Introducción al Desarrollo Web

Ing. Marco Aedo López

EPIS UNSA – 2025 maedol@unsa.edu.pe marcoaedo@hotmail.com

Fundamentos del desarrollo Frontend

JavaScript

Tema 6

Objetivos

- Comprender los fundamentos del lenguaje JavaScript: variables, tipos de datos, operadores, estructuras de control, funciones y manejo del DOM
- Aplicar JavaScript para crear interactividad en páginas web, incluyendo validación de formularios, manipulación dinámica de elementos HTML y respuesta a eventos del usuario
- Desarrollar proyectos prácticos utilizando buenas prácticas de programación, integrando JavaScript con HTML y CSS para construir aplicaciones web funcionales y escalables

1. ¿Qué es JavaScript?

- Lenguaje de programación interpretado, no compilado y basado en prototipos
- Usado principalmente para dar interactividad a páginas web
- Compatible con todos los navegadores modernos
- No requiere compilación, se ejecuta en el navegador

1. ¿Qué es JavaScript?

- Tipado dinámico
- Débilmente tipado
- Multiparadigma
- Gran ecosistema

2. ¿Por qué aprender JavaScript?

1. Lenguaje universal del desarrollo web

- Es el único lenguaje que funciona de forma nativa en todos los navegadores
- Es indispensable para crear **interfaces interactivas** y mejorar la experiencia del usuario
- Se creó para que la web deje de ser estática y limitada
- Permite desarrollo frontend, backend, aplicaciones móviles y escritorio
- Alta demanda laboral

2. ¿Por qué aprender JavaScript?

2. Ecosistema enorme y en constante evolución

- Gran cantidad de librerías y frameworks (React, Angular, Vue, Svelte, etc.)
- Con Node.js, puedes usar JavaScript en el servidor
- Amplio soporte de la comunidad, tutoriales y documentación accesible
- Innovación continua gracias a actualizaciones de ECMAScript

2. ¿Por qué aprender JavaScript?

3. Independencia tecnológica

- Permite crear proyectos propios sin depender de herramientas o plataformas externas
- Control total sobre el comportamiento del navegador y las interacciones del usuario

3. Sintaxis básica

- JavaScript es sensible a mayúsculas y minúsculas (case-sensitive)
- Se recomienda usar punto y coma (;) al final de cada sentencia
- Es zero-based

4. Variables y Constantes

- No es obligatorio declarar variables pero es altamente recomendable
- **var**: declaración antigua, ámbito global o de función
- **let**: declaración moderna, ámbito de bloque
- **const**: declaración para valores constantes, no se puede reasignar. Ámbito de bloque
- Buenas prácticas: usar **let** y **const** en lugar de var

4. Variables y Constantes

```
let edad;            // Declaración sin valor   undefined
edad = 25;           // Asignación posterior
let nombre = 'Ana'; // Declaración e inicialización
let x = 10, y = 20; // Declaración de varias variables

const PI = 3.1416; // Obligatorio inicializar
const NOMBRE_UNIVERSIDAD = "UNSA";

var edad;            // Declaración sin valor
edad = 25;           // Asignación posterior
var nombre = "Ana"; // Declaración e inicialización
var x = 10, y = 20, ciudad="Arequipa";
```

5. Reglas para nombres de identificadores

- Se aplican a variables, constantes, funciones, clases, parámetros, etc.
- Pueden contener letras, números, guion bajo (_) o signo de dólar ($)
- No pueden comenzar con un número
- No se permiten espacios ni caracteres especiales (excepto _ y $)
- Usar **camelCase** para nombres compuestos: `miVariableEjemplo`

```
let nombre;
let _contador;
let $precio;
let edad2;
let camelCaseVariable;
```

```
let 2x;        // ❌ comienza con número
let mi-nombre; // ❌ guion medio no permitido
let class;     // ❌ palabra reservada
```

6. Convenciones para nombres de identificadores

- Variables y funciones: camelCase → `nombreUsuario`
- Constantes: MAYUS_CON_GUIONES_BAJOS → `PI_VALOR`
- Clases: PascalCase → `MiClase`

```
// Variables (camelCase)
let nombreCliente = "María";
let totalCompra = 1200;
let descuentoAplicado = true;

// Constantes (MAYÚS_CON_GUIONES_BAJOS)
// Screaming Snake Case
const TASA_IGV = 0.18;
const URL_SERVIDOR = "https://api.mitienda.com";
```

```
// Clase (PascalCase)
class Producto {

}
```

