TESLA
TECHNOLOGIES

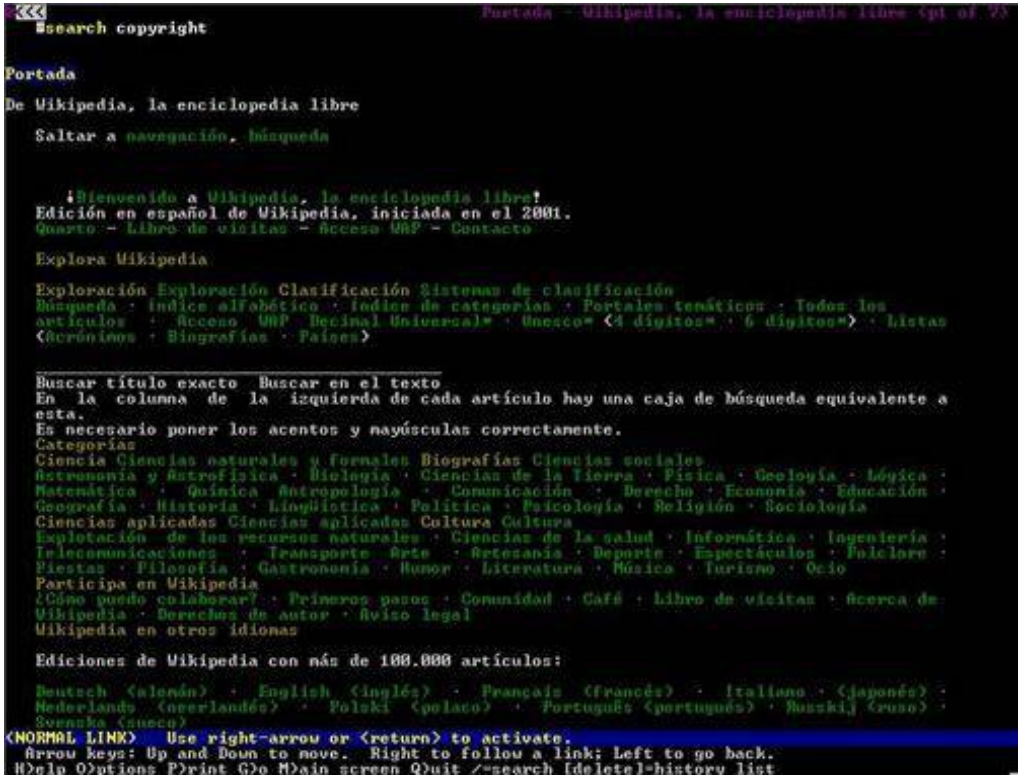
Título de la ponencia	Introducción a HTML5

