



<codoc  
codoc/>



Maurisandev



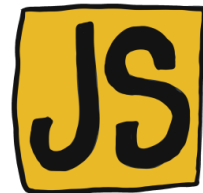
@MauriDeveloper



mau.sanchez@bue.edu.ar

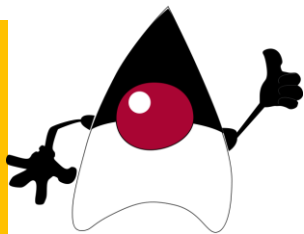
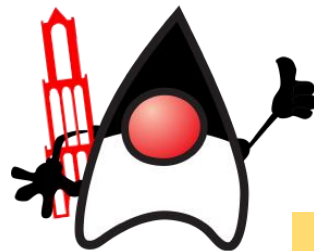


# Curso Java FullStack



Codo a Codo  
4.0

Clase-18  
JavaScript





# FUNCIONES



Cuando se desarrolla una aplicación o sitio web, es muy habitual utilizar una y otra vez las mismas instrucciones.

En programación, una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta, que luego se pueden reutilizar a lo largo de diferentes instancias del código.

```

//! FUNCION CONOCIDA
prompt("INGRESAR SUS DATOS");

//! FUNCION PARA VER EN CONSOLA
console.log("VAMOS A VER POR CONSOLA");

//? OTRA FUNCION CONOCIDA
alert("SALIDA DE DATOS POR ALERT");

//? FUNCION CREADA POR EL DEV
mi_funcion();
```





# *JavaScript - Funciones*

Las funciones son uno de los pilares fundamentales en JavaScript. Una función es un procedimiento en JavaScript—un conjunto de sentencias que realizan una tarea o calculan un valor. Para usar una función, debe definirla en algún lugar del ámbito desde el cual desea llamarla.

```
// Declaración de Función ( Function Declaration )  
function calcularEdad(anioNacimiento) {  
    return 2020 - anioNacimiento  
}
```

# JavaScript – Funciones Declaración

La definición de una función consiste de la palabra clave (reservada) function, seguida por:

El nombre de la función (opcional).

Una lista de argumentos para la función, encerrados entre paréntesis y separados por comas (,).

Las sentencias JavaScript que definen la función, encerradas por llaves, { }.

A continuación podemos observar la declaración de la función calcularEdad, la cual recibe un parámetro.

```
// Declaración de Función ( Function Declaration )  
function calcularEdad(anioNacimiento) {  
    return 2020 - anioNacimiento  
}
```

```
// Expresión de Función (Function Expression)  
const calcularEdad = function() {  
    return 2020 - anioNacimiento;  
}
```





<codigo  
codigo/>



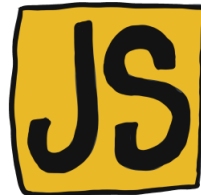
# *¿y qué ventajas me da?*

Las principales ventajas del uso de funciones son:

Ahorro de código.

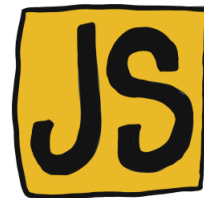
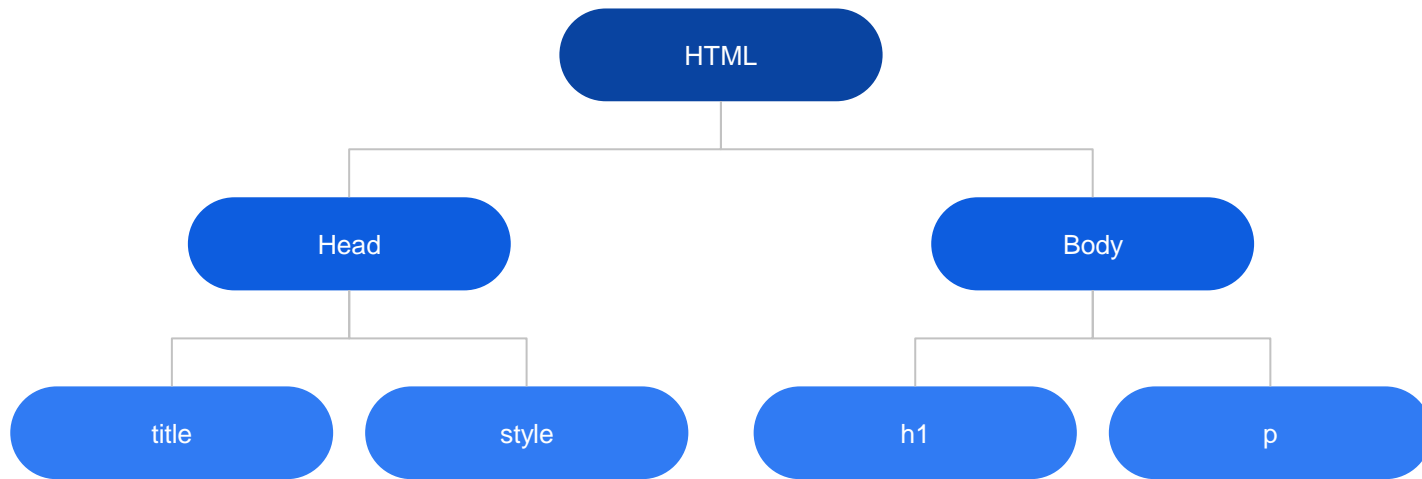
Aporta ordenamiento y entendimiento al código.

