Pràctica d'integració de continguts multimèdia

Gracies als nous elements afegits amb HTML5 (audio i video) es possible crear reproductors personalitzats fent servir nomes les tecnologies proporcionades pels navegadors, afegint llistes de reproduccio que carreguin la informacio de fonts externes com per exemple un servidor de video extern o un fitxer de dades.

Un altre us molt frequent dels recursos multimedia es la creacio de baners publicitaris, ja que aprofitant les caracteristiques del llenguatge, en lloc de fer servir baners estatics amb codi HTML, es poden crear baners dinamics que mostrin un anunci o un altre segons les dades que s'hagin carregat (o incrustat).

En particular, els dos dels llenguatges de marcat mes populars per transmetre aquesta informacio son XML i JSON (JavaScript Object Notation). Un avantatge d'aquest ultim es que es tracta del mateix format que fa servir JavaScript i per tant una estructura de dades en JavaScript es identica a la seva representacio en JSON

Creació d'un reproductor de vídeo

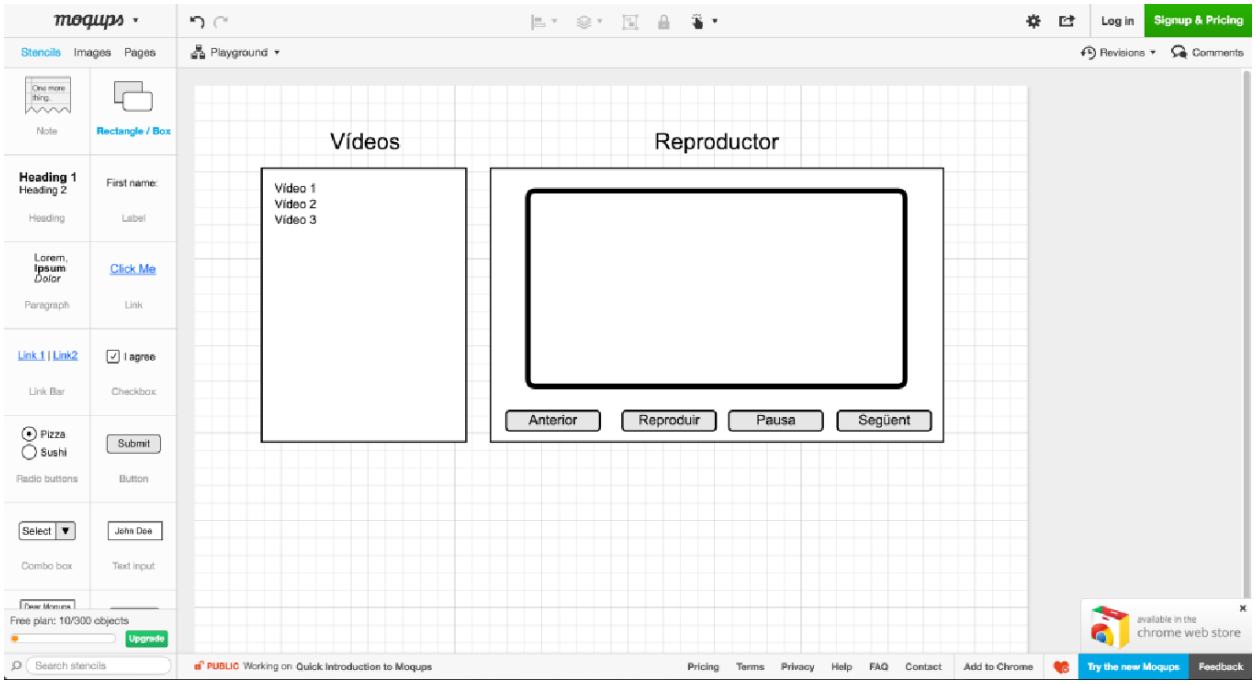
En primer lloc, es desenvolupara un reproductor de video, pas a pas, al qual s'integraran efectes de so i animacions amb CSS. Les caracteristiques d'aquest reproductor seran les seguents:

- A la seccio esquerra mostrara una llista generada dinamicament a partir d'una estructura de dades JSON (en el nostre cas, un objecte literal de JavaScript)
- A la seccio dreta mostrara una caixa on es reproduira el video i a sota els botons de reproduccio.
- En passar el cursor sobre qualsevol boto o element de la llista es reproduira un so i s'executara una petita animacio.
- En fer clic sobre un element de la llista es reproduira un so i comencara a reproduir-se el video associat.
- En fer clic sobre un boto es reproduira un so i es realitzara una accio diferent, segons el boto clicat:

  – Anterior: es desplacara el video seleccionat una posicio cap enrere, de manera que si era seleccionat el primer video, passara a seleccionarse l'ultim, i comencara la reproduccio.
  – Seguent: al contrari que el boto *Anterior*, selecciona el seguent video de la llista, i si era l'ultim, se selecciona el primer. A continuacio comenca la reproduccio.
  – Reproduir: inicia la reproduccio del video seleccionat des del comencament, i, si ja s'estava reproduint, tornara a comencar.
  – Aturar: si el video estava reproduint-se l'atura, i, si estava aturat, continua reproduint des del mateix punt.

Prototip de la interfície del reproductor

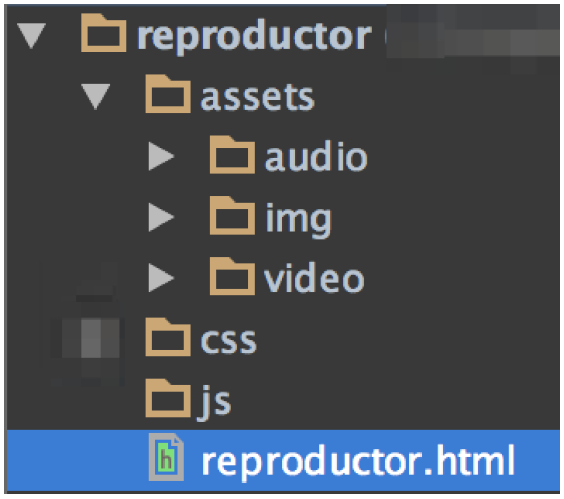
El primer que s'ha de fer abans de comencar a codificar la solucio es fer un prototip de la interficie. D'aquesta manera, a l'hora de portar a terme la implementacio tindreu clar com ha de funcionar l'aplicacio. Aquest prototip el podeu dissenyar directament sobre paper o fer servir alguna eina de *wireframes* o *mockups*. Es recomana fer servir programari especific per a la creacio d'aquests, ja que ofereixen elements neutres per crear les interficie que no distreuen del seu objectiu.



Una vegada tingueu clara la distribucio de la interficie de la vostra aplicacio podeu comencar la seguent fase de preparacio.

