BACKBONE.JS

Backbone.Events

Events is a module that can be mixed in to any object, giving the object the ability to bind and trigger custom named events. Events do not have to be declared before they are bound, and may take passed arguments. For example:

```
var object = {};

_.extend(object, Backbone.Events);

object.bind("alert", function(msg) {
  alert("Triggered " + msg);
});

object.trigger("alert", "an event");
```

bind `object.bind(event, callback, [context])`
Bind a **callback** function to an object. The callback will be invoked whenever the event (specified by an arbitrary string identifier) is fired. If you have a large number of different events on a page, the convention is to use colons to namespace them: "poll:start", or "change:selection"

To supply a **context** value for this when the callback is invoked, pass the optional third argument: model.bind('change', this.render, this)

Callbacks bound to the special "all" event will be triggered when any event occurs, and are passed the name of the event as the first argument. For example, to proxy all events from one object to another:

```
proxy.bind("all", function(eventName) {
```

BACKBONE.JS

```
_.extend(object, Backbone.Events);

object.bind("alert", function(msg) {
  alert("Triggered " + msg);
});

object.trigger("alert", "an event");
```

bind `object.bind(event, callback, [context])`

Bind a **callback** function to an object. The callback will be invoked whenever the event (specified by an arbitrary string identifier) is fired. If you have a large number of different events on a page, the convention is to use colons to namespace them: "poll:start", or "change:selection"

To supply a **context** value for this when the callback is invoked, pass the optional third argument: model.bind('change', this.render, this)

Callbacks bound to the special "all" event will be triggered when any event occurs, and are passed the name of the event as the first argument. For example, to proxy all events from one object to another:

```
proxy.bind("all", function(eventName) {
  object.trigger(eventName);
});
```

unbind `object.unbind([event], [callback])`

Remove a previously-bound callback function from an object. If no callback is specified, all callbacks for the event will be removed. If no event is specified, all event callbacks on the object will be removed.