



IT Academy
by KIBERNUM

Programa: Desarrollo de Aplicaciones Front-End

Módulo 2: Fundamentos de Desarrollo Front-End



Aprendizaje esperado

Utilizar código JavaScript para la personalización de eventos sencillos dentro de un documento html dando solución al problema planteado.



**¿Qué saben sobre JavaScript y
cómo creen que influye en las
páginas web modernas?**





JavaScript: El Lenguaje del Desarrollo Web

JavaScript es un lenguaje de programación fundamental para el desarrollo web moderno, permitiendo la creación de páginas dinámicas e interactivas. Este lenguaje, que se ejecuta de forma nativa en todos los navegadores web, ha transformado la experiencia del usuario, haciendo que las interacciones sean más fluidas y rápidas.

Historia de JavaScript



01. **1995**
Brendan Eich crea JavaScript en Netscape Communications Corporation, inicialmente llamado Mocha.
02. **2005**
AJAX revoluciona la interacción web, permitiendo el manejo asíncrono de datos y mejorando la experiencia del usuario.
03. **2009**
Node.js se introduce, expandiendo el alcance de JavaScript al desarrollo del lado del servidor.

Relevancia de JavaScript



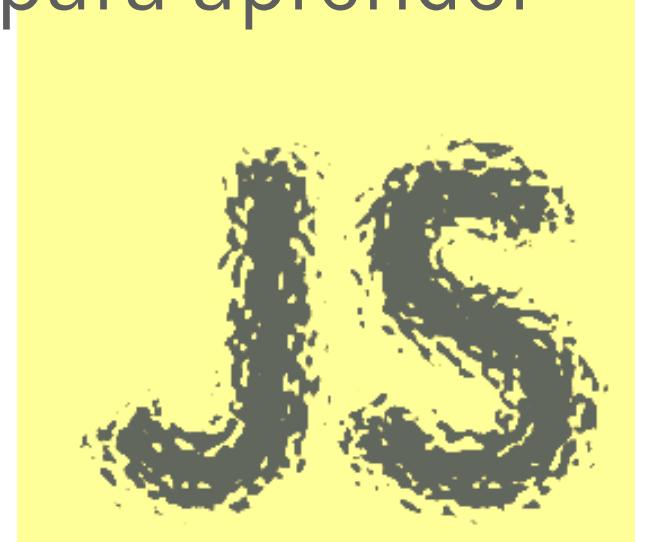
¿Por qué es importante JavaScript?

JavaScript es el **lenguaje esencial del desarrollo web moderno**, y su relevancia se refleja en que **más del 98 % de los sitios web lo utilizan**.

Gracias a él, las páginas web dejan de ser estáticas y se convierten en **experiencias dinámicas, interactivas y personalizadas** para cada usuario.

- ◆ **Presencia universal:** está integrado en todos los navegadores y sistemas operativos, sin necesidad de instalación.
- ◆ **Versatilidad:** permite programar tanto el **front-end** (interfaz) como el **back-end** mediante entornos como Node.js.
- ◆ **Puerta de entrada al desarrollo web:** dominar JavaScript abre el camino para aprender frameworks, librerías y arquitecturas más avanzadas.

💡 *Aprender JavaScript es comprender cómo “piensa” la web: su lógica, eventos y estructura.*



Relevancia de JavaScript



Interactividad y dinamismo

JavaScript otorga **vida** a las páginas web al permitir que **respondan de forma inmediata** a las acciones del usuario.

Cada clic, movimiento del mouse o pulsación de tecla puede desencadenar una reacción que mejora la **usabilidad** y la **experiencia** del visitante.

Ejemplos de interactividad cotidiana:

- Mostrar u ocultar menús de navegación.
- Validar formularios en tiempo real.
- Actualizar contenido sin recargar la página (*AJAX / Fetch API*).
- Animar transiciones, carruseles o efectos visuales.

Esta capacidad de respuesta instantánea convierte a JavaScript en el **motor del dinamismo web**, transformando un sitio estático en una aplicación fluida e intuitiva.

⭐ *Con JavaScript, las páginas dejan de “mostrar información” y comienzan a “interactuar con las personas”.*



Relevancia de JavaScript



Un ecosistema robusto y en evolución

El éxito de JavaScript no se limita al lenguaje: se apoya en un **ecosistema enorme y en constante crecimiento** que impulsa la innovación digital.

