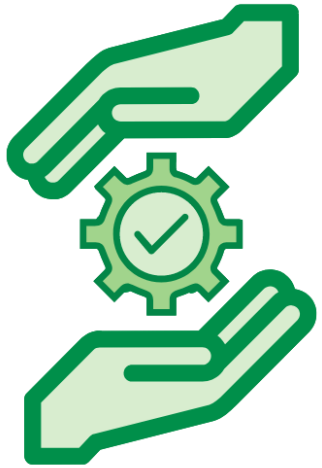


Bootcamp de Desarrollo Web

Clase 10

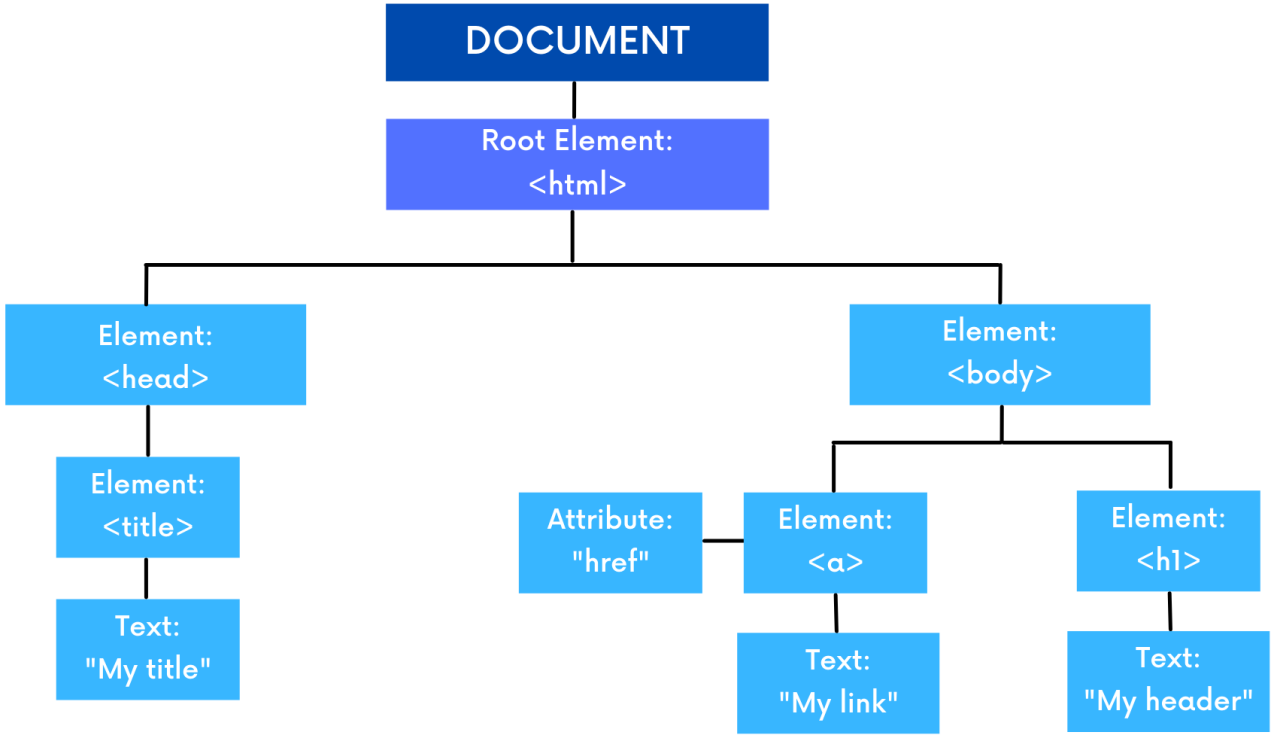
Javascript & Jquery

¿Qué es el DOM?

El **DOM**, o **Document Object Model** (Modelo de Objetos del Documento en español), es la representación la estructura de la página web y permite a los desarrolladores web acceder y manipular dinámicamente los elementos de la página mediante código JavaScript.

