Marionette.Layout

`Layout` - это специальный гибрид, представляющий собой нечто промежуточное между `ItemView` и коллекцией из `Region` объектов, который используется для формирования макета приложения со множеством суб-регионов, которые будут управляться с помощью определенных Marionette.RegionManager.

Layout может быть включен в приложение в качестве управляющей структуры, и использован вместо `composite-view` для работы с областями экрана, относящимися к подчиненным приложениям (sub-application), - так реализуется отображение множества представлений внутри соответствующих регионов, посредством динамически отображаемого HTML.

Базовое применение

`Layout` происходит непосредственно от `ItemView` и отличается возможностью задавать прикрепленные к layout-у регионы - экземпляры объекта `Region`.

```
<script id="layout-template" type="text/template">
  <section>
    <navigation id="menu">...</navigation>
    <article id="content">...</article>
  </section>
</script>
```

```
AppLayout = Backbone.Marionette.Layout.extend({
  template: "#layout-template",
  regions: {
    menu: "#menu",
    content: "#content"
  }
});
var layout = new AppLayout();
layout.render();
```

Поскольку layout определен, возможен прямой доступ ко всем зарегистрированным в нем регионам, играющим роль region managers.

```
layout.menu.show(new MenuView());
layout.content.show(new MainContentView());
```

Задание Regions посредством функции

Регионы могут быть заданы внутри Layout-а посредством функции, возвращающей объект с дифиницией региона. Данный объект определяет регион по тем же правилам, что и описанный выше объектный литерал.

```
Marionette.Layout.extend({
  // ...
  regions: function(options){
    return {
      fooRegion: "#foo-element"
    };
  },
  // ...
});
```

Обратите внимание, что функция принимает `options` в качестве аргумента, которые были переданы в конструктор view. Когда регион впервые инициализован, значение `this.options` еще не доступно, так что options должны быть получены из этого параметра.

Доступ к региону

Любой определенный внутри layout-а регион будет доступен этому layout-у или любому другому коду сразу после того, как layout был инициализован.
Благодаря этому любой layout может быть вставлен в любой DOM-элемент на HTML странице, со всеми определенными в нем регионами, готовыми к работе, без необходимости вызова метода render() или каких-либо других дополнительных действий.

Однако, следует заметить, что регион может быть наполнен содержанием только в том случае, если layout имеет доступ к элементу, определенному при дифиниции региона.
То-есть это значит, что если представление еще не отрендерилось, регионы могут не найти соответствующие им DOM-элементы, и связанный с ними сценарий не внесет никаких изменений в DOM.

Повторный рендеринг layout-а

Layout может быть отрендерен столько раз, сколько это необходимо, но инициализирующий render и все последующие имеют в своем поведении некоторые отличия.

Первый раз, когда layout отрендерен, ничего особенного не происходит. Единственно - вызывается метод render() прототипа `ItemView` на данном layout-е.

В следующий раз, когда нужно будет отрендерить layout, метод render(), кроме вышеназванного, запустит код первичной или повторной инициализации регионов.

После первого рендеринга каждый последующий будет принудительно вызывать метод close() для каждого региона, закрывая его.
Это также повлечет закрытие каждого присутствующего (показанного, отрисованного) в регионе представления (view), и каждого связанного с этим представлением sub-view. Как только регионы окажутся закрыты,
все регионы будут сброшены при помощи свойственного им метода reset() , чтобы они больше не ссылались на элементы предыдущего рендеринга.

И уже когда layout закончит процесс собственного ре-рендеринга, вызов метода show для каждого региона и связанный с ним показ представления
приведет к тому, что регион свяжет себя с новым элементом в layout-е.

Как избежать ререндеринга целого layout-а

Случается, что необходим повторный рендеринг всего layout-а. Однако, как следует из вышеописанного, это может привести к

чрезмерно большому количеству необходимой
для полной перезагрузки layout-а и всех отображенных в нем представлений работы.

Это значит, что вы должны избегать ре-рендеринга целого layout-а без крайней необходимости. Вместо этого, вам следует привязать шаблон layout-а к модели и, прослушивая события изменения модели,
обновлять layout небольшими порциями, работая только с необходимыми DOM-элементами.

Встроенные Layouts и Views

Поскольку `Layout` происходит непосредственно от `ItemView`, он наследует всю базовую функциональность `ItemView`.
Что включает в себя и все методы, необходимые для показа внутри существующего region manager-а. (???)

```
MyApp = new Backbone.Marionette.Application();
MyApp.addRegions({
 mainRegion: "#main"
});

var layout = new AppLayout();

MyApp.mainRegion.show(layout);

layout.show(new MenuView());
```

Ограничений на глубину встраивания layout-в в region managers нет, это дает возможность создавать структуры представлений любой степени сложности.

(См. также статью автора фреймворка "Managing Layouts And Nested Views With Backbone.Marionette")

Закрытие экземпляра layout-а

Когда вы заканчиваете работу с layout-ом, вы можете вызвать его метод `close`. Это будет гарантировать, что все region managers внутри layout-а будут корректно выключены,
что в свою очередь гарантирует, что все views, показанные внутри регионов (см. метод show() региона) будут выключены (закрыты) корректно.

Если вы показываете layout внутри родительского region manager-а, замена layout-а другим layout-ом или другим view закроет текущий layout точно так же, как закрыл бы view.

Вышеописанное поведение гарантирует корректное закрытие layout-в и содержащихся в них представлений.

Пользовательские типы регионов

Если вам необходимо заменить `Region` с помощью реализованного вами класса, вы можете указать этот класс посредством свойства `regionType` Layout-а:

```
MyLayout = Backbone.Marionette.Layout.extend({
  regionType: SomeCustomRegion
});
```

Вы можете также задавать пользовательские `Region`-классы для любого региона, определенного в Layout-е:

```
AppLayout = Backbone.Marionette.Layout.extend({
  template: "#layout-template",
  regionType: SomeDefaultCustomRegion,
  regions: {
    menu: {
      selector: "#menu",
      regionType: CustomRegionTypeReference
    },
    content: {
      selector: "#content",
      regionType: CustomRegionType2Reference
    }
  }
});
```

Добавление и удаление регионов

Регионы могут добавляться и удаляться по мере необходимости в экземпляре Layout-а с помощью следующих методов:

- addRegion
- addRegions
- removeRegion

addRegion:

```
var layout = new MyLayout();
// ...
layout.addRegion("foo", "#foo");
layout.foo.show(new someView());
```

addRegions:

```
var layout = new MyLayout();
// ...
layout.addRegions({
  foo: "#foo",
  bar: "#bar"
});
```

removeRegions:

```
var layout = new MyLayout();
// ...
layout.removeRegion("foo");
```

Любой регион может быть удален, независимо от того, был ли он определен в аттрибутах региона или в его дефиниции, или добавлен позже.

За дальнейшей информацией по использованию этих методов, следует обращаться к документации по `Marionette.regionManager`.