Material descargable

Javascript: Conceptos básicos

A continuación te dejamos los conceptos básicos que te ayudarán a realizar los ejercicios de los próximos pasos:

¿Qué es el Document Object Model (DOM)?

El Document Object Model (DOM) es una interfaz de programación que representa los documentos HTML y XML como una estructura de árbol. Permite a los lenguajes de programación, como JavaScript, acceder y manipular el contenido, la estructura y el estilo de las páginas web. Las características clave son:

Estructura de Árbol: El DOM organiza los elementos de una página web en una jerarquía de árbol. Cada elemento, atributo y texto se convierte en un nodo de este.

Independiente del Lenguaje: Aunque se usa comúnmente con JavaScript, el DOM es independiente del lenguaje de programación.

Dinámico: Los cambios realizados en el DOM mediante scripts se reflejan inmediatamente en la visualización de la página web.

Interactivo: Permite la interacción del usuario y los scripts, facilitando tareas como la captura de eventos o la modificación de elementos en la página.

Estandarizado: El World Wide Web Consortium (W3C) le da soporte, asegurando la compatibilidad y consistencia entre diferentes navegadores y plataformas.

En JavaScript se puede acceder al `DOM` mediante la variable global `document`.

Una vez aclarado qué es el DOM, dejaremos algo en claro: Existe una propiedad de cada elemento llamada `innerHTML`, la cual modifica el HTML del mismo y no es considerada buena práctica. Usaremos la propiedad `textContent`, la cual solo permite texto junto con el método `createElement`.

En los siguientes pasos usaremos `métodos` para obtener y/o modificar los elementos del DOM.

createElement: Es un método en JavaScript utilizado para crear un nuevo elemento HTML. Este método es parte del objeto `document` y permite generar dinámicamente elementos en una página web.

Por ejemplo, para crear un nuevo elemento div, usarías el siguiente código:

```
Unset
const nuevoDiv = document.createElement('div');
```

querySelector(): `Retorna` el primer elemento que coincida con un grupo específico de `selectores CSS` dentro del documento. Por ejemplo, si quieres seleccionar el primer elemento con la clase `.my-class` en un documento HTML, deberías utilizar `document.querySelector('.my-class')`. Este método es útil para manipular elementos del DOM basado en sus atributos de estilo, identificadores, clases, etc., permitiendo una interacción dinámica en páginas web.

getElementById: Es otra forma de seleccionar elementos en un documento HTML. Este método también forma parte del objeto `document` y permite acceder a un elemento específico por su atributo id. El id de un elemento es único dentro de una página, por lo que `getElementById` siempre devuelve como máximo un solo elemento.

.classList.add(): Se utiliza para agregar una o más clases CSS a un elemento del DOM. Es parte de la propiedad `classLis`t, que proporciona una forma conveniente de acceder y modificar las clases de un elemento.

Por ejemplo, si tienes un elemento HTML con el id `myElemen`t y quieres añadirle la clase CSS `new-class`, usarías el siguiente código:

```
Unset
document.getElementById('myElement').classList.add('new-class'
);
```

También se puede agregar más de una clase al mismo tiempo:

```
Unset
document.getElementById('myElement').classList.add('class1',
'class2', 'class3');
```

appendChild: Se usa para agregar un nuevo nodo al final de los hijos de un elemento padre especificado.

```
Unset
// Crear un nuevo elemento
const nuevoElemento = document.createElement("div");
nuevoElemento.textContent = "¡Hola Mundo!";

// Seleccionar el elemento padre
const padre = document.getElementById("elementoPadre");

// Añadir el nuevo elemento al padre
padre.appendChild(nuevoElemento);
```

Antes:

```
Unset
  <div id="elementoPadre">
    <!-- hijos existentes -->
  </div>
```

Después:

```
Unset
  <div id="elementoPadre">
    <!-- hijos existentes -->
    <div>
      ¡Hola Mundo!
    </div>
  </div>
```

insertAdjacentElement: Se usa para insertar un elemento en una posición especificada relativa a otro elemento. Tiene dos argumentos:

- **Posición**: Una cadena que especifica dónde insertar el elemento. Las opciones son:
    - "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.
- Elemento a Insertar: El elemento que deseas insertar.

```
Unset
// Crear un nuevo elemento
const nuevoElemento = document.createElement("span");
nuevoElemento.textContent = "Nuevo texto";

// Seleccionar el elemento de referencia
const elementoReferencia =
document.getElementById("elementoExistente");

// Insertar el nuevo elemento antes del elemento de referencia
elementoReferencia.insertAdjacentElement("beforebegin",
nuevoElemento);
```

Antes:

```
Unset
<div id="contenedor">
  <div id="elementoExistente">Elemento Existente</div>
</div>
```

Después:

```
Unset
<div id="contenedor">
  <span>Nuevo texto</span>
  <div id="elementoExistente">Elemento Existente</div>
</div>
```