Estructura de directoris i preparació de recursos

El seguent pas es crear la estructura de directoris, preparar els recursos multimedia necessaris i copiar-los als directoris pertinents Es possible que l'estructura de directoris us vingui donada per les tecnologies que empreu; en aquest cas, fareu servir una estructura propia:



• El fitxer HTML es trobara a l'arrel del projecte.
• Els fitxers CSS es trobaran dins del directori */css*.
• Els fitxers amb codi JavaScript aniran dins del directori */js*.
• Els fitxers de recursos multimedia es trobaran dins del directori */assets*, i dins d'aquests:
  – Els fitxers de video dins de la carpeta */video*.
  – Els fitxers d'audio dins de la carpeta */audio*.
  – Els fitxers d'imatge dins de la carpeta */img* (encara que en aques projecte no es fara servir cap).

Pas 1: preparació de l'estructura de la pàgina HTML

Arribats a aquest punt, ja podeu comencar a codificar. Primer haureu de crear el fitxer HTML sense aplicar cap estil ni cap identificador, aixo ho fareu en el seguent pas. Nomes heu de fixar-vos en el disseny del prototip i pensar com ha d'estar estructurat el codi.

Temporalment, es fara servir una llista (amb els elements ul i li) per representar els videos que poden seleccionar-se, ja que no s'afegira la creacio dinamica fins mes endavant.
Aquesta seria una possible implementacio:

```
<head>
  <title>Reproductor</title>
</head>

<body>
  <main>
    <div>
      <h1>Vídeos</h1>
      <ul>
        <li>Vídeo 1</li>
        <li>Vídeo 2</li>
        <li>Vídeo 3</li>
```

```
    </ul>
   </div>
   <div>
    <h1>Reproductor</h1>
    <div>
     <video width="430px" height="315px" type="video/mp4"></video>
    </div>
    <div>
     <div>ANTERIOR</div>
     <div>REPRODUIR</div>
     <div>ATURAR</div>
     <div>SEGÜENT</div>
    </div>
   </div>
  </main>
</body>
```

Pas 2: afegir el full d'estil

Una vegada tingueu l'estructura llesta fareu servir el full d'estil per donar-li un format que s'ajusti al nostre prototip. El primer que heu de fer es crear un fitxer de text pla anomenat reproductor.css, que guardareu dins del directori */css*. Acontinuacio, l'enllacareu amb el document HTML afegint el seguent codi dins de l'element head:

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

Seguidament, procediu a afegir els identificadors i classes a diferents seccions del codi HTML que us facilitara l'assignacio d'estils:

- identificador per a la llista
- identificador per al visor
- identificador per al reproductor

- identificador per als controls
- classes per als botons

Freqüentment, en treballar amb elements flotants, els elements que apareixen a continuació no es mostren correctament. Una de les solutions possibles es afegir un element div buit amb una classe que apliqui l'estil CSS clear:both. Per convenció, a aquesta classe se l'anomena clear o clearfix.

Ara només resta aplicar els estils apropiats:

```
* {
  border: 1px dotted lightgray;
}

main {
  max-width: 960px;
  margin: 0 auto;
}

#llista,
#visor {
  float: left;
}

#llista {
  width: 30%;
}

#visor {
  width: 65%;
  text-align: center;
}

#controls {
  display: flex;
  justify-content: space-around;
}

.boto {
  flex-basis: 20%;
}

.clearfix {
  clear: both;
  border: none;
}
```

Fixeu-vos que s'ha afegit una vora a tots els elements del document:

```
* {
border: 1px dotted lightgray;
}
```

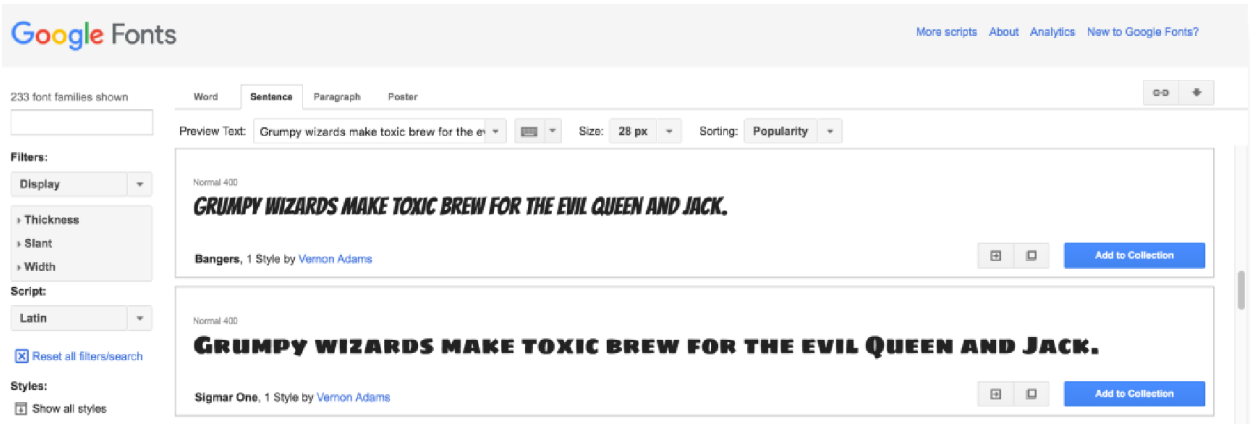
Aquesta vora s'ha afegit de manera temporal per ressaltar les delimitacions de tots els elements, i no formarà part del disseny final.

Pas 3: afegir fonts amb Google Fonts

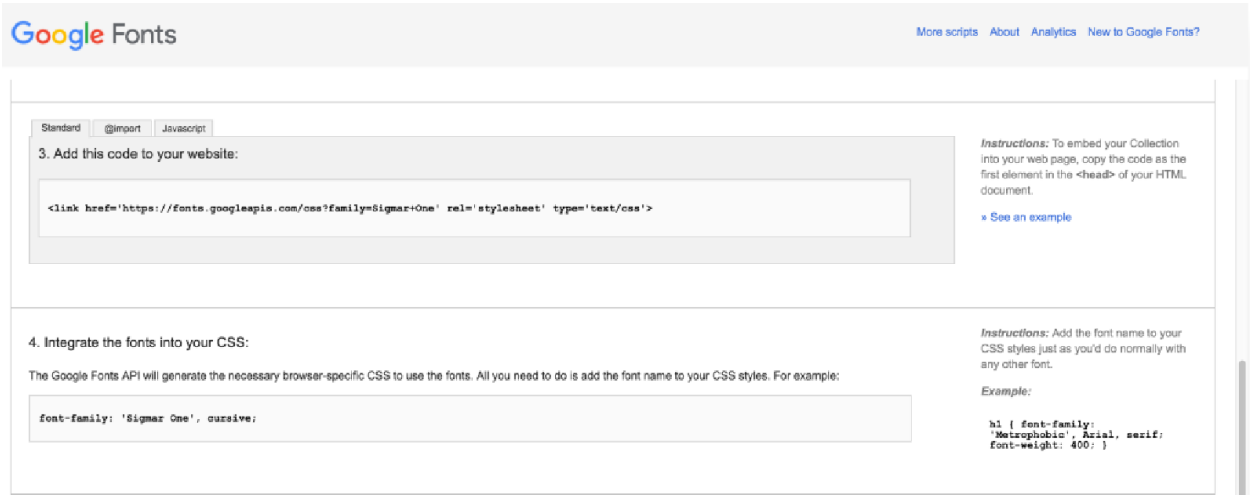
Una manera molt simple de fer els vostres titols i capçaleres més atractives es fer servir fonts externes, com les que ofereix Google Fonts.

