TEMA 22
Audio en HTML

Elemento <audio>

El elemento <audio> permite incrustar audio fácilmente en un archivo html.

En este tema vamos a poder realizar las siguientes acciones:

- Reproducir audio.
- Controlar el audio (controles personalizados).
- Grabar desde el micrófono.
- Descargar audios grabados.
- Analizar el sonido.
- Convertir texto en voz.

Para continuar con el tema vamos a crear una página denominada: Tema22.html:

```
tema22.html > html
1   <!DOCTYPE html>
2   <html lang="es">
3
4   <head>
5     <meta charset="UTF-8">
6     <!-- ETIQUETA PARA HACER RESPONSIVE LA PÁGINA, ADAPTACIÓN DE ESCALAS -->
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <title>Tema 22 html</title>
9     <!-- ICONO DE VENTANA -->
10    <link rel="icon" type="image/x-icon" href="imagenes/favicon.ico">
11    <!-- HOJA DE ESTILOS EXTERNA-->
12    <link rel="stylesheet" href="estilos/estilostema22.css">
13  </head>
14
15  <body>
16    <h2>INCORPORACIÓN DE AUDIO EN HTML</h2>
17    <div style="text-align:center">
18    </div>
19  </body>
```

La creamos con lo básico, creamos su hoja de estilo:

Estilostema22.css:

```
estilos > # estilostema22.css > ...
1   body {
2       font-family: Arial, sans-serif;
3       margin-left: 10px;
4       padding: 5px;
5   }
6
7   h2,h4 {
8       color: #4d77ca;
9       text-align: center;
10  }
11
```

1. REPRODUCCIÓN DE AUDIO

Vamos a escribir el siguiente código en nuestro archivo:

```
<body>
  <h2>INCORPORACIÓN DE AUDIO EN HTML</h2>
  <div style="text-align:center">
    <audio controls>
      <source src="https://www.soundhelix.com/examples/mp3/SoundHelix-Song-1.mp3" type="audio/mpeg">
      Tu navegador no soporta el elemento de audio.
    </audio>
  </div>
</body>
```

Es Parecido a la carga de vídeos, como podéis apreciar, si el navegador no dispone del software disponible para reproducir el audio saldría el mensaje que aparece en la imagen. Por otro lado el atributo "controls" hace que aparezcan los controles de audio en pantalla, para poder reproducirlo, cambiar volumen, cambiar velocidad, descargarlo, etc…

La pantalla saldría de la siguiente forma:

Podemos utilizar dos propiedades más:

- muted: el audio aparecerá de primeras con el volumen muteado.
- loop: El audio se reproducirá indefinidamente.

```
<audio controls muted loop>
  <source src="https://www.soundhelix.com/e
  Tu navegador no soporta el elemento de au
</audio>
```

2. CONTROLES PERSONALIZADOS

Vamos a escribir el siguiente código en nuestro archivo:

Ponemos los siguientes botones en nuestro código:

```
<h2>INCORPORACIÓN DE AUDIO EN HTML</h2>
<div style="text-align:center">
  <audio id="miAudio" loop>
    <source src="https://www.soundhelix.com/examples/mp3/SoundHelix-Song-1.mp3" type="audio/mpe
    Tu navegador no soporta el elemento de audio.
  </audio>
  <br>
  <h4>CONTROLES</h4>
  <button onclick="playAudio()">Reproducir</button>
  <button onclick="pauseAudio()">Pausar</button>
</div>
```

Podemos observar que hemos quitado en principio la propiedad “muted” para que no esté el volumen quitado y también la propiedad “controls” para que no aparezcan los controles nativos.

Una vez definidos los botones, vamos a poner el código javascript al que llaman los mismos:

```
    <button onclick="playAudio()">Reproducir</button>
    <button onclick="pauseAudio()">Pausar</button>
  </div>

  <!-- JAVASCRIPT PARA CONTROLAR EL AUDIO -->
  <script>
    const audio = document.getElementById("miAudio");

    function playAudio() {
      audio.play();
    }

    function pauseAudio() {
      audio.pause();
    }
  </script>

</body>
```

Ahora saldrá de la siguiente forma la pantalla:

Vamos a incorporar algunos controles más:

- Subir/Bajar volumen gradualmente.
- Mutear/Desmutear el audio.
- Avanzar/Retroceder 5 segundos.

