1.3.- Acceso a los nodos element.

Método `getElementsByName() - Ejercicio`

Crea una página HTML con un radio button con 3 opciones de deporte.

Desde Javascript haz que siempre esté seleccionada la última opción.

Desde Javascript recorre los elementos del radiobutton con el método `getElementsByName` y `forEach()` para ver cuál de ellos tiene la propiedad `checked` a `true`. El elemento seleccionado será el que tenga dicha propiedad a `true`.

Ejercicio `getElementsByClassName()`

Ejercicio:

Crea una página HTML con:

- **Una clase de estilos llamada “importante” que ponga color a red**
- **5 elementos div. 2 de ellos tendrán la clase “importante”.**

Desde Javascript calcula el número de elementos con la clase “`importante`”

Modificación del contenido de un element

`element.innerHTML:` para modificar el contenido del elemento (nodo texto hijo), indicando al navegador que el texto lo interprete como código HTML. No modifica el propio elemento HTML

`element.outerHTML:` para modificar el contenido del elemento (nodo texto hijo), indicando al navegador que el texto lo interprete como código HTML, y también modifica el elemento HTML.

`element.innerText:` para modificar el contenido del elemento (nodo texto hijo), indicando al navegador que el texto lo interprete como texto plano.

Modificación del contenido de un nodo element

Ejercicio:

Tenemos una página HTML con 4 párrafos, cada uno de ellos con un id.

- Modifica el primero para que aparezca el texto “Esta clase es estupenda”
- Modifica el segundo párrafo para que sea sustituido por la siguiente lista (comprueba que el elemento párrafo desaparece):
  - **DWEC**
  - **DIW**
- Modifica el tercer párrafo dando el valor “Hola” a la propiedad `innerText`. ¿Qué ocurre?, ¿por qué?
- Modifica el primer párrafo para que aparezca una línea más en dicho párrafo con “**DWEC** es fantástico”
- Modifica el cuarto párrafo para que aparezca lo mismo que en el primero pero sin las negritas utilizando la propiedad `innerText`.

Acceso a los nodos element-continuación

`elem.children:` devuelve una colección con todos los nodos hijos de tipo element de un elemento (`elem`). El orden es el de aparición en la página HTML

En esta página HTML de ejemplo:

```
<div>
<p>Primera línea</p>
<p>Segunda línea</p>
<p>Tercera línea</p>
</div>
var elemDiv=document.getElementsByTagName("div")[0]
elemDiv.children[0]// es el párrafo cuyo contenido es "Primera línea"
elemDiv.children[1]// es el párrafo cuyo contenido es "Segunda línea"
elemDiv.children[elemDiv.children.length-1]// es "Tercera línea"
```

La colección que devuelve este método no permite utilizar el método **`forEach()`** Para ello habría que pasar la colección a Array así:

```
Array.from(elem.children).forEach()
```

`elem.childElementCount:` devuelve el nº de nodos hijos de tipo element de un elemento (`elem`). Equivalente a: **`elem.children.length`**

```
elemDiv.children[elemDiv.childElementCount-1]// es "Tercera línea"
```

Acceso a los nodos element.

`elem.firstElementChild:` devuelve el primer nodo hijo de tipo element de un elemento (`elem`). El orden es el de aparición en la página HTML

En esta página HTML de ejemplo:

```
<div>
<p>Primera línea</p>
<p>Segunda línea</p>
<p>Tercera línea</p>
</div>
var elemDiv= document.getElementsByTagName("div")[0]
elemDiv.firstElementChild // es el de contenido "Primera línea"
//igual elemDiv.children[0]
```

`elem.lastElementChild:` devuelve el último nodo hijo de tipo element de un elemento (`elem`)

```
elemDiv.lastElementChild // es el de contenido "Tercera línea"
// igual a elemDiv.children[elemDiv.childElementCount-1]
```

`elem.parentNode:` es el nodo padre del elemento `elem`.

```
elemDiv.firstElementChild.parentNode // es elemDiv
```

Acceso a los nodos element.

`elem.nextElementSibling`: devuelve el siguiente nodo de tipo element a `elem` que hay en el mismo nivel del árbol DOM. El orden es el de aparición en la pág. HTML

En esta página HTML de ejemplo:

```
<div>
<p>Primera línea</p>
<p>Segunda línea</p>
<p>Tercera línea</p>
</div>
var elemDiv= document.getElementsByTagName("div")[0]

elemDiv.firstElementChild.nextElementSibling()
// es el párrafo de contenido "Segunda línea"
```

`elem.previousElementSibling`: devuelve el nodo de tipo element anterior a `elem` que hay en el mismo nivel del árbol DOM. El orden es el de aparición en la página HTML

```
elemDiv.lastElementChild.previousElementSibling() //el de "Segunda línea"
elemDiv.firstElementChild.previousElementSibling() //es null
```

Modificación del árbol DOM

- **`cloneNode(true)`** **copia un nodo** ya existente, **pero no lo añade automáticamente a ningún nodo. Con true copia también todos sus nodos hijos si los tuviera.**