Esteban Vázquez Ferreiro
Antonio Varela Nieto

5 de Marzo 2013

TESLA TECHNOLOGIES

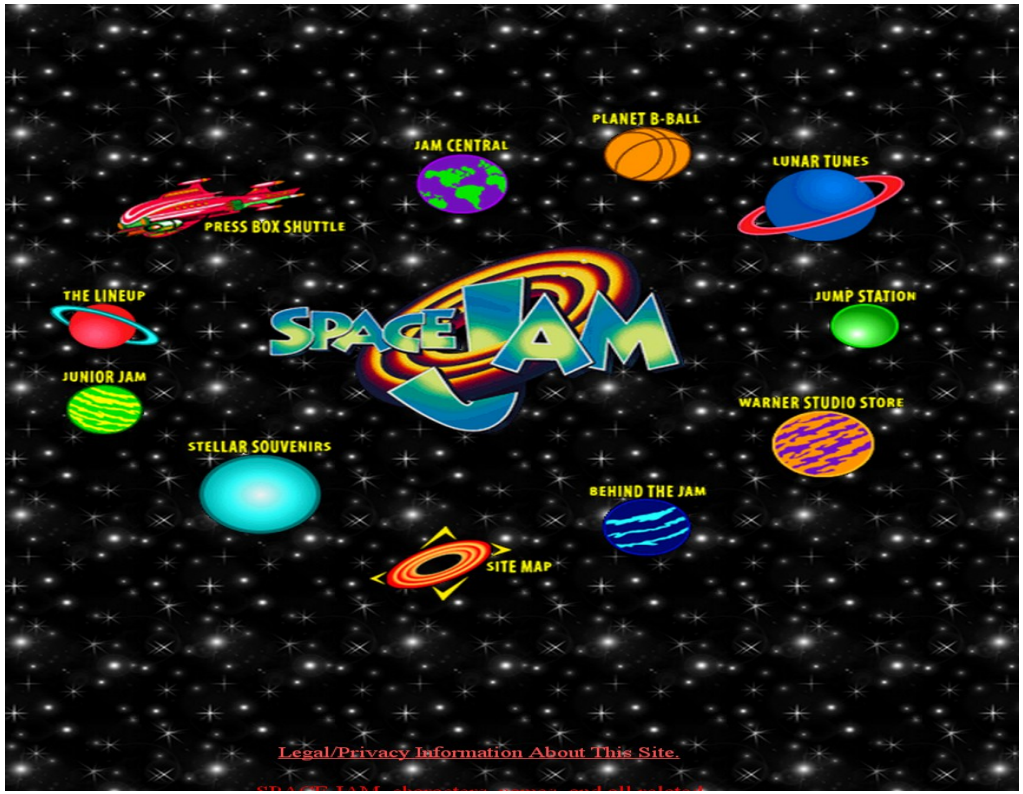
Introducción: El origen de los tiempos



http://lynx.browser.org/

TESLA
TECHNOLOGIES

Introducción: El origen de los tiempos



http://www2.warnerbros.com/spacejam/movie/jam.htm

TESLA TECHNOLOGIES

Introducción: El origen de los tiempos

http://wonder-tonic.com/geocitiesizer/

TESLA
TECHNOLOGIES

Introducción: El origen de los tiempos

http://sprite.com.br/

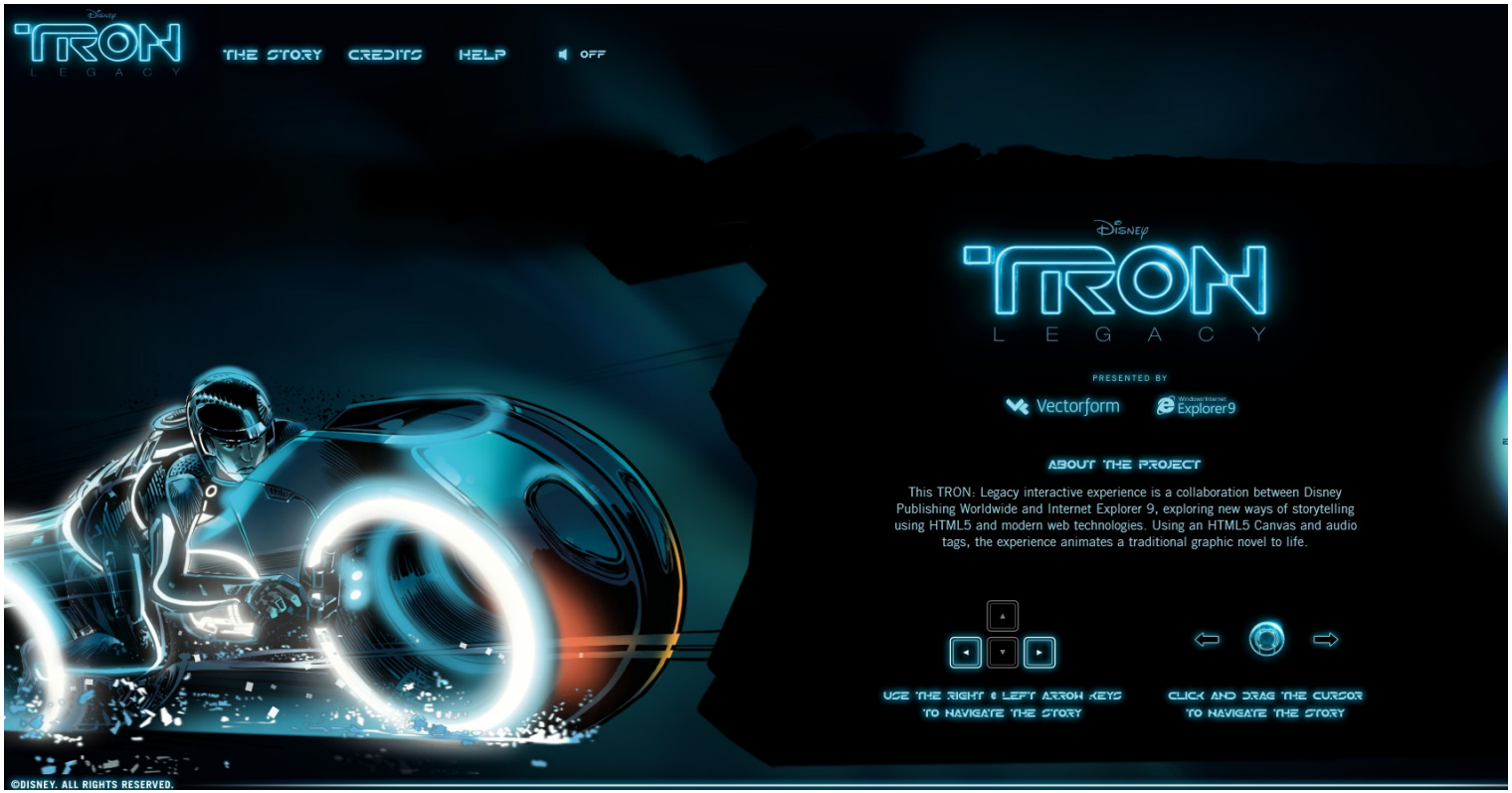
TESLA
TECHNOLOGIES

Introducción: El origen de los tiempos

http://www.teslatechnologies.net/html5

TESLA
TECHNOLOGIES

Introducción: El origen de los tiempos



http://disneydigitalbooks.go.com/tron/

TESLA
TECHNOLOGIES

Introducción: El estándar

- El futuro de la web pasa por el nuevo estándar de la **W3C**: **HTML5**.

