Primitive Data Types

String: Represents textual data.

Number: Represents both integer and floating-point numbers.

Boolean: Represents true or false values.

Undefined: Represents a variable that has been declared but not assigned a value.

Null: Represents an intentional absence of any object value.

Symbol: Represents a unique identifier.

BigInt: Represents large integers.

Non-Primitive Data Types

Object: Represents collections of key-value pairs.

Array: Represents ordered collections of values.

Function: Represents executable code.

Date: Represents dates and times.

RegExp: Represents regular expressions.

==========

In JavaScript- let, var, and const are used to declare variables

let is used to declare block-scoped variables.

It can be reassigned but not redeclared within the same scope.

var is function-scoped or globally-scoped if declared outside a function.

It can be reassigned and redeclared within the same scope.

const

const is used to declare block-scoped variables that cannot be reassigned.

However, the contents of objects and arrays declared with const can be modified.

==========
=

```
import {test,chromium} from '@playwright/test';
test('check variable types', async ({ page }) => {
        let str = "Hello, Playwright!";
        let num = 42;
        let bool = true;
        let obj = { key: "value" };
        let arr = [1, 2, 3];

        let date = new Date();
        let regex = /abc/;
        let arr1 = [1, 2, 3];


   console.log(typeof str);
   console.log(typeof num);
   console.log(typeof bool);
   console.log(typeof obj);
   console.log(typeof arr);

        console.log(date instanceof Date);
        console.log(regex instanceof RegExp);
```

```
        console.log(arr instanceof Array);

        });


test('let example', async ({ page }) => {
        let count = 0;
        await page.goto('https://www.google.com/');

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

        console.log(count);
        });


test('var example', async ({ page }) => {
        var message = "Hello";
        await page.goto('https://www.google.com/');

        if (true) {
        var message = "Welcome";
        console.log(message);
        }

        console.log(message);
        });
```

```
test('const example', async ({ page }) => {
        const url = 'https://example.com';
        await page.goto(url);

        const user = { name: 'Rathi', age: 30 };
    user.age = 37;

        console.log(user);

        const numbers = [1, 2, 3];
    numbers.push(4);

        console.log(numbers);
        });
```

==========

loops:

```
console.log("For Loop:");
for (let i = 1; i <= 5; i++) {
  console.log(`Iteration ${i}`);
}

// While loop
console.log("\nWhile Loop:");
```

```
let j = 1;
while (j <= 5) {
  console.log(`Iteration ${j}`);
  j++;
}

// Do-While loop
console.log("\nDo-While Loop:");
let k = 1;
do {
  console.log(`Iteration ${k}`);
  k++;
} while (k <= 5);

// For...of loop (array iteration)
console.log("\nFor...of Loop:");
const fruits = ["apple", "banana", "cherry"];
for (const fruit of fruits) {
  console.log(fruit);
}

// For...in loop (object iteration)
console.log("\nFor...in Loop:");
const person = { name: "Alice", age: 30, city: "Chennai" };
for (const key in person) {
  console.log(`${key}: ${person[key]}`);
}
```

Simple if condition

```
const age = 20;

if (age >= 18) {
  console.log("You are eligible to vote.");
}
```

If -else:

```
const score = 45;

if (score >= 50) {
  console.log("You passed the test.");
} else {
  console.log("You failed the test.");
}
```

If else- if else

```
const marks = 85;

if (marks >= 90) {
  console.log("Grade: A");
} else if (marks >= 75) {
  console.log("Grade: B");
} else {
  console.log("Grade: C");
}
```

```
================================================================================
=
// Array declaration
const numbers = [10, 20, 30, 40, 50];
const fruits = ['apple', 'banana', 'cherry'];

// forEach - iterate and print each number
console.log("Using forEach:");
numbers.forEach((num, index) => {
  console.log(`Index ${index}: ${num}`);
});

// map - create a new array with doubled values
const doubled = numbers.map(num => num * 2);
console.log("\nDoubled values using map:", doubled);

// filter - get numbers greater than 25
const filtered = numbers.filter(num => num > 25);
console.log("\nFiltered values > 25:", filtered);

// reduce - sum of all numbers
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
console.log("\nSum using reduce:", sum);

// find - find the first number greater than 30
const found = numbers.find(num => num > 30);
console.log("\nFirst number > 30 using find:", found);
```

```
// includes - check if array contains a value
console.log("\nDoes array include 20?", numbers.includes(20));


// push - add an element
fruits.push('date');
console.log("\nFruits after push:", fruits);


// pop - remove last element
fruits.pop();
console.log("Fruits after pop:", fruits);
```

OOP Concepts Covered:

- **Class**
- **Object**
- **Constructor**
- **Encapsulation**
- **Inheritance**
- **Polymorphism**

```
// Base class (Parent)
class User {
  constructor(name, role) {
        this.name = name;
    this.role = role;
  }

  displayInfo() {
    console.log(`Name: ${this.name}, Role: ${this.role}`);
  }
}

// Derived class (Child) - Inheritance
class Admin extends User {
  constructor(name, permissions) {
    super(name, 'Admin');
    this.permissions = permissions;
  }

  // Method overriding - Polymorphism
  displayInfo() {
```

```
    super.displayInfo();
    console.log(`Permissions: ${this.permissions.join(', ')}`);
  }
}

// Another derived class
class Guest extends User {
  constructor(name) {
    super(name, 'Guest');
  }

  displayInfo() {
    super.displayInfo();
    console.log('Limited access');
  }
}

// Object creation
const adminUser = new Admin('Ambika', ['read', 'write', 'delete']);
const guestUser = new Guest('Hari');

// Method calls
adminUser.displayInfo();
console.log('---');
guestUser.displayInfo();

const { chromium } = require('playwright'); // or 'firefox' or 'webkit'


(async () => {

  // Launch browser

  const browser = await chromium.launch({ headless: false }); // set headless: true to run without
UI

  const context = await browser.newContext();

  const page = await context.newPage();


  // Navigate to a website

  await page.goto('https://playwright.dev/');
```

```
  // Print the page title
  const title = await page.title();
  console.log('Page title:', title);

  // Close browser
  await browser.close();
})();
```

==========