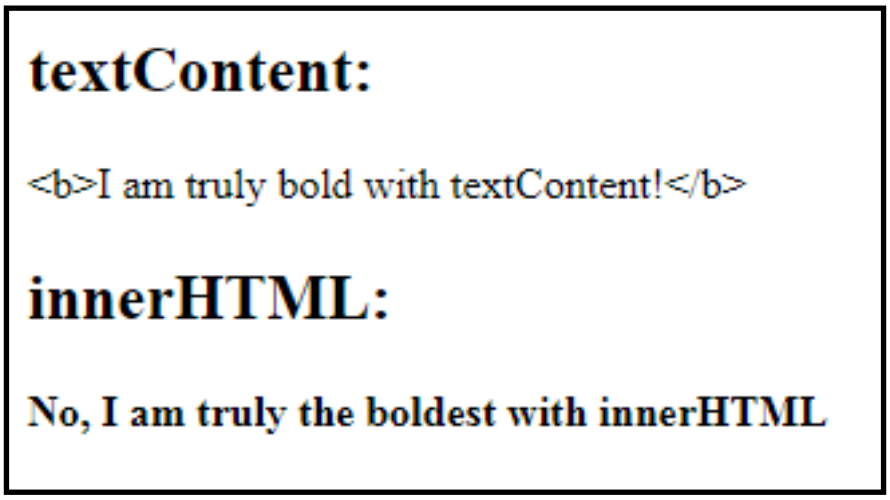
3. Manipulación del DOM

La manipulación del DOM en JavaScript permite cambiar dinámicamente el contenido, los atributos y los estilos de los elementos de una página web. Esto posibilita la creación de aplicaciones web interactivas y dinámicas. Algunos métodos y propiedades comunes para la manipulación del DOM son:



- **textContent**: Permite obtener o establecer el contenido de texto de un elemento.
  const miElemento = document.getElementById('miElemento');
  miElemento.textContent = 'Nuevo texto';

- **innerHTML**: Permite obtener o establecer el contenido HTML de un elemento.
  const miElemento = document.getElementById('miElemento');
  miElemento.innerHTML = 'Nuevo contenido';

3.1. setAttribute()

El atributo "setAttribute" es una función que permite modificar o establecer atributos en elementos HTML. La sintaxis general para `setAttribute` es la siguiente:

elemento.setAttribute(nombreAtributo, valorAtributo);

- **elemento**: Es la referencia al elemento HTML al que añadir o modificar un atributo.
- **nombreAtributo**: Es una cadena de texto que representa el nombre del atributo que se quiere modificar o agregar.
- **valorAtributo**: Es el nuevo valor que deseas asignar al atributo especificado.

```
const miElemento = document.getElementById('miElemento');
miElemento.setAttribute('class', 'nuevaClase');
```

<div id=“miDiv">

Este es un div.

** **

Se utilizará `setAttribute` para cambiar el valor del atributo "id" del elemento:

var miDiv = document.getElementById("miDiv");

miDiv.setAttribute("id", "nuevoId");

Después de ejecutar este código:

<div id=“nuevoId">

Este es un div.

** **

EJERCICIO A: Crear una página HTML que contenga un div con un ID ("miDiv") y un botón. Al hacer clic en el botón, se llama a la función `cambiarIdDelDiv`. Esta función utiliza `document.getElementById` para obtener una referencia al div y luego utiliza `setAttribute` para cambiar el valor del atributo "id" del div de "miDiv" a "nuevoId" cuando se hace clic en el botón. Diseña un sitio web que utilice lo anterior.

EJERCICIO B: Validación de Correo Electrónico. Crea un formulario de registro que solicite al usuario su nombre y dirección de correo electrónico. Implementa una validación que cuando el usuario introduzca una dirección de correo electrónico que no cumpla con el formato esperado, se resalte el campo con un color de fondo rojo y muestra un mensaje de error.

