienza con la cadena seguida de un guion (-).
  - \$= Un valor que termina con una subcadena determinada.
- Seleccionar por el valor de varios atributos  
`$( '[atributo1="valor"][atributo2="valor"]' );`

© JMA 2015. All rights reserved

## Filtros por posicionamiento

Filtro	Descripción
:first	primer elemento coincidente.
:last	último elemento coincidente.
:eq()	elemento en el índice dado.
:gt()	elementos cuyo índice es superior al índice dado.
:lt()	elementos cuyo índice es inferior al índice dado.
:even	elementos impares, en base cero.
:odd	elementos impares, en base cero.
:first-child	elementos que son el primer hijo de su padre.
:first-of-type	elementos que son los primeros entre hermanos del mismo nombre del elemento.
:last-child	elementos que son el último hijo de su padre.
:last-of-type	elementos que son el último entre los hermanos del mismo nombre del elemento.
:only-child	elementos que son el único hijo de su padre.
:only-of-type	elementos que no tienen hermanos con el mismo nombre del elemento.

© JMA 2015. All rights reserved

## Filtros básicos y de estilo

Filtro	Descripción
:not()	elementos que no coinciden con el selector dado.
:focus	elemento que tiene foco.
:header	elementos que son encabezados, como h1, h2, h3 y así sucesivamente.
:contains()	elementos que contienen el texto especificado.
:empty	elementos que no tienen hijos (incluidos los nodos de texto).
:has()	elementos que contienen al menos un elemento que coincide con el selector especificado.
:root	elemento que es la raíz del documento.
:parent	elementos que tienen al menos un nodo hijo (ya sea un elemento o texto).
:hidden	elementos que están ocultos.
:visible	elementos que son visibles.
:animated	elementos que están en ejecutando una animación en el momento en el que selector se ejecuta.
:target	elemento de destino indicado por el identificador de fragmento de URI del documento.
:lang()	elementos del idioma especificado.

© JMA 2015. All rights reserved

# Filtros para formularios

Filtro	Descripción
:input	elementos de entrada, área de texto, seleccione y botones.
:button	elementos de los botones y elementos de tipo botón.
:submit	elementos del tipo botón submit.
:reset	elementos del tipo botón reset.
:image	elementos de tipo imagen.
:checkbox	elementos de tipo checkbox.
:radio	elementos de tipo de radiobutton.
:checked	elementos que están marcados o seleccionados.
:text	elementos de entrada de tipo texto.
:password	elementos de tipo contraseña.
:file	elementos del tipo de archivo.
:selected	elementos que están seleccionados.
:disabled	elementos que están deshabilitados.
:enabled	elementos que están habilitados.

© JMA 2015. All rights reserved

## Filtrado del resultado de la selección

- Una vez realizada la selección de los elementos, es posible utilizarlos en conjunto con diferentes métodos.
  - Si en una selección se realiza una llamada a un método, y éste devuelve un objeto jQuery, es posible seguir un "encadenado" de métodos en el objeto.
- ```
$('#content')
    .find('h3')
    .eq(2)
    .html('nuevo texto para el tercer elemento h3');
```

- Si desea volver a la selección original en el medio del encadenado, jQuery ofrece el método end() para poder hacerlo.

```
$('#content')
    .find('h3').eq(2)
    .html('nuevo texto para el tercer elemento h3')
    .end() // reestablece la selección
    .first().html('nuevo texto para el primer elemento h3');
```

© JMA 2015. All rights reserved

# Métodos de filtrado

- `.filter():` Reducir el conjunto de elementos coincidentes a los que coinciden con el selector o pasan la prueba de la función.
- `.first():` Reducir el conjunto de elementos emparejados a la primera en el conjunto.
- `.last():` Reducir el conjunto de elementos coincidentes a la final en el conjunto.
- `.eq():` Reducir el conjunto de elementos coincidentes a la que está en el índice especificado.
- `.has():` Reducir el conjunto de elementos coincidentes a los que tienen un descendiente que coincide con el selector o elemento DOM.
- `.is():` Compruebe el actual conjunto combinado de elementos contra un selector, elemento u objeto jQuery y devolver true si al menos uno de estos elementos partidos los argumentos dados.
- `.map():` Pase cada elemento en el conjunto combinado corriente a través de una función, la producción de un nuevo objeto jQuery que contiene los valores de retorno.
- `.slice():` Reducir el conjunto de elementos emparejados a un subconjunto especificado por una gama de índices.
- `.contents():` Obtener los hijos de cada elemento en el conjunto de los elementos coincidentes, incluyendo texto y nodos comentario.

© JMA 2015. All rights reserved

# Métodos de filtrado

- `.find():` Obtener los descendientes de cada elemento en el conjunto actual de los elementos coincidentes, filtrados por un selector, un objeto jQuery o un elemento.
- `.children():` Obtener los hijos de cada elemento en el conjunto de elementos emparejados, opcionalmente filtrada por un selector.
- `.closest():` Para cada elemento en el conjunto, obtener el primer elemento que coincide con el selector probando el elemento en sí y que atraviesa a través de sus ancestros en el árbol DOM.
- `.parent():` Obtener el padre de cada elemento en el conjunto actual de los elementos coincidentes, opcionalmente filtrado por un selector.
- `.offsetParent():` Obtener el elemento antecesor más cercano que se coloca.
- `.parents():` Obtener los antepasados de cada elemento en el conjunto actual de los elementos coincidentes, opcionalmente filtrados por un selector.
- `.parentsUntil():` Obtener los antepasados de cada elemento en el conjunto actual de los elementos coincidentes, hasta, pero no incluyendo el elemento emparejado por el selector, el nodo DOM, o un objeto jQuery.
- `.siblings():` Obtener los hermanos de cada elemento en el conjunto de elementos emparejados, opcionalmente filtrada por un selector.

© JMA 2015. All rights reserved

# Métodos de filtrado

- `.prev():` Obtener el hermano inmediatamente anterior de cada elemento en el conjunto de elementos emparejados, opcionalmente filtrada por un selector.
- `.prevAll():` Obtener todos los hermanos anteriores de cada elemento en el conjunto de elementos emparejados, opcionalmente filtrada por un selector.
- `.prevUntil():` Obtén todos los hermanos anteriores de cada elemento hasta, pero sin incluir el elemento emparejado por el selector, el nodo DOM, o un objeto jQuery.
- `.next():` Obtener el hermano inmediatamente después de cada elemento en el conjunto de elementos emparejados. Si se proporciona un selector, recupera el siguiente hermano sólo si coincide con el selector.
- `.nextAll():` Obtener todos los siguientes hermanos de cada elemento en el conjunto de elementos emparejados, opcionalmente filtrado por un selector.
- `.nextUntil():` Obtener todos los siguientes hermanos de cada elemento hasta, pero sin incluir el elemento emparejado por el selector, el nodo DOM, o un objeto jQuery pasado.
- `.add():` Crear un nuevo objeto jQuery con elementos añadidos al conjunto de elementos coincidentes.
- `.addBack():` Añadir el conjunto anterior de elementos en la pila para el conjunto actual, opcionalmente filtrada por un selector.
- `.end():` Poner fin a la operación de filtrado más reciente en la cadena actual y devolver el conjunto de elementos coincidentes a su estado anterior.
- `.not():` Eliminar elementos del conjunto de elementos coincidentes.

© JMA 2015. All rights reserved

## EVENTOS

© JMA 2015. All rights reserved

# Introducción

- jQuery provee métodos para asociar controladores de eventos (en inglés event handlers) a selectores.
- Cuando un evento ocurre, la función provista es ejecutada.
- Dentro de la función, la palabra clave this hace referencia al elemento en que el evento ocurre.
- Para más detalles sobre los eventos en jQuery, se puede consultar <http://api.jquery.com/category/events>.
- La función del controlador de eventos puede recibir un objeto.
- Este objeto puede ser utilizado para determinar la naturaleza del evento o, por ejemplo, prevenir el comportamiento predeterminado de éste.
- Para más detalles sobre el objeto del evento, se puede consultar <http://api.jquery.com/category/events/event-object>.

