Diferenças entre CommonJS e ES Modules

■ CommonJS (CJS)

Origem: Criado para o Node.js.

Sintaxe: const fs = require('fs'); module.exports = {...}

Execução: Carregamento síncrono (require acontece linha a linha).

Extensão padrão: .cjs ou .js (se não houver 'type: module' no package.json).

Compatibilidade: Padrão histórico do Node.js, muito usado em pacotes antigos do npm.

Limitações: Não funciona nativamente no navegador; suporte limitado para tree-shaking.

■ ES Modules (ESM)

Origem: Padrão oficial do JavaScript (ES6, 2015).

Sintaxe: import fs from 'fs'; export function ...

Execução: Carregamento assíncrono (útil no navegador).

Extensão padrão: .mjs ou .js (se o package.json tiver 'type: module').

Compatibilidade: Funciona tanto no Node.js quanto nos navegadores; melhor suporte para tree-shaking.

Vantagens: Sintaxe clara e padronizada, suporta import dinâmico (import()).

■ Quando usar cada um

✔■ Use CommonJS (CJS) se:
Está trabalhando em projetos legados ou precisa de compatibilidade com pacotes npm antigos.

Seu código roda apenas no Node.js e não precisa de suporte a browsers modernos.

✔■ Use ES Modules (ESM) se:
Está iniciando um projeto novo.

Precisa que o mesmo código funcione tanto em Node.js quanto em browsers.

Quer tirar proveito de tree-shaking, bundlers modernos e performance.

Deseja usar uma sintaxe padrão do JavaScript (futuro da linguagem).

■ Comparação rápida

Aspecto	CommonJS (CJS)	ES Modules (ESM)
Origem	Node.js	ECMAScript (ES6)
Sintaxe	require / module.exports	import / export
Execução	Síncrono	Assíncrono

Extensões	.cjs / .js	.mjs / .js
Compatibilidade	Node.js antigo / pacotes legados	Node.js moderno / navegadores
Tree-shaking	Limitado	Suportado