- Crear un formulario HTML que incluya los siguientes elementos:
  - Un campo de texto para el nombre.
  - Un campo de texto para la dirección de correo electrónico.
  - Un botón de envío.
- Escribe una función en JavaScript que se ejecute cuando el usuario introduzca texto en el campo de correo electrónico. Esta función debe verificar si la dirección de correo electrónico es válida. Para la validación de direcciones de correo electrónico, puedes utilizar expresiones regulares o funciones que verifiquen si el formato es válido (por ejemplo, si contiene "@" y un dominio válido).
- Si la dirección de correo electrónico no es válida, cambia el "id" del campo de correo electrónico a "emailError" usando setAttribute. Esto permitirá aplicar estilos CSS específicos al campo con errores.
- Aplica un estilo CSS al campo con el "id" "emailError" para resaltarlo, estableciendo un fondo rojo y mostrando un mensaje de error junto al campo.
- **Mini Anexo:** La expresión regular `var regex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;` es utilizada para validar direcciones de correo electrónico. Las expresiones regulares son patrones de búsqueda que se utilizan para verificar si una cadena de texto cumple con un formato específico.

  - `^`: Representa el inicio de la cadena, la validación debe comenzar desde el principio de la cadena.
  - `[a-zA-Z0-9._-]+`: Esto coincide con uno o más caracteres que pueden ser letras mayúsculas o minúsculas (a-z, A-Z), dígitos (0-9), puntos (`.`), guiones bajos (`_`) o guiones (`-`). Esto representa la parte del nombre de usuario en una dirección de correo electrónico.
  - `@`: Coincide con el carácter "@" que separa el nombre de usuario del dominio en una dirección de correo electrónico.
  - `[a-zA-Z0-9.-]+`: Nuevamente, esto coincide con uno o más caracteres que pueden ser letras mayúsculas o minúsculas, dígitos, puntos o guiones. Representa la parte del dominio.
  - ‘\.`: Coincide con un punto literal (`.`), que es necesario en todas las direcciones de correo electrónico válidas.
  - `[a-zA-Z]{2,4}`: Esto coincide con entre 2 y 4 letras, lo que representa la extensión del dominio (como "com" o "edu").
  - `$`: Representa el final de la cadena, lo que significa que la validación debe terminar al final de la cadena.

3.2. createElement()

En JavaScript, puedes crear y modificar elementos del DOM utilizando diversas funciones y métodos. **Crear un nuevo elemento y añadirlo al DOM,** se utiliza createElement() para crear un nuevo elemento y appendChild() o insertBefore() para añadirlo a un elemento existente. La sintaxis básica del método es:

var nuevoElemento = document.createElement(“nombreDelElemento");

- nombreDelElemento: Es una cadena de texto que especifica el tipo del elemento HTML que se va a crear, por ejemplo, "div", "p", "img", "a", "h1", etc.

```
const nuevoParrafo = document.createElement('p');
// Agregar texto al párrafo
nuevoParrafo.textContent = 'Este es un nuevo párrafo';
// Añadir el párrafo al body
document.body.appendChild(nuevoParrafo);
```

Ejemplo: Crear un párrafo y agregarlo a la página:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<button onclick="agregarParrafo()">Agregar Párrafo</button>
<script>
function agregarParrafo() { // Crear un nuevo elemento de párrafo
    var nuevoParrafo = document.createElement("p"); // Personalizar el párrafo (agregar texto)
    nuevoParrafo.textContent = "Este es un nuevo párrafo creado dinámicamente."; // Obtener el elemento en el que deseamos agregar el nuevo párrafo
    var contenedor = document.body; // Agregar el nuevo párrafo al DOM
    contenedor.appendChild(nuevoParrafo);
}
</script>
</body>
</html>
```

3.3. appendChild()

appendChild es un método en JavaScript que se utiliza para agregar un nodo (elemento, atributo, texto, etc.) como un hijo de otro nodo. La sintaxis general es:

parentNode.appendChild(childNode);

- **parentNode**: Este es el nodo al que deseas agregar el childNode como hijo.
- **childNode**: Este es el nodo que deseas agregar como hijo al parentNode.

