

JS

UT6. MANEJO DEL DOM (DOCUMENT OBJECT MODEL)

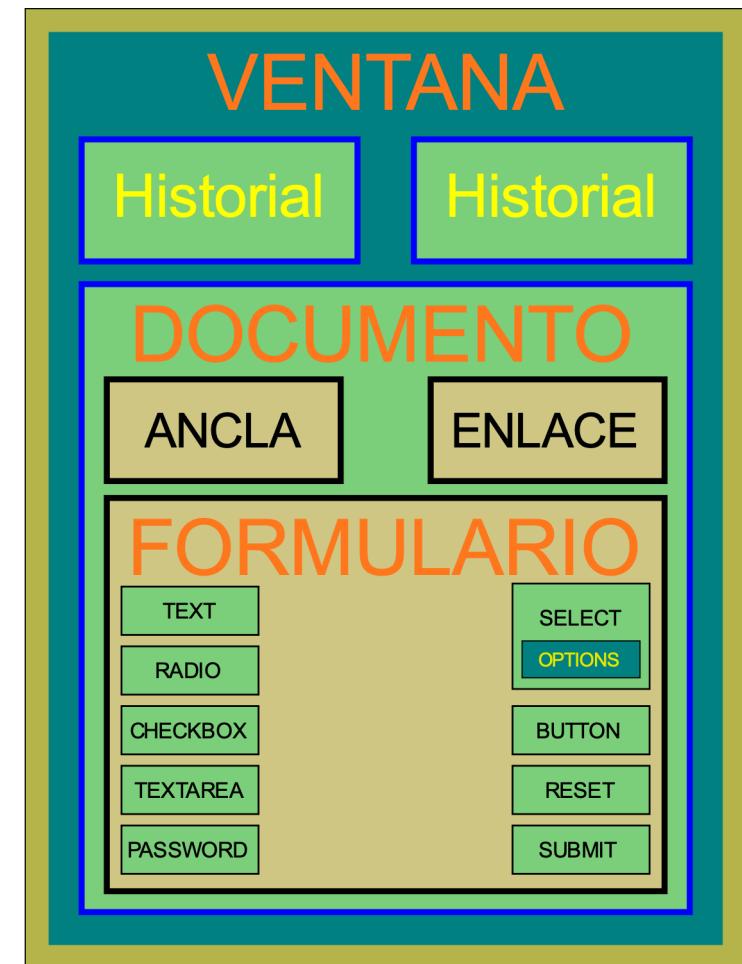
Desarrollo Web en Entorno
Cliente
2ºDAW

1. EL MODELO DE OBJETOS DEL DOCUMENTO (DOM)

Es un **estándar de W3C** que define cómo acceder a los documentos, como por ejemplo HTML y XML.

Es una **interfaz de programación de aplicaciones (API)** de la plataforma de W3C.

Permite a los scripts acceder y actualizar dinámicamente su contenido, estructura y estilo de documento.



1. EL MODELO DE OBJETOS DEL DOCUMENTO (DOM)

Fue utilizado por primera vez con el navegador Netscape Navigator V.2.0.

A esta primera versión de DOM se le denomina **DOM nivel 0**.

El primer navegador de Microsoft que utilizó el DOM nivel 0 fue IE 3.0.

Debido a las diferencias entre los navegadores, W3C emitió una especificación a finales de **1998** que llamó **DOM nivel 1**.

En esta especificación ya se consideraba la manipulación de todos los elementos existentes en los archivos HTML.