© JMA 2015. All rights reserved

# Vinculación de eventos

- jQuery ofrece métodos para la mayoría de los eventos — entre ellos \$.fn.click, \$.fn.focus, \$.fn.blur, \$.fn.change, etc. Estos últimos son formas reducidas del método \$.fn.bind de jQuery.
- El método on es útil para vincular la misma función de controlador a múltiples eventos, para cuando se desea proveer información al controlador de evento, cuando se está trabajando con eventos personalizados o cuando se desea pasar un objeto a múltiples eventos y controladores.
- Vincular un evento utilizando un método reducido  
`$( 'p' ).click(function() { ... });`
- Vincular un evento utilizando el método \$.fn.on  
`$( 'p' ).on('click', function() { ... });`
- Vincular varios eventos al mismo controlador  
`$( 'p' ).on('click change', function() { ... });`

© JMA 2015. All rights reserved

# Vinculación de eventos

- Vincular un evento con información asociada

```
$('input').on(
    'click change',
    // se debe pasar la información asociada como argumento
    { foo : 'bar' },
    function(eventObject) {
        console.log(eventObject.type, eventObject.data);
        // registra el tipo de evento y
        // la información asociada { foo : 'bar' }
    });
});
```

- Vincular múltiples eventos a un elemento

```
$('p').on({
    'click': function() { ... },
    'mouseover': function() { ... }
});
```

© JMA 2015. All rights reserved

# Vinculación de eventos

- Vincular eventos para ejecutar una vez

```
$(p).one('click', function() { ... });
```

- Vincular varios controladores al mismo evento

```
var controlador1 = function() { ... };
var controlador2 = function() { ... };
$('p')
    .on('click', controlador1)
    .on('click', controlador2);
```

- Desvincular todos los controladores del evento click en una selección

```
$(p).off('click');
```

- Desvincular un controlador particular del evento click

```
$(p).off('click', controlador1);
```

© JMA 2015. All rights reserved

# Métodos de eventos

| Método        | Descripción   |
|---------------|---|
| .ready()      | Especificar la función que se ejecuta cuando el DOM está totalmente cargado.  |
| .resize()     | Enlazar un controlador de eventos para el evento "redimensionar".   |
| .scroll()     | Enlazar un controlador de eventos para el evento "scroll".  |
| .trigger()    | Ejecutar todos los manipuladores y comportamientos vinculados a los elementos coincidentes para el tipo de evento dado.   |
| .bind()       | Adjuntar un manejador a un evento para los elementos.   |
| .unbind()     | Retirar un controlador de eventos previamente conectado a los elementos.  |
| .on()         | Adjuntar una función de controlador de eventos para uno o más eventos a los elementos.  |
| .off()        | Eliminar un controlador de eventos.   |
| .delegate()   | Adjuntar un manejador de uno o más eventos para todos los elementos que coinciden con el selector, ahora o en el futuro, sobre la base de un conjunto específico de elementos raíz. |
| .undelegate() | Retirar un manejador del evento para todos los elementos que coinciden con el selector actual, en base a un conjunto específico de elementos raíz.                                  |
| .one()        | Adjuntar un manejador a un evento para los elementos. El manejador se ejecuta como máximo una vez por elemento según el tipo de evento.   |

© JMA 2015. All rights reserved

# Eventos específicos

| Método         | Descripción  |
|----------------|--|
| .change()      | Se lanza cuando cambia el valor de un <input>, <textarea> y <select>.  |
| .focus()       | Se lanza cuando un elemento recibe el foco.  |
| .focusin()     | Se lanza cuando un elemento recibe el foco (soporta la propagación).   |
| .blur()        | Se lanza cuando un elemento pierde el foco.  |
| .focusout()    | Se lanza cuando un elemento pierde el foco (soporta la propagación).   |
| .select()      | Se lanza cuando el usuario realiza una selección de texto en un <input type = "text"> o <textarea>.                                    |
| .submit()      | Se lanza cuando el usuario intenta enviar el formulario, solo para <form>.   |
| <b>Teclado</b> | .keydown(), .keypress(), .keyup()  |
| <b>Ratón</b>   | .click(), .dblclick(),<br>.mouseenter(), .mouseleave(), .hover(),<br>.mouseover(), .mousemove(), .mouseout(), .mousedown(), .mouseup() |

© JMA 2015. All rights reserved

# Objeto Event

- El sistema de eventos de jQuery normaliza el objeto Event de acuerdo a los estándares del W3C.
- Esta garantizado que el objeto Event se pasa al controlador de eventos.
- La mayoría de las propiedades de evento original se copian y se normaliza para el nuevo objeto de evento.
- El constructor `jQuery.Event` está expuesto y se puede utilizar cuando se invoca a un trigger. El operador new es opcional.  

```
var e = jQuery.Event("click");
// Desencadenar un evento click artificialmente
jQuery("body").trigger(e);
```
- jQuery normaliza las siguientes propiedades para la consistencia entre navegadores:
  - target, relatedTarget, pageX, pageY, which, metaKey
- Las siguientes propiedades también se copian en el objeto de evento, aunque algunos de sus valores pueden quedar indefinidos en función del evento:
  - altKey, bubbles, button, cancelable, charCode, clientX, clientY, ctrlKey, currentTarget, data, detail, eventPhase, metaKey, offsetX, offsetY, originalTarget, pageX, pageY, relatedTarget, screenX, screenY, shiftKey, target, view, which

© JMA 2015. All rights reserved

## Objeto Event: Propiedades

| Propiedades                 | Descripción   |
|-----------------------------|---|
| <code>type</code>           | Describe la naturaleza del evento.  |
| <code>data</code>           | Opcional. Un objeto de datos asociado al enlazar el controlador de ejecución actual.  |
| <code>timeStamp</code>      | Milisegundos entre el momento en que se ha creado el evento y 01/01/1970.   |
| <code>result</code>         | El último valor devuelto por un controlador de eventos que se desencadenó por este evento, a menos que el valor era indefinido. |
| <code>which</code>          | Indica la tecla o botón específico que se presionó.   |
| <code>namespace</code>      | El espacio de nombres especificado cuando se activa el evento.  |
| <code>metaKey</code>        | Indica si se ha pulsado la tecla META cuando el evento disparó.   |
| <code>target</code>         | El elemento DOM que inició el evento.   |
| <code>delegateTarget</code> | El elemento donde estaba conectada la llamada actualmente controlador de eventos jQuery.  |
| <code>currentTarget</code>  | El elemento DOM actual dentro de la fase de propagación de eventos.   |
| <code>relatedTarget</code>  | El otro elemento DOM involucrado en el evento, si los hay.  |
| <code>pageX</code>          | La posición del ratón respecto al borde izquierdo del documento.  |
| <code>pageY</code>          | La posición del ratón con relación al borde superior del documento.   |

© JMA 2015. All rights reserved

## Objeto Event

| Método                          | Descripción   |
|---------------------------------|---|
| preventDefault()                | Si se llama a este método, no se activará la acción predeterminada del evento.  |
| isDefaultPrevented()            | Devuelve si Event.preventDefault () nunca fue llamado en este objeto de evento.   |
| stopPropagation()               | Previene que el evento se propague por el árbol DOM, evitando cualquier controlador de los contenedores reciban la notificación del evento. |
| isPropagationStopped()          | Devuelve si Event.stopPropagation () nunca fue llamado en este objeto de evento.  |
| stopImmediatePropagation()      | Impide al resto de los manipuladores empezar la ejecución y evita que el evento se propague por el árbol DOM.                               |
| isImmediatePropagationStopped() | Devuelve si Event.stopImmediatePropagation () nunca fue llamado en este objeto de evento.   |

© JMA 2015. All rights reserved

## MANIPULACIÓN DE LOS ELEMENTOS

© JMA 2015. All rights reserved

## Sobrecarga

- jQuery "sobrecarga" sus métodos, en otras palabras, el método para establecer un valor posee el mismo nombre que el método para obtener un valor.
- Cuando un método es utilizado para obtener (o leer) un valor, es llamado obtenedor (en inglés getter) y no suele recibir argumentos. Devuelven el valor por el cual se consultó.  
`$( 'h1' ).html();`
- Cuando un método es utilizado para establecer un valor, es llamado método establecedor (en inglés setter) y recibe el nuevo valor como argumento. Devuelven un objeto jQuery, permitiendo continuar con la llamada de más métodos en la misma selección.  
`$( 'h1' ).html("<b>Hello world!</b>");`
- Los métodos establecedores permiten la modificación del valor con una función que debe devolver el nuevo valor y tiene dos parámetros: el índice en la selección y el valor original a modificar. A través de la referencia `this` se accede al elemento actual que se está modificando.

```
$("#test2").html(function(i, origText){
    return "Old html: " + origText + " New html: Hello <b>world!</b>
(index: " + i + ")";
});
```

© JMA 2015. All rights reserved

## Modificar el valor

- Para obtener o establecer el contenido en formato HTML:  
`$("#test2").html("<b>Hello world!</b>");`
- Para obtener o establecer el contenido en formato texto, las entidades HTML se auto “escapan”:  
`$("#test1").text("Hello world!");`
- Para obtener o establecer el valor de los campos de un formulario:  
`$("#test3").val("Your name");`

