Recuerden poner a grabar la clase

Clase 7

DOM ||

Diplomatura UNTREF

Temario

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

- Metodo getElementByClassName
- Metodo getElementsByTagName
- Template string
- Template Literal
- Manejo de atributos de un HTMLElement (className - ClassList add y remove)

Metodo getElementByClassName

Metodo getElementByClassName

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

El método **getElementsByClassName** es una función de JavaScript que nos permite seleccionar elementos HTML por su clase.

Su propósito principal es ayudarnos a obtener una colección de elementos que comparten una misma clase CSS.

Es importante destacar que este método devuelve una colección de elementos (tipo HTMLCollection), no un único elemento.

Sintaxis y uso

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

La sintaxis básica del método **getElementsByClassName** es la siguiente:

document.getElementsByClassName("nombre-de-clase").

En este caso, "nombre-de-clase" representa el nombre de la clase CSS que queremos seleccionar.

La búsqueda se realiza dentro del documento HTML o dentro de un elemento específico, si se proporciona uno como contexto.

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de getElementsByClassName</title>
</head>
<body>
    <div class="destacado">Elemento 1</div>
    <div class="destacado">Elemento 2</div>
    <div class="normal">Elemento 3</div>
    <div class="destacado">Elemento 4</div>
    <script>
        const elementosDestacados = document.getElementsByClassName("destacado");
        console.log(elementosDestacados);
    </script>
</body>
</html>
```

Acceso y Manipulacion de Elementos

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

Para acceder a los elementos seleccionados por **getElementsByClassName**, podemos utilizar un bucle (como for o forEach) o acceder directamente por índice.

Los elementos seleccionados se comportan como una colección similar a un array (tipo HTMLCollection).

Podemos manipular propiedades y contenido de los elementos, cambiar estilos, agregar eventos, etc.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Acceso y Manipulación de Elementos</title>
</head>
<body>
    <div class="destacado">Elemento 1</div>
    <div class="destacado">Elemento 2</div>
    <div class="normal">Elemento 3</div>
    <div class="destacado">Elemento 4</div>
    <script>
        const elementosDestacados = document.getElementsByClassName("destacado");
        for (let i = 0; i < elementosDestacados.length; i++) {
            elementosDestacados[i].style.color = "red";
        }
    </script>
</body>
</html>
```

Combinación con otros métodos

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

Podemos combinar **getElementsByClassName** con otros métodos, como querySelector, para realizar selecciones más específicas.

Esto nos permite seleccionar elementos con una clase específica dentro de un contenedor con un ID o una etiqueta específica.

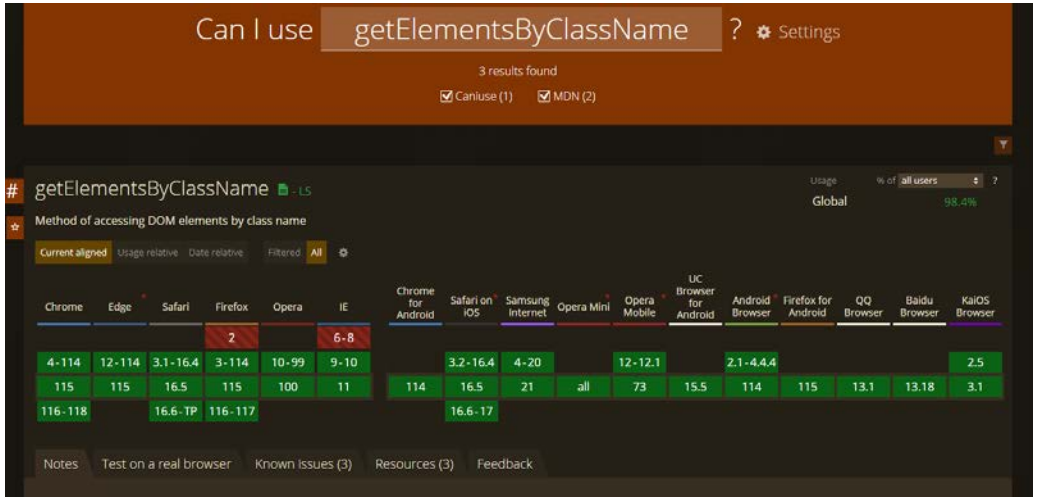
```html
<!DOCTYPE html>
<html>
<head>
    <title>Combinando con otros Métodos</title>
</head>
<body>
    <div id="contenedor">
        <div class="destacado">Elemento 1</div>
        <div class="destacado">Elemento 2</div>
    </div>
    <div class="destacado">Elemento 3</div>
    <div class="destacado">Elemento 4</div>
    <script>
        const contenedorDestacados =
document.getElementById("contenedor").getElementsByClassName("destacado");
        console.log(contenedorDestacados);
    </script>
</body>
</html>
```

