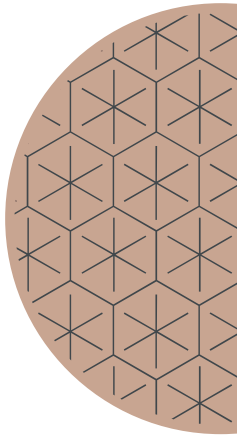
Utilización del Modelo de Objetos del Documento (DOM).

Utilización del Modelo de Objetos del Documento (DOM).

JavaScript es un increíble lenguaje de scripting del lado del cliente que puede conectarse al DOM y manipular el contenido de la página. Este capítulo cubrirá todo lo que necesitas saber sobre el DOM y hacer nuestras páginas web más dinámicas usando el objeto documento.



Árbol y nodos DOM

Árbol y nodos DOM

Como hemos visto anteriormente, mediante Javascript podemos hacer cambios en el DOM que se reflejan en el front-end del sitio web.

Además si refrescas esta página, todo se revertirá como antes de que comenzáramos la manipulación de JavaScript.

Árbol y nodos DOM

Todos los elementos del DOM se conocen como nodos. El DOM representa un documento HTML como una estructura de árbol. Cada uno de los elementos del DOM, como los elementos, atributos, texto, etc., está organizado en una estructura similar a un árbol que se conoce como árbol del DOM. Como en cualquier otro árbol, el árbol DOM tiene ramas, y cada una de estas ramas termina en un nodo.

Árbol y nodos DOM

Tenemos diferentes tipos de nodos en el árbol DOM, pero principalmente necesitamos conocer estos tres:

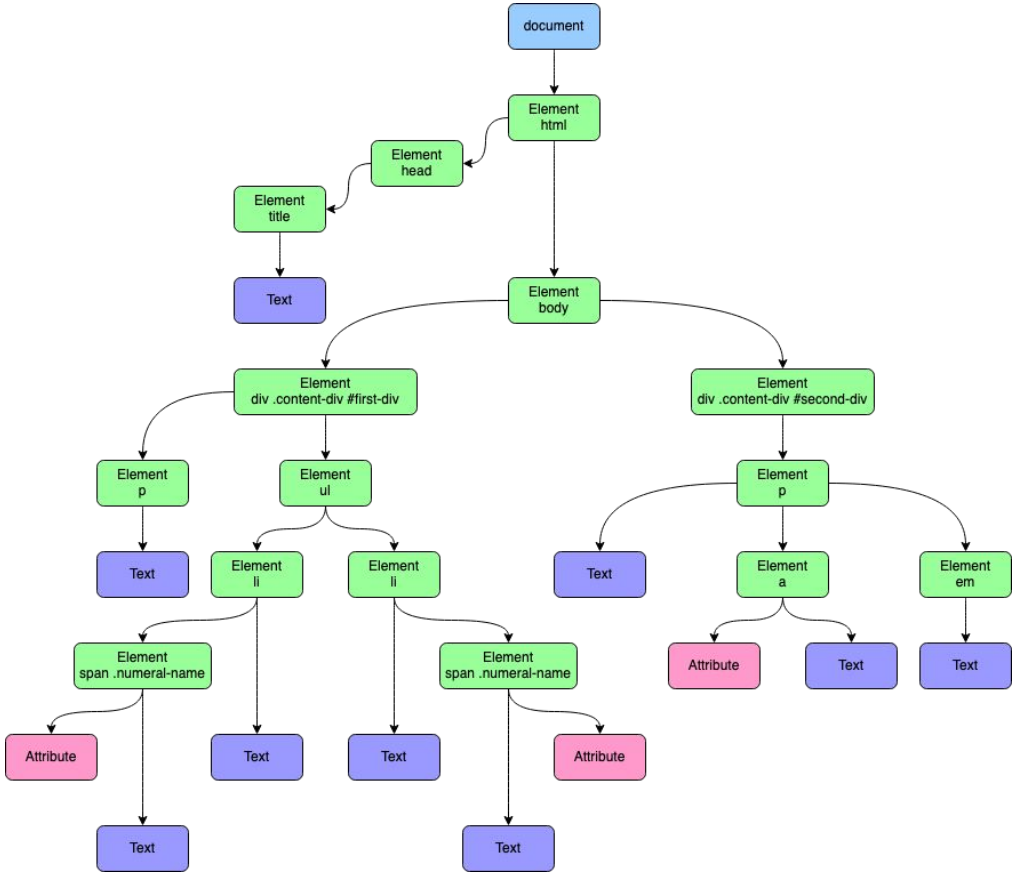
- Nodos de elementos
- Nodos de texto
- Nodos de comentario