Cada elemento en el documento (como encabezados, párrafos, imágenes, etc.) es representado por un **nodo** en el árbol del DOM, y estos nodos pueden ser manipulados para cambiar el contenido, el estilo y la estructura de la página web de forma dinámica.

Breve Historia de Javascript



Interactuando con el DOM

Todo en el DOM recae en alguna de estas dos categorías: selección de elementos y manipulación de elementos.

Selección: Hacemos referencia a todos los elementos que queremos hacer dinámicos desde nuestro código HTML y les asignamos variables en nuestro archivo JavaScript.

Manipulación: Una vez que hemos seleccionado y enlazado las variables, creamos las diversas funciones responsables de la manipulación y luego las enlazamos a las variables.

Interactuando con el DOM

Para manipular elementos HTML con JavaScript, es necesario que JavaScript sea **consciente** de la existencia de dichos elementos, lo que comúnmente se conoce como **selección** de elementos. La selección implica enlazar o hacer referencia a los elementos.

En el DOM, no hay una única forma de localizar y referenciar un elemento para su manipulación. Esto dependerá del selector utilizado en la etiqueta del elemento.

Para lograr esto, se asigna el elemento a una variable, siguiendo un formato específico. Es importante destacar que todos los selectores del DOM están precedidos por el objeto de documento y un punto.

Query Selector

Este método devuelve el primer elemento que coincide con un selector CSS especificado.

Sintaxis: **document.querySelector(selector)**

Ejemplo:

```
<script>
  // Seleccionar el primer párrafo dentro del div con id "miDiv"
  var primerParrafo = document.querySelector('#miDiv .parrafo');
  console.log(primerParrafo.textContent); // Resultado: "Primer párrafo'
</script>
```

Query Selector All

Este método devuelve una NodeList (lista de nodos) que contiene todos los elementos que coinciden con un selector CSS especificado.

Sintaxis: **document.querySelectorAll(selector)**

Ejemplo:

```
  // Seleccionar todos los párrafos dentro del div con id "miDiv"
  var todosLosParrafos = document.querySelectorAll('#miDiv .parrafo');

  // Iterar sobre la NodeList e imprimir el texto de cada párrafo
  todosLosParrafos.forEach(function(parrafo) {
    console.log(parrafo.textContent);
  });
  // Resultado:
  // "Primer párrafo"
  // "Segundo párrafo"
</script>
```

getElementsByClassName

El método **getElementsByClassName** es otro método que se utiliza para seleccionar elementos del DOM basándose en sus clases.

A diferencia de querySelector y querySelectorAll, getElementsByClassName se enfoca específicamente en seleccionar elementos por su **clase**.

El siguiente ejemplo, **getElementsByClassName('parrafo')** devuelve una **HTMLCollection** que contiene todos los elementos con la clase "parrafo". Luego, se utiliza un bucle for para iterar sobre la colección y se imprime el texto de cada párrafo.

getElementsByClassName

```
<head>
  <meta charset="UTF-8">
  <title>getElementsByClassName Ejemplo</title>
</head>
<body>
  <div id="miDiv">
    <p class="parrafo">Primer párrafo</p>
    <p class="parrafo">Segundo párrafo</p>
    <p class="otraClase">Tercer párrafo</p>
  </div>

  <script>
    // Seleccionar todos los elementos con la clase "parrafo"
    var parrafos = document.getElementsByClassName('parrafo');

    // Iterar sobre la HTMLCollection e imprimir el texto de cada párr
    for (var i = 0; i < parrafos.length; i++) {
      console.log(parrafos[i].textContent);
    }
    // Resultado:
    // "Primer párrafo"
    // "Segundo párrafo"
  </script>
</body>
</html>
```

getElementById

El método **getElementById** es otro método utilizado para seleccionar un elemento del DOM, pero en este caso, se basa en el atributo id del elemento.

