

## Resolvendo um problema com o paradigma funcional

### Transcrição

Talvez você tenha encontrado uma solução diferente da minha, mas darei a minha "versão dos fatos". Nosso objetivo é que o `Date` receba no construtor o `ano`, `mês` e `dia`. Faremos alguns ajustes no código:

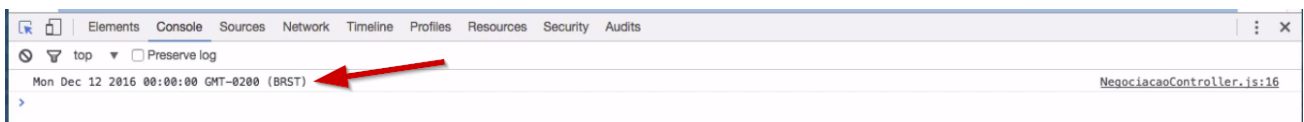
```
adiciona(event) {  
  
    event.preventDefault();  
  
    let data = new Date(  
        this._inputData.value  
    );  
    console.log(data);  
}
```

Lembrando que cada um dos itens será um parâmetro do construtor. Mas o mês tem que ser decrescido de `1`. Em seguida, transformaremos a *string* em um *array*, utilizando o `split('-')`. Ao fazermos isto, a string `2016-11-12` se tornará um *array* de três elementos. Sabemos que se passarmos da forma como já está, conseguiremos o resultado desejado. Porém, não quero que o `Date` receba um *array*, queremos que ele receba a primeira posição do *array* como a primeira posição do construtor e que o processo se repita com o segundo e terceiro elemento do *array*.

No ES6, temos o recurso **spread operator**. Observe o que faremos no código:

```
adiciona(event) {  
  
    event.preventDefault();  
  
    let data = new Date(...this._inputData.value.split('-'));  
    console.log(data);  
}
```

Adicionamos `...` (reticências) posicionado antes do `this`. Com este *spread operator*, indicamos que o *array* será desmembrado - e o primeiro item do *array*, e cada parâmetro do `Date` será posicionado na mesma ordem no construtor. Como já construímos o código, a data será passada, mas o mês ficará incorreto e não será subtraído `1`. Executaremos o código como está e a data aparecerá como o mês de `Dezembro`.



Mas conseguimos demonstrar que o *spread operator* está funcionando corretamente. Porém, temos que encontrar uma forma de, antes de reposicionarmos cada parâmetro para o construtor, decrementarmos `1` do valor do mês. Para isto, trabalharemos com a função `map()`, bem conhecida no mundo JavaScript e que nos permitirá subtrair `1`. Então, iremos chamar a função `map` no *array* criado e dependendo do elemento, iremos diminuir `-1`.

```
adiciona(event) {  
  
    event.preventDefault();  
  
    let data = new Date(...  
        this._inputData.value  
        .split('-')  
        .map(function(item) {  
            return item;  
        })  
    );  
    console.log(data);  
}
```

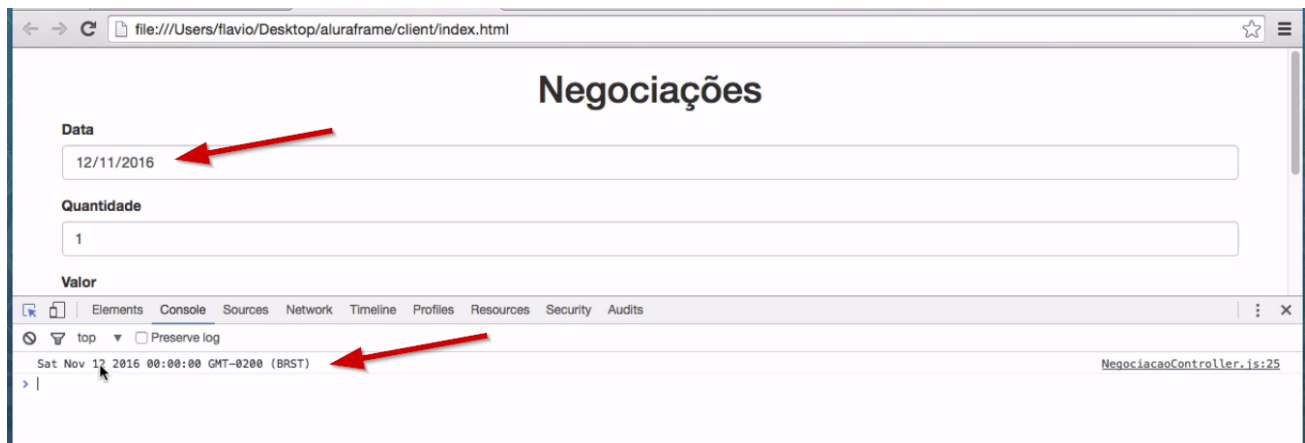
Com o `return item`, varremos o array onde temos ano, mês e dia, e recriaremos a mesma lista de elementos. Ou seja, não teremos nenhuma transformação. Se executássemos o código como ele está, a data continuaria com o mesmo problema. Nós precisamos que o mês `11` se torne `10`.

