

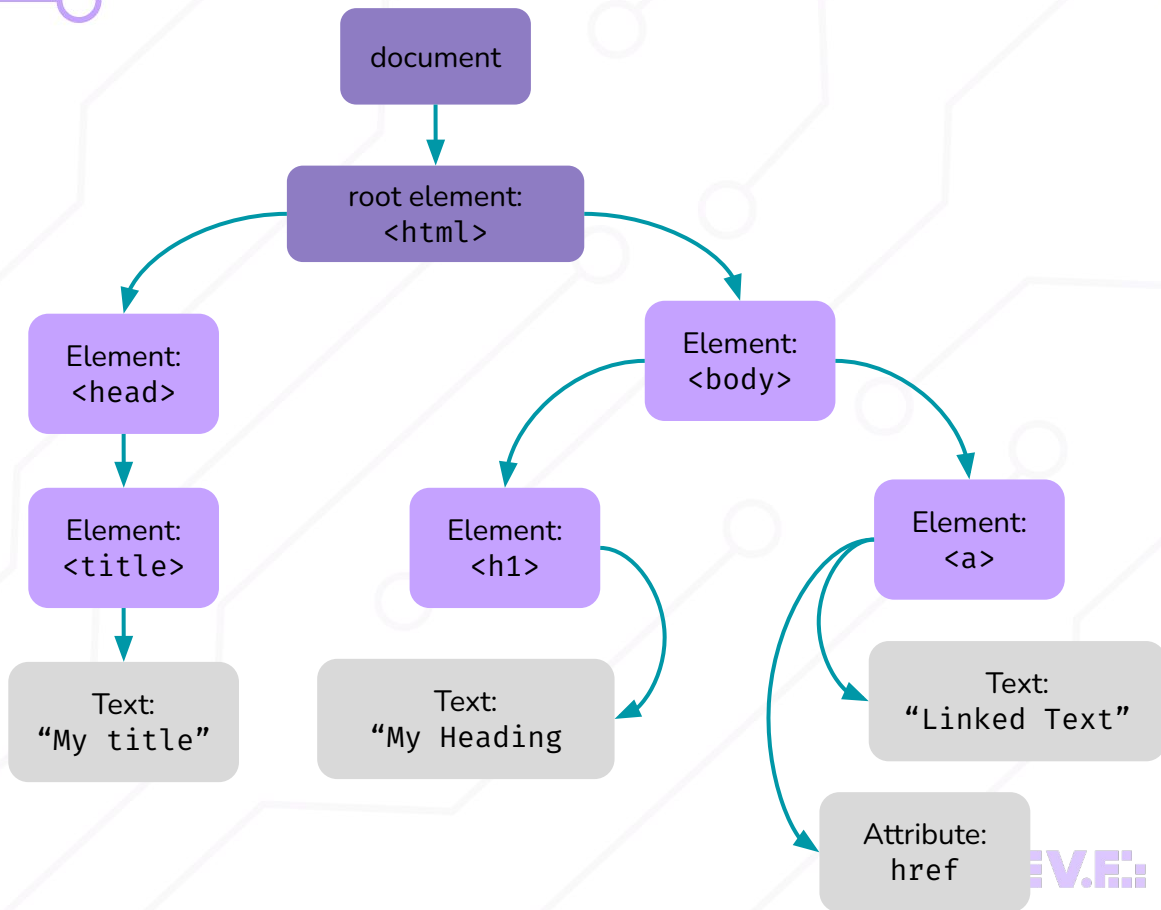
# DOM

(Document Object Model)

**DEV.FX**  
DESARROLLAMOS(PERSONAS);

# DOM (Document Object Model)

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.



# DOM (Document Object Model)

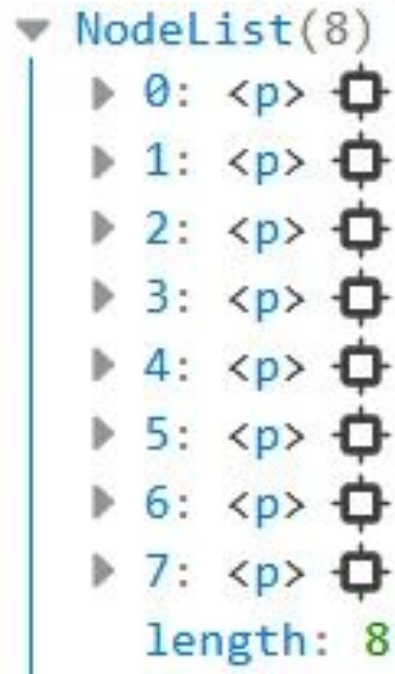
En Javascript, la forma de acceder al DOM es a través de un objeto llamado **document**, que representa el árbol DOM de la página o, más concretamente, la página de la pestaña del navegador donde nos encontramos. En su interior pueden existir varios tipos de elementos, pero principalmente serán **ELEMENT**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

