**Day 2:Prompts, Arithmetic and logical operator, variables and data types.**

**1.Alert and Prompts**

**Code:**

alert("Hello, world!"); // alert pop’s message in the display

**output:**

“Hello world”

**Code:**

let userName = prompt("What is your name?");

alert("Hello, " + userName + "!");

**Output:**

“Hello joe!”

**2.Arithmetic and logical operators**

**Code:**

let a = 5;

let b = 3;

let sum = a + b; // Addition

let difference = a - b; // Subtraction

let product = a \* b; // Multiplication

let quotient = a / b; // Division

let remainder = a % b; // Modulus

console.log("Sum: " + sum); // console.log display’s output in console

console.log("Difference: " + difference);

console.log("Product: " + product);

console.log("Quotient: " + quotient);

console.log("Remainder: " + remainder);

**Output:**

Sum: 8

Difference: 2

Product: 15

Quotient: 1.6666666666666667

Remainder: 2

**Code:**

let x = true;

let y = false;

let andResult = x && y; //and both input should be true to get true output

let orResult = x || y; //or Pop’s true in distinct input

let notResult = !x; //not contradict’s the input

console.log("AND Result: " + andResult);

console.log("OR Result: " + orResult);

console.log("NOT Result: " + notResult);

**Output:**

AND Result: false

OR Result: true

NOT Result: false

**3.Variables and Data types**

**var**: The traditional way to declare a variable. It has function scope and can be re-assigned

**Code:**

var name = "Alice";

var age = 25;

console.log(name); // Output: Alice

console.log(age); // Output: 25

name = "Bob"; // Re-assigning value

console.log(name); // Output: Bob

**let**: Introduced in ES6, it has block scope and can be re-assigned

**Code:**

let city = "New York";

let temperature = 75;

console.log(city); // Output: New York

console.log(temperature); // Output: 75

city = "Los Angeles"; // Re-assigning value

console.log(city); // Output: Los Angeles

**const**: Declares a block-scoped constant (cannot be reassigned).

**Code:**

const pi = 3.14; console.log(pi); // Output: 3.14

**Examples of different data types**

let name = "Alice"; // String

let age = 30; // Number

let isStudent = true; // Boolean

let person = { name: "Bob", age: 25 }; // Object

let numbers = [1, 2, 3]; // Array

let notDefined; // Undefined

let nothing = null; // Null

console.log(name); // Output: Alice

console.log(age); // Output: 30

console.log(isStudent); // Output: true

console.log(person); // Output: { name: 'Bob', age: 25 }

console.log(numbers); // Output: [1, 2, 3]

console.log(notDefined); // Output: undefined

console.log(nothing); // Output: null

**Day 3:functions, returning values, conditional, switch and looping statements.**

**1.Function declaration:**

**Code:**

// Function declaration

function square(number) {

return number \* number;

}

// Function can be called before its definition due to hoisting

console.log(square(5)); // Output: 25

// Function call after definition

console.log(square(10)); // Output: 100

**2.Function expression:**

const add = function(a, b) {

return a + b;

};

console.log(add(2, 3)); // Output: 5

**3.Arrow function:**

const square = x => x \* x;

console.log(square(6)); // Output: 36

**4.** **Immediately Invoked Function Expressions (IIFE):**

(function() {

let msg = "This is an IIFE!";

console.log(msg); // Output: This is an IIFE!

})();

**5.Anonymous function:**

const numbers = [1, 2, 3, 4];

const doubled = numbers.map(function(number) {

return number \* 2;

});

console.log(doubled); // Output: [2, 4, 6, 8]

**6.Higher order function:**

function outerFun() {

function nestedFun() {

let local = "I am a local variable";

return local;

}

let result = nestedFun();

console.log(result);

}

outerFun(); // output:I am a local variable

The above code shows us we can call a local scope’s value outside its local function.

**7.call back function:**

function sayHello() {

console.log("Hello!");

}

function executeCallback(callback) {

callback();

}

executeCallback(sayHello); // Output: Hello!

**2.Returing Values**

let a = 10;

function outerFun() {

{

console.log("This function runs immediately!");

//Output:this function runs immediately

}

Function nestedFun(a,b){

return a + b;

}

console.log(nestedFun(2, 3)); //Output:5

}

outerFun();

**3.Conditional Statements**

**1.if-else:**

let age = 18;

if (age >= 18) {

console.log("You are an adult."); //Output:You are an adult

} else {

console.log("You are a minor.");

}

**2.if-else if-else ladder:**

let score = 85;

if (score >= 90) {

console.log("Grade: A");

} else if (score >= 80) {

console.log("Grade: B");

} else if (score >= 70) {

console.log("Grade: C");

} else {

console.log("Grade: F");

}

**3.Nested if-else:**

let num = 15;

if (num > 0) {

if (num % 2 === 0) {

console.log("The number is positive and even.");

} else {

console.log("The number is positive and odd.");

} //Output:The number id positive and odd.

