

# Integrando JavaScript en el sitio web

## Introducción

¡Te damos la bienvenida a una nueva clase! 

En la clase anterior aprendimos sobre los conceptos básicos de JavaScript y creamos una función simple para mostrar una alerta en una página HTML.

Hoy exploraremos formas más avanzadas de integrar JavaScript en un sitio web; veremos las **estructuras de control de flujo** y el **Modelo de Objeto de Documento (DOM)**, entenderemos qué son y cómo gestionar *eventos y oyentes de eventos*, manejo y validación de *formularios*.

Al final de esta clase, lograrás implementar formularios y su validación correspondiente en la página HTML sobre la que vienes trabajando.

¡Empecemos! 

## Estructuras de control de flujo

Las **estructuras de control de flujo** son utilizadas para controlar el flujo de ejecución del programa y tomar decisiones basadas en ciertas condiciones.

Algunos ejemplos de *estructuras de control de flujo* incluyen:

- **Estructuras condicionales: if, else, else if** → Se usan para ejecutar diferentes bloques de código dependiendo de si una condición es verdadera o falsa.
- **Estructuras de bucles: for, while, do-while** → Se usan para repetir un bloque de código hasta que se cumpla una condición.
- **Estructuras de control de excepciones: try, catch, finally** → Se usan para manejar errores en el código.

Las estructuras de control de flujo **se pueden anidar unas dentro de otras, lo que permite crear programas complejos y avanzados**. También es importante tener en cuenta la eficiencia del código y evitar anidaciones excesivas o redundantes que puedan afectar el rendimiento del programa.

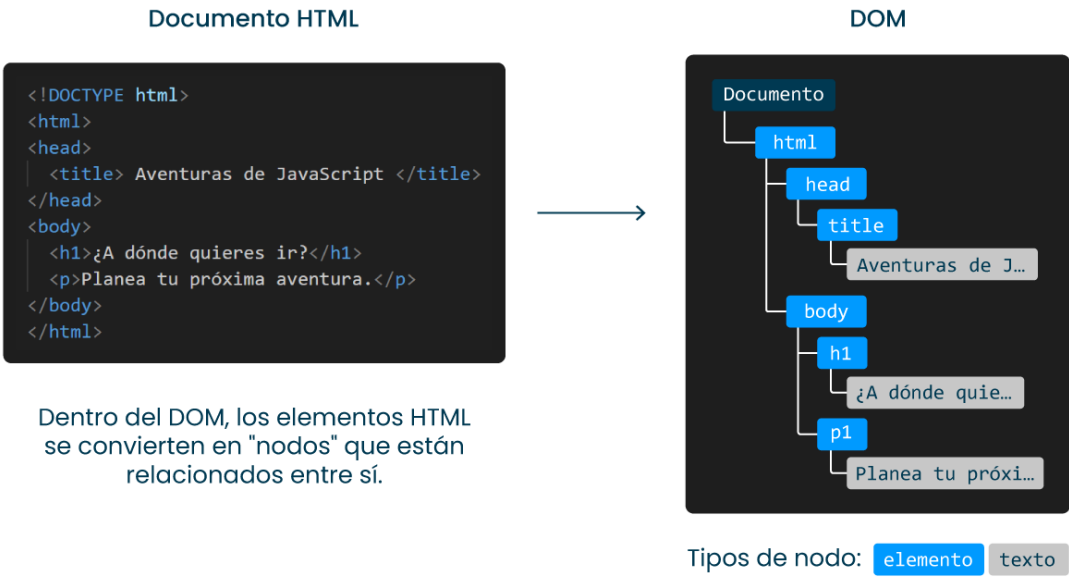
💡 *Para poder profundizar más sobre este tema, vas a poder acceder al material complementario de las [Tipo de “estructuras de control de flujo”](#).*

## Manipulación del DOM

El **DOM** (“Document Object Model” o “Modelo de Objetos del Documento”) es una representación de la estructura de un documento HTML, que se organiza como un árbol de objetos (“nodos”), donde cada objeto representa una parte del documento, como un elemento o un atributo.

En otras palabras, es una interfaz de programación de aplicaciones (API) que proporciona una forma estándar para que los programadores accedan y manipulen los elementos HTML de un documento web.

