Manipulando el DOM con Vanilla-JS

Antes de empezar con este capítulo, quiero aclarar que Vanilla-JS es un término que se utiliza para referirse a JavaScript puro, es decir, sin utilizar librerías o frameworks.

Para manipular el DOM con Vanilla-JS, vamos a vincular un archivo JavaScript a nuestro documento HTML. Para ello, vamos a utilizar la etiqueta `<script>`.

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1>DOM</h1>
    <h2>El DOM es una representación del documento HTML.</h2>
    <script src="main.js"></script>
  </body>
</html>
```

En este ejemplo, el archivo `main.js` se encuentra en la misma carpeta que el archivo HTML. Sin embargo, también podemos utilizar una ruta relativa para vincular el archivo JavaScript.

```
<script src="js/main.js"></script>
```

Es decir que la estructura de nuestro proyecto sería la siguiente:

```
.
├── index.html
└── js
    └── main.js
```

Acceder al DOM desde JavaScript

Para acceder al DOM desde JavaScript, vamos a utilizar el objeto `document`. Este objeto representa el documento HTML y nos permite acceder a todos los elementos del DOM.

```
console.log(document);
```

lo que veremos por consola será lo siguiente:

```
# Document
```

Si desplegamos el objeto `document`, veremos que tiene una estructura similar a la siguiente:

```
# Document
  # head
  # body
    # h1
    # h2
    # script
```

Es decir que dispondremos de una propiedad para acceder a cada elemento del DOM como este caso, el h1 o el h2.

Capturando elementos del DOM

Para capturar elementos del DOM, existen varios métodos que nos permiten acceder a los elementos del DOM. En este capítulo, vamos a ver los siguientes métodos:

- getElementById
- getElementsByTagName
- getElementsByClassName
- querySelector
- querySelectorAll

getElementById

La forma más sencilla de acceder a un elemento del DOM es utilizando el método `getElementById`. Este método recibe como parámetro el valor del atributo `id` del elemento que queremos seleccionar.

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1 id="title">DOM</h1>
    <h2>El DOM es una representación del documento HTML.</h2>
    <script src="main.js"></script>
  </body>
</html>
```

```
const title = document.getElementById('title');
console.log(title);
```

lo que veremos por consola será lo siguiente:

```
# <h1 id="title">DOM</h1>
```

getElementsByTagName

El método `getElementsByTagName` nos permite acceder a todos los elementos que coincidan con el nombre de la etiqueta que pasamos como parámetro.

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1 id="title">DOM</h1>
    <h2>El DOM es una representación del documento HTML.</h2>
    <script src="main.js"></script>
  </body>
</html>
```

```
const headings = document.getElementsByTagName('h2');
console.log(headings);
```

lo que veremos por consola será lo siguiente:

```
# HTMLCollection(1) [h2]
```

getElementsByClassName

El método `getElementsByClassName` nos permite acceder a todos los elementos que coincidan con el nombre de la clase que pasamos como parámetro.

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1 id="title">DOM</h1>
    <h2 class="subtitle">El DOM es una representación del documento HTML.
</h2>
    <script src="main.js"></script>
  </body>

</html>
```

```
const subtitles = document.getElementsByClassName('subtitle');
console.log(subtitles);
```

lo que veremos por consola será lo siguiente:

```
# HTMLCollection(1) [h2.subtitle]
```

querySelector

El método `querySelector` nos permite acceder al primer elemento que coincida con el selector que pasamos como parámetro. Este selector puede ser el nombre de una etiqueta, el valor de un atributo `id` o el valor de un atributo `class`.

Para indicar si lo que queremos seleccionar es una etiqueta, un `id` o una clase, debemos utilizar los siguientes selectores:

- Para etiquetas: `nombreDeEtiqueta`
- Para id: `#nombreDeId`
- Para clases: `.nombreDeClase`

Esto probablemente te suene familiar si has utilizado CSS ya que los selectores son los mismos.

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1 id="title">DOM</h1>
    <h2 class="subtitle">El DOM es una representación del documento HTML.
</h2>
    <script src="main.js"></script>
  </body>
</html>
```

```
const title = document.querySelector('#title');
console.log(title);
```

lo que veremos por consola será lo siguiente:

```
# <h1 id="title">DOM</h1>
```

```
const subtitle = document.querySelector('.subtitle');
console.log(subtitle);
```

lo que veremos por consola será lo siguiente:

```
# <h2 class="subtitle">El DOM es una representación del documento HTML.
</h2>
```

querySelectorAll

El método `querySelectorAll` nos permite acceder a todos los elementos que coincidan con el selector que pasamos como parámetro. Este selector puede ser el nombre de una etiqueta, el valor de un atributo `id` o el valor de un atributo `class`.

En la mayoría de los casos, usamos esto para seleccionar elementos que comparten la misma clase.

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1 id="title">DOM</h1>
    <h2 class="subtitle">El DOM es una representación del documento HTML.
</h2>
    <h2 class="subtitle">El DOM es una representación del documento HTML.
</h2>
    <script src="main.js"></script>
  </body>
</html>
```

```
const subtitles = document.querySelectorAll('.subtitle');
console.log(subtitles);
```

lo que veremos por consola será lo siguiente:

```
# NodeList(2) [h2.subtitle, h2.subtitle]
```

Modificando elementos del DOM

Una vez que tenemos acceso a los elementos del DOM, podemos modificarlos utilizando las propiedades que nos ofrece cada elemento.

