2024/25

U.D.5. INTEGRACIÓN DE CONTENIDO INTERACTIVO Y MULTIMEDIA.

5.1 Elementos multimedia e interactivos básicos

ÍNDICE

1. El sonido en la web

En la actualidad, es posible incorporar a un sitio web casi cualquier tipo de elemento, incluyendo elementos multimedia como el sonido. El uso de las nuevas tecnologías, si estas son usadas de forma correcta, incrementa el grado de sensación de satisfacción de un usuario al navegar en una página web, objetivo final de cualquier diseño de interfaces web.

Es común el uso de los elementos de audio en páginas web; una de las maneras más habituales consiste en aparecer como **música de fondo** que se reproduce mientras el usuario está navegando por el sitio. En estos casos, **es aconsejable que se facilite** a dicho usuario **la posibilidad de silenciarla**.

Para incrustar un archivo de audio basta con utilizar la etiqueta `<audio>`, seguida de la ruta del fichero.

La **sintaxis** para añadir **sonido de fondo** es la siguiente:

```
<audio src="nombreFicheroMusica.extension" opciones> … </audio>
```

Este elemento incorpora un conjunto de parámetros interesantes que permiten definir las características de la reproducción. De esta forma, es posible adaptarlo a la experiencia de navegación que se desee para el usuario final. Los **parámetros** más habituales son:

- `src`. Indica la dirección y nombre del archivo contenedor de la música que se desea reproducir. Este atributo **puede ser reemplazado por** el elemento `<source>` y su propio atributo `src` para declarar varias fuentes con diferentes formatos.
- `width` y `height`. Tamaño de los controles del reproductor que aparece en la pantalla.
- `autoplay`. Si se indica a true, inicia de forma automática la reproducción. El autoplay en las últimas versiones de los navegadores web **está bloqueado**.
- `loop`. Si se indica a true, reproduce en bucle el archivo hasta que el usuario abandona la página.
- `controls`: muestra los controles de audio que nos ofrece el navegador. Cuando se incluye el atributo, el navegador activará su propia interfaz de control del audio.
- `hidden`. Si se indica a true, oculta el reproductor.
- `preload`: indica si se cargará el archivo con la página (`auto`, se carga al cargar la página; `meta`, solo carga metadatos; y `none` no hay precarga o no se cachea).

En el siguiente **ejemplo**, se indica que se reproduzca un archivo de audio llamado *audio.ogg*. La reproducción comenzará en cuanto el usuario entre en el sitio web (*autoplay*), y estará ejecutándose en bucle hasta el que se abandone la página (*loop*).

```
<audio src="audio.ogg" autoplay loop> </audio>
```

A continuación, se muestra un **ejemplo** de código sencillo en donde se inserta audio:

```
<body>
   <section>
       <article>
           <figure>
               <h1>Mientras navegas, escuchas música de ambiente</h1>
               <audio src="audio/canción1.mp3" controls autoplay></audio>
           </figure>
       </article>
   </section>
</body>
```

La etiqueta `<audio>` tiene **otros atributos** y puede actuar como contenedor e incorporar otras etiquetas:

- *<source>:* permite indicar **varias rutas** de ficheros de audio con diversas extensiones para tratar de asegurar que, al menos, una sea reconocida por el navegador y pueda reproducirse.
  Cuando utilizamos *<source>* es recomendable utilizar el atributo *type*, el cual permite que el navegador conozca el tipo MIME y los tipos de códecs que debe utilizar antes de descargar el audio. Si no indicamos dicho atributo, el navegador intentará averiguar, mediante prueba y error, cuál es el tipo adecuado.

- *<track>:* define un fichero de subtítulos. También se puede usar con la etiqueta *<video>*.

```
<!DOCTYPE html>
<html>
<body>

<audio controls autoplay>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  <track src="fgsubtitles_en.vtt" kind="subtitles" srclang="en" label="English">

</audio>

</body>
</html>
```

Otra de las opciones de uso de sonido en la web se realiza a través de **enlace** conocido como **sonido manual**; es decir, la señal acústica solo se reproduce si el usuario realiza alguna acción directa sobre el elemento asociado al sonido. Por **ejemplo**, si se rellena un campo de forma errónea, se activa un aviso acústico.

