

HTML5

1. Definición de HTML5

HTML5 es una combinación de nuevas etiquetas de markup (lenguaje) HTML, propiedades CSS3, JavaScript y algunas tecnologías complementarias de apoyo, pero que técnicamente son independientes de la propia especificación HTML5. Por ello vamos a distinguir entre la especificación HTML5 en sí y la *familia HTML5*.

Podemos definir la especificación HTML5 como nuevos elementos de markup o sintaxis, utilizados por los diseñadores para crear páginas web junto con las etiquetas utilizadas a día de hoy. Muchos de estos nuevos elementos ya son conocidos para los diseñadores que trabajan con las etiquetas HTML tradicionales, como `<p>`, `` o `<div>`. Estas nuevas etiquetas suponen para desarrolladores y diseñadores, unas herramientas más avanzadas y se traducen en mejores experiencias para el usuario final.

La familia HTML5 incluye las nuevas etiquetas y tecnologías como CSS3, Geolocalización, Almacenamiento Web (Web Storage), Web Workers y Web Sockets entre otras. Todas ellas suponen una actualización de gran potencia al conjunto de herramientas ya existente, y con ellas se pueden crear páginas web más sofisticadas y útiles.

Los nuevos navegadores incorporan funcionalidades para responder a las expectativas de los consumidores y también como fruto de la evolución natural de la propia tecnología. A medida que las aplicaciones web van ganando en capacidad de respuesta y velocidad y son capaces de resolver tareas complejas como la edición de imágenes, representación de mapas, hojas de cálculo o vídeos, los usuarios ya exigen este mismo nivel de rendimiento en todas las aplicaciones de la web. Existen limitaciones con las capacidades de los lenguajes de programación actuales y no todas las funcionalidades se pueden implementar e incorporar de manera sencilla. HTML5 aporta nuevas funcionalidades y herramientas con el fin de conseguir que los sitios web sean más interesantes, atractivos y útiles.

2. El lenguaje HTML5. Nuevas etiquetas.

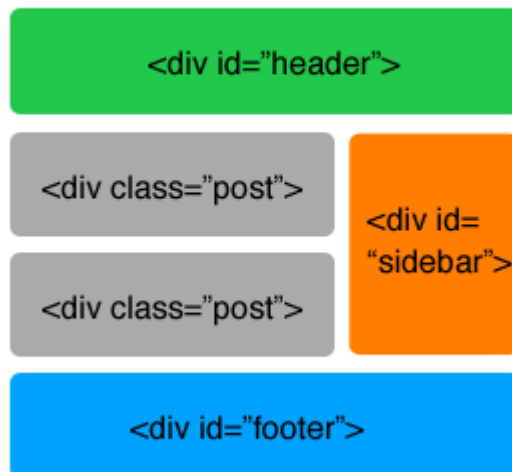
El lenguaje (o "markup") HTML5 incorpora algunas etiquetas nuevas pensadas para hacer que la estructura de la página web sea más lógica y funcional. Antes de HTML5, la estructura de una página dependía fuertemente de las etiquetas <div>, generalmente asociadas a una clase CSS o un ID. Por ejemplo, en HTML 4.0 es una práctica comúnmente aceptada definir la cabecera de una página web de esta forma:

```
<div id="header" > Esta es la cabecera </div>
```

En este caso, el código destacado en color rojo es el ID de CSS que sirve para definir la anchura y altura de la cabecera así como su color de fondo. En el código CSS podríamos tener algo así:

```
#header {  
    width:960px;  
    height:100px;  
    background-color:gray;  
}
```

Antigua estructura:



HTML5 dispone de unos cuantos elementos como:

<header> Define una región encabezado o header de una página o de una sección.

<footer> Define una región pie de página o footer de una página o de una sección.

<nav> Define una región de navegación de una página o de una sección.

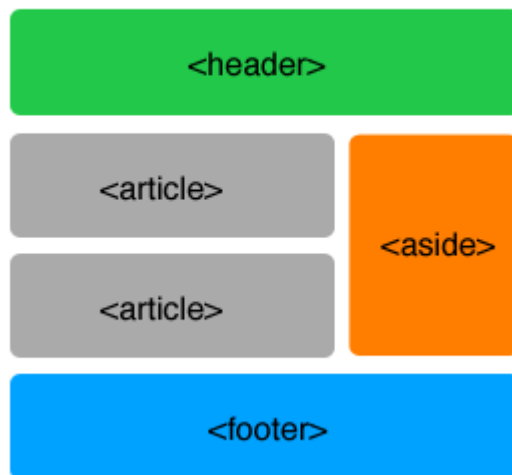
<section> Define una región lógica de una página o un grupo de contenido.

<article> Define un artículo o una pieza completa de contenido.

<aside> Define un contenido secundario.

Estos nombres se basan en los que se vienen utilizando habitualmente para distinguir secciones dentro de las páginas web que vemos a diario (div id="footer", div id="nav", etc.). El objetivo de los nuevos elementos HTML5 no es otro que evitar una excesiva dependencia de las etiquetas <div> y sustituirlas por una estructura de página más consistente y legible.

Nueva estructura:



`<header>` `<h1>` Título de la página `</h1>` `</header>`

`<article>` Contenido del post `</article>`

`<article>` Contenido del post `</article>`

`<aside>` `` `` Elementos típicos del sidebar `` `` `</aside>`

`<footer>` `<p>` Elementos típicos del footer `</p>` `</footer>`



Ejemplo con todas las etiquetas nuevas de secciones:

Encabezado

- [Deportes](#)
- [Videos](#)
- [Novedades](#)

Noticias

Subida de temperaturas

La semana empieza con el cielo despejado y con una leve subida de las temperaturas que afectará a toda la Península...