Vamos a poner el siguiente código:

```
    <h4>CONTROLES</h4>
    <!-- Controles de reproducción y pausa -->
    <button onclick="playAudio()">Reproducir</button>
    <button onclick="pauseAudio()">Pausar</button>
    <!-- Controles de volumen -->
    <button onclick="changeVolume(0.1)">Subir Volumen</button>
    <button onclick="changeVolume(-0.1)">Bajar Volumen</button>
    <button onclick="toggleMute()">Mute</button>
    <!-- Controles de reproducción -->
    <button onclick="seek(-5)">Retroceder 5s</button>
    <button onclick="seek(5)">Avanzar 5s</button>
</div>
```

Se debería ver así:

Una vez definidos los botones con sus llamadas a funciones de javascript, vamos a añadir estas funciones de javascript.

```
function pauseAudio() {
  audio.pause();
}

// Cambiar volumen gradualmente
function changeVolume(amount) {
  let newVolume = Math.min(1, Math.max(0, audio.volume + amount));
  audio.volume = newVolume;
}

// Mute / Unmute
function toggleMute() {
  audio.muted = !audio.muted;
}

// Avanzar o retroceder en el audio
function seek(seconds) {
  audio.currentTime = Math.max(0, audio.currentTime + seconds);
}
```

Ahora podemos probar estos botones y sus funcionalidades.

Como pensada fíjate qué cómodo sería realizar un mapa de imagen de una mezcladora de Audio y ponerles enlaces a estos controles, haciendo que se sincronizara con una pantalla de vídeo. No hay que hacerlo en clase por falta de tiempo, pero no por falta de ganas.

3. GRABAR DESDE EL MICRÓFONO

Vamos a escribir el siguiente código en nuestro archivo:

```
    <button onclick="changeVolume(-0.1)">Bajar Volumen</button>
    <button onclick="toggleMute()">Mute</button>
    <!-- Controles de reproducción -->
    <button onclick="seek(-5)">Retroceder 5s</button>
    <button onclick="seek(5)">Avanzar 5s</button>
    <br><br>
    <h4>CONTROLES DE GRABACIÓN</h4>
    <!-- Controles de grabación de audio -->
    <button onclick="startRecording()">Iniciar Grabación</button>
    <button onclick="stopRecording()">Detener Grabación</button>
</div>
```

Debería quedar así:

Ahora vamos a poner el código javascript al que llaman estos botones:

```
//Código para controlar la grabación de audio:
let mediaRecorder;
let audioChunks = [];

async function startRecording() {
  const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
  mediaRecorder = new MediaRecorder(stream);
  mediaRecorder.ondataavailable = event => audioChunks.push(event.data);

  mediaRecorder.onstop = () => {
    const audioBlob = new Blob(audioChunks, { type: "audio/mp3" });
    const audioURL = URL.createObjectURL(audioBlob);
    document.getElementById("miAudio").src = audioURL;
    audioChunks = [];
  };

  mediaRecorder.start();
}

function stopRecording() {
  mediaRecorder.stop();
}
```

El grabar es una función asíncrona porque no salta cuando se le llama directamente, se queda esperando a que se le de permiso al navegador de poder grabar audio, por eso se define como “async”.

Cada vez que en una función se utilice una llamada del tipo “await” hay que definir la función en la que se encuentra del tipo “async”.

El await espera a que se den permisos para seguir ejecutando el código.

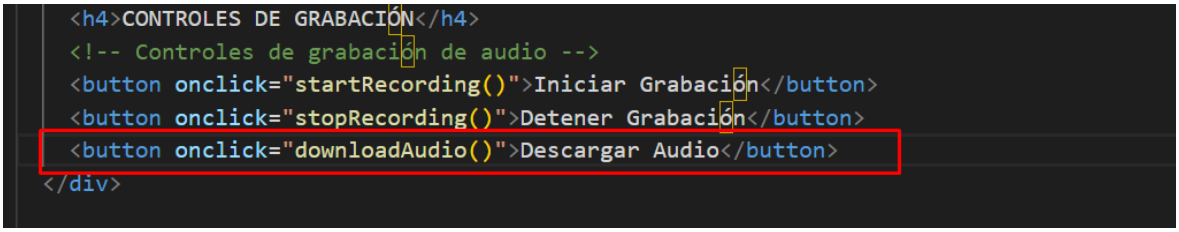
Podéis observar que utilizamos el mismo control de audio, lo que hacemos es cambiarle la fuente para que apunte al objeto nuevo que se ha grabado.