7. Tipos de Datos

- Primitivos: string, number, boolean, null, undefined, symbol, bigint
- Objetos: arrays, funciones, objetos literales, clases
- JavaScript es dinámicamente tipado: el tipo puede cambiar en tiempo de ejecución
- Conversión implícita y explícita de tipos
- `typeof` permite saber el tipo de dato

7. Tipos de datos primitivos

Tipo	Descripción	Ejemplo
string	Representa texto. Se define entre comillas simples ' ', dobles " " o backticks ` ` (template literals)	"Hola", 'JavaScript’ `Hola ${nombre}`
number	Números enteros o decimales. JavaScript no diferencia entre int y float	42, 3.14, -10
bigint	Enteros de tamaño arbitrario, para números más grandes que Number.MAX_SAFE_INTEGER. Se define con n al final	9007199254740991n 123456789012345678901n
boolean	Valores lógicos: verdadero o falso	true, false
undefined	Variable declarada pero sin valor asignado	let x; // undefined
null	Ausencia intencional de valor (lo pone el programador)	let persona = null;
symbol	Valores únicos e inmutables, usados como identificadores de propiedades de objetos	Symbol("id")

8. Principales Operadores

Tipo de operador	Operador	Descripción	Ejemplo	Resultado
Aritmético	+	Suma	5 + 3	8
	-	Resta	5 - 3	2
	*	Multiplicación	4 * 2	8
	/	División	10 / 2	5
	%	Módulo (resto)	10 % 3	1
	**	Exponente	2 ** 3	8
Asignación	= += -= *= /= %= **=	Asignación	let a = 5	5
Incremento / Decremento	++	Incrementa en 1	a++	a + 1
	--	Decrementa en 1	a--	a - 1
Igualdad	==	Igualdad (solo valor)	5 == '5'	true
	===	Igualdad estricta (valor y tipo)	5 === '5'	false
	!=	Diferente (solo valor)	5 != '6'	true
	!==	Diferente estricta (valor y tipo)	5 !== '5'	true

8. Principales Operadores

Tipo de operador	Operador	Descripción	Ejemplo	Resultado
Comparación	>	Mayor que	7 > 3	true
	<	Menor que	3 < 7	true
	>=	Mayor o igual que	7 >= 7	true
	<=	Menor o igual que	3 <= 5	true
Lógicos	&&	AND (y)	true && false	false
	||	OR (o)	true || false	true
	!	NOT (negación)	!true	false
Cadena	+	Concatenación	"Hola " + "Mundo"	"Hola Mundo"
Tipo	typeof	Devuelve tipo de dato	typeof 5	"number"

9. Expresiones

- Cualquier fragmento de código que **genera un valor**
- Ese valor puede ser un número, un string, un booleano, un objeto, una función, etc.

```
5 + 3              // produce 8
"Hola " + "Mundo" // produce "Hola Mundo"
```

9. Expresiones

- Simples: un literal tal como: `42` `"Juancito"` `true`
- Complejas: involucran operadores, llamadas a funciones u otras expresiones anidadas
- Pueden formar parte de sentencias

```
let resultado = 2 + 2;
```

10. Tipos de Expresiones

Tipo de expresión	Descripción	Ejemplo
Literal	Valor escrito directamente	42, "Hola", true
Aritmética	Operaciones matemáticas	5 + 3 (a * b) / 2
De asignación	Asigna un valor	x = 10 y += 5
Lógica	Devuelven un valor booleano	x > 10 a && b
De cadena	Concatenación o manipulación de textos	"Hola " + " Mundo"
De llamada a función	Invocan funciones y producen un valor	Math.max(5, 10)
Condicional (ternaria)	Evalúa condición y devuelve un valor	edad >= 18 ? "Sí" : "No"

11. Estructura de una Sentencia

- Se compone de palabras clave, identificadores, operadores y valores
- Ejemplo:

```
let suma = a + b + 5;
```

- Los espacios en blanco no afectan la ejecución, pero mejoran la legibilidad
- Las sentencias pueden estar en una sola línea o en varias