Cada elemento en HTML puede tener un identificador único, y **getElementById** se utiliza para seleccionar el elemento que tiene un ID específico.

getElementById

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>getElementById Ejemplo</title>
</head>
<body>
  <div id="miDiv">
    <p id="primerParrafo">Primer párrafo</p>
    <p id="segundoParrafo">Segundo párrafo</p>
  </div>

  <script>
    // Seleccionar el elemento con el id "primerParrafo"
    var parrafo = document.getElementById('primerParrafo');
    console.log(parrafo.textContent); // Resultado: "Primer párrafo"
  </script>
</body>
</html>
```

GetElementsByTagName

El método **getElementsByTagName** es utilizado para seleccionar elementos del DOM basándose en su nombre de etiqueta (tag name).

Devuelve una colección de todos los elementos con el nombre de etiqueta especificado.

GetElementsByTagName

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>getElementsByTagName Ejemplo</title>
</head>
<body>
  <div>
    <p>Primer párrafo</p>
    <p>Segundo párrafo</p>
    <span>Tercer elemento</span>
  </div>

  <script>
    // Seleccionar todos los elementos 'p' en el documento
    var parrafos = document.getElementsByTagName('p');

    // Iterar sobre la HTMLCollection e imprimir el texto de cada párraf
    for (var i = 0; i < parrafos.length; i++) {
      console.log(parrafos[i].textContent);
    }
    // Resultado:
    // "Primer párrafo"
    // "Segundo párrafo"
  </script>
</body>
</html>
```

¿Qué son los Eventos?

En JavaScript, los eventos son acciones o sucesos que ocurren en el navegador y que pueden ser detectados y manejados mediante código.

Los eventos pueden estar relacionados con la interacción del usuario, como hacer **clic** en un elemento, **mover el ratón**, o también pueden ser eventos relacionados con el ciclo de vida de la página, como la carga completa del documento.

Event Listener

En JavaScript, un **event listener** (escucha de eventos) es una función que espera la ocurrencia de un evento específico y responde a él ejecutando un conjunto de instrucciones.

Los event listeners son fundamentales para la programación de eventos y la interactividad en las aplicaciones web.

La función addEventListener se utiliza para adjuntar un event listener a un elemento del DOM. Su sintaxis básica es la siguiente:

```
elemento.addEventListener(evento, funciónDeManejo);
```

removeEventListener

removeEventListener es un método que se utiliza en JavaScript para eliminar un event listener previamente añadido mediante addEventListener.

Esto es útil cuando ya no se necesita escuchar un evento en un elemento específico o cuando deseas cambiar la función de manejo asociada a un evento.

```
elemento.removeEventListener(evento, funciónDeManejo);
```

Event Listener

elemento: Es el elemento del DOM al cual se le está añadiendo el event listener.

evento: Es el tipo de evento al cual se está suscribiendo el event listener (por ejemplo, "**click**", "**mouseover**", "**keydown**", etc.).

funciónDeManejo: Es la función que se ejecutará cuando ocurra el evento.

```
elemento.addEventListener(evento, funciónDeManejo);
```

Event Listener

Es importante notar que, para que **removeEventListener** funcione correctamente, la función de manejo que estás intentando quitar debe ser la misma función de manejo que se usó al añadir el event listener originalmente.

Esto significa que, si estás utilizando funciones anónimas, no podrás quitar el event listener directamente ya que no tienes una referencia a esa función.

Eventos de Interacción del Usuario

click: Se dispara cuando se hace clic en un elemento.

```
document.getElementById('miBoton').addEventListener('click', function() {
  console.log('Botón clickeado');
});
```

Eventos de Interacción del Usuario

mouseover y mouseout: Se disparan cuando el ratón entra o sale de un elemento.

```
document.getElementById('miElemento').addEventListener('mouseover', function() {
  console.log('El ratón está sobre el elemento');
});

document.getElementById('miElemento').addEventListener('mouseout', function() {
  console.log('El ratón salió del elemento');
});
```

Eventos de Interacción del Usuario

keydown, keypress y keyup: Se disparan cuando se presiona una tecla, cuando se mantiene presionada y cuando se suelta, respectivamente.

```
document.addEventListener('keydown', function(event) {
  console.log('Tecla presionada:', event.key);
});
```

Eventos de Interacción del Usuario

change: Se dispara cuando el valor de un elemento de formulario cambia (input, select, etc.).

```
document.getElementById('miInput').addEventListener('change', function() {
  console.log('Valor del input cambió');
});
```