JavaScript

Sintaxis y Fundamentos

🖍️ 2026-01 — ⏱️ 60 min.

Variables y Tipos de Datos

Declaración de Variables

```
// var - ya no se usa
var nombre = 'Juan';

// let - para variables que cambian
let edad = 25;
edad = 26; // ✅ Puede cambiar

// const - para constantes
const PI = 3.14159;
PI = 3.14; // ❌ Error: no se puede reasignar
```

Nombres de Variables

Reglas:

- Empezar con letra, `_`, o `$`
- Pueden contener letras, números, `_`, `$`
- No usar palabras reservadas (`if`, `let`, `function`, etc.)
- Case-sensitive (`nombre` ≠ `Nombre`)

Convención (camelCase):

```
let nombreCompleto = 'Juan Pérez';
let edadUsuario = 25;
let estaActivo = true;
```

Tipos de Datos Primitivos (1/2)

```
// String
let nombre = "María";
let apellido = 'González';
let mensaje = `Hola ${nombre}`; // Template literals

// Number
let entero = 42;
let decimal = 3.14;
let negativo = -10;
```

Tipos de Datos Primitivos (2/2)

```
// Boolean
let esVerdad = true;
let esFalso = false;

// Undefined
let sinValor;
let sinValor2 = undefined;

// Null
let vacio = null;
```

Operador typeof

```
console.log(typeof "Hola");      // "string"
console.log(typeof 42);          // "number"
console.log(typeof true);        // "boolean"
console.log(typeof undefined);   // "undefined"
console.log(typeof null);        // "object"
console.log(typeof {});          // "object"
console.log(typeof []);          // "object"
console.log(typeof function(){}); // "function"
```

Operadores

Operadores Aritméticos

```
let a = 10;
let b = 3;

console.log(a + b);  // 13
console.log(a - b);  // 7
console.log(a * b);  // 30
console.log(a / b);  // 3.333...
console.log(a % b);  // 1
console.log(a ** b); // 1000

// Incremento y decremento
let x = 5;
x++;  // x = 6
x--;  // x = 5
```

Operadores de Comparación

```
// Igualdad
console.log(5 == 5);   // true

// Desigualdad
console.log(5 != 5);   // false

// Mayor y menor
console.log(10 > 5);     // true
console.log(10 < 5);     // false
console.log(10 >= 10);   // true
console.log(10 <= 5);    // false
```

Operadores de Comparación

```
console.log(5 == "5"); // ?
```

Operadores de Comparación

```
console.log(5 == "5"); // true
console.log(5 !== "5"); // false
```

Operadores de Comparación

```
// Igualdad
console.log(5 == "5");   // true
console.log(5 === "5");  // false

// Desigualdad
console.log(5 != "5");   // false
console.log(5 !== "5");  // true

// Mayor y menor
console.log(10 > 5);     // true
console.log(10 < 5);     // false
console.log(10 >= 10);   // true
console.log(10 <= 5);    // false
```

Mejor práctica: Usar siempre `===` y `!==` (comparación de tipo y de valor).

Operadores Lógicos (1/2)

```
// AND (&&) - ambos deben ser verdaderos
console.log(true && true);   // true
console.log(true && false);  // false

// OR (||) - al menos uno debe ser verdadero
console.log(true || false);  // true
console.log(false || false); // false

// NOT (!) - invierte el valor
console.log(!true);          // false
console.log(!false);         // true
```

Operadores Lógicos (2/2)

```
// Ejemplo práctico
let edad = 20;
let tieneLicencia = true;

if (edad >= 18 && tieneLicencia) {
    console.log("Puede conducir");
}

// Otro ejemplo
let esFinDeSemana = true;
let tieneEnergia = false;

if (esFinDeSemana || tieneEnergia) {
    console.log("Vamos a programar!");
}
```

Condicionales

`if` / `else`

```
let edad = 18;

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

// if / else if / else
let nota = 85;

if (nota >= 90) {
    console.log("Excelente");
} else if (nota >= 70) {
    console.log("Aprobado");
} else {
    console.log("Reprobado");
}
```

Operador Ternario

```
// Sintaxis: condición ? valorSiTrue : valorSiFalse

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

// Anidado (con 2 ternarios ya es bastante)
let nota = 85;
let resultado = nota >= 90 ? "A" :
                nota >= 80 ? "B" :
                nota >= 70 ? "C" : "F";
```

`switch`

```
let dia = "lunes";

switch (dia) {
    case "lunes":
        console.log("Inicio de semana");
        break;
    case "viernes":
        console.log("Fin de semana laboral");
        break;
    case "sábado":
    case "domingo":
        console.log("Fin de semana");
        break;
    default:
        console.log("Día entre semana");
}
```

⚠️ No olvidar `break` o `return` en los casos.

