Material descargable

Programación Web Desde Cero

egg

¿Qué es JavaScript?

JavaScript es un lenguaje de programación utilizado principalmente para hacer que las páginas web sean interactivas. Permite agregar funcionalidades a un sitio web, como botones que responden cuando se hace clic en ellos, formularios que validan la información ingresada por los usuarios y elementos que se actualizan automáticamente sin necesidad de recargar la página.

Mientras que las páginas web tradicionales sirven principalmente como fuentes de información y tienen interactividad limitada (suelen llamarse páginas estáticas), las aplicaciones web modernas ofrecen una experiencia de usuario rica y dinámica, similar a los programas de escritorio, aprovechando las capacidades avanzadas de JavaScript y el ecosistema de herramientas y frameworks disponibles hoy en día.

¿ Qué son las variables?

Las variables son como “contenedores” para almacenar datos de diferentes tipos, como números, textos, listas y elementos más complejos. Las variables te permiten referenciar y manipular estos datos a lo largo de tu código.

Para definir una variable en JavaScript, puedes usar las palabras clave **let** o **const**, seguidas del nombre de la variable y, opcionalmente, inicializarla con un valor.

```
let nombre
let edad = 30
```

La variable **nombre** no se inicia con un valor, mientras que la variable **edad** se **inicializa** con el valor 30. Luego, para asignar un valor a la variable nombre es

necesario llamar o “invocar” a la variable con el nombre correspondiente y definir su valor:

```
nombre = 'Eggsy'
```

De la misma forma, se puede modificar el valor de una variable ya inicializada:

```
edad = 30
```

Utiliza **const** para todas las variables que no necesitan cambiar su valor después de ser inicializadas, lo cual ayuda a prevenir modificaciones accidentales:

```
const documento = 99123456
```

¿Cómo se nombran las variables?

Las convenciones de nomenclatura son reglas o pautas que se siguen al nombrar variables en un lenguaje de programación para mejorar la legibilidad del código y facilitar su comprensión. La convención de nomenclatura más común es **camelCase**.

Para aplicar esta convención cada palabra en el nombre de la variable comienza con mayúscula, excepto la primera. Por ejemplo:

```
let datosDelUsuario
const crearProducto
```

Usa nombres descriptivos para los nombres de las variables para que tu código sea fácil de leer y mantener.

¿Cómo se define una variable numérica?

Para declarar una variable numérica en JavaScript, puedes usar las palabras reservadas let/const, seguido del nombre de la variable y luego asignarle un valor numérico (entero o decimal):

```
let hermanos = 4
```

```
const pi = 3.14159
```

Cada una de estas declaraciones crea una variable que almacena un número, que luego puede ser utilizada para operaciones aritméticas.

¿Cómo se define una variable con texto?

Para declarar una variable con texto o cadena de texto (comúnmente llamado **string**), puedes usar las palabras reservadas let/const, seguido del nombre de la variable y luego asignarle la letra, la palabra o frase entre comillas (simple o doble):

```
let vocal = 'a'
const frase = "hola mundo"
```

Cada una de estas declaraciones crea una variable que almacena una cadena de texto, que luego puede ser utilizada para operaciones como la impresión en consola, manipulación de texto, entre otras.

¿Qué es y cómo se define una variable booleana?

Una variable booleana es un tipo de dato que solo puede tener dos posibles valores: true (verdadero) o false (falso). Este tipo de variable es fundamental en la programación, ya que se usa para tomar decisiones a través de estructuras de control como condicionales y bucles, permitiendo ejecutar diferentes partes del código según si una condición es verdadera o falsa.

Para definir una variable booleana en JavaScript, simplemente asignas true o false a una variable utilizando let/const:

```
let esVerdadero = true
const esFalso = false
```

¿Cómo se clasifican los tipos de datos en JS?

