EXERCISES QUE TRABAJAREMOS EN EL CUE:

- EXERCISE 1: MANIPULACIÓN DE ELEMENTOS Y SUS PROPIEDADES CSS.
- EXERCISE 2: MANEJADORES DE EVENTOS.

EXERCISE 1: MANIPULACIÓN DE ELEMENTOS Y SUS PROPIEDADES CSS

Durante este curso hemos visto como manipular la información de un elemento HTML con JavaScript puro, y también analizado como aplicar y cambiar propiedades CSS de manera directa en nuestro código. Ahora, estudiaremos cómo utilizar los métodos que provee jQuery, para lograr estos tipos de alteraciones sobre los elementos de un HTML.

Para empezar, vamos a crear un nuevo HTML en nuestro IDE Visual Studio Code. Nuestro primer paso es siempre incluir jQuery entre las etiquetas `<head>` de nuestro proyecto, pues así no tendremos problemas al momento de desarrollar.

Para manipular el contenido de los elementos del DOM, jQuery ofrece tres métodos que permiten obtener o sobrescribir información. Estos son: `text()`, `html()` y `val()`.

`text()` - Fija o retorna el contenido de texto de un elemento seleccionado.

`html()` - Fija o retorna el contenido de un elemento, incluyendo los elementos HTML.

`val()` - Fija o retorna el valor de los campos de un formulario (Por ejemplo: de un input).

Veamos un ejemplo práctico. Plantearemos el siguiente HTML:

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Ejercicio JQuery</title>

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
script>
</head>

```

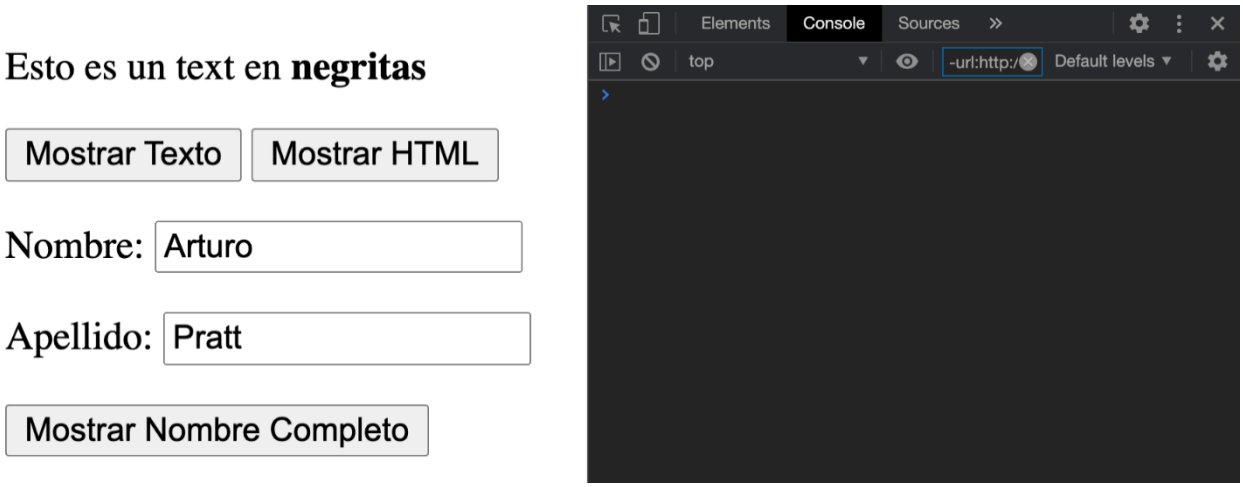
```
<body>

    <p id="test">Esto es un text en <b>negritas</b></p>
    <button id="btn1">Mostrar Texto</button>
    <button id="btn2">Mostrar HTML</button>

    <p>Nombre: <input type="text" id="valorNom" value="Arturo"></p>
    <p>Apellido: <input type="text" id="valorAp" value="Pratt"></p>
    <button id="btn3">Show Value</button>

