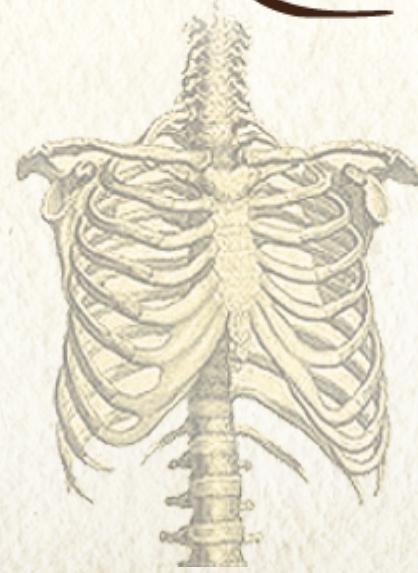


Sed sed orci. Nulla
Donec a turpis in

THE ANATOMY OF Backbone.js

Donec
ed sed orci.
turpis in erat



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

Sed sed orci. Nulla
Donec a turpis in

Introduction

- LEVEL 1 -

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

Introducing the Todo App

To get the todo data

```
$.getJSON('/todo', function(data) {
```

The data returned

```
{ description: 'Pick up milk', status: 'incomplete', id: 1 }
```



todo list!

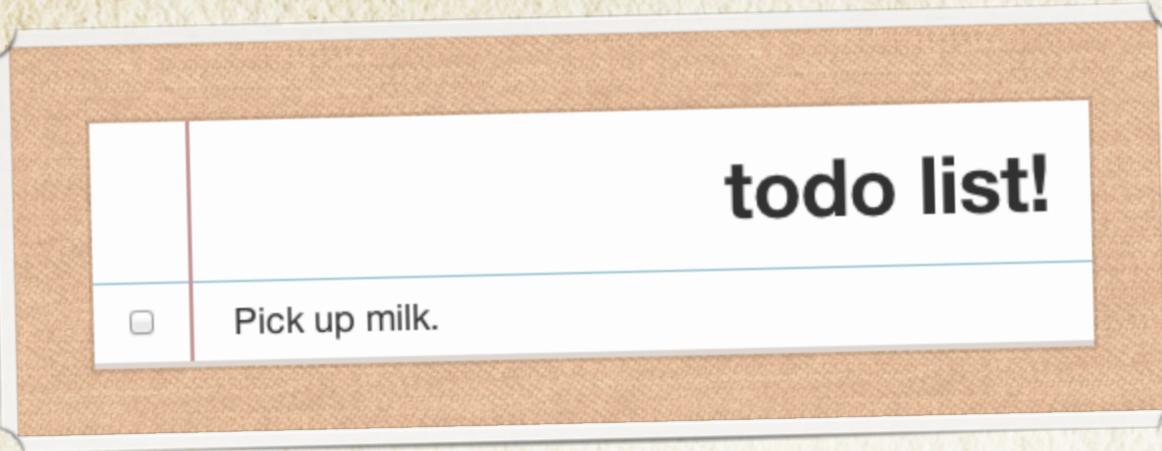


Pick up milk.

Introduction

Backbone.js

Adding functionality



Checking off a todo item

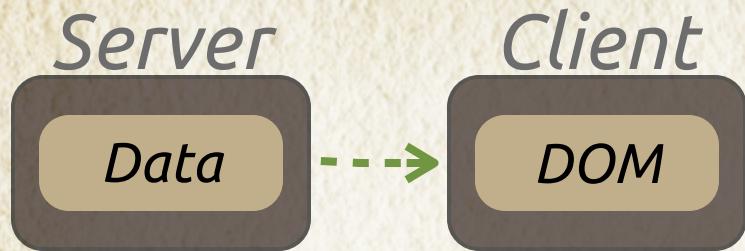
Add deadlines

Reorder & sort



**Methods can be disorganized
We lose the data structure**

Without Backbone.js



```
{ description: 'Pick up milk', status: 'incomplete', id: 1 }
```

```
<h3 class='incomplete'>
<input type='checkbox' data-id='1' />
Pick up milk
</h3>
```



We need an object to maintain the data

Introducing Backbone.js



"Get your truth out of the DOM"

