Visor Avanzado

JAUME CLOQUELL CAPO, BARTOMEU BERNAT MESTRE CRESPÍ

Tabla de contenido

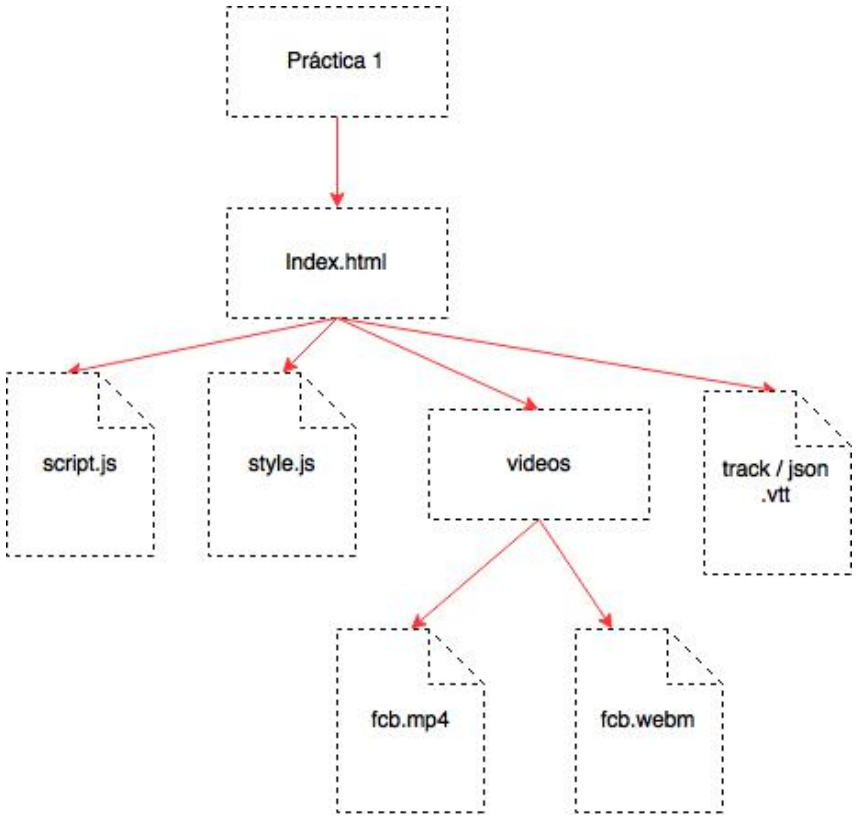
Aspectos generales

Introducción

El “Bareto de jugones” es una página web de carácter deportivo, en al que puedes visualizar los últimos partidos del FC Barcelona, así como elegir el gol que m’has te guste para poder volver a ver.
El objetivo de la práctica era diseñar un reproductor de videos HTML5 (utilizando el tag <video>) con controles personalizados y incorporar funcionalidades basadas en canvas i text trac WEBVTT.

Estructura

La web se inicia en una página principal (index.html) que es desde donde importaremos el resto de librerías i funcionalidades necesarias para la implementación de la práctica. En dicho documentos importaremos tanto los archivos javascript (script.js y bootstrap.min.js), los archivos de estilo (style.js y bootstrap.min.css) y finalmente videos y los tracks.