Mi Publicidad

Texto de la publicidad

Otros temas relacionados

Información corporativa | Copyright © empresa, S. A. 2013, Todos los derechos reservados | Aviso legal | Política de Privacidad | Gestión publicitaria

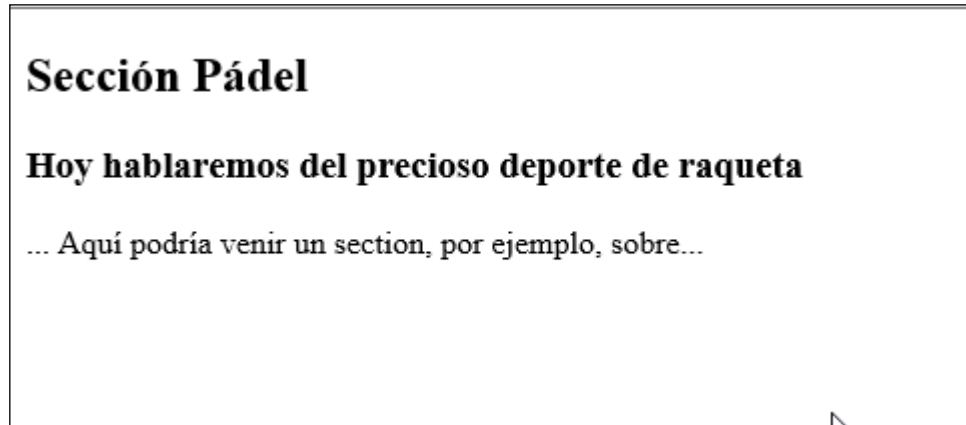
```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Titulo</title>
</head>
<body>
<header><h1> Encabezado </h1></header>
<nav>
  <ul>
    <li><a href="Deportes.html">Deportes</a></li>
    <li><a href="Videos.html">Videos</a></li>
    <li><a href="Novedades.html">Novedades</a></li>
  </ul>
</nav>
<section>
  <h1>Noticias</h1>
  <article>
    <section>
      <h2>Subida de temperaturas</h2>
      <p>La semana empieza con el cielo despejado y con una leve subida
de las temperaturas que afectará a toda la Península...</p>
    </section>
  </article>
</section>
<aside>
  <h1>Mi Publicidad</h1>
  <p>Texto de la publicidad</p>
  <p>Otros temas relacionados</p>
</aside>

<footer>
  Información corporativa | Copyright © empresa, S. A. 2013, Todos los derechos
reservados | Aviso legal | Política de Privacidad | Gestión publicitaria
</footer>
</body>
</html>
```

3. <hgroup>

Este elemento no tiene excesivo secreto, si nos fijamos en la documentación es simple y su única función es agrupar uno o más elementos de <h1> a <h6>, siempre que semánticamente valga la pena.

Es más, no hay demasiadas webs que lo utilicen por norma, pero hay que crecer en el uso de elementos semánticos, como en nuestro lenguaje cotidiano intentamos crecer día a día, y hablar más correctamente, con html tiene que suceder lo mismo, debemos "hablar" mejor ;-)

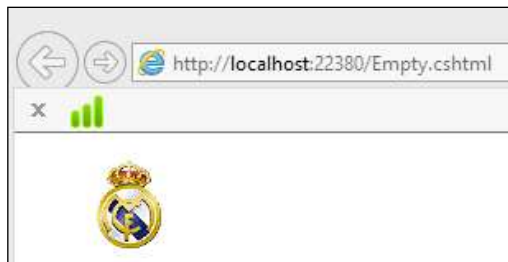


```
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Titulo</title>
</head>
<body>
  <hgroup>
    <h2>
      Sección Pádel
    </h2>
    <h3>
      Hoy hablaremos del precioso deporte de raqueta
    </h3>
  </hgroup>
  ... Aquí podría venir un section, por ejemplo, sobre...
</body>
</html>
```

4. <figure> y <figcaption>

<figure> está principalmente pensado para utilizar junto a <figcaption> y en resumen, es para diagramas, ilustraciones, fotos (sería perfecto, por ejemplo, para grandes fragmentos de código en blogs de programación).

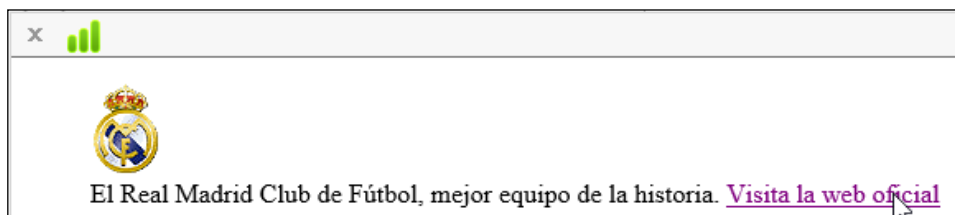
<figcaption> es opcional, y podemos ponerlo antes o después de la imagen en cuestión, es algo así como el "pie de foto".



```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Titulo</title>
</head>
<body>
  <figure>
    
  </figure>

</body>
</html>
```

Usando **<figcaption>** dentro.



```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Titulo</title>
</head>
<body>

  <figure>
    

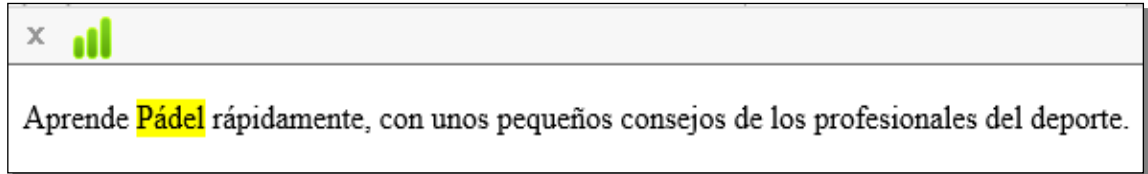
    <figcaption>
      El Real Madrid Club de Fútbol, mejor equipo de la historia.
      <a href="http://www.realmadrid.com">Visita la web oficial</a>
    </figcaption>
  </figure>

</body>
</html>
```

También podríamos ponerlo antes de la imagen, o podríamos utilizar más de una imagen, y poner varias con un mismo **<figcaption>** dentro.