### ¿Cómo se ve la estructura del DOM?



**JavaScript se utiliza para acceder a los objetos del DOM y manipularlos, lo que permite actualizar dinámicamente el contenido y la estructura de una página web en tiempo real.**

La manipulación del DOM puede incluir la creación, eliminación, modificación y movimiento de elementos y atributos en una página web.

 *Es importante tener en cuenta que la manipulación del DOM debe hacerse con cuidado y no en exceso, ya que puede afectar el rendimiento de la página.*

## Encontrando los elementos HTML

Como mencionamos, JavaScript se puede usar para manipular elementos HTML y, para hacerlo, primero se deben encontrar esos elementos.

Se pueden encontrar a través de las siguientes maneras:

- Encontrando *elementos HTML* por su **id**

```
const elemento = document.getElementById( "id" );
```

- Encontrando *elementos HTML* por su **nombre de etiqueta**

```
const paragraphs = document.getElementsByTagName( "p" );
```

- Encontrando *elementos HTML* por su **nombre de clase**

```
const myClasses = document.getElementsByClassName( "miClase" );
```

## Propiedades y Métodos del DOM

En el DOM, tanto los métodos como las propiedades son utilizados para manipular y acceder a los *elementos HTML* en una página web.

Las **propiedades del DOM** son valores que pueden ser leídos o modificados directamente. Algunos ejemplos de propiedades incluyen `.innerText`, `.innerHTML`, `.value`, `.src`, `.href`, `.id`, `.className`, `.style`, entre otras. Estas propiedades pueden ser accedidas utilizando la notación de punto o la notación de corchetes.

Por otro lado, los **métodos del DOM** son funciones que se utilizan para realizar una acción en un elemento HTML, como agregar un nuevo elemento, eliminar un elemento existente, cambiar su estilo, etc. Algunos ejemplos de métodos incluyen

`createElement()`, `appendChild()`, `removeChild()`, `setAttribute()`, `addEventListener()`, entre otros.

Demasiada información, ¿verdad? “¡Que no panda el cúnico!” 😊 Para una mayor claridad de estos conceptos, veamos el siguiente video:

 [Ver video](#)

## Eventos



Los **eventos** son *acciones que ocurren en la página web o en el navegador*, como hacer clic en un botón, pasar el mouse sobre una imagen, escribir en un campo de texto, entre otros.

Estos eventos pueden ser detectados y manejados a través de JavaScript para realizar acciones específicas. Es decir, **son una herramienta poderosa para interactuar con el usuario y mejorar su experiencia en la página web.**

## Tipos de eventos

En JavaScript hay una gran cantidad de eventos disponibles. Algunos de los más comunes son:

- **Eventos de ratón:** Se activan cuando el usuario interactúa con el mouse, como hacer clic, mover el mouse, presionar y soltar el botón del mouse, etc.
- **Eventos de teclado:** Se activan cuando el usuario interactúa con el teclado, como presionar una tecla, soltar una tecla, mantener presionada una tecla, etc.
- **Eventos de formulario:** Se activan cuando el usuario interactúa con un formulario, como enviar un formulario, hacer clic en un botón de envío, cambiar el valor de un campo de formulario, etc.
- **Eventos de ventana:** Se activan cuando el usuario interactúa con la ventana del navegador, como cambiar el tamaño de la ventana, minimizar o maximizar la ventana y cerrar la ventana, etc.

 *Es importante mencionar que la mayoría de los eventos son "disparados" por los usuarios, pero también se pueden activar mediante código. Para conocer más los diferentes eventos, puedes acceder al siguiente link*  [Tipos de eventos en JavaScript.](#)

El **manejo de eventos** es una parte importante de la programación en Javascript, y permite a los desarrolladores crear interacciones dinámicas y personalizadas en

sus páginas o aplicaciones web.

A continuación, lo veremos en más detalle...

## Manejo de eventos

Para manejar un evento en JavaScript se utiliza un oyente de eventos (*"event listener"*).

