JavaScript Variables and Constants - Complete Lecture Notes

Based on Hindi JavaScript Tutorial Series

October 16, 2025

Comprehensive notes covering JavaScript Variables, Constants, and Declaration Methods

Contents

1 Introduction and Mindset

Learning with Purpose

The teacher emphasizes that learning JavaScript should not be just for coding syntax, but with the goal of building real products like e-commerce websites, social media applications, or mobile apps.

Teacher's Important Message

"People who learn programming just to get a job don't last long in the industry. Those who stay are the ones who want to build something - create products. That's what makes history."

Practical Goal Setting

Set a concrete goal for this series: "I will build an e-commerce website" or "I will create a social media web application." This practical approach makes learning variables, constants, and loops meaningful rather than just theoretical.

2 Real-World Context: User Registration

Practical Application Scenario

We're learning variables and constants in the context of building user registration for an e-commerce website. We need to store user information like:

- Account ID
- Email
- Password
- City
- State

E-commerce Registration Requirements

- Unique ID for each user
- Email address for communication
- Secure password
- Location information (city, state)
- Temporary storage in memory before database

3 Variables and Constants Fundamentals

What are Variables and Constants?

- **Constants**: Values that cannot be changed once declared
- **Variables**: Values that might change later in the program

Both require memory space to store information temporarily.

Memory Space Analogy

"Think of variables and constants as reserved memory spaces where you can temporarily store information. The entire game of programming is about storing this information optimally, retrieving it, and manipulating it efficiently."

4 JavaScript Declaration Methods

4.1 Constants: const keyword

Constant Declaration Syntax:

```
const accountId = 144553;
const accountEmail = "hitesh@chaicode.com";
```

const Behavior

"Once you assign a value to a constant, it gets locked there. You cannot change it. Many times in programming, intentionally locking values is better than making everything changeable with let."

const Investigation Results

```
const accountId = 144553;
accountId = 123456; // This will cause an error!
```

Output: Assignment to constant variable is not allowed

4.2 Variables: let keyword

Variable Declaration with let:

```
let accountEmail = "hitesh@chaicode.com";
let accountPassword = "12345";
let accountCity = "Jaipur";
```

let Modification Examples

```
let accountEmail = "hitesh@chaicode.com";
accountEmail = "contact@hitesh.com"; // This works!

let accountCity = "Jaipur";
accountCity = "Bangalore"; // This also works!
```

4.3 Console Output Methods

`Individual Console Output:`

```
console.log(accountId);
console.log(accountEmail);
console.log(accountPassword);
console.log(accountCity);
```

`Table Output Method:`

```
console.log([accountId, accountEmail, accountPassword, accountCity])
    ;
```

Console.table() Advantage
"Using console.table() displays all variables in a nice tabular structure, making it easier to compare values and see everything at once rather than individual console.log statements."

5 The var vs let Controversy

Historical Context of var
JavaScript originally had only `var` for variable declaration. However, it had scope-related issues that caused problems in larger applications.

The Problem with var
Scope Issues: var doesn't respect block scope (curly braces {}). If multiple programmers used the same variable name in different files or conditions, they would accidentally modify each other's variables.

```
// Problem scenario with var
var accountPassword = "12345";

// In another file or condition
for (var i = 0; i < 5; i++) {
    var accountPassword = "newPassword"; // This changes the
        original!
}
```

Modern JavaScript Practice

"From now on, we will use only two things: **const** for constants and **let** for variables. Forget that var even existed!"

```
//     Modern approach
const accountId = 144553;  // For values that won't change
let accountEmail = "hitesh@chaicode.com";  // For values that might
   change
```

Multi-line Comments

```
/*
Please please never use var
Because of issues in block scope and functional scope
*/
```

This type of comment uses /* */ syntax and can span multiple lines.

6 Undefined Variables and Declaration

`Declaring Without Initializing:`

```
let accountState;  // No value assigned
console.log(accountState);  // Output: undefined
```

Undefined Behavior

"When you declare a variable without assigning any value, JavaScript considers it as `undefined`. This means the variable exists but has no value assigned to it yet."

Real Use Case for Undefined Variables

```
// Client wants account state but default is unknown
let accountState;  // Will assign value later based on user input

console.log(accountState);  // Output: undefined
```

7 Variable Naming Conventions

Naming Philosophy

There are many opinions about variable naming conventions:

- `accountId` (camelCase - most common in JavaScript)
- `account_id` (snake_case)
- `AccountID` (PascalCase)

Teacher's Simple Rule

"My simple rule for variable names: **The name should be easily readable**. That's the only goal. Don't worry about what they do in Java or C++. What matters is that when another programmer reads your code, they should easily understand what the variable represents."

8 Investigation-Based Learning Approach

Learning Methodology

The teacher emphasizes learning through investigation rather than just following examples:

- Try different approaches
- Make changes and observe results
- Understand why things work the way they do
- Read documentation to investigate behavior

Investigation Process Demonstrated

1. Declare constants and try to change them → Observe error
2. Declare variables with let and modify them → Observe success
3. Compare const vs let behavior
4. Investigate var issues through historical context
5. Test undefined variable behavior

9 Complete Code Example

Complete Variables and Constants Example:

```javascript
// Constants - cannot be changed
const accountId = 144553;

// Variables - can be changed
let accountEmail = "hitesh@chaicode.com";
let accountPassword = "12345";
let accountCity = "Jaipur";

// Variable without initialization
let accountState;

// Modifying variables
accountEmail = "contact@hitesh.com";
accountPassword = "212121";
accountCity = "Bangalore";

// Output methods
console.log(accountId);
console.log([accountId, accountEmail, accountPassword, accountCity])
    ;
console.log(accountState); // Output: undefined

/*
Important Note:
Always use const and let
Never use var due to scope issues
*/
```

10 Key Takeaways Summary

Summary of JavaScript Declaration Methods

- **const**: For values that should never change
- **let**: For values that might change
- **var**: Avoid completely due to scope issues
- **No keyword**: Possible but not recommended

Critical Rules to Remember

1. Use `const` by default for values that shouldn't change
2. Use `let` only when you know the value needs to change
3. Never use `var` in modern JavaScript
4. Variable names should be easily readable and meaningful
5. Undefined variables have no value assigned

Learning Philosophy

"JavaScript is very simple if you learn it through investigation. We'll continue learning this way - through investigation and project-oriented approach, because we need to learn with experience plus we need to know what we're building."