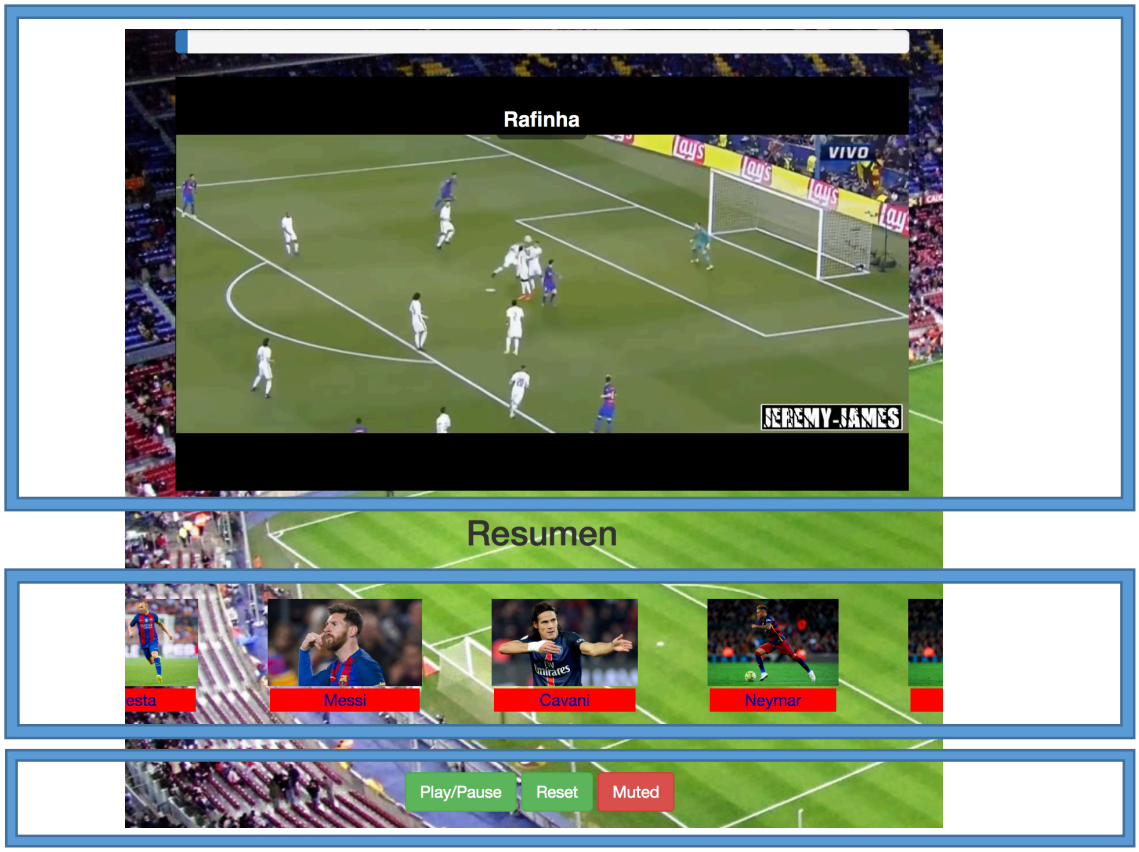


Funcionalidades

Las funcionalidades más relevantes se describirán a continuación:

Controles de video personalizados – F01

El diseño del reproductor del video consta de dos partes. En primer lugar encontramos el video junto con una barra de progreso, encima. Esta barra nos da una estimación del tiempo restante del video.



En segundo lugar nos encontramos los controles del video, que se encuentran por debajo del resumen de los goles y poseemos las funcionalidades de para o reproducir el volumen, volver a inicial el video, ponerlos en silencio y finalmente modificar el volumen.

Funcionalidades

Play/Pause

Esta función pausa el video o lo reanuda según su estado actual utilizando las funciones *javascript* para los elementos de video html5 *play()* y *pause()*.

```
function playPause() {

    if (video.paused)
        video.play();
    else
        video.pause();
}
```

Visualizar el tiempo de reproducción restante

```
//inicializamos las variables
progress = document.getElementById("progress");
chapterMenuDiv = document.querySelector("#chapterMenu");
trackElems = document.querySelectorAll("track");
video = document.querySelector("#myVideo");

//Creamos un evento que nos cargue caga vez que el tiempo del video cambia
//lo utilizamos para actualizar nuestra barra de progresso
video.ontimeupdate = function () {
    var total = (video.currentTime * 100) / video.duration;
    progress.style.width = total + "%";
};
```

Para visualizar el tiempo de reproducción lo que hacemos, es crear un evento que se inicialice cada vez que el tiempo del video se modifique. Utilizamos funcionalidades propias del tag <video>.

Una vez que lanza el evento, mediante una simple operación aritmética, obtenemos el valor actual y a partir de él, obtenemos el porcentaje total de la duración completa del video. Una vez tenemos el tanto por ciento, solo hace falta actualizar la barra de progreso, al estado actual, modificando su valor.

Modificar el volumen

Estas funciones se encargan simplemente de silenciar directamente el video, modificando el valor muted con un boolean o en caso de añadir o reducir volumen, se añade un valor definido entre un rango de 0 a 100.

```
function muted() {

    if (!video.muted) {
        video.muted = true;
    } else {
        video.muted = false;
    }
}

function setVolumen(value) {
    video.volume = value / 100;
}
```

Subtítulos con Text Track WebVTT – F02

Como se ha comentado en el apartado anterior, los elementos *<track>* nos permiten añadir subtítulos y metadatos a los elementos *<video>* de *html5*. En este caso en formato *.vtt*.

Para ello hemos utilizado un archivo como el que se muestra a continuación para el video de la web.

```
WEBVTT

1
00:00:00.500 --> 00:00:01.000 line:6% size:110%
<v Proog>La recibe Sergio</v>

2
00:00:01.000 --> 00:00:02.000 line:6% size:110%
<v Proog>Mascherano</v>

3
00:00:02.000 --> 00:00:03.000 line:6% size:110%
<v Proog>Rafinha</v>
```

Para añadirlo en el video, lo importamos directamente dentro del tag video con el parámetro “DEFAULT” para que así el navegador lo elija directamente al iniciar el video. Los comentarios tienen su propio estilo, para que se vean más grandes y en el lado superior del video.

```
<div>
    <video id="myVideo" preload="metadata" crossOrigin="anonymous"
    style="width: 50%" autoplay>
        <source <source src="./gdie02/video/fcb.webm"
        type="video/webm">
        <source src="./gdie02/video/fcb.mp4" type="video/mp4">

        <track label="Deutsch subtitles" kind="subtitles"
        src="./gdie02/track.vtt" default>
        <track label="English chapters" kind="chapters"
        src="./gdie02/json.vtt">
    </video>
    <h2>Resumen</h2>
</div>
```