5. <mark> y <progress>

<mark> Representa una extensión de texto marcado o resaltado para fines de referencia.



```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Titulo</title>
</head>
<body>
  <p>
    Aprende <mark>Pádel</mark> rápidamente, con unos pequeños consejos de
    los profesionales del deporte.
  </p>
</body>

</html>
```

<progress> Dibuja una barra de progreso, como las típicas barras que te van indicando el tiempo que llevan ejecutándose y el que falta por terminar... Por ejemplo, descargando un archivo de algún servidor.



```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Titulo</title>
</head>
<body>
  <p>
    Tiempo de espera restante: <progress
value="85"></progress>
  </p>

</body>
</html>
```

<progress> es un elemento perfecto para ir cambiando mediante javascript el atributo "value",

6. input, placeholder, email, url, date, autofocus, required

<input> de HTML, es una etiqueta mediante la cual pasamos datos a través de un formulario **<form>**.

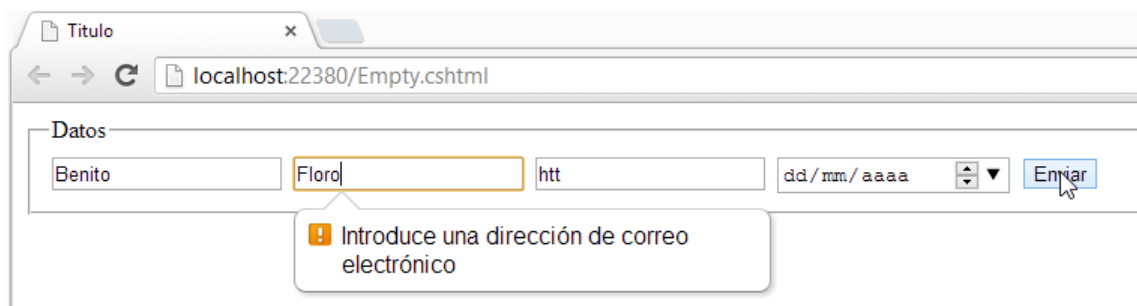
```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Titulo</title>
</head>
<body>
<form action="" method="">

<fieldset>
  <legend>Datos</legend>
  <input type="text" placeholder="Tu nombre" name="nombre" required
autofocus>
  <input type="email" placeholder="Correo electrónico" name="email"
required>
  <input type="url" placeholder="Tu web" name="url">
  <input type="date" name="fecha">
  <input type="submit" value="Enviar">
</fieldset>
</form>

</body>
</html>
```

Novedades en los controles form:

- El atributo "placeholder" es un texto que se pone sobre el input, una vez haces focus sobre él, se quita, y si no pones nada en el interior del input, al quitar el foco sobre el mismo, vuelve a aparecer ese texto...
- El atributo "required", hace que al realizar submit el formulario, no lo envíe si no está rellenado, y te avise mediante una alerta.
- El atributo "autofocus", hace que nada más cargar la página, el cursor se sitúe en ese input.
- El atributo "type = email", obliga a que sea una dirección de email correcta.
- El atributo "type = url", obliga a que sea una dirección web correcta.
- El atributo "type = date", te despliega opciones de fecha.



The screenshot shows a web browser window with a single tab titled 'Titulo'. The address bar shows 'localhost:22380/Empty.cshtml'. The form, titled 'Datos', contains four input fields and a submit button labeled 'Enviar'. The first field (name) contains 'Benito', the second (email) contains 'Floro', the third (URL) contains 'htt', and the fourth (date) shows a calendar icon and the text 'dd/mm/aaaa'. A tooltip is displayed over the email field, containing an information icon and the text 'Introduce una dirección de correo electrónico'.

NOTA: IE todavía no reconoce todos los controles.

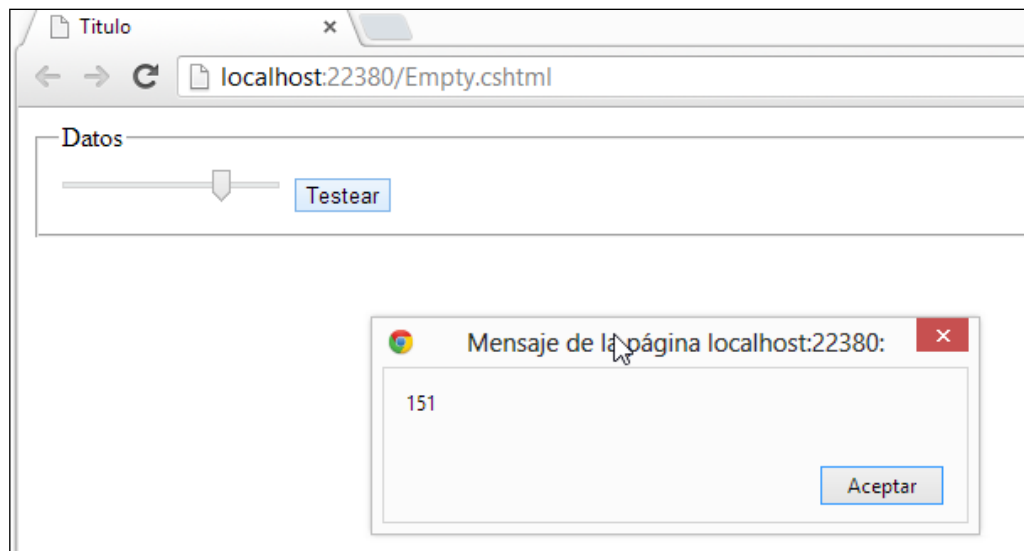
The screenshot shows a web browser window with two tabs, both titled 'Titulo'. The address bar shows 'localhost:22380/Empty.cshtml'. The page content is a form titled 'Datos' with three input fields: 'Benito', 'BenitoFloro@machote.es', and 'http://elgranbenitofloro.es'. To the right of these fields is a date picker showing 'mayo de 2013' and a calendar grid. The calendar grid has columns for days of the week (lu., ma., mi., ju., vi., sá., do.) and rows of dates. The date '16' is highlighted. An 'Enviar' button is located to the right of the date picker.