Combinación con otros métodos

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

Es importante tener en cuenta la compatibilidad del método getElementsByClassName con diferentes navegadores.

En general, este método es compatible con la mayoría de los navegadores modernos, pero algunos navegadores más antiguos pueden tener limitaciones.



Metodo getElementByTagName

Método getElementByTagNam

El método getElementsByTagName es una función de JavaScript que nos permite seleccionar elementos HTML por su etiqueta.

Su objetivo principal es obtener una colección de elementos que comparten una misma etiqueta, como "div", "p", "h1", etc.

Es importante destacar que este método también devuelve una colección de elementos, no un único elemento.

Combinación con otros metodos

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

Podemos combinar **getElementsByClassName** con otros métodos, como querySelector, para realizar selecciones más específicas.

Esto nos permite seleccionar elementos con una clase específica dentro de un contenedor con un ID o una etiqueta específica.

Uso básico

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

Supongamos que deseamos obtener todos los elementos de la etiqueta <p> en la página.

La función nos devuelve una colección de elementos que coinciden con la etiqueta <p>.

```
<!DOCTYPE html>
<html>
<head>
  <title>Uso básico de getElementByTagName</title>
</head>
<body>
  <p>Primer párrafo</p>
  <p>Segundo párrafo</p>
  <p>Tercer párrafo</p>

  <script>
    const elementosP = document.getElementsByTagName('p');
    console.log(elementosP); // Mostrará una colección con los tres elementos <p>
  </script>
</body>
</html>
```

Iterar sobre una colección de elementos

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

Ahora que tenemos nuestra colección de elementos, ¿qué podemos hacer con ella?

Podemos recorrerla usando un bucle for para realizar una acción en cada elemento.

```html
<!DOCTYPE html>
<html>
<head>
  <title>Iterar sobre una colección de elementos</title>
</head>
<body>
  <p>Primer párrafo</p>
  <p>Segundo párrafo</p>
  <p>Tercer párrafo</p>

  <script>
    const elementosP = document.getElementsByTagName('p');
    for (let i = 0; i < elementosP.length; i++) {
      elementosP[i].textContent = '¡Texto cambiado!';
    }
  </script>
</body>
</html>
```

