

HTML

2.8 Etiquetas de scripting



Autor: Lisa ERIKSEN

Fecha: 2025 / 2026

1. Descubrimos las etiquetas interactivas

Con la llegada de HTML5, se añadieron etiquetas que permiten una mayor interacción entre el usuario y la página. Estas etiquetas facilitan la creación de elementos interactivos que mejoran la experiencia del usuario.

1.1. Despegables con `<detail>` y `<summary>`

La etiqueta `<details>` permite crear un elemento desplegable que inicialmente no está visible.

```
<details>
  <summary>Titulo Principal</summary>
  <h3>Titulo</h3>
  <p>Un parrafo de presentacion</p>
</details>
```

Actividad: Imagina que creas una página para una plataforma de cine digital. Quieres mostrar información adicional sobre una película solo cuando el visitante la solicite.

▼ Más información sobre la película
Interstellar (2014)

Un grupo de exploradores viaja a través de un agujero de gusano en busca de un nuevo hogar para la humanidad. Una mezcla épica de ciencia, emociones y aventura espacial dirigida por Christopher Nolan.



Preguntas:

- ¿Qué parte del contenido está visible al principio?
- ¿Qué ocurre al hacer clic en el texto de `<summary>`?
- ¿Qué ventaja tiene esta etiqueta frente a mostrar todo el texto directamente?

1.2. Ventanas de diálogo con <dialog>

Los **modales** son ventanas que bloquean la página hasta que el usuario realiza una acción (como cerrar).

Actividad: Ahora trabajas en una web de bienvenida. Quieres que aparezca una pequeña ventana con un saludo cuando el usuario lo pida.

```
<dialog id="saludo">
  <p>👋 ¡Bienvenido/a a nuestra página!</p>
  <button onclick="document.getElementById('saludo').close()">Cerrar</button>
</dialog>

<button onclick="document.getElementById('saludo').show()">Mostrar saludo</button>
```

Preguntas:

- ¿Qué ocurre cuando haces clic en “Mostrar saludo”?
- ¿Para qué sirve el método .show()?

1.3. Crea tu propio modal

Actividad: Ahora modifica el código anterior para que el mensaje se muestre **como modal**, bloqueando el resto de la página.

Pista: Usa el método `showModal()` en lugar de `show()`.

Preguntas:

- ¿Qué diferencia notas entre `.show()` y `.showModal()`?
- ¿Puedes interactuar con la página mientras el modal está abierto?
- ¿Qué aplicaciones reales podría tener este tipo de ventana?

1.4. Modal bloqueante con contenteditable

Actividad: Ahora diseñas un pequeño bloc de notas en línea. Quieres permitir que el usuario modifique un texto directamente desde la página.

Copia este ejemplo:

```
<h2 contenteditable>📝 Haz clic aquí para editar el título</h2>  
  
<div contenteditable> Escribe aquí tus ideas... </div>
```

Puedes añadir un poco de CSS en la parte <style> de la pagina HTML y usando el class siguiente:

```
.contenteditable {  
    border: 1px solid;  
    padding: 10px;  
    width: 250px;  
}
```

Preguntas:

- Cambia el texto directamente desde el navegador.
- Actualiza la página: ¿los cambios se guardan?

Preguntas:

- ¿Qué ocurre al hacer clic dentro del texto?
- ¿Qué significa que el contenido sea “editable”?
- ¿Qué pasa con tus cambios al recargar la página?

2. Descubrimos las etiquetas de script

Comprender cómo funciona la etiqueta <script> y sus atributos para añadir interactividad y lógica a una página web.

2.1. Tu primer script JavaScript

Actividad: Imagina que eres desarrollador de una página de bienvenida y quieres mostrar un mensaje al entrar.

```
<h1>Mi primera página con JavaScript</h1>
<script> alert("¡Bienvenido a mi página!"); </script>
```

Preguntas:

- ¿Cuándo aparece el mensaje?
- ¿Qué pasaría si colocas el <script> dentro del <head>?
- ¿Qué tipo de carga se está usando aquí (síncrona, asíncrona o diferida)?

2.2. Separar el código – script externo

Actividad: Ahora el equipo quiere mantener el código más limpio y crear una pagina externa para el script JavaScript.

1. Crea un archivo nuevo llamado **saludo.js** con este contenido:

```
alert("¡Hola desde un archivo externo!");
```

2. En tu página HTML, añade la etiqueta <script> dentro del <body> y asigna como fuente tu archivo JavaScript externo.

```
<script src="saludo.js"></script>
```

Preguntas:

- ¿Por qué puede ser más práctico separarlo del HTML?
- Si el archivo saludo.js no existe, ¿qué ocurre?

