

Sed sed orci. Nulla  
Donec a turpis in

# Views

- LEVEL 3 -



Donec a  
sed sed orci.  
ipis in erat  
erat



Sed sed orci. Nullam a lac  
Donec a turpis in erat cur  
fringit. Sed sed orci. Nullam



# More on the View Element

---

```
var SimpleView = Backbone.View.extend({});  
var simpleView = new SimpleView();
```

```
console.log(simpleView.el);
```

---> <div></div>

```
var SimpleView = Backbone.View.extend({tagName: 'li'});  
var simpleView = new SimpleView();
```

```
console.log(simpleView.el);
```

---> <li></li>

*tagName can be any HTML tag*



# More on the View Element

---

```
var TodoView = Backbone.View.extend({  
  tagName: 'article',  
  id: 'todo-view',  
  className: 'todo'  
});
```

```
var todoView = new TodoView();  
console.log(todoView.el);
```

--> <article id="todo-view" class="todo"></article>



# More on the View Element

```
var todoView = new TodoView();  
console.log(todoView.el);
```

--> `<article id="todo-view" class="todo"></article>`

*I want to use a jQuery method*

`$('#todo-view').html();`



*el is a DOM Element*

`$(todoView.el).html();`

*Shortcut*

`todoView.$el.html();`



*Good since the el's id may  
be dynamic*

VIEWS

Backbone.js



# Back in Level 1

```
var TodoView = Backbone.View.extend({  
  render: function(){  
    var html = '<h3>' + this.model.get('description') + '</h3>';  
    $(this.el).html(html);  
  }  
});
```

```
var todoView = new TodoView({ model: todoItem });  
todoView.render();  
console.log(todoView.el);
```

```
<div>  
  <h3>Pick up milk</h3>  
</div>
```



# Adding the `el` attributes

```
var TodoView = Backbone.View.extend({
  tagName: 'article',
  id: 'todo-view',
  className: 'todo',
  render: function(){
    var html = '<h3>' + this.model.get('description') + '</h3>';
    $(this.el).html(html); !
  }
});
```



# fixing the `el`

```
var TodoView = Backbone.View.extend({
  tagName: 'article',
  id: 'todo-view',
  className: 'todo',
  render: function(){
    var html = '<h3>' + this.model.get('description') + '</h3>';
    this.$el.html(html);
  }
});
```

`$(this.el).html(html);`

```
var todoView = new TodoView({ model: todoItem });
todoView.render();
console.log(todoView.el);
```

---> `<article id="todo-view" class="todo">`  
      `<h3>Pick up milk</h3>`  
      `</article>`

# Using a Template

```
var TodoView = Backbone.View.extend({
  ...
  template: _.template('<h3><%= description %></h3>'),
  render: function(){
    var attributes = this.model.toJSON();
    this.$el.html(this.template(attributes));
  }
});
```

*The underscore library*

```
var todoView = new TodoView({ model: todoItem });
todoView.render();
console.log(todoView.el);
```

--> `<article id="todo-view" class="todo">`  
    `<h3>Pick up milk</h3>`  
    `</article>`

VIEWS

Backbone.js



# Templating Engines

---

## Underscore.js

```
<h3><%= description %></h3>
```

## Mustache.js

```
<h3>{{description}}</h3>
```

## Haml-js

```
%h3= description
```

## Eco

```
<h3><%= @description %></h3>
```

---

VIEWS

Backbone.js



# Adding View Events

---

```
<h3><%= description %></h3>
```

*In jQuery to add an alert on click*

```
$("#h3").click(alertStatus);  
  
function alertStatus(e) {  
  alert('Hey you clicked the h3!');  
}
```



*Not how we do things in Backbone*



# View Events

---

*Views are responsible for responding to user interaction*

```
var TodoView = Backbone.View.extend({  
  events: {  
    "click h3": "alertStatus"  
  },  
  alertStatus: function(e){  
    alert('Hey you clicked the h3!');  
  }  
});
```

← "<event> <selector>": "<method>"

*Selector is scoped to the el*


```
this.$el.delegate('h3', 'click', alertStatus);
```



# Views Can Have Many Events

```
var DocumentView = Backbone.View.extend({  
  events: {  
    "dblclick" : "open",  
    "click .icon.doc" : "select",  
    "click .show_notes" : "toggleNotes",  
    "click .title .lock" : "editAccessLevel",  
    "mouseover .title .date" : "showTooltip"  
  },  
  ...  
});
```

*anywhere on EL*





# View Event Options

---

```
var SampleView = Backbone.View.extend({  
  events: {  
    "<event> <selector>": "<method>"  
  },  
  ...  
});
```

## Events

change	click	dblclick	focus	focusin
focusout	hover	keydown	keypress	load
mousedown	mouseenter	mouseleave	mousemove	mouseout
mouseover	mouseup	ready	resize	scroll
select	unload			