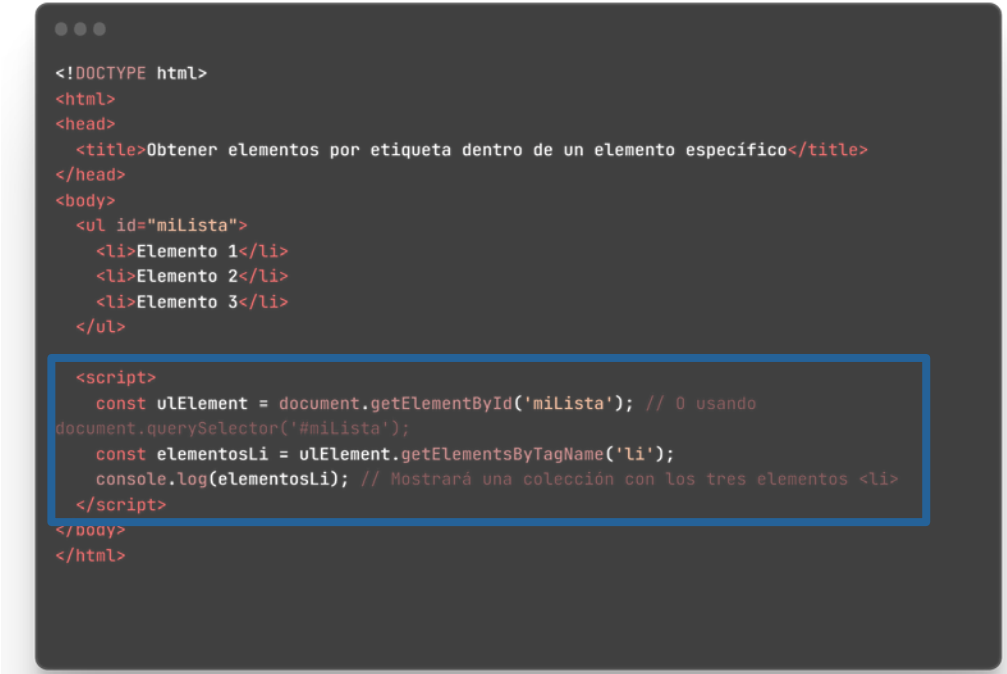
Obtener elementos por etiqueta dentro de un elemento específico

UNTREF
UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

A veces, queremos buscar elementos por etiqueta solo dentro de un área específica del DOM.

Para ello, podemos combinar **getElementsByTagName** con **getElementById** o **querySelector.**

Veamos cómo obtener todos los elementos <li> dentro de una lista no ordenada (<ul>) con un identificado



Template Strings/ template Literals

Template Strings

Los **Template Strings**, también conocidos como "**plantillas de cadenas**", son una característica introducida en ECMAScript 6 (ES6) para mejorar la forma en que manejamos cadenas de texto en JavaScript.

Los **Template Strings** se crean utilizando comillas invertidas (`) en lugar de comillas simples o dobles.

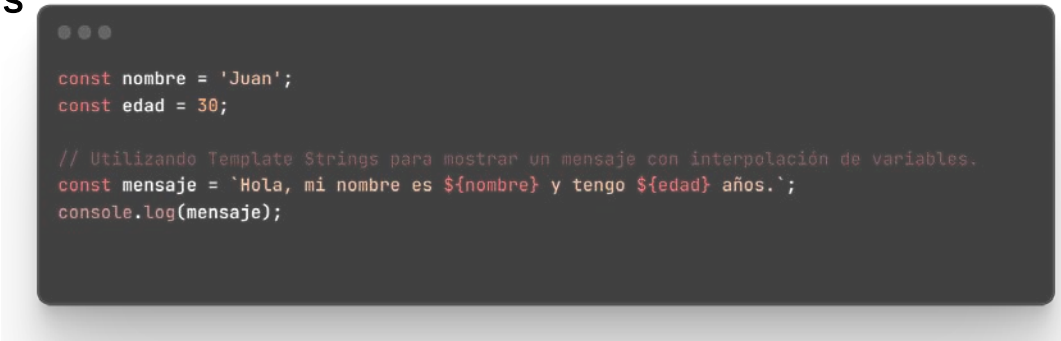
Esto nos permite incluir expresiones y variables dentro de las cadenas de forma más sencilla y legible.

Interpolación de variables con Template Strings

La principal ventaja de los **Template Strings** es la interpolación de variables.

Para insertar una variable dentro de un **Template String**, simplemente se coloca la variable entre llaves **(${variable}).**

El contenido de la variable se evalúa y se sustituye en el lugar correspondiente dentro de la cadena.

