10

Resumindo

Transcrição

Vamos revisar o que estudamos até aqui: vimos a criação de uma classe, utilizando novos recursos JavaScript que favorecem a implementação do paradigma orientado a objeto. Vimos também que uma classe possui um `constructor` com os quais definimos atributos - que chamamos de **propriedades** - e que podemos materializar uma abstração do mundo real usando um modelo, por meio de uma classe. Outro assunto abordado é que podemos passar parâmetros no construtor de uma classe e dessa forma, garantindo que no momento em que a instância de uma classe é criada, já tenha todos os dados necessários.

Por convenção, adotamos que os atributos **privados** devem usar o prefixo `_` (*underline*), indicando para o desenvolvedor que ele só pode acessá-lo. Vimos como adicionar métodos nas classes, e estes, sim, podem acessar os atributos privados.

Apresentamos uma maneira de criar um atributo, que na prática é um método, e ao acessá-lo, podemos executar o código. Moral da história: temos um método que conseguimos acessar como uma propriedade, bastando ser antecedido pela palavra especial `get`, desta forma, estaríamos gerando um `getter`. Quem acessa a sua classe acredita que se trata de uma propriedade, mas na verdade, trata-se por "debaixo dos panos" de um método.

No entanto, isto não era suficiente para garantir a integridade da nossa negociação, que não pode ser alterada. Por isso, usamos o `Object.freeze()` para congelar um objeto depois de criado. Como `Object.freeze()` é *shallow* (raso), ele será aplicado nas propriedades do objeto, mas as propriedades que são objetos não serão todas congeladas. A ação ficará apenas na superfície. Para resolver esta questão, falamos um pouco sobre programação defensiva. Quando alguém tentar acessar a data, nós retornaremos uma nova data. Fizemos o mesmo com o construtor e com isso, evitamos que alguém consiga de fora da classe alterar algum item do estado interno.

O que vimos foi relevante porque o modelo é uma das coisas mais importantes quando desenvolvemos o sistema. Agora que temos o modelo pronto, a aplicação poderá crescer, tendo-o como base. No fim, deixamos a sugestão da adoção de um novo hábito: substituir nas declarações de variáveis o uso de `var` por `let`, que permite um escopo de bloco e evita que as mesmas vazem para um escopo global. Antes do ES6, em JavaScript, era comum o uso de funções para a criação de um escopo para a variável.

Vamos continuar com os nossos estudos e façam os exercícios para praticar os conceitos vistos.