MODIFICAR ESTILOS CSS

Es posible modificar el CSS de los elementos mediante el método setAttribute, a través de los atributos **style** o **class**. JavaScript proporciona métodos especiales para manipular las clases CSS de un elemento

PROPIEDAD STYLE

Los elementos poseen una propiedad llamada style que permite acceder a las propiedades CSS de un elemento. Lo que realmente hace es modificar el atributo style del elemento, modifican los valores de la misma. La idea es que style es un objeto que tiene como propiedades todas las propiedades CSS. Por ejemplo:

let parrafo=document.getElementsByTagName("p")[0]; //selecciona un elemento párrafo
parrafo.style.color="red"; //pone en color rojo el texto del parrafo
parrafo.style.border="1px solid black"; //

De esta forma podemos ir modificando todo el CSS que queramos. El problema viene cuando la modificación es sobre propiedades como **border-bottom**, **background-color** que por la sintaxis de JavaScript darían confusión con el signo de resta "-", de hecho los identificadores de propiedades, funciones, métodos, etc… no pueden tener guiones para evitar confusión con la resta. Para poder manipular propiedades CSS como las mencionadas (separadas por guiones) se convierte su sintaxis de tal forma que se eliminan los guiones, por ejemplo, **border-bottom** pasa a **borderBottom**, **text-align** a **textAlign**, **background-color** a **backgroundColor** y así con todas las propiedades. Para el ejemplo anterior:

parrafo.style.backgroundColor="white"; //establece el color de fondo del parrafo a blanco.

No obstante, la mayoría de los navegadores actuales acepta también el formato idéntico a CSS a través de corchetes:

parrafo.style["background-color"]="white";

OBTENER LOS ESTILOS CSS QUE SE APLICAN A UN ELEMENTO

JavaScript dispone de un método en el objeto window llamado **getComputedStyle**. Este método devuelve un objeto en el que se pueden consultar las propiedades CSS que se están aplicando a un elemento en concreto. Solo permite ver las propiedades del elemento pero no modificarlas. Ejemplo:

let cssParrafo=window.getComputedStyle(parrafo);
console.log(cssParrafo.fontFamily); //mostrará el valor de la propiedad Font-family que tiene
//el párrafo pasado como parámetro.

MANIPULAR LAS CLASES

Los elementos disponen de una propiedad llamada className. Mediante esa propiedad podemos asignar una clase CSS a un elemento de la página. Por ejemplo:

parrafo.className="nuevaClase"; //asignamos a párrafo una nueva clase.
console.log(párrafo.className); // mostramos el valor del atributo class de párrafo

Si un elemento en su atributo class tiene más de un elemento a la hora de mostrarla no hay problema pero si surge un problema si queremos añadir o eliminar alguna clase más ya que habría que tratar el string que continen el valor o valores de class. Ejemplo:

Si tenemos el párrafo:
<p id="nota" class=" notas anotacion">

Si ejecutamos el código: ***console.log(document.getElementsById("nota").className;***
//muestra las dos clases: notas y anotación. Añadir o quitar supone modificar el string siendo más incómodo.

Por esa razón JavaScript incorporó la propiedad **classList** a los elementos del DOM. Se trata de un objeto parecido a un array que representa las clases aplicadas a un elemento. Dispone de propiedad length, podemos acceder a sus elementos con un índice y se pueden usar los bucles de recorrido habituales para los array pero muchos de los métodos de los array no están disponibles.
Así, obtener el nombre de las clases de un elemento mediante ***classList*** se haría, por ejemplo:

```
for(let clase of parrafo.classList){
   console.log(clase);
}
```
// mostrará las clases notas y anotacion

La propiedad **classList** dispone de estas interesantes propiedades:

- **add(nombreClasel [, nombreClase2]):** Permite añadir una clase al elemento. Basta con indicar el nombre de la clase entre comillas. Admite indicar varias clases si se separan con comas.
- **remo ve (nombreCl asel [, nombreClase2¡):** Método contrario al anterior. Retira del elemento la clase o clases que se indiquen.
- **toggle(nombreClase [, forzar]):** Si se usa un solo parámetro, será el nombre de una clase. El método hace que si el elemento no tiene asignada esa clase, se asigna. Si el elemento ya tenía esa clase, la quita. Es un método muy interesante y que se usa muy a menudo.

  El segundo parámetro ***(forzar)*** es un valor booleano que si vale **true** entonces añade la clase, independientemente de si el elemento ya la tenía o no. Si vale false, entonces quita la clase del elemento. Es decir, este método es capaz de hacer lo que hacían los otros dos.

- **contains(nombreClase):** Devuelve **true** si el elemento tiene asignada la clase cuyo nombre se indica en el parámetro.
- **replace(nombreViejo, nombreNuevo):** Permite cambiar una clase por otra en el elemento.

OBTENER ATRIBUTOS DATA

En los documentos HTML existe la posibilidad de crear atributos **data.** Se trata de un tipo de atributos creados por los propios desarrolladores a voluntad. Se usan para almacenar información que puede ser manipulada desde JavaScript.

Los atributos data empiezan por ese mismo término (data) seguidos de un guión y luego el nombre en sí que queramos data al atributo. Por ejemplo:

```
<p id="pl" data-tipo="Inicio de libro" data-1ibro="Don Quijote"
    data-autor-principal="Miguel de Cervantes">
En un lugar de La Mancha...
</p>
```

Para manipular estos atributos desde JavaScript, disponemos de un objeto **dataset.**

dataset es un objeto que contiene como propiedades cada una de los atributos data del elemento actual. Para acceder concretamente a uno se usa una sintaxis de tipo **Camel Case** (de la que ya hablamos en el apartado de la propiedad style). Concretamente en este ejemplo, podemos ver los valores de los tres atributos de esta forma:

```
consolé.log(pl.dataset.libro);
consolé.log(pl.dataset.autorPrincipal);
consolé.log(pl.dataset.tipo);
```

Muestra:

```
Don Quijote
Miguel De Cervantes
Inicio de libro
```

Es posible también modificar un atributo data, o crearlo si no existe:

```
pl.dataset.publicacion="Siglo XVII";
```