2.3. Scripts asincrónos diferidos

Actividad: Eres parte de un equipo que diseña una web con muchos scripts. Debes decidir cuál usar para optimizar la carga de la página.

```
<h2>Prueba de carga</h2>
<script src="uno.js" async></script>
<script src="dos.js" defer></script>
```

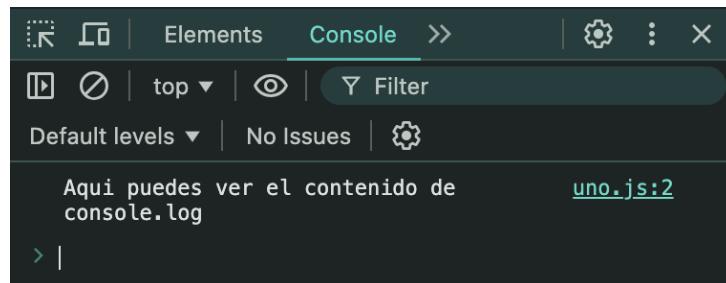
1. Crea un archivo nuevo llamado **uno.js** con este contenido:

```
console.log("Script 1 (async)");
```

2. Crea un archivo nuevo llamado **dos.js** con este contenido:

```
console.log("Script 2 (defer)");
```

3. Abre tu página en el navegador y utiliza el inspector de código. En la consola podrás ver el resultado de tu script JavaScript.



Preguntas:

- ¿Qué mensaje aparece primero en la consola?
- ¿Por qué crees que el navegador ejecuta los scripts en ese orden?
- ¿Cuándo conviene usar `async` y cuándo `defer`?

2.4. Modulos – Reutiliza tu código

Actividad: Ahora trabajas en un proyecto grande y necesitas compartir variables o funciones entre varios archivos.

1. Crea un archivo nuevo llamado **datos.js** con este contenido:

```
export const nombre = "Lucía";  
✓ export function saludar() {  
  |   console.log(`Hola ${nombre} !`);  
  }  
  }
```

2. En tu HTML, importa la función saludar en la etiqueta **<script>**:

```
<script type="module">  
  |   import { saludar } from './datos.js';  
  |   saludar();  
</script>
```

Preguntas:

- ¿Qué indica el atributo `type="module"`?
- ¿Cómo cambia el comportamiento del script con respecto a un `<script>` normal?
- ¿Qué ventaja tiene usar `import` y `export`?

2.5. Compatibilidad – Navegadores modernos y antiguos

Actividad: Algunos usuarios usan navegadores que **no soportan módulos**. Tu tarea es mostrar mensajes diferentes según el navegador.

Copia este código en tu pagina HTML y utiliza la consola del inspector de código para observar el resultado:

```
<script nomodule>
| console.log("Soy un navegador antiguo 😞");
</script>
```

```
<script type="module">
| console.log("Soy un navegador moderno 😎");
</script>
```

Preguntas:

- ¿Qué parte se ejecuta en tu navegador actual?
- ¿Por qué es útil el atributo nomodule?

2.6. Plantillas invisibles – Etiqueta <template>

Actividad: Vas a crear una lista de alumnos que se generará dinámicamente con JavaScript.

- Escribe el código HTML siguiente en tu pagina y observa en tu navegador:

```
<h2>Lista de alumnos</h2>
<table id="tabla">
  <tr>
    <th>Nombre</th>
    <th>Edad</th>
  </tr>

  <template id="alumno">
    <tr>
      <td>Pablo</td>
      <td>22</td>
    </tr>
  </template>
</table>
```

Prueba:

- ¿Ves la fila del usuario? ¿Por qué el contenido del <template> no aparece?

- Añade este script JavaScript en tu pagina HTML y observa en tu navegador.

```
<script>
  const template = document.getElementById("alumno");
  const tabla = document.getElementById("tabla");
  const copia = template.content.cloneNode(true);
  tabla.appendChild(copia);
</script>
```

Preguntas:

- ¿Qué hace la línea const template = document.getElementById("alumno");?
- ¿Para qué sirve const tabla = document.getElementById("tabla");?
- ¿Qué significa template.content.cloneNode(true); y por qué lo usamos?
- ¿Qué hace tabla.appendChild(copia); y qué cambia en la página al ejecutarlo?