Árbol y nodos DOM

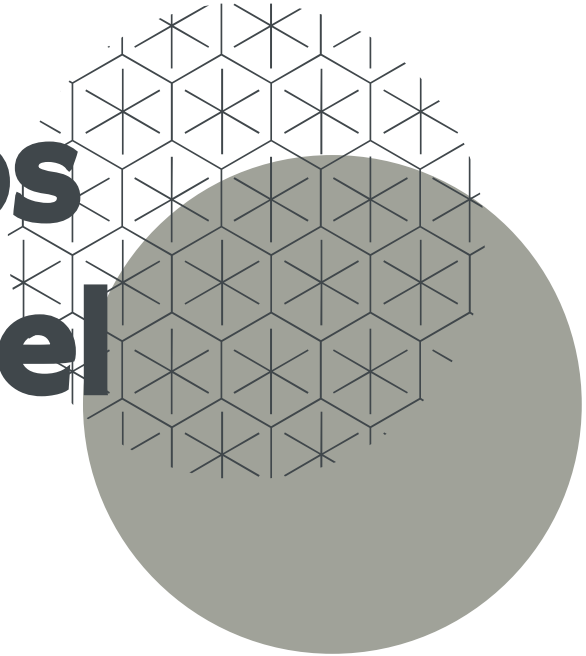
Cada nodo en nuestro DOM tiene un tipo de nodo, y la propiedad nodeType puede ayudarnos a descubrir el tipo de nodo.

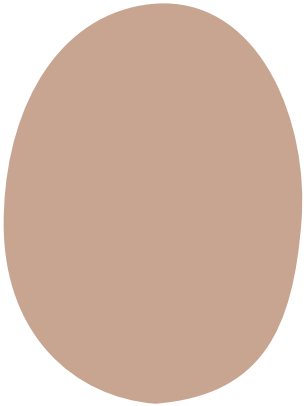
El sitio web MDN tiene una lista completa y actualizada de todos los tipos de nodo válidos:

https://developer.mozilla.org/en-US/docs/Web/API/Node/nodeType


document
Element
html
Element
head
Element
title
Text
Element
body
Element
div .content-div #first-div
Element
div .content-div #second-div
Element
p
Element
ul
Element
p
Text
Element
li
Element
li
Text
Element
a
Element
em
Element
span .numeral-name
Element
span .numeral-name
Attribute
Text
Text
Attribute
Text
Text
Attribute
Text
Text

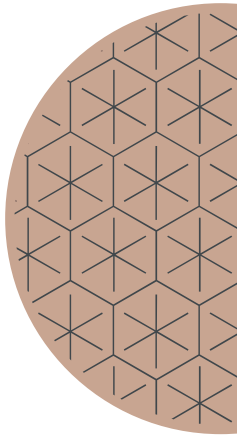
Acceder a los elementos del DOM





Acceder a los elementos del DOM

Hasta ahora, hemos visto cómo podemos utilizar el ID de HTML con el método getElementByID(). Me gustaría señalar que hay algunos métodos más que podemos utilizar para la manipulación del DOM



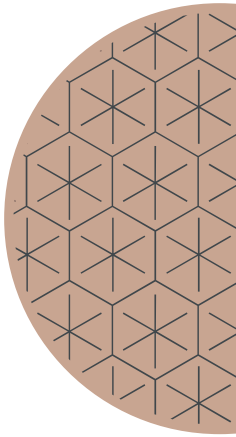
Acceder a los elementos del DOM

Attribute	Selector	Method
ID	#id	getElementById()
Class	.className	getElementByClassName()
Tag	<p>	getElementByTagName()
Single Selector		querySelector()
Select All		querySelectorAll()