- ◆ **Frameworks y librerías:**

React, Angular y Vue.js facilitan la creación de **interfaces dinámicas y escalables** con estructuras modulares.

- ◆ **Entornos de ejecución:**

Con Node.js es posible usar JavaScript en el servidor, integrando lógica de negocio, APIs y bases de datos.

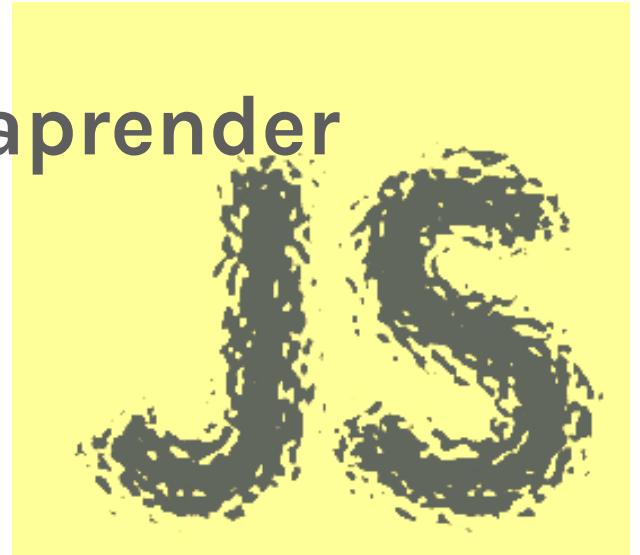
- ◆ **Comunidad activa:**

Millones de desarrolladores contribuyen a librerías, herramientas y recursos que fortalecen la evolución del lenguaje.

- ◆ **Aprendizaje continuo:**

Su documentación abierta y la abundancia de tutoriales y foros hacen que **aprender JavaScript sea accesible y colaborativo**.

 *JavaScript no solo es un lenguaje: es un ecosistema que impulsa la innovación tecnológica global.*



Reto: Explorando JS en la Web

◆ **Instrucciones:** Visita diferentes sitios web y observa si hay elementos que cambian o responden sin recargar la página.

◆ **Ejemplos:**

- Formularios interactivos
- Efectos de desplazamiento
- Menús desplegables
- Notificaciones en tiempo real



Capacidades y Limitaciones de JavaScript



1

¿Qué puede hacer?

Manipular el DOM

Manejar eventos

Comunicación asíncrona

Validación de formularios

2

¿Qué no puede hacer?

Acceso a archivos locales

Limitaciones de seguridad

Acceso a hardware



Incorporación de JavaScript en HTML

JavaScript en Línea

Incluir JavaScript directamente en las etiquetas HTML utilizando atributos como onclick.

JavaScript Interno

Escribir JavaScript dentro de la misma página HTML utilizando la etiqueta.

JavaScript Externo

Para proyectos más grandes o complejos, es una buena práctica colocar el código JavaScript en un archivo separado.

