07

Casting explícito

Transcrição

A classe `NegociacaoController` não compila. Precisamos ajustá-la:

```
// app/ts/controllers/NegociacaoController.ts

    class NegociacaoController {

        private _inputData: HTMLInputElement;
        private _inputQuantidade: HTMLInputElement;
        private _inputValor: HTMLInputElement;

       // código posterior omitido
```

Nossos inputs do formulário são do tipo HTMLInputElement. Inclusive, o próprio VSCode já nos lista todos os tipos possíveis da interface DOM. Por enquanto, entenda que o TypeScript já vem com uma lista de definições para esses tipos que são lidas automaticamente pelo VCode.

No entanto, nosso código ainda não compila, inclusive o compilador indica um erro agora dentro do `constructor()` de `NegociacaoController`. O problema é que `document.querySelector()` retorna elementos do tipo `Element`, um tipo mais genérico, pois seu retorno pode ser elementos que representam inputs, tabelas, destaques entre outros. Mas ao tratá-los todos como `Element`, deixamos de ter acesso a métodos e propriedades específicas dos elementos retornados, no caso, a propriedade `value` que todo input possui.

O primeiro passo para resolvermos nosso problema, é utilizarmos o tipo correto nas propriedades da nossa classe. Vamos utilizar o tipo `HTMLInputElement`:

```
// app/ts/controllers/NegociacaoController.ts

    class NegociacaoController {

        private _inputData: HTMLInputElement;
        private _inputQuantidade: HTMLInputElement;
        private _inputValor: HTMLInputElement;

        constructor() {
            // continua com erro de compilação
            this._inputData = document.querySelector('#data');
            this._inputQuantidade = document.querySelector('#quantidade');
            this._inputValor = document.querySelector('#valor');
        }
```

O problema é que geramos um erro de compilação dentro do `constructor()` da classe. Não podemos pegar o valor de um tipo mais genérico e atribuí-lo a um tipo mais específico, apenas o contrário é permitido automaticamente pela linguagens. Porém, como temos certeza que estamos buscando elementos do tipo `input`, podemos assumir a responsabilidade de conversão, isto é, realizando uma conversão explícita:

```
// app/ts/controllers/NegociacaoController.ts

    class NegociacaoController {

        private _inputData: HTMLInputElement;
        private _inputQuantidade: HTMLInputElement;
        private _inputValor: HTMLInputElement;

        constructor() {

            this._inputData = <HTMLInputElement>document.querySelector('#data');
            this._inputQuantidade = <HTMLInputElement>document.querySelector('#quantidade');
            this._inputValor = <HTMLInputElement>document.querySelector('#valor');
        }
```

Esse processo de conversão explícita é chamado de `casting`. No caso, estamos forçando uma conversão de um tipo mais genérico para um tipo mais específico. Pode ocorrer o casting implícito, quando atribuímos um tipo mais específico a um tipo mais genérico. Por enquanto, apenas o casting explícito se apresentou necessário em nossa aplicação.

Estamos quase lá! Mas por mais que tenhamos resolvido os problemas de compilação, novos apareceram. No método adiciona, como o retorno de um HTMLInputElement é sempre uma string, ela não se coaduna com os tipos esperados pelo `constructor()` de `Negociacao`. Precisamos converter os dados antes que sejam passados para o `constructor()`.