



JAVASCRIPT Web APIs

© 2017, ACTIBYTI PROJECT SLU, Barcelona
Autor: Ricardo Ahumada



UNIÓN EUROPEA
Fondo Social Europeo
"El FSE invierte en tu futuro"

ÍNDICE DE CONTENIDOS

1. Caso práctico
2. Introducción
3. Drag & drop
4. Geolocalización
5. Local storage
6. Indexed DB
7. Archivos locales
8. Web Workers
9. Web sockets
10. API History
- A1. Librerías Auxiliares

1

CASO PRÁCTICO

Caso Práctico: BananaTube Local

“BananaTube” es el proyecto estrella de Banana Apps.

BananaTube será el próximo boom! de las redes sociales; permitirá a sus usuarios gestionar videos, exponerlos en su muro, comentar videos propios y de sus amigos, calificarlos y compartirlos en varios canales.

En esta etapa del proyecto se quiere incorporar funcionalidades especiales al prototipo: almacenar los videos, comentarios, etc del muro en una base de datos local; usar un mecanismo de drag&drop para reordenar videos. Suscribirse a mensajes del servidor. Geolocalizar a los usuarios.

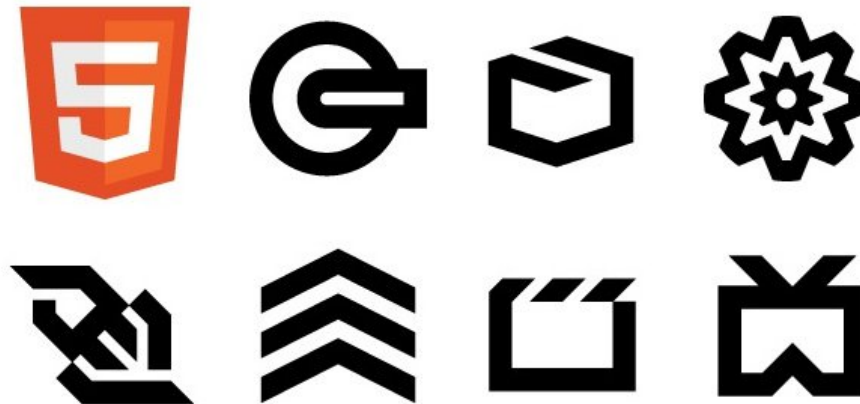
Discutamos

- Seguimos trabajando con Javascript?
- Necesitamos algo más?
- Y nuestro entorno?
- Qué herramientas necesitaremos?

2

INTRODUCCIÓN

APIs HTML5



- Una interfaz de programación de aplicaciones (API: Application Programming Interface) es una colección de instrucciones y estándares de programación para acceder a una aplicación de software.
- Con una API, es posible diseñar productos basados en el servicio que proporciona la API.
- HTML5 expone todo un set de tecnologías que se pueden usar en la generación de la capa de presentación de una aplicación web.
- Usaremos Javascript para acceder a las API.

Referencias

- El estado actual de dichas APIs se pueden consultar en el sitio W3C
 - https://www.w3.org/standards/techs/js#w3c_all
- En el sitio MDN tenemos una referencia práctica de las mismas
 - <https://developer.mozilla.org/en-US/docs/Web/API>

Ejemplos de APIs

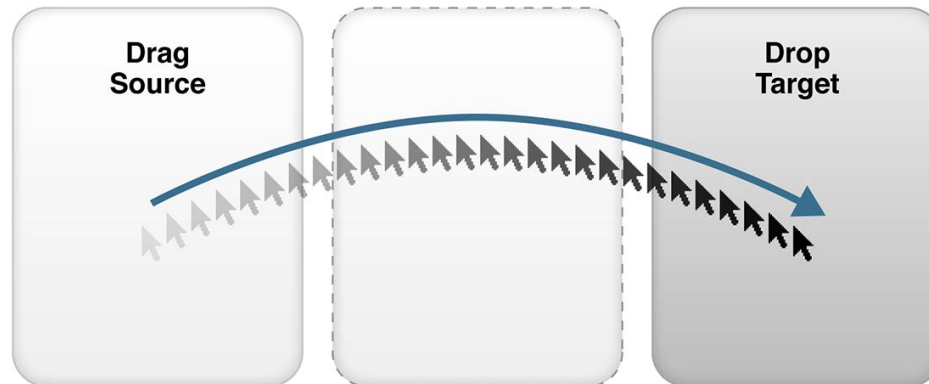
- Una API de dibujo en 2D utilizada con el nuevo elemento de lienzos para representar gráficos y otras imágenes visuales
- Un mecanismo API de memoria caché que soporta aplicaciones web offline
- Una API para reproducir video y audio utilizado con los nuevos elementos de video y audio
- Una API de historial que hace que el historial de navegación sea accesible y permite que las páginas se agreguen a este
- Una API de arrastrar y soltar para ser utilizada con el atributo draggable
- Una API de edición para ser utilizada con el atributo contenteditable
- Almacenamiento del lado del cliente con APIs de JavaScript para pares de valor clave y también bases de datos SQL intercaladas