7. Input type="range"

Con "range" obtenemos algo similar a la barra de progress que hemos comentado pero esta vez, el valor lo escoge el usuario, es decir, con su interacción selecciona el rango que desea (y que se le permite).

```
<!DOCTYPE html>
<html lang="es">
<head>

<title>Titulo</title>
</head>
<body>
  <form action="" method="">
    <fieldset>
      <legend>Datos</legend>
      <input id="range" type="range" min="1" max="201" value="5"
step="50">
      <input type="button"
onclick="alert(document.getElementById('range').value);" value="Testear">
    </fieldset>
  </form>
</body>
</html>
```



NOTA: WEB DONDE PODEMOS VER QUE PARTE FUNCIONA CON LOS DIFERENTES NAVEGADORES: <http://caniuse.com/>

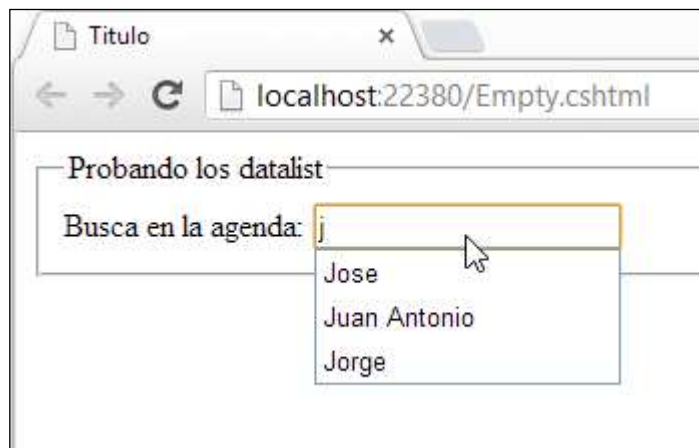
8. DataList

Podríamos decir, para simplificar mucho, que es como un **<select>**, pero en lugar de seleccionar en primera instancia la opción, puedes ir escribiéndola y te va presentando las posibles opciones.

Por tanto, con html5 simplificamos el uso de scripts, ya que, todos los que son más comunes ya vendrán nativamente en los navegadores con html5, y así evitamos el uso de tanto scripts.

```
<!DOCTYPE html>
<html lang="es">
<head>

<title>Titulo</title>
</head>
<body>
<form>
    <fieldset>
        <legend>Probando los datalist</legend>
        <label>Busca en la agenda:
            <input type="text" name="agenda" list="agenda" />
            <datalist id="agenda">
                <option value="Angelina">
                <option value="Jose">
                <option value="Rosa">
                <option value="Juan Antonio">
                <option value="Jorge">
            </datalist>
        </label>
    </fieldset>
</form>
</body>
</html>
```



9. Reproducción de vídeos en internet

Añadir video y sonido a una página web es una manera de hacerla más atractiva. Los contenidos multimedia captan la atención del visitante y nos permiten llegar a audiencias que, muy a menudo, adoptan una actitud de rechazo hacia las páginas con texto abundante. En HTML5 se resuelve directamente la construcción de páginas con audio y vídeo. Antes de la llegada de HTML5 dependíamos del uso de complementos instalables, como Flash, QuickTime o Silverlight para poder mostrar contenidos de vídeo. Con HTML5 se resuelve esta necesidad ya que especifica un elemento HTML llamado `<video>` que funciona de manera nativa en el navegador y se integra con Javascript.

Ejemplo de carga de video:

```
<body>

    <video width="640" height="360" poster="dostontosmuytontos.jpg" controls
autoplay>
        <!-- El atributo "poster", muestra una imagen al empezar, hasta que
no se le da al "play"-->
        <source src="PeliHTML5.mp4" type="video/mp4" /><!-- Safari , IE9 --
>
        <source src="PeliHTML5.webm" type="video/webm" /><!-- Chrome10+,
Ffx4+, Opera10.6+ -->
        <source src="PeliHTML5.ogv" type="video/ogg" /><!-- Firefox3.6+ /
Opera 10.5+ -->
        <object width="640" height="360" type="application/x-shockwave-
flash" data="player.swf">

            <param name="movie" value="PeliHTML5.swf" />
            <param name="flashvars"

value="autostart=true&controlbar=over&image=poster.jpg&file=archivo.mp4" />
            <!-- Imágen antes de la carga -->
            
        </object>
    </video>

</body>
```



- **SRC:** Nos enlaza el archivo de video que queremos reproducir.
- **WIDTH:** Nos define el ancho del video en pixeles.
- **HEIGHT:** Nos define la altura del video en pixeles.
- **CONTROLS:** Nos permite implementar los controles del reproductor por defecto del navegador como, botón play-pause, seek y volumen.
- **AUTOPLAY:** Nos permite reproducir el archivo de video desde que se carga la página.
- **PRELOAD:** Nos carga un poco el archivo de video antes de iniciar la reproducción en el buffer para que no se trabe por reproducir más de lo que carga.

