Desafío - Javascript

- Para realizar este desafío debes haber estudiado previamente todo el material disponibilizado correspondiente a la unidad.
- Una vez terminado el desafío, comprime la carpeta que contiene el desarrollo de los requerimientos solicitados y sube el .zip en el LMS.
- Desarrollo desafío:
    - El desafío se debe desarrollar de manera Grupal.

Capítulos

El desafío está basado en los siguientes capítulos de la lectura:

- Introducción a JavaScript
- Avisos alert y console log
- Agregando JS
- Console log
- Declaraciones y sintaxis
- Tipos de datos y variables
- Operadores y asignación

Descripción

Este desafío proporcionará al estudiante los conocimientos básicos y necesarios para enfrentar el módulo introductorio a JavaScript. Ya que se desarrollaran distintos problemas y ejercicios que abordan dos formas de trabajar con JavaScript desde un documento HTML directamente, como es el caso de atributos para implementar JS directo desde las etiquetas HTML o mediante el uso especial de la etiqueta `<script>` para solamente escribir código JS. Los ejercicios solicitados al estudiante van desde la captura de eventos, la modificación de elementos y la utilización de funciones y métodos especiales de JavaScript, es decir, manipulación del DOM para modificar estilos, capturar eventos y leer elementos HTML.

Indicaciones

Este desafío se debe trabajar en grupos de 2 personas, y consiste en una guía en donde aprenderemos a modificar una página web con HTML y Javascript sin utilizar jQuery. Al final de este desafío debes tener los siguientes archivos con los requerimientos pedidos en cada sección. Durante la guía se dará la indicación de los requerimientos que debe tener cada uno.

- Desafío 1: cambiando-elemento-distinto-v2.html
- Desafío 2: cambiando-elemento-distinto-v3.html
- Desafío 3: menu-top.html
- Desafío 4: menu-top-v2.html
- Desafío 5: box-color-v2.html

Es importante indicar que para implementar JavaScript en los desafíos indicados anteriormente, debes utilizar la etiqueta "script" para desarrollar toda la logica de programacion. Permitiendo utilizar un solo archivo con la extensión de HTML pero con la lógica de programación en JavaScript en el mismo archivo. Esta técnica es una de las maneras de incluir JavaScript en nuestras páginas web, ya que en los siguientes módulos utilizarás la tercera forma de incluir JavaScript mediante archivos externos, técnica más utilizada y empleada actualmente.

Javascript y HTML

Podemos ejecutar código javascript en respuesta a eventos generados en nuestra página web, por ejemplo cuando se realiza un click.

```
<!-- Prueba1.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <title>HTML Con Javascript</title>
</head>
<body>
  <button onclick="alert('hola')"> Hey </button>
</body>
</html>
```

<button onclick="alert('hola')"> Hey </button>

Al hacer click en el botón veremos un mensaje de alerta. La directiva onclick también podemos agregarla a párrafos y otros elementos.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prueba 1</title>
</head>
<body>
  <p onclick="alert('hola')"> Hey </p>
</body>
</html>
```

Con javascript podemos obtener el elemento seleccionado utilizando la palabra reservada `this`.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prueba 1</title>
</head>
<body>
  <p onclick="console.log(this)"> Hey </p>
</body>
</html>
```

Al saber que elemento estamos seleccionado podemos modificarlo.

```
<!-- cambiocolor.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prueba 1</title>
</head>
<body>
  <p onclick="this.style.color='red'"> Hey </p>
</body>
</html>
```

Lo anterior se lee como a este párrafo del estilo cambiaremos el color a rojo. Podemos probarlo copiando y pegando el código y haciendo click en el párrafo.

Cambiando el CSS

Por regla general deberíamos evitar cambiar estilos directamente y ocupar clases, esto es especialmente importante a medida que nuestro sitio va creciendo pues nos permite mantener una coherencia visual y hacer sencillos los cambios de estilo en todas las secciones a la vez.

```
<!-- cambioclase.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prueba 1</title>
  <style>
      .warning{
        color:red;
      }
  </style>
</head>
<body>
  <p onclick="this.classList.add('warning')"> Solo Añadir </p>
  <p onclick="this.classList.toggle('warning')"> Toggle </p>
</body>
</html>
```

add agrega la clase, toggle la agrega o la quita dependiendo de si está presente o no, prueba haciendo click en el elemento con el inspector abierto para poder observar los cambios.

