UNIDAD 6 - ESTRUCTURA DOM

1. DOM (Document Object Model):

La creación del *Document Object Model* o **DOM** es una de las innovaciones que más ha influido en el desarrollo de las páginas web dinámicas y de las aplicaciones web más complejas.

DOM permite a los programadores web acceder y manipular las páginas XHTML como si fueran documentos XML. De hecho, DOM se diseñó originalmente para manipular de forma sencilla los documentos XML.

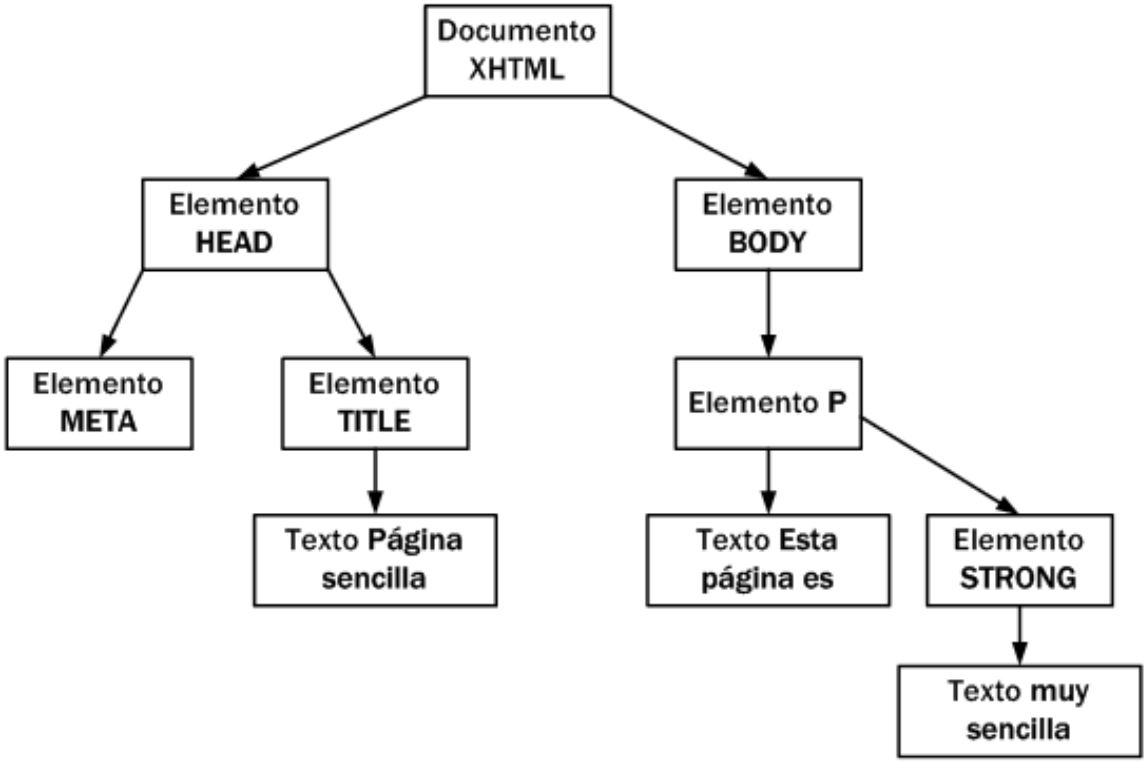
1.1 Arbol de nodos:

Una de las tareas habituales en la programación de aplicaciones web con JavaScript consiste en la manipulación de las páginas web. De esta forma, es habitual obtener el valor almacenado por algunos elementos (por ejemplo los elementos de un formulario), crear un elemento (párrafos, , etc.) de forma dinámica y añadirlo a la página, aplicar una animación a un elemento (que aparezca/desaparezca, que se desplace, etc.).

Todas estas tareas habituales son muy sencillas de realizar gracias a DOM. Sin embargo, para poder utilizar las utilidades de DOM, es necesario *"transformar"* la página original. Una página HTML normal no es más que una sucesión de caracteres, por lo que es un formato muy difícil de manipular. Por ello, los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular.