La **sintaxis** para añadir **sonido manual** es el siguiente:

```
<a href="nombreFicheroSonido.extension"> ELEMENTO_ACCION </a>
```

A continuación, se presenta un **ejemplo** sencillo:

```
<body>
<a href="cancion1.mp3"><button>Haz click aqui</button> </a>
</body>
```

1.1 Soporte en navegadores web

MP3 está bajo licencia comercial, así que no es soportado por navegadores como Firefox u Opera. OGG es soportado por estos navegadores, pero no por Safari e Internet Explorer. Por este motivo, es **necesario subir los dos tipos (OGG y MP3) de archivos en nuestras páginas web**.

1.2 Consideraciones para el uso de audio en un sitio web

Ya hemos visto cómo insertar audio en la web. Para hacerlo, conviene tener en cuenta algunas consideraciones sobre su uso:

1) El audio se usa como contenido de dicha web.
2) Se usará como elemento decorativo (música de fondo).

En el primer caso, el uso del audio se hace imprescindible para dicha página. Respecto al uso de **música de fondo**, cabe destacar que es una técnica en **desuso**. Sin embargo, en el

caso de que decidamos incluir música de fondo, deberemos **mostrar** también **botones** de reproducción para poder iniciar y parar la música cuando el usuario así lo desee.

Los archivos de **audio** solo deben utilizarse **cuando sea imprescindible**, ya que ocupan mucho tamaño, y optimizando su reproducción.

2. El uso del vídeo en la web

Para incorporar un vídeo en la web deben tenerse en cuenta varios factores en función de lo que se desea hacer con dicho recurso.

- En el caso de ser un **vídeo en streaming**, ha de tenerse en cuenta la tasa de transferencia o, lo que es lo mismo, el número de bits por segundo al que debe realizarse la transmisión para que el usuario lo vea sin cortes.
- Otra opción es que el **archivo se descargue** desde el sitio web, pero no se vaya a visualizar directamente en la página; en tal caso, el factor más importante será el peso del fichero.

Para incrustar un video en una página web, que se reproduce desde esta, se utiliza la etiqueta `<video>` seguida de la ruta en la que se encuentra alojado el fichero que se desea reproducir. De manera opcional, pueden indicarse las dimensiones del vídeo utilizando los parámetros de altura y anchura, `height` y `width`.

La **sintaxis** para añadir un **vídeo** es el siguiente:

```
<video src="rutaFicheroVideo" width="xxx" height="xxx"> </video>
```

El elemento `<video>` dispone de varios atributos que nos permiten establecer sus diferentes valores de comportamiento.

- `width` y `height`, definen las dimensiones del reproductor de vídeo, tal y como ya se ha mencionado.
- `src`. Indica la fuente del vídeo. Este atributo puede ser reemplazado por el elemento `<source>` y su propio atributo `src` para declarar varias fuentes con diferentes formatos.

  Al especificar el atributo `type` permitimos que el navegador conozca el tipo MIME (Multi-Purpose Internet Mail Extensions) y los tipos de códecs que debe utilizar antes de descargar el vídeo. Si no indicamos dicho atributo, el navegador intentará averiguar, mediante prueba y error, cuál es el tipo adecuado.

La etiqueta `<video>` permite **indicar varios formatos** de fichero puesto que, en función del navegador, funcionan diferentes formatos. Para ello, basta con repetir la siguiente línea las veces que se desee:

```
<source src="rutaFicheroVideo.extension" type="video/tipoFormato">
```

Por **ejemplo**:

```
<source src="rutaFicheroVideo.mp4" type="video/mp4">
```

A continuación, se muestra un **ejemplo** de código sencillo en donde se inserta vídeo:

```
<html>
   <head> INSERTANDO VIDEO</head>
   <body>
    <section>
      <article>
        <figure>
          <h1>Si pulsas PLAY comenzará la reproducción: </h1>
          <video>
            <source src="media/tipo1.mp4" controls preload>
            <source src="media/tipo2.mov" controls preload>
            <source src="media/tipo3.wmv" controls preload>
            <source src="media/tipo4.ogg" controls preload>
          </video>
        </figure>
      </article>
    </section>
   </body>
</html>
```

