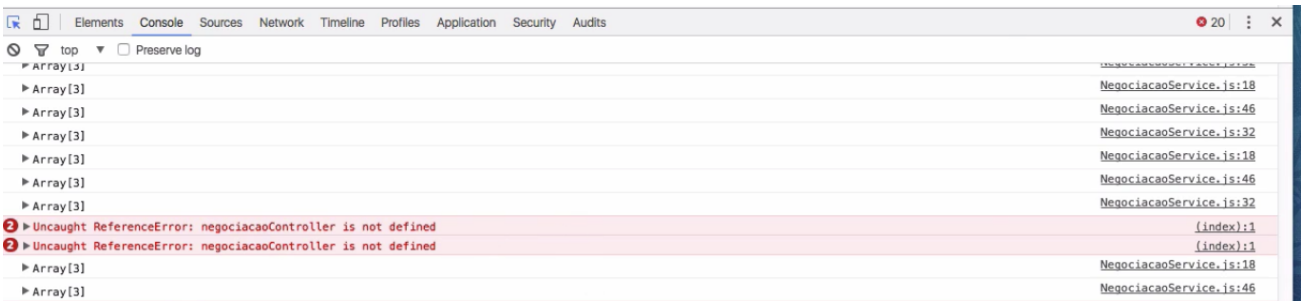
07

Delegação de eventos

Transcrição

Nossa aplicação parece estar funcionando, migramos para o sistema de módulos do ECMAScript 2015. No entanto, quando tentamos ordenar os dados das colunas, já não somos bem-sucedidos na tarefa. Se abrimos o Console, veremos diversas mensagens de que `NegociacaoController` não está definido.



Se analisarmos o `NegociacaoView`, veremos que adicionamos o evento `onclick` chamando o `negociacaoController`.

```
<thead>
    <tr>
        <th onclick="negociacaoController.ordena('data')">DATA</th>
        <th onclick="negociacaoController.ordena('quantidade')">QUANTIDADE</th>
        <th onclick="negociacaoController.ordena('valor')">VALOR</th>
        <th onclick="negociacaoController.ordena('volume')">VOLUME</th>
    </tr>
</thead>
```

Como ocorreu anteriormente, tivemos problemas porque o `NegociacaoController` não faz mais parte do escopo global. Para que a aplicação possa funcionar, teremos que encontrar alguma forma de importar o arquivo. No entanto, não podemos simplesmente importar o `NegociacaoController`.

```
import {View} from './View';
import {DateHelper} from '../helpers/DateHelper';
import {NegociacaoController} from '../controllers/NegociacaoController'

export class NegociacoesView extends View {

  constructor(elemento) {

      super(elemento);
  }
//...
```

Se fizermos isto, teremos um nova instância de negociação e isso nos trará problemas. Ao tentarmos ordenar as negociações, elas não terão os dados incluídos ou alterados. Isto significa que a solução é ter acesso a mesma instância. Como faremos isto? No arquivo `NegociacaoController.js`, não exportaremos mais a classe `NegociacaoController`. Iremos remover a palavra `export`:

```
class NegociacaoController {

    constructor() `{

        let $ = document.querySelector.bind(document);

        this._inputData = $('#data');
        this._inputQuantidade = $('#quantidade');
        this._inputValor = $('#valor');
//...
```

No entanto, se não exportamos, ninguém poderá usar seu conteúdo. A solução será exportar uma função que criaremos a seguir: `currentInstance()` :

```
let negociacaoController = new NegociacaoController();

export function currentInstance() {

    return negociacaoController;

}
```

Dentro do módulo, teremos a variável `negociacaoController` e que só existirá na nele. Quando a função for chamada, retornaremos a instância `negociacaoController` . Agora importaremos uma única instância. Faremos alterações em `boots.js` , adicionando o `currentInstance` :

```
import {currentInstance} from './controllers/NegociacaoController';
import {} from '/polyfill/fetch';

let negociacaoController = currentInstance();

document.querySelector('.form').onsubmit = negociacaoController.adiciona.bind(negociacaoControl
document.querySelector('[type=button]').onclick = negociacaoController.apaga.bind(negociacaoCont
```

A função nos retornará a instância de `negociacaoController` do módulo. Também importaremos o `currentInstance` em `NegociacaoView.js` .

```
import {View} from './View';
import {DateHelper} from '../helpers/DateHelper';
import {currentInstance} from '../controlers/NegociacaoController';

export class NegociacaoView extends View{

    constructor(elemento) {

        super(elemento);
    }
//...
```