Lo que estamos haciendo es asegurándonos con las diferentes etiquetas source es que en todos los navegadores actuales haya una solución de visionado que esté nativamente implantada en ellos, es decir, que no tengan que instalar nada para poder verlo correctamente... y en el caso de que el navegador sea antiguo o no pueda reproducirlo, se le da la opción del visionado con flash (si no lo tiene tampoco, entonces sí tendrá que instalarlo, como hasta ahora...).

10. Output

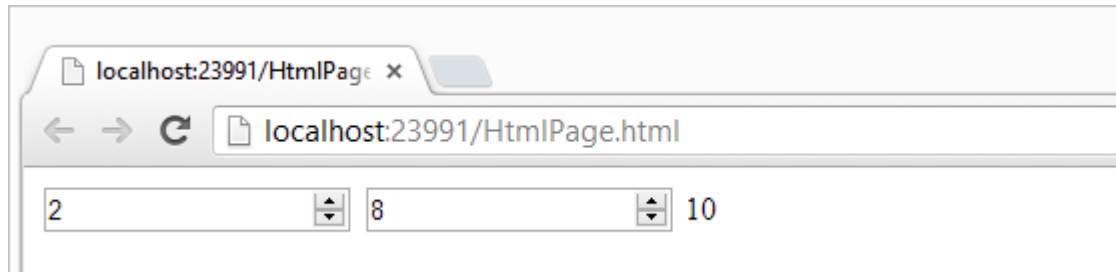
HTML5 nos ofrece un nuevo elemento llamado `<output>`. Este se incorporó para representar el resultado de un cálculo.

`<body>`

```

        <form      oninput="txtResultado.value      =      parseInt(txt1.value)      +
parseInt(txt2.value)">
            <input type="number" name="txt1" />
            <input type="number" name="txt2" />
            <output name="txtResultado">0</output>
        </form>
</body>

```



HTML5 introduce una nueva propiedad llamada valueAsNumber. Esta retorna el valor como un número en vez de como un string. Por lo tanto, ya no es más necesario usar el parseInt o parseFloat y el operador "+" suma en vez de concatenar.

```

<body>

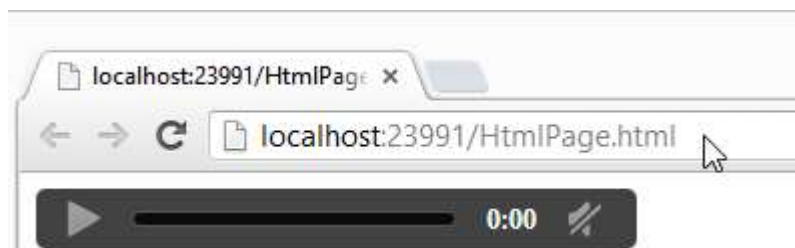
        <form      oninput="txtResultado.value      =      txt1.valueAsNumber      +
txt2.valueAsNumber">
            <input type="number" name="txt1" />
            <input type="number" name="txt2" />
            <output name="txtResultado">0</output>
        </form>

</body>

```

11. Audio

Etiqueta mediante el cual enlazamos un archivo de audio.



```

<body>

    <audio src="Halamadrid.oga" controls>
        <a href="Halamadrid.oga">Himno del Real Madrid</a>
    </audio>
    <br/>
</body>

```

Tenemos muchos navegadores con diferente compatibilidad, pero lo que más nos interesa son los **Motores de Renderizado**, estos se encargan de renderizar el código de nuestra página web e implementar ahora el contenido multimedia.

Veamos cuales motores corresponden a cada navegador:

- Google Chrome: WebKit
- Safari: WebKit
- Mozilla FireFox: Gecko
- Internet Explorer: Trident
- Opera: Presto

Estos son los Navegadores más importantes hasta ahora, y ahora veamos la compatibilidad con los archivos de audio.

	WebKit	Gecko	Trident (IE9)	Presto
Mp3	*		*	
Ogg	*(Safari no)	*		*
Wav	*(Safari si)	*	*	*

Con la etiqueta **SOURCE** podemos decirle que archivo va enlazar y que tipo de audio es.

```
<audio controls autoplay preload>
<source src="Halamadrid.ogg" type="audio/ogg" />
<source src="Halamadrid.mp3" type="audio/mpeg" />
<source src="Halamadrid.wav" type="audio/wav" />
Tu navegador no soporta esta característica
</audio>
```

12. ALMACENAMIENTO WEB

Existen dos tipos de almacenamiento que trabajan prácticamente igual, es decir las funciones para guardar, recuperar y eliminar son los mismos, la diferencia radica en que el **sessionStorage** funciona mientras el usuario tiene el browser abierto y se guarda en el servidor (sin necesidad de utilizar ningún lenguaje de servidor), y el **localStorage** persiste aun después de cerrar el explorador (persiste incluso entre pestañas).

SessionStorage se tiene que utilizar con algún servidor (ejecutarlo vía http), no se puede ejecutar directamente la página.

LocalStorage se puede cargar la página en local para probarlo.

SessionStorage

Si hay algo que siempre es extraño de HTML es la forma de almacenar datos, que ayude al usuario a una mejor movilidad mientras navega por nuestras páginas. Esto se puede lograr mediante el uso de las llamadas cookies, el problema es que

tiene ciertas restricciones. Ya con HTML5 y Javascript esto se puede lograr mediante un atributo del DOM llamado sessionStorage que se utiliza como objeto global de Javascript, el cual dispone de métodos que nos ayudan a manipular los datos que queramos utilizar.

¿Cómo funciona SessionStorage?

