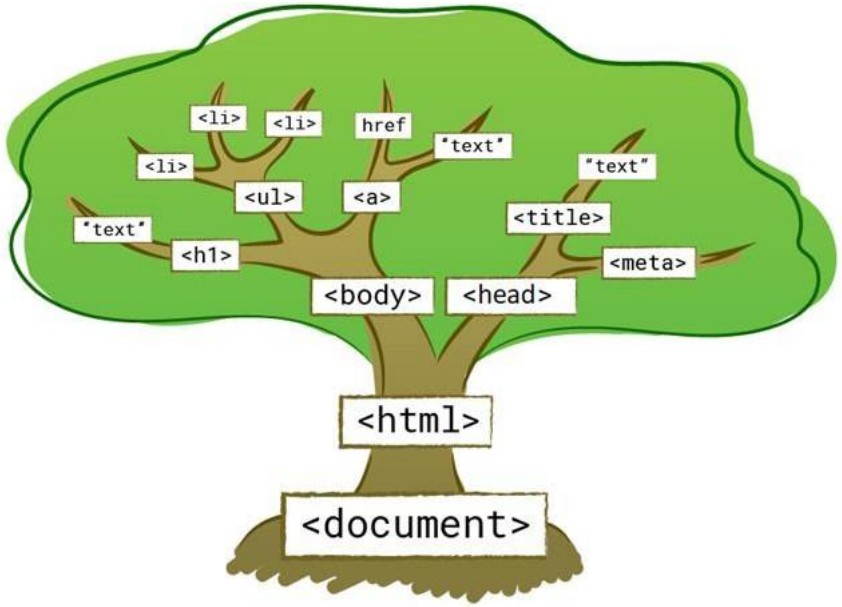
D.O.M.

