INDICE

INTRODUCCIÓN

Creación, sustitución y borrado de elementos del DOM.

- Crea una plantilla HTML con el <style> exactamente igual que en el ejercicio anterior y donde ahora, en el , tengas un <textarea cuyo id = “texto” y 5 botones que sean “crearParrafo”,”crearImagen” ,”borrarUltimo” ,”borrarPrimero” y ”sustituirPrimeroVacío”.

Function creaParrafo:

- Crear desde JS un elemento <p> cuyo texto sea el que escribamos en el textarea. Además, añade un atributo class = “miClase”. Este elemento <p> deberá estar contenido en el existente.

Function crearImagen:

- Añadir un elemento img con su atributo src. La ruta de alojamiento la preguntaremos desde un prompt al usuario. De manera análoga a la primera función creada, añadiremos ese hijo a al elemento DIV.

Function borrarUltimo:

- Quita el último elemento que contenga DIV.

Function borrarPrimero:

- Quita el primer elemento que contenga DIV.

Function sustituirPrimeroVacío:

- Crear un párafo P asociandole un texto que se denomine Vacío. En ese momento, busca la manera en que puedas sustituir el primer hijo por este nuevo párrafo que hemos creado.

EJERCICIOS

Creación, sustitución y borrado de elementos del DOM.

Código:

```
<script>
    const textArea = document.createElement('textarea');
    const div = document.createElement('div');

    document.body.append(div);
    div.setAttribute('class', 'container');
    textArea.setAttribute('id', 'texto');
    textArea.setAttribute('rows', '10');
    textArea.setAttribute('cols', '50');
    div.append(textArea);

    const div1 = document.createElement('div');
    div1.setAttribute('class', "buttonContainer")
    div.append(div1)

    const botones = ["CrearParrafo", "CrearImagen", "BorrarUltimo", "BorrarPrimero", "SustituirPrimeroVacio"];
    for (let i = 1; i <= 5; i++) {
        const button = document.createElement('button');
        button.innerText = botones[i - 1];
        button.setAttribute('id', botones[i - 1]);
        button.setAttribute('class', "botones");
        div1.append(button);
    }

    const div2 = document.createElement('div');
    document.body.append(div2);
    div2.setAttribute('class', 'container2');
```

Resultado:

Explicación:

Para la interfaz, el código establece tres variables globales: div, div2 y textArea. Se establece un div como contenedor principal y se incorpora al body; luego, se agrega el textArea con id "texto" y dimensiones fijas para la introducción de información. Después, se establece un arreglo que contiene cinco nombres de botones encargados de controlar la aplicación. Para ubicar los botones de forma visualmente centrada en el formulario, se genera un contenedor intermedio (div1). Un bucle for recorre los nombres del array y genera un botón para cada uno de esos elementos. A cada botón se le asigna un nombre, tanto como texto como id, para relacionar las funciones de JavaScript. Por último, se genera el div de resultados (div2) con la clase container2 y se incorpora al body, quedando todo preparado para la lógica de eventos.

Function creaParrafo

Código:

```
function creaParrafo() {
    const textAreaId = document.getElementById("texto");

    const p = document.createElement('p');
    p.setAttribute('id', 'parrafo');
    p.innerHTML = textAreaId.value;
    p.setAttribute("class", 'miClase');
    div2.append(p);

    if (div2.children.length > 0) div2.style.display = 'block';
    textAreaId.value = '';
}
```

Resultado:

Explicación:

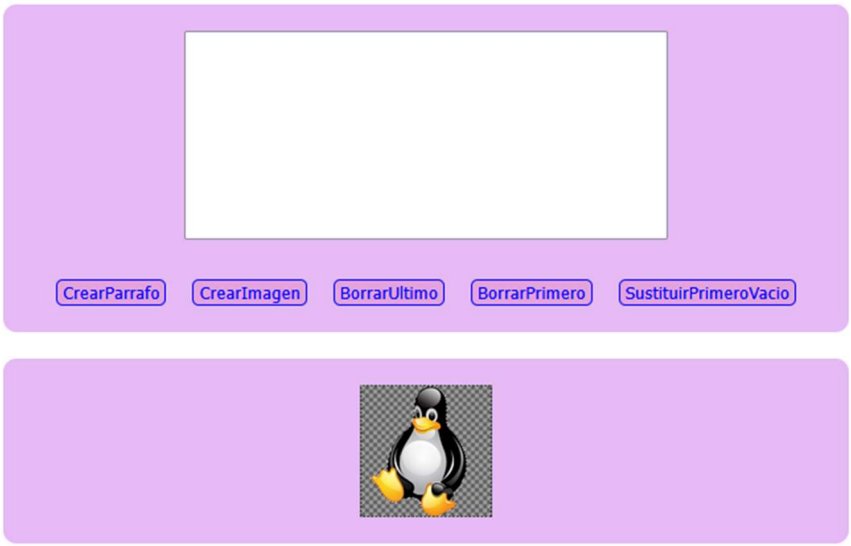
La función creaParrafo() se inicia accediendo al textarea a través de su id para obtener el texto. Cree un nuevo párrafo (p) y se asigna una clase CSS llamada miClase y un id estático. El contenido interno del nuevo párrafo se define con el texto que proporciona el usuario, logrado mediante textAreaId.value. Este párrafo se incluye de inmediato después del contenedor de resultados div2. Por último, se comprueba si div2 tiene hijos para garantizar que sea visible (cambiando su display a block), y luego se limpia el campo de texto de entrada vaciando su value.

