
12 horas con HTML5, CSS3 y Javascript

CURSO ACELERADO

JS

HTML

CSS



HTML 5



“HTML5 es el presente de la web y si no estás asimilando lo que está pasando ya eres parte de la vieja generación de desarrolladores. Eso tendría que tenerte preocupado.”

HTML5 implica saber que hoy en día no nos conectamos solo desde el ordenador sino desde teléfonos móviles, tabletas, eBooks, netbooks, y otra gama de dispositivos. El html5 supuso que se acabaron los webmasters independientes y hoy hablamos de equipos multidisciplinarios de

empresas de tecnología que cuentan con frontends, backends, sysadmins, mobile devs, community managers y arquitectos de información en los proyectos que están reiventando mercados

Podemos hablar de todas las empresas gigantes de la web: Microsoft, Google, Apple, Adobe, Facebook, Yahoo, Mozilla y miles de proyectos tecnológicos que hoy se basan en HTML5, lo apoyan y tienen propuestas que los hacen competir en código en el navegador más cercano y desde cualquier dispositivo.



Como dato curioso el HTML5 tiene logo de superhéroe y vende incluso camisetas para apoyar al movimiento?

HTML5 es la actualización de HTML, el lenguaje en el que es creada la web. HTML5 también es un termino de marketing para agrupar las nuevas tecnologías de desarrollo de aplicaciones web: HTML5, CSS3 y nuevas capacidades de Javascript.

La versión anterior y más usada de HTML, HTML4, carece de características necesarias para la creación de aplicaciones modernas basadas en un navegador. El uso fuerte de Javascript ha ayudado a mejorar esto, gracias a frameworks como jQuery, entre otros.

LA LENTA EXTINCIÓN DE FLASH

Flash en especial ha sido usado en reemplazo de HTML para desarrollar web apps que superaran las habilidades de un navegador: Audio, video, webcams, micrófonos, datos binarios, animaciones vectoriales, componentes de interfaz complejos, entre muchas otras cosas.

Ahora HTML5 es capaz de hacer esto sin necesidad de plugins y con una gran compatibilidad entre navegadores.

No obstante flash no ha muerto y sigue usándose en muchos sitios webs y aplicaciones multimedia

ETIQUETAS DE HTML5

HTML4 y HTML5 son 100% compatibles entre sí. Todo el código que tienes en HTML normal seguirá funcionando sin problemas en HTML5. Para empezar a usar HTML5 lo único que tienes que hacer es colocar este DOCTYPE4 antes de la etiqueta <html>:

<!DOCTYPE HTML>

Es un DOCTYPE mucho más simplificado que XHTML5 (cuyas reglas siguen siendo usadas) y te permite usar todas las habilidades de HTML5 sin que nada de lo que ya tienes programado deje de funcionar.

Es muy importante escribir correctamente tanto el código CSS como el XHTML. Y sobre todo, este último, pues no podemos olvidar que XHTML, es la base de la WEB, y el armazón sobre el que se aplicará el CSS.

Para que el código XHTML sea adecuado y rígido, existe el DOCTYPE. Sin extenderme mucho, es una declaración para hacer que el navegador, entienda qué es

lo que va a recibir, y cómo deberá procesarlo. De esta manera, la misma página, con distintos doctype, o peor aún, sin él, se verá de distinta manera, incluso en el mismo navegador.

Por ello, es necesario elegir adecuadamente un doctype, y seguirlo.

¿Las posibilidades? Unas cuantas, pero mi primer consejo es que se usen Doctype para XHTML. El segundo: que se elija "XHTML STRICT".

De esta manera, le advertiremos al navegador de que va a recibir código XHTML estándar (W3C).

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

["http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">](http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd)

VALIDAR LA WEB:

Se trata de verificar que el código (x)HTML de tu WEB es coherente con su

doctype. ¿Cómo hacerlo? Pues mediante el validador de la W3C.

VALIDAR EL CSS:

Se trata de verificar que el CSS que hemos escrito, es igualmente adecuado y sin errores.

Esto debemos hacerlo para asegurarnos la compatibilidad entre navegadores.

La W3C también nos ofrece su Validador de CSS,

<HEADER>

Hacer cosas como `<div id="header">` es un poco extraño cuando el 99% de los proyectos web tienen una cabecera. `<header>` está diseñada para reemplazar la necesidad de crear divs sin significado semántico.

<HGROUP>

Muchos headers necesitan múltiples títulos, como un blog que tiene un título y un tagline explicando el blog. `<hgroup>` permite colocar un `h1`, `h2` y `h3` dentro del header sin afectar el SEO, permitiendo usar otro `h1` en el sitio. En el HTML actual, sólo puedes usar `h1` una vez por sitio o el `h1` pierde prioridad de SEO.

<NAV>

Igual que `<header>`, `<nav>` está diseñado para que ahí coloques la botonera de navegación principal. Puedes colocar cualquier etiqueta dentro, aunque lo recomendado es usar listas ``.

<SECTION>

Define un área de contenido única dentro del sitio. En un blog, sería la zona donde están todos los posts. En un video de youtube, habría un section para el video, uno para los datos del video, otro para la zona de comentarios.

<ARTICLE>

Define zonas únicas de contenido independiente. En el home de un blog, cada post sería un article. En un post del blog, el post y cada uno de sus comentarios sería un `<article>`.

<ASIDE>

Cualquier contenido que no esté relacionado con el objetivo primario de la página va en un aside. En un blog, obviamente el aside es la barra lateral de información. En el home de un periódico, puede

ser el área de indicadores económicos, o muy normalmente la publicidad.

<FOOTER>

Este es obvio. Es el pie de página y todo lo que lo compone.

Atención `<div>` no está muerto: Estas nuevas etiquetas no significan que ya no se use `<div>`. Div siempre debe usarse cuando necesites una caja con objetivos de diseño gráfico o cualquier cosa que no tenga significado semántico. Sólo usa las etiquetas semánticas de HTML5 donde sean necesarias.

EJEMPLO DE NUEVAS ETIQUETAS HTML5

```
<header>
<hgroup>
<h1>El blog de html5</h1>
<h2>Este es el blog de html5</h2>
</hgroup>
</header>
<nav>
Aquí va la botonera de navegación
</nav>
<section>
<article>Aquí va un post, con su titulo en
h2</article>
<article>Aquí va un post, con su titulo en
h2</article>
<article>Aquí va un post, con su titulo en
h2</article>
</section>
<aside>
Barra lateral con cosas como cuentas de
twitter, facebook, posts viejos, etc.
</aside>
<footer>
Pie de pagina, copyright, etc.
</footer>
```


NUEVAS ETIQUETAS DE HTML 5

Video, audio y animación vectorial están en la lista de prioridades de todas las personas que enseñan el html5. Específicamente, las nuevas etiquetas son:

<VIDEO>

Inserta video sin necesidad de plugins. Es muy fácil usarla, pero cada navegador soporta codecs diferentes de video, lo que hace necesario recodificar un video en múltiples codecs. En un futuro capítulo hablaremos un poco del problema que este tag está generando y cómo utilizarla.

<AUDIO>

Lo mismo que video, pero sin video. Puede usar múltiples formatos, en especial mp3, pero también depende del navegador.

<INPUT >

Input ya existía como la etiqueta para insertar cajas de texto y botones. Ahora es más poderosa, con la capacidad de insertar cajas tipo “email” que se autovalidan, calendarios tipo “date”, sliders, números, entre otras.

<CANVAS>

Un área de dibujo vectorial y de bitmaps con Javascript. Es un API de dibujo entero para Javascript.

<SVG>

Una etiqueta, igual que , para insertar dibujos y animaciones vectoriales al estilo de Flash. Todo basado en el estándar abierto SVG (Scalable Vector Graphics), derivado de XML.

TABLA PERIÓDICA DE LAS ETIQUETAS HTML

Es recomendable visitar este sitio web para obtener una visión de las etiquetas html 4 y 5.

Hay etiquetas que han caído en desuso y otras características también interesantes que no he mencionado, podéis consultar todas las etiquetas en forma de tabla periódica en el siguiente enlace

<http://joshduck.com/periodic-table.htm>

FORMULARIOS WEB

Para el usuario, los formularios tienen ahora en HTML5 un comportamiento más inteligente, un campo de fecha mostrará al usuario un calendario interactivo, o hints (pistas) sobre qué ingresar en ese campo, foco automático en el campo inicial, formato correcto a medidas de tiempo o seleccionar color mediante una paleta de colores.

Html5 agrega MUCHAS novedades, nuevos atributos, elementos y tipos de campos.

Todos los ejemplos que salen a continuación funcionan en el navegador Opera, pues es el que actualmente soporta mejor la mayoría de las nuevas características, también muchas funcionan bien en Chrome, Firefox y hasta en Explorer, pero varias sólo están disponibles para Opera actualmente. Pero esto no es razón para no usarlos, sino al contrario, porque no serán visibles para algunos pero les será muy útil a quienes utilicen navegadores modernos.

13 NUEVOS ATRIBUTOS:

AUTOFOCUS

Hacemos foco en el input que tenga asociado éste valor al cargar la página

PLACEHOLDER

Para ofrecer una pista de lo que el usuario debe ingresar

ACCEPT

Permite que solo el tipo de archivo determinado pueda ser cargado en el formulario

MULTIPLE

Permite seleccionar múltiples archivos para ser cargados de una vez por el formulario

MAX / MIN / STEP

Podemos delimitar rango de valores numéricos permitidos máximos, mínimos y múltiplos dentro de un rango.

FORM NO DISPONIBLE

Este atributo nos permite colocar un elemento en cualquier parte de la página, no solamente dentro del form tag, y que éste siga siendo procesado por el form. Además permite asociar un mismo

elemento a mas de un form simultaneamente

REQUIRED

Atributo booleano que determina si el elemento debe ser obligatorio o no

(Utilizado junto con “pattern”, “max / min”, “email” y otros nuevos atributos, nos permite prescindir en gran medida de Javascript para validar nuestros formularios y hacero sólo con el navegador.)

