05

Construindo armadilhas de escrita

Transcrição

Não estamos interessados em executar um código quando ocorrer a leitura, e sim, quando acontecer a modificação de alguma propriedade. Em seguida, faremos pequenas alterações no arquivo `index.html`. Atualmente, o `get` está assim:

```
<script>

    let negociacao = new Proxy(new Negociacao(new Date(), 1, 100), {
    get: function(target, prop, receiver) {

        console.log(`a propriedade "${prop}" foi interceptada`);
        return Reflect.get(target, prop, receiver);
    }
    });
    negociacao.quantidade = 10;
    negociacao.valor = 100;
</script>
```

Mas esse código ainda não funcionará, porque `quantidade` e `valor` em `Negociacao.js` são getters. Por isso, não podemos fazer uma atribuição, considerando que são apenas leitura e não podem ser alterados. De volta ao `index.html`, faremos uma "licença poética" e acessaremos diretamente as propriedades *privates*.

```
negociacao._quantidade = 10;
negociacao._valor = 100;
```

Vamos desrespeitar a convenção da nossa propriedade, mas com isso, poderemos disparar a nossa armadilha.

A primeira coisa que devemos fazer para executar uma armadilha quando estou atribuindo, é alterarmos de `get` para `set`. Também adicionaremos outro parâmetro: `value`.

```
<script>

    let negociacao = new Proxy(new Negociacao(new Date(), 1, 100), {
    set: function(target, prop, value, receiver) {

        console.log(`a propriedade "${prop}" foi interceptada`);
        return Reflect.set(target, prop, value, receiver);
    });
    negociacao._quantidade = 10;
    negociacao._valor = 100;
</script>
```

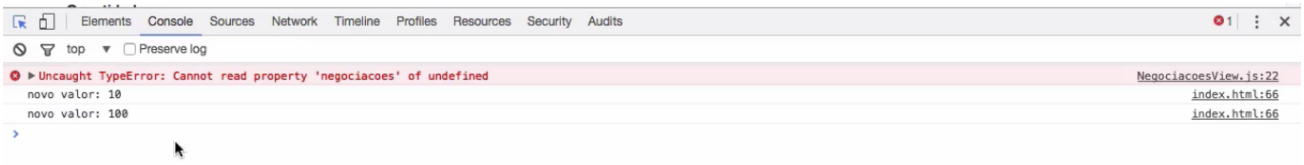
Quando atribuímos `10` para `_quantidade`, a função no `set` será chamada e retornará os quatro parâmetros. No `console.log()` queremos exibir o valor atual e o que será exibido depois.

```
<script>

    let negociacao = new Proxy(new Negociacao(new Date(), 1, 100), {
    set: function(target, prop, value, receiver) {

        console.log(`novo valor: ${value}`);
        return Reflect.set(target, prop, value, receiver);
}
    });
    negociacao._quantidade = 10;
    negociacao._valor = 100;
</script>
```

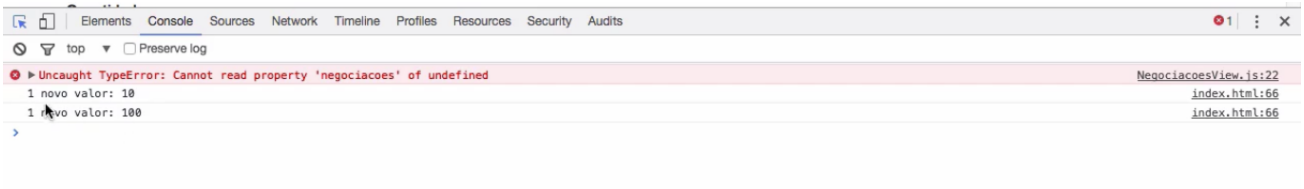
Com o `value` no `console.log` exibiremos o valor que queremos atribuir.



A armadilha foi executada e os valores exibidos estão corretos. Mas também queremos mostrar o valor antigo, para isto, só teremos acesso ao `target` que é o objeto `Negociacao`. Vamos tentar usar o seguinte código no `console.log`:

```
console.log(`${target.quantidade} novo valor: ${value}`);
return Reflect.set(target, prop, value, receiver);
```

No Console. veremos os seguintes valores:



Para `quantidade`, o valor antigo ficou igual a `1`, e o mesmo para o `valor`. A explicação é que a linha do `console.log` está sendo executada também quando tentamos descobrir o `negociacao._valor`. Então, o valor de `quantidade` deve ser igual ao valor de `prop`. Mas como conseguiremos fazer a *property* dinâmica? Não temos a opção de usar `${target.prop}`, mas podemos usar um recurso do JS, podemos passar `${target[prop]}`. O trecho do código ficará assim:

```
<script>

  let negociacao = new Proxy(new Negociacao(new Date(), 1, 100), {
    set: function(target, prop, value, receiver) {

        console.log(`${target[prop]} novo valor: ${value}`);
        return Reflect.set(target, prop, value, receiver);
}
    });
    negociacao._quantidade = 10;
    negociacao._valor = 100;
</script>
```

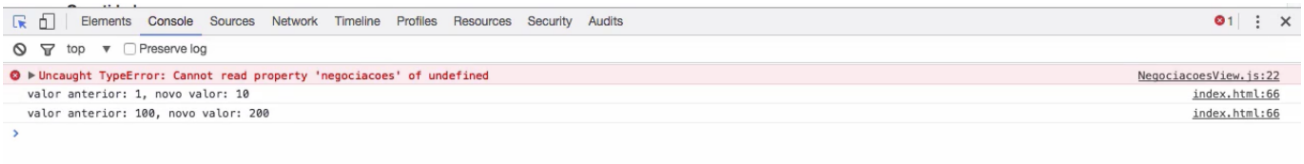
O `prop` junto com o `target` será o `_quantidade` e conseguirá ler o valor. Em seguida, adicionaremos `valor anterior:` à linha.

```
<script>

  let negociacao = new Proxy(new Negociacao(new Date(), 2, 100), {
    set: function(target, prop, value, receiver) {

        console.log(`valor anterior: ${target[prop]} novo valor: ${value}`);
        return Reflect.set(target, prop, value, receiver);
}
    });
    negociacao._quantidade = 10;
    negociacao._valor = 200;
</script>
```

No Console, veremos os seguintes valores:



Com isso, resolvemos o problema da execução de um código que só atualizará a View quando `_quantidade` e `_valor` forem alterados.