© JMA 2015. All rights reserved

# Crear y borrar elementos

- Crear nuevos elementos  

```
var e =$('<p>Un nuevo párrafo</p>');
```
- Crear un nuevo elemento con atributos utilizando un objeto  

```
var e ='<a/>', {
    html : 'Un <strong>nuevo</strong> enlace',
    'class' : 'new',
    href : 'foo.html'
});
```
- Añadir el elemento a la página  

```
($('body').append(e);
$('ol').append('<li>Appended item</li>');
$('ol').append(li1,li2,li3);
```
- Borrar elementos  

```
$("#test1").remove();
$("p").remove(".test");
```

© JMA 2015. All rights reserved

# Clonar, reemplazar y envolver elementos

- Para clonar un elemento  

```
$("#t1").clone().appendTo("#content");
```
- Para reemplazar un elemento  

```
$("#t1").replaceWith("<h2>has been replaced</h2>" );
```
- Para envolver un elemento, introducir un nuevo elemento cuyo contenido el elemento original a sustituir  

```
 $("p").wrap("<div></div>");
```
- Para envolver el contenido de un elemento  

```
 $("p").wrapInner("<b></b>");
```
- Para envolver el conjunto resultante  

```
 $(".inner").wrapAll("<div class='new' />");
```
- Para eliminar el elemento padre:  

```
 $("p").unwrap();
```

© JMA 2015. All rights reserved

# Métodos para insertar y borrar

| Método          | Descripción   |
|-----------------|---|
| .append()       | Añade el parámetro después de cada elemento del conjunto seleccionado.              |
| .appendTo()     | Añade el elemento seleccionado después de cada elemento del conjunto de destino.    |
| .after()        | Inserta el parámetro después de cada elemento del conjunto seleccionado.            |
| .insertAfter()  | Inserta el elemento seleccionado después de cada elemento del conjunto de destino.  |
| .before()       | Inserta el parámetro antes de cada elemento del conjunto seleccionado.              |
| .insertBefore() | Inserta el elemento seleccionado antes de cada elemento del conjunto de destino.    |
| .prepend()      | Inserta el contenido al principio de cada elemento del conjunto.                    |
| .prependTo()    | Inserta cada elemento del conjunto al comienzo del destino.                         |
| .replaceAll()   | Reemplaza con el elemento cada elemento del conjunto destino.                       |
| .empty()        | Borra todos los nodos secundarios del conjunto de elementos coincidentes de la DOM. |
| .remove()       | Borra el conjunto de elementos coincidentes de la DOM.                              |
| .detach()       | Separa el conjunto de elementos coincidentes de la DOM.                             |

© JMA 2015. All rights reserved

# Atributos y Propiedades

- Por atributos se entiende los atributos de las etiquetas HTML y por propiedades se entiende las propiedades de los HTMLElement. En algunos casos se solapan unos a otros pero en otros no.
- Un valor se modifica con dos parámetros (nombre y valor) o varios valores mediante un JSON con pares nombre/valor.
- .attr(): Obtiene el valor de un atributo para el primer elemento del conjunto o establecer uno o más atributos para cada elemento coincidente.  

```
var ref = $('a').attr('href');
$('a').attr('href', 'allMyHrefsAreTheSameNow.html');
$('a').attr({
  'title' : 'all titles are the same too',
  'href' : 'somethingNew.html'
});
```
- .prop(): Obtiene el valor de una propiedad para el primer elemento del conjunto o establecer una o más propiedades para cada elemento coincidente.
- .removeAttr(): Elimina el atributo de cada elemento del conjunto.  

```
($('a').removeAttr('href');
```
- .removeProp(): Elimina la propiedad de cada elemento del conjunto.

© JMA 2015. All rights reserved

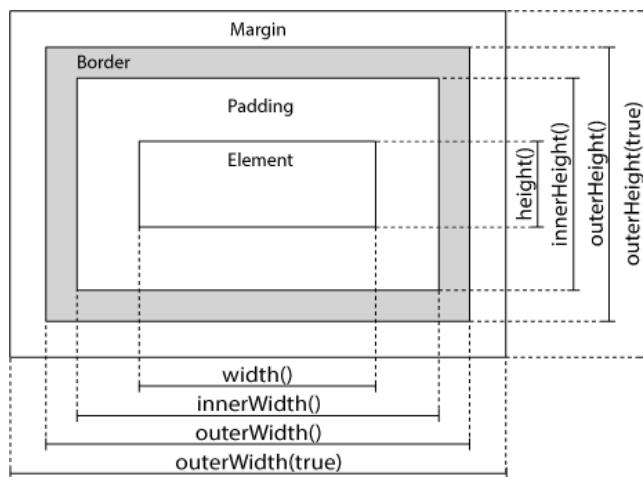
# Estilo

- Modificar el atributo style de las etiquetas:
 

```
$('.h1').css('fontSize');
$('.h1').css('fontSize', '100px');
$('.h1').css({ 'fontSize' : '100px', 'color' :
'red' });
```
- Modificar el atributo class de las etiquetas:
  - .hasClass(), .addClass(), .removeClass(), .toggleClass()
- Redimensionar las etiquetas:
  - .height(), .innerHeight(), .outerHeight()
  - .width(), .innerWidth(), .outerWidth(),
  - .position(), .offset()
  - .scrollLeft(), .scrollTop()

© JMA 2015. All rights reserved

# Dimensionado



© JMA 2015. All rights reserved

## EFFECTOS

© JMA 2015. All rights reserved

### Introducción

- Los efectos permiten a un elemento cambiar de un aspecto a otro realizando una transición entre ellos.
- A nivel interno, los efectos cambian y combinan los valores de las propiedades de estilo a lo largo de una línea temporal.
- Con jQuery, agregar efectos a una página es muy fácil, que poseen una configuración predeterminada pero también es posible proveerles parámetros personalizados.
- Es posible crear animaciones particulares estableciendo valores de propiedades CSS.
- Es posible encadenar varios efectos a la vez.

© JMA 2015. All rights reserved

## Efectos predefinidos

- .show: Muestra el elemento seleccionado.
- .hide: Oculta el elemento seleccionado.
- .toggle: Realiza el efecto show/hide contrario al actual.
- .fadeIn: Cambia la opacidad del elemento seleccionado al 100%.
- .fadeOut: Cambia la opacidad del elemento seleccionado al 0%.
- .fadeTo: Cambia la opacidad del elemento seleccionado al % indicado.
- .fadeToggle: Realiza el efecto fadeIn/fadeOut contrario al actual.
- .slideDown: Muestra el elemento con un movimiento de deslizamiento vertical.
- .slideUp: Oculta el elemento con un movimiento de deslizamiento vertical.
- .slideToggle: Realiza el efecto slideDown/slideUp contrario al actual.

© JMA 2015. All rights reserved

## Configuración

- Al aplicar el efecto se puede indicar con el primer parámetro la duración en milisegundos el efecto o la cadena con la duración definida en:
 

```
jQuery.fx.speeds: {
    slow: 600,
    fast: 200,
    _default: 400
}
```
- Para aplicar el efecto deseado:
 

```
$( 'h1' ).fadeIn(3000);
$( 'h1' ).fadeOut('slow');
```
- Se pueden añadir velocidades personalizadas:
 

```
jQuery.fx.speeds.miRitmo = 123;
```
- Como segundo parámetro se puede establecer la función que se invocará al terminar el efecto:
 

```
$( "p" ).hide("slow", function(){...});
```

© JMA 2015. All rights reserved

# Animaciones

- Es posible realizar animaciones en propiedades CSS utilizando el método `.animate`, que permite realizar una animación estableciendo valores a propiedades CSS o cambiando sus valores actuales.  
`$( "div" ).animate( {  
 left: '250px',  
 opacity: '0.5',  
 height: '150px',  
 width: '150px'  
}, 3000);`
- Se puede especificar el valor de una propiedad de animación como "show", "hide", or "toggle":  
`$( "div" ).animate( {height: 'toggle'} , 'slow');`
- También es posible definir valores relativos al valor actual poniendo `+ =` o `- =` delante del valor:  
`$( "div" ).animate( {  
 height: '+=150px',  
 width: '-=150px'  
}, 3000);`

© JMA 2015. All rights reserved

# Animaciones

- El Easing describe la manera en que un efecto ocurre, es decir, si la velocidad durante la animación es constante o no. jQuery incluye solamente dos métodos de easing: swing y linear. Si se desea transiciones más naturales en las animaciones, existen varias extensiones que lo permiten.  
`$( "div" ).animate( {  
 left : [ "+=50", "swing" ],  
 opacity : [ 0.25, "linear" ]  
}, 300);`
- De forma predeterminada, jQuery cuenta con cola para animaciones, de tal forma que, si se escriben múltiples animaciones consecutivas, se crea una cola de "interna" con las llamadas y se llama una por una al terminar la anterior.  
`var div = $( "div" );  
div.animate( {left: '100px'}, "slow");  
div.animate( {fontSize: '3em'}, "slow");`