- `controls`. Define si se muestran los controles del vídeo que nos ofrece el navegador (reproducir, pausar...).
- `autoplay`. El video se reproduce en cuanto se carga, no es necesario activarlo. En la mayoría de los navegadores, este atributo está **bloqueado**.
- `loop`. El vídeo se reproduce en bucle hasta que el usuario abandona la página.
- `mutted`. El vídeo comienza su reproducción sin sonido.
- `preload`. Puede recibir tres valores:
  - *none*: el vídeo no se cachea o no se carga hasta que el usuario no lo decida pulsando sobre él.
  - *metadata*: recomienda al navegador que capture información acerca de la fuente (dimensión, duración, etc).

- *auto*: la carga del vídeo comienza cuando el usuario accede a la página, y por tanto, descarga el archivo lo más pronto posible.

- `poster`. Provee una imagen que será mostrada mientras el usuario espera a que cargue el vídeo.

```
<!DOCTYPE html>
<html>
<body>

<h1>The video poster attribute</h1>

<video width="320" height="240"
poster="/images/w3schools_green.jpg" controls>
   <source src="movie.mp4" type="video/mp4">
   <source src="movie.ogg" type="video/ogg">
   Your browser does not support the video tag.
</video>

</body>
</html>
```

En la nueva versión HTML5 se incorpora un **nuevo atributo** al elemento de enlace `<a>`, denotado por **`download`**, aunque esta opción puede utilizarse para la **descarga de cualquier recurso** que se desee descargar, se expone el ejemplo para un elemento de vídeo. En este caso, el atributo `'href'` indica la ruta donde está el elemento que se desea descargar, y será `download` el que permite la descarga. Si se indica una ruta a continuación del atributo, este se almacenará allí: si no, en el lugar por defecto del dispositivo del usuario. Un **ejemplo** sería el siguiente:

```
<a href="videos/video.mp4.html" download> Descargar Archivo </a>
```

Por último, **no será necesario que tengamos almacenado siempre el archivo de vídeo** para insertarlo en la web, ya que también podemos enlazar vídeos de distintas webs, como de **YouTube**. En estos casos, los vídeos disponen de un enlace para insertarlos en páginas web usando la mayoría de las veces la etiqueta *<iframe>*. También se podrán usar estilos CSS, y su **sintaxis** es la siguiente:

```
<iframe src="url" title="description"></iframe>
```

3. Control de reproducción de vídeo y audio con JavaScript

Como hemos visto en las secciones anteriores, los navegadores trabajan de forma diferente y algunos de los atributos de `<audio>` y `<video>` pueden estar o no habilitados por defecto.

Para poder controlar todos los elementos de nuestro reproductor podemos utilizar **JavaScript** y aprovechar los nuevos **métodos, propiedades y eventos** incorporados en HTML5.

3.1 Métodos

Para gestionar la reproducción de medios en HTML5 disponemos de una lista de métodos que se pueden llamar desde JavaScript. Los métodos más utilizados son los siguientes:

- `play()`: comienza a reproducir el medio desde el inicio, siempre que el medio no haya sido pausado previamente.
- `pause()`: pausa la reproducción.
- `load()`: carga el archivo del medio. Es útil para cargar el medio anticipadamente.
- `canPlayType(formato)`: para comprobar si el formato del archivo se soporta por el navegador.

