# Accediendo a un elemento HTML a través del DOM (getElementById)

El primer método que nos provee el DOM a través del objeto document es:

document.getElementById("nombre del Id del elemento HTML")

Nos retorna una referencia a la etiqueta en sí misma. Con esta referencia podemos acceder a sus propiedades como puede ser su contenido, color, fuente, etc. A este método lo emplearemos constantemente durante el curso.

Para disponer un ejemplo muy sencillo dispondremos un elemento <h1> en la página y luego por medio de dos botones cambiaremos su color, y su tamaño de fuente.

El archivo html es el siguiente:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

<link rel="stylesheet" href="estilos.css">

</head>

<body>

<h1 id="titulo">Este es un título dinámico</h1>

<input type="button" value="Cambiar Color" onClick="cambiarColor()">

<input type="button" value="Cambiar Tamaño de Fuente" onClick="cambiarTamanoFuente()">

<script src="funciones.js"></script>

</body>

</html>

Es importante notar que a los eventos para los dos botones los definimos en un archivo separado llamado funciones.js que lo incluimos con el siguiente código:

<script src="funciones.js"></script>

A la hoja de estilo también la definimos en un archivo separado que la incluimos con el siguiente código:

<link rel="stylesheet" href="estilos.css">

Para poder acceder a la marca h1 debemos definir la propiedad id:

<h1 id="titulo">Este es un título dinámico</h1>

Para capturar los eventos de los dos botones inicializamos la propiedad onclick de cada botón (más adelante veremos que hay otros métodos para acceder a los eventos de los objetos que se definen directamente en el archivo JavaScript sin tener que llamar desde HTML a las funciones):

<input type="button" value="Cambiar Color" onClick="cambiarColor()">

<input type="button" value="Cambiar Tamaño de Fuente"

onClick="cambiarTamanoFuente()">

Por otro lado definimos el archivo estilos.css:

#titulo {

text-align: center;

}

#boton1,

#boton2 {

text-align: center;

width: 200px;

}

y el archivo funciones.js:

function cambiarColor() {

let tit = document.getElementById('titulo')

tit.style.color = '#ff0000'

}

function cambiarTamanoFuente() {

let tit = document.getElementById('titulo')

tit.style.fontSize = '60px'

}

Notemos que para recuperar una referencia a la etiqueta h1 debemos indicar el siguiente código:

let tit = document.getElementById('titulo')

Ahora, en la varible tit tenemos una referencia a la marca h1 de la página.  
Si queremos cambiar el color del texto debemos acceder a la propiedad style y de ésta a la propiedad color:

tit.style.color = '#ff0000'

Es similar para cambiar la fuente del título (accedemos a la propiedad fontSize):

let tit = document.getElementById('titulo')

tit.style.fontSize = '60px'

### Acotaciones

En muchas situaciones podemos implementar un código más conciso sin tener que almacenar en una variable la referencia a la etiqueta:

function cambiarColor() {

document.getElementById('titulo').style.color = '#ff0000'

}

function cambiarTamanoFuente() {

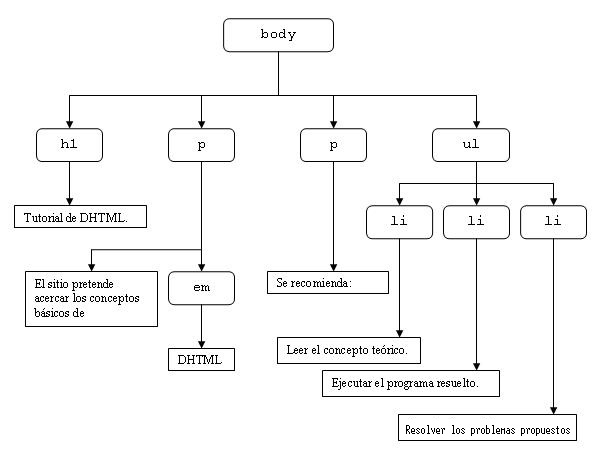
document.getElementById('titulo').style.fontSize = '60px'

}

# Accediendo a los nodos hijos de un elemento HTML

# (childNodes y children)

Cada nodo tiene un solo nodo padre, pero puede tener muchos hijos.

