P1: VISOR INTERACTIVO

How to cook Chili

21753 - Gestión y Distribución de la Información Empresarial

Curso 2020/21

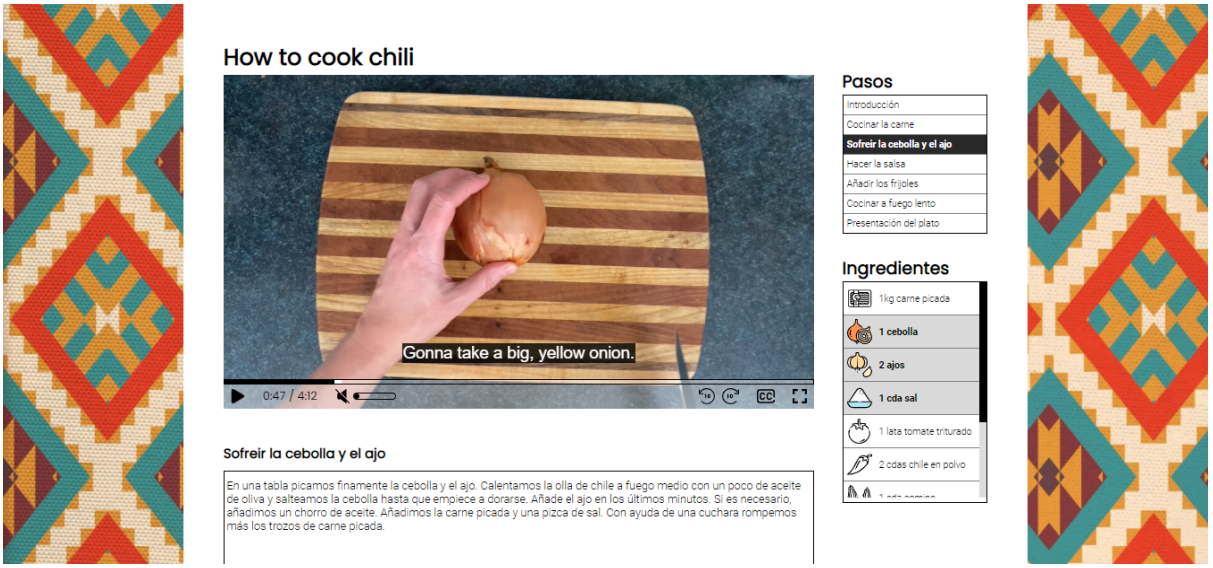
Integrantes:

Sergio Garcia Puertas - [redacted]
[redacted] - [redacted]

Índice

1. Funcionalidades y tecnologías utilizadas

La página web que hemos desarrollado tiene los siguientes elementos principales:



https://alumnes-ltim.uib.es/gdie2104/

- **Un reproductor de video con las funcionalidades**:
  - **Pausar/Reanudar**: clic en el botón de la barra de controles o directamente en el video.
  - **Tiempo actual**: se muestra el tiempo actual del video respecto al tiempo total.
  - **Ajustar el volumen**: arrastrar el slider de la barra de controles.
  - **Avanzar/Retroceder 10 seg**: dos botones de la barra de controles que respectivamente modifican el tiempo del video.
  - **Cambiar los subtítulos**: un botón en la barra de controles que despliega un menú. Hay tres opciones de subtítulos: Español, Inglés y Desactivado. Por defecto están desactivados.
  - **Pantalla completa**: botón en la barra de controles que pone el video en pantalla completa. Para salir se puede volver a hacer clic en el botón o pulsar la tecla “ESC”.
  - **Saltar a cualquier punto del video**: la barra de progreso del video permite hacer clic en cualquier punto del video y saltar al minuto del video correspondiente.

  Hemos utilizado JavaScript y jQuery para añadir eventos a los diferentes elementos del reproductor con tal de poder implementar las funcionalidades anteriores.

- **Lista de pasos**: contenedor web que contiene todos los pasos que aparecen en la receta del video. Si haces click en uno de estos pasos, se resalta y te lleva a la parte del video donde se realiza este paso. También se resaltan los ingredientes, de la lista de ingredientes, que se utilicen en ese paso.