Por **ejemplo**, si queremos iniciar la reproducción de un vídeo después de pulsar un botón haremos lo siguiente:

```
<video id="video" width="700" height="350">
    <source src="video.mp4">
    <source src="video.ogv">
</video>

<input type="button" id="boton" value="Reproducir">

<script>
  function iniciar() {
     var boton=document.getElementById('boton');
     boton.addEventListener('click', presionar, false);
  }
  function presionar() {
     var video=document.getElementById('video');
     video.play();
  }
  window.addEventListener('load', iniciar, false);
</script>
```

3.2 Propiedades

Con HTML5 también disponemos de algunas propiedades que nos darán información sobre el medio, la mayoría de ellas son de lectura/escritura, por lo tanto, también podemos cambiar su valor. Las siguientes son las propiedades más utilizadas:

- ***paused***: tiene valor «true» si la reproducción del medio está actualmente pausada o no ha comenzado.
- ***ended***: tiene valor «true» si la reproducción del medio ha finalizado porque se llegó al final.
- ***duration***: contendrá la duración del medio en segundos.
- ***currentTime***: Esta propiedad de lectura/escritura contendrá un valor para informar sobre la posición en segundos en la cual el medio está siendo reproducido, o especifica una nueva posición donde continuar reproduciendo.
- ***error***: tiene el valor del error ocurrido.
- ***muted***: para desactivar (true) o activar (false) el sonido del vídeo.
- ***volume***: para indicar el volumen del vídeo. El valor del volumen debe ser un número entre 0 y 1, por ejemplo si indicamos medio.volume = 0.3 estaríamos configurando el volumen al 30%.

Por ejemplo, podemos cambiar la función presionar() del ejemplo anterior, para que se inicie la reproducción del vídeo al presionar el botón y se pause la reproducción si se vuelve a presionar. En el siguiente código, cambiamos también el texto del botón indicando la operación a realizar.

```
function presionar() {
     if(!medio.paused && !medio.ended)   {
        medio.pause();
        reproducir.value='Reproducir';
     }
     else
     {
        medio.play();
        reproducir.value='Pausa';
     }
}
```

3.3 Eventos

También disponemos de diferentes eventos que nos permiten conocer la situación del medio. Los eventos más importantes son los siguientes:

- ***progress***: progreso de la descarga del medio. La información estará disponible a través del atributo buffered.
- ***canplaythrough***: se dispara cuando el medio completo puede ser reproducido sin interrupción.
- ***ended***: se dispara cuando el reproductor llega al final del medio.
- ***pause***: se dispara cuando el reproductor es pausado.
- ***play***: se dispara cuando el medio comienza a ser reproducido.
- ***error***: se dispara cuando ocurre un error.

Más eventos JS

A continuación, se muestra un **ejemplo** donde se implementan las funcionalidades de los botones play/pause.

Código HTML:

```
<video id="medio" width="720" height="400">
  <source src="http://clips.vorwaerts-gmbh.de/big_buck_bunny.mp4">
  <source src="http://clips.vorwaerts-gmbh.de/big_buck_bunny.ogv">
</video>

<nav>
      <input type="button" id="reiniciar" value="reiniciar">
      <input type="button" id="retrasar" value="&laquo;">
      <input type="button" id="play" value="&#9658;">
      <input type="button" id="adelantar" value="&raquo;">
      <input type="button" id="silenciar" value="silenciar">
      <label>Volumen</label>
      <input type="button" id="menosVolumen" value="-">
      <input type="button" id="masVolumen" value="+">
</nav>
```

Código JS:

```
function accionPlay() {
	if(!medio.paused && !medio.ended)
	{	medio.pause();
		play.value='\u25BA'; //\u25BA
		document.body.style.backgroundColor = '#fff';	}
	else
	{	 medio.play();
		play.value='||';
		document.body.style.backgroundColor = 'grey';	}
}

function accionReiniciar() {
  //Usar propiedad .currentTime
  //Reproducir el vídeo
  //Cambiar el valor del botón a || }

function accionRetrasar() {
   //Usar propiedad .curentTime
  //Contemplar que no existen valores negativos }

function accionAdelantar() {
  //Contemplar que no existen valores mayores a medio.duration   }

function accionSilenciar() {
	//Utilizar medio.muted = true; o medio.muted = false; }

function accionMasVolumen() {
	//Contemplar que el valor máximo de volumen es 1  }

function accionMenosVolumen() {
	//Contemplar que el valor mínimo de volumen es 0  }

function iniciar() {
	medio=document.getElementById('medio');
	play=document.getElementById('play');
	reiniciar=document.getElementById('reiniciar');
	retrasar=document.getElementById('retrasar');
	adelantar=document.getElementById('adelantar');
	silenciar=document.getElementById('silenciar');

	play.addEventListener('click', accionPlay);
	reiniciar.addEventListener('click', accionReiniciar);
	retrasar.addEventListener('click', accionRetrasar);
	adelantar.addEventListener('click', accionAdelantar);
	silenciar.addEventListener('click', accionSilenciar);
	menosVolumen.addEventListener('click', accionMenosVolumen);
	masVolumen.addEventListener('click', accionMasVolumen);
}

window.addEventListener('load', iniciar, false);
```

