CommonJS Modules (CJS)

CommonJS is a module system used in Node.js to handle JavaScript modules. Here's how it works:

- **`require`** : To import modules in CommonJS, you use the `require()` function.
- **`module.exports`** : To export something from a module, you use `module.exports` .

Example of CommonJS:

```
// math.js
module.exports.add = function (a, b) {
  return a + b;
}

// main.js
const math = require('./math');  // Importing the math module
console.log(math.add(2, 3));     // Output: 5
```

Key Points:

- Modules are loaded synchronously (one after the other).
- It's the default module system in Node.js (but you can use ES modules now as well).

ECMAScript Modules (ESM)

ECMAScript modules (also called ES modules or ESM) are the standard module system introduced in JavaScript (starting from ES6/ES2015). ES modules are natively supported in modern JavaScript environments like browsers and Node.js.

- **`import`** : Used to import a module.
- **`export`** : Used to export functions, objects, or variables from a module.

Example of ECMAScript modules:

```
// math.js
export function add(a, b) {
  return a + b;
}

// main.js
import { add } from './math.js';  // Importing the math module
console.log(add(2, 3));           // Output: 5
```

Key Points:

- ES modules are loaded asynchronously (they don't block the rest of the code).
- Used in both browser environments and Node.js (with the `.mjs` extension or `type: module` in `package.json`).

CommonJS vs ECMAScript Modules

Feature	CommonJS (CJS)	ECMAScript Modules (ESM)
Syntax	`require()` to import, `module.exports` to export	`import` to import, `export` to export
Load Type	Synchronous (one after the other)	Asynchronous (non-blocking)
Default in Node.js	Yes (traditional default)	Supported with `.mjs` extension or `type: "module"` in `package.json`
Use Case	Mostly in Node.js	In both browsers and Node.js
Exports	Exports a value as a whole (`module.exports = ...`)	Can export multiple values (named exports)
Support	Supported in older versions of Node.js	Introduced in ES6 (ES2015) and widely supported today

Key Differences:

- **Syntax**: CJS uses `require()` and `module.exports`, while ESM uses `import` and `export`.
- **Asynchronous vs Synchronous**: ESM is asynchronous and doesn't block other code, while CJS loads modules synchronously.
- **Cross-environment**: ESM works in both browsers and Node.js, while CJS is mainly for Node.js. Browsers don't natively support CJS without tools like `webpack`.

When to Use Which?

- **CommonJS (CJS)** is still widely used in many Node.js projects and libraries. If you're working with older projects or packages, you might still use CJS.
- **ECMAScript Modules (ESM)** are the future standard for JavaScript modules, and if you're working with modern JavaScript or targeting browsers, ES modules are recommended.