  
  
En este ejemplo el nodo body tiene 4 hijos, la marca h1 tiene 1 hijo, lo mismo cada marca li tienen un hijo. Los nodos que no tienen descendientes son los nodos hoja.

El DOM provee a cada nodo de un vector que almacena la referencia a cada nodo hijo, la propiedad se llama:

childNodes

Este vector almacena una referencia a cada nodo hijo.

Los nodos pueden ser de tipo: ELEMENT\_NODE (nodo elemento) o TEXT\_NODE (nodo texto).

Confeccionar una página que contenga un div con tres párrafos. Luego acceder a la propiedad childNodes y verificar cuantos hijos tiene y de que tipo de nodos se tratan.  
El archivo html es:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<div id="bloque">

<p>Este es el primer párrafo</p>

<p>Este es el segundo párrafo</p>

<p>Este es el tercer párrafo</p>

</div>

<input type="button" value="Cantidad de Hijos del DIV" onClick="cantidadHijos()">

<script src="funciones.js"></script>

</body>

</html>

El archivo javascript funciones.js es el siguiente:

function cantidadHijos() {

let lista = document.getElementById('bloque')

alert('La cantidad de hijos del nodo div es:' + lista.childNodes.length)

let cadena = ""

for (let x = 0; x < lista.childNodes.length; x++) {

if (lista.childNodes[x].nodeType == Node.TEXT\_NODE)

cadena = cadena + "Nodo tipo Texto\n"

if (lista.childNodes[x].nodeType == Node.ELEMENT\_NODE)

cadena = cadena + "Nodo tipo Elemento\n"

}

alert(cadena)

}

Es importante notar que sólo definimos la propiedad id para el elemento div, ya que teniendo la referencia de esta podemos acceder a sus hijos.

Entre la marca <div id="bloque"> y la marca <p> hay un salto de línea (el navegador crea un nodo de texto, este es el primer nodo).  
Luego el primer párrafo es el segundo nodo.  
Entre el primer párrafo y el segundo hay otro salto de línea (este es el tercer nodo) y así sucesivamente. Si contamos los párrafos y saltos de línea podremos contabilizar los 7 hijos.  
Si no dejamos el salto de línea entre una párrafo y otro y tampoco dejamos salto de línea después del div y antes del cerrado del div, los hijos serán 3.

Para obtener la referencia a la marca div hacemos como lo vimos en el concepto anterior:

let lista = document.getElementById('bloque')

Imprimimos la cantidad de nodos hijo del div accediendo a la propiedad length de childNodes:

alert('La cantidad de hijos del nodo div es:' + lista.childNodes.length)

Como ya sabemos la cantidad de nodos que tiene podemos mediante un for recorrer el vector y analizar el tipo de cada nodo:

let cadena = ""

for (let x = 0; x < lista.childNodes.length; x++) {

if (lista.childNodes[x].nodeType == Node.TEXT\_NODE)

cadena = cadena + "Nodo tipo Texto\n"

if (lista.childNodes[x].nodeType == Node.ELEMENT\_NODE)

cadena = cadena + "Nodo tipo Elemento\n"

}

alert(cadena)

### propiedad children

En cambio la propiedad children retorna solo los nodos de tipo elemento. Volvamos a codificar el problema anterior para mostrar la cantidad de nodos de tipo elemento:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<div id="bloque">

<p>Este es el primer párrafo</p>

<p>Este es el segundo párrafo</p>

<p>Este es el tercer párrafo</p>

</div>

<input type="button" value="Cantidad de Hijos tipo elemento del DIV" onClick="cantidadHijosElemento()">