reiniciar « ▶ » silenciar Volumen - +

Actividad propuesta

*Partiendo del código del ejemplo anterior, implementa las siguientes **funcionalidades extra**:*

a) *Al pulsar el botón «reiniciar» si el vídeo está iniciado se reiniciará, es decir, comenzará a reproducirse de nuevo desde el inicio.*

b) *Al pulsar el botón "retrasar" la reproducción del vídeo saltará 5 segundos hacia atrás.*

c) *Al pulsar el botón "adelantar" la reproducción del vídeo saltará 5 segundos hacia delante.*

d) *Al pulsar el botón «silenciar» el sonido del vídeo se desactivará y el texto del botón cambiará a "escuchar". Al volver a pulsar el botón se activará el sonido y se cambiará de nuevo el texto a "silenciar".*

e) *Al pulsar el botón "menosVolumen" se bajará el volumen del vídeo 0.1 puntos hasta llegar a 0.*

f) *Al pulsar el botón masVolumen ( ) se subirá el volumen del vídeo 0.1 puntos hasta llegar a 1.*

4. Elementos interactivos

Los elementos interactivos permiten acercar la comunicación entre el sitio web y el usuario, esto es, **hacerlo partícipe** y que consiga una experiencia de navegación única, como si la página estuviera diseñada para él, creando una **web** más **dinámica**. Por lo tanto, **para conseguir un sitio interactivo, el foco principal del diseño de la web siempre será el usuario**. Nunca deben perderse de vista la accesibilidad ni la usabilidad a costa de ganar interactividad.

4.1 ¿Qué son los elementos interactivos?

Se define como elemento interactivo aquel que **cambia su estado o su valor cuando el usuario interactúa con él o sobre él**. Existen multitud de elementos interactivos. A continuación, se verán algunos **ejemplos**:

a) **Enlaces**. Cuando el usuario actúa sobre ellos, se produce un cambio en la página, bien por el redireccionamiento hacia otro sitio o página web, bien por la descarga de algún tipo de fichero (con la propiedad 'download'), etc. En función de si un enlace ha sido visitado o no, se puede modelar su apariencia.
b) **Botones**. La acción de pulsar un botón puede implicar múltiples acciones, desde el envío de un formulario, al borrado de la información contenido en un formulario, la selección de una opción, etc. De nuevo, al igual que ocurre con los enlaces, el estado de un botón cambia en función de si ha sido pulsado o no.
c) **Cuadros de texto**. Son elementos en los cuales el usuario puede escribir una determinada información, por lo tanto, su aspecto va cambiando.
d) **Casillas de verificación** o **cuadros check** (checkboxes). Permiten escoger una serie de opciones; dependiendo de si están seleccionados o no, su estado es diferente.

4.2 Comportamientos interactivos

Los elementos anteriores, de por sí, no implican un comportamiento interactivo. Para añadir un comportamiento interactivo a dichos elementos, y que el usuario tenga la sensación de tener el control de la página, podemos hacer uso de **HTML5**, **CSS** y lenguajes de programación dinámicos, como es el caso de **JavaScript** y su librería o biblioteca **jQuery**.