- **Lista de ingredientes**: contenedor web que contiene todos los ingredientes que se utilizan en la receta del video. Si haces click en uno de estos ingredientes, se resalta

a los demás y te lleva al paso donde se utiliza este ingrediente. Esto hace que se resalte el paso que se ha puesto. Si en ese paso se utiliza más de un ingrediente, también se resaltan los demás.

- **Descripción del paso de la receta**: Por defecto, al cargarla página no aparece este contenedor web. Una vez iniciado el video se muestra el título y la descripción correspondientes al paso de la receta en el que se encuentra la reproducción del video.

  Las últimas 3 funcionalidades mencionadas las hemos implementado usando un archivo *.vtt* de metadata, donde en cada *cue* hay un *JSON* con la información necesaria. Además, también hemos utilizado JavaScript y jQuery para poder manipular la página web con la información que contiene el *JSON* de cada cue.

2. Material multimedia

2.1 Descarga del video

Para descargarnos el vídeo de youtube en máxima calidad (4k) hemos utilizado una herramienta llamada *youtube-dl*. Para utilizarla es necesario descargarse un archivo .exe y desde la consola del ordenador posicionarnos en el directorio en que se encuentra este archivo. Si utilizamos el comando `.\youtube-dl -F "[link del video]"`, nos aparece un listado de todos los formatos y calidades que podemos elegir para descargar el video. En nuestro caso, este es el listado de opciones:

```
PS C:\Users\gserg\OneDrive\Documents\youtube-dl> .\youtube-dl -F "https://www.youtube.com/watch?v=ZT3G1ECfYoU&ab_channel
=YouSuckAtCooking"
[youtube] ZT3G1ECfYoU: Downloading webpage
[youtube] ZT3G1ECfYoU: Downloading API JSON
[youtube] ZT3G1ECfYoU: Downloading API JSON
[info] Available formats for ZT3G1ECfYoU:
format code  extension  resolution note
249          webm       audio only tiny   51k , webm_dash container, opus @ 51k (48000Hz), 1.91MiB
250          webm       audio only tiny   62k , webm_dash container, opus @ 62k (48000Hz), 2.32MiB
251          webm       audio only tiny  115k , webm_dash container, opus @115k (48000Hz), 4.27MiB
140          m4a        audio only tiny  129k , m4a_dash container, mp4a.40.2@129k (44100Hz), 4.77MiB
160          mp4        256x144    144p   59k , mp4_dash container, avc1.4d400c@  59k, 30fps, video only, 2.20MiB
394          mp4        256x144    144p   70k , mp4_dash container, av01.0.00M.08@  70k, 30fps, video only, 2.60MiB
278          webm       256x144    144p   77k , webm_dash container, vp9@  77k, 30fps, video only, 2.84MiB
395          mp4        426x240    240p  118k , mp4_dash container, av01.0.00M.08@ 118k, 30fps, video only, 4.37MiB
133          mp4        426x240    240p  126k , mp4_dash container, avc1.4d4015@ 126k, 30fps, video only, 4.65MiB
242          webm       426x240    240p  152k , webm_dash container, vp9@ 152k, 30fps, video only, 5.61MiB
396          mp4        640x360    360p  213k , mp4_dash container, av01.0.01M.08@ 213k, 30fps, video only, 7.86MiB
134          mp4        640x360    360p  239k , mp4_dash container, avc1.4d401e@ 239k, 30fps, video only, 8.80MiB
243          webm       640x360    360p  314k , webm_dash container, vp9@ 314k, 30fps, video only, 11.58MiB
397          mp4        854x480    480p  357k , mp4_dash container, av01.0.04M.08@ 357k, 30fps, video only, 13.17MiB
135          mp4        854x480    480p  388k , mp4_dash container, avc1.4d401f@ 388k, 30fps, video only, 14.32MiB
244          webm       854x480    480p  551k , webm_dash container, vp9@ 551k, 30fps, video only, 20.30MiB
398          mp4        1280x720   720p  634k , mp4_dash container, av01.0.05M.08@ 634k, 30fps, video only, 23.37MiB
136          mp4        1280x720   720p  679k , mp4_dash container, avc1.4d401f@ 679k, 30fps, video only, 25.00MiB
247          webm       1280x720   720p 1084k , webm_dash container, vp9@1084k, 30fps, video only, 39.94MiB
399          mp4        1920x1080  1080p 1069k , mp4_dash container, av01.0.08M.08@1069k, 30fps, video only, 39.38MiB
248          webm       1920x1080  1080p 2035k , webm_dash container, vp9@2035k, 30fps, video only, 74.92MiB
137          mp4        1920x1080  1080p 2353k , mp4_dash container, avc1.640028@2353k, 30fps, video only, 86.65MiB
400          mp4        2560x1440  1440p 3988k , mp4_dash container, av01.0.12M.08@3988k, 30fps, video only, 146.82MiB
271          webm       2560x1440  1440p 5977k , webm_dash container, vp9@5977k, 30fps, video only, 220.01MiB
401          mp4        3840x2160  2160p 7060k , mp4_dash container, av01.0.12M.08@7060k, 30fps, video only, 259.89MiB
313          webm       3840x2160  2160p 15684k , webm_dash container, vp9@15684k, 30fps, video only, 577.32MiB
18           mp4        640x360    360p  574k , avc1.42001E, 30fps, mp4a.40.2 (44100Hz), 21.15MiB
22           mp4        1280x720   720p 2140k , avc1.64001F, 30fps, mp4a.40.2 (44100Hz) (best)
PS C:\Users\gserg\OneDrive\Documents\youtube-dl>
```

