JavaScript

**JavaScript Basics**

1. **What is JavaScript? Purpose and usage ?**

* JavaScript is a **programming language** primarily used for creating interactive and dynamic content on websites.
* It is a lightweight, high-level, and versatile language supported by all major web browsers.

**Purpose of JavaScript:**Enhancing User Interactivity - validation, drop-down menus,

**Dynamic Content -** real-time without reloading the page

Control of Multimedia - Handles audio, video, and animations

Client-Side Processing - Reduces server load by processing data ->

**Usage of JavaScript** : Web - Front , Back-End

Mobile App - React Native

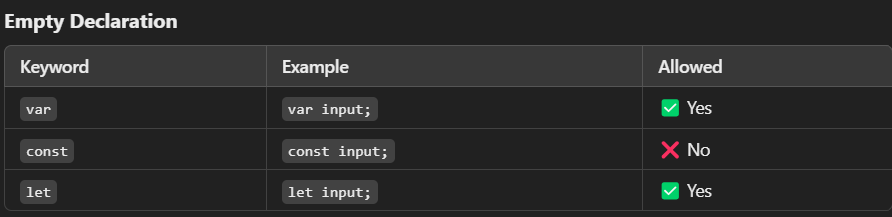
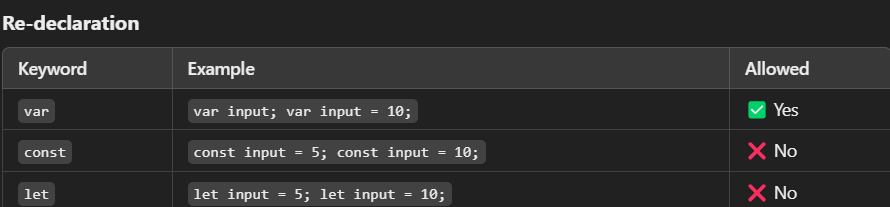
Game Development - browser-based games using libraries like Phaser

1. **Setting Up the JavaScript Environment**

Browser Console : inspect

Code Editor : install node local and extension

1. **Variables: var, let, const**

#### **Scope Examples**

**Var (Function Scoped)**

var input=5;

if (true) {

var input = 15; // Redefines the same variable

console.log(input); // 15

}console.log(input); // 15

**Const(Block Scoped)**

const input = 5;

if (true) {

const input = 15; // Creates a new variable within the block

console.log(input); // 15

}

console.log(input); // 5

**Let (Block Scoped)**

let input = 5;if (true) {

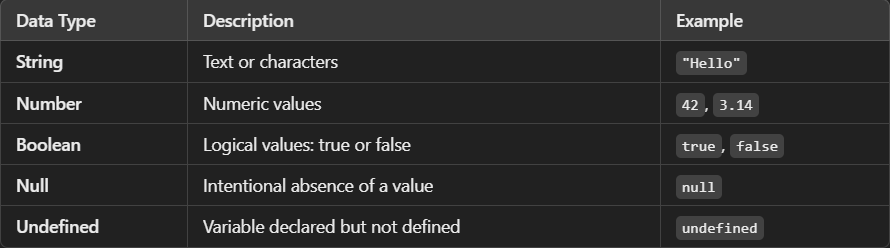
let input = 15; // Creates a new variable within the block

console.log(input); // 15

}console.log(input); // 5



**4.Data Types: string, number, boolean, null, undefined.**

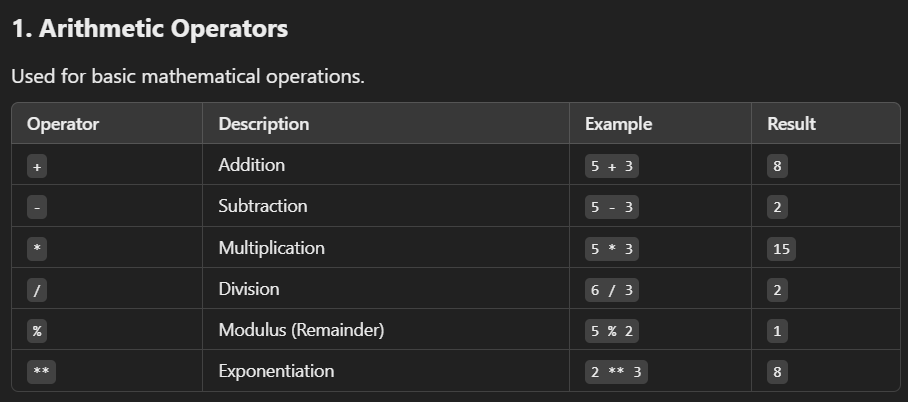
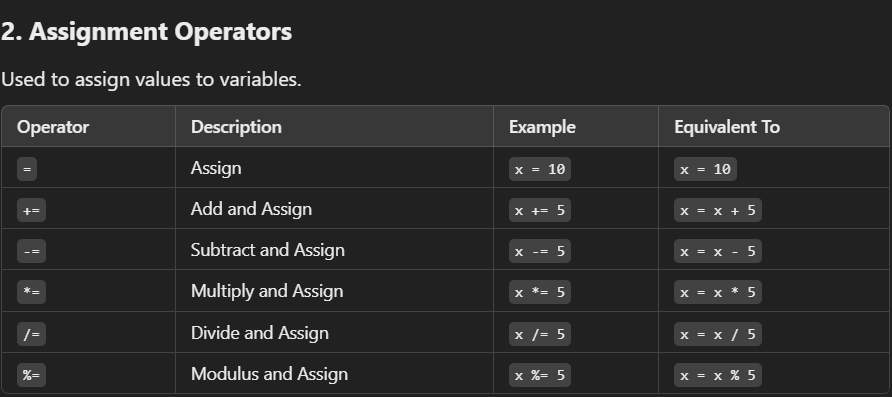