Cuando abrimos una página web con el navegador pedimos información al servidor y en ese momento se inicia una sesión, con el cual podemos guardar datos que estará sólo accesible para el navegador con el cual estamos conectados. Ahora, si con el navegador no realizamos ninguna actividad con el servidor durante al menos 15 minutos, entonces el servidor borrará esos datos de la sesión.

Lamentable los datos almacenados son guardados como cadenas y al recuperarlo con el método getItem() se obtienen cadenas.

Ejemplo:

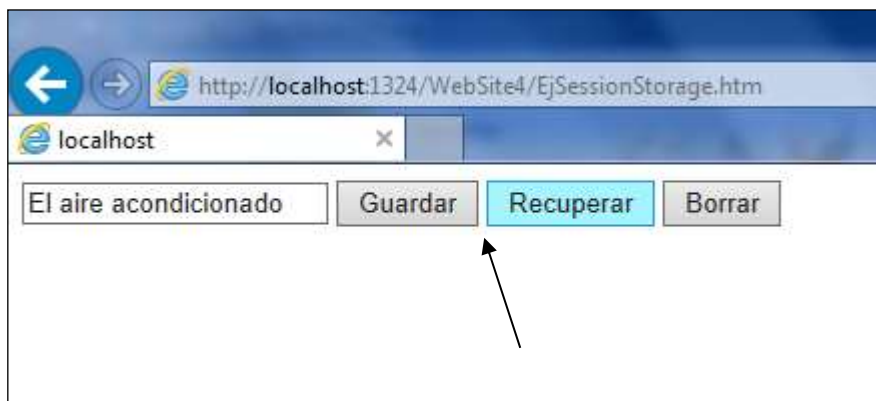
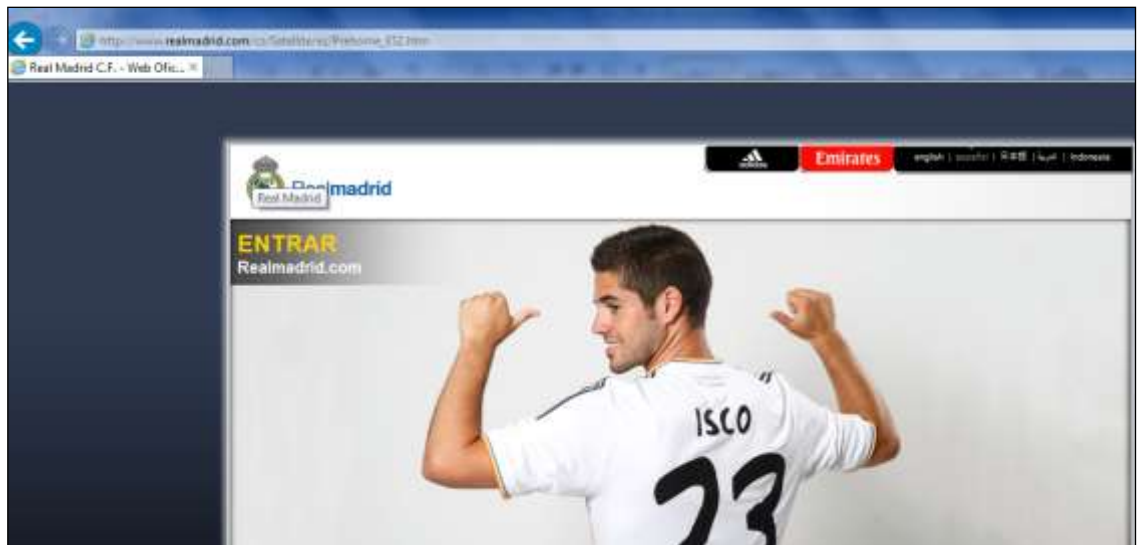
- Diseñaremos una página html con tres botones y una caja de texto.
- Al pulsar sobre el botón guardar, almacenaremos la información escrita sobre la caja de texto en un objeto SessionStorage.

```
sessionStorage.setItem(clave, valor);
```



- Si navegamos a otra página y después volvemos a nuestro sitio, comprobaremos que la información se guardó.

```
sessionStorage.getItem(clave);
```



- La opción borrar eliminará la variable guardada en el objeto SessionStorage.

```
sessionStorage.removeItem(clave);
```

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

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title></title>
  <script type="text/javascript">
    function guardarenSession() {
      var valor = document.getElementById('txtDatos').value;
      sessionStorage.setItem('Datos', valor);
    }
    function recuperarSession() {
      var valor = sessionStorage.getItem('Datos');
      document.getElementById('txtDatos').value = valor;
    }
    function borrarSession() {
      sessionStorage.removeItem('Datos');
      document.getElementById('txtDatos').value = "";
    }
  </script>
</head>
<body>
```



```
<input type="text" id="txtDatos"/>
<input type="button" onclick="guardarenSession()" value="Guardar"/>
<input type="button" onclick="recuperarSession()" value="Recuperar"/>
<input type="button" onclick="borrarenSession()" value="Borrar"/>
</body>
</html>
```

LocalStorage

Local Storage (almacenamiento local) es una base de datos de tipo key — value (clave — valor). Los datos se almacenan en lado de cliente.

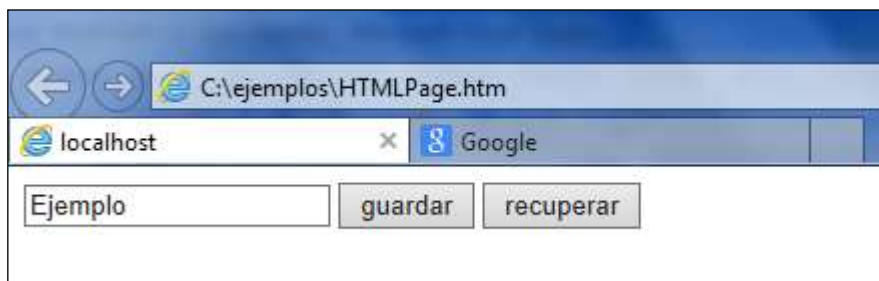
Las principales ventajas antes de los cookies:

- El tamaño del almacenamiento es más grande (se limita en los ajustes de navegador)
- Alto rendimiento (debido a que los datos se almacenan en el HDD del cliente)
- Tiempo de vida illimitado

Ejemplo

- Diseñaremos una página html con dos botones y una caja de texto.
- Al pulsar sobre el botón guardar, almacenaremos la información escrita sobre la caja de texto en un objeto LocalStorage.