- Cada vez son más los navegadores y dispositivos que aumentan la compatibilidad con **HTML5**.

- Pero, ¿**cuál es el origen del estándar de HTML**?
  - Gran parte de las etiquetas se fueron incorporando al estándar a raíz de discusiones en foros especializados y listas de correo.
  - Este sistema se remonta al año 1993, antes de la creación de la **W3C**.

Introducción: El estándar

- ¿Quién participaba en esas discusiones?
  - **Marc Andreessen**: Padre de **X Mosaic**, precursor de **Netscape** y luego **Mozilla**.
  - **Tony Johnson**: Padre de **Midas** (uno los primeros navegadores multiplataforma).
  - **Tim Beners-Lee**: El padre de la web actual, entre otros muchos.

TESLA
TECHNOLOGIES

Introducción: El estándar

- Hechos destacados:

- En **diciembre** de **1997**, la **World Wide Web Consorcium** publicó el **estándar HTML4**.

- De la especulación de insertar **XML** en **HTML** nació el **XHTML 1.0** en **diciembre** de **1998**.
  - Y su revisión **XHTML 1.1** en **mayo** de **2001**.

Hablamos de estándares de hace 10 años, respecto de los cuales se han detectado algunos problemas como la **polémica** entre **HTML** y **XHTML** sobre **la gestión draconiana de los errores.**

TESLA
TECHNOLOGIES

Introducción: Objetivos de la web actual

- ¿Qué se busca en la web actual?

1. **Compatibilidad con versiones anteriores**. HTML, CSS, DOM y Javascript.
2. **Gestión de errores clara**. Buscar que los navegadores no se tengan que inventar su propio código de error.
3. **Los usuarios no deben de estar expuestos a errores de autorización**. La especificación debe recoger cómo recuperarse de un error en cada escenario posible.
4. **Uso práctico**. Cada característica nueva debe tener un caso de uso práctico que la justifique.
   - A la inversa no tiene por qué cumplirse: un caso de uso no implica una nueva característica.

TESLA
TECHNOLOGIES

Introducción: Objetivos de la web actual

1. **Se debe evitar el Scripting**. En caso de uso, debe ser un lenguaje declarativo.
2. **Se deben evitar las características específicas**. Las aplicaciones web tienen que ser independientes de dónde se ejecuten.
3. **Debe ser un proceso abierto.**

La web siempre se ha beneficiado de la construcción sobre estándares libres, objeto de discusión entre los múltiples implicados.

TESLA
TECHNOLOGIES

HTML5: Nuevas etiquetas semánticas

TESLA TECHNOLOGIES

Doctype

- Para ver la estructura de un documento en **HTML5** debemos comenzar por definir el Doctype. Un ejemplo clásico es el siguiente:

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

TESLA
TECHNOLOGIES

Doctype

- La definición de **doctype** tiene una larga historia, incluyendo una serie de problemas que le surgieron a Microsoft cuando estaba desarrollando la versión de **Explorer** para **Mac**:

- Su nuevo **navegador** cumplía bien el nuevo estándar pero era **incapaz** de leer las páginas antiguas.

TESLA
TECHNOLOGIES

Doctype

- En la actualidad tenemos la nada despreciable cifra de **15 doctypes distintos**, soportados por los diferentes navegadores.

- Podemos mantener la estructura actual del doctype o bien podemos cambiarla a **HTML5**, lo cual nos proporciona un doctype mucho mas corto y fácil de usar:

<!DOCTYPE html>

Elemento raíz o root

- Una página en **HTML** se compone de una serie de elementos que situamos juntos para conseguir una representación. Estos elementos siguen una estructura que se asemeja a un árbol: elementos raíz, elemento hoja, etc.

- Existe un elemento que está por encima de todos ellos, llamado "**root element**" y que se corresponde con <html>.

<html xmlns=http://www.w3.org/1999/xhtml lang="es" xml:lang="es">

TESLA TECHNOLOGIES

Elemento raíz o root

- Esta declaración es perfectamente válida en **HTML5**. No obstante, podemos transformarla un poco ya que, como en el caso del doctype, **HTML5** también nos **simplifica** esto.

- Ello es debido a **dos motivos**:
- El primero es que el **atributo xmls** no tiene sentido en este caso, ya que este atributo lo que nos dice es que los elementos de la página pertenecen al espacio de nombres de **XHTML**, pero en **HTML5** todos los elementos pertenecen ya implícitamente a este espacio.