- Jeremy Ashkenas

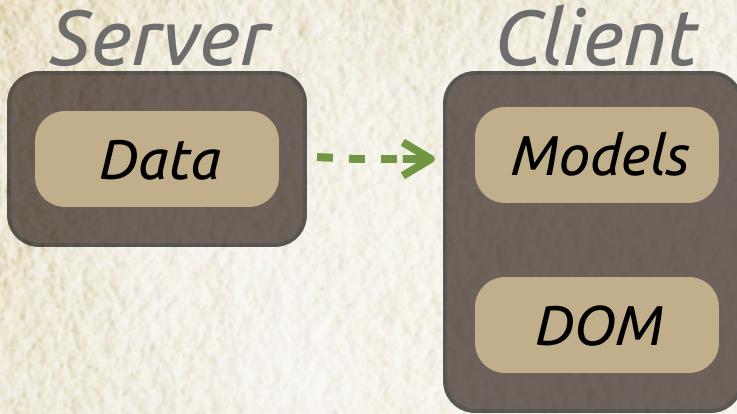
Provides client-side app structure

Models to represent data

Views to hook up models to the DOM

Synchronizes data to/from server

With Backbone.js



To create a model class

```
var TodoItem = Backbone.Model.extend({});
```

To create a model instance

```
var todoItem = new TodoItem(  
  { description: 'Pick up milk', status: 'incomplete', id: 1 }  
);
```

Backbone Models

Models

```
var todoItem = new TodoItem(  
  { description: 'Pick up milk', status: 'incomplete', id: 1 }  
);
```

To get an attribute

```
todoItem.get('description');      - - -> 'Pick up milk'
```

To set an attribute

```
todoItem.set({status: 'complete'});
```

Sync to the server

```
todoItem.save();
```



Configuration needed

Introduction

Backbone.js

Displaying the Data



To create a view class

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

To create a view instance



```
var todoView = new TodoView({ model: todoItem });
```

Rendering the View

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

Every view has a top level **E**Lement

<div>

<p>

<header>

<section>

...

default

Rendering the View

```
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>
```

Sed sed orci. Nulla
Donec a turpis in

Models

- L E V E L 2 -

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

Reviewing Models

Generating a model class

```
var TodoItem = Backbone.Model.extend({});
```

Generating a model instance

```
var todoItem = new TodoItem(  
  { description: 'Pick up milk', status: 'incomplete' }  
);
```

To get an attribute

```
todoItem.get('description');      ----> 'Pick up milk'
```

To set an attribute

```
todoItem.set({status: 'complete'});
```

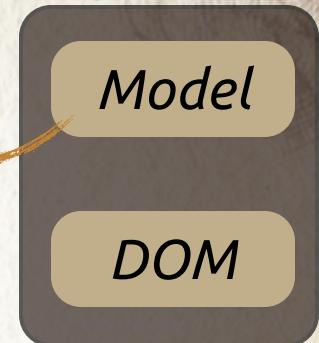
Fetching Data from the Server

```
var todoItem = new TodoItem();
```

Server



Client



URL to get JSON data for model

```
todoItem.url = '/todo';
```

To populate model from server

```
todoItem.fetch();
```

```
----> { id: 1, description: 'Pick up milk', status: 'incomplete' }
```

```
todoItem.get('description');
```

```
----> 'Pick up milk'
```



/todo isn't a good URL

Models

Backbone.js

Fetching Data from the Server

```
var TodoItem = Backbone.Model.extend({urlRoot: '/todos'});
```

Fetch todo with id = 1

```
var todoItem = new TodoItem({id: 1})
```

*RESTful web service
(Rails flavor)*

```
todoItem.fetch();   GET /todos/1
```

---> { id: 1, description: 'Pick up milk', status: 'incomplete' }

Update the todo

```
todoItem.set({description: 'Pick up cookies.'});
```

```
todoItem.save();   PUT /todos/1
```

with JSON params

Creating & Destroying a New Todo

```
var todoItem = new TodoItem();
```

```
todoItem.set({description: 'Fill prescription.'});
```

```
todoItem.save();
```

**POST /todos
with JSON params**

```
todoItem.get('id');
```

---> 2

```
todoItem.destroy();
```

DELETE /todos/2

Get JSON from model

```
todoItem.toJSON();
```

---> { id: 2, description: 'Fill prescription', status: 'incomplete' }

Default Values

```
var TodoItem = Backbone.Model.extend({  
  defaults: {  
    description: 'Empty todo...',  
    status: 'incomplete'  
  }  
});
```

```
var todoItem = new TodoItem();
```

```
todoItem.get('description');
```

