



UNIDAD TEMÁTICA V

JAVASCRIPT: AGREGANDO COMPORTAMIENTO A LA WEB.



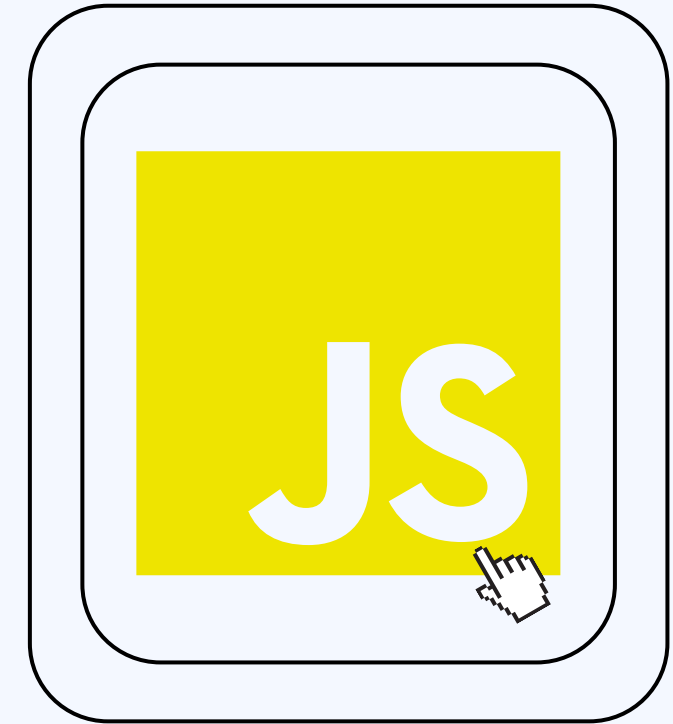
ING. RAIKJARS AFRICANO



¿QUÉ ES JAVASCRIPT?

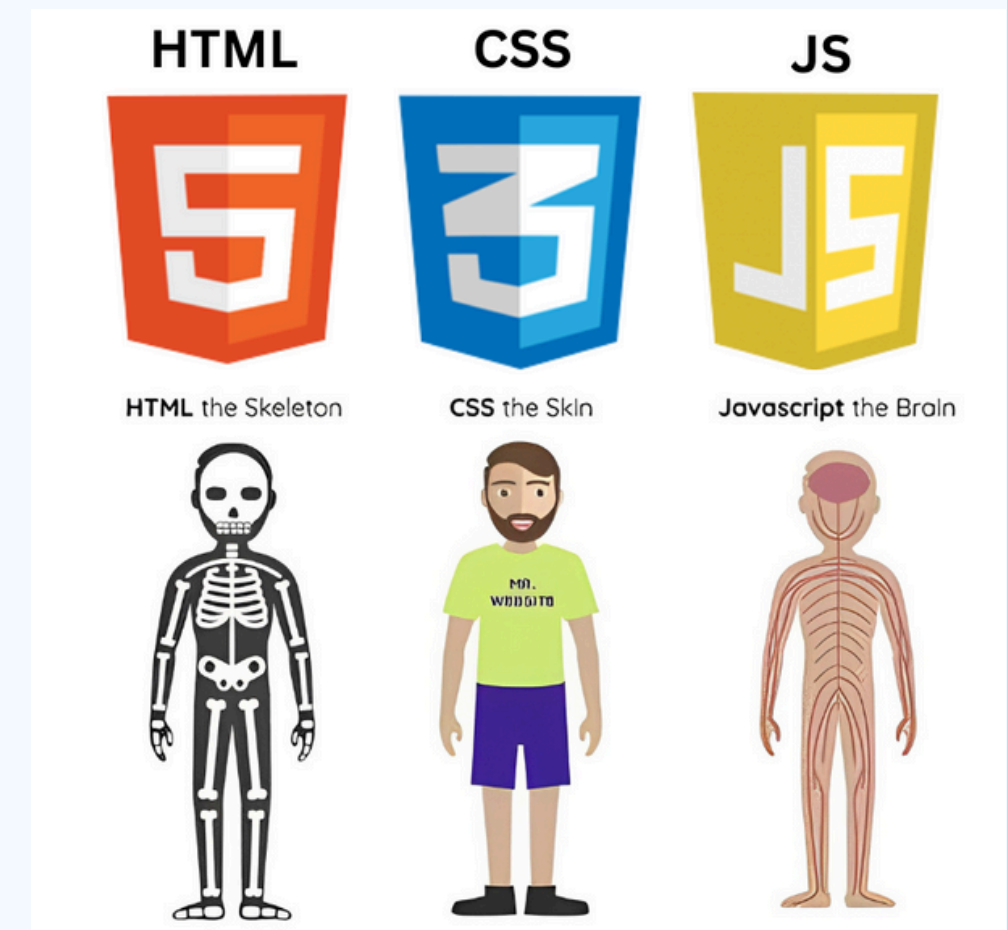
JavaScript (JS) es un lenguaje de programación de alto nivel que se utiliza principalmente para la creación de contenido dinámico y la interacción en páginas web. Permite agregar funcionalidad interactiva, como menús desplegables, actualizaciones de contenido sin recargar la página, animaciones y efectos visuales.