Ciclos

Ciclo `for`

```
// for clásico
for (let i = 0; i < 5; i++) {
    console.log(i); // 0, 1, 2, 3, 4
}

// Iterar hacia atrás
for (let i = 5; i > 0; i--) {
    console.log(i); // 5, 4, 3, 2, 1
}

// Saltos de 2 en 2
for (let i = 0; i < 10; i += 2) {
    console.log(i); // 0, 2, 4, 6, 8
}
```

Ciclo `while` y `do-while`

```
// while - verifica condición antes de ejecutar
let i = 0;
while (i < 5) {
    console.log(i);
    i++;
}

// do-while - ejecuta al menos una vez
let j = 0;
do {
    console.log(j);
    j++;
} while (j < 5);
```

`break` y `continue`

```
// break - sale del ciclo
for (let i = 0; i < 10; i++) {
    if (i === 5) break;
    console.log(i); // 0, 1, 2, 3, 4
}

// continue - salta a la siguiente iteración
for (let i = 0; i < 5; i++) {
    if (i === 2) continue;
    console.log(i); // 0, 1, 3, 4 (salta el 2)
}

for (let i = 0; i < 10; i++) {
    if (i % 2 !== 0) continue;
    console.log(i); // 0, 2, 4, 6, 8
}
```

Funciones

Declaración de Funciones

```
// Función tradicional
function saludar(nombre) {
    return `Hola, ${nombre}!`;
}

console.log(saludar("María")); // "Hola, María!"

// Función sin parámetros
function obtenerFecha() {
    return new Date().toLocaleDateString();
}

// Función sin retorno (void)
function mostrarMensaje() {
    console.log("Esto es un mensaje");
}

// Tambien función sin retorno (void)
function mostrarMensaje() {
    return console.log("Esto es un mensaje");
}
```

Parámetros y Argumentos

```
// Múltiples parámetros
function sumar(a, b) {
    return a + b;
}

console.log(sumar(5, 3)); // 8

// Parámetros por defecto
function saludar(nombre = "Usuario") {
    return `Hola, ${nombre}!`;
}

console.log(saludar());        // "Hola, Usuario!"
console.log(saludar("Ana"));   // "Hola, Ana!"

// "Rest" parameters
function sumarTodos(...numeros) {
    return numeros.reduce((total, n) => total + n, 0);
}

console.log(sumarTodos(1, 2, 3, 4)); // 10
```

Arrow Functions (Funciones Flecha)

```
// Función tradicional
function suma(a, b) {
    return a + b;
}

// Arrow function equivalente
const suma = (a, b) => {
    return a + b;
};

// Arrow function con retorno implícito (una línea)
const suma = (a, b) => a + b;

// Con un solo parámetro (sin paréntesis)
const cuadrado = x => x * x;

// Sin parámetros
const obtenerPI = () => 3.14159;

console.log(suma(5, 3));      // 8
console.log(cuadrado(4));     // 16
console.log(obtenerPI());     // 3.14159
```

Arrays

Crear y Acceder a Arrays

```
// Crear array
let frutas = ["manzana", "banana", "naranja"];
let numeros = [1, 2, 3, 4, 5];
let mixto = [1, "texto", true, null];

// Acceder a elementos
console.log(frutas[0]);  // "manzana"
console.log(frutas[2]);  // "naranja"

// Modificar elementos
frutas[1] = "fresa";
console.log(frutas); // ["manzana", "fresa", "naranja"]

// Longitud del array
console.log(frutas.length); // 3

// Último elemento
console.log(frutas[frutas.length - 1]); // "naranja"
```

Métodos de Arrays Comunes (1/2)

```
let frutas = ["manzana", "banana"];

// Agregar al final
frutas.push("naranja");
// ["manzana", "banana", "naranja"]

// Eliminar del final
frutas.pop();
// ["manzana", "banana"]

// Agregar al inicio
frutas.unshift("fresa");
// ["fresa", "manzana", "banana"]

// Eliminar del inicio
frutas.shift();
// ["manzana", "banana"]
```

Métodos de Arrays Comunes (2/2)

```
let frutas = ["manzana", "banana", "naranja"];

// Encontrar índice
let indice = frutas.indexOf("banana"); // 1

// Verificar si existe
let existe = frutas.includes("manzana"); // true

// Extraer una porción
let algunas = frutas.slice(1, 3);
// ["banana", "naranja"]

// Unir en string
let texto = frutas.join(", ");
// "manzana, banana, naranja"
```