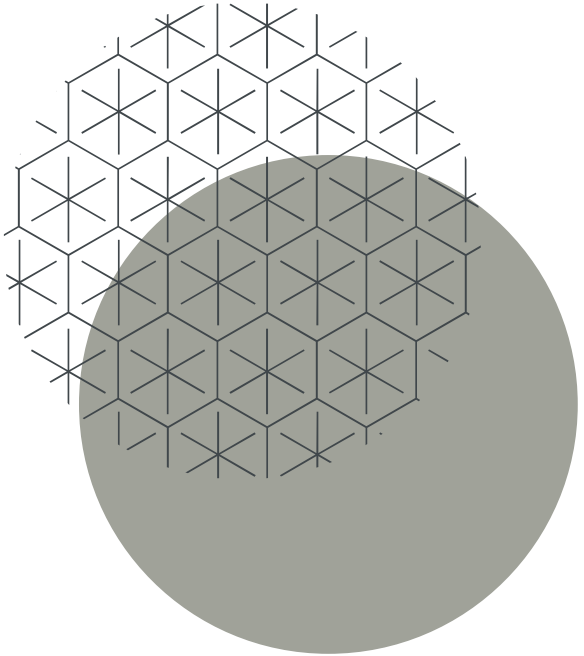
Acceder a los elementos del DOM

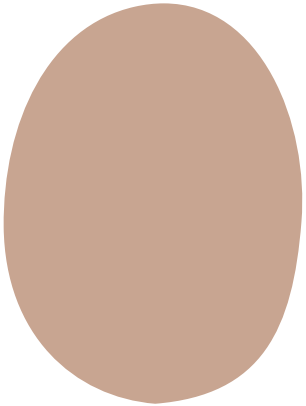
Veamos un ejemplo en el GitHub.

https://github.com/DAW-2022-2023/DWEC



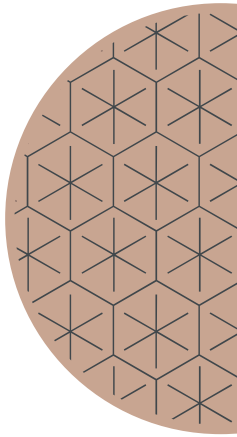
Recorrer el DOM





Recorrer el DOM

Hasta ahora, hemos aprendido sobre los métodos incorporados más útiles que podemos utilizar para acceder a los elementos dentro de nuestro documento DOM. En esta sección, aprenderemos a navegar por el árbol del DOM, un árbol de nodos.



Root Nodes

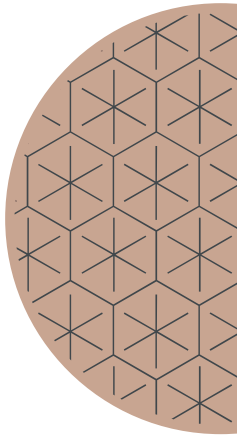
Cuando cargamos un archivo HTML en nuestro navegador, aunque no contenga ningún código, el navegador creará automáticamente tres nodos. Estos nodos serán accesibles y analizados en el documento DOM.

El objeto **document** es el más importante, que es la raíz de todos los demás nodos que tenemos en nuestro DOM. El objeto document es una propiedad del objeto **window**, y este objeto window se denomina objeto global. Con el objeto window, podemos acceder a la altura y anchura de la pantalla y tener acceso a sus métodos como los métodos alert y prompt.

Parent Nodes

Dependiendo de la relación con otros nodos, los nodos del DOM se denominan:

- Padre
- Hijos
- Hermanos



El nodo padre debe estar un nivel por encima del resto de los nodos, y siempre está más cerca del nodo 'documento' en el árbol de nodos del DOM.

Veamos un ejemplo en GitHub.

Parent Nodes

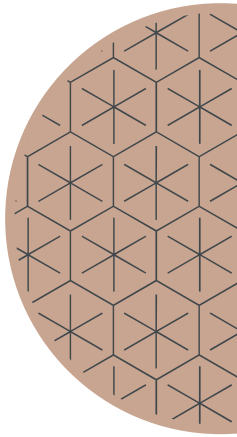
En nuestro ejemplo, el nodo HTML es el padre de los nodos head y body.

