
## Destroying or removing a model

At this step we will bring the Remove button to life and implement functionality to remove events.

Destroying a model is really simple. Run the following commands in the console:

```
var event = Organizer.EventsList.models[0];
event.destroy();
```

Note that the view was automatically re-rendered. The `destroy` event bubbles up to the collection to which the model belongs. However, we are not listening to `destroy` event anywhere! To understand this, re-write our event listeners like this:

```
this.collection.on('remove', function() {console.log('remove');});
this.collection.on('destroy', function() {console.log('destroy');});
```

Now destroy another event:

```
var event = Organizer.EventsList.models[0];
event.destroy();
```

As you see both remove and destroy events were called; remove was called the first. The idea behind the `destroy` method is that it sends the HTTP DELETE request to the server and removes item from the corresponding collection.

Bring back event handlers and code `removeEvent` function:

```
removeEvent: function(e) {
    e.preventDefault();
  this.model.destroy();
}
```

Our remove button however is not very user-friendly. You may click it by occasion and remove an event without an option to restore it. So it would be great to ask the user if he really wants to remove this item. This is simple with JavaScript's confirm dialog:

```
removeEvent: function(e) {
  e.preventDefault();
  if (confirm('Are you sure?')) {
    this.model.destroy();
  }
}
```

## Downsides of JavaScript dialogs

I want to warn you though that these default JS dialogs have some downsides. First of all, you can't style them so in different browsers and operating systems they will look differently. Next, if I click on some other Remove button, Firefox will ask me if I want to hide these dialogs completely. Obviously, we do not want this to happen – you can't trust user on such things. And lastly, while this dialog is being displayed, all background code processing is stopped.

So I'd recommend using pseudo-modal boxes built with HTML, CSS and JS instead, like jQuery UI dialog or Bootstrap's modals.