Per afegir alguna d'aquestes fonts heu de visitar la página de Google Fonts i a continuació:

- Seleccioneu una de les fonts llistades, per exemple Sigmar One, de Vernon Adams.
- Afegiu-la a la col·lecció, fent clic al boto *Add to Collection*.



• Feu clic al boto *use*, que us portara a un'altra pagina on podeu veure més informació sobre la font, el seu pes i el codi per afegir tant al fitxer HTML i el codi CSS per fer-la servir,

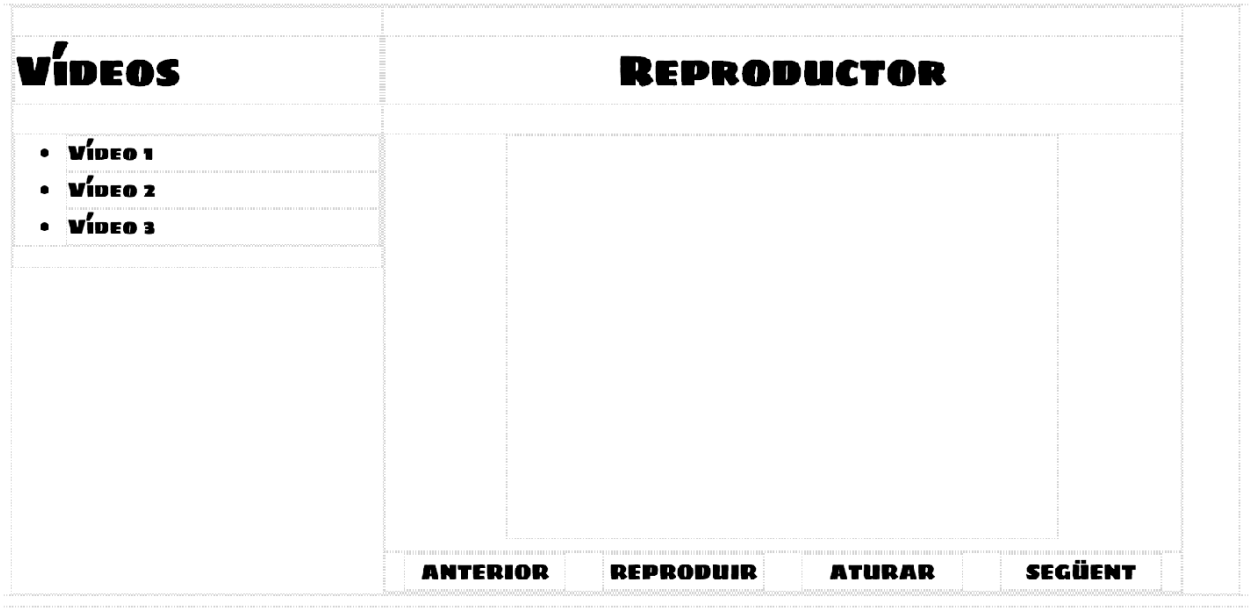


Ara que ja teniu tota la informació que necessiteu, actualitzeu el vostre fitxer HTML amb el codi obtingut de la página, i mes a mes afegeix el joc de caracteres per evitar problemes de representacio:

```
<head>
        <meta charset="utf-8">
        <title>Reproductor</title>
         <link href="css/reproductor.css" rel="stylesheet" type="text/css"/>
         <link href="https://fonts.googleapis.com/css?family=Sigmar+One" rel="stylesheet" type="text/css">
</head>
```

Per assegurar que tots els caracters de la pagina es mostren correctament s'ha d'afegir a la capcalera l'etiqueta: <meta charset="utf-8">. També s'ha de modificar el fitxer CSS per fer servir aquesta font a tots els elements, ja que no s'inclourà cap altre tipus d'element de text a banda de les capcalera i els botons, i s'obtindrà el resultat

```
* {
border: 1px dotted lightgray;
font-family: 'Sigmar One', cursive;
}
```



Pas 4: afegir icones amb Font Awesome

Per fer més clara la utilitat dels botons afegireu icones de Font Awesome. Com que precisament inclou un joc d'icones per a reproductors, facilita molt la feina. L'avantatge de fer servir icones en lloc d'imatges es que les fonts funcionen com a imatges vectorials i, per tant, poden escalar-se a la mida que necessiteu sense perdre qualitat.
El primer que heu de fer per poder emprar les icones de Font Awesome es enllacar amb el fitxer que conte el codi CSS:

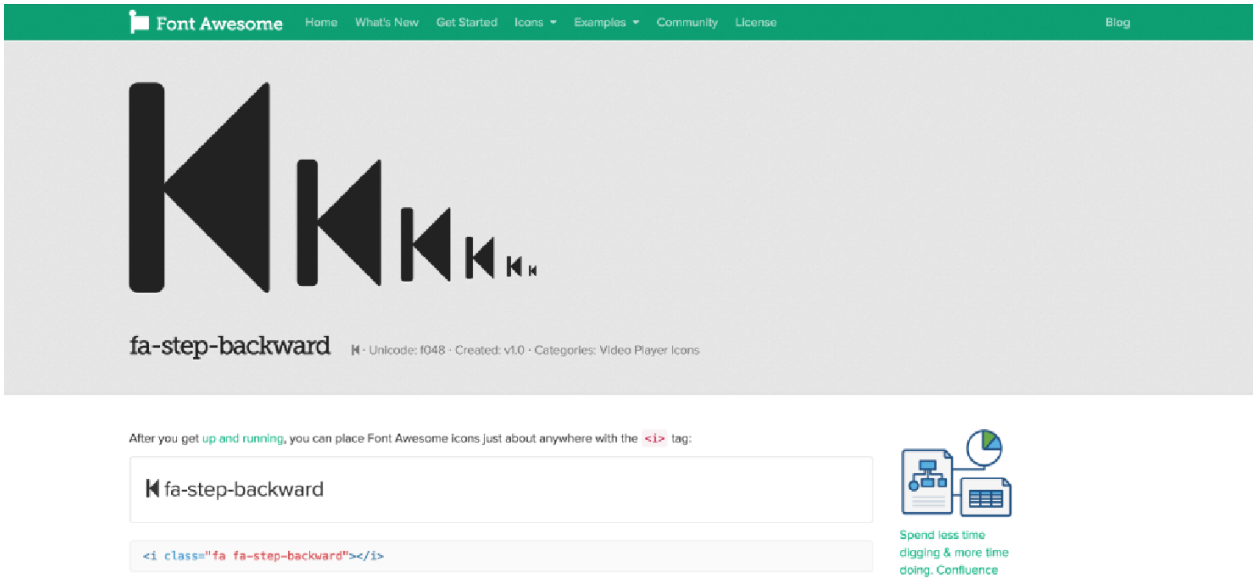
<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css" rel="stylesheet">

Tant **Google Fonts** com **Font Awesome**, a banda del fitxer CSS, realitzaran peticions extres per accedir als fitxers de fonts. Una vegada enllacat, ja podeu afegir les icones al reproductor; en aquest cas, s'ha decidit fer servir les icones:

- Step-backward
- play
- pause
- step-forward

Per afegir un boto directament al codi HTML nomes hem de cercar la icona que us interessi dins del lloc web de Font Awesome i fer-hi clic. Us portara a una altra pagina, on trobareu el codi HTML que heu de fer servir.

