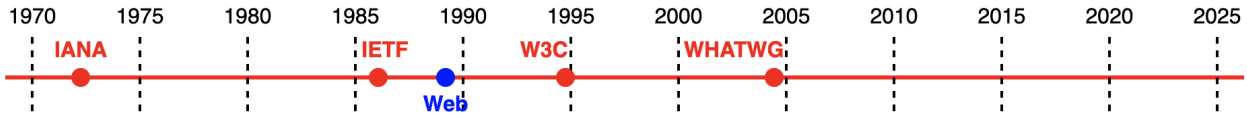
Actividad opcional 1. El modelo DOM

Historia, orígenes y representación del modelo DOM

El DOM o *Document Object Model*, cuya estandarización principal corre a cargo del W3C, es una interfaz de plataforma o API de programación para documentos HTML y XML que proporciona una representación estructurada del documento, facilita un conjunto estándar de objetos para representar dichos documentos y define, además, de qué manera los programas pueden acceder a ellos y manipularlos.

Pero, ¿cómo y cuándo surgen el DOM y el W3C? Todo empezó en 1989 en el CERN, el laboratorio europeo de física de partículas próximo a Ginebra – Suiza, cuando Tim Berners-Lee, el principal creador de la *World Wide Web*, se propuso crear un sistema de hipertexto para facilitar el intercambio de información entre los investigadores. Tras su éxito, en el verano de 1991, este sistema se puso a disposicion de todos los usuarios de Internet y en el 1993 la licencia del navegador y del servidor pasaron a ser de dominio público. Así, en 1994 se crea el W3C (*World Wide Web Consortium*) que es un organismo formado por empresas y universidades de todo el mundo que se encarga de desarrollar por consenso especificaciones relativas a la web que reciben el nombre de recomendaciones.



El DOM, sin embargo, nació de la mano de *Netscape Navigator* de *Netscape Communications*, pensado para detectar eventos generados por el usuario y modificar el documento HTML. A esta primera generación del DOM se le conoce como DOM *Level 0* o *Legacy* DOM. *Internet Explorer* de *Microsof*t le siguió y ambos navegadores incluían soporte para HTML dinámico (DHTML) permitiendo a los desarrolladores crear páginas con interactividad del lado del cliente, al manipular el documento a través del DOM. Pero había un problema y es que un mismo documento no estaba representado de la misma manera por estos navegadores.

Otro conflicto entre navegadores vino dado por el uso de intérpretes o motores para JavaScript distintos. Diferentes fabricantes de navegadores inventaron variantes de modelos de HTML dinámico y a medida que todo esto evolucionaba, los desarrolladores se veían obligados a escribir diferentes versiones según para qué navegador.

Las aplicaciones web de entonces se veían afectadas por códigos y soluciones temporales y funcionalidades restingidas además de los continuos cambios. Era necesario verificar todo el tiempo cuál era el navegador y la versión utilizada para que DOM pudiera funcionar. Algunas webs incluso utilizaban un *banner* en el pie de la página, informando qué navegador y resolución era la más apropiada para visualizar el sitio.

Los primeros estándares DOM del W3C surgieron en un clima de tensión conocido como la «guerra de los navegadores» en un intento de fusionar, estandarizar y finalmente reemplazar las diversas técnicas patentadas por cada navegador en favor del desarrollador y de los usuarios de

Internet. Podemos definir así, las versiones del DOM según se han ido incorporando los cambios más importantes del siguiente modo:

Versiones del DOM

- ***DOM Level 0***: Permitía la validación de eventos generados por el usuario y ligeras modificaciones en el HTML.
- ***DOM Level 1***: En 1998 se lanzó la primera recomendación W3C sobre DOM, que consistió en dividirlo en dos secciones principales: Core y HTML. Se considera la primera versión de DOM estándar. Esta recomendación trataba sobre documentos HTML y XML.
- ***DOM Level 2***: Dos años después, se alcanzó un nuevo estado de recomendación W3C, que empezaba a complicarse. La especificación se dividió en seis secciones: *Core*, HTML, *Views*, *Style Events* y *Traversal-Range*. Se añadieron nuevos aspectos como el tratamiento de espacios de nombres XML o de eventos.
- ***DOM Level 3***: De nuevo se dividió esta recomendación en seis secciones, aunque diferentes a las anteriores: *Core*, *Events*, *Load and Save*, *Validation*, *Xpath* y *Views*. Se agregaron además, el tratamiento de *XML Information Set* o *XML Base*.
- ***DOM Level 4***: Vigente desde 2014, es la versión del DOM que conocemos actualmente. A partir de mayo del 2019, el W3C ha dejado el desarrollo del DOM oficialmente en manos del WHATWG.