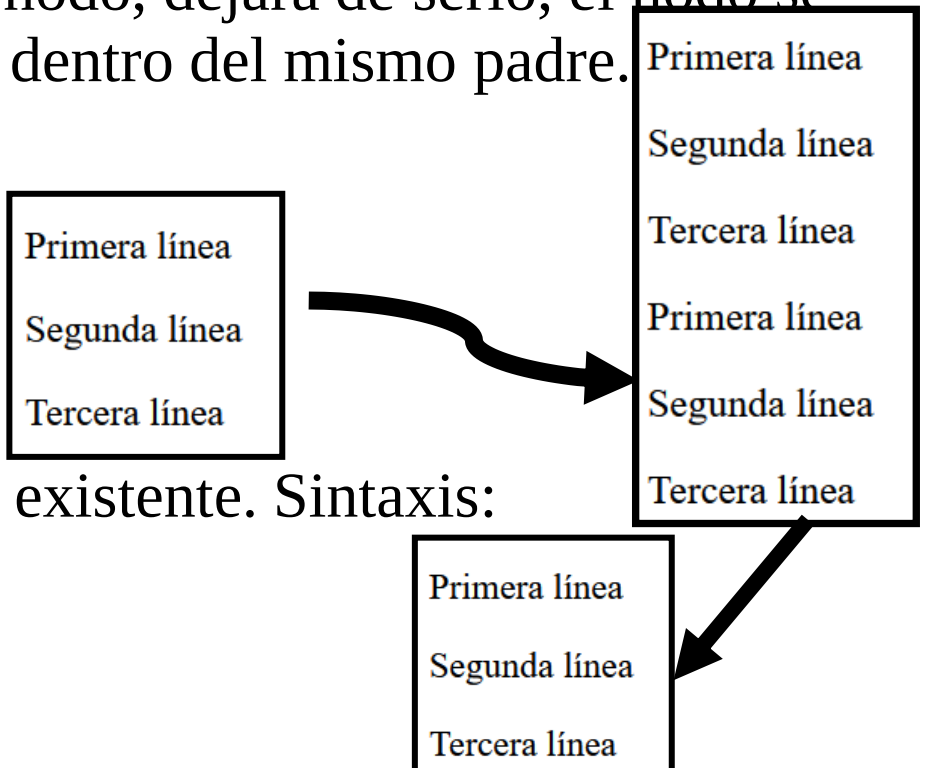
```
var nodoNuevo=nodo.cloneNode(true);
```

Ejercicio: crea una copia del primer párrafo del div.

- **`appendChild()`** Para insertar un hijo después del último nodo hijo. Si el hijo insertado era hijo previamente de otro nodo, dejará de serlo, el nodo se mueve de un padre a otro. Se puede mover dentro del mismo padre.

```
nodoPadre.appendChild(nuevoHijo);
```

Ejercicio: añade al final del body la copia del div.



- **`removeChild()`** para borrar un hijo existente. Sintaxis:

```
nodoPadre.removeChild(nodoHijo).
```

Ejercicio: borra la copia del div.

Nota extra: El nodo seguirá existiendo pero ya no será hijo de `nodoPadre`. Si antes de borrarlo guardamos su referencia podremos verlo, y veremos que su `parentNode` es `null`

Modificación del árbol DOM

- **`insertBefore()`** Para insertar un hijo antes de otro nodo hijo. Sintaxis:

```
nodoPadre.insertBefore(nuevoHijo,refHijo);
```

Va a añadir al nodo padre un hijo delante del nodo hijo `refHijo`

Ejemplo: https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_node_insertbefore

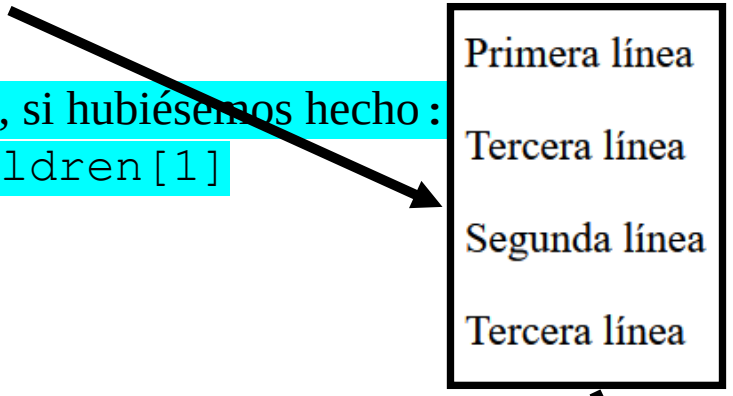
Ejercicio: crea una copia del ultimo párrafo del div e insértalo antes del segundo párrafo del div.