La icona step-backward es pot trobar en el següent enllaç: goo.gl/MGm4PD. Allà es pot trobar el codi corresponent, class="fa fa-step-backward"></i> i la visualització en diferents mides.



Reemplaceu el codi dels controls pel seguent, on s'han afegit les icones per al reproductor i un salt de linia per fer que el text quedi sempre en una nova linia:

```
<div id="controls">
 <div class="boto"><i class="fa fa-step-backward fa-3x"></i><br>ANTERIOR</div>
 <div class="boto"><i class="fa fa-play fa-3x "></i><br>REPRODUIR</div>
 <div class="boto"><i class="fa fa-pause fa-3x"></i><br>ATURAR</div>
 <div class="boto"><i class="fa fa-step-forward fa-3x"></i><br>SEGÜENT</div>
 </div>
```

Pas 5: canviar l'estil dels botons

El seguent pas consisteix en donar estil als botons per fer-los mes atractius. S'ha decidit fer servir dos tons de blau: un de clar per a l'estat normal i un de mes fosc quan el cursor sigui a sobre del boto, i aplicar cantonades arrodonides. Al fitxer CSS canvieu l'estil dels botons pel seguent:

```
.boto {
  flex-basis: 20%;
  border-radius: 10px;
  border: 2px solid #105F85;
  padding: 5px;
  color: white;
  background-color: #2897E8;
  cursor: pointer;
}

.boto:hover {
  background-color: #105F85;
}
```

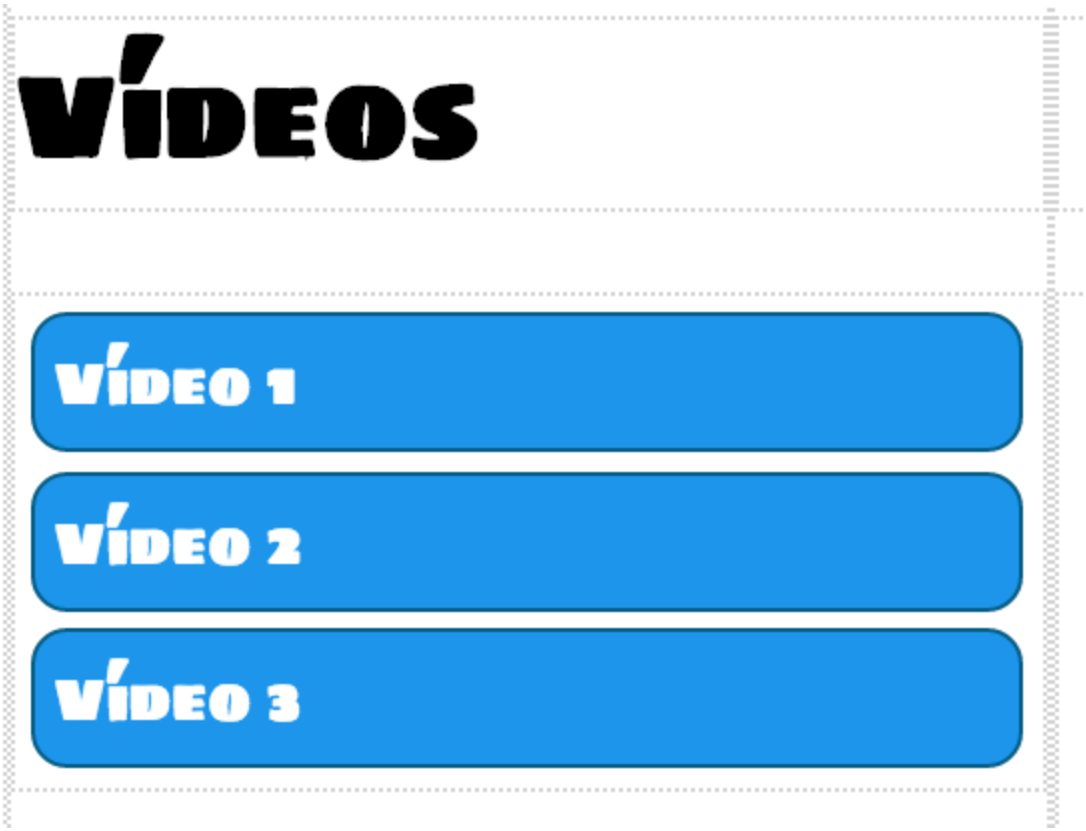
Pas 6: canviar l'estil de la llista

De manera semblant, modifiqueu l'estil de la llista aplicant el mateix tipus de vora, cantonades i colors:

```
#llista ul {
        padding: 0;
}

#llista li {
        list-style: none;
        border-radius: 10px;
        border: 2px solid #105F85;
        padding: 5px;
        color: white;
        background-color: #2897E8;
        cursor: pointer;
        margin: 5px;
}
```

```
#llista li:hover {
	background-color: #105F85;
}
```



Pas 7: refacció del full d'estil

Us haureu adonat que el codi CSS tant del boto com dels elements de la llista es practicament identic. Arribats a aquest punt, es una bona idea fer una refaccio per evitar repetir els blocs de codi. Una possible solucio seria la seguent:

```
#llista ul {
	padding: 0;
}

#llista li {
	list-style: none;
	margin: 5px;
}

.boto {
```

```
        flex-basis: 20%;
}

.boto, #llista li {
        border-radius: 10px;
        border: 2px solid #105F85;
        padding: 5px;
        color: white;
        background-color: #2897E8;
        cursor: pointer;
}

.boto:hover, #llista li:hover {
        background-color: #105F85;
}
```

Pas 8: afegir efectes amb CSS3

Ara que ja teniu definits la llista i els botons, us resta afegir un fons per a tota la pagina i fer algun petit retoc. S'ha decidit fer servir un degradat mitjancant CSS3, i per ajudar-vos a generar el codi CSS podeu fer servir una de les multiples eines en linia, per exemple la que es troba a www.colorzilla.com/gradient-editor.