© JMA 2015. All rights reserved

## Control de efectos

- El método `.stop([borrarCola] [, irAlFinal])` detiene el efecto actual, por parámetro se puede indicar si se continua con el resto de los efectos encolados y si se va al punto final de la animación.
- El método `.finish()` finaliza la cola saltando a los puntos finales sin esperar.
- Con el método `.delay(ms)` se puede establecer un temporizador en milisegundos para retrasar la ejecución del siguiente elemento en la cola.

```
var p = $("#p1");
p.slideUp(2000, function() {
    p.css("color", "red");
})
.delay(2000)
.slideDown(2000);
```

© JMA 2015. All rights reserved

## UTILIDADES JQUERY

© JMA 2015. All rights reserved

## data()

- jQuery ofrece una manera sencilla para poder guardar información relacionada a un elemento, y la misma biblioteca se ocupa de manejar los problemas que pueden surgir por falta de memoria.
- A través del método data() expone una colección donde es posible, para cada elemento, guardar cualquier tipo de información asociada a una clave.
- Para crear y modificar la colección:  

```
$("body").data("foo", 52);
$("body").data("bar", { myType: "test", count: 40 });
$("body").data({ baz: [ 1, 2, 3 ] });
```
- Para consultar los valores almacenados:  

```
var a=$("body").data("foo");
var b=$("body").data("bar").myType;
var c=$("body").data("baz")[2];
```
- Para eliminar los valores almacenados:  

```
$("body").removeData("baz");
```

© JMA 2015. All rights reserved

## each ()

- El método .each () está diseñado para hacer construcciones de bucle concisas y menos propensos a errores sobre los elementos DOM .
- Permite asociar la función que se quiere ejecutar para cada elemento de la selección.
- Cuando se invoca itera sobre los elementos DOM que forman parte del objeto jQuery.
- Cada vez que se invoca la función se le pasa el índice de la iteración, comenzando desde 0.
- La función se ejecuta en el contexto del elemento DOM actual, por lo que la palabra clave this hace referencia al elemento.  

```
$( "li" ).each(function( index ) {
  console.log( index + ": " + $( this ).text() );
});
```

© JMA 2015. All rights reserved

# Comprobación de tipos

| Método                 | Descripción   |
|------------------------|---|
| jQuery.type()          | Determinar la clase interna del JavaScript de un objeto, similar al <code>typeof</code> . |
| jQuery.isArray()       | Determina si el argumento es una matriz.  |
| jQuery.isEmptyObject() | Determina si un objeto está vacío (no contiene propiedades enumerables).                  |
| jQueryisFunction()     | Determina si el argumento pasado es un objeto función de Javascript.                      |
| jQuery.isNumeric()     | Determina si su argumento es un número.   |
| jQuery.isPlainObject() | Determina si un objeto es un objeto plano (creados con "{}" o "new Object").              |
| jQuery.isWindow()      | Determina si el argumento es una ventana.   |
| jQuery.isXMLDoc()      | Comprueba si un nodo DOM está dentro de un documento XML (o es un documento XML).         |

© JMA 2015. All rights reserved

# Utilidades de matrices

| Método             | Descripción   |
|--------------------|---|
| jQuery.inArray()   | Busca un valor determinado en una matriz y devuelve su índice o -1 si no lo encuentra.    |
| jQuery.grep()      | Encuentra los elementos de una matriz que cumplan con la función de filtro suministrada.  |
| jQuery.makeArray() | Convierte un objeto-array en un verdadero Array JavaScript.                               |
| jQuery.map()       | Genera un nuevo array convirtiendo cada uno de los elementos con la función suministrada. |
| jQuery.merge()     | Combina el contenido de dos matrices en la primera matriz.                                |
| jQuery.unique()    | Ordena una matriz de elementos DOM eliminando los elementos duplicados eliminados.        |

© JMA 2015. All rights reserved

# Utilidades y analizadores

| Método              | Descripción  |
|---------------------|--|
| jQuery.noop()       | Función vacía para cuando es obligatorio suministrar una.                          |
| jQuery.now()        | Devuelve un número que representa la hora actual, equivale a (new Date).getTime(). |
| jQuery.trim()       | Quita el espacio en blanco del principio y final de una cadena.                    |
| jQuery.globalEval() | Ejecutar código JavaScript en el contexto global.                                  |
| jQuery.parseHTML()  | Analiza una cadena en una matriz de nodos DOM.                                     |
| jQuery.parseJSON()  | Toma una cadena JSON bien formada y devuelve el valor resultante JavaScript.       |
| jQuery.parseXML()   | Analiza una cadena en un documento XML.  |
| jQuery.extend()     | Combinar las propiedades de dos o más objetos en el primer objeto.                 |
| jQuery.contains()   | Comprueba si un elemento DOM es un descendiente de otro elemento DOM.              |

© JMA 2015. All rights reserved

## JQUERY AJAX

© JMA 2015. All rights reserved

# Introducción

- AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications).
- A través de varios métodos, jQuery provee soporte para Ajax, permitiendo abstraer las diferencias que pueden existir entre navegadores.
- El método `$.ajax()` encapsula todas las tareas necesarias para realizar la petición asíncrona y procesar la respuesta.
- El método `$.ajax()` devuelve el objeto `jqXHR`, que es un supraconjunto del objeto XMLHttpRequest.
- jQuery suministra atajos preconfigurados mediante los métodos `$.get()`, `$.getScript()`, `$.getJSON()`, `$.post()` y `$(()).load()`.
- A pesar que la definición de Ajax posee la palabra XML, la mayoría de las aplicaciones no utilizan dicho formato para el transporte de datos, sino que en su lugar se utiliza HTML plano o información en formato JSON.

© JMA 2015. All rights reserved

# Tipos de datos

- jQuery soporta los formatos mas habituales de intercambio de datos:
  - text: Para el transporte de cadenas de caracteres simples.
  - html: Para el transporte de bloques de código HTML que serán ubicados en la página.
  - script: Para añadir un nuevo script con código JavaScript a la página.
  - json: Para transportar información en formato JSON, el cual puede incluir cadenas de caracteres, arrays y objetos.
  - jsonp: Para transportar información JSON de un dominio a otro.
  - xml: Para transportar información en formato XML.
- Cuando el tipo de dato no este es especificado por el nombre del método habrá que indicarlo en la configuración.

© JMA 2015. All rights reserved

# Petición AJAX

```
$.ajax({
    url : 'post.php',      // la URL para la petición
    data : { id : 123 }, // la información a enviar
    type : 'GET',         // especifica si será una petición POST o GET
    dataType : 'json',    // el tipo de información de la respuesta
    // código a ejecutar si la petición es satisfactoria
    success : function(json) {
        $('<h1>').text(json.title).appendTo('body');
        $('<div class="content">')
            .html(json.html).appendTo('body');
    },
    // código a ejecutar si la petición falla
    error : function(xhr, status) {
        alert('Disculpe, existió un problema');
    }
});
```

© JMA 2015. All rights reserved

## Configuración básica

- **url:** Establece la URL en donde se realiza la petición. La opción url es obligatoria para el método `$.ajax()`;
- **type/method:** Verbo HTTP: GET, POST, PUT, DELETE, HEAD, ... Por defecto GET. Algunos verbos pueden no estar soportados por todos los navegadores.
- **data:** Establece la información que se enviará al servidor. Puede ser: un objeto, un array o una cadena de datos.
- **dataType:** Establece el tipo de información que se espera recibir como respuesta del servidor. Si no se especifica ningún valor, de forma predeterminada, jQuery revisa el tipo de MIME recibido con la respuesta.
- **jsonp:** Establece el nombre de la función de devolución de llamada a enviar cuando se realiza una petición JSONP. De forma predeterminada el nombre es callback

© JMA 2015. All rights reserved

# Configuración específica

- **async:** Establece si la petición será asíncrona o no. De forma predeterminada el valor es true. Debe tener en cuenta que si la opción se establece en false, la petición bloqueará la ejecución de otros códigos hasta que dicha petición haya finalizado.
- **username:** El usuario que utilizará XMLHttpRequest en respuesta a una solicitud de autenticación de acceso HTTP.
- **password:** La contraseña que utilizará XMLHttpRequest en respuesta a una solicitud de autenticación de acceso HTTP.
- **headers:** Permite añadir los pares clave/valor adicionales a enviar en el encabezado de las peticiones. El encabezado "X-Requested-With: XMLHttpRequest" siempre se añade.
- **cache:** Establece si la petición será guardada en la cache del navegador. De forma predeterminada es true para todos los dataType excepto para script y jsonp. Cuando posee el valor false, se agrega una cadena de caracteres anti-cache al final de la URL de la petición.