<script src="funciones.js"></script>

</body>

</html>

El archivo JavaScript:

function cantidadHijosElemento() {

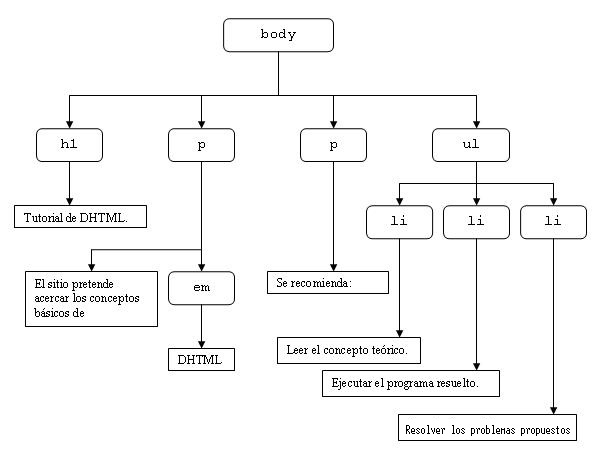
let lista = document.getElementById('bloque')

alert('La cantidad de hijos de tipo elemento del div es:' + lista.children.length)

**Accediendo a un nodo de texto de un elemento HTML**

**(childNodes - nodeValue)**

Es importante notar que el texto contenido en un elemento de HTML no pertenece al nodo, sino que se encuentra en otro nodo especial que se llama nodo texto (TEXT\_NODE)

  
  
En esta árbol de nodos podemos identificar que todos los rectángulos que no tienen bordes redondeados son nodos texto.  
Ahora para acceder a dichos nodos podemos hacerlo a partir del nodo padre con la propiedad childNodes.

El siguiente ejemplo cambia el texto que contiene el elemento "h1" accediendo al nodo de texto.  
El archivo pagina.html es:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<h1 id="titulo">Este es el título original</h1>

<input type="button" value="Cambiar Texto" onClick="cambiarTexto()">

<script src="funciones.js"></script>

</body>

</html>

El archivo funciones.js es:

function cambiarTexto() {

let tit = document.getElementById('titulo')

tit.childNodes[0].nodeValue = 'Ahora vemos el nuevo título'

}

Como ya hemos visto rescatamos la referencia al elemento h1 por medio de la función getElementById, luego mediante la propiedad childNodes accedemos a la primer componente que tiene la referencia al nodo de tipo texto. Por último asignamos el nuevo texto a la propiedad nodeValue.

# Accediendo a un nodo padre (parentNode)

Si tenemos la referencia a un nodo podemos obtener fácilmente la referencia del nodo padre mediante la propiedad parentNode.  
Veamos un ejemplo sencillo donde obtenemos la referencia de un párrafo contenido en un div. Luego por medio de esta referencia obtenemos la referencia al div y cambiamos el color de fondo del div.  
pagina.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<div>

<p id="parrafo1">Este es el primer párrafo</p>

<p>Este es el segundo párrafo</p>

<p>Este es el tercer párrafo</p>

</div>

<input type="button" value="Cambiar color de fondo del div" onClick="cambiarColor()">

<script src="funciones.js"></script>

</body>

</html>

funciones.js

function cambiarColor() {

let puntero = document.getElementById('parrafo1')

let padre = puntero.parentNode

padre.style.backgroundColor = '#ff0000'

}

Obtenemos la referencia del párrafo:

let puntero = document.getElementById('parrafo1')

Ahora accedemos a la propiedad parentNode de la variable inicializada previamente:

let padre = puntero.parentNode

Finalmente modificamos el color del div:

padre.style.backgroundColor = '#ff0000'

### Acotaciones

Es muy común utilizar una sintaxis más concisa sin tener que definir variables locales:

function cambiarColor() {

document.getElementById('parrafo1').parentNode.style.backgroundColor = '#ff0000'

}

# Accediendo a un nodo hermano (nextSibling y previousSibling - nextElementSibling y previousElementSibling)

Si tenemos la referencia de un nodo podemos fácilmente acceder a los otros nodos que se encuentran en la misma altura dentro del árbol que genera el DOM.