} else {

console.log("The number is negative or zero.");

}

**4.Switch statements**

let i = 3;

let order;

switch (i) {

case 1:

order = "laptop";

break;

case 2:

order = "mobile";

break;

case 3:

order = "tv"; //Output:tv

break;

default:

order = "unknown";

break;

}

console.log(order);

**5.Looping statements**

**1.For loop:**

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

console.log("Iteration number: " + i);

}

Output:

Iteration number: 0

Iteration number: 1

Iteration number: 2

Iteration number: 3

Iteration number: 4

**2.while loop:(Entry check loop)**

let i = 0;

while (i < 5) {

console.log("Iteration number: " + i);

i++;

}

Output:

Iteration number: 0

Iteration number: 1

Iteration number: 2

Iteration number: 3

Iteration number: 4

**3.do-while loop:(Exit check loop)**

let i = 0;

do {

console.log("Iteration number: " + i);

i++;

} while (i < 5);

Output:

Iteration number: 0

Iteration number: 1

Iteration number: 2

Iteration number: 3

Iteration number: 4

**Day 4:Hoisting and scope, Array and array methods, object and json iteration.**

**1.Hoiting:**

Function hoisting:

console.log(add(2, 4)); // Function is called before its declaration

//Output:6

function add(a, b) {

return a + b;

}

Variable hoisting:

console.log(a); // Output: undefined

var a = 10;

console.log(a); // Output: 10

**2.Scope:(local and global scope):**

var globalVar = "I am global";

function myFunction() {

var localVar = "I am local";

console.log(globalVar); // I am global

console.log(localVar); // I am local

return localVar;

}

var result = myFunction(); // Store the returned value of localVar

console.log(result); // Now you can access the value of localVar

**3.Array and Array methods:**

// Initial array

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

// push - Add elements to the end of the array

numbers.push(6); // [1, 2, 3, 4, 5, 6]

console.log('After push:', numbers);

// pop - Remove the last element from the array

let lastElement = numbers.pop(); // [1, 2, 3, 4, 5]

console.log('After pop:', numbers);

console.log('Removed element:', lastElement);

// shift - Remove the first element from the array

let firstElement = numbers.shift(); // [2, 3, 4, 5]

console.log('After shift:', numbers);

console.log('Removed element:', firstElement);

// unshift - Add elements to the beginning of the array

numbers.unshift(1); // [1, 2, 3, 4, 5]

console.log('After unshift:', numbers);

// slice - Create a new array by extracting a portion of the original array

let slicedArray = numbers.slice(1, 4); // [2, 3, 4]

console.log('Sliced array:', slicedArray);

// splice - Modify the contents of an array by removing or replacing elements

numbers.splice(2, 1, 10, 20); // Remove 1 element at index 2, and add 10 and 20

console.log('After splice:', numbers);

// map - Create a new array with the results of calling a function on every element

let doubledNumbers = numbers.map(num => num \* 2); // [2, 4, 20, 40]

console.log('Doubled numbers:', doubledNumbers);

// filter - Create a new array with all elements that pass the test implemented by the provided function

let evenNumbers = numbers.filter(num => num % 2 === 0); // [2, 20]

console.log('Even numbers:', evenNumbers);

// forEach - Execute a provided function once for each array element

console.log('Numbers array elements:');

numbers.forEach(num => console.log(num));

**4.Object and JSON iteration:**

**1. Iterating Over Objects:**

**Using for-in Loop:**const person = {

name: "John",

age: 25,

city: "San Francisco"

};

// Using for...in to iterate over the object's properties

for (let key in person) {

console.log(key + ": " + person[key]);

}

Output:

name: John

age: 25

city: San Francisco

**2.Using Object.keys() and Object.values():**const person = {

name: "John",

age: 30,

city: "New York"

};

// Get an array of the object's keys

let keys = Object.keys(person);

// Get an array of the object's values

let values = Object.values(person);

console.log(keys); // ["name", "age", "city"]

console.log(values); // ["John", 30, "New York"]

Output:

["name", "age", "city"]

["John", 30, "New York"]

**3.Iterating Over JSON:**

let jsonData = [

{ name: "suresh", age: 35, city: "coimbatore" },

{ name: "vel", age: 33, city: "salem" },

{ name: "rajesh", age: 34, city: "erode" }

];

// Using forEach to iterate over the array and log the details

jsonData.forEach((i, ind) => {

console.log(i.name);

console.log(i.age);

console.log(i.city);

});

// Using map to filter and transform the array based on a condition

let result = jsonData.map((i, ind) => {

if (i.age > 33) { // Condition: Age must be greater than 33

console.log(i.name);

console.log(i.age);

console.log(i.city);

return { name: i.name, age: i.age, city: i.city }; // Returning the filtered user object

}

return null; // Return null if the condition is not met

});

console.log(result);

Output:

suresh

35

coimbatore

vel

33

salem

rajesh

34

erode