ThePower business school

Etiquetas de scripting HTML🤖👨‍💻

A medida que aprendemos HTML vemos que se trata de un lenguaje de marcas estupendo y muy potente, pero también nos vamos dando cuenta que ciertos detalles no se pueden cubrir sólo con HTML y CSS. Para ello tenemos que hacer uso de un lenguaje de programación llamado **Javascript**.

Etiquetas interactivas

En HTML5 se añadieron etiquetas que posibilitan la interacción del usuario con la página. Por esta razón, el usuario es parte fundamental de este proceso.

Etiquetas interactivas: Estas etiquetas pueden ser

<details>. Crea un elemento desplegable, que contendrá todo el contenido. Actúa como un div, pero inicialmente no aparecerá desplegado. Lo que veremos será el título que le hayamos dado con el <summary> junto a una flecha que indica que es un desplegable.

```
<details>
  <summary>Más información</summary>
  <h2>Nombre</h2>
  <img src="foto.jpg" alt="Foto" />
</details>
```

<summary>. Título del elemento que se verá estando o no desplegado.

<dialog>. Ventana de diálogo que puede contener y mostrar información. Se utiliza sobre todo para enviar alertas al usuario a través de la pantalla. Es la mejor opción para sustituir al alert de javascript.

```
<dialog id="alert">
  <p>¡Hola!</p>
  <button id="close" onclick="document.getElementById('alert').close()">Ok!</button>
</dialog>
<button id="show" onclick="document.getElementById('alert').show()">Mostrar</button>
```

`onclick` será un evento de javascript de tipo clic, que indicará una acción que debe ocurrir cuando se produzca ese clic.

Otra opción muy buena es la de crear modales desde HTML. Los modales son ventanas que aparecen en pantalla y que no se quitan hasta que el usuario realiza la acción que se le pide desde el modal. Por ejemplo, muchos avisos de cookies que salen en algunas webs o cuando tienes un bloqueador de anuncios y te salta un modal que, hasta que no desactivas el bloqueador no te deja navegar.

Para crear este modal podemos usar la función showModal() como en el ejemplo:

```
<dialog id="gandalfModal">
  <img src="gandalf.jpg" alt="Gandalf" />
  <p>¡NO.. PUEDES... PASAR!</p>
  <button id="close" onclick="document.getElementById('gandalfModal').close()">Ok!</button>
</dialog>
<button id="show" onclick="document.getElementById('gandalfModal').showModal()">Mostrar</button>
```

Si copiamos y pegamos esto veremos el siguiente modal:

Si, por otro lado, queremos dar la posibilidad al usuario de modificar un elemento, existe el atributo contentEditable

```
<h3 contenteditable>Título</h3>
<div contenteditable>
  <p>Este es un párrafo modificable de ejemplo</p>
</div>
```

Pero esto no persistirá, es decir, si modificamos el contenido y recargamos la página no veremos reflejados los cambios. Esto quiere decir que el atributo funciona como si cambiásemos el contenido del html a través del inspector de elementos del navegador. Para hacer que los cambios persistan deberíamos guardar el contenido editado mediante javascript.

Etiquetas interactivas: Qué hemos aprendido

1. Qué son y para qué se usan las etiquetas interactivas.
2. Cuáles existen y qué atributos las acompañan.
3. Ejemplo práctico de crear modales, algo que os será muy útil.

Etiquetas script

Si bien HTML y CSS son capaces de realizar muchas de las necesidades que tendremos cuando queramos hacer una web, tienen sus limitaciones. Ahí es donde entrará javascript, que se enlazarán al HTML con la etiqueta <script>.

Esta etiqueta sirve para cargar contenido Javascript en nuestro documento HTML y podrá recibir los siguientes atributos:

Atributo	Valor	Descripción
src	URL	Dirección URL del script externo a cargar.
type	tipo	Tipo de script a cargar. Si se omite, se asume text/javascript como valor.
nomodule	boolean	Si se define este atributo, el script no se carga en navegadores modernos.
async	boolean	Ejecuta el script cuando se haya descargado, sin bloquear el navegador.

defer	boolean	Aplaza la ejecución del script, lo ejecuta al final, cuando haya descargado todo.