AUTOCOMPLETE

Permite especificar si un elemento puede o no ser autocompletado por el navegador basado en entradas previas del usuario

PATTERN

Para validar un elemento en base a una expresión regular (RegExp)

DIRNAME NO DISPONIBLE

Envía la dirección en que fue ingresado el texto en un input determinado (ej: de derecha a izquierda)

NOVALIDATE

Evita la validación del elemento al enviar el formulario

FORMACTION

Para sobrescribir el comportamiento por defecto del formulario

FORMENCTYPE

Para sobrescribir la codificación por defecto del formulario

FORMMETHOD

Para sobrescribir el método de envío por defecto del formulario

FORMNOVALIDATE

Para sobrescribir novalidate de un formulario (Esto es útil por ejemplo si tenemos un formulario con dos botones u acciones, uno de envío y otro para guardar cambios, pero queremos que la validación ocurra al enviar un formulario y no si el formulario es guardado)

FORMTARGET

Para sobrescribir la ventana destino por defecto del form

5 NUEVOS ELEMENTOS:

PROGRESS

Representa el grado de progreso de una tarea o acción (ej: Representar la carga de una imagen)

METER

Se utiliza para medir algo con una escala determinada, como temperatura o distancias.

DATALIST

Si asociamos un text input a un Datalist (lista de valores) al hacer foco en ese

input aparece un dropdown mostrando el contenido del elemento Datalist

KEYGEN

Genera un par de claves de control privada que se guarda en local y pública que se envía al servidor

OUTPUT

Se utiliza para realizar cálculos entre campos (ej: La suma de 2 números en distintos inputs)

13 NUEVOS TIPOS DE CAMPOS:

TEL

Para un número telefónico (En realidad, no prueba que sea un número, para validar un formato numérico en particular se debe complementar con “pattern”)

SEARCH

Sugiere ingreso de texto en el input (La diferencia entre search y un input de texto común es que puede ser reconocido por la plataforma en la cual es visto para igualar su estilo)

URL

Para ingresar direcciones web absolutas

EMAIL

Para valores únicos o múltiples de direcciones de email

DATETIME

Formato de fecha y hora con zona horaria UTC

DATE

Formato de fecha sin zona horaria

MONTH

Mes y año sin zona horaria

WEEK

Fecha para ingresar semana del año

TIME

Hora completa sin zona horaria

datetime-local

Fecha y hora sin zona horaria

NUMBER

Valores numericos

RANGE

Valor numerico para controlar slider

COLOR

Para elegir colores de una paleta

Recuerda que elementos y atributos como placeholder, autofocus, email, datalist y required, entre los más comunes, ya tienen un amplio soporte, y como alternativa, siempre tenemos la posibilidad de ofrecer una experiencia similar utilizando soporte de librerías de javascript (como modernizr) para navegadores que no soportan aún estos avances

de forma tal de ir incorporando poco a poco esta nueva experiencia al diseño cotidiano.

Fuentes:

W3c Specification

* <http://wufoo.com/html5/>

RESUMEN DE LAS ETIQUETAS EN LOS FORMULARIOS EN HTML 5

NUEVOS TIPOS DE ENTRADA:

```
<input type="email" />
<input type="url" />
<input type="date" />
<input type="time" />
<input type="datetime" />
<input type="month" />
<input type="week" />
<input type="number" />
<input type="range" />
<input type="tel" />
<input type="search" />
<input type="color" />
```

NUEVOS CONTROLES:

```
<output></output>
```

NUEVOS ATRIBUTOS:

```
autofocus
min
max
pattern
placeholder
required
step
```

EJEMPLO DE FORMULARIO HTML5

SIMPLE

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML5 test</title>
  <meta content="text/html;
charset=ISO-8859-1" http-equiv="content-
type" />
</head>
<body>
  <header>
    <nav>
      <ul>
        <li><a href="#">Inicio</a></li>
        <li><a href="#">Pedidos</a></li>
        <li><a href="#">Facturas</a></li>
        <li><a href="#">Administración</a></li>
      </ul>
    </nav>
    <form>
      <input type="search" name="q"
placeholder="Búsqueda" />
    </form>
  </header>
  <section>
    <article>
      <header>
        <h2>Noticia #1</h2>
      </header>
      <section>
        <p>Texto de la noticia #1...</p>
      </section>
    </article>
    <article>
      <header>
        <h2>Noticia #2</h2>
      </header>
```

```
    <section>
      <p>Texto de la noticia #2...</p>
    </section>
  </article>
</section>
<aside>
  <form>
    <p><label>Nombre: <input name="name"
required></label></p>
    <p><label>E-mail: <input name="email"
type="email" required></label></p>
    <p><label>Edad: <input type="number"
min="0" max="99" step="1" value="33"></
label></p>
    <p><label>Edad: <input type="range"
min="0" max="10" step="2" value="6"></
label></p>
    <p><label>URL: <input name="url"
type="url"></label></p>
    <p><label>Comentario: <textarea
name="comment" required></textarea></
label></p>
    <p><input type="submit"
value="Enviar"></p>
  </form>
</aside>
<footer>
  <p>Copyleft 2010</p>
</footer>
</body>
</html>
```


EJEMPLO DE FORMULARIO HTML5 COMPLEJO

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
















<title>Ejemplo nuevos controles</title>
</head>

<body>
<form action="."
oninput="range_control_value.value =
range_control.valueAsNumber">
<p>
Nombre: <input type="text"
name="name_control" autofocus required />
<br />
Correo Electrónico: <input type="email"
name="email_control" required />
<br />
URL: <input type="url" name="url_control"
placeholder="Escribe la URL de tu página
web personal" />
<br />
Fecha: <input type="date"
name="date_control" />
<br />
Tiempo: <input type="time"
name="time_control" />
<br />
Fecha y hora de nacimiento: <input
type="datetime" name="datetime_control" />
<br />
Mes: <input type="month"
name="month_control" />
```

```
<br />
Semana: <input type="week"
name="week_control" />
<br />
Número (min -10, max 10): <input
type="number" name="number_control"
min="-10" max="10" value="0" />
<br />
Intervalo (min 0, max 10): <input
type="range" name="range_control" min="0"
max="10" value="0" /> <output
for="range_control"
name="range_control_value" >0</output>
<br />
Teléfono: <input type="tel"
name="tel_control" />
<br />
Término de búsqueda: <input type="search"
name="search_control" />
<br />
Color Favorito: <input type="color"
name="color_control" />
<br />
<input type="submit" value="Submit!" />
</p>
</form>
</body>
</html>
```

EL PROBLEMA DE COMPATIBILIDAD CON LOS NAVEGADORES

VALIDACIÓN EN CLIENTE DE FORMULARIOS

























Browser Support for New HTML5 Input Types								
								
		Firefox	Safari	Safari	Chrome	Opera	IE	Android
Email		4+	5+	3.1+	6+ / 10+	10.6+	10+	2.3-
Tel		4+	5+	3.1+	6+	10.6+	10+	2.3+
Url		4+	5+	3.1+	6+ / 10+	10.6+	10+	2.3-
Search		3.6 / 4+	5+	4+	6+	10.6+	9 / 10+	2.3-
Color		11-	5.2-	5-	20+	11+	10-	2.3-
Number		11-	4 / 5.2+	4+	9- / 10+	11+	10-	2.3+
Range		11-	4+	5+	6+	9+	10+	2.3-
Date		11-	5+	5-	10 / 17 / 20+	10.6+	10-	2.3-












Hasta ahora la forma que se utiliza para validar los formularios “del lado del cliente” es javascript para ayudar al usuario a completar correctamente los formularios para recopilar coherentemente su información, desde el registro y acceso de una cuenta hasta procesos de pago de compra y otros.

Las nuevas características para manejo de formularios html5 permiten prescindir de javascript para realizar validaciones “del lado del cliente”, esto me permite como diseñador crear un formulario completo sin recurrir a un desarrollador, aumentando la productividad, sobre todo en equipos pequeños de trabajo.

En el ejemplo anterior el formulario ya está validado.

Ahora presentaremos un ejercicio simple de formulario validado con una hoja de estilos

Browser Support for New HTML5 Input Attributes								
								
		Firefox	Safari	Safari	Chrome	Opera	IE	Android
Placeholder		4+	4/5+	4+	10+	11.10/11.50	10+	2.3+
Autofocus		4+	5+	5-	6+	11+	10+	2.3-
Maxlength		3.6/4+	5+	4+	6+	11+	9/10	2.3+
List (Datalist)		4+	5-	5-	20+	10.6+	10+	2.3-
Autocomplete		4+	5.2+	N/A	17+	10.6+	10-	2.3-
Required		6+	5-	5-	6+	10.6+	10+	2.3-
Pattern		4+	5-	5-	6/10+	10.6/11+	10+	2.3-
Spellcheck		3.6+	4+	5-	10+	11+	10+	N/A
Novalidate		4+	5-	5-	6+	10.6+	10+	2.3-
Formnovalidate		4+	5.2-	5-	6+	10.6+	10+	2.3-
Formaction		4+	5.2+	5+	10+	10.6+	10+	2.3-
Formmethod		4+	5.2+	5+	10+	10.6+	10+	2.3-
Formtarget		4+	5.2+	5+	10+	10.6+	10+	2.3-
Formenctype		?	?	?	?	?	?	?
Accept		11+	5.2+	N/A	10+	10.6+	10+	2.3-
Multiple		3.6+	5+	N/A	6+	11+	10+	2.3-
Min / Max / Step		11-	5+	5-	6+	10.6+	10+	2.3-

Browser Support for New HTML5 Form Elements								
								
		Firefox	Safari	Safari	Chrome	Opera	IE	Android
Meter		16+	5.2+	5-	6+	11+	10-	2.3-
Progress		6+	5.2+	5-	6+	10.6+	10+	2.3-
Output		6+	5.1+	5+	13+	9.2+	10+	2.3+
Keygen		3.6+	4+	5-	6+	10.6+	10-	2.3+

```

html5.html x  estilo.css x
html body form fieldset input#nombre
</head>
<body>
<form action="#" >
  <fieldset>
    <h2>Actualizar datos</h2>
    <input type="text" id="nombre" name="nombre" required="required" placeholder="Usuario"
      pattern="[a-zA-Z]{4,}" maxlength="60">
    <div class="mensaje">Este campo debe tener + de 4 letras de la "aA" a la "zZ"</div>
    <input type="email" id="email" name="email" placeholder="correo@dominio.com" required>
    <div class="mensaje">Su email no debe tener caracteres diferentes a los permitidos</div>
    <input type="submit" id="enviar" name="enviar" value="enviar">
  </fieldset>
</form>
</body>
</html>

```

```

html5.html x  estilo.css x
html {
  font-size: 62.5%;
}
body {
  font-family: arial, sans-serif;
  color: #313030;
  font-size: 1.2em;
}

```

```

fieldset {
  width: 400px;
}
input {
  display: block;
}
.mensaje {
  padding: 6px;
}
input:valid + .mensaje {
  visibility: hidden;
}
input:valid {
  border: 1px solid gray;
}
input:invalid {
  border: 1px solid red;
}

```


AUDIO Y VIDEO

Hablar de vídeo en internet es hablar de ciclos, de batallas entre los CEOs más influyentes en el mundo de tecnología, de la compleja estandarización de un codec o de un player. Hay un avance importante con el tag <video> pero esto apenas empieza y mucha de la historia seguramente se va a repetir nuevamente.