Ejemplo: JavaScript en línea

Puedes incluir JavaScript directamente en las etiquetas HTML utilizando atributos como onclick para ejecutar código cuando se hace clic en un botón.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript en línea</title>
</head>
<body>
    <h1>Ejemplo de JavaScript en línea</h1>
    <button onclick="alert('¡Hola, Mundo!')">Haz clic aquí</button>
</body>
</html>
```

Ejemplo: JavaScript interno

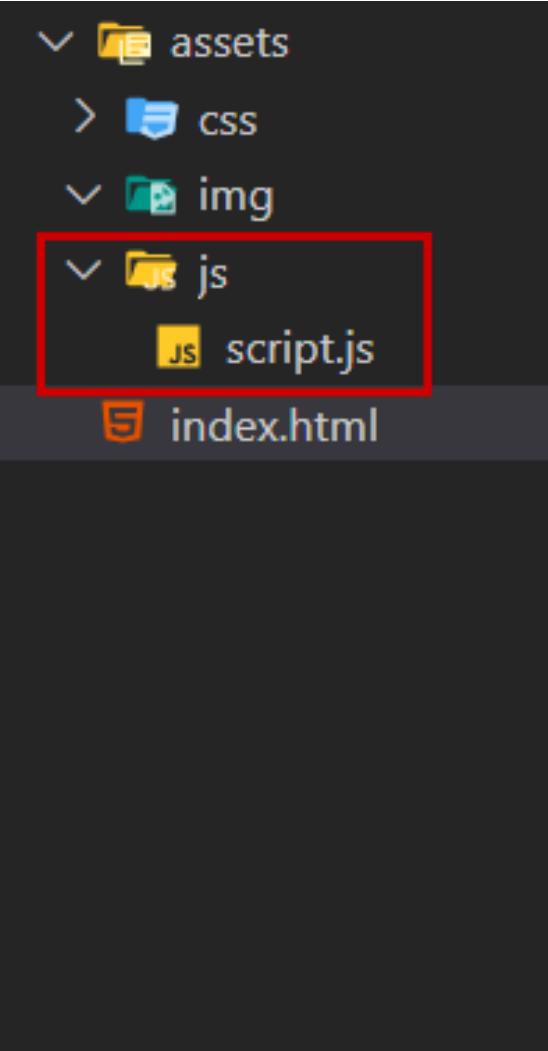
Puede escribirse dentro de la misma página HTML utilizando la etiqueta `<script>` en la sección `<head>` o al final del `<body>`. Aquí se separa del HTML, pero sigue dentro del mismo archivo.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript Interno</title>
</head>
<body>
    <h1>Ejemplo de JavaScript Interno</h1>
    <button onclick="mostrarMensaje()">Haz clic aquí</button>

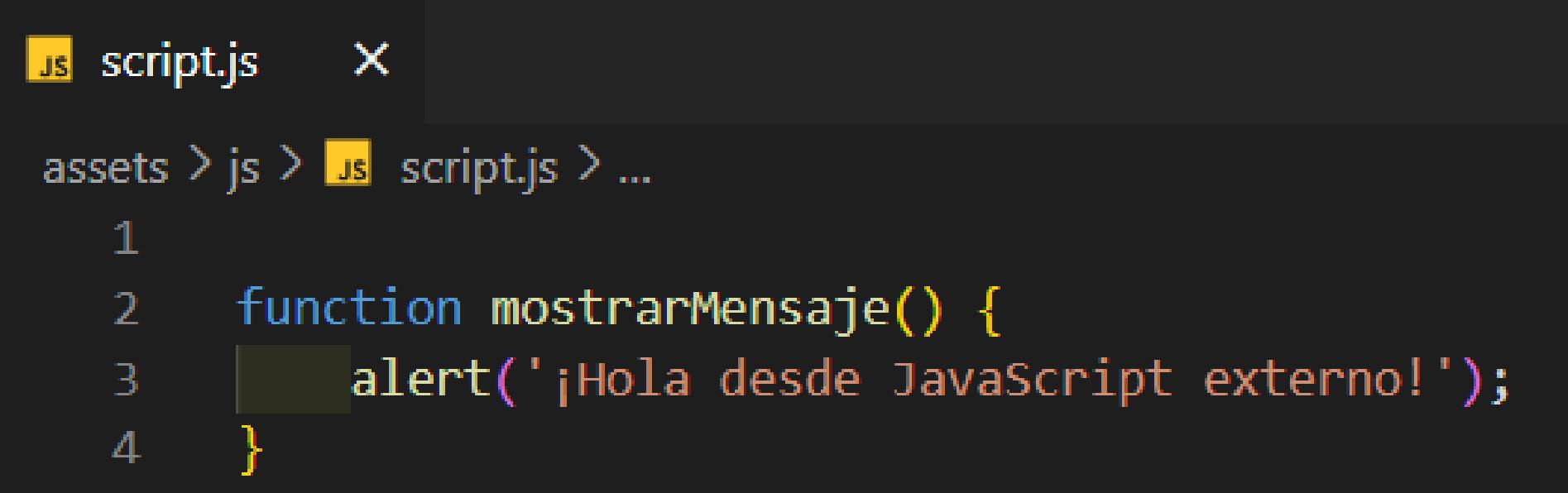
    <script>
        function mostrarMensaje() {
            alert('¡Hola desde JavaScript interno!');
        }
    </script>
</body>
</html>
```

