

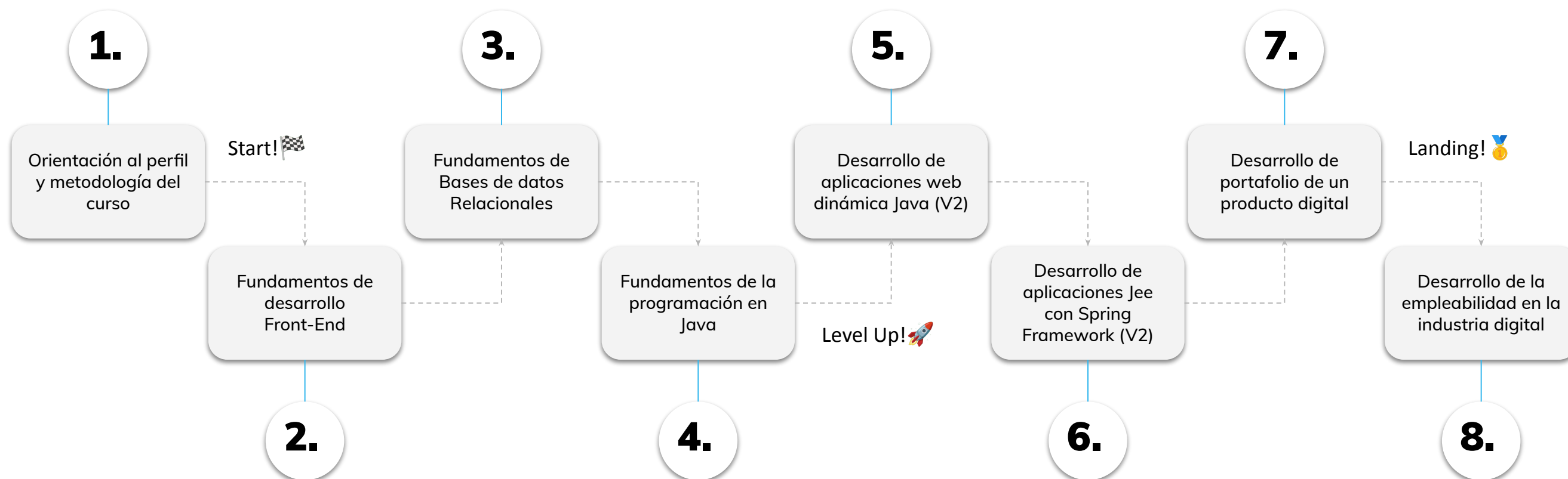
# ➤ Bases del lenguaje Javascript

---

**Plan formativo:** Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

# HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



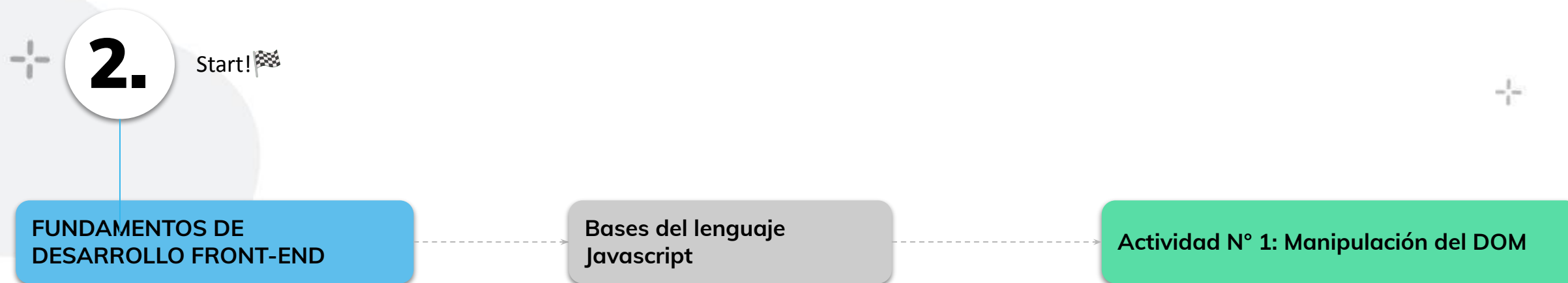
# REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

- ✓ Comprendes el concepto de funciones en JavaScript
- ✓ Declarar funciones, incluyendo la especificación de parámetros y el uso de return para devolver valores.
- ✓ Utilizar la función flecha y la función anónima
- ✓ Comprender cómo debugger pausa la ejecución en un punto específico

# LEARNING PATHWAY

¿Sobre qué temas trabajaremos?



