07

Para saber mais: strictNullChecks, exemplo 2

Temos a seguinte função:

```
function minhaFuncao(flag: boolean) {

    let valor = null;
    if(flag) return null;
    return true;
}

let x = minhaFuncao(false);
```

O código compila, pois `minhaFuncao` pode retornar `null` ou um `boolean` dependendo do valor do parâmetro `flag` passado. O TypeScript também consegue inferir os dois tipos retornados pela função.

Mas se quisermos tipar o retorno de `minhaFuncao` como `boolean` ? Teremos um erro de compilação:

```
function minhaFuncao(flag: boolean): boolean {

    let valor = null;

    // erro aqui!
    // Type 'null' is not assignable to type 'boolean'.

    if(flag) return null;
    return true;
}

let x = minhaFuncao(false);
```

Com `strictNullChecks` ativado não podemos retornar null para uma função que explicitamente indicamos que retorna boolean. Se a opção do compilador não estivesse ativa, funcionaria pois `null` pode ser atribuído a qualquer tipo do TypeScript. Mas será que conseguimos fazer esse código compilar sem termos que desativar esta `strictNullChecks` no compilador?

Podemos indicar que a função pode devolver mais de um tipo, no caso ela devolverá `boolean` ou `null` :

```
// deixarmos explícitos que a função pode retornar boolean ou null
function minhaFuncao(flag: boolean): boolean | null{

    let valor = null;
    if(flag) return null;
    return true;
}

let x = minhaFuncao(false);
```

Agora, como explicitamos que seu retorno pode ser também `null`, nosso código passará pelo `strictNullChecks`.

Curiosamente, linguagens como a Golang permitem uma função ou método ter mais de um tipo de retorno.