Tip: Ocupando transiciones en CSS puedes lograr efectos de fade in, prueba agregando a la clase warning: `transition: all 800ms;`

Combinando lo aprendido con funciones

```
<!-- cambioclase.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prueba 1</title>
  <style>
      .warning{
        color:red;
      }
  </style>
</head>
<body>
  <p onclick="this.classList.add('warning')"> Solo Añadir </p>
</body>
</html>
```

Selectores

Podemos seleccionar cualquier elemento de nuestra página web utilizando selectores. la forma más fácil de realizarlo es cuando el elemento tiene un id. Estos selectores son muy útiles cuando queremos modificar un elemento distinto al que estamos seleccionado, por ejemplo si hacemos click en un botón y un párrafo debe cambio de color.

```
<!-- selectores1.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Selectores</title>
</head>
<body>
  <p id="par-1"> 1 </p>
  <p id="par-2"> 2 </p>
  <p id="par-3"> 3 </p>
  <p id="par-4"> 4 </p>
</body>
</html>
```

Utilizando el inspector de elementos podemos ejecutar código javascript directamente en el tab console, ahí probemos escribiendo la siguiente expresión: `document.getElementById('par-1')`

También podemos modificar directamente un elemento desde la consola, probemos con:

`document.getElementById('par-1').style.color="red"` Al ejecutar esto deberíamso ver el primer párrafo de color rojo.

Cambiando un elemento distinto al seleccionado

Para lograr esto combinamos nuestros dos últimos aprendizajes, agregamos onclick al elemento disparador y dentro de este seleccionamos el otro elemento y lo modificamos.

```
<!-- cambiando-elemento-distinto.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Selectores</title>
</head>
<body>
  <p id="par-1"
onclick="document.getElementById('par-2').style.color='red'"> 1 </p>
  <p id="par-2"> 2 </p>
  <p id="par-3"> 3 </p>
  <p id="par-4"> 4 </p>
</body>
</html>
```

Nota: Existen distintos tipos de selectores, pero en este desafío solo trabajaremos con `getElementById()`

Requerimientos

1. Desafío 1: Crear un nuevo archivo a partir de lo anteriormente aprendido. Este archivo se debe llamar `cambiando-elemento-distinto-v2.html` y debes hacer que al hacer click al segundo párrafo el cuarto cambie a color azul y al hacerle click al tercero este cambie a color verde.

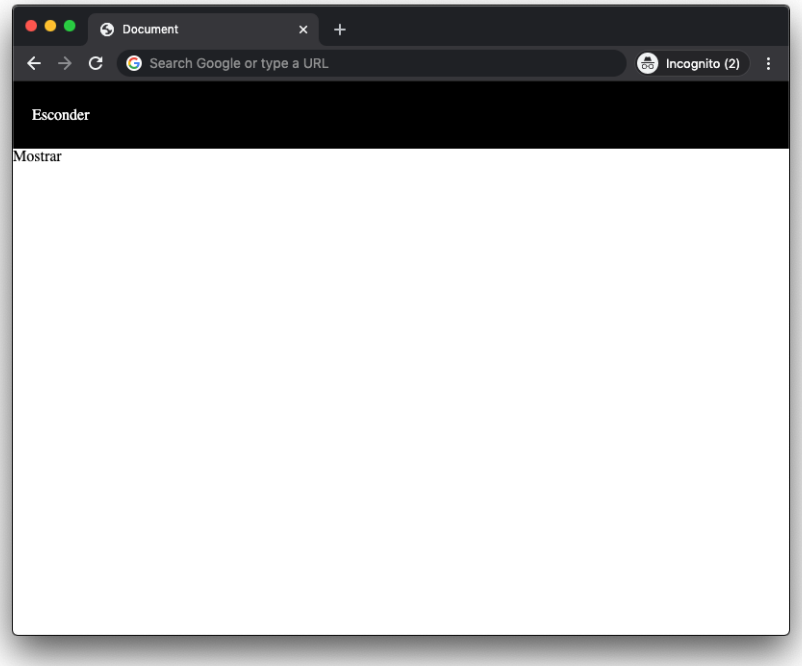
2. Desafío 2: A partir del Desafío 1, crear un nuevo archivo llamado `cambiando-elemento-distinto-v3.html` , en el que se debe utilizar clases para lograr todos los cambios de colores.

