

Desarrollo Web

© JMA 2015. All rights reserved

EVOLUCIÓN DEL DESARROLLO WEB

© JMA 2015. All rights reserved

1990

Cliente

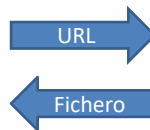
Tim Berners-Lee (CERN)

- ENQUIRE
 - Hipertexto
 - HTTP
 - HTML

Servidor

Robert Cailliau

- CERN httpd
 - Servidor de ficheros
 - HTTP – URL
 - Ficheros estáticos de texto



© 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

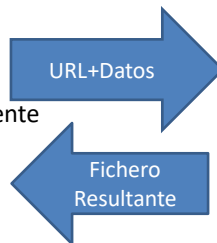
- Por la necesidad que el servidor Web pudiese devolver páginas Web dinámicas y no únicamente contenido estático residente en ficheros HTML se desarrolló la tecnología CGI (Common Gateway Interface) donde el servidor Web invocaba un programa el cual se ejecutaba, devolvía la página Web y el servidor 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

Cliente

- Navegadores
 - HTML 3.2
 - Formularios
 - Plug-in
 - Applet
 - ActiveX
 - Scripting de cliente
 - JavaScript



Servidor

- Servidor Web
 - Servidor de ficheros
 - Ficheros estáticos
 - Texto, HTML, Imágenes, ...
 - Ficheros dinámicos
 - Scripting de servidor
 - ASP
 - PHP
 - ...
 - JSP

© JMA 2015. All rights reserved

Scripting

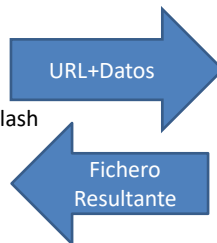
- CGI era una solución cómoda de realizar páginas Web dinámicas pero tenía un grave problema de rendimiento que lo hizo insostenible en cuanto la demanda de la Web comenzó a disparar las peticiones de los servidores Web.
- Para agilizar esto, los principales servidores Web del momento (Netscape e IIS) 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

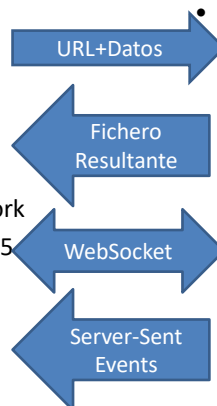
- Servidor Web
 - Servidor de ficheros
 - Ficheros estáticos
 - Ficheros dinámicos
 - Scripting de servidor
 - Servidor de aplicaciones
 - ASP.NET
 - J2EE
 - Web Services
 - WS XML
 - RestFul

© JMA 2015. All rights reserved

Actualidad

Cliente

- Navegadores
 - HTML 5
 - CSS 3
 - ~~RIA~~
 - Móviles
 - JS RIA Framework
 - EcmaScript 2015



Servidor

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

© JMA 2015. All rights reserved

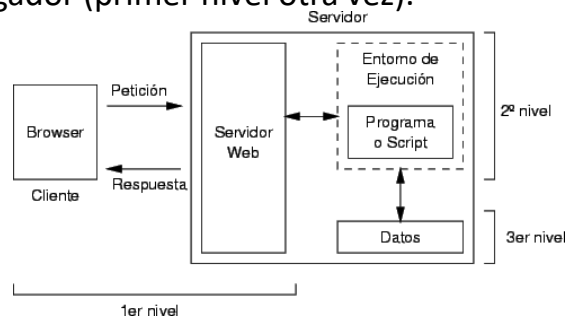
Aplicación Web

- 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 código 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 los 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/>
 - <https://www.w3counter.com/globalstats.php>

© JMA 2015. All rights reserved



© 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

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.

© JMA 2015. All rights reserved

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_caracter;
 - &letra_o_vocal_mnemotécnico;
- Mnemotécnicos:
 - acute (á), grave (è), circ (î), uml (Ü), tilde (Ñ), ring (â), ...

© JMA 2015. All rights reserved

Nombres de Entidades

Resultado	Descripción	Nombre Entidad	Número Entidad
	non-breaking space	 	
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
¢	cent	¢	¢
£	pound	£	£
¥	yen	¥	¥
€	euro	€	€
©	copyright	©	©
®	registered trademark	®	®
™	trademark	™	™

© JMA 2015. All rights reserved

27

Entidades habituales

á	á	Á	Á
é	é	É	É
í	í	Í	Í
ó	ó	Ó	Ó
ú	ú	Ú	Ú
ü	ü	Ü	Ü
ñ	ñ	Ñ	Ñ
¿	Símbolo ¿	¡	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 `
`
- 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/>` `
`.

© 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

- Título 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 paginas 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 via HTTP. Los servidores HTTP utilizaran 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 generara 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 pagina dentro de un documento extenso.
- Los atributos son:
 - REL: Tipo de relación con el documento.
 - HREF: Especifica la pagina enlazada.
 - HREFLANG: Idioma de la página relacionada. (opcional)
 - TYPE: Tipo MIME de la pagina, 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 esta 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" />
</frameset>
...
</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.
-
 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.
- `...Texto o imagen del enlace...`
- El texto del enlace aparece automáticamente subrayado de azul (o de morado si se visitó recientemente)
- Los atributos son:
 - ID: 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 `ejemplo` de enlaces.
- Para enlazar a otra parte de la misma página:
 - Inserte un enlace con nombre: ` Referencias`
 - Y enlace a la misma con: ` Mis referencias `
- Para enlazar a un anclaje con nombre de una página diferente:
 - ` Mis referencias `
- Para enviar un correo electrónico:
 - `...Texto para enviar mensaje...`
- Para recibir un fichero:
 - `...Texto indicando la descarga...`

© 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
 - ``
- 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 `` final, porque `` 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.

```


<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="poly" coords="140,121,181,116,204,160,204,270,124,122"
    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 reinializa 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
`

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

Radio buttons:

☐ male

☒ female

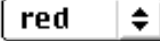
- Si uno o mas botones de radio tienen el mismo NAME (grupo), el usuario solo puede seleccionar uno de ellos, desmarcándose automáticamente el resto.
- Al servidor se envía el NAME común asociado al VALUE del INPUT seleccionado.
- Al igual que los checkboxes, los radio buttons no contienen texto indicativo.

© 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"><-- 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>Who are you?</p>
<form method="post" action="">
 <p>Name:
 <input type="text" name="textfield">
 </p>
 <p>Gender:
 <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

## 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:

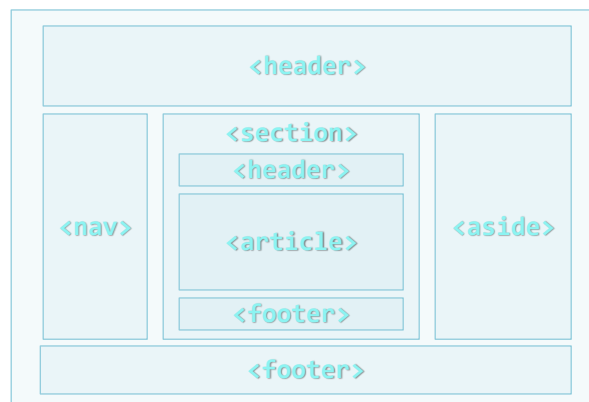
```
<link rel="stylesheet" href="style-original.css">
```

© 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 relacionado 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>Titulo de la sección</h1>
 <h2>Subtitulo</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 resaltado 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 URIs

lachlan.hunt@lachy.id.au

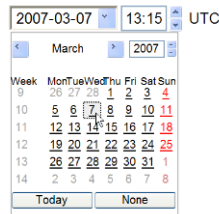
http://lachy.id.au

<http://lachy.id.au> Lachlan Hunt: Web Development Guru  
<http://lachy.id.au/log/> Lachy's Log

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

© JMA 2015. All rights reserved

## Fecha y hora



- `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-local`: para introducir una fecha y una hora.
- `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:

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 esta 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 códecs y, lo que es peor los mismo
 - 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 Oggtheora 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 vídeo.
 - descriptions:** Descripciones, indica que contiene texto que será mostrado cuando no esté disponible el componente de vídeo.
 - 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:ss.sss --> [hh:]mm:ss.sss

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 conteniendo 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.

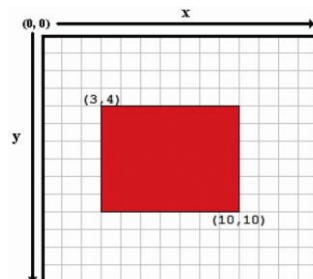
```
function iniciar(){
    var elemento=document.getElementById('lienzo');
    lienzo=elemento.getContext('2d');
}
window.addEventListener("load", iniciar, false);
```

- 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 del 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 rellenar 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(imágen, tipo)` El atributo `imágen` 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(imágen, 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(imágen, 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(imágen, 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 dsi 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 esta 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 sobre coste

© 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 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 {
    // Con 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 cacheada:
 - `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 metodo `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);
worker.addEventListener('error', function(e) {...}, 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.

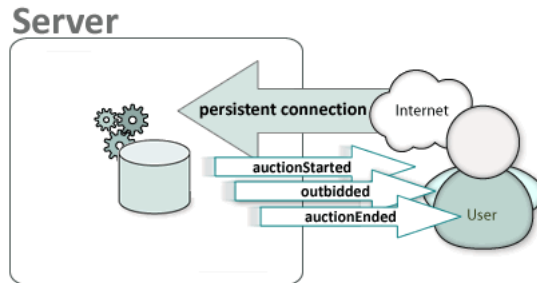
© JMA 2015. All rights reserved

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 amortizados

© 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 padre 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

Pseudoclasses o Pseudoelementos

- Las pseudoclasses 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 pseudoclasses clasifican las etiquetas sobre características diferentes a su nombre, atributos o contenido.
- La sintaxis de pseudoclasses o pseudoelementos:

```
selector:pseudo { ... }
selector.class:pseudo { ... }
```

© JMA 2015. All rights reserved

Pseudoclasses 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(/g/)	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

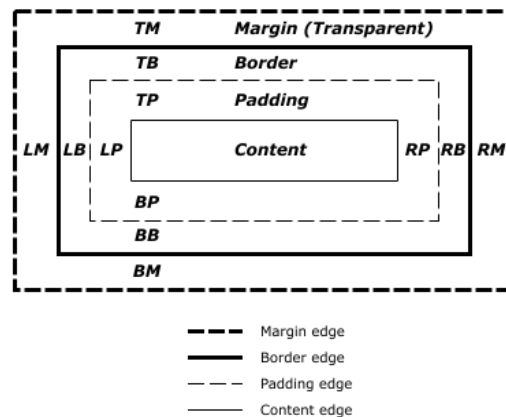
© JMA 2015. All rights reserved

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 "#ff0000"
 - 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

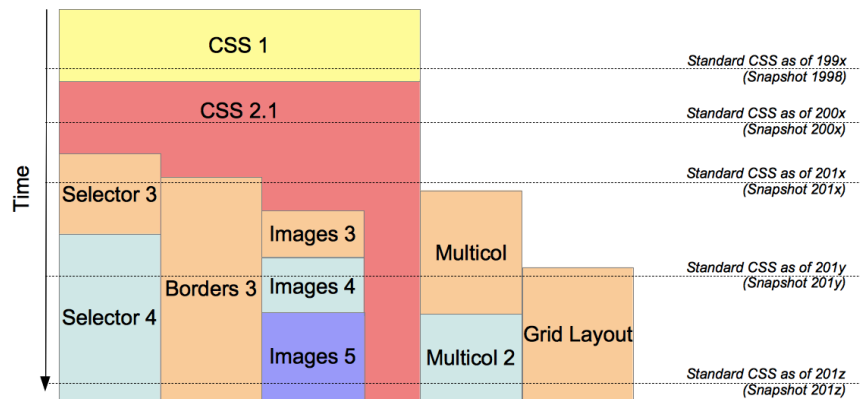
© JMA 2015. All rights reserved

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

Propiedades personalizadas (variables)

- En CSS, las propiedades personalizadas (también conocidas como variables) son entidades definidas por autores de CSS que contienen valores específicos que se pueden volver a utilizar en un documento.
- Se declaran en las reglas CSS con un nombre personalizado, sensible a la tipografía y prefijados por --(doble guion).


```
:root {
  --error-color: red;
}
```
- Pueden tener un alcance global o local. El alcance es global si se declara dentro del pseudo selector :root. El alcance local permite su uso en el selector donde se declara o en cualquier selector anidado, primando las definiciones mas locales.
- La función var() permite recuperar el valor de la variable para definir el valor de otra propiedad. Opcionalmente se puede definir un valor por defecto para cuando exista la variable.


