appacademy / composite_view Private

The canonical a/A composite view implementation.

README.md

Backbone.CompositeView

A Backbone library that defines a class with utility methods for views that house other views.

Dependencies

Backbone.CompositeView requires Underscore version `>= 1.8.0`. `backbone-on-rails` versions after `1.1.2.1` satisfy this dependency, or you can add `underscore.js` to your `vendor/assets/javascripts`.

Installing

Add `composite_view.js` to your Backbone project. A common place for this is in a rails app might be `app/assets/javascripts/utils/composite_view.js`.

Make sure to include the `utils` directory in your `application.js` file (after `backbone`, but before `views`):

```
//= require backbone
...
//= require_tree ./utils
...
//= require_tree ./views
```

API

addSubview (selector, subview, [prepend])

Adds the subview to the parent view's store, calls the subview's `render` method, and attempts to attach it. The subview can be prepended, rather than appended, by passing `true` as the optional third argument.

attachSubview (selector, subview, [prepend])

Appends the subview to the provided selector, delegating events and attaching any children. Available for public use, but primarily for use by `addSubview` and `attachSubviews`.

attachSubviews

Iterates over the `subviews` object and the contained arrays, attaching each subview to its selector with `attachSubview`.

eachSubview (callback)

Iterates over each array of subviews contained in the `subviews` object, calling `callback` once for each subview. The `callback` is passed two arguments: the `subview` and its `selector`.

onRender

Recursively calls `onRender` on all child views. Useful if the behavior of child views depends on their `$el`s being in the DOM. Child `CompositeView`s can extend this method:

```
onRender: function () {
  Backbone.CompositeView.prototype.onRender.call(this);
  // ...
}
```

remove

Overrides the default implementation to call `remove` on all child views, ensuring that they can be garbage collected.

removeSubview (selector, subview)

Calls `remove` on the provided subview and removes it from the `subviews` object.

removeModelSubview (selector, model)

Attempts to find a subview matching the provided selector and model. If found, it will be removed from the DOM and `splice`d out of the selector's array of `subviews`.

subviews ([selector])

Reader method that provides access to the `subviews` object. Takes a CSS selector as an optional argument. If called without a selector, returns the whole object; if called with a selector, returns the array of views for that selector. The returned object or array is wrapped with Underscore as a convenience.

unshiftSubview (selector, subview)

Alias for `addSubview`, passing in `true` for `prepend`.

How to use

Collection Add

- Write a `addMyView(model)`

```
addPhotoView: function (photo) {
  var subview = new PhotoView({ model: photo });
  this.addSubview('.photos', subview);
}
```

- In initialize, iterate over the collection, calling `addMyView`.
- Call `addMyView` on the `add` event of the collection

```
initialize: function () {
  this.listenTo(this.collection, 'add', this.addPhotoView);
  this.collection.each(this.addPhotoView.bind(this));
}
```

- Call `attachSubviews` in render

```
render: function () {
  var content = this.template();
  this.$el.html(content);
  this.attachSubviews();
  return this;
}
```

Collection Remove

- Write a `removeMyView(model)`

```
removePhotoView: function (photo) {
  this.removeModelSubview('.photos', photo);
}
```

- Call `removeMyView` on the `remove` event of the collection

```
initialize: function () {
  this.listenTo(this.collection, 'remove', this.removePhotoView);
}
```