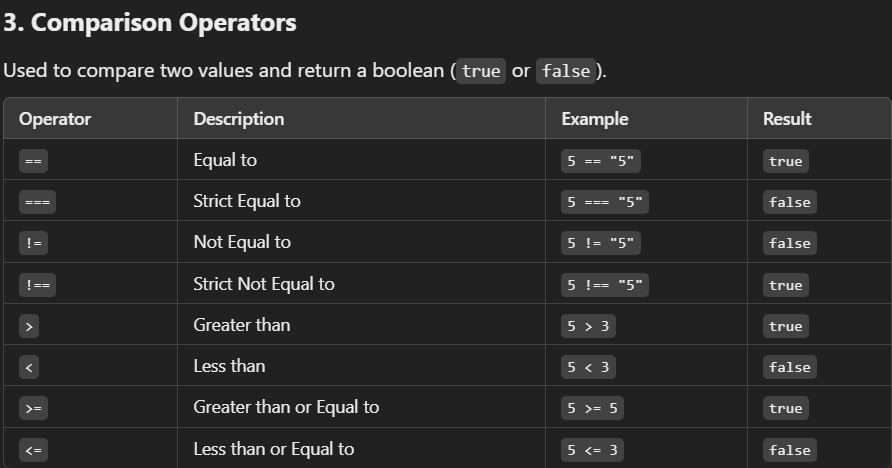
### Key Differences

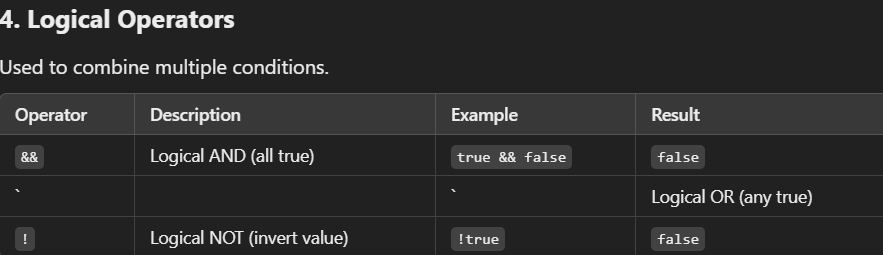
* **letand var** allow reassignment with any data type.
* const requires an initial value and cannot be reassigned.

All variable declaration types (var, let, const) can store any data type in JavaScript. The choice depends on scope, reassignment, and coding style.

**Operators: Arithmetic, Assignment, Comparison, Logical.**





**==========================Control Flow============================**  
  
**Conditional Statements**: if, else if, else, switch.

**Loops:** for, while, do...while, for...of, for...in.

**Logical operators** in flow control.

-----------**Conditional Statement------------**

let age = 20;

**if** (age < 18) { console.log("You are a minor."); }

**else if** (age >= 18 && age < 65) { console.log("You are an adult."); }

**else** { console.log("You are a senior citizen."); }

If - Checks the first condition and runs the code if it is true

Else - Runs when none of the above conditions are true

Else if - Adds more conditions to check between if and else

**Switch**-Used for multiple conditions,often simpler than if-else for equality checks

let day = 3;

switch (day) {

case 1:

console.log("Monday");

break;

case 2:

console.log("Tuesday");

break;

case 3:

console.log("Wednesday");

break;

default:

console.log("Invalid day");

}

**--------------Loops** ------------

**For-Loop -** Repeats as long as the condition is true

**for** (let i = 0; i < 5; i++) {

console.log(i); // Output: 0, 1, 2, 3, 4

}

**While-Loop -** Repeats as long as the condition is true

**let** i = 0;

**while** (i < 5) {

console.log(i);

i++;

}

**Do-WhileLoop -** Executes code at least once,then repeats as long as the condition is true

**let** i = 0;

**do** {

console.log(i);

i++;

} **while** (i < 5);