3

DRAG & DROP

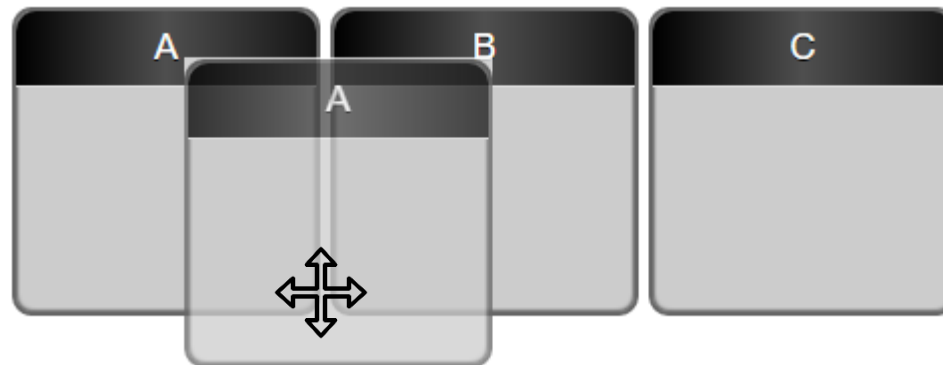
Arrastrar y Soltar

- Es un mecanismo basado en eventos, el API de JavaScript y atributos (**draggable**), para permitir que prácticamente cualquier tipo de elemento de una página se pueda arrastrar.
- El proceso es muy similar al de cualquier mecanismo de comunicación;
 - **Emisor**: el usuario
 - **Receptor**: el elemento sobre el que se suelta el objeto
 - **Canal**: el contexto de ejecución
 - **La información transmitida**: objeto inicial que el usuario quiere arrastrar



Elementos arrastables

```
<div id="columns">  
  <div class="column" draggable="true"><header>A</header></div>  
  <div class="column" draggable="true"><header>B</header></div>  
  <div class="column" draggable="true"><header>C</header></div>  
</div>
```



Mecanismos de escucha

- El mecanismo de escucha se implementa mediante eventos
- El elemento arrastrado lanza los eventos **dragstart**, **drag** y **dragend**.
- **dragstart**: Este evento es disparado en el momento en el que el arrastre comienza. Los datos asociados con el elemento origen son definidos en este momento en el sistema.
 - El objeto ***dataTransfer*** contiene la información sobre el objeto arrastrado.
 - se establece en el evento dragstart y se lee/procesa en el evento drop.
 - `e.dataTransfer.setData(format, data)`
- El evento **drag** es similar al evento mousemove, excepto que será disparado durante una operación de arrastre por el elemento origen.
- **dragend** se dispara cuando la operación de arrastrar y soltar termina (con éxito o no)

Eventos disparados en receptor

- **dragenter** Cuando el puntero del ratón entra dentro del área ocupada por los posibles elementos destino durante una operación de arrastrar y soltar, este evento es disparado.
- **dragover** Este evento es similar al evento mousemove, excepto que es disparado durante una operación de arrastre por posibles elementos destino.
- **drop** Cuando el elemento origen es soltado durante una operación de arrastrar y soltar, este evento es disparado por el elemento destino.
- **dragleave** Este evento es disparado cuando el ratón sale del área ocupada por un elemento durante una operación de arrastrar y soltar. (se usa junto con dragenter para mostrar una ayuda visual al usuario que permita identificar el elemento destino).

Asignación de eventos

- Mediante la asignación de los eventos citados conseguimos un buen nivel de control de la operación, pudiendo tratar el elemento soltado según su tipo.

```
function iniciarDD() {  
    origen1 = document.getElementById('grafico');  
    origen1.addEventListener('dragstart', arrastrado, false);  
  
    destino = document.getElementById('cajasoltar');  
  
    destino.addEventListener('dragenter', function (e) {  
        e.preventDefault();  
    }, false);  
  
    destino.addEventListener('dragover', function (e) {  
        e.preventDefault();  
    }, false);  
  
    destino.addEventListener('drop', soltadoGrafico, false);  
}
```

```
function arrastrado(e) {  
    var codigo = '<img src="' +  
    origen1.getAttribute('src') + '>';  
    e.dataTransfer.setData('Text', codigo);  
}  
  
function soltadoGrafico(e) {  
    e.preventDefault();  
    destino.innerHTML =  
    e.dataTransfer.getData('Text');  
}  
  
window.addEventListener('load',  
    iniciarDD, false);
```



Pongámoslo en práctica

- Haz que al arrastrar una imagen, esta desaparezca del origen.

