

Imports & Exports

Como reutilizar funções e módulos no JS/TS

Sumário

- Motivação
- Tipos de **Export**
 - Nomeado
 - Padrão (default)
 - Inline x no final
- Tipos de **Import**
 - Nomeado
 - Alias
 - Default
 - Tudo
 - Misturado
- Boas práticas

Motivação

Modularização

Dividir o código em partes menores e **reaproveitáveis**

Tipos de **EXPORT**


```
10  module.exports = router;
```

Nomeado

- Permite exportar múltiplos elementos de um arquivo

VANTAGENS

- Vários exports por arquivo



```
export const PI = 3.14;
```

```
export function soma(a: number, b: number) {  
  return a + b;  
}
```

Padrão (Default)

- Só pode haver **um por arquivo**
- usado quando o módulo tem um valor principal que precisa ser exportado



```
export default function() { console.log('função principal'); }
```

Inline x no final

Não tem nenhuma funcionalidade *especial*, apenas por uma questão de organização e preferência

INLINE



```
export function ola() {}
```

NO FINAL



```
function ola() {}  
function tchau() {}  
export { ola, tchau  
};
```

Dúvidas?

Tipos de **IMPORT**

```
1 import express from "express";
```

Nomeado


- Importa exatamente os nomes *exportados*



```
import { PI, soma } from './math.js';
```

Alias (apelido)

- Renomeia as variáveis importadas



```
import { soma as somar } from './math.js';
```

Default

É livre para ser nomeado como o usuário quiser



```
import qualquerNome from './arquivo.js';
```

Tudo (*)

Importa o módulo todo, fazendo ele se comportar como um objeto

SINTAXE

Modulo.funcao()



```
import * as MathUtils from './math.js';  
MathUtils.soma(2, 3);
```

Dúvidas?

Boas práticas

Use ***EXPORT DEFAULT*** quando o módulo tiver ***um único propósito***

Use ***EXPORTS NOMEADOS*** quando o arquivo tiver ***vários utilitários***

Sempre dê nomes claros e consistentes para ***facilitar o reuso***

Evite usar *import ** sem necessidade -> segurança e poluição

Bora construir exemplos