La función “onstop” se llama automáticamente cuando se para la grabación y ahí es cuando se hace el cambio de fuente de audio. Ahora todos los controles manipulan el audio nuevo.

4. DESCARGAR EL AUDIO GRABADO

Vamos a descargar el audio grabado sin necesidad de utilizar los controles por defecto:

Añadimos el siguiente botón a nuestra lista de controles de grabación:



Ahora añadimos la función en javascript de descarga de audio “downloadAudio”:

Para poder hacer la descarga vamos a definir nuestra variable que guarda la creación de audio en una variable global para luego poder utilizarla en la función de descarga:

```
//Código para controlar la grabación de audio:
let mediaRecorder;
let audioChunks = [];
let audioBlob;

async function startRecording() {
  const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
  mediaRecorder = new MediaRecorder(stream);
  mediaRecorder.ondataavailable = event => audioChunks.push(event.data);

  mediaRecorder.onstop = () => {
    audioBlob = new Blob(audioChunks, { type: "audio/mp3" });
    const audioURL = URL.createObjectURL(audioBlob);
    document.getElementById("miAudio").src = audioURL;
    audioChunks = [];
  };

  mediaRecorder.start();
}
```

Ahora definimos la función de descarga:

```
// Javascript para descargar audio:
function downloadAudio() {
  const a = document.createElement("a");
  a.href = URL.createObjectURL(audioBlob);
  a.download = "grabacion.mp3";
  a.click();
}
</script>

</body>
```

Ahora podemos probar a grabar un archivo de audio y descargar el mismo con el nombre de archivo que hemos definido: “grabación.mp3”.

5. ANALIZAR EL SONIDO

Añadimos un canvas a nuestra página que sirva para dibujar el spectro de frecuencias de audio:

```
<h4>CONTROLES DE GRABACION</h4>
<!-- Controles de grabación de audio -->
<button onclick="startRecording()">Iniciar Grabación</button>
<button onclick="stopRecording()">Detener Grabación</button>
<button onclick="downloadAudio()">Descargar Audio</button>
<!-- Análisis de frecuencias del audio -->
<br><br>
<h4>ANÁLISIS DE FRECUENCIA</h4>
<canvas id="canvas" width="500" height="100" style="border: 1px solid black"></canvas>
</div>

<!-- JAVASCRIPT PARA CONTROLAR EL AUDIO -->
```

Ahora nuestra página tendrá el siguiente aspecto:

Ahora incorporamos las funciones y código script:

```
//Script para definir el spectro de frecuencia de audio y recoger el id del canvas:
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext("2d");

const audioCtx = new (window.AudioContext || window.webkitAudioContext)();
const analyser = audioCtx.createAnalyser();
const source = audioCtx.createMediaElementSource(miAudio);
source.connect(analyser);
analyser.connect(audioCtx.destination);

//Función que dibuja el spectro de audio:
function draw() {
  requestAnimationFrame(draw);
  const data = new Uint8Array(analyser.frequencyBinCount);
  analyser.getByteFrequencyData(data);

  ctx.clearRect(0, 0, canvas.width, canvas.height);
  data.forEach((value, i) => {
    ctx.fillStyle = `rgb(${value},50,50)`;
    ctx.fillRect(i * 3, canvas.height - value, 2, value);
  });
}

//Se ejecuta cuando se pulsa el audio:
miAudio.onplay = () => { audioCtx.resume(); draw(); };
```