Veamos con un ejemplo como obtenemos la referencia de los nodos hermanos, implementaremos una página que contenga tres párrafos y sólo disponemos el id del primero de ellos.

pagina.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<div>

<p id="parrafo1">Este es el primer párrafo</p>

<p>Este es el segundo párrafo</p>

<p>Este es el tercer párrafo</p>

</div>

<input type="button" value="accediendo a los tres párrafos" onClick="proximoParrafo()">

<script src="funciones.js"></script>

</body>

</html>

funciones.js

function proximoParrafo() {

let puntero1 = document.getElementById('parrafo1')

while (puntero1 != null) {

if (puntero1.nodeType == Node.ELEMENT\_NODE)

alert(puntero1.childNodes[0].nodeValue)

puntero1 = puntero1.nextSibling

}

}

La propiedad nextSibling retorna la referencia del nodo hermano que se encuentra inmediatamente más abajo en el archivo HTML, pero a la misma altura si lo pensamos al archivo HTML como un árbol. También existe una propiedad llamado previusSibling que retorna la referencia del nodo hermano que se encuentra inmediatamente más arriba en el archivo HTML.  
nextSibling y previusSibling retornan null en caso de no existir más nodos en dicho nivel.

**Recordar:**Algo muy importante que hay que tener en cuenta es que si hay un salto de línea entre dos párrafos lo interpreta y crea un nodo de texto y la propiedad nextSibling y previusSibling lo accede.

Ahora entonces para poder recorrer la lista de nodos hermanos podemos disponer una estructura repetitiva y avanzar por la lista. Mediante un if verificamos si se trata de un nodo de tipo elemento y procedemos a acceder a su nodo hijo que contiene el valor del texto del párrafo:

let puntero1 = document.getElementById('parrafo1')

while (puntero1 != null) {

if (puntero1.nodeType == Node.ELEMENT\_NODE)

alert(puntero1.childNodes[0].nodeValue)

puntero1 = puntero1.nextSibling

}

### Propiedades nextElementSibling y previusElementSibling

el atributo nextElementoSibling avanza al siguiente nodo de tipo elemento, a diferencia de la propiedad nextSibling que avanza tanto a nodos de tipo elemento como texto.

Luego para recorrer todos los nodos de tipo elemento el algoritmo nos queda:

let puntero1 = document.getElementById('parrafo1')

while (puntero1 != null) {

alert(puntero1.childNodes[0].nodeValue)

puntero1 = puntero1.nextElementSibling

}

No nos hace falta hacer un if dentro del while, debido a que la propiedad nextElementSibling avanza al siguiente nodo de tipo Elemento.

**Otras formas de acceder a los nodos hijos (firstChild y lastChild - firstElementChild y lastElementChild)**

Habiamos visto anteriormente que podemos acceder a todos los hijos de un nodo por medio del vector childNodes o todos los hijos de tipo elemento con children. Otra forma de acceder al primer y último nodo hijo es por medio de las propiedades firstChild (retorna la referencia del primer hijo, es lo mismo que poner obj.childNodes[0]) y obj.lastChild (retorna la referencia del último hijo, es sinónimo de poner obj.childNodes[obj.childNodes.length-1])

Veamos con un ejemplo como utilizar estas dos propiedades, dispondremos tres párrafos dentro de un div. Acceder al primer hijo del div y luego avanzando con el método nextSibling llegar hasta el final de la lista de hermanos, mostrar el texto de cada párrafo.

pagina.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<div id="parrafos">

<p>Este es el primer párrafo</p>

<p>Este es el segundo párrafo</p>

<p>Este es el tercer párrafo</p>

</div>

<input type="button" value="accediendo a los tres párrafos" onClick="proximoParrafo()">

<script src="funciones.js"></script>

</body>

</html>

funciones.js

function proximoParrafo() {

let puntero1 = document.getElementById('parrafos')

let puntero2 = puntero1.firstChild

while (puntero2 != null) {

if (puntero2.nodeType == Node.ELEMENT\_NODE)

alert(puntero2.childNodes[0].nodeValue)

else

alert('No es un nodo tipo elemento')

puntero2 = puntero2.nextSibling

}

}

Primero obtenemos la referencia al div:

let puntero1 = document.getElementById('parrafos')

Seguidamente obtenemos la referencia al primer nodo hijo del div:

let puntero2 = puntero1.firstChild

Mediante una estructura repetitiva avanzamos por cada nodo hermano y aquellos que son de tipo elemento procedemos a obtener la referencia del nodo hijo que es de tipo texto:

while (puntero2 != null) {

if (puntero2.nodeType == Node.ELEMENT\_NODE)

alert(puntero2.childNodes[0].nodeValue)

else

alert('No es un nodo tipo elemento')

puntero2 = puntero2.nextSibling

}

### fistElementChild y lastElementChild

Con el atributo fistElementChild obtenemos la referencia del primer nodo de tipo elemento.