Walter Arias Aguirre

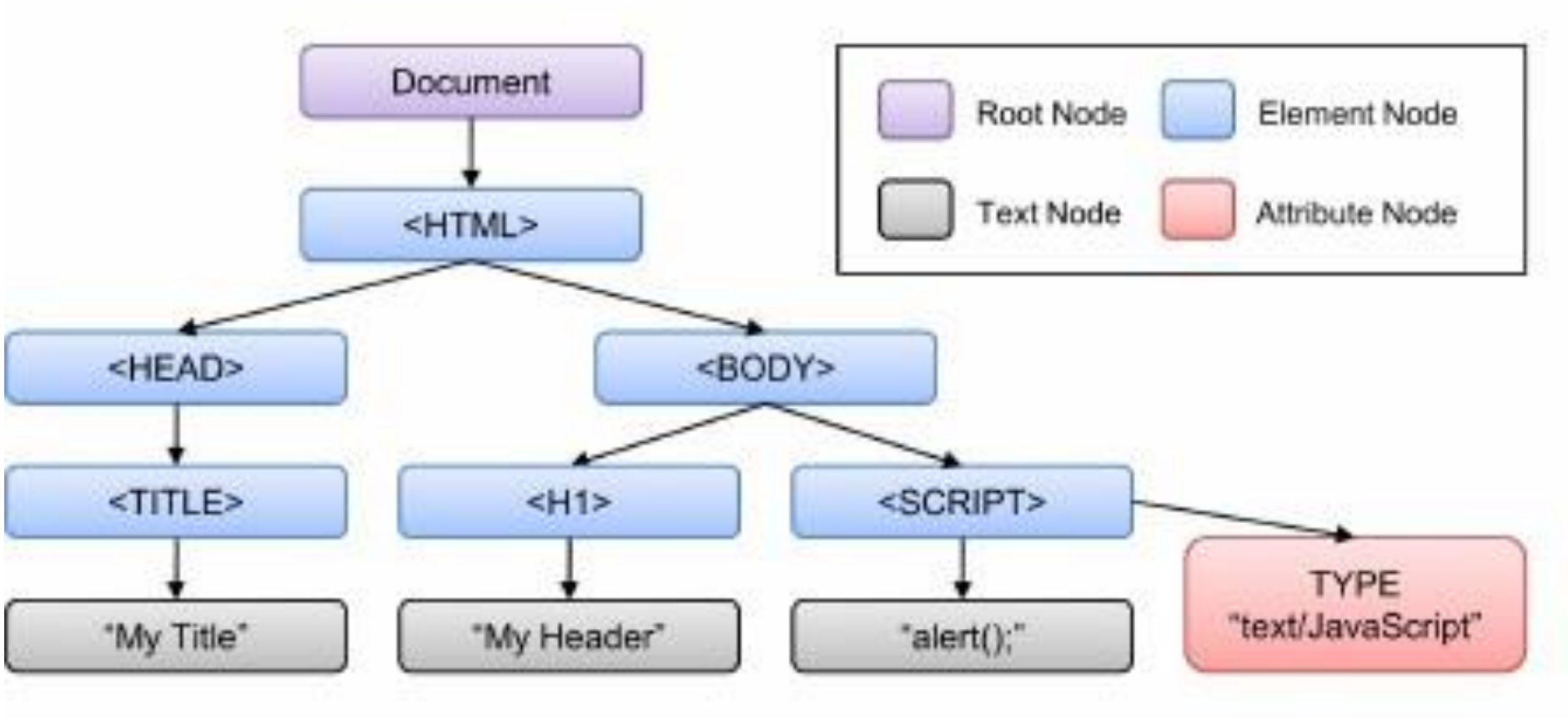
Definición

El Modelo de Objetos del Documento (D.O.M) es una interfaz (API) de programación para documentos HTML y XML. Representa la página para que los programas puedan cambiar la estructura, el estilo y el contenido del documento. El DOM representa el documento como nodos y objetos.

Dentro de una página web todos los objetos (body, h1, forms, a href, etc.) se representan mediante una jerarquía; con JavaScript se puede de forma dinámica crear, modificar y eliminar dichos elementos.



Nodos o Elements del DOM



DOCUMENT

Accediendo a los elementos(element)

SERVICIO NACIONAL DE APRENDIZAJE

- JavaScript representa los nodos del documento mediante el tipo **document**. En los navegadores, **el objeto document** es una instancia de HTMLDocument (que hereda de Document) y representa toda la página HTML.
- El objeto documento es una propiedad de **window** y por lo tanto es accesible globalmente.

Propiedades del Objeto document

Propiedad	Descripción
lastModified	La fecha de la última modificación de la página
referrer	La URL desde la que se accedió a la página (es decir, la página anterior en el array history)
title	El texto de la etiqueta <title>
URL	La URL de la página actual del navegador

Otras Propiedades

SERVICIO NACIONAL DE APRENDIZAJE

Array	Descripción
anchors	Contiene todas las "anclas" de la página (los enlaces de tipo)
applets	Contiene todos los applets de la página
embeds	Contiene todos los objetos embebidos en la página mediante la etiqueta <embed>
forms	Contiene todos los formularios de la página
images	Contiene todas las imágenes de la página
links	Contiene todos los enlaces de la página (los elementos de tipo)

Métodos del document

- **write (texto), writeln (texto) :** se utiliza para escribir texto HTML en el documento. El método writeln añade un salto de línea al final del texto que se haya pasado como argumento, pero si la ejecución se realiza entre las etiquetas <pre></pre> de HTML.

- **METODOS PARA ACCEDER A LOS NODOS Y ELEMENTOS**

- **getElementById (identificador):** Va a devolver el elemento del documento cuyo atributo ID coincida con el parámetro identificador. Si no existe el identificador devuelve nulo

- **getElementsByName (nombre):** Retorna una colección de elementos cuyo atributo NAME coincida con el indicado.

- **getElementsByTagName (Etiqueta):** Devuelve la lista de elementos cuya etiqueta sea la misma que la especificada por el parámetro. Si lo que se quiere es obtener la lista de todos los elementos hay que usar el valor especial asterisco, «*».