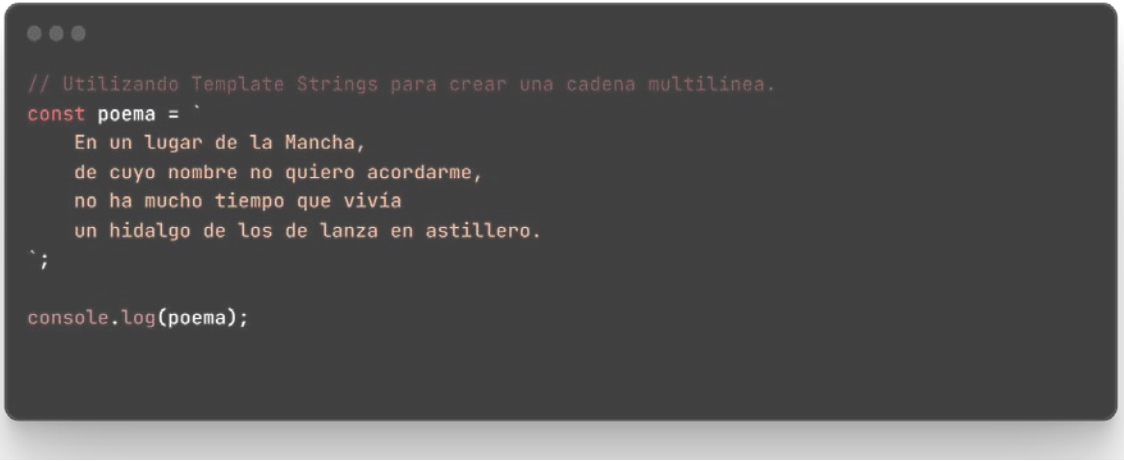


```
const nombre = 'Juan';
const edad = 30;

// Utilizando Template Strings para mostrar un mensaje con interpolación de variables.
const mensaje = `Hola, mi nombre es ${nombre} y tengo ${edad} años.`;
console.log(mensaje);
```

Multilínea con Template Strings

Otro beneficio de los **Template Strings** es la capacidad de crear cadenas multilínea de forma más sencilla.

En lugar de usar caracteres especiales como **"\n"** para representar saltos de línea, simplemente **se pueden agregar líneas adicionales** dentro del Template String.



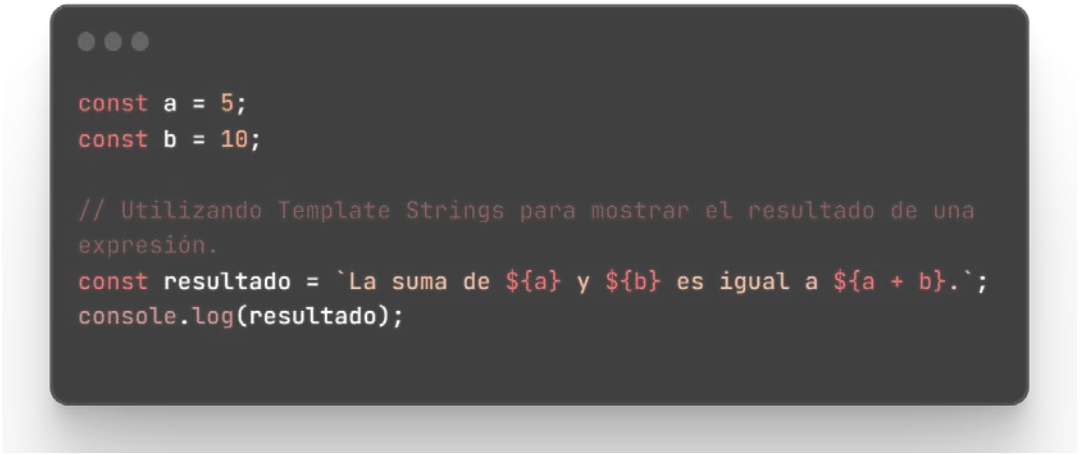
```
// Utilizando Template Strings para crear una cadena multilínea.
const poema = `
    En un lugar de la Mancha,
    de cuyo nombre no quiero acordarme,
    no ha mucho tiempo que vivía
    un hidalgo de los de lanza en astillero.
`;

console.log(poema);
```

Expresiones en Template Strings

Además de variables, también podemos incluir expresiones dentro de los **Template Strings.**

Las expresiones son evaluadas y su resultado es incorporado en la cadena.

