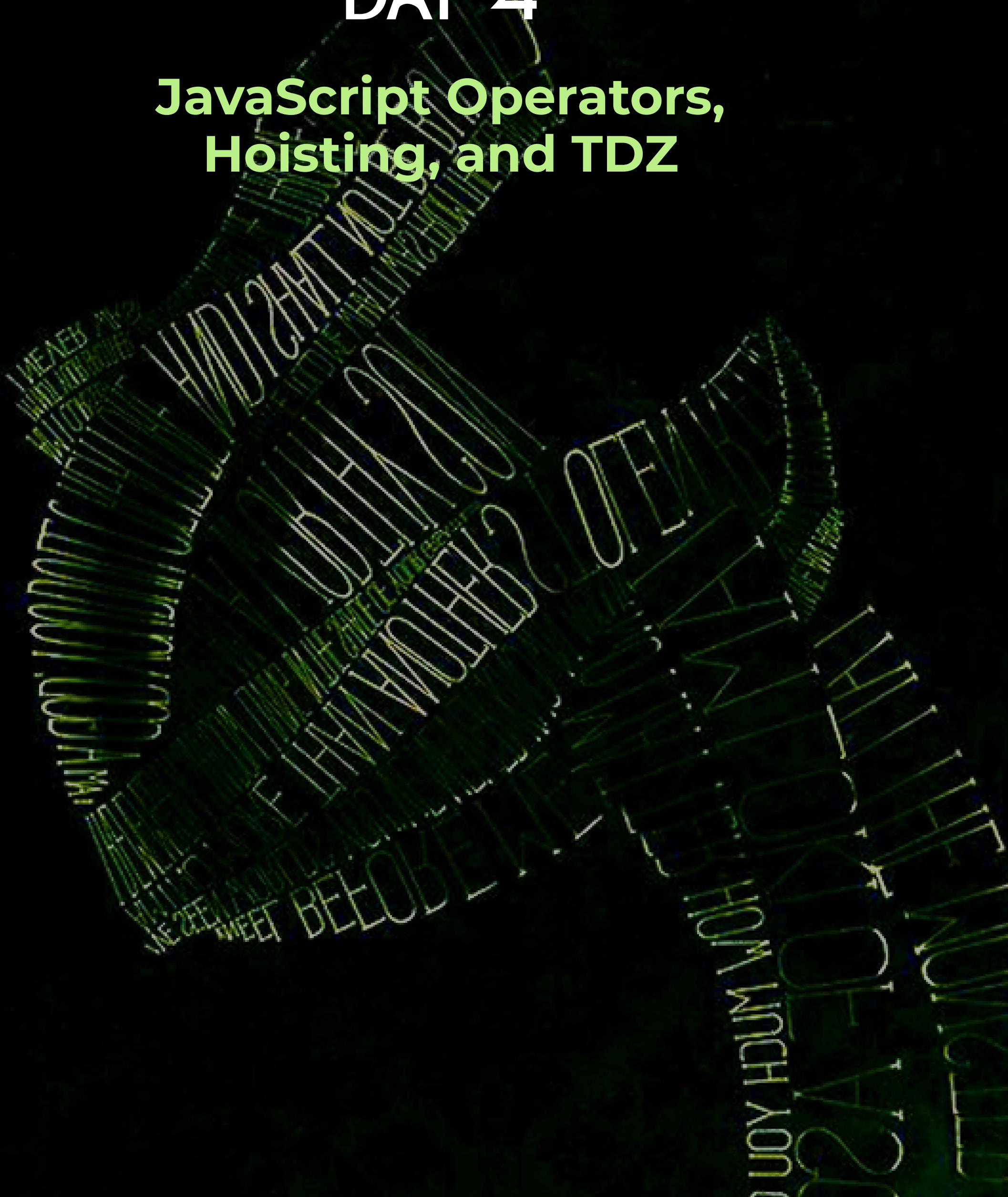
Sheryians Coding School

Live Cohort

DAY-4

JavaScript Operators, Hoisting, and TDZ



JavaScript Operators, Hoisting, and TDZ

This guide explains all major **operators** in JavaScript and introduces two important concepts — **Hoisting** and the **Temporal Dead Zone (TDZ)**.

- ## JS Operators

Arithmetic Operators

Used to perform mathematical operations.

Operator	Description	Example	Output
+	Addition	5 + 2	7
-	Subtraction	5 - 2	3
*	Multiplication	5 * 2	10
/	Division	10 / 2	5
%	Modulus (Remainder)	5 % 2	1
**	Exponentiation	2 ** 3	8

JavaScript Operators, Hoisting, and TDZ

Assignment Operators

Used to assign values to variables.

