Marionette.CollectionView

`CollectionView` проходит циклом по всем моделям, представленным в данной коллекции,
отображает их посредством определенного `itemView`, и все `el` из `itemView`, с отрендеренным в них html, добавляет в `el`
`collectionView`.

CollectionView: `itemView`

Задает `itemView` для `collectionView`, и является не конкретным экземпляром, а объектом, выводимым (extended) из `Backbone.View`
или `Marionette.ItemView`.

```
MyItemView = Backbone.Marionette.ItemView.extend({});
Backbone.Marionette.CollectionView.extend({
 itemView: MyItemView
});
```

В качестве альтернативы, можно определить `itemView` в конструкторе:

```
MyCollectionView = Backbone.Marionette.CollectionView.extend({...});
new MyCollectionView({
 itemView: MyItemView
});
```

Если `itemView` не определен, будет брошено исключение с соответствующим предупреждением.

Если выбор itemView зависит от данных модели, `getItemView` может быть переопределен:

```
Backbone.Marionette.CollectionView.extend({
 getItemView: function(item) {
  // some logic to calculate which view to return
  return someItemSpecificView;
 }
})
```

CollectionView: `itemViewOptions`

Встречаются случаи, когда необходимо передать данные из родительского CollectionView в каждый экземпляр itemView. Для этого в

CollectionView нужно задать `itemViewOptions` в качестве объектного литерала, тогда эти данные добавятся к `options` конструктора при создании экземпляра itemView.

```
ItemView = Backbone.Marionette.ItemView({
  initialize: function(options){
    console.log(options.foo); // => "bar"
  }
});
CollectionView = Backbone.Marionette.CollectionView({
  itemView: ItemView,
  itemViewOptions: {
    foo: "bar"
  }
});
```

Вы можете также задать в качестве `itemViewOptions` функцию, тогда объектный литерал будет вычислен и возвращен на этапе исполнения кода. Одним из параметров для этой функции является модель, поскольку логика вычисления `itemViewOptions` основывается на данных модели. Атрибуты возвращенного объекта добавятся к `options` конструктора при создании экземпляра itemView.

```
CollectionView = Backbone.Marionette.CollectionView({
  itemViewOptions: function(model, index) {
  // производим вычисления на основании данных модели
   return {
    foo: "bar",
    itemIndex: index
   }
  }
});
```

Атрибут `emptyView` CollectionView

Если коллекция еще (или уже) пуста, и вам необходимо вывести представление, отличающееся от списка отдельных itemView, то вы можете определить атрибут `emptyView` в CollectionView.

```
NoItemsView = Backbone.Marionette.ItemView.extend({
  template: "#show-no-items-message-template"
});
Backbone.Marionette.CollectionView.extend({
  // ...
  emptyView: NoItemsView
});
```

В этом примере `emptyView` отобразится на месте списка, если тот пуст.

Атрибут `buildItemView` CollectionView

Если вам нужно создать экземпляр пользовательского представления для `itemView`, переопределите метод `buildItemView`. Этот метод принимает три параметра: item, ItemViewType, itemViewOptions.

```
buildItemView: function(item, ItemViewType, itemViewOptions){
  // формируем конечный список параметров для item view, который будет выстраиваться по определенному типу
  var options = _.extend({model: item}, itemViewOptions);
  // создаем новый экземпляр
  var view = new ItemViewType(options);
  // возвращаем его
  return view;
},
```

Методы обратного вызова

`CollectionView` может предоставлять ряд коллбэков.

onBeforeRender

`onBeforeRender` вызовется непосредственно перед рендерингом представления.

```
 Backbone.Marionette.CollectionView.extend({
  onBeforeRender: function(){
    // делаем что-то
  }
});
```

onRender

Сразу после того, как представление отреднерено, будет вызван метод `onRender`.
Вы можете реализовать его в представлении, например, чтобы взаимодействовать с объектом представления `el`, который также уже будет отреднерен.

```
Backbone.Marionette.CollectionView.extend({
 onRender: function(){
   // делаем что-то
 }
});
```

onItemAdded

Данный коллбэк уведомляет, что к collection view был добавлен элемент, либо экземпляр item view. Внутри него мы получаем доступ к соответствующему элементу коллекции экземпляру itemView.

```
Backbone.Marionette.CollectionView.extend({
 onItemAdded: function(itemView){
   // делаем что-то с экземпляром itemView
 }
});
```

onBeforeClose

Данный метод вызывается прямо перед закрытием представления.

```
Backbone.Marionette.CollectionView.extend({
 onBeforeClose: function(){
   // делаем что-то
 }
});
```

onClose

Данный метод вызывается сразу после закрытия представления.

```
Backbone.Marionette.CollectionView.extend({
 onClose: function(){
   // делаем что-то
 }
});
```