1. EL MODELO DE OBJETOS DEL DOCUMENTO (DOM)

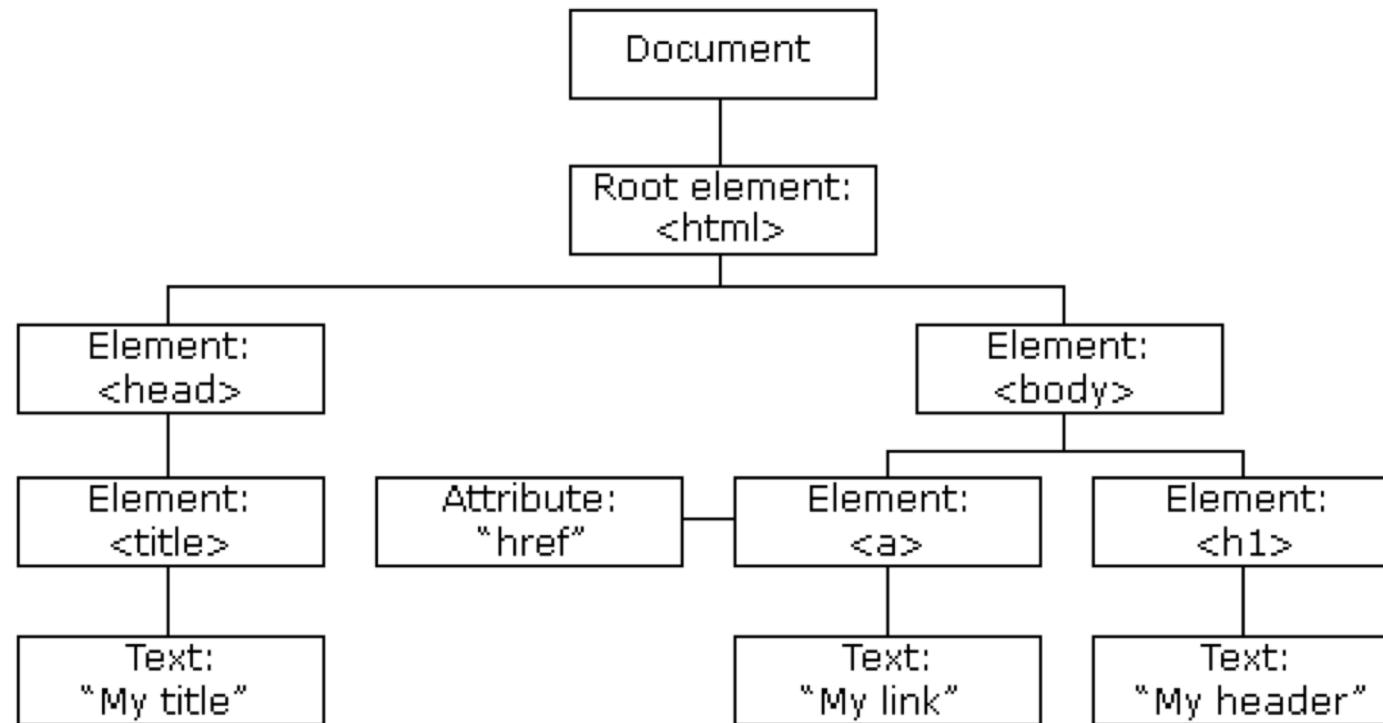
A finales del año **2000**, W3C emitió **DOM nivel 2**, en la cual se incluía el manejo de eventos en el navegador y la interacción con hojas de estilo CSS.

En **2004** se emitió **DOM nivel 3**, en la cual se utiliza la definición de tipos de documento (DTD) y la validación de documentos.

Actualmente DOM se divide en tres partes según la W3C:

- Núcleo del DOM.
- XML DOM.
- HTML DOM.

1.2 ESTRUCTURA DEL ÁRBOL DOM



1.2 ESTRUCTURA DEL ÁRBOL DOM

Para ordenar la estructura del árbol, existe una serie de reglas:

- En el árbol de nodos, al nodo superior (document), se le llama **raíz**.
- **Cada nodo**, exceptuando el nodo raíz, **tiene un parente**.
- Un nodo puede tener **cualquier número de hijos**.
- Una **hoja** es un **nodo sin hijos**.
- Los nodos que comparten el mismo parente, son **hermanos**.

2.1 OBJETOS DEL MODELO.

Objetos del modelo:

- **Node.** Es una interfaz abstracta para indicar un nodo del árbol. Esos nodos pueden ser un Document, un Element, un DocumentFragment...
 - <https://developer.mozilla.org/en-US/docs/Web/API/Node>
- **Document.** Es el nodo raíz del documento HTML.
- **DocumentType.** Es un nodo que indica la representación del DTD de la página.
- **Element:** Es un nodo que representa un elemento HTML.
- **Attr:** Este nodo representa un atributo de un elemento.
- **Text:** Este nodo almacena el texto contenido dentro de un nodo Element.

