Tema 2. DOM (Document Object Model)

2.2. Atributos

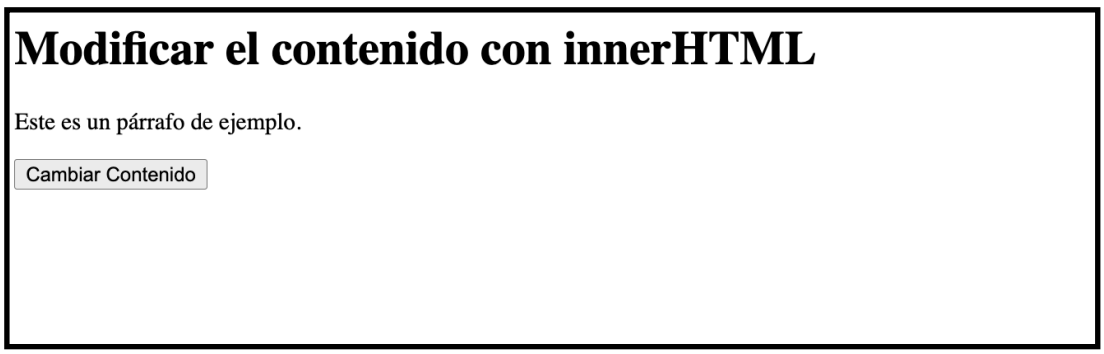
Cuando trabajamos con JavaScript y el DOM (Document Object Model), es esencial comprender cómo acceder y manipular los elementos de una página web. Para ello, existen una serie de atributos clave que nos permiten interactuar con los elementos de manera efectiva.

Estos atributos abarcan desde la obtención del texto contenido en un elemento hasta la modificación de estilos y clases, e incluso la gestión de valores en formularios. A continuación, exploraremos algunos de estos atributos comunes

textContent: `textContent` representa el texto contenido dentro de un elemento. No interpreta ni ejecuta código HTML y devuelve el texto como está.

```
const elemento = document.getElementById('miElemento');
console.log(elemento.textContent);
```

Ejercicio A: Modificación del contenido de un párrafo con textContent. Crea una página web que contenga un párrafo inicial con el texto "Este es un párrafo de ejemplo." y un botón con el texto "Cambiar Contenido". Cuando el usuario haga clic en el botón, el contenido del párrafo deberá cambiar dinámicamente a "¡El contenido ha sido cambiado!" utilizando la propiedad textContent en una función JavaScript.



innerHTML: innerHTML permite acceder al contenido HTML dentro de un elemento. Puede ser utilizado para obtener o modificar el contenido HTML.

```
const elemento = document.getElementById('miElemento');
console.log(elemento.innerHTML);
```

EJERCICIO B. Diferencias entre innerHTML y textContent. Crea una página web que incluya un `div` que contiene un párrafo inicial con el siguiente texto: "Este es un párrafo de ejemplo." Además, agrega dos botones: uno con el texto "Cambiar con innerHTML" y otro con el texto "Cambiar con textContent". Cuando el usuario haga clic en estos botones, deberán ejecutar funciones JavaScript correspondientes que cambien el contenido del `div`.

value: value se aplica a elementos de formulario como input, select y textarea. Representa el valor introducido por el usuario en el campo de entrada.

```
const inputElemento = document.getElementById('miInput');
console.log(inputElemento.value);
```

EJERCICIO C: Uso de la propiedad `value` en un campo de entrada. Crea una página web que contenga un campo de entrada de tipo texto (`<input>`) con un valor inicial "Texto inicial". También incluye un botón con el texto "Cambiar Value". Cuando el usuario haga clic en el botón, deberá ejecutarse una función JavaScript que cambie el valor del campo de entrada.

Diferencias clave:

- **textContent** y **innerHTML** trabajan con contenido de texto y HTML, respectivamente, mientras que **value** se aplica principalmente a elementos de formulario para obtener o establecer el valor.

- **textContent** no interpreta HTML y solo devuelve el texto plano. **innerHTML** interpreta y devuelve el HTML contenido dentro del elemento. **value** se aplica a elementos de formulario y representa el valor del elemento.
- **innerHTML** puede ser usado para cambiar la estructura y el contenido HTML de un elemento. Por otro lado, **textContent** y **value** no cambian la estructura HTML sino solo el contenido o valor.

Se puede tener diferente value y textContent. En este ejemplo, cada opción tiene un `value` **diferente y un texto (**`textContent`**) asociado. Por ejemplo, si seleccionas "Opción 2" del combo, el valor** 2 **sería el que se obtendría al acceder a** `miCombo.value` **y el texto "Opción 2" sería el que se obtendría al acceder a** `miCombo.options[1].textContent`.

```
<select id="miCombo">
  <option value="1">Opción 1</option>
  <option value="2">Opción 2</option>
  <option value="3">Opción 3</option>
  <option value="4">Opción 4</option>
</select>
```

EJERCICIO D. Diferencias entre value, innerHTML y textContent. Crea una página web que incluya un campo de entrada de tipo texto (`<input>`) con un valor inicial "Texto de entrada" y un párrafo (`<p>`) con el texto "Este es un párrafo de ejemplo." Además, agrega tres botones:

- El primer botón debe tener el texto "Cambiar Value". Al hacer clic en este botón, se debe ejecutar una función JavaScript que cambie el valor del campo de entrada a "Nuevo valor del campo" utilizando la propiedad `value`.

