JS Data Types

# 1. What are Data Types?

Data types specify the type of data that a variable can hold. In JavaScript, there are two main categories of data types:

**1. Primitive Data Types**

**2. Non-Primitive (Reference) Data Types**

# 2. Primitive Data Types

Primitive data types are the most basic data types in JavaScript. They represent **single values** and are **immutable**, meaning their values cannot be changed once created.

## a. String

**- What It Is:** Represents textual data.

**- How to Use:** Enclosed in single quotes `' '`, double quotes `" "`, or backticks `` ` ` ``.

**Example:**

let greeting = "Hello, Mahesh!";

let singleQuote = 'Lets learn JavaScript';

let templateString = `I can embed variables like ${greeting}`;

console.log(greeting);

console.log(singleQuote);

console.log(templateString)

## b. Number

**- What It Is:** Represents both integer and floating-point numbers.

**- How to Use:** Written without quotes.

let age = 25;

let pie = 3.14;

## c. Boolean

**- What It Is:** Represents logical values—true or false.

**- How to Use:** No quotes, directly using `true` or `false`.

let welearnjs = true;

let welearnmysql = false;

console.log(welearnjs);

console.log(welearnmysql)

## d. Undefined

**- What It Is:** A variable that has been declared but not assigned a value.

**- How to Use:** Automatically assigned by JavaScript.

let notDefined;

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

## e. Null

**- What It Is:** Represents the intentional absence of any object value.

**- How to Use:** Assigned by the programmer to indicate "no value".

let emptyValue = null;

console.log(emptyValue); // Output: null

## f. Symbol (ES6)

**- What It Is:** Represents a unique and immutable value, often used as object keys.

**- How to Use:** Created using `Symbol()`.

let uniqueId = Symbol('id');

let anotherUniqueId = Symbol('id');

console.log(uniqueId === anotherUniqueId); // Output: false

## g. BigInt (ES2020)

**- What It Is:** Represents integers with arbitrary precision, allowing the representation of very large numbers.

**- How to Use:** Appended with `n` or using the `BigInt()` function.

let bigNumber = 1234567890123456789012345678901234567890n;

let anotherBigNumber = BigInt("12345678901234567890");

# 3. Non-Primitive (Reference) Data Types

Non-primitive data types are more complex and can hold collections of values or more complex entities. They are mutable, meaning their contents can be changed.

## a. Object

**- What It Is:** Represents a collection of key-value pairs.

**- How to Use:** Defined using curly braces `{}`.

let person = {

name: "Mahesh",

age: 25,

};

## b. Array

**- What It Is:** A special type of object used to store ordered collections of values.

**- How to Use:** Defined using square brackets `[]`.

let fruits = ["Apple", "Banana", "Orange"];

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

## c. Function

**- What It Is:** A block of code designed to perform a particular task.

**- How to Use:** Defined using the `function` keyword or arrow syntax.

// Ex:10 (Functions)

function greet() {

    console.log("Hello Mahesh");

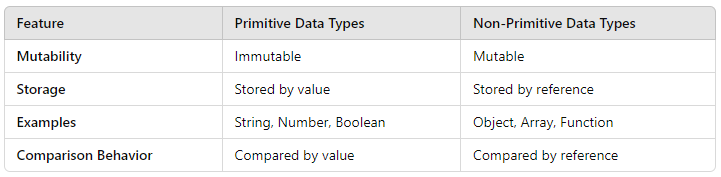
}

greet()

let add = (a, b) => console.log(a + b);

add(10, 20)

# 4. Comparing Primitive and Non-Primitive Data Types



# 5. Tips for Remembering Data Types

**1. Primitives are Simple:** Think of them as single, indivisible values like a single word (e.g., a name) or number.

**2. Non-Primitives are Complex:** Think of them as collections, like lists of items (arrays) or detailed profiles (objects).

**3. Immutability vs. Mutability:** Primitive values can't be changed once set, while non-primitive types can be modified.

# 6. Practice Time!

Let’s solidify your understanding with some simple exercises.

## Exercise 1: Identify the Data Types

For each variable below, identify its data type.

let name = "Manick"; // ?

let score = 95.5; // ?

let isPassed = true; // ?

let data; // ?

let emptyValue = null; // ?

let uniqueKey = Symbol('key'); // ?

let bigNumber = 12345678901234567890n; // ?

let person = { name: "Manick", age: 1 }; // ?

let fruits = ["Apple", "Banana", "Cherry"]; // ?

let add = (a, b) => a + b; // ?

## Exercise 2: Create Variables with Different Data Types

1. Create a string variable with your favorite quote.

2. Create a number variable representing your birth year.

3. Create a boolean variable indicating if you like JavaScript.

4. Create an undefined variable.

5. Create a null variable.

6. Create a symbol for a unique identifier.

7. Create a BigInt representing a very large number.

8. Create an object representing a laptop with properties like `brand`, `model`, and `year`.

9. Create an array of your top 3 favorite movies.

10. Create a function that takes two numbers and returns their sum.

## Exercise 3: Manipulate Data Types

1. String: Concatenate two strings to form a full name.

2. Number: Perform arithmetic operations (addition, subtraction) on two numbers.

3. Boolean: Use logical operators to combine two boolean values.

4. Array: Add a new item to an array and remove the last item.

5. Object: Add a new property to an object and update an existing property.