   Dato: También es posible ejecutar múltiples expresiones sobre un único elemento separándolos con `;`.

   ```
   <p id="par-1"
   onclick="document.getElementById('par-2').style.color='red';
   console.log('hola')"> 1 </p>
   ```

   Pero en este caso es mejor utilizar funciones las cuales estudiaremos más adelante.

3. Desafío 3: Crear un archivo denominado `menu-top.html` con un menú que pueda colapsar, ocupando un link y otro que restaure su posición original (No es necesario esconder el link de mostrar).



Se recomienda utilizar la siguiente base para el ejercicio. Se pueden agregar identificados adicionales si se requiere.

```
<!-- menu-top.html -->
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Document</title>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/meyer-reset/2.0/reset.css">

  <style>
    nav{
      height: 70px;
      background-color: black;
      color: white;
      padding: 0px;
      line-height: 70px;
      padding-left: 20px;
    }

    nav.hide{
      display: none;
    }

  </style>
</head>
<body>
  <nav id="nav">
    <a> Esconder </a>
  </nav>
  <a> Mostrar </a>
</body>
</html>
```

Consejo: Hay varias formas de resolver un ejercicio como este, pero se puede resolver únicamente con los elementos cubiertos en la lectura.

4. Desafío 4: Crear un archivo con el nombre de `menu-top-v2.html`, este debe ser creado como una copia del desafío anterior donde adicional a los requerimientos realizados se debe mostrar nuevamente el menú en el caso que se encuentre oculto.

Hasta el momento, la forma de emplear JavaScript como un atributo sobre las etiquetas HTML, no es muy utilizada cuando se trabaja con JavaScript puro (o vanilla JavaScript), ya que es una forma intrusiva de escribir código JavaScript. Pero, algunas librerías y framework de JavaScript, como Vue, React o Angular, implementan estas metodologías con atributos propios creados especialmente para realizar ciertas acciones, o se basan en los atributos tradicionales entre HTML y JavaScript. Por consiguiente, otra técnica o método para desarrollar el código anterior de una manera más limpia y clara, dejando las etiquetas de HTML sin ningún tipo de código en JavaScript, es mediante la utilización de la etiqueta <script> para agregar código en JS interno o externo, dicha técnica explicaremos y utilizaremos a partir de este momento.

En la siguiente sección aprenderemos sobre funciones en JavaScript y cómo utilizarlas para modificar nuestro HTML mediante la escucha de eventos y la ejecución de acciones dentro de las funciones, así como la utilización de la etiqueta <script> para escribir todo el código que sea JavaScript.

¿Cuáles son las funciones?

Las funciones son un mecanismo por el cual podemos agrupar múltiples instrucciones y llamarlas cuando las necesitemos.

```
function x() {
  console.log("hola")
  console.log("yo")
  console.log("soy")
  console.log("una")
  console.log("función")
}

x()
```

```
hola
yo
soy
una
función
```

Partes de una función

Una función primero se define y luego puede ser llamada cuantas veces sea necesario. Cuando definimos una función tenemos que darle un nombre, en el caso del ejemplo anterior la llamamos x.

```
// Definimos la función
function x() {
  console.log("hola")
  console.log("mundo")
}

// Llamamos a la función
x()
x()
x()
```

```
hola
mundo
hola
mundo
hola
mundo
```

Las funciones también pueden recibir parámetros.

```
// Definimos la función
function saludar(nombre) {
  console.log("hola", nombre);
}

saludar("Javiera");
saludar("Camila");
```

```
hola Javiera
hola Camila
```

¿Qué son los eventos y el DOM?

