CLASE NRO. 14 – MANIPULACIÓN DINÁMICA DEL DOM

El manejo dinámico del DOM se refiere a la capacidad de crear, modificar y eliminar elementos del DOM utilizando JavaScript en tiempo real, basándonos en acciones del usuario u otros eventos.

El manejo dinámico apunta principalmente a la creación de elementos/objetos HTML como etiquetas, botones, imágenes, contenedores (div) y así todo tipo de etiquetas que se van a crear mediante programación y se van a incrustar en en DOM (document object model). Es decir, son etiquetas HTML que no estaban escritas originalmente en el documento HTML estático y por lo tanto se van a crear posteriormente, dependiendo de las interacciones y acciones que realice el usuario. Todas estas etiquetas se generarán mediante programación, se ejecutarán mediante instrucciones que permitirán construir tanto la etiqueta como así también su comportamiento.

Existen básicamente tres formas ó técnicas de incorporar elementos al DOM de forma programática (cuando nos referimos a la forma programática nos referimos mediante programación con el lenguaje JavaScript).

- **Forma 1: Manipulación Programática del DOM (Programmatic DOM Manipulation):**

  Incluye métodos como

  ```
  document.createElement(),
  document.createTextNode(),
  document.createDocumentFragment()
  document.createComment(data)
  ```

- **Forma 2: Manipulación del DOM mediante HTML Interno (Inner HTML Manipulation):**

  Utiliza propiedades como innerHTML para establecer o devolver el contenido HTML de un elemento. Características: Facilita la creación rápida de estructuras HTML complejas, pero puede ser menos seguro y más susceptible a vulnerabilidades XSS.

- **Forma 3: Plantillas del DOM (Template-Based DOM Manipulation):**
  Utiliza elementos <template> para definir fragmentos de HTML que no se renderizan inmediatamente y pueden ser clonados e insertados en el DOM cuando sea necesario.

  Características: Permite la reutilización de fragmentos de HTML, mejora el rendimiento y la organización del código.

Forma 1: Manipulación Programática del DOM (Programmatic DOM Manipulation):

- document.createElement(tagName)

Descripción: Crea un nuevo elemento HTML con el nombre de etiqueta especificado.
var nuevoDiv = document.createElement('div');

- document.createTextNode(data)
  Descripción: Crea un nuevo nodo de texto con el texto especificado.
  var texto = document.createTextNode('Hola, mundo!');

- element.innerHTML

Descripción: Establece o devuelve el contenido HTML de un elemento. Puede usarse para crear múltiples elementos o nodos de texto de una sola vez.

```
var contenedor = document.getElementById('contenedor');
contenedor.innerHTML = '<div class="nuevo">Nuevo contenido</div>';
```

- element.insertAdjacentHTML(position, text)

Descripción: Inserta el texto HTML especificado en una posición determinada en relación con el elemento.
Posiciones posibles:

'beforebegin': Antes del elemento mismo.
'afterbegin': Justo dentro del elemento, antes de su primer hijo.
'beforeend': Justo dentro del elemento, después de su último hijo.
'afterend': Después del elemento mismo.

```
var contenedor = document.getElementById('contenedor');
contenedor.insertAdjacentHTML('beforeend', '<p>Nuevo párrafo</p>');
```

- document.createDocumentFragment()

Descripción: Crea un nodo de documento vacío que puede contener nodos. Es útil para agregar múltiples elementos al DOM de una vez, mejorando el rendimiento.

```
let fragmento = document.createDocumentFragment();
for (let i = 0; i < 5; i++)
{
        let nuevoP = document.createElement('p');
        nuevoP.textContent = 'Párrafo ' + (i + 1);
        fragmento.appendChild(nuevoP);
}

document.getElementById('contenedor').appendChild(fragmento);
```

FORMA1 :

Ventajas:

Control Granular: Permite un control detallado sobre cada aspecto del elemento. Puedes establecer propiedades, atributos, estilos y eventos de manera individual.

Seguridad: Es menos susceptible a ataques XSS (Cross-Site Scripting) porque no se interpreta HTML como código.

Mejor Rendimiento: Especialmente cuando se manipulan muchos elementos o se realizan múltiples operaciones en el DOM, ya que evita el reflujo y el repintado innecesarios del navegador.

Facilidad de Uso con Eventos: Permite agregar fácilmente event listeners directamente al elemento.

Desventajas:

Verbosidad: Puede ser más largo y detallado al escribir comparado con

FORMA 2:

Ventajas:

Simplicidad: Es más conciso y fácil de escribir cuando se necesita agregar múltiples elementos con HTML complejo.
Rapidez en Implementación: Ideal para agregar rápidamente bloques de HTML sin mucho detalle.

Desventajas:

Menos Seguro: Es más susceptible a ataques XSS, especialmente si el contenido HTML incluye datos no confiables.

Rendimiento: Puede causar reflujo y repintado innecesarios en el navegador, ya que toda la estructura interna del elemento se recrea.

Difícil Manipulación Posterior: No es tan conveniente para agregar event listeners directamente a los elementos creados de esta manera.