Este lenguaje **NO** tiene relación con Java, El nombre fue una estrategia de marketing y es uno de los lenguajes más populares Según GitHub y Stack Overflow.



Características de Javascript

- **Lenguaje interpretado:** No requiere compilación, se interpreta en tiempo real.
- **Tipado débil:** El tipo de una variable no necesita ser declarado explícitamente, lo que facilita la escritura de código.
- **Orientado a eventos:** Permite asociar acciones a eventos específicos de la página web, como clics, cambios de entrada o eventos de teclado.
- **Asincronía:** Permite realizar tareas de forma asíncrona, como esperar datos del servidor sin bloquear otras operaciones, mejorando la eficiencia de las aplicaciones.
- **Contenido dinámico:** JavaScript permite crear contenido que se actualiza dinámicamente en la página web. .
- **Compatibilidad:** Funciona en todos los navegadores modernos y también en servidores (Node.js), ejecutándose desde el navegador del usuario.





MÉTODOS PARA APLICAR JAVASCRIPT

Método	Descripción	Ventajas	Desventajas	Caso de Uso
Inline (en HTML)	Usar el atributo onclick, onload, etc., directamente en elementos HTML.	<ul style="list-style-type: none">- Rápido para pruebas simples.- No requiere archivos externos.	<ul style="list-style-type: none">- Difícil de mantener.- Mezcla HTML/JS.	Obsoleto, no es recomendado de usar.
Embebido <script>	Insertar código JS dentro de <script> en el <head> o <body> del HTML.	<ul style="list-style-type: none">- Fácil de implementar.- Útil para scripts pequeños.	<ul style="list-style-type: none">- Puede ralentizar la carga si es muy grande.- No es reutilizable (mala práctica).	Scripts pequeños o pruebas rápidas.
Archivo externo (Recomendado)	Vincular un archivo .js con <script src="archivo.js"></script>.	<ul style="list-style-type: none">- Reutilizable.- Mejor organización.- Caché del navegador.	Requiere archivo adicional	Proyectos grandes o código reutilizable.



Estilos en Linea

```
<button onclick="alert('¡Hola!')">Click aquí</button>
```



Embebido

```
<script>console.log("Código JS embebido");</script>
```



CSS Externo

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo JS Externo</title>
  </head>
  <body>
    <!-- Contenido de mi pagina web -->
    <script src="script.js"></script>
  </body>
</html>
```


SINTAXIS BASICA EN JAVASCRIPT



SENTENCIAS

Las sentencias en **JavaScript** son bloques de código que, cuando se ejecutan, realizan una acción o una serie de acciones

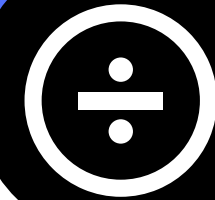
- terminan con un punto y coma (;), aunque no es obligatorio.
- Es case-sensitive (miVariable ≠ Mivariable) y se ejecuta secuencialmente (de arriba a abajo).



DECLARACIÓN DE VARIABLES

Se declaran utilizando las palabras clave **var**, **let** o **const**. Los nombres de las variables pueden contener letras, dígitos, símbolos, etc, pero deben comenzar con una letra o un guion bajo.

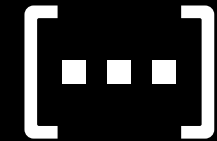
- **var (obsoleto)**: Alcance global o de función.
- **let**: Alcance de bloque, valor reasignable.
- **const**: Alcance de bloque, valor no reasignable.



OPERADORES

JavaScript utiliza diversos operadores para realizar operaciones matemáticas, comparaciones, asignaciones, etc.

- **Aritméticos**: +, -, *, /, %.
- **Comparación**: ==, === (igualdad estricta), >, <, !=.
- **Lógicos**: && (AND), || (OR), ! (NOT).



TIPOS DE DATOS

JavaScript soporta diversos tipos de datos, incluyendo números, cadenas, booleanos y objetos.

Primitivos:

- number (ej: 42, 3.14).
- string (ej: "Hola").
- boolean (true/false).
- null (valor nulo)
- undefined (no definido).

Objetos:

- object (ej: { nombre: "Ana" }), array (ej: [1, 2, 3]).



VARIABLES Y TIPOS DE DATOS

TIPOS DE DATOS PRIMITIVOS



Los datos primitivos (también llamados tipos de datos primitivos) son los valores básicos que no son objetos y no tienen métodos ni propiedades. Son los bloques de construcción fundamentales para representar datos en **JavaScript**.

Cuándo usar datos primitivos:

- **Almacenar valores simples:** Para almacenar números, texto, valores booleanos (true o false), o valores especiales como null o undefined.
- **Cuando se necesita un rendimiento óptimo:** Los datos primitivos suelen ser más eficientes en términos de espacio de memoria y tiempo de procesamiento que los objetos.
- **Comparaciones de valores:** para comparar valores, se utilizan los datos primitivos. Por ejemplo, para verificar si dos números son iguales.
- **Datos que no cambian:** Si los datos no cambian de valor después de la creación, es mejor usar primitivos.

TIPOS DE DATOS NO PRIMITIVOS



son estructuras de datos más complejas que pueden contener múltiples valores y que se almacenan en memoria por referencia, no por valor. Por lo que su tamaño puede cambiar a medida que se agregan o eliminan elementos.

La desventaja mas notable es que no se pueden copiar simplemente con el operador de asignación (=).

DECLARACIÓN DE VARIABLES (VAR, LET, CONST)



- **var (obsoleto, no recomendado):**

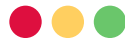
```
var edad = 30;
```

- **let (recomendado para valores cambiantes):**