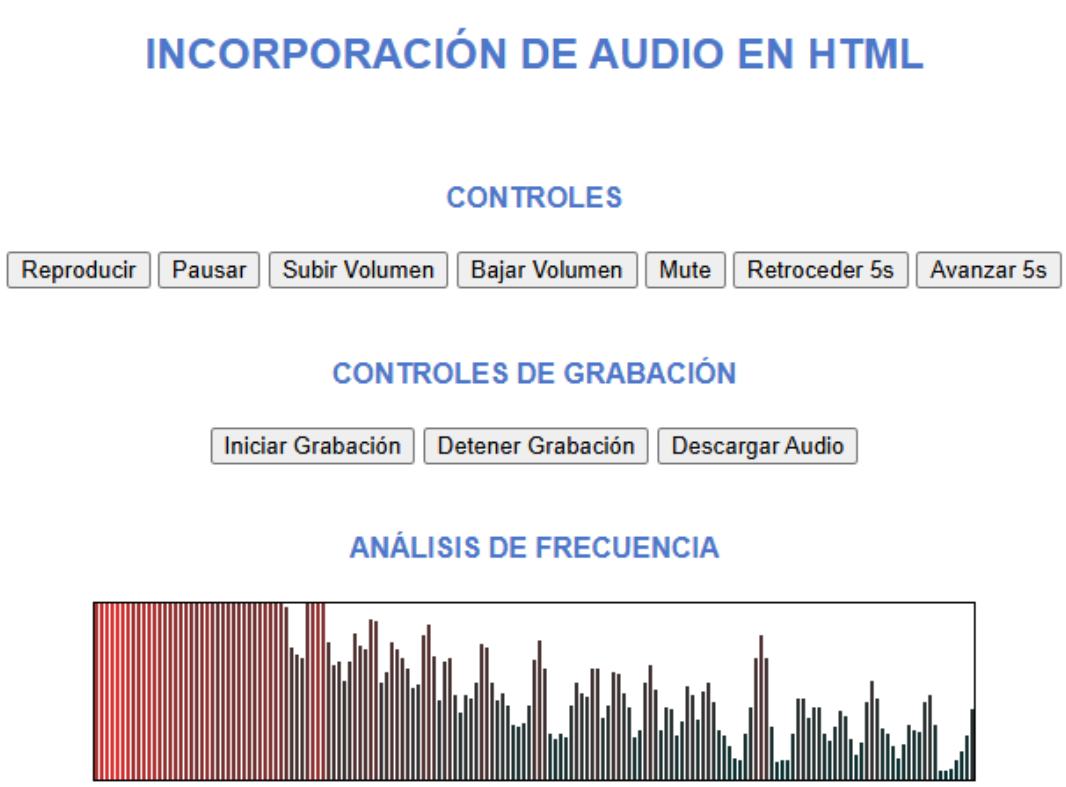
Por motivos de seguridad de cors, tenemos que abrir un Chrome con la seguridad desactivada:

DESACTIVAR SEGURIDAD CHROME (Cambiar ruta según ubicación del Chrome en el pc):

cd C:\Program Files (x86)\Google\Chrome\Application

chrome.exe --disable-web-security --user-data-dir="C:\chrome_dev"

El resultado es el siguiente:



No me imagino que de cosas podemos hacer ya, entre el mapa de imagen, controles y el gráfico de frecuencia (equalizador), parece que estamos en plan DJ.

6. CONVERTIR TEXTO EN VOZ

Por último, en el audio vamos a ver como podemos convertir texto en voz, muy útil cuando estamos hablando de accesibilidad y mostrar información a personas con cierta discapacidad visual.

Vamos a poner el botón que lea un mensaje en nuestro archivo html:

```
    <h4>ANÁLISIS DE FRECUENCIA</h4>
    <canvas id="canvas" width="500" height="100" style="border: 1px solid black"></canvas>
    <!-- Convertir texto en voz-->
    <br><br>
    <h4>CONVERTIR TEXTO EN VOZ</h4>
    <button onclick="speak()">Leer Texto</button>
  </div>
```

Y a continuación creamos nuestra función “speak()”:

```
// Función que convierte texto en voz:
function speak() {
  const mensaje = new SpeechSynthesisUtterance("Hola, esto es una prueba de texto a voz.");
  window.speechSynthesis.speak(mensaje);
  alert("Hola, esto es una prueba de texto a voz.");
}

</script>
```

Ahora podemos probar que aparte de salir el mensaje en pantalla con un “alert”, también sale el audio del propio mensaje informativo.

Por último dos propiedades avanzadas del textspeak:

```
// Función que convierte texto en voz:
function speak() {
  const mensaje = new SpeechSynthesisUtterance("Hola, esto es una prueba de texto a voz.");
  // Cambiar idioma a español (España)
  mensaje.lang = "es-ES"; //en-EN para inglés, ....
  // Ajustar la velocidad de habla (1 es normal, 0.5 es más lento, 2 es más rápido)
  mensaje.rate = 1;
  window.speechSynthesis.speak(mensaje);
  alert("Hola, esto es una prueba de texto a voz.");
}
```

Atributo “lang”: El idioma que lee el mensaje.

Atributo “rate”: Velocidad de lectura.