Iterar sobre Arrays

```
let numeros = [1, 2, 3, 4, 5];

// for clásico
for (let i = 0; i < numeros.length; i++) {
    console.log(numeros[i]);
}

// for...of (más moderno)
for (let numero of numeros) {
    console.log(numero);
}

// forEach
numeros.forEach(function(numero) {
    console.log(numero);
});

// forEach con arrow function
numeros.forEach(numero => console.log(numero));
```

Métodos Funcionales: map, filter, reduce

```
let numeros = [1, 2, 3, 4, 5];

// map - transforma cada elemento
let duplicados = numeros.map(n => n * 2);
// [2, 4, 6, 8, 10]

// filter - filtra elementos
let pares = numeros.filter(n => n % 2 === 0);
// [2, 4]

// reduce - reduce a un valor único
let suma = numeros.reduce((total, n) => total + n, 0);
// 15

// Combinando métodos
let sumaPares = numeros
    .filter(n => n % 2 === 0)
    .reduce((total, n) => total + n, 0);
// 6
```

Métodos Funcionales: find, some, every

```
let numeros = [1, 2, 3, 4, 5];

// find - encuentra el primer elemento que cumple
let encontrado = numeros.find(n => n > 3);
// 4

// findIndex - encuentra el índice del primer elemento
let indice = numeros.findIndex(n => n === 3);
// 2

// some - verifica si alguno cumple
let hayPares = numeros.some(n => n % 2 === 0);
// true

// every - verifica si todos cumplen
let todosPares = numeros.every(n => n % 2 === 0);
// false
```

Método flat

```
// flat - aplanar arrays anidados
const nested = [1, [2, 3], [4, [5]]];

// Un nivel
const flat1 = nested.flat();
// [1, 2, 3, 4, [5]]

// Dos niveles
const flat2 = nested.flat(2);
// [1, 2, 3, 4, 5]

// Todos los niveles
const deepNested = [1, [2, [3, [4, [5]]]]];
const flatAll = deepNested.flat(Infinity);
// [1, 2, 3, 4, 5]
```

Objetos

Crear y Acceder a Objetos

```
// Crear objeto
let persona = {
    nombre: "Juan",
    edad: 30,
    ciudad: "Madrid",
    activo: true
};

// Acceder con notación de punto
console.log(persona.nombre);  // "Juan"

// Acceder con notación de corchetes
console.log(persona["edad"]); // 30

// Modificar propiedades
persona.edad = 31;
persona["ciudad"] = "Barcelona";

// Agregar nuevas propiedades
persona.profesion = "Desarrollador";

// Eliminar propiedades
delete persona.activo;
```

Métodos en Objetos

```
let persona = {
    nombre: "María",
    edad: 25,
    // Método tradicional
    saludar: function() {
        return `Hola, soy ${this.nombre}`;
    },
    // Método abreviado (ES6)
    despedir() {
        return `Adiós, soy ${this.nombre}`;
    },
    // Arrow function (this no funciona igual)
    obtenerEdad: () => {
        return this.edad; // ⚠️ this no es el objeto
    }
};

console.log(persona.saludar());   // "Hola, soy María"
console.log(persona.despedir());  // "Adiós, soy María"
```

Iterar sobre Objetos (1/2)

```
let persona = {
    nombre: "Carlos",
    edad: 35,
    ciudad: "Lima"
};

// for...in - itera las keys
for (let clave in persona) {
    console.log(`${clave}: ${persona[clave]}`);
}
// nombre: Carlos
// edad: 35
// ciudad: Lima
```

Iterar sobre Objetos (2/2)

```
let persona = { nombre: "Carlos", edad: 35, ciudad: "Lima" };

// Object.keys() - obtiene las keys
let claves = Object.keys(persona);
console.log(claves);
// ["nombre", "edad", "ciudad"]

// Object.values() - obtiene los valores
let valores = Object.values(persona);
console.log(valores);
// ["Carlos", 35, "Lima"]

// Object.entries() - obtiene pares [key, value]
let entries = Object.entries(persona);
// [["nombre", "Carlos"], ["edad", 35], ["ciudad", "Lima"]]
```