Aunque **jQuery** es utilizado por el **77%** de todos los sitios web según la w3techs (https://w3techs.com/technologies/overview/javascript_library), actualmente **JavaScript puede resolver** de manera nativa aquellos problemas que en 2006 llevaron a John Resig a crear jQuery:

- Una **forma unificada** de acceder al DOM (antes no había un método estándar).
- Poder **interactuar** con datos de un servidor.
- Crear **animaciones** (antes era muy complejo).

¿Entonces por qué aún se sigue utilizando? Debido a que se empezó a utilizar de forma masiva, y su migración a otra librería más moderna o a un framework, resulta compleja.

Aunque jQuery es más popular incluso que Bootstrap, utilizar jQuery para tu página web trae más **desventajas** que beneficios:

- Requiere de un motor interno llamado Sizzle cuyo tamaño del archivo llega a los 65kb y supera las 2000 líneas de código.
- Una tarea tan sencilla como seleccionar un elemento del DOM para poder manipularlo llega a ralentizar la carga de nuestra web.
- Tiene una gestión compleja sobre estilos CSS. Su lenguaje no es estándar y por tanto, es difícil de gestionar con CSS.
- Hay librerías más modernas como React o Angular que nos permiten hacer lo mismo que jQuery, pero siguiendo un estándar, y por tanto, nos permiten tener un código más limpio.

Lo ideal es **NO depender de ninguna librería** para crear sitios web con resultados profesionales y, además, los tiempos de carga serán mucho mejores.

Más adelante veremos algunas nociones y características principales de jQuery.

4.2.1 Interactividad con HTML5

Todos los elementos interactivos mencionados en el apartado 3.1 ya se han visto, por lo que en este apartado veremos tres etiquetas adicionales: `details`, `summary` y `dialog`.

4.2.1.1 La etiqueta <details>

Esta etiqueta oculta una zona de la página web a través de un **icono en forma de triángulo**, que se sitúa en la parte derecha del elemento. Si se pulsa sobre este, se hace visible la parte recogida dentro de esta etiqueta y, cuando se vuelve a pulsar, se oculta o viceversa.

Su **sintaxis** es la siguiente:

Oculto-visible: `<details> …… </details>`

Visible-oculto: `<details open> …… </details>`

Será el atributo ‘`open`’, situado en la etiqueta `<details>`, el que determine si el elemento objeto de la interacción aparece visible al inicio de la carga de la página o no.

Ejemplo sin el atributo ‘open’; **no aparece visible**.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Etiqueta details</title>
</head>
<body>
    <p> Si no aparece el atributo open incluido en la etiqueta details ...</p>
    <details>
        <p>... este texto aparece oculto hasta que se despliega pulsando sobre Detalles </p>
    </details>
</body>
</html>
```

Si no aparece el atributo open incluido en la etiqueta details ...

▶ Detalles

Ejemplo con atributo ‘open’; **aparece visible**.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Etiqueta details visible</title>
</head>
<body>
    <p> Si el atributo open está incluído en la etiqueta details ...</p>
    <details open>
        <p>... este texto aparece visible hasta que se pulsa sobre Detalles </p>
    </details>
</body>
</html>
```

Si el atributo open está incluído en la etiqueta details ...

▼ Detalles

... este texto aparece visible hasta que se pulsa sobre Detalles

4.2.1.2 La etiqueta ‘summary’

La propiedad ‘`summary`’ siempre aparece contenida dentro de una etiqueta `<details>`. Se trata del elemento que **modifica el valor de la etiqueta de texto que aparece junto al icono de triángulo** sobre el que se debe pulsar para mostrar u ocultar el texto. Si no se utiliza este elemento, se muestra el valor por defecto “detalles”.