Regresemos en el tiempo al año 2004. No existía Youtube. Por lo mismo, si querías alojar y compartir en un navegador cualquier vídeo te enfrentabas a dramas de plataforma, de ancho de banda y de problemas de usabilidad para el usuario final.

Si encontrabas algún enlace que te llevara a ver videos en la web seguramente te sonarían los logos de Real Player, de Windows Media Player o incluso de Quicktime. Del lado del servidor los sysadmins tenían que pelear con Real Media Server, Windows Media Server que era parte del IIS y otro montón de opciones, pero era complicado unificar.

Adobe por su parte había logrado importantes avances para que desde archivos .swf pudieras incorporar videos, pero no había un solo canal.

En el año 2005, tres jóvenes dejaron de trabajar en Paypal y emprendieron el sueño de crear una plataforma universal para compartir video. Ese año nace Youtube.

Paralelamente, Adobe demostró que su esfuerzo para crear un sistema para reproducir vídeo iba a ser valorado y aprovecharía el dominio que su plugin de flash tenía en todos los navegadores del planeta. Los sysadmins educaron a sus usuarios a crearse una cuenta en youtube y que dejaran de molestar con complejos servidores para alojar y reproducir vídeos.

En abril del 2010 una carta de Steve Jobs a todos los clientes de apple sobre sus pensamientos de Flash arranca uno de esos nuevos ciclos donde una platafor-

ma disruptiva que cada día ganaba mercado le daba la espalda al famoso plugin de Adobe en sus dispositivos móviles. Apple le prometía larga vida al HTML5 porque todo se podía resolver en temas de vídeo con un tag aprobado por el estándar. Hoy volvemos a tener una batalla de grandes empresas por el codec luego de que están todos de acuerdo que el navegador lleva una opción amigable para incluir un elemento de video, porque este elemento es tan importante como las imágenes.

Y podemos usar perfectamente el ejemplo de las imágenes para entender lo que pasa con el video. Todos los navegadores permiten que en un documento HTML incluyas un tag `` al cual debes decirle la ruta a una imagen. Y la imagen puede ser un .jpg, un .gif, un .bmp o un .png (entre otros formatos). Dependerá de la capacidad de tu navegador para reconocer todos los contenidos de esta imagen, interpretarla y mostrarla. Con el video pasa lo mismo.

¿Con la etiqueta `<video>` dejamos de usar video en flash?

El debate sobre Flash y HTML5 da por mucho. Adobe tiene tecnología sólida y es de los reproductores que mejor entiende de codecs y licencias. Es una solución práctica para que no tengamos que enfrentarnos con esos problemas y sigue siendo la base de Youtube. Youtube sigue siendo la solución práctica para compartir en segundos un vídeo, le generemos difusión y hagamos además social media.

Pero incluso Youtube está haciendo esfuerzos importantes en miras de HTML5, resolviendo otros dramas que incluye este tag.

Revisa <http://youtube.com/html5> con un navegador moderno y conoce más de sus experimentos.

No dejamos de usar flash, es más, es uno de los planes más seguros para que muchas personas y navegadores viejos puedan ver tus contenidos en vídeo.

Uso del tag `<video>` y el soporte de formatos en diferentes navegadores

Si hablamos de navegadores, recuerda que hay una versión donde el video empezó a existir. Antes, simplemente no va a mostrar contenido y detectarlo para mostrar un player alternativo (en flash) es recomendado:

IE 9+, Firefox 3.5+, Chrome 3.0+, Safari 3.0+, Opera 10.5+, Android Browser 2.0+, Safari Mobile 1.0+

Para incluir un video con HTML5 usas el siguiente formato:

```
<video src="un-video.avi" width="320" height="240"></video>
```

Adicionalmente al ancho y alto, hay atributos adicionales que puede usar:

preload = que empezará a precargar el video independientemente de las acciones del usuario sobre el player.

```
<video src="un-video.avi" width="320" height="240" preload></video>
```

Recomendamos utilizarlo si la función de la página es mostrar un video. Si por el contrario, el video únicamente ayudará a complementar la información (un post que tiene múltiples videos o referencias, no lo recomendamos)

```
<video src="un-video.mp4" width="320" height="240" preload="none"></video>
```

Y el anterior que le dirá explícitamente que no tiene que precargar el video.

autoplay = dará play al video en cuanto cargue la página sin acción del usuario sobre los controles.

```
<video src="un-video-mlw.avi" width="320" height="240" autoplay></video>
```

controls = hace que vayan a incluirse los controles (play, pausa, volumen, etc.) en el player del video. Estos controles están predefinidos en cada navegador y como veremos más adelante en algunos pla-

yers opciones pueden ser modificados con javascript + css3.

```
<video src="un-video.avi" width="320" height="240" controls></video>
```

Aquí es super importante entender una diferencia entre los tipos de archivo y los codex con que el video ha sido procesado. Un .avi, .mp4 (o .m4v), un .flv (flash video) y un .ogv simplemente contienen un video, pero cada formato puede tener diferente forma de codificarlo.

Los navegadores y las compañías que los producen han elegido soportar únicamente a algunos codecs y lamentablemente no existe una alternativa definitiva y universal. En serio, no existe, tenemos tag <video> y el drama ha quedado por aquí, en que tienes que codificar tus videos al menos en algunos formatos para asegurarte que funcione en todos lados. ¿Ven por qué Youtube sigue siendo una opción maravillosa?

CODECS.

Hay 3 que importan hoy en día: H.264, VP8 y Theora.

H.264

Conocido como MPEG-4 Advanced Video Codiging, es el más popular hoy en día, lo usa youtube cuando muestra videos desde el flash player (el flash player lo soporta sin problema). Es el que le da vida a los videos en un iPhone, en tu iPad. Incluso muchos reproductores de Blue-ray lo soportan. Es genial, es asombroso, pero tiene algunos temas legales detrás que complica su existencia. Hay un grupo, el MPEG LA group que tiene patente sobre este formato y le quiere cobrar a todos los que decodifiquen su formato. Google no es muy amigo de este formato. Chrome ya tiene el 20% del mercado y Android sigue creciendo. En enero le dijeron adiós.

<http://blog.chromium.org/2011/01/html-video-codec-support-in-chrome.html>

VP8.

Google compró hace unos años a una empresa llamada On2 por una millonada porque tenían avances con desarrollos de codecs de videos. Y unos años des-

pués ponen a disposición del mundo el VP8. Es todo lo bueno que puedas pedir en este tipo de problemas legales ya que lo liberaron gratis, sin pagar derechos a quien lo quiera usar. Es un regalito de Google para el mundo. Pero Microsoft y Apple no quieren regalos de nadie, así que ya entenderás cuál es el problema de este formato.

THEORA.

No tiene dramas de patentes, es royalty free, funciona en Linux y es el que viene en archivos Ogg. De hecho, se basa en desarrollos de On2 (la empresa que eventualmente fue adquirida por Google) y llevó esto por otro camino que es muy libre, transparente, bueno, pero menos popular. Puedes instalar decodificadores en windows, en mac. Hoy lo soporta Chrome (junto a VP8) y suena maravilloso pero como todo lo gratis y open source, Google es amigo de este formato.

Resumen de estos formatos, tienes que usar al menos h.264 y Theora o VP8 para que la web te vea. No puedes usar solo uno, toca elegir dos.

Y por cierto, solo hemos hablado de codecs de video. El drama con codecs de au-

dio también nos trae opciones y complicaciones que te recomendaría investigar.

La opción de “controls” en el tag video abre un mundo de posibilidades para que te pongas a generar un reproductor visible al usuario personalizado. Y los hackers del mundo han

visto opción para poner su talento artístico a disposición de los demás.

ENLACES A PLAYERS

Hay muchas opciones de players de video que puedes utilizar e incluir en tus páginas HTML5. La mejor opción para conocer este confuso panorama es la tabla que encontramos en esta web:

<http://praegnanz.de/html5video>

Las opciones son muchas y hay demasiados grupos de desarrolladores asegurando generar buenas alternativas al estándar definido en cada navegador.

Entre las opciones más populares allí descritas y que hemos seguido:

Videojs

<http://videojs.com>

Han sabido vender la propuesta de su player con varios skins que imitan los de los principales sitios de video en la web.

Sublime Video

<http://sublimevideo.net>

Un proyecto con un fin comercial que quiere hacerse cargo del player para que te encargues de generar el contenido. Aunque sigo sin verle la proposición de

valor por sobre lo que me puede entregar Youtube.

Media Elements

<http://mediaelementjs.com>

Otra de las propuestas, con un player para <video> y <audio> que además tiene versiones compatibles con un flash player para navegadores viejos. Es de esos que te permiten aprender bastante mientras juegas con su código.

Importante con el tema del video es entender que HTML lo ha reconocido como una de las bases más importantes de la web del presente y su mundo de posibilidades es increíble.