```
let contador = 0;
```

- **const (para constantes):**

```
const PI = 3.1416;
```



TIPOS DE DATOS PRIMITIVOS

Tipo	Descripción	Ejemplo	Características Clave
number	Números enteros o decimales.	42, 3.14, -10	Soporta operaciones matemáticas.
string	Secuencia de caracteres (texto).	"Hola", 'Mundo'	Usa comillas simples, dobles o backticks (`).
boolean	Valor lógico (true o false).	true, false	Usado en condicionales y operaciones lógicas.
null	Valor nulo (asignado intencionalmente).	null	Representa la ausencia de valor.
undefined	Valor no definido (variable sin valor).	let x; → x es undefined	Valor por defecto de variables no inicializadas.
symbol	Valor único e inmutable (ES6+).	Symbol("id")	Usado para claves únicas en objetos.
bigint	Números enteros muy grandes (ES11+).	12345678901234567890n	Se agrega n al final.



TIPOS DE DATOS NO PRIMITIVOS

Tipo	Descripción	Ejemplo	Características Clave
object	Estructura de pares { clave: valor }.	{ nombre: "Ana", edad: 30 }	Colección de propiedades.
array	Lista ordenada de elementos.	[1, "texto", true]	Índices numéricos (comienza en 0).
function	Bloque de código reutilizable.	function suma(a, b) { return a + b; }	Es un objeto invocable.
Date	Fecha y hora.	new Date()	Métodos para manejar fechas.
RegExp	Expresiones regulares (patrones).	/\d+/g	Para búsqueda y validación de texto.
Map / Set	Colecciones (ES6+).	new Map(), new Set([1, 2, 3])	Map (clave-valor), Set (valores únicos).
Promise	Operaciones asíncronas.	fetch(url).then(...)	Maneja resultados futuros.
Error	Información de errores.	new Error("Algo falló")	Captura y personaliza errores.



TIPOS DE DATOS NO PRIMITIVOS

Tipo	Descripción	Ejemplo	Características Clave
object	Estructura de pares { clave: valor }.	{ nombre: "Ana", edad: 30 }	Colección de propiedades.
array	Lista ordenada de elementos.	[1, "texto", true]	Índices numéricos (comienza en 0).
function	Bloque de código reutilizable.	function suma(a, b) { return a + b; }	Es un objeto invocable.
Date	Fecha y hora.	new Date()	Métodos para manejar fechas.
RegExp	Expresiones regulares (patrones).	/\d+/g	Para búsqueda y validación de texto.
Map / Set	Colecciones (ES6+).	new Map(), new Set([1, 2, 3])	Map (clave-valor), Set (valores únicos).
Promise	Operaciones asíncronas.	fetch(url).then(...)	Maneja resultados futuros.
Error	Información de errores.	new Error("Algo falló")	Captura y personaliza errores.



ESTRUCTURAS DE CONTROL

son mecanismos que permiten alterar el flujo de ejecución del código, determinando qué bloques de código se ejecutan y en qué orden, según ciertas condiciones o la necesidad de repetir un proceso. Se utilizan para tomar decisiones, repetir bloques de código y modificar el flujo de ejecución del programa.

Tipos de estructura de control

- **Condicionales (if, else, else if):** Permiten ejecutar diferentes bloques de código según si una condición es verdadera o falsa.
- **Bucles (for, while, do-while):** Permiten repetir un bloque de código varias veces mientras una condición se cumpla.
- **Switch:** Permite evaluar una expresión y ejecutar el código asociado a un valor específico de esa expresión.
- **Try-catch:** Permite manejar errores o excepciones que puedan ocurrir durante la ejecución del código.

Escenario	Estructura Recomendada
Tomar decisiones simples	if, else, else if
Múltiples casos posibles	switch
Repetición con contador	for
Repetición con condición	while / do...while
Iterar arrays/strings	for...of
Iterar objetos	for...in (o Object.keys())
Manejo de errores	try...catch



Web

HTML

CSS

Javascript

UX

UI

PWA

Responsive

¡MUCHAS GRACIAS!

¿Dudas?



UNLOCKED ✓



Correo
rafrican@ucab.edu.ve



Modulo 7
Mensajes directos y foros



Salon A5-42
Hasta las 10:00 AM