DOM: Document Object Model (Modelo de Objetos del Documento)

Es una interfaz de programación para documentos HTML definiendo:

- Objetos: Elementos HTML
- Propiedades de los elementos HTML
- Métodos de manipulación de elementos HTML
- Eventos para los elementos HTML

Es decir, proporciona un estándar de cómo obtener, modificar,añadir o eliminar elementos HTML. Por ejemplo, crear un botón nuevo, añadir una fila a una tabla, cambiar el contenido de un etiqueta…

En DOM los documentos no se representan como texto plano, sino en **forma de árbol de nodos**. Por lo tanto para manipular el documento, tendremos que manipular los nodos:

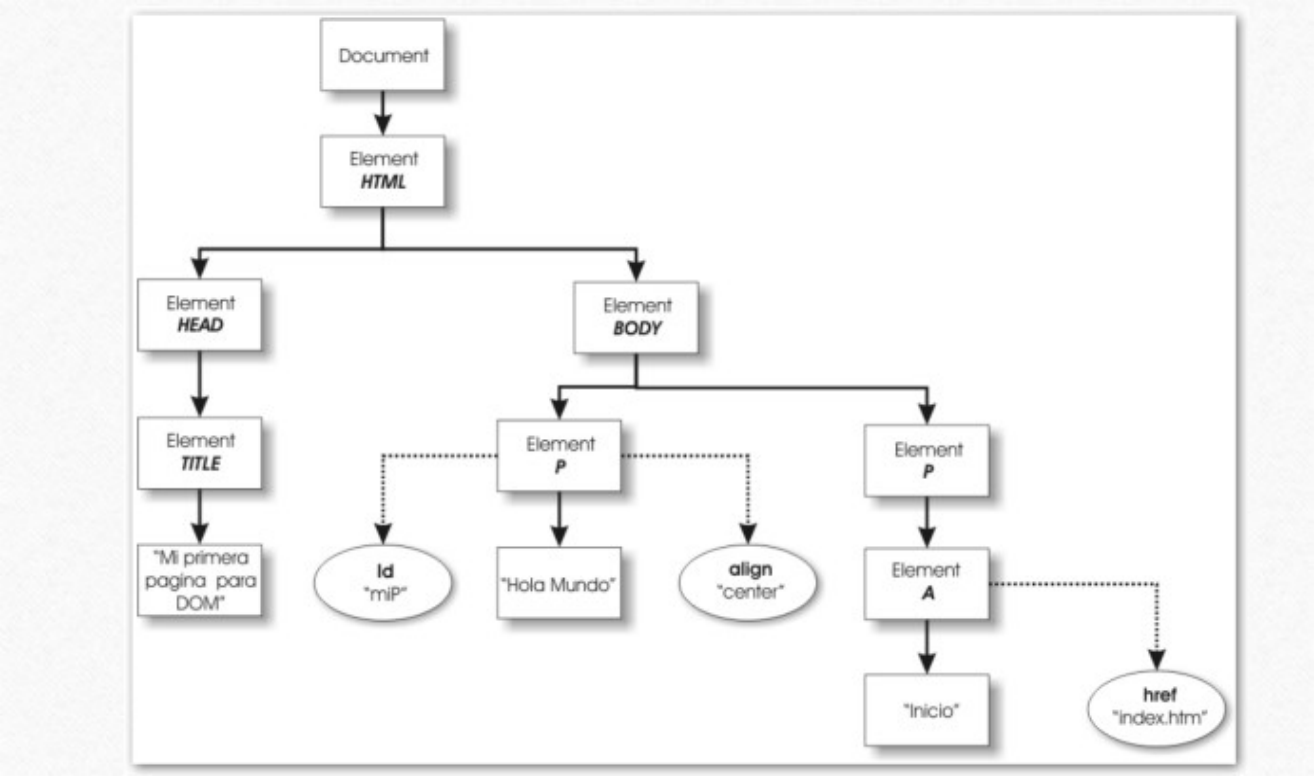
- Todos los nodos son de tipo **Node,** pero hay subtipos: Element, Text, Comment, etc...
- La raíz del árbol es un nodo de tipo documento.
- Los nodos más usuales pueden ser:
  - **Elemento**: etiquetas del HTML
  - **Atributos**: atributos de los elementos. Realmente, los atributos no están en el árbol, y accederemos a ellos a través de métodos del nodo que los posea.
  - **Texto**: Texto plano entre etiquetas de inicio y fin
  - **Comment**: Comentarios HTML