A pesar de sus orígenes, una historia llena de crisis y de conflictos enquistados, el DOM se ha convertido en una utilidad disponible para la mayoría de lenguajes de programación (*Java*, *Python, PHP, JavaScript*), cuyas diferencias se encuentran básicamente en la forma de implementarlo.

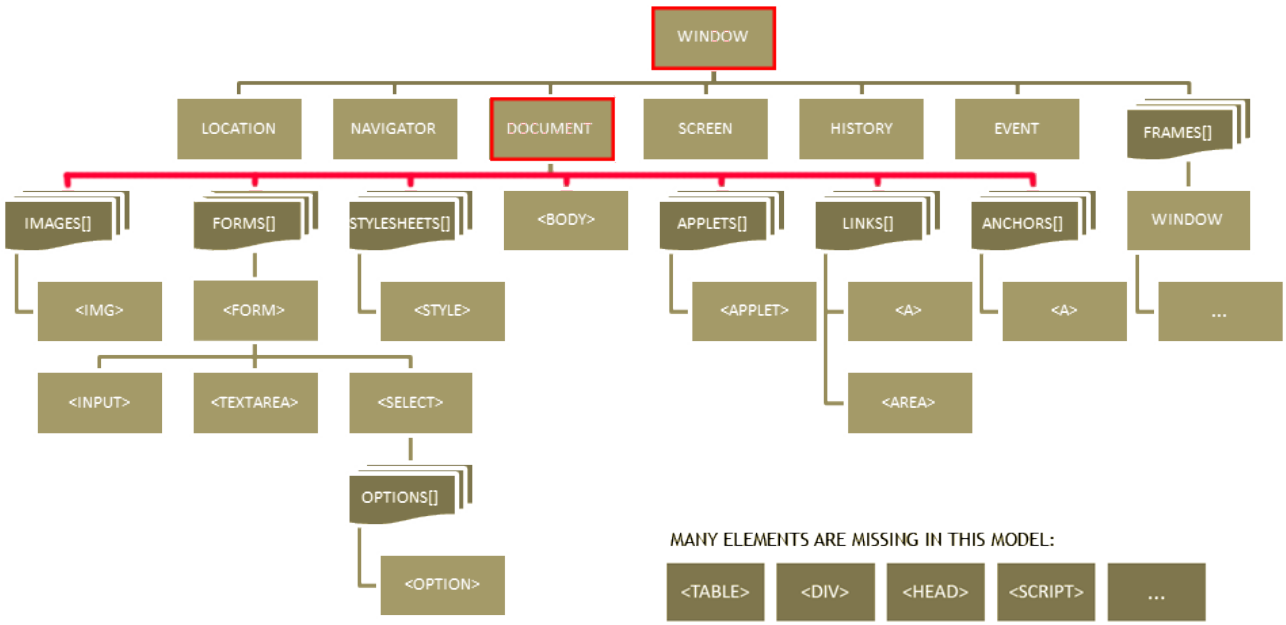
El DOM tiene infinitas posibilidades. Algunas tareas habituales en la programación de aplicaciones web con *JavaScript* son obtener el valor almacenado por elementos de un formulario, crear o eliminar un elemento de forma dinámca o manipular (poner/quitar/cambiar) una clase de un elemento, por poner algunos ejemplos.

Estructura jerárquica propuesta en el modelo DOM y sus nodos

El DOM ofrece una representación del documento como un grupo de nodos y objetos estructurados que tienen propiedades y métodos. El navegador interpreta las etiquetas de un documento HTML y confecciona una estructura DOM en la que organiza jerárquicamente todos los elementos que conforman la web y entrega una representacion estructurada en forma de árbol con ramas.

El DOM se divide en tres partes principales: el árbol de nodos, el API de acceso al documento y el API de manipulación.

El árbol de nodos es la representación en memoria de la estructura de la página web. Cada elemento HTML es un nodo en el árbol y cada nodo puede tener hijos o descendientes y padres o ascendentes.



