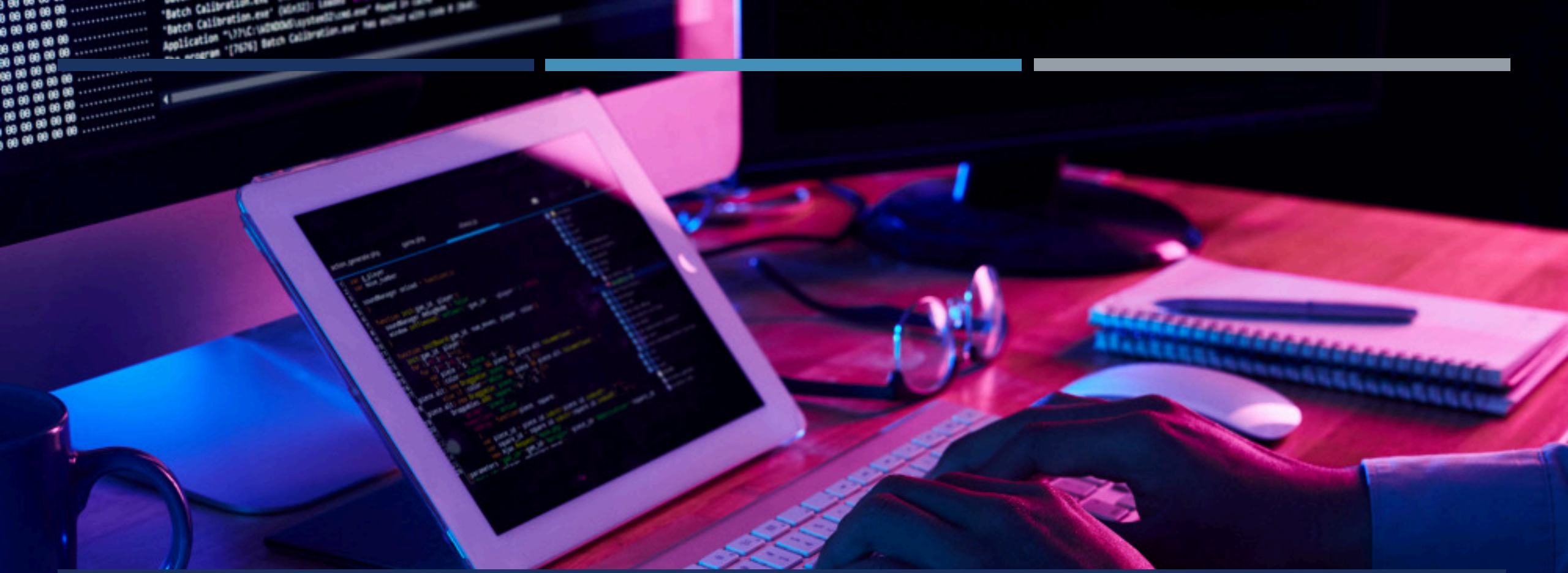


PROGRAMACIÓN WEB EN EL ENTORNO CLIENTE

MF0491_3

Gobierno de Canarias
Servicio Canario de Empleo

FUERTEVENTURA 2000
Centro de Formación Profesional

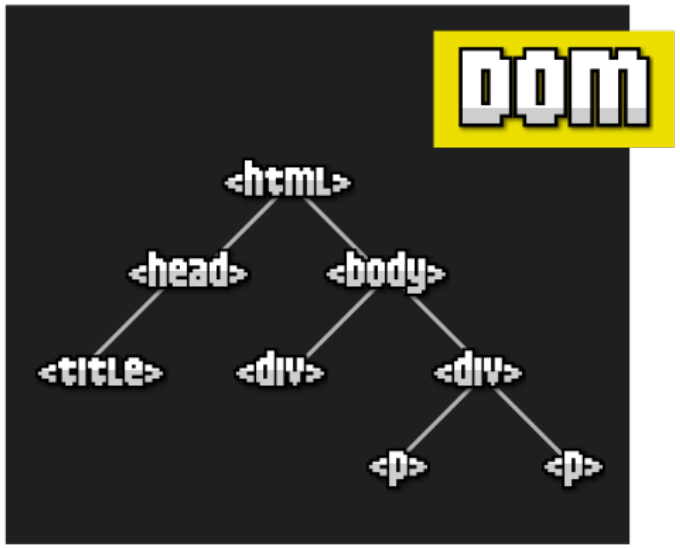
PIEC
PLAN INTEGRAL DE EMPLEO DE CANARIAS

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL
MINISTERIO DE TRABAJO Y ECONOMÍA SOCIAL
SERVICIO PÚBLICO DE EMPLEO ESTATAL
SEPE

Realizado por Manuel Macías

DOM

Las siglas DOM significan Document Object Model, o lo que es lo mismo, la estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol DOM (o simplemente DOM).