Aporta facilidad y rapidez para hacer modificaciones.





# *Arbol del DOM*

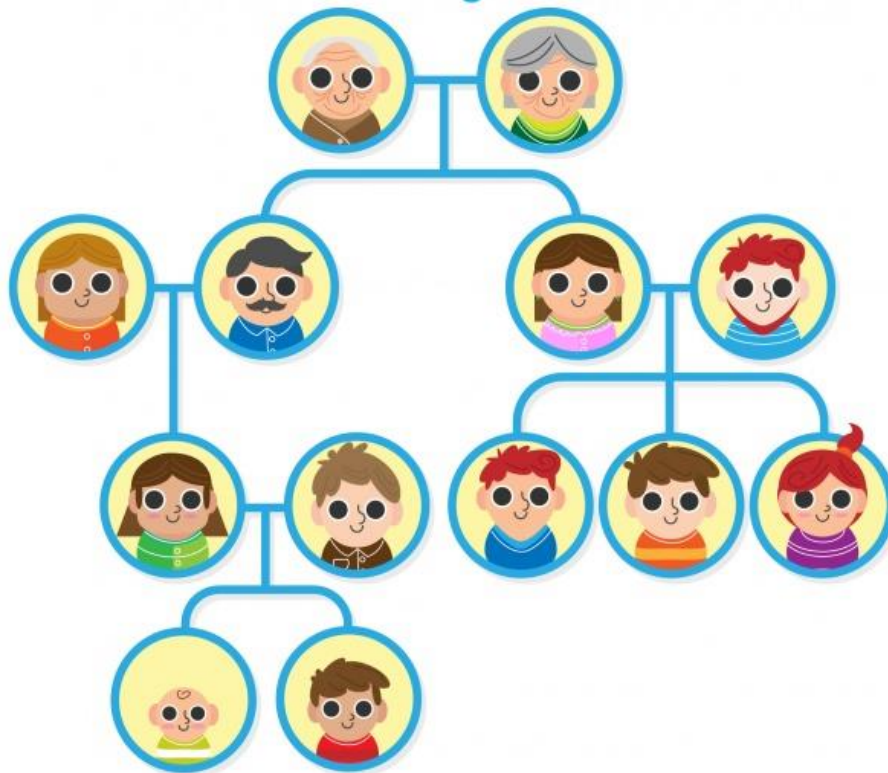




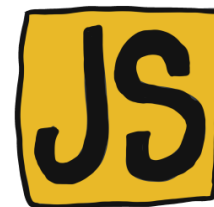
<code>  
code/>



# Family Tree



designed by  freepik.com





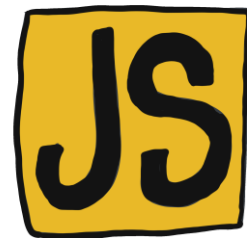
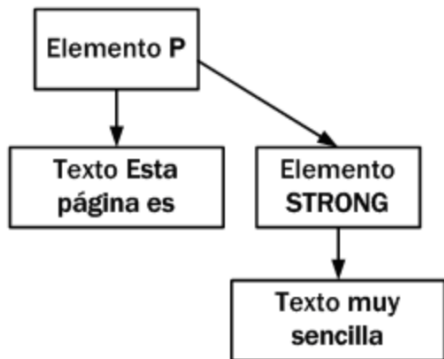
<codoa  
codo/>

