03

Realizando o parse da resposta

Transcrição

Para termos acesso às requisições que foram retornadas pelo servidor, vamos usar a propriedade `xhr.responseText`. Ela também vai capturar a mensagem de erro vinda do servidor:

```
//...

    if(xhr.readyState == 4) {

        if(xhr.status == 200) {

            console.log('Obtendo as negociações do servidor.');
            console.log(xhr.responseText);
        }else{
            console.log('Não foi possível obter as negociações do servidor.');
            console.log(xhr.responseText);
        }
    }
};
xhr.send();
}
```

Em seguida, recarregaremos a página e veremos o que será exibido no Console:

Vemos um texto sendo exibido e, não, efetivamente um *array*. Isto acontece porque o `JSON` tem um formato textual para que os dados possam trafegar pela internet. Para transformarmos o JSON (texto) em um array de objetos, usaremos a função `JSON.parse()`, existente desde a versão anterior do JavaScript. Cada item do *array* é um objeto JavaScript. Mas nós poderíamos fazer como no exemplo abaixo?

```
if(xhr.readyState == 4) {
    if(xhr.status == 200) {

        this._listaNegociacoes.adiciona(JSON.parse(xhr.responseText));

    } else {

      console.log('Não foi possível obter as negociações do servidor.');
      console.log(xhr.responseText);
```

```
    }
}
```

Não poderíamos fazer isto, porque o `adiciona()` recebe uma negociação por vez. Na verdade, ela recebe uma instância de negociação. Se observarmos o *array* exibido no Console, **não** veremos uma instância de negociação, na verdade, cada item é um *object*. Primeiramente teremos que converter o JSON, fazendo o `parse()` e para cada item do *array*, teremos que criar uma negociação. Para realizarmos esta ação, usaremos a função `map()` - que varrerá o array e criará um novo com modificações. Usaremos uma *arrow function*:

```
if(xhr.readyState == 4) {
    if(xhr.status == 200) {

        JSON.parse(xhr.responseText)
         .map(objeto => new Negociacao(objeto.data, objeto.quantidade, objeto.valor));

    } else {

      console.log('Não foi possível obter as negociações do servidor.');
      console.log(xhr.responseText);
    }
}
```

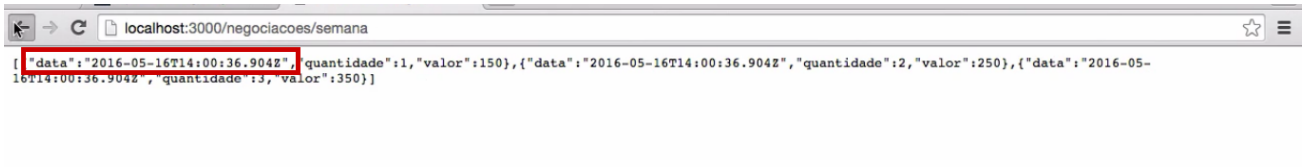
Os itens serão chamados de `objeto` e para cada um deles, retornaremos uma `Negociacao()`. Como trabalhamos com uma *arrow function*, não precisaremos usar o `return`. Da lista que foi criada pelo `map`, vamos iterar com o `forEach()`, e para cada item teremos um negociação. A seguir, vamos incluir `this._listaNegociacoes`:

```
if(xhr.readyState == 4) {
    if(xhr.status == 200) {

        JSON.parse(xhr.responseText)
         .map(objeto => new Negociacao(objeto.data, objeto.quantidade, objeto.valor))
         .forEach(negociacao => this._listaNegociacoes.adiciona(negociacao));

    } else {

      console.log(xhr.responseText);
      this._mensagem.texto = 'Não foi possível obter as negociações da semana';
    }
}
```

Vamos entender o que foi feito: `xhr.responseText` é um texto, o `JSON()` irá transforma-lo para o formato textual para objeto. Para cada um dos objetos listados, converteremos em uma instância de negociação, no fim, um novo array será gerado. Depois, com o `forEach()`, percorreremos cada item e adiciono na minha lista de negociações. Dentro do `else` alteramos as mensagens no caso de erro. No console, será exibido o `xhr.responseText` e para o usuário, será mostrada uma mensagem de alto nível.