Los tipos de carga pueden ser síncrono, que es el que se establece por defecto y paraliza la página hasta que la carga del contenido javascript esté listo, carga asíncrona que usaréis mucho y sirve para descargar el script en cuanto esté disponible y no detiene la carga del HTML. Una vez descargado lo ejecuta y detiene la carga HTML. Una vez ejecutado terminará de cargar el resto del HTML. Y por último el atributo defer, que hace que la carga del script se realice al final del todo, una vez el resto de la página ha cargado.

De esta forma, además podemos cargar javascript en línea o externo. Funciona igual que el código CSS:

Ejemplo de código javascript en línea:

```
<script>
  alert('¡Hola!');
</script>
```

Ejemplo de javascript externo:

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

En los últimos años Javascript ha incorporado los módulos. Estos módulos posibilitan importar funcionalidades mediante import y export. Para ello, debemos añadir en nuestra etiqueta <script> el atributo type=”module”:

```
<script type="module">
   import { pi } from './pi-module.js';
</script>
```

Al ser una funcionalidad “nueva” no tiene soporte con la totalidad de los navegadores, por lo que podemos asignar una alternativa que será el atributo nomodule por si el navegador no contempla

el uso de estos módulos de javascript:

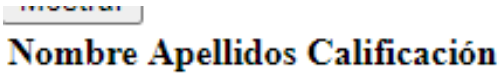
```
<script type="module">
  /* Código para navegadores modernos */
  import { pi } from './pi-module.js';
</script>
<script nomodule>
  /* Código para navegadores antiguos */
</script>
```

Por último, puede darse el caso que tengamos código reutilizable que queremos usar en varios sitios de nuestra aplicación. Para facilitar esto, existe la etiqueta <template>

Si usamos en nuestro documento HTML este código:

```
<table id="tabla">
  <tr>
    <th>Nombre</th>
    <th>Apellidos</th>
    <th>Calificación</th>
  </tr>
  <template id="usuario">
    <tr>
      <td>1</td>
      <td>2</td>
      <td>3</td>
    </tr>
  </template>
</table>
```

y después recargamos el navegador, veremos que no hay ni rastro de esa fila con el 1, 2 y 3, pero si abrimos el inspector de elementos veremos en nuestro árbol html que sí que se ha renderizad como un <template>:



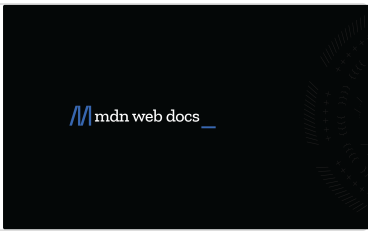

```
<table id="tabla">
  <tbody>
    <tr>
      <th>Nombre</th>
      <th>Apellidos</th>
      <th>Calificación</th>
    </tr>
    <template id="usuario"> == $0
      #document-fragment
        <tr>
          <td>1</td>
          <td>2</td>
          <td>3</td>
        </tr>
    </template>
  </tbody>
</table>
</div>
```

Este contenido será inerte hasta que lo clonemos a través de javascript y lo añadamos de forma dinámica.

: The Content Template element - HTML: HyperText Markup Language | MDN

The HTML element is a mechanism for holding HTML that is not to be rendered immediately when a page is loaded but may be instantiated subsequently during runtime using JavaScript. Think of a template as a content fragment that is being stored

https://developer-mozilla-org.translate.goog/en-US/docs/Web/HTML/Element/template?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=sc



Etiquetas script: Qué hemos aprendido

1. Cuáles son las etiquetas HTML de scripts.
2. Los tipos de carga de contenido javascript que existen y su significado.
3. Cómo usar los módulos de javascript.

Eventos

Un **evento Javascript** es una característica especial que ha sucedido en nuestra página y a la cuál le asociamos una funcionalidad, de modo que se ejecute cada vez que suceda dicho evento.