Luego el algoritmo para recorrer todos los nodos de tipo elemento queda:

let puntero1 = document.getElementById('parrafos')

let puntero2 = puntero1.firstElementChild

while (puntero2 != null) {

alert(puntero2.childNodes[0].nodeValue)

puntero2 = puntero2.nextElementSibling

}

**Accediendo a un conjunto de marcas HTML similares (getElementsByTagName)**

Disponemos en el objeto document de un método llamado:

getElementsByTagName("Nombre de la Marca")

Retorna un vector con la referencia a todas las marcas de dicho tipo.  
A diferencia del método getElementById que retorna la referencia de una sola marca, el método getElementsByTag retorna un conjunto de referencias, las mismas se almacenan en un vector.

Vamos a desarrollar una aplicación para agregar un punto al final de cada párrafo.

La página html es:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<p>Este es el primer párrafo</p>

<p>Este es el segundo párrafo</p>

<p>Este es el tercer párrafo</p>

<input type="button" value="Agregar un punto al final de cada párrafo" onClick="cambiarParrafos()">

<script src="funciones.js"></script>

</body>

</html>

Luego el archivo funciones.js es:

function cambiarParrafos() {

let lista = document.getElementsByTagName('p')

for (let f = 0; f < lista.length; f++)

lista[f].childNodes[0].nodeValue = lista[f].childNodes[0].nodeValue + '.'

}

Hay que tener en cuenta que en el vector lista se almacena la referencia a todas las marcas de tipo párrafo (p) de la página. Luego para acceder al contenido de cada párrafo debemos hacerlo mediante el vector childNodes y mediante la propiedad nodeValue acceder al texto contenido en dicho párrafo. Disponemos el subíndice cero en la propiedad childNodes ya que los párrafos definidos en el ejemplo tienen como hijo sólo un texto.

Para agregar un punto al final de cada párrafo sólo debemos almacenar el contenido actual más el string '.':

let lista = document.getElementsByTagName('p')

for (let f = 0; f < lista.length; f++)

lista[f].childNodes[0].nodeValue = lista[f].childNodes[0].nodeValue + '.'

### Acotaciones

Podemos recorrer el vector de elementos con el for of:

function cambiarParrafos() {

let lista = document.getElementsByTagName('p')

for (let elemento of lista)

elemento.childNodes[0].nodeValue = elemento.childNodes[0].nodeValue + '.'

}

# Accediendo a un conjunto de marcas HTML similares a partir de un Nodo (getElementsByTagName)

Podemos obtener la referencia a una marca HTML, y a partir de esta referencia llamar al método getElementsByTagName visto en un concepto anterior:

let obj=document.getElementById("bloque2")

let lista=obj.getElementsByTagName('p')

La cual retorna un vector con todas las marcas contenidas en ese bloque (es decir, no es respecto a todo el documento como el problema anterior)

Esto puede ser de gran utilidad cuando queremos hacer cambios a un conjunto de marcas similares del documento, pero que no afecten a todo el documento.  
Para probar el método, agregaremos un punto al final de cada párrafo que están contenidos en una marca (div).

La página html es:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<div id="bloque1">

<h1>Primer Bloque de párrafos.</h1>

<p>Este es el primer párrafo</p>

<p>Este es el segundo párrafo</p>

<p>Este es el tercer párrafo</p>

</div>

<div id="bloque2">

<h1>Segundo Bloque de párrafos.</h1>

<p>Este es el primer párrafo</p>

<p>Este es el segundo párrafo</p>

<p>Este es el tercer párrafo</p>

</div>

<input type="button" value="Agregar un punto al final de cada párrafo" onClick="cambiarParrafos()">

<script src="funciones.js"></script>

</body>

</html>

Agrupamos los párrafos en dos div

function cambiarParrafos() {

let obj = document.getElementById("bloque2")

let lista = obj.getElementsByTagName('p')

for (let elemento of lista)

elemento.childNodes[0].nodeValue = elemento.childNodes[0].nodeValue + '.'

