



Componentes multimedia en HTML5

Índice



1 Componentes multimedia en HTML5	3
1.1 Compatibilidad entre navegadores	6
1.2 Consulta de tablas de compatibilidad	7
1.3 Usar librerías	8

1. Componentes multimedia en HTML5

Además de los cambios que hemos visto anteriormente, se han añadido a HTML5 elementos que permiten contenido multimedia sin necesidad de plugins externos como Adobe Flash, Microsoft Silverlight o los Applets de Java.

A continuación se detallan algunas de las nuevas etiquetas de HTML5 creadas como componentes multimedia:

- **audio:** permite la carga de ficheros de audio. Cada navegador puede soportar los formatos que se crea oportunos, es decir, el estándar

HTML5 no regula que estándares de audio son permitidos. Los más habituales son mp3, ogg y wav.

La etiqueta *audio* cuenta con los siguientes atributos:

- **src:** URL donde se encuentra el fichero de audio que queremos que se reproduzca.
- **autoplay:** indica si el archivo se reproduce automáticamente al cargar la página.

- **loop:** el fichero se reproduce en bucle.
- **controls:** la presencia de este atributo indica que se debe mostrar el interfaz visual (reproducir, pausa, parar, volumen, etc.)
- **autobuffer:** su presencia indica que el fichero debe descargarse completamente antes de comenzar a ejecutarse.



VISUALIZACIÓN DE LA ETIQUETA AUDIO

Como no hay un formato de audio estándar adoptado por todos los navegadores, podemos agregar distintas fuentes eliminando el atributo `src` y añadiendo como contenido una etiqueta `source` por cada formato de audio que tengamos.

```
<audio autoplay controls loop>
  <source src="sonido.ogg">
  <source src="sonido.mp3">
</audio>
```

ETIQUETA AUDIO CON MÚLTIPLES FUENTES

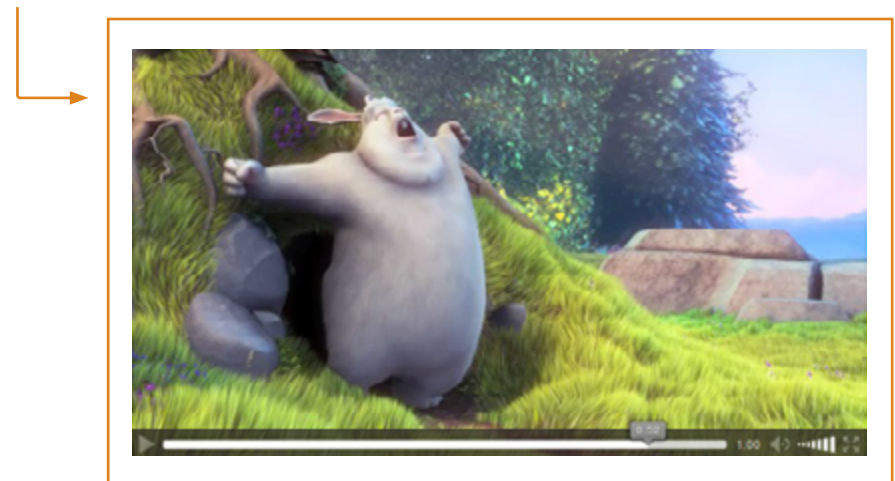
El elemento `source` indica, a través de la propiedad `src`, la ubicación del archivo de audio. El orden en el que añadimos estas fuentes es importante. Primero el navegador busca la primera fuente y verifica que puede reproducirla en ese formato. En caso afirmativo, reproduce ese fichero y, en caso negativo, pasa a la siguiente fuente.

- **video:** este elemento nos permite mostrar un video sin necesidad de un plugin externo. En este momento, los navegadores permiten mostrar una cantidad de formatos de video muy limitados. Atributos:
- **src:** URL donde se encuentra el fichero de video que queremos reproducir.
- **controls:** visualiza los controles de reproducción del video (reproducir, pausa, parada, volumen, etc.).