**NODE:** Es la representación genérica de una etiqueta: **HTMLElement**.

**ELEMENT:** Es una unidad más básica, la cual puede ser un **ELEMENT** o un nodo de texto.

# Seleccionar Elementos del DOM

Para hacer modificaciones en un elemento de la página HTML, lo primero que debemos hacer es buscar dicho elemento. Para ello, se suele intentar identificar el elemento a través de alguno de sus atributos más utilizados, generalmente el `id` o `class`.




# Seleccionar Elementos del DOM

Tipo	Método de Búsqueda	Descripción
ELEMENT	<code>getElementById('')</code>	Busca el elemento HTML por su id.
ARRAY	<code>getElementsByClassName('')</code>	Busca elementos con la clase especificada.
ARRAY	<code>getElementsByTagName('')</code>	Busca elementos con el atributo name o value especificado.
ARRAY	<code>getElementsByTagName('')</code>	Busca elementos con el nombre de etiqueta especificado.

# Seleccionar Elementos del DOM

Tipo	Método de Búsqueda	Descripción
ELEMENT	<code>querySelector('')</code>	Busca el primer elemento que coincide con el selector CSS
ARRAY	<code>querySelectorAll('')</code>	Busca todos los elementos que coinciden con el selector CSS

# HTML Collection .vs. Node List

	HTML Collections	Node List
Métodos que los retornan	<code>getElementByTagName()</code> <code>getElementsByClassName()</code>	<code>getElementsByName()</code> <code>querySelectorAll()</code>
Contiene	Únicamente nodos	nodos, atributos y nodos de texto
name e index	Podemos acceder a los nodos a través de su name, atributos, id, e índice. <code>boxes.namedItem("box1")</code>	Podemos acceder a los nodos a través de su índice
Vivos o estáticos	vivos	estáticos  <b>Excepción:</b> <code>getElementsByName()</code> regresa <code>NodeList</code> – pero viva.

# Crear Elementos HTML

Tipo	Método de Búsqueda	Descripción
ELEMENT	<code>parent.createElement('tag')</code>	Crea y devuelve el elemento HTML definido por la etiqueta.
NODE	<code>parent.createTextNode('')</code>	Crea y devuelve un nodo HTML con el texto text.
BOOLEAN	<code>element.isConnected</code>	Indica si el nodo HTML está insertado en el documento HTML.



Creados mas NO añadidos al documento HTML



# Obtener Atributos HTML

Tipo	Método de Búsqueda	Descripción
BOOLEAN	<code>element.hasAttributes()</code>	Indica si el elemento tiene atributos HTML.
BOOLEAN	<code>element.hasAttribute('')</code>	Indica si el elemento tiene el atributo HTML especificado.
ARRAY	<code>element.getAttributeNames()</code>	Devuelve un con los atributos del elemento.
STRING	<code>element.getAttribute('')</code>	Devuelve el valor del atributo especificado del elemento o null si no existe.

# Modificar o Eliminar Atributos HTML


Método	Descripción
<code>element.setAttribute(attr, value)</code>	Añade o cambia el atributo attr al valor value del elemento HTML.
<code>element.toggleAttribute(attr)</code>	Añade atributo attr si no existe, si existe lo elimina.
<code>element.removeAttribute(attr)</code>	Elimina el atributo attr del elemento HTML.

# Modificar o Eliminar Clases CSS

Tipo	Método de Búsqueda	Descripción
STRING	<code>element.className</code>	Acceso directo al valor del atributo HTML class. También se puede asignar.
OBJECT	<code>element.classList</code>	Objeto especial para manejar clases CSS. Contiene métodos y propiedades de ayuda.