Por otro lado, el cuerpo es el padre de h1,h2,p y ul. Verás, aquí no he mencionado el elemento li, pero ¿por qué?

Bueno, el elemento li es el hijo del elemento ul, y éste se encuentra dos niveles por debajo del body, por lo que no es un hijo real. Considere que el elemento li es el nieto del elemento body. x

Parent Nodes

Entonces, ¿cómo podemos comprobar el padre de, por ejemplo, el elemento h1? Podemos utilizar la propiedad **parentNode**, que nos llevará al nodo padre.



Children Nodes

Como vimos en el ejemplo anterior, el cuerpo es el padre de los nodos h1,h2,p y ul. Estos nodos son hijos del nodo body,

Si tenemos nodos de más de un nivel de profundidad, entonces no se denominan hijos sino descendientes

Children Nodes

Property	Returns
childNodes	Nodos hijos
firstChild	El primer nodo hijo
lastChild	El último nodo hijo
firstElementChild	Primer nodo elemento
lastElementChild	Último nodo elemento hijo
children	Todos los nodos hijos elementos

Children Nodes

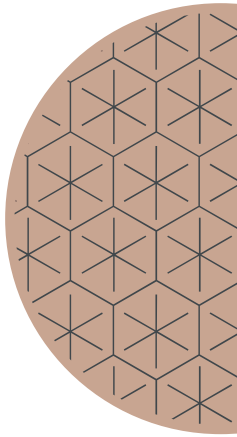
Dependiendo de lo que queremos usaremos un método u otro, veamos el GitHub.

https://github.com/DAW-2022-2023/DWEC

Sibling Properties

Al igual que las propiedades de los hijos, también tenemos las propiedades de los hermanos, y funcionan de forma similar. Cuando se trata de hermanos de un nodo, éste puede ser cualquier nodo del mismo nivel en el árbol DOM.

El nodo hermano no tiene que ser del mismo tipo de nodo. Puede ser un comentario, un texto o un elemento siempre que esté en el mismo nivel de la jerarquía DOM.

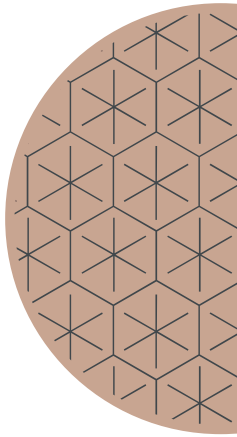


Sibling Properties

Property	Returns
nextSibling	Next Sibling Node
previousSibling	Previous Sibling Node
nextElementSibling	Next Sibling Element Node
previousElementSibling	Previous Sibling Element Node

Sibling Properties

Las dos primeras propiedades de la tabla anterior son para recorrer los nodos solamente, y las dos últimas propiedades son para recorrer los nodos de los elementos.



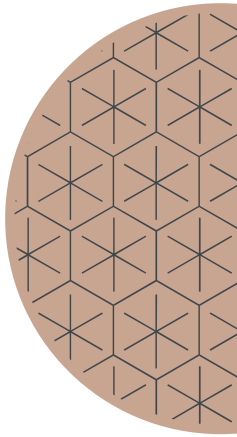
Dirección recorrer DOM

Hemos cubierto cómo podemos atravesar el DOM, pero no hemos mencionado nada sobre las direcciones de los desplazamientos. Podemos recorrer el DOM en estas tres direcciones:

- Hacia abajo
- Lateralmente
- Hacia arriba

Dirección hacia abajo

Al recorrer el documento hacia abajo desde un elemento específico, podemos utilizar uno de los dos selectores CSS, el querySelector o el querySelectorAll. También podemos utilizar los hijos, que nos darán la opción de seleccionar los descendientes directos. Como sabes, los hijos producirán una colección HTML en la que podemos utilizar los índices de los elementos para seleccionar el elemento que queramos.



Veamos un ejemplo en el GitHub.