© JMA 2015. All rights reserved

# Configuración avanzada

- **context:** Establece el alcance en que la/las funciones de devolución de llamada se ejecutarán (define el significado de this dentro de las funciones). De manera predeterminada this hace referencia al objeto originalmente pasado al método \$.ajax.
- **timeout:** Establece un tiempo en milisegundos para considerar a una petición como fallada.
- **contentType:** Indica al servidor el tipo de contenido. El valor predeterminado es "application/x-www-form-urlencoded; charset=UTF-8 ", lo cual es adecuado para la mayoría de los casos.
- **crossDomain:** true si se desea forzar una solicitud crossdomain (como JSONP) en el mismo dominio. Esto permite, por ejemplo, la redirección del lado del servidor a otro dominio. Por defecto: false para solicitudes del mismo dominio y true para las solicitudes a otros dominios.
- **converters:** Conjunto de convertidores para los tipo de datos recibidos. Un convertidor es una función que devuelve el valor transformado de la respuesta. Por defecto: {"text": window.String, "text html": true, "text json": jQuery.parseJSON, "text xml": jQuery.parseXML}

© JMA 2015. All rights reserved

# Configuración de Eventos

- **complete:** Establece una función de devolución de llamada que se ejecuta cuando la petición está completa, haya fallado o no. La función recibe como argumentos el objeto de la petición en crudo y el código de estatus de la misma petición (Event event, jqXHR jqXHR, PlainObject ajaxOptions).
- **success:** Establece una función a ejecutar si la petición ha sido satisfactoria. Dicha función recibe como argumentos la información de la petición (convertida a objeto JavaScript en el caso que dataType sea JSON), el estatus de la misma y el objeto de la petición en crudo (Event event, jqXHR jqXHR, PlainObject ajaxOptions, PlainObject data).
- **error:** Establece una función de devolución de llamada a ejecutar si resulta algún error en la petición. Dicha función recibe como argumentos el objeto de la petición en crudo y el código de estatus de la misma petición (Event event, jqXHR jqXHR, PlainObject ajaxSettings, String errorThrown).
- **statusCode:** Establece un conjunto de pares código/función. Permite asociar una función a cada código HTTP de estado.

© JMA 2015. All rights reserved

# Atajos

- Los métodos que provee la biblioteca son:
  - `$.get()`: Realiza una petición GET.
  - `$.getScript()`: Realiza una petición GET y añade un script a la página.
  - `$.getJSON()`: Realiza una petición GET y espera que el dato devuelto sea JSON.
  - `$.post()`: Realiza una petición POST.
  - `$(()).load()`: Obtiene el código HTML de una URL y rellena a los elementos seleccionados con la información obtenida.
- Los métodos pueden tener los siguientes argumentos, salvo la URL obligatoria, en el siguiente orden:
  - `url`: La URL en donde se realizará la petición.
  - `data`: La información que se enviará al servidor (NO en `$.getScript()`).
  - `success`: Función que se ejecuta en caso que petición haya sido satisfactoria (String script, String textStatus, jqXHR jqXHR).
  - `dataType`: El tipo de dato que se espera recibir desde el servidor: xml, json, jsonp, script, text o html (NO en `$.getJSON()` ni en `$.getScript()`).
- Con `$.ajaxSetup()` se fijan las opciones de configuración por defecto.

© JMA 2015. All rights reserved

# Ayudantes para formularios

- Para transformar la información de un formulario a una cadena POST  

```
$('#myForm').serialize();
Resultado:
field1=123&field2=hello+world
```
- Para crear un array de objetos contenido información de un formulario  

```
($('#myForm').serializeArray();
Resultado:
[
    { name : 'field1', value : 123 },
    { name : 'field2', value : 'hello world' }
]
```
- Para enviar el formulario:  

```
("button").click(function(){
    $.post("demo_test_post.asp",
        $('#myForm').serialize(),
        function(data, status){
            alert("Data: " + data + "\nStatus: " + status);
        });
});
```

© JMA 2015. All rights reserved

# Controladores de eventos globales

- Se pueden registrar controladores globales que se llevan a cabo para cualquier petición Ajax en la página.
  - Solo se dispararon si la propiedad global en `jQuery.ajaxSetup ()` es true, como es por defecto.
  - Los eventos disponibles
    - `.ajaxComplete()`: Cuando se completa la petición.
    - `.ajaxSuccess()`: Cuando se completa la petición satisfactoriamente.
    - `.ajaxError()`: Cuando se completa la petición con un error.
    - `.ajaxSend()`: Antes de enviar una petición Ajax.
    - `.ajaxStart()`: Cuando comienza la primera petición Ajax.
    - `.ajaxStop()`: Cuando todas las peticiones Ajax han completado.
- ```
$(document).ajaxError(function(event, request, settings) { ...
})
$('#trabajandoAJAX')
    .ajaxStart(function() { $(this).show(); })
    .ajaxStop(function() { $(this).hide(); });
```

© JMA 2015. All rights reserved

## Promesas (> 1.5)

- El objeto jqXHR devuelto por `$.ajax()`, a partir de jQuery 1.5, implementa la interfaz Promesa, dándole todas las propiedades, métodos y comportamiento de las mismas.
- Estos métodos toman uno o más argumentos función que se invocan cuando finaliza la solicitud `$.ajax()`, lo que permite asignar múltiples devoluciones de llamada en una sola solicitud. También permiten generar la llamada en un punto y tratar la respuesta en otro.
- Los métodos Promesa disponibles del objeto jqXHR incluyen:
  - `jqXHR.done(function(data, textStatus, jqXHR) {});` Para cuando se ha recibido correctamente la respuesta.
  - `jqXHR.fail(function(jqXHR, textStatus, errorThrown) {});` Para cuando se ha fallado la petición.
  - `jqXHR.always(function(datos | jqXHR, textStatus, jqXHR | errorThrown) {});` Siempre falle o no la petición, los parámetros son los de done o fail según corresponda (agregado en jQuery 1.6)
  - `jqXHR.then(function(data, textStatus, jqXHR) {}, function(jqXHR, textStatus, errorThrown) {});` Incorpora la funcionalidad de los métodos `.done()` y `.fail()`, lo que permite manipular (a partir de jQuery 1.8) el Promise subyacente.

© JMA 2015. All rights reserved



© JMA 2015. All rights reserved

RWD – Responsive Web Design

## DISEÑO WEB ADAPTATIVO

© JMA 2015. All rights reserved

### Diseño Adaptativo

- Es un enfoque de diseño destinado a la elaboración de sitios/aplicaciones para proporcionar un entorno óptimo de:
  - Lectura Fácil
  - Navegación correcta con un número mínimo de cambio de tamaño
  - Planificaciones y desplazamientos
- Con la irrupción multitud de nuevos dispositivos y el que el acceso a internet se realiza ya mayoritariamente desde dispositivos diferentes a los tradicionales ordenadores ha obligado a seguir dicho enfoque en las aplicaciones WEB.
- Contempla la definición de múltiples elementos antes de la realización de la programación real.
  - Elementos de página en las unidades de medidas correctas
  - Imágenes flexibles
  - Utilización de CSS dependiendo de la aplicación

© JMA 2015. All rights reserved

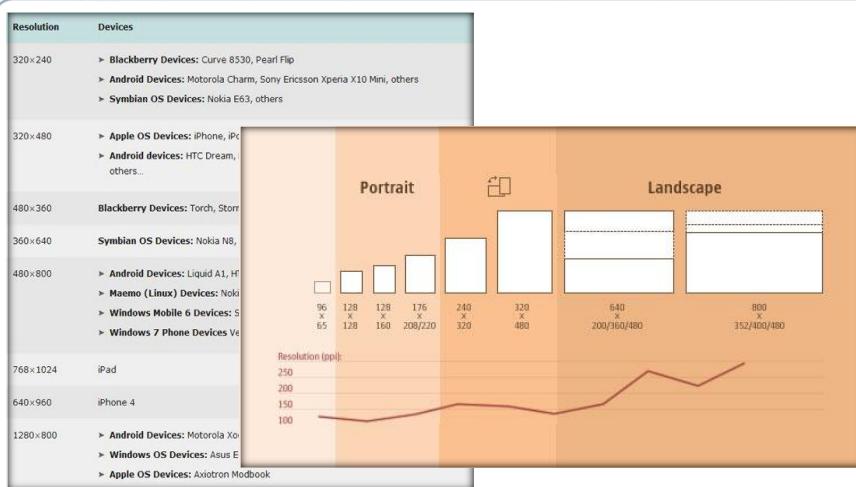
# Resolución

- Los dispositivos móviles tienen una característica distintiva, y es su resolución de pantalla.
- Es necesario conocer cuales son las resoluciones más comunes en este tipo de dispositivos móviles, de los gadgets más utilizados, etc.
- Las resoluciones van cambiando de forma muy rápida y en dispositivos nuevos