- **getElementsByClassName (clase):** Localiza todos los elementos que tengan la clase indicada (atributo CLASS). Es posible indicar más de una clase, separadas por un espacio y sin importar el orden, para encontrar los elementoso que contengas todas ellas.

Métodos del document

- **querySelector (Selector):** Nos permite encontrar el primer elemento que cumpla el selector CSS. por ejemplo, las clases CSS deben indicarse con un punto delante del nombre (.nombreClase) y los identificadores con una almohadilla (#nombreId). Si no hay coincidencias, devuelve nulo.
- **querySelectorAll (selector):** Es idéntico al anterior, pero nos devuelve todos los elementos que coinciden con el selector.
- **createElement (etiqueta):** Va a permitir crear nuevos elementos dinámicamente por código. Tendremos que indicar el nombre de la etiqueta que corresponde al elemento que queremos crear: párrafo (p), capa (div), etc.
- **removeElement (etiqueta):** Elimina elementos dinámicamente Tendremos que indicar el nombre de la etiqueta que corresponde al elemento que queremos eliminar: párrafo (p), capa (div), etc.

ELEMENT

El Tipo ELEMENT

SERVICIO NACIONAL DE APRENDIZAJE

- Junto al tipo Document, el tipo Element es el más utilizado en la programación web. **El tipo Element representa un elemento XML o HTML**, proporcionando acceso a información como su nombre de etiqueta hijos y atributos.

- Todos los elementos HTML están representados por el tipo HTMLElement, ya sea directamente o a través de **subtipos**.

- Cada element tiene los siguientes atributos estándar que están disponibles en cada elemento HTML:

  - ***id-****Un identificador único para el elemento en el documento.*
  - ***title****-Información adicional sobre el elemento, normalmente representada como un tooltip.*
  - ***lang****-Código de idioma para el contenido del elemento (raramente utilizado).*
  - ***dir-****La dirección del idioma, "ltr" (de izquierda a derecha) o "rtl" (de derecha a izquierda); también rara vez se utiliza.*
  - ***className-****El equivalente del atributo class, que se utiliza para especificar las clases CSS en un elemento.*

EJEMPLO

- **MOSTRANDO ATRIBUTOS PARA LOS ELEMENT**

```
let div = document.getElementById("myDiv");
alert(div.getAttribute("id")); // "miDiv"
alert(div.getAttribute("class")); // "bd"
alert(div.getAttribute("title")); // "Cuerpo del texto".
alert(div.getAttribute("lang")); // "en"
alert(div.getAttribute("dir")); // "ltr"
```

Propiedades de los element

SERVICIO NACIONAL DE APRENDIZAJE

- **Element.attributes** Sólo lectura : Devuelve un objeto NamedNodeMap que contiene los atributos asignados del elemento HTML correspondiente.
- **Element.childElementCount** Sólo lectura : Devuelve el número de elementos hijos de este elemento.
- **Element.children** Sólo lectura : Devuelve los elementos hijos de este elemento.
- **Element.clientHeight** Sólo lectura: Devuelve un número que representa la altura interior del elemento.
- **Element.clientLeft** Sólo lectura: Devuelve un número que representa la anchura del borde izquierdo del elemento.

Propiedades de los element

- **Element.clientTop** Sólo lectura :Devuelve un número que representa la anchura del borde superior del elemento.
- **Element.clientWidth** Sólo lectura :Devuelve un número que representa la anchura interior del elemento.
- **Element.firstElementChild** Sólo lectura :Devuelve el primer elemento hijo de este elemento.
- **Elemento.id:**Es un DOMString que representa el id del elemento.
- **Element.innerHTML :**Es un DOMString que representa el contenido del elemento.
- **Element.lastElementChild** Sólo lectura :Devuelve el último elemento hijo de este elemento.

