educación IT
CODE YOUR FUTURE

# JavaScript desde cero

Módulo 5

educación IT
CODE YOUR FUTURE

# Estilos y Clases CSS desde JavaScript

# Introducción

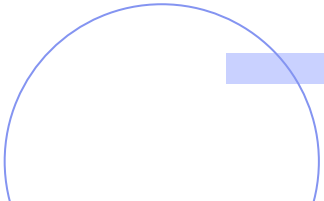
Cuando se comienza a trabajar con elementos HTML, desde JS, no solamente se puede leer su contenido y especificarlo; también existe la posibilidad de manipular los estilos CSS de estos elementos HTML.

Para ello, JS nos brinda dos posibles opciones que permiten manipular CSS fácilmente.

Antes de comenzar a tratar estos temas debemos tener presente que, **todo lo que haga referencia a estilos CSS deben manejarse siempre desde CSS.**

**La intervención de JS sobre los estilos queda limitada solamente a pequeños cambios que se aplica sobre los elementos HTML que CSS puro no puede manipular.**

Allí ingresará JS para intervenir en este tipo de escenarios específicos.



# La propiedad *style*

***HTMLElement Style*** es el mecanismo técnico que maneja el CORE de los motores web, para referenciar al DOM trabajado desde JavaScript (*HTMLElement*).

**Este CORE incluye un objeto básico, denominado `Style`**, que representa los estilos definidos para el DOM, a través del mecanismo de interfaces.

Tiene un punto de complejidad medio, si queremos trabajar directamente sobre cada uno de los estilos definidos en los documentos CSS integrados a un sitio web.

Aunque también, el objeto básico **`Style`**, forma parte del objeto **`document`**, con el que enlazamos a elementos HTML para manipularlos desde JS.

Por lo tanto, tendremos la posibilidad de hacer ajustes específicos sobre los estilos de los elementos HTML, una vez enlazados a ellos desde JS.

# Objeto *style*

El objeto **`style`** permite acceder a los estilos de un elemento con el elemento **`document`** como intermediario.

Por ejemplo:

JS Objeto Style

```
let imgLogo = document.querySelector("img#logo")
```

Luego, se pueden obtener los estilos de un elemento o sobreescribirlos, como se observa en la imagen de la derecha:

Es importante entender que **solo se sobreescribe esa propiedad, no las restantes que ese elemento pueda tener**.

También, este estilo se agregará en línea y no de manera externa, dado que no podemos modificar una hoja externa de CSS con JS.

JS Objeto Style

```
imgLogo.style.color = 'black'
imgLogo.style.backgroundColor = 'yellowgreen'
```

## Objeto *style* y sus propiedades

- **Cada propiedad CSS se trabaja desde JavaScript con el mismo nombre**.

  La única diferencia a tener presente es que, las propiedades CSS con **nombre combinado**, separado por un guión, **(*kebab-case*), se transforman en propiedades (*camelCase*)**.

  **Ejemplo:**

| CSS | JS |
|---|---|
| `border-width` | `borderWidth` |

- Los **valores** que asignaremos a cada propiedad CSS, desde JS, **deben encerrarse entre comillas**.

Propiedades CSS en el objeto JS Style

```
logo.style.width = '80px'
logo.style.height = '80px'
logo.style.padding = '20px'
logo.style.margin = '20px'
logo.style.borderWidth = '1px'
logo.style.borderStyle = 'solid'
```

educación IT
CODE YOUR FUTURE

# Acceso a clases CSS (*className*)

# Acceso a clases CSS

Prácticamente **todo elemento HTML depende del uso de clases CSS para adquirir estilos**: colores, espacios, bordes, ubicación en pantalla, entre otros.

Para este trabajo, JavaScript cuenta con una serie de métodos dedicados, que facilitan la interacción.

El siguiente código posee un **`<div>`** que oficia de *card HTML* y posee el atributo **`class`** con dos clases CSS referenciadas.

```
<div id="card" class="card card-product">
    <h1>Título del producto</h1>
    <img src="images/producto01.jpg">
    <button>AGREGAR</button>
</div>
```

Nos enlazamos con el elemento ** ** y desde la propiedad **className**, se podrán leer las clases CSS que tenga definidas.

```
const divCard = document.getElementById("card")

console.log(divCard.className)  //retorna card card-product
```

A través de la propiedad **.className**, es posible **vaciar todas las clases de un elemento HTML o agregar una nueva** a las existentes.

Se debe tener la precaución de **dejar un espacio, al agregar una nueva clase CSS**, porque sino quedará pegada a la que exista previamente. Ambas no funcionarán como tales.

```
divCard.className = ""    //eliminamos a todas

divCard.className += " otra-clase-css"  // agregamos una
```

**Nota:** el **atributo HTML `class`** se denomina **`className`** en **JavaScript**, dado que la palabra `class` en JS está reservada para la creación y manejo de clases basadas en métodos constructor (objetos).

educación IT
CODE YOUR FUTURE

# Acceso a clases CSS (*classList*)

# Manejo de clases con *classList*

La propiedad **`classList`** permite manejar clases CSS, en un elemento HTML, pero **en forma de *arrays***.

De la misma manera que accedemos a `.className`, podemos hacerlo con **`.classList`** aunque, en este caso, **retorne un *array* con cada una de las clases integradas en un elemento HTML**.

El elemento HTML **`textArea`** con el cual nos enlazamos desde JS previamente, brinda a través de su propiedad **`.classList`** todas las clases CSS que posee definidas. Como vemos en el gráfico de abajo, **todas las clases CSS genéricas están representadas en un *array***. Por lo tanto, podríamos acceder a ellas a través del índice que generan los *arrays* sobre sus elementos.

## Métodos

Pero para trabajar con las clases CSS de forma más práctica, tenemos una serie de métodos que facilitan realizar las operaciones básicas sobre clases CSS.

Veamos a estos en la siguiente tabla:

| Método | Descripción |
|---|---|
| `.add("nombre-clase")` | Agrega una clase CSS al elemento HTML. |
| `.remove("nombre-clase")` | Remueve la clase CSS indicada, del elemento HTML. |
| `.toggle("nombre-clase")` | Si la clase CSS no se encuentra listada en el elemento HTML, la agrega.<br>Si se encuentra, la quita. |
| `.replace("clase-vieja", clase-nueva")` | Permite reemplazar una clase CSS por otra clase CSS en el elemento HTML. |

## Definición

En todos los casos, el nombre de la clase CSS se define **entre los paréntesis del método a utilizar** y se evita anteponer el punto (**.**).

Métodos de la propiedad classList

```
divCard.classList.add("shadow-lightgray");

divCard.classList.remove("text-blue");

divCard.classList.toggle("text-medium-size");
```

## Beneficios

El grandioso beneficio que se obtiene con **`classList`** y sus métodos asociados, es que podemos **controlar CSS de una forma mucho más precisa, y aplicar pequeños cambios** que CSS, de forma directa, no puede controlar, porque son parte de la lógica de una aplicación web.

Ejemplo:

```
const buttonCart = document.querySelector("button.button-cart.add-to-cart")

buttonCart.addEventListener("click", ()=> {
      carrito.push(selectedProduct)    //agregamos el producto al carrito
      selectedProduct.stock--          //restamos una unidad de su stock
      if (selectedProduct.stock <= 0) {
          buttonCart.disabled = true
          buttonCart.classList.replace("add-to-cart", "cart-disabled")
      }
})
```

# ¡Sigamos trabajando!