En nuestro caso, para la primera práctica vamos a utilizar el video con la máxima calidad posible tanto de video como de audio. Para esto basta que utilicemos el comando

`.\youtube-dl -f bestvideo+bestaudio "[link del video]"`. Al ejecutarse obtenemos dos archivos, uno con el video y otro con el audio.

2.2 Conversión del video

Una vez tenemos el video y el audio por separados debemos crear, combinando el video y audio, un archivo *.mp4* y otro archivo *.webm*.

Para esto hemos utilizado la herramienta *ffmpeg*. Primero debemos descargarnos la herramientas y después, desde la consola, nos posicionamos en el directorio donde tenemos los videos.

Para crear el archivo .mp4 hemos utilizado la siguiente instrucción:

```
ffmpeg -i chili-video.webm -ss 00:00:00 -to 00:04:12 -i chili-audio.m4a -ss 00:00:00 -to 00:04:12 -c:v libx264 -c:a aac videomp4.mp4
```

- -i → indica el archivo que se utiliza como input.
- -ss → indica el tiempo de inicio del video que queremos codificar.
- -to → indica el tiempo final del video que queremos codificar. Junto con el parámetro anterior se utilizan para recortar el video.
- -c:v → indica la librería (y por tanto el formato) que se utiliza para codificar el video.
- -c:a → indica la librería (y por tanto el formato) que se utiliza para codificar el audio.

Para crear el archivo .webm hemos utilizado la siguiente instrucción:

```
ffmpeg -i chili-video.webm -ss 00:00:00 -to 00:04:12 -i chili-audio.m4a -ss 00:00:00 -to 00:04:12 -c:v libvpx-vp9 -c:a libvorbis videowebm.webm
```

Se utilizan los mismos parámetros que en el caso anterior, pero cambiando las librerías que se utilizan para codificar y el nombre de la salida.

2.3 Publicación del video

Una vez tenemos los dos archivos que vamos a utilizar, debemos especificar en el elemento <video> del html que este contiene dos <source> diferentes:

```
<video id="myVideo" muted>
    <source src="videos/videomp4.mp4" type="video/mp4">
    <source src="videos/videowebm.webm" type="video/webm">
    <track src="chilidatos.vtt" kind="metadata">
    <track src="sub_eng.vtt" kind="subtitles" label="English" srclang="en">
    <track src="sub_esp.vtt" kind="subtitles" label="Español" srclang="es">
</video>
```

De esta manera, el navegador intentará utilizar el archivo .mp4 como primera opción. Si este funciona, ignorará el archivo *.webm*. Si, por el contrario, el archivo *.mp4* falla, el navegador utilizará el archivo *.webm*.