Esto es útil para realizar operaciones aritméticas, llamadas a funciones, entre otras cosas.



```
const a = 5;
const b = 10;

// Utilizando Template Strings para mostrar el resultado de una expresión.
const resultado = `La suma de ${a} y ${b} es igual a ${a + b}.`;
console.log(resultado);
```

Manejo de atributos de un HTMLElement

(className - ClassList add y remove)

¿Qué son los atributos HTMLElements?

Los atributos de un **HTMLElement** son metadatos que describen y definen características específicas de un elemento HTML.

Cada elemento puede tener varios atributos que influyen en su comportamiento, apariencia o interacción con el usuario.

Algunos atributos comunes incluyen **id, class, src, href, style**, entre otros.

Acceso a los atributos de un HTMLElement

Para acceder a los atributos de un **HTMLElement**, podemos utilizar varias opciones en JavaScript.

Podemos utilizar la propiedad **getAttribute()** para obtener el valor de un atributo específico.

También podemos acceder a los atributos directamente a través de las propiedades del elemento, como **id, className, src, etc.**

```
// Obtener el valor de un atributo usando getAttribute()
const miElemento = document.getElementById('miElemento');
const valorSrc = miElemento.getAttribute('src');

// Acceder al atributo directamente usando la propiedad
const valorId = miElemento.id;
```

Conoceremos cómo trabajar con el atributo **className** y los métodos **classList.add()** y **classList.remove()** para agregar y eliminar clases de un elemento.

Atributos HTMLElement = className

El atributo **className** es una propiedad de los elementos del **DOM** que permite obtener y establecer la lista de clases de un elemento como una sola cadena de texto separada por espacios.

Representa todas las clases aplicadas a un elemento y es una de las formas más antiguas y sencillas de manipular las clases de un HTMLElement.

Atributos HTMLElement = className

El atributo **className** tiene algunas limitaciones, como que reemplaza todas las clases existentes en el elemento. Por esta razón, se introdujo el uso de la propiedad **classList**, que proporciona métodos más potentes para trabajar con las clases de un **HTMLElement**.

```html
<!-- HTML -->
<div id="miDiv" class="rojo borde-grueso"></div>
```

```javascript
// JavaScript
const miElemento =
document.getElementById('miDiv');

// Accediendo al atributo className
console.log(miElemento.className); // Output:
"rojo borde-grueso"

// Modificando el atributo className
miElemento.className = "azul borde-fino";
console.log(miElemento.className); // Output:
"azul borde-fino"
```

Atributos HTMLElement = classList.add()

El método **classList.add()** es parte del objeto **classList** y se utiliza para agregar una o varias clases a un elemento. Es una forma conveniente y más moderna de agregar clases sin eliminar las ya existentes.

Puede recibir múltiples parámetros de clase separados por comas o una lista de clases en forma de array.

```html
<!-- HTML -->
<div id="miDiv" class="rojo"></div>
```

```javascript
// JavaScript
const miElemento =
document.getElementById('miDiv');

// Agregando clases usando classList.add()
miElemento.classList.add('borde-grueso',
'redondeado');
// El elemento ahora tiene las clases: "rojo
borde-grueso redondeado"
```

Atributos HTMLElement = classList.remove()

El método **classList.remove()** es otra función del objeto **classList**, que se utiliza para eliminar una o varias clases de un elemento. Al igual que **classList.add()**, puede recibir múltiples parámetros de clase o una lista de clases en forma de array.