© JMA 2015. All rights reserved

# Resolución



© JMA 2015. All rights reserved

# Resolución

- También deberemos tener en cuenta la resoluciones de otros dispositivos como:
  - Tablets
  - TV SmartTV
  - Pizarras electrónicas, etc



**Pizarras** 10.1" y 11.6" (2560x1440, 1920x1080, 1366x768), 17" (1920x1080)

**PC** 12" (1280x800), 14" (1920x1080, 1366x768), 15.6" (1920x1080)

**Family hub** 23" (1920x1080), 27" (2560x1440)

© JMA 2015. All rights reserved

# Orientación de Página

- La orientación del papel es la forma en la que una página rectangular está orientada y es visualizada.
- Los dos tipos más comunes son:
  - Landscape (Horizontal)
  - Portrait (Vertical)



© JMA 2015. All rights reserved

## Recomendaciones de Diseño

1. Utilizar porcentajes y “ems” como unidad de medida en lugar de utilizar los valores determinados como definición de pixel.

**Las ems son unidades relativas,**  
así que más exactamente 1 em equivale  
al cien por cien del tamaño inicial de  
fuente.

© JMA 2015. All rights reserved

## Recomendaciones de Diseño

2. Determinar el tamaño y definición de las imágenes a utilizar

- Recortar, Ajustar, etc



© JMA 2015. All rights reserved

## Recomendaciones de Diseño

3. El contenido y funcionalidad BÁSICA debe de ser accesible por todos los navegadores



© JMA 2015. All rights reserved

## Recomendaciones de Diseño

4. Definición correcta de contenidos en función del dispositivo



© JMA 2015. All rights reserved

## Ventajas

- Soporte de dispositivos móviles.
- Con una sola versión en HTML y CSS se cubren todas las resoluciones de pantalla.
- Mejora la experiencia de usuario.
- Se reducen los costos de creación y mantenimiento cuando el diseño de las pantallas es similar entre dispositivos de distintos tamaños.
- Evita tener que desarrollar aplicaciones específicas para cada sistema operativo móvil.
- Facilita la referenciación y posicionamiento en buscadores, versión única contenido/página.

© JMA 2015. All rights reserved

## Mobile First



Responsive Web Design

Mobile First Web Design



© JMA 2015. All rights reserved

# INTRODUCCIÓN

© JMA 2015. All rights reserved

## Características

- Twitter Bootstrap es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web adaptativo (diseño).
- Bootstrap se puede descargar y usar de forma totalmente gratuita.
- Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales que son opcionales.
- Bootstrap es el framework libre de cliente más rápido y fácil para el desarrollo web adaptativo apto para dispositivos móviles
- Permite la creación de sitios web que se ajustan a sí mismos automáticamente para que visualicen correctamente en todos los dispositivos, desde pequeños teléfonos a grandes televisores.
- Todos los plugins JavaScript de Bootstrap requieren la librería jQuery para funcionar, por lo que se deberá incluir.

© JMA 2015. All rights reserved

# Añadir Bootstrap a una página

- Se pueden utilizar dos estrategias diferentes:
  - Incluir ficheros locales.
  - Incluir ficheros compartidos en una CDN (Content Delivery Network).
- Se pueden descargar los ficheros locales desde <http://getbootstrap.com/getting-started/>:
  - Versión de producción: para servidores web se encuentra minimizada y compactada para reducir al máximo su tamaño.
  - Versión de desarrollo: para desarrollo y pruebas sin comprimir y depurable.

```
<head>
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <script type="text/javascript" src="/js/jquery.min.js"></script>
  <script type="text/javascript" src="/js/bootstrap.min.js"></script>
</head>
```
- Las CDN permiten compartir contenidos comunes entre diferentes sitios y evitar descargas al aprovechar la cache de los navegadores:
 

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">
<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
```
- Es conveniente incluirlos después de las hojas de estilos para asegurar la importación de todos los estilos o al final del cuerpo para mejorar la percepción del usuario.

© JMA 2015. All rights reserved

## Contenidos descargados

### Versión compilada

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

### Versión original

```
bootstrap/
  less/
  js/
  fonts/
  dist/
    css/
    js/
    fonts/
  docs/
    examples/
```

© JMA 2015. All rights reserved

# Personalización

- Bootstrap viene con SASS CSS, pero el código fuente utiliza dos de los más populares preprocesadores CSS: Less y Sass.
- La utilidad Less está disponible online para personalizar el Bootstrap en:
  - <http://getbootstrap.com/customize/>
- La personalización sigue los siguientes pasos:
  1. Seleccionar de los elementos que se van a utilizar y se desean conservar: CSS Común, Componentes Bootstrap, componentes JavaScript y plugins jQuery.
  2. Cambiar los valores de las variables Less que controlan los colores, tamaños, estilos y otros utilizadas dentro de las hojas de estilo Bootstrap.
  3. Generar y descargar la versión personalizada.
  4. Descomprimir y ubicar en los directorios de nuestro sitio web.

© JMA 2015. All rights reserved

# Plantilla

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Plantilla</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/css/bootstrap.min.css">
</head>
<body>
    <div class="container">

        </div>
        <script src="/js/jquery.min.js"></script>
        <script src="/js/bootstrap.min.js"></script>
</body>
</html>
```

© JMA 2015. All rights reserved

# ESTRUCTURA

© JMA 2015. All rights reserved

## Sintaxis

- Bootstrap define muchas clases CSS para personalizar los diferentes elementos.
- Utiliza una sintaxis declarativa:
  - Atributo CLASS
  - Atributos personalizados
  - Estilos por defecto
- El elemento base es la rejilla. Bootstrap incluye una rejilla o retícula fluida, pensada para móviles, que administra el espacio disponible (posicionamiento y tamaños) para cumplir con el diseño web adaptativo.
- 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 los diseños.

© JMA 2015. All rights reserved

# Clases predefinidas

- Clases contenedoras:
  - .container (ancho fijo)
  - .container-fluid
- Clase fila:
  - .row
- Clases columnas:
  - col-*escala-nº de columnas*
    - Escala: xs (muy pequeños), sm (pequeños), md (medianos), lg (grandes)
    - Nº de columnas: 1..12
- Las clases se aplican normalmente a etiquetas DIV en su atributo CLASS.

© JMA 2015. All rights reserved

## Características por tamaños

	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 mínima del contenedor	Ninguna (auto)	728px	940px	1170px
Prefijo de las clases CSS	.col-xs-	.col-sm-	.col-md-	.col-lg-
Número de columnas	12			
Anchura máxima de columna	auto	~62px	~81px	~97px
Separación entre columnas	30px (15px a cada lado de la columna)			
Permite anidación	Si			
Desplazar columnas	Si			
Reordenación de columnas	Si			

© JMA 2015. All rights reserved

# Cuadricula

```
<div class="container">
  <!-- Stack the columns on mobile by making one full-width and the other half-width -->
  <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>
  <!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on desktop -->
  <div class="row">
    <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
    <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
    <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  </div>
  <!-- Columns are always 50% wide, on mobile and desktop -->
  <div class="row">
    <div class="col-xs-6">.col-xs-6</div>
    <div class="col-xs-6">.col-xs-6</div>
  </div>
</div>
```

© JMA 2015. All rights reserved

# Rejilla

- Las filas siempre se definen dentro de un contenedor de tipo .container (anchura de columna fija) o de tipo .container-fluid (anchura de columna variable). De esta forma las filas se alinean bien y muestran el padding correcto.
- Las filas se utilizan para agrupar horizontalmente a varias columnas.
- El contenido siempre se coloca dentro de las columnas, ya que las filas sólo deberían contener como hijos elementos de tipo columna.
- 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.
- Las columnas de la rejilla definen su anchura especificando cuántas de las 12 columnas de la fila ocupan.
- Una columna puede tener varias definiciones, separadas por espacios, para los diferentes tamaños de dispositivos.

