How to reduce code duplication

Our views share some common patterns and it would be great to extract them to some base class and just use extending mechanism. We will require two base classes: one for a single item view (like a single event or form) and one for a list of items (that is, list of events).

Go ahead and create a new *base.js* file inside the views directory. Don't forget to hook it up on the main page:

```
<script src="js/app.js"></script>
<script src="js/models/event.js"></script>
<script src="js/collections/events.js"></script>
<script src="js/views/base.js"></script>
<script src="js/views/events.js"></script>
<script src="js/router.js"></script>
```

First of all, create the base class for a single item view. Every such view has a `render` function that compiles the corresponding template, perhaps by passing some model's data. Let's call it `ItemView`:

```
Organizer.ItemView = Backbone.View.extend({
  render: function() {
    var template = Handlebars.compile($(this.template).html());

    var data = {};
    if (this.model) {
      data = _.isFunction(this.model.toJSON) ? this.model.toJSON() : this.model;
    }
    this.$el.html( template(data) );

    return this;
  }
});
```

The template's id will be passed as an attribute. Next we have to check if the view has any model attached to it. Remember, that in some cases we have to use `toJSON` method like inside the `EventView` and in some cases we don't as the model is already converted to JSON (like in `ShowEventView`). Therefore we check if the model can be converted to JSON and if yes we do it, otherwise leave it as is. Lastly compile the template and update the element. We are not

adding any attributes like `tagName` or `className` or an `initialize` function, because these will be set specifically for each child view.

Let's do the same for the list view:

```
Organizer.ListView = Backbone.View.extend({
  initialize: function() {
    this.collection.on('reset', this.render, this);
    this.collection.on('add', this.render, this);
    this.collection.on('remove', this.render, this);
  },
  render: function() {
    var els = [];
    var that = this;
    this.collection.each(function(element) {
      var itemView = new that.ItemView({model: element});
      els.push(itemView.render().el);
    });
    this.$el.html(els);
    return this;
  }
});
```

We are working with a collection here, so binding all these events makes sense. Note that inside the each cycle I've replaced `new Organizer.EventView` with `new that.ItemView`. As long as this is a base class, every child class will have its own item view, therefore it will be provided as a property. The other code seems very similar to the one we saw earlier. The only difference is that we are not rendering anything to the DOM.

Re-write our views:

```
Organizer.EventsListView = Organizer.ListView.extend({
  tagName: 'ul',
  className: 'list-group',
  ItemView: Organizer.EventView
});
```

The `EventView`:

```
Organizer.EventView = Organizer.ItemView.extend({
  tagName: 'li',
  className: 'list-group-item',
  template: '#event-template',
  events: {
    'click .btn-danger': 'removeEvent'
  },
  removeEvent: function(e) {
    e.preventDefault();
```

```
    if (confirm('Are you sure?')) {
      this.model.destroy();
    }
  }
});
```

I've removed `render` function and added `template` property. Everything else stayed the same.

`NewEventView`:

```
Organizer.NewEventView = Organizer.ItemView.extend({
  tagName: 'form',
  events: {
    'submit': 'createEvent'
  },
  initialize: function() {
    this.render();
  },
  template: '#event-form-template',
  createEvent: function(e) {
    e.preventDefault();
    var that = this;
    var title = this.$('#event_title').val();
    var description = this.$('#event_description').val();
    var model = new Organizer.Event();
    model.save({
      title: title,
      description: description,
      position: Organizer.events.nextPosition()
    }, {
      success: function () {
        Organizer.events.add(model);
        that.el.reset();
        that.$('.has-error').removeClass('has-error');
      }
    });
  }
});
```

`ShowEventView`:

```
Organizer.ShowEventView = Organizer.ItemView.extend({
  initialize: function() {
    this.render();
  },
  template: '#show-event-template'
});
```

We've removed a lot of duplicating code and views are looking much better, but we effectively removed all our append instructions, so nothing is added to the DOM. Therefore let's proceed to the next step and fix this right away!