2.1 OBJETOS DEL MODELO.

La interfaz **Node**:

- Para poder manipular la información de los nodos, JavaScript crea un objeto denominado Node.
- En este objeto se definen las propiedades y los métodos para procesar los documentos.
- Este objeto define una serie de constantes que identifican los tipos de nodo.

<https://developer.mozilla.org/en-US/docs/Web/API/Node>

2.1 OBJETOS DEL MODELO.

Constantes de Node:

Tipo de nodo=Valor	
Node.ELEMENT_NODE = 1	Node.PROCESSING_INSTRUCTION_NODE = 7
Node.ATTRIBUTE_NODE = 2	Node.COMMENT_NODE = 8
Node.TEXT_NODE = 3	Node.DOCUMENT_NODE = 9
Node.CDATA_SECTION_NODE = 4	Node.DOCUMENT_TYPE_NODE = 10
Node ENTITY_REFERENCE_NODE = 5	Node.DOCUMENT_FRAGMENT_NODE = 11
Node ENTITY_NODE = 6	Node.NOTATION_NODE = 12

2.1 OBJETOS DEL MODELO.

Algunas propiedades y métodos de Node:

Propiedad / Método	
nodeName	previousSibling
nodeValue	nextSibling
nodeType	hasChildNodes ()
ownerDocument	attributes
firstChild	appendChild (nodo)
lastChild	removeChild (nodo)
childNodes	replaceChild (nuevoNodo, anteriorNodo)
parentNode	insertBefore (nuevoNodo, anteriorNodo)

3. ACCESO AL DOCUMENTO DESDE CÓDIGO

Una vez cargados por el navegador todos los elementos en el árbol del DOM (Evento **DOMContentLoaded**) ya podemos acceder a cualquier nodo.

Los nodos hijos de cada nodo se almacenarían en un array (**nodo.childNodes**)

3. ACCESO AL DOM CON CÓDIGO

Podemos **movernos por el árbol del DOM** desde un nodo dado con estas propiedades:

nodo.**firstChild** → primer nodo hijo de un nodo

nodo.**lastChild** → último nodo hijo de un nodo

nodo.**childNodes[2]** → nodo hijo en tercera posición de un nodo

nodo.**childNodes.length** → número de hijos de un nodo.

nodo.**parentNode** → Nodo padre de un nodo

nodo.**previousSibling** → anterior hermano de un nodo.

nodo.**nextSibling** → siguiente hermano de un nodo.

3. ACCESO AL DOM CON CÓDIGO

Podemos acceder al **tipo de nodo** con la propiedad **nodeType**:

```
obj_tipo_document = document.nodeType; // 9
```

```
obj_tipo_element = document.documentElement.nodeType; // 1
```

Tipo de nodo=Valor	
Node.ELEMENT_NODE = 1	Node.PROCESSING_INSTRUCTION_NODE = 7
Node.ATTRIBUTE_NODE = 2	Node.COMMENT_NODE = 8
Node.TEXT_NODE = 3	Node.DOCUMENT_NODE = 9
Node.CDATA_SECTION_NODE = 4	Node.DOCUMENT_TYPE_NODE = 10
Node.ENTITY_REFERENCE_NODE = 5	Node.DOCUMENT_FRAGMENT_NODE = 11
Node.ENTITY_NODE = 6	Node.NOTATION_NODE = 12

3.1 ACCESO DIRECTO A LOS NODOS

DOM añade una serie de métodos para acceder de manera más directa a los nodos de tipo Element:

- `getElementsByTagName(cadena)`
- `getElementById(cadena)`
- `getElementsByName(cadena)`
- `getElementsByClassName(cadena)`
- `querySelector(cadena)`
- `querySelectorAll(cadena)`

3.2 ACCESO A LOS ATRIBUTOS DE UN NODO TIPO ELEMENT

DOM permite acceder directamente a todos los atributos de una etiqueta.