---> 'Empty todo...'

```
todoItem.get('status');
```

---> 'incomplete'

Models Can Have Events

To listen for an event on a model

```
todoItem.on('event-name', function(){  
  alert('event-name happened!');  
});
```

Run the event

```
todoItem.trigger('event-name');
```

Special Events

To listen for changes

```
todoItem.on('change', doThing);
```

```
var doThing = function() {  
  ...  
}
```



Event triggered on change

```
todoItem.set({description: 'Fill prescription.'});
```

Set without triggering event

```
todoItem.set({description: 'Fill prescription.',  
             {silent: true}});
```

Remove event listener

```
todoItem.off('change', doThing);
```

Special Events

```
todoItem.on(<event>, <method>);
```

Built-in events

change	<i>When an attribute is modified</i>
change:<attr>	<i>When <attr> is modified</i>
destroy	<i>When a model is destroyed</i>
sync	<i>Whenever successfully synced</i>
error	<i>When model save or validation fails</i>
all	<i>Any triggered event</i>

Sed sed orci. Nulla
Donec a turpis in

Views

- LEVEL 3 -

Sed sed orci. Nullam a lac
Donec a turpis in erat cur
fringil. 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);
```

--->

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*

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>
```

VIEWS

Backbone.js

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!');  
  }  
});
```



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

events:

"dblclick"

"click .icon.doc"

"click .show_notes"

"click .title .lock"

"mouseover .title .date"

: "open",

: "select",

: "toggleNotes",

: "editAccessLevel",

: "showTooltip"

},

...

});

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			

Sed sed orci. Nulla
Donec a turpis in

Models & Views

- LEVEL 4 -

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

Review our Model View

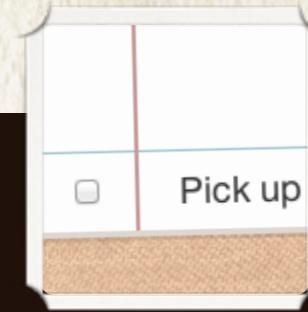
```
var TodoView = Backbone.View.extend({
  template: _.template('<h3><%= description %></h3>'),
  render: function(){
    this.$el.html(this.template(this.model.toJSON()));
  }
});
```

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

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

Adding a checkbox

```
var TodoView = Backbone.View.extend({
  template: _.template('<h3>' +
    '<input type=checkbox ' +
    '<% if(status === "complete") print("checked") %>/>' +
    '<%= description %></h3>'),
  render: function(){
    this.$el.html(this.template(this.model.toJSON()));
  }
});
```



How do we update the model when
checkbox changes?

View events update the Model

Server



Server



Models & Views

Backbone.js

Update model on VI event

```
var TodoView = Backbone.View.extend({  
  
  events: {  
    'change input': 'toggleStatus'  
  },  
  
  toggleStatus: function(){  
    if(this.model.get('status') === 'incomplete'){  
      this.model.set({'status': 'complete'});  
    }else{  
      this.model.set({'status': 'incomplete'});  
    }  
  }  
});
```



Model logic in view

Refactor to the Model

```
var TodoView = Backbone.View.extend({  
  events: {  
    'change input': 'toggleStatus'  
  },  
  toggleStatus: function(){  
    this.model.toggleStatus();  
  }  
});  
  
var TodoItem = Backbone.Model.extend({  
  toggleStatus: function(){  
    if(this.get('status') === 'incomplete'){  
      this.set({'status': 'complete'});  
    }else{  
      this.set({'status': 'incomplete'});  
    }  
  }  
});
```



Model logic in Model

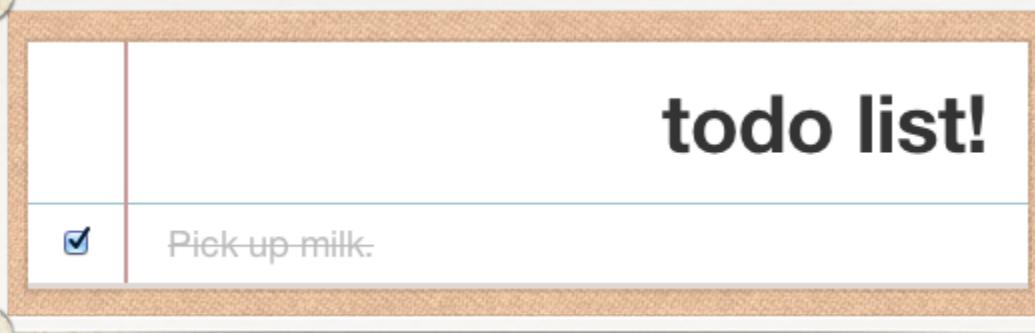
Sync changes to server