En esta lección, exploraremos el concepto de selectores en JavaScript y su importancia para acceder a elementos específicos del DOM. Familiarizarse con el selector `getElementById` como una forma de seleccionar un elemento por su identificador único. Adquirir habilidades para modificar dinámicamente el contenido de un elemento del DOM cambiando su texto o valor mediante JavaScript.



# OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



**Comprender la manipulación del DOM con JavaScript**



**Familiarizarse con el selector `getElementById` como una forma de seleccionar un elemento por su identificador único.**



**Aprender cómo obtener el contenido de texto de un elemento del DOM utilizando propiedades como `textContent` o `innerText`.**





# Rompehielo 🧊



Si 3 gatos cazan 3 ratones  
en 3 minutos.

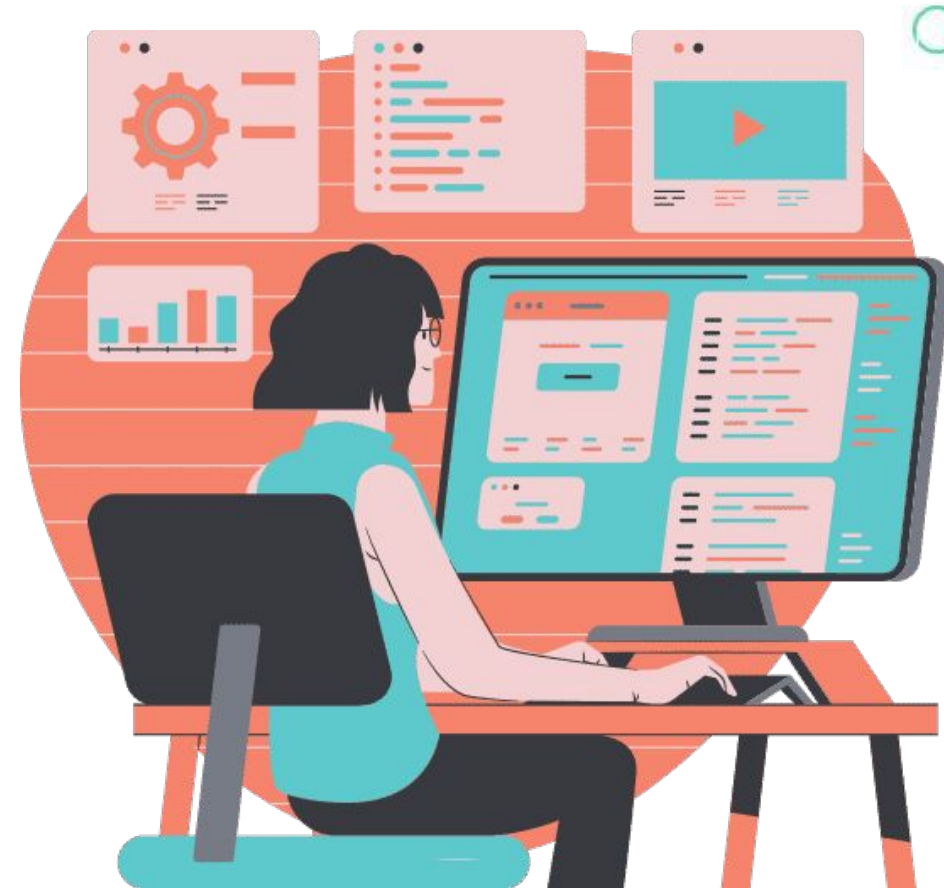
¿cuántos gatos  
cazarán 10 ratones  
en 10 minutos?