<html lang="es" xml:lang="es">

Elemento raíz o root

- El segundo motivo es debido a que nos basta con dejar solamente el atributo de idioma, ya que la etiqueta con el **XML** es un vestigio de **XHTML**. Por lo tanto, nuestro elemento raíz quedaría de la siguiente forma:

<html lang="es">

TESLA TECHNOLOGIES

La etiqueta <head>

- Normalmente, el primer hijo de la **etiqueta html** es la etiqueta **head**. Esta etiqueta suele contener una **serie de metadatos**, que es información sobre la página.

- Aquí podemos destacar que debemos definir siempre la codificación de caracteres que vamos a usar en nuestra página.

<META HTTP-EQUIV=Content-Type CONTENT=text/html; charset=ISO-8859-1>

TESLA
TECHNOLOGIES

Introducción: Definiendo relaciones entre links en HTML5

- Los tradicionales links (usando a href=) son la forma más simple y clásica de apuntar a otra página. El **atributo rel** es el encargador de categorizar dichos links.
- En **HTML5**, estas relaciones se dividen en dos grandes grupos:
  - **Links a recursos externos**, que son links a una serie de recursos que se pueden usar en el documento actual.
  - **Hipervínculos**, que son links a otros documentos.

TESLA
TECHNOLOGIES

Introducción: Definiendo relaciones entre links en HTML5

- Ahora os dejamos un pequeño listado con los principales links:

· rel = “stylesheet”
Se suele utilizar para apuntar a una serie de reglas css (las clásicas hojas de estilos).

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

· rel = “alternate”
En este caso suele usarse de una forma conjunta con un RSS o un Atom Media.

<link rel="alternate" type="application/atom+xml" title="Feed de prueba" href="/feed/" />

TESLA
TECHNOLOGIES

Introducción: Definiendo relaciones entre links en HTML5

· rel="archives"
Se utiliza para indicar que el documento al cual se referencia describe una colección de registros, documentos o cualquier otro material que pueda tener interés histórico.

· rel=”autor”
Se suele utilizar para realizar links hacia información del autor de la página.

· rel=”start”, rel=”prev”, rel=“end” y rel=”next”
Se suelen emplear para definir relaciones en páginas que pertenecen a una serie, como los capítulos de un libro.
HTML 5 incluye una más que es **rel=”first”**, para indicar el comienzo en lugar de start.

TESLA
TECHNOLOGIES

Introducción: Definiendo relaciones entre links en HTML5

· **rel=”icon”**
Como su nombre indica, es para hacer referencia al link de un icono.

<link rel="shortcut icon" href="favicon.ico">

· **rel=”license”**
Sirve para indicar los términos de la licencia a los cuales se acoge la página.

TESLA
TECHNOLOGIES

Nuevos elementos semánticos entre links en HTML5

- **<section>** El elemento section representa una sección genérica de un documento o aplicación. Una sección, en este contexto, es una agrupación temática de contenido, típicamente con un encabezado.

- **<nav>** El elemento nav representa una sección de una página que enlaza a otras páginas o a partes dentro de la página: una sección con enlaces de navegación.

- **<article>** El elemento artículo representa una composición contenida en un documento, página, aplicación o sitio concebida para ser distribuida independientemente o ser reutilizada.

Nuevos elementos semánticos entre links en HTML5

- **<aside>** El elemento aside representa una sección de una página que consta de contenido relacionado tangencialmente; es decir, aquello que podría ser considerado de forma separada respecto del contenido original.

- **<hgroup>** El elemento hgroup representa el encabezando de una sección. Consiste en un conjunto de elementos de tipo encabezado, que van desde h1 hasta h6.

- **<header>** El elemento header representa un grupo de introducción o ayudas a la navegación. Contiene normalmente las cabeceras de las secciones, pudiéndose usar para tablas de contenidos, formularios de búsqueda o cualquier logo relevante.

TESLA
TECHNOLOGIES

Nuevos elementos semánticos entre links en HTML5

- **<footer>** El elemento footer representa el elemento más lejano a la raíz del documento. Un footer contiene información sobre su sección como quién la escribió, enlaces a otros documentos relacionados, datos de copyright o información de licencias. Los footers no tienen por qué aparecer necesariamente al final de una sección, pero es donde suelen hacerlo.

- **<time>** El elemento time representa un tiempo en un reloj de 24 horas o una fecha precisa en el formato de calendario gregoriano.

- **<mark>** El elemento mark representa texto en un documento marcado o destacado con propósitos de referencia.

Cabeceras <header>

Tradicionalmente, para definir la cabecera de nuestra web solíamos recurrir a un código como el siguiente:

```
<div id="header">
        <h1>Titulo</h1>
        <p class="tagline">Curso de HTML5</p>
</div>

<div class="entrada">
        <h2>Primera clase</h2>
</div>
```

TESLA
TECHNOLOGIES

Cabeceras <header>

El código anterior es perfectamente válido en **HTML5**, pero este estándar nos ofrece algunos elementos semánticos propios para las cabeceras y las secciones de la web:

```
<header>
        <h1>Titulo</h1>
        <p class="tagline">Curso de HTML5</p>
</header>
```

TESLA
TECHNOLOGIES

Artículos <articles>

- El elemento article deberá utilizarse cuando se tenga un contenido que sea independiente, distribuible o reusable, es decir, cuando al mover el contenido a otro lugar de la página, a otra página distinta o se quite, no afecte al resto de la página.

