Clase: Manipulación del DOM y Eventos en JavaScript

Objetivo general:

Al finalizar la clase, los estudiantes serán capaces de manipular el DOM (Document Object Model) de una página web usando JavaScript. También aprenderán a manejar eventos en los elementos HTML y comprenderán cómo interactuar con el DOM de manera eficiente.

---

1. Introducción al DOM (30 minutos)

¿Qué es el DOM? El **Document Object Model (DOM)** es una representación estructural de un documento HTML o XML. Cada elemento HTML se convierte en un objeto que puede ser accedido y manipulado mediante JavaScript.

Relación entre HTML y el DOM: Cuando un navegador carga una página web, convierte el archivo HTML en una estructura jerárquica de nodos conocida como DOM. Cada nodo representa una parte de la página, como un párrafo, una imagen o una lista.

Ejemplo básico de una estructura DOM: Supongamos que tenemos el siguiente código HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo DOM</title>
  </head>
  <body>
    <h1 id="titulo">Bienvenido</h1>
    <p class="parrafo">Este es un párrafo.</p>
    <p class="parrafo">Este es otro párrafo.</p>
  </body>
```

```
</html>
```

El navegador convierte esto en una estructura jerárquica (DOM) que JavaScript puede manipular.

---

2. Selección de Elementos en el DOM (30 minutos)

Para manipular el DOM, primero necesitamos seleccionar los elementos con los que queremos trabajar. Hay varias formas de seleccionar elementos en JavaScript.

Métodos de selección más comunes:

1. **getElementById()**: Selecciona un elemento por su id. Es uno de los métodos más rápidos, ya que los IDs son únicos en el documento.
    - **Ejemplo**:

```
const titulo = document.getElementById("titulo");
console.log(titulo.textContent);  // Muestra el texto "Bienvenido"
```

2. **getElementsByClassName()**: Selecciona todos los elementos que tienen una clase específica y devuelve una colección HTML.
    - **Ejemplo**:

```
const parrafos = document.getElementsByClassName("parrafo");
console.log(parrafos.length);  // Muestra 2, ya que hay dos
     párrafos con la clase "parrafo"
```

3. **querySelector() y querySelectorAll()**:
    - **querySelector()**: Selecciona el primer elemento que coincida con un selector CSS.
    - **querySelectorAll()**: Selecciona todos los elementos que coincidan con un selector CSS.

- **Ejemplo**:

```
const primerParrafo = document.querySelector(".parrafo");

const todosLosParrafos = document.querySelectorAll(".parrafo");

console.log(primerParrafo.textContent);  // Muestra el texto del primer párrafo

console.log(todosLosParrafos.length);  // Muestra 2
```

Actividad práctica:

1. Crear una página HTML simple con varios elementos (encabezados, párrafos, listas).
2. Usar diferentes métodos de selección (getElementById, getElementsByClassName, querySelector) para seleccionar elementos y mostrar su contenido en la consola.

---

3. Manipulación del DOM (45 minutos)

Una vez que seleccionamos los elementos, podemos manipularlos. Podemos cambiar el contenido, las clases, los estilos o incluso agregar y eliminar elementos del DOM.

Manipulación de texto y contenido HTML:

- **textContent**: Cambia el texto dentro de un elemento.
    - **Ejemplo**

```
const titulo = document.getElementById("titulo");

titulo.textContent = "¡Bienvenido al DOM!";  // Cambia el texto del h1
```

- **innerHTML**: Cambia el contenido HTML dentro de un elemento.
    - **Ejemplo**:

```
const parrafo = document.querySelector(".parrafo");
parrafo.innerHTML = "<strong>Este es un nuevo contenido</strong>";
```

Manipulación de estilos: Podemos cambiar el estilo de un elemento a través de la propiedad style.

- **Ejemplo**

```
const titulo = document.getElementById("titulo");
titulo.style.color = "blue";  // Cambia el color del texto a azul
titulo.style.fontSize = "30px";  // Cambia el tamaño de la fuente
```

Agregar y eliminar clases:

- **classList.add()**: Agrega una clase al elemento.
- **classList.remove()**: Elimina una clase.
- **classList.toggle()**: Alterna entre agregar y eliminar una clase.
    - **Ejemplo**:

```
const parrafo = document.querySelector(".parrafo");
parrafo.classList.add("resaltado");  // Agrega la clase "resaltado"
```

Crear y eliminar elementos del DOM:

- **createElement()**: Crea un nuevo elemento.
- **appendChild()**: Agrega un elemento hijo al DOM.
- **remove()**: Elimina un elemento del DOM.
    - **Ejemplo:**

```
const nuevoParrafo = document.createElement("p");
```

```
nuevoParrafo.textContent = "Este es un párrafo creado
      dinámicamente.";

document.body.appendChild(nuevoParrafo);  // Agrega el nuevo
      párrafo al final del body


nuevoParrafo.remove();  // Elimina el nuevo párrafo
```

Actividad práctica:

1. Crear una lista vacía en HTML y, usando JavaScript, agregar varios elementos de lista dinámicamente.
2. Cambiar el estilo de un párrafo cuando se hace clic en un botón.
3. Alternar una clase en un párrafo cuando se presiona un botón, usando classList.toggle().

---

4. Manejo de Eventos (45 minutos)

Los eventos permiten a JavaScript reaccionar a las acciones del usuario, como hacer clic en un botón, mover el ratón o escribir en un campo de texto.

Agregar eventos a los elementos: El método más común para manejar eventos en JavaScript es addEventListener(). Permite asociar una función que se ejecuta cuando ocurre un evento específico.

Ejemplo básico de un evento click:

```
const boton = document.getElementById("miBoton");

boton.addEventListener("click", () => {

  alert("¡Hiciste clic en el botón!");

});
```

Eventos comunes:

- **click**: Se dispara cuando el usuario hace clic en un elemento.

- **mouseover**: Se dispara cuando el ratón pasa por encima de un elemento.
- **input**: Se dispara cuando el valor de un campo de texto cambia.
- **submit**: Se dispara cuando un formulario es enviado.

Prevención de eventos por defecto: Algunos eventos tienen comportamientos por defecto que pueden ser indeseables, como el envío de un formulario recargando la página. Esto se puede prevenir usando event.preventDefault().

Ejemplo de submit:

```
const formulario = document.getElementById("miFormulario");

formulario.addEventListener("submit", (event) => {

  event.preventDefault();  // Previene el comportamiento por defecto

  alert("Formulario enviado sin recargar la página");

});
```

El objeto event: Cuando ocurre un evento, JavaScript crea un objeto event que contiene información sobre el evento, como el tipo de evento, el elemento que lo generó, y las coordenadas del ratón (si es un evento de ratón).

Ejemplo de event:

```
document.addEventListener("click", (event) => {

  console.log(event.clientX, event.clientY);  // Muestra las coordenadas del clic

});
```

Actividad práctica:

1. Crear un botón que cambie el color de un párrafo cuando se hace clic en él.
2. Implementar un formulario que no recargue la página al enviarlo y que muestre un mensaje de confirmación.
3. Usar el evento mouseover para cambiar el color de fondo de un elemento cuando el ratón pasa por encima.

Actividad Final (10 minutos)

Objetivo: Crear una mini aplicación interactiva usando DOM y eventos.

Descripción del proyecto:

1. Crear un formulario con un campo de texto y un botón de enviar.
2. Al hacer clic en el botón, agregar el texto del campo de texto como un nuevo elemento de lista en la página.
3. Cada vez que se agregue un elemento a la lista, cambiar el color de fondo del elemento

```
const form = document.querySelector("form");
const input = document.querySelector("input");
const lista = document.querySelector("ul");

form.addEventListener("submit", (event) => {
  event.preventDefault();
  const nuevoElemento = document.createElement("li");
  nuevoElemento.textContent = input.value;
  nuevoElemento.style.backgroundColor = "lightblue";
  lista.appendChild(nuevoElemento);
  input.value = "";  // Limpia el campo de texto
});
```

Material adicional para consulta

- MDN Web Docs - DOM
- Event Reference