Su **sintaxis** es la siguiente:

```
<details>
      <summary>
            ……
      </summary>

</details>
```

A continuación, se presenta un **ejemplo** con el atributo ‘open’; aparece visible.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Etiqueta summary</title>
</head>
<body>
    <p> Si el atributo open está incluído en la etiqueta details ...</p>
    <details open>
        <summary> PULSA </summary>
        <p>... este texto aparece visible hasta que se pulsa sobre PULSA. </p>
    </details>
</body>
</html>
```

Si el atributo open está incluído en la etiqueta details ...

▼ PULSA

... este texto aparece visible hasta que se pulsa sobre PULSA.

4.2.1.3 La etiqueta 'dialog'

La etiqueta `<dialog>` define el contenido de una ventana de diálogo independiente de la página web, es decir, **abre una nueva ventana que se superpone a la principal**. Puede incorporar el atributo `'open'`, que indica que el cuadro de diálogo está activo y el usuario puede interactuar con él.

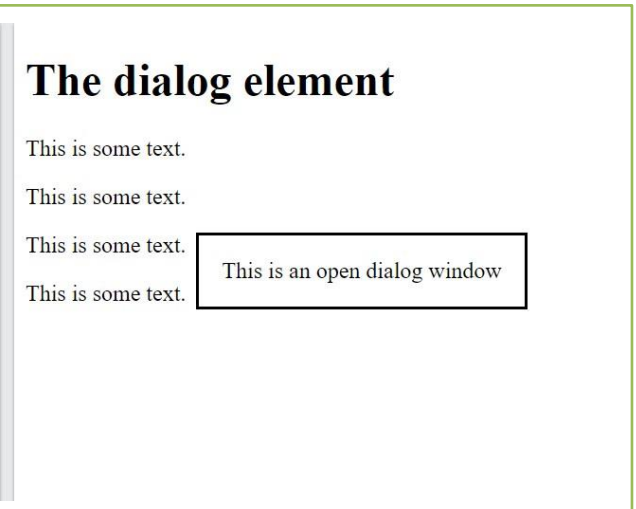
Su **sintaxis** es la siguiente: `<dialog> …… </dialog>`

La **posición por defecto** en la que se mostrará este elemento es **centrada horizontalmente**. De la misma forma, la apariencia por defecto será con color de fondo blanco. Si se desean **modificar** estas y otras propiedades, es posible hacerlo utilizando el pseudoelemento ***::backdrop***.

```css
dialog::backdrop {
    background-color: rgba(255, 255, 0, 0.7);
}
```

A continuación, se presenta un **ejemplo**:

```
<!DOCTYPE html>
<html>
<body>

<h1>The dialog element</h1>

<p>This is some text.</p>

<p>This is some text.</p>

<dialog open>This is an open dialog window</dialog>

<p>This is some text.</p>

<p>This is some text.</p>

</body>
</html>
```



4.2.2 Interactividad con CSS3

Podemos utilizar las reglas de estilo para simular la interacción con los botones, enlaces, elementos de formulario, etc. Para ello nos servimos de las pseudoclases `link`, `visited`, `hover`, `active`, y `focus` que ya vimos.

En la siguiente tabla se muestran algunos de los muchos usos que se le pueden dar a las hojas de estilo para dotar a una interfaz de mayor interactividad.

Mapas interactivos	Creando cualquier **mapa de imagen** que**, al pasar el puntero** del ratón por encima, **despliegue una información adicional**, haga una sustitución de una imagen, haga un cambio de color, etc.
Menús de navegación	El empleo de reglas de estilo para diseñar los menús de navegación es algo muy habitual para evitar la apariencia tan poco atractiva de los enlaces. Estos menús se pueden crear: • Utilizando las propiedades de los **colores de la fuente y del fondo** y variar propiedades usando las pseudoclases `hover` para definir los colores cuando el usuario pasa por

	encima, o `active` para el momento en que el usuario hace clic sobre la opción del menú. • Utilizando **imágenes de fondo** que se pueden cambiar empleando las mismas pseudoclases.
Alternancia de imágenes	Se pueden usar la pseudoclase `hover` sobre un elemento de imagen para cambiar el archivo de imagen a mostrar cuando el usuario pasa el ratón por encima de la imagen.
Formularios más interactivos	Podemos **alterar el color de fondo y de la fuente** de cualquier elemento del formulario empleando las pseudoclases: • `hover` cuando el usuario pasa el ratón por encima del elemento del formulario. • `focus` cuando el usuario accede a un elemento ya sea con un clic de ratón o empleando la tecla de tabulación que se desplaza entre los diferentes elementos.