Nomes heu de seleccionar el tipus de degradat, modificar-lo i copiar el codi que es genera com a propietat background de l'element body al fitxer CSS:

Cal tenir en compte que si nomes heu de preocupar-vos per navegadors moderns, el codi per generar el degradat es reduiria a body {background: linear-gradient(top, rgba(167,207,223,1) 0%,rgba(35,83,138,1) 100%)}.

```
body {
        /* Permalink - use to edit and share this gradient:
        http://colorzilla.com/gradient-editor/#a7cfdf+0,23538a+100;Blue+3d+%238 */
        background: rgb(167,207,223); /* Old browsers */
        background:-moz-lineaR-gradient(top, rgba(167,207,223,1) 0%, rgba (35,83,138,1)
        100%); /* FF3.6-15 */
        background: -webkit-Linear-gradient(top, rgba(167,207,223,1) 0%,rgba(35,83,138,1)
        100%); /* Chrome10-25,Safari5.1-6 */
        background: linear-gradient(to bottom, rgba(167,207,223,1) 0%,rgba(35,83,138,1)
        100%); /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
}
```

Hi ha un inconvenient: en casos com aquest, en que la llargaria de la pagina no es suficient per cobrir-la sencera, el degradat s'atura on acaba el contingut. Una possible solucio es fer servir

un degradat lineal, i com a color del fons de l'element html el mateix color amb el qual acaba el degradat. D'aquesta manera, la transicio es inapreciable:

```
html {
 background: rgb(35,83,138);
 }

 body {
          margin: 0;
          padding-top: 20px;
          /* Permalink - use to edit and share this gradient:
         http://colorzilla.com/gradient-editor/#a7cfdf+0,23538a+100;Blue+3d+%238 */
           background: rgb(167,207,223); /* Old browsers */
           background: -moz-linear-gradient(top, rgba(167,207,223,1) 0%, rgba(35,83,138,1) 100%);
                  /* FF3.6-15 */
           background: -webkit-linear-gradient(top, rgba(167,207,223,1) 0%,rgba(35,83,138,1) 100%);
         /* Chrome10-25,Safari5.1-6 */
          background: linear-gradient(to bottom, rgba(167,207,223,1) 0%,rgba(35,83,138,1) 100%); /*
         W3C, IE10+, FF16+, Chrome26+, Opera12+,Safari7+ */
}
```

A continuacio es canviara el color de les capcaleres i l'alineacio, i s'afegira una ombra al text:

```
h1 {
 text-align: center;
 color: white;
 text-shadow: 2px 2px 2px black;
 }
```

Seguidament s'aplica com a fons un color de tipus RGBA (color de 24 bits amb transparencia) i s'afegeix una ombra a tot el panell:

```
#llista, #visor {
 background: rgba(0, 0, 0, 0.75);

  border-radius: 10px;

  border: 1px solid black;
  padding: 5px;
  margin: 5px;

  /* Ombra */
  -webkit-box-shadow: 10px 10px 20px 0px rgba(0,0,0,0.75);
  -moz-box-shadow: 10px 10px 20px 0px rgba(0,0,0,0.75);
   box-shadow: 10px 10px 20px 0px rgba(0,0,0,0.75);
}
```

En canvi, per al reproductor de video s'afegeix un fons negre solid, de manera que queda molt clar on es reproduiran els videos:

```
video {
background:black;
margin: 10px;
}
```

Finalment, elimineu la vora que es va afegir per comprovar facilment les delimitacions de cada element:

```
  * {
font-family: 'Sigmar One', cursive;
}
```

Pas 9: enllaçar la biblioteca jQuery i afegir efectes de so

Per continuar afegireu efectes de so als vostres botons: un quan el cursor estigui a sobre i un altre en fer-hi clic.

Primerament, heu d'obtenir els sons, per qualsevol mitja; per exemple, sons descarregats de biblioteques de so amb llicencia lliure, creats per vosaltres mateixos a traves d'eines en linia o enregistrats i modificats amb Audacity.

Necessitareu obtenir dos fitxers d'audio, que haureu de copiar dins del directori *assets/audio* del vostre projecte amb aquests noms:

- **selecciona.mp3**: aquest so es reproduira en fer clic sobre un boto o element de la llista.
- **a-sobra.mp3**: aquest so es reproduira quan passem el cursor per sobre d'un boto o element de la llista.

Una vegada tingueu els vostres fitxers d'audio preparats, enllaceu la biblioteca jQuery i el fitxer amb el codi JavaScript, ja que fareu servir un fitxer extern per tenir el codi millor organitzat.

El primer que heu de fer es crear un nou fitxer de text buit dins de la carpeta *js* anomenat reproductor.js, i a continuacio enllaceu tant la biblioteca jQuery com el vostre fitxer afegint el seguent codi dins de la capcalera:

```
<script src="//code.jquery.com/jquery-3.1.1.min.js"></script>
<script src="js/reproductor.js"></script>
```

Sempre han d'enllacar-se tots els **fitxers amb codi CSS** abans que els fitxers amb JavaScript, i quan treballem amb biblioteques, aquestes han de carregarse sempre abans que els fitxers amb el codi que les fan servir.

Primerament s'afegira la classe sonor als elements que vulgueu que produeixin so en passar el cursor per sobre. Comenceu afegint-la als elements de la llista de videos:

```
<ul>
<li class="sonor">Vídeo 1</li>
<li class="sonor">Vídeo 2</li>
<li class="sonor">Vídeo 3</li>
</ul>
```

I a continuacio, afegiu tambe la classe dels botons:

```
<div id="controls">
<div class="boto sonor"><i class="fa fa-step-backward fa-3x"></i><br>ANTERIOR</div>
<div class="boto sonor"><i class="fa fa-play fa-3x "></i><br>REPRODUIR</div>
<div class="boto sonor"><i class="fa fa-pause fa-3x"></i><br>ATURAR</div>
<div class="boto sonor"><i class="fa fa-step-forward fa-3x"></i><br>SEGÜENT</div>
```

Seguidament, afegiu el seguent codi JavaScript al fitxer reproductor.js que detectara quan es dispara l'*event* mouseenter i reproduira el so:

```
$(document).ready(function () {
        var a_sobra = new Audio('assets/audio/a-sobra.mp3');
        $('.sonor').on('mouseenter', function() {
                a_sobra.play();
        });
});
```

Si proveu ara veureu que funciona, pero no tal com s'espera. El so es reprodueix, pero nomes torna a comencar quan s'ha acabat la reproduccio i no tan aviat com el cursor es col·loca sobre altre element.

Pas 10: creació d'un 'sound pool'

El so no sembla reproduir-se correctament. Aquesta es una limitacio de l'element audio d'HTML. La solucio es crear un conjunt de sons (*sound pool* en angles), de manera que cada vegada que es demani reproduir un so s'agafara el seguent de la llista.

> **Utilització de 'pools'**
> En casos de jocs que poden tenir desenes de sons reproduint-se pràcticament al mateix temps, com IOC Invaders (https://github.com/XavierGaro/ioc-invaders), o d'imatges que es repeteixen, la reutilització d'aquests recursos representa una necessitat, ja que si no s'apliquen aquestes tècniques el nombre de quadres per segon es redueix dràsticament.

Haureu de substituir el codi del fitxer reproductor.js pels seguents fragments. En primer lloc, afegireu la crida a la funcio inicialitzarSoundPool, a la qual passareu dos arguments: el nom pel qual volem identificar el so i l'URL on es troba el fitxer d'audio.