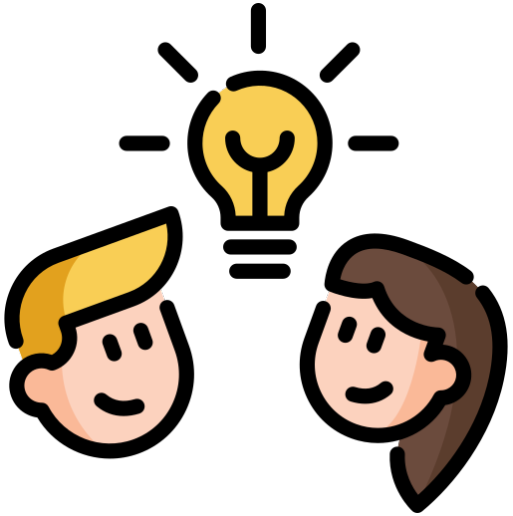
```html
<!-- HTML -->
<div id="miDiv" class="rojo borde-grueso
redondeado"></div>
```

```javascript
// JavaScript
const miElemento = document.getElementById('miDiv');

// Eliminando clases usando classList.remove()
miElemento.classList.remove('rojo', 'borde-grueso');
// El elemento ahora tiene las clases: "redondeado"
```

Consideraciones

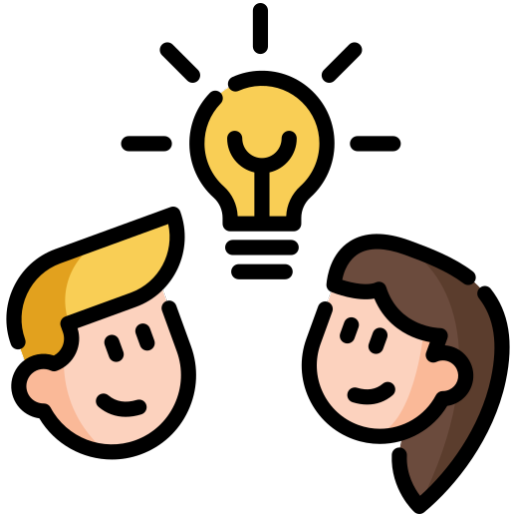
Estos métodos (**classList.add()** y **classList.remove()**) son especialmente útiles cuando deseamos agregar o eliminar clases dinámicamente según ciertas condiciones o eventos en nuestra aplicación web.

Además, **classList** también proporciona otros métodos útiles, como **toggle()** para alternar la presencia de una clase y **contains()** para verificar si un elemento tiene una clase específica.



Consideraciones

Recuerda que el uso de **classList** es preferible cuando trabajas con clases en el **DOM**, ya que proporciona una interfaz más segura y conveniente para manejar las clases de un HTMLElement, especialmente en comparación con el atributo **className**.



Método togle()

El método toggle() se utiliza para alternar la presencia de una clase en un elemento. Si la clase ya está presente en el elemento, toggle() la eliminará; de lo contrario, si la clase no está presente, toggle() la agregará. Esto hace que sea muy conveniente para activar o desactivar estilos o comportamientos basados en la presencia de una clase.

"Este método es especialmente útil cuando deseas cambiar el estado de un elemento en respuesta a un evento o acción del usuario, como alternar un menú desplegable o cambiar el color de un botón"

Método contains()

El método **contains()** se utiliza para verificar si un elemento tiene una clase específica.
Retorna un valor **booleano** (**true o false**) que indica si la clase está presente o no en el elemento.

```
<!-- HTML -->
<div id="miDiv" class="rojo borde-grueso"></div>
---------
// JavaScript
const miElemento = document.getElementById('miDiv');

// Verificando si el elemento tiene la clase "rojo"
usando classList.contains()
const tieneClaseRojo =
miElemento.classList.contains('rojo');
console.log(tieneClaseRojo); // Output: true

// Verificando si el elemento tiene la clase "azul"
const tieneClaseAzul =
miElemento.classList.contains('azul');
console.log(tieneClaseAzul); // Output: false
```

*"El método **contains()** es útil cuando necesitas realizar acciones basadas en la presencia o ausencia de ciertas clases en un elemento. Puedes utilizarlo para condicionar acciones, estilos o comportamientos según si un elemento tiene o no una clase específica."*

¿Dudas o consultas?

¡Muchas Gracias!