Tip: `origen1.parentNode.removeChild(origen1);`

4

GEOLOCALIZACIÓN

Geo-Localización

- Consiste en la capacidad de un sitio o aplicación web para determinar la procedencia geográfica del visitante (o la ubicación del usuario)
 - Responder adecuadamente
 - Auditoría de uso
 - Estadísticas
 - Control de accesos
 - Personalización de la interfaz
 - Oferta de servicios asociados a una ubicación
- Se basa en un servicio “on-line” al que se consulta para obtener los datos de ubicación del solicitante
- Los datos son devueltos en términos de latitud y longitud geográficas, junto a un conjunto adicional de metadatos
- Latitud y longitud pueden ser expresadas en forma decimal (Lat. 42,123122) o en formato de sexagesimal (Lat. 42° 23' 14")
- El API de geo-localización nos devuelve la información en formato decimal



¿De dónde proviene la información?

- El API utilizado simplemente expone un conjunto de mecanismos de acceso a la información, pero no garantiza la fiabilidad más allá de un cierto punto
- Se puede utilizar cualquier de los métodos siguientes para acceder a esa información
 - Dirección IP
 - Triangulación de coordenadas
 - Sistema de posicionamiento global (GPS)
 - Wi-Fi con direcciones MAC desde RFID, Wi-Fi y Bluetooth
 - Identificadores telefónicos tipo GSM o CDMA
 - Definida por el usuario

Privacidad

> Privacidad

- > El estándar recomienda que estas opciones solo estén disponibles **cuando el usuario permita dicho acceso**
- > Todos los navegadores disponen de un mecanismo de protección que está activado de forma predeterminada
- > Cuando accedemos a una página que solicita este tipo de información, el mecanismo se activa
- > Los navegadores pedirán permiso para realizar la operación

> Soporte nativo

- > Naturalmente, el caso ideal es aquel en que no necesitamos soporte de librerías adicionales porque lo obtenemos directamente del propio navegador
- > Los datos geográficos en este caso se obtienen directamente del DOM, accediendo al objeto navigator.geolocation

Obteniendo los datos

- Con **navigator.geolocation**, controlamos el soporte, a la vez que programamos las peticiones y respuestas
- <https://developer.mozilla.org/en-US/docs/Web/API/Navigator/geolocation>

```
function obtenerLocalizacion() {  
    if(navigator.geolocation) {  
        document.getElementById("nivelSoporte").innerHTML =  
            "La Geo-Localizacion HTML5 está soportada en este navegador.";  
        navigator.geolocation.getCurrentPosition(updateLocation);  
    }  
}
```

```
function updateLocation(position) {  
    console.log('position:',position)  
    var lat = position.coords.latitude;  
    var lon = position.coords.longitude;  
    var pre = position.coords.accuracy;  
    ....  
}
```

Objeto position

- El objeto clave que nos devuelve la información pertinente es **position**
- En concreto este objeto contiene las siguientes propiedades:

latitude	(Latitud)
longitude	(Longitud)
accuracy	(Precisión)
altitude	(Altitud)
altitudeAccuracy	(Precisión de la altitud)
heading	(Dirección del movimiento, respecto al norte geográfico)
speed	(Velocidad del movimiento, en metros por segundo)

Mapas

- Podemos usar los datos devueltos y apoyarnos en APIs de mapas (Bing, Google, OpenMaps ..) .
- Bing Maps, ofrece servicios muy precisos de geo-localización
- Basta con entrar en el sitio <http://bing.com/maps> y nos ubica automáticamente en el punto del mapa más cerca de donde nos encontramos físicamente

```
if (navigator.geolocation) {  
    navigator.geolocation.getCurrentPosition(  
        function (position) {  
            //TODO: Centrar el mapa según las coordenadas  
  
        },  
        function (error) {  
            //TODO Manejar errores  
        }); } else {  
    //TODO Solución para resolución IP  
}
```



Pongámoslo en práctica

- Examina las demos de geolocalización y su integración con otros mecanismos como mapas

5

LOCAL STORAGE

Almacenamiento local y de sesión

- El objetivo de estas API es la persistencia de información entre distintas peticiones Web.
- Más adecuada que mediante cookies (más tamaño)
- Servicios accesibles en el objeto window:
 - **window.sessionStorage**
 - **window.localStorage**
- Para su funcionamiento se usa una "base de datos" manejada internamente por el navegador y accedida mediante estas API.



sessionStorage

- Almacenamiento y recuperación de datos en una sesión
- En la actualidad **solo soporta almacenar textos (strings)**
- La **escritura** de datos se hace mediante
`setItem(clave, valor)`
 - **clave:** el código que queremos darle al texto a almacenar para después recuperarlo
 - **valor:** el texto a almacenar
- La **lectura** de datos almacenados se hace mediante
`getItem(clave)`
 - permite recuperar el texto guardado con la clave indicada

Almacenamiento local y de sesión

- Podemos usar una interfaz de usuario sencilla con dos botones “Grabar” y “Leer”, y una caja de texto para almacenar los datos de entrada y mostrar la salida.

```
<body>
  <form action="Valid1.html" id="Formulario" >
    <p>Almacenamiento de valores de sesión</p>
    <input type="text" id="clave1" />
    <input type="button" value="Grabar"
      onclick="escribirClave('Dato');" /><br />
    <input type="text" id="claveLeida" />
    <input type="button" value="Leer"
      onclick="leerClave('Dato');" />
  </form>
</body>
```

