Rendering collection

We now have a model and a collection in place and it is high time to use them in practice and render some data into the view.

First of all let's instantiate collection inside the *app.js& and fetch data.

```
var events = new Organizer.Events();
events.fetch();
```

Your local storage is going to be empty for now, so manually populate your collection:

```
events.reset( [{title: 'event 1'}, {title: 'event 2'}, {title: 'event 3'}] );
```

Now just provide your collection to the view as an attribute:

```
var eventsList = new Organizer.EventsListView({collection: events});
```

Inside the view we have to remove the array with demo data and tweak the cycle a bit:

```
this.collection.each(function(event) {
```

Reload the page and note that nothing is being rendered? Debug with the help of `console.log`:

```
console.log(this.model);
```

We've already seen that model's attributes cannot be accessed directly therefore we need to turn the model to an object using `toJSON`:

```
this.$el.html( template(this.model.toJSON()) );
```

`fetch` can accept two callbacks: `success` and `error`:

```
events.fetch({
  success: function() {
    console.log('okay!')
  },
  error: function() {
    console.log('error');
  }
});
```

Each callback accepts a collection, response and options as arguments. For now we may put these lines

```
events.reset( [{title: 'event 1'}, {title: 'event 2'}, {title: 'event 3'}] );
eventsList.render();
```

inside the `success` callback. In the next step I'll show you another, more preferable solution.