```
.error-msg {
  color: var(--error-color, red);
  margin-top: calc(var(--gap) * 1px);
}
```
- Las variables CSS están en el DOM, por lo que se pueden cambiar con JavaScript.

© 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("nombreachivo.woff") format("woff"),
        url("nombreachivo.otf") format("opentype"),
        url("nombreachivo.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)
 - 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.google.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

Enalemanlaspa
abrasecon

Con word-wrap

Enalemanlaspa
abraseconform
anavecesuniend
oseaotraspalabr
asporloquesevu
elvenenormes

- 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.
 - background-color: 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

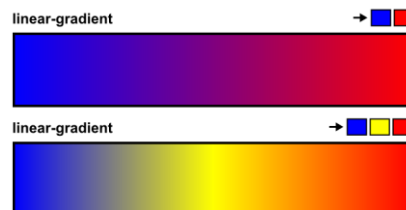
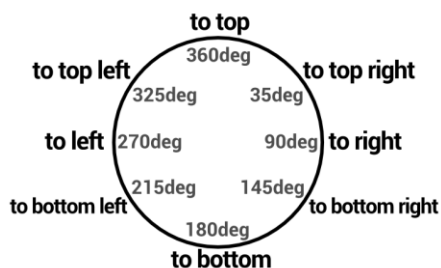
Gradientes de color

- 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], ...)



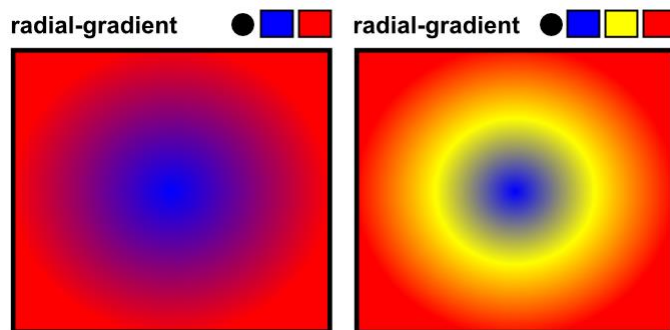
© JMA 2015. All rights reserved

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

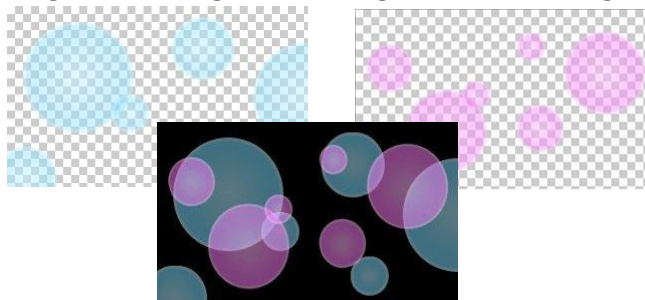


© JMA 2015. All rights reserved

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

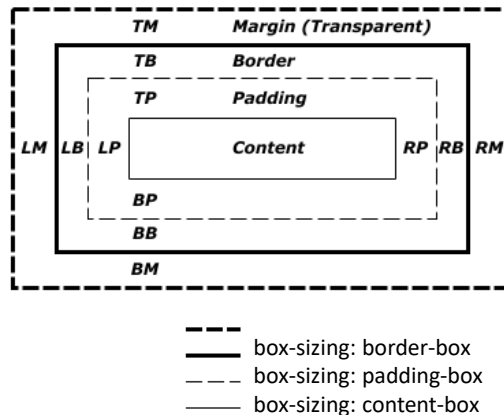
{width:300px; height:100px; border:1px; **box-sizing: border-box;**}

.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)

- El elemento padre que tiene establecido display: flex se llama contenedor flexible.
- Los elementos que se presentan como cajas flexibles dentro del contenedor flexible se denominan ítems o elementos flexibles.
- Los elementos dentro del contenedor se distribuyen con respecto a dos ejes:
 - El eje principal (main axis) es el eje que corre en la dirección en que se colocan los elementos flexibles. El inicio y el final de este eje se denominan inicio principal (main start) y final principal (main end).
 - El eje transversal (cross axis) es el eje que corre perpendicular a la dirección en la que se colocan los elementos flexibles. El inicio y el final de este eje se denominan inicio transversal (cross start) y extremo transversal (cross end).
- El eje puede ser horizontal (fila) o vertical (columna).
- El inicio y el final del eje es un concepto relativo que estará en función a la definición. El inicio puede estar a la izquierda o arriba y, con reverse, a derecha o abajo. Los valores -start y -end de las propiedades hacen referencia al main start y al main end.

© JMA 2015. All rights reserved

Modelo de caja flexible (Flexbox)

- Las propiedades que permiten definir al contenedor son:
 - flex-direction: define el eje principal y el inicio: row | row-reverse | column | column-reverse
 - flex-wrap: define si ajusta o desborda los elementos: nowrap | wrap | wrap-reverse
 - flex-flow: define la flex-direction y la flex-wrap
 - justify-content: especifica la alineación de los elementos del contenedor flexible cuando los elementos no utilizan todo el espacio disponible en el eje principal (horizontalmente): flex-start | flex-end | center | space-between | space-around | space-evenly
 - align-items: especifica la alineación predeterminada para los elementos dentro del contenedor flexible: stretch | center | flex-start | flex-end | baseline
 - align-content: especifica la alineación de las franjas desbordadas (wrap): stretch | center | flex-start | flex-end | space-between | space-around

© JMA 2015. All rights reserved

Modelo de caja flexible (Flexbox)

- Las propiedades que permiten definir ítems o elementos flexibles son:
 - `order`: especifica el orden de los ítems.
 - `flex-grow`: especifica el factor que crecerá un ítem en relación con el resto de los ítems.
 - `flex-shrink`: especifica el factor que encogerá un ítem en relación con el resto de los ítems.
 - `flex-basis`: especifica la longitud inicial (`height` o `width`) de un ítem.
 - `flex`: propiedad abreviada con `flex-grow`, `flex-shrink` y `flex-basis`
 - `align-self`: especifica la alineación del ítem seleccionado dentro del contenedor flexible (anula el valor `align-items` del contenedor): `auto` | `stretch` | `center` | `flex-start` | `flex-end` | `baseline`

© JMA 2015. All rights reserved

Grid Layout

- Grid Layout presenta un sistema de cuadrícula bidimensional para CSS. Las cuadrículas se pueden utilizar para posicionar áreas principales de la página o pequeños elementos de la interfaz de usuario.
- Una cuadrícula es un conjunto de líneas horizontales y verticales que separan filas y columnas, creando celdas. Los elementos se pueden colocar en la cuadrícula respetando estas columnas y filas.
- Se puede crear una cuadrícula con tamaños fijos (utilizando píxeles, por ejemplo) o tamaños flexibles (con porcentajes o con la nueva unidad de medida `fr`, fracción, diseñada para este propósito). Permite agregar filas y columnas adicionales cuando sea necesario.
- Se puede colocar elementos en una ubicación precisa en la cuadrícula utilizando números de línea, nombres o seleccionando un área de la cuadrícula. Grid también contiene un algoritmo para controlar la ubicación de elementos que no tienen una posición explícita en la cuadrícula. Contiene características de alineación para poder controlar la forma cómo se alinean los elementos una vez colocados en un área de cuadrícula y cómo está alineada toda la cuadrícula.

© JMA 2015. All rights reserved

Grid Layout

- El contenedor de Grid es un elemento que declara la propiedad `display: grid` o `display: inline-grid`. Todos los hijos directos de dicho elemento se convertirán en elementos de la cuadrícula.
- Las propiedades que permiten definir al contenedor son:
 - `grid-template-columns`: especifica los anchos (que determinan el número) de columnas.
 - `grid-template-rows`: especifica las alturas (que determinan el número) de filas.
 - `grid-template-areas`: define el diseño de la cuadrícula mediante los nombres de las áreas que van a alojar. Las filas se definen delimitándolas por apóstrofes, que contienen los nombres de las áreas, cada área en una columna. Para que un área ocupe más de una fila o columna es necesario repetir su nombre.
 - `grid-auto-columns`, `grid-auto-rows`: establece el tamaño de las columnas y de las filas autogeneradas (no definidas explícitamente).
 - `grid-auto-flow`: controla cómo se insertan los elementos sin posicionamiento explícito: `row` | `column` | `dense` | `row dense` | `column dense`
 - `grid-column-gap`, `grid-row-gap`, `grid-gap`: definen el tamaño del espacio entre las columnas y las filas. (CSS3: renombradas a `column-gap`, `row-gap` y `gap`)

© JMA 2015. All rights reserved

Grid Layout

- El posicionamiento se puede realizar indicando las líneas que delimitan la posición (empezando en 1) que puede contener varias celdas. Se puede simplificar indicando el número de posición de la columna o fila (empezando en 1), con `span` se puede indicar que ocupe mas de una. Si no se indica el posicionamiento se añade en la siguiente celda libre y el orden al escribir los elementos importa.
- Las propiedades que permiten posicionar los elementos en la cuadrícula:
 - `grid-column-start`, `grid-column-end`, `grid-row-start`, `grid-row-end`: definen las líneas que delimitan la posición.
 - `grid-column`, `grid-row`: especifican la ubicación parcial (y el tamaño) de un elemento en la cuadrícula en un diseño de cuadrícula.
 - `grid-area`: especifica la ubicación completa (y el tamaño) del elemento (abreviatura de las anteriores). También se puede utilizar para asignar un nombre del área si la cuadrícula se ha definido con `grid-template-areas`.

© 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

Cursores 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

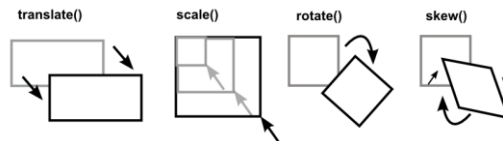
Transformaciones

- 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

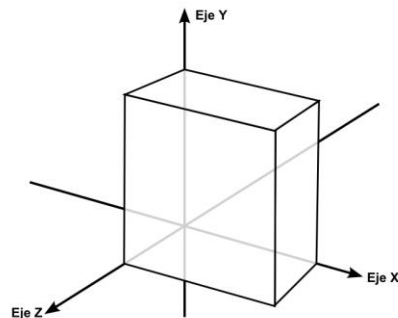
- `translate()`: mueve un elemento desde su posición actual (de acuerdo con los parámetros dados para el eje X y el eje Y). Métodos para un solo eje: `translateX()` y `translateY()`.
- `rotate()`: hace girar un elemento sobre un punto de acuerdo con un determinado grado.
- `scale()`: aumenta o disminuye 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` propiedad especifica un retraso (en segundos) para el efecto de transición.

© JMA 2015. All rights reserved

Animaciones

- Las 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

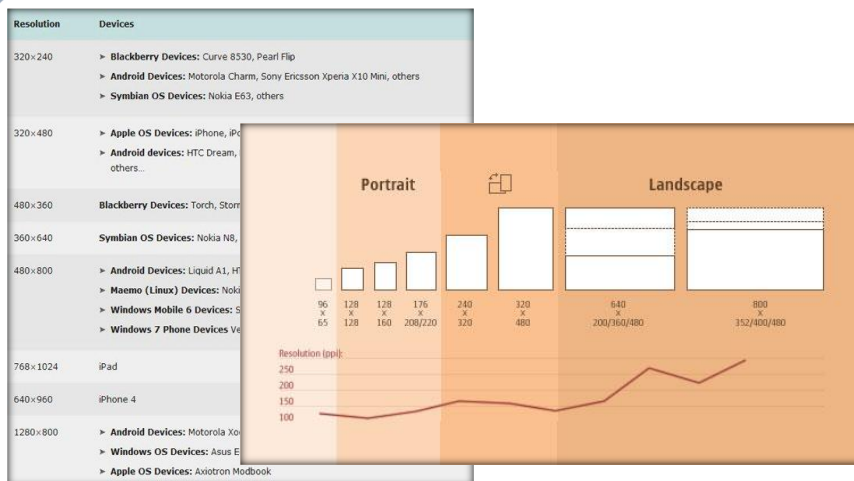
Resolución

- 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

Mobile First



Responsive Web Design

Mobile First Web Design



© 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

Unidades Relativas

Unidad	Relativa a
em	Tamaño de letra del elemento padre, en el caso de propiedades tipográficas como font-size, y tamaño de la fuente del propio elemento en el caso de otras propiedades, como width.
ex	Altura x de la fuente del elemento.
ch	La medida de avance (ancho) del glifo "0" de la letra del elemento.
rem	Tamaño de la letra del elemento raíz.
lh	Altura de la línea del elemento.
vw	1% del ancho de la ventana gráfica.
vh	1% de la altura de la ventana gráfica.
vmin	1% de la dimensión más pequeña de la ventana gráfica.
vmax	1% de la dimensión más grande de la ventana gráfica.

© 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;
}
```
- 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