```
var TodoItem = Backbone.Model.extend({  
  toggleStatus: function(){  
    if(this.get('status') === 'incomplete'){  
      this.set({'status': 'complete'});  
    }else{  
      this.set({'status': 'incomplete'});  
    }  
    this.save();  
  }  
});
```



PUT /todos/1

Update view to reflect changes



```
.complete {  
  color: #bbb;  
  text-decoration: line-through;  
}
```

update TodoView template:

```
template: _.template('<h3 class="<%= status %>">' +  
  '<% if(status === "complete") print("checked") %>/>' +  
  ' <%= description %></h3>')
```

How should we update the view
when the model changes?

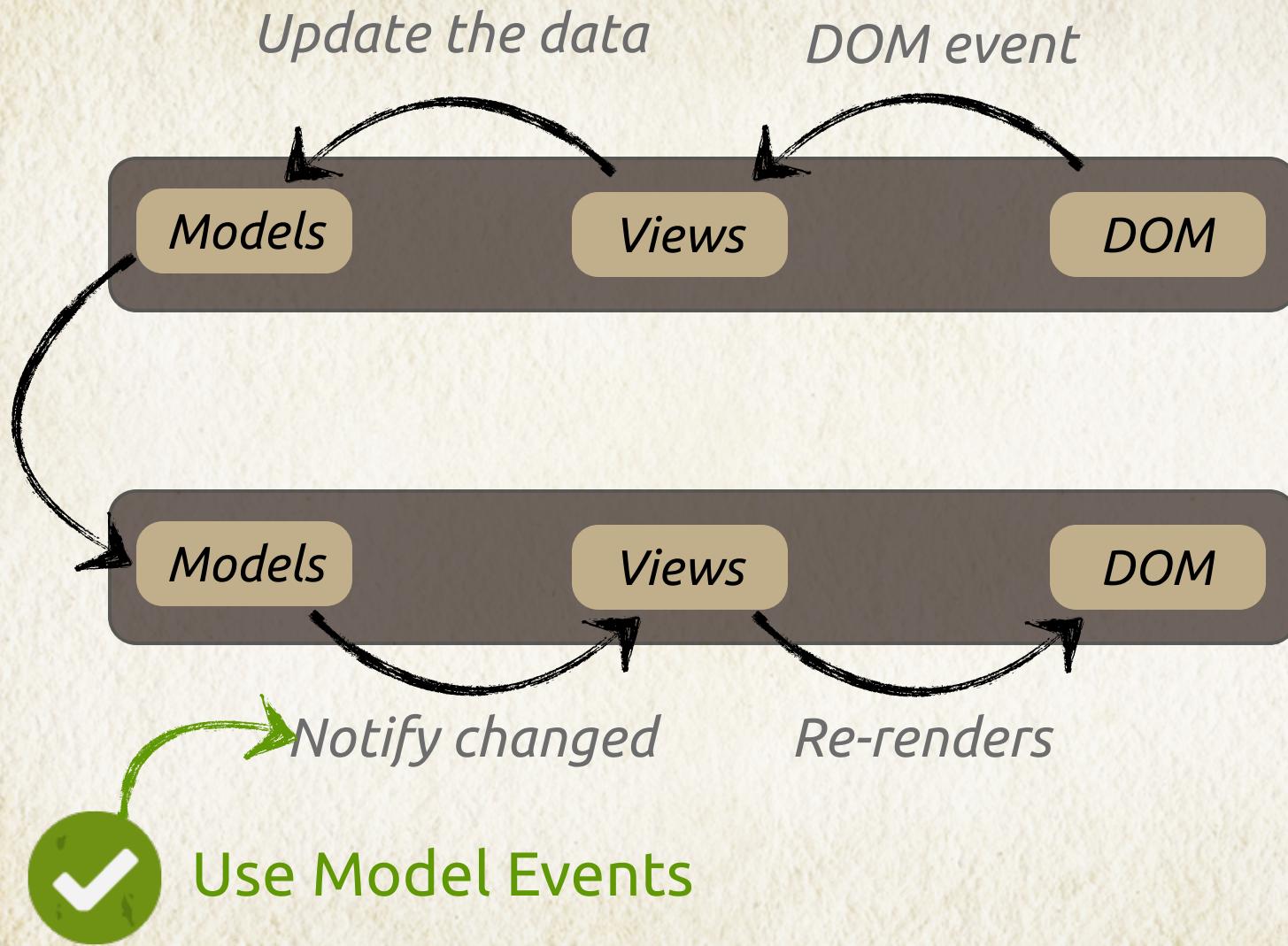
Re-render the view