© JMA 2015. All rights reserved

# Columnas

- Desplazamiento de columnas
  - Mueva las columnas hacia la derecha, usando clases .col-md-offset-\*.
  - Estas clases aumentan el margen izquierdo de una columna por \* columnas.
- Columnas anidadas
  - Se pueden anidar columnas dentro de otras columnas.
  - Para ello, dentro de una columna con la clase col-md-\* crea un nuevo elemento con la clase .row y añade una o más columnas con la clase .col-md-\*.
  - Las columnas anidadas siempre tienen que sumar 12 columnas de anchura.
- Cambio de orden de columnas
  - Se puede cambiar fácilmente el orden de las columnas de cuadrícula preconfiguradas con las clases de modificador .col-md-push-\* y .col-md-pull-\*.

© JMA 2015. All rights reserved

# Control de la visualización

Clase	Teléfonos	Tablets	Ordenador	Ordenador grande
.visible-xs	Visible	Oculto	Oculto	Oculto
.visible-sm	Oculto	Visible	Oculto	Oculto
.visible-md	Oculto	Oculto	Visible	Oculto
.visible-lg	Oculto	Oculto	Oculto	Visible
.hidden-xs	Oculto	Visible	Visible	Visible
.hidden-sm	Visible	Oculto	Visible	Visible
.hidden-md	Visible	Visible	Oculto	Visible
.hidden-lg	Visible	Visible	Visible	Oculto
Clase	Navegador		Impresora	
.visible-print	Oculto		Visible	
.hidden-print	Visible		Oculto	

© JMA 2015. All rights reserved

## ESTILOS CSS

© JMA 2015. All rights reserved

### Tipografía

- Redefine los estilos de:
  - h1 ... h6, p, mark, ins, del, s, u, strong, small, em, abbr, address, blockquote,
- Clases de alineación
  - text-left, text-center, text-right, text-justify, text-nowrap
- Clases de transformación de texto
  - text-lowercase, text-uppercase, text-capitalize
- Clases para listas
  - ul, ol: list-unstyled, list-inline
  - dl: dl-horizontal
- Crea estilos específicos para las etiquetas de código:
  - code, kbd, var, pre,

© JMA 2015. All rights reserved

# Imágenes

- Bootstrap define varias clases CSS para decorar las imágenes del sitio web:
  - .img-rounded, añade unas pequeñas esquinas redondeadas en todos los lados de la imagen aplicando el estilo border-radius: 6px.
  - .img-thumbnail, muestra la imagen con un relleno blanco y un borde fino simulando el aspecto de las fotografías de las antiguas cámaras instantáneas. Añade además una breve animación para hacer que la imagen aparezca al cargar la página.
  - .img-circle, convierte la imagen en un círculo aplicando el estilo border-radius: 50%



© JMA 2015. All rights reserved

# Tablas

- Estilo básico con líneas de separación de filas:  
`<table class="table">`
- Filas marcadas con franjas:  
`<table class="table table-striped">`
- Con todos los bordes en la tabla y las celdas:  
`<table class="table table-bordered">`
- Remarcado de la fila sobre la que está el ratón:  
`<table class="table table-hover">`
- Tabla compacta, disminuye a la mitad el espacio entre las celdas:  
`<table class="table table-condensed">`

© JMA 2015. All rights reserved

# Tablas

- Las clases contextuales colorean las filas de la tabla o celdas individuales.

.active	Aplica el color de desplazamiento del ratón a una fila o celda específicas.
.success	Indica una acción exitosa o positiva.
.info	Indica un cambio o acción informativos neutrales.
.warning	Muestra una advertencia que puede hacer falta solucionar.
.danger	Indica un acción peligrosa o potencialmente negativa

- Para que aparezca la barra de desplazamiento horizontal en dispositivos pequeños

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

© JMA 2015. All rights reserved

# Formularios

- Bootstrap aplica por defecto algunos estilos a todos los componentes de los formularios.
- Para optimizar el espaciado, se utiliza la clase .form-group para encerrar cada campo de formulario con su <label>.
- Para que el formulario ocupe el menor espacio posible, añade la clase .form-inline para que las etiquetas <label> se muestren a la izquierda de cada campo del formulario.
- Para alinear los elementos <label> y los campos de formulario mediante las clases CSS utilizadas para definir las rejillas se añade la clase .form-horizontal al formulario, que modifica la clase .form-group para que se comporte como la fila de una rejilla.
- Si se añades la clase .form-control a los elementos <input>, <textarea> y <select>, su anchura se establece a width: 100%.

© JMA 2015. All rights reserved

# Formulario

```
<form class="form-inline">
<div class="form-group">
<label class="sr-only" for="idAmount">Amount (in dollars)</label>
<div class="input-group">
<div class="input-group-addon">$</div>
<input type="text" class="form-control" id="idAmount" placeholder="Amount">
<div class="input-group-addon">.00</div>
</div>
</div>
<div class="form-group">
<div class="col-sm-offset-2 col-sm-10">
<div class="checkbox">
<label>
<input type="checkbox"/> Remember me
</label>
</div>
</div>
</div>
<button type="submit" class="btn btn-primary">Transfer cash</button>
</form>
```

© JMA 2015. All rights reserved

## Controles compatibles

- INPUT
  - text, password, datetime, datetime-local, date, month, time, week, number, email, url, search, tel y color.
- TEXTAREA
- SELECT
- Casillas de verificación y botones de radio
  - .disabled .radio, .radio-inline, .checkbox, .checkbox-inline

© JMA 2015. All rights reserved

## Estados de formulario

- Bootstrap aplica una sombra a los campos seleccionados mediante la propiedad box-shadow de CSS aplicada a la pseudo-clase :focus del elemento.
  - Añadiendo el atributo disabled a cualquier campo de texto se evita que el usuario pueda introducir información y Bootstrap lo muestra con un aspecto muy diferente.
  - Además de deshabilitar campos individuales, también es posible añadir el atributo disabled a un elemento <fieldset> para deshabilitar cualquier campo de formulario que se encuentre en su interior.
  - 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 y .has-success para cuando el valor es correcto.
  - Estas clases se pueden aplicar a cualquier elemento que contenga una de las tres siguientes clases: .control-label, .form-control y .help-block.
  - Se utiliza la clase help-block para mostrar los mensajes de ayuda de los campos del formulario.
- <span class="help-block">Un texto de ayuda que ocupa dos líneas porque es muy largo, pero aún así se ve bien gracias a los estilos de Bootstrap.</span>

© JMA 2015. All rights reserved

## Botones

- Las siguientes clases se pueden aplicar a etiquetas para mostrar botones:
  - <a>, <button> e <input>.
- Se pueden crear diferentes tipos de botones con ayuda de cualquiera de las clases CSS definidas por Bootstrap
  - btn-default (normal), btn-primary (destacado), btn-success (éxito), btn-info (información), btn-warning (advertencia), btn-danger (peligro) y btn-link (enlace).
- Cuando se necesite crear botones más grandes o más pequeños que el tamaño estándar:
  - .btn-lg (grande), .btn-sm (pequeño) y .btn-xs (extra pequeño).
- Si se quiere forzar a que el botón muestre el aspecto presionado, se añade la clase .active.
- Se añade el atributo disabled para dar un aspecto desactivado a los elementos <button>.

© JMA 2015. All rights reserved

# Utilidades

- Bootstrap define la clase `.close` para mostrar la entidad HTML &times; como si fuera la típica X asociada con el cierre de una ventana o aplicación.  
`<button type="button" class="close" aria-hidden="true">&times;</button>`
- Un elemento flotante a la derecha o a la izquierda es muy habitual en la mayoría de diseños web, por eso define dos clases CSS genéricas llamadas `.pull-left` y `.pull-right` que se puede aplicar sobre cualquier elemento.
- 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>`
- Se aplica la clase especial `center-block` para centrar horizontalmente cualquier elemento (el elemento centrado se convierte en un elemento de bloque).
- Con las clases `.show` y `.hide`, que muestran y ocultan cualquier elemento. La clase `.sr-only` marca un contenido como oculto y que sólo esté disponible para los lectores ("screen readers").

© JMA 2015. All rights reserved

# Aspecto visual

## Colores contextuales

- `text-muted`
- `text-primary`
- `text-success`
- `text-info`
- `text-warning`
- `text-danger`

## Fondos contextuales

- `bg-primary`
- `bg-success`
- `bg-info`
- `bg-warning`
- `bg-danger`

© JMA 2015. All rights reserved

<http://getbootstrap.com/components/>

## COMPONENTES

© JMA 2015. All rights reserved

## Componentes

- Iconos (glyphicons)
- Menús desplegables
- Grupos de botones
- Botones desplegables
- Grupos de campos de formulario
- Elementos de navegación
- Barras de navegación
- Migas de pan
- Paginadores
- Etiquetas
- Insignias (Badges)
- Jumbotron
- Encabezado de página
- Imágenes en miniatura
- Mensajes de alerta
- Barras de progreso
- Objetos multimedia
- Listas de elementos
- Paneles
- Contenido empotrado
- Pozos

© JMA 2015. All rights reserved

# JAVASCRIPT

© JMA 2015. All rights reserved

## Plug-in

- Se pueden usar todos los plug-in de Bootstrap solamente por medio de JavaScript.
- Todos los API públicos son métodos encadenables y únicos y generan la colección sobre la que se actúa.
- Transiciones (`transition.js`)
  - Para efectos sencillos de transición
- Modales (`modal.js`)
  - Los modales son cuadros de diálogo con mensajes en pantalla flexibles y eficientes, con una funcionalidad requerida mínima y valores predeterminados inteligentes.
- Desplegables (`dropdown.js`)
  - Permite agregar menús desplegables a casi cualquier cosa, incluyendo en barras de navegación, fichas y píldoras.