Continuaremos tentando resolver o problema, adicionando um segundo parâmetro na função `map()`: `indice`. Incluiremos um `if`, no qual especificaremos que quando passarmos pelo elemento `1` (o segundo) do array, vamos subtrair `1`.

```
adiciona(event) {  
  
    event.preventDefault();  
  
    let data = new Date(...  
        this._inputData.value  
        .split('-')  
        .map(function(item, indice) {  
            if(indice == 1) {  
                return item - 1;  
            }  
            return item;  
        })  
    );  
    console.log(data);  
}
```

Apesar de `item` ser uma string, quando usamos `-1`, o JS entende que deverá fazer uma operação numérica e faz a conversão implícita da string. Caso não seja o segundo elemento, a posição do mês, o retorno será equivalente ao valor do elemento do array.

Veremos se o código vai funcionar. No formulário, preencheremos o campo da data com `12/11/2016`. Desta vez, a data que aparecerá no Console estará correta.



Nós conseguimos, encadeando uma série de funções, converter só o segundo item do array e depois, decrementá-lo. Mas é possível eliminar o `if`. Em vez do laço, faremos alterações na função `map()`:

```
return item - indice % 2
```

O trecho do código ficará da seguinte forma:

```
//...
adiciona(event) {

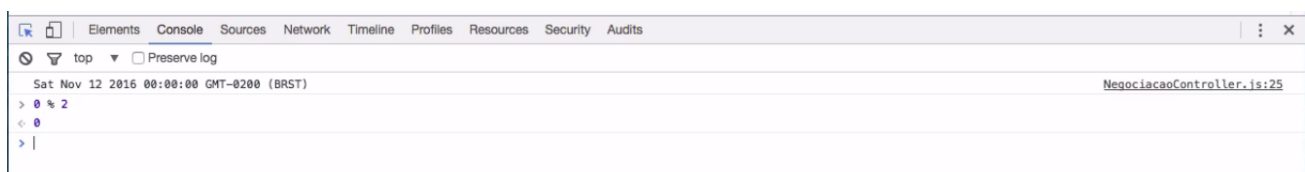
  event.preventDefault();

  let data = new Date(...this._inputData
    .value.split('-'))
    .map(function(item, indice) {
      return item - indice % 2;
    });
}
```

Se estamos na primeira posição do array, o valor de `indice` é `0`. Por isso, o resultado de `indice % 2` será igual a `0` também. Se subtrairmos este valor de `item`, nada irá mudar. Mas quando estivermos na segunda posição do array, o `indice` será igual a `1`. Agora, quando calcularmos `1` módulo de `2`, o resultado será `1`. E quando estivermos na terceira posição do array, `2` módulo de `2`, também será igual a `0`. Não diminuiremos nada do valor do item. Dessa forma conseguimos evitar a criação de um `if`.

Para entendermos melhor o que fizemos, calcularemos os módulos no Console:

```
0 % 2
```



A última linha retornou o valor do resultado: `0`, ou seja, a primeira posição do array. Se calcularmos os módulos dos demais, teremos os seguintes valores.



Desta forma, conseguimos evitar o `if` e ajustar apenas o mês. Até aqui, o nosso código ficou assim:

```
class NegociacaoController {

  constructor() {

    let $ = document.querySelector.bind(document);
    this._inputData = $('#data');
    this._inputQuantidade = $('#quantidade');
    this._inputValor = $('#valor');

  }

  adiciona(event) {

    event.preventDefault();

    let data = new Date(...this._inputData
      .value.split('-')
      .map(function(item, indice) {
        return item - indice % 2;
      }));

    let negociacao = new Negociacao(
      data,
      this._inputQuantidade.value,
      this._inputValor.value
    );

    console.log(negociacao);
  }
}
```

Se ele for executado, veremos que a data que surgirá no Console será 12 de novembro de 2016.