Ejemplo

```
<input type="text" id="inputTemporal" />  
<input type="button" value="Grabar" onclick="GrabarClaveSesion('claveTemporal');" />  
    <input type="text" id="valorTemporal" />  
<input type="button" value="Leer" onclick="LeerClaveSesion('claveTemporal');" />
```

```
function GrabarClaveSesion(clave) {  
    var valor = document.getElementById("inputTemporal").value;  
    window.sessionStorage.setItem(clave, valor);  
}
```

```
function LeerClaveSesion(clave) {  
    var valor = window.sessionStorage.getItem(clave);  
    document.getElementById("valorTemporal").value = valor;  
}
```

Simplificación de sintaxis

- Puede simplificarse la sintaxis aún más, utilizando las llamadas “propiedades de expansión”
- Las siguientes instrucciones son equivalentes
 - `window.sessionStorage.setItem(clave1, valor);`
 - `window.sessionStorage.clave1 = valor;`
- Y lo mismo podemos hacer en el proceso de lectura
 - `window.sessionStorage.getItem(clave1);`
 - `window.sessionStorage.clave1`


Persistencia

- La persistencia tiene un tiempo de caducidad
- Cada vez que cierre o la página (o la solapa que contenga una página con este mecanismo), la información se perderá
 - Un cambio de página dentro del mismo sitio no produce la pérdida de la información
- Para casos en los que es necesario que la información persistida dure más allá de la sesión o del cierre del navegador, la opción adecuada es el almacenamiento local (**localStorage**)
- Estas API pueden sustituir ventajosamente a las cookies, y permiten cubrir de manera muy sencilla las necesidades de almacenamiento locales

```
window.localStorage.clave = valor;  
....  
var valor = window.localStorage.clave;
```

Almacenamiento local y de sesión

- Finalmente, una comparativa entre ambas revela las diferencias más importantes:

sessionStorage	
La persistencia se limita a la solapa de navegación (pestañas)	
Los valores almacenados	

- También existen limitaciones a la cantidad de información que puede guardarse en el apartado valor vinculado con una clave dada
- Existen unos valores de **cuota** que deberán de ser configurables por el usuario



Pongámoslo en práctica

- Crea una lista de objetos de usuarios y guárdalos en localStorage
- Recupera la lista y formatealo en una lista HTML
- **Tips:**
 - JSON.parse(string) → Object
 - JSON.stringify(object) → String



Pongámoslo en práctica

- Activa las herramientas de desarrollador con F12
- Ve a la pestaña de Application y examina los datos de Local y Session storage