Ejemplo: JavaScript externo

Código escrito en un archivo separado .js y enlazado con <script src="archivo.js"></script>.



```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>JavaScript Externo</title>
7
8  </head>
9  <body>
10     <h1>Ejemplo de JavaScript Externo</h1>
11     <button onclick="mostrarMensaje()">Haz clic aquí</button>
12
13
14     <script src=".//assets/js/script.js"></script>
15
16  </body>
17  </html>
```



```
1
2  function mostrarMensaje() {
3      alert('¡Hola desde JavaScript externo!');
4  }
```



Variables en JavaScript

Las variables son la forma en que JavaScript almacena datos.

Se pueden declarar con las palabras clave: var, let o const.

Let y Const



Let

se introdujo en ES6 (ECMAScript 2015)

- ◆ **Características**

- Permite declarar variables cuyo valor puede cambiar.
- Tiene alcance de bloque (limitado al bloque donde se define).

Const

Se introdujo en ES6 (ECMAScript 2015).

- ◆ **Características**

- Se utiliza para declarar variables cuyo valor no debe cambiar.
- Tiene alcance de bloque.
- No permite re-asignación.

Var y Hoistng



◆ Var

Fue la forma tradicional de declarar variables antes de ES6.

Características

- ✓ Tiene alcance de función (no de bloque).
- ✓ Sufre de hoisting (se "mueve" al principio de su contexto).

◆ Hoisting (Levantamiento)

Comportamiento en JavaScript donde las declaraciones de variables y funciones se "levantan" al principio de su contexto de ejecución.

Características

- ✓ Permite usar una variable o llamar a una función antes de declararla.
- ✓ Puede llevar a resultados inesperados.

Ejemplo: Variables de JavaScript

◆ Ejemplo let

JavaScript

```
let nombre = "Ana";
nombre = "Carlos"; // Se puede reasignar
```

◆ Ejemplo var

JavaScript

```
var fruta = "Manzana";
if (true) {
|   var fruta = "Pera";
}
console.log(fruta); // "Pera" (No tiene alcance de bloque)
```

◆ Ejemplo const

JavaScript

```
const PI = 3.1416;
PI = 3.15; // ✗ Error: No se puede reasignar
```

Ejemplo: Hoisting (levantamiento)