Un **oyente de eventos** es una función que se encarga de "escuchar" estos eventos específicos y ejecutar ciertas acciones en función de lo ocurrido. Algunos ejemplos podrían ser:

- Cambio de contenido al hacer clic en una pestaña.
- Una validación al enviar un formulario.
- Expandir y colapsar elementos al hacer clic en un botón.
- Cargar datos adicionales al llegar al final de una página (scroll infinito).
- Cambiar el estilo de la barra de navegación al desplazarse.

El método *"addEventListener()"* es el que se utiliza para agregar un oyente de eventos a un elemento HTML. Este método toma dos argumentos:

- **Tipo de evento** que se desea escuchar.
- **Función** que se ejecutará cuando se active el evento.

Todo ok con la teoría, pero veamos cómo ponerla en práctica:

Supongamos que tenemos un botón en nuestra página HTML con el siguiente código:

```
<button id="miBoton">Haz clic aquí</button>
```

Queremos que cuando el usuario haga clic en el botón, se muestre un mensaje de alerta en la pantalla. Para hacer esto, necesitamos añadir un "oyente de eventos" al botón, que esté "escuchando" cuando se hace clic en el mismo.

Esto se puede hacer en JavaScript de la siguiente manera:

```
// Obtenemos el botón por su id
let boton = document.getElementById("miBoton");

// Añadimos un oyente de eventos que "escuche" el evento "click"
boton.addEventListener("click", function() {
    alert("¡Haz hecho clic en el botón!");
});
```

En el código anterior, primero obtenemos el botón por su *id* utilizando el método `getElementById()` para almacenarlo en la variable `boton`. Luego, utilizamos el método `addEventListener()` para añadir un oyente de eventos al botón. Este método toma dos argumentos en consideración:

- Tipo de evento que se desea escuchar → en este caso, el evento `"click"`.
- Función que se ejecutará cuando se produzca ese evento → una *función anónima que muestra un mensaje de alerta en la pantalla*.

Ahora, cada vez que el usuario haga clic en el botón se ejecutará la función que hemos definido y se mostrará el mensaje de alerta.

Muy bien, ahora es tu turno:

## ¡Manos a la obra!

1. Abrir Visual Studio Code y crear un nuevo archivo HTML (`"index.html"`).

2. Agregar el siguiente código HTML en el archivo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Actividad de oyente de eventos en JavaScript</title>
  </head>
  <body>
    <button id="miBoton">Haz clic aquí</button>

    <script src="index.js"></script>
  </body>
</html>
```

3. Crear un nuevo archivo JavaScript (*"index.js"*) y "encontrar" el elemento HTML al que quieres agregarle el oyente de eventos. Para ello, vas a poder usar el método *"getElementById"* y pasarle el ID del botón como argumento:

```
// Encuentra al elemento HTML (el botón) mediante su ID
const miBoton = document.getElementById('miBoton');
```

4. Definir la *función* que se ejecutará cuando se haga clic en el botón. En este caso, se va a mostrar una alerta en la pantalla. La función puede tener cualquier nombre, por ejemplo, *"handleClick"*:

```
// Define la función que se ejecutará cuando se haga clic en el botón
function handleClick() {
  alert('¡Hola! Has hecho clic en el botón.');
```

5. Agregar un oyente de eventos al botón. Para ello, se utiliza el método *"addEventListener"* y se pasa el evento que se quiere escuchar, en este caso *"click"*, y la función que se va a ejecutar cuando se dispare el evento, *"handleClick"*:

```
// Agrega un oyente de eventos para escuchar el clic del botón
miBoton.addEventListener('click', handleClick);
```

6. Abrir el archivo *"index.html"* en tu navegador web.

7. Hacer clic en el botón de la página web.

¡Listo! Debería aparecer una alerta con el mensaje *"¡Hola! Has hecho clic en el botón."*

## Formularios

Un **formulario** es un elemento que se utiliza para recopilar información ingresada por el usuario como nombres, correos electrónicos, contraseñas, comentarios, entre otros. Son esenciales en las páginas web para permitir interacciones entre el usuario y el servidor, como iniciar sesión, registrarse, enviar comentarios y realizar pedidos en línea.



## Formularios

Nombre

Apellido

Email

Mensaje

Enviar

Un formulario típico de HTML se crea utilizando la etiqueta `<form>` y contiene varios elementos de entrada como `<input>`, `<textarea>`, `<select>` y `<button>` que facilitan la recopilación de datos de los usuarios.

Veamos un ejemplo de una estructura HTML básica para un formulario:



```

<form>
  <!-- Campo para nombre -->
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" name="nombre">

  <!-- Campo para correo electrónico -->
  <label for="email">Correo electrónico:</label>
  <input type="email" id="email" name="email">

  <!-- Campo para mensaje -->
  <label for="mensaje">Mensaje:</label>
  <textarea id="mensaje" name="mensaje"></textarea>

  <!-- Campo para selección de país -->
  <label for="pais">País:</label>
  <select id="pais" name="pais">
    <option value="argentina">Argentina</option>
    <option value="chile">Chile</option>
    <option value="mexico">México</option>
    <option value="peru">Perú</option>
  </select>

  <!-- Botón de envío -->
  <button type="submit">Enviar</button>
</form>

```

- **<form>**: Envuelve todos los campos del formulario. Se utiliza para agrupar los elementos relacionados en un solo bloque.
- **<label>**: Se utiliza para etiquetar los campos del formulario. El atributo "for" se utiliza para vincular la etiqueta con el campo.
- **<input>**: Se utiliza para crear campos de entrada de texto. En este caso, se utilizó el tipo "text" para el campo de nombre y el tipo "email" para el campo de correo electrónico. El atributo "id" se utiliza para identificar el campo y el atributo "name" se utiliza para definir el nombre del campo en el servidor.
- **<textarea>**: Se utiliza para crear áreas de texto más grandes. En este caso, se utiliza para el campo de mensaje.
- **<select>**: Se utiliza para crear listas desplegables. En este caso, se utiliza para el campo de selección de país. Cada opción dentro del elemento <select> se crea con el elemento <option>, que contiene un valor y el texto a mostrar en la lista.
- **<button>**: Se utiliza para crear botones en el formulario. En este caso, se utiliza para el botón de envío del formulario. El atributo "type" se establece en "submit" para enviar el formulario.

¡Genial! Ahora, pasemos a ver cómo manejar y validar los formularios con JavaScript...

# Manejo y validación de formularios

El **manejo y validación de formularios** es un aspecto clave para garantizar que los datos ingresados por los usuarios sean correctos y estén en el formato esperado antes de ser procesados o almacenados en un servidor.

El diagrama muestra un formulario de registro con cuatro campos de entrada:

- Usuario:** Contiene el texto "john123". Tiene un icono de error (X roja) a la derecha. Debajo del campo, un mensaje de error indica: "El usuario tiene que ser de 4 a 16 dígitos y solo puede contener números, letras y guion bajo."
- Nombre:** Contiene el texto "Carlos Arturo". Tiene un icono de éxito (✓ verde) a la derecha.
- Contraseña:** Contiene diez puntos para ocultar el texto. Tiene un icono de éxito (✓ verde) a la derecha.
- Repetir Contraseña:** Contiene diez puntos para ocultar el texto. Tiene un icono de error (X roja) a la derecha. Debajo del campo, un mensaje de error indica: "Ambas contraseñas deben ser iguales."

💡 *Esta práctica mejora la calidad de los datos y la experiencia del usuario, al tiempo que minimiza posibles problemas o errores en el procesamiento de la información.*

El **manejo de formularios** en JavaScript implica la *interacción con elementos de formulario HTML* como campos de entrada, botones, casillas de verificación y listas desplegables.

Algunas tareas comunes en el manejo de formularios incluyen:

- Acceder a los valores de los campos de entrada.
- Cambiar dinámicamente el contenido o atributos de un elemento de formulario.
- Enviar un formulario o realizar una acción cuando se hace clic en un botón.
- Utilizar eventos de formulario, como submit, change, focus y blur, para detectar y responder a las interacciones del usuario.
- Validación de formularios

La **validación de formularios** es el *proceso de verificar que los datos ingresados por el usuario cumplen con ciertos criterios o reglas*. Estos pueden incluir:

- Comprobar que un campo es obligatorio y no está vacío.
- Validar que el valor ingresado sea del tipo de datos esperado como un número, una dirección de correo electrónico o una fecha.
- Verificar que el valor esté dentro de un rango específico como una cantidad mínima y máxima, o una longitud mínima y máxima.

- Comprobar que el valor cumpla con un patrón específico como una expresión regular para una contraseña segura o un formato de número de teléfono.

## Desafío del día

El desafío de hoy consiste en **agregar un formulario y su validación en la página HTML** sobre la que viniste trabajando en los encuentros anteriores.

⚠ Antes de comenzar, recuerda **crear una nueva carpeta** con el nombre de la sección “Sección 5 - Integrando JavaScript en el sitio web” y hacer una copia del documento “.html” sobre el que venías trabajando.

Tu código debería ser similar al siguiente:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
  </head>
  <body>
    <div class="container">
      <h1>Bienvenidos a mi primer Website</h1>
      <div class="row">
        <div class="column">
          <h2>Presentación</h2>
          <p>Soy [Tu nombre], ¡y esta es mi primera página web!</p>
          <p>
            Estoy aprendiendo desarrollo web para crear sitios web
            visualmente atrapantes y funcionales.
          </p>
        </div>
        <div class="column">
          <h2>Lo que se viene...</h2>
          <p>
            A lo largo de este curso estaré aprendiendo HTML, CSS
            y JavaScript. Estos lenguajes me ayudarán a crear una
            base sólida en el desarrollo web front-end.
          </p>
        </div>
      </div>
    </div>
    <button onclick="showAlert()">¡Haz click en mí!</button>
    <script src="script.js"></script>
  </body>
</html>
```