2.4 Estructura de las carpetas

La carpeta *GDIEpractica* tiene los siguientes archivos:

- *index.html*
- *style.css*
- *interaccion.js*
- *reproductor.js*
- *chilidatos.vtt*
- *sub_esp.vtt*
- *sub_eng.vtt*
- *mexican.jpg* // imagen del fondo de la página
- videos // carpeta con los videos
    - *videomp4.mp4*
    - *videowebm.webm*
- iconos // carpeta con los iconos usados
    - .svg de los elementos de la barra de control de video
    - blanco // carpeta de iconos
        - .svg de los ingredientes en blanco y negro
    - color // carpeta de iconos
        - .svg de los ingredientes a color

3. Desarrollo de las funcionalidades

3.1 Archivos TextTrack

Tenemos tres archivos de TextTrack:

1. **chilidatos.vtt**: es un TextTrack de metadatos. Contiene los capítulos del video, que corresponden a los diferentes pasos de la receta de chili. Cada uno de los siete capítulos tiene información en formato JSON. Todos contienen el título del paso de la receta (title), la descripción del paso de la receta (description) y un array con los ingredientes usados en cada paso (ingredientes). En el caso en que no se use ningún ingrediente, el array de ingredientes estará vacío.

```
WEBVTT

1
00:00:00.000 --> 00:00:09.000
{
"title": "Introducción",
"description": "Hay miles de recetas de chili, algunas con decenas de ingredientes.
"ingredientes":[]
}
```

2. **sub_eng.vtt**: contiene los subtítulos en inglés del video. Cada una de las líneas de subtítulo tiene la información asociada: line:-3, la cual tiene la funcionalidad de representar dicha línea de subtítulo 3 posiciones por encima de la posición inicial. Está hecho para garantizar que aunque la barra de controles del video esté visible, se puedan seguir leyendo los subtítulos.

3. **sub_esp.vtt**: contiene los subtítulos en español del video. Al igual que el sub_eng.vtt, cada línea de subtítulos tiene la información line:-3.

```
WEBVTT

1
00:00:00.000 --> 00:00:03.000 line:-3
Some chilies have 20, 40, 600-plus ingredients.

2
00:00:03.000 --> 00:00:06.000 line:-3
This one has 10, 'cause 10 is all you need.
```

3.2 Archivos JavaScript

Tenemos dos archivos JavaScript:

1. **interaccion.js**: contiene las instrucciones relacionadas con la interacción del video y los TextTracks.
   Al cargar la página web, se desactivan los subtítulos por defecto. Para ello, guardamos los TextTrack del video y los ponemos en modo "hidden".

```
videoElement = document.getElementById('myVideo');
//guardamos las textTracks del video
var textTracks = videoElement.textTracks;
//textTrack metadata
var textTrack = textTracks[0];
textTrack.mode = "hidden";
//textTrack subtítulos
var subEng = textTracks[1];
subEng.mode = "hidden";
var subEsp = textTracks[2];
subEsp.mode = "hidden";
```

   Añadimos un listener al TextTrack de metadatos (chilidatos.vtt) para detectar cuando cambia de *cue*. Cada vez que detecta un cambio de *cue* se realizan los siguientes pasos:

   1. Se recoge la información del JSON del paso que ha activado el cambio de *cue*. Se actualiza el título y descripción del paso del *cue* al que se ha cambiado y se muestra en el contenedor web de debajo del video. Cuando cogemos la información de la cue, lo hacemos de la última cue que se ha añadido a la variable "*textTrack*".

      Esto lo hacemos para que en caso de haber dos o más cues activas en el momento en que se produce un evento *cuechange*, cojamos la información de la última *cue*, que será la que ha activado el evento.

```
var datos = JSON.parse(textTrack.activeCues[textTrack.activeCues.length - 1].text);
//Actualizamos el título de la descripción
var tituloDesc = document.getElementById('pasoActual');
tituloDesc.innerHTML = datos.title;
//Actualizamos la descripción
var textoDesc = document.getElementById('descripcion');
textoDesc.innerHTML = datos.description;
```

2. Se ponen todos los elementos de las listas de pasos y de ingredientes en su estado neutro. Todos los elementos se ponen con letra normal y fondo blanco. Además, las imágenes de los ingredientes que estaban en color se sustituyen por las mismas imágenes en blanco y negro.

```
    //Reseteamos todo (pasos e ingredientes)
$("li").css("background-color","white");
$("li").find('button').css("font-weight","Normal");
$("li").find('button').css("color","black");
    //Resetear las imágenes
for(let i = 0; i<ultimosIngredientes.length; i++){
    $("#"+ultimosIngredientes[i]).find('img').attr("src","iconos/blanco/"+ultimosIngredientes[i]+".svg");
}
```