© JMA 2015. All rights reserved

# Plug-in

- ScrollSpy (scrollspy.js)
  - Sirve para actualizar automáticamente los objetos de navegación (menú), con base en la posición de desplazamiento del contenido.
- Fichas comutables (tab.js)
  - Permiten gestionar interfaces basados en solapas.
- Herramientas de ayuda o consejo (tooltip.js)
  - Muestran la ayuda emergente cuando el usuario detiene el ratón en un elemento o realiza un tap largo.
- Popovers (popover.js)
  - Agregar pequeñas cubiertas de contenido, como las del iPad, a un elemento para albergar información secundaria.
- Mensajes de alerta (alert.js)
  - Con este plug-in, se puede agregar /descartar la funcionalidad para todos los mensajes de alerta.

© JMA 2015. All rights reserved

# Plug-in

- Botones (button.js)
  - Controla los estados de botones o crea grupos de botones para obtener más componentes como barras de herramientas.
- Colapsar (collapse.js)
  - Plug-in que utilizan distintas clases para expandir y colapsar el contenido
- Bandeja circular (carousel.js)
  - Gestiona el componente de bandeja circular que rota imágenes en una determinada secuencia. Generalmente no es compatible con los estándares de accesibilidad.
- Affix - anexar (affix.js)
  - Gestiona el posicionamientos de los elementos afijos.

© JMA 2015. All rights reserved



© JMA 2015. All rights reserved

## Introducción

- Modernizr es una librería JavaScript que nos permite conocer el soporte que da el navegador a las tecnologías HTML5 y CSS3, lo que nos permitirá desarrollar sitios web que se adapten a las capacidades cada navegador.
- En lugar de poner en la lista negra rangos completos de características no soportadas por la mayoría de los navegadores, Modernizr usa la detección de características para permitir adaptar fácilmente las experiencias de usuario en función de las capacidades reales de su navegador.
- Sabiendo que el navegador soporta ciertas capacidades de CSS3 o de HTML5, podremos utilizarlas con libertad. De modo contrario, si sabemos que un navegador no es compatible con determinada funcionalidad, podremos implementar variantes que sí soporte y así crear sitios web que se adaptan perfectamente al cliente web de cada visitante.
- Existen dos herramientas principales en Modernizr que se pueden utilizar para detectar las funcionalidades que están presentes en un navegador:
  - A través de JavaScript
  - Directamente sobre el CSS.

© JMA 2015. All rights reserved

# Personalización

- La web de Modernizr permite generar y descargar un fichero personalizado, minimizado o extendido, que contenga exclusivamente los elementos necesarios para la aplicación:
  - [https://modernizr.com/download?do\\_not\\_use\\_in\\_production](https://modernizr.com/download?do_not_use_in_production)
- La personalización sigue los siguientes pasos:
  1. Seleccionar de los elementos que se van a utilizar.
  2. Marcar o demarcar minify.
  3. Generar y descargar la versión personalizada.
  4. Añadir a la aplicación web en la carpeta de código JavaScript de terceros.
  5. Referenciar en las páginas html: se recomienda añadirlo en la cabecera antes que otros scripts y después de los estilos:

```
<script src="vendor/modernizr-custom.min.js"></script>
```

© JMA 2015. All rights reserved

# Objeto Modernizr

- El objeto Modernizr expone como propiedades booleanas las características a detectar (etiquetas, atributos, propiedades, objetos ES6, ...) para implementar el código alternativo:
 

```
if (Modernizr.canvas) {
  // Se puede utilizar el canvas
} else {
  // El elemento canvas no está disponible
}
```
- En algunos casos las propiedades son contenedores de características:
 

```
if(Modernizr.input.max){ ... }
```
- Alternativamente se puede utilizar el método Modernizr.on :
 

```
Modernizr.on('flash',function( result ) {
  if (result){ // the browser has flash
  } else { // the browser does not have flash
  }
});
```

© JMA 2015. All rights reserved

## Agregar nuevas características

- La forma más común de crear una nueva característica detectables es utilizando el método Modernizr.addTest asociando una cadena (preferiblemente solo en minúscula, sin ningún signo de puntuación) a una función que devolverá un resultado booleano: true si cuenta con la característica y false en caso contrario:  

```
Modernizr.addTest('itsTuesday', function() {
    return (new Date()).getDay() === 2;
});
```
- También se puede hacer asociando una constante:  

```
Modernizr.addTest('hasJquery', 'jQuery' in window);
```
- Se invoca como una característica mas:  

```
if(Modernizr.hasJquery) { ... }
```

© JMA 2015. All rights reserved

## Otras consultas:

- Modernizr.hasEvent(eventName,[element]): permite determinar si el navegador es compatible con un evento suministrado.
- Modernizr.mq(mq): permite verificar programáticamente si el estado actual de la ventana del navegador coincide con una consulta de medios.  

```
if (Modernizr.mq('(min-width: 900px)')) { ... }
```
- Modernizr.testAllProps(prop,[value],[skipValueTest]): determina si el navegador admite una propiedad de CSS o valor determinado, en alguna forma prefijada.  

```
testAllProps('boxSizing') // true
testAllProps('display', 'flex') // true
```

© JMA 2015. All rights reserved

## Características mediante CSS

- Al cargar la página, Modernizr añade al atributo class de la etiqueta <html> una clase de estilo por cada característica, prefijada por "no-" las no soportadas:

```
<html class=" js flexbox no-flexboxlegacy canvas canvastext webgl no-touch
geolocation postmessage no-websqldatabase indexeddb hashchange history
draganddrop websockets rgba hsla multiplebgs backgroundsize borderimage
borderradius boxshadow textshadow opacity cssanimations csscolumns
cssgradients no-cssreflections csstransforms csstransforms3d csstransitions
fontface generatedcontent video audio localstorage sessionstorage webworkers
applicationcache svg inlinesvg no-smil svgclippaths">
```

- La hoja de estilos se diseña en función a dichas clases:

```
.no-cssgradients .header {
    background: url("images/glossybutton.png");
}
.cssgradients .header {
    background-image: linear-gradient(cornflowerblue, rebeccapurple);
}
```

© JMA 2015. All rights reserved

## Modernizr.load()

- El método Modernizr.load() es una manera sencilla de cargar librerías sólo cuando los usuarios las necesitan, según una característica esté o no disponible, ahorrando ancho de banda y mejorando el rendimiento de la aplicación.
- Con éste método podemos indicar a los navegadores que no soporten ese API que carguen el polyfill correspondiente, de modo que se pueda utilizar esa característica de HTML5 en ellos también.

```
Modernizr.load({
    test: Modernizr.geolocation,
    yep : 'geo.js',
    nope: 'geo-polyfill.js'
});
```

© JMA 2015. All rights reserved

## Yepnope

- <http://yepnopejs.com>
- Un gestor de scripts para la carga dinámica y condicional.

```
yepnope({
  test : Modernizr.geolocation,
  yep  : 'normal.js',
  nope : ['polyfill.js', 'wrapper.js']
});
```

© JMA 2015. All rights reserved

## Carga dinámica de scripts

- La tendencia actual es no utilizar frameworks de carga dinámica de scripts.
- La alternativa es seguir un enfoque utilizando módulos y empaquetar la aplicación utilizando require.js, webpack, browserify o cualquier otra de las excelentes herramientas de compilación administradas por dependencias.
- Cuando se trata de cargar cosas condicionalmente, la sugerencia es generar una compilación para cada combinación de las cosas.
- Esto puede parecer que generará muchos archivos (y podría), pero las herramientas actuales son bastante buenas en eso.
- Luego se incorpora una secuencia de comandos inline en la página que solo carga (¡asincrónicamente!) un único script integrado que está sintonizado con las características de ese usuario. Toda la ganancia de rendimiento de la carga condicional, y ninguno de los problemas de latencia de cargar 100 cosas a la vez.

© JMA 2015. All rights reserved