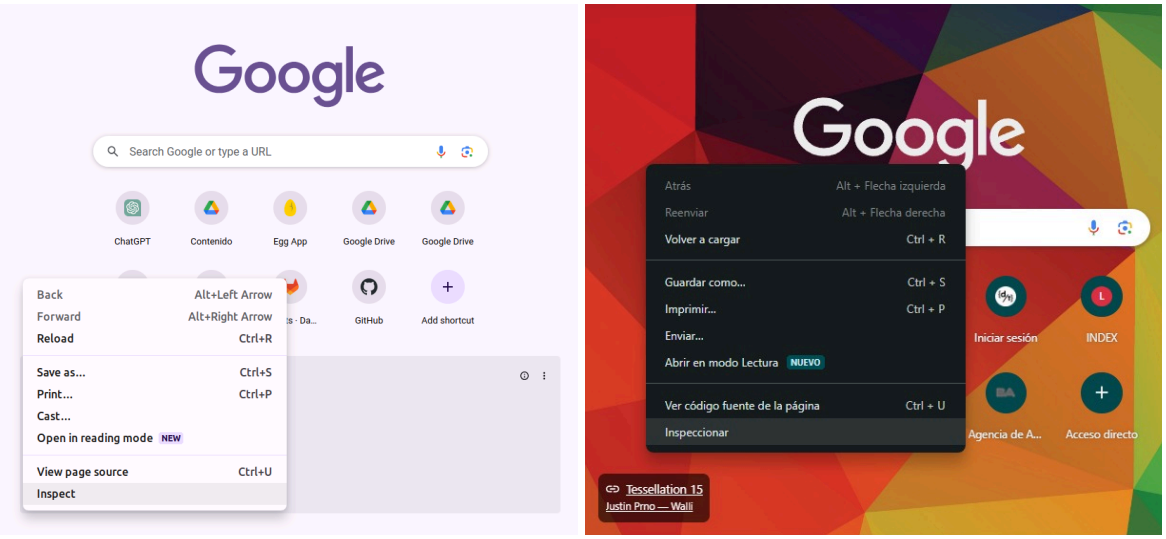
Tipo Primitivo	
Dato	**Descripción**
Numérico	Números, ya sea enteros o reales.
String	Cadenas de texto.
Boolean	Valores lógicos como true o false.
null	Cuando un dato no existe.
undefined	Cuando no se le asigna un valor a la variable.

Tipo Objeto	
Tipo De Objeto	**Descripción**
Predefinido de JavaScript	Date: fechas Error: datos de erros RegExp: expresiones regulares
Definidos por el programador	Funciones Clases
Arrays	Serie de elementos o formación tipo vector o matriz. Es considerado un objeto especial.

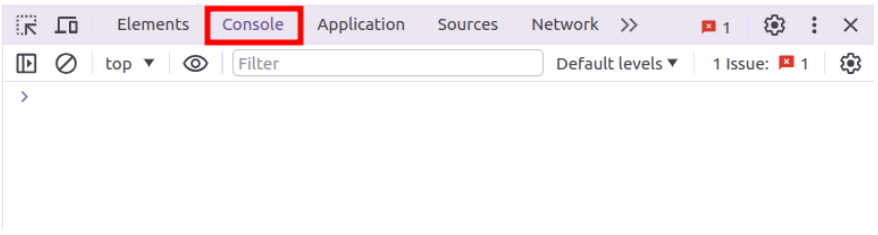
¿Cómo pruebo JS en el navegador?

Probar JavaScript directamente en el navegador es un proceso sencillo y eficaz para desarrolladores de todos los niveles. Casi todos los navegadores modernos vienen con herramientas de desarrollo integradas, incluyendo una consola JavaScript donde puedes escribir y ejecutar código JavaScript en tiempo real.

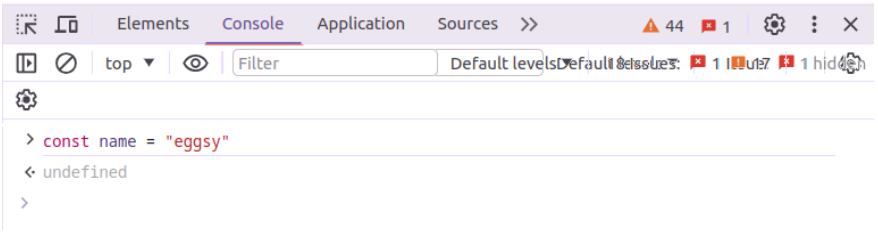
En tu navegador de preferencia, haz clic derecho en la página y selecciona la opción "Inspect", "Inspeccionar" o "Inspeccionar elemento".



Se abrirá una sección en la cual debes dirigirte a la pestaña "Consola":



Luego estarás en condiciones de probar el código de JS:



¿Qué es el objeto window de JS?

El objeto **window** representa la ventana del navegador. Piensa en él como una caja que contiene la página web que estás viendo. Está disponible automáticamente en el entorno de JavaScript, y puedes usarlo para controlar aspectos de la ventana del navegador, como el tamaño, la navegación (ir hacia adelante, hacia atrás) y mucho más.

