Aula 6

Módulos em JavaScript (ES6 Modules)

Prof. Bruno Torres

Objetivos 🎯

- Entender o que são módulos
- Dividir código em arquivos reutilizáveis
- Aprender a usar `export` e `import`
- Aplicar em projeto prático

Pré-requisitos (Node.js + Módulos) ⚙️

Se for usar Node.js para testar:

1. Inicie um projeto com:

```
npm init -y
```

2. Ative suporte a módulos ES6:

```
npm pkg set type=module
```

Pré-requisitos (Node.js + Módulos) ⚙️

Ou renomeie os arquivos para `.mjs`

> 💡 **Dica:** Em navegadores, basta usar `<script type="module">`.

O que são módulos? 📦

- Arquivos JS com código **isolado e reutilizável**
- Melhor organização e manutenção do projeto
- Evita conflitos de nomes (escopo)

Criando um módulo 🛠️

```
// saudacao.js
export function saudacao(nome) {
  return `Olá, ${nome}!`;
}
```

Importando o módulo 📥

```
// app.js
import { saudacao } from "./saudacao.js";
console.log(saudacao("Bruno"));
```

Export default 🚀

```
// math.js
export default function soma(a, b) {
  return a + b;
}
```

```
// app.js
import soma from "./math.js";
```

Vários exports 📚

```
export const PI = 3.14;
export function area(r) {
  return PI * r * r;
}
```

```
import { PI, area } from "./figuras.js";
```

Renomeando e agrupando 🧩

```
import { area as calcularArea } from "./figuras.js";
import * as figuras from "./figuras.js";
```

HTML com módulo 🌐

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

- Necessário para ativar os imports/exports

Instalando o Live Server 🖥️

Para evitar erros ao usar módulos com HTML:

1. Instale a extensão **Live Server** no VS Code.
2. Clique com o botão direito no arquivo HTML → **"Open with Live Server"**.
3. Isso abrirá a página no navegador com um servidor local (http://127.0.0.1...).

Por que usar Live Server? ❓

- Navegadores **bloqueiam módulos** com `file://` por segurança (CORS).
- Live Server simula um ambiente real de servidor.
- Permite usar `<script type="module">` sem problemas.

Exemplo prático com HTML 📝

```html
<!-- index.html -->
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8" />
  <title>Exemplo Módulo</title>
</head>
<body>
  <script type="module" src="./main.js"></script>
</body>
</html>
```

Arquivo: saudacao.js

```
// saudacao.js
export function ola(nome) {
  return `Olá, ${nome}!`;
}
```

Arquivo: main.js

```
// main.js
import { ola } from "./saudacao.js";
console.log(ola("Bruno"));
```

Evitando erro CORS 🚫

- Usar servidor local como o **Live Server**
- Módulos não funcionam via `file://`

Projeto prático 🛠️

Objetivo: Separar responsabilidades

- `math.js` → funções matemáticas
- `mensagens.js` → textos
- `app.js` → lógica principal

math.js 📐

```
export function somar(a, b) {
  return a + b;
}
export function multiplicar(a, b) {
  return a * b;
}
```

mensagens.js ✉️

```
export function boasVindas(nome) {
  return `Bem-vindo, ${nome}!`;
}
```

app.js 🖥️

```
import { somar } from "./math.js";
import { boasVindas } from "./mensagens.js";

console.log(boasVindas("Bruno"));
console.log(somar(3, 4));
```

Conclusão ✅

- Organize seu código com módulos
- Separe responsabilidades
- Use sempre `type="module"` no HTML

Obrigado! 🙌

Dúvidas? Vamos praticar!