04

Arrow function e seu escopo léxico

Transcrição

Nós queremos que `function()` execute, mas que `this` seja o `NegociacaoController` sem precisarmos passar um contexto:

```
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);
        });
//...
```

Começaremos removendo o `this` da função:

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

Vamos retirar também o `contexto` de `ListaNegociacoes` :

```
class ListaNegociacoes {

  constructor(armadilha) {

        this._negociacoes = [];
        this._armadilha = armadilha;
  }
  adiciona(negociacao) {
      this._negociacoes.push(negociacao);
      this._armadilha(this);
  }

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

  esvazia() {
      this._negociacoes = [];
      this._armadilha(this);
  }
```

```
//...
```

Retiramos o `Reflect.apply()` e deixamos o código como estava anteriormente. Agora será preciso encontrar um forma para que quando o `_armadilha(this)` seja executado o contexto seja `NegociacaoController` .

Primeiramente, faremos um pequeno ajuste no `NegociacaoController` , ao adicionarmos uma *arrow function*, usando `=>` :

```
class NegociacaoController {

    constructor() {

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

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

Porém, se executarmos o código, seremos surpreendidos com o formulário funcionando corretamente. Como isso é possível? Isto ocorre porque a *arrow function* não é apenas uma maneira sucinta de escrever uma função, ela também tem um característica peculiar: o escopo de `this` é **léxico**, em vez de ser dinâmico como a outra função. Isto significa que o `this` não mudará de acordo com o contexto. Da maneira como estruturamos o código, o `this` será `NegociacaoController` - esta condição será mantida independente do local em que chamemos a *arrow function*, porque ela está amarrada a um escopo imutável.

Então, o `this` de uma *arrow function* é **léxico**, enquanto o `this` de uma função padrão é **dinâmico**. Com esse ajuste, conseguimos deixar o nosso código mais sucinto.