Destructuración

```
// Destructuración de objetos
let persona = { nombre: "Ana", edad: 28, ciudad: "Bogotá" };

let { nombre, edad } = persona;
console.log(nombre); // "Ana"
console.log(edad);   // 28

// Con alias
let { nombre: n, edad: e } = persona;
console.log(n); // "Ana"

// Destructuración de arrays
let frutas = ["manzana", "banana", "naranja"];

let [primera, segunda] = frutas;
console.log(primera);  // "manzana"
console.log(segunda);  // "banana"

// Saltar elementos
let [, , tercera] = frutas;
console.log(tercera); // "naranja"
```

Strings

Métodos de Strings (1/2)

```
let texto = "Hola Mundo";

// Longitud
console.log(texto.length); // 10

// Mayúsculas y minúsculas
console.log(texto.toUpperCase()); // "HOLA MUNDO"
console.log(texto.toLowerCase()); // "hola mundo"

// Buscar
console.log(texto.includes("Mundo"));  // true
console.log(texto.indexOf("Mundo"));   // 5
console.log(texto.startsWith("Hola")); // true
console.log(texto.endsWith("Mundo"));  // true
```

Métodos de Strings (2/2)

```
let texto = "Hola Mundo";

// Extraer
console.log(texto.slice(0, 4));     // "Hola"
console.log(texto.substring(5, 10)); // "Mundo"

// Reemplazar
console.log(texto.replace("Mundo", "JavaScript"));
// "Hola JavaScript"

// Dividir
let palabras = texto.split(" ");
console.log(palabras); // ["Hola", "Mundo"]

// Limpiar espacios
let conEspacios = "  hola  ";
console.log(conEspacios.trim()); // "hola"
```

Template Literals

```
let nombre = "María";
let edad = 25;

// String tradicional
let mensaje1 = "Hola, " + nombre + ". Tienes " + edad + " años.";

// Template literal (backticks)
let mensaje2 = `Hola, ${nombre}. Tienes ${edad} años.`;

// Expresiones dentro de ${}
let mensaje3 = `El próximo año tendrás ${edad + 1} años.`;

// Multi-línea
let html = `
    <div>
        <h1>${nombre}</h1>
        <p>Edad: ${edad}</p>
    </div>
`;

console.log(mensaje2);
```

Conceptos Importantes

Truthy y Falsy

Valores *Falsy* (se evalúan como `false`):

```
false
0
"" (string vacío)
null
undefined
NaN // Not a Number
```

Valores *Truthy*: Todo lo demás

Null Coalescing Operator (??) - Parte 1

```
// ?? retorna el valor derecho si el izquierdo es null o undefined
let valor1 = null ?? "valor por defecto";
console.log(valor1); // "valor por defecto"

let valor2 = undefined ?? "valor por defecto";
console.log(valor2); // "valor por defecto"

let valor3 = 0 ?? "valor por defecto";
console.log(valor3); // 0 (no es null ni undefined)

let valor4 = "" ?? "valor por defecto";
console.log(valor4); // "" (no es null ni undefined)
```

Null Coalescing Operator (??) - Parte 2

```
// Diferencia con ||
let x = 0 || 100;
console.log(x); // 100 (0 es falsy)

let y = 0 ?? 100;
console.log(y); // 0 (0 no es null/undefined)

let texto1 = "" || "default";
console.log(texto1); // "default" ("" es falsy)

let texto2 = "" ?? "default";
console.log(texto2); // "" ("" no es null/undefined)
```

Usa `??` cuando solo quieres valores por defecto para `null` y `undefined`.

Optional Chaining (?.)

```
let usuario = {
    nombre: "Juan",
    direccion: {
        ciudad: "Madrid"
    }
};

// Sin optional chaining
// console.log(usuario.contacto.telefono); // ❌ Error

// Con optional chaining
console.log(usuario.contacto?.telefono); // undefined

// Con arrays
let usuarios = null;
console.log(usuarios?.[0]?.nombre); // undefined

// Con funciones
let obj = {};
obj.funcion?.(); // No hace nada si funcion no existe

// Combinado con ??
let telefono = usuario.contacto?.telefono ?? "Sin teléfono";
console.log(telefono); // "Sin teléfono"
```

Conceptos Adicionales

Manejo de Errores

Try Catch

```
try {
    // Código que puede fallar
    const resultado = funcionQuePuedeFallar();
    console.log(resultado);
} catch (error) {
    // Manejar el error
    console.error('Ocurrió un error:', error.message);
} finally {
    // Siempre se ejecuta
    console.log('Limpieza');
}

// Lanzar errores personalizados
function dividir(a, b) {
    if (b === 0) {
        throw new Error('No se puede dividir por cero');
    }
    return a / b;
}
```