03

Instalando TypeScript Definitions

Transcrição

jQuery foi criado com TypeScript? Com certeza não e, se não foi, não possui variáveis, propriedades e parâmetro de métodos e funções tipados. Então, como suportar os recursos da linguagem TypeScript com uma biblioteca que não foi criada utilizando nessa linguagem?

Os criadores da biblioteca ou terceiros podem criar um arquivo chamado TypeScript Declaration File. Este arquivo possui informações dos nomes de métodos e funções, inclusive tipos que podem ser utilizados pelo TypeScript. Quando carregado, o TypeScript conseguirá, baseado nesse arquivo, realizar checagem estática inclusive lançar mão de todos seu poder através de seu editor ou IDE favorita.

No caso, vamos instalar o tipo do jQuery. Vale lembrar que esse tipo não foi definido pela equipe do jQuery:

```
npm install @types/jquery@2.0.42 --save-dev
```

Você pode instalar o type definiton da sua biblioteca favorita, contanto que ela exista. Inclusive podem haver arquivos de definições criados por mais de um colaborador, no final, somos nós que devemos escolher o arquivo que for mais atualizado. Não há solução mágica, é necessário realizar pesquisas na Internet. Inclusive, pode ser que nem existe um arquivo `tsd` para sua biblioteca favorita, sendo assim, a solução com `declare var` continua sendo válida.

Vamos abrir e fechar o Visual Studio Code. Por padrão, o compilador do TypeScript procurará por padrão todos os @types dentro da pasta `node_modules` sem termos que nos preocupar.

Agora, com o VSCode reaberto, vamos remover o `declare var` . Em seguida, vamos usar o tipo `jQuery` para `_elemento` , propriedade da nossa classe. Esse tipo não está disponível antes, passou a estar depois de termos instalado o tds do jQuery:

```
abstract class View<T> {

    private _elemento: JQuery;

    constructor(seletor: string) {

        this._elemento = $(seletor);
    }

    update(model: T) {

        this._elemento.html(this.template(model));
    }

    abstract template(model: T): string;
}
```

Fantástico! Agora, podemos usar o jQuery na declaração da nossa classe `NegociacaoController` :

```
class NegociacaoController {

    private _inputData: JQuery;
    private _inputQuantidade: JQuery;
    private _inputValor: JQuery;
    private _negociacoes = new Negociacoes();
    private _negociacoesView = new NegociacoesView('#negociacoesView');
    private _mensagemView = new MensagemView('#mensagemView');


    constructor() {
        this._inputData = $('#data');
        this._inputQuantidade = $('#quantidade');
        this._inputValor = $('#valor');
        this._negociacoesView.update(this._negociacoes);
    }

    adiciona(event: Event) {

        event.preventDefault();

        const negociacao = new Negociacao(
            new Date(this._inputData.val().replace(/-/g, ',')),
            parseInt(this._inputQuantidade.val()),
            parseFloat(this._inputValor.val())
        );

        this._negociacoes.adiciona(negociacao);
        this._negociacoesView.update(this._negociacoes);
        this._mensagemView.update('Negociação adicionada com sucesso');
    }
}


// app/ts/app.ts

const controller = new NegociacaoController();

$('.form').submit(controller.adiciona.bind(controller));
```

Bibliotecas como `Lodash`, `Underscore`, inclusive plugins do jQuery UI possuem seu respectivo arquivo `tsd`. Um repositório do git que centraliza algumas desses arquivo é `https://github.com/DefinitelyTyped/DefinitelyTyped`.