События CollectionView

В течение жизненного цикла collection view порождается ряд событий.
Каждое из этих событий генерится с помощью функции Marionette.triggerMethod, которая одновременно вызывает соответствующий событию метод "on{EventName}".

Событие "before:render" или метод onBeforeRender

Порождается непосредственно перед рендерингом представления. Синонимом данного события является событие "collection:before:render" , которому соответствует метод `onCollectionBeforeRender`.

```
MyView = Backbone.Marionette.CollectionView.extend({...});
var myView = new MyView();
myView.on("before:render", function(){
  alert("the collection view is about to be rendered");
});
myView.render();
```

Событие "render" или метод onRender

Порождается сразу после рендеринга представления. Синонимом данного события является событие "collection:rendered", которому соответствует метод `onCollectionRendered`.

```
MyView = Backbone.Marionette.CollectionView.extend({...});
var myView = new MyView();
myView.on("render", function(){
  alert("the collection view was rendered!");
});
myView.on("collection:rendered", function(){
  alert("the collection view was rendered!");
});
myView.render();
```

Событие "before:close" или метод onBeforeClose

Порождается непосредственно перед закрытием представления. Одновременно с ним породится событие "collection:before:close", которому соответствует метод `onCollectionBeforeClose`.

```
MyView = Backbone.Marionette.CollectionView.extend({...});
var myView = new MyView();
myView.on("collection:before:close", function(){
  alert("the collection view is about to be closed");
});
myView.close();
```

Событие "closed" / "collection:closed"

Порождается сразу после закрытия представления, с вызовом соответствующих методов.

```
MyView = Backbone.Marionette.CollectionView.extend({...});
var myView = new MyView();
myView.on("collection:closed", function(){
  alert("the collection view is now closed");
});
myView.close();
```

(TODO: проверить существование события "closed" - прим.пер.)

События "before:item:added" / "after:item:added"

Событие "before:item:added" (которому соответствует метод `onBeforeItemAdded`) порождается сразу после создания нового экземпляра itemView для элемента, добавленного в коллекцию, но перед тем как представление будет отрендерено и добавлено в DOM.

Событие "after:item:added" (которому соответствует метод `onAfterItemAdded`) порождается сразу после рендеринга представления.

```
var MyCV = Marionette.CollectionView.extend({
  // ...
  onBeforeItemAdded: function(){
    // ...
  },
  onAfterItemAdded: function(){
    // ...
  }
});
var cv = new MyCV({...});
cv.on("before:item:added", function(viewInstance){
  // ...
});
cv.on("after:item:added", function(viewInstance){
  // ...
});
```

Событие "item:removed" или метод onItemRemoved

Событие "item:removed" порождается сразу после того, как экземпляр itemView будет закрыт или удален (ищутся методы 'close' или 'remove'), что произойдет в свою очередь после удаления соответствующего ему элемента из коллекции.

```
 cv.on("item:removed", function(viewInstance){
  // ...
});
```

Всплытие события "itemview:*" из встроенных представлений

Когда item view внутри collection view порождает какое-либо событие, оно всплывает сквозь родительский collection view, и название события предваряется конструкцией "itemview:" заданной по умолчанию.

То-есть, если встроенное представление сгенерит "do:something", родительское представление сгенерит соответственно "itemview:do:something".

```
// задаем базовую коллекцию
var myModel = new MyModel();
var myCollection = new MyCollection();
myCollection.add(myModel);

//  создаем и рендерим collection view
colView = new CollectionView({
  collection: myCollection
});
colView.render();

// привязываем всплывающие события
colView.on("itemview:do:something", function(childView, msg){
  alert("I said, '" + msg + "'");
});

// это хак - просто чтобы сгенерить событие для нашего примера
var childView = colView.children[myModel.cid];
childView.trigger("do:something", "do something!");
```

В результате будет выведено предупреждение с текстом "I said, 'do something!'".