Por ejemplo, una entrada en un blog, una noticia en un periódico o un comentario en un foro.

TESLA
TECHNOLOGIES

Fechas y horas

Entre las nuevas funcionalidades que incorpora **HTML 5,** también existe una etiqueta para dotar de una mayor información semántica a la fecha de publicación de un articulo, para lo cual se ha introducido la etiqueta **<time>**.

<time datetime="2012-02-10" pubdate>10 Febrero de 2012</time>

TESLA
TECHNOLOGIES

Fechas y horas

- Como podemos ver, esta etiqueta incluye tres partes:
  - Una fecha (timestamp) pensada para las máquinas.
  - Texto fecha para humanos.
  - Un parámetro adicional que es pubdate. Este atributo tiene dos posibles significados en función de dónde nos encontremos: si estamos en un artículo, indicará la fecha de publicación del artículo, mientras que si está en otra parte del documento, hará referencia a la fecha de publicación de todo el documento.

Fechas y horas

- Asimismo, también podemos incluir la hora en esta etiqueta quedándonos de la siguiente manera:

<time datetime="2012-02-10 T16:00:00" pubdate>10 Febrero de 2012</time>

- La T indica que el formato de la hora es de 24 horas.

TESLA
TECHNOLOGIES

Pie de página <footer>

- En el caso del pie de página nos sucede exactamente lo mismo que con los menús de navegación. Si queremos dotarlos de una mayor carga semántica, debemos marcar nuestros pies con la nueva etiqueta **<footer>**.

- Sobre qué contenido debemos poner en esta etiqueta, la especificación de **HTML5** nos dice que un pie típicamente contiene información relativa al autor, links a documentos relacionados, información de copyright, etc.; es decir, *lo que ya se venía ubicando hasta ahora usualmente en los pies de página.*

TESLA
TECHNOLOGIES

HTML5: Formularios

TESLA
TECHNOLOGIES

Formularios HTML5

- **HTML5** da una vuelta de tuerca a todo lo relacionado con los formularios. Ahora pasan a incluirse nuevas etiquetas para diferentes controles en función de la utilidad que le podamos dar, lo cual nos va a dotar de una mayor flexibilidad.

- Se incluye además una validación de los datos nativa, simplificándose muchísimo la tarea a la hora de realizar formularios.

Placeholder Text

- Una de las primeras novedades que nos vamos a encontrar en los formularios **HTML5** es la posibilidad de introducir lo que se llama un placeholder text a nuestros inputs.

- Un **placeholder text** es el típico texto que aparece mientras el control no tiene el foco encima; en cuanto lo tiene, ese texto desaparece.

```
<form>
        <input name="query" placeholder="Busca en esta web">
        <input type="submit" value="Buscar">
</form>
```

Campos con Autofocus

- La mayor parte de los sitios web incluyen un cuadro de texto sobre el cual se le aplica el foco de forma automática, como es el caso de **Google**. No obstante, eso presenta algunos problemas.

- Para resolverlos, **HTML5** nos incluye un atributo **autofocus**. Este atributo hace lo que se le supone, en cuanto se carga la página nos lleva el foco de forma automática al control.

```
<form>
        <input name="txtbuscar" autofocus>
        <input type="submit" value="Buscar">
</form>
```

TESLA
TECHNOLOGIES

Direcciones de email

- Este es el primer control nuevo que vamos a ver en HTML 5: la especificación define un tipo de input para direcciones de email.

- En caso de que el navegador no reconozca la etiqueta, lo que sucederá es que el navegador la ignorará y tratará el tipo email como un campo de texto normal.

```
<form>
        <input type="email">
        <input type="submit" value="Enviar">
</form>
```

Direcciones Web

- Las direcciones web o URL son otro tipo de texto especial. Como todos sabemos, la sintaxis de una URL está contenida dentro del estándar de internet, muestra el protocolo, etc.

- De la misma forma que con las direcciones de email, si el navegador no la soporta, la ignorará dejándola como un simple campo de texto.

```
<form>
        <input type="url">
        <input type="submit" value="Enviar">
</form>
```

Números y Spinboxes

- Muchas veces queremos incluir números en nuestros formularios que a su vez queremos que sólo se puedan modificar mediante el uso de un spinbox:

```
<input type="number" min="0" max="10" step="2" value="6">
```

- Admite varios parámetros:

  - **Min**, el número mínimo aceptable.
  - **Max**, el número máximo.
  - **Step**, define los números aceptables en el rango.
  - **Value**, es el valor por defecto.

