

# PRACT 6: EVENTOS – MANEJO ESTRUCTURA DOM

## Introducción:

### MODIFICAR ATRIBUTOS, CLASES Y ESTILOS EN EL DOM

Los elementos HTML a menudo tienen información adicional asignada en forma de atributos. Los atributos pueden incluir pares de nombre y valor, y algunos de los más comunes son class y style. Pueden ser modificados a través de los métodos de la estructura DOM

#### MODIFICAR ATRIBUTOS EN EL DOM

Los atributos son valores que contienen información adicional sobre elementos HTML. Normalmente vienen en pares de **nombre y valor**, y pueden ser esenciales según elemento.

En JavaScript, tenemos cuatro métodos para modificar atributos de elementos:

Método	Descripción	Ejemplo
<code>hasAttribute()</code>	Muestra un booleano true o false	<code>element.hasAttribute('href');</code>
<code>getAttribute()</code>	Muestra el valor de un atributo especificado o null	<code>element.getAttribute('href');</code>
<code>setAttribute()</code>	Agrega o actualiza el valor de un atributo especificado.	<code>element.setAttribute('href', 'index.html');</code>
<code>removeAttribute()</code>	Elimina un atributo de un elemento.	<code>element.removeAttribute('href');</code>

#### Ejemplo:

Crearemos un nuevo archivo HTML con una etiqueta img con un atributo. Estableceremos un vínculo a una imagen pública disponible a través de una URL, pero puede cambiarla por una imagen local alternativa si trabaja sin conexión.

```
<!DOCTYPE html>  
<html lang="en">  
<body>
```

```

```

```
</body>
```

```
</html>
```

Podemos probar todos los métodos de atributos sobre la marcha.

```
// Assign image element  
const img = document.querySelector('img');
```

```
img.hasAttribute('src');           // returns true
img.getAttribute('src');          // returns "...shark.png"
img.removeAttribute('src');       // remove the src attribute and value
```

En este punto, habrá eliminado el atributo src y el valor asociado con img, pero puede reiniciar ese atributo y asignar el valor a una imagen alternativa con **img.setAttribute()**:

```
img.setAttribute('src', 'https://js-tutorials.nyc3.digitaloceanspaces.com/octopus.png');
```

También podemos modificar el atributo de esta forma:

```
img.src = 'https://js-tutorials.nyc3.digitaloceanspaces.com/shark.png';
```

Cualquier atributo puede editarse de esta forma y también con los métodos anteriores.

Los métodos **hasAttribute()** y **getAttribute()** se utilizan normalmente con instrucciones condicionales y los métodos **setAttribute()** y **removeAttribute()** se usan para modificar directamente el DOM.

## MODIFICAR CLASES EN EL DOM

Las clases CSS se utilizan para aplicar estilos a varios elementos, a diferencia de los ID que solo pueden existir una vez por página. En JavaScript, contamos con las propiedades **className** y **classList** para trabajar con el atributo de clase.

Método/Propiedad	Descripción	Ejemplo
<code>className</code>	Obtiene o establece un valor de clase.	<code>element.className;</code>
<code>classList.add()</code>	Agrega uno o más valores de clase.	<code>element.classList.add('active');</code>
<code>classList.toggle()</code>	Activa o desactiva una clase.	<code>element.classList.toggle('active');</code>
<code>classList.contains()</code>	Comprueba si el valor de clase existe.	<code>element.classList.contains('active');</code>
<code>classList.replace()</code>	Sustituye un valor de clase existente por uno nuevo.	<code>element.classList.replace('old', 'new');</code>
<code>classList.remove()</code>	Elimina un valor de clase.	<code>element.classList.remove('active');</code>