La propiedad **attributes** permite acceder a los atributos de un nodo de tipo element mediante los siguientes métodos:

- **getNamedItem(nomAttr)**. Devuelve el nodo de tipo attr(atributo), cuya propiedad nodeName(Nombre del nodo) contenga el valor nomAttr.
- **removeNamedItem(nomAttr)**. Elimina el nodo de tipo attr(atributo) en el que la propiedad nodeName (Nombre del nodo) coincide con el valor nomAttr.
- **setNamedItem(nodo)**.Este método añade el nodo attr(atributo) a la lista de atributos del nodo element. Lo indexa según la propiedad nodeName(del atributo).
- **item(pos)**. Devuelve el nodo correspondiente a la posición indicada por el valor numérico pos.

3.2 ACCESO A LOS ATRIBUTOS DE UN NODO TIPO ELEMENT

DOM proporciona algunos métodos que facilita el acceso directo a la modificación, inserción y borrado de los atributos de una etiqueta:

- **getAttribute(nomAtributo)**. Este método equivale a:
attributes.getNamedItem(nombAtributo).
- **setAttribute(nomAtributo, valorAtributo)**. Este método equivale a la estructura: attributes.getNamedItem(nomAtributo).value= valor
- **removeAttribute(nomAtributo)**. Este método equivale a la estructura:
attributes.removeNamedItem(nomAtributo)

3.2 ACCESO A LOS ATRIBUTOS DE UN NODO TIPO ELEMENT

También podemos acceder y modificar el valor de un atributo también directamente con . seguido del nombre del atributo

```
const valor = elemento.value;  
elemento.value = "nuevo valor";  
elemento.checked = 1;
```

3.3 LA PROPIEDAD STYLE DE UN ELEMENT

Podemos modificar directamente las propiedades CSS de un elemento accediendo a su atributo `style` directamente:

```
elemento.style.color = "blue";  
elemento.style.fontFamily = "Arial";  
elemento.style.display = "block";
```

3.4 LA PROPIEDAD CLASSLIST DE UN ELEMENT

El atributo classList de un nodo tipo Element nos permite gestionar el listado de clases a los que pertenece el elemento HTML. El atributo es de solo lectura pero tiene varios métodos que facilitan su consulta y modificación:

- **add(String [, String])**Añade las clases indicadas. Si estas clases existieran en el atributo del elemento serán ignoradas.
- **remove(String [, String])**Elimina las clases indicadas.
Nota: Eliminar una clase que no existe NO produce un error.
- **item(Number)**Devuelve el valor de la clase por índice en la colección.
- **toggle(String [, force])**Cuando sólo hay un argumento presente: Alterna el valor de la clase; ej., si la clase existe la elimina y devuelve false, si no, la añade y devuelve true.
Cuando el segundo argumento está presente: Si el segundo argumento se evalúa como true, se añade la clase indicada, y si se evalúa como false, la elimina.
- **contains(String)**Comprueba si la clase indicada existe en el atributo de clase del elemento.
- **replace(oldClass, newClass)**Reemplaza una clase existente por una nueva.

<https://developer.mozilla.org/es/docs/Web/API/Element/classList>

3.5 CREACIÓN DE NUEVOS NODOS

- **createElement(etiqueta)** → crea un nuevo nodo del tipo pasado el parámetro etiqueta.
- **createTextNode(texto)** → crea un nodo de texto con el texto pasado en el parámetro
- **createAttribute(nombreAtributo)** → crea un nodo tipo atributo con el nombre dado.
- **createCDATASection(texto)** → crea una nodo tipo CDATA. con el texto pasado (El navegador lo interpretará como caracteres, no como marcas o referencias de entidad).
- **createComment(texto)** → se creará un nodo tipo comentario.
- **createDocumentFragment()** → se creara un nodo tipo DocumentFragment. (Un documento al que podemos anexar nodos y construir un árbol DOM fuera de pantalla).

3.6 ANEXAR NODOS A OTRO NODO

Para anexar un nodo a otro nodo existente tenemos dos métodos

elemento.append(child) → Este método acepta como hijo otro nodo pero también aceptará una cadena DOMString (una cadena de texto plano y HTML). Este método aceptaría múltiples nodos hijo.

elemento.appendChild(child) → Este método sólo acepta como hijo otro elemento tipo DOM. Este método solo acepta un parámetro hijo.

https://dev.to/ibn_abubakre/append-vs-appendchild-a4m