Un altre canvi es que en lloc de reproduir directament el so, ara es cridara una altra funcio, tambe propia, amb el nom reproduirSo, passant com a argument l'identificador que hem assignat al so.

```
$(document).ready(function() {
        inicialitzarSoundPool('a_sobra', 'assets/audio/a-sobra.mp3')
        $('.sonor').on('mouseenter', function() {
                reproduirSo('a_sobra');
        });
});
```

Seguidament s'afegeixen dues variables globals:

- soundPool: un objecte literal de JavaScript buit que es fara servir com a diccionari de dades.
- MAX_SOUNDS: una variable que es fara servir com si fos una constant, i que te la finalitat de limitar el nombre de sons identics que poden reproduir-se al mateix temps.

```
var soundPool = {},
MAX_SOUNDS = 10;
```

Vegeu ara la implementacio de la funcio inicialitzarSoundPool, que te la finalitat de crear l'estructura de dades necessaria per gestionar els sons i crear tots els elements d'audio necessaris:

```
function inicialitzarSoundPool(nom, url) {
        soundPool[nom] = {
        sons: [],
        actual: 0
        };
        for (var i = 0; i < MAX_SOUNDS; i++) {
                soundPool[nom].sons.push(new Audio(url));
        }
}
```

Com podeu veure, la funcio es molt senzilla, pero pot resultar una mica estranya si encara no es domina el llenguatge JavaScript.

El primer que s'ha fet es afegir a l'estructura de dades global soundPool un nou objecte literal. La clau per accedir a aquest objecte dins de soundPool sera el nom que s'ha passat com a argument, i l'objecte creat tindra dues propietats:

- **sons**: un *array* que emmagatzemara tots els elements d'audio creats per reproduir aquest so.
- **actual**: un *enter* que servira per saber quin es l'element d'audio que s'ha de reproduir en cada moment.

A continuacio s'utilitza un bucle for per crear tants elements d'audio com indica la variable global MAX_SOUNDS, i s'afegeixen a l'*array* de l'objecte creat amb el metode push:

```
soundPool[nom].sons.push(new Audio(url));
```

Fixeu-vos que tots els elements d'audio aixi creats s'afegeixen a l'objecte guardat al *sound pool* amb el nom passat com a argument i amb el fitxer d'audio especificat com a url.

Per exemple, si el nom es *a_sobra* i la url fos *assets/audio/a-sobra.mp3*, internament cada linia del bucle s'interpretaria aixi:

```
soundPool['a_sobra'].sons.push(new Audio('assets/audio/a-sobra.mp3'));
```

Per acabar, afegiu el metode reproduirSo, que agafara un nou element de so del *sound pool* cada vegada que es cridi fins a arribar a l'ultim element, on tornara a comencar:

```
function reproduirSo(nom) {
        var index = soundPool[nom].actual;
        soundPool[nom].sons[index % MAX_SOUNDS].play();
        soundPool[nom].actual++;
}
```

El mètode push és propi dels arrays de JavaScript i permet afegir un element al final de l'array

El primer que es fa es obtenir l'index del so corresponent al nom passat com a argument que toca reproduir.

A continuacio, s'aprofita la propietat del modul per evitar haver de fer la comprovacio quan s'arriba al final de l'*array*. Ates que la mida d'aquest es igual a MAX_SOUNDS, es pot fer servir l'operacio index % MAX_SOUNDS per obtenir sempre un valor dins del rang de l'*array* (entre 0 i MAX_SOUNDS - 1).

Finalment, s'incrementa el valor del so actual per deixar llest el *sound pool* per reproduir el seguent.

Pas 11: reutilització de funcions

Ara que ja funciona correctament, afegiu el so per a l'*event* click, afegint al final de la funcio cridada per ready el seguent codi:

```
inicialitzarSoundPool('selecciona', 'assets/audio/selecciona.mp3')

$('.sonor').on('click', function () {
        reproduirSo('selecciona');
});
```

Com podeu veure, una vegada implementat el sistema de *pools* d'audio es molt facil afegir qualsevol quantitat de sons. Nomes s'ha hagut de canviar el nom del so, l'URL del fitxer que es vol reproduir i l'*event* al qual es vol associar aquest nou so.

Pas 12: afegir animacions CSS

Per acabar amb els efectes dels botons i de la llista de videos s'incloura una petita animacio fent servir CSS; afegiu el seguent codi al final del fitxer reproductor.css:

```
.sonor {
        transition: 0.5s ease-out;
}

.sonor:hover {
        animation-name: zoom;
        animation-duration: 0.5s;
        animation-direction: alternate;
}
```

```
@keyframes zoom {
        from {
                transform: scale(1.0) rotate(5deg);
        }

        to {
                transform: scale(1.1) rotate(-5deg);
        }
}
```

Pas 13: creació d'elements de la llista i reproducció de vídeo

Fins ara, la llista de videos que es mostrava era fixada pel codi HTML, pero aixo no es gaire practic, ja que aquesta llista ha de ser dinamica per poder afegir, eliminar o passar al seguent video.

En un cas real, segurament aquesta llista es carregaria fent servir AJAX, pero per simplificar afegireu la informacio dels videos directament al vostre fitxer reproductor.js amb una estructura de dades de tipus JSON, i es generaran els elements de la llista via codi.

El primer que fareu es eliminar els elements de la llista de videos i afegir un identificador per facilitar la feina d'afegir-los. La llista al vostre codi HTML ha de quedar aixi:

```
<div id="llista">
        <h1>Vídeos</h1>
        <ul id="videos">
        </ul>
</div>
```

A continuacio afegiu a la declaracio de variables general del fitxer reproductor.js la informacio dels videos com un *array* d'objectes de JavaScript creats amb *notacio literal*, juntament amb una variable anomenada selecciona que inicialment tindra el valor 0 per establir com a seleccionat el primer video:

```
var soundPool = {},
        MAX_SOUNDS = 10,
        videos = [
                {
                        titol: 'El primer vídeo',
                        descripcio: 'Aquesta és la descripció del primer vídeo',
                        url: 'assets/video/video-1.mov'
```

```
        },
        {
            titol: 'El segon vídeo',
            descripcio: 'Aquesta és la descripció del segon vídeo',
            url: 'assets/video/video-2.mov'
        },
        {
            titol: 'El tercer vídeo',
            descripcio: 'Aquesta és la descripció del tercer vídeo',
            url: 'assets/video/video-3.mov'
        }
 ],
seleccionat = 0;
```

Objectes a partir de notació literal: A JavaScript es possible crear objectes de forma literal, aixo facilita la creacio d'estructures que es poden utilitzar com a diccionaris de dades, permetent tractar els elements emmagatzemats com a parells de *clau-valor*

Seguidament, afegireu una funcio per inicialitzar la llista de videos, la finalitat de la qual sera la seguent:

- Recorrer a totes les posicions de l'*array* de videos.
- Crear un element de tipus li per a cada element de l'*array*.
- Afegir la classe sonor per conservar els efectes CSS afegits anteriorment.
- Afegir un identificador únic que estarà format pel mot video- i l'índex que li correspongui a l'element de l'array.
- Afegir el text a mostrar, que es correspondrà amb el valor del titol .
- Afegir l'atribut title amb el valor de descripcio ; això fa que en posar el cursor sobre l'element es mostri la descripció del vídeo.
- Afegir el nou element a la llista de videos .
- Afegir la detecció de l'event click per establir aquest vídeo com el seleccionat i iniciar la seva reproducció.

