03

API Reflection e as facetas de this

Transcrição

Temos uma função com o escopo dinâmico:

```
class NegociacaoController {

    constructor() {

        let $ = document.querySelector.bind(document);
        this._inputData = $('#data');
        this._inputQuantidade = $('#quantidade');
        this._inputValor = $('#valor');

        this._listaNegociacoes = new ListaNegociacoes(function(model) {
            this._negociacoesView.update(model);
        });
//...
```

O `this` irá variar de acordo com o contexto da execução. Como a função será chamada dentro da classe `ListaNegociacoes`, ele será usado como contexto do `this` da função.

```
this._listaNegociacoes = new ListaNegociacoes(function(model) {
    this._negociacoesView.update(model);
});
```

Mas para que o código funcione, queremos que o `this` tenha como contexto o `NegociacaoController`. Da mesma maneira que o `this` é dinâmico, nós programaticamente podemos modificá-lo. Como faremos isso? Primeiramente, no construtor de `_listaNegociacoes`, adicionaremos o primeiro parâmetro: `this` - referente ao `NegociacaoController`.

```
class NegociacaoController {

    constructor() {

        let $ = document.querySelector.bind(document);
        this._inputData = $('#data');
        this._inputQuantidade = $('#quantidade');
        this._inputValor = $('#valor');

        this._listaNegociacoes = new ListaNegociacoes(this, function(model) {
            this._negociacoesView.update(model);
        });
```

Este que será recebido depois em `ListaNegociacoes.js`, como `contexto`:

```
class ListaNegociacoes {
```

```
    constructor(contexto, armadilha) {

        this._negociacoes = [];
        this._armadilha = armadilha;
        this._contexto = contexto;
    }
//...
```

Observe que criamos o atributo `_contexto` .

Agora o construtor do model recebe o contexto, no qual queremos que ele execute uma função. Mas para que o `this` seja realmente o `NegociacaoController` , teremos que mudar a maneira de chamar a função utilizada. Pediremos uma ajuda para a **API de Reflexão** do JavaScript, adicionando em `ListaNegociacoes.js` o `Reflect.apply` . Com isto, chamaremos o método estático da classe:

```
adiciona(negociacao) {
    this._negociacoes.push(negociacao);
    //this._armadilha(this);
    Reflect.apply(this._armadilha, this._contexto, [this]);
}

get negociacoes() {
    return [].concat(this._negociacoes);
}

esvazia() {
    this._negociacoes = [];
    //this._armadilha(this);
    Reflect.apply(this._armadilha, this._contexto, [this]);
}
```

Observe que fizemos algumas alterações nos métodos `adiciona()` e `esvazia()` . O `Reflect.apply` recebeu o `this._armadilha` como primeiro parâmetro e o segundo é `this._contexto` . O terceiro parâmetro é o `[this]` , que será a própria `ListaNegociacoes` . Depois, adicionamos o `Reflect.apply()` também no `esvazia()` .

Se executarmos o código, o formulário continua funcionando normalmente.

Nós conseguimos redefinir o contexto em que queremos executar a função de `_listaNegociacoes` , utilizando o `Reflect.apply()` . O método `apply()` recebeu a função executada, depois o contexto e os parâmetros que serão

passados para a função. Fizemos isto com o `adiciona()` e o `esvazia()`.

```
adiciona(negociacao) {
    this._negociacoes.push(negociacao);
    //this._armadilha(this);
    Reflect.apply(this._armadilha, this._contexto, [this]);
}
```

A função espera receber um modelo no `NegociacaoController`, dentro do `constructor`:

```
this._listaNegociacoes = new ListaNegociacoes(this, function(model) {
    this._negociacoesView.update(model);
});
```

No método `adiciona()`, passaremos o `[this]` que será o `ListaNegociacoes`. O nosso código está funcionando corretamente, mas será que existe outra maneira de conseguirmos o mesmo efeito, sem ter que passar o contexto para o `model`? Veremos mais adiante.