Esta transformación la realizan todos los navegadores de forma automática y nos permite utilizar las herramientas de DOM de forma muy sencilla. El motivo por el que se muestra el funcionamiento de esta transformación interna es que condiciona el comportamiento de DOM y por tanto, la forma en la que se manipulan las páginas.

DOM transforma todos los documentos XHTML en un conjunto de elementos llamados *nodos*, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos. Por su aspecto, la unión de todos los nodos se llama *"árbol de nodos"*.



En el esquema anterior, cada rectángulo representa un nodo DOM y las flechas indican las relaciones entre nodos. Dentro de cada nodo, se ha incluido su tipo y su contenido.

La raíz del árbol de nodos de cualquier página XHTML siempre es la misma: un nodo de tipo especial denominado *"Documento"*. A partir de ese nodo raíz, cada etiqueta XHTML se transforma en un nodo de tipo *"Elemento"*. La conversión de etiquetas en nodos se realiza de forma jerárquica. De esta forma, del nodo raíz solamente pueden derivar los nodos HEAD y BODY. A partir de esta derivación inicial, cada etiqueta XHTML se transforma en un nodo que deriva del nodo correspondiente a su *"etiqueta padre"*. La transformación de las etiquetas XHTML habituales genera dos nodos: el primero es el nodo de tipo *"Elemento"* (correspondiente a la propia etiqueta XHTML) y el segundo es un nodo de tipo

1.2 Tipos de nodos:

La especificación completa de DOM define 12 tipos de nodos, aunque las páginas XHTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

- ***Document***, nodo raíz del que derivan todos los demás nodos del árbol.
- ***Element,*** *representa cada una de las etiquetas XHTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.*

- ***Attr,*** se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas XHTML, es decir, uno por cada par atributo=valor.
- ***Text,*** nodo que contiene el texto encerrado por una etiqueta XHTML.
- ***Comment,*** representa los comentarios incluidos en la página XHTML.

1.3 Acceso directo a los nodos:

Una vez construido automáticamente el árbol completo de nodos DOM, ya es posible utilizar las funciones DOM para acceder de forma directa a cualquier nodo del árbol. Como acceder a un nodo del árbol es equivalente a acceder a "un trozo" de la página, una vez construido el árbol, ya es posible manipular de forma sencilla la página: acceder al valor de un elemento, establecer el valor de un elemento, mover un elemento de la página, crear y añadir nuevos elementos, etc.

Es importante recordar que el acceso a los nodos, su modificación y su eliminación solamente es posible cuando el árbol DOM ha sido construido completamente, es decir, después de que la página XHTML se cargue por completo.

1.4 Funciones:

- **getElementsByTagName():** La función getElementsByTagName(nombreEtiqueta) obtiene todos los elementos de la página XHTML cuya etiqueta sea igual que el parámetro que se le pasa a la función. El valor que devuelve la función es un array con todos los nodos que cumplen la condición de que su etiqueta coincide con el parámetro proporcionado. El valor devuelto es un array de nodos DOM.

- **getElementsByName():** La función getElementsByName() es similar a la anterior, pero en este caso se buscan los elementos cuyo atributo name sea igual al parámetro proporcionado.

- **getElementById():** La función getElementById() es la más utilizada cuando se desarrollan aplicaciones web dinámicas. Se trata de la función preferida para acceder directamente a un nodo y poder leer o modificar sus propiedades. La función getElementById() devuelve el elemento XHTML cuyo atributo id coincide con el parámetro indicado en la función. Como el atributo id debe ser único para cada elemento de una misma página, la función devuelve únicamente el nodo deseado.

1.5 Creación de elementos XHTML simples:

Como se ha visto, un elemento XHTML sencillo, como por ejemplo un párrafo, genera dos nodos: el primer nodo es de tipo **Elemen**t y representa la etiqueta <p> y el segundo nodo es de tipo y representa el contenido textual de la etiqueta <p>. Por este

motivo, crear y añadir a la página un nuevo elemento XHTML sencillo consta de cuatro pasos diferentes:

1. Creación de un nodo de tipo Element que represente al elemento.
2. Creación de un nodo de tipo Text que represente el contenido del elemento.
3. Añadir el nodo Text como nodo hijo del nodo Element.
4. Añadir el nodo Element a la página, en forma de nodo hijo del nodo correspondiente al lugar en el que se quiere insertar el elemento.

