RESUMO DA AULA

20 Export default e Export nomeado

Nesta aula, vamos explorar dois tipos de exporta o que podemos utilizar no JavaScript: a exporta o padr o (default) e a exporta o nomeada. Entender como e quando usar cada uma essencial para organizar e gerenciar m dulos de maneira eficiente no seu projeto. 1. Exporta o Padr o (Default Export) A exporta o padr o indicada quando um m dulo exporta apenas uma funcionalidade principal. No JavaScript, usamos export default para definir a exporta o padr o de um m dulo. Exemplo de exporta o padr o: No arquivo operacoes-matematicas.js, criamos uma fun o simples de soma e a exportamos como padr o: function somar(numero1, numero2) { return numero1 + numero2; } export default somar; Outra forma de definir a exporta o padr o declarar a fun o diretamente junto com o export default: export default function somar(numero1, numero2) { return numero1 + numero2; } Como importar um export padr o: Ao importar um m dulo que foi exportado como padr o, n o precisamos utilizar chaves {}. Podemos simplesmente escrever: import somar from './operacoes-matematicas.js'; Ou seja, o nome somar que usamos na importa o definido por n s e n o precisa corresponder ao nome da fun o no m dulo. Se quisermos, podemos mudar o nome da vari vel que recebe a fun o: import xpto from './operacoes-matematicas.js'; console.log(xpto(1, 3)); // Output: 4 Isso ocorre porque, na exporta o padr o, o nome que definimos na importa o independente do nome do m dulo. Por que usar export default? O export default til quando temos apenas uma funcionalidade principal em um m dulo. Ele ideal para casos como classes, fun es utilit rias nicas, ou um nico componente em frameworks como React. 2. Exporta o Nomeada (Named Export) Com a exporta o nomeada, podemos exportar v rias funcionalidades de um nico m dulo. Usamos export para definir fun es, vari veis ou classes que queremos tornar dispon veis para importa o. Exemplo de exporta o nomeada: Vamos adicionar uma segunda fun o ao nosso arquivo operacoes-matematicas.js e usar a exporta o nomeada: function somar(numero1, numero2) { return numero1 + numero2; } function multiplicar(numero1, numero2) { return numero1 * numero2;

} export { somar, multiplicar }; Ao usar { somar, multiplicar }, estamos definindo explicitamente quais fun es queremos exportar e tornar acess veis para outros m dulos. Como importar uma exporta o nomeada: Para importar fun es ou vari veis que foram exportadas de forma nomeada, precisamos usar {}: import { somar, multiplicar } from './operacoes-matematicas.js'; Se tentarmos importar com um nome diferente, como { soma } no lugar de { somar }, o JavaScript lan ar um erro, pois o nome da importa o deve corresponder exatamente ao nome da exporta o. Qual a diferen a pr tica? Exporta o Padr o (export default): Apenas um default por m dulo. O nome na importa o pode ser qualquer um. til quando o m dulo exporta uma funcionalidade principal. Exporta o Nomeada (export { ... }): Permite m ltiplas exporta es no mesmo m dulo. Os nomes das importa es devem corresponder aos das exporta es. til quando o m dulo cont m v rias fun es ou classes que podem ser usadas individualmente. 3. Qual usar? Embora ambas as abordagens tenham suas utilidades, em geral, preferimos a exporta o nomeada (export { ... }) por alguns motivos: Legibilidade e Consist ncia: A exporta o nomeada deixa claro quais partes de um m dulo est o sendo exportadas e importadas. Refatora o Segura: Se alterarmos o nome de uma fun o exportada no m dulo, o JavaScript lan ar um erro em todos os arquivos que importam essa fun o. Isso facilita a manuten o e evita falhas silenciosas. Escalabilidade: Em projetos grandes, comum que um m dulo tenha v rias fun es ou classes. A exporta o nomeada facilita a organiza o e a clareza do c digo. Por exemplo, se mudarmos o nome da fun o somar para adicionar no arquivo operacoes-matematicas.js: function adicionar(numero1, numero2) { return numero1 + numero2; } export { adicionar }; Ao tentar importar com o nome antigo ({ somar }), o JavaScript lan ar um erro, indicando que o nome foi alterado. Isso nos for a a atualizar todos os locais que dependem do m dulo, garantindo consist ncia no projeto. 4. Como funcionam os dois tipos juntos? Podemos usar a exporta o padr o e a exporta o nomeada no mesmo m dulo. No exemplo abaixo, usamos export default para uma funcionalidade principal e export para outras fun es auxiliares: function subtrair(numero1, numero2) { return numero1 - numero2; } function dividir(numero1, numero2) { return numero1 / numero2; } function somar(numero1, numero2) { return numero1 + numero2; } export { subtrair, dividir }; export default somar; E podemos importar assim: import somar, { subtrair,

dividir } from './operacoes-matematicas.js'; console.log(somar(10, 5)); // Output: 15 console.log(subtrair(10, 5)); // Output: 5 console.log(dividir(10, 5)); // Output: 2 Aqui, somar a exporta o padr o, enquanto subtrair e dividir s o exporta es nomeadas. Resumo Export Default: Apenas um por m dulo. Nome flex vel na importa o. Ideal para m dulos que t m uma nica responsabilidade. Export Nomeado: V rias exporta es por m dulo. Nome deve corresponder exatamente na importa o. Ideal para m dulos que exportam v rias funcionalidades. Na pr xima aula, veremos como aplicar esses conceitos em um projeto maior para otimizar a organiza o de c digo e o uso de m dulos no JavaScript.