METODOLOGÍAS PARA ORGANIZAR EL CÓDIGO CSS

© JMA 2015. All rights reserved

Introducción

- En los sitios de web más pequeños, la forma en que organiza sus estilos no suele ser una gran preocupación. Entrás ahí, escribes algo de CSS, o tal vez incluso algo de SASS. Lo compila todo en una sola hoja de estilo con la configuración de producción de SASS y se agrega para obtener todas las hojas de estilo de los módulos en un paquete ordenado y agradable.
- Sin embargo, cuando se trata de proyectos más grandes y complejos, la forma en que se organiza el código es la clave para la eficiencia y afecta en al menos tres aspectos: cuánto código se tendrá que escribir, cuánto tiempo lleva escribir código, y cuánto tendrá que cargar su navegador. Esto se vuelve especialmente importante cuando se trabaja con equipos de temas y cuando el alto rendimiento es esencial.
- Existen muchas metodologías que tienen como objetivo reducir la huella de CSS, organizar la cooperación entre programadores y mantener grandes bases de código CSS. Esto es obvio en proyectos grandes como Twitter, Facebook y Github , pero otros proyectos a menudo evolucionan a un estado de "archivo CSS enorme" con bastante rapidez.
 - OOCSS: Separar contenedor y contenido con "objetos" CSS
 - BEM: Organizar pos Bloques Elementos Modificadores
 - SMACSS: Guía de estilo para escribir su CSS con cinco categorías para reglas CSS
 - SUITCSS: Nombres de clases estructurados y guiones significativos
 - Atomic: Dividiendo estilos en piezas atómicas o indivisibles

© JMA 2015. All rights reserved

OOCSS

- La metodología Object-Oriented CSS (OOCSS) es un enfoque para que escribir CSS sea rápido, fácil de mantener y basado en estándares. Agrega una previsibilidad muy necesaria a CSS para que incluso los principiantes puedan participar en la redacción de hermosos sitios web.
- Básicamente, un "objeto" de CSS es un patrón visual repetido, que se puede abstraer en un fragmento independiente de HTML , CSS y posiblemente JavaScript. Luego, dicho objeto se puede reutilizar en todo un sitio.
- Se basa en los siguientes dos principios:
 - Separar la estructura del diseño.
 - Separar contenedor del contenido.

© JMA 2015. All rights reserved

OOCSS

- En CSS hay un grupo de propiedades de estructura que son invisibles al usuario, pero modifican el tamaño y posición de un elemento, y otro grupo de propiedades que modifican el aspecto visual de un elemento, como los colores.
- Para aplicar la regla de separar la estructura del diseño en OOCSS se diferencian las propiedades que modifican la estructura de las propiedades que modifican el diseño, para ello se crean diferentes estilos sin mezclar propiedades de estos dos grupos.
- Esto significa definir características visuales repetidas (como estilos de fondo y borde) como "máscaras" separadas que se pueden mezclar y combinar con los diversos objetos para lograr una gran cantidad de variedad visual sin mucho código.
- La separación de estructura y aspecto también puede significar el uso de clases para nombrar los objetos y sus componentes, en lugar de depender únicamente de la semántica de HTML.
- Dado que "rara vez se utilizan estilos que dependen de la ubicación", la segunda regla trata de evitar los selectores anidados, que los estilos dependan del contenedor en el que están dada la estructura de HTML ya que esto hace que el estilo aplicado al contenido no sea reutilizable.

© JMA 2015. All rights reserved

OOCSS

```

/* structure */
.block {
  width: 200px;
  padding: 10px;
}
.line {
  padding: 10px;
}
/* skin */
.red {
  border: solid 1px #000;
  background: linear-gradient(#ccc, #f00);
}
.blue {
  border: solid 1px #000;
  background: linear-gradient(#ccc, #00f);
}

```

```

<div class="block red"></div>
<div class="block blue"></div>
<div class="line red"></div>
<div class="line blue"></div>

```

© JMA 2015. All rights reserved

BEM

- BEM significa Bloques Elementos Modificadores (Block Element Modifier) por sus siglas en inglés. Sugiere una manera estructura de nombrar tus clases, basado en las propiedades del elemento en cuestión, para que puedas crear una estructura de código más consistente, no duplicar estilos y traer claridad al código definiendo y estableciendo mejor la jerarquía en el proyecto. (<http://getbem.com/>)
- Si alguna vez has visto un nombre de una clase como `header__from--login` eso es precisamente BEM en acción.
- Cuando utilices la metodología BEM, debes tomar en cuenta que solamente se usan nombres de clases (no IDs). Los nombres de clases permiten repetir el nombre BEM si es necesario, así como crear una estructura de código más consistente (en ambos archivos el HTML y CSS/Saas).

© JMA 2015. All rights reserved

Características

- Ventajas:
 - Ofrece un sistema para organizar y nombrar clases.
 - Permite trabajar de forma coordinada y asíncrona a los miembros de un equipo.
 - Permite conocer de un vistazo la estructura, tanto del documento HTML, como de las clases CSS.
 - Permite evolucionar y modificar elementos con facilidad (creando clases de modificadores)
- Inconvenientes:
 - Hay que ajustarse a las normas y reglas de la metodología.
 - La cantidad de clases utilizadas puede ser más elevada que otras alternativas.
 - La teoría de que "cambiando una clase se cambia el aspecto de una web", no aplica aquí: para ser coherente hay que crear clases que modifique el aspecto de los elementos, y añadirlos al código HTML.

© JMA 2015. All rights reserved

Bloque

- Encapsula una entidad independiente que es significativa por sí misma. Si bien los bloques se pueden anidar e interactuar entre sí, semánticamente permanecen iguales; no hay precedencia ni jerarquía. Las entidades holísticas sin representación DOM (como controladores o modelos) también pueden ser bloques.
- Nomenclatura:
 - Los nombres de bloque pueden constar de letras latinas, dígitos y guiones. En los nombres complicados que requieran mas de una palabra se utiliza un guion como separador.
- HTML
 - Cualquier nodo DOM puede ser un bloque si acepta un nombre de clase.
`<div class = "block"> ... </div>`
- CSS
 - Usar solo el selector de nombre de clase, sin nombre de etiqueta ni ID, sin dependencia de otros bloques o elementos en una página.
 - Bien:


```
.block {color: # 042; }
```

© JMA 2015. All rights reserved

Elemento

- Partes de un bloque que no tienen un significado independiente, cualquier elemento está semánticamente ligado a su bloque.
- Nomenclatura
 - Los nombres de los elementos pueden consistir en letras latinas, dígitos, guiones y guiones bajos. La clase CSS se forma como un nombre de bloque más dos guiones bajos más el nombre del elemento: `block__elem`
- HTML
 - Cualquier nodo DOM dentro de un bloque puede ser un elemento. Dentro de un bloque dado, todos los elementos son semánticamente iguales.
`<div class = "bloquear">`
:
` `
`</div>`
- CSS
 - Usar solo el selector de nombre de clase, sin nombre de etiqueta ni ID, sin dependencia de otros bloques o elementos en una página
 - Bien


```
.block__elem {color: # 042; }
```
 - Malo


```
.block .block__elem {color: # 042; }  
div.block__elem {color: # 042; }
```

© JMA 2015. All rights reserved

Modificador

- Cualificador de bloques o elementos usado para cambiar su apariencia, comportamiento o estado.
- Nomenclatura
 - Los nombres de los modificadores pueden consistir en letras latinas, dígitos, guiones y guiones bajos. La clase CSS se forma como el nombre del bloque o elemento más dos guiones: `.block--mod` o `.block__elem--mod` y `.block--color-black` con `.block--color-red`.
- HTML
 - El modificador es un nombre de clase adicional que se agrega a un nodo DOM de bloque/elemento solo cuando se cualifican, conservándose la clase original:
 - Bien


```
<div class="block block--mod">...</div>
<div class="block block--size-big block--shadow-yes">...</div>
```
 - Malo


```
<div class="block--mod">...</div>
```
- CSS
 - Usar el nombre de la clase modificadora como selector: `.block--hidden { }`
 - Para alterar elementos basados en un modificador a nivel de bloque:
 - `.block--mod .block__elem { }`
 - Modificador de elemento: `.block__elem--mod { }`

© JMA 2015. All rights reserved

Ejemplo

- Para un bloque de formulario con elementos input y submit y modificadores `theme:"xmas"` y `simple: true`, mas un elemento submit con su propio modificador `disabled: true` para no enviar el formulario mientras no está lleno.
- HTML


```
<form class="form form--theme-xmas form--simple">
  <input class="form__input" type="text" />
  <input type="submit" class="form__submit form__submit--disabled" />
</form>
```
- CSS


```
.form { }
.form--theme-xmas { }
.form--simple { }
.form__input { }
.form__submit { }
.form__submit--disabled { }
```

© JMA 2015. All rights reserved

SMACSS

- SMACSS son las siglas de Scalable and Modular Architecture for CSS (Arquitectura en CSS Escalable y Modular) y es una metodología CSS basada en la categorización. Al clasificar las reglas CSS, comenzamos a ver patrones y podemos definir mejores prácticas en torno a cada uno de estos patrones.
- Cada categoría tiene ciertas pautas que se aplican a ella. Gran parte del propósito de categorizar cosas es codificar patrones, cosas que se repiten dentro de nuestro diseño. La codificación de patrones implica menos código, mantenimiento más fácil y mayor consistencia en la experiencia del usuario. Las excepciones a la regla pueden ser ventajosas, pero deben justificarse.
- Se clasifican las reglas en las cinco categorías. Se utiliza una convención de nomenclatura que es beneficiosa para comprender de inmediato a qué categoría pertenece un estilo en particular y su función dentro del alcance general de la página. En proyectos grandes, es más probable que los estilos se dividan en varios archivos, la convención de nomenclatura también facilita encontrar a qué archivo pertenece un estilo.