# › Selectores básicos: getElementById

# Selectores básicos: getElementById

El Modelo de Objetos del Documento (DOM) es una representación en forma de árbol de la estructura de un documento HTML o XML. Proporciona una interfaz para acceder y manipular los elementos del documento, permitiendo a los programadores interactuar con el contenido, la estructura y los estilos de una página web.







# Selectores básicos: getElementById

El acceso al DOM se realiza a través del objeto document, que es parte del entorno JavaScript en el navegador. El objeto document representa el documento HTML cargado y proporciona métodos y propiedades para interactuar con él.

A partir de ahí, podemos utilizar métodos como getElementById(), querySelector(), createElement(), entre otros, para seleccionar, crear y manipular elementos dentro del DOM.





# Selectores básicos: getElementById

## Estructura del DOM y Nodos

- Cada etiqueta HTML se transforma en un nodo de tipo "Elemento". La conversión se realiza de forma jerárquica.
- De esta forma, del nodo raíz solamente pueden derivar los nodos HEAD y BODY.
- Cada etiqueta HTML se transforma en un nodo que deriva del correspondiente a su "etiqueta padre".
- La transformación de las etiquetas HTML habituales genera dos nodos
  - Nodo elemento: correspondiente a la propia etiqueta HTML.
  - Nodo texto: contiene el texto encerrado por esa etiqueta HTML.



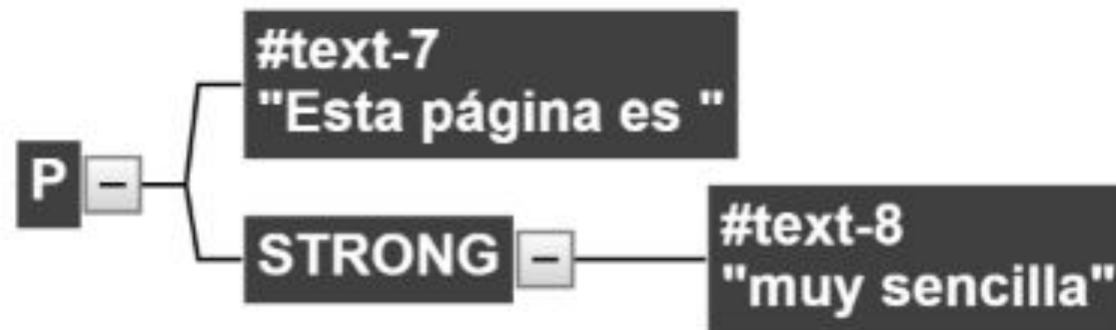
```
<p>Esta página es <strong>muy sencilla</strong></p>
```



# Selectores básicos: getElementById

## Estructura del DOM y Nodos

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





# Selectores básicos: getElementById

## Acceso por objeto document

El acceso a body usando la referencia document.body requiere que el script se incluya al final del <body> en el HTML.

```
console.dir(document)
console.dir(document.head)
console.dir(document.body)
```





# Selectores básicos: getElementById

## Tipos de Nodos

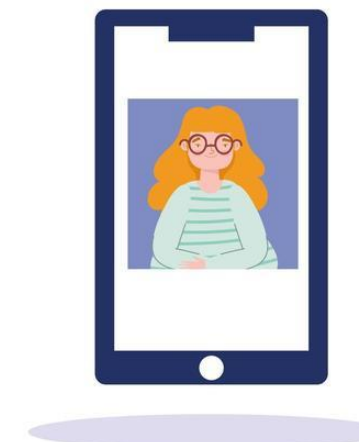
La especificación completa de DOM define 12 tipos de nodos, los más usados son:

### Document:

- Nodo raíz del que derivan todos los demás nodos del árbol.

### Element:

- Representa cada una de las etiquetas XHTML. Puede contener atributos y derivar otros nodos de él.





# Selectores básicos: getElementById

## Attr:

- Se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par atributo=valor.

## Text:

- Nodo que contiene el texto encerrado por una etiqueta HTML.

## Comment:

- Representa los comentarios incluidos en la página HTML.