IMPORTANTE: El orden de los nodos es crucial

Ejemplo:

```
<HTML>
     <HEAD>
          <TITLE>Mi primera página para DOM</TITLE>
     </HEAD>
     <BODY>
      <P id="primera" align="center">Hola Mundo!</P>
      <P id="segunda">
          <A href="index.htm">Inicio</A>
      </P>
     </BODY>
</HTML>
```

DOM asociado:



Podemos acceder a los elementos del DOM:

- Por su ID: let myID = document.getElementById(“idMiBoton”);
- Por su etiqueta: let myEnlace = document.getElementsByTagName(“a”)[0];
- Por su name: let myLibro = document.getElementsByName(“libros”)[0];
- Por su nombre de clase let myFilas = document.getElementsByClassName(“row”);

Una vez obtenida la referencia a un nodo, podemos obtener sus propiedades:

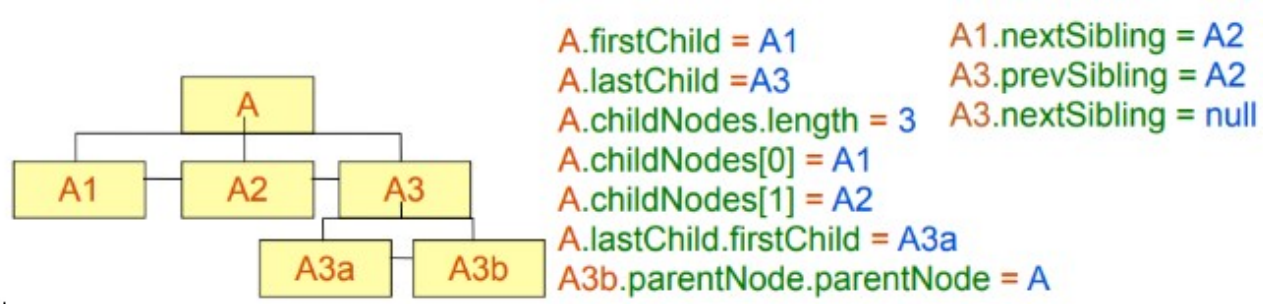
- **nodeType**: constante entera que representa el tipo del nodo
- **nodeName**: nombre
- **nodeValue**: valor

Tipo de nodo	nodeType	nodeName	nodeValue
Etiqueta	1 (Node.ELEMENT_NODE)	Nombre de la etiqueta sin los “<>” y en máyúsc.	null
Texto	3 (Node.TEXT_NODE)	#text	Texto del nodo
Comentario	8 (Node.COMMENT_NODE)	#comment	Texto del comentario
DOCTYPE	10(Node.DOCUMENT_TYPE_NODE)	Nombre de la etiq. raíz del DOCTYPE	null
Documento	9 (Node.DOCUMENT_NODE)	#document	null

Acceso a un nodo desde otro

Cada nodo tiene una seria de propiedades que reflejan el “parentesco” con otros. Algunas de las cuales son:

- **ChildNodes / Children:** Array con los nodos hijos
- **FirstChild / FirstElementChild:** Primer nodo hijo
- **LastChild / lastElementChild**: Último nodo hijo
- **ParentNode / parentElement:** Nodo padre
- **NextSibling / nextElementSibling:** siguiente hermano al mismo nivel
- **PrevSibling / previousElementSibling:** hermano anterior.



Veamos diferencias:

- childNodes vs children
  - **childNodes** devuelve un array con todos los nodos hijos (elementos, texto, comentarios..)
  - **children** devuelve un array con todos los nodos hijos de tipo elemento
- firstChild vs firstElementChild
  - **firstChild**: primer hijo de un nodo
  - **firstElementChild**: primer hijo de tipo Element.
- lastChild vs lastElementChild
  - **lastChild**: último hijo de un nodo
  - **lastElementChild**: último hijo de tipo Element.
- previousSibling vs previousElementSibling
  - **previousSibling**: anterior hijo de un nodo
  - **previousElementSibling**: anterior hijo de tipo Element.

- nextSibling vs nextElementSibling
  - **nextSibling**: siguiente hijo de un nodo
  - **nextElementSibling**: siguiente hijo de tipo Element.

- parentNode vs parentElement
  - **parentNode**: padre del nodo
  - **parentElement**: padre del nodo de tipo Element.

Cuidado con los espacios en blanco: se interpretan como nodos de texto.

- Vamos a practicar:

```
<div id="myDIV">
    <p>First p element</p>
    <p>Second p element</p>
</div>
```

1.- Poner el fondo del primer <p> de color amarillo.
2.- Poner el fondo del segundo <p> de color naranja.

