JavaScript Data Types - Complete Summary Notes

Based on Hindi JavaScript Tutorial Series

October 17, 2025

Comprehensive Data Types Summary with Interview Perspective

Contents

1 Introduction and Learning Approach

Interview Focused Learning
"This video provides a brush-up on data types from an interview perspective. Next video will cover memory-related concepts that will help you crack interviews. Obviously, it's not just about reading - you need to crack interviews and build good applications too."

Repository Access
"All code files are available on GitHub repository 'js-hindi-youtube' under Hitesh Choudhary. You can get all files from there."

2 Official Data Type Classification

Primitive vs Non-Primitive
JavaScript officially divides all data types into two main categories:

- **Primitive Types**: Call by value types
- **Non-Primitive Types**: Reference types

Memory Storage Basis
"The categorization is based on how data is stored in memory and how you can access your data. This is frequently asked in interviews: 'How is data stored in memory and accessed?' Based on this, data has two categorizations: Primitive and Non-Primitive."

3 Primitive Data Types

7 Primitive Categories
JavaScript has 7 primitive data types that are call-by-value:

1. **String**: Text data
2. **Number**: Numeric data (including decimals)
3. **Boolean**: true/false values
4. **null**: Empty value
5. **undefined**: Unassigned variable
6. **Symbol**: Unique identifiers
7. **BigInt**: Very large numbers

Call by Value Behavior
"All primitive types are call-by-value. When you copy them, you get the actual data, not the memory reference. Whatever changes you make happen in the copy."

3.1 String Type

String Examples:

```
const name = "hitesh";
const greeting = "Hello World";
```

3.2 Number Type

Number Examples:

```
const score = 100;
const scoreValue = 100.3; // Also number type
```

Number Includes Decimals
"There's no special type for decimal values like float. Number is number - whether it's decimal or any other format."

3.3 Boolean Type

Boolean Examples:

```
const isLoggedIn = false;
const isUserActive = true;
```

Real Usage
"You might track if user is logged into website or not, then you simply give it false. This becomes your simple boolean type."

3.4 null Type

null Example:

```
const outsideTemp = null;
```

null vs Empty
"null doesn't mean value is zero. null means completely empty. Remember our temperature example from server - when temperature didn't come, it wasn't zero, it was empty. So that was null."

3.5 undefined Type

undefined Examples:

```
let userEmail; // undefined
let user = undefined; // explicitly undefined
```

Undefined Definition
"You declared a variable but haven't decided what value to put in it yet. Memory space is declared but what value will come is unknown, so it's called undefined."

3.6 Symbol Type

Symbol Examples:

```
const id = Symbol('123');
const anotherId = Symbol('123');

console.log(id === anotherId); // false
```

Symbol Uniqueness
"Symbol is used to make any value unique. Even if you pass the same string value, the return type you get is a different data type called Symbol. Both values are not equal even though they look the same."

Symbol Usage Context
"You'll use Symbols extensively in advanced JavaScript frontend development. When you have components and need to identify individual components uniquely, that's when we wrap them in Symbol."

3.7 BigInt Type

BigInt Examples:

```
const bigNumber = 12345678901234567890123456789012345678901234567890n;
```

BigInt Declaration
"Just add 'n' at the end of the number and it automatically becomes BigInt. All your values normally get covered in Number type, but some scientific values or very large values that aren't being handled go in BigInt."

4 Reference Types (Non-Primitive)

Reference Types
Reference types (also called Non-Primitive types) are data types where memory reference can be directly allocated to you.

Three Main Reference Types

- **Arrays**: Collection of values
- **Objects**: Key-value pairs
- **Functions**: Executable code blocks

Master JavaScript Objects
"If you want to master JavaScript, master JavaScript Objects and Browser Web Events. These two topics will make you a JavaScript master."

4.1 Array Type

Array Examples:

```
const heroes = ["shaktiman", "naagraj", "doga"];
```

4.2 Object Type

Object Examples:

```
const myObj = {
    name: "hitesh",
    age: 22,
    isLoggedIn: true
};
```

Object Flexibility
"Objects can contain any data type - strings, numbers, booleans, functions, even other objects. This makes them very powerful for organizing data."

4.3 Function Type

Function Examples:

```
const myFunction = function() {
    console.log("Hello world");
};
```

Function as Variable
"In JavaScript, you can treat function like any other variable. You can store function definition in a variable just like you store other variables."