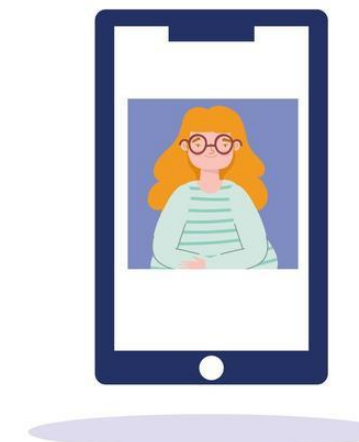
# Selectores básicos: getElementById

## Acceso al DOM

Existen distintos métodos para acceder a los elementos del DOM empleando en la clase Document.

Los más comunes son:

- `getElementsByTagName()`
- `getElementsByClassName()`
- `getElementById()`





# Selectores básicos: getElementById

## getElementById():

El método getElementById() sirve para acceder a un elemento de la estructura HTML, utilizando su atributo ID como identificación.

```
//código HTML
<div id="mi-id">
  <p>Lorem</p>
</div>
//código JS
const miElemento = document.getElementById('mi-id');
```







# Selectores básicos: getElementById

## getElementsByTagName():

Este método permite seleccionar elementos por su etiqueta HTML. Recibe como argumento el nombre de la etiqueta y devuelve una colección de elementos que coinciden con esa etiqueta.

```
//código HTML
<div id="mi-id">
  <p>Lorem</p>
</div>
//código JS
const elementosP = document.getElementsByTagName('p');
```





# Selectores básicos: getElementById

## getElementsByClassName():

Este método permite seleccionar elementos por su clase CSS. Recibe como argumento el nombre de la clase y devuelve una colección de elementos que tienen esa clase.

```
//código HTML
<div id="mi-id">
  <p class="mi-clase">Este es un párrafo con la clase "mi-clase".</p>
</div>
//código JS
const elementosClase = document.getElementsByClassName('mi-clase');
```



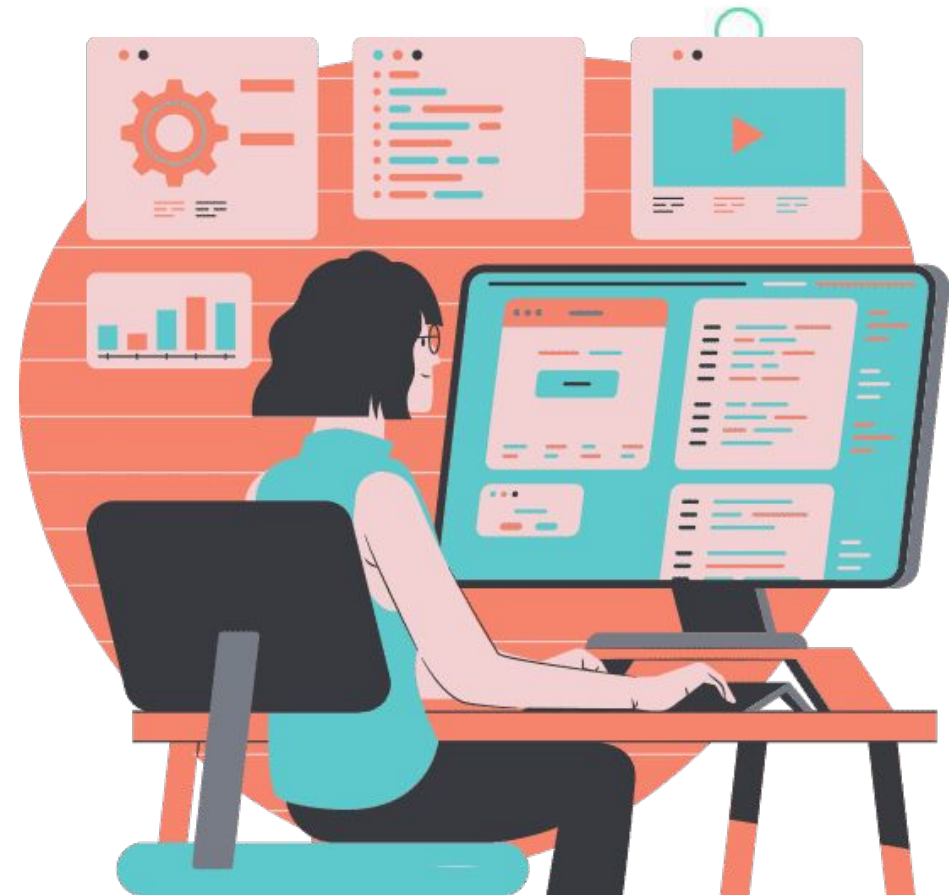
# ➤ Obtención y manipulación de valores y textos de los elementos del DOM