Mas quando executarmos o código como está, teremos problemas.

Se acessarmos o endereço `localhost:3000/negociacoes/semana`, veremos que a data está no formato de uma string um pouco diferente.

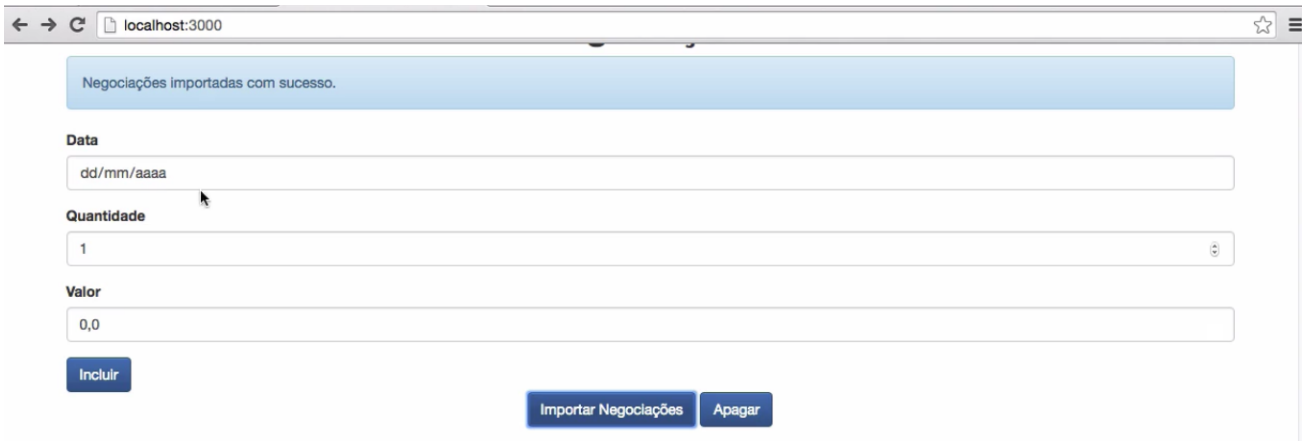


Nós estamos tentando passar `objeto.data` (com formato de texto) diretamente para `Negociacao`. A seguir, vamos passar a string `objeto data` para um construtor de `Date()`.

```
if(xhr.status == 200) {
        JSON.parse(xhr.responseText)
      .map(objeto => new Negociacao(new Date(objeto.data), objeto.quantidade, objeto.valor))
      .forEach(negociacao => this._listaNegociacoes.adiciona(negociacao));
}
```

Então, vamos instanciar uma nova data, baseada na string `objeto.data`. Recarregaremos a página do formulário e tudo funcionará corretamente. Para exibirmos uma mensagem de sucesso, adicionaremos o `this._mensagem.texto`. Faremos ajustes na mensagem de erro também.

```
if(xhr.status == 200) {
      JSON.parse(xhr.responseText)
        .map(objeto=> new Negociacao(new Date(objeto.data), objeto.quantidade, objeto.valor))
        .forEach(negociacao => this._listaNegociacoes.adiciona(negociacao))
        this._mensagem.texto = 'Negociações importadas com sucesso.';
  } else {
      console.log(xhr.responseText);
      this._mensagem.texto = 'Não foi possível obter as negociações.';
  }
}
```



A mensagem será exibida corretamente e podemos considerar esta primeira parte finalizada. Depois, faremos uma pequena brincadeira para vermos o que acontece em um caso de erro. Mudaremos o endereço no `importaNegociacoes()` para `negociacoes/xsemana`:

```
importaNegociacoes() {

    let xhr = new XMLHttpRequest();
    xhr.open('GET', 'negociacoes/xsemana');

//...
```

Neste caso, quando tentarmos recarregar a página, o usuário verá a seguinte mensagem de erro:

No Console, veremos a mensagem de log:

Conseguimos implementar a nossa solução com AJAX, usando JavaScript "puro" e sem utilizar bibliotecas como jQuery.