3. Se destaca el elemento de la lista de pasos que ha disparado el cambio de cue poniendo el fondo en gris oscuro y la letra en negrita y de color blanco. Se recogen los ingredientes del JSON, y para cada uno se destaca en la lista de ingredientes poniendo la letra en negrita y poniendo la imagen asociada en color.

```
    //Marcamos el paso en concreto
$("#"+idPaso).css("background-color","#292929");
$("#"+idPaso).find('button').css("font-weight","Bold");
$("#"+idPaso).find('button').css("color","white");
//Actualizamos los ingredientes utilizados
ultimosIngredientes = [];
if(datos.ingredientes.length > 0){
    for (let i = 0; i < datos.ingredientes.length; i++) {
        $("#"+datos.ingredientes[i]).css("background-color","#d9d9d9");
        $("#"+datos.ingredientes[i]).find('button').css("font-weight","Bold");
        $("#"+datos.ingredientes[i]).find('img').attr("src","iconos/color/"+datos.ingredientes[i]+".svg");
        ultimosIngredientes.push(datos.ingredientes[i]);
    }
}
```

En este archivo JavaScript también tenemos la función *"cambiarTiempo()"*. Cada vez que se pulsa un botón para seleccionar un paso o un ingrediente, se llama a esta función pasándole por parámetro el paso al que se debe ir. Por otro lado tenemos definida una array con los tiempos del video en que empieza cada paso.

Por tanto, esta función lo que hace es coger el tiempo del video respectivo al paso que se la ha indicado por parámetro, y actualizar el "*currentTime*" del video. Esto hará que se active la *cue* pertinente.

```
var tiempos = ["0","9","46","85","130","176","219"];
```

```
function cambiarTiempo(tiempo){
    videoElement.currentTime = tiempos[tiempo - 1];
}
```

2. **reproductor.js**: contiene las instrucciones relacionadas con las funcionalidades del reproductor de video.

   Para **pausar/reanudar el video** hemos puesto dos listeners; uno en el botón de play y otro en el contenedor del video y barra de controles. Según el estado del video, se pausa o reanuda y se cambia la imagen del botón de play.

```
playButton.addEventListener("click",function(){
    if (video.paused){
        video.play();
        document.getElementById("botonPP").src = "iconos/pausa.svg";
    }else{
        video.pause();
        document.getElementById("botonPP").src = "iconos/play2.svg";
    }
});

video.addEventListener("click", function() {
    if (video.paused){
        video.play();
        document.getElementById("botonPP").src = "iconos/pausa.svg";
    }else{
        video.pause();
        document.getElementById("botonPP").src = "iconos/play2.svg";
    }
});
```

   Cada vez que se detecta un cambio en el **tiempo del video** se recalcula el tiempo de reproducción. Se puede detectar por el avance natural del video o porque se ha saltado a un cierto punto del video. Se transforma el formato del tiempo de reproducción a mm:ss y se actualiza el texto que representa el tiempo en la barra de control. También se actualiza la barra de tiempo de video transcurrido.

```
video.addEventListener("timeupdate",function(){
    //Tiempo actual
    let segundos = Math.floor(video.currentTime % 60);
    let minutos = Math.floor(video.currentTime / 60);
    segundos = segundos >= 10 ? segundos : '0' + segundos;

    //Tiempo total
    let segTot = Math.floor(video.duration % 60);
    let minTot = Math.floor(video.duration / 60);
    segTot = segTot >= 10 ? segTot : '0' + segTot;
    var tiempoTotal = `${minTot}:${segTot}`;

    tiempo.innerText = `${minutos}:${segundos} / ${tiempoTotal}`;

    //Actualizamos la barra
    let posBarra = (video.currentTime / video.duration) * 10000;
    timeBar.value = posBarra;
    //Pintamos la barra
    var x = (timeBar.value/10000) *100;
    if(x<15 && !fullScreenActivada){
        x+=1;
    }
    var color = 'linear-gradient(90deg, rgb(0, 0, 0)' + x + '%, rgb(255, 255, 255, 0)' + x + '%)';
    timeBar.style.background = color;
});
```

Cada vez que se detecta un cambio de tiempo a mano, al hacer **clic en la barra de tiempo de video**, se actualiza la barra de tiempo y el tiempo de video al indicado. Al actualizar el tiempo, automáticamente se disparará el *listener "timeupdate"*.

```
timeBar.addEventListener("change", function() {
    // Calcular el tiempo nuevo
    var time = video.duration * (timeBar.value / 10000);

    // Actualizar el tiempo de video
    video.currentTime = time;
});
```