## EJEMPLO

```
classes.html
<!DOCTYPE html>
<html lang="en">

<style>
  body {
    max-width: 600px;
    margin: 0 auto;
    font-family: sans-serif;
  }
  .active {
    border: 2px solid blue;
  }
  .warning {
```

```

        border: 2px solid red;
    }

    .hidden {
        display: none;
    }

    div {
        border: 2px dashed lightgray;
        padding: 15px;
        margin: 5px;
    }
</style>

<body>
    <div>Div 1</div>
    <div class="active">Div 2</div>
</body>

</html>

```

La propiedad **className** se introdujo para evitar conflictos con la palabra clave de class que se encuentra en JavaScript y otros lenguajes que tienen acceso al DOM. Puede utilizar className para asignar un valor directamente a la clase.

```

// Select the first div
const div = document.querySelector('div');

// Assign the warning class to the first div
div.className = 'warning';

```

Tenga en cuenta que si ya hay clases en el elemento, esto las anulará. Puede agregar varias clases delimitadas por espacios usando la propiedad className o usarla sin los operadores de asignación para obtener el valor de la clase en el elemento.

La otra forma de modificar clases es a través de la propiedad **classList**, que incluye algunos métodos útiles.

```

// Select the second div by class name
const activeDiv = document.querySelector('.active');

activeDiv.classList.add('hidden');           // Add the hidden class
activeDiv.classList.remove('hidden');         // Remove the hidden class
activeDiv.classList.toggle('hidden');         // Switch between hidden true and false
activeDiv.classList.replace('active', 'warning'); // Replace active class with warning class

```

A diferencia del ejemplo de **className**, al utilizarse **classList.add()** se añadirá una nueva clase a la lista existente. También puede agregar varias clases como cadenas separadas por comas. También es posible utilizar **setAttribute** para modificar la clase de un elemento.

## MODIFICAR ESTILOS EN EL DOM

La propiedad de estilo representa los estilos incorporados en un elemento HTML. A menudo, se aplicarán estilos a elementos a través de una hoja de estilos, pero a veces debemos agregar o editar un estilo alineado directamente.

## EJEMPLO

```

styles.html
<!DOCTYPE html>
<html lang="en">

<body>

    <div style="height: 100px;
        width: 100px;
        border: 2px solid black;">Div</div>

</body>

</html>
    
```

Una opción para editar los estilos es `setAttribute()`.

```

// Select div
const div = document.querySelector('div');
// Apply style to div
div.setAttribute('style', 'text-align: center');
    
```

Sin embargo, esto eliminará todos los estilos alineados existentes del elemento. Debido a que este probablemente no sea el efecto deseado, es mejor utilizar el atributo `style` directamente.

```

div.style.height = '100px';
div.style.width = '100px';
div.style.border = '2px solid black';
    
```

Las propiedades de CSS se escriben en **kebab-case**, que son palabras minúsculas separadas por guiones. Tenga en cuenta que las propiedades de CSS de kebab-case no se pueden utilizar en la propiedad de estilo de JavaScript. Se sustituirán por su equivalente en camelCase, en el cual la primera palabra se escribe con minúscula y las siguientes con mayúsculas. En otras palabras, en vez de `text-align` utilizaremos `textAlign` para la propiedad de estilo de JavaScript.

```

// Make div into a circle and vertically center the text
div.style.borderRadius = '50%';
div.style.display = 'flex';
div.style.justifyContent = 'center';
div.style.alignItems = 'center';
    
```

Si deben aplicarse muchos cambios estilísticos a un elemento, lo mejor es aplicar los estilos a una clase y agregar una nueva clase. Sin embargo, hay algunos casos en los que será necesario o más práctico modificar el atributo de estilo alineado.

## Secuencia y desarrollo:

1. **Ejercicio 1:** A partir del fichero `p6_ejercicio1` facilitado, escribe el código Javascript necesario que permita la introducción de un código de 5 dígitos. Si el código introducido es igual que el

que se almacena en la variable `codigoValido`, la cual se genera de forma aleatoria al cargar la página.

