Uma das situações em que o TypeScript pode ser útil é quando há muitos desenvolvedores trabalhando em um projeto grande e complexo, com muitos arquivos e interações entre diferentes partes do código.

Com o JavaScript puro, é fácil cometer erros de digitação ou confundir o tipo de uma variável ou parâmetro. Esses erros podem passar despercebidos e causar problemas no código em tempo de execução.

Com o TypeScript, os desenvolvedores podem definir tipos para variáveis, parâmetros de função, objetos e outros elementos do código. Isso permite que o compilador do TypeScript detecte e reporte erros de tipo em tempo de compilação, antes mesmo de executar o código.

Por exemplo, suponha que você tenha uma função que recebe uma string e um número como parâmetros e retorna uma string concatenada com o número. Em JavaScript puro, a definição dessa função seria assim:

javascript Copy code

```
function concatenateStringAndNumber(str, num) { return str + num; }
```

Agora, suponha que outro desenvolvedor, em outra parte do código, chama essa função passando uma string e um boolean em vez de uma string e um número:

arduino Copy code

```
const myString = "Hello"; const myBoolean = true; const result =
concatenateStringAndNumber(myString, myBoolean);
```

Em JavaScript, esse código seria executado normalmente, mas o resultado seria uma string concatenada com "true", que não é o que se esperava. Em um projeto grande, pode ser difícil encontrar esse erro de tipo.

Com o TypeScript, a definição dessa função seria assim:

typescript Copy code

```
function concatenateStringAndNumber(str: string, num: number): string {
return str + num; }
```

Agora, se outro desenvolvedor tentar chamar essa função passando uma string e um boolean, o compilador do TypeScript irá reportar um erro de tipo em tempo de compilação, impedindo que o código seja executado com esse erro.

vbnet Copy code

```
const myString = "Hello"; const myBoolean = true; const result =
concatenateStringAndNumber(myString, myBoolean); // Erro: o tipo boolean
não pode ser atribuído ao tipo número
```