Los eventos son acciones u ocurrencias que suceden en una página web, las cuales pueden ser capturadas mediante la programación debido para reaccionar a ese evento en específico. Por ejemplo, si el usuario hace clic en un botón en una página web, es posible que desee responder a esa acción mostrando una ventana emergente o modificando un texto. Por consiguiente, los eventos se envían para notificar al código en JavaScript que algo ha ocurrido. Cada evento está representado por un objeto que se basa en la interfaz Event, y puede tener campos y/o funciones personalizadas adicionales para obtener más información acerca de lo sucedido. Los eventos más comunes son los de dispositivo de entrada, como el "click, que utilizamos en los primeros desafíos.

Por otra parte, el Modelo de Objetos del Documento (Document Object Model o DOM), es un modelo de documento que se carga en el navegador web y que representa el documento como un árbol de nodos, en donde cada nodo representa una parte del documento (puede tratarse de un elemento, una cadena de texto o un comentario). Por consiguiente, al tener en cuenta estos términos, podemos ahora capturar estos eventos de una manera más dinámica y limpia directamente desde el código en JavaScript, mediante la utilización de la etiqueta <script> para albergar el código. Como se muestra a continuación:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prueba 1</title>
</head>
<body>
  <p> Elementos de HTML  -  Clic Aquí</p>
  <script>
    //código en el lenguaje de programación JavaScript
    alert("mensaje desde JS");
 </script>
</body>
</html>
```

Si ejecutamos el código anterior, se podrá observar esta vez que la ventana abre automáticamente sin el control o la ejecución de un clic por parte del usuario cuando se carga la página web en el navegador. Ahora, si queremos lograr el mismo resultado que el ejemplo original mostrado al inicio de este documento, es decir, que la ventana emergente aparezca cuando el usuario haga un clic sobre el texto, se debe implementar los evento "click", el cual, debe ser escuchado en nuestro código mediante la función "addEventListener", el cual, permite capturar los eventos indicados cuando se ejecutan en un elemento en específico. Por consiguiente, debemos primeramente leer el elemento al cual queremos aplicar el evento, es decir, el párrafo " Elementos de HTML - Clic Aquí ". Para leer este elemento utilizaremos la ayuda de los selectores como el id y el método "getElementById" con el que trabajamos anteriormente. Ahora debemos modificar el código anterior agregando un "id" a la etiqueta , realizando los siguientes pasos:

Paso 1:

Modifica la etiqueta en el HTML, agregando un id con nombre "elementoP", por ejemplo.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prueba 1</title>
</head>
<body>
  <p id="elementoP"> Elementos de HTML  -  Clic Aquí</p>
  <script>
    //código en el lenguaje de programación JavaScript
    alert("mensaje desde JS");
 </script>
</body>
</html>
```

Paso 2:

Luego, dentro de las etiquetas <script>, procederemos a leer el elemento por su id mediante el método “getElementById” guardando el resultado en una variable, quedando el código de la siguiente manera:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prueba 1</title>
</head>
<body>
  <p id="elementoP"> Elementos de HTML  -  Clic Aquí</p>
  <script>
    var elementoP = document.getElementById('elementoP');
 </script>
</body>
</html>
```

Paso 3:

Ya que tenemos el elemento guardado, procedemos a agregarle el escucha para que ejecute una función cuando el evento que deseamos escuchar, en este caso el “click” lo active el usuario y muestre una ventana de alerta con el mensaje: “Mensaje desde JS activado por evento click”.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prueba 1</title>
</head>
<body>
  <p id="elementoP"> Elementos de HTML  -  Clic Aquí</p>
  <script>
    var elementoP = document.getElementById('elementoP');
    elementoP.addEventListener('click',function () {
            alert("Mensaje desde JS activado por evento click")
    });
 </script>
</body>
</html>
```

Al ejecutar el código anterior, se podrá observar que la venta de alerta solo ocurre cuando se hace un clic sobre el párrafo. Por ende, esta metodología o forma de trabajo aplicaremos a partir de este momento, es decir, primero leyendo el elemento HTML mediante un selector, luego aplicando un escucha con el evento "click", y finalmente dentro de la función del escucha realizar las acciones correspondientes.

