07

A instância é imutável mesmo?

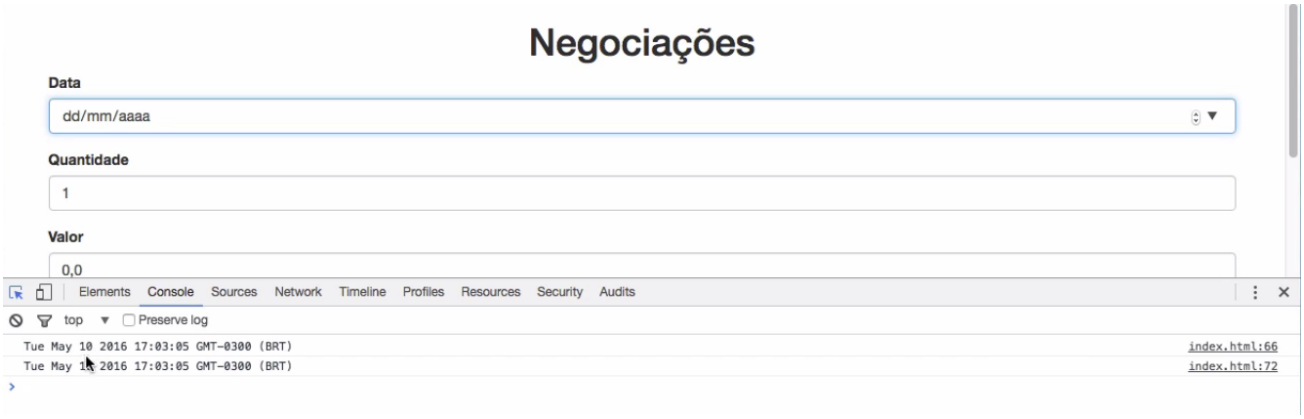
Transcrição

Pela regra de negócio, uma negociação deve ser imutável e não pode ser alterada depois de criada. Mas faremos um novo teste e nele vamos alterar a data da negociação. Já fizemos isto com a quantidade. Agora, criaremos a variável `amanha` que será referente a data de amanhã.

Usaremos o `new Date()` somado com `1`. Sempre que trabalharmos com data, utilizaremos o `setDate()`. Como a data atual é `10`, setaremos com o valor `11`. Então, diremos que `n1.data` será igual a variável `amanha`.

```
<script src="js/app/models/Negociacao.js"></script>
<script>

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

    console.log(n1.data);

    var amanha = new Date();
    amanha.setDate(11);

    n1.data = amanha;

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

Para verificar se o código funciona corretamente, adicionamos o `console.log()`. Se Negociação for realmente imutável, a data não poderá ser alterada.



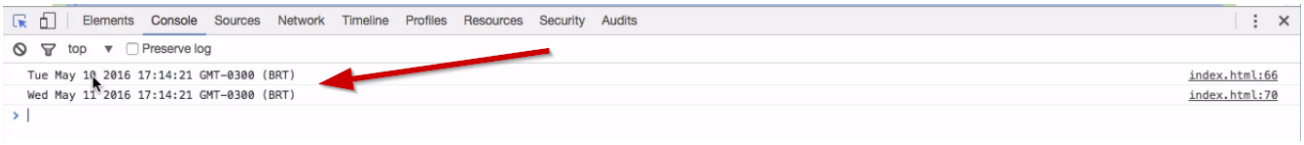
Vemos que ela continuou imutável.

Agora, quero fazer um teste diferente. Não vamos mais trabalhar com a variável `amanha` e vamos substituí-la por `n1.data.setDate()`. Como não é um método que existe na data, iremos defini-lo com `11`:

```
<script>

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

```
    console.log(n1.data);

    n1.data.setDate(11);

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

Será que a data será alterada?



Veja que a data foi alterada, no segundo `console` a data exibida foi `11` . Isto significa que a nossa `Negociacao` é mutável. Assim, alguém pode acessar o sistema e alterá-lo, o que não pode acontecer.

Mas se nós utilizamos o `Object.freeze()` , por que isso aconteceu? O `Object.freeze` é shallow, ou seja, ele ficará na superfície. Quando congelamos um valor, não podemos alterá-lo. Porém, como `_data` é um objeto, não entrará como uma das propriedades do objeto. Então, não é feito o que chamamos de *deep freeze*. Quando trabalhamos com um objeto e dentro dele temos outras propriedades, estas não ficarão congeladas. Para resolver isso, aplicaremos uma aplicação defensiva mais adiante.