EL DOM *(Modelo de objeto de documento)*

El DOM define un estándar para acceder a los documentos: "El Modelo de objetos de documento (DOM) del W3C es una plataforma y una interfaz de lenguaje neutral que permite que los programas y los scripts accedan y actualicen dinámicamente el contenido, la estructura y el estilo de un documento"

Cuando se carga una página web, el navegador crea el DOM (Document Model Object)

- **Modelo**: Significa que el DOM ofrece una representación estructurada de tu documento HTML.
- **Objeto**: Cada elemento del DOM es un objeto al que puedes acceder y manipular con JavaScript.
- **Documento**: Se refiere a la página HTML.

El modelo HTML DOM se construye como un árbol de objetos. Con este, JavaScript obtiene toda la potencia que necesita para crear HTML dinámico:

- JavaScript puede cambiar todos los *elementos* HTML en la página
- JavaScript puede cambiar todos los *atributos* HTML en la página
- JavaScript puede cambiar todos los estilos *CSS* en la página
- JavaScript puede *eliminar* elementos y atributos HTML existentes
- JavaScript puede *agregar* nuevos elementos y atributos HTML
- JavaScript puede *reaccionar* a todos los eventos HTML existentes en la página
- JavaScript puede *crear* nuevos eventos HTML en la página

¿Qué es el HTML DOM?

El HTML DOM es un modelo de objetos estándar y una interfaz de programación para HTML que define cómo obtener, modificar, agregar o eliminar elementos HTML.

La interfaz de programación DOM

Se puede acceder al HTML DOM con JavaScript. En el DOM, todos los elementos HTML se definen como objetos. La interfaz de programación son las propiedades y métodos de cada objeto.

- Una propiedad es un valor que puede obtener o establecer (como cambiar el contenido de un elemento HTML).
- Un método es una acción que puede hacer (como agregar o eliminar un elemento HTML).

Por ejemplo:

- `getElementById` es un método que accede a un elemento por el ID.
- `innerHTML` es una propiedad para obtener o remplazar el contenido de elementos HTML.

```
<html>
        <body>
                <p id="demo"></p>

                <script>
                        document.getElementById("demo").innerHTML = "Hello
World!";
                </script>
        </body>
</html>
```

El objeto de Documento DOM HTML

El DOM (Document Object Model) representa la estructura de la página web. Para acceder y manipular elementos HTML, usamos el objeto `document`. Aquí hay algunos ejemplos:

```
//
**********************************************************************
**
// Encontrar elementos HTML

document.getElementById("id");              // Encuentra un elemento por su ID
document.getElementsByTagName("name");      // Encuentra elementos por etiqueta
HTML
document.getElementsByClassName("name");    // Encuentra elementos por clase
CSS
document.querySelector("css");              // Selecciona el primer elemento
que coincida con el selector CSS
document.querySelectorAll("css");           // Selecciona todos los elementos
que coincidan con el selector CSS

//
**********************************************************************
**
// Cambiar elementos HTML

// Propiedades
element.innerHTML = "nuevo contenido";      // Cambia el contenido HTML de un
elemento
```

```
element.attribute = "nuevo valor";          // Cambia el valor de un atributo
de un elemento HTML
element.style.property = "nuevo estilo";    // Cambia el estilo de un elemento
HTML

// Métodos
element.setAttribute("attribute", "value"); // Cambia el valor de un atributo
de un elemento HTML
// Nota: `setAttribute` es un método que requiere el nombre y valor del
atributo como argumentos.

//
**********************************************************************
**
// Añadir y eliminar elementos

document.createElement("element");          // Crea un nuevo elemento HTML
document.removeChild(element);              // Elimina un elemento HTML
document.appendChild(element);              // Añade un elemento HTML como hijo
de otro
document.replaceChild(newElement, oldElement); // Reemplaza un elemento HTML
por otro
document.write("texto");                    // Escribe texto o HTML
directamente en la página

//
**********************************************************************
**
// Agregar controladores de eventos

document.getElementById("id").onclick = function() {
  // Código para gestionar el evento `onclick` del elemento con el ID
especificado
};

//
**********************************************************************
**
// Encontrar objetos HTML

// HTML DOM Nivel 1 (1998) define 11 objetos HTML que siguen válidos en HTML5
document.anchors;      // Anclas en el documento
document.body;         // Cuerpo del documento
document.cookie;       // Cookies del documento
document.doctype;      // Tipo de documento
document.forms;        // Formularios en el documento
```

```
document.images;       // Imágenes en el documento
document.links;        // Enlaces en el documento
document.readyState;   // Estado de carga del documento
```

Nodos DOM

De acuerdo con el estándar W3C HTML DOM, todo en un documento HTML es un nodo:

- Todo el documento es un nodo de documento.
- Cada elemento HTML es un nodo de elemento.
- El texto dentro de los elementos HTML son nodos de texto.
- Cada atributo HTML es un nodo de atributo (en desuso).
- Todos los comentarios son nodos de comentarios.