</body>
```

En primer lugar, veremos cómo usar estos métodos para retornar la información desde los elementos. Para hacerlo, estableceremos lo siguiente:

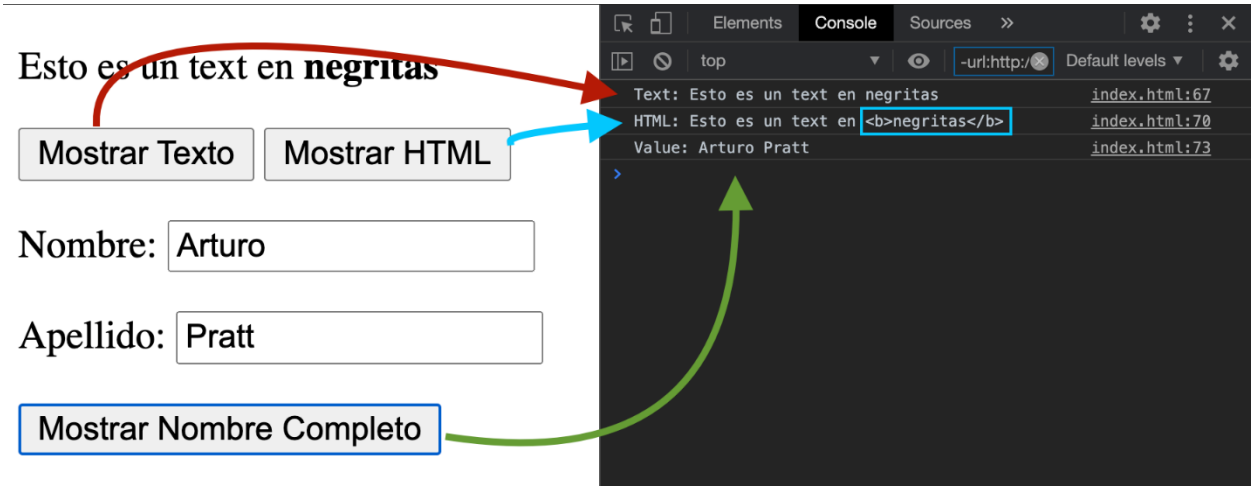
```
<script>
   $(function () {
       $("#btn1").click(function () {
           console.log("Text: " + $("#test").text());
       });
       $("#btn2").click(function () {
           console.log("HTML: " + $("#test").html());
       });
       $("#btn3").click(function () {
           console.log("Value: " + $("#valorNom").val() + " " +
$("#valorAp").val());
       });
   });
</script>
```

En esta función vamos a mostrar por consola la información de texto y de HTML de un elemento con el id **“test”**. Luego, los valores de dos inputs que contienen la información de un nombre y un apellido; y un **String** con ambos valores juntos.

El código planteado se ve de la siguiente forma en nuestro navegador:



Al hacer **CLIC** en “Mostrar Texto”, “Mostrar HTML” y “Mostrar Nombre Completo”, podremos apreciar lo siguiente:



En base a este resultado, sabemos que el método `text()` nos muestra el contenido de texto de un elemento seleccionado, mientras que el método `html()` enseña todo el contenido de un elemento, incluyendo las etiquetas HTML, lo cual tenemos destacado con una cajita azul donde se pueden ver las etiquetas `<b>`. Además, con esto aprendemos que el método `val()` devuelve o retorna el valor ingresado en elementos de formularios, como los inputs. Si ingresáramos otros valores en los inputs, en la consola veríamos reflejado ese cambio al hacer **CLIC** en el último botón.

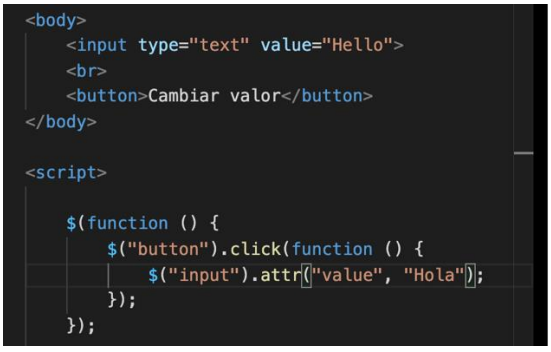
Recién estudiamos cómo retornar valores desde los elementos, pero para setear o fijarlos, basta con pasar por parámetro el valor que deseamos fijar. Por ejemplo, si usamos el mismo HTML (solo cambiaremos el texto dentro de los botones), podemos incorporar las siguientes modificaciones a nuestro código para que, en vez de retornar valores, los fije:

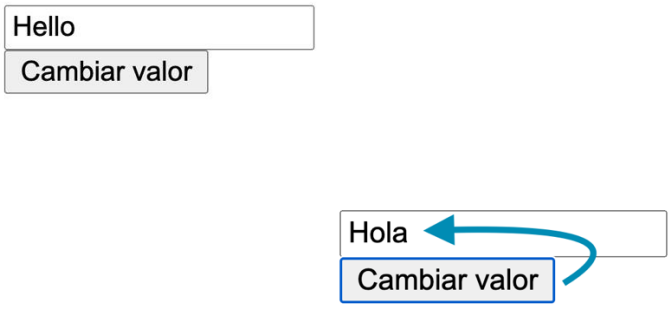
```
<script>
    $(function () {
        $("#btn1").click(function () {
            $("#test").text("Reemplazamos este texto");
        });
        $("#btn2").click(function () {
            "HTML: " + $("#test").html("<h1>Luego, podemos hacerlo más
<i>GRANDE</i></h1>");
        });
        $("#btn3").click(function () {
            "Value: " + $("#valorNom").val("Salvador") + " " +
$("#valorAp").val("Dalí");
        });
    });
</script>
```

El resultado en nuestro navegador es el siguiente:

Otra manera en que podemos cambiar los atributos es utilizando el método `attr()`, pasándole por parámetro el atributo en especifico que queremos cambiar, junto con su valor. En el siguiente caso ocupamos este método para cambiar el valor de un input.

