JavaScript Study Notes – Variables & Data Types

---

1. What is a Variable?

- A **variable** is like a container that stores data in memory.
- Instead of writing values directly, we give them a **name** so we can use them later.
- Example:

```
let age = 20;   // age is a variable storing number 20
const name = "Umesh";  // name is a variable storing string "Umesh"
```

---

2. Primitive Data Types:

- **Number** → integers, decimals, etc. → `let x = 10;`
- **String** → text enclosed in quotes → `let name = "Umesh";`
- **Boolean** → true/false values → `let isStudent = true;`
- **Null** → represents "nothing" → `let empty = null;`
- **Undefined** → variable declared but no value assigned → `let a;`
- **Symbol** → unique value (used in advanced coding)

2. Types of Variables in JavaScript

There are three keywords to declare variables:

a) `var`

- **Scope:** Function-scoped (available throughout a function).
- **Redeclaration:** Allowed.
- **Reassignment:** Allowed.
- **Hoisting:** Variables declared with `var` are "hoisted" (moved to the top of their scope) but initialized with `undefined`.
- **Use case:** Not recommended in modern JS (can create bugs).

Example:

```
var x = 10;
var x = 20;  // redeclaration allowed
console.log(x); // 20
```

---

b) `let`

- **Scope:** Block-scoped (only available inside `{}` where declared).
- **Redeclaration:** ❌Not allowed in the same block.
- **Reassignment:** ✅Allowed.
- **Hoisting:** Hoisted but not initialized (gives error if used before declaration).
- **Use case:** Best choice for variables that change.

Example:

```
let y = 30;
y = 40; // allowed
console.log(y); // 40
// let y = 50; ❌ Error
```

---

c) `const`

- **Scope:** Block-scoped.
- **Redeclaration:** ❌Not allowed.
- **Reassignment:** ❌Not allowed (value is constant).
- **Hoisting:** Hoisted but not initialized.
- **Use case:** Use for values that should never change.

Example:

```
const PI = 3.14159;
console.log(PI);
// PI = 3.14; ❌ Error
```

✅When to use what?

- Use `const` by default.
- Use `let` if the value will change.
- Avoid `var` in modern JS.

---

3. Data Types in JavaScript

Data type = kind of value a variable can hold.

JavaScript has **two categories of data types**:

(A) Primitive Data Types

These are **simple values** stored directly in memory.
They are **immutable** (cannot be changed directly).

1. **Number**
2. Stores integers and decimals.
3. Example:

```
let num1 = 42;
let num2 = 3.14;
console.log(typeof num1); // number
```

4. Special values: `Infinity`, `-Infinity`, `NaN` (Not a Number).
5. **String**
6. Stores text inside quotes (`"`, `'`, or `` ` ``).
7. Supports template literals with backticks.
8. Example:

```
let str = "Hello";
let message = `Hi, ${str}`;
console.log(message); // Hi, Hello
```

9. **Boolean**
10. Stores `true` or `false`.
11. Example:

```
let isAI = true;
let isHuman = false;
```

12. **Null**
13. Represents "empty" or "nothing".
14. Example:

```
let data = null;
console.log(data); // null
console.log(typeof data); // object (bug in JS)
```

15. **Undefined**

16. Variable declared but not assigned any value.

17. Example:

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

18. **Symbol**

19. Used to create unique identifiers.

20. Mostly used in advanced coding (objects).

21. Example:

```
let sym1 = Symbol("id");
let sym2 = Symbol("id");
console.log(sym1 === sym2); // false
```

22. **BigInt**

23. For very large numbers (greater than Number limit).

24. Example:

```
let big = 1234567890123456789On;
console.log(typeof big); // bigint
```

(B) Non-Primitive Data Types (Reference Types)

These are **complex data structures**. They store **references (addresses in memory)**.

1. **Object**

2. Collection of key-value pairs.

3. Example:

```
let person = { name: "Umesh", age: 20 };
console.log(person.name); // Umesh
```

4. **Array**

5. Ordered collection of values.

6. Example:

```
let colors = ["red", "blue", "green"];
console.log(colors[0]); // red
```

7. **Function**

8. A block of code that performs a task.

9. Example:

```
function greet() {
  return "Hello!";
}
console.log(greet());
```

---

4. Type Checking – `typeof`

The `` operator tells us the type of a value.

Examples:

```
console.log(typeof 100);       // number
console.log(typeof "AI");      // string
console.log(typeof true);      // boolean
console.log(typeof undefined); // undefined
console.log(typeof null);      // object (special case)
```

```
console.log(typeof Symbol());  // symbol
console.log(typeof 123n);      // bigint
```

---

5. Type Conversion

Sometimes we need to change a value from one type to another.

a) Convert to String

```
let num = 123;
console.log(String(num));   // "123"
console.log(num.toString()); // "123"
```

b) Convert to Number

```
let str = "456";
console.log(Number(str)); // 456
console.log(parseInt("10.5")); // 10
console.log(parseFloat("10.5")); // 10.5
```

c) Convert to Boolean

```
console.log(Boolean(1));   // true
console.log(Boolean(0));   // false
console.log(Boolean(""));  // false
console.log(Boolean("AI")); // true
```

---

6. Mini Exercise

```
let age = 20;
const name = "Umesh";

console.log(`Name: ${name}, Age: ${age}`);
```

Output:

```
Name: Umesh, Age: 20
```

---

7. Importance for AI & Web Development

- **Variables** → Store and manage data (datasets, inputs, results).
- **Data types** → Ensure correct handling of data (numbers for calculations, strings for text, booleans for conditions).
- **Type conversion** → Required when working with **APIs, databases, ML models**.
- In AI, handling data properly is **critical**.
- In JavaScript → used in **AI web apps** (TensorFlow.js, data visualization, dashboards).

---

8. Key Takeaways

- Variables = containers for data.
- Use `let` and `const` (avoid `var`).
- Data types are divided into:
- **Primitive** (Number, String, Boolean, Null, Undefined, Symbol, BigInt).
- **Non-Primitive** (Objects, Arrays, Functions).
- Always use `typeof` to check data type.
- Understand type conversion for handling different inputs.