04

Implementando um EventEmitter

Transcrição

Vamos criar o módulo `app/utils/event-emitter.js`. Utilizarmos um `Map` para associar um **event** a todos os seus **listeners**. Como a chave do `Map` será o `event`, não corremos o risco de criar uma nova chave em nosso `Map` para um `event` já existente:

```
// app/utils/event-emitter.js

const events  = new Map();
```

Pronto. Um ponto a destacar é que `events` não será uma propriedade do objeto `EventEmitter` que criaremos. A variável `events` vive no escopo do módulo. Essa abordagem evitará que outra parte do nosso código, que não seja o próprio `EventEmitter` interaja e bagunce com `events`. Em outras palavras, estamos encapsulando a mapa de eventos.

Vamos exportar o objeto `EventEmitter` que terá os métodos `on` e `emit`:

```
const events  = new Map();

export const EventEmitter = {

    on(event, listeners) {
        /* falta implementar */
    },

    emit(event, data) {
        /* falta implementar */
    }
};
```

Vamos começar pela implementação do método `on`. A primeira coisa que ele fará é verificar se o `event` que desejamos escutar já existe. Se não existir, criamos uma chave em nosso `Map` com o nome do `event` e guardamos um array vazio que armazenará todos os seus `listeners`. É a partir dessa garantia que podemos adicionar o `listener` ao `event`:

```
const events  = new Map();

export const EventEmitter = {

    on(event, listener) {
        if (!events.has(event)) events.set(event, []);
        events.get(event).push(listener);
    },

    emit(event, data) {
        /* falta implementar */
    }
};
```

Só falta o método `emit`. Quando for chamado, ele obterá a lista de `listeners` para o `event` passado como parâmetro. Se o `event` ainda não foi criado por nenhum `listener`, nada acontecerá. Cada `listener` associado ao `event` será chamado recebendo como parâmetro o dado recebido pela função `emit`. Esse passo é fundamental, caso contrário os `listeners` não terão acesso ao dado emitido com o `event`:

```
const events  = new Map();

export const EventEmitter = {

    on(event, listener) {
        if (!events.has(event)) events.set(event, []);
        events.get(event).push(listener);
    },

    emit(event, data) {
        const listeners = events.get(event);
        if (listeners) listeners
            .forEach(listener => listener(data));
    }
};
```

Chegou a hora de utilizarmos nosso `EventEmitter`.