06

Para saber mais: strictNullChecks, exemplo 1

O compilador do TypeScript possui uma série de configurações e uma delas é a `strictNullChecks`. Neste modo, null e undefined não fazem parte do domínio dos tipos e só podem ser atribuídos a eles mesmos. Com a exceção de undefined que pode ser atribuído a void. Isso pode ser interessante para evitarmos valores nulos e indefinidos em nosso projeto.

Que tal vermos mais um exemplo no qual ele pode nos ajudar a detectar erros em nosso código?

Vamos buscar em um HTML hipotético um elemento com ID `cartao_1` e a partir dele acessar seu elemento pai através de `parentNode`:

```
const elCartao: HTMLDivElement = <HTMLDivElement> document.querySelector('#cartao_1');
let elPai = elCartao.parentElement;
```

Excelente. Mas o mesmo código não compilará com `strictNullChecks` com a seguinte alteração:

```
const elCartao: HTMLDivElement = <HTMLDivElement> document.querySelector('#cartao_1');
let elPaiDoPai = elCartao.parentElement.parentElement; // [ts] Object is possibly 'null'
```

Veja que o aviso do compilador TypeScript faz todo sentido, porque pode ser que o elemento pai do pai do elemento `elCartao` não exista e, se não existir, o valor de `parentNode` será null.

Dessa forma, o programador terá que lidar antecipadamente com essa situação em seu código:

```
const elCartao: HTMLDivElement = <HTMLDivElement> document.querySelector('#cartao_1');
let elPaiDoPai;
if(elCartao.parentElement) {
    elPaiDoPai = elCartao.parentElement.parentElement;
}
console.log(elPaiDoPai);
```