Contenido

LA API CLASSLIST DE JAVASCRIPT

1 MANIPULACIÓN DE CLASES.

En CSS es muy común utilizar múltiples **clases CSS** para asignar estilos relacionados dependiendo de lo que queramos. Para ello, basta hacer cosas como la que veremos a continuación:

```
<div class="element shine dark-theme"></div>
```

Observa que tenemos un elemento ** ** que tiene las siguientes clases:

- La clase **element** sería la clase general que representa el elemento, y que tiene estilos fijos.
- La clase **shine** podría tener una animación CSS para aplicar un efecto de brillo.
- La clase **dark-theme** podría tener los estilos de un elemento en un tema oscuro.

Todo esto se utiliza sin problema de forma estática, pero cuando comenzamos a programar en Javascript, buscamos una forma **dinámica**, práctica y cómoda de hacerlo desde Javascript, y es de lo que tratará este apartado.

1.1 LA PROPIEDAD className

Javascript tiene a nuestra disposición una propiedad **.className** en todos los elementos HTML. Dicha propiedad contiene el valor del atributo HTML **class** como un **STRING** , y puede tanto leerse como reemplazarse:

Propiedad	Descripción
.className	Acceso directo al valor del atributo HTML **class**. También se puede asignar.
.classList	Objeto especial para manejar clases CSS. Contiene métodos y propiedades de ayuda.

La propiedad **.className** viene a ser la modalidad directa y rápida de utilizar el getter **.getAttribute("class")** y el setter **.setAttribute("class", v)**. Veamos un ejemplo utilizando estas propiedades y métodos y su equivalencia:

```
const div = document.querySelector(".element");

// Obtener clases CSS
div.className;          // "element shine dark-theme"
div.getAttribute("class");  // "element shine dark-theme"

// Modificar clases CSS
div.className = "elemento brillo tema-oscuro";
div.setAttribute("class", "elemento brillo tema-oscuro");
```

Trabajar con **.className** tiene una limitación cuando trabajamos con **múltiples clases CSS**, y es que puedes querer realizar una manipulación sólo en una clase CSS concreta, dejando las demás intactas. En ese caso, modificar clases CSS mediante una asignación **.className** se vuelve poco práctico.

Probablemente, la forma más interesante de manipular clases desde Javascript es mediante el objeto **.classList**.

1.2 El objeto .classlist

Para trabajar más cómodamente, existe un sistema muy interesante para trabajar con clases: la propiedad **classList**. Se trata de una propiedad que contiene una serie de métodos que permiten trabajar con las clases de forma más intuitiva y lógica.

Si accedemos a **.classList**, nos devolverá un **ARRAY** (*no es exactamente un array, sino un DOMTokenList*) de clases CSS de dicho elemento. Pero además, incorpora una serie de * métodos que nos harán muy **sencillo** trabajar con clases CSS:

Método	Descripción
ARRAY .classList	Devuelve la lista de clases del elemento HTML.
NUMBER .classList.length	Devuelve el número de clases del elemento HTML.
STRING .classList.item(n)	Devuelve la clase número **n** del elemento HTML. si no existe.
UNDEFINED .classList.add(c1, c2, ...)	Añade las clases **c1**, **c2**... al elemento HTML.
UNDEFINED .classList.remove(c1, c2, ...)	Elimina las clases **c1**, **c2**... del elemento HTML.
BOOLEAN .classList.contains(clase)	Indica si la **clase** existe en el elemento HTML.
BOOLEAN .classList.toggle(clase)	Si la **clase** no existe, la añade. Si no, la elimina.
BOOLEAN .classList.toggle(clase, expr)	Si **expr** es **true**, añade **clase**. Si no, la elimina.

Método	Descripción
BOOLEAN .classList.replace(old, new)	Reemplaza la clase **old** por la clase **new**.