EJEMPLOS DE MULTIMEDIA

```
html5.html x estilo.css x
html body section audio
<body>
  <h1>Multimedia en html5</h1>
  <section>
    <h2>Video ejemplo</h2>
    <video controls loop poster="poster.jpg" preload>
      <source src="MIB.mp4">
      <source src="MIB.webm">
      <source src="MIB.ogv">
    </video>
  </section>
  <section>
    <h2>Audio ejemplo</h2>
    <audio controls="controls" preload="metadata" >
      <source src="MIB.wav">
      <source src="MIB.mp3">
      <source src="MIB.ogg">
    </audio>
  </section>
</body>
</html>
```

```
html5-fallback-inicial.html x estilo.css x
html body section video p
</head>
<body>
  <h1>Multimedia en html5</h1>
  <section>
    <h2>Video ejemplo</h2>
    <video controls poster="poster.jpg" >
      <source src="MIB.mp4">
      <source src="MIB.webm">
      <source src="MIB.ogv">
      <p>Su navegador no soporta video html5</p>
    </video>
  </section>
  <section>
    <h2>Audio ejemplo</h2>
    <audio controls="controls">
      <source src="MIB.wav">
      <source src="MIB.mp3">
      <source src="MIB.ogg">
      <object width="353" height="132"><embed src="http://www.goeear.com/files/external.swf?file=07242fff" type="applica
      </embed>
    </object>
    </audio>
  </section>
```

CSS 3



Diseñar en CSS ha sido una mezcla entre risas y lloros. No sólo por la falta de compatibilidad con IE, sino porque cosas como bordes redondeados en tamaños dinámicos requiere múltiples divs, estilos y cuatro PNGs diferentes en el mejor de los casos. Ya no más, CSS3 trae opciones que hacen el diseño realmente fácil.

@FONT-FACE

Es la capacidad de usar CUALQUIER FUENTE EN HTML. Sin necesidad de Flash, Cufon, SiFR u otras cosas. ¿ Esto

funciona perfecto desde Internet Explorer 6 para arriba. NADIE LO USA.

Importante, antes que entender @font-face tienes que aprender a usar una maravilla que nos trajo Google y que le hizo la

vida más fácil a todos los amantes de la tipografía que estaban cansados de usar Arial y Helvetica. Visita Google WebFonts y aprovéchalo.

<http://www.google.com/webfonts/v2>

SELECTORES CSS

¿Te ha tocado hacer un diseño donde una lista o tabla tiene algunos elementos en blanco y los otros en gris? Como una cebra. Antes, la única forma era hacerlo a mano o con un script del lado del server. Ahora, con CSS3, sólo tienes que especificar un color para “odd” y otro para “even” y listo.

Igualmente, puedes crear estilos para el primer elemento y otro para el último. O estilos para etiquetas iguales con ciertos atributos diferentes en HTML. Y esto es muy compatible desde IE8.

COLUMNAS DE TEXTO

¿Sabes cómo se hacía antes que varios párrafos de texto se dividiera en columnas con HTML? No se podía. Ahora sólo requieres un atributo CSS para lograrlo. Y puedes controlar la cantidad de colum-

nas, el espacio entre ellas, líneas de separación, etc.

OPACIDAD, TRANSPARENCIA, CANALES ALPHA, CONTRASTE, SATURACIÓN Y BRILLO

Vuelve lo que quieras transparente u opaco con una instrucción. Imágenes, textos, sombras, bordes, lo que sea. O si quieres convertir una foto en blanco y negro o sepia, lo puedes hacer con sólo CSS.

ANIMACIONES DE TRANSICIÓN Y TRANSFORMACIÓN

Las animaciones que antes lograbas con jQuery o Javascript ahora pueden ser logrados sólo con CSS. Con una ventaja adicional, al hacerlo con CSS, las animaciones vendrán aceleradas por hardware. Mucho más veloces, sobre todo en dispositivos móviles.

<http://www.cristalab.com/tips/animaciones-css3-en-html-con-jquery-c916361>

SOMBRA, GRADIENTES Y TRANSFORMACIONES

BORDES REDONDEADOS

Sí. Con una instrucción puedes hacer que cualquier caja tenga bordes redondeados como quieras. Olvidate de crear múltiples divs, cortar pngs y otros temas arcaicos.

<http://www.cristalab.com/tips/bordes-redondeados-con-css3-c69441/>

<http://www.cristalab.com/tips/bordes-redondeados-en-css-3-con-border-radius-c913771/>

Con las siguientes nuevas reglas de estilo incluídas en CSS3 podemos crear bordes redondeados para nuestros tags en XHTML. Todo sin usar algún tipo de "truco" complejo o librería de Javascript dedicada exclusivamente a ello.

Este es el código que deberá estar incluído en tu archivo CSS para que funcione apropiadamente en Firefox 3 y posteriores:

Código :

#ejemplo

```
{  
-moz-border-radius-topleft: 10px;  
-moz-border-radius-topright: 10px;  
-moz-border-radius-bottomright: 10px;  
-moz-border-radius-bottomleft: 10px;  
}
```

Por otro lado, este es el encargado de lograr el mismo efecto en iPhone, Safari y Chrome:

Código :

#ejemplo

```
{  
-webkit-border-top-left-radius: 10px;  
-webkit-border-top-right-radius: 10px;  
-webkit-border-bottom-left-radius: 10px;  
-webkit-border-bottom-right-radius: 10px;  
}
```

Sin duda es una de las características mas adorables al momento de diseñar, y que por si fuera poco es un estándar bien aceptado por la W3C.

Border-Radius

Entrar a <http://border-radius.com> y nos encontramos con esto:

Webkit: Es un motor renderizado. Funciona como base para Safari y Google Chrome. En el CSS se representa:

Código :

- webkit-border-radius
- webkit-border-top-left-radius
- webkit-border-top-right-radius
- webkit-border-bottom-right-radius
- webkit-border-bottom-left-radius

Gecko: Es el motor de render usado por Firefox y en CSS se representa con :

- moz-border-radius
- moz-border-radius-topleft
- moz-border-radius-topright
- moz-border-radius-bottomright
- moz-border-radius-bottomleft

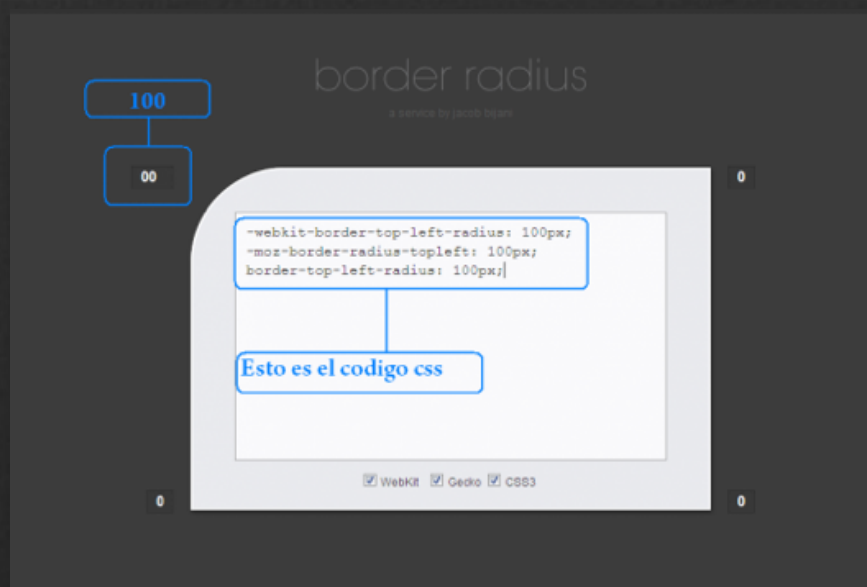
CSS3 o W3C: Esta es la forma correcta, que aun no la implementan muchos. Cada navegador (menos Opera y Explorer) tienen su propia implementación, pero esta es la forma estándar:

Código :

- border-radius
- border-top-left-radius
- border-top-right-radius
- border-bottom-right-radius

border-bottom-left-radius

En esas esquinas marcadas en la aparte anterior es donde debes ingresar el numero en pixeles de el grado de redondez deseada lo cual nos dará:



REFLEXIONES, GRADIENTES Y SOMBRAS

Si no has superado la web 2.0, puedes poner reflexiones a cualquier elemento HTML. Pero lo interesante es crear gradientes para fondos y sombras para cajas o texto, todo en una sola línea de código y con el mismo nivel de complejidad que logras con una sombra en Photoshop o Fireworks.

JQUERY CSSANIMATE

Download

<https://github.com/bipsa/css3Animate>

La implementación de este plugin es sencilla, solo se hace el selector y se llama el método CSSAnimate, como pueden notar el plugin ya valida si soporta CSS transitions, y de no soportarlo simplemente asigna los estilos a su selección.

Imaginemos que creas una clase llamada ".mySelector" que permitirá rotar un elemento HTML.

Código :

