JavaScript II - HAB (https://remoto.hackaboss.com/trainer/course/id:134)

3. DOM II

Modificando atributos de los elementos

Podemos acceder a los atributos de los elementos usando varias propiedades:

- **getAttribute:** muestra el valor de un atributo.
- **setAttribute:** establece el valor de un atributo.
- **hasAttribute:** comprueba si existe un atributo.
- **removeAttribute:** elimina un atributo.

```
// <button class="boton" type="submit">Enviar</button>
const el = document.querySelector('button');

// Imprime en la consola el valor "submit".
console.log(el.getAttribute('type'));

// Hace que el botón no sea clicable.
el.setAttribute('disabled', true);

// Comproba si el elemento contiene el atributo "class".
el.hasAttribute('class'); //true

// Vuelve a activar el botón.
el.removeAttribute('disabled');
```

En caso de atributos de tipo **data** (que nos permitian añadir atributos personalizados a los elementos) podemos acceder a ellos de manera más sencilla:

```
// <li data-time="23:45" data-username="root">Texto</li>
const el = document.querySelector('ul.messages li:first-child');

console.log(el.dataset.userbane); // root
console.log(el.dataset.time); // 23:45

// Cambia el valor del atributo data-username a 'guest'
el.dataset.username = 'guest';
```

Modificando el CSS de los elementos

La propiedad **style** de un **Element** nos permite ver y establecer de forma rápida propiedades de CSS de un elemento. Podemos hacerlos de varias formas:

```
const el = document.getElementById('titulo');

// Imprime en la consola las propiedades de CSS dun elemento establecidas en el a
tributo "style".
console.log(el.style);
console.log(el.style.backgroundColor);

// Establece varias propiedades de una sola vez.
el.style.cssText = 'text-decoration: none; background: gold';

// Lo mismo que el anterior usando la propiedad "setAttribute".
el.setAttribute('style', 'text-decoration: none; background: gold');

// establece una a una.
el.style.textDecoration = 'none';
el.style.background = 'gold';
```

Estas propiedades muestran y establecen el estilo en el atributo **style** del elemento y no al estilo recibido por CSS especificado en hojas de estilo externas o etiqueta **<style>**. Veremos más adelante como manejar estilos externos.

Para ver todas las propiedades de estilo que afectan a un elemento usamos un método del objeto **window**:

```
const el = document.getElementById('titulo');
const allStyle = window.getComputedStyle(el);
```

Este método nos devuelve un objeto de tipo **CSSStyleDeclaration** que podemos modificar usando estos métodos (https://developer.mozilla.org/en-US/docs/Web/API/CSSStyleDeclaration).

Modificando las clases CSS de los elementos

Aún que mediante la anterior propiedad **style** podemos modiciar facilmente el estilo del elemento, si tenemos un CSS bien escrito, la mayoría de las veces lo que buscaremos será modificar las classes que le afectan a un elemento. Para esto podemos usar la propiedad **classList** y los métodos que proporciona.

```
// <p class="importante inicial">lorem ipsum</p>
const el = document.querySelector('.importante');

el.classList.add('destacado'); // Añade la clase ".destacado" al elemento.
el.classList.remove('importante'); //Le quita la clase ".importante" al elemento.
el.classList.togggle('importante'); // Le quita la clase ".importante" al elemento si la tiene o se la añade si no la tiene.
el.classList.contains('inicial'); // Devuelve true si el elemento tiene la clase ".inicial" y false si no la tiene.
el.classList.replace('importante', 'secundario'); // Reemplaza la clase ".importante" por ".secundario".
```

Creando, borrando y sustituyendo elementos

El método **createElement** del objeto **document** nos permite crear nuevos elementos. A estos elementos podemos añadirles otros usando el método **appendChild** o añadirles texto usando el método de document **createTextNode**.

```
// Seleccionamos el body.
const body = document.querySelector('body');

// Creamos un elemento de tipo "ul".
const list = document.createElement('ul');

// Creamos un elemento de tipo "li".
const item = document.createElement('li');

// Creamos un nodo de texto con el valor "Texto de la lista";
const itemContent = document.createTextNode('Texto de la lista');

// Añadimos el nodo de texto al elemento de tipo "li" creado anteriormente.
item.appendChild(itemContent);

// Añadimos el elemento "li" a la lista.
list.appendChild(item);

// Añadimos la lista al body.
body.appendChild(list);

/* Resultado:
  <body>
    <ul>
      <li>Texto de la lista</li>
    </ul>
  </body>
*/
```

appendChild añade el elemento al final de la lista de hijos del elemento padre. Si queremos añadirlo a otra posición podemos usar el método **insertBefore**:

```
/* Tenemos este HTML:
  <ul>
    <li class="uno"></li>
    <li class="dous"></li>
    <li class="tres"></li>
  </ul>
*/

// Seleccionamos la lista.
const list = document.querySelector('ul');

// Creamos un nuevo elemento de la lista y le asignamos un texto.
const newItem = document.createElement('li');
newItem.textContent = 'antes de dos';
newItem.classList.add('before-two');

// Seleccionamos el segundo elemento de la lista.
const secondItem = list.querySelector('.dos');

// Añadimos el nuevo elemento antes del segundo elemento.
list.insertBefore(newItem, secondItem);

/* Resultado:
  <ul>
    <li class="uno"></li>
    <li class="before-two">antes de dos</li>
    <li class="dous"></li>
    <li class="tres"></li>
  </ul>
*/
```

Hay otros métodos para añadir elementos:

- **insertAdjacentElement:** permite insertar un elemento en diferentes posiciones https://developer.mozilla.org/en-US/docs/Web/API/Element/insertAdjacentElement.

- Los métodos modernos **append (https://developer.mozilla.org/en-US/docs/Web/API/ParentNode/append)** y **prepend (https://developer.mozilla.org/en-US/docs/Web/API/ParentNode/prepend):** permiten añadir uno o más elementos al principio y final de la lista de hijos del padre.

Para eliminar un elemento podemos user el método **removeChild** del padre:

```
/*
  <ul>
    <li>uno</li>
    <li>dos</li>
  </ul>
*/

const lastItem = document.querySelector('ul li:last-child');
const list = lastItem.parentElement;

// Elimina el "lastItem".
list.removeChild(lastItem);
```

Una forma más moderna de hacerlo es usar directamente el método **.remove( )** del propio elemento:

```
/*
  <ul>
    <li>uno</li>
    <li>dos</li>
  </ul>
*/

const lastItem = document.querySelector('ul li:last-child');

// Elimina el "lastItem".
lastItem.remove();
```

Completado. Continuamos.