De este modo, si se quiere añadir un párrafo simple al final de una página XHTML, es necesario
incluir el siguiente código JavaScript:

```
// Crear nodo de tipo Element
var parrafo = document.createElement("p");
// Crear nodo de tipo Text
var contenido = document.createTextNode("Hola Mundo!");
// Añadir el nodo Text como hijo del nodo Element
parrafo.appendChild(contenido);
// Añadir el nodo Element como hijo de la pagina
document.body.appendChild(parrafo);
```

El proceso de creación de nuevos nodos puede llegar a ser tedioso, ya que implica la utilización
de tres funciones DOM:

- **createElement(etiqueta)**: crea un nodo de tipo Element que representa al elemento XHTML cuya etiqueta se pasa como parámetro.
- **createTextNode(contenido)**: crea un nodo de tipo Text que almacena el contenido textual de los elementos XHTML.
- **nodoPadre.appendChild(nodoHijo)**: añade un nodo como hijo de otro nodo. Se debe utilizar al menos dos veces con los nodos habituales: en primer lugar se añade el nodo Text como hijo del nodo Element y a continuación se añade el nodo Element como hijo de algún nodo de la página.

Diferencia entre append y appendChild: La principal diferencia es que appendChild es una función DOM mientras que append es una función de JavaScript.

La función **appendChild** necesita un "elemento" como un parámetro.

NO PUEDES hacer esto:

```
document.getElementById("yourId").appendChild("<p></p>");
```

Pero PUEDES hacer esto.

```
var p = document.createElement("p");
document.getElementById("yourId").appendChild(p);
```

En todos los casos en los que pueda usar appendChild , puede usar append. Pero no al revés.

Prepend: El prepend()método inserta un conjunto de Nodeobjetos u DOMStringobjetos después del primer hijo de un nodo padre.

1.6 Eliminación de nodos:

Afortunadamente, eliminar un nodo del árbol DOM de la página es mucho más sencillo que añadirlo. En este caso, solamente es necesario utilizar la función **removeChild():**

```
var parrafo = document.getElementById("provisional");
parrafo.parentNode.removeChild(parrafo);
<p id="provisional">...</p>
```

La función removeChild() requiere como parámetro el nodo que se va a eliminar. Además, esta función debe ser invocada desde el elemento padre de ese nodo que se quiere eliminar. La forma más segura y rápida de acceder al nodo padre de un elemento es mediante la propiedad nodoHijo.parentNode.

Así, para eliminar un nodo de una página XHTML se invoca a la función removeChild() desde el valor parentNode del nodo que se quiere eliminar. Cuando se elimina un nodo, también se eliminan automáticamente todos los nodos hijos que tenga, por lo que no es necesario borrar manualmente cada nodo hijo.

Una vez que se ha accedido a un nodo, el siguiente paso natural consiste en acceder y/o modificar sus atributos y propiedades. Mediante DOM, es posible acceder de forma sencilla a todos los atributos XHTML y todas las propiedades CSS de cualquier elemento de la página.

1.7 Acceso directo a los atributos XHTML:

Los atributos XHTML de los elementos de la página se transforman automáticamente en propiedades de los nodos. Para acceder a su valor, simplemente se indica el nombre del atributo XHTML detrás del nombre del nodo.

El siguiente ejemplo obtiene de forma directa la dirección a la que enlaza el enlace:

```
var enlace = document.getElementById("enlace");
alert(enlace.href); // muestra http://www...com

<a id="enlace" href="http://www...com">Enlace</a>
```

Las propiedades CSS no son tan fáciles de obtener como los atributos XHTML. Para obtener el valor de cualquier propiedad CSS del nodo, se debe utilizar el atributo style. El siguiente ejemplo obtiene el valor de la propiedad margin de la imagen:

```
var imagen = document.getElementById("imagen");
alert(imagen.style.margin);

<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

Si el nombre de una propiedad CSS es compuesto, se accede a su valor modificando ligeramente
su nombre:

```
var parrafo = document.getElementById("parrafo");
alert(parrafo.style.fontWeight); // muestra "bold"