- ◆ Ejemplo var

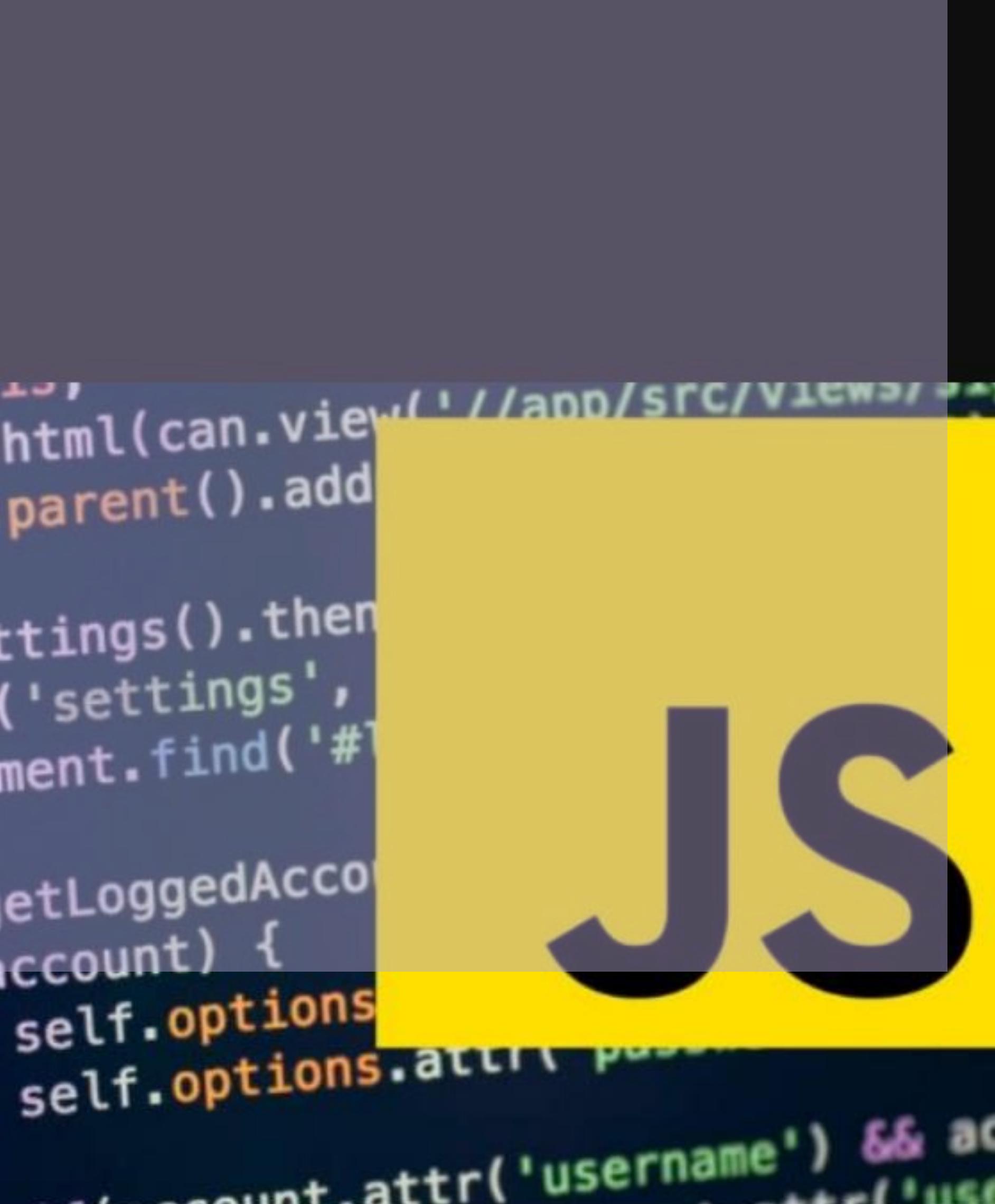
JavaScript

```
console.log(saludo); // undefined (pero no error)
var saludo = "Hola";
```

