1. WHY DO WE NEED DATA TYPES?

Definition:
Data types define what kind of value a variable can hold — for example, text, number, boolean, object, etc.
Why needed:
JavaScript needs to know how to store and operate on your data.
Without data types, it wouldn't know if 10 + 5 means adding numbers or joining strings.

2. DATA TYPES IN JAVASCRIPT

- Two main categories:
  - Primitive Data Types – simple, immutable values
  - Non-Primitive (Reference) Data Types – complex structures like objects, arrays, functions

PRIMITIVE DATA TYPES (7 TOTAL)

Type	Example	Description
Number	`let age = 25;`	Represents integers and floating-point numbers
String	`let name = "John";`	Represents text (sequence of characters)
Boolean	`let isOnline = true;`	Logical true or false
Undefined	`let city;`	Variable declared but not assigned a value
Null	`let car = null;`	Represents intentional empty value
Symbol	`let id = Symbol("id");`	Unique and immutable identifier
BigInt	`let big = 1234567890123456789012345678901234567890n;`	Large integers beyond Number limit

NON-PRIMITIVE DATA TYPES

Type	Example	Description
Object	`let person = { name: "Alice", age: 25 };`	Collection of key–value pairs
Array	`let colors = ["red", "green", "blue"];`	Ordered list of values
Function	`function greet() { console.log("Hi!"); }`	Reusable block of code

EXAMPLES OF DATA TYPES

```
let age = 25; // Number
let price = 99.99; // Number
let name = "John Doe"; // String
let isStudent = true; // Boolean
let phone; // Undefined
let car = null; // Null
let id = Symbol('userID'); // Symbol
let bigNum = 12345678901234567890n; // BigInt
let user = { name: "Alice", age: 22 }; // Object
let numbers = [1, 2, 3, 4]; // Array
function sayHello() { console.log("Hello JS!"); } // Function
```

3. VARIABLES IN JAVASCRIPT

Definition:
A variable is like a container that stores a value in memory.
Syntax:
`let variableName = value;`

TYPES OF VARIABLE DECLARATIONS

Keyword	Scope	Reassignable	Redeclarable	Example
`var`	Function-scoped	✅ Yes	✅ Yes	`var x = 10;`
`let`	Block-scoped	✅ Yes	❌ No	`let y = 20;`

Keyword	Scope	Reassignable	Redeclarable	Example
`const`	Block-scoped	❌ No	❌ No	`const z = 30;`

EXAMPLE OF VARIABLES

`var name = "John";` // function-scoped
`let age = 25;` // block-scoped
`const country = "USA";` // cannot change

`age = 30;` // ✅ allowed
// `country = "UK";` // ❌ error: const cannot be reassigned

✅ Use `let` when you know value might change.
✅ Use `const` for fixed values.
🚫 Avoid `var` (older JS) — use `let` or `const`.

4. OPERATORS IN JAVASCRIPT

Operators perform actions on variables and values.

ARITHMETIC OPERATORS

Operator	Description	Example	Result
`+`	Addition	`5 + 2`	7
`-`	Subtraction	`5 - 2`	3
`*`	Multiplication	`5 * 2`	10
`/`	Division	`10 / 2`	5
`%`	Modulus (remainder)	`10 % 3`	1
`**`	Exponentiation	`2 ** 3`	8
`++`	Increment	`let x=1; x++`	2
`--`	Decrement	`let y=2; y--`	1

COMPARISON OPERATORS

Operator	Description	Example	Result
`==`	Equal to (value only)	`5 == "5"`	true
`===`	Strict equal (value + type)	`5 === "5"`	false
`!=`	Not equal (value only)	`5 != 6`	true
`!==`	Strict not equal	`5 !== "5"`	true
`>`	Greater than	`10 > 5`	true
`<`	Less than	`5 < 10`	true
`>=`	Greater or equal	`10 >= 10`	true
`<=`	Less or equal	`8 <= 10`	true

LOGICAL OPERATORS

<table>
<tr><th>Operator</th><th>Description</th><th>Example</th><th>Result</th></tr>
<tr><td><code>&&</code></td><td>Logical AND</td><td><code>true && false</code></td><td>false</td></tr>
<tr><td><code>||</code></td><td>Logical OR</td><td><code>true || false</code></td><td>true</td></tr>
<tr><td><code>!</code></td><td>Logical NOT</td><td><code>!true</code></td><td>false</td></tr>
</table>

ASSIGNMENT OPERATORS

Operator	Example	Meaning
`=`	`x = 10`	Assigns value
`+=`	`x += 5`	Adds and assigns
`-=`	`x -= 5`	Subtracts and assigns
`*=`	`x *= 2`	Multiplies and assigns

Operator	Example	Meaning
`/=`	`x /= 2`	Divides and assigns
`%=`	`x %= 2`	Remainder and assigns

TYPE OPERATORS

Operator	Description	Example	Result
`typeof`	Returns data type	`typeof "hello"`	`"string"`
`instanceof`	Checks instance of object	`arr instanceof Array`	true

5. TYPE CONVERSION IN JAVASCRIPT

Converting values between data types is called Type Conversion.

🔹 **Implicit Conversion (Type Coercion)**

JavaScript automatically converts types.

```
console.log("5" + 2); // "52" (string + number → string)
console.log("5" - 2); // 3 (string - number → number)
console.log(true + 1); // 2 (true → 1)
```

🔹 **Explicit Conversion**

You convert manually.

➤ **String Conversion**

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

➤ **Number Conversion**

```
let str = "123";
console.log(Number(str)); // 123
```

➤ **Boolean Conversion**

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

6. BONUS: TYPEOF OPERATOR EXAMPLES

```
typeof 123; // "number"
typeof "Hello"; // "string"
typeof true; // "boolean"
typeof undefined; // "undefined"
typeof null; // "object" <-- known JS quirk
typeof {}; // "object"
typeof []; // "object"
typeof function(){}; // "function"
typeof 123n; // "bigint"
```

7. PRACTICE EXAMPLES (10+)

```
let x = 5;
let y = "10";

console.log(x + y); // "510" (string)
console.log(Number(y) + x); // 15 (number)
console.log(typeof x); // "number"
console.log(typeof y); // "string"

let isAdult = x > 3 && y == 10; // true
console.log(isAdult);

let name = "Alice";
console.log(`Hello ${name}, your age is ${x}`);

let count = 0;
count++; // increment
console.log(count); // 1
count--;
console.log(count); // 0

let result = null;
console.log(Boolean(result)); // false

let total = 10;
total += 5; // 15
console.log(total);
```