NAVEGAR POR EL DOM

OBTENER LOS HIJOS DE UN ELEMENTO

Todos los elementos disponen de una propiedad llamada **childNodes,** la cual devuelve una lista de nodos. Como ejemplo, utilizaremos el párrafo identificado como párrafol, para obtener su lista de nodos hijo a través de la consola de JavaScript:

```
let parrafoí=document.getElementById("parrafol");
for(let hijo of parrafol.childNodes){
      consolé.log('Texto: "${hijo.nodeValue}"\n' +
                  ` Tipo de nodo: ${hijo.nodeType}, ${hijo.nodeName}');
}
```

Hay una propiedad llamada **children** que funciona igual que childNodes, pero solo obtiene los nodos hijos que sean elementos:

```
for( let hijo of parrafol.children){
      consolé.log('Texto: "${hijo.nodeValue}"\n'+
                  ` Tipo de nodo: ${hijo.nodeType}, ${hijo.nodeName}');
}
```

PROPIEDADES QUE FACILITAN LA NAVEGACIÓN POR EL DOM

PROPIEDAD	USO
firstChild	Selecciona el primer nodo hijo del elemento actual.
lastChild	Selecciona el último nodo hijo del elemento actual.
nextSibling	Selecciona el siguiente hermano respecto del nodo actual. Son hermanos los nodos que están al mismo nivel en el árbol DOM. Dicho de otra forma, son hermanos los nodos que tienen el mismo padre.
previousSibling	Obtiene el hermano anterior al nodo actual.
parentNode	Obtiene el padre del nodo actual.
firstElementChild	Obtiene el primer hijo del nodo actual que sea un elemento.
lastElementChild	Obtiene el último hijo del nodo actual que sea un elemento.
nextElementSibling	Obtiene el siguiente hermano del anterior que sea de tipo elemento
previouslementSibling	Obtiene el elemento hermano anterior al nodo actual
childElementCount	Número de elementos hijos del nodo actual

CREAR ELEMENTOS

El método **createElement** del objeto document es el que permite crear elementos. Requiere que indiquemos el nombre de la etiqueta de dicho elemento:

```
let capal=document.createElement("div");
```

La variable ***capal*** es la referencia al nuevo elemento. Este elemento no se mostrará todavía en el documento, porque realmente no forma parte de los nodos visibles al no haber indicado su posición en el árbol de nodos que cuelgan de **document.**

Sí tiene disponibles las propiedades habituales de los elementos.

```
capal.innerHTML="<p>Esta es mi nueva capa</p>";
capal.setAttribute("id","capal");
```

CREAR NODOS DE TEXTO

El método **createTextNode** crea un nodo de tipo texto. Para ello basta con indicar el texto que deseamos colocar en el nodo:

```
let nodoTl=document.createTextNode("Este texto no se muestra todavía");
```

Al igual que ocurría con createElement, los nodos recién creados no se muestran hasta que no los coloquemos dentro de document.

AÑADIR UN NODO HIJO

El método **appendChild** permite colocar un nodo como hijo de otro elemento. Es un método que poseen todos los nodos del DOM y que tiene como único parámetro el nodo que deseamos colocar como hijo.

Ejemplo:

```
<main id="principal">
      <p>Soy el primer párrafo</p>
      <p>Soy el segundo párrafo</p>
      <p>Soy el tercer párrafo</p>
</main>
<script>
      let capa=document.getE1ementById("principal");
      let nuevoP=document.createElement("p");
      nuevoP.textContent="Soy el nuevo párrafo";
      capa.appendChild(nuevoP);
</script>
```

INSERCIÓN DE NODOS EN POSICIONES CONCRETAS

El método anterior tiene como inconveniente, que el elemento que deseamos añadir debe ir siempre colocado al final. Por ello disponemos de otro método llamado **insertBefore** que permite indicar la posición en la que queremos colocar el elemento. Para ello el nodo que será padre del que queremos añadir, debe de tener nodos hijo. De otro modo deberíamos usar **appendChild.**

El método **insertBefore** tiene dos parámetros: el primero es el nodo que deseamos añadir y el segundo es el nodo hijo delante del cual quedará colocado el que deseamos añadir. Evidentemente no podemos, con este método, añadir un nodo al final, pero para eso disponemos de appendChild.

Usando el mismo código HTML, veamos cómo colocar un nuevo párrafo como segundo párrafo:

```
let capa=document.getElementById("principal");
let nuevoP=document.createElement("p");
nuevoP.textContent="Soy el nuevo párrafo";
//El elemento pPosterior es el segundo párrafo de la capa principal
let pPosterior=document.querySelectorAl1(
                                    "«principal p:nth-of-type(2)")[0];
capa.insertBefore(nuevoP,pPosterior);
```

REEMPLAZAR ELEMENTOS

El método **replaceChild** es más agresivo. Permite cambiar un nodo por otro. Necesita como parámetros: primero el nuevo nodo que se desea colocar y después el nodo que se desea quitar. El nuevo nodo reemplaza el antiguo.

El método devuelve una referencia al nodo retirado, que realmente no se quita del todo, sino que simplemente se retira del árbol del DOM. Podemos usar la referencia al nodo quitado para colocarlo dentro de otro elemento.

Ejemplo:

```
let capa=document.getElementById("principal");
let nuevoP=document.createElement("p");
nuevoP.textContent="Soy el nuevo párrafo";
//El elemento pPosterior es el segundo párrafo de la capa principal
let pPosterior=document.querySelectorAl1(
                                        "«principal p:nth-of-type(2)")[0];
capa.replaceChi Id(nuevoP,pPosterior);
Ha desaparecido el segundo párrafo y en su posición aparece el nuevo
nodo.
```

ELIMINAR ELEMENTOS

El método **removeChild** es un método de los nodos que recibe como único parámetro el nodo a eliminar. Ese nodo no se elimina realmente de la memoria solo se retira del árbol de elementos, pero permanece en memoria. De hecho, el método devuelve una referencia al objeto retirado.

En este ejemplo se quitaría el segundo párrafo y se colocaría al final con ayuda de **appendChild:**

```
let capa=document.getElementById("principal");
let pSegunda=document.querySelectorAl1(
                                        "«principal p:nth-of-type(2)")[0];
//se elimina el segundo párrafo
capa.removeChild(pSegunda);
capa.appendChild(pSegunda);
```