

Backbone.js Views

Liam McLennan

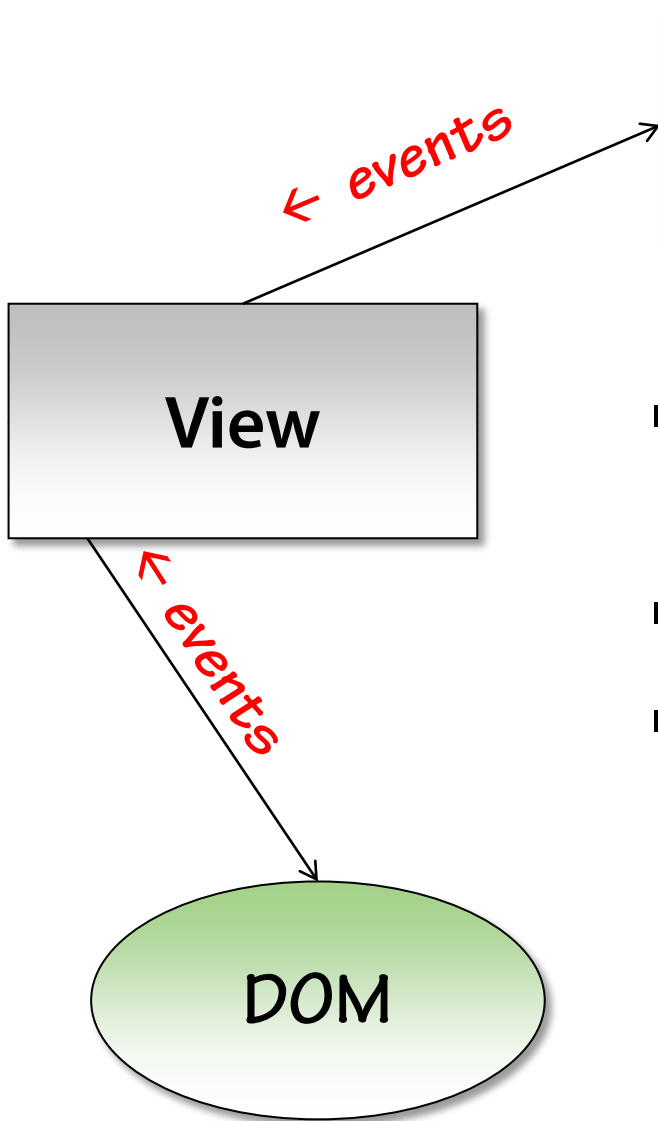
@liammclennan



Outline

- What are views for?
- Defining view types
- Instantiating views
- View properties
- Rendering
- Views and models
- View events

Views



- Provide the 'glue' between models and the document
- Handle model events
- Handle DOM events

Defining New View Types

- Define new View types by extending `Backbone.View`

```
var VehicleListView = Backbone.View.extend({  
    // properties  
});
```

- All views have an associated DOM element at all times (`.el`)

Views that Create New Elements

- The new element is defined by the id, tagName, className and attributes

```
var V = Backbone.View.extend({
  tagName: 'li',
  id: 'thing',
  className: 'active',
  attributes: {
    'data-value': 12345
  }
});
var v = new V();
$('body').prepend(v.el);
```

Views that Attach to Existing Elements

- Pass a 'el' property to the view's constructor

```
var V = Backbone.View.extend({});
```

```
var v = new V({el: '#test'});
```

```
v.$el.css('background-color', 'CornflowerBlue');
```

<http://jsfiddle.net/R9Eb6/3/>

Instantiating Views

- To create a new view object call its constructor function with the 'new' operator
- The simplest case is to create an instance of `Backbone.View`

```
var view = new Backbone.View();
```

- Usually you will want to instantiate instances of your own view type

```
var VehicleListView = Backbone.View.extend({});  
var myView = new VehicleListView();
```

Instantiating Views (cont.)

- Often you will pass a model to the view constructor

```
var myView = new VehicleListView({  
    model: myModelObject  
});
```


Instantiating Views (cont.)

- Any of the following properties will be attached directly to the view object if passed to the constructor
 - model, collection, el, id, className, tagName, attributes

```
var myView = new VehicleListView({  
  model: myModelObject,  
  className: 'model-object'  
});
```

el

- All views have an 'el' property that references the views DOM element

```
var v = new Backbone.View({el: 'body'});  
v.el // <body></body>
```

<http://jsfiddle.net/bv6T8/>

\$el

- **\$el is a cached jQuery (or zepto) wrapper around el**
 - Avoids repeated `$(this.el)`

```
var v = new Backbone.View({el: 'body'});  
v.$el // [<body></body>]
```

<http://jsfiddle.net/H8mpG/>

this.\$

- **this.\$ is the jQuery (or zepto) function scoped to the current view**
 - this.\$('selector') is equivalent to this.\$el.find('selector')

render

- Render is the function that render's the views element (.el)
- The default implementation is a no-op. Provide an implementation with your view definitions.
- Should return 'this'

```
var V = Backbone.View.extend({  
  render: function () {  
    this.$el.html('some content');  
    return this;  
  }  
});
```

Combining Views and Models

- Pass the model to the view's constructor

```
var v = new View({  
  model: myModel  
});
```

- Bind the view's render method

```
myModel.on('change', function () {  
  $('body').append(v.render().el);  
});
```

make

- Helpful function for creating DOM elements

```
var el = new Backbone.View().make(  
  'h3',  
  {class: 'not-very-important'},  
  'Preliminary Version'  
);  
  
// <h3 class="not-very-important">Preliminary Version</h3>
```

remove

- Remove is a shortcut method to remove the view from the DOM
- Equivalent to:

```
$el.remove();
```


events

- **Declarative syntax to register handlers for DOM events**

```
var FormView = Backbone.View.extend({  
  events: {  
    'click .clickable': 'handleClick',  
  },  
  handleClick: function () {}  
})
```

- **Equivalent to:**

```
this.$('.clickable').click(handleClick);
```

View Guidelines

- **Views should render self-contained DOM element**
 - Do not attach to existing elements
 - Do not access DOM elements the view does not own
- **Pass el to the constructor of self-updating view**

Summary

- **Views**
 - Are the link between model's and UI
 - Are responsible for rendering model's
 - Responsible for handling DOM, model and collection events