JavaScript Hoisting – Explained Simply 🚀

👉 We've all heard about hoisting, but what does it actually mean? Let's break it down.

What is Hoisting?

- Hoisting means JavaScript moves declarations to the top of the scope before execution.

- In simple terms: You can use variables or functions before you declare them.

Variable Hoisting

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

👉 **No error, because var is hoisted as undefined.**

- var → hoisted with undefined
- let & const → hoisted but stay in the Temporal Dead Zone (TDZ)

Function Hoisting

✅ Function Declarations → hoisted

❌ Function Expressions / Arrow → not hoisted

```js
greet(); // ✅ Works

function greet() {
  console.log("Hello!");
}
```

👉 Function declarations are fully hoisted.

But:

```js
greet(); // ❌ Error
const greet = () => {};
```

👉 Function expressions & arrow functions are not hoisted.

Why Does It Matter?

Hoisting can cause hidden bugs & interview questions!

- Makes code behavior easier to understand
- Helps avoid unexpected undefined bug
- Frequently asked in JavaScript interviews

Key Takeaways

Hoisting =

Declarations are moved, but initializations are not.

💡 Use let & const instead of var

💡 Use function declarations if you rely on hoisting

👉 **Have you ever faced a tricky bug because of hoisting?**

Share in the comments!