Mover con `insertBefore`: podemos mover un hijo de posición, si hubiésemos hecho : `div.insertBefore(div.lastElementChild,div.children[1]`
Habría quedado:

Primera línea

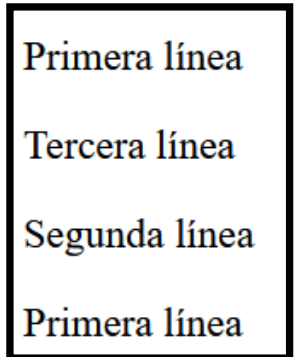
Tercera línea

Segunda línea



- **`replaceChild()`** **reemplaza un nodo por otro,** el nodo sustituido dejará de ser hijo pero seguirá existiendo. Si el nodo que reemplaza era hijo de otro nodo dejará de serlo, y por tanto se mueve. Sintaxis: `nodoPadre.replaceChild(nuevoHijo,antiguoHijo);`

Ejercicio: crea una copia del primer párrafo del div y reemplaza el ultimo párrafo por esta copia.



Nota extra: el nodo reemplazado sigue existiendo en memoria pero ya no es hijo del `div`. Si antes de ser reemplazado guardamos su dirección `let ultimo=`

`div.lastElementChild`, y después lo reemplazásemos, veríamos que `ultimo` sigue existiendo, pero ya no tiene padre (`ultimo.parentNode` es `null`)

1.4.- Acceso a los nodos de tipo atributo.

Ejercicio:

Crea una aplicación web con:

1. **Un input de tipo text con id “nombre”.**
2. **Un botón llamado “Mostrar” que al pulsarlo muestra todos los atributos del input.**
3. **Un botón llamado “Agregar tamaño” que al pulsarlo agrega un atributo llamado “size” y ponle el valor 100**
4. **Un botón “Eliminar tamaño” que al pulsarlo elimina el atributo “size”**

Ejecuta la aplicación, pulsa “Mostrar”, luego “Agregar tamñao”, luego “Mostrar”, después “Eliminar tamaño”, y por último “Mostrar”

Otros métodos y propiedades de los nodos

`nodo.childNodes`: propiedad que es una colección con los nodos hijos (tipo element y text) de `nodo`. `nodo.children` es una colección de nodos hijos tipo `element`.

`nodo.hasChildNodes()`: devuelve `true` si `nodo` tiene nodos hijos (tipo element o text), y `false` si no tiene.

```
<p id="demo"></p>

demo.hasChildNodes() //devuelve false
demo.innerHTML="contenido"  // Queda así: <p id="demo">contenido</p>
demo.hasChildNodes() //devuelve true porque ya tiene un nodo hijo tipo
text
```

`nodo.nodoType`: tiene valor:

- 1 si el nodo es de tipo `Element`
- 2 si el nodo es de tipo `atributo`.
- 3 si el nodo es de tipo `text`.

```
demo.nodeType // es el nodo párrafo, su tipo de nodo vale 1
// es el atributo id del párrafo que es un nodo cuyo tipo vale 2
demo.attributes[0].nodeType
// es el nodo hijo (del párrafo) de tipo texto ("contenido") y el tipo es 3.
demo.childNodes[0].nodeType
```

Ejercicio-Acceso a los nodos element.

Dada esta tabla:

```
 <table style="width:300px">
   <tbody>
   <tr>
     <td><button style="font-size:26px;width:80px">1</button></td>
     <td><button style="font-size:26px;width:80px">2</button></td>
     <td><button style="font-size:26px;width:80px">3</button></td>
   </tr>
   <tr>
     <td><button style="font-size:26px;width:80px" >4</button></td>
     <td><button style="font-size:26px;width:80px" >5</button></td>
     <td><button style="font-size:26px;width:80px">6</button></td>
   </tr>
   <tr>
     <td><button style="font-size:26px;width:80px">7</button></td>
     <td><button style="font-size:26px;width:80px">8</button></td>
     <td><button style="font-size:26px;width:80px">9</button></td>
   </tr>

</tbody>
   </table>
```

Ejercicio-Modificación del árbol DOM.

En la tabla anterior añade los siguientes botones:

```
<input type="button" value="subir" onclick="subir()">
<input type="button" value="bajar" onclick="bajar()">
<input type="button" value="izquierda" onclick="izquierda()">
<input type="button" value="derecha" onclick="derecha()">
<input type="button" value="aleatorio" onclick="aleatorio()">
<input type="button" value="reiniciar" onclick="reiniciar()">
```

Codifica las funciones de la siguiente manera (Importante: No se puede modificar el atributo value de los botones**):**

`subir()`: sube todas las filas de la tabla un nivel más arriba de forma circular, es decir, la primera fila será la última después de subir.

`bajar()`: baja todas las filas de la tabla un nivel más arriba de forma circular

`izquierda()`: desplaza todos los botones una posición a la izquierda de forma circular.

`derecha()`: desplaza todos los botones una posición a la derecha de forma circular.

`aleatorio()`: desordena todos los botones de forma aleatoria. Sugerencia: calcular 2 posiciones de `tds` que se van a intercambiar, para ello clona dichos `tds`.

`reiniciar()`: deja los botones como estaban al comienzo.