<p id="parrafo" style="font-weight: bold;">...</p>
```

La transformación del nombre de las propiedades CSS compuestas consiste en eliminar todos los guiones medios (-) y escribir en mayúscula la letra siguiente a cada guión medio. A continuación se muestran algunos ejemplos:

- ***font-weight se transforma en fontWeight***
- ***line-height se transforma en lineHeight***
- ***border-top-style se transforma en borderTopStyle***
- ***list-style-image se transforma en listStyleImage***

El único atributo XHTML que no tiene el mismo nombre en XHTML y en las propiedades DOM es el atributo class. Como la palabra class está reservada por JavaScript, no es posible utilizarla para acceder al atributo class del elemento XHTML. En su lugar, DOM utiliza el nombre **className** para acceder al atributo class de XHTML:

```
var parrafo = document.getElementById("parrafo");
alert(parrafo.class); // muestra "undefined"
alert(parrafo.className); // muestra "normal"
<p id="parrafo" class="normal">...</p>
```

2. LISTA DE PROPIEDADES DE JAVASCRIPT PARA MODIFICAR EL ESTILO DE LAS PAGINAS:

Propiedad	Descripcion
background	Establece todas las propiedades del background o fondo en una única declaración de forma abreviada, son 5 las propiedades separadas por un espacio: *background-color, background-image, background-repeat, background-attachment, background-position* Se usa de la siguiente forma: *Object.style.background="color image repeat attachment position"* Por ejemplo: *document.body.style.background="#f3f3f3 url('foto.png') no-repeat right top";*
backgroundAttachment	Especifica cuando la imagen empleada como fondo permanece fija o se desplaza.Las opciones son: *scroll* Se desplaza con la pagina (Predeterminado) *fixed* Permanece fija
backgroundColor	Especifica el color del fondo
backgroundImage	Especifica la ubicación de la imagen a emplear como fondo. Por ejemplo: *document.body.style.backgroundImage="url('foto.png')";*
backgroundPosition	Posición de la imagen utilizada. Se usan dos valores, si solo se especifica uno la imagen será centrada. Los valores pueden ser: *top left, top center, top right, center left, center center, center right, bottom left, bottom center, bottom right* *x% y%* *xpos ypos*
backgroundRepeat	Especifica si la imagen empleada se repite para llenar toda el área de la página. *repeat* Es repetida en las dos dimensiones, es el valor predeterminado.

	repeat-x Solo se repite en el eje horizontal *repeat-y* Solo se repite en el eje vertical *no-repeat* No es repetida
border	Establece las propiedades del borde en una sola declaración de la siguiente forma: *Object.style.border="width style color"* Por ejemplo: *document.getElementById("ejemplo").style.border="thick solid green";* Las restantes propiedades que se pueden usar de forma individual para definir el estilo del borde son: *borderBottom* *borderBottomColor* *borderBottomStyle* *borderBottomWidth* *borderColor* *borderLeft* *borderLeftColor* *borderLeftStyle* *borderLeftWidth* *borderRight* *borderRightColor* *borderRightStyle* *borderRightWidth* *borderStyle* *borderTop* *borderTopColor* *borderTopStyle* *borderTopWidth* *borderWidth*
outline	Especifica las propiedades de outline (borde de fuente) en una sola declaración de la siguiente forma: *Object.style.outline="width style color"* Por ejemplo: *document.getElementById("ejemplo").style.outline="thick solid #0000FF";* Las restantes propiedades que se pueden usar de forma individual

	para definir el estilo outline son: *outlineColor* *outlineStyle* *outlineWidth*
listStyle	Especifica las siguientes propiedades en una sola declaración: list-style-image, list-style-position y list-style-type, se usa: *Object.style.listStyle="type position image"* También se pueden usar de forma individual: *listStyleImage* *listStylePosition* *listStyleType*
margin	Establece todas las propiedades de los márgenes en una sola declaración. Esta propiedad puede establecerse indicando desde 1 a 4 valores. • Un valor, por ejemplo: div {margin: 50px} todos los márgenes serán de 50px • Dos valores, por ejemplo: div {margin: 50px 10px} top (superior) y bottom (inferior) serán de 50px, left (izquierda) y right (derecha) serán de 10px. • Tres valores, por ejemplo: div {margin: 50px 10px 20px} el valor de top será 50px, left y right será 10px, bottom sera 20px. • Cuatro valores, por ejemplo: div {margin: 50px 10px 20px 30px} el valor de top será 50px, right será de 10px, bottom será de 20px, left será de 30px. Los valores se pueden definir de tres formas alternativas *"% length auto"* Por ejemplo: *document.getElementById("ejemplo").style.margin="2px 2px 5px 5px";* También se pueden usar de forma individual: *marginBottom* *marginLeft* *marginRight* *marginTop*