5 JavaScript: Dynamically Typed Language

Dynamic vs Static Typing

- **Statically Typed**: Variable types must be declared (C++, Java, TypeScript)
- **Dynamically Typed**: Variable types are determined at runtime (JavaScript)

No Type Declarations

"In JavaScript, we never define the type to the language. If I define a constant with value 100, I never told JavaScript that number type will come in this. Similarly, if I want to define false in it, I never told the language to give me boolean type value."

Research Assignment

"Tell me through research: Is JavaScript statically typed or dynamically typed? I've given you good hints. TypeScript requires colon and writing number, but in JavaScript we never define language types."

6 typeof Operator

Checking Data Types:

```
console.log(typeof bigNumber);          // "bigint"
console.log(typeof outsideTemp);        // "object" (special case)
console.log(typeof score);              // "number"
console.log(typeof myFunction);         // "function"
console.log(typeof id);                 // "symbol"
console.log(typeof undefinedValue);     // "undefined"
```

Special typeof Cases

- `typeof null` returns `"object"` - This is a famous interview question
- `typeof function` returns `"function"` (but it's actually function object)
- All non-primitive types return some form of object

typeof null Quirk

"typeof null returns 'object' - this is unique and obvious. This is frequently asked in interviews because it's unique and returns object."

7 Complete Data Types Code Example

Complete Data Types Demonstration:

```
// Primitive Types
const score = 100;                                  // Number
const scoreValue = 100.3;                           // Number
const isLoggedIn = false;                           // Boolean
const outsideTemp = null;                           // null
let userEmail;                                      // undefined
const id = Symbol('123');                           // Symbol
const anotherId = Symbol('123');                    // Symbol
const bigNumber = 123456789012345678900n;           // BigInt

// Reference Types
const heroes = ["shaktiman", "naagraj", "doga"];  // Array
const myObj = {                                     // Object
    name: "hitesh",
    age: 22,
    isLoggedIn: true
};
const myFunction = function() {                     // Function
    console.log("Hello world");
};

// Checking types
console.log(typeof score);          // "number"
console.log(typeof scoreValue);     // "number"
console.log(typeof isLoggedIn);     // "boolean"
console.log(typeof outsideTemp);    // "object" (special case)
console.log(typeof userEmail);      // "undefined"
console.log(typeof id);             // "symbol"
console.log(typeof bigNumber);      // "bigint"
console.log(typeof heroes);         // "object"
console.log(typeof myObj);          // "object"
console.log(typeof myFunction);     // "function"

// Symbol uniqueness check
console.log(id === anotherId);      // false
```

8 Memory Management: Primitive vs Reference

Call by Value vs Call by Reference

- **Primitive Types**: Call by Value (copy of value)
- **Reference Types**: Call by Reference (memory reference)

Key Difference

"When you copy primitive types, you get the actual data. When you copy reference types, you get the memory reference. This affects how changes propagate through your code."

9 Interview Preparation Summary

Data Types Return Types Summary

Data Type	typeof Returns
String	"string"
Number	"number"
Boolean	"boolean"
null	"object"
undefined	"undefined"
Symbol	"symbol"
BigInt	"bigint"
Array	"object"
Object	"object"
Function	"function"

Critical Interview Questions

1. How are data types categorized in JavaScript?
2. What's the difference between primitive and reference types?
3. Why does `typeof null` return "object"?
4. Is JavaScript statically or dynamically typed?
5. How do Symbols ensure uniqueness?
6. When would you use BigInt vs Number?

Assignment for Students

"Your assignment is to go to the comment section and write a nice comment about all these data types - what return type comes for each. This will help others who want to read. Please comment and push so this code file becomes available to you."

10 Key Takeaways

Data Types Mastery

- **7 Primitive Types**: String, Number, Boolean, null, undefined, Symbol, BigInt
- **3 Reference Types**: Array, Object, Function
- **Memory**: Primitives are call-by-value, References are call-by-reference
- **Typing**: JavaScript is dynamically typed
- **Checking**: Use `typeof` operator but know the special cases
- **Practice**: Essential for interviews and building applications

Final Note

"All code is available in the repository. Quickly check it out and do the assignment in comments. See you in the next video where we'll cover memory-related concepts!"