Formas de acceso a todos los nodos del mismo tipo:

Ejemplo1: Poner el color a rojo de todos los elementos p

```
let nodos = document.getElementsByTagName(“p”);
for (var i = 0; i < nodos.length; i++){
    nodos[i].style.color = “red";  //la propiedad style representa el estilo CSS, con subpropiedades que son nombres
}                                  //de propiedades CSS
```

Ejemplo2: Obtener todas las filas de una tabla con id “tabla1”.

```
let tabla1 = document.getElementById("tabla1");
let filasDeTabla1 = tabla.getElementsByTagName("tr");
```

Modificando valores: setAttribute, nodeValue, innerHTML, textContent

```
<input type="button" value="Enviar" id="b1">
<p id="p" >TEXTO</p>


<script language="JavaScript">
    let p = document.getElementById("p");
    let boton = document.getElementById(“b1”);
    boton.setAttribute(“value”, “Enviado”);     // modifico el atributo value del botón
    p.setAttribute("align","right");              // modifico el atributo align de la etiqueta <p>
    p.firstChild.nodeValue="He cambiado el texto del p";
    p.innerHTML=”Otro texto”;
    p.textContent =”Nuevo texto”;
</script>
```

Crear nuevos nodos

- **document.createElement(“etiqueta”):** Crea un nodo etiqueta. Se le pasa el nombre de la etiqueta sin “<>”.
- **document.createTextNode(“texto”):** Crea un nodo de texto, con el contenido especificado.

IMPORTANTE: Hay que insertar los nodos creados en el lugar apropiado del árbol:

```
<body id="cuerpo">
<script language="JavaScript">
  var par = document.createElement("p");
  var texto = document.createTextNode("Yo antes no existía!”);
  par.appendChild(texto);
  document.getElementById("cuerpo").appendChild(par);
</script>
</body>
```

¿Qué hace este código?

Insertar o eliminar Nodos

Para insertar y para eliminar nodos, se usan estos métodos que tienen que ser llamados desde el padre del nodo a insertar o a eliminar:

- padre.**appendChild**(nuevoHijo) : Añade al final de todos los hijos actuales del padre el nuevoHijo.
- padre.**insertBefore**(nuevoHijo, hijoReferencia): Inserta el nuevoHijo justo antes del hijoReferencia.
- padre.**removeChild**(hijoABorrar): Se borra el hijoABorrar que pertenece al padre.
- padre.**replaceChild**(nuevoHijo, hijoAntiguo): reemplaza un hijo por otro nuevo

Ejercicio 1: Dado este código, añadir un tercer elemento <p>

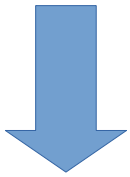
```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
   <p>This is new.</p>
</div>
```

¿Y si queremos asignarle un id?

Ejercicio 2: Dado este código, añadir un tercer elemento <p> antes del primero

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```



```
<div id="div1">
   <p>This is new.</p>
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

Ejercicio 3: Dado este código, eliminar el primer elemento

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

Ejercicio 4: Reemplazar el primer <p> por otro elemento <p> que ponga: This is new

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```
<div id="div1">
   <p>This is new.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

Recordatorio:

```
let formularios = document.forms;

let imagenes = document.images;

let links = document.links;

let body = document.body;
```

```
//Reducir el tamaño de todas las imágenes a la mitad, con DOM core
imags = document.getElementsByTagName("img");
for(i=0; i<imags.length; i++) {
    imags[i].setAttribute("width",imags[i].getAttribute("width")/2);
    imags[i].setAttribute("height",imags[i].getAttribute("height")/2);
}
//idem con DOM HTML
for(var i=0; i<document.images.length; i++) {
    document.images[i].width /= 2;
    document.images[i].height /= 2;
}
```

Ejercicios:

5.- Crear una página HTML vacia. Añadir un párrafo (<p>) con un texto a la página HTML.
6.- Crear 3 párrafos con sus id individuales. Desde javascript borrar el primero de los párrafos.
7.- Dado un enlace en una pagina html, acceder desde javascript a su atributo href.
8.- Dada una imagen en la pagina html con margen, acceder desde javascript a su atributo margen
9.- Dado un párrafo con la letra en negrita(Font-weight:bold), acceder desde javascript a dicho atributo. Asignarle una clase de estilo y acceder al nombre de la clase desde js.
10.- Dada la siguiente lista en html: Añadir un botón que al pulsarse añada un elemento nuevo li a la lista.

