02-Node-Module-System/01-global-objects.md

🌍 Global Objects in Node.js

🎯 What are Global Objects?

Global objects are available everywhere in your application - in all files without needing to import them.

Example: `console`

```
console.log('Hello World');
```

The `console` object is **global** - you can use it anywhere!

🔧 VS Code IntelliSense

💡 **Tip:** VS Code will automatically show you available global objects and their methods as you type!

Start typing `console.` and watch IntelliSense suggest methods like:

- `console.log()`
- `console.error()`
- `console.warn()`
- `console.table()`

⏰ Common Global Functions

Timer Functions

These work in **both Browser and Node.js**:

```
// Call function after delay
setTimeout(() => {
    console.log('Executed after 2 seconds');
}, 2000);

// Stop a timeout
const timer = setTimeout(() => {}, 2000);
clearTimeout(timer);

// Repeatedly call function
const interval = setInterval(() => {
    console.log('Executed every second');
}, 1000);

// Stop the interval
clearInterval(interval);
```

> 📖 **Full list of globals:** nodejs.org/api/globals.html

🌐 Browser: The `window` Object

In the Browser

Everything global is attached to the `window` object:

```
// These are equivalent in the browser:
console.log('Hello');
window.console.log('Hello');

setTimeout(() => {}, 1000);
window.setTimeout(() => {}, 1000);
```

Variables in Browser

```
var message = 'test';
console.log(window.message); // 'test'
```

Variables declared with `var` are added to the `window` object!

🟢 Node.js: The `global` Object

No `window` in Node.js!

Node.js has a `global` object instead:

```
// These are equivalent in Node.js:
console.log('Hello');
global.console.log('Hello');

setTimeout(() => {}, 1000);
global.setTimeout(() => {}, 1000);
```

⚠️ Important Difference: Variable Scope

In Node.js, variables are NOT added to `global`!

Create a file `test.js`:

```
var message = 'test';
console.log(global.message);
```

Run it:

```
milan@first-app~\ node test.js
undefined
milan@first-app~\
```

Why `undefined`?

In Node.js, **variables are scoped to their file** (module), not added to the global object!

🔒 Scope in Node.js

File-Level Scope

```
// app.js
var message = 'Hello';
console.log(message); // ✅ Works
```

```
// otherFile.js
```

```
console.log(message); // ❌ ReferenceError: message is not defined
```

Variables are limited to the file where they are declared!

🎨 JavaScript Global Scope (Browser)

Problem with Browser Global Scope

In traditional JavaScript (browser), functions and variables are added to the global scope:

```
// file1.js
function sayHello() {
    console.log('Hello from file1');
}

// file2.js
function sayHello() {  // ⚠️ Overwrites the first one!
    console.log('Hello from file2');
}
```

The Problem

- Different files with **identical declarations** will override each other
- Hard to maintain large applications
- Name conflicts are common

🧩 The Solution: Modularity

Why We Need Modules

Small building blocks = Modules

Each file becomes a module with its own scope:

- ✅ Variables are private by default
- ✅ No naming conflicts
- ✅ Explicit imports/exports
- ✅ Better code organization

📦 What is a Module?

In Node.js

- Each file is a **module**
- Variables declared in a file have **scope of that file only**
- Similar to **private** in Object-Oriented Programming
- Must be **explicitly exported** to be used elsewhere
- Every Node app has at least **one main module**

🔑 Key Takeaways

Concept	Browser	Node.js
Global Object	`window`	`global`
Variables	Added to `window`	NOT added to `global`
Scope	Global by default	File (module) scoped

Functions	Global if not in module	Private to file

💡 Best Practice

Module-Based Architecture

✅ **DO**: Use modules to organize your code
✅ **DO**: Keep variables private unless needed elsewhere
✅ **DO**: Explicitly export what's needed
❌ **DON'T**: Rely on global scope
❌ **DON'T**: Add everything to the global object

🔜 What's Next?

Now that you understand global objects and scope, let's dive deeper into **how the module system works**!

🏠 Course Home | 📘 Chapter 2 Home

← Previous: Chapter 2 Intro | Next: Module System Basics →