Dirección hacia arriba

Podemos recorrer el DOM hacia arriba utilizando los dos métodos

- parentElement
- closest

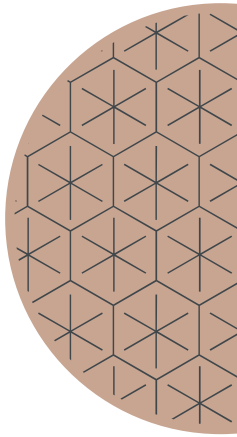
parentElement lo conocemos, el método closest empezará a buscar desde el elemento actual, y luego irá hacia arriba hasta llegar al objeto documento, entonces se detendrá y devolverá el primer elemento que encuentre.

Veamos un ejemplo en el GitHub.

Dirección a los lados

Podemos recorrer el DOM lateralmente utilizando uno de estos métodos:

- previosElementSibling
- nextElementSibling
- Combinación de (parentElement, children, and index)



Creación, inserción y eliminación de nodos del DOM

Hasta ahora, sabemos cómo obtener elementos específicos de nuestro documento DOM. También hemos aprendido cómo podemos recorrer el DOM utilizando diferentes propiedades.

Ahora sabemos lo importante que son los ids, las etiquetas, los selectores y las clases porque podemos localizar fácilmente cualquier nodo que queramos. También podemos utilizar diferentes propiedades para encontrar los nodos anteriores o siguientes en el mismo documento..

Creación, inserción y eliminación de nodos del DOM

Pero hay más cosas que podemos hacer con nuestro documento DOM. También podemos añadir, eliminar, reemplazar o editar nodos.

Crear nuevos nodos

Lo vimos en el tema 5, modelo de objetos predefinidos. Había tres manera posibles:

- Document.createElement()
- InnerHTML
- insertAdjacentHTML

Insertar los nodos creados en el DOM

Si solo creas un elemento no verás ningún cambio porque nunca lo hemos insertado en nuestro DOM. Para ver estos nodos recién creados, tenemos que utilizar uno de estos tres métodos:

- node.appendChild()
- node.insertBefore()
- node.replaceChild()

Insertar los nodos creados en el DOM

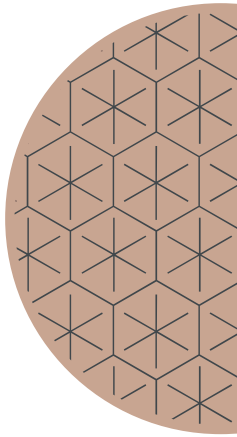
El método appendChild() añadirá el nodo como último hijo del elemento padre.

El método 'insert before' inserta el nodo creado antes del nodo hermano especificado.

El último método, 'replaceChild', sustituirá el nodo existente por uno nuevo.

Modificar clases y atributos del DOM

En JavaScript, tenemos dos propiedades que pueden trabajar con los atributos de clase. Se llaman las propiedades className y classList.



Modificar clases y atributos del DOM

Property	Description
className	Gets the class name/value
classList.add()	Adds one or more class values/names
classList.remove()	Remove class name/value
classList.toggle()	Toggles a class name on or off
classList.contains()	Check if a class name/value exists
classList.replace()	Replace class name with new class name

Modificación de atributos

Los atributos son los pares nombre/valor que contienen información adicional sobre nuestros elementos HTML. Por ejemplo, hemos visto atributos comunes de elementos HTML como classes, ids, styles, src, href, etc.

la lista completa de los atributos HTML:
https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes

La pregunta es, ¿cómo podemos modificar los valores de los atributos en JavaScript?

Modificar atributos

Property	Description
hasAttribute()	Returns true or false
getAttribute()	Returns the value of the attribute or null
setAttribute()	Add or updates new attribute value
removeAttribute()	Remove an attribute from the element

Eliminar elementos DOM

Para eliminar un nodo del DOM solo tienes que llamar a la función .remove()

```
<div id="div-01">Here is div-01</div>
<div id="div-02">Here is div-02</div>
<div id="div-03">Here is div-03</div>

const element = document.getElementById('div-02');
element.remove();
```

FIN