02-Node-Module-System/04-builtin-modules.md

🔧 Built-in Node Modules

📚 Node.js Core Modules

Powerful Modules Ready to Use

No installation required!

📖 Documentation

Official Docs

- 📘 Node.js v25.x API
- 📘 Node.js Latest API

Note: Not everything is a module! Some are objects like `Console`.

🗂️ Important Built-in Modules

Module	Purpose
path	Work with file/directory paths
fs	File system operations

os	Operating system information
http	Create web servers
events	Event handling
stream	Work with streaming data

🛤️ Path Module

Working with File Paths

```
const path = require("path"); // No ./ for built-in modules!

const pathObj = path.parse(__filename);
console.log(pathObj);
```

Output:

```
{
  root: '/',
  dir: '/Users/milan/Dev/first-app/les2',
  base: 'pathvb.js',
  ext: '.js',
  name: 'pathvb'
}
```

Common Path Methods

```
path.join("/users", "milan", "file.txt"); // Join paths
path.resolve("file.txt"); // Absolute path
path.basename("/users/milan/file.txt"); // 'file.txt'
path.dirname("/users/milan/file.txt"); // '/users/milan'
path.extname("/users/milan/file.txt"); // '.txt'
```

📖 Path Module Docs

💻 OS Module

System Information

⚠️ Only works in Node.js, not in browsers!

```
const os = require("os");

const totalMemory = os.totalmem();
const freeMemory = os.freemem();
const osType = os.type();
const uptime = os.uptime();

console.log(`Total Memory: ${totalMemory}`);
console.log(`Free Memory: ${freeMemory}`);
console.log(`OS Type: ${osType}`);
```

Output:

```
milan@les2~ node osvb.js
Total Memory: 8589934592
OS Type: Darwin
milan@les2~
```

ES6 Template Strings

Note the backticks!

```
// Old way
console.log("OS type is: " + osType);

// ES6 way (template literal)
console.log(`OS type is: ${osType}`);
```

📖 OS Module Docs

📁 File System Module

⚠️ Synchronous vs Asynchronous

The File System module has two versions of most methods:

❌ Synchronous (Blocking)	✅ Asynchronous (Non-blocking)
`fs.readdirSync()`	`fs.readdir()`
Blocks thread	Non-blocking
Returns data directly	Uses callback

Synchronous Example (Don't Use!)

```
const fs = require("fs");
const files = fs.readdirSync("./");
```

```
console.log(files);
```

Output:

```
['app.js', 'fsvb.js', 'logger.js', 'osvb.js', 'pathvb.js']
```

✅ File System: Async (Always Prefer This!)

Asynchronous Example

```
const fs = require("fs");

fs.readdir("./", function (err, files) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Result", files);
  }
});
```

How It Works

1. **Two parameters:** Path and callback function
2. **Callback has two parameters:** `err` and `result`
3. **Only one contains data:** Either error OR result (not both)

Output:

```
milan@les2 ~ node fsvb.js
Result ['app.js', 'fsvb.js', 'logger.js', 'osvb.js', 'pathvb.js']
```

```
milan@les2~\
```

📖 File System Docs

🎯 Best Practices

DO ✅

- **Always use async methods** for file operations
- Check for errors first in callbacks
- Use `path` module for cross-platform paths
- Use template literals (backticks) for strings

DON'T ❌

- Don't use synchronous methods (blocks the thread)
- Don't ignore error handling
- Don't hardcode file paths
- Don't use + for string concatenation when template literals work better

---

🧪 Try It Yourself

Exercise

1. Use `os` module to get system info
2. Use `path` module to parse a file path
3. Use `fs` (async) to read files in current directory
4. Display results using template literals

🏠 Course Home | 📘 Chapter 2 Home

← Previous: Creating Modules | Next: Events →