Operator	Description	Example	Output
=	Assign	x = 10	10
+=	Add and assign	x += 5	x = x + 5
-=	Subtract and assign	x -= 5	x = x - 5
*=	Multiply and assign	x *= 5	x = x * 5
/=	Divide and assign	x /= 5	x = x / 5
%=	Modulus and assign	x %= 5	x = x % 5

JavaScript Operators, Hoisting, and TDZ

Comparison Operators

Used to compare two values and return a boolean.

Operator	Description	Example	Output
==	Equal to	5 == '5'	true
===	Strict equal (type + value)	5 === '5'	false
!=	Not equal	5 != 6	true
!==	Strict not equal	5 !== '5'	true
>	Greater than	6 > 5	true
<	Less than	5 < 6	true
>=	Greater or equal	5 >= 5	true
<=	Less or equal	4 <= 5	true

Logical Operators

Used to combine conditional statements.

Operator	Description	Example	Output
&&	AND	(5 > 3 && 6 > 4)	true
||	OR	(5 > 10 || 6 > 4)	true
!	NOT	!(5 > 3)	false

JavaScript Operators, Hoisting, and TDZ

Ternary Operator

A shorthand for `if...else`.

```
condition ? doThis : doThat;
```

Example:

```
let age = 20;
let message = age >= 18 ? "Adult" : "Minor";
console.log(message); // "Adult"
```

Type Checking Operators

Used to check the data type or object type.

Operator	Description	Example	Output
typeof	Returns the data type	typeof "Hello"	"string
instance of	Checks object type	[] instanceof Array	true

JavaScript Operators, Hoisting, and TDZ

String Operator

Only one — `+` — is used for concatenation.

🧩 Example:

```javascript
let name = "Ritik";
let greet = "Hello " + name;
console.log(greet); // Hello Ritik
```

Spread / Rest Operator

Very important in modern JS.

🧩 Example:

```javascript
// Spread
let arr = [1, 2, 3];
let copy = [...arr];
console.log(copy); // [1, 2, 3]

// Rest
function sum(...numbers) {
  return numbers.reduce((a, b) => a + b);
}
console.log(sum(1, 2, 3)); // 6
```

Nullish Coalescing Operator

Provides a fallback **only** for `null` or `undefined`.

🧩 Example:

```
let name = null;
let displayName = name ?? "Guest";
console.log(displayName); // Guest
```

🔟 Optional Chaining

Safely access nested properties without throwing an error.

🧩 Example:

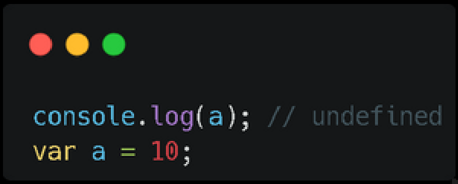
```
let user = { profile: { name: "Ritik" } };
console.log(user?.profile?.name); // Ritik
console.log(user?.address?.city); // undefined (no error)
```

JavaScript Operators, Hoisting, and TDZ

• Hoisting in JavaScript

Hoisting is JavaScript's behavior of moving variable and function declarations to the top of their scope **before code execution**.

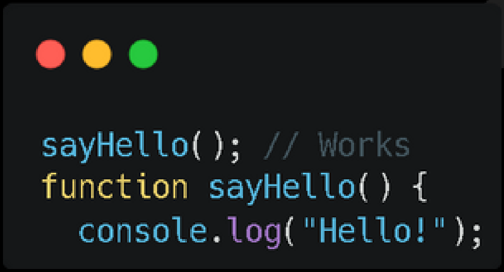
Example 1: Variable Hoisting



```javascript
console.log(a); // undefined
var a = 10;
```

👉 The declaration `var a` is hoisted, but not the initialization.

Example 2: Function Hoisting



```javascript
sayHello(); // Works
function sayHello() {
  console.log("Hello!");
```

👉 Function declarations are **fully hoisted**, meaning you can call them before they are defined.

Example 3: `let` and `const` are not hoisted the same way

```
console.log(x); // ReferenceError ❌
let x = 5;
```

Variables declared with `let` and `const` are **hoisted** but not **initialized** — this leads to the **Temporal Dead Zone (TDZ).**

⚡ Temporal Dead Zone (TDZ)

The ***Temporal Dead Zone** is the period between the start of a scope and the actual declaration of a `let` or `const` variable.

Any attempt to access the variable before declaration results in a **ReferenceError**.

🧩 Example:

```
console.log(message); // ❌ ReferenceError
let message = "Hello TDZ!";
```

🧩 Explanation:

The variable `message` is hoisted but remains **uninitialized** until the `let` statement executes — during this time, it exists in the **TDZ**

✅ Key Takeaways

- `var` is **function-scoped** and hoisted with `undefined` initialization.
- `let` and `const` are **block-scoped** and hoisted **without initialization**, causing the **TDZ**.
- Accessing variables in the TDZ throws a `ReferenceError`.
- Function declarations are **fully hoisted**, but function expressions are not.

Author: Ritik Rajput
Created with ❤️ using JavaScript & Markdown