Números y Sliders

- Como hemos visto en el punto anterior, tal y como a veces nos interesa tener números limitados por un par de botones, otras veces nos interesa limitar los números usando barras de desplazamientos. Para ello, la especificación de **HTML5** nos incluye un nuevo tipo que es **range**.

```
<input type="range" min="0" max="10" step="2" value="6">
```

TESLA
TECHNOLOGIES

Date Pickers

- **HTML5** define una forma de incluir un control nativo para introducir fechas sin necesidad de ningún script externo. En realidad no se trata de un único control sino que son 6 estructurados en función de lo que necesitemos:

- **Data, Month, Week,Time, Datetime, Datetime-local.**

```
<form>
        <input type="date">
</form>
```

Cajas de Búsqueda

- En la nueva especificación se ha incluido el tipo **search** para poder diferenciar los inputs de texto normales y los de búsqueda.

- Si no reconoce este tipo, el navegador lo único que hace es tratarlo como una caja de texto normal.

```
<form>
        <input name="txtbusqueda" type="search">
        <input type="submit" value="Encontrar">
</form>
```

Color Pickers

- Entre las novedades de **HTML5** también se incluye un campo de texto que nos permite seleccionar un color en hexadecimal, para ello se ha introducido el tipo color.

- El problema es que esta es la etiquetas que menos soporte tiene actualmente.

```
<form>
        <input type="color">
</form>
```

TESLA
TECHNOLOGIES

HTML5: Audio/Vídeo

Etiqueta vídeo

- Todos hemos insertado algún vídeo en nuestra web, y gracias a servicios como youtube el proceso se ha visto muy simplificado. Pero no existía ninguna forma estándar de incluir un vídeo almacenado en nuestro servidor, siempre teníamos que recurrir a plugins.

- **HTML5** define un nuevo estándar para incrustar vídeos en las páginas webs, usando para ello la **etiqueta <video>**.

- **El soporte de esta etiqueta aun está siendo implementado**, así que se espera que con el tiempo consiga una mayor compatibilidad y más funcionalidades.

TESLA
TECHNOLOGIES

Contenedores de vídeos

- Muchas veces tendemos a pensar que los archivos AVI o MP4 son los propios vídeos, pero eso no es cierto al 100% sino que solamente son contenedores de ese vídeo.

- Los vídeos están formados internamente, dicho de una manera muy sencilla, por una serie de pistas tanto de audio como de vídeo y una serie de metadatos.

TESLA
TECHNOLOGIES

Contenedores de vídeos

- **MPEG-4**

Normalmente su extensión es .mp4 o .m4v. El MPEG-4 contenedor está basado en QuickTime (.mov).

- **FlashVideo**

Aparece normalmente con una extensión .flv Hasta hace poco era el único formato que admitía Flash, pero en las últimas versiones nos permite usar MPEG-4.

- **Ogg**

Normalmente con extensión .ogv Es un estándar abierto sin el estorbo de patentes.
Tenemos soporte nativo en Firefox 3.5, Chrome 4 y Ópera 10.5.
Ogg para vídeo se llama **Theora** y Ogg para audio se llama **Vorbis**.

Contenedores de vídeos

- **WebM**

Tiene una extensión .webm. Es un formato de contenedor nuevo técnicamente muy similar a otro formato llamado Matroska. WebM estuvo anunciado en Google I/O 2010. Está diseñado para ser utilizado exclusivamente con el VP8 vídeo códec y el Vorbis audio códec. Tiene soporte nativo en Chromium, Chrome, y Ópera. En las últimas versiones de Flash también tendrá soporte.

- **Vídeo de audio Interleave (AVI)**

Normalmente aparece con una extensión .avi. Es un formato de contenedor que fue inventado por Microsoft en un tiempo en que todo era más sencillo, cuándo el hecho de que los ordenadores reprodujesen vídeo era considerado bastante asombroso.

TESLA TECHNOLOGIES

¿Qué soporte tiene?

- **Cómo está el soporte a fecha de hoy**:

  • **Mozilla Firefox** (3.5 y posteriores): apoya Theora vídeo y Vorbis audio en un contenedor Ogg.

  • **Opera** (10.5 y posteriores) apoya Theora vídeo y Vorbis audio en un contenedor Ogg.

  • **Google Chrome** (3.0 y posteriores) apoya Theora vídeo y Vorbis audio en un contenedor Ogg. También apoya H.264 vídeo (todos los perfiles) y AAC audio (todos los perfiles) en un contenedor MP4.

TESLA
TECHNOLOGIES

¿Qué soporte tiene?

• **Google Chrome** y versiones experimentales de **Opera** soportan **VP8** vídeo y **Vorbis** audio en un contenedor **WebM**.

• **Safari** en **Mac** y **Windows PCs** (3.0 y posteriores) soporta todo aquello que sea soportado por **QuickTime.** Es una lista larga, pero no incluye Theora vídeo, Vorbis audio, o el contenedor Ogg. Aun así, QuickTime apoya H.264 vídeo (perfil principal) y AAC audio en un contenedor MP4.