# Obtención y manipulación de valores y textos de los elementos del DOM

La propiedad `innerText` solo muestra el texto visible en el elemento y no incluye el texto oculto por estilos CSS, mientras que la propiedad `textContent` muestra todo el contenido textual, incluyendo el texto oculto.

Si necesitas modificar todo el contenido textual, incluyendo el oculto, puedes usar `textContent` en su lugar. Al utilizar `innerText`, se aplicarán las reglas de escape de HTML, lo que significa que cualquier etiqueta HTML dentro del texto será mostrada como texto plano. Si deseas modificar el contenido HTML completo de un elemento, debes utilizar la propiedad `innerHTML` en su lugar.






# Obtención y manipulación de valores y textos de los elementos del DOM



## Innet Text:

La propiedad innerText de un nodo nos permite modificar su nodo de texto. Es decir, acceder y/o modificar el contenido textual de algún elemento del DOM.



```
const elemento = document.getElementById('mi-elemento');  
elemento.innerText = 'Nuevo texto';
```





# Obtención y manipulación de valores y textos de los elementos del DOM



innerHTML permite definir el código html interno del elemento seleccionado. El navegador lo interpreta como código HTML y no como contenido de texto, permitiendo desde un string crear una nueva estructura de etiquetas y contenido.

Al pasar un string con formato de etiquetas html y contenido a través de la propiedad innerHTML, el navegador genera nuevos nodos con su contenido dentro del elemento seleccionado.





# Obtención y manipulación de valores y textos de los elementos del DOM



## Class Name

A través de la propiedad `className` de algún nodo seleccionado podemos acceder al atributo `class` del mismo y definir cuáles van a ser sus clases:



```
const elemento = document.getElementById('mi-elemento');  
elemento.className = 'clase-1 clase-2';
```

Al seleccionar un elemento con el ID utilizando `getElementById()` asignamos un nuevo valor a la propiedad `className` del elemento.

Esto reemplazará todas las clases existentes del elemento por estas nuevas clases.

# Obtención y manipulación de valores y textos de los elementos del DOM

## Class Name

Podes utilizar className para agregar o quitar clases individuales del elemento:

```
const elemento = document.getElementById('mi-elemento');  
elemento.className += ' nueva-clase';
```

Utilizamos el operador += para concatenar la clase "nueva-clase" al atributo class existente del elemento, sin eliminar las clases previas.





# Obtención y manipulación de valores y textos de los elementos del DOM

Es importante destacar que al utilizar `className`, debes tener en cuenta que estás sobrescribiendo por completo el atributo `class`.

Si necesitas realizar operaciones más complejas con las clases, como agregar o quitar clases específicas, te recomendaría utilizar la propiedad `classList` en su lugar, que ofrece métodos más flexibles para trabajar con las clases de un elemento.



# ➤ Agregar o quitar Nodos



# Agregar o quitar Nodos

## Creación de elementos

Para crear elementos se utiliza la función `document.createElement()`, y se debe indicar el nombre de etiqueta HTML que representará ese elemento. Luego debe agregarse como hijo el nodo creado con `append()`, al body o a otro nodo del documento actual.