```
var TodoView = Backbone.View.extend({  
  events: {  
    'change input': 'toggleStatus'  
  },  
  toggleStatus: function(){  
    this.model.toggleStatus();  
    this.render();  
  },  
  render: function(){  
    this.$el.html(this.template(this.model.toJSON()));  
  }  
});
```



Doesn't work for other model changes

Model updates change the View



Models & Views

Backbone.js

Re-render the view

```
var TodoView = Backbone.View.extend({  
  events: {  
    'change input': 'toggleStatus'  
  },  
  initialize: function(){  
    this.model.on('change', this.render, this);  
  },  
  toggleStatus: function(){  
    this.model.toggleStatus();  
  },  
  render: function(){  
    this.$el.html(this.template(this.model.toJSON()));  
  }  
});
```



Why the third argument?

What is this?

```
this.model.on('change', this.render);
```

render()

window

```
render: function(){
  this.$el.html(this.template(this.model.toJSON()));
}
```



render context is not the view

What is this?

```
this.model.on('change', this.render, this);
```

render()

todoView

```
render: function(){
  this.$el.html(this.template(this.model.toJSON()));
}
```

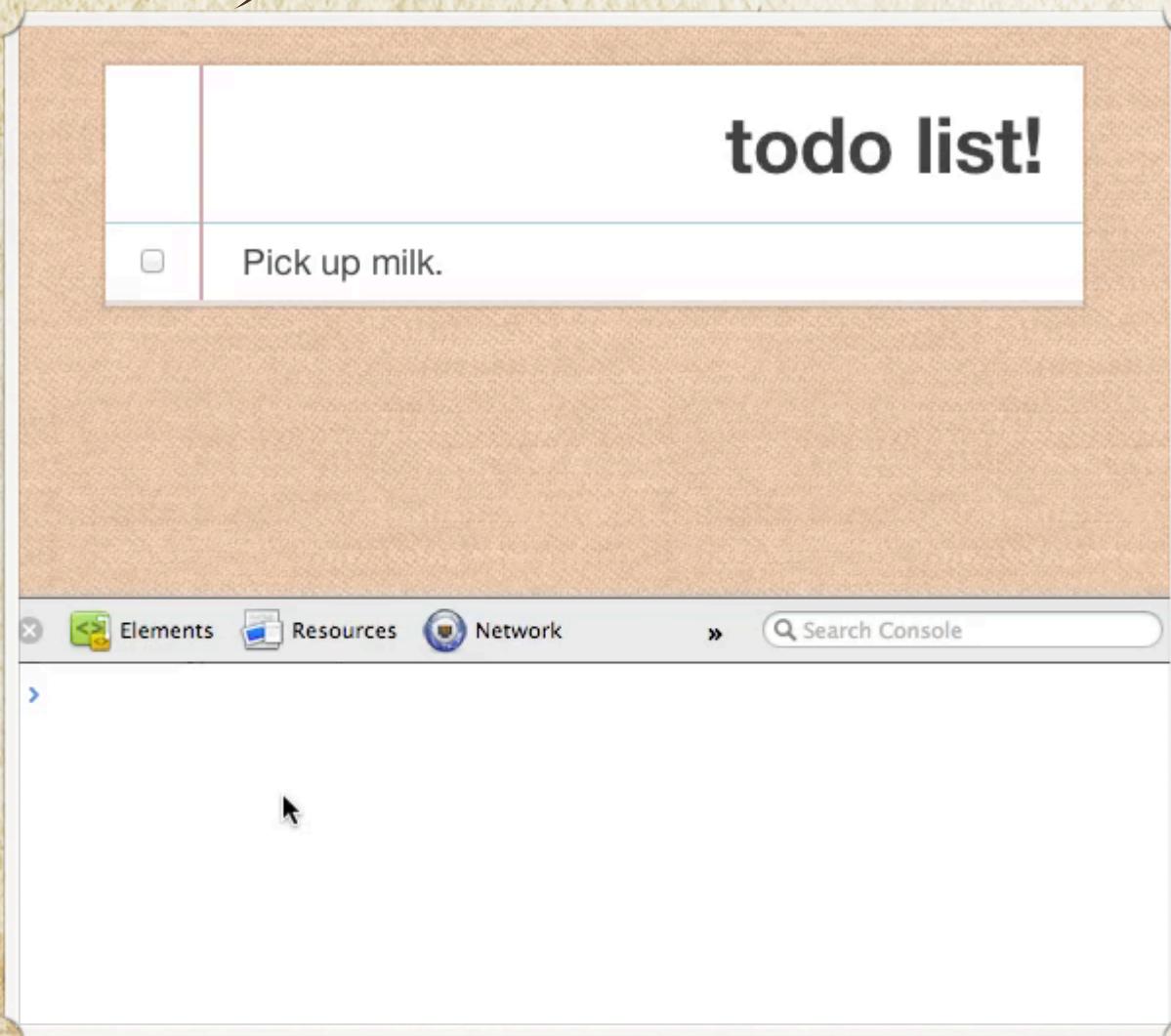


render context is bound to the view

Remove view on model destroy