12. Bloques de Código

- Un bloque se define con llaves `{ ... }`
- Sirven para agrupar varias instrucciones
- Ejemplo en un if:

```
if (condición) { ... }
```

- El bloque define el alcance (scope) de las variables `let` y `const`

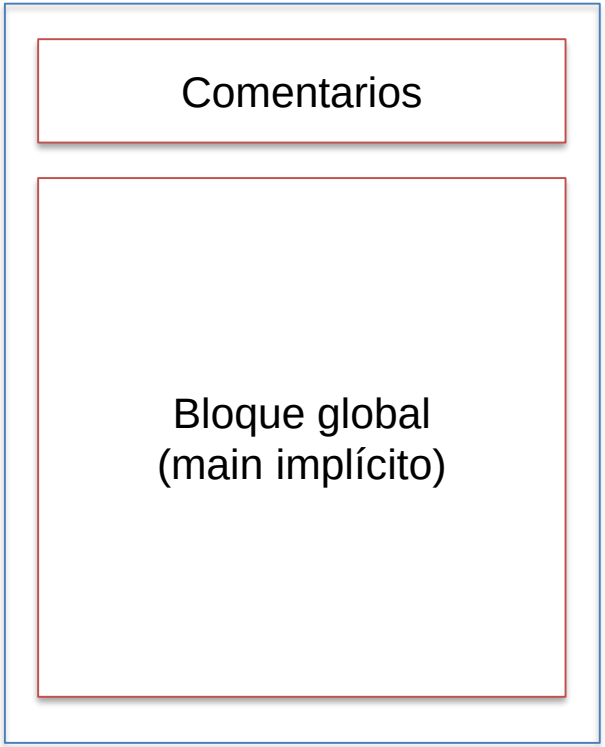
13. Comentarios en JavaScript

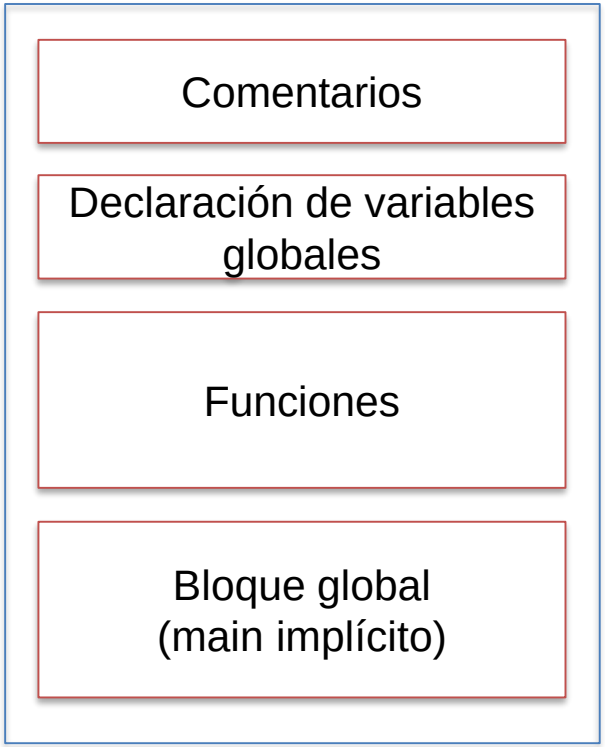
- Una línea: `// Esto es un comentario`
- Varias líneas:

```
/* Esto es un comentario de
varias líneas */
```

- Sirven para documentar y aclarar el código
- No afectan la ejecución del programa

Estructura de Programa





14. Tipo de dato string

- Es un **tipo de dato primitivo** que representa una **secuencia de caracteres** (letras, números, símbolos o espacios)
- Se utiliza para **manejar texto** en los programas
- Formas de definir un string
  - Comillas simples ' '
  - Comillas dobles " "
  - Template literals (backticks) ` `
    - Interpolar variables
    - Escribir texto en varias líneas

14. Tipo de dato string

```
let nombre = 'Juancito';
console.log("Hola "+nombre);


let nombre = "Juancito";
console.log('Hola '+nombre);

let lenguaje = "JavaScript";
let mensaje = `Estoy aprendiendo ${lenguaje}`;
console.log("Hola "+mensaje);
```