# Agregar o quitar Nodos

## **createElement()**

Para crear un nuevo elemento utilizamos el método `document.createElement()`. Luego, asignamos un contenido de texto al elemento utilizando la propiedad `textContent`. Luego utilizamos el método `appendChild()` para agregarlo al árbol del nodo.


```
// Crear un elemento <p>
const nuevoParrafo = document.createElement('p');
// Agregar contenido al elemento
nuevoParrafo.textContent = 'Este es un nuevo párrafo';
// Obtener el nodo padre al que deseas agregar el nuevo elemento
const contenedor = document.getElementById('mi-contenedor');
// Agregar el nuevo elemento como hijo del nodo padre
contenedor.appendChild(nuevoParrafo);
```



# Agregar o quitar Nodos

## Eliminar elementos

Para quitar nodos del DOM, puedes utilizar el método `removeChild()` para eliminar un nodo específico o utilizar `parentNode.removeChild()` para eliminar un nodo hijo de su nodo padre.



```
// Obtener el nodo que deseas eliminar
const nodoEliminar = document.getElementById('mi-nodo');

// Obtener el nodo padre del nodo a eliminar
const nodoPadre = nodoEliminar.parentNode;

// Eliminar el nodo del DOM
nodoPadre.removeChild(nodoEliminar);
```



# Agregar o quitar Nodos

## **remove()**

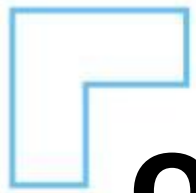
El método `remove()` te permite eliminar un nodo seleccionado del DOM. puedes llamar a este método en el nodo que deseas eliminar y se eliminará del árbol DOM.

```
const elemento = document.getElementById('mi-elemento');  
elemento.remove();
```

Es importante destacar que al utilizar `remove()`, el nodo eliminado no se puede recuperar, por lo que debes tener cuidado al utilizar este método y asegurarte de que realmente deseas eliminar el nodo del DOM.

Hay que tener en cuenta que el método `remove()` no es compatible con todos los navegadores, especialmente versiones antiguas.

# ➤ Obtener datos de Inputs



# Obtener datos de Inputs

Para obtener o modificar datos de un formulario HTML desde JS, podemos hacerlo mediante el DOM. Accediendo a la propiedad value de cada input seleccionado.







# Obtener datos de Inputs

## input

tenemos un formulario con dos campos de entrada:

- uno para el nombre
- otro para el email.

Luego, tenemos un botón que al hacer clic invoca la función `mostrarDatos()`.

```
<form id="mi-formulario">
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" />
  <label for="email">Email:</label>
  <input type="email" id="email" />
  <button type="button" onclick="mostrarDatos()">Mostrar Datos</button>
</form>
```



# Obtener datos de Inputs

## input



En la función mostrarDatos(), utilizamos getElementById() para obtener los elementos de entrada del formulario por su ID.

Luego, accedemos a la propiedad value de cada elemento para obtener su valor actual.



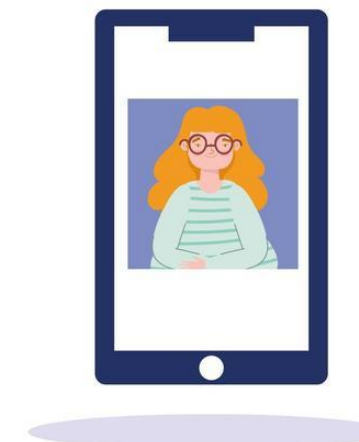
```
function mostrarDatos() {  
  // Obtener los valores de los campos de entrada  
  const nombre = document.getElementById('nombre').value;  
  const email = document.getElementById('email').value;  
  
  // Mostrar los datos en la consola o realizar alguna acción con ellos  
  console.log('Nombre:', nombre);  
  console.log('Email:', email);  
}
```

# › Query Selector



# Query Selector

El método `querySelector()` nos permite seleccionar nodos del DOM utilizando selectores CSS. Funciona de manera similar a cómo seleccionamos elementos con CSS en nuestros estilos.





# Obtener datos de Inputs

## querySelector()

Devuelve el primer elemento que coincide con el selector especificado. Si hay varios elementos que coinciden con el selector, solo se seleccionará el primero.

```
<div id="mi-elemento" class="clase1 clase2">
  <p>Contenido del elemento</p>
</div>

// Seleccionar el elemento por su ID
const elementoID = document.querySelector('#mi-elemento');

// Seleccionar el elemento por su clase
const elementoClase = document.querySelector('.clase1');

// Seleccionar el elemento por su etiqueta
const elementoEtiqueta = document.querySelector('p');
```



