# Data Types and Variables:

**Different Data Types in JavaScript:**

**Primitive Data Types:**

Number: Represents both integers and floating-point numbers.

String: A sequence of characters, e.g., `"hello"`.

Boolean: Logical values `true` or `false`.

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

Null: Represents the intentional absence of any object value.

Symbol: A unique and immutable data type often used as object keys.

BigInt: Can represent numbers larger than the limit of the `Number` type.

**Non-Primitive Data Types:**

Object: Used to store collections of data and more complex entities.

**Difference Between `var`, `let`, and `const`:**

**var:** Declares a variable globally or function-scoped. Variables can be re-declared or updated.

**let:** Declares a block-scoped variable. Variables can be updated but not re-declared within the same block.

**const:** Declares a block-scoped constant. It cannot be re-assigned or re-declared.

**Why JavaScript Allows Assigning Different Data Types to the Same Variable:**

JavaScript is a dynamically typed language, meaning variables are not bound to a specific type. You can change a variable’s type by simply assigning a new value of a different type.

**Handling Variables Declared But Not Initialized:**

A variable that is declared but not initialized has the value `undefined`.

**code example**

let x;

console.log(x); // undefined

**Significance of Variable Names:**

Variable names in programming are important for clarity, readability, and maintainability. In JavaScript, meaningful variable names help in understanding the purpose of the variable and its role in the program.

# Numeric Data Types:

**Various Numeric Data Types**

Integer: Whole numbers like `10`, `-5`.

loating-point (double): Numbers with decimal points like `3.14`, `-2.718`.

Infinity: A special value that results from dividing a non-zero number by `0`, e.g., `10 / 0`.

**Difference Between Integers, Doubles, and Infinity:**

Integers: Whole numbers without a fractional part.

Doubles (floating-point): Numbers with a fractional part.

Infinity: Represents a value larger than any number or negative infinity for the smallest value.

**Handling Arithmetic Operations:**

JavaScript handles arithmetic operations by automatically converting numeric types when necessary (e.g., adding an integer to a floating-point number).

# string Data Type:

**String Representation in JavaScript**:

Strings are sequences of characters enclosed in either single (`'`) or double (`"`) quotes.

**Difference Between Single and Double Quotes**:

There is no functional difference between single and double quotes. They can be used interchangeably, but consistency is key.

**Automatic Conversion of Characters to Strings:**

In JavaScript, any character or group of characters enclosed in quotes is treated as a string, regardless of the number of characters.

# Boolean and Undefined Data Types:

**Purpose of Boolean Variables:**

Boolean variables represent logical entities and can have two values: `true` or `false`. They are useful for controlling flow in conditions.

**Concept of Undefined:**

A variable that is declared but not assigned a value will have the value `undefined`.

**Example code**

let y;

console.log(y); // undefined

**Use of Booleans in Conditional Statements:**

Boolean variables are critical in conditional logic, enabling the execution of code blocks based on `true` or `false` values.

**Code example**

let isValid = true;

if (isValid) {

console.log('Valid!');

}

# Null Data Type:

**Significance of `null`:**

`null` is used to represent an intentional absence of any object value. It’s a placeholder for an object that doesn’t yet exist.

**Difference Between `null` and `undefined`:**

**`undefined`** means a variable has been declared but has no value.

**`null`** is an assignment value that represents "no value" intentionally.

**Example:**

let x = null;

console.log(x); // null

# Object Data Type:

**How Objects are Represented in JavaScript:**

Objects are collections of key-value pairs, where the keys are strings (or Symbols), and the values can be any data type.

**example**

let person = {

name: "John",

age: 30

};

**Structure of the `countryInfo` Object:**

Suppose the `countryInfo` object contains data about countries. It would look like:

let countryInfo = {

name: "Kenya",

capital: "Nairobi"

};

**Nesting Objects:**

Objects can contain other objects as values.

**example**

let countryInfo = {

name: "Kenya",

capital: "Nairobi",

population: {

males: 1000000,

females: 1100000

}

};

# Array Data Type:

**Purpose and Structure of Arrays:**

Arrays store ordered lists of values, which can be any data type.

**example**

let fruits = ["apple", "banana", "orange"];

**Arrays with Different Data Types:**

Arrays can hold multiple data types.

**example**

let mixedArray = [42, "hello", true, null];

**Array of Arrays (2D Arrays):**

Arrays can contain other arrays, allowing for multi-dimensional data representation.

**example**

let matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

];

# **Variable Naming Conventions:**

**Naming Conventions:**

- Variable names can contain letters, digits, underscores, and dollar signs.

- They must begin with a letter, underscore, or dollar sign.

- Camel case is commonly used in JavaScript (`myVariableName`).

**Importance of Descriptive Names:**

Descriptive names make code more readable and maintainable. It’s good practice to choose names that convey the purpose of the variable.

**Naming Conventions Followed or Violated:**

Ensure that variables are properly named following camel case and not starting with digits.

# Constants in JavaScript:

**Use of `const` Keyword:**

Constants (`const`) are block-scoped and cannot be re-assigned once defined. They ensure that a variable’s value remains constant throughout the program.

**Error when Reassigning a Constant:**

Reassigning a constant leads to an error.

**example**

const PI = 3.14;

PI = 3.14159; // Error: Assignment to constant variable.

**Example from Code:**

const MAX\_HEIGHT = 200;

console.log(MAX\_HEIGHT); // 200