}

Lo más importante del problema es ver como llamamos al método getElementsByTagName. Si queremos la referencia a todas las marcas de la página lo hacemos :

let lista=document.getElementsByTagName('p')

Si queremos la referencia a todas las marcas similares contenidas en un div lo hacemos:

let obj = document.getElementById("bloque2")

let lista = obj.getElementsByTagName('p')

Es decir, primero obtenemos la referencia del div con id llamado bloque2 y luego, mediante el objeto devuelto, llamamos al método getElementsByTagName.

Nos queda luego recorrer, en este caso con un for of los elementos del array y proceder a modificar el valor almacenado en cada párrafo:

for (let elemento of lista)

elemento.childNodes[0].nodeValue = elemento.childNodes[0].nodeValue + '.'

# Agregar un nodo de texto (appendChild - createTextNode)

Para la creación de un nodo de texto disponemos del siguiente método de la clase document:

let nt=document.createTextNode('Texto del nodo')

Para añadirlo luego a un párrafo por ejemplo debemos llamar al método appendChild:

let nparrafo=document.getElementById('parrafo')

nparrafo.appendChild(nt)

Veamos un ejemplo en el que cada vez que se presione un botón se añada un nodo de texto a un párrafo mostrando cuántos nodos de texto se han añadido:

pagina1.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<p id="parrafo">Texto inicial:</p>

<input type="button" value="Agregar nodo de texto" onClick="agregar()">

<script src="funciones.js"></script>

</body>

</html>

funciones.js

let contador = 1

function agregar() {

let nt = document.createTextNode('Nuevo texto ' + contador + '-')

let nparrafo = document.getElementById('parrafo')

nparrafo.appendChild(nt)

contador++

}

A la variable contador la definimos fuera de la función para que sea global.  
Cada vez que se presiona el botón creamos un nodo de texto agregándole el valor del contador:

let nt = document.createTextNode('Nuevo texto ' + contador + '-')

Seguidamente obtenemos una referencia al párrafo:

let nparrafo = document.getElementById('parrafo')

finalmente añadimos el nodo de texto al párrafo:

nparrafo.appendChild(nt)

# Eliminar un nodo de texto (removeChild) y verificar si tiene nodos hijo (hasChildNodes)

Para eliminar un nodo de texto disponemos de un método llamada:

removeChild

Para comprender el funcionamiento de la misma implementaremos un programa que permita añadir y eliminar nodos de texto de un párrafo.

pagina.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<p id="parrafo">Texto inicial:</p>

<input type="button" value="Agregar nodo de texto" onClick="agregar()">

<input type="button" value="Eliminar nodo de texto" onClick="eliminar()">

<script src="funciones.js"></script>

</body>

</html>

funciones.html

let contador = 1

function agregar() {

let nt = document.createTextNode('Nuevo texto ' + contador + '-')

let nparrafo = document.getElementById('parrafo')

nparrafo.appendChild(nt)

contador++

}

function eliminar() {

let nparrafo = document.getElementById('parrafo')

if (nparrafo.hasChildNodes()) {

nparrafo.removeChild(nparrafo.lastChild)

contador--

}

}

La función agregar ya la explicamos en un concepto anterior al ver la función createTextNode.  
Para eliminar un nodo de texto primero debemos obtener la referencia de una marca html, en este caso el párrafo:

let nparrafo = document.getElementById('parrafo')

Otro método importante que contiene todo nodo de tipo elemento (son los nodos que apuntan a una marca HTML, en este caso de tipo p (párrafo)) es hasChildNodes, la misma retorna true en caso que el nodo tenga nodos hijos (sean nodos de tipo texto o nodos de tipo elemento).  
Es decir que mediante el siguiente if verificamos si el nodo tiene nodos texto aún:

if (nparrafo.hasChildNodes()) {

nparrafo.removeChild(nparrafo.lastChild)

contador--

}

En caso que la condición se verifique verdadera entrará al bloque del if y se ejecutará la llamada a la función removeChild. La misma requiere una referencia al nodo texto que queremos borrar. Y lo último que aparece es la propiedad lastChild que tiene todo nodo de tipo elemento, el mismo almacena la referencia al último nodo hijo que contiene dicha marca.