Los nodos son objetos que forman el DOM y como tal tienen propiedades y métodos. Las propiedades son atributos que permiten identificar el nodo y saber su relación con el resto del DOM. Vamos a ver las propiedades más importantes.

Propiedades de los Nodos DOM

```
// Propiedad `nodeName`: especifica el nombre de un nodo.
let elemento = document.getElementById("h1");
alert(elemento.nodeName); // Muestra "H1"

// - `nodeName` es de solo lectura.
// - Para un nodo de elemento, `nodeName` es el nombre de la etiqueta en
MAYÚSCULAS.
// - Para un nodo de atributo, `nodeName` es el nombre del atributo.
// - Para un nodo de texto, `nodeName` siempre es "#text".
// - Para el nodo de documento, `nodeName` siempre es "#document".

//
**********************************************************************
**
// Propiedad `nodeValue`: especifica el valor de un nodo.
console.log(elemento.nodeValue);          // `null` para nodos de elemento
console.log(elemento.firstChild.nodeValue); // Texto de un nodo de texto

// - Para elementos de nodo, `nodeValue` es `null`.
// - Para nodos de texto, `nodeValue` es el texto en sí.
// - Para nodos de atributo, `nodeValue` es el valor del atributo.

//
**********************************************************************
```

```
**
// Propiedad `nodeType`: devuelve el tipo de un nodo. Es de solo lectura.

console.log(elemento.nodeType); // Devuelve 1 para ELEMENT_NODE

// Tipos de nodos más comunes:
const nodeTypes = {
  ELEMENT_NODE: 1,       // <h1 class="heading">Titulo</h1>
  ATTRIBUTE_NODE: 2,     // class = "heading" (obsoleto)
  TEXT_NODE: 3,          // Texto de un elemento
  COMMENT_NODE: 8,       // <!-- This is a comment -->
  DOCUMENT_NODE: 9,      // El documento HTML mismo (padre de <html>)
  DOCUMENT_TYPE_NODE: 10 // <!Doctype html>
};
```

Propiedades de relación entre los nodos

```
// Propiedades para navegar entre nodos
let nodoEjemplo = document.getElementById("p1");

let nodoPadre = nodoEjemplo.parentNode;             // Nodo padre de un nodo
let nodoHijos = nodoEjemplo.childNodes;             // Colección de hijos de
un nodo
let primerHijo = nodoEjemplo.firstChild;            // Primer hijo de un nodo
let ultimoHijo = nodoEjemplo.lastChild;             // Último hijo de un nodo
let siguienteHermano = nodoEjemplo.nextSibling;     // Siguiente nodo hermano
let anteriorHermano = nodoEjemplo.previousSibling;  // Anterior nodo hermano

//
**********************************************************************
**
// Nodos secundarios y valores de nodo
// Un nodo de elemento no contiene texto directamente, sino un nodo de texto
con el valor.

<p id="p1">Tutorial DOM</p>;

let miP1 = document.getElementById("p1").innerHTML;           // "Tutorial
DOM"
let miP2 = document.getElementById("p1").firstChild.nodeValue; // "Tutorial
DOM"
let miP3 = document.getElementById("p1").childNodes[0].nodeValue; // "Tutorial
DOM"

// Ejemplo: copiar el texto de un <h1> a un <p> en tres formas
```

```
<h1 id="h1">Mi pagina</h1>
<p id="p1"></p>

<script>

document.getElementById("p1").innerHTML =
document.getElementById("h1").innerHTML;
document.getElementById("p1").innerHTML=document.getElementById("h1").firstChi
ld.nodeValue;

document.getElementById("p1").innerHTML=document.getElementById("h1").childNod
es[0].nodeValue;

</script>

// Nota: `innerHTML` es un método rápido para acceder al contenido HTML, pero
conocer otros métodos ayuda a entender la estructura de árbol y navegación del
DOM.

// Para nodos de texto, también se puede acceder al contenido mediante
`textContent`.
let textoNodo = document.getElementById("demo").textContent;
```

