09

Modelando com TypeScript

Transcrição

O código de `Negociacao.ts` não compila, porque apesar do TS ser um superset do ES2015, ele necessita uma declaração especial para propriedade de classes. Essas são declaradas no corpo da classe:

```
class Negociacao {

     // declaração das propriedades de classe
    _data;
    _quantidade;
    _valor;

    constructor(data, quantidade, valor) {

        this._data = data;
        this._quantidade = quantidade;
        this._valor = valor;
    }

    get data() {

        return this._data;
    }

    get quantidade() {

        return this._quantidade;
    }

    get valor() {

        return this._valor;
    }

    get volume() {

        return this._quantidade * this._valor;
    }
}
```

Se rodarmos nosso compilador mais uma vez através do comando `npm run compile` nenhuma mensagem de será exibida, indicando que nosso código compila. Um ponto curioso é que o resultado da compilação em `app/js/models/Negociacao.js` não possui as propriedades de classe. Está corretíssimo, pois se tivesse, não seria reconhecido pelo navegador. Lembre-se que todo código TypeScript deve ser transformado em um código em JavaScript para que seja compreendido pelo navegador.

Mas parece que escrevemos mais para pouca coisa, pois ainda podemos acessar a propriedade `_quantidade` mesmo com um código escrito em TypeScript.

Podemos lançar mão de um recurso exclusivo da linguagem, o modificador `private` para tornar as propriedades da classe privadas. Propriedades privadas só podem ser acessadas pelos métodos da própria classe, resultando em um erro de compilação caso seja feito um acesso externo.:

```
// app/js/models/Negociacao.js

class Negociacao {

    private _data;
    private _quantidade;
    private _valor;

    constructor(data, quantidade,  valor) {

        this._data = data;
        this._quantidade = quantidade;
        this._valor = valor;
    }

    get data() {

        return this._data;

    }

    get quantidade() {

        return this._quantidade;

    }

    get valor() {

        return this._valor;
    }

    get volume() {

        return this._quantidade * this._valor;
    }
}
```

Agora, em `app.ts` tentarmos acessar a propriedade `_quantidade` , por exemplo, receberemos um erro de compilação:

```
const negociacao = new Negociacao(new Date(), 1, 100);
negociacao._quantidade = 3; // O VSCODE indica um erro de compilação aqui
console.log(negociacao.quantidade);
```

Inclusive o VCode indicará para nós que estamos tentando acessar uma propriedade com modificador de acesso privado. Contudo, mesmo com o erro de compilação, será gerado um arquivo `.js` . Não queremos isso, queremos apenas arquivos `.js` gerados a partir de arquivos `.ts` sintaticamente corretos. Para isso, vamos adicionar em `tsconfig.js` mais uma configuração, a `"noEmiteOnError"` :

```
{
    "compilerOptions": {
        "target": "es6",
        "outDir": "app/js",
        "noEmitOnError": true
    },
    "include": [
        "app/ts/**/*"
    ]
}
```

Agora, só serão gerados arquivos `.js` apenas se o arquivo `.ts` não tiver nenhum erro de compilação.