- El segundo botón debe tener el texto "Cambiar InnerHTML". Al hacer clic en este botón, se debe ejecutar una función JavaScript que cambie el contenido del párrafo a "¡El contenido ha sido cambiado con innerHTML**!" utilizando la propiedad `innerHTML`.**

- El tercer botón debe tener el texto "Cambiar TextContent". Al hacer clic en este botón, se debe ejecutar una función JavaScript que cambie el contenido del párrafo a "¡El contenido ha sido cambiado con textContent!" utilizando la propiedad `textContent`.

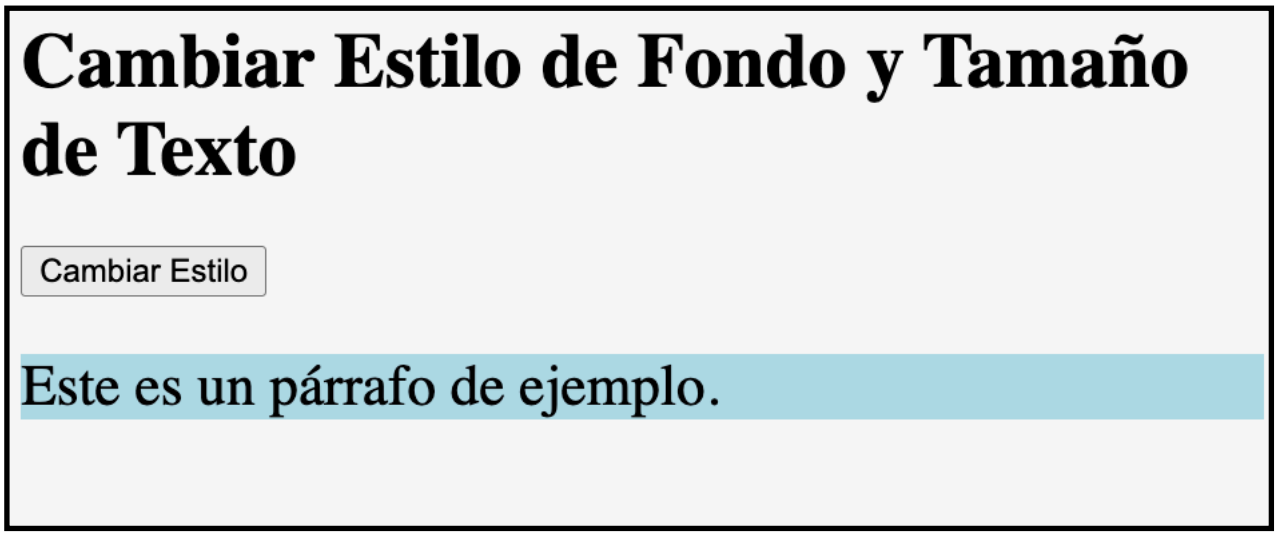
style: style proporciona acceso a las propiedades de estilo CSS de un elemento. Puedes modificar el estilo directamente a través de JavaScript**.**

```
const elemento = document.getElementById('miElemento');
elemento.style.color = 'blue';
```

1. `backgroundColor`: Cambia el color de fondo del elemento.
2. `color`: Cambia el color del texto.
3. `fontSize`: Cambia el tamaño de fuente del texto.
4. `fontFamily`: Cambia la familia de fuentes utilizada.
5. `fontWeight`: Cambia el grosor de la fuente (por ejemplo, "bold" para negritas).
6. `textDecoration`: Cambia la decoración del texto (por ejemplo, "underline" para subrayar).
7. `textAlign`: Cambia la alineación del texto (por ejemplo, "left", "center" o "right").
8. margin: Cambia los márgenes del elemento.
9. padding: Cambia el espacio de relleno alrededor del contenido del elemento.
10. border: Cambia las propiedades del borde (por ejemplo, ancho, estilo y color).
11. width y height: Cambian el ancho y la altura del elemento.
12. display: Cambia el comportamiento de visualización del elemento (por ejemplo, "block", "inline", "none", etc.).

13. `position`: Cambia la posición del elemento en relación con su contenedor
14. `top`, `right`, `bottom`, `left`: Cambian las coordenadas de posición del elemento cuando se utiliza `position: absolute` o `position: relative`.
15. `opacity`: Cambia la opacidad del elemento (un valor entre 0 y 1).
16. `transition`: Controla las transiciones CSS para animar cambios en las propiedades de estilo.

EJERCICIO E. Cambiar el Estilo de Fondo y el Tamaño del Texto. Crea una página web que tenga un botón y un elemento de párrafo (<p>) con un texto inicial. Cuando SE haga clic en el botón, se deberá ejecutar una función JavaScript que cambie el estilo de fondo y el tamaño del texto del párrafo.



src: src se aplica a elementos como imágenes.

```
const imagenElemento = document.getElementById('miImagen');
console.log(imagenElemento.src);
```

EJERCICIO F. Cambio de Imagen. Crea una página web que presente una imagen y un botón, al pulsar el botón esta imagen debe cambiar a otra. Incluye hasta 3 cambios.

href: href se aplica a elementos link. Representa el destino del enlace.

```
const enlaceElemento = document.getElementById('miEnlace');

console.log(enlaceElemento.href);
```

EJERCICIO G. Modificar el atributo href de un enlace . Crea una página web que contenga un enlace (<a>) y un botón. El objetivo es modificar el atributo href del enlace cuando el usuario haga clic en el botón.

checked: checked se aplica a elementos de tipo checkbox y radio. Indica si el elemento está seleccionado o no.

```
const checkboxElemento = document.getElementById('miCheckbox');

console.log(checkboxElemento.checked);
```

EJERCICIO H. Cambiar el estado "checked" de un checkbox con el DOM. Crea una página web que contenga un elemento de tipo "checkbox", un botón y un elemento de párrafo. El objetivo es cambiar el estado "checked" del checkbox cuando el usuario haga clic en el botón utilizando el DOM