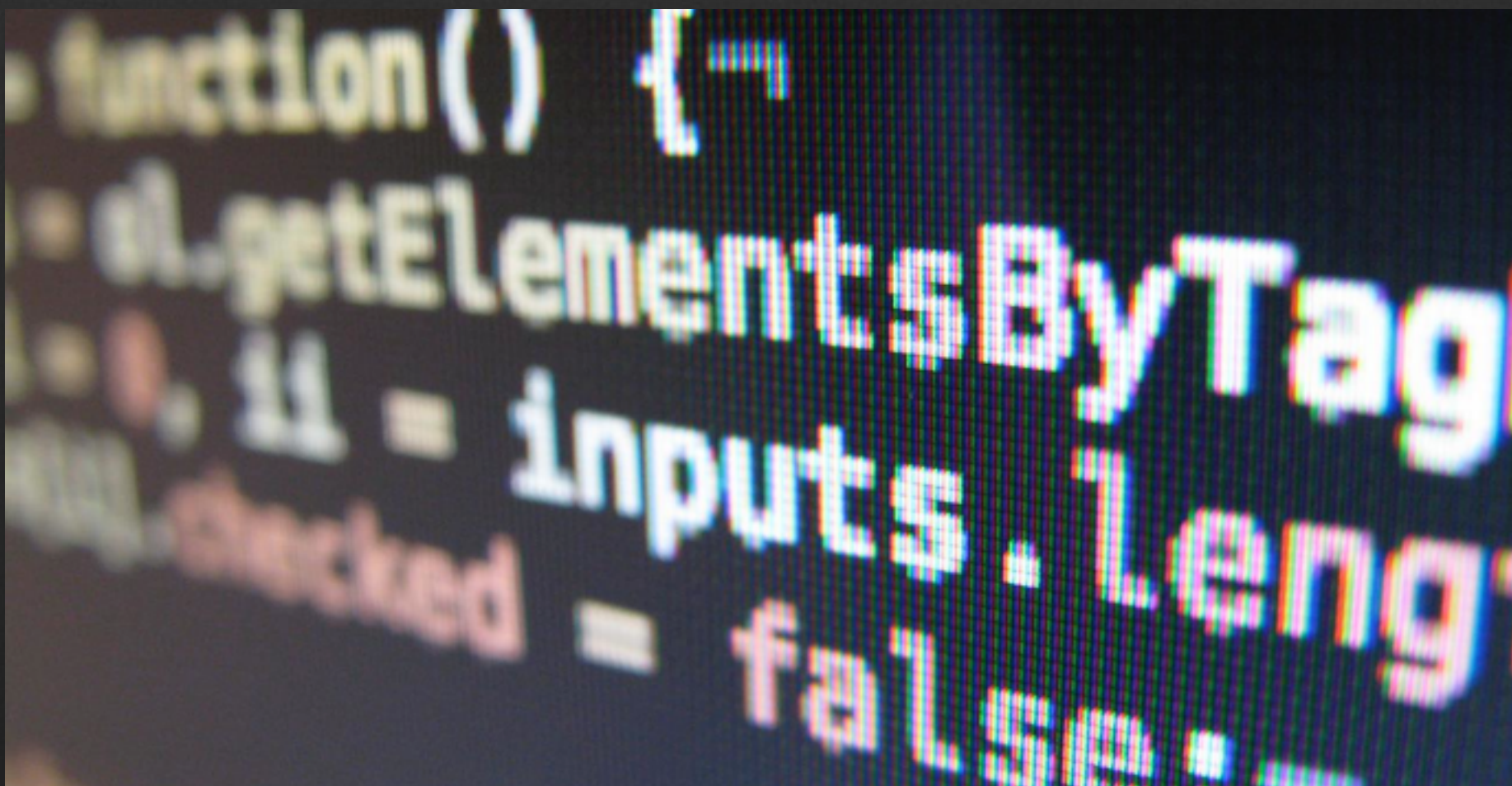
```
.mySelector {  
    -moz-transform : rotate(0deg) scale(1)  
    rotate(0deg);
```

```
    webkit-transform : rotate(0deg) scale(1)  
    rotate(0deg);  
}
```

Código :

```
$(".mySelector").CSS3Animate({  
    "property": ["-moz-transform", "webkit-transform"],  
    "transition-duration": 1000,  
    "css": {  
        "-moz-transform" : "rotate(350deg)  
scale(1) rotate(15deg)",  
        "webkit-transform" : "rotate(350deg)  
scale(1) rotate(15deg)"  
    },  
    "oncomplete" :function(){  
        alert("listo!")  
    }  
});
```


JAVASCRIPT



JavaScript se utiliza en miles de millones de páginas Web para añadir funcionalidad, validar formularios, comunicarse con el servidor y mucho más.

JavaScript o también abreviado muchas veces JS por la extensión de los archivos “.js”. Sirve para desencadenar o activar eventos motivados por acciones del cliente (ocurre del lado del navegador) por eso se lo conoce como un lenguaje del lado del cliente. Entonces esto se resume a

que JavaScript nos va ayudar a dar dinamismo a la página.

¿QUÉ ES UN EVENTO EN JAVASCRIPT?

Es cuando desencadenamos una porción de código en base a una acción del usuario. , como clic o doble clic, cargar la página, salir de la página, etc.

Debemos recordar siempre que en JavaScript: Es sensible a mayúsculas y minúsculas (Case Sensitive).

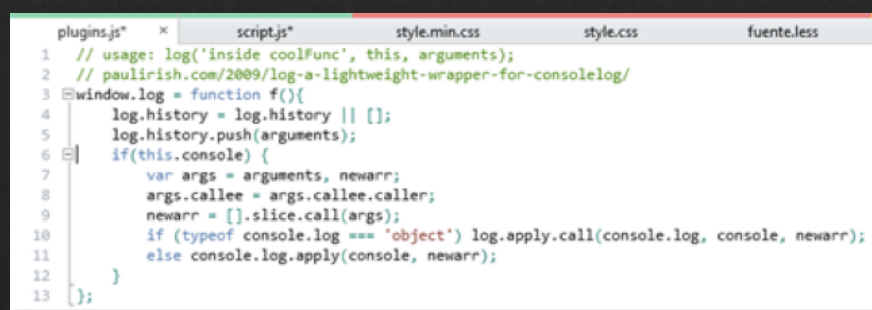
Se puede comentar de dos maneras:

```
/*  
se comenta esto  
*/
```

```
// este es el segundo comentario
```

Los “{ }” se utilizan para definir fragmentos de código.

El “;” nos ayuda a marcar el final de una sentencia.

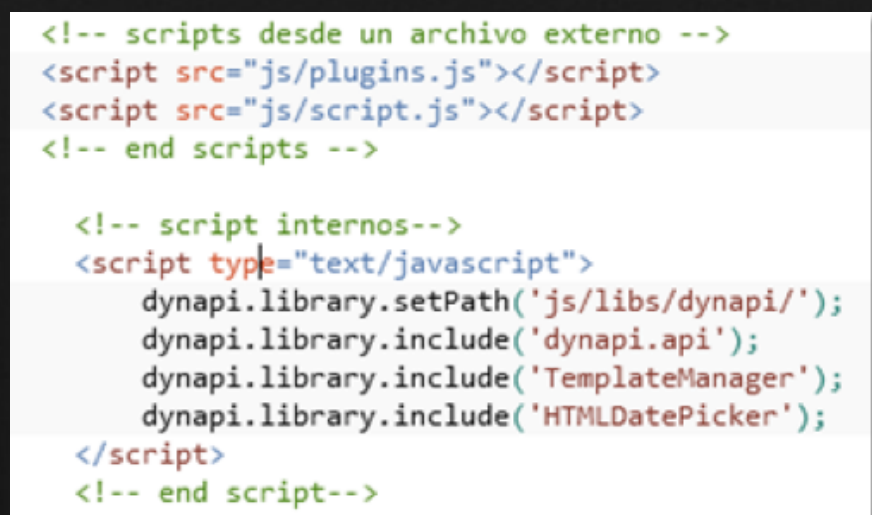


```
1 // usage: log('inside coolFunc', this, arguments);  
2 // paulirish.com/2009/log-a-lightweight-wrapper-for-consolelog/  
3 window.log = function f(){  
4   log.history = log.history || [];  
5   log.history.push(arguments);  
6   if(this.console) {  
7     var args = arguments, newarr;  
8     args.callee = args.callee.caller;  
9     newarr = [].slice.call(args);  
10    if (typeof console.log !== 'object') log.apply.call(console.log, console, newarr);  
11    else console.log.apply(console, newarr);  
12  }  
13 };
```

Existen dos formas de incorporar código JavaScript en nuestro proyecto:

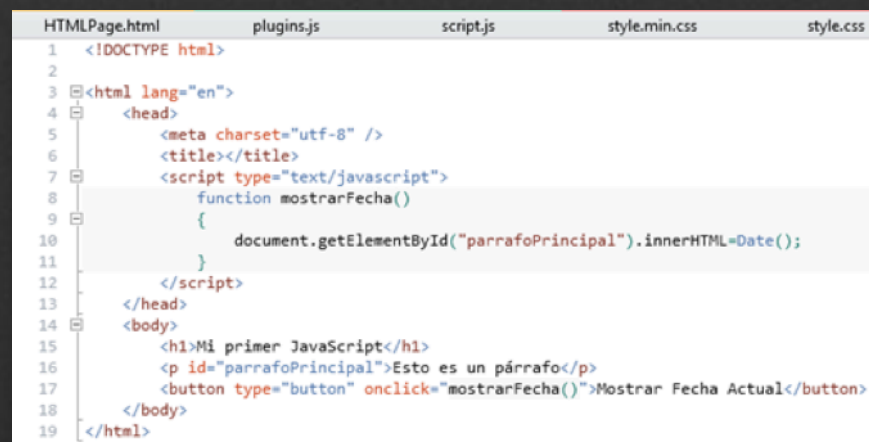
Externamente: cuando se hace referencia aún archivo “.js”.

Internamente: cuando el código JavaScript se encuentra en la misma página.



```
<!-- scripts desde un archivo externo -->  
<script src="js/plugins.js"></script>  
<script src="js/script.js"></script>  
<!-- end scripts -->  
  
<!-- script internos-->  
<script type="text/javascript">  
  dynapi.library.setPath('js/libs/dynapi/');  
  dynapi.library.include('dynapi.api');  
  dynapi.library.include('TemplateManager');  
  dynapi.library.include('HTMLDatePicker');  
</script>  
<!-- end script-->
```

Ejemplo de JavaScript básico



```
1 <!DOCTYPE html>  
2  
3 <html lang="en">  
4 <head>  
5   <meta charset="utf-8" />  
6   <title></title>  
7   <script type="text/javascript">  
8     function mostrarFecha()  
9     {  
10      document.getElementById("parrafoPrincipal").innerHTML=Date();  
11    }  
12  </script>  
13 </head>  
14 <body>  
15   <h1>Mi primer JavaScript</h1>  
16   <p id="parrafoPrincipal">Esto es un párrafo</p>  
17   <button type="button" onclick="mostrarFecha()">Mostrar Fecha Actual</button>  
18 </body>  
19 </html>
```

Lo que hemos hecho es abrir los tags desde la línea 7 para crear nuestro código JavaScript y procedimos a crear una función que inspeccionara nuestro documento en búsqueda del Id “parrafoPrincipal” para almacenar la fecha actual dentro de la etiqueta HTML que posea ese Id.

POSIBILIDADES INFINITAS PARA JAVASCRIPT

Chrome Experiments, es un sitio dedicado a la difusión de experimentos con tecnologías de punta para navegadores web. Fue lanzado hace unos 18 meses bajo el auspicio de Google, quien recién celebró la llegada del experimento número 100. Por supuesto, JavaScript y HTML5, son las tecnologías más relevantes usadas en el sitio para poner a prueba tu(s) navegador(es).

