06

Objetos imutáveis

Transcrição

Nós usaremos um artifício existente há algum tempo na linguagem JavaScript: "congelaremos" um objeto e com isso, qualquer alteração nas suas propriedades será ignorada. Isso funcionará no caso da classe `Negociacao`, porque nem mesmo os métodos da classe podem alterar as propriedades de uma negociação criada.

Para isso, usaremos o método `Object.freeze()` e vamos congelar o `n1`.

```
<script>

    var n1 = new Negociacao(new Date(), 5, 700);

    Object.freeze(n1);
    n1._quantidade = 100000000000;

    console.log(n1.quantidade);
    console.log(n1.data);
    console.log(n1.valor);
    console.log(n1.volume);
</script>
```

Por convenção, não podemos fazer isso. Mas será que conseguimos alterar um objeto congelado? Faremos um teste. No nosso caso, qual será o valor da `quantidade`: `5` ou `100000000000`?

No Console, veremos o valor `5`, isto significa que não conseguimos alterar e o objeto foi congelado. Lembrando que `quantidade`, é uma propriedade que chamamos em `Negociacao.js`, que por "debaixo dos panos", rodará como um método e retornará `this._quantidade`:

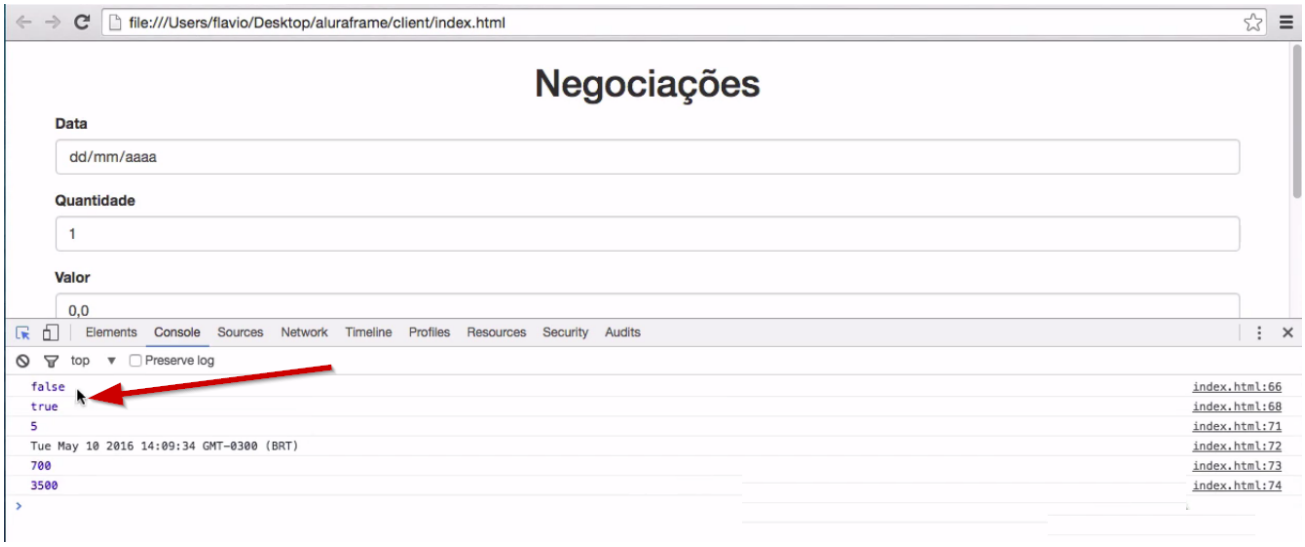
```
get quantidade() {
     return this._quantidade;
}
```

Mesmo colocando `n1._quantidade = 100000000000;` no `index.html`, esse valor parece ter sido ignorado. Isso é bom, agora, o desenvolvedor já não conseguirá alterar esta quantidade. Vamos fazer um pequeno teste que nos mostrará o que está congelado no Console:

```
<script>

    var n1 = new Negociacao(new Date(), 5, 700);

    console.log(Object.isFrozen(n1));
    Object.freeze(n1);
```

```
    console.log(Object.isFrozen(n1));
    n1._quantidade = 100000000000;

    console.log(n1.quantidade);
    console.log(n1.data);
    console.log(n1.valor);
    console.log(n1.volume);
</script>
```

No Console, veremos o seguinte resultado:



O objeto antes não era congelado e depois, foi. E por isso, não conseguimos alterar `_quantidade`, o atributo que por convenção definimos ser privado. Porém, essa solução é procedural, porque você terá sempre que lembrar de congelar a instância. No entanto, queremos que ao utilizarmos o `new Negociacao`, ele já devolva uma instância congelada. Nós podemos fazer isso, congelando a instância no construtor. Em `Negociacao.js`, adicionaremos `Object.freeze()` após último `this`:

```
class Negociacao {

    constructor(data, quantidade, valor) {

        this._data = data;
        this._quantidade = quantidade;
        this._valor = valor;
        Object.freeze(this);

    }

//...
```

No entanto, você se lembra que o `this` é uma variável implícita? E quando algum método é chamado, temos acesso à instância trabalhada? O `this` do `Object.freeze()` será o `n1` no `index.html`.

```
<script>

    var n1 = new Negociacao(new Date(), 5, 700);

    n1._quantidade = 100000000000;
```

```
    console.log(n1.quantidade);
    console.log(n1.data);
    console.log(n1.valor);
    console.log(n1.volume);
</script>
```

Então, quando damos uma `new Negociacao`, ele já devolverá uma instância congelada. Aparentemente, resolvemos o nosso problema. Não é uma solução 100% de encapsulamento, porque ainda conseguimos enxergar o `n1._quantidade` - em linguagens como Java ou C#, não iríamos enxergá-lo. Em JS, enxergamos a `_quantidade`, mas não conseguimos alterá-la. Desta forma, garantimos que a negociação não será modificada.

Vamos verificar mais adiante se de fato a `Negociacao` não será alterada.