- ◆ Ejemplo let

JavaScript

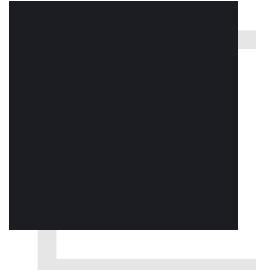
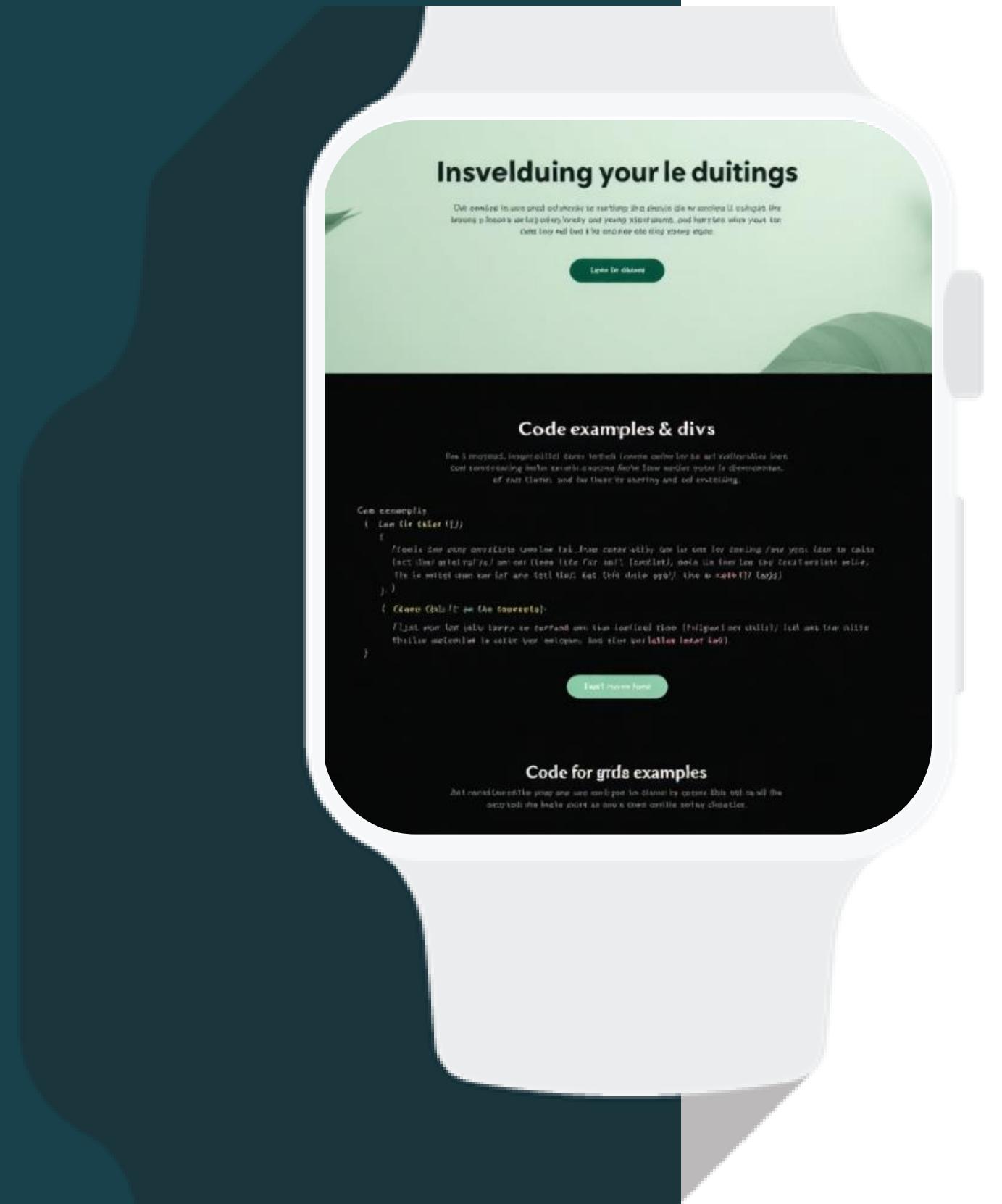
```
console.log(saludo); // ✗ Error: No está definida aún
let saludo = "Hola";
```



Expresiones Aritméticas

JavaScript soporta operadores aritméticos básicos para realizar cálculos y operaciones matemáticas como suma, resta, multiplicación, división y módulo.

Expresiones Aritméticas en JavaScript



Definición:

JavaScript soporta operadores aritméticos básicos para realizar cálculos y operaciones matemáticas.