SALVANDO PREFERENCIAS DE USUARIO

El almacenamiento local HTML5 se utiliza para almacenar pares de clave-valor en el lado del cliente. Estos pares de clave-valor se pueden recuperar en páginas HTML que provengan del mismo dominio. Los datos de almacenamiento local se almacenan en el disco y se conservan después de reiniciar las aplicaciones. Para obtener información adicional sobre el almacenamiento local, lea la especificación en

<http://dev.w3.org/html5/webstorage>.

USO DEL ALMACENAMIENTO LOCAL HTML5 EN LAS APLICACIONES

El almacenamiento local HTML5 se puede utilizar en aplicaciones que deben guardar datos y preferencias de los usuarios de manera que se conserven después de haber reiniciado las aplicaciones. A continuación se ofrecen algunos ejemplos del uso del almacenamiento local HTML5 a fin de aumentar la funcionalidad de una aplicación web.

Juegos: El almacenamiento local se puede utilizar para guardar el progreso de un juego o las puntuaciones más altas, que más adelante se podrán recuperar.

Multimedia: Si una aplicación web integra una secuencia de audio o vídeo, la aplicación puede almacenar la marca de hora de estas secuencias en el almacenamiento local. Dado que estos datos se conservan entre distintos reinicios de la aplicación, puede iniciar la secuencia de audio o vídeo desde la última ubicación pausada.

Las API JavaScript* siguientes se pueden utilizar para almacenar, recuperar, eliminar o borrar los datos:

Almacenar un valor:

```
window.localStorage.setItem(keyinput, valinput)
```

```
var keyinput = "key10";  
var valoutput = "value10";  
if(typeof(window.localStorage) !=  
'undefined'){
```



```

window.localStorage.setItem(keyinput, valinput);
    }
    else{
        throw "window.localStorage, not
defined";
    }

```

Recuperar un valor:

```

window.localStorage.getItem (keyinput);
    var valoutput ;
    if(typeof(window.localStorage) !=
'undefined'){
        valoutput =
window.localStorage.getItem ("key10");
    }
    else{
        throw "window.localStorage, not
defined";
    }

```

Eliminar un valor:

```

window.localStorage.removeItem(keyinput);
    var valoutput ;
    if(typeof(window.localStorage) !=
'undefined'){
        valoutput = window.localStorage.
removeItem ("key10");
    }
    else{
        throw "window.localStorage, not
defined";
    }

```

Borrar todos los valores:

```

window.localStorage.clear()

        if(typeof(window.localStorage) !=
'undefined'){
            window.localStorage. clear() ;
        }
        else{
            throw "window.localStorage, not
defined";
        }

```

Código de ejemplo:
localStorageSample.html

```
<!DOCTYPE HTML>

<html>
  <head>
    <TITLE>HTML5 local storage tester
(testbed)</TITLE>
  </head>
  <body>
    <div id="output_area"
      style="position:relative;width:
100%;height:200px;overflow:auto;
border: dotted 1px #ff0000;">
    </div>

    <div>
      <input id="local_storage_key"
type="text" />
      <input
id="local_storage_value" type="text" />
    </div>

    <div>
      <input type="button"
value="load" onclick="load_local();" />
      <input type="button"
value="store" onclick="store_local();" />
    </div>
    <script id="this_page_js"
type="text/javascript">

      /* store some string data to local
storage

      This is using some text input
elements on the page
```

```
for input.*/

function store_local(domid_prefix){
  var keyinput =
document.getElementById('local_storage_ke
y').value;

  var valinput =
document.getElementById('local_storage_val
ue').value;

  try{

if(typeof(window.localStorage) != 'undefined')
{

window.localStorage.setItem(keyinput, valinp
ut);

      }
      else{
        throw
"window.localStorage, not defined";
      }
    }
    catch(err){

output_str("store_local,error," + err);

      }
    }

/* load some string data from local
storage

This is using some text input
elements on the page
for the key name input and
value output.*/

function load_local(domid_prefix){
```

```

        var keyinput =
document.getElementById('local_storage_key').value;
        var valoutput =
document.getElementById('local_storage_value');
        try {
            if(typeof(window.localStorage)
!= 'undefined') {
                valoutput.value =
window.localStorage.getItem(keyinput);
            }
            else {
                throw "window.localStorage,
not defined";
            }
        }
        catch(err) {
            output_str("store_local,error,"
+ err);
        }
    }

    /* function to print to the debug
area */

    function output_str(str){
        var da =
document.getElementById("output_area");
        da.innerHTML += str + "<br/
>";
        da.scrollTop = da.scrollHeight;
    }
</script>
</body>
</html>

```


WEB SOCKETS

Igual que XMLSockets en Actionscript, Web Sockets permite hacer aplicaciones multiusuario en tiempo real, como juegos, chats, notificaciones, etc. Si el navegador no tiene soporte de Web Sockets, es posible usar implementaciones multiuser en Javascript como PubSubHubBub

<http://code.google.com/p/pubsubhubbub/>

WebSockets es una tecnología que está surgiendo de la mano de HTML5 y estará pronto disponible en todos nuestros browsers en poco tiempo.

WebSockets es una tecnología que nos da canales de comunicación bidireccional, full-duplex a través de un sencillo socket TCP. Traducido ésto a nuestro ámbito más cotidiano, es simplemente server push. Vamos a poder comunicarnos real-time con nuestros clientes conectados cosas que trataba de emular la técnica Comet la cual hacía vivir un request http durante X tiempo y donde el server escribía en ese response emulando ser un push.

Hoy no son muchas las versiones estables de browsers que lo soportan. No es el caso de Google Chrome el cual viene con soporte para WebSockets desde su versión 4.

Google Chrome 4 +

Internet Explorer 9 beta +

Firefox 4 beta +

Safari 5 +

Opera 10.70 +

Ahora veamos un poco de código. Lo primero que tenemos que saber es si el browser conectado soporta WebSockets, lo cual podemos comprobarlo de manera sencilla.

view plain**copy to clipboard****print?**

```
var support = window.WebSocket != null;
```

Ahora que sabemos que nuestro browser soporta WebSockets podemos empezar creando uno.

view plain**copy to clipboard****print?**

```
var socket = new WebSocket("/mysocket");
```

Una vez creado el objeto WebSocket vamos a agregarle una serie de listeners para poder manipular los eventos que ocurran en el mismo.

view plaincopy to clipboardprint?

```
socket.onopen = function(e) {  
    alert("Socket is connected");  
};
```

```
socket.onclose = function(e) {  
    alert("Socket is closed :(");  
}
```

// y finalmente el que mas nos interesa, el push del server

```
socket.onmessage = function(e) {  
    alert("Server sent a message: " + e.data);  
}
```

Sin dudas que el evento más importante acá es el onmessage. El parámetro e.data puede ser texto plano, json, xml o cualquier formato que estén acostumbrados a utilizar a diario.

Ahora vamos a ver un ejemplo de cuando el cliente quiere comunicarse con el server enviándole un sencillo mensaje.

view plaincopy to clipboardprint?

```
socket.send("data goes here");
```

De ésta manera podemos tener una comunicación bidireccional de manera sencilla.

Ahora veamos algunos escenarios donde aplica la utilización de WebSockets

Es aplicable donde un servicio TCP debe ser llevado a una arquitectura web (como por ejemplo el servicio de mensajería XMPP)

Donde una aplicación web necesita comunicar datos en real-time a sus clientes (por ejemplo una aplicación de stocks online)

WebSockets es un avance significativo para la web donde las aplicaciones en browsers se parecen cada vez más a las de escritorio pero a su vez hiper conectadas entre sí.

ALGUNOS RECURSOS

WebSockets –

<http://www.websockets.com>

W3C WebSockets –

<http://dev.w3.org/html5/websockets/>

WebSockets en Wikipedia –

<http://en.wikipedia.org/wiki/WebSockets>

GEOLOCALIZACIÓN

Una de las principales novedades de HTML5 fue la aparición de nuevas APIs de Javascript que aumentan la potencia de este lenguaje.

Una de ellas es la nueva API de geolocalización, que nos permite conocer la ubicación geográfica del usuario, siempre y cuando esté usando un navegador que la tenga implementada y que el usuario dé su permiso. El navegador hará uso de muchos métodos (GPS, Skyhook, Google Geo, IP) para darte la latitud y longitud de tus usuarios. Obviamente, ellos tienen que dar permiso. Lo mejor es que funciona en cualquier PC, no sólo en teléfonos

Aunque la primera impresión sea que sólo será útil para usuarios de navegadores móviles, la realidad es que éste utiliza otros medios además del GPS para calcular la ubicación del usuario, como por ejemplo a través de su dirección IP.

Antes de empezar a usar el API de geolocalización tendremos que comprobar que el navegador la soporta:

```
if (navigator.geolocation)
{
    // Código de la aplicación
}
else
{
    // No hay soporte para la geolocalización: podemos desistir o utilizar algún método alternativo
}
```

LA FUNCIÓN

NAVIGATOR.GEOLOCATION.GETCURRENTPOSITION

La ubicación del usuario se obtiene a través de la siguiente función:

```
navigator.geolocation.getCurrentPosition(funcExito, funcError, opciones);
```

Esta función tiene tres parámetros (los dos últimos opcionales):

funcExito: función de retorno que se ejecutará si se obtiene la posición. Se le pasará como parámetro un objeto `Position`.

funcError: función de retorno que se ejecutará si no se obtiene la posición. Se le pasará como parámetro un objeto `PositionError`.

opciones: un objeto `PositionOptions` con parámetros para la obtención de la localización.

La función intentará obtener la posición del usuario. Si lo consigue llamará a la función que le pasemos como primer parámetro pasándole un objeto `Position` con esos datos. Si, por el contrario, no lo consigue, llamará a la función que le pasemos como segundo parámetro pasándole un objeto `PositionError` que indicará la razón por la que no lo ha conseguido.

