01

Modelo e reutilização em projetos

Transcrição

Você pode fazer o download (https://github.com/alura-cursos/javascript-avancado-ii/archive/aula1.zip) completo até aqui e continuar seus estudos.

Para liberarmos o desenvolvedor da responsabilidade de atualizar programaticamente a View sempre que o modelo fosse atualizado, nós colocamos "armadilhas": funções que eram chamadas quando métodos específicos eram executados. Desta forma, chamávamos automaticamente a atualização da View. Nós declaramos o modelo no inicio, definimos as ações que deveriam acontecer quando ocorria a modificação, e assim liberamos o desenvolvedor da responsabilidade.

No entanto, esta solução deixa a desejar porque coloca código de infraestrutura - ou seja, de atualização da View - no modelo. Geralmente, a parte mais reutilizada de um sistema é o modelo. Então, ao acessarmos um modelo de negociação e encontrarmos um atributo chamado `_armadilha`, por exemplo:

```
class ListaNegociacoes {

    constructor(armadilha) {

        this._negociacoes = [];
        this._armadilha = armadilha;
    }
//...
```

O que `_armadilha` tem a ver com a lista de negociação? Ela foi usada apenas para aplicar o artifício que chama a View automaticamente. E se tivéssemos outros métodos que quiséssemos monitorar e executar uma armadilha? Teríamos que alterar a classe do modelo. Então, o modelo é a parte mais reutilizável. Se agora não quisermos mais utilizar um sistema baseado em MVC, podemos optar em usar o AngularJS ou outro framework.

Mas se começamos a incluir diversos itens de infraestrutura, de recursos para que ela gere benefícios - como a atualização de View - começamos a não reutilizar continuamente o modelo. Encontraremos uma forma de manter o modelo intacto, sem utilizarmos armadilhas e ainda assim, conseguir executar um código arbitrário quando algum método for chamado. A seguir, encontraremos uma solução para a questão.