En Javascript, cuando nos referimos al DOM nos referimos a esta estructura de árbol, mediante la cuál podemos acceder a ella y modificar los elementos del HTML desde Javascript, añadiendo nuevas etiquetas, modificando o eliminando otras, cambiando sus atributos HTML, añadiendo clases, cambiando el contenido de texto, etc...

El objeto document

En Javascript, la forma de acceder al DOM es a través de un objeto llamado document, que representa el árbol DOM de la página o, más concretamente, la página de la pestaña del navegador donde nos encontramos.
En su interior pueden existir varios tipos de elementos, pero principalmente serán element o node :

- element: no es más que la representación genérica de una etiqueta: HTMLElement.
- node: es una unidad más básica, la cuál puede ser o un nodo de texto.

Todos los elementos HTML, dependiendo del elemento que sean, tendrán un tipo de dato específico.

Algunos ejemplos:

Tipo de dato genérico	Tipo específico	Etiqueta	Descripción
HTMLElement	HTMLDivElement		Etiqueta divisoria (en bloque).
HTMLElement	HTMLSpanElement		Etiqueta divisoria (en línea).
HTMLElement	HTMLImageElement	<img>	Imagen.
HTMLElement	HTMLAudioElement	<audio>	Contenedor de audio.

Aunque los métodos tradicionales para realizar búsquedas de elementos en el DOM son más que suficiente para buscar elementos, actualmente tenemos a nuestra disposición dos nuevos métodos de búsqueda de elementos que son mucho más cómodos y prácticos si conocemos y dominamos los selectores CSS.

Es el caso de los métodos **.querySelector()** y **.querySelectorAll()**:

querySelector()

El método **.querySelector()** devuelve el primer elemento que encuentra que encaja con el selector CSS suministrado por parámetro. Siempre devuelve un solo elemento y en caso de no coincidir con ninguno, devuelve null:

```
const page = document.querySelector("#page");          // <div id="page"></div>
const info = document.querySelector(".main .info");    // <div class="info"></div>
```

querySelectorAll()

Por otro lado, el método **.querySelectorAll()** realiza una búsqueda de elementos en el DOM, sólo que como podremos intuir por ese **All()**, devuelve un con todos los elementos que coinciden con el CSS.

```
// Obtiene todos los elementos con clase "info"
const infos = document.querySelectorAll(".info");
// Obtiene todos los elementos con atributo name="nickname"
const nicknames = document.querySelectorAll('[name="nickname"]');
// Obtiene todos los elementos <div> de la página HTML
const divs = document.querySelectorAll("div");
```

El método **.querySelectorAll()** siempre nos devolverá un **array** de elementos. El número de elementos del **array** dependerá de los que encuentre.

Creando elementos con **createElement()**

Mediante el método **.createElement()** podemos crear un HTML en memoria (¡no estará insertado aún en nuestro documento HTML!).

Con dicho elemento almacenado en una variable o constante, podremos modificar sus características o contenido, para posteriormente insertarlo en una posición determinada del DOM o documento HTML.


```
const div = document.createElement("div"); // Creamos un <div></div>
const span = document.createElement("span"); // Creamos un <span></span>
 const img = document.createElement("img"); // Creamos un <img>
```

El método .appendChild()

Uno de los métodos más comunes para añadir un elemento HTML creado con Javascript es .appendChild(). Como su propio nombre indica, este método añade o inserta un nuevo elemento, como si fuera un hijo al final de todos los hijos del elemento sobre el que se realiza.