Manipulando atributos para los element

- Cada elemento puede tener cero o más atributos, que suelen utilizarse para dar información adicional sobre el elemento en particular o su contenido. Los tres métodos principales del DOM para trabajar son:

  - **getAttribute(),**
  - **setAttribute()**
  - **removeAttribute().**

- Estos métodos están pensados para trabajar con cualquier atributo, incluyendo los definidos como propiedades en el tipo HTMLElement. He aquí un ejemplo:

Creando Element

SERVICIO NACIONAL DE APRENDIZAJE

- Se pueden crear nuevos elementos utilizando el método document.createElement(). Este método acepta un solo argumento, que es el nombre de la etiqueta del elemento a crear. En los documentos HTML, el nombre de la etiqueta no distingue entre mayúsculas y minúsculas, mientras que sí lo hace en los documentos XML (incluido XHTML). Para crear un elemento , se puede usar el siguiente código:

```
let div= documento.createElement("div");
```

- El uso del método createElement() crea un nuevo elemento y establece su propiedad. En este punto, puede manipular los atributos del elemento, agregarle más elementos secundarios, etc. Considera el siguiente ejemplo:

```
div.id  =  "miNuevoDiv";

div.className = "caja";
```

Creando Element

- Establecer estos atributos en el nuevo elemento ***asigna solo información***. Porque el elemento no es parte del árbol del documento, no afecta la visualización del navegador. El elemento se puede agregar al documento. árbol utilizando **append(), appendChild(), insertBefore() o replaceChild().** El siguiente código agrega el elemento recién creado al elemento <body> del documento:

```
document.body.appendChild(div)
```

Borrando element

- Para ello usamos el método element.remove()

Objeto NODELIST

DEFINICIÓN

- Son colecciones (parecidos a los arrays) de nodos.
- Se pueden obtener con el selector querySelectorAll().
- A pesar de no ser arreglos se pueden iterar o recorrer con el bucle foreach o for.

Propiedades interesantes de NodeList

SERVICIO NACIONAL DE APRENDIZAJE

- Length = retorna el número de nodos
- childNodes = devuelve los hijos del nodo en forma de lista o colección.

Métodos

- Nodelist.item() = devuelve un elemento de acuerdo al índice de la colección.
- Nodelist.entries() = devuelve todas las entradas de a
- Nodelist.keys() = devuelve todas las llaves o keys de la colección.
- Nodelist.values() = devuelve todos los valores de la colección
- Nota: todos estos métodos se pueden recorrer usando foreach, o el método for.

Ejemplo de for

- For (const key of datos.entries()){
  console.log(key)
- }

STYLES

Manipulando Estilos

- Los estilos CSS se pueden manejar dinámicamente con el comando style ejemplo:

```
milista.style.backgroundColor = "blue"
milista.style.color ="white"
milista.style.padding ="20px"
```

Uso más práctico

- Usando **setAttribute** del element

ejemplo:

```
input2.setAttribute(
"style",
" background-color: red;  border: solid 5px black; padding: 10px;"
);
```

Clases CSS Dinámicas con DOM - atributos

.classList Devuelve la lista de clases del elemento HTML.

.classList.length Devuelve el número de clases del elemento HTML.

.classList.item(n) Devuelve la clase número n del elemento HTML. si no existe.

.classList.contains(clase) Indica si la clase existe en el elemento HTML.

Métodos de classlist

- .classList.add(c1, c2, ...) Añade las clases c1, c2... al elemento HTML.
- .classList.remove(c1, c2, ...) Elimina las clases c1, c2... del elemento HTML.
- .classList.toggle(clase) Si la clase no existe, la añade. Si no, la elimina.
- .classList.toggle(clase, expr) Si expr es true, añade la clase. Si es false, la elimina.
- .classList.replace(old, new) Reemplaza la clase old por la clase new.

RECORRIENDO EL DOM

TRAVERSAL DOM

CONCEPTO CLAVE

- ELEMENTO O ELEMENT: Cualquier etiqueta HTML (es a la vez un nodo)
- NODO: Puede ser un elemento(etiqueta html) o texto, atributo, propiedad, comentario.