Es como el control remoto de la ventana del navegador que te permite interactuar con ella. Los métodos **prompt**, **alert**, y **console** son herramientas que

ese control remoto te ofrece para interactuar con el usuario o para ayudarte durante el desarrollo de tus aplicaciones web:

- **prompt**: Muestra un cuadro de diálogo con un mensaje que solicita al usuario que ingrese algún texto. A modo de ejemplo, para guardar el nombre de un usuario en una variable, podrías usar prompt para preguntarlo:

```
let nombre = prompt("¿Cómo te llamas?")
```

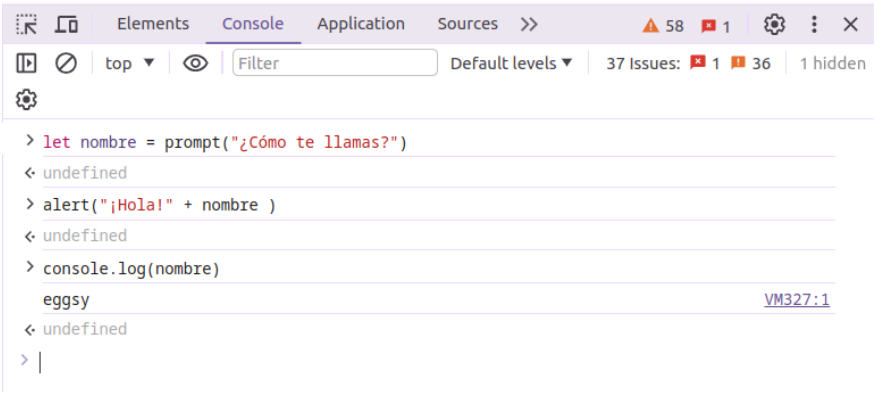
- **alert**: Muestra un cuadro de diálogo con un mensaje, pero solo tiene un botón para cerrarlo. Es útil para informar algo rápidamente al usuario. A modo de ejemplo, si quieres decirle "Hola" al usuario, podrías hacerlo así:

```
alert("¡Hola! " + nombre )
```

- **console**: Te permite imprimir mensajes en la consola del navegador, lo cual es útil para depurar tu código. A modo de ejemplo, para mostrar el nombre en la consola:

```
console.log(nombre)
```

Te animamos a probar estas líneas en el navegador para ver cómo funcionan.



¿Cómo se conecta JS con la página HTML?

Para agregar código JavaScript a una página web y hacer que tu sitio sea interactivo, tienes que agregar la etiqueta **<script>** sobre el final del archivo HTM:

```
<!-- archivo index.html -->

<html>
    <head>
        <title>Mi Página Web</title>
    </head>
    <body>
        <!-- Contenido de la página -->
        <script>
            let nombre = 'Eggsy'
            console.log(nombre)
        </script>
    </body>
</html>
```

Dentro de estas etiquetas debes desarrollar toda la lógica que tu sitio necesite.

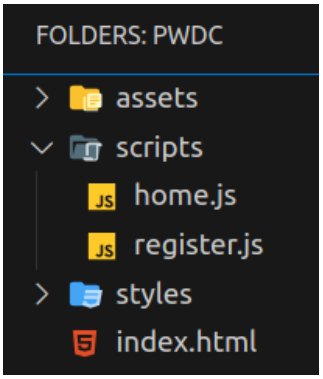
Se recomienda separar la lógica (JavaScript) de la maqueta (HTML) creando un archivo de extensión **.js** y conectando con la siguiente etiqueta:

```
<!-- archivo index.html -->

<html>
    <head>
        <title>Página Web Desde Cero</title>
    </head>
    <body>
        <!-- Contenido de la página -->
        <script src="home.js"></script>
    </body>
</html>
```

En este ejemplo, el archivo HTML se conecta con un archivo de JS llamado **home.js**. Este archivo debe contener todas las líneas de código necesarias para la funcionalidad deseada:

```
<!-- archivo script.js -->

let nombre = 'Eggsy'
console.log(nombre)
```

A medida que la página se vuelve más grande y se agrega más funcionalidades, es necesario crear más archivos de javascript. Es importante guardar todos estos archivos de forma ordenada en una carpeta llamada scripts.



Al incorporar esta carpeta a la estructura de archivos y carpetas de la página web, se debe enrutar correctamente hacia el archivo script que corresponde. A modo de ejemplo:

```
<!-- archivo index.html -->

<script src="./scripts/home.js"></script>
```