# Obtener datos de Inputs

## Query Selector All

Query Selector me retorna el primer elemento que coincida con el parámetro de búsqueda, o sea un sólo elemento. Si quiero obtener una colección de elementos puede utilizar el método `querySelectorAll()` siguiendo el mismo comportamiento.

```
<ul>
  <li>Elemento 1</li>
  <li>Elemento 2</li>
  <li>Elemento 3</li>
</ul>

// Seleccionar todos los elementos <li> dentro del <ul>
const elementos = document.querySelectorAll('ul li');
// Iterar sobre la colección de elementos
elementos.forEach((elemento) => {
  console.log(elemento.textContent);
});
```

# LIVE CODING

Ejemplo en vivo

**Selectores básicos: getElementById y Obtención y manipulación de valores y textos de los elementos del DOM.**

*En este live coding, aprenderemos a utilizar el selector básico getElementById en JavaScript para acceder a elementos específicos del DOM (Document Object Model) por su ID.*

*Luego, exploraremos cómo obtener y manipular los valores y textos de esos elementos. Este conocimiento es fundamental para interactuar con y modificar la página web en tiempo real.*

  **Tiempo: 10 minutos**



# LIVE CODING

Ejemplo en vivo

*Crea una página HTML con al menos tres elementos diferentes que tengan IDs únicos (por ejemplo, un párrafo, un encabezado y un botón).*

*Utiliza JavaScript para realizar las siguientes acciones:*

- *Selecciona uno de los elementos por su ID utilizando getElementById.*
- *Obtén y muestra en la consola el contenido HTML y de texto de ese elemento.*
- *Modifica el contenido de ese elemento (cambia el texto o el estilo) utilizando JavaScript.*
- *Actualiza la visualización de la página para reflejar los cambios.*



# Evaluación Integradora ✨



## Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase

[CLICK AQUÍ](#)

para ver la consigna completa



# **Ejercicio N° 1**

# **Manipulación del DOM**



# Manipulación del DOM

Breve descripción

## Consigna: 🛠️

Crea una página HTML con al menos un elemento en el que puedas aplicar la manipulación del DOM con innerHTML. Puedes usar un párrafo, un div, o cualquier otro elemento de tu elección. Utiliza JavaScript para realizar las siguientes acciones:

- Selecciona el elemento utilizando getElementById u otro selector.
- Utiliza innerHTML para cambiar, agregar o eliminar contenido HTML dentro de ese elemento.
- Actualiza la visualización de la página para reflejar los cambios.

Ejemplos de acciones que puedes realizar:

- Cambiar el texto dentro del elemento.
- Agregar contenido HTML, como listas o imágenes.
- Eliminar contenido HTML del elemento.

**Tiempo🕒: 15 minutos**



# Manipulación del DOM

Breve descripción

## Contexto: 🙌

Exploraremos cómo utilizar la propiedad `innerHTML` en JavaScript para manipular el contenido de elementos HTML en el DOM (Document Object Model). `innerHTML` nos permite cambiar, agregar o eliminar contenido HTML de manera dinámica en una página web.



## Paso a paso: ⚙️

En JavaScript, crea una función que se ejecute después de que se cargue el DOM utilizando `document.addEventListener('DOMContentLoaded', function () {...});`.

Dentro de la función, utiliza `document.getElementById` u otro selector para seleccionar el elemento que deseas manipular.

Utiliza la propiedad `innerHTML` para obtener o establecer el contenido HTML del elemento seleccionado.

Puedes asignar nuevo contenido HTML al elemento seleccionado, modificar el contenido existente o eliminarlo.

○

# ¿Alguna consulta?

+



# RESUMEN

¿Qué logramos en esta clase?

- ✓ Manipular el Document Object Model (DOM) de una página web utilizando JavaScript
- ✓ Seleccionar elementos HTML en una página utilizando selectores como getElementById y querySelector.
- ✓ Manipulación de contenido HTML mediante propiedades como innerHTML



# #WorkingTime

Continuemos ejercitando

**¡Antes de cerrar la clase!** Te invitamos a: 🙌 🙌 🙌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
  - a. Material 1: Lección 5: Bases del lenguaje Javascript: 39-48
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

# ¡Muchas Gracias!

Nos vemos en la próxima clase 🙌