EL OBJETO POSITION

Es el objeto que nos indicará la ubicación del usuario si el navegador puede determinarla. Este objeto consta de los siguientes atributos:

Atributo

Tipo de dato

Descripción

`coords.latitude`

double

Latitud en grados decimales

`coords.longitude`

double

Longitud en grados decimales

`coords.accuracy`

double

Precisión en metros

timestamp

`DOMTimeStamp`

Momento de la toma de estos datos

`coords.altitude`

double o null

Altitud en metros

`coords.altitudeAccuracy`

double o null

Precisión de la altitud en metros

`coords.heading`

double o null

Orientación en grados decimales en el sentido de las agujas del reloj

`coords.speed`

double o null

Velocidad en metros/segundo

De todos los atributos sólo

`coords.latitude`, `coords`

De todos los atributos sólo

`coords.latitude`, `coords.longitude` y `coords.accuracy` es seguro que tendrán valor. Los otros dependen de las aptitudes del dispositivo que esté usando el usuario.

EL OBJETO POSITIONERROR

Es el objeto que nos indicará la causa por la que no se ha podido determinar la ubicación del usuario. Este objeto consta de dos atributos: `code` y `message`. De estos el que nos interesa es `code` que será el que nos indique el error de forma más eficiente (`message` hace lo mismo pero con una cadena explicativa del error).

Valor

Valor numérico

Descripción

`PERMISSION_DENIED`

1 El usuario ha denegado el acceso a la obtención de su ubicación

`POSITION_UNAVAILABLE`

2 No se ha podido obtener la ubicación del usuario por alguna razón

`TIMEOUT`

3 Se ha agotado el tiempo de espera para obtener la ubicación

El objeto `PositionOptions`

Es el objeto que nos permitirá poner algunas condiciones a la obtención de la ubicación del usuario. Este objeto consta de los siguientes atributos:

Atributo

Tipo de dato

Valor por defecto

Descripción

`enableHighAccuracy`

boolean

false

Si el dispositivo y el usuario lo permiten el navegador intentará obtener la ubicación del usuario con una mayor preci-

sión. Esto suele suponer un mayor coste de recursos.

timeout

long

No tiene

Tiempo de espera máximo para obtener la posición en milisegundos. Este tiempo empieza a contar desde que el usuario da su permiso, no antes.

maximumAge

long

0

Antigüedad máxima en milisegundos. Con el valor por defecto (0), cada vez que se pide la posición se vuelve a calcular. Si ponemos algún valor mayor que cero, se busca en la caché y si hay una posición tomada anteriormente no más antigua que el valor dado, se devuelve inmediatamente, ahorrando muchos recursos. Con el valor Infinity siempre se devolverá un valor de la caché.

Hay que tener en cuenta que algunos dispositivos tienen distintos métodos para obtener la posición si enableHighAccuracy está activado y si está desactivado, con lo es posible que una llamada a get-

CurrentPosition con esta opción activada dé error mientras que puede tener éxito con ella desactivada.

OBTENER LA POSICIÓN DEL USUARIO

Veamos ahora el código para obtener la posición del usuario con: comprobación de si el navegador soporta geolocalización, recogida de los datos de latitud y longitud, control de errores y pasándole las opciones de tiempo máximo de espera y antigüedad.

```
(function(){  
    var content =  
document.getElementById("geolocation-  
test");  
  
    if (navigator.geolocation)  
    {  
  
navigator.geolocation.getCurrentPosition(function(objPosition)  
    {  
        var lon =  
objPosition.coords.longitude;  
        var lat =  
objPosition.coords.latitude;  
  
        content.innerHTML =  
"<p><strong>Latitud:</strong> " + lat + "</p><p><strong>Longitud:</strong> " + lon +  
"</p>";  
    }  
    }  
})
```



```

    }, function(objPositionError)
    {
        switch (objPositionError.code)
        {
            case
objPositionError.PERMISSION_DENIED:
                content.innerHTML =
"No se ha permitido el acceso a la posición
del usuario.";
                break;
            case
objPositionError.POSITION_UNAVAILABLE:
                content.innerHTML =
"No se ha podido acceder a la información
de su posición.";
                break;
            case
objPositionError.TIMEOUT:
                content.innerHTML = "El
servicio ha tardado demasiado tiempo en
responder.";
                break;
            default:
                content.innerHTML =
"Error desconocido.";
        }
    }, {
        maximumAge: 75000,
        timeout: 15000
    });
}
else
{
    content.innerHTML = "Su navegador no
soporta la API de geolocalización.";
}
})();

```

Y en el siguiente cuadro podemos ver el resultado de este script aplicado en esta página:

Latitud: 40.4373898

Longitud: -3.6987106

EJEMPLO DE CÓMO CAPTURAR Y MOSTRAR LOS DATOS DE GEOLOCALIZACIÓN

```
<html>
```

```
<head>
```

```
<title>Localizador</title>
```

```
<script type="text/javascript"
src="jquery-1.7.1.min.js"></script>
```

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

```
<style>
```

```
body{
```

```
font-family: "Helvetica Neue", "Helvetica",
Arial, Verdana, sans-serif;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<header>
```

```
<h1>Localizador mediante HTML5</h1>
```

```
</header>
```

```
<section>
```

```
<article>
```

```
<div id='map_canvas' style='width:100%;
height:400px;'></div>
```

```
</article>
```

```
<div id="respuesta">
```

```
</div>
```

```
</section>
```

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

```
<script type="text/javascript">
```

```
var map;
```

```
var latitud;
```

```
var longitud;
```

```
var precision;
```

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

```
localizame();
```

```
});
```

```
function localizame() {
```

```
if (navigator.geolocation) {
```

```
navigator.geolocation.getCurrentPosition(coo
rdenadas, errores);
```

```
}else{
```

```
alert('Oops! Tu navegador no soporta
geolocalización. Bájate Chrome, que es
gratis!');
```

```
}
```

```

}

function coordenadas(position) {

    latitud = position.coords.latitude;

    longitud = position.coords.longitude;

    precision = position.coords.accuracy;

    cargarMapa();

    alert("Datos con una precisión de " +
    precision/1000 + " km, " + precision + "
    metros");

}

function errores(err) {

    if (err.code == 0) {

        alert("Oops! Algo ha salido mal");

    }

    if (err.code == 1) {

        alert("Oops! No has aceptado compartir tu
        posición");

    }

    if (err.code == 2) {

        alert("Oops! No se puede obtener la posición
        actual");
    }

```

```

}

if (err.code == 3) {

    alert("Oops! Hemos superado el tiempo de
    espera");

}

}

function cargarMapa() {

    var latlon = new
    google.maps.LatLng(latitud,longitud);

    var myOptions = {

        zoom: 17,

        center: latlon,

        mapTypeId:
        google.maps.MapTypeId.ROADMAP

    };

    map = new google.maps.Map($
    ("#map_canvas").get(0), myOptions);

    var coorMarcador = new
    google.maps.LatLng(latitud,longitud);
    var marcador = new google.maps.Marker({

        position: coorMarcador,
    }

```


map: map,

title: "Dónde estoy?"

});

}

</script>

</body>

</html>

WEB WORKERS

¿Sabían que Javascript sólo puede hacer una cosa al tiempo? Gran parte de la razón por la que Wave falló y las web apps son simples es porque la multitarea es imposible. Web Workers soluciona eso. Web Workers permite tener multiples .js corriendo en paralelo en una misma página. Haciendo tareas complejas más veloces gracias al multithreading.

Los web workers permiten la ejecución de scripts en segundo plano, son procesos de larga duración y pueden consumir mucha memoria que hasta ahora se podían implementar con la función `window.setTimeout()`. Los workers tienen las siguientes ventajas: se ejecutan en threads separados, de forma concurrente, no bloquean la interfaz de usuario, pueden ser dedicados (al tab) o compartidos por varios tabs o por la ventana e, incluso, podrían persistir al cierre de la misma.

EJEMPLO WEB WORKERS

```
<script language="javascript">
  var worker = new Worker('myjavascript.js');
  worker.onmessage = function(event)
  { alert(event.data); };
</script>
```

OTROS PUNTOS A TENER EN CUENTA CON WEBWORKERS

Creación de un worker

Múltiples workers

Parámetros en el worker

Operaciones con parámetros

Procesar matrices

DOS EJEMPLOS DE HTML Y JS TRABAJANDO JUNTOS

```
index.html | index.html | index2.html | index.html | worker.js
1 <!doctype html>
2 <html>
3   <body>
4     <p>El numero primo mas alto hasta el momento es: <output
id="resultado"></output></p>
5     <script>
6       var worker = new Worker('worker.js');
7       worker.onmessage = function(event){
8         document.getElementById('resultado').textContent =
event.data;
9       };
10    </script>
11  </body>
12 </html>

index.html | index.html | index2.html | index.html | *worker.js
1 var n=1;
2 search: while(true){
3   n +=1;
4   for(var i = 2; i<= Math.sqrt(n);i+=1)
5     if(n % i == 0)
6       continue search;
7
8   postMessage(n);
9 }
```

```
1 <html>
2   <head>
3     <script type="text/javascript">
4       var worker = new Worker('worker.js');
5       worker.onmessage = function(event){
6         alert("Del worker, he recibido el siguiente
mensaje:"+event.data);
7       }
8       worker.postMessage("|Yo digo hola");
9     </script>
10  </head>
11  <body>
12  </body>
13 </html>

index.html | worker.js
1 this.onmessage = function(event){
2   postMessage("El worker dice que: "+event.data);
3 }
```


WEBGL

WebGL creció desde los experimentos del canvas 3D comenzados por Vladimir Vukićević en Mozilla. El primero mostró un prototipo de Canvas 3D en 2006. A finales de 2007, tanto Mozilla² como Opera³ habían hecho sus propias implementaciones separadas. A principios de 2009 Mozilla y Khronos formaron el WebGL Working Group (Grupo de Trabajo del WebGL).

El Grupo de Trabajo del WebGL incluye Apple, Google, Mozilla, y Opera, y WebGL ya está presente en las builds de Mozilla Firefox, Mozilla Fennec, Google Chrome y también en la versión de Safari incorporada en OS X Lion (Safari 5.1).

Notables primeras aplicaciones de WebGL incluyen Google Maps y Zygote Body.

