01

Classe Mensagem

Transcrição

Começando deste ponto? Você pode fazer o DOWNLOAD (https://github.com/alura-cursos/javascript-avancado-i/archive/aula6.zip) completo do projeto até aqui e continuar seus estudos.

Nossa aplicação já está funcionando, mas vamos melhorar a experiência do usuário. Quando adicionamos uma negociação, os dados são inseridos na tabela. Nosso objetivo será exibir uma mensagem para o usuário, na qual informaremos que a negociação foi incluída com sucesso. Para fazer isto, vamos criar um novo modelo que chamaremos de `Mensagem.js`. Nele, adicionaremos um texto e sempre que quisermos exibir uma mensagem, será instanciado um objeto da minha classe `Mensagem`. O texto que será exibido, ficará guardado nesta classe.

```
class Mensagem {

    constructor() {

      this._texto;
    }

    get texto() {

        return this._texto;

    }
}
```

Nós usamos a convenção do prefixo `_` para manter o `_texto` privado. Usamos um `get` que terá um `return this._texto`.

Nós queremos também ser capazes de alterar o texto, faremos isto, adicionando o `set texto()`. Assim como temos a opção de usar o `get`, usaremos o `set`:

```
set texto(texto) {

    this._texto = texto;
}
```

Mas seria possível aceitar um intervenção como a variável `let`, como nas linhas abaixo:

```
let mensagem = new Mensagem();
mensagem.texto = 'x';
```

O valor dentro do `mensagem.texto` será enviado por debaixo dos panos para o método `texto()` e depois, será atribuído a `mensagem`. Porém, quando criamos uma mensagem nova, o valor deve estar com uma *string* em branco. Mas ainda é possível alterar o texto da mensagem em branco:

```
let mensagem = new Mensagem('xxxx');
mensagem.texto = 'nova mensagem';
```

Temos ainda a opção de já passar a mensagem no `constructor()`:

```
class Mensagem {

    constructor(texto) {

        this._texto = texto;
    }

    get texto() {

        return this._texto;
    }

    set texto(texto) {

        this._texto = texto;
    }
}
```

Agora podemos usar a variável `let` e passaremos o texto dentro da `Mensagem()`.

```
let mensagem = new Mensagem('Flávio Almeida');
console.log(mensagem.texto)
```

A mensagem `Flávio Almeida` poderá ser visualizada no Console, quando executarmos o código. Mas e nos casos em que não sabemos qual será a mensagem do objeto `Mensagem()`? Qual será o valor padrão do texto? Teremos que passar como parâmetro uma *string* vazia.

```
class Mensagem {

    constructor(texto) {

        this._texto = texto;
    }

    get texto() {

        return this._texto;
    }

    set texto(texto) {

        this._texto = texto;
    }
}

let mensagem = new Mensagem('');
```

Para resolver a questão, o ES6 permite atribuir um valor padrão para parâmetros do `constructor()` ou de funções do JS. Se não passarmos no construtor da `Mensagem()` um texto, ele adotará como padrão uma *string* em branco.

```
class Mensagem {

    constructor(texto='') {

        this._texto = texto;
    }
//...
```

Mas se abaixo, adicionamos um texto e `Mensagem`, ele entende que não poderá usar o valor padrão. Vamos testar o código.

Antes, importaremos o arquivo `Mensagem.js` em `index.html`.

```
<script src="js/app/models/Negociacao.js"></script>
<script src="js/app/controllers/NegociacaoController.js"></script>
<script src="js/app/helpers/DateHelper.js"></script>
<script src="js/app/models/ListaNegociacoes.js"></script>
<script src="js/app/views/NegociacoesView.js"></script>
<script src="js/app/models/Mensagem.js"></script>
<script>
    let negociacaoController = new NegociacaoCOntroller();
</script>
```

Em seguida, digitaremos as seguintes linhas no Console do navegador:

```
let mensagem = new Mensagem();
undefined
mensagem.texto
""
```

Ao imprimirmos o `mensagem.texto`, o retorno é uma string em branco. Se colocarmos como valor padrão do `constructor()` o texto `Olá`, o código fica assim:

```
class Mensagem {

    Constructor(texto='Olá') {

        this._texto = texto;
    }
//...
```

Ao recarregarmos o Console, o retorno será:

```
let mensagem = new Mensagem();
undefined
```

```
mensagem.texto
"Olá"
```

Ele imprimiu o valor padrão. Mas se adotarmos como padrão outro texto, `Tchau!` , por exemplo, o retorno será diferente.

```
let mensagem = new Mensagem();
undefined
mensagem.texto
"Tchau!"
```

Este é um recurso interessante, porque podemos definir um parâmetro default, tanto no construtor quanto no método.