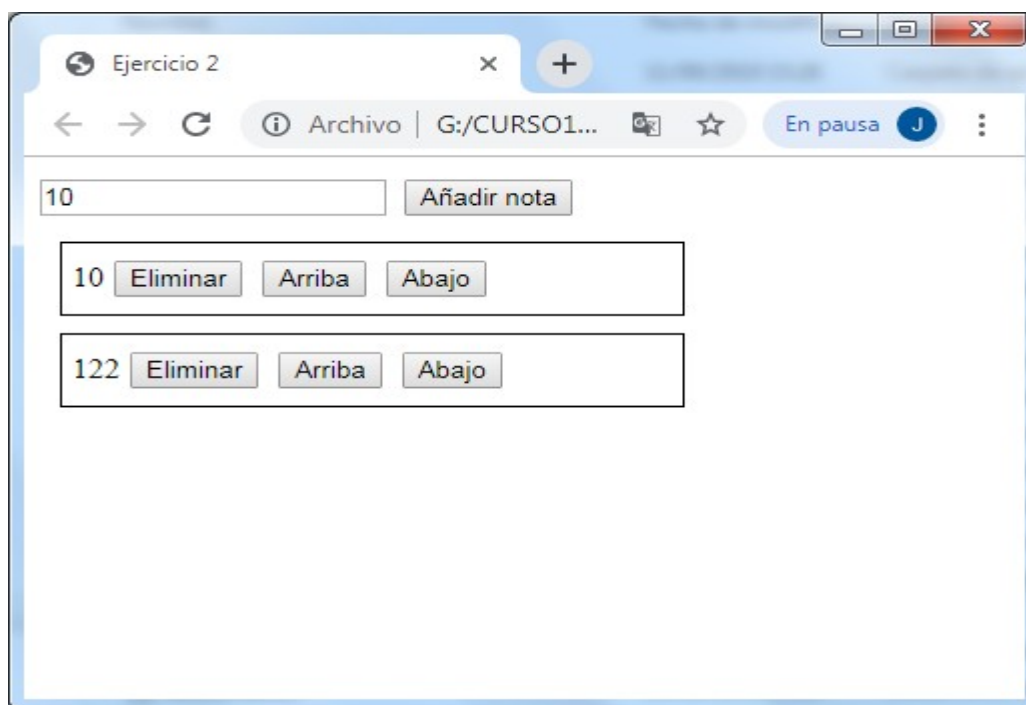
- Deberá mostrarse un mensaje de alerta indicando que el código introducido es correcto, y mostrando dicho código.
- Otro mensaje de alerta indicando que el mensaje introducido no es correcto.

## 2. Ejercicio 2: Realizar la siguiente práctica teniendo en cuenta las siguientes especificaciones.



Al cargar la página, el usuario introduce el número de cajas que quiere crear. Una vez que han sido creadas, al pasar con el ratón por encima cambia de color, y al salir de la caja vuelve al color inicial. El cuadro de texto asociado al botón **antes**, indica la posición índice para crear una caja y añadir antes de la posición indicada (en el ejemplo es la caja de color verde) Comportamiento similar al texto asociado al botón **después**. (En el ejemplo es la caja de color naranja)

## 3. Ejercicio 3: Crea una aplicación web con un botón y un cuadro de texto. Cuando el usuario pulse añadir nota se crea una capa con la nota y tres botones, los cuales al pulsar o se elimina, pasa a ser el primero o el último.



The screenshot shows a web browser window with the title "Ejercicio 2". The address bar shows "G:/CURSO1...". The page content includes a text input field with the value "10" and a button "Añadir nota". Below this, there are two rows of data, each in a bordered box. The first row contains the number "10" followed by buttons "Eliminar", "Arriba", and "Abajo". The second row contains the number "122" followed by buttons "Eliminar", "Arriba", and "Abajo".

Nota	Eliminar	Arriba	Abajo
10			
122			