BACKBONE.JS

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.

on	object.on(event, callback, [context])[context])	Bind a callback function to an object.
off	object.off([event], [callback], [context])	Remove a previously-bound callback function from an object.
trigger	object.trigger(event, [*args])	Trigger callbacks for the given event, or space-delimited list of events.

Model

Models are the heart of any JavaScript application, containing the interactive data as well as a large part of the logic surrounding it: conversions, validations, computed properties, and access control.

extend	Backbone.Model.extend(properties, [classProperties])	To create a Model class of your own, you extend Backbone.Model and provide instance properties, as well as optional classProperties to be attached directly to the constructor function.
constructor / initialize	new Model([attributes])	When creating an instance of a model, you can pass in the initial values of the attributes, which will be set on the model.
get	model.get(attribute)	Get the current value of an attribute from the model.
set	model.set(attributes, [options])	Set a hash of attributes (one or many) on the model.
escape	model.escape(attribute)	Similar to get, but returns the HTML-escaped version of a model's attribute.
has	model.has(attribute)	Returns true if the attribute is set to a non-null or non-undefined value.
unset	model.unset(attribute, [options])	Remove an attribute by deleting it from the internal attributes hash.
clear	model.clear([options])	Removes all attributes from the model.
id	model.id	A special property of models, the id is an arbitrary string (integer id or UUID).
idAttribute	model.idAttribute	A model's unique identifier is stored under the id attribute.
cid	model.cid	A special property of models, the cid or client id is a unique identifier automatically assigned to all models when they're first created.
attributes	model.attributes	The attributes property is the internal hash containing the model's state.
defaults	model.defaults or model.defaults()	The defaults hash (or function) can be used to specify the default attributes for your model.
toJSON	model.toJSON()	Return a copy of the model's attributes for JSON stringification.
fetch	model.fetch([options])	Resets the model's state from the server.
save	model.save([attributes], [options])	Save a model to your database (or alternative persistence layer), by delegating to Backbone.sync.
destroy	model.destroy([options])	Destroys the model on the server by delegating an HTTP DELETE request to Backbone.sync.
validate	model.validate(attributes)	Validate is called before set and save, and is passed the attributes that are about to be updated.
isValid	model.isValid()	Call model.isValid() to check if the model is currently in a valid state, according to your validate function.
url	model.url()	Returns the relative URL where the model's resource would be located on the server.
urlRoot	model.urlRoot	Specify a urlRoot if you're using a model outside of a collection, to enable the default url function to generate URLs based on the model id.
parse	model.parse(response).url()	Parse is called whenever a model's data is returned by the server, in fetch, and save.
clone	model.clone()	Returns a new instance of the model with identical attributes.
isNew	model.isNew()	If the model does not yet have an id, it is considered to be new.
change	model.change()	Manually trigger the "change" event and a "change:attribute" event for each attribute that has changed.
hasChanged	model.hasChanged([attribute])	Has the model changed since the last "change" event?
changedAttributes	model.changedAttributes([attributes])	Retrieve a hash of only the model's attributes that have changed.
previous	model.previous(attribute)	During a "change" event, this method can be used to get the previous value of a changed attribute.
previousAttributes	model.previousAttributes()	Return a copy of the model's previous attributes.

Collection

Collections are ordered sets of models.

extend	Backbone.Collection.extend(properties, [classProperties])	To create a Collection class of your own, extend Backbone.Collection, providing instance properties, as well as optional classProperties to be attached directly to the collection's constructor function.
model	collection.model	Override this property to specify the model class that the collection contains.
constructor / initialize	new Collection([models], [options])	When creating a Collection, you may choose to pass in the initial array of models. The collection's comparator function may be included as an option.
models	collection.models	Raw access to the JavaScript array of models inside of the collection.
toJSON	collection.toJSON()	Return an array containing the attributes hash of each model in the collection.

add	collection.add(models, [options])	Add a model (or an array of models) to the collection.
remove	collection.remove(models, [options])	Remove a model (or an array of models) from the collection.
get	collection.get(id)	Get a model from a collection, specified by id.
getByCid	collection.getByCid(cid)	Get a model from a collection, specified by client id.
at	collection.at(index)	Get a model from a collection, specified by index.
length	collection.length	Like an array, a Collection maintains a length property, counting the number of models it contains.
comparator	collection.comparator	If you define a comparator, it will be used to maintain the collection in sorted order.
sort	collection.sort([options])	Force a collection to re-sort itself.
pluck	collection.pluck(attribute)	Pluck an attribute from each model in the collection.
url	collection.url or collection.url()	Set the url property (or function) on a collection to reference its location on the server.
parse	collection.parse(response)	Parse is called by Backbone whenever a collection's models are returned by the server, in fetch.
fetch	collection.fetch([options])	Fetch the default set of models for this collection from the server, resetting the collection when
reset	collection.reset(models, [options])	Adding and removing models one at a time is all well and good, but sometimes you have so many models to change that you'd rather just update the collection in bulk.
create	collection.create(attributes, [options])	Convenience to create a new instance of a model within a collection.

Router

Router provides methods for routing client-side pages, and connecting them to actions and events.

extend	Backbone.Router.extend(properties, [classProperties])	Define actions that are triggered when certain URL fragments are matched.
routes	router.routes	The routes hash maps URLs with parameters to functions on your router
constructor / initialize	new Router([options])	When creating a new router, you may pass its routes hash directly as an option, if you choose.
route	router.route(route, name, [callback])	Manually create a route for the router.
navigate	router.navigate(fragment, [options])	Call navigate in order to update the URL.

History

History serves as a global router (per frame) to handle hashchange events or pushState, match the appropriate route, and trigger callbacks.

start	Backbone.history.start([options])	Call Backbone.history.start() to begin monitoring hashchange events, and dispatching routes.

Sync

Sync is the function that Backbone calls every time it attempts to read or save a model to the server.

emulateHTTP	Backbone.emulateHTTP = true	
emulateJSON	Backbone.emulateJSON = true	

View

The general idea is to organize your interface into logical views, backed by models, each of which can be updated independently when the model changes, without having to redraw the page.

extend	Backbone.View.extend(properties, [classProperties])	
constructor / initialize	new View([options])	
el	view.el	All views have a DOM element at all times (the el property)
$el	view.$el	A cached jQuery (or Zepto) object for the view's element.
setElement	view.setElement(element)	Call setElement to apply a Backbone view to a different DOM element.
attributes	view.attributes	A hash of attributes that will be set as HTML DOM element attributes on the view's el.
$ (jQuery or Zepto)	view.$(selector)	Function that runs queries scoped within the view's element.
render	view.render()	Override this function with your code that renders the view template from model data, and updates this.el with the new HTML.
remove	view.remove()	Convenience function for removing the view from the DOM.
make	view.make(tagName, [attributes], [content])	Convenience function for creating a DOM element of the given type (tagName).
delegateEvents	delegateEvents([events])	Uses jQuery's delegate function to provide declarative callbacks for DOM events within a view.
undelegateEvents	undelegateEvents()	Removes all of the view's delegated events.

Utility

noConflict	var backbone = Backbone.noConflict();	Returns the Backbone object back to its original value.
setDomLibrary	Backbone.setDomLibrary(jQueryNew);	Tell Backbone to use a particular object as it's DOM / Ajax library.