Para ajustar el **volumen** del video utilizamos un slider. Cuando se detecta un cambio en el slider se activa un evento que pone el volumen del video en función del valor del slider. Además, dependiendo de si el volumen es 0 o no, se muestra un icono u otro.

```
volumeBar.addEventListener("change", function(){
    //Actualizamos el volumen con el valor del slider
    if (volumeBar.value > 0) {
        video.muted = false;
        document.getElementById("volImg").src = "iconos/volumen.svg";
    }
    else {
        video.muted = true;
        document.getElementById("volImg").src = "iconos/mute.svg";
    }
    video.volume = volumeBar.value;
});
```

Para **avanzar o retroceder 10 segundos** hay dos *listeners*, uno en cada respectivo botón. Se suman o sustraen 10 segundos al tiempo del video en ese momento, teniendo en cuenta no sumar más allá del tiempo total o restar por debajo de 00:00.

```
prev10s.addEventListener ("click", function() {
    if (video.currentTime <= 10) {
        video.currentTime = 0;
    } else {
        video.currentTime -= 10;
    }
});

next10s.addEventListener ("click", function() {
    if (video.currentTime + 10 >= video.duration) {
        video.currentTime = video.duration;
    } else {
        video.currentTime += 10;
    }
});
```

Para los **subtítulos**, al pulsar el botón se activa o desactiva la clase *"show"* de un contenedor que tiene las diferentes opciones.

```
subtitles.addEventListener("click", function() {
    document.getElementById("idiomas").classList.toggle("show");
});
```

Cada opción está dentro de un botón, que al pulsarlos llama a la función *"cambiarSubtitulos()"* pasándole por parámetro el idioma en que se quieren los subtítulos o 'none' en caso de que se quieran desactivar. Esta función primero desactiva todos los subtítulos y, si se quieren poner en algún idioma, entonces activa los subtítulos de ese idioma. También se resalta la opción activada.

```
function cambiarSubtitulos(idioma) {
    for (var i = 0; i < video.textTracks.length; i++) {
        video.textTracks[i].mode = 'hidden';
    }

    document.getElementById("subEsp").classList.remove("subactivo");
    document.getElementById("subEng").classList.remove("subactivo");
    document.getElementById("subNone").classList.remove("subactivo");

    // Elegir los subtitulos
    if (idioma == "es") {
        video.textTracks[2].mode = "showing";
        document.getElementById("subEsp").classList.add("subactivo");
    } else if (idioma == "en") {
        video.textTracks[1].mode = "showing";
        document.getElementById("subEng").classList.add("subactivo");
    } else {
        document.getElementById("subNone").classList.add("subactivo");
    }

    document.getElementById("idiomas").classList.toggle("show");
}
```

Al poner la pantalla completa, se activa un evento que mira si hay un elemento en *full screen,* si no lo hay el reproductor (vídeo y controles), se pone en pantalla completa. Si ya hay un elemento en *full screen* se sale del modo *full screen*. En cualquier caso se llama a la función *setVideoSize()* que ajusta el tamaño del reproductor.

```
fullscreen.addEventListener("click", function(){
    if (!isInFullScreen()) {
        fullScreenActivada = true;
        if (player.requestFullscreen) {
            player.requestFullscreen();
        } else if (player.mozRequestFullScreen) {
            player.mozRequestFullScreen();
        } else if (player.webkitRequestFullScreen) {
            player.webkitRequestFullScreen();
        } else if (player.msRequestFullscreen) {
            player.msRequestFullscreen();
        }
    } else {
        fullScreenActivada = false;
        if (document.exitFullscreen) {
            document.exitFullscreen();
        } else if (document.webkitExitFullscreen) {
            document.webkitExitFullscreen();
        } else if (document.mozCancelFullScreen) {
            document.mozCancelFullScreen();
        } else if (document.msExitFullscreen) {
            document.msExitFullscreen();
        }
    }
    setVideoSize();
});
```

```
function setVideoSize() {
    if(fullScreenActivada){
        player.style.width = 'window.innerWidth';
    }else{
        player.style.width = '100%';
    }
  }
```