Function crearImagen

Código:

```
function crearImagen() {
    const rutaImagen = prompt("Introduzca la url de la imagen: ");
    const imagen = document.createElement('img');
    imagen.setAttribute('src', rutaImagen);
    div2.append(imagen);
    if (div2.children.length > 0) div2.style.display = "block";
}
```

Resultado:



Explicación:

La función crearImagen() pide al usuario que introduzca una URL de imagen a través de un cuadro de diálogo prompt y almacena la entrada. Seguidamente, genera un nuevo elemento <img> y define su atributo src con la URL que el usuario acaba de ingresar. Este nuevo componente de imagen se incorpora al contenedor de resultados, conocido como div2. Por último, el código verifica si div2 contiene algo y, en caso afirmativo, garantiza que se vea el contenedor obligando a su propiedad display a block.

Function borrarUltimo

Código:

```
function borrarUltimo() {
    const ultimoElementoDiv2 = div2.lastElementChild;
    if (ultimoElementoDiv2) div2.removeChild(ultimoElementoDiv2);
    if (div2.children.length === 0) div2.style.display = "none";
}
```

Resultado:

Explicación:

La propiedad lastElementChild es utilizada por la función borrarUltimo() para identificar el último elemento hijo dentro del contenedor div2. Después, verifica si existe un elemento que deba eliminarse; si es así, lo quita del DOM utilizando el método removeChild() del contenedor. Tras la eliminación, el código comprueba si no hay hijos en div2; si es así y el contenedor está vacío, lo oculta por completo al establecer su propiedad display a none.

Function borrarPrimero

Código:

```
function borrarPrimero() {
    const primerElementoDiv2 = div2.firstElementChild;
    if (primerElementoDiv2) div2.removeChild(primerElementoDiv2);
    if (div2.children.length === 0) div2.style.display = "none";
}
```

Resultado:

Explicación:

La función borrarPrimero() se inicia localizando, mediante la propiedad firstElementChild, al primer elemento hijo que se encuentra en el contenedor div2. Después, verifica si hay un elemento que borrar; si existe, lo quita del DOM invocando el método removeChild() del contenedor padre. Luego de la posible eliminación, el código comprueba si div2 no tiene hijos, o sea, si su longitud es cero. Si el contenedor está vacío, la función hace que se oculte completamente al fijar su propiedad display a none.

Function sustituirPrimeroVacío

Código:

```
function sustituirPrimeroVacio() {
    const p = document.createElement("p");
    p.innerText = "Vacío";
    const primerElemento = div2.firstElementChild;
    div2.replaceChild(p, primerElemento);
}

document.getElementById("CrearParrafo").addEventListener("click", creaParrafo);
document.getElementById("CrearImagen").addEventListener("click", crearImagen);
document.getElementById("BorrarUltimo").addEventListener("click", borrarUltimo);
document.getElementById("BorrarPrimero").addEventListener("click", borrarPrimero);
document.getElementById("SustituirPrimeroVacio").addEventListener("click", sustituirPrimeroVacio);
```

Resultado:

Explicación:

La función sustituirPrimeroVacio() se inicia al crear un nuevo párrafo (p) y definir el texto que contiene como "Vacio". Después, la función consigue el nodo que es en este momento el primer elemento hijo dentro del contenedor div2. Luego, emplea el método replaceChild() del contenedor div2 para sustituir el primer elemento hallado por el nuevo párrafo "Vacio". Por último, las cinco líneas finales del código asocian los manejadores de eventos click a los cinco botones de control, conectando cada uno con su función correspondiente para manipular el DOM y que así el programa funcione.

BIBLIOGRAFIA

Para hacer este ejercicio acerca de la manipulación dinámica del DOM con JavaScript, revisé la documentación oficial de MDN Web Docs, que explica claramente los procedimientos para crear, alterar y quitar elementos HTML. Algunos ejemplos de estos procedimientos son createElement(), appendChild(), removeChild() y replaceChild().
Enlace: https://developer.mozilla.org/es/docs/Web/API/Document/createElement

También consulté las guías de W3Schools, que me sirvieron mucho para comprender cómo emplear addEventListener() para fortalecer el uso de eventos y cómo utilizar setAttribute() para acceder a los atributos y modificarlos.
Enlace: https://www.w3schools.com/js/js_htmldom.asp

Asimismo, utilicé la documentación de MDN acerca del modelo de eventos en JavaScript, que explica cómo agregar controladores y cómo interactuar eficazmente con el DOM.
Enlace: https://developer.mozilla.org/es/docs/Learn/JavaScript/Building_blocks/Events

Finalmente, se emplearon los recursos del Manual de HTML y CSS de Educamadrid para reforzar la estructura de estilos, el diseño del contenedor y la disposición visual de los componentes generados dinámicamente.
Enlace: https://www.educa2.madrid.org/web/educamadrid