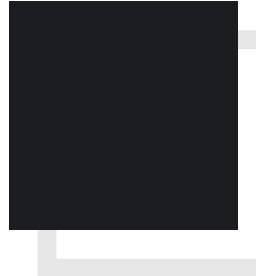
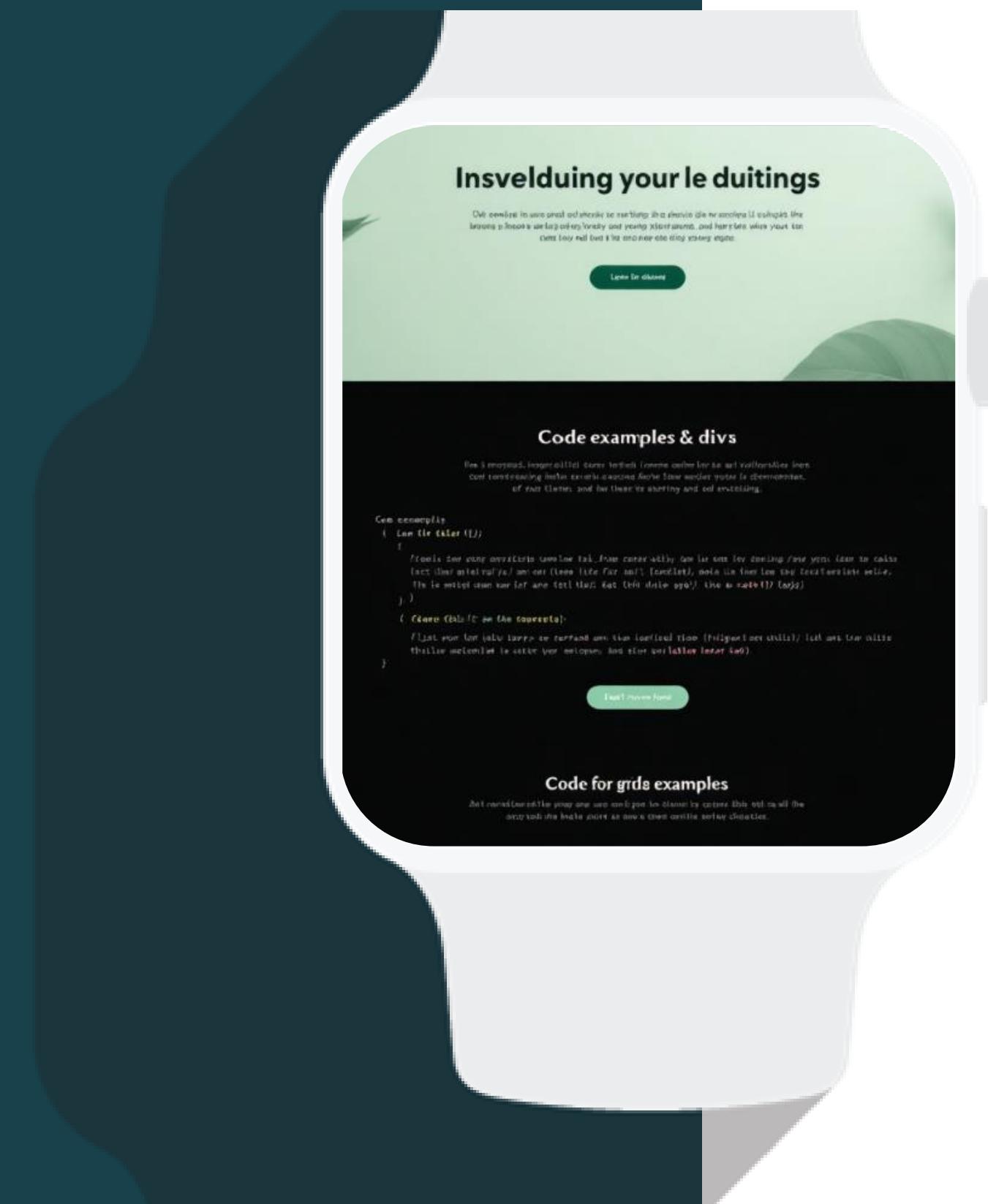
◆ Operadores comunes:

Suma (+), Resta (-), Multiplicación (*), División (/), Módulo (%).

◆ Ejemplo:

- ✓ Se declaran variables para cada operación aritmética.
- ✓ Los resultados se muestran mediante alert(), que crea una ventana emergente.

Uso de alert() para mostrar resultados



- ◆ **Características:**
 - ✓ alert() es una forma sencilla de mostrar resultados sin manipular el DOM.
 - ✓ Ideal para demostraciones rápidas o pruebas.
- Los resultados se muestran mediante alert(), que crea una ventana emergente.
-
- ◆ **Ejemplo:** Al abrir el archivo en un navegador, aparecerá una ventana emergente con los resultados de las operaciones aritméticas.

Ejercicio guiado: Expresiones aritméticas

Objetivo: aplicar operaciones aritméticas simples en JavaScript usando **valores ingresados por el usuario.**

Instrucciones paso a paso:

- 1** Crea un nuevo archivo operaciones.js.
- 2** Usa prompt() para pedir al usuario dos números.
- 3** Convierte los valores a tipo numérico usando Number().
- 4** Suma ambos valores y guarda el resultado en una variable.
- 5** Muestra el resultado en pantalla usando alert().



TIPS: todo lo que se ingresa por prompt() llega como texto, debes convertirlo antes de operar.

Ejercicio guiado: Expresiones aritméticas

```
// Pedir datos al usuario
let num1 = Number(prompt("Ingresa el primer número:"));
let num2 = Number(prompt("Ingresa el segundo número:"));

// Calcular la suma
let resultado = num1 + num2;

// Mostrar resultado
alert("La suma de ambos números es: " + resultado);
```

127.0.0.1:5500 dice

La suma de ambos números es: 7

Aceptar



IT Academy

by KIBERNUM