- **autoplay:** el video se reproduce inmediatamente.
- **width:** anchura del video en pixeles
- **height:** altura del video en pixeles.

Al igual que con la etiqueta `video`, podemos utilizar múltiples fuentes para un mismo video, suprimiendo el atributo `src` y añadiendo tantos elementos `source` como fuentes tengamos. El elemento `source` también acepta el atributo `type`.

```
<video controls>
  <source src="http://www.quicksmode.org/html5/
  videos/big_buck_bunny.mp4" type="video/mp4">
</video>
```



VIDEO

- **canvas**: con esta etiqueta podemos crear en un documento un área en la que es posible dibujar mediante JavaScript. El objetivo de este componente es generar gráficos en el cliente.

Para poder hacer un uso correcto de este componente, debemos manejar el lenguaje JavaScript que veremos más adelante. De momento nos limitaremos a ver ejemplos para ver cómo se comporta. Atributos de la etiqueta:

- **width**: anchura en pixeles del canvas.
- **height**: altura en pixeles del canvas.

Es conveniente definir el atributo *id* para poder referenciar cómodamente el componente desde JavaScript.

El elemento *canvas* nos permite crear gráficos en 2d y 3d. A continuación veremos un ejemplo de *canvas*. Veremos que es un ejemplo más complejo que los anteriores. El ejemplo cuenta con los siguientes elementos:

- 1. Elemento canvas.** Se declara en donde queramos tener el lienzo, y le damos un tamaño. También hemos añadido el atributo *id*.
- 2. Código Javascript.** Vemos que se declara la función *window.onload*, cuyo código se ejecuta cuando la página ha cargado por completo en el navegador.

Dentro de dicha función, se recupera el elemento canvas de HTML5 mediante la función *getElementById* del objeto *document* (mas adelante entraremos en detalles) y se guarda en la variable "lienzo".

Después, creamos el contexto del lienzo en el que vamos a dibujar. En este caso, esta tarea la realiza la llamada a la función *crearContexto2d* que crea un contexto para gráficos en 2d se guarda en la variable *contexto2d*.

Por último, se llama a la función *pintar rectángulo* que recibe como parámetro el contexto, y realiza el dibujo sobre él.

- 3. Representación del *canvas*** en el navegador una vez pintado nuestro rectángulo.

⏮
⏪
⏩
⏭

```
<canvas id="Lienzo1" width="300" height="200"
class="borde-gris"></canvas>
```


+

⏮
⏪
⏩
⏭

```
<script>
  var crearContexto2d = function(lienzo) {
    if(!lienzo) return null;
    if(!lienzo.getContext) return null;

    return lienzo.getContext('2d');
  }
  var pintarRectangulo = function(contexto2d) {
    if(!contexto2d) return;

    contexto2d.fillStyle = 'rgb(200,0,0)';
    contexto2d.fillRect = (10, 10, 100, 100);
  }
  window.onload = function() {
    var lienzo = document.getElementById('lienzo1');
    var contexto2d = crearContexto2d(lienzo1);
    pintarRectangulo(contexto2d);
  }
</script>
```



