



export & import

CHEAT SHEET

Um guia rápido sobre como exportar e importar dados entre módulos com Javascript ES6



explorer



rocketseat



O que é modularização no Javascript?

A modularização é um recurso de programação que também está presente no Javascript, ganhando mais reforço a partir da versão ES6 (versão 6.0 do Ecma Script).

Em resumo, a modularização no JavaScript é uma maneira de **organizar** nosso código em pedaços menores e independentes, chamados módulos.

Cada módulo pode ter dados e funções que queremos compartilhar com outros arquivos ou mantê-los privados.

Um módulo em Javascript é um arquivo que contém código com informações ou de importação (com import) ou de exportação (com export).



Sintaxe da modularização

Para usar módulos, é preciso adicionar a propriedade `type="module"` no script principal, ao declará-lo no HTML:

```
<script src=".main.js" type="module"></script>
```





Exportação de dados em módulos JS

A regra é: tudo o que for declarado dentro de um módulo fica restrito àquele módulo e não pode ser acessado por outro arquivo.

Para permitir que um dado seja acessível fora do arquivo é necessário exportar esse dado de forma **explícita**, com o **export**.

É possível utilizar *export* para *function*, *class*, *var*, *let*, ou *const*, desde que o dado não esteja dentro de algum escopo no arquivo.

Não é possível exportar dados dentro de condicionais *if* ou *funções*, por exemplo.

💡 O QUE É ESCOPO?

Em programação, "**escopo**" refere-se à área do código onde uma determinada variável, função ou outro elemento é acessível e pode ser referenciado.

Por exemplo: se criamos uma variável dentro de uma função, o escopo dessa variável será essa função. Porém, se essa variável for criada na raiz do código, seu escopo passa a ser todo o arquivo.



Tipos de exportações e importações

Exportação padrão (default)

Exportação definida como padrão do módulo. Sua sintaxe é **export default**
Importante: só pode existir 1 exportação padrão por arquivo.



```
export default function MyFunction() { }  
// exportando uma função
```

```
export default title = "aqui vai um título"  
// exportando uma variável
```

Na **importação padrão**, é possível renomear o dado para qualquer outro nome:



```
import 'arquivo.js'  
// importa um arquivo inteiro e já executa
```

```
import qualquerNome from './arquivo.js'  
// é possível dar qualquer nome para o dado
```

Tipos de exportações e importações

Exportação nomeada (named)

Nesse tipo de exportação, definimos (geralmente no final do código) quais dados serão exportados.

Sua sintaxe é “**export dado**” ou “**export { dado }**”



```
export let dado2 = 54  
export const name = "Ana"
```

```
export { dado2, name }  
// é possível exportar vários dados diferentes em uma mesma  
// linha de código, geralmente declarada no final do arquivo.
```

```
export { dado2 as dado }  
// é possível renomear o dado ANTES de exportar
```

Na **importação**, é necessário que o dado tenha o **mesmo nome**:



```
import { dado2 } from './arquivo.js'  
  
import { name as userName } from './arquivo.js'  
// é possível renomear o dado com "as novo_nome"
```



Tipos de exportações e importações

Exportação mista (mixed export)

Quando um arquivo tem exportação padrão e nomeada simultaneamente:



```
export let dado2 = 42  
  
export default function myFunction() {  
    console.log("hello")  
}
```

Na importação:



```
import myFunctionRenamed, { dado2 } from './arquivo.js'  
// é possível fazer importações mistas de um mesmo arquivo
```

Importando **todos** os dados:



```
import 'arquivo.js'  
// aqui o arquivo é importado e já executado
```

```
import * as namedImports from 'arquivo.js'  
// Importa todas as named exports como namedImports.  
// Para acessar, use: namedImports.nome_do_dado (ex: namedImports.dado2)
```



Conclusão

A modularização é um recurso poderoso de arquitetura nas aplicações. Cada módulo pode ter dados e funções que queremos compartilhar com outros arquivos ou mantê-los privados.

Para tornar algo acessível fora do módulo, usamos a palavra-chave "**export**". Podemos exportar dados individuais com o "**named export**" ou apenas uma coisa principal com o "**default export**".

Para usar o que exportamos em outro arquivo, usamos a palavra-chave "**import**".

Assim, conseguimos criar projetos mais organizados e fáceis de trabalhar, especialmente à medida que nossos programas se tornam maiores e mais complexos.

PARA SABER MAIS – DICAS DE CONTEÚDO



ECMAScript 2015 Language Specification - ECMA-262 6th Edition

<https://262.ecma-international.org/6.0/#sec-modules>



import - JavaScript | MDN

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import>

export - JavaScript | MDN

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/export>