**Understanding "use strict" in JavaScript 🚀**

**1️. What is "use strict"?**

"use strict" is a special directive in JavaScript that **enables strict mode**, which helps catch common coding mistakes and prevents unsafe actions.

🔹 **Why use strict mode?**  
✅ Helps find **silent errors** that JavaScript normally ignores.  
✅ Prevents the use of **undeclared variables**.  
✅ Makes debugging **easier**.  
✅ Improves performance by enabling **optimizations** in modern JavaScript engines.

**2️. How to Enable Strict Mode**

There are **two ways** to use strict mode:

**1. Strict Mode for the Whole Script**

"use strict"; // Enable strict mode for the entire file

let name = "John"; // ✅ Works fine

console.log(name);

undeclaredVar = 10; // ❌ Error: undeclaredVar is not defined

**🔍 Without "use strict", JavaScript would allow undeclaredVar = 10; without an error!**

**2. Strict Mode for a Specific Function**

function test() {

"use strict";

let x = 5; // ✅ Works fine

undeclaredVar = 10; // ❌ Error: undeclaredVar is not defined

}

test();

🔹 This ensures **only this function runs in strict mode**, while the rest of the script is in normal mode.

**3️. Common Errors Prevented by Strict Mode**

**1. Using Undeclared Variables**

"use strict";

x = 10; // ❌ ReferenceError: x is not defined

🔹 **Fix:** Always declare variables with let, const, or var.

"use strict";

let x = 10; // ✅ Correct

console.log(x);

**2. Prevents Accidental Global Variables**

"use strict";

function myFunction() {

testVar = "Hello"; // ❌ ReferenceError: testVar is not defined

}

myFunction();

🔹 **Fix:** Use let or const.

"use strict";

function myFunction() {

let testVar = "Hello"; // ✅ Correct

}

**3. Disallows Duplicating Function Parameter Names**

"use strict";

function sum(a, a) { // ❌ SyntaxError: Duplicate parameter name not allowed in strict mode

return a + a;

}

🔹 **Fix:** Use unique parameter names.

"use strict";

function sum(a, b) { // ✅ Correct

return a + b;

}

**4. Prevents Assignment to Read-Only Properties**

"use strict";

const obj = Object.freeze({ name: "John" });

obj.name = "Mike"; // ❌ TypeError: Cannot assign to read-only property 'name'

🔹 **Fix:** Avoid modifying frozen objects.

**5. Prevents this from Being undefined in Functions**

"use strict";

function showThis() {

console.log(this); // ❌ Outputs `undefined` instead of global object

}

showThis();

🔹 **Fix:** Use arrow functions or bind this explicitly.

const showThis = () => {

console.log(this); // ✅ Works correctly

};

showThis();

**4️. Should You Always Use Strict Mode?**

✅ **Yes!** It helps prevent bugs and improves performance.  
🔹 **Best practice:** Always add "use strict"; at the **top of your JavaScript files**.

**5️. Summary**

| **Feature** | **Without Strict Mode 😴** | **With Strict Mode 🚀** |
| --- | --- | --- |
| Undeclared variables | Allowed | ❌ Error |
| Duplicate function parameters | Allowed | ❌ Error |
| this in functions | Global object | undefined |
| Writing to read-only properties | Allowed | ❌ Error |