Encontrar elementos HTML por colecciones de objetos HTML

Como vimos anteriormente al lenguaje se le incorporaron objetos y colecciones para agrupar elementos concretos. Podemos realizar búsquedas sobre esos elementos también. Es decir:

```
let misImagenes= document.images // podemos acceder a todas las imagenes del
documento
let misLinks= document.links // podemos acceder a todos las links del
documento
let misForms=document.forms // podemos acceder a todos las formularios del
documento
```

Modificar los elementos del DOM

Una vez que hemos accedido a los elementos del DOM con cualquiera de las formas que hemos visto, podemos modificar las propiedades de los elementos modificando dinámicamente los elementos HTML de nuestra página. Las modificaciones son de dos tipos de atributos o de estilo CSS.

- Atributo: `elemento.atributo=valor`.

```
<img id="miImagen" src="imagen1.gif">

<script>
        document.getElementById("miImagen").src = "imagen2.jpg";
</script>
```

- Estilo: `elemento.style.property = estilo`.

```
<p id="p1">Hola mundo!</p>

<script>
        document.getElementById("p1").style.color = "blue";
</script>
```

Operaciones sobre nodos DOM

Hasta ahora hemos accedido a elementos que ya existían en el DOM y una vez que teníamos acceso hemos podido modificarle las propiedades. En muchas ocasiones lo que vamos a necesitar es crear, modificar, eliminar o clonar nodos.

Crear nuevos elementos HTML (nodos)

```
let imagen = document.createElement("img");  // Crear nodo <img>
imagen.id = "mi_foto";  // Asignar id
imagen.src = "./fotos/logo.jpg";  // Asignar src para definir la imagen

// Añadir imagen al contenedor 'fotos'
let divFotos = document.getElementById("fotos");
divFotos.appendChild(imagen);  // Insertar nodo en el DOM

// Ejemplo con nodo de texto
let miParrafo = document.createElement("p");  // Crear nodo <p>
let miTexto = document.createTextNode("Texto del parrafo.");  // Crear nodo de
texto
miParrafo.appendChild(miTexto);  // Agregar texto al <p>

// Añadir párrafo al contenedor 'div1'
let padre = document.getElementById("div1");
padre.appendChild(miParrafo);  // Insertar nodo en el DOM

// textContent para acceder o cambiar texto sin crear nodo de texto
```

```
const parrafo = document.getElementById("miParrafo");
console.log(parrafo.textContent);  // Obtener contenido de texto
parrafo.textContent = "Nuevo contenido para el párrafo.";  // Cambiar
contenido

// insertBefore para añadir en una posición específica
let miparrafo = document.createElement("p");
let miTexto2 = document.createTextNode("Parrafo nuevo");
miparrafo.appendChild(miTexto2);
let elementoHijo = document.getElementById("p1");  // Nodo de referencia
padre.insertBefore(miparrafo, elementoHijo);  // Insertar antes de
elementoHijo
```

Eliminar elementos HTML existentes

```
let parrafoEliminar = document.getElementById("p1");
parrafoEliminar.remove();  // Eliminar nodo directamente
```

Sustitución de elementos HTML

```
let nuevoParrafo = document.createElement("p");
let nodoTexto = document.createTextNode("Parrafo nuevo");
nuevoParrafo.appendChild(nodoTexto);
let child = document.getElementById("p1");  // Nodo a reemplazar
padre.replaceChild(nuevoParrafo, child);  // Reemplazar nodo
```

Clonar elementos HTML

```
// Método cloneNode(deep): deep = true copia el nodo y todos los descendientes
let clon = document.getElementById("miParrafo").cloneNode(true);
```

Asignar eventos utilizando el HTML DOM

Los eventos permiten ejecutar funciones en respuesta a acciones del usuario (clic, cambio, tecla pulsada, etc.).

```
// Asignación de evento onclick a un botón existente para mostrar la hora
actual
document.getElementById("btn1").onclick = displayDate;

function displayDate() {
    document.getElementById("p1").innerHTML = Date();  // Muestra la fecha y
```

```
hora actuales en el párrafo
}

// Crear un nuevo botón y asignarle un evento onclick usando una función
anónima
let miBoton = document.createElement("button");        // Crear nodo <button>
miBoton.innerHTML = "Púlsame...";                      // Asignar texto al
botón

miBoton.onclick = function() {                         // Asignar función
anónima al evento onclick
    alert(this.innerHTML);                             // Muestra el texto del
botón en un alert
};

document.body.appendChild(miBoton);                    // Añadir el botón al
body
```