Modificando el contenido de un elemento

Para modificar el contenido de un elemento, podemos utilizar la propiedad `innerHTML`. Esta propiedad nos permite acceder al contenido HTML de un elemento.

Vamos a modificar el contenido del elemento `h1` del siguiente ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1 id="title">DOM</h1>
    <h2 class="subtitle">El DOM es una representación del documento HTML.
</h2>
    <h2 class="subtitle">El DOM es una representación del documento HTML.
</h2>
    <script src="main.js"></script>
  </body>
</html>
```

```
const title = document.querySelector('#title');
title.innerHTML = 'DOM - Document Object Model';
```

lo que veremos en el navegador será lo siguiente:

```
<h1 id="title">DOM - Document Object Model</h1>
```

Como podemos observar en el ejemplo anterior, la propiedad `innerHTML` nos permite modificar el contenido HTML de un elemento.

Modificando el texto de un elemento

Para modificar el texto de un elemento, podemos utilizar la propiedad `textContent`. Esta propiedad nos permite acceder al contenido de texto de un elemento.

Vamos a modificar el texto del elemento `h1` del siguiente ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1 id="title">DOM</h1>
    <h2 class="subtitle">El DOM es una representación del documento HTML.
</h2>
    <h2 class="subtitle">El DOM es una representación del documento HTML.
</h2>
    <script src="main.js"></script>
  </body>
</html>
```

```
const title = document.querySelector('#title');
title.textContent = 'DOM - Document Object Model';
```

lo que veremos en el navegador será lo siguiente:

```
<h1 id="title">DOM - Document Object Model</h1>
```

Como podemos observar en el ejemplo anterior, la propiedad `textContent` nos permite modificar el contenido de texto de un elemento. La diferencia con la propiedad `innerHTML` es que `textContent` no nos permite modificar el contenido HTML de un elemento, mientras que `innerHTML` sí.

Modificando el valor de un atributo

Para modificar el valor de un atributo, podemos utilizar la propiedad `setAttribute`. Esta propiedad nos permite modificar el valor de un atributo de un elemento.

Vamos a modificar el valor del atributo `href` del elemento `a` del siguiente ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <a href="https://www.google.com">Google</a>
    <script src="main.js"></script>
  </body>
</html>
```

```
// Seleccionamos el elemento a
const link = document.querySelector('a');

// Modificamos el valor del atributo href
link.setAttribute('href', 'https://www.youtube.com');
```

lo que veremos en el navegador será lo siguiente:

```
<a href="https://www.youtube.com">Google</a>
```

Como podemos observar en el ejemplo anterior, la propiedad `setAttribute` nos permite modificar el valor de un atributo de un elemento. En este caso, modificamos el valor del atributo `href` del elemento `a`. Por lo cual, el enlace que antes nos llevaba a Google, ahora nos lleva a YouTube.

Modificando el estilo de un elemento

Para este ejemplo agregaremos un archivo CSS para darle estilo a nuestro documento HTML.

La estructura de nuestro repositorio será la siguiente:

```
.
├── index.html
├── js
│   └── main.js
└── styles
    └── main.css
```

Para vincular el archivo CSS a nuestro documento HTML, debemos agregar la siguiente etiqueta en el `head` del documento HTML.

```
<link rel="stylesheet" href="styles/main.css" />
```

Para modificar el estilo de un elemento, podemos utilizar la propiedad `style`. Esta propiedad nos permite modificar el estilo de un elemento.

En el siguiente ejemplo tendremos un div que representa un cuadrado de color rojo. Vamos a cambiar la forma del cuadrado a un círculo y el color a azul.

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="styles/main.css" />
    <title>DOM</title>
  </head>
  <body>
    <div id="square"></div>
    <script src="main.js"></script>
  </body>
</html>
```

styles/main.css

```
#square {
  width: 100px;
  height: 100px;
  background-color: red;
  border-radius: 0;
}
```

js/main.js

```
// Seleccionamos el elemento div
const square = document.querySelector('#square');

// Modificamos el estilo del elemento div
square.style.borderRadius = '50%';
square.style.backgroundColor = 'blue';
```

lo que veremos en el navegador será lo siguiente:

```
<div id="square" style="border-radius: 50%; background-color: blue;">
</div>
```

Como podemos observar en el ejemplo anterior, la propiedad `style` nos permite modificar el estilo de un elemento. En este caso, modificamos el estilo del elemento `div`.