Existen multitud de eventos y para usarlos es tan fácil como elegir la etiqueta a la que queremos ponerle el evento, añadirle un atributo que detecte si se produce o no el evento (que tendrá el prefijo on) y por último indicar la función/acción que debe ocurrir cuando el evento ocurra.

```
<div>
  <img src="gandalf.jpg" alt="Gandalf" onClick="alert('¡Has hecho clic!')" />
</div>
```

En el ejemplo, cuando se haga click en la imagen saldrá el alert con el texto que hemos indicado. Como hemos dicho existen multitud de eventos, os vamos a dejar los más conocidos.

Eventos: Eventos de documento

`onLoad`, que ocurrirá cuando la página cargue.

```
<img style="width: 300px;"
      src="https://e00-marca.uecdn.es/assets/multimedia/imagenes/2022/10/18/16661071524623.jpg"
      id="gandalfImage" alt="Gandalf">
 <p id="textId"></p>

<script>
    document.getElementById("gandalfImage").addEventListener("load", loadImage);
    function loadImage() {
        document.getElementById("textId").innerHTML = "Image loaded";
    }
</script>
```

`onScroll`, cuando se haga scroll en la página.

```
<textarea style="width:100% height: 100px" id="scrollExample">
        HTML stands for Hyper Text Markup Language.
      It is used to design web pages using markup language.
      HTML is the combination of Hypertext and Markup language.
      Hypertext defines the link between the web pages.
      Markup language is used to define the text document
      within tag which defines the structure of web pages.
      HTML is a markup language which is used by the browser
      to manipulate text, images and other content to
      display it in required format.
    </textarea>

    <p id="example"></p>

<script>
    document.getElementById(
```

```
      "scrollExample").addEventListener("scroll", onScrollAction);

    function onScrollAction() {
        document.getElementById(
          "example").innerHTML = "Has hecho scroll en el textArea.";
    }
</script>
```

Eventos: Eventos de foco

Cuando elementos como input, textarea, select o anchor pueden ser seleccionables por el usuario y ocurrirán cuando se haga/se pierda foco sobre ellos.

`onBlur`, el elemento ha perdido foco.

```
<h2>onblur event example</h2>
<input onblur="myFunction()">

<script>
    function myFunction() {
        alert('foco perdido!');
    }
</script>
```

`onFocus`, el elemento está en foco.

```
<h2>onFocus event example</h2>
<input onfocus="myFunction()">

<script>
    function myFunction() {
        alert('Estoy en foco!');
    }
</script>
```

Eventos: Eventos de ratón

Se utilizan para acciones que el usuario hace a través del ratón.

onClick	El usuario ha pulsado (y soltado) el elemento.
onDblclick	El usuario ha hecho doble clic en el elemento.

onMousedown	El usuario ha pulsado (aún sin haber soltado) el elemento.
onMouseup	El usuario ha soltado el botón pulsado en un elemento.
onMousemove	El usuario ha movido el ratón.
onMouseenter	El usuario ha movido el ratón dentro de un elemento.
onMouseleave	El usuario ha movido el ratón fuera de un elemento.
onMouseout	El usuario ha movido el ratón fuera de un elemento (bubbles).
onMouseover	El usuario ha movido el ratón dentro de un elemento (bubbles).
onWheel	El usuario ha movido la rueda del ratón.

```
<!-- EJEMPLO DE ONCLICK QUE CUANDO HAGAMOS CLICK EN EL <p>

EL TEXTO SE CAMBIARÁ A ROJO -->

<p id="example" onclick="myFunction()">Haz click para cambiarme de color.</p>

<script>
function myFunction() {
  document.getElementById("example").style.color = "red";
}
</script>
```

Para no llenar la página con ejemplos de código en los que simplemente modificamos el tipo de evento, os dejamos el ejemplo de código de arriba que es aplicable a cualquier tipo de evento de ratón. Tan solo tendréis que cambiar el tipo de evento en la línea que os hemos remarcado para poder probarlo!

Eventos: Qué hemos aprendido

1. Qué son los eventos de javascript.
2. Las diferentes categorías o grupos que existen.