```
localStorage.setItem(clave, valor);
```



- Si cerramos el navegador o abrimos otra pestaña y después volvemos a nuestro sitio, comprobaremos que la información se guardó.

```
localStorage.getItem(clave);
```

```
<script>
function guardar() {
    var miDato = document.getElementById('miInput').value;
    localStorage.setItem('miDato', miDato);
}
function recuperar() {
    var miDato = localStorage.getItem('miDato');
    document.getElementById('miInput').value = miDato;
}
</script><input type="text" id="miInput">
<input type="button" onclick="guardar()" value="guardar">
<input type="button" onclick="recuperar()" value="recuperar">
```

13. DRAG & DROP

Drag and Drop es una nueva API incluida en HTML5. Para crear un elemento arrastrable simplemente tenemos que añadirle `draggable=true`. Además, esta API incluye 7 nuevos eventos a HTML que nos permiten controlar las todas las acciones posibles del elemento.

Los elementos arrastrables pueden producir tres eventos:

- `dragstart`: lanzado cuando se empieza arrastrar un elemento.
- `drag`: mientras se arrastra un elemento.
- `dragend`: lanzado cuando se termina de arrastrar un elemento.

Y los contenedores que pueden recibir los elementos arrastrables tienen 4 eventos posibles.

- `dragenter`: cuando un elemento arrastrable se arrastra dentro de un elemento.
- `dragleave`: cuando el elemento arrastrable sale del elemento.
- `dragover`: lanzado cuando un elemento se mueve dentro de un elemento.
- `drop`: lanzado cuando un objeto arrastable es soltado dentro.

Con esto ya podremos crear un sitio con Drag and Drop como veremos a continuación.

En el siguiente ejemplo veremos la forma de arrastrar elementos con HTML5:

Ejemplo antes de arrastrar

Por favor posicione a los alumnos en el lugar que corresponda	
Alumnos	Fernando Antonio H. Miguel Gonzalo Antonio G. Isaac C. Roberto Alfonso I. Pablo David Isaac S. Javier Carlos Dan Oscar Wilmar Analle Laura Gema Ylana Arianne Ana Alex
Ganadores de Pádel	
Tromistas	
Expertos en cerveza	
Apuesta bica	
Ojales	
Reyes del ping pong	
Profesor de Flash/Flash	

Ejemplo después de arrastrar

Por favor posicione a los alumnos en el lugar que corresponda

Alumnos	
Ganadores de Pádel	Miguel Alex
Tenistas	Antonio H. Gema Aurora Laura María Aurora Ana
Expertos en cerveza	Antonio G. Oscar Alfonso Roberto
Apreta bici	Héctor Fernando Javier
Gigolos	Gonzalo Dani
Reyes del ping-pong	David Isaac S. Isaac G.
Profesor de Flash Blend	Carlos J. Pablo

```

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Arrastrar y soltar</title>
</head>
<body>
  <div class="container_4">
    <div class="grid_4" style="margin-top:15px">
      <h1>Por favor posicione a los alumnos en el lugar que corresponda </h1>
      <section id="user-levels">
        <div id="unassigned" class="drop" ondrop="dropUser(this, event)"
ondragenter="return false" ondragover="return false">
          <span>Alumnos</span>
          <a draggable="true" class="user" id="A28" ondragstart="dragUser(this,
event)">Fernando</a>
          <a draggable="true" class="user" id="A29" ondragstart="dragUser(this,
event)">Antonio H.</a>
          <a draggable="true" class="user" id="A30" ondragstart="dragUser(this, event)">Miguel</a>
          <a draggable="true" class="user" id="A31" ondragstart="dragUser(this,
event)">Gonzalo</a>
          <a draggable="true" class="user" id="A32" ondragstart="dragUser(this,
event)">Antonio G.</a>
          <a draggable="true" class="user" id="A1" ondragstart="dragUser(this,
event)">Isaac G.</a>
          <a draggable="true" class="user" id="A2" ondragstart="dragUser(this,
event)">Roberto</a>
          <a draggable="true" class="user" id="A3" ondragstart="dragUser(this,
event)">Alfonso</a>
          <a draggable="true" class="user" id="A4" ondragstart="dragUser(this,
event)">J. Pablo</a>
          <a draggable="true" class="user" id="A5" ondragstart="dragUser(this,
event)">David</a>
          <a draggable="true" class="user" id="A6" ondragstart="dragUser(this, event)">Isaac S.</a>
          <a draggable="true" class="user" id="A7" ondragstart="dragUser(this,
event)">Javier</a>
          <a draggable="true" class="user" id="A8" ondragstart="dragUser(this,
event)">Carlos</a>

```

