Marionette: функции

Marionette предоставляет ряд полезных функций, которые обеспечивают общее поведение объектов в рамках всего фреймворка.

- Marionette.extend
- Marionette.getOption
- Marionette.triggerMethod
- Marionette.bindEntityEvent

Marionette.extend

Очень полезна функция `extend`, присутствующая в Backbone. Она нашла свое применение и в различных местах кода Marionette. Для совпадения функций и во избежание двусмысленности, Marionette.extend является псевдонимом Backbone.extend.

```
var Foo = function(){};
// используем Marionette.extend для того, чтобы сделать объект Foo наследуемым, наподобие других Backbone и Marionette объектов
Foo.extend = Marionette.extend;

// Теперь Foo может быть наследован (расширен), для создания экземпляров объекта нового типа, с рядом новых методов
var Bar = Foo.extend({
 someMethod: function(){ ... }
  // ...
 });
// создаем экземпляр Bar
var b = new Bar();
```

Marionette.getOption

Дает возможность получить атрибуты объекта или непосредственно из самого объекта, или из `this.options`, если при создании экземпляра объекта они были заданы.

```
var M = Backbone.Model.extend({
 foo: "bar",
 initialize: function(){
  var f = Marionette.getOption(this, "foo");
  console.log(f);
 }
});

new M(); // => "bar"
new M({}, { foo: "quux" }); // => "quux"
```

Полезно, когда при построении объекта в конструктор передаются инициализирующие аргументы.

Свойства объекта с логическим значением false (Falsey values)

Функция `getOption` будет пытаться прочесть значения полей, приводимых к логическому false, непосредственно из объекта.

Пример:

```
var M = Backbone.Model.extend({
 foo: "bar",
 initialize: function(){
  var f = Marionette.getOption(this, "foo");
  console.log(f);
 }
});

new M(); // => "bar"

var f;
new M({}, { foo: f }); // => "bar"
```

В обоих случаях возвращается "bar", поскольку во втором случае `f` имеет неопределенное значение.

Marionette.triggerMethod

Порождает событие и запускает соответствующий метод на целевом объекте.

Когда событие порождено, происходит следующее: к его названию, возведенному в верхний регистр по первой букве, добавляется приставка "on". Например:

* `triggerMethod("render")` будет пытаться запустить функцию "onRender"
* `triggerMethod("before:close")` будет пытаться запустить функцию "onBeforeClose"

Все аргументы, переданные в triggerMethod, будут транслированы одновременно как в событие, так и в соответствующую функцию - за тем исключением,
что название события не передастся в функцию.

`triggerMethod("foo", bar)` вызовет `onFoo: function(bar){...})`

Marionette.bindEntityEvents

Этот метод привязывает backbone'овскую "сущность" (collection/model) к методу целевого объекта.

```
Backbone.View.extend({
 modelEvents: {
  "change:foo": "doSomething"
 },
 initialize: function(){
  Marionette.bindEntityEvents(this, this.model, this.modelEvents);
 },
 doSomething: function(){
  // событие "change:foo" произошедшее в модели, запустит код, находящийся здесь
 }
});
```

Первый параметр, `target`, должен иметь метод `listenTo` из объекта EventBinder.

Второй параметр - сущность (Backbone.Model или Backbone.Collection), к которой привязывается событие.

Третий параметр - хеш из конфигурации { "event:name": "eventHandler" }. Если событию соответствует множество обработчиков, они должны быть разграничены пробелом. Вместо строкового названия обработчика может стоять функция.

(Методу Marionette.bindEntityEvents сопоставлен метод Marionette.unbindEntityEvents, но в документации, по крайней мере пока, от этом не говорится. Однако, можно найти упоминание в annotated source - прим. пер.)