14. Tipo de dato string

```
let texto = "Ella dijo: \"Hola\"";
console.log(texto);
let salto = "Primera línea\nSegunda línea";
console.log(salto);



//Propiedad útil
let frase = "Hola";
console.log(frase.length);
```

```
//Métodos útiles
//Manipulación
console.log("hola".toUpperCase()); // "HOLA"
console.log("ADIOS".toLowerCase()); // "adios"

//Extracción
console.log("JavaScript".charAt(0));   // "J"
console.log("JavaScript".slice(0, 4)); // "Java"
console.log("JavaScript".substring(4, 10)); // "Script"
console.log("JavaScript".substring(0, 10)); // "JavaScript"

//Búsqueda
console.log("Programar".includes("ma")); // true
console.log("Programar".indexOf("a"));   // 5

//Reemplazo
console.log("Hola mundo".replace("mundo", "amigos")); // "Hola amigos"
```

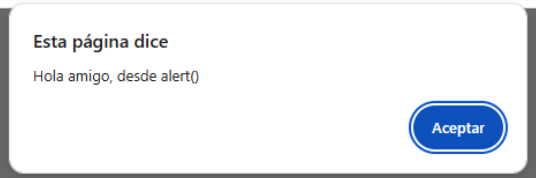
15. Salida de datos

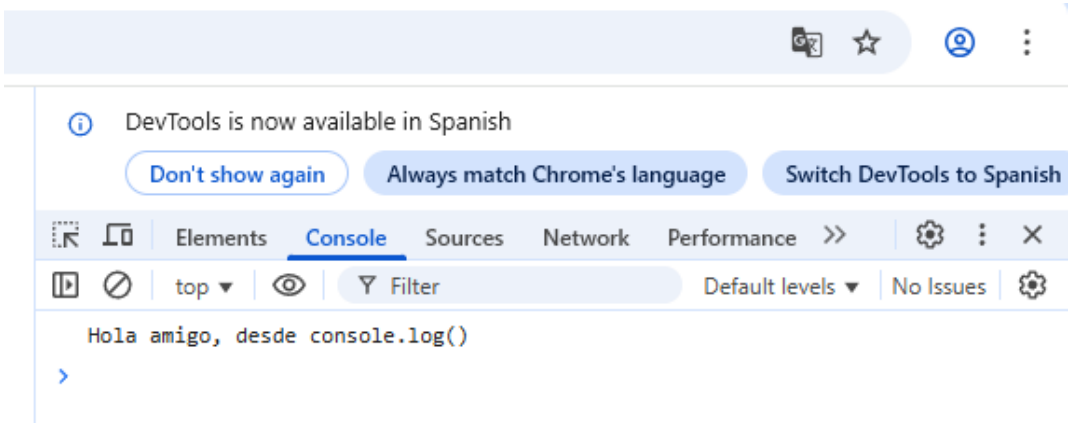
- alert()
  - Muestra un cuadro emergente en el navegador
  - Es la forma más directa de salida de datos
- console.log()
  - Envía datos a la consola del navegador. Herramientas de desarrollador (F12 o Inspeccionar) → pestaña Console
  - Ideal para depuración y para uso de la consola
- document.write()
  - Inserta texto directamente en el html
  - Ya no se usa profesionalmente

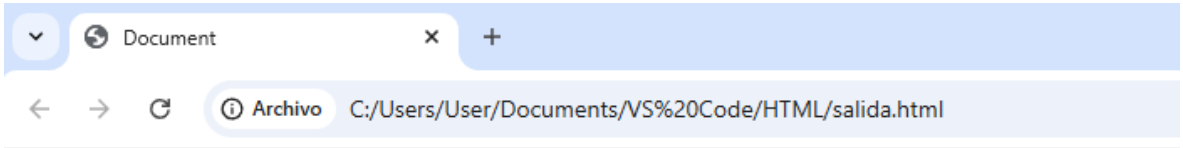
15. Salida de datos

- Dentro del `<script>` en el archivo html

```
<body>
    <h1>Prueba de salidas</h1>

    <script>
        // Salida con alert
        alert("Hola amigo, desde alert()");

        // Salida en la consola
        console.log("Hola amigo, desde console.log()");

        // Salida en la página con document.write
        document.write("Hola amigo, desde document.write()");

    </script>
</body>
```







Prueba de salidas

Hola amigo, desde document.write()