6

INDEXED DB

Indexed DB

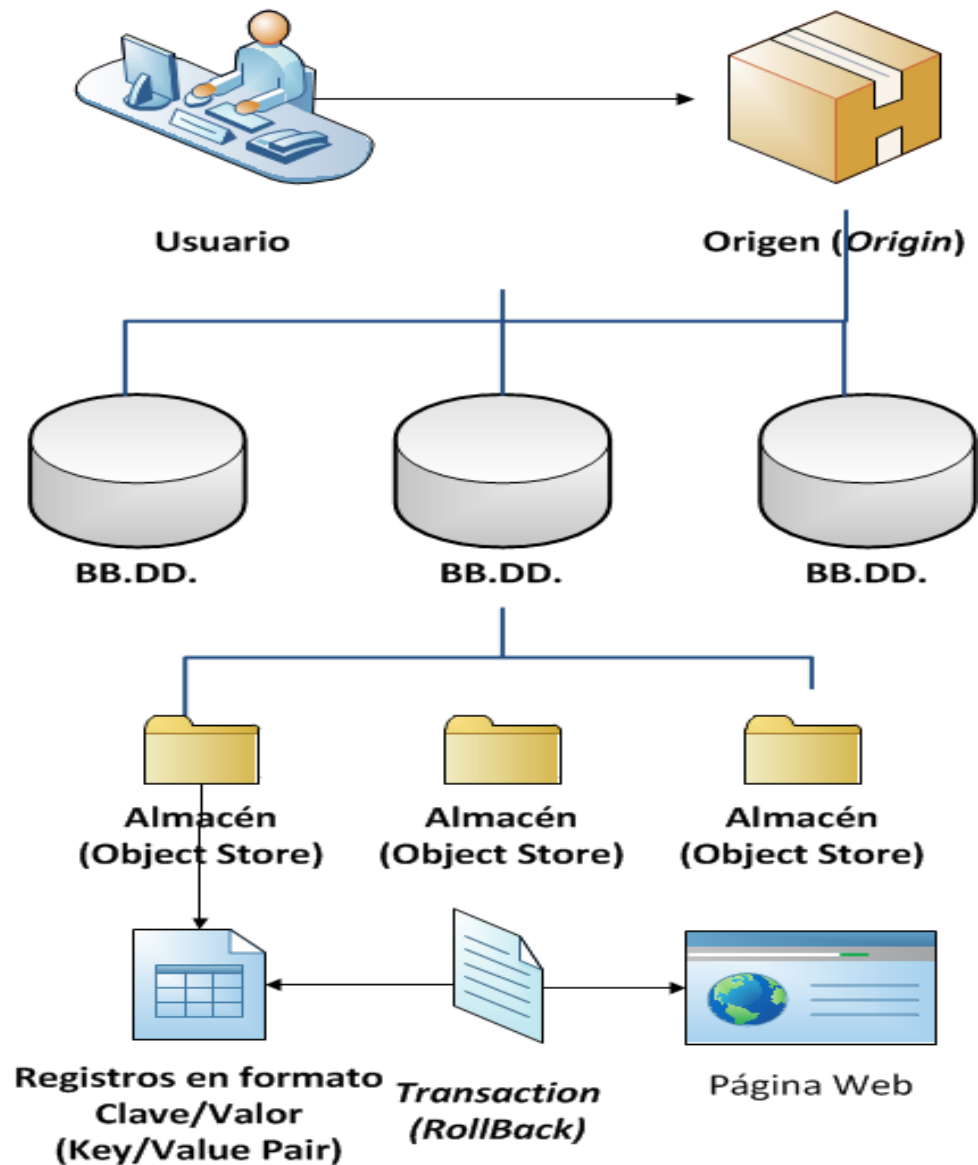
- Es una API para crear bases de datos locales, que extiende las posibilidades mostradas para las opciones de almacenamiento anteriores
 - <http://www.w3.org/TR/IndexedDB/>
- Permite el almacenamiento de valores simples y de objetos jerárquicos
- Cada registro consiste en una clave y un valor asociado
- La base de datos mantiene índices para manejar los registros guardados
- El acceso a un registro se puede realizar mediante el uso de la clave o del índice asociado
- Para el caso de muchos registros...
- Resulta especialmente útil para el uso en aplicaciones “off-line” que necesitan mantener grandes cantidades de información.

Indexed DB - origin

- Cada usuario dispone de un origen (origin) de información.
- Cada origin permite disponer de un conjunto de bases de datos asociadas.
- Cada base de datos está compuesta por uno o más almacenes de datos (object store)
- Cada object store almacena un conjunto de registros en un formato *key/value pair* (parejas clave/valor).
- Las listas se ordenan por la clave en orden ascendente
- Cada object store tiene un nombre que es único en la base de datos a la que pertenece.
- Todas las operaciones que tienen lugar sobre una base de datos de este tipo debe ser realizadas mediante una **transaction**.
- Implica capacidades de **rollback**
- Este tipo de operaciones implica acciones de tipo asíncrono

Indexed DB

- El esquema siguiente plantea esa arquitectura:





Pongámoslo en práctica

- Examina y analiza los ejemplos de Comentarios y FlightLog
- Activa las herramientas de desarrollador con F12
- Ve a la pestaña de Application y examina los datos de la base de datos



Pongámoslo en práctica

- Modifica el script de comentarios para crear una nueva tabla que almacene el número de comentarios por fecha.

7

ARCHIVOS LOCALES

File API



- Permite programar acciones sobre uno o varios archivos en la máquina del usuario
 - <https://developer.mozilla.org/en/docs/Web/API/File>
- Funciona conjuntamente con el elemento `<input type="file">` para la selección de los archivos
- El atributo **multiple** habilita la selección de más de un archivo
- Cuando se produce un proceso de interacción con el usuario, el elemento `<input>` recoge en su **atributo files** el conjunto de ficheros seleccionado por el usuario
- Si queremos obtener información sobre ellos, podemos programar una función en **JavaScript** que responda al evento **onchange** de este elemento
- El proceso es simple, inicialmente, y permite la manipulación o el manejo de archivos con propósito de lectura, subida al servidor, etc.

Ejemplo

- Por ejemplo, podemos programar una sencilla interfaz que habilite la selección de uno o más archivos por parte del usuario y muestre posteriormente la información de detalles de esos archivos en la página.

```
<h3>Seleccionar fichero(s)</h3>  
<p><input id="files-upload" type="file" multiple /></p>  
  
<h3>Ficheros subidos</h3>  
<ul id="file-list"><li>(no hay ficheros todavía)</li></ul>
```

Ejemplo (cont)

```

var filesUpload = document.getElementById("files-upload"),           fileList
= document.getElementById("file-list");

filesUpload.onchange = function () {
    traverseFiles(this.files);
};

function traverseFiles(files) {
    var li,file,fileInfo;
    fileList.innerHTML = "";

    for (var i = 0, il = files.length; i < il; i++) {
        li = document.createElement("li");
        file = files[i];
        fileInfo = "<div><strong>Fichero:</strong> " + file.name + "</div>";
        fileInfo += "<div><strong>Tamaño:</strong> " + file.size + " bytes</div>";
        fileInfo += "<div><strong>Tipo:</strong> " + file.type + "</div>";
        li.innerHTML = fileInfo;
        fileList.appendChild(li);
    };
};

```



Pongámoslo en práctica

- Examina y analiza el ejemplo TipoArchivo.html



Pongámoslo en práctica- Ficheros en un directorio.

