**Proxy Intercepta métodos?**

[Sugerir melhoria](https://cursos.alura.com.br/course/javascript-es6-orientacao-a-objetos-parte-2/task/17486)

Temos a seguinte declaração de classe:

class Pessoa {

constructor(nome) {

this.\_nome = nome;

}

get nome() {

return this.\_nome;

}

set nome(nome) {

this.\_nome = nome;

}

grita(frase) {

return `${this.\_nome} grita ${frase}`;

}

}

Criando uma instância e chamando o método grita:

let pessoa = new Pessoa('Barney');

pessoa.grita('Olá');

E se quisermos interceptar a chamada do método grita? A má notícia é que toda Proxy criada, por padrão, não esta preparada para interceptar métodos (getters e setters são exceções a este problema). Essa limitação ocorre porque sempre que um método de um objeto (que não deixa de ser uma propriedade que armazena uma função) é chamado, primeiro é realizado uma operação de leitura (get, do nosso handler da Proxy) e depois os parâmetros são passados através de Reflect.apply. O problema é que como o método é interceptado pelo get do handler passado para o Proxy não temos acesso aos seus parâmetros. E agora?

Uma solução é implementar o seguinte código:

let pessoa = new Proxy(new Pessoa('Barney'), {

get(target, prop, receiver) {

if(prop == 'grita' && typeof(target[prop]) == typeof(Function)) {

// essa função retornada irá substituir o método 'gita' no proxy!!! Ou seja, estamos usando o handler do proxy para modificar o próprio proxy, que loucura!

return function() {

console.log(`Método chamado: ${prop}`);

// Quando usarmos Reflect.apply, Reflect.get e Reflect.set precisamos retornar o resultado da operação com return

// arguments é uma variável implícita que dá acesso à todos os parâmetros recebidos pelo método/função

return Reflect.apply(target[prop], target, arguments);

}

}

// só executa se não for função

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

}

});

pessoa.grita('Olá');