15. Salida de datos

- Dentro de un archivo .js y enlazado con el `<script>` en el archivo .html

```html
<body>
    <h1>Prueba de salidas</h1>

    <script src="salida.js"></script>
</body>
```

JS salida.js

```js
// Salida con alert
alert("Hola amigo, desde alert()");

// Salida en la consola
console.log("Hola amigo, desde console.log()");

// Salida en la página con document.write
document.write("Hola amigo, desde document.write()");
```

16. Entrada de datos

- Prompt (método clásico y sencillo)
  - Muestra un cuadro de diálogo en el navegador para que el usuario escriba algo
  - Ideal para aprendizaje básico, no recomendado en producción

```
let nombre = prompt("Ingresa tu nombre:");
```

- Formularios HTML
- Eventos y elementos interactivos
- Otros

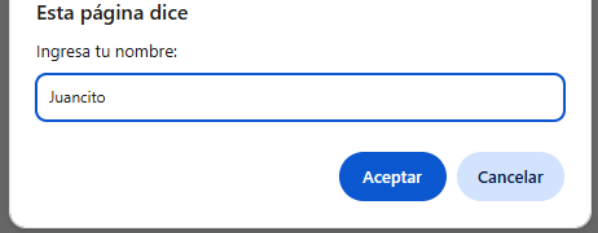
16. Entrada de datos

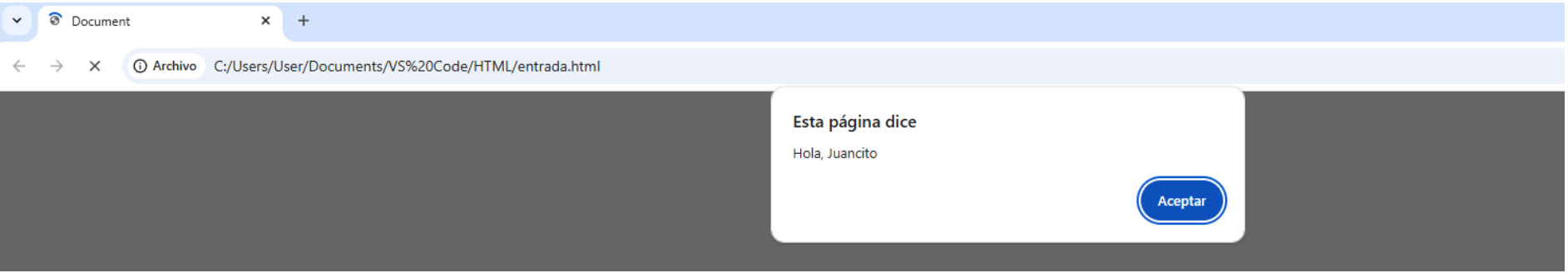
- Dentro de un archivo .js y enlazado con el `<script>` en el archivo .html

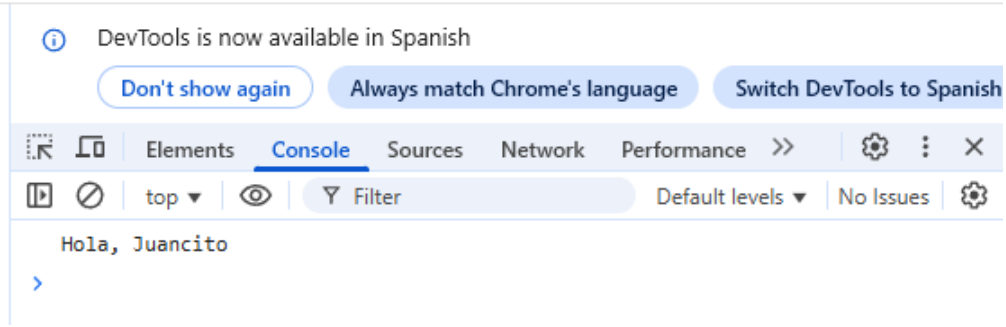
```html
<body>
    <h1>Prueba de entradas</h1>

    <script src="entrada.js"></script>
</body>
```

entrada.js > ...

```
1   let nombre = prompt("Ingresa tu nombre:");
2   alert("Hola, " + nombre);
3
4   console.log("Hola, " + nombre);
5
6   document.write("Hola "+nombre+" desde document.write()");
7
```


