CommonJS vs ES Modules

No ecossistema JavaScript, existem dois principais sistemas de módulos: CommonJS (CJS) e ECMAScript Modules (ESM). Ambos servem para organizar e reutilizar código, mas diferem em sintaxe, comportamento e suporte em diferentes ambientes.

CommonJS (CJS)

Prós:

- Sintaxe simples com `require` e `module.exports`.
- Suporte consolidado no Node.js (principalmente versões anteriores ao ES Modules).
- Carregamento dinâmico de módulos em tempo de execução.

Contras:

- Carregamento síncrono, o que pode ser ineficiente para aplicações em browsers.
- Menos compatível com ferramentas modernas de front-end.
- Compatibilidade futura limitada, já que o padrão ECMAScript é ESM.

ECMAScript Modules (ESM)

Prós:

- Sintaxe padronizada pelo ECMAScript (`import` e `export`).
- Carregamento assíncrono, ideal para navegadores.
- Melhor suporte para tree-shaking e otimizações de bundlers.
- É o padrão oficial e recomendado para novos projetos.

Contras:

- Compatibilidade mais limitada em versões antigas do Node.js e browsers antigos.
- Carregamento dinâmico mais verboso (`import()`).
- Migração de projetos antigos em CJS pode exigir ajustes.

Diferenças principais

- Sintaxe: CommonJS usa `require()`/`module.exports`, enquanto ESM usa `import`/`export`.
- Carregamento: CJS é síncrono; ESM é assíncrono.
- Padrão: CJS foi criado para o Node.js; ESM é o padrão oficial do JavaScript.
- Compatibilidade: CJS funciona em versões antigas do Node; ESM é necessário para novas ferramentas e bundlers modernos.

Em resumo, CommonJS ainda é usado em projetos antigos e possui forte integração no ecossistema Node.js, mas os ECMAScript Modules representam o futuro do JavaScript, oferecendo melhor compatibilidade, desempenho e suporte a ferramentas modernas. Para novos

projetos, ESM é a escolha recomendada.