Ahora sí, ¡en marcha!

1.Crear un formulario HTML con campos de entrada y un botón de envío:

```
<form id="myForm">
  <label for="email">Correo electrónico:</label>
  <input type="email" id="email" name="email" required>
  <button type="submit">Enviar</button>
</form>
```

2.Agregar un oyente de eventos al formulario para manejar el evento "submit":

```
const form = document.getElementById('myForm');
form.addEventListener('submit', function(event) {
  event.preventDefault();
  validateForm();
});
```

En este ejemplo se utiliza *"event.preventDefault()"* para evitar que el formulario se envíe de la manera predeterminada, lo que permite realizar la validación antes de enviar los datos.

3.Sumar una función de validación que verifique los criterios deseados:

```
function validateEmail(email) {
  const regex = /^[^\s@]+@[^\s@]+\.[^\s@]{2,7}$/;
  return regex.test(email)
}

function validateForm() {
  const emailInput = document.getElementById('email');
  const email = emailInput.value;

  if (!validateEmail(email)) {
    alert('Por favor ingrese un correo electrónico válido.');
```

```
  } else {
```

```
    alert('Correo electrónico enviado correctamente.');
```

```
  }
```

```
}
```

Aquí la función *"validateEmail()"* indica que el mail contenga un "@" y un "." para ser válido. No te preocupes por cómo crear esta función podrás aprender a usarla en nuestro curso de Frontend y FullStack.

**¡Desafío terminado!** 🎉

# Resolución del desafío

En el siguiente video te compartimos un paso a paso de cómo resolver el desafío anterior:

 [Resolución de Ejercicio | Integrando JavaScript en el sitio web | Egg](#)

¡Felicidades! 

Has integrado con éxito conceptos avanzados de JavaScript junto con la creación y validación de un formulario. Estas técnicas te ayudarán a crear páginas web más interactivas que respondan a las acciones del usuario y brinden una experiencia más enriquecedora para sus visitantes.

En la próxima clase practicaremos los temas vistos hasta el momento para poder incorporarlos de una forma 100% práctica y con ejemplos de uso en el día a día de los programadores.

## Valida tus conocimientos

Te compartimos los enlaces del **material complementario** para que puedas continuar profundizando sobre el siguiente tema en el momento que lo desees:

- [Tipos de “Estructuras de control de flujo”](#)
- [Tipos de eventos en JavaScript](#)

Por último te proponemos realizar la autoevaluación para poner a prueba los principales aprendizajes del día de hoy. ¿Vamos?

 [Realizar test](#)

Ten en cuenta que si has llegado hasta aquí ya has cumplido con el objetivo del encuentro.

¡Hasta la próxima! 

## Mapa de conceptos vistos