```
const padre = document.getElementById('padre');

const nuevoElemento = document.createElement('div');

padre.appendChild(nuevoElemento);
```

Por ejemplo, si tenemos un elemento HTML existente y deseas agregar un nuevo elemento como su hijo, puedes usar `appendChild`:

```
<!DOCTYPE html>
<html> <head> </head>
 <body>
<div id="contenedor">
<p>Este es un párrafo existente.</p>
</div>
<script> // Crear un nuevo elemento
    var nuevoParrafo = document.createElement("p");
    nuevoParrafo.textContent = "Este es un nuevo párrafo."; // Obtener el contenedor existente
    var contenedor = document.getElementById("contenedor"); // Agregar el nuevo párrafo como hijo del contenedor
    contenedor.appendChild(nuevoParrafo);
</script>
</body>
</html>
```

EJERCICIO C: Crear un nuevo elemento de párrafo utilizando document.createElement() y agregarlo al Document Object Model (DOM) de una página web. Se utiliza un botón que, al hacer clic en él, llama a una función que crea el párrafo y lo inserta como hijo de un div existente en la página. Usa appendChild y createElement.

3.4. insertBefore()

`insertBefore` es un método que se utiliza para agregar un nuevo nodo como hijo de un elemento específico en el DOM antes de un nodo hijo existente. La sintaxis general de `insertBefore` es la siguiente:

```
parentNode.insertBefore(newNode, referenceNode);
```

- `parentNode`: Es el nodo padre al que deseas agregar el nuevo nodo.
- `newNode`: Es el nodo que deseas agregar como nuevo hijo.
- `referenceNode`: Es el nodo hijo existente antes del cual deseas insertar `newNode`.

```
// Crear un nuevo elemento
var newElement = document.createElement("p");
newElement.textContent = "Nuevo párrafo";

// Obtener el elemento padre al que deseas agregar el nuevo elemento
var parentElement = document.getElementById("contenedor");

// Obtener el elemento hijo existente antes del cual deseas insertar el nuevo elemento
var referenceElement = parentElement.firstChild;

// Insertar el nuevo elemento antes del elemento de referencia
parentElement.insertBefore(newElement, referenceElement);
```

En este ejemplo, hemos creado un nuevo elemento `p` y le hemos asignado texto. Luego, hemos obtenido el elemento padre al que deseamos agregar el nuevo elemento (`contenedor`) y el elemento hijo existente antes del cual deseamos insertar el nuevo elemento. Finalmente, hemos utilizado `insertBefore` para insertar el nuevo elemento antes del elemento de referencia.

3.5. removeChild()

removeChild es un método en JavaScript que se utiliza para eliminar un nodo hijo específico de un elemento padre. Recuerda, el método removeChild se llama en el nodo padre (parentNode) y toma como argumento el nodo hijo (childNode) que se desea eliminar. La sintaxis general del método removeChild es la siguiente:

```
parentNode.removeChild(childNode);
```

- **parentNode**: Es el nodo padre del que se quiere eliminar un hijo.
- **childNode**: Es el nodo que se quiere eliminar.

También se puede acceder de la siguiente forma:

```
const hijo = document.getElementById('hijo');
hijo.parentNode.removeChild(hijo);
```

```
<!DOCTYPE html> <html>
<body>
<div id="contenedor">
<p>Este párrafo será eliminado.</p>
<button onclick="eliminarParrafo()">Eliminar Párrafo</button>
</div>
<script>
function eliminarParrafo() { // Obtener el nodo padre
        var contenedor = document.getElementById("contenedor"); // Obtener el nodo hijo que queremos eliminar var
        parrafoAEliminar = document.querySelector("#contenedor p"); // Eliminar el nodo hijo del nodo
        padre contenedor.removeChild(parrafoAEliminar);
}
</script>
</body>
</html>
```

EJERCICIO D: Crea una web que al pulsar el botón "eliminarParrafo" que se llame a la función eliminarParrafo(). Esta función obtiene el tercer párrafo dentro del div con el ID "contenedor" y, si existe, lo elimina del DOM utilizando removeChild. Esto resulta en la eliminación del tercer párrafo de la lista de seis párrafos.