# Crear y agregar un nodo de tipo elemento (createElement - appendChild)

Para crear nodos de tipo elemento disponemos del método createElement que contiene el objeto document:

let elemento=document.createElement("Aca indicamos el nombre del elemento HTML a crear")

Para ver el funcionamiento confeccionemos un programa que cree un párrafo en forma dinámica y lo añada a un div.  
pagina.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<div id="parrafos">

</div>

<input type="button" value="Agregar nodo tipo elemento" onClick="agregar()">

<script src="funciones.js"></script>

</body>

</html>

funciones.js

function agregar() {

let elemento = document.createElement('p')

let texto = document.createTextNode('Hola Mundo')

elemento.appendChild(texto)

let obj = document.getElementById('parrafos')

obj.appendChild(elemento)

}

En la función agregar se crea un nodo de tipo elemento (tipo párrafo):

let elemento = document.createElement('p')

Pero recordemos que los nodos elemento no contienen el texto, sino que hay que crear un nodo de texto y añadirlo al nodo de tipo elemento:

let texto = document.createTextNode('Hola Mundo')

elemento.appendChild(texto)

Por último obtenemos una referencia al div y añadimos el párrafo que acabamos de crear:

let obj = document.getElementById('parrafos')

obj.appendChild(elemento)

# Agregar un nodo de tipo elemento en medio (insertBefore - createElement)

En caso que necesitemos agregar un nodo entre varios que se encuentran en el mismo nivel, el método appendChild no nos servirá. Para estos casos tenemos el método insertBefore que inserta un nodo previo a otro que le pasamos como referencia en el segundo parámetro.

Supongamos que tenemos cinco párrafos todos contenidos en un div y necesitamos insertar otro párrafo entre el segundo y el tercero. Para resolver este problema debemos obtener la referencia del tercer párrafo y la del div donde se insertará nuestro nuevo párrafo.

pagina.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<div id="parrafos">

<p>Este es el primer párrafo</p>

<p>Este es el segundo párrafo</p>

<p id="p3">Este es el tercer párrafo</p>

<p>Este es el cuarto párrafo</p>

<p>Este es el quinto párrafo</p>

</div>

<input type="button" value="insertar otro párrafo entre el segundo y el tercero" onClick="insertarEnMedio()">

<script src="funciones.js"></script>

</body>

</html>

funciones.js

function insertarEnMedio() {

let elemento = document.createElement('p')

let nodotexto = document.createTextNode('Párrafo entre segundo y tercero')

elemento.appendChild(nodotexto)

let puntero = document.getElementById('p3')

let pa = document.getElementById('parrafos')

pa.insertBefore(elemento, puntero)

}

Lo primero es crear nuestro párrafo y un nodo texto que se lo añadimos al nodo elemento:

let elemento = document.createElement('p')

let nodotexto = document.createTextNode('Párrafo entre segundo y tercero')

elemento.appendChild(nodotexto)

Ahora obtenemos la referencia al tercer párrafo:

let puntero = document.getElementById('p3')

y la referencia al div:

let pa = document.getElementById('parrafos')

Por último llamamos al método insertBefore a partir de la referencia del div y le pasamos como parámetro el nodo a insertar y el nodo previo a donde se debe insertar:

pa.insertBefore(elemento, puntero)

# Eliminar un nodo de tipo elemento (removeChild)

Para eliminar un nodo de tipo elemento disponemos del método removeChild, este método lo llamamos a partir del nodo padre.  
Veamos un ejemplo que permita eliminar el primer o último nodo elemento:  
pagina.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<div id="parrafos"></div>