© JMA 2015. All rights reserved

Categorización

- Reglas básicas: se establecen los estilos por defectos de los elementos HTML individuales, los típicos selectores CSS de tipo. Por ejemplo, reset, html, body, button, h1, etc.
- Reglas del layout: se divide la página en secciones (header, main content, footer, navigation, etc.) y se asignan los estilos relacionados con su estructura.
- Reglas de módulos: elementos que pueden ser reusables, modulares e independientes del contexto. Por ejemplo: llamadas a la acción o galerías de imágenes.
- Reglas de estados: se define el comportamiento del layout y sus módulos en diferentes estados: hover, active, diferentes resoluciones, etc.
- Reglas de temas: estilos que afectan al layout y módulos. Estas reglas son opcionales, y las puedes utilizar para estilos que puede que quieras reemplazar (por ejemplo, uno oscuro y uno claro).

© JMA 2015. All rights reserved

Reglas de nomenclatura

- Estas reglas son mas recomendaciones que imposiciones, como utilizar el - como separador. Se usa un prefijo para diferenciar entre reglas de diseño, estado y módulo.
- Para Layout, se utiliza l- pero layout- funcionaría igual de bien. El uso de prefijos como grid- también proporciona suficiente claridad para separar los estilos de diseño de otros estilos.
- Los módulos serán la mayor parte de cualquier proyecto. Como resultado, hacer que cada módulo comience con un prefijo como .module- sería innecesariamente detallado. Los módulos solo usan el nombre del módulo en sí. Los elementos relacionados dentro de un módulo usan el nombre base como prefijo. Los módulos que son una variación de otro módulo también deben usar el nombre del módulo base como prefijo.
- Para las reglas de estado, se puede utilizar is- como en is-hidden o is-collapsed, se pueden combinar con el módulo para estados específicos como is-tab-active. Esto hace que sean muy legibles.
- Los temas pueden tener nombres de clases que indiquen claramente qué estilos son parte del tema y cuáles no.

© JMA 2015. All rights reserved

Ejemplos

```
/* Módulos y submódulos */
.header, .article, .footer, .nav { ... }
.article { ... }
.article-title { ... }
.article-text { ... }
/* Estados */
.is-selected { ... }
.article.is-selected, is-article-selected { ... }
/* Layout */
.l-fixed { ... }
.l-flex { ... }
.l-fixed .article { ... }
/* Temas */
.theme-dark { ... }
.theme-light { ... }
.theme-dark .article { ... }
```

© JMA 2015. All rights reserved

SASS

© JMA 2015. All rights reserved

Contenidos

- Introducción e instalación
 - Sintaxis, Comentarios, Expresiones
 - Variables, Interpolación, Alcance
 - Importaciones y Módulos
 - Reglas, anidamiento, selectores
 - Reglas de control de flujo
 - Mixin/Include, extend, function
 - Depuración
-

© 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

SASS

- Sass (acrónimo de Syntactically Awesome StyleSheets) es un metalenguaje dinámico de hojas de estilo, basado en la sintaxis de CSS, a la que ha incorporado variables, anidamiento, herencia, operadores, mixins y funciones, lo que es válido en CSS es válido en Sass con la misma semántica.
- Diseñado por inicialmente diseñado por Hampton Catlin y desarrollado por Natalie Weizenbaum.
- La implementación oficial de Sass es open-source y está escrita en Ruby, sin embargo existen otras implementaciones, incluyendo una en JavaScript para NodeJS.
- Sass dispone en dos sintaxis. La sintaxis original, llamada indented syntax (sintaxis con sangrado) que usa una sintaxis similar al Haml. Este usa la sangría para separar bloques de código y el carácter nueva línea para separar reglas. La sintaxis más reciente, SCSS, usa el formato de bloques como CSS. Este usa llaves para denotar bloques de código y punto y coma (;) para separar las líneas dentro de un bloque.
- En función a la sintaxis, los ficheros tienen las extensiones .sass o .scss.

© JMA 2015. All rights reserved

Uso

- Instalación:
 - `npm install -g sass`
- Compilación en línea de comandos:
 - `sass styles.scss styles.css`
- Compilación continua (detección de cambios) en línea de comandos:
 - `sass --watch styles.scss styles.css`
- Hay disponibles extensiones para los principales editores y entornos de desarrollo que generan automáticamente la versión `.css` al guardar los ficheros `.sass` o `.scss`

© JMA 2015. All rights reserved

Sintaxis

- Una hoja de estilo de Sass se compone de una serie de declaraciones, que se evalúan para construir el CSS resultante.
- Algunas declaraciones pueden tener bloques, sangrados o definidos mediante `{ y }`, que contienen otras declaraciones.
- Sass soporta todas las reglas `@` (también llamadas "reglas at") definidas por CSS3. Además, Sass incluye varias reglas específicas llamadas directivas.
- Las declaraciones pueden ser:
 - Universales: Estos tipos de declaraciones se pueden usar en cualquier lugar de una hoja de estilo de Sass:
 - Declaraciones de variables, como `$var: value`.
 - Control de flujo en reglas, como `@if` y `@each`.
 - Las reglas `@error`, `@warn` y `@debug`.

© JMA 2015. All rights reserved

Sintaxis

- Las declaraciones pueden ser:
 - CSS: Estas declaraciones producen CSS. Se pueden usar en cualquier lugar excepto dentro de `@function`:
 - Reglas de estilo, como `h1 { /* ... */ }`.
 - CSS at-rules, como `@media` y `@font-face`.
 - Mixin usando `@include`.
 - La regla `@at-root`.
 - De nivel superior: Estas declaraciones solo se pueden usar en el nivel superior de una hoja de estilo o anidadas dentro de una declaración CSS en el nivel superior:
 - Cargas de módulo y importaciones, usando `@use` y `@import`.
 - Definiciones Mixin usando `@mixin`.
 - Definiciones de funciones usando `@function`.
 - Otras declaraciones
 - La regla `@extend` solo se puede utilizar dentro de las reglas de estilo.

© JMA 2015. All rights reserved

Comentarios

- Sass soporta el mismo tipo de comentarios que CSS, que utilizan los delimitadores `/*` y `*/` y pueden ocupar una o más líneas.
- Además, Sass también soporta los comentarios en línea (hasta el final de la línea) que utilizan los delimitadores `//` y que son muy comunes en todos los lenguajes de programación.
- La principal diferencia entre estos dos tipos de comentarios es que los comentarios tradicionales (`/* ... */`) se añaden en el código CSS generado, mientras que los comentarios en línea (`// ...`) se eliminan y no aparecen en el código CSS generado.
- Los comentarios de documentación permiten documentar las librerías, son comentarios silenciosos, son comentarios en línea pero escritos con tres barras (`///`) directamente encima de lo que está documentando.

© JMA 2015. All rights reserved

Importaciones

- Sass mejora la regla `@import` de CSS para poder importar también archivos SCSS y Sass. Todos los archivos importados, independientemente de su tipo, acaban fusionándose antes de generar el archivo CSS final. Además, cualquier variable o mixin definidos en los archivos importados se pueden utilizar en la hoja de estilos principal.
`@import "library";`
- La regla se deja tal cual al generar el CSS si:
 - la extensión del archivo importado es `.css`
 - el nombre del archivo empieza por `http://`
 - el nombre del archivo se indica mediante `url()`
 - la regla `@import` tiene alguna media query

© JMA 2015. All rights reserved

Expresiones

- Una expresión es todo lo que se encuentra en el lado derecho de una declaración de propiedad o variable. Cada expresión produce un valor. Cualquier valor de propiedad CSS válido también es una expresión Sass, pero las expresiones Sass son mucho más poderosas que los valores CSS simples. Se pasan como argumentos a mixins y funciones, se utilizan para controlar el flujo con la regla `@if` o se calculan aritméticamente.
- Sass define una serie de operaciones:
 - `==`, `!=`, `<`, `<=`, `>` y `>=` se utilizan para comparar dos valores.
 - `+`, `-`, `*`, `/` y `%` tienen su significado matemático habitual para los números.
 - `and`, `or` y `not` tienen el comportamiento booleano habitual. Sass considera que todos los valores son "verdaderos" excepto `false` y `null`.
 - `+`, `-` y `/` se puede utilizar para concatenar cadenas.
 - `(y)` se puede utilizar para controlar explícitamente el orden de precedencia de las operaciones.

