06

Ah se meu código funcionasse no Edge!

Até sua versão 13, o Microsoft Edge não possui o método `includes` de `Array`. Veja que isso nos causará problemas. Vejamos o código do nosso `ProxyFactory`:

```
class ProxyFactory {

    static create(objeto, props, acao) {

        return new Proxy(objeto, {

                get(target, prop, receiver) {

                    // uso includes aqui!
                    if(props.includes(prop) && ProxyFactory._ehFuncao(target[prop])) {

                        return function() {

                            console.log(`interceptando ${prop}`);
                            let retorno = Reflect.apply(target[prop], target, arguments);
                            acao(target);
                            return retorno;
                        }
                    }

                    return Reflect.get(target, prop, receiver);
                },

                set(target, prop, value, receiver) {

                    let retorno = Reflect.set(target, prop, value, receiver);

                    // uso includes aqui!
                    if(props.includes(prop)) acao(target);
                    return retorno;
                }
        });
    }

    static _ehFuncao(func) {

        return typeof(func) == typeof(Function);
    }
```

E agora? Bom, para resolver o problema do `includes` podemos criar um **polyfill** *extreme go horse* que será suficiente para resolver o problema. Mas o que é um polyfill?

Criando um polyfill

Um polyfill é um script que emula o comportamento de um recurso quando esse não é suportado para garantir que nosso código funcione sem termos que abdicar do que é mais novo.

Crie o arquivo `aluraframe/client/js/app/polyfill/es6.js`. Nele, vamos adicionar no `prototype` de `Array` o método `includes` que usa por debaixo dos panos o já conhecido `indexOf`. Mas, Flávio, é assim que o `includes` oficial é implementado? Não faço ideia, o importante é que o resultado final seja o mesmo, e usar o `indexOf` por debaixo dos panos resolve isso perfeitamente. Veja que o método só é adicionando se ele não existir:

```
// aluraframe/client/js/app/polyfill/es6.js

if(!Array.prototype.includes) {

    // Se não existir, adiciona

    console.log('Polyfill para Array.includes aplicado.');

    Array.prototype.includes = function(elemento) {
        return this.indexOf(elemento) != -1;
    };
}
```

Quando adicionamos métodos no `prototype` de uma classe ou função construtora, todas as instâncias dessa função construtora ou classe terão o método.

Agora vamos importar esse script no `head` da nossa página. Isso é necessário porque ele deve alterar `Array` antes que ele seja usado pela nossa aplicação. Alterando `aluraframe/client/index.html`:

```
<!-- aluraframe/client/index.html -->

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Negociações</title>
    <link rel="stylesheet" href="css/bootstrap.css">
    <link rel="stylesheet" href="css/bootstrap-theme.css">

    <!-- carregando nosso polyfill -->
    <script src="js/app/polyfill/es6.js"></script>

</head>
```

Excelente, mas isso não é suficiente. Sabe por quê? Porque o Edge 13 não suporta parâmetros opcionais, apenas o Edge 14 suportará. E agora?

Edge 13 não suporta parâmetros opcionais do ES6

Aqui não há como fazer um polyfill como o que fizemos sem termos que escrever muito código hack. Aqui eu deixo ao aluno decidir ou não se abdica do parâmetro opcional usando uma estratégia antiga de JavaScript. Que tal revisitarmos esse antigo truque?

Hoje nossa classe `Mensagem` é assim:

```
class Mensagem {

    // o valor padrão quando o parâmetro não é chamado é ''
    constructor(texto='') {

        this._texto = texto;
    }

    get texto() {

        return this._texto;
    }

    set texto(texto) {

        this._texto = texto;
    }
}
```

Precisamos alterá-la dessa forma para funcionar no `Edge` :

```
class Mensagem {

    // DEIXOU DE RECEBER O PARÂMTRO OPCIONAL
    constructor(texto) {

        this._texto = texto || ''; // se texto for undefined, vai passar ''
    }

    get texto() {

        return this._texto;
    }

    set texto(texto) {

        this._texto = texto;
    }
}
```

Pronto. Com essas alterações (e mais a solução que usamos para o Firefox do campo date) nosso código funcionará no Microsot Edge.