Veamos un ejemplo de uso de cada método de ayuda. Supongamos que tenemos el siguiente elemento HTML en nuestro documento. Vamos a acceder a el y a utilizar el objeto **.classList** con dicho elemento:

```
<div id="page" class="info data dark" data-number="5"></div>
```

Observa que dicho elemento HTML tiene:

- Un atributo **id**
- Tres clases CSS: **info**, **data** y **dark**
- Un metadato HTML **data-number**

1.2.1 ACCEDER A CLASES CSS

Al margen de acceder a la lista de clases mediante **.classList** y al número de clases del elemento con **.classList.length**, es posible acceder a la propiedad **.classList.values** para obtener un como lo haría **.className**:

```
const div = document.querySelector("#page");

div.classList;          // ["info", "data", "dark"] (DOMTokenList)
div.classList.length;   // 3
Array.from(div.classList)  // ["info", "data", "dark"] (Array)
[...div.classList];     // ["info", "data", "dark"] (Array)
div.classList.values;   // "info data dark" (String)

div.classList.item(0);  // "info"
div.classList.item(1);  // "data"
div.classList.item(3);  // null
```

Recuerda que el objeto **.classList** aunque parece que devuelve un no es un array, sino un que actúa de forma similar a un array, por lo que puede carecer de algunos

métodos o propiedades concretos. Si quieres convertirlo a un array real, utiliza **Array.from()** o desestructuración con **[...div.classList]**.

Por último, observa que disponemos del método **.classList.item()** que nos devuelve un con la clase específica en esa posición. Si no existe una clase en esa posición, nos devolverá .

1.2.2 AÑADIR Y ELIMINAR CLASES CSS

Los métodos **.classList.add()** y **.classList.remove()** permiten indicar una o múltiples clases CSS a añadir o eliminar. Observa el siguiente código donde se ilustra un ejemplo:

```
const div = document.querySelector("#page");

div.classList.add("uno", "dos");    // No devuelve nada.
div.classList;  // ["info", "data", "dark", "uno", "dos"]

div.classList.remove("uno", "dos"); // No devuelve nada.
div.classList;  // ["info", "data", "dark"]
```

Al utilizar los métodos **.add()** o **.remove()**, en el caso de que se añada una clase CSS que ya existía previamente, o que se elimine una clase CSS que no existía, simplemente no ocurrirá nada.

1.2.3 COMPROBAR SI EXISTEN CLASES CSS

Con el método **.classList.contains()** podemos comprobar si existe una clase en un elemento HTML, ya que nos devuelve un indicandonos si está presente o no:

```
const div = document.querySelector("#page");

div.classList;                  // ["info", "data", "dark"]
div.classList.contains("info");    // Devuelve `true` (existe esa clase)
div.classList.contains("warning");  // Devuelve `false` (no existe esa clase)
```

Esto puede resultar interesante en algunas situaciones, donde queremos averiguar mediante Javascript si existe una clase.

1.2.4 CONMUTAR O ALTERNAR CLASES CSS

Otro método muy interesante es**.classList.toggle()**, que lo que hace es **añadir o eliminar la clase CSS** dependiendo de si ya existía previamente. Es decir, añade la clase si no existía previamente o elimina la clase si existía previamente:

```
const div = document.querySelector("#page");
div.classList; // ["info", "data", "dark"]
div.classList.toggle("info"); // Como "info" existe, lo elimina. Devuelve "false"
div.classList; // ["data", "dark"]
div.classList.toggle("info"); // Como "info" no existe, lo añade. Devuelve "true"
div.classList; // ["info", "data", "dark"]
```

Observa que **.toggle()** devuelve un que será **true** o **false** dependiendo de si, tras la operación, la clase sigue existiendo o no. Ten en cuenta que en **.toggle()**, al contrario que **.add()** o **.remove()**, sólo se puede indicar una clase CSS por parámetro.

1.2.5 REEMPLAZAR UNA CLASE CSS

Por último, tenemos un método **.classList.replace()** que nos permite reemplazar la primera clase indicada por parámetro, por la segunda. Veamos este método en acción:

```
const div = document.querySelector("#page");
div.classList; // ["info", "data", "dark"]
div.classList.replace("dark", "light");      // Devuelve `true` (se hizo el cambio)
div.classList.replace("warning", "error");   // Devuelve `false` (no existe la clase
warning)
```

Con todos estos métodos de ayuda, nos resultará mucho más sencillo manipular clases CSS desde Javascript en nuestro código.