
🔹 CommonJS (CJS)

É o sistema de módulos **clássico do Node.js**.

- **Sintaxe**
- `// exportar`
- `module.exports = function soma(a, b) {`
- `  return a + b;`
- `};`
-
- `// importar`
- `const soma = require('./soma');`
- `console.log(soma(2, 3));`
- **Características**:
  - Executado de forma **síncrona** (por isso nasceu para o Node, que carregava arquivos do disco).
  - Usa `require` e `module.exports`.
  - Padrão até o Node 12 (ainda é amplamente usado).
  - Não funciona nativamente no navegador (precisa de bundler como Webpack, Browserify, etc).

🔹 ES Modules (ESM)

É o sistema **padrão do JavaScript moderno (ECMAScript 2015)**.

- **Sintaxe**
- `// exportar`
- `export function soma(a, b) {`
- `  return a + b;`
- `}`
-
- `// importar`
- `import { soma } from './soma.js';`
- `console.log(soma(2, 3));`
- **Características**:
  - Suporte nativo em navegadores modernos **e** no Node (a partir da versão 12 com `"type": "module"` no `package.json`).
  - Executado de forma **assíncrona** (o `import` pode carregar módulos sob demanda).
  - Usa `import` e `export`.
  - Melhor compatibilidade futura (é o padrão da linguagem).

🔹 Diferenças principais

Aspecto	CommonJS (`require`)	ES Modules (`import/export`)
Origem	Node.js	ECMAScript (padrão JS)
Carregamento	Síncrono	Assíncrono
Sintaxe	`require` / `module.exports`	`import` / `export`
Suporte	Nativo no Node	Nativo em navegadores + Node
Árvore de dependências	Mais difícil de otimizar	Mais fácil de otimizar (tree-shaking em bundlers)

🔹 Quando usar cada um?

- ☑ **CommonJS**
  - Se você está mantendo ou trabalhando em um projeto **antigo** de Node.
  - Se usa pacotes que ainda só exportam em CommonJS.
- ☑ **ES Modules**
  - Para **projetos novos** (tanto em Node quanto no navegador).
  - Se precisa de **compatibilidade entre frontend e backend**.
  - Quando quer aproveitar otimizações modernas (ex.: **tree-shaking** em bundlers).

👉 Resumindo:

- **CJS** → legado, ainda muito usado no backend.
- **ESM** → padrão oficial e recomendável para projetos novos.

Quer que eu faça um **exemplo comparando os dois lado a lado** em um projetinho Node simples (com `math.js` e `index.js`) para ficar ainda mais claro?