```
<body>
    <input type="text" value="Hello">
    <br>
    <button>Cambiar valor</button>
</body>
<script>
    $(function () {
        $("button").click(function () {
            $("input").attr("value", "Hola")
        })
    })
</script>
```





De esta manera queda demostrado como manipular la información de elementos HTML, para poder retornar y fijar información en el DOM.

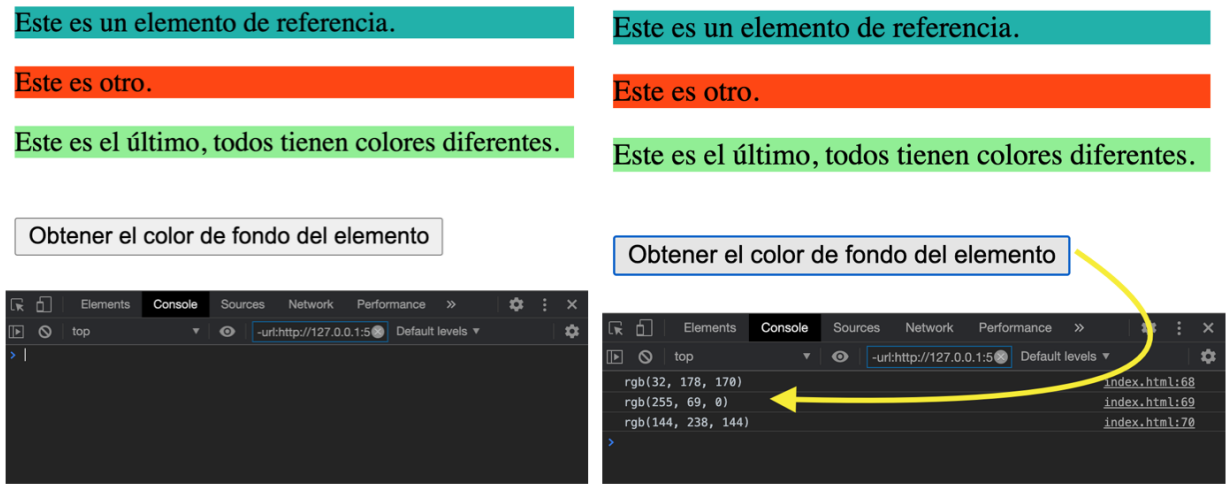
Ahora veremos como cambiar las propiedades CSS de los elementos. El método que nos permite obtener y fijar propiedades es `css()`. Para obtener una propiedad basta con pasar por parámetro el nombre de ella. Por ejemplo, si tenemos el siguiente HTML:

```
<body>
   <p style="background: lightseagreen;">
      Este es un elemento de referencia.
   </p>
   <p style="background: orangered;" id="p2">
      Este es otro.
   </p>
```

```
    <p style="background: lightgreen;" id="p3">
        Este es el último, todos tienen colores diferentes.
    </p>
    <br>
    <button>Obtener el color de fondo del elemento</button>
</body>
<script>
    $(function () {
        $("button").click(function () {
            console.log($("p").css("background-color"));
            console.log($("#p2").css("background-color"));
            console.log($("#p3").css("background-color"));
        });
    });
</script>
```

Veremos que en nuestra función mostraremos por consola, el valor del color de fondo de todos los elementos `<p>` de nuestro HTML. Es importante destacar que este método retorna el valor del primer elemento que cumple con su criterio, y es por eso que en nuestro primer console.log colocamos el nombre de la etiqueta del elemento, y después usamos los id de los otros dos.

En nuestro navegador el resultado es el siguiente:

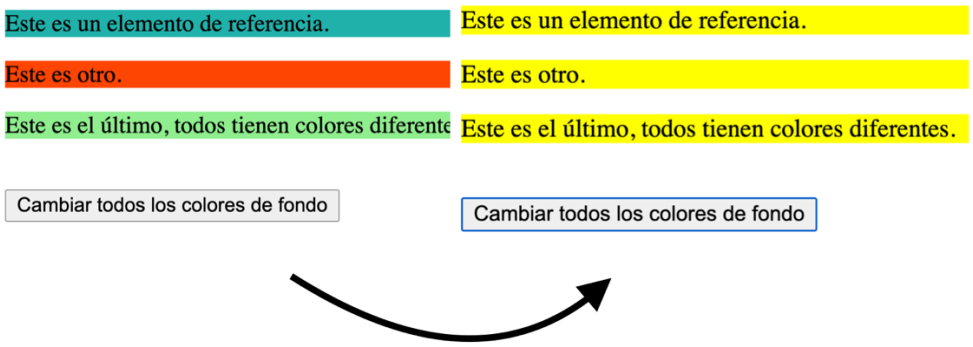


Como se puede observar, al consultar por el valor del color de fondo de estos atributos, en la consola recibimos el valor de cada uno de ellos en formato RGB. Este método lo podemos usar para obtener la información de cualquier otro atributo de CSS que se requiera.

Para fijar valores CSS utilizando el mismo método, basta con pasar por parámetro el nombre de la propiedad y su valor. Por ejemplo, vamos a cambiar el color de fondo de todos los elementos a amarillo, con el siguiente código:

```
$(function () {
        $("button").click(function () {
                $("p").css("background-color", "yellow");
        });
});
```

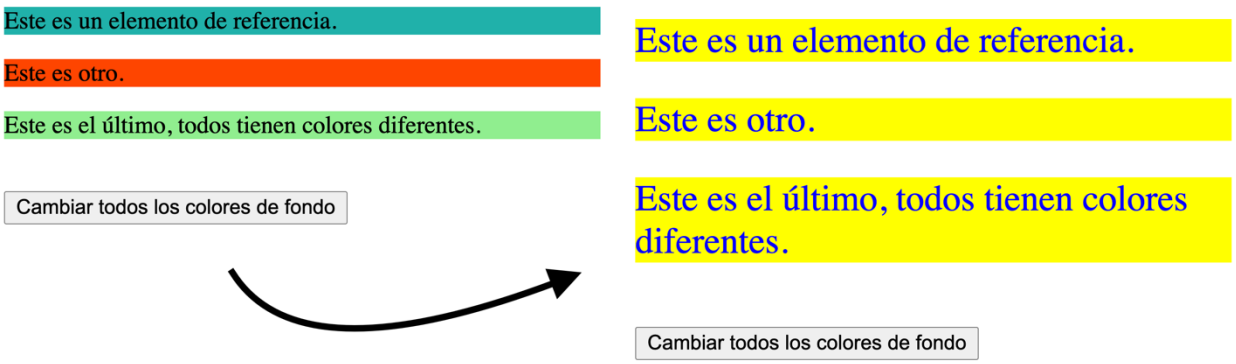
En este caso, fijamos el color de fondo de todos los elementos `<p>` en el DOM. El resultado en nuestro navegador es el siguiente:



Con este método también podemos fijar múltiples propiedades CSS, ¿cómo se hace?: éstas se pasan por parámetro con la sintaxis de un Array. Por ejemplo, si quisiéramos cambiar el color de fondo y el tamaño de la letra junto a su color, podemos hacerlo de la siguiente manera:

```
$(function () {
        $("button").click(function () {
                $("p").css({
                        "background-color": "yellow",
                        "font-size": "150%",
                        "color": "blue"
                });
        });
});
```

Acá vamos a manipular tres propiedades CSS: el color de fondo, el tamaño de la letra y el color del texto. Las introducimos como parámetro en forma de un arreglo de pares propiedad/valor, tal como un JSON. El resultado en nuestro navegador es el siguiente:



De esta forma podemos utilizar métodos de jQuery para manipular el contenido, como las propiedades de estilo de los elementos dentro del DOM.

EXERCISE 2: MANEJADORES DE EVENTOS

Para explicar este punto, primero debemos definir lo que son eventos: acciones realizadas por usuarios, a las cuales una página web puede responder. Es decir, un evento representa el momento exacto cuando algo ocurre en nuestra página. Por ejemplo: "moviendo el mouse o cursor sobre un elemento", "seleccionando un botón", o "haciendo clic sobre un elemento".

Teniendo esto en cuenta, los **Manejadores de eventos** son métodos que nos permiten ejecutar instrucciones, en base a las acciones que realiza el usuario sobre la página web. Sin haberlo notado, ya los hemos estado usando. Por ejemplo, hemos trabajado con el siguiente manejador de eventos:

```
1 $(document).ready()
```

Este método incorpora un manejador de eventos al documento HTML, el cual nos permite ejecutar una función cuando el documento esté completamente cargado. En otras palabras, el *evento* corresponde a cuando el HTML se carga. El hecho de poder llevar a cabo instrucciones una vez que se termine de ejecutar la acción, hace que este método sea un **Manejador de eventos**.

Lo mismo sucede con el siguiente método que también hemos estado usando:

```
1 $("div").click()
```

Éste agrega un manejador de eventos al elemento `<div>`, que nos permite ejecutar una función al hacer clic sobre un elemento especificado.

Dado que ya hemos trabajado con estos dos manejadores de eventos, a continuación, estudiaremos cuatro tipos que nunca hemos usado. Además, analizaremos otros manejadores más que podremos investigar.

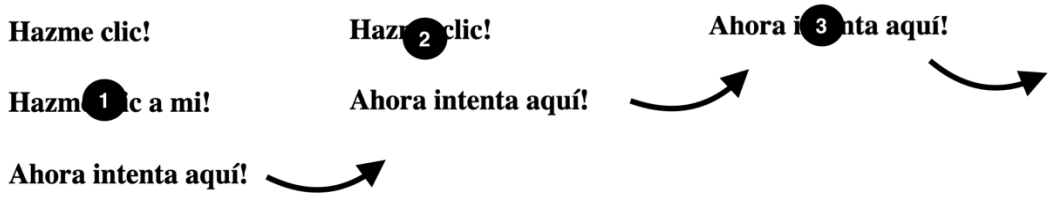
Para empezar, vamos a repasar el manejador `click()`. Partimos creando un nuevo archivo HTML en nuestro IDE Visual Studio Code. Luego, en el body de nuestro archivo, plantearemos el siguiente código:

```
<body>
    <h4>Hazme clic!</h4>
    <h4>Hazme clic a mi!</h4>
    <h4>Ahora intenta aquí!</h4>
</body>
```

A este HTML le vamos a incorporar el siguiente manejador de eventos, el cual nos permitirá hacer desaparecer los elementos `h4` cuando le hacemos clic:

```
<script>
    $(function () {
        $("h4").click(function () {
            $(this).fadeOut();
        });
    });
</script>
```

En nuestro navegador podemos apreciar lo siguiente: (las enumeraciones representan en donde hicimos clic en nuestro HTML).



Ahora vamos a hablar sobre dos manejadores nuevos: `mouseenter()` y `mouseleave()`. El primer método nos permite ejecutar una función, justo en el momento en que nuestro mouse o cursor entra en el espacio que ocupa el elemento definido. El segundo método se comporta de manera inversa, permitiéndonos realizar comandos cuando el mouse *sale* del espacio que ocupa el elemento definido.

Para ponerlos en práctica, vamos a plantear lo siguiente:

En nuestro HTML:

```
<body>
    <h4>Finalmente acá</h4>
    <h4>Luego aquí...</h4>
    <h4>Primero coloca el cursor aquí!</h4>
</body>
```

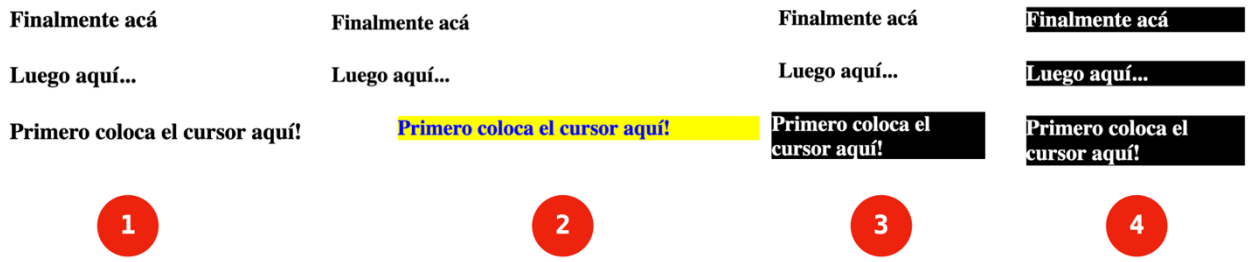
Y en nuestro `<script>`:

```
<script>
    $(function () {
        $("h4").mouseenter(function () {
            $(this).animate({
                marginLeft: '50px'
            }, "slow", function () {
                $(this).css({ "color": "blue", "backgroundColor": "yellow"
})
            })
        });
        $("h4").mouseleave(function () {
            $(this).animate({
                marginRight: '50px',
                width: '10em'
            }, "fast", function () {
                $(this).css({ "color": "white", "backgroundColor": "black"
})
            })
        });
    });
</script>
```

Con estos métodos manejadores de eventos establecemos que, al momento en que nuestro cursor entre en el espacio de cualquiera de los elementos `h4`, se cambiará el color del texto a azul, y hará que el color de fondo del elemento sea amarillo. Además, desplazará el elemento hacia la derecha unos 50 pixeles.

Cuando nuestro cursor sale del espacio que ocupan los `h4`, entonces cambiará el color de fondo y de texto a blanco y negro. Aparte, reducirá el ancho del elemento y lo desplazará a su posición original.

En nuestro navegador el resultado es el siguiente:



De esta forma hemos empezado a familiarizarnos con los manejadores de eventos, usando los métodos de jQuery.