Para el siguiente ejemplo necesitaremos una página web que llamaremos box.html que tenga un div con ancho y alto definido y un color de fondo, como se muestra a continuación. Importante: para no utilizar otros archivos externos, trabajaremos con la etiqueta `<style>` para agregar los estilos correspondientes, igualmente con el código en JavaScript que será realizado dentro de las etiquetas `<script>`.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Colores</title>
  <style>
    #box {
      width: 170px;
      height: 170px;
      background-color: red;
    }
  </style>
</head>
<body id="body">
  <div id="box"></div>
</body>
</html>
```

Al ejecutar el archivo con el código anterior en el navegador web, obtendremos el siguiente resultado:

Ahora bien, para cambiar el color de fondo de la caja necesitamos obtener el elemento, referirnos a su estilo y cambiarlo (aunque también podemos hacerlo con clases). Por lo tanto debemos implementar los siguientes pasos:

a):

Lo primero es agregar las etiquetas `<script>` que nos permitirán desarrollar el código en JavaScript, luego leer el elemento que vamos a querer modificar por su selector, en este caso un id con el nombre de "box".

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Colores</title>
  <style>
    #box {
      width: 170px;
      height: 170px;
      background-color: red;
    }
  </style>
</head>
<body id="body">
  <div id="box"></div>
  <script>
        var box = document.getElementById('box');
  </script>
</body>
</html>
```

b):

Ahora que ya tenemos identificado y guardado el elemento que deseamos modificar, utilizando las propiedades del HTML DOM Style Object, en donde el objeto Style representa una declaración de estilo individual, procederemos a modificar el estilo del color de fondo del elemento, originalmente se encuentra en rojo, y por ejemplo, lo cambiaremos a verde, para eso utilizaremos `style.backgroundColor='green'`.

Quedando el código de la siguiente manera:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Colores</title>
  <style>
    #box {
      width: 170px;
      height: 170px;
      background-color: red;
    }
  </style>
</head>
<body id="body">
  <div id="box"></div>
  <script>
        var box = document.getElementById('box');
        box.style.backgroundColor = "green";
  </script>
</body>
</html>
```

c):

Al ejecutar el archivo box.html en el navegador web, el resultado obtenido ahora será:

Continuemos con el ejercicio anterior, pero esta vez agregaremos más elementos al HTML para interactuar con el evento “click”. Para ello, si quisiéramos crear botones para cambiar los colores a verde, amarillo y rojo tendríamos que repetir mucho código, aquí es donde las

funciones nos pueden ayudar a escribir menos código y llevar un mejor orden. Por lo tanto, el ejemplo solicita crear un cuadrado de color rojo, mas tres botones que permitan al usuario modificar el color del cuadrado a verde, amarillo o regresarlo a su color por defecto que es el rojo. Para poder realizar esto debemos seguir los siguientes pasos:

1: Comencemos por crear los tres botones, agregandolos exactamente al final del elemento que contiene el cuadrado, agregando un id a cada uno de estos botones creados con la etiqueta `<button>`. Seguidamente borramos todo el código JS escrito dentro de las etiquetas `<script>`, ya que vamos a crear un nuevo código. Quedando de la siguiente manera:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Colores</title>
  <style>
    #box {
      width: 170px;
      height: 170px;
      background-color: red;
    }
  </style>
</head>
<body id="body">
  <div id="box"></div>
  <button id="green"> Verde </button>
  <button id="yellow"> Amarillo </button>
  <button id="red"> Rojo </button>
  <script>
  </script>
</body>
</html>
```