- Modifica el script para mostrar los ficheros en un directorio.
- Tip en la siguiente diapositiva

Tip



```
var _getAllFilesFromFolder = function(dir) {  
  
    var filesystem = require("fs");  
    var results = [];  
  
    filesystem.readdirSync(dir).forEach(function(file) {  
  
        file = dir+'/'+file;  
        var stat = filesystem.statSync(file);  
  
        if (stat && stat.isDirectory()) {  
            results = results.concat(_getAllFilesFromFolder(file))  
        } else results.push(file);  
  
    });  
  
    return results;  
  
};
```

8

WEB WORKERS

Web Workers



- Los WebWorkers permiten implementar procesos locales asíncronos para mejorar el nivel de respuesta de la interfaz de usuario.
 - https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers
- Para crear un worker se asigna a una variable una llamada al objeto **worker**, que recibe el nombre de **un fichero JavaScript** con la tarea a ejecutar.

```
var worker = new Worker("tarea.js");
```

- El worker dispone de un evento **postmessage** para **transmitir** información de vuelta al mecanismo de invocación
- El objeto implicado en la transmisión de los datos en ambos sentidos es el propio objeto event que se envía/recibe en los métodos de llamada y recepción.

Ejemplo

- En el archivo principal definimos un worker (tarea.js) y le enviamos un **mensaje**

```
var worker = new Worker("tarea.js");

//Esto establece el parámetro que se pasa al worker
worker.postMessage("Datos");

// Cuando el proceso termina se pasa por este método
worker.onmessage = function (event) {
    var message = "Mensaje del Worker: " + event.data;
    // Actualiza la interfaz de usuario
    document.getElementById("MensajeWorker").innerHTML = message;
}
```

Ejemplo (cont)

- En el archivo de worker (tarea.js) estamos a la escucha de mensajes (**onmessage**) y lo procesamos con una función de callback (**mensajeVuelta**)

```
// Asignamos la función a ejecutar
onmessage = mensajeVuelta;

function mensajeVuelta(event) {
  console.log('mensajeVuelta',event);
  // El objeto event transporta la información en su propiedad data
  if (event.data == "Datos") {
    setTimeout(function() {
      postMessage("Datos de vuelta");
    }, 5000);
  } else {
    // Error intencionado
    1/x;
  }
}
```

9

WEB SOCKETS

Web Sockets

- Es un API de nueva implantación, todavía en fase de especificación
 - [https://
developer.mozilla.org/en-US/docs/Web/API/WebSockets_API](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API)
- Permite mantener una conexión abierta con un servicio web, de forma que, cada vez que exista un cambio de estado en el servicio, podamos recibir una notificación y obrar en consecuencia.
 - Usa el protocolo WS://
- Podemos pensar en ellos como en una conexión telefónica abierta entre nuestra página cliente y el servicio Web.
- Su mecanismo de funcionamiento sería similar a lo siguiente:

```
// Creamos una variable para almacenar las referencias  
var socket = new WebSocket("ws://yourdomain/yourservice");
```

Web Sockets

- A continuación programamos una función para recibir las notificaciones del servicio asignándola al evento **onopen**:

```
// Asignamos el evento onopen
socket.onopen = function(){
    alert("Socket abierto con servicio Ajedrez "on-line");
}
```

- Podemos enviar mensajes al servicio mediante el método **postmessage**:

```
socket.postMessage("Blancas mueven: d4");
```

- Y podemos recibir mensajes registrando otro manejador

```
socket.onmessage = function(event) {
    alert("Respuesta de las negras: " + event.data);
};
```

Web Sockets

- Se trata de una de las API más esperanzadoras en cuanto a las posibilidades potenciales
- Pero, al mismo tiempo, en palabras de uno de los responsables del estándar es “una de las más arriesgadas, de todo el nuevo paquete de API propuestas para el estándar HTML5”
- Debido a esto, el nivel de seguridad debe alcanzar unos mínimos establecidos antes de ser implantado y soportado por las nuevas versiones de los navegadores.

