JavaScript Data Types - Complete Lecture Notes

Based on Hindi JavaScript Tutorial Series

October 16, 2025

Comprehensive notes covering JavaScript Data Types, Strict Mode, and Documentation

Contents

1 Introduction and Learning Philosophy

Learning Approach
The teacher emphasizes practical, keyboard-based learning rather than traditional pen-and-paper methods. Programming is best learned by writing code extensively on your keyboard.

Teacher's Strong Opinion
"If you're learning programming by writing everything on paper and PDFs, you'll never learn coding properly. The most important thing for learning coding is practice on your keyboard - writing as much code as possible."

Practical Learning Strategy

- Write notes directly in code files
- Practice extensively on keyboard
- Avoid excessive pen-and-paper methods
- Focus on code readability and future-proofing

2 JavaScript Strict Mode

What is Strict Mode?
Strict mode (`"use strict"`) is a way to opt into a restricted variant of JavaScript that helps catch common coding errors and "unsafe" actions.

Strict Mode Declaration:

```
"use strict";
// All code here will be treated as newer JavaScript version
```

Historical Context
"JavaScript from 8-10 years ago was different - it didn't have classes, modules, arrow functions. The TC39 organization didn't want old code to break, so they introduced strict mode to treat all code as newer version standards."

Modern JavaScript Practice
"Nowadays, JavaScript automatically runs in strict mode, so you don't necessarily need to write `"use strict"`. However, it's good practice to explicitly declare that you're using modern syntax standards."

3 Environment Differences: Node.js vs Browser

Alert Statement Behavior

The `alert()` function behaves differently in Node.js vs Browser environments:

- **Browser**: Shows popup dialog
- **Node.js**: Not available by default (causes error)

Alert Examples:

```
// This works in browser but causes error in Node.js
alert("Hello");
alert(3 + 3); // Shows 6 in browser popup

// In Node.js, we use console.log instead
console.log(3 + 3); // Outputs 6 to terminal
```

Common Environment Confusion

Many beginners get confused when `alert()` doesn't work in Node.js. Remember: JavaScript engine is embedded in browsers, while Node.js provides a different runtime environment.

4 Code Readability and Best Practices

Code Readability Priority

"Code execution working doesn't mean everything is fine. Code should also be readable and future-proof. Otherwise, what's the use of experience?"

Good vs Bad Code Examples

```
//     BAD: Poor readability
console.log(3+3)console.log("hitesh");

//     GOOD: Readable and maintainable
console.log(3 + 3);
console.log("hitesh");

//     BAD: Unnecessary complexity
console.log(3
+
3);

//     GOOD: Clean and understandable
console.log(3 + 3);
```

Semicolon Usage
"While JavaScript automatically inserts semicolons, it's better practice to include them explicitly for code clarity and to avoid unexpected behavior."

5 JavaScript Documentation Sources

Official Documentation

- **MDN Web Docs**: Developer-friendly documentation
- **TC39**: Official ECMAScript specification
- **ECMA-262**: Original JavaScript standards

Historical Standards Issue
"In early days, JavaScript had different standards across browsers. The ECMA Script organization was formed to define unified standards - specifying how loops should output, how input should be taken, etc."

Learning Documentation
"We'll mostly use MDN documentation as it's easier to understand. The TC39 specifications are more for browser developers and engine writers, but we'll refer to them for important concepts."

6 JavaScript Data Types

6.1 Primitive Data Types

Primitive Data Types
Primitive data types are the basic building blocks in JavaScript. They are immutable and stored directly in memory.

Data Type Examples:

```
// String
let name = "hitesh";

// Number
let age = 18;

// Boolean
let isLoggedIn = true;
let hasCreditCard = false;

// Undefined
let accountState; // undefined

// Null
let temperature = null;

// Symbol
let id = Symbol('id');

// BigInt
let bigNumber = 1234567890123456789012345678901234567890n;
```

6.2 Number Type

Number Range and BigInt

- **Number**: Standard numeric type with range up to 2^{53} (or 2^{52})
- **BigInt**: For very large numbers beyond standard number range

When to Use BigInt

"You won't use BigInt often in regular programming journey, but it's essential for:

- Stock market/trading applications
- Very large websites like Reddit
- Applications handling extremely large numbers

"

6.3 Boolean Type

Boolean Values

Boolean represents only two values: `true` or `false`. It's used for yes/no decisions in programming.