2: Ya contamos con los botones en el HTML, cada uno de ellos con un ID, esto permitirá ahora escribir el código en JS para leer cada uno de los elementos HTML y a su vez poder escuchar cada evento ejecutado por cada botón. Todos estos eventos ejecutarán una función que a su vez llamará a una única función que permitirá modificar el color de fondo del cuadrado dependiendo del botón presionado. Esta función llevará por nombre `cambiar_color`. La cual, recibirá el color seleccionado por el usuario y modifica el elemento con ese color:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Colores</title>
  <style>
    #box {
      width: 170px;
      height: 170px;
      background-color: red;
    }
  </style>
</head>
<body id="body">
  <div id="box"></div>
  <button id="green"> Verde </button>
  <button id="yellow"> Amarillo </button>
  <button id="red"> Rojo </button>
  <script>
    var caja = document.getElementById("box");
        var verde = document.getElementById('green');
        var amarillo = document.getElementById('yellow');
        var rojo = document.getElementById('red');

        verde.addEventListener('click', function () {
               cambiar_color("green");
        });
        amarillo.addEventListener('click', function () {
               cambiar_color("yellow");
        });
        rojo.addEventListener('click', function () {
               cambiar_color("red");
        });
  </script>
</body>
</html>
```

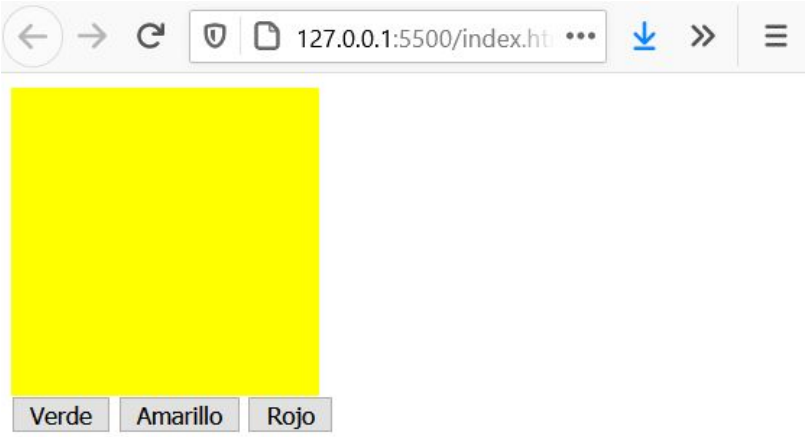
3: solo queda crear la función cambiar_color() para que reciba un valor (color seleccionado por el usuario) y modifique en el elemento el color del fondo mediante el estilo.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Colores</title>
  <style>
    #box {
      width: 170px;
      height: 170px;
      background-color: red;
    }
  </style>
</head>
<body id="body">
  <div id="box"></div>
  <button id="green"> Verde </button>
  <button id="yellow"> Amarillo </button>
  <button id="red"> Rojo </button>
  <script>
    var caja = document.getElementById("box");
        var verde = document.getElementById('green');
        var amarillo = document.getElementById('yellow');
        var rojo = document.getElementById('red');

        verde.addEventListener('click', function () {
               cambiar_color("green");
        });
        amarillo.addEventListener('click', function () {
               cambiar_color("yellow");
        });
        rojo.addEventListener('click', function () {
               cambiar_color("red");
        });

function cambiar_color(color){
               caja.style.backgroundColor = color;
        }
  </script>
</body>
</html>
```

4: Gracias a lo anterior, ahora cada vez que el usuario haga un clic sobre uno de los tres botones, el color del cuadrado se modificara al color seleccionado. Por consigueinte, al ejecutar el código anterior y hacer un clic sobre el boton amarillo, el resultado en el navegador web sera:



5. Crea el archivo `box-color-v2.html`. Dentro de él agrega dos botones con la finalidad de modificar el tamaño de la caja principal, es decir, un botón para modificar la dimensión actual de la caja por el tamaño de 400x400 y otro botón para modificar la dimensión de la caja actual por el tamaño de 100x100. Los textos de los botones deben especificar la dimensión. Deberás crear una segunda función debajo de la primera para lograrlo.

Evaluación de aprendizaje

- Desafío 1: Aprender la sintaxis de onlick.
- Desafío 2: Modificar un código existente para alterar las propiedades de elementos a partir de sus clases.
- Desafío 3: Implementar una funcionalidad con JavaScript a partir desde el HTML utilizando instrucciones previamente aprendidas.
- Desafío 4: Utilizar selectores para modificar el comportamiento de un elemento distinto al seleccionado.
- Desafío 5: Crear una función desde cero en JavaScript para cambiar el tamaño de un elemento existente.