```

        <a          draggable="true"          class="user"          id="A9"
ondragstart="dragUser(this, event)">Dani</a>
        <a          draggable="true"          class="user"          id="A10"
ondragstart="dragUser(this, event)">Óscar</a>
        <a          draggable="true"          class="user"          id="A11"
ondragstart="dragUser(this, event)">Héctor</a>
        <a          draggable="true"          class="user"          id="A12"
ondragstart="dragUser(this, event)">Amelia</a>
        <a          draggable="true"          class="user"          id="A13"
ondragstart="dragUser(this, event)">Laura</a>
        <a          draggable="true"          class="user"          id="A14"
ondragstart="dragUser(this, event)">Gema</a>
        <a          draggable="true"          class="user"          id="A17"
ondragstart="dragUser(this, event)">Nuria</a>
        <a          draggable="true"          class="user"          id="A15"
ondragstart="dragUser(this, event)">Aurora</a>
        <a          draggable="true"          class="user"          id="A16"
ondragstart="dragUser(this, event)">Ana</a>
        <a draggable="true" class="user" id="A26" ondragstart="dragUser(this,
event)">Alex</a>
    </div>

    <div id="Div11" class="drop" ondrop="dropUser(this, event)"
ondragenter="return false" ondragover="return false"><span>Ganadores de
Paddel</span></div>
    <div id="Div12" class="drop" ondrop="dropUser(this, event)"
ondragenter="return false" ondragover="return false"><span>Tronistas</span></div>
    <div id="Div13" class="drop" ondrop="dropUser(this, event)"
ondragenter="return false" ondragover="return false"><span>Expertos en
cerveza</span></div>
    <div id="Div1" class="drop" ondrop="dropUser(this, event)"
ondragenter="return false" ondragover="return false"><span>Apreta
bicis</span></div>
    <div id="Div2" class="drop" ondrop="dropUser(this, event)"
ondragenter="return false" ondragover="return false"><span>Gigolos</span></div>
    <div id="Div3" class="drop" ondrop="dropUser(this, event)"
ondragenter="return false" ondragover="return false"><span>Reyes del ping
pong</span></div>
    <div id="Div4" class="drop" ondrop="dropUser(this, event)"
ondragenter="return false" ondragover="return false"><span>Profesor de
Flash/Blend</span></div>
    <div class="clear"></div>
</section>

<script type="text/javascript">
    function dragUser(user, event) {
        event.dataTransfer.setData('User', user.id);
    }

    function dropUser(target, event) {
        var user = event.dataTransfer.getData('User');
        target.appendChild(document.getElementById(user));
    }
</script>

<style type="text/css">
    .drop {
        width: 100%;
        padding: 20px;
        height: 18px;
        margin: 30px auto;
        background: #f7f7f7;

```

```

        border: 1px solid #333;
    }

    .drop span {
        margin-right: 20px;
        display: block;
        float: left;
    }

    .drop a {
        color: #FFF;
        background: #FF4b33;
        border-radius: 2px;
        padding: 5px;
    }
}

</style>

</div>
</div>
</body>
</html>

```

Primero tenemos que añadir `draggable="true"` al elemento que queremos arrastrar:

Fernando

Con esto ya podremos arrastrar el enlace, pero necesitamos indicarle al navegador la información que contiene el elemento que hemos cogido.

Fernando

Y añadirle la función javascript:

```

function dragUser(user, event) {
    event.dataTransfer.setData('User', user.id)
}

```

Ahora que tenemos definido el elemento que vamos a arrastrar necesitamos definir un área donde soltarlo:

Unassigned

Y el evento que se lanzará cuando soltemos un elemento sobre él:

```

function dropUser(target, event) {
    var user = event.dataTransfer.getData('User');
    target.appendChild(document.getElementById(user));
}

```

En el evento de comenzar a arrastrar creamos una variable llamada `user` en la que guardamos los datos del elemento que estamos arrastrando. Ahora cogemos esos datos y los introducimos en el elemento.

14. GEOLOCALIZACIÓN

El localizador de posición que ofrece HTML5 permite obtener la ubicación del usuario de forma precisa y visual, demarcaremos la posición del usuario utilizando los mapas de Google Maps.

La geolocalización de usuario en HTML5 se lleva a cabo gracias al potente API JavaScript combinada con el uso de Google Maps. De esta manera como desarrolladores, obtendremos la ubicación de los usuarios sobre un mapa y dispondremos de una importante base de datos para ofrecer a los usuarios servicios cercanos su domicilio.

Mensaje mientras encuentra la localización

Buscando su localización...



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

```
<script>
function success(position) {
    var status = document.querySelector('#status');
    status.innerHTML = "¡Su ubicación!";

    var mapcanvas = document.createElement('div');
    mapcanvas.id = 'mapcanvas';
    mapcanvas.style.height = '400px';
    mapcanvas.style.width = '560px';
```

```

        document.querySelector('#map').appendChild(mapcanvas);

        var latlng = new google.maps.LatLng(position.coords.latitude,
position.coords.longitude);
        var myOptions = {
            zoom: 15,
            center: latlng,
            mapTypeControl: false,
            navigationControlOptions: { style:
google.maps.NavigationControlStyle.SMALL },
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };
        var map = new google.maps.Map(document.getElementById("mapcanvas"),
myOptions);

        var marker = new google.maps.Marker({
            position: latlng,
            map: map,
            title: "Usted está aquí."
        });
    }

    function error(msg) {
        var status = document.getElementById('status');
        status.innerHTML = "Error [" + error.code + "]: " + error.message;
    }

    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(success, error, {
maximumAge: 60000, timeout: 4000 });
    } else {
        error('Actualiza el navegador web para usar el API de localización');
    }

</script>

<p id="status">Buscando su localizaci&ocute;n...</p>
<div id="map"></div>

```

Explicación código:

Lo primero que hacemos es detectar si el navegador soporta la geolocalización, luego pediremos las coordenadas del usuario (previa aceptación del usuario) y una vez recibidas las coordenadas mostramos la ubicación del usuario en un mapa de Google. (Llamada a la Geolocation API, Mapa Google centrado en la posición del usuario, Error, Otras propiedades de getCurrentPosition() y La función watchPosition().)