<input type="button" value="Agregar nodo tipo elemento" onClick="agregar()">

<input type="button" value="Eliminar primer nodo tipo elemento" onClick="eliminarPrimero()">

<input type="button" value="Eliminar último nodo tipo elemento" onClick="eliminarUltimo()">

<script src="funciones.js"></script>

</body>

</html>

funciones.js

let contador = 1

function agregar() {

let nuevoelemento = document.createElement("p")

let nuevotexto = document.createTextNode("Hola Mundo " + contador + " - ")

contador++

nuevoelemento.appendChild(nuevotexto)

let puntero = document.getElementById("parrafos")

puntero.appendChild(nuevoelemento)

}

function eliminarPrimero() {

let puntero = document.getElementById("parrafos")

if (puntero.children.length > 0)

puntero.removeChild(puntero.children[0])

}

function eliminarUltimo() {

let puntero = document.getElementById("parrafos")

if (puntero.children.length > 0)

puntero.removeChild(puntero.children[puntero.children.length - 1])

}

La primera función nos permite crear un nodo de tipo elemento y añadirlo a otro nodo elemento de la página:

function agregar() {

let nuevoelemento = document.createElement("p")

let nuevotexto = document.createTextNode("Hola Mundo " + contador + " - ")

contador++

nuevoelemento.appendChild(nuevotexto)

let puntero = document.getElementById("parrafos")

puntero.appendChild(nuevoelemento)

}

Lo nuevo se encuentra en la función de eliminarPrimero:

function eliminarPrimero() {

let puntero = document.getElementById("parrafos")

if (puntero.children.length > 0)

puntero.removeChild(puntero.children[0])

}

Obtenemos una referencia al nodo elemento llamado "parrafos", si este nodo tiene nodos hijos procedemos a llamar al método removeChild y le pasamos como parámetro la referencia a su primer hijo (puntero.children[0]).

Este control lo hacemos para validar que el nodo tenga nodos hijo.  
Para eliminar el último nodo:

function eliminarUltimo() {

let puntero = document.getElementById("parrafos")

if (puntero.children.length > 0)

puntero.removeChild(puntero.children[puntero.children.length - 1])

}

Enviamos como parámetro al método removeChild la referencia del último hijo al que accedemos por medio del vector children.

# Crear un atributo y agregárselo a un nodo de tipo elemento (setAttribute)

Hay muchas marcas HTML que pueden tener definidos atributos. Muchos de estos son casi obligatorios para su funcionamiento. Imaginemos la marca HTML <a> , si no definimos el atributo href con la dirección del sitio poco sentido tendrá incluirla en la página.

Veamos con un ejemplo como proceder a definir un atributo a un nodo de tipo elemento:  
pagina.html:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Prueba</title>

</head>

<body>

<input type="button" value="Crear nodo tipo elemento y definir el atributo href" onClick="crearElementoyAtributo()">

<div id="direccion">

</div>

<script src="funciones.js"></script>

</body>

</html>

funciones.js:

function crearElementoyAtributo() {

let elemento = document.createElement('a')

let puntero = document.getElementById('direccion')

puntero.appendChild(elemento)

let nodotexto = document.createTextNode('google')

elemento.appendChild(nodotexto)

elemento.setAttribute('href', 'https://www.google.com.ar')

}

Como funciona:  
Creamos un nodo de tipo elemento:

let elemento = document.createElement('a')

Obtenemos una referencia al div llamado direccion:

let puntero = document.getElementById('direccion')

Añadimos el nodo de tipo elemento al div:

puntero.appendChild(elemento)

Creamos un nodo de texto:

let nodotexto = document.createTextNode('google')

Ahora añadimos el nodo de texto al nodo de tipo elemento que acabamos de crear:

elemento.appendChild(nodotexto)

Por último y lo nuevo, es definir el atributo href al ancla que acabamos de crear:

elemento.setAttribute('href', 'https://www.google.com.ar')

El primer parámetro es el nombre de la propiedad (en este caso href) y el segundo es el valor que toma la propiedad.