Existen 12 tipos de nodos en el DOM pero aquellos que se utilizan con más frecuencia son:

- ***document***: Es el nodo raíz del que derivan todos los demás nodos del árbol.
- ***element***: Representa cada una de las etiquetas HTML. Se trata del único nodo que puede contener atributos y del que pueden derivar otros nodos.
- ***attr***: Se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par atributo=valor.
- ***text***: Nodo que contiene el texto delimitado por una etiqueta HTML.

El DOM API o API de acesso al Documento proporciona una interfaz para acceder a cada nodo en el árbol mediante métodos como *getElementById*, obtiene un elemento por su identificador único, *getElementsByTagName*, devuelve una colección de elementos HTML que tienen un nombre de etiqueta común, o *querySelector*, que permite seleccionar elementos utilizando selectores CSS.

El API de manipulación permite modificar la estructura, el contenido y los estilos de los elementos HTML en una página web, mediante métodos que posibilitan agregar, eliminar y modificar elementos HTML, cambiar estilos o manejar eventos, entre otras acciones. Por ejemplo, algunos métodos de esta API son *createElement*, crea el nuevo elemento dado, *appendChild*, agrega un nodo a un elemento del DOM, *removeChild*, elimina un hijo de un elemento, o *setAttibute*, que establece el valor de un atributo en un elemento.

Métodos y/o propiedades

Estos son algunos de los métodos y/o propiedades de JavaScript que permiten hacer manipulaciones avanzadas en el árbol DOM de una web:

- ***offsetWidth* y *offsetHeight***: Estas propiedades de sólo lectura, devuelven el ancho y alto del *layout* del elemento como un entero, respectivamente. Esta medida incluye los bordes y el *padding* asignados al elemento en los estilos CSS.

```
.rect {
   padding: 5px 10px;
   border: 1px solid;
}
<div id="myElement" class="rect" width="50" height="50"></div>

let widthElement = document.getElementById('myElement').offsetWidth; // 62
let heightElement = document.getElementById('myElement').offsetHeight; // 72
```

- ***firstElementChild* y *lastElementChild***: Estas propiedades de sólo lectura de la interfaz *Node* devuelven el primer y último elemento hijo, respectivamente, o *null* si no tiene elementos hijos.

```
<div id="myElement">
    <p>Text 1</p>
    <p>Text 2</p>
</div>

let firstChild = document.getElementById('myElement').firstElementChild;
console.log(firstChild.textContent) // devuelve Text 1

let lastChild = document.getElementById('myElement').lastElementChild;
console.log(lastChild.textContent) // devuelve Text 2
```

- ***classList***: Devuelve una colección activa de *DOMTokenList* de los atributos de clase del elemento. Esta propiedad de sólo lectura aunque puede ser modificada usando los métodos *add()*, agrega las clases indicadas, *remove()*, elimina las clases indicadas, *toogle()*, alterna el valor de la clase eliminándola si existe y añadiéndola si no, y *replace()*, que reemplaza una clase existente por una nueva.

```
<div id="myElement" class="foo old-class">Text</div>

const element = document.getElementById('myElement');

// añade la clase new-class
let addClass = element.classList.add('new-class');

// elimina la clase old-class
let removeClass = element.classList.remove('old-class');

// agrega la clase active
let toogleClass = element.classList.toogle('active');

// cambia la clase new-class a replace-class
let replaceClass = element.classList.replace('foo', 'bar');
```

Referencias bibliográficas

- Fundación Wikipedia. (2024). *Document Object Model.* https://es.wikipedia.org/wiki/Document_Object_Model
- HostGator. (2024). *Qué es DOM y ¿por qué esta interface es esencial en la web?*. https://www.hostgator.mx/blog/dom-document-object-model/
- Campus Training. (2024). *Dom: ¿qué es en informática?.* https://www.campustraining.es/noticias/dom-que-es-informatica/
- desarrolloweb.com. (2024). *Historia del estándar DOM del W3C*. https://desarrolloweb.com/articulos/historia-estandar-dom.html
- mclibre.org. (2024). *Historia de la Web: Normas y recomendaciones*. https://www.mclibre.org/consultar/htmlcss/otros/historia.html
- 3CON14-LAB. (2024). *DOM*. http://3con14.biz/code/javascript/9-dom.html
- Mozilla. (2024). *HTMLElement: offsetWidth property*. https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/offsetWidth
- Mozilla. (2024). *Node: firstElementChild property*. https://developer.mozilla.org/en-US/docs/Web/API/Element/firstElementChild
- Mozilla. (2024). *Element.classList*. https://developer.mozilla.org/en-US/docs/Web/API/Element/classList