padding	Especifica de forma conjunta los valores del padding (espaciado)de un elemento, se pueden usar hasta cuatro valores. Para emplear los valores utiliza el mismo método de margin. Por ejemplo:

	document.getElementById("ejemplo").style.padding="10px 0 5px 0"; También se pueden usar de forma individual: *paddingBottom* *paddingLeft* *paddingRight* *paddingTop*
cssText	Especifica una declaración de estilo como una cadena de texto.
clear	Especifica la posición de un elemento de forma relativa a un objeto que flota.
clip	Especifica que parte de un elemento posicionado es visible.
cssFloat	Establece la alineación horizontal de un elemento. Pueden usarse tres valores de la siguiente forma: *Object.style.cssFloat="left|right|none"* Por ejemplo: *document.getElementById("ejemplo").style.cssFloat="left";*
cursor	Establece el estilo a emplearse en el cursor del ratón de la siguiente forma: *Object.style.cursor="valor"* Los valores pueden ser: *auto(predeterminado), crosshair, e-resize, help, move, n-resize, ne-resize, nw-resize, pointer, s-resize, se-resize, sw-resize, text, url, w-resize, wait.* Por ejemplo: *document.getElementById("ejemplo").style.cursor="pointer";*
display	Define la forma en que se muestra un elemento HTML, los métodos más empleados son: "inline" o "block", también es muy usado para ocultar elementos de la forma siguiente: *document.getElementById("ejemplo").style.display="none";*
overflow	Especifica que hacer si el contenido de un elemento rebasa el espacio que proporciona este.
position	Especifica el tipo de posicionamiento usado para un elemento, pueden usarse los siguientes valores (*static, relative, absolute o fixed*)
verticalAlign	Especifica la alineación vertical de un elemento
visibility	Especifica la visibilidad de un elemento
zIndex	Especifica el orden del posicionamiento de un elemento
orphans	Especifica el mínimo numero de líneas para un elemento que tiene que

	ser visible en la parte inferior de la página.
widows	Especifica el mínimo numero de líneas para un elemento que tiene que ser visible en la parte superior de la página.
borderCollapse	Establece si el borde de una tabla debe colapsar en una simple línea
borderSpacing	Espaciado del borde en una tabla
captionSide	Posición del elemento caption de una tabla
emptyCells	Especifica si se debe mostrar el borde y fondo de una celda vacía
color	Establece el color del texto.
direction	Establece la dirección del texto
font	Establece todas las propiedades del elemento fuente en una sola declaración. Pueden usarse los valores:*font-style, font-variant, font-weight, font-size, line-height y font-family*. Hazlo de la siguiente forma: *Object.style.font="style variant weight size/lineHeight family"* Por ejemplo: *document.getElementById("ejemplo").style.font="italic bold 18px arial,serif";* También pueden usarse de forma individual las siguientes propiedades: *fontFamily* *fontSize* *fontSizeAdjust* *fontStyle* *fontVariant* *fontWeight*
letterSpacing	Especifica el espacio entre caracteres en el texto.
lineHeight	Especifica el espacio entre líneas en el texto.
textAlign	Especifica la alineación horizontal del texto.
textDecoration	Especifica el estilo de decoración del texto.
textIndent	Establece la indentacion utilizada en la primera línea del texto.
textTransform	Establece la propiedad Transform usada en el texto.
wordSpacing	Establece el espacio entre palabras en el texto.
	Para la posición y tamaño de cada elemento se pueden usar las

	siguientes propiedades: *Bottom, left, right, top* *Height, width, maxHeight, maxWidth , minHeight ,minWidth*

3. Funciones para el manejo de la estructura DOM

Funciones que permiten acceder a cualquier node identificado por un selector CSS específico.

3.1 QuerySelector: Devuelve un elemento en el documento que coinciden con el selector CSS especificado. El método querySelector () simplemente devuelve el primer elemento que coincide con el selector especificado. Si tiene que devolver todos los elementos, utilice el método querySelectorAll () en su lugar.

Selectores