Tipo	Método	Descripción
OBJECT	<code>element.classList</code>	Objeto especial para manejar clases CSS. Contiene métodos y propiedades de ayuda.
NUMBER	<code>.classList.length</code>	Devuelve el número de clases del elemento HTML.
STRING	<code>.classList.item(n)</code>	Devuelve la clase número n del elemento HTML. si no existe.
BOOLEAN	<code>.classList.contains(class)</code>	Indica si la clase existe en el elemento HTML.
	<code>.classList.add(c1, c2, ...)</code>	Añade las clases c1, c2... al elemento HTML.
	<code>.classList.remove(c1, c2, ...)</code>	Elimina las clases c1, c2... del elemento HTML.
BOOLEAN	<code>.classList.toggle(class)</code>	Si la clase no existe, la añade. Si no, la elimina.
BOOLEAN	<code>.classList.replace(old, new)</code>	Reemplaza la clase old por la clase new.

# Contenido en el DOM

Tipo	Método de Búsqueda	Descripción
STRING	<code>element.textContent</code>	Devuelve el contenido de texto del elemento. Se puede asignar para modificar.
STRING	<code>element.innerHTML</code>	Devuelve el contenido HTML del elemento. Se puede usar asignar para modificar.
STRING	<code>element.innerText</code>	Versión no estándar de <code>.textContent</code> de Internet Explorer  EVITAR

# .textContent .vs. .innerText .vs. .innerHTML

This is my **link** collection:

- [Texto bold](#)
- [Texto 2 énfasis](#)

```
Sources Elements Console
<html>
  <head></head>
  <body>
    <div id="mylinks">
      " This is my "
      <b>link collection</b>
      ": "
      <ul>
        <li>
          ::marker
          <a href="#">
            "Texto "
            <b>bold</b>
          </a>
        </li>
        <li>
          ::marker
          <a href="#">
            "Texto 2 "
            <em>énfasis</em>
          </a>
        </li>
      </ul>
    </div>
  </body>
</html>
```

```
> const div = document.querySelector("#mylinks");
< undefined
> console.log(div.textContent);
```

This is my link collection:

Texto bold  
Texto 2 énfasis

```
> console.log(div.innerText);
```

This is my link collection:  
Texto bold  
Texto 2 énfasis

```
> console.log(div.innerHTML);
```

This is my <b>link collection</b>:

```
<ul>
  <li><a href="#">Texto <b>bold</b> </a></li>
  <li><a href="#">Texto 2 <em>énfasis</em></a></li>
</ul>
```

# Nodos

Tipo	Método de Búsqueda	Descripción
NODE	<code>parent.appendChild(node)</code>	Añade como hijo el nodo node. Devuelve el nodo insertado.
NODE	<code>parent.removeChild(node)</code>	Elimina y devuelve el nodo hijo node.
NODE	<code>parent.replaceChild(new, old)</code>	Reemplaza el nodo hijo old por new. Devuelve old.
NODE	<code>parent.insertBefore(new, node)</code>	Inserta el nodo new antes de node y como hijo del nodo actual.

# Eventos

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev



# ¿Qué son los eventos?

Hay situaciones en las que necesitamos realizar una determinada acción cuando ocurra un determinado caso. En estas situaciones, no sabemos exactamente cuando tenemos que activar nuestra funcionalidad, ya que no podemos predecir cuando el usuario de nuestra página realizará la acción necesaria (y podrá ser diferente en cada situación).

En estas situaciones es cuando entran en juego los eventos.



# Formas de manejar eventos

Forma	Ejemplo
Atributos HTML	<pre>&lt;button   onClick="sayHello()"&gt; &lt;/button&gt;</pre>
Propiedades Js	<pre>element.onClick = sayHello()</pre>
<code>addEventListener</code>	<pre>elemento.addEventListener ( 'listener', sayHello() )</pre>

[Lista de eventos en HTML.](#)

# Objeto Event

El objeto Event representa un evento que tiene lugar en el DOM.

**Un evento** es disparado para **una acción del usuario** (click en un botón, scroll en una sección, presionar una tecla, etc.). Todo evento que se dispara permite capturar información sobre quien lo lanzó y una de las más importantes es **event.target**.

[Doc de Event.](#)

# Guías de apoyo

- ❑ <https://lenguajejs.com/javascript/dom/que-es/>
- ❑ <https://www.freecodecamp.org/news/what-is-the-dom-document-object-model-meaning-in-javascript/>
- ❑ <https://baja.website/manipulacion-del-dom-y-eventos-con-selectores-javascript/>