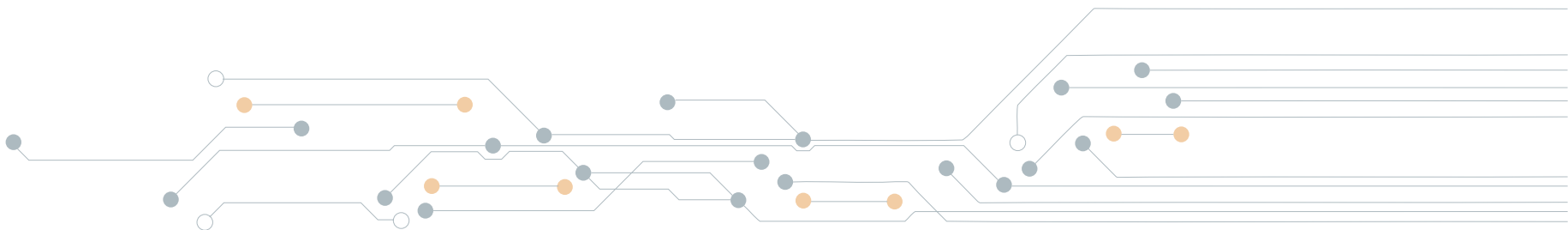
1.1 | Compatibilidad entre navegadores

El motor de renderizado es el encargado de leer los datos recibidos de una web (HTML, XML, CSS, XSL...) y representarlos en pantalla. Cada navegador tiene su motor de renderizado, como Webkit, el motor de Safari, Opera y hasta hace poco de Chrome (ahora utiliza Blink); Gecko, el motor de Firefox; o Trident, el motor de Internet Explorer.

Pues bien, aunque en gran medida todos cumplen los estándares y representan las etiquetas de la misma forma, hay elementos que no todos interpretan igual, o incluso debido a su novedad, algunos no llegan ni tan siquiera a soportar. Por ejemplo, hasta hace poco Trident, el motor de Internet Explorer, era incapaz de interpretar la gran mayoría de etiquetas HTML5.

Si quieres utilizar por ejemplo, el elemento canvas (el cual está subiendo en popularidad como la espuma), es posible que tengas más de un problema al probar con los distintos navegadores.

Pero si las etiquetas pueden ser un problema, los estilos CSS lo son aun más. Cosas como las sombras en los textos funcionan en algunos navegadores si y en otros no, el tratamiento de los bordes no es igual en Firefox que en Explorer, lo que puede causar que se descuadre algún elemento, hacer bordes redondeados es diferente según el navegador, e incluso algo tan básico como los márgenes (margin) difiere dependiendo del navegador y su versión.

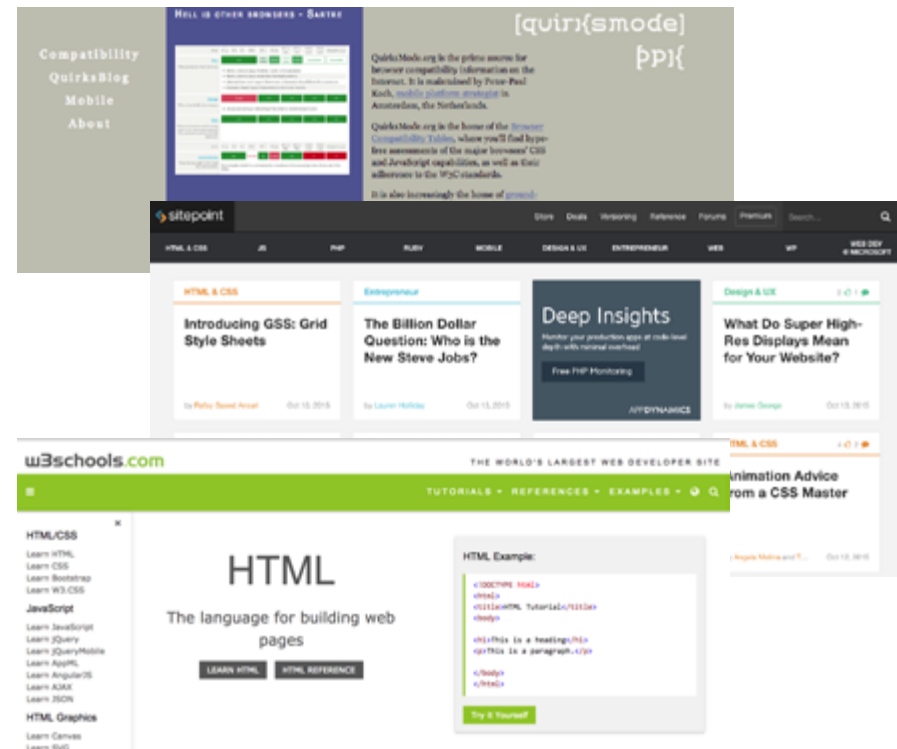


1.2 | Consulta de tablas de compatibilidad

Una de las primeras cosas a tener en cuenta son las tablas de compatibilidades, o lo que es lo mismo, tablas que te indican si el recurso que quieres utilizar es compatible con todos los navegadores o no. Podemos consultar las siguientes:

- **quirksmode.org:** sin duda la pagina más completa sobre el tema. Hay disponibles tablas de compatibilidad tanto de CSS, el DOM y test de elementos HTML5. A día de hoy está completamente actualizado con las últimas versiones de los navegadores (como IE11).
- **sitepoint:** sitepoint tiene un apartado llamado reference en el que no sólo encontraremos tablas de compatibilidades sino también explicaciones sobre cada atributo, posibles valores, etc. El gran problema es que no esta tan actualizado como quirksmode y hay tablas algo antiguas.
- **w3schools:** aunque w3schools es más útil para conocer tags, atributos y otros elementos del estándar, junto a cada uno de ellos nos encontraremos los iconos con los navegadores compatibles.

Por desgracia únicamente tiene en cuenta las últimas versiones de los mismos, por lo que no nos ayuda a hacer compatible nuestra página con navegadores antiguos.



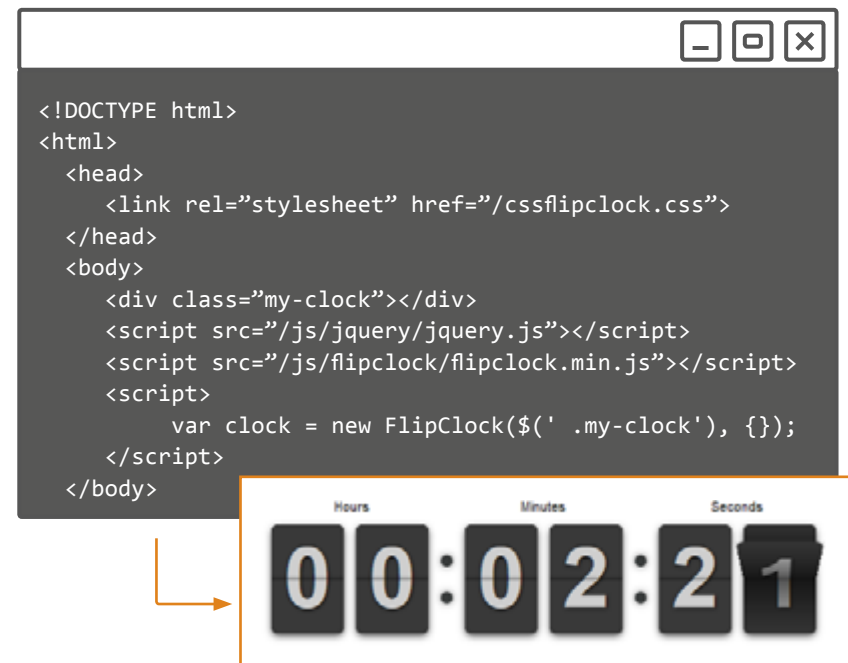
1.3 | Usar librerías

Puedes utilizar librerías que hacen lo que quieres. Se trata de un conjunto de elementos de código reutilizables que no conforman una aplicación en sí misma, si no que están orientadas a ser utilizadas en otros programas. Suelen estar enfocadas a un campo concreto (por ejemplo, librerías para manejar fechas, operaciones matemáticas, o en este caso, a uso de componentes HTML).

Usar librerías en el desarrollo de páginas web es una forma de evitarte los problemas de compatibilidad entre navegadores. Por lo general estarán bien probadas y los problemas serán mucho menores.

Por ejemplo, si queremos incluir un reloj en nuestra página, podemos acudir a bastantes librerías existentes. Por ejemplo, la librería FlipClock nos permite incluir un reloj, temporizador o cronometro con efecto moneda. A su vez, FlipClock usa otra librería llamada JQuery, por lo que también la necesitaremos para que funcione FlipClock.

Para usarla, necesitaríamos importar las librería JavaScript y sus estilos CSS y usarla tal y como dice su documentación. A continuación vemos como se usa:



Telefonica

EDUCACIÓN DIGITAL