Introduction to Backbone.js validation

In this step we are going to add **validations** to our model.

Validation means checking if model's attributes satisfy input criteria before saving it. It occurs when a model is persisted via the save method, however you can also call it before set when `validate: true` option is passed.

Do not trust your users as they might set inappropriate data simply because they do not understand which data is actually appropriate. Therefore you also have to set validation logic both on client-side and on server-side and display user-friendly error messages if validation fails.

Adding validations

Let's add yet another field to the form to allow users enter event's description. Open *event.js* file and provide a default value

```
description: ''
```

Modify the form template:

```
<div class="form-group">
  <label for="event_description">Description</label>
  <textarea id="event_description" class="form-control" name="description"></textarea>
</div>
```

Modify the `createEvent` function to fetch description's value:

```
e.preventDefault();
var that = this;
var title = this.$('#event_title').val();
var description = this.$('#event_description').val();
this.model.save({
  title: title,
  description: description
}, {
```

```
  success: function () {
    Organizer.EventsList.add(that.model);
    that.el.reset();
  }
});
```

Validate is a function literal defined for a model that receives attributes as well as provided options as an arguments. Please note that validate only works with attributes that have changed.

```
validate: function(attrs, options) {

}
```

Backbone's documentation instructs us not to return anything from this method if attributes are correct. If something is not correct, an error message or an error object should be returned. If validation fails, save will not continue and an `invalid` event will get triggered. Model will also have a `validationError` property set to an error returned by the validate method.

Suppose we want event's title to be present. UnderscoreJS provides us with a handy method called `isEmpty`:

```
validate: function(attrs, options) {
  if (_.isEmpty(attrs.title)) {
    return "Title has to be present!"
  }
}
```

Now we need to listen for an `invalid` event on the model and perform some action when it happens:

```
initialize: function() {
  this.on('invalid', function(model, error) { console.log(error) })
},
```

So now this error is being showed in my console, but that's not quite helpful – you can't expect user to open the console and look for errors. Instead it would be better to highlight a field in some way.

So it's time to use an error object. I'll just provide field's id as a key and the actual error message as a value:

```
if (_.isEmpty(attrs.title)) {
  return {'event_title': 'Title has to be present!'}
}
```

Re-write the event handler:

```
this.on('invalid', function(model, error) {
  $('#' + _.keys(error)[0]).parent().addClass('has-error');
})
```

keys is an Underscore's method that returns all the names of object's properties. We then just get the first key and pass it as an id's name to the jQuery selector. Next we grab this element's parent and assign `has-error` class to it. This class is provided by Bootstrap and it has to be assigned to the whole `form-group` instead of an individual field.

I'd like to highlight the label as well. For this to work label must be assigned with `control-label` class:

```
<script id="event-form-template" type="text/x-handlebars-template">
  <div class="form-group">
    <label for="event_title" class="control-label">Title</label>
    <input type="text" id="event_title" class="form-control" name="title">
  </div>

  <div class="form-group">
    <label for="event_description" class="control-label">Description</label>
    <textarea id="event_description" class="form-control" name="description"></textarea>
  </div>
  <input type="submit" class="btn btn-primary" value="Add an event">
</script>
```

If a user enters the correct data and submit the form, the highlighting shold be removed:

```
success: function () {
  Organizer.EventsList.add(that.model);
  that.el.reset();
  that.$('.has-error').removeClass('has-error');
}
```