**For-of -** Iterates over iterable objects like arrays

let arr = [10, 20, 30];

for (**let** value **of** arr) {

console.log(value); // Output: 10, 20, 30

}

**For-in -** Iterates over the keys of an object

let obj = { a: 1, b: 2, c: 3 };

for (let key in obj) {

console.log(key); // Output: a, b, c

}

**---------------Logical Operators in Flow Control---------------**

**1.**&& **(AND)**: Both conditions must be true

**let** age = 25;

**if** (age > 18 **&&** age < 30) {

console.log("Young adult");

}

**2.|| (OR)**: At least one condition must be true

**let** isWeekend = true;

**let** isHoliday = false;

**if** (isWeekend || isHoliday) {

console.log("Relax, it's a break!"); // output will show

}

3.**! **(NOT)****: Inverts the condition

**let** isLoggedIn = false;

**if** (!isLoggedIn) {

console.log("Please log in."); // output will show

}

**Example** // even or odd using conditionals and loops

**let** numbers = [1, 2, 3, 4, 5];

**for** (**let** num **of** numbers) {

**if** (num % 2 === 0) {

console.log(`${num} is even.`);

}

**else** {

console.log(`${num} is odd.`);

}

}

### **============================**Functions and Scope in JavaScript**======================**

### **//**Functions are reusable blocks of code that perform specific tasks.

#### ****1.Function Declaration****

* Declared using the **function** keyword
* Can be called anywhere in the code (**hoisted**)

**Example: function** greet(name) {

return `Hello, ${name}!`;

}

console.log(greet("John")); // Output: Hello, John!

**2.Function Expression**

* Stored in a variable and is not hoisted.

**Example** : **const** greet = **function**(name) {

return `Hello, ${name}!`;

};

console.log(greet("John")); // Output: Hello, John!

**3. Arrow Functions**

const greet = (name) => `Hello, ${name}!`;

console.log(greet("John")); // Output: Hello, John!

### ****------------------Parameters and Return Values------------------------****

* **Parameters**: Variables passed into the function.
* **Return Value**: The result a function gives back after execution

**function** add(a, b) {

return a + b; // Returns the sum of a and b

}

console.log(add(3, 5)); // Output: 8

**------------------Scope--------------------**

**//** a program where a variable is accessible . variables, functions, and objects

### ****Why Scope Matters****

* Helps avoid naming conflicts.
* Controls variable visibility, improving code security and maintainability

### ****Key Points to Remember****

1. **Global Scope**: Variables accessible everywhere.
2. **Local Scope**: Variables restricted to the function they are declared in.
3. **Block Scope**: Variables restricted to the block (for let and const)

#### 1. ****Global Scope****

* Variables declared outside of any function or block have a global scope.
* These variables are accessible **anywhere in the program**

**let** globalVar = "I am global";

**function** showGlobal() {

console.log(globalVar); **// Accessible**

}

showGlobal();

console.log(globalVar); **// Accessible**

#### 2. ****Local Scope (Function Scope)****

* Variables declared inside a function are local to that function.
* They are not accessible outside the function

**function** localScope() {

**let** localVar = "I am local";

console.log(localVar); **// Accessible inside the function**

}

localScope();

// console.log(localVar); // **Error**: localVar is not defined

#### ****3. Block Scope****

* Variables declared with let or const are limited to the block they are defined in.

**if** (true) {

**let** blockVar = "I am block scoped";

console.log(blockVar); // **Output**: I am block scoped

}

// console.log(blockVar); // **Error**: blockVar is not defined

**4.closure**

**//** a function to "remember" variables ,To **protect or encapsulate data** in a function

**function** addPrefix(prefix) {

**return function** (name) {

**return** `${prefix} ${name}`;

};

}

**const** addMr = addPrefix("Mr.");

console.log(addMr("John")); // **Output**: Mr. John

console.log(addMr("Doe")); // **Output**: Mr. Doe

### ****5. IIFE (Immediately Invoked Function Expression)****

// Functions that are executed immediately after they are defined.like **let**, **const**, and modules often reduce the need for them

**Example 1:**

(**function**() {

console.log("I run immediately!");

})**()**; // **Output**: I run immediately!

**Example 2:**

**function** outer() {

**let** count = 0; // Declare count in the outer function.

**return function** () { // Return an inner function.

count++; // Increment count.

**return** count; // Return the updated count.

};

}

const counter = outer(); // Call outer() and store the returned function in counter.

console.log(counter()); // Output: 1

console.log(counter()); // Output: 2

**Example: finding the factorial**

**function** factorial(n) {

if (n < 0) return "negative numbers";

if (n === 0 || n === 1) return 1;

let result = 1;

for (let i = 2; i <= n; i++) {

result \*= i;

}

return result;

}

console.log(factorial(5)); // Output: 120

console.log(factorial(0)); // Output: 1

console.log(factorial(-3)); // Output: negative numbers

**Reverse String**

**function** reverse(str){

**return** str.**split**("").**reverse**().**join**("");

}

console.log(reverse("siva")); // Output: "avis"