Resumen de los goles –F03

```
WEBVTT

chapter-1
00:00:10.000 --> 00:00:12.000
{
    "description": "Suarez",
    "image":
    "https://www.100xcientofutbol.com/wp-content/uploads/092915-
    is-Suarez-pi-ssm.vresize.1200.675.high_.69.jpg"
}
```

Para poder añadir los resúmenes de los goles y que una vez que cliquemos encima, este nos lleva al momento exacto del video, hemos utilizado otro archivo track para ello pero con un estructura un poco

diferente. En el hemos añadido, un objeto json formato por 2 parámetros, url del video y quien realiza el gol.

La implementación de dicha funcionalidad la dividimos en 3 partes:

1. Carga del archivo vtt
2. Obtención de datos
3. Generación de las imágenes en caliente en la web

La primera parte, se basa en obtener el archivo vtt rápidamente al iniciar la página web. Para ello realizaremos un recorridos secuencial, en la función buildChapterMenu().

```
    // Obtenemos todos los tracks
    tracks = video.textTracks;
    buildChapterMenu('chapters');
};

function buildChapterMenu(kind) {
    // localizamos los tracks con kind="chapters"
    for (var i = 0; i < tracks.length; i++) {
        var track = tracks[i];
        var trackAsHtmlElem = trackElems[i];
        if ((track.kind === kind)) {
            // current chapter while the video is playing
            track.mode = "showing";
            trackAsHtmlElem.addEventListener('load', function (e) {
                displayChaptersMarkers(track);
            });
        }
    }
}
```

En la función recorremos los tracks y solo obtendremos aquellos que su kind tenga el valor de “chapter” para descartar el otro archivo vtt.

Una vez que accedemos al track especifico, creamos una función que nos lanzar aun evento cuando el track específico esté listo para ser usado. Este evento lo tuvimos que realizar porque en ocasiones la aplicación fallaba al intentar acceder a un elemento, cuyo valores aun no estaban cargados en memoria.

```
function displayChaptersMarkers(track) {
    var cues = track.cues;

    // We should not see the cues on the video.
    track.mode = "hidden";

    for (var i = 0, len = cues.length; i < len; i++) {
        var cue = cues[i];
        var cueObject = JSON.parse(cue.text);
        var description = cueObject.description;
        var imageFileName = cueObject.image;
        var imageURL = imageFileName;
        // añadimos las imagenes
        var figure = document.createElement('figure');
        figure.classList.add("img");
        console.log(cue);
        figure.innerHTML = "<img onclick='jumpTo(" + cue.startTime + ");'
        class='thumb' src='" + imageURL + "'><figcaption class='desc'>" +
        description + "</figcaption></figure>";
        chapterMenuDiv.insertBefore(figure, null);
    }
}
```

La segunda parte, se basa en como obtenemos los datos una vez que se haya realizado el evento y por tanto, ejecutemos la función displayChpatersMarjers().

Como podemos observar en la imagen de la izquierda obtenemos el json del track especifico haciendo uso de la función JSON.PARSE() y con ello accedemos a los valores definidos que son: url de la imagen y una descripción.

Finalmente, en la tercera parte, se realiza la operación de generación de las imágenes y con ello, una función específica para que al hacer un click, el visualizador modifique el tiempo actual del video.

Simplemente se trata de añadir un pequeño código HTML5, cuyos elementos son un img con la dirección obtenida anteriormente, y el nombre del goleador, que lo obtendremos a partir de la descripción.
Hay que destacar que gracias a esta funcionalidad, podríamos modificar el video, sy con ello el número de goles sin necesidad de introducir ni modificar ninguna línea de código.

El evento onclick() definido en la imagen, nos ayuda a detectar cuando el usuario realiza un click conel ratón encima de la imagen. Cada vez que se ejecuta este evento, llamaremos a la función jumpTo() donde simplemente realizamos una función setTime() con el tiempo definido obtenido e partir del json del track.

Codificación

Los videos se obtuvieron de Youtube. Para la práctica se han utilizado dos formatos distintos de video a fin de poder ejecutarla en el número máximo de navegadores posibles.
Esos formatos son:

- mp4
- webm

Evaluación en diferentes plataformas

	Safari	Firefox	Chrome
F01			
F02			
F03			

El único problema, lo encontramos en Firefox, donde no podemos cargar el archivo vtt específico para la implementación de la funcionalidad 03.

Opinión personal

La práctica, podría mejorar bastante en la parte de responsive. A pesar de utilizar las librerías de bootrstap, el visualizador de resúmenes de goles no se adapta correctamente a los dispositivos móviles. A pesar de ello, hemos conseguido que funciona casi a 100% en los navegadores más usados hoy en día.