© JMA 2015. All rights reserved

Expresiones

- SASS permite expresiones aritméticas 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

Variables

- Permiten asociar nombres a valores y utilizar los nombres en sustitución de los valores reales en las reglas. Sass usa el \$ símbolo para convertir algo en variable.
- Durante la traducción, los nombres se sustituyen por valores reales en el documento CSS de salida.
`$colorLetra: #3d3d3d;`
`$tamanoLetra: 12pt;`

`#titulo {`
 `color: $colorLetra;`
 `font-size:$tamanoLetra;`
`}`
`h2 {`
 `color: $colorLetra;`
`}`

© JMA 2015. All rights reserved

Interpolación

- Si la variable contiene el nombre de una clase, etiqueta, propiedad o parte de una cadena debe ir entre #{ } 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

Anidamiento

- SASS 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: $tamanoLetra;
  a { ... }
  a:hover { ... }
}
```
- Genera:

```
.menu { font-size: $tamanoLetra; }
.menu a { ... }
.menu a:hover { ... }
```

© 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

Propiedades anidadas

- CSS define varias propiedades cuyos nombres parecen estar agrupados de forma lógica. Así por ejemplo, las propiedades font-family, font-size y font-weight están todas relacionadas con el grupo font. En CSS es obligatorio escribir el nombre completo de todas estas propiedades. Sass permite:

```
.funky {
  font: {
    family: fantasy;
    size: 30em;
    weight: bold;
  }
}
```

© JMA 2015. All rights reserved

Módulos

- Se pueden crear archivos Sass parciales que contengan pequeños fragmentos para incluirlos en otros archivos Sass. Esta es una excelente manera de modularizar su CSS y ayudar a que las cosas sean más fáciles de mantener.
- Un módulo es un archivo parcial Sass nombrado con un guion bajo inicial (Ej: `_base.scss`). El guion bajo le permite a Sass saber que el archivo es solo un archivo parcial y que no debe generar un archivo CSS. Los Sass parciales o módulos se utilizan con la regla `@use`.

```
@use 'base';
h2 {
  color: $colorLetra;
}
```
- Cuando se usa un archivo, no necesita incluir la extensión del archivo ni el guion bajo inicial. Sass es inteligente y lo resolverá por sí mismo.

© JMA 2015. All rights reserved

Mixin / Include

- 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. Incluso puede pasar valores para hacer que su mezcla sea más flexible.
- Para crear un mixin, se usa la directiva `@mixin` que le da un nombre. Usando una variable `$` dentro de paréntesis se pueden pasar valores, con `:` se le asigna el valor por defecto. El mixin se puede usar como una declaración CSS comenzando por `@include` seguido del nombre del mixin.

```
@mixin theme($theme: DarkGray) {
  background: $theme;
  box-shadow: 0 0 1px rgba($theme, .25);
  color: #fff;
}
.info {
  @include theme;
}
.alert {
  @include theme($theme: DarkRed);
}
```

© JMA 2015. All rights reserved

Extender / Herencia

- Esta es una de las características más útiles de Sass. El uso de `@extend` permite compartir un conjunto de propiedades CSS de un selector a otro, crea una regla con todas las clases que lo extienden, a diferencia del mixin que incluye (duplica) su contenido cada vez que se utiliza. Un selector "marcador de posición" es un tipo especial de clase, precedida por %, que solo se genera cuando está extendida y ayudar a mantener limpio y ordenado el CSS generado.

```
%message-shared
  border: 1px solid #ccc
  padding: 10px
.message
  @extend %message-shared
.success
  @extend %message-shared
  border-color: green
// CSS generado
.message, .success {
  border: 1px solid #ccc;
  padding: 10px;
}
.success {
  border-color: green;
}
```

© JMA 2015. All rights reserved

Alcance de las variables

- Las variables declaradas en el nivel superior de una hoja de estilo son globales . Esto significa que se puede acceder a ellas en cualquier lugar de su módulo después de haber sido declaradas. Las declaradas en bloques (llaves en SCSS o código sangrado en Sass) suelen ser locales y solo se puede acceder a ellas dentro del bloque en el que fueron declaradas.
- Las variables locales incluso se pueden declarar con el mismo nombre que una variable global. Si esto sucede, en realidad hay dos variables diferentes con el mismo nombre: una local y otra global.
- Normalmente, cuando asigna un valor a una variable, si esa variable ya tenía un valor, se sobrescribe su valor anterior. Pero si se está escribiendo una biblioteca Sass, es posible que se desee permitir que los usuarios configuren las variables de su biblioteca antes de usarlas para generar CSS.
- La bandera `!default` asigna un valor a una variable solo si esa variable no está definida o su valor es null. De lo contrario, se utilizará el valor existente. Para cargar y asignar los valores por defecto se utiliza clausula `with` de `@use`.

```
// _library.scss
$black: #000 !default;
:
// style.scss
@use 'library' with ( $black: #222 );
```

© JMA 2015. All rights reserved

Reglas de control de flujo

- Sass proporciona una serie de reglas que permiten controlar si los estilos se generan o se deben generar varias veces con pequeñas variaciones. También se pueden usar en mixins y funciones para escribir pequeños algoritmos para facilitar la escritura de Sass. Sass admite cuatro reglas de control de flujo:
 - `@if` controla si se evalúa o no un bloque.
 - `@each` evalúa un bloque para cada elemento en una lista o cada par en un mapa.
 - `@for` evalúa un bloque un cierto número de veces.
 - `@while` evalúa un bloque mientras que se cumple una determinada condición.
- Las reglas de control son una característica muy avanzada que rara vez se utiliza directamente en las hojas de estilos. Sin embargo, son muy útiles para definir mixins y otras características avanzadas de librerías como Compass.

© JMA 2015. All rights reserved

La regla `@if` `@else`

- La regla `@if` evalúa una expresión SASS y solamente incluye los estilos definidos en su interior si la expresión devuelve un valor distinto a `false` o `null`.
- `@if` puede ir seguida de una o más reglas `@else if` y una última regla `@else`. Si la expresión evaluada por `@if` es `false` o `null`, Sass evalúa por orden el resto de reglas `@else if` hasta que alguna no devuelva `false` o `null`. Si ninguna regla `@else if` llega a ejecutarse, pasará a ejecutar la regla `@else` si existe.


```
$type: monster;
p {
  @if $type == ocean {
    color: blue;
  } @else if $type == purple {
    color: red;
  } @else {
    color: black;
  }
}
```
- La función `if()` permite tomar decisiones en una expresión.


```
margin-left: if($type == ocean, 10px, 15px);
```

© JMA 2015. All rights reserved

La regla @each

- La regla @each facilita la emisión de estilos o la evaluación de código para cada elemento de una lista o cada par en un mapa. Es ideal para estilos repetitivos que solo tienen algunas variaciones entre ellos. El bloque se evalúa para cada elemento de la lista por turno, que se asigna al nombre de variable dado.
- Con listas:


```
$sizes: 40px, 50px, 80px;
@each $size in $sizes {
  .icon-#{ $size } { font-size: $size; height: $size; width: $size; }
}
```

 - Genera:


```
.icon-40px { font-size: 40px; height: 40px; width: 40px; }
.icon-50px { font-size: 50px; height: 50px; width: 50px; }
.icon-80px { font-size: 80px; height: 80px; width: 80px; }
```
- Con mapas:


```
$icons: ("eye": "\f112", "start": "\f12e", "stop": "\f12f");
@each $name, $glyph in $icons {
  .icon-#{ $name }:before { display: inline-block; font-family: "Icon Font"; content: $glyph; }
}
```

 - Genera:


```
.icon-eye:before { display: inline-block; font-family: "Icon Font"; content: "\f112"; } ...
```

© JMA 2015. All rights reserved

La regla @for

- La regla @for muestra repetidamente un conjunto de estilos. En cada repetición se utiliza el valor de una variable de tipo contador para ajustar el resultado mostrado. La directiva puede utilizar dos sintaxis: @for \$var from <inicio> through <final> and @for \$var from <inicio> to <final>.
- El \$var es una variable, mientras que <inicio> y <final> son expresiones que deben devolver números enteros. Cuando el valor de <inicio> es mayor que el de <final> el valor del contador se decrementa en vez de incrementarse.
- En cada repetición del bucle, la directiva @for asigna a la variable \$var el valor del contador y repite los estilos utilizando el nuevo valor de \$var. Con through, los estilos se repiten desde <inicio> hasta <final> (ambos inclusive). Con to los estilos se repiten desde <inicio> hasta <final>, sin incluir el último.


```
@for $i from 1 to 4 {
  .item-#{ $i } { width: 2em * $i; }
}
```
- Genera:


```
.item-1 { width: 2em; }
.item-2 { width: 4em; }
.item-3 { width: 6em; }
```

© JMA 2015. All rights reserved

La regla @while

- La regla @while repite indefinidamente los estilos hasta que la expresión da como resultado false. Aunque esta directiva se usa muy poco, se puede utilizar para crear bucles más avanzados que los que se crean con la directiva @for.