```
<ul id="lista">
        <li>Lorem ipsum dolor sit amet</li>
        <li>Consectetuer adipiscing elit</li>
        <li>Sed mattis enim vitae orci</li>
        <li>Phasellus libero</li>
        <li>Maecenas nisl arcu</li>
</ul>
```

11.- Realizar los apartados siguientes:
A. Dado un array con las estaciones del año, crear desde javascript una lista con cada uno de los valores del array. Tendrá un titulo Estaciones del año. Utilizar createElement
B. Dado un array con los continentes, crear ahora la lista utilizando innerHTML.

12.- Crear 3 parrafos con <p> con los id= contenidos_1, contenidos_2 y contenidos_3. Detrás de cada párrafo, poner un enlace <a> con el texto ‘Ocultar Contenido’ y con sus id= enlace_1, enlace_2 y enlace_3.
Realizar un programa que desde código javascript al pulsar un enlace se ocultará el texto del parrafo.correspondiente. Cuando se oculte el texto del enlace deberá mostrar ‘Mostrar Contenido’ y al pulsarlo se visualizará el parrafo volviendo a mostrar el enlace el texto ‘ocultar contenido’

13.- Crear un botón dentro de una etiqueta section (<section id="ContentFormulario">)
de tal forma que al pulsarlo, desde javascript se genere un formulario. Dicho formulario debe tener los siguientes atributos: una anchura de 300px, un action a la página de google y el method será get.
Además, el formulario debe contener:
- Un input de tipo text para el nombre, con el atributo placeholder ‘Nombres’ y un estilo:
width:100%;margin: 10px 0px;padding: 5px
- Un input de tipo text para los apellidos, con el atributo placeholder ‘Apellidos’ y un estilo:
width:100%;margin: 10px 0px;padding: 5px

- Un input de tipo text para el email, con el atributo placeholder ‘Email’ y un estilo:
width:100%;margin: 10px 0px;padding: 5px
- Un input de tipo text para el asunto, con el atributo placeholder ‘Asunto’ y un estilo:
width:100%;margin: 10px 0px;padding: 5px
- Un input de tipo text para el Mensaje, con el atributo placeholder ‘Mensaje’ y un estilo:
width:100%;height:200px;margin: 10px 0px;padding: 5px
- Un botón con el valor ‘Enviar’ con estilo width:100px;margin: 10px 0px;padding: 10px;background:#F05133;color:#fff;border:solid 1px #000; y con un mensaje de alert cuando se pulse el botón.

14- A partir de una lista de 10 elementos (ejemplo lorem), preguntar al usuario con mensaje de prompt un texto a introducir y una posición del 1 al 10. Colocar el texto en la posición indicada. Si la posición no es correcta indicarlo.
NOTA: si se quiere añadir en la última posición, el texto nuevo tendrá que quedar el último de la lista.

15.- A partir del siguiente código html

```html
<h1>Tabla HTML dibujada con JS</h1>
<table>
    <thead>
        <tr>
            <th>Nombre</th>
            <th>Precio</th>
            <th>Código</th>
        </tr>
    </thead>
    <tbody id="cuerpoTabla">
    </tbody>
</table>
```

Crear desde javascript las filas y columnas de dicha tabla dentro del cuerpo de la tabla (tbody). Crear un **array** de 4 elementos, donde cada elemento será un **objeto literal** con los campos: id, nombre, precio y código. Cada objeto literal tendrá los datos de una fila. El id del objeto tendrá que asignárselo al id de la fila(<tr>).

16.- Dada una página con un botón y dos párrafos de texto, poner el fondo del segundo párrado en color rojo desde código javascript. LOS PARRAFOS NO TIENEN ID.

17.- Crear una tabla desde javascript con 2 filas y dos columnas. El texto de cada valor será: ‘Posición: ij’. La primera columna tendrá un fondo de color rojo, la segunda columna debe ocultarse.

18.- Partir del ejercicio 15 que mostraba una tabla con 4 filas pero ahora añadir desde javascript una columna nueva, que será un checkbox. Cada checkbox tiene un id diferente: “checkbox1”, “checkbox2”,… y un name con el valor “marcar.”

19.- Partir del ejercicio 18. Debajo de la tabla, habrá 3 botones:

• Uno que seleccione todos los checkbox de la tabla
• Otro que deseleccione todos los checks marcados de la tabla
• Otro que elimine aquellas filas que estén marcadas con el checkbox.

20.- Partir del ejercicio 19. El formulario ahora tendrá además 3 input de tipo text antes de la tabla para poder almacenar el nombre , el precio y el codigo de un nuevo producto. También hay un botón que cuando se pulse, cree una nueva fila con los datos introducidos. ¿Cómo podemos optimizar el código?

21.- Partir del ejercicio anterior para que cuando se pase por una fila de la tabla, pondrá su fondo de color amarillo.