Еще раз повторим, что в данном примере мы использовали хак с получением прямой ссылки на встроенное представление просто для генерации события, обычно все обстоит иначе - генерация событий в item view обусловлена его подпиской на прослушивание DOM-событий или событий, связанных с изменением модели.

CollectionView: `itemViewEventPrefix`

Префикс события itemView, всплывающего сквозь collection view, может быть кастомизирован. Для этого нужно задать атрибут `itemView EventPrefix` в экземпляре collection view.

```
var CV = Marionette.CollectionView.extend({
  itemViewEventPrefix: "some:prefix"
});
var c = new CV({
  collection: myCol
});
c.on("some:prefix:render", function(){
  // item view был отрендерен, поэтому что-то делаем
});
c.render();
```

`itemViewEventPrefix` может быть определен в дефиниции представления или в вызове конструктора.

Рендеринг CollectionView

Метод `render` collection view отвечает за отображение всей коллекции. Он проходит циклом по всем элементам коллекции и отображает их индивидуально в `itemView`.

```
MyCollectionView = Backbone.Marionette.CollectionView.extend({...});
new MyCollectionView().render().done(function(){
  // все встроенные представления к этому моменту уже отобразились, делаем что-то
});
```

CollectionView: автоматический рендеринг

Collection view прослушивает события "add", "remove" и "reset" коллекции.

Как только коллекция будет переустановлена (произойдет событие "reset"), представление вызовет собственный метод `render` и заново отобразит обновленную коллекцию.

Когда в коллекцию будет добавлена модель, collection view отобразит данную модель в наборе item views.

Когда модель будет удалена из коллекции, collection view закроет или удалит соответствующий item view.

CollectionView: повторное отображение коллекции

Если необходимо перерендерить целиком всю коллекцию, можно вызвать метод `view.render`.
Этот метод позаботится о закрытии всех встроенных представлений, которые были открыты ранее.

CollectionView: appendHtml

По умолчанию collection view вызывает метод jQuery `.append`, чтобы добавить HTML-содержимое экземпляра item view к объекту `el` collection view.

Это поведение может быть переопределено в методе `appendHtml` в конкретном представлении.
Данный метод принимает три параметра и не возвращает значений.

```
Backbone.Marionette.CollectionView.extend({
 // The default implementation:
  appendHtml: function(collectionView, itemView, index){
    collectionView.$el.append(itemView.el);
  }
});
```

Первый параметр - экземпляр CollectionView, второй - текущий экземпляр item view, предоставляющий готовый для рендеринга HTML в своем itemView.el.

Третий параметр - `index`. Это индекс модели, которую представляет экземпляр itemView. Он может оказаться полезным при сортировке коллекции и отображении отсортированного списка в должном порядке.

CollectionView: встроенные представления

CollectionView использует Backbone.BabySitter для хранения и организации встроенных представлений, что предполагает легкий доступ к представлениям внутри базового представления, перебор и нахождение их по данному индексу точно так же, как если бы мы работали с моделями внутри коллекции, и ряд других действий.

```
var cv = new Marionette.CollectionView({
  collection: someCollection
});
cv.render();

// нахождение представления по модели
var v = cv.children.findByModel(someModel);

// перебор представлений и их обработка
cv.children.each(function(view){
  // делаем что-то с представлением
});
```

Большая информация доступна в документации по Backbone.BabySitter.

Закрытие CollectionView

CollectionView реализует метод `close`, вызываемый автоматически из region managers.
В частности, происходит следующее:

- удаляется подписка на все события, задействованные в `listenTo`
- удаляется подписка на все события, определенные пользователем

- удаляется подписка на все DOM-события
- удаляются все отрендеренные item views
- `this.el` удаляется из DOM
- вызывается метод `onClose` представления, если он определен

Путем добавления обработчика события 'close' - метода `onClose` в представление, можно запустить пользовательский код с какими-то дополнительными действиями (например, по очистке ресурсов), который выполнится, когда представление будет закрыто.

```
Backbone.Marionette.CollectionView.extend({
  onClose: function(){
    // здесь код, который должен быть выполнен после закрытия представления
  }
});
```