```
$i: 2;
@while $i <= 10 {
  .item-#{ $i } { width: 2em * $i; }
  $i: $i + 2;
}
```

- Genera:

```
.item-2 { width: 4em; }
.item-4 { width: 8em; }
.item-6 { width: 12em; }
.item-8 { width: 16em; }
.item-10 { width: 20em; }
```

© JMA 2015. All rights reserved

Funciones

- Las funciones permiten definir cálculos complejos de valores que se pueden reutilizar en toda la hoja de estilo. Facilitan la abstracción de fórmulas y comportamientos comunes de una manera legible.
- Las funciones se definen utilizando la regla at @function. El nombre de una función puede ser cualquier identificador Sass. La regla at @return indica el valor a usar como resultado de la llamada a la función. Las funciones se llaman utilizando la sintaxis de función CSS normal.

```
@function pow($base, $exponent: 2) {
  $result: 1;
  @for $_ from 1 through $exponent {
    $result: $result * $base;
  }
  @return $result;
}

.sidebar {
  margin-left: pow(4, $factor) * 1px;
}
```

© JMA 2015. All rights reserved

Funciones

- Los parámetros permiten personalizar el comportamiento de las funciones cada vez que se llaman. Los parámetros se especifican en la regla `@function` después del nombre de la función, como una lista de nombres de variables entre paréntesis. La función debe llamarse con el mismo número de argumentos que parámetros. Los valores de estas expresiones están disponibles dentro del cuerpo de la función como las variables correspondientes.
 - Normalmente, cada parámetros que declara una función debe pasarse cuando se invoca esa función. Sin embargo, un parámetros es opcional al definir un valor predeterminado que se usará si esos argumentos no se pasan. Los valores predeterminados utilizan la misma sintaxis : que las declaraciones de variables.
 - Cuando se llama a una función, los argumentos se pueden pasar por nombre además de pasarlos por su posición en la lista de argumentos. Esto es especialmente útil para funciones con múltiples argumentos opcionales o con argumentos booleanos cuyos significados no son obvios sin un nombre que los acompañe.
- margin-left: pow(\$base: 4, 3) * 1px;

© JMA 2015. All rights reserved

Funciones

- Además de las funciones definidas por el usuario, Sass proporciona una biblioteca de funciones integradas que siempre están disponibles para su uso que proporciona una gran variedad de funciones que transforman colores, manipulan cadenas y hacen cálculos matemáticos.
- Estos módulos se pueden cargar con la regla `@use` como cualquier hoja de estilo definida por el usuario, y sus funciones se pueden llamar como cualquier otro miembro del módulo. Sass proporciona los siguientes módulos integrados:
 - `sass:math` proporciona funciones que operan con números.
 - `sass:string` facilita la combinación, la búsqueda o la división de cadenas.
 - `sass:color` genera nuevos colores basados en los existentes, lo que facilita la creación de temas de color.
 - `sass:list` permite acceder y modificar valores en listas.
 - `sass:map` permite buscar el valor asociado con una clave en un mapa y mucho más.
 - `sass:selector` proporciona acceso al potente motor selector de Sass.
 - `sass:meta` expone los detalles del funcionamiento interno de Sass.

© JMA 2015. All rights reserved

La regla @at-root

- Las directivas @at-root hacen que una o más reglas se generen en la raíz de la hoja de estilos en vez de anidarse en sus selectores. Se puede utilizar tanto con selectores individuales como con bloques de selectores.

```
// selector individual
.parent {
  @at-root .child { ... }
}

// bloques de selectores
.parent {
  @at-root {
    .child1 { ... }
    .child2 { ... }
  }
}
```

- Genera:


```
.child { ... }
.child1 { ... }
.child2 { ... }
```

© JMA 2015. All rights reserved

Depuración

- La regla @debug muestra por la consola el valor de la expresión Sass indicada. Se trata de una regla útil para depurar hojas de estilos muy complejas y que utilizan expresiones Sass muy avanzadas.

```
@debug 10em + 12em;
```

- Las reglas @warn y @error muestran el valor de una expresión Sass en forma de mensaje de aviso o error. Se trata de una regla muy útil para que los creadores de las librerías avisen a los diseñadores sobre el uso de características que se han declarado obsoletas y para mostrar errores en el uso de mixins que Sass ha podido corregir automáticamente. A diferencia de @debug:
 - Se pueden desactivar los mensajes de error con la opción --quiet de la línea de comandos o con la opción de configuración :quiet de Sass.
 - Los mensajes de error también se incluyen en la hoja de estilos generada para que el usuario pueda ver tanto los errores como el lugar exacto en el que se producen.

```
@mixin adjust-location($x, $y) {
  @if unitless($x) {
    @warn "Assuming #{ $x } to be in pixels";
    $x: 1px * $x;
  }
  @if unitless($y) {
    @warn "Assuming #{ $y } to be in pixels";
    $y: 1px * $y;
  }
  position: relative; left: $x; top: $y;
}
```

© 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 habilidades 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.
- **Comprensible:** 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 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, exceda 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

© 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

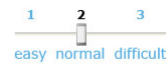
- Por medio del teclado y a través del atributo `tabindex` podemos navegar entre los diferentes elementos de una web, pero sólo algunos elementos soportan este atributo y son capaces de recibir el foco: A, AREA, BUTTON, INPUT, OBJECT, SELECT, y TEXTAREA. Por tanto, un simple elemento DIV no puede ser 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

UTILIDADES

© JMA 2015. All rights reserved

Instalación de utilidades

Consideraciones previas

- Las utilidades son de línea de comandos.
- Para ejecutar los comandos es necesario abrir la consola comandos (Símbolo del sistema)
- Siempre que se realice una instalación o creación es conveniente “Ejecutar como Administrador” para evitar otros problemas.
- En algunos casos el firewall de Windows, la configuración del proxy y las aplicaciones antivirus pueden dar problemas.

GIT: Software de control de versiones

- Descargar e instalar: <https://git-scm.com/>
- Verificar desde consola de comandos:
 - git

Node.js: Entorno en tiempo de ejecución

- Descargar e instalar: <https://nodejs.org>
- Verificar desde consola de comandos:
 - node --version

© JMA 2015. All rights reserved

npm: Node Package Manager

- Aunque se instala con el Node es conveniente actualizarlo:
 - `npm update -g npm`
- Verificar desde consola de comandos:
 - `npm --version`
- Configuración:
 - `npm config edit`
 - `proxy=http://usr:pwd@proxy.dominion.com:8080` ← Símbolos: %HEX ASCII
- Generar fichero de dependencias `package.json`:
 - `npm init`
- Instalación de paquetes:
 - `npm install -g grunt-cli karma karma-cli` ← Global (CLI)
 - `npm install jasmine-core tslint --save --save-dev`
 - `npm install` ← Dependencias en `package.json`
- Arranque del servidor:
 - `npm start`

© JMA 2015. All rights reserved

Generación del esqueleto de aplicación

- Configurar un nuevo proyecto puede ser un proceso complicado y tedioso, con tareas como:
 - Crear la estructura básica de archivos y bootstrap
 - Configurar Browserify o WebPack para transpilar el código
 - Crear scripts para ejecutar el servidor de desarrollo, tester, publicación, ...
- Disponemos de diferentes opciones de asistencia:
 - Proyectos semilla (seed) disponibles en github
 - Generadores basados en Yeoman
 - Herramientas oficial de gestión de proyectos (CLI)
- Una *Command Line Interface* permite generar proyectos desde consola, así como ejecutar un servidor de desarrollo o lanzar los tests de la aplicación.

© JMA 2015. All rights reserved

Express: Infraestructura de aplicaciones web Node.js

- <http://expressjs.com/>
- Instalación:
 - `npm install -g express-generator`
- Generar un servidor y sitio web:
 - `express cursoserver`
- Descargar dependencias
 - `cd cursoserver && npm install`
- Levantar servidor:
 - `SET DEBUG=cursoserver:* & npm start`
- Directorio de cliente de la aplicación web
 - `cd cursoserver\public` ← Copiar app. web

© JMA 2015. All rights reserved

Bower: Gestor de dependencias del frontend web

- <http://bower.io/>
- Instalación:
 - `npm install -g bower`
- Generar fichero de dependencias bower.json:
 - `bower init`
- Descargar, instalar y registrar dependencia de una librería o framework:
 - `bower install jquery -save`

© JMA 2015. All rights reserved

Grunt: Automatización de tareas

- Instalación general (<https://gruntjs.com/>):
 - npm install -g grunt-cli
 - npm install -g grunt-init
- Instalación local de los módulos en la aplicación (añadir al fichero package.json de directorio de la aplicación o crearlo si no existe):