Una vegada es té clar en què consisteix la implementació, fent servir jQuery trobareu que és pràcticament més simple que redactar-la, ja que per a cada una d'aquestes accions hi ha un mètode o funció de jQuery que facilita la tasca. Afegiu el següent codi al final del fitxer reproductor.js:

```
function inicialitzarLlistaVideos() {
    $.each(videos, function (index, value) {
        // Creem un nou node de tipus LI fent servir jQuery
        var $node = $('<li></li>');

        // S'afegeix la classe sonor
        $node.addClass('sonor');
```

```
                // S'afegeix un identificador únic basat en el seu índex
                $node.attr('id', 'video − '+index);

                // S'afegeix el contingut de l'element
                $node.text(value.titol);

                // S'afegeix la descripció per mostrar quan deixem el cursor a sobre uns segons
                $node.attr('title', value.descripcio);

                // S'afegeix el node a la llista de vídeos
                $('#videos').append($node);

                // S'afegeix la detecció del esdeveniment click per establir aquest vídeo com el
                seleccionat i iniciar la reproducció
                $node.on('click', function() {
                        seleccionat = index;
                        reproduirVideo();
                })
        });
}
```

Ara que ja teniu la funció que inicialitza la llista de vídeos només cal cridar-la. Haureu d'afegir la crida a aquesta funció just abans de la inicialització dels pools, perquè si ho afegiu després els elements de la llista no quedaran enllaçats amb aquests:

```
$(document).ready(function () {
        inicialitzarLlistaVideos();

 function reproduirVideo() {
         var video = videos[seleccionat];
         $('video').attr('src', video.url);
         $('video').get(0).play();
 }
```

Aquesta funcio es molt simple: primer s'agafen les dades del video seleccionat de l'*array*, a continuacio se selecciona l'element video i es canvia el seu atribuit src a l'URL del video.

Fixeu-vos en un petit detall: el metode play no pertany a jQuery, sino que forma part de l'element video de la pagina, aixi que primer s'ha de cridar el metode get(0), que retorna el primer element de tipus video que es trobi a la pagina, i a continuacio cridar el seu metode play per reproduir el video.

Finalment, només resta implementar la funció reproduirVideo i obtindreu un resultat com:

Pas 14: gestió del reproductor a través dels botons

Ja gairebe heu enllestit el vostre reproductor i nomes cal afegir la funcionalitat als botons. El primer que haureu de fer es afegir un identificador unic a cadascun d'aquests botons, modificant el codi HTML:

```
<div id="controls">
        <div id="anterior" class="boto sonor"><i class= "fa fa-step-backward fa-3x">
        </i><br>ANTERIOR</div>
        <div id="reproduir" class="boto sonor"><i class="fa fa-play fa-3x "></i><br>
        REPRODUIR</div>
        <div id="aturar" class="boto sonor"><i class="fa fa-pause fa-3x"></i><br>
        ATURAR</div>
        <div id="seguent" class="boto sonor"><i class="fa fa-step-forward fa-3x"></i>
        <br>SEGÜENT</div>
</div>
```

A continuacio, heu d'afegir la deteccio de l'esdeveniment clic per a cadascund'ells; ho podeu fer a continuacio de la inicialitzacio dels efectes de so, associant cada boto amb una funcio diferent:

```
$('#anterior').click(anteriorVideo);
```

```
$('#reproduir').click(reproduirVideo);
$('#aturar').click(aturarVideo);
$('#seguent').click(seguentVideo);
```

Aquest cop, en lloc de fer servir el metode on de jQuery, s'ha fet servir el metode click. En aquesta situacio es indiferent, i es pot fer servir l'un o l'altre indistintament. jQuery facilita una extensa llista de dreceres per escoltar *events*: click, dblclick, mousedown, etc.

Si necessiteu escoltar **multiples *events*** simultaniament heu de fer servir el metode on de jQuery. Per exemple: $('li').on('click dblclick', function() { alert("S'ha fet clic!"); });.

Una altra diferencia que es pot apreciar es que en lloc de cridar una funcio com en els exemples anteriors s'ha posat nomes el nom de les funcions corresponents. En tots dos casos es tracta del que anomenem *callback*, funcions que son cridades per altres funcions. Per exemple, quan es dispara l'*event* associat com en aquest cas, quan es finalitza un temporitzador (amb les funcions setInterval o setTimeout) o quan retorna una peticio AJAX del servidor amb exit.

El seguent pas es implementar aquestes funcions. Com que la funcio reproduirVideo s'ha implementat anteriorment, nomes cal codificar les noves:

```
function aturarVideo() {
        var video = $('video').get(0);
        if (video.paused) {
                video.play();
        } else {
                video.pause();
        }
}

function anteriorVideo() {
        seleccionat--;
        if (seleccionat < 0) {
                seleccionat = videos.length - 1;
        }
        reproduirVideo();
}

function seguentVideo() {
        seleccionat++;
        if (seleccionat === videos.length) {
                seleccionat = 0;
        }
        reproduirVideo();
}
```

Vegeu amb detall aquestes funcions. En primer lloc, teniu la funcio aturarVideo. El seu comportament es diferent del de les altres, ja que ha de controlar l'estat de reproduccio; per aquesta rao, el que s'ha fet es obtenir l'element de video, i a partir de la propietat paused determinar si cal aturar-lo (video.pause()) o reproduir-lo (video.play()).

Les funcions anteriorVideo i seguentVideo son practicament identiques, nomes redueixen o augmenten el valor de l'element seleccionat, canviant a l'ultima o primera posicio respectivament si el valor es troba fora de rang, i a continuacio inicien la reproduccio del video automaticament.

Pas: 15. afegir funcionalitat al teclat

Per acabar amb aquest exemple s'afegira una funcionalitat mes: quan es premi la tecla espai, el video alternara entre pausa i reproduccio. Afegiu el seguent codi a continuacio del codi per gestionar els *events* dels botons:

```
$(document).keypress(function(evt) {
        if (evt.charCode==32) {
                aturarVideo();
        }
});
```

Com veieu, ha estat molt simple afegir aquesta funcionalitat. S'ha fet una subscripcio per escoltar l'*event* keypress de tot el document, i quan es dispara es comprova si la propietat charCode te el valor 32 (que correspon a la barra d'espai). Si es aixi, es crida aturarVideo, que l'aturara o el reproduira segons l'estat actual.

Cal tenir en compte que no tots els navegadors retornen la mateixa informacio a l'objecte *event*, sobretot en referencia als *events* de teclat i per tant es recomanable fer servir biblioteques per controlar aquest comportament.