TESLA TECHNOLOGIES

¿Qué soporte tiene?

- Dispositivos móviles como **iPhone** y **Android** soportan **H.264** vídeo y **AAC** audio en un contenedor **MP4**.

- **Adobe Flash** (9 y posteriores) soporta **H.264** vídeo y **AAC** audio en un contenedor **MP4**.

- **IE9** soporta **H.264** vídeo y **AAC** audio en un contenedor **MP4**.

- **IE8 y anteriores** no soportan vídeo.

Etiqueta audio

- Se usa de una forma idéntica a la etiqueta vídeo.

- Sucede lo mismo con los códecs, los formatos y los controles. Para reproducir, resulta suficiente esto:

```
<audio src="prueba.mp3" autoplay></audio>
```

- Pero si queremos asegurarnos compatibilidad, necesitaremos lo siguiente:

```
<audio controls>
    <source src="prueba.ogg">
    <source src="prueba.mp3">
</audio>
```

TESLA
TECHNOLOGIES

HTML5: Geolocalización

Geolocalización

- La **geolocalización**, como su nombre indica, es situarnos **en qué punto del mundo estamos**. Existen múltiples técnicas de detección, posicionamiento por GPS, triangulación de señales de móvil, localización por wifi, etc.

- El API de geolocalización solo comparte nuestra ubicación con sitios web a parir del momento que le damos permiso.

- La latitud y longitud están disponible a través de **Javascript** en la página. A partir de aquí, podemos enviar la información al servidor remoto y podremos, por ejemplo, mostrar nuestra ubicación en un mapa.

TESLA
TECHNOLOGIES

Geolocalización

➢ Tenemos un objeto denominado position, el cual tiene las siguientes propiedades:

- **coords.latitude**
- **coords.longitude**
- **coords.altitude**
- **coords.accuracy**
- **coords.altitudeAccuracy**
- **coords.heading (grados desde el norte)**
- **coords.speed (metros por egundo)**
- **timestamp**

➢De estas propiedades sólo podemos garantizar que existan 3: latitud, longitud y precisión. El resto dependen del sistema que esté utilizando el agente.

TESLA
TECHNOLOGIES

HTML5: Introducción a Canvas

TESLA
TECHNOLOGIES

Canvas

➢En **HTML5** se define **Canvas** como **“un bitmap, que puede ser usado para renderizar gráficos, juegos, o otros elementos visuales en directo**”.

Básicamente Canvas es un rectángulo en nuestra página sobre el que podemos dibujar cualquier cosa gracias a Javascript.

➢La etiqueta de Canvas es muy sencilla, viéndose reducida, por ejemplo, a lo siguiente:

```
<canvas width="300" height="225"></canvas>
```

Objetos básicos

- Una vez hemos hecho esto, vamos a explicar paso a paso.
- En la primera línea vemos lo siguiente:

var ejemplo_canvas = document.getElementById("prueba");

- Lo único que hace es seleccionarlos el objeto canvas de la escena y almacenarlo en la variable ejemplo_canvas. Hasta aquí es idéntico a trabajar con capas normales y javascript.
- En la segunda línea ya empieza el tema interesante:

var ejemplo _context = ejemplo _canvas.getContext("2d");

Objetos básicos

➢Algunos ejemplos de estas propiedades (sólo hablando de dibujar rectángulos) son:

•La propiedad **FillStyle**: el relleno. Puede ser un color CSS, un patrón o un degradado. El fillStyle por defecto es de color negro sólido, pero podemos cambiarlo a nuestro gusto.

•La propiedad **fillRect** (x, y, ancho, alto) dibuja un rectángulo relleno con el estilo de relleno actual.

TESLA
TECHNOLOGIES

Objetos básicos

•La propiedad **strokeStyle** es como **fillStyle**, pudiendo ser de un color CSS, un patrón o un gradiente.

•La propiedad **strokeRect** (x, y, ancho, alto) dibuja un rectángulo con el estilo de trazo actual. **strokeRect** no se llena en el centro, sino que sólo dibuja los bordes.

•La propiedad **clearRect** (x, y, ancho, alto) borra los píxeles en el rectángulo especificado.

Caminos en Canvas

➢**Canvas** nos permite dibujar caminos, es decir una serie de líneas que unan dos o mas vértices. Hacer esto es bastante sencillo, se utilizan los dos siguientes métodos:

- **moveTo (x, y)**: se mueve el lápiz hacia el punto de partida.
- **lineTo (x, y)**: dibuja una línea hasta el punto final especificado.

➢Cuanto más se llama a **moveTo ()** y **lineTo ()**, mayor será el camino.

➢Podemos llamarlos todas las veces que queramos, pero no se dibujará nada en pantalla hasta que llamemos a alguno de los métodos de pintar.