Esta página dice
Ingresa tu nombre:
Juancito
Aceptar
Cancelar


Document
Archivo
C:/Users/User/Documents/VS%20Code/HTML/entrada.html
Esta página dice
Hola, Juancito
Aceptar


DevTools is now available in Spanish
Don't show again
Always match Chrome's language
Switch DevTools to Spanish
Elements
Console
Sources
Network
Performance
top
Filter
Default levels
No Issues
Hola, Juancito


Document
Archivo
C:/Users/User/Documents/VS%20Code/HTML/entrada.html
Prueba de entradas
Hola Juancito desde document.write()

```
let x=parseInt(prompt("Ingresa primer numero:"));
let y=parseInt(prompt("Ingresa segundo numero:"));

console.log("la suma es: "+(x+y));
console.log("la resta es: "+(x-y));
console.log("la multiplicación es: "+x*y);
console.log("la división es: "+x/y);
console.log("el módulo es: "+x%y);
console.log("la potencia es: "+x**y);
```

```
const PI = 3.1416;
let radio = prompt("Ingresa el radio del círculo:");
radio = parseFloat(radio);
let area = PI * radio ** 2;
console.log("El área del círculo es: " + area);
```

```javascript
// Entrada de datos
let nombre = prompt("¿Cuál es tu nombre?");
let edad = parseFloat(prompt("¿Cuántos años tienes?"));
let estatura = parseFloat(prompt("¿Cuál es tu estatura en metros?"));

// Salida de datos
let mensaje = `Hola ${nombre}! tienes ${edad} años y mides ${estatura} metros.`;
console.log(mensaje);
```

```javascript
let nombre = prompt("Ingrese su nombre:");
let peso = parseFloat(prompt("Ingrese su peso en kilogramos:"));
let estatura = parseFloat(prompt("Ingrese su estatura en metros:"));

let imc = peso / (estatura ** 2);
console.log("==================================");
console.log("Nombre: " + nombre);
console.log("Peso: " + peso + " kg");
console.log("Estatura: " + estatura + " m");
console.log("IMC: " + imc);
console.log("==================================");
```

17. Estructuras de Control

- Condicionales: `if`, `else if`, `else`, `switch`
- Bucles: `for`, `while`, `do...while`, `for...of`, `for...in`
- `break` y `continue` para controlar el flujo

18. Estructuras de Control Condicionales

```
//if    Ejecuta un bloque si la condición es verdadera.

let edad = 20;

if (edad >= 18) {
  console.log("Eres mayor de edad");
}
```

```
//if...else
//Agrega un bloque alternativo si la condición es falsa.

let edad = 16;

if (edad >= 18) {
  console.log("Eres mayor de edad");
} else {
  console.log("Eres menor de edad");
}
```

```
//if...else if...else
//Permite evaluar múltiples condiciones.
//Forma más legible que ifs anidados

let nota = 18;

if (nota >= 18) {
  console.log("Excelente");
} else if (nota >= 11) {
  console.log("Aprobado");
} else {
  console.log("Desaprobado");
}
```

```
//Operador ternario ? :
//Forma corta de if...else.

let edad = 20;
let mensaje = (edad >= 18) ? "Mayor de edad" : "Menor de edad";
console.log(mensaje);
```

```
//switch
//Cuando se tienen muchos casos posibles.
//puede ser con cualquier tipo de datos.
//usa igualdad estricta (===)

let dia = 3;

switch (dia) {
  case 1: console.log("Lunes");
          break;
  case 2: console.log("Martes");
          break;
  case 3: console.log("Miércoles");
          break;
  default: console.log("Día no válido");
}
```