```
{
  "name": "my-project-name",
  "version": "0.1.0",
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0",
    "grunt-shell": "~0.7.0"
  }
}
```

© JMA 2015. All rights reserved

Grunt: Automatización de tareas

- Descargar e instalar localmente los módulos en la aplicación (situarse en el directorio de la aplicación)
 - npm install --save-dev
- Fichero de tareas: gruntfile.js
- Ejecutar tareas:
 - grunt
 - grunt jshint
 - grunt reléase
 - grunt serve

© JMA 2015. All rights reserved

Gulp: Automatización de tareas

- Instalación general o local (<https://gulpjs.com/>):
 - npm install gulp-cli -g
 - npm install --save-dev gulp
- Instalación de las dependencias que sean necesarias:
 - npm install --save-dev gulp-concat gulp-uglify
- Crear el archivo gulpfile.js con las diferentes tareas:


```
var gulp = require('gulp'), concat = require('gulp-concat'), uglify = require('gulp-uglify');
gulp.task('tarea', function () {
  gulp.src('js/*.js')
    .pipe(concat('unico.js'))
    .pipe(uglify())
    .pipe(gulp.dest('build/'))
});
```
- Ejecutar una tarea:
 - gulp tarea

© JMA 2015. All rights reserved

Webpack

- Webpack (<https://webpack.github.io/>) es un empaquetador de módulos, es decir, permite generar un archivo único con todos aquellos módulos que necesita la aplicación para funcionar.
- Toma módulos con dependencias y genera archivos estáticos correspondientes a dichos módulos.
- Webpack va mas allá y se ha convertido en una herramienta muy versátil. Entre otras cosas, destaca que:
 - Puede generar solo aquellos fragmentos de JS que realmente necesita cada página.
 - Dividir el árbol de dependencias en trozos cargados bajo demanda
 - Haciendo más rápida la carga inicial
 - Tiene varios loaders para importar y empaquetar también otros recursos (CSS, templates, ...) así como otros lenguajes (ES6 con Babel, TypeScript, SaSS, etc).
 - Sus plugins permiten hacer otras tareas importantes como por ejemplo minimizar y ofuscar el código.

© JMA 2015. All rights reserved

JSLint, JSHint y TSLint

- Los analizadores de código son herramientas que realizan la lectura del código fuente y devuelve observaciones o puntos en los que tu código puede mejorarse desde la percepción de buenas prácticas de programación y código limpio.
- JSHint es un analizador online de código JavaScript (basado en el JSLint creado por Douglas Crockford) que nos permitirá mostrar puntos en los que tu código no cumpla unas determinadas reglas establecidas de “código limpio”.
- El funcionamiento de JSHint es el siguiente: toma nuestro código, lo escanea y, si encuentra un problema, devuelve un mensaje describiéndolo y mostrando su ubicación aproximada.
- Para descargar e instalar:
 - `npm install -g jshint`
- Existen “plug-in” para la mayoría de los entornos de desarrollo (<http://jshint.com>). Se puede automatizar con GRUNT o GULP.

© JMA 2015. All rights reserved

ESLint

- ESLint (<https://eslint.org/>) es una herramienta para identificar e informar sobre patrones encontrados en código ECMAScript/JavaScript, con el objetivo de hacer que el código sea más consistente y evitar errores.
- Se puede instalar ESLint usando npm:
 - `npm install eslint --save-dev`
- Luego de debe crear un archivo de configuración `.eslintrc.json` en el directorio, se puede crear con `--init`:
 - `npx eslint --init`
- Se puede ejecutar ESLint con cualquier archivo o directorio:
 - `npx eslint **/*.js`

© JMA 2015. All rights reserved

Karma: Gestor de Pruebas unitarias de JavaScript

- Instalación general:
 - npm install -g karma
 - npm install -g karma-cli
- Generar fichero de configuración karma.conf.js:
 - karma init
- Ingenierías de pruebas unitarias disponibles:
 - <http://jasmine.github.io/>
 - <http://qunitjs.com/>
 - <http://mochajs.org>
 - <https://github.com/caolan/nodeunit>
 - <https://github.com/nealxyc/nunit.js>

© JMA 2015. All rights reserved

Yeoman: Generador del esqueleto web

- <http://Yeoman.io>
- Instalación:
 - npm install -g yo
 - npm install -g generator-angular
 - npm install -g generator-karma
- Generar un servidor y sitio web:
 - yo angular
- Descargar dependencias si es necesario
 - bower install & npm install
- Preparar entorno de ejecución y levantar el servidor en modo prueba:
 - grunt serve

© JMA 2015. All rights reserved

generator-webapp

- Instalar:
 - `npm install --global yo gulp-cli bower generator-webapp`
- Para generar una nueva aplicación
 - `md miapp && cd miapp`
 - `yo webapp`
- Para instalar dependencias frontend
 - `bower install --save <package>`
- Preparar entorno de ejecución y levantar el servidor en modo prueba:
 - `gulp serve`
- Para ejecutar las pruebas en el navegador
 - `gulp serve:test`
- Para generar la aplicación web para producción (/dist):
 - `gulp`

© JMA 2015. All rights reserved

DOCUMENTADOR

© JMA 2015. All rights reserved

Documentación

- Hacer la documentación del código fuente puede llegar a ser muy tedioso, todos los programadores prefieren ir directo al grano, escribir su código y pasar de largo esta aburrida tarea. Por fortuna, actualmente existen un montón de herramientas para agilizar la documentación del código sin tener que dedicarle más tiempo del imprescindible.
- JSDoc es una sintaxis para agregar comentarios con documentación al código fuente de JavaScript.
- La sintaxis JSDoc es similar a la sintaxis de Javadoc, usado para documentar el código de Java, pero se ha especializado para trabajar con la sintaxis de JavaScript, es más dinámico y, por tanto único, ya que no es totalmente compatible con Javadoc. Sin embargo, como Javadoc, JSDoc permite al programador crear Doclets y Taglets que luego se pueden traducir en formatos como HTML o RTF.

```
/**
 * Create a dot.
 * @param {number} x - The x value.
 * @param {number} y - The y value.
 * @param {number} width - The width of the dot, in pixels.
 */
constructor(x, y, width) {
```

© JMA 2015. All rights reserved

JSDoc

Etiqueta	Descripción
@author	nombre del autor.
@constructor	indica el constructor.
@deprecated	indica que ese método es deprecated.
@exception	sinónimo de @throws.
@param	parámetros de documentos y métodos.
@private	indica que el método es privado.
@return	indica que devuelve el método.
@see	Indica la asociación con otro objeto.
@throws	Indica la excepción que puede lanzar un método.
@version	indica el número de versión o librería.

© JMA 2015. All rights reserved

Documentador

- JDOC
 - <http://usejsdoc.org/index.html>
 - <https://github.com/jsdoc3/jsdoc>
 - http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=881:guia-de-estilo-javascript-comentarios-proyectos-jsdoc-param-return-extends-ejemplos-cu01192e&catid=78&Itemid=206
- Doxygen
 - <http://www.stack.nl/~dimitri/doxygen/>
- Compodoc
 - <https://compodoc.github.io/website/>
- Typedoc
 - <https://typedoc.org>

© JMA 2015. All rights reserved



© 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 la 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 retícula 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

Cuadrícula

```
<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
<code>.visible-xs</code>	Visible	Oculto	Oculto	Oculto
<code>.visible-sm</code>	Oculto	Visible	Oculto	Oculto
<code>.visible-md</code>	Oculto	Oculto	Visible	Oculto
<code>.visible-lg</code>	Oculto	Oculto	Oculto	Visible
<code>.hidden-xs</code>	Oculto	Visible	Visible	Visible
<code>.hidden-sm</code>	Visible	Oculto	Visible	Visible
<code>.hidden-md</code>	Visible	Visible	Oculto	Visible
<code>.hidden-lg</code>	Visible	Visible	Visible	Oculto
Clase	Navegador		Impresora	
<code>.visible-print</code>	Oculto		Visible	
<code>.hidden-print</code>	Visible		Oculto	

© JMA 2015. All rights reserved

ESTILOS CSS

© JMA 2015. All rights reserved

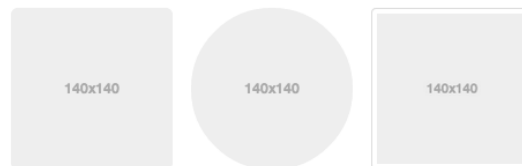
Tipografía

- 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 espaciado de 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 `×` como si fuera la típica X asociada con el cierre de una ventana o aplicación.
`<button type="button" class="close" aria-hidden="true">×</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 conmutables (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 - anexas (affix.js)
 - Gestiona el posicionamientos de los elementos afijos.

© JMA 2015. All rights reserved