Depois, trabalharemos com a **delegação de eventos**. Começaremos removendo o `onclick` do arquivo.

```
<thead>
    <tr>

      <th>DATA</th>
      <th>QUANTIDADE</th>
      <th>VALOR</th>
      <th>VOLUME</th>

    </tr>
</thead>
```

Quando clicarmos na coluna (na tag `<th>`), o JavaScript possui um sistema de eventos chamado ***event bubbling***. Com ele, ao clicarmos, o evento "subirá" até a tag `<tr> - que é o pai - e seguirá subindo até o`

`, perguntando se todos respondem pelo evento do clique. Nós vamos especificar que, se o evento de clique chegar até o` negociacoesView `, ele responderá ao evento da` `:

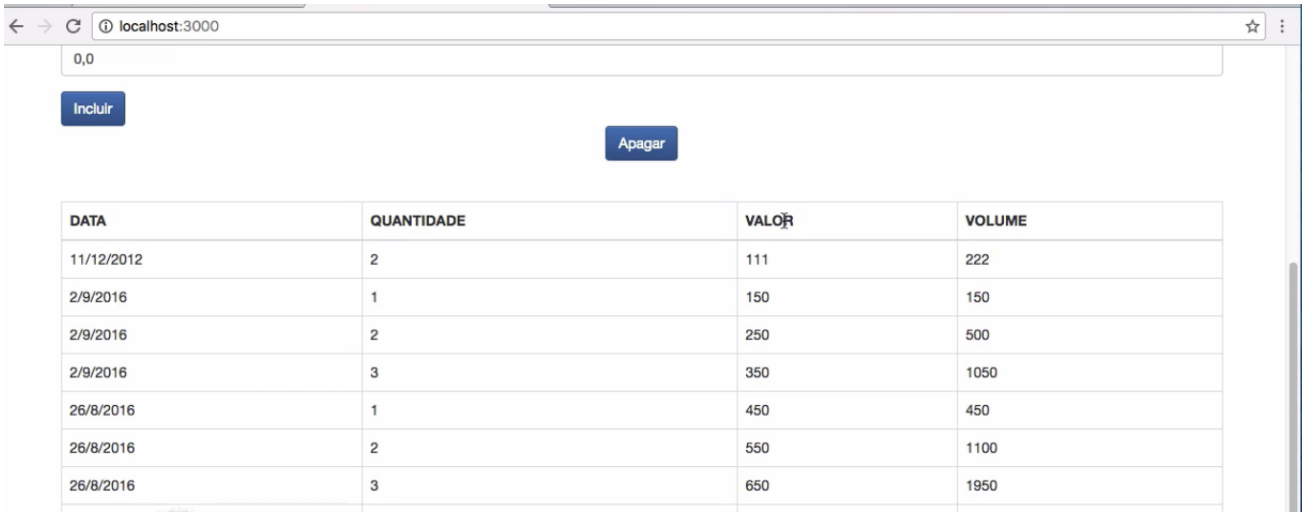
```
<div id="negociacoesView"></div>
```

Se não fizermos isto, cada `<th>` terá que responder pelo evento. Em `NegociacoesView`, vamos adicionar o `event.target` dentro do `if`:

```
export class NegociacoesView extends View {

    constructor(elemento) {

        super(elemento);

        elemento.addEventListener('click', function(event) {

            if(event.target.nodeName == 'TH') {

                currentInstance().ordena(event.target.textContent.toLowerCase());
            }
          })
    }
}
```

Nós adicionaremos o nome de cada `<th>` e vamos converter para cada caso (`toLowerCase()`). Se clicamos na `TH`, ela não estará preparada para o esforço... O evento subirá de hierarquia até chegar em `elemento`, então, perguntará se o alvo (`target`) era uma `TH`. Como a resposta será sim, pegaremos a instância atual da Controller, e chamaremos o `ordena()` passando o nome da `th` - indicado pelo `event.target.textContent`.

Se executarmos a página, veremos que o código funciona. Os dados também serão ordenados quando clicarmos em um coluna da tabela.



O sistema de módulos do ECMAScript 2015 também nos permite exportar instâncias, por meio da chamada da função `currentInstance`. Com isto, não temos mais nada quebrando na aplicação e terminamos o projeto com o código totalmente convertido.