# CONCEPTO DE DOM



```
<p>Esta página es <strong>muy sencilla</strong></p>
```

La etiqueta `<p>` se transforma en los siguientes nodos del DOM:







# Manipulación del DOM

## ¿Qué es el DOM?

Las siglas DOM significan Document Object Model, o lo que es lo mismo, la estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol DOM (o simplemente DOM).

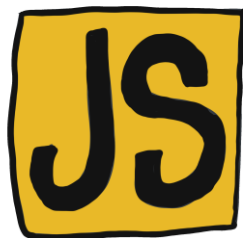
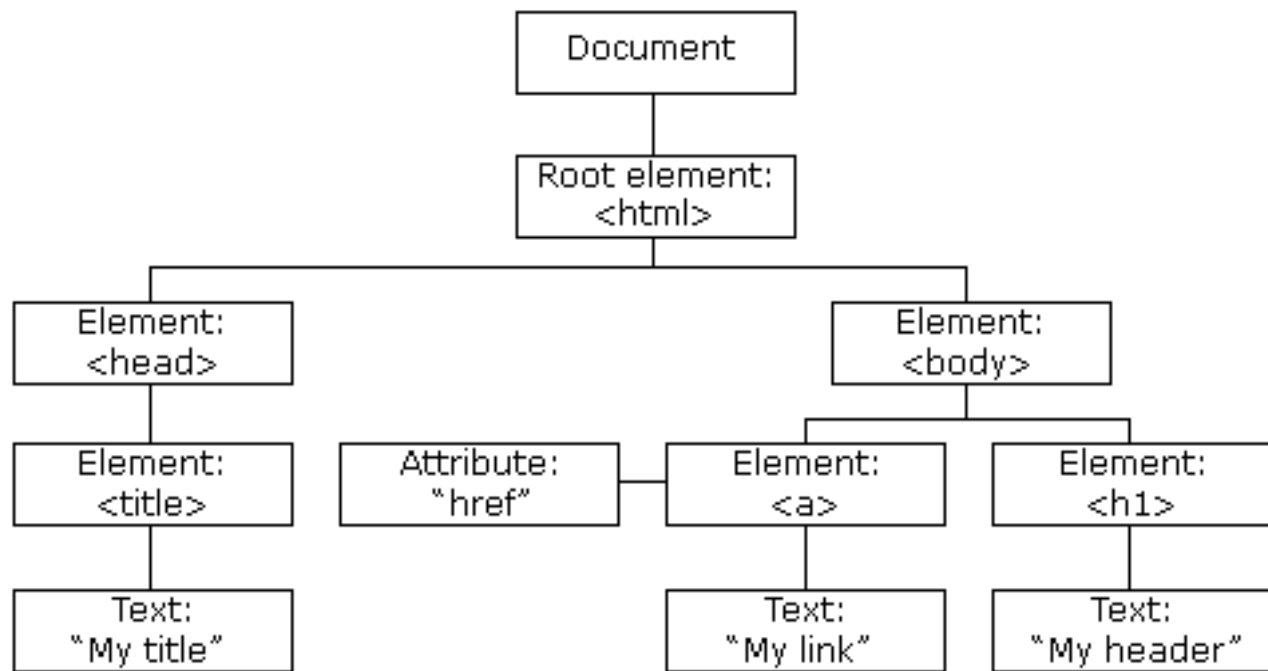
En Javascript, cuando nos referimos al DOM nos referimos a esta estructura, que podemos modificar de forma dinámica desde Javascript, añadiendo nuevas etiquetas, modificando o eliminando otras, cambiando sus atributos HTML, añadiendo clases, cambiando el contenido de texto, etc...





<codoo  
codoo/>

# Manipulación del DOM



# SELECTORES

Se utilizan para identificar y seleccionar uno o más elementos del DOM.