DISEÑO

WebGL está basado en OpenGL ES 2.0 y proporciona una API para gráficos 3D. Se utiliza el elemento canvas HTML5 y se accede mediante interfaces Document Object Model. Gestión de memoria auto-

mática se proporciona como parte del lenguaje JavaScript.

WebGL carece de las rutinas matemáticas matriz eliminadas en OpenGL 3.0. Esta funcionalidad debe ser proporcionada por el usuario en el espacio de código JavaScript; este código necesario se complementa con frecuencia con una biblioteca de matriz tal como glMatrix, TDL, o MJS.

IMPLEMENTACIÓN

Actualmente es soportado principalmente en Internet Explorer (versión 11), Google Chrome y Mozilla Firefox aunque también funciona con limitaciones en Opera browser y en Safari bajo el sistema operativo de Apple para computadoras de escritorio.

Numerosas implementaciones han sido desarrolladas además para buscadores de plataformas móviles.

BIBLIOTECAS EN JAVASCRIPT

Como WebGL es una tecnología diseñada para trabajar directamente con el

GPU (unidad de procesamiento gráfico) es difícil de codificar en comparación con otros estándares web más accesibles, es por eso que muchas bibliotecas de JavaScript han surgido para resolver este problema:

C3DL, CopperLight, Curve3D, CubicVR, EnergizeGL, GammaJS, GLGE, GTW, JS3D, Kuda, O3D, OSG.JS, PhiloGL, Pre3d, SceneJS, SpiderGL, TDL, Three.js, X3DOM.

Entre ellas Three.js es la más popular en términos de número de usuarios. Es ligera y tiene un bajo nivel de complejidad en comparación con la especificación WebGL original.

CREACIÓN DEL CONTENIDO

Las escenas WebGL se pueden crear sin necesidad de programación utilizando una herramienta de creación de contenidos, como Blender o con Autodesk Maya. Las escenas luego se exportan a WebGL. Esto fue posible por primera vez Inka3D, un plugin de exportación WebGL para Maya. También hay servicios para publicar contenido en línea 3D interactivo utilizando WebGL como p3d.in y Sketchfab.

IMAGEN INTERACTIVA 3.1 Ejemplo de animación html 5 integrada



Realizado con el programa Hype

DRAG & DROP

Vete a gmail, crea un email e intenta arrastrar un archivo del explorador de archivos al mail. Verás que es posible adjuntarlo con sólo arrastrarlo. El gesto de arrastrar y soltar ahora es posible gracias a HTML5. Puedes traer trozos de datos o archivos enteros.

EJEMPLO DE DRAG AND DROP

```
<!DOCTYPE HTML>
<html> <head>
<script>
function allowDrop(ev)
{
ev.preventDefault();
}

function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}

function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElement
ById(data));
```

```
}
</script> </head>
<body>

<div id="div1" ondrop="drop(event)"
ondragover="allowDrop(event)"></div>



</body>
</html>
```


SVG Y CANVAS

SVG

renderización de gráficos en función de un documento SVG (Scalable Vector Graphics),

```
<svg>
  <circle id="myCircle"
    cx="100" cy="75" r="50"
    fill="blue"
    stroke="firebrick"
    stroke-width="3" />
  <text x="60" y="155">Hello World</text>
</svg>
```

CANVAS

nos permitirá renderizar imágenes sin necesidad de plugins, generando de forma dinámica el contenido a través de javascript , y es el elemento de la polémica, en el que muchos han visto la muerte del flash, como hasta ahora lo conocíamos

```
<canvas id="mycanvas" width="200"
height="100">
```

Your browser does not support the

```
<code>canvas</code> element.
```

```
</canvas>
```

```
<script language="javascript">
```

```
var canvas =
document.getElementById('mycanvas');
```

```
var context = canvas.getContext('2d');
```

```
contexto.fillRect(50, 0, 10, 150);
```

```
</script>
```

HTML5 EN IPHONE, ANDROID E IPAD

El objetivo es lograr que las páginas web se vean correctamente en todos los navegadores de teléfonos y otros equipos portables como tabletas y eBooks. Códigos y scripts para identificar el navegador del usuario y cargar el estilo CSS que permita mostrar una página web correctamente.

El gran problema, no es solo el de crear nuevo contenido que sea compatible, sino también el de adaptar páginas creadas con anterioridad y que no se muestren debidamente en estos dispositivos.

Quizás no sea necesario adaptar todas, pero si las más demandadas en cada caso.

Existen varias opciones o alternativas para lograrlo:

- Crear versiones de páginas optimizadas para móviles, detectar y dirigir estos dispositivos a ellas.
- Crear versiones de páginas optimizadas para móviles y solo colocar un link en la parte superior de las páginas normales,

para que el usuario manualmente las cargue.

- Seguir utilizando las mismas páginas estándar pero utilizar un script que detecte los móviles y en ese caso carguen un estilo específico solo para ellos.
- Utilizar páginas optimizadas para móviles pero utilizar un script para que los navegadores estándar de escritorio usen un estilo apropiado para ellos.
- Por último el método que recomienda Google, usar Responsive web design. Algo en español como "diseño web responsable o adaptable".

DETECTAR NAVEGACIÓN CON IPHONE, IPOD O IPAD

Una vez tomada la decisión de que necesitamos una adaptación de nuestros contenidos para iPhone, lo que tenemos que hacer es colocar un código en la cabecera de nuestras páginas, de este modo, cada

vez que un usuario acceda a través de su iPhone a nuestra página, será redirigido a la adaptación que hemos diseñado para él.

El código en javascript

```
<script type="text/javascript"> if
((navigator.userAgent.indexOf('iPhone') != -1)
| (navigator.userAgent.indexOf('iPod') != -1) )
| (navigator.userAgent.indexOf('iPad') != -1) )
{
/* iPhone user */
window.location.href="iphone/";
}
</script>
```

Lo único que debéis cambiar es el `window.location` y sustituir “iphone/” por la url en la que tengáis colocada vuestra adaptación web para iPhone.

SCRIPT PARA DETECTAR DISPOSITIVOS MÓVILES Y REDIRIGIRLOS A UN DIRECTORIO

El siguiente script en lenguaje PHP, insertado al comienzo de una página `index.php`, re-direccionará a los dispositivos móviles cuyos agentes de usuario se especifiquen, a un directorio llamado en este ejemplo `mobile`, que contiene las ver-

siones optimizadas para estos dispositivos.

En este ejemplo se especifican solo tres agentes de usuario (iphone, ipad y kindle), pero se pueden usar varios.

Los navegadores estándar continuaran leyendo el resto de la página original.

```
<?php
$navigator_user_agent =
(isset($_SERVER['HTTP_USER_AGENT'])) ?
strtolower($_SERVER['HTTP_USER_AGENT'])
:'';
if(
  strstr($navigator_user_agent, "iphone")or
  strstr($navigator_user_agent, "ipad")or
  strstr($navigator_user_agent, "kindle")
)
{
  header("Location: mobile/index.html");
}
?>
<html>
Resto del código fuente de la página
</html>
```

Responsive design

Responsive web design no es más que servir a todo tipo de dispositivos, la misma página o sea el mismo contenido HTML, pero utilizar CSS3 para definir la forma en que se representa, de acuerdo al tamaño de su pantalla.

Para eso se pueden emplear los comandos CSS: "@media handheld" y "@media screen".

EJEMPLO DEL USO DE @MEDIA HANDHELD

En el siguiente ejemplo, de forma predefinida el navegador cargará el archivo de estilo normal.css, si es un dispositivo portable el que carga la página, entonces leerá el estilo inline, que modificará el ancho y tamaño de la fuente especificado en el archivo anterior

```
<link rel="stylesheet" media="screen" type="text/css" href="normal.css" />
```

```
<style type="text/css">
@media handheld
body{width:98%;font-family:Verdana;font-size:16px;}
</style>
```

Usar el comando @media screen

@media screen es algo parecido, la diferencia es que se aplica un estilo determinado solo cuando cambia el tamaño de la pantalla del dispositivo.

En el siguiente ejemplo, se usa para cambiar el tamaño de la fuente de la página y

ocultar la barra lateral, se activa cuando el ancho del navegador es inferior a 800px.

Se puede probar reduciendo el ancho de la ventana del navegador, incluso con un navegador de escritorio como Google Chrome.

```
@media screen and (min-width:300px) and (max-width:800px) {
body{font-size:0.8em;}
sidebar{display:none;}
}
```

Usar CCS3 para disminuir el tamaño de las imágenes

Unas de las reglas fundamentales para lograr una buena experiencia en los dispositivos móviles, es evitar la barra horizontal que incomoda y casi hace imposible el desplazamiento por la página al usuario. Se puede hacer que las imágenes en una página, se reduzcan de forma automática a la pantalla del dispositivo para evitar que aparezca la barra deslizante horizontal de la siguiente forma:

```
<style>
img{max-width:100%;}
</style>
```

EJEMPLO PRÁCTICO DEL USO DE CSS PARA REDUCIR Y ENCAJAR LAS IMÁGENES EN UNA PAGINA

Como comprobar cómo se ven las páginas web en dispositivos diferentes

Usar la Vista de diseño adaptable del navegador Firefox

El navegador Firefox incluye desde la versión 15 en las Herramientas de desarrolladores, una nueva utilidad llamada: "Vista de diseño adaptable".

Permite comprobar cómo se ve la página en que se está navegando, en las pantallas de los más populares dispositivos portables.

En el menú se puede escoger entre las siguientes resoluciones:

320 x 480 - iPhone (3G - 3GS)

360 x 640

768 x 1024 - iPad 1 & 2

800 x 1280 - Nexus 4 Nokia Lumia

980 x 1280 Kindle Fire HD 7

1280 x 600 iPhone 5

1920 x 900 Microsoft Surface

Para abrir la Vista de diseño adaptable usa las teclas Control + Mayus + M. Está en Herramientas > desarrollo web

En la actualidad varios servicios de internet permiten comprobar cómo se ve una página web específica en distintos dispositivos, solo es necesario introducir la dirección URL y ver el resultado, de acuerdo a las posibilidades que brinde el servi-

cio. Uno de los servicios más eficientes es Pikock.com

Para probarlo carga en tu navegador:

<http://www.pikock.com/en/pi-responsive.htm>

Escoge en el menú de la parte superior el dispositivo que se necesita simular.