10

API HISTORY

API History

- El historial de un navegador almacena una referencia a todas las páginas web (URLs) visitadas por el usuario durante una sesión
- El navegador ofrece mecanismos JavaScript para el desplazamiento por páginas visitadas a través de la propiedad **history** que pertenece al objeto **window** (las flechas del navegador implementan este comportamiento)
- Desde ese objeto window (o incluso, sin mencionarlo explícitamente) podemos simular las flechas de navegación con los siguientes métodos y propiedades:
 - **back()**: retrocede un paso en el historial (flecha izquierda del navegador).
 - **forward()**: avanza un paso en el historial (flecha derecha del navegador).
 - **go(steps)**: avanza o retrocede en el historial la cantidad de pasos especificados.
 - **Length**: total de elementos en el historial

Simulación de URLs

- Desde la aparición de AJAX la navegación no es tan simple como la indicación de una URL, al haber fragmentos (URL, en realidad) que se incrustan en otra página.
- La novedad es poder añadir URL "falsas" al historial de navegación y poder controlar mejor la actividad del usuario.
- Se han creado nuevos métodos al objeto History para poder manejar esta situación:
 - **pushState**(estado, título, url): añade una página falsa
 - **replaceState**(estado, título, url)
 - **state**: si una página falsa ha sido visitada
 - **popstate**: se genera cada vez que una página falsa se crea mediante código

Ejemplo

- Las URLs generadas usando métodos como `pushState()` son URLs falsas en el sentido de que los navegadores nunca controlan su validez.

```
$(document).ready(function () {  
    $('#url').click(function () {  
        $('#cajadatos').html('La URL es pagina2');  
        history.pushState(null, null, 'pagina2.html');  
    });  
});
```



Pongámoslo en práctica

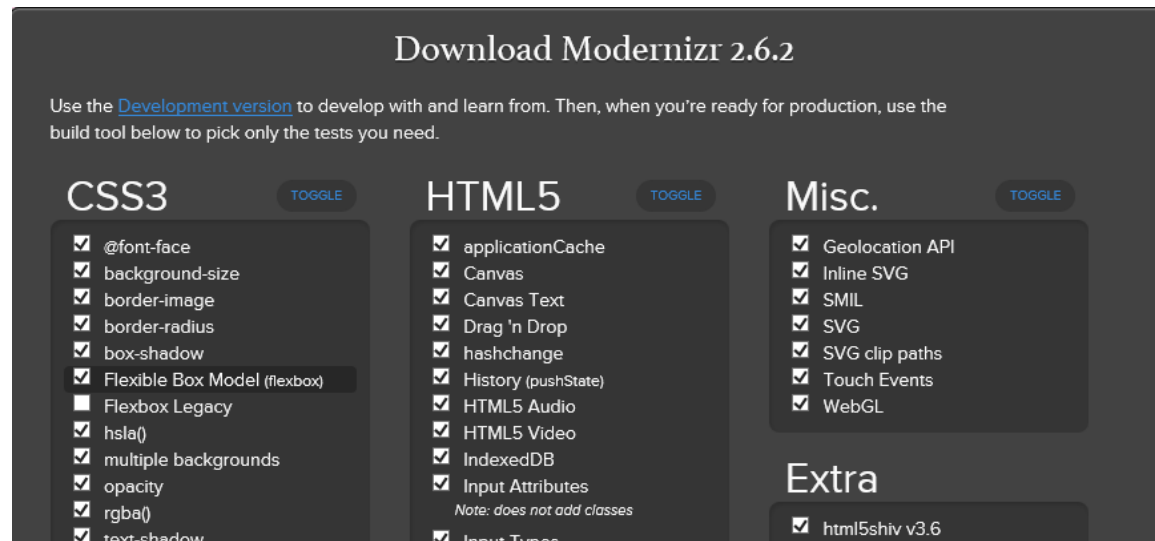
- Añade un botón “guardar” a la página y asocia el evento onclick con añadir una nueva versión a la URL de la página
- Tip: observa el comportamiento de jsbin.com para ello

A1

LIBRERÍAS AUXILIARES

Modernizr

- Librería muy completa para garantizar al desarrollador el soporte de los estándares HTML5, CSS y API's de JavaScript.
- Las últimas versiones y la documentación se encuentra en su sitio oficial: <http://modernizr.com/>
- Cuenta siempre con versiones de desarrollador y de producción.
- Podemos personalizarla para solo descargar un subconjunto de la librería que incluye aquellas partes que nos interesa detectar



Modernizr - Características

- Permite establecer una estrategia de comportamiento para navegadores antiguos (o incluso en los nuevos, para aquellas características que no estén soportadas)
- Recordemos que el estándar insiste en que NO se compruebe la versión del navegador sino el soporte que ofrece.
- Cuando Modernizr detecte que una característica no está soportada, podemos utilizar un nuevo concepto de fallback llamado **Polyfill**.
- Un Polyfill soluciona un problema concreto para dar soporte de lo que no existe, gracias a sus librerías

Ejemplo de uso de Modernizr

- La referencia a la librería se situa en <head>
- Crea un objeto JavaScript llamado Modernizr, al que podemos interrogar por la funcionalidad necesaria
- Controla tanto aspectos de elementos del DOM, como de CSS 3 o de API de JavaScript

```
<script src="modernizr.js"></script>
```

```
if (Modernizr.canvas) {  
    alert("This browser supports HTML5 canvas!");  
}
```

```
Modernizr.load({  
    test: Modernizr.canvas,  
    nope: 'http://flashcanvas.net/bin/flashcanvas.js'  
});
```