## SELECTORES

#emailHelp

div

.form-group

Id

Elemento

Clase

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1">
    <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="form-group">
    <label for="exampleSelect1">Example select</label>
    <select class="form-control" id="exampleSelect1">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      <option>4</option>
      <option>5</option>
    </select>
  </div>
</form>
```





<codoo  
codoo/>

# EVENTOS EN EL DOM

```
<!DOCTYPE html>
<html>

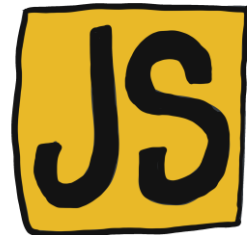
<body>
  <p>Ingresa tu nombre:</p>
  <input type="text" id="nombre">

  <script>
    document.getElementById("nombre").onchange = mostrarAlert;

    function mostrarAlert() {
      alert('Escribiste algo');
    }
  </script>

</body>
</html>
```

Los eventos son la manera que tenemos en Javascript de controlar las acciones de los usuarios y definir un comportamiento de la página o aplicación cuando se produzcan.





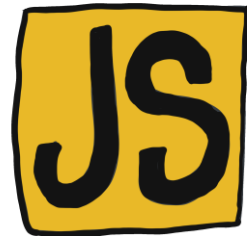
# *Manipulación del DOM*

## **El objeto document**

**En Javascript, la forma de acceder al DOM es a través de un objeto llamado document, que representa el árbol DOM de la página o pestaña del navegador donde nos encontramos. En su interior pueden existir varios tipos de elementos, pero principalmente serán Element o Node:**

**Element no es más que la representación genérica de una etiqueta: HTML.**






**Node es una unidad más básica, la cuál puede ser Element o un nodo de texto.**





# Manipulación del DOM

- **API nativa de Javascript**
- Javascript nos proporciona un conjunto de herramientas para trabajar de forma nativa con el DOM de la página, entre las que se encuentran:

Capítulo del DOM	Descripción
 Buscar etiquetas	Familia de métodos entre los que se encuentran funciones como <code>.getElementById()</code> , <code>.querySelector()</code> o <code>.querySelectorAll()</code> , entre otras.
 Crear etiquetas	Una serie de métodos y consejos para crear elementos en la página y trabajar con ellos de forma dinámica.
 Insertar etiquetas	Las mejores formas de añadir elementos al DOM, ya sea utilizando propiedades como <code>.innerHTML</code> o método como <code>.appendChild()</code> , <code>.insertAdjacentHTML()</code> , entre otros.
 Gestión de clases CSS	Consejos para la utilización de la API <code>.classList</code> de Javascript que nos permite manipular clases CSS desde JS, de modo que podamos añadir, modificar, eliminar clases de CSS de un elemento de una forma práctica y cómoda.
 Navegar entre elementos	Utilización de una serie de métodos y propiedades que nos permiten «navegar» a través de la jerarquía del DOM, ciñéndonos a la estructura del documento y la posición de los elementos en la misma.

# Manipulación del DOM

- **Métodos tradicionales**

- Existen varios métodos, los más clásicos y tradicionales para realizar búsquedas de elementos en el documento. Observa que si lo que buscas es un elemento específico, lo mejor sería utilizar `getElementById()`, en caso contrario, si utilizamos uno de los 3 siguientes métodos, nos devolverá un array donde tendremos que elegir el elemento en cuestión posteriormente:

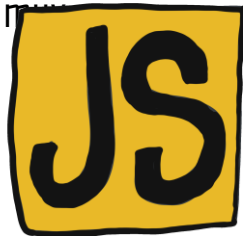
Métodos de búsqueda	Descripción
Element <code>.getElementById(id)</code>	Busca el elemento HTML con el id id. Si no, devuelve NULL .
Array <code>.getElementsByClassName(class)</code>	Busca elementos con la clase class. Si no, devuelve [].
Array <code>.getElementsByName(name)</code>	Busca elementos con atributo name, name. Si no, devuelve [].
Array <code>.getElementsByTagName(tag)</code>	Busca elementos tag. Si no encuentra ninguno, devuelve [].



<codigo  
codigo/>

# Manipulación del DOM

- **Métodos modernos**
- Aunque podemos utilizar los métodos tradicionales que acabamos de ver, actualmente tenemos a nuestra disposición dos nuevos métodos de búsqueda de elementos que son mucho más cómodos y prácticos si conocemos y dominamos los selectores CSS. Es el caso de los métodos `.querySelector()` y `.querySelectorAll()`:
- Con estos dos métodos podemos realizar todo lo que hacíamos con los métodos tradicionales mencionados anteriormente e incluso muchas más cosas (en menos código), ya que son más flexibles y potentes gracias a los selectores CSS.







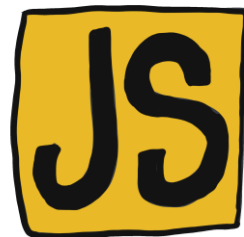
# Manipulación del DOM

Método de búsqueda	Descripción
Element .querySelector(sel)	Busca el primer elemento que coincide con el selector CSS sel. Si no, NULL .
Array .querySelectorAll(sel)	Busca todos los elementos que coinciden con el selector CSS sel. Si no, [].

Ejemplos:

```
/ <div id="page"></div> selecciona el elemento con id page
const page = document.querySelector("#page");

/ <div class="info"></div> Selecciona el primer elemento
/con clase info que se encuentre
/dentro de otro elemento con clase main
const info = document.querySelector(".main .info");
```





# Manipulación del DOM

- **Crear elementos HTML**
- Existen una serie de métodos para crear de forma eficiente diferentes elementos HTML o nodos, y que nos pueden convertir en una tarea muy sencilla el crear estructuras dinámicas, mediante bucles o estructuras definidas:

Métodos	Descripción
Element .createElement(tag, options)	Crea y devuelve el elemento HTML definido por el String tag.
Nodo .createComment(text)	Crea y devuelve un nodo de comentarios HTML <!-- text -->.
Node .createTextNode(text)	Crea y devuelve un nodo HTML con el texto text.
Node .cloneNode(deep)	Clona el nodo HTML y devuelve una copia. deep es false por defecto.
Boolean .isConnected	Indica si el nodo HTML está insertado en el documento HTML.





# Manipulación del DOM

## Atributos HTML de un elemento

- Hasta ahora, hemos visto cómo crear elementos HTML con Javascript, pero no hemos visto cómo modificar los atributos HTML de dichas etiquetas creadas. En general, una vez tenemos un elemento sobre el que vamos a crear algunos atributos, lo más sencillo es asignarle valores como propiedades de objetos:

```
const div = document.createElement("div"); // <div></div>
div.id = "page"; // <div id="page"></div>
div.className = "data"; // <div id="page" class="data"></div>
div.style = "color: red"; // <div id="page" class="data" style="color: red"></div>
```



# Manipulación del DOM

## Reemplazar contenido

Podemos reemplazar el contenido de una etiqueta HTML.

Las propiedades son las siguientes:

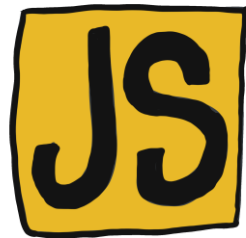
Propiedades	Descripción
.textContent	Devuelve el contenido de texto del elemento. Se puede asignar para modificar.
.innerHTML	Devuelve el contenido HTML del elemento. Se puede usar asignar para modificar.

```
/La propiedad textContent: nos devuelve el contenido de texto de un elemento HTML.  
/Es útil para obtener (o modificar) sólo el texto dentro de un  
/elemento, obviando el etiquetado HTML:  
  
const div = document.querySelector("div"); // <div></div>  
div.textContent = "Hola a todos"; // <div>Hola a todos</div>  
div.textContent; // "Hola a todos"  
  
/La propiedad innerHTML: nos permite hacer lo mismo, pero interpretando el código HTML indicado y  
renderizando sus elementos:  
  
const div = document.querySelector(".info"); // <div class="info"></div>  
div.innerHTML = "<strong>Importante</strong>"; // Interpreta el HTML  
div.innerHTML; // "<strong>Importante</strong>"  
div.textContent; // "Importante"
```



<codoo  
codoo/>

# Manipulación del DOM



