

# JavaScript Data Types – Detailed Notes

JavaScript has two major categories of data types:

1. Primitive Data Types
  2. Non-Primitive Data Types
- 

## 1. Primitive Data Types

Primitive data types are **basic, fundamental values** that are **not objects**.

They are **immutable** (cannot be changed once created) and **stored directly in memory**.

### ✓ List of Primitive Data Types:

1. Number
  2. String
  3. Boolean
  4. Undefined
  5. Null
  6. BigInt
  7. Symbol
- 

#### ◆ 1. Number

Represents numeric values — both integers and decimals.

```
let age = 20;
```

```
let price = 99.99;
```

---

## ◆ 2. String

Text data enclosed in quotes.

```
let name = "Ali";
let message = 'Hello World!';
```

---

## ◆ 3. Boolean

Returns `true` or `false`.

```
let isLoggedIn = true;
```

---

## ◆ 4. Undefined

A variable declared but not assigned any value.

```
let x;
console.log(x); // undefined
```

---

## ◆ 5. Null

Represents empty or non-existent value (manually assigned).

```
let data = null;
```

---

## ◆ 6. BigInt

Helps store numbers larger than JavaScript's safe integer limit.

```
let bigNumber = 12345678901234567890n;
```

---

## ◆ 7. Symbol

Unique values often used as object keys.

```
let id = Symbol("id");
```

---

# Characteristics of Primitive Types

- Immutable
  - Stored by **value**
  - Fast and simple
  - Cannot contain methods or properties (except via wrapper objects internally)
- 

# 2. Non-Primitive Data Types

Non-Primitive data types are **objects**.

They are **mutable** (can be changed).

Stored by **reference**, not value.

### Includes:

- **Arrays**
- **Objects**
- **Functions**

(We will focus on Arrays & Objects as requested.)

---

## 2.1 Arrays

An array is a **collection of values** stored at **indexed positions** (0, 1, 2, ...).

```
let fruits = ["apple", "banana", "mango"];
```

---

### ◆ Array Characteristics

- Ordered collection
  - Index starts from 0
  - Can store mixed data types
  - Mutable (can be changed)
- 

## Array Methods

### 1. push()

Adds element **at the end**.

```
fruits.push("orange");
```

---

### 2. pop()

Removes **last element**.

```
fruits.pop();
```

---

### 3. unshift()

Adds element **at the beginning**.

```
fruits.unshift("kiwi");
```

---

### 4. shift()

Removes the **first element**.

```
fruits.shift();
```

---

## Loops with Arrays

### ✓ 1. for loop

Used when you need the **index**.

```
for (let i = 0; i < fruits.length; i++) {  
  console.log(fruits[i]);  
}
```

---

### ✓ 2. for...in loop

Used to loop **over indices** of an array.

(Not recommended for arrays but good for practice.)

```
for (let index in fruits) {  
  console.log(index, fruits[index]);  
}
```

---

## ✓ 3. for...of loop

Used to loop **over values** of an array.

```
for (let item of fruits) {  
  console.log(item);  
}
```

---

## 2.2 Objects

Objects store data in **key-value pairs**.

```
let student = {  
  name: "Ali",  
  age: 18,  
  grade: "A"  
};
```

---

## Object Features

- Keys are always **strings** (or symbols).
  - Values can be anything: strings, numbers, arrays, functions, even other objects.
  - Accessed using **dot notation** or **bracket notation**.
- 

### ◆ Dot Notation

Used for simple and valid key names.

```
console.log(student.name);
```

```
student.age = 19;
```

---

## ◆ Bracket Notation

Used when:

- Key contains **spaces**
- Key starts with a **number**
- Key includes **special characters**
- Accessing key using a **variable**

```
console.log(student["grade"]);
```

```
let key = "name";
console.log(student[key]); // dynamic access
```

---

## Examples of When to Use Bracket Notation

**Key has spaces:**

```
let person = {
  "first name": "Sara"
};

console.log(person["first name"]);
```

**Key stored in variable:**

```
let k = "age";  
console.log(student[k]);
```

---

## Summary Table: Dot vs Bracket

Feature	Dot Notation	Bracket Notation
Easy to write	✓ Yes	—
Works with spaces	✗ No	✓ Yes
Works with variables	✗ No	✓ Yes
Works with special characters	✗ No	✓ Yes

---

## 🧠 Key Differences: Primitive vs Non-Primitive

Feature	Primitive	Non-Primitive
Stored By	Value	Reference
Mutable?	✗ No	✓ Yes
Types	Number, String, etc.	Objects, Arrays
Example	<code>let x = 5</code>	<code>let arr = []</code>

---



# Final Quick Recap

## ✓ Primitive Data Types

- Basic values
- Immutable
- Example: number, string

## ✓ Non-Primitive Data Types

- Arrays & Objects
- Mutable
- Stored as references

## ✓ Arrays

- Ordered data
- Methods: push, pop, shift, unshift
- Loops: for, for-in, for-of

## ✓ Objects

- Key-value pairs
- Access using dot & bracket notation