Boolean Use Cases

- User logged in or not
- Credit card details available or not
- Server response received or not
- Any binary decision-making scenario

Boolean Simplicity

”Boolean is very simple - it’s just about ’yes’ or ’no’. JavaScript isn’t complicated, and boolean is one of the most useful data types we’ll work with extensively.”

6.4 Null vs Undefined

Null and Undefined Difference

- **Undefined**: Variable declared but no value assigned
- **Null**: Intentional empty value representation

Practical Example

”Suppose you request temperature from a server. If server sends 0, that’s an actual temperature. But if there’s a server issue and it can’t send temperature, you’d prefer null (empty) rather than 0 (which is a valid temperature).”

Null vs Undefined Examples

```
// Undefined - value not assigned
let accountState;
console.log(accountState); // undefined

// Null - intentional empty value
let temperature = null; // Server couldn't provide temperature
let userResponse = null; // User didn't provide input
```

6.5 Symbol Type

Symbol Data Type

Symbol is a unique and immutable primitive value that may be used as the key of an Object property. It’s mainly used for creating unique identifiers.

Symbol Usage Context

”You’ll see Symbols extensively when we talk about React on this channel. When you create many components and need identification for individual components, or when you need to create unique keys, that’s when we use Symbols heavily.”

7 typeof Operator

Using typeof Operator:

```
// Checking data types
console.log(typeof "hitesh");    // "string"
console.log(typeof age);         // "number"
console.log(typeof isLoggedIn);  // "boolean"
console.log(typeof undefined);   // "undefined"

// Interesting cases
console.log(typeof null);        // "object" (historical reason)
```

The null typeof Quirk

"When you check typeof null, it returns 'object'. This is considered a historical mistake or language quirk in JavaScript. Many people call it a language error, and it's a common interview question."

Common Confusion

- `typeof undefined` → `"undefined"` (correct)
- `typeof null` → `"object"` (historical quirk)
- This causes many errors in JavaScript programs

8 Objects and Future Topics

Object Data Type

Objects are non-primitive data types that can store collections of data and more complex entities. We'll dedicate 5-6 videos to thoroughly understand objects.

Coming Attractions

"In future videos, we'll explore:

- Different ways to define strings and numbers
- Practical usage scenarios for each data type
- Object-oriented programming with objects
- Advanced data type manipulations

"

9 Assignment and Practice

Homework Assignment
"Your assignment is to visit the ECMAScript specification. You don't need to read everything, but at least look at the specifications while the videos aren't coming. We'll mostly spend our time on MDN documentation."

Repository Access
"All exercise files are available on GitHub repository. Please star the repository and fork it so you have your own copy. The files will be available in the 01-basics folder."

10 Complete Code Example

Complete Data Types Example:

```javascript
"use strict";

// Different data types examples
let name = "hitesh";          // String
let age = 18;                 // Number
let isLoggedIn = true;        // Boolean
let accountState;             // Undefined
let temperature = null;       // Null
let id = Symbol('id');        // Symbol
let bigNumber = 1234567890123456789012345678901234567890n; // BigInt

// Using typeof operator
console.log(typeof name);        // "string"
console.log(typeof age);         // "number"
console.log(typeof isLoggedIn);  // "boolean"
console.log(typeof accountState); // "undefined"
console.log(typeof temperature);  // "object" (historical quirk)
console.log(typeof id);          // "symbol"
console.log(typeof bigNumber);   // "bigint"

// Console output methods
console.log([name, age, isLoggedIn, accountState, temperature]);
```

11 Key Takeaways Summary

Data Types Summary

- **String**: Text data
- **Number**: Numeric data
- **Boolean**: True/False values
- **Undefined**: Unassigned variables
- **Null**: Intentional empty values
- **Symbol**: Unique identifiers
- **BigInt**: Very large numbers
- **Object**: Complex data structures (coming soon)

Critical Points to Remember

1. Use strict mode for modern JavaScript features
2. Understand environment differences (Node.js vs Browser)
3. Prioritize code readability over clever tricks
4. Know the difference between null and undefined
5. Remember typeof null returns "object" (historical quirk)
6. Practice extensively on keyboard, not just pen-and-paper

Learning Mindset
"Today's content is enough for now. We'll explore more interesting topics in upcoming videos. Remember: the key to learning JavaScript is consistent practice and understanding the 'why' behind each concept."