19. Estructuras de Control Repetitivas

```
//for
//Se usa cuando sabemos cuántas veces queremos repetir una acción

for (let i = 1; i <= 5; i++) {
  console.log("Iteración número: " + i);
}
```

```
//while
//Cuando no sabemos cuántas veces se repetirá el ciclo
//pero sí tenemos una condición lógica que debe cumplirse para continuar

let contador = 1;

while (contador <= 5) {
  console.log("Contador: " + contador);
  contador++;
}
```

```
//do...while
//Parecido a while, pero con una diferencia importante:
//ejecuta al menos una vez el bloque de código,
//incluso si la condición es falsa desde el inicio

let numero = 5;

do {
  console.log("Número: " + numero);
  numero++;
} while (numero < 5);
```

```
//break: rompe el ciclo y lo termina
//continue: salta a la siguiente iteración sin ejecutar lo que sigue en el bloque

for (let i = 1; i <= 5; i++) {
  if (i === 3) continue; // salta el 3
  if (i === 5) break;    // rompe el ciclo en 5
  console.log(i);
}
```

```
//for...of
//Recorre directamente los elementos de una cadena o arreglo

let palabra = "JavaScript";

for (let letra of palabra) {
  console.log(letra);
}
```

```
//for...in
//recorre las propiedades (claves)

let palabra = "Hola";

for (let indice in palabra) {
  console.log(indice + " → " + palabra[indice]);
}
```

19. Estructuras de Control Repetitivas

- En desarrollo web se usan mucho:
  - Para recorrer arreglos de datos (por ejemplo, productos en un carrito de compras)
  - Para generar dinámicamente HTML
  - Para iterar resultados de una API y mostrarlos en pantalla

Ejercicios

1. Crea un programa que lea tres notas y sus respectivos pesos (porcentaje de evaluación), y calcule el promedio ponderado final del estudiante.

   Muestra el resultado con dos decimales. Sugerencia: num.toFixed(2)

2. Escribe un programa que lea una temperatura en grados Celsius y muestre su equivalente en Fahrenheit y Kelvin.

   Considerar: F = C * 9/5 + 32 K = C + 273.15

3. Crea un programa que lea las coordenadas de dos puntos (x1, y1) y (x2, y2), y calcule la distancia entre ellos. Usa Math.sqrt() como si fuera Java

4. Escribe un programa que calcule el monto final de una inversión usando interés compuesto. El usuario ingresa el capital, la tasa de interés anual (%) y el número de años. Mostrar el monto final y el interés ganado.

   Considerar: Monto = Capital * (1 + tasa/100) ^ tiempo

5. Crea un programa que lea una velocidad en kilómetros por hora (km/h) y la convierta a: metros por segundo (m/s) y millas por hora (mph)

   Considerar: 1 milla = 1.60934 kilómetros

6. Crea un programa que lea tres números y muestre el número mayor
7. Crea un programa que lea por separado dos números y un operador (+, -, *, /, %, **).
   Realizar la operación correcta según el operador.
   Si el operador no es válido, muestra un mensaje de error.
8. Crea un programa que lea peso y estatura, calcule el IMC y lo clasifique:
   IMC < 18.5 → Bajo peso
   18.5 ≤ IMC < 25 → Normal
   25 ≤ IMC < 30 → Sobrepeso
   IMC ≥ 30 → Obesidad
   Muestra un mensaje indicando la categoría correspondiente.

$$\text{IMC} = \frac{\text{peso (kg)}}{\text{estatura (m)}^2}$$

9. Crea un programa que lea una nota numérica (0–100) y la convierta a una calificación con letra:
   90–100 → A
   80–89 → B
   70–79 → C
   60–69 → D
   0–59 → F
   Valida que la nota ingresada esté entre 0 y 100.
10. Crea un programa que lea las tres longitudes de un triángulo y determine su tipo:
    Equilátero → los tres lados iguales Isósceles → dos lados iguales
    Escaleno → todos los lados diferentes
    También valida si los lados forman un triángulo válido (suma de dos lados > tercer lado).

11. Programa que pida al usuario que ingrese N números.

    Calcula y muestra la suma total y su promedio redondeado a 2 decimales.

12. Programa que lea un número entero positivo n y muestre los primeros n términos de la serie de Fibonacci 0, 1, 1, 2, 3, 5, 8, …

13. Programa que pida al usuario que ingrese la edad de personas (no se sabe la cantidad).

    Calcula y muestra: Promedio de edades y cantidad de personas mayores de edad

14. Programa que pida un número n y que muestre todos los números primos desde 2 hasta n, y la cantidad de números primos que encontró

15. Genera un número aleatorio entre 1 y 100. El usuario tiene 6 intentos para adivinarlo.

    Cada vez que ingresa un número, el programa indica si es mayor o menor que el número secreto.

    Termina cuando adivina o se acaban los intentos. Deberá mostrar si ganó o perdió, si ganó mostrar en cuántos intentos lo hizo