09

Monkey Patch

Herculano criou a simples classe `Pessoa`, que recebe em seu construtor um nome e sobrenome. Inclusive, ele criou o método `obterNomeCompleto`, que retorna a junção do nome com o sobrenome, separados por um espaço.

```
class Pessoa {

    constructor(nome, sobrenome) {

        this.nome = nome;
        this.sobrenome = sobrenome;
    }

    obterNomeCompleto() {

        return `${this.nome} ${this.sobrenome}`;
    }
}

let pessoa1 = new Pessoa('Flávio', 'Almeida');
pessoa1.obterNomeCompleto();

let pessoa2 = new Pessoa('Almeida', 'Flávio');
pessoa2.obterNomeCompleto();
```

Contudo, em determinada parte do seu código, ele viu a necessidade de que o método `obterNomeCompleto` retornasse o nome e o complemento separados por um hífen. Ele lembrou das aulas de Proxy, mas achou que naquela situação teria muito trabalho em implementar algo como nosso `ProxyFactory`.

Ele lembrou que seu amigo Hugo costumava resolver esse problema através de *monkey patch*, que consiste em mudar o método ou função dinamicamente. Ele tem interesse em mudar o método de uma instância e não de todas.

Qual das opções abaixo Herculano realiza *monkey patch* do método `obterNomeCompleto` de `pessoa2`?

Selecione uma alternativa

A

```
let pessoa2 = new Pessoa('Almeida', 'Flávio');

pessoa2.obterNomeCompleto() = function() {

  return `${this.nome} - ${this.sobrenome}`;
}
console.log(pessoa2.obterNomeCompleto());
```

B

```
let pessoa2 = new Pessoa('Almeida', 'Flávio');

pessoa2.obterNomeCompleto = () => {
```

```
  return `${this.nome} - ${this.sobrenome}`;
}

console.log(pessoa2.obterNomeCompleto());
```

C

```
let pessoa2 = new Pessoa('Almeida', 'Flávio');

pessoa2.obterNomeCompleto = function() {

  return `${this.nome} - ${this.sobrenome}`;
}
console.log(pessoa2.obterNomeCompleto());
```