El método .prepend() y .append()

Con estos dos métodos podemos insertar elementos en sus elementos hijos, al principio o al final. El método .prepend() permite insertar uno o varios elementos antes del primer elemento hijo de nuestro elemento base. En el caso de append() ocurre lo mismo, pero después del último elemento hijo:

Método .addEventListener()

Con el método **.addEventListener()** permite añadir una escucha del evento indicado (primer parámetro), y en el caso de que ocurra, se ejecutará la función asociada indicada (segundo parámetro).

```
const button = document.querySelector("button");
button.addEventListener("click", function() {
alert("Hello!");
});
```

El objeto Event

Cuando se disparan ciertos eventos, hay casos en los que nos podría interesar obtener información relacionada con la naturaleza del evento en cuestión. Por ejemplo, si estamos escuchando un evento de tipo click de ratón, nos podría interesar saber con que botón del ratón se ha hecho click, en que punto concreto de la pantalla se ha hecho click, etc.

Estos detalles se pueden obtener de forma opcional a través de un objeto event que se proporciona en la función asociada al evento. Para ello, sólo es necesario indicar un primer parámetro en la función que gestiona el evento, y dicho parámetro, será de tipo evento con dicha información asociada.

Observa el siguiente ejemplo de código:

```
const button = document.querySelector("button");
button.addEventListener("click", (event) => {
console.log(event);
});
```

BOM – Browser Object Model

Mediante BOM, es posible redimensionar y mover la ventana del navegador, modificar el texto que se muestra en la barra de estado y realizar muchas otras manipulaciones no relacionadas con el contenido de la página HTML.

El mayor inconveniente de BOM es que, al contrario de lo que sucede con DOM, ninguna entidad se encarga de estandarizarlo o definir unos mínimos de interoperabilidad entre navegadores.

Algunos de los elementos que forman el BOM son los siguientes:

- Crear, mover, redimensionar y cerrar ventanas de navegador.
- Obtener información sobre el propio navegador.
- Propiedades de la página actual y de la pantalla del usuario.
- Gestión de cookies.

JSON) – JavaScript Object Notation

Es un formato de intercambio de datos muy ligero y está basado en la notación para la representación de objetos. A pesar de estar basado en JavaScript, es independiente del lenguaje. Puede ser utilizado para intercambiar datos entre programas escritos en lenguajes totalmente diferentes. Es un formato de texto, por lo que puede ser leído por máquinas y humanos.
http://www.JSON.org/.

JSON define seis tipos de valores: **objects, arrays, strings, numbers, booleans** y el valor especial **null**. Los espacios (espacios en blanco, tabuladores, retornos de carro y nueva línea) pueden introducirse antes o después de cualquier valor, sin afectar a los valores representados. Esto hace que un texto JSON sea mucho más fácil de leer por humanos.
Un objeto JSON es un contenedor, no ordenado de parejas clave/valor. Una clave puede ser un string, y un valor puede ser un valor JSON (tanto un array como un objeto). Los objetos JSON se pueden anidar hasta cualquier profundidad. Un array JSON es una secuencia ordenada de valores, donde un valor puede ser un valor JSON (tanto un array como un objeto).

Usando la API de almacenamiento web (LocalStorage)

La API de almacenamiento web proporciona los mecanismos mediante los cuales el navegador puede almacenar información de tipo clave/valor, de una forma mucho más intuitiva que utilizando cookies.

```
// Guardar datos
localStorage.setItem("listado", "Listado de
Alumnos inscritos");

// Recuperar datos
localStorage.getItem("listado");

// Borrar Datos
localStorage.removeItem("alumno");

//  Vaciar LocalStorage
localStorage.clear();
```

Método Fetch

Realizar Peticiones AJAX de una manera más simple de que como se hacía antes.
Una petición AJAX es una llamada a un servicio REST, API REST, Backend y recibir resultados externos.

```
fetch('https://reqres.in/api/users?page=2') // Realizamos la petición
    .then(data => data.json()) // Promesa para convertir en JSON
    .then(users => { // Promesa para devolver los datos
        usuarios = users.data;
        console.log(usuarios);
           usuarios.map((user, i) => {
           let nombre = document.createElement('h3’);
           nombre.innerHTML = (i+1) + '.- '+ user.first_name + ' ' + user.last_name;
           div_usuarios.appendChild(nombre);
        });
});
```

Programación web en el entorno cliente

GRACIAS

MANUEL MACÍAS

TUTORIAS@MANUELMACIAS.ES

PROGRAMACIÓN JAVASCRIPT