Nodos /elementos

- Element.parentNode
- Element.parentElement

- Aumentar la jerarquía de acuerdo a la profundidad del arbol

Nodos /elementos Hijos

- Element.children
- Element children[index]
- Element.firstChild
- Element.firstelementChild
- Element.lastChild
- Element.lastElementChild

Nodos/elementos Hermanos

- Element.previouselementsibling
- Element.nextelementsibling

Element Table

Atributos del Element Table

SERVICIO NACIONAL DE APRENDIZAJE

- El element <table> añade lo siguiente:
  - **caption-**Indicador del elemento <caption> (si existe).
  - **tBodies-**Una colección HTMLC de elementos <tbody>.
  - **tFoot-**Indicador del elemento <tfoot> (si existe).
  - **tHead-**Pointer al elemento <thead> (si existe).
  - **rows-**Una colección HTML de todas las filas de la tabla.

Métodos del element table

- **createTHead()-**Crea un elemento <thead>, lo coloca en la tabla y devuelve una referencia.
- **createTFoot()-**Crea un elemento <tfoot>, lo coloca en la tabla y devuelve una referencia.
- **createCaption()-**Crea un elemento <caption>, lo coloca en la tabla y devuelve una referencia.
- **deleteTHead()-**Elimina el elemento <thead>.
- **deleteTFoot()-**Elimina el elemento <tfoot>.
- **deleteCaption()-**Elimina el elemento <caption>.
- **deleteRow(pos)-**Elimina la fila en la posición dada.
- **insertRow(pos)-**Inserta una fila en la posición dada en la colección de filas.

Element tbody

- **rows:** una colección o arreglo *HTMLCollection* de filas en el elemento <tbody>.
- **deleteRow(pos):** Elimina la fila en la posición dada.
- **insertRow(pos):** Inserta una fila en la posición dada en la colección de filas y devuelve una referencia a la nueva fila.

Element TR

- **cells-**Una colección *HTMLCollection* de celdas en el elemento <tr>.
- **deleteCell(pos)-**Elimina la celda en la posición dada.
- **insertCell(pos)-**Inserta una celda en la posición dada en la colección de celdas y devuelve una referencia a la nueva celda.

NODOS DE TEXTO

Nodos de Texto

- Los nodos de texto están representados por el tipo Texto y contienen texto sin formato que se interpreta literalmente y puede contener caracteres HTML escapados, pero no código HTML

- **CREACIÓN DE NODOS DE TEXTO**

- Se pueden crear nuevos nodos de texto utilizando el método document.createTextNode(), que acepta un solo argumento: el texto que se insertará en el nodo.

- Al igual que con la configuración del valor de un nodo de texto existente, el texto estará codificado en HTML o XML, como se muestra en este ejemplo:

```
let texto = document.createTextNode("<b>Hola</b>”)
```

Métodos Importantes

- **appendData(texto):** agrega texto al final del nodo.
- **deleteData(offset, count):** elimina el número de caracteres que comienzan en la posición de OFFSET.
- **insertData(OFFSET, texto):** inserta texto en la posición de OFFSET
- **replaceData(offset, conteo):** reemplaza el texto que comienza en offset hasta el conteo final de offset
- **splitText(offset):** divide el nodo de texto en dos nodos de texto separados en la posición de offset.
- **substringData(offset, conteo):** extrae una cadena del hasta que comienza en offset hasta el conteo final de offset.

- Además de estos métodos, *la propiedad* ***length*** devuelve la cantidad de caracteres en el nodo.

LIBROS

- **Profesional Javascript for web developers.** Matt Frisbie
- **Eloquent Javascript.** Marijn Haverbeke.
- **Mastering Javascript.** Ved Antani

REFERENCIA TOTAL DEL DOM

SERVICIO NACIONAL DE APRENDIZAJE

- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

APIS DISPONIBLES PARA JAVASCRIPT

https://developer.mozilla.org/es/docs/Web/API