TESLA
TECHNOLOGIES

Caminos en Canvas

- Si, por ejemplo, intentamos crear una cuadrícula podemos hacer lo siguiente:

```
for (var x = 0.5; x < 500; x += 10)
 {
        context.moveTo(x, 0);
        context.lineTo(x, 375);
}
for (var y = 0.5; y < 375; y += 10)
{
        context.moveTo(0, y);
        context.lineTo(500, y);
}
```

Caminos en Canvas

- Como hemos visto, esto no resulta suficiente sino que deberíamos llamar a algunos de los métodos que pintan. Para ello, en este caso usaremos el método stroke(), que básicamente recogerá todos los caminos que hemos definido y nos los pintará.

```
context.strokeStyle = "#eee";
context.stroke();
```

TESLA
TECHNOLOGIES

Texto en Canvas

- Además de pintar líneas y formas en C**anvas**, podemos introducir textos.

- Tenemos que tener presente que esto no es **HTML**; por consiguiente, no funciona ninguna de las propiedades a las que estábamos acostumbrados en CSS: no hay márgenes, paddings, separación de palabras, etc.

TESLA
TECHNOLOGIES

Gradientes en Canvas

➢**Además de colores sólidos podemos hacer degradados**, los cuales serán de dos tipos:

- Lineales.
- Radiales.

TESLA
TECHNOLOGIES

Imágenes en Canvas

- En canvas también es posible pintar imágenes.

- Para pintar una imagen se utiliza el método **drawImage**
  - Este método pertenece al contexto, al igual que los métodos **stroke** y **fill**.

- El primer parámetro de este método siempre es una referencia a un mapa de bits.
  - Podemos usar **javaScrip** para recuperar dicha referencia

```
Var  img = new Image();
Img.src = 'imagen.png';
```

TESLA
TECHNOLOGIES

Imágenes en Canvas

- La forma más básica del método **drawImage** tiene tres parámetros. La fuente y la posición x e y donde comenzar a pintar.

```
contexto.drawImage(img,100,100);
```

- En una forma un poco más completa se añaden dos parámetros más para definir un rectángulo, donde se pintara la imagen.

```
contexto.drawImage(img,100,100, 50,50);
```

- En la forma más completa se incorporan 4 nuevos parámetros a continuación de la fuente, para definir un área rectangular de recorte sobre la imagen original.

```
contexto.drawImage(img, 20,30, 50,60, 100,100, 50,50);
```

TESLA
TECHNOLOGIES

Frameworks: three.js

- Es un motor de renderizado en 3D.
- Acepta funcionalidades de WebGL.
- Podemos conseguir cosas como con 30 líneas montar una escena en 3D.
- Tiene múltiples funcionalidades nativas:
  - Calculo de colisiones
  - Manejo de sprites
  - Tipos de cámaras

http://mrdoob.github.com/three.js/

TESLA
TECHNOLOGIES

Frameworks: box2d

- Implementa un motor físico en dos dimensiones.
- Aplicaciones muy usadas lo utilizan, como Angry Birds.
- Box2d funciona con física de cuerpos rígidos, todo en Box2D es un cuerpo con una forma asociada
- Es multiplataforma. Esta escrita en C++ y existen versiones para Flash y Javascript.

http://box2d.org/

TESLA
TECHNOLOGIES

SVG vs. Canvas vs. Flash

- Actualmente tenemos 3 opciones mayoritarias para gráficos y animaciones: **Flash**, **Canvas** y **SVG**.

- ¿Cuál es la mejor para nosotros? Teniendo en cuenta que esto es un curso de **HTML5 + Canvas**, deberíamos responder que canvas siempre.

- Pero todo depende de lo que necesitemos, ya que cada uno tiene sus ventajas e inconvenientes.

TESLA
TECHNOLOGIES

SVG vs. Canvas vs. Flash

- Flash ahora mismo tiene el mejor rendimiento.

- Canvas es una buena alternativa a **Flash**, no tan rápida pero sí lo suficiente para ser una opción viable, siempre y cuando el navegador lo soporte.

- La implementación de animaciones en **SVG** es considerablemente más lenta que **Flash** o **Canvas** (**HTML5**).

- Canvas nos proporciona un futuro, ya que los dispositivos móviles lo soportan y **Flash** dejará en breve de tener soporte para dichos dispositivos.

TESLA
TECHNOLOGIES

SVG vs. Canvas vs. Flash

- La mejora de los motores de javascript de los navegadores hace que cada vez **Canvas** sea mas rápido, reduciendo su desventaja con **Flash** día a día.

- **SVG** puede resultar más sencillo para las animaciones, ya que la programación de las mismas es más simple.

- No obstante, **SVG** pierde rápidamente rendimiento a medida que nuestros documentos se complican; en **Canvas** y **Flash** eso no sucede.

- **SVG** es mejor para realizar imágenes vectoriales, mientras que canvas es superior en el manejo de gráficos y nos da mucha más versatilidad.

TESLA
TECHNOLOGIES

Esteban Vázquez
esteban.vazquez@teslatechnologies.com

Antonio Varela
antonio.varela@teslatechnologies.com

Web: http://teslatechnologies.es
Twitter: @tesla_tec