A continuacio podeu veure el llistat complet dels fitxers que componen el reproductor comencant per reproductor.html:

```
<html>
<head>
        <meta charset="utf-8">
        <title>Reproductor</title>
        <link href="https://fonts.googleapis.com/css?family=Sigmar+One" rel="stylesheet" type="text/css">
        <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css"
      rel="stylesheet">
</head>

<body>
        <main>
                <div id="llista">
                        <h1>Vídeos</h1>
                        <ul id="videos">
                        </ul>
                </div>
                <div id="visor">
```

```
                    <h1>Reproductor</h1>
                <div id="reproductor">
                    <video width="430px" height="315px"></video>
                </div>
                <div id="controls">
                    <div id="anterior" class="boto sonor"><i class="fa fa-step-backward
                    fa-3x"></i><br>ANTERIOR</div>
                    <div id="reproduir" class="boto sonor"><i class="fa fa-play
                    fa-3x"></i><br>REPRODUIR</div>
                    <div id="aturar" class="boto sonor"><i class="fa fa-pause fa-3x"></i><br>ATURAR</div>
                    <div id="seguent" class="boto sonor"><i class="fa fa-step-forward
                    fa-3x"></i><br>SEGÜENT</div>
                </div>
            </div>
            <div class="clearfix"></div>
        </main>
</body>

</html>
```

Aquest es el contingut de reproductor.css:

```
* {
  font-family: 'Sigmar One', cursive;
}

h1 {
  text-align: center;
  color: white;
  text-shadow: 2px 2px 2px black;
}

html {
  background: rgb(35, 83, 138);
}

body {
  margin: 0;
  padding-top: 20px;
  /* Permalink - use to edit and share this gradient:
http://colorzilla.com/gradient-editor/#a7cfdf+0,23538a+100;Blue+3d+%238 */
  background: rgb(167, 207, 223);
  /* Old browsers */
  background: -moz-linear-gradient(top, rgba(167, 207, 223, 1) 0%, rgba(35, 83, 138, 1) 100%);
  /* FF3.6-15 */
  background: -webkit-linear-gradient(top, rgba(167, 207, 223, 1) 0%, rgba(35, 83, 138, 1) 100%);
  /* Chrome10-25,Safari5.1-6 */
  background: linear-gradient(to bottom, rgba(167, 207, 223, 1) 0%, rgba(35, 83, 138, 1) 100%);
  /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
}

main {
  max-width: 960px;
  margin: 0 auto;
}

video {
  background: black;
  margin: 10px;
}
```

```
#llista,
#visor {
  float: left;
}

#llista {
  width: 30%;
}

#visor {
  width: 65%;
  text-align: center;
}

#controls {
  display: flex;
  justify-content: space-around;
}

#llista ul {
  padding: 0;
  margin: 0;
}

#llista li {
  list-style: none;
  margin: 5px;
}

.boto {
  flex-basis: 20%;
}

.boto,
#llista li {
  border-radius: 10px;
  border: 2px solid #105F85;
  padding: 5px;
  color: white;
  background-color: #2897E8;
  cursor: pointer;
}

.boto:hover,
#llista li:hover {
  background-color: #105F85;
}

.clearfix {
  clear: both;
  border: none;
}

#llista,
#visor {
  background: rgba(0, 0, 0, 0.75);
  border-radius: 10px;
  border: 1px solid black;
  padding: 5px;
  margin: 5px;
  /* Ombra */
```

```
  -webkit-box-shadow: 10px 10px 20px 0px rgba(0, 0, 0, 0.75);
  -moz-box-shadow: 10px 10px 20px 0px rgba(0, 0, 0, 0.75);
  box-shadow: 10px 10px 20px 0px rgba(0, 0, 0, 0.75);
}

.sonor {
  transition: 0.5s ease-out;
}

.sonor:hover {
  animation-name: zoom;
  animation-duration: 0.5s;
  animation-direction: alternate;
}

@keyframes zoom {
  from {
    transform: scale(1.0) rotate(5deg);
  }
  to {
    transform: scale(1.1) rotate(-5deg);
  }
}
```

I finalment, el contingut de reproductor.js:

```
$(document).ready(function() {
  inicialitzarLlistaVideos();

  inicialitzarSoundPool('a_sobra', 'assets/audio/a-sobra.mp3')

  $('.sonor').on('mouseenter', function() {
    reproduirSo('a_sobra');
  });

  inicialitzarSoundPool('selecciona', 'assets/audio/selecciona.mp3')

  $('.sonor').on('click', function() {
    reproduirSo('selecciona');
  });

  $('#anterior').click(anteriorVideo);
  $('#reproduir').click(reproduirVideo);
  $('#aturar').click(aturarVideo);
  $('#seguent').click(seguentVideo);
});

var soundPool = {},
  MAX_SOUNDS = 10,
  videos = [{
    titol: 'El primer vídeo',
    descripcio: 'Aquesta es la descripció del primer vídeo',
    url: 'http://vid456.photobucket.com/albums/qq284/Mascasesos/video-1_zpsr5deccka.mp4'
  }, {
```

```
    titol: 'El segon vídeo',
    descripcio: 'Aquesta es la descripció del segon vídeo',
    url: 'http://vid456.photobucket.com/albums/qq284/Mascasesos/video-2_zpsirkgj4lj.mp4'
  }, {
    titol: 'El tercer vídeo',
    descripcio: 'Aquesta es la descripció del tercer vídeo',
    url: 'http://vid456.photobucket.com/albums/qq284/Mascasesos/video-3_zpsqey3tss5.mp4'
  }],
  seleccionat = 0;

function inicialitzarSoundPool(nom, url) {
  soundPool[nom] = {
    sons: [],
    actual: 0
  };

  for (var i = 0; i < MAX_SOUNDS; i++) {
    soundPool[nom].sons.push(new Audio(url));
  }
}

function reproduirSo(nom) {
  var index = soundPool[nom].actual;
  soundPool[nom].sons[index % MAX_SOUNDS].play();
  soundPool[nom].actual++;
}

function inicialitzarLlistaVideos() {
  $.each(videos, function(index, value) {
    // Creem un nou node de tipus LI fent servir jQuery
    var $node = $('<li></li>');

    // Afegim la classe sonor
    $node.addClass('sonor');

    // Afegim un identificador únic basat en el seu index
    $node.attr('id', 'video-' + index);

    // Afegim el contingut del element
    $node.text(value.titol);

    // Afegim la descripció per mostrar quan deixem el cursor a sobre uns egons
    $node.attr('title', value.descripcio);

    // Afegim el node a la llista de vídeos
    $('#videos').append($node);
```

```
    // Afegim la detecció del esdeveniment click per establir aquest vídeo com el seleccionat i
iniciar la reproducció
    $node.on('click', function() {
      seleccionat = index;
      reproduirVideo();
    })
  });
}

function reproduirVideo() {
  var video = videos[seleccionat];
  $('video').attr('src', video.url);
  $('video').get(0).play();
}

function aturarVideo() {
  var video = $('video').get(0);

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

function anteriorVideo() {
  seleccionat--;
  if (seleccionat < 0) {
    seleccionat = videos.length - 1;
  }
  reproduirVideo();
}

function seguentVideo() {
  seleccionat++;
  if (seleccionat === videos.length) {
    seleccionat = 0;
  }
  reproduirVideo();
}
```

Per finalitzar, podeu veure l'aspecte final del reproductor en funcionament:


Vídeos
El primer vídeo
El segon vídeo
El tercer vídeo
Reproductor
Anterior
Reproduir
Aturar
Següent