```
<div id="myDiv" class="divClass"></div>
<script>
   document.querySelector('#myDiv');
</script>
```

```
// By class name
document.querySelectorAll('.divClass');
// By type
document.querySelectorAll('div');
```

Es posible que se desee realizar una búsqueda un poco más compleja combinando una o más características que pudiera contener el nodo que se desea obtener.

```
<div class="divClass"></div>
<div class="divClass"></div>
<div class="divClass active"></div>

<div class="divClass"></div>
<div class="divClass"></div>
<div class="divClass active"></div>
```

Si se quiere obtener el div con clase divClass que a su vez sea activese pueden combinar ambos criterios de la siguiente manera:

```
document.querySelector('div.divClass.active');
document.querySelector('div[class="divClass active"]');
```

document.querySelector('div.divClass.active');
document.querySelector('div[class="divClass active"]');

La ventaja de comprender un poco más a fondo cómo funcionan estos selectores, es que se pueden utilizar para acceder a un nodo con criterios muy específicos, sin necesidad de hacer una búsqueda genérica y luego tratar de filtrar cada resultado*:*

3.2 **getElementByClassName:** Retorna un objecto similar a un array de los elementos hijos que tengan todos los nombres de clase indicados. Cuando es llamado sobre el objeto document , la busqueda se realiza en todo el document, incluido el nodo raíz. También puedes llamar **getElementsByClassName()** sobre cualquier elemento; en ese caso retornara sólo los elementos hijos del elemento raíz indicado que contengan los nombres de clase indicados.

3.3 **Append, prepend:** append() y prepend() son métodos prácticos para manipular el DOM de manera eficiente y directa. Ambos métodos son ideales para cuando se quiere agregar elementos sin sobrescribir o eliminar los existentes.

3.3.1 **append():** es útil cuando se desea mantener el contenido existente y agregar algo al final (por ejemplo, cuando se muestran mensajes nuevos en una lista o se añaden más resultados a una página).

Ejemplo:

```
<div id="contenedor">
  <p>Primer párrafo</p>
</div>
<script>
        const contenedor = document.getElementById('contenedor');
        const nuevoParrafo = document.createElement('p');
        nuevoParrafo.textContent = "Este es un nuevo párrafo agregado al principio.";
      // Añadir el nuevo párrafo al final del contenedor
        contenedor.append(nuevoParrafo);
</script>
```

3.3.2 **Prepend():** es útil cuando quieres colocar algo al principio (como un mensaje que debe aparecer primero o nuevos elementos en una lista que deben mostrarse antes de los anteriores

Ejemplo:

```
<div id="contenedor">
  <p>Primer párrafo</p>
</div>
<script>
        const contenedor = document.getElementById('contenedor');
        const nuevoParrafo = document.createElement('p');
        nuevoParrafo.textContent = "Este es un nuevo párrafo agregado al principio.";
      // Añadir el nuevo párrafo al principio del contenedor
        contenedor.prepend(nuevoParrafo);
</script>
```

3.4 innerHTML e insertAdjacentHTML

3.4.1 innerHTML obtiene o establece el HTML o XML contenido dentro del elemento.

3.4.2 insertAdjacentHTML() analiza el texto especificado como HTML o XML e inserta los nodos resultantes en el árbol DOM en una posición especificada

Sintaxis:

element.insertAdjacentHTML(position, text);

- **Parámetros**
- **position**
  - **beforebegin**: Antes del elemento.
  - **afterbegin**: Justo dentro del element, antes de su primer hijo.
  - **beforeend**: Justo dentro del element, después de su último hijo.
  - **afterend**: Después del elemento .
- **Text**: La cadena que se analizará como HTML o XML y se insertará en el árbol.