```
//Insertar elementos  
const img = document.createElement("img");  
img.src =  
"https://lenguajejs.com/assets/logo.svg";  
img.alt = "Logo Javascript";  
  
document.body.appendChild(img);
```

En este ejemplo podemos ver como creamos un elemento `<img>` que aún no está conectado al DOM. Posteriormente, añadimos los atributos `src` y `alt`, obligatorios en una etiqueta de imagen. Por último, conectamos al DOM el elemento, utilizando el método `.appendChild()` sobre `document.body` que no es más que una referencia a la etiqueta `<body>` del documento HTML.

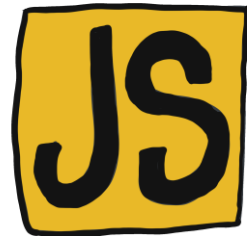


# Manipulación del DOM

## El método remove()

Probablemente, la forma más sencilla de eliminar nodos o elementos HTML es utilizando el método `.remove()` sobre el nodo o etiqueta a eliminar:

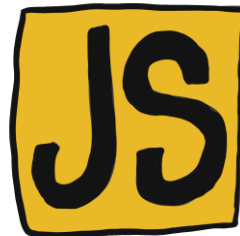
```
//Remove()
const div =
document.querySelector(".deleteme");
div.isConnected;    // true
div.remove();
div.isConnected;    // false
```

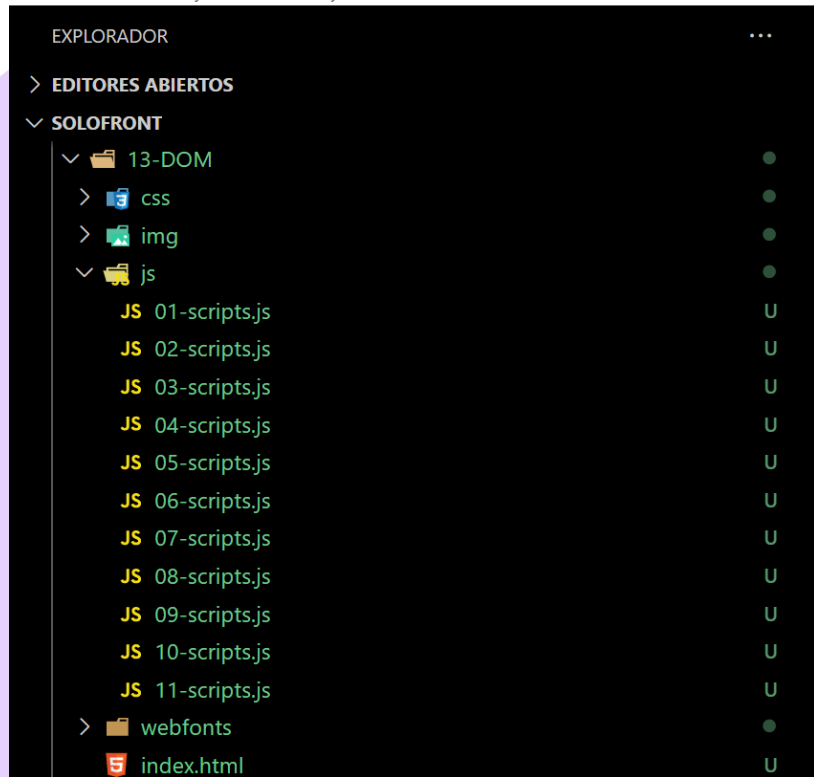




# LINKS COMPLEMENTARIOS

- [https://www.w3schools.com/js/js\\_htmlDOM\\_methods.asp](https://www.w3schools.com/js/js_htmlDOM_methods.asp)
- [https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side\\_web\\_APIs/Manipulating\\_documents](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents)
- <https://www.hongkiat.com/blog/dom-manipulation-javascript-methods/>
- <https://dev.to/giorgosilias/manipulating-dom-in-javascript-for-beginners-4kac>
- <https://dev.to/desoga/7-javascript-methods-that-aids-dom-manipulation-kkj>
- <https://www.freecodecamp.org/news/dom-manipulation-in-vanilla-js-2036a568dcd9/>
- <https://www.tutorialrepublic.com/javascript-tutorial/javascript-dom-manipulation.php>





# Challenge

Apliquemos la teoría y  
Repasemos todos los  
ejercicios del  
Repositorio Git!







Maurisandev



@MauriDeveloper



mau.sanchez@bue.edu.ar



# MUCHAS GRACIAS!



\*\*\*\*\*

**NOS VEMOS EL  
martes!!**