Librerías adicionales

> PolyFills

- > Lo que sigue (y merece la pena incluirlo), es la lista más completa de PolyFills que podemos encontrar hasta el momento (tomada directamente en inglés del sitio oficial que la renueva periódicamente)

> SVG

- > [svgweb](#) (Brad Neuberg)
Fallback via Flash
- > [Raphaël](#) (Dmitry Baranovsky)
API abstracta. Añade características. Fallback para IE via VML
- > [Ample](#) SDK (Sergey Ilinsky)
- > [canvg](#) (Gabe Lerner)
Escribe SVG en canvas. Buena para Android

Lista de PolyFills

> SVG

> [SVG Boilerplate](#)

En Alpha y con bugs, pero maneja multiples shims concurrentes de SVG

> [SIE SVG library](#) (dhrname). Fallback a VML para viejos IE

> [dojo gfx](#) (Eugene Lazutkin, Kun Xi, Chris Mitchell)

Fallback via VML, Canvas, Silverlight y Flash

> [fabric.js](#) (kangax)

Puede visualizar SVG via canvas

> [inline SVG polyfill](#) (mstolfoort)

> Canvas

> [FlashCanvas](#) (Shinya Muramatsu)

Mejora 33 veces el rendimiento ([33x better performance](#)) respecto a excanvas

> [excanvas](#) (google, erik arvidsson)

> [slcanvas project](#) ([Original Silverlight bridge](#), [demo page](#))

> [canvas-text](#)

Solo necesario para Texto Canvas en IE, realmente

> [fxCanvas](#)

> [Kinetic.js](#) (Eric Drowel)

Lista de PolyFills (II)

> Web Storage (LocalStorage y sessionStorage)

- > [storage](#) polyfill (Joshua Bell)
- > [storage](#) polyfill (Remy Sharp)
- > [sessionstorage](#) (Andrea Giammarchi)
- > [Amplify.js](#) (appendTo)
- > [YUI3 CacheOffline](#) (YUI team)
- > [textStorage.js](#) (sofish) Usa localStorage API, fallback para IE6+

> Soluciones sin API -HTML5

- > [ssw](#) (Matthias Schäfer)
- > [\\$.store](#) (Rodney Rehm)
- > [lawnchair](#) (Brian Leroux)
- > [webstorage](#) (Ryan Westphal)
- > [store.js](#) (Marcus Westin)
- > [PersistJS](#) (Paul Duncan)
- > [Squirrel.js](#) (Aaron Gustafson)

Librerías adicionales (Lista de Polyfills)

> Sectioning Elements

- > [html5shiv](#) (afarkas, Jon Neal y la comunidad)
- > Permite usar todos los nuevos elementos en casi todos los navegadores anteriores
- > Ocupa solo ¡4k!
- > La librería que incluyo en la documentación contiene el proyecto completo (por si alguien quiere modificarlo...☺)
- > Se anuncia como la librería estándar"de facto" para usar el estándar sin problemas de reconocimiento del DOM
Se puede usar también para Impresión

> Otras dignas de mención

- > [Linq2IndexedDB](#), [Geolocation-API-Polyfill](#), [HTML5-History-API](#), [FileAPI](#), [web worker api shim](#)
- > La lista completa de Polyfills, incluye más de 100 entradas con alternativas para todos los estándares
 - > <https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills>

Recursos relacionados

- Sitio de la revista DNM con noticias de desarrollo:
www.dnmpius.net
- MSDN y HTML5:
<http://msdn.microsoft.com/en-us/ie/aa740469>
- Internet Explorer 10 Guide for Developers:
<http://msdn.microsoft.com/en-us/ie/gg192966>
- Demos de Mozilla FireFox con HTML 5:
<https://demos.mozilla.org/es/>
- HTML 5 Rocks (Para formación y demos con Chrome):
<http://www.html5rocks.com/en/>
- Opera y HTML 5:
<http://www.opera.com/docs/specs/presto25/html5/>
- Safari y HTML 5:
<http://www.apple.com/html5/>



Almacenando los datos de BananaTube en local

- Implementa el almacenamiento en local de los videos, sus comentarios y calificaciones presentes en la vista cargada del muro
- El almacenamiento se debe poner en marcha en cuanto no exista comunicación con el servidor.



[...] netmind

WeKnowIT

Barcelona

Madrid

C. Almogàvers, 123
08018 Barcelona
Tel. 93 304.17.20
Fax. 93 304.17.22

Plaza Carlos Trías Bertrán, 7
28020 Madrid
Tel. 91 442.77.03
Fax. 91 442.77.07

www.netmind.es



MINISTERIO
DE ENERGÍA, TURISMO
Y AGENDA DIGITAL

red.es



UNIÓN EUROPEA

Fondo Social Europeo
"El FSE invierte en tu futuro"