```
var TodoView = Backbone.View.extend({  
  initialize: function(){  
    this.model.on('change', this.render, this);  
    this.model.on('destroy', this.remove, this);  
  },  
  
  render: function(){  
    this.$el.html(this.template(this.model.toJSON()));  
  },  
  
  remove: function(){  
    this.$el.remove();  
  }  
});
```

Watch it in action



Models & Views

Backbone.js

Sed sed orci. Nulla
Donec a turpis in

Donec
ed sed orci.
turpis in erat

Collections

- LEVEL 5 -

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

Set of Models

```
var todoItem1 = new TodoItem();  
var todoItem2 = new TodoItem();  
var todoList = [todoItem1, todoItem2];
```



```
var TodoList = Backbone.Collection.extend({  
  model: TodoItem  
});  
var todoList = new TodoList();
```



TodoList manages a set of
TodoItem model instances

Add/Remove/Get

get number of models

```
todoList.length;           → 2
```

add a model instance

```
todoList.add(todoItem1);
```

get model instance at index 0

```
todoList.at(0);           → todoItem1
```

get by id 1

```
todoList.get(1);           → todoItem1
```

removing a model instance

```
todoList.remove(todoItem1);
```

~~Bulk Population~~

reset

```
var todos = [  
  {description: 'Pick up milk.', status: 'incomplete'},  
  {description: 'Get a car wash', status: 'incomplete'},  
  {description: 'Learn Backbone', status: 'incomplete'}  
];  
  
todoList.reset(todos);
```

todoList

TodoItem

TodoItem

TodoItem

Each object in Array becomes a TodoItem

Collections

Backbone.js

Fetching Data from the Server

URL to get JSON data from

```
var TodoList = Backbone.Collection.extend({  
  url: '/todos'  
});
```

populate collection from server

```
todoList.fetch();  GET /todos
```



```
[  
  {description: 'Pick up milk.', status: 'incomplete', id: 1},  
  {description: 'Get a car wash', status: 'incomplete', id: 2}  
]
```

```
todoList.length;  - - -> 2
```

Collections Can Have Events

To listen for an event on a collection

```
todoList.on('event-name', function(){  
  alert('event-name happened!');  
});
```

Run the event

```
todoList.trigger('event-name');
```



Works just like models!

Special Events

listen for reset

```
todoList.on('reset', doThing);
```



```
var doThing = function() {  
  ...  
}
```

Event triggered on reset & fetch

```
todoList.fetch();  
todoList.reset();
```

without notification

```
todoList.fetch({silent: true});  
todoList.reset({silent: true});
```

Remove event listener

```
todoList.off('reset', doThing);
```

Special Events on Collection

```
todoList.on(<event>, <function>);
```

Built-in Events

add

When a model is added

remove

When a model is removed

reset

When reset or fetched

```
todoList.on('add', function(todoItem){  
  ...  
});
```

todoItem is the model being added

Model Events

Models in collection

change	<i>When an attribute is modified</i>
change:<attr>	<i>When <attr> is modified</i>
destroy	<i>When a model is destroyed</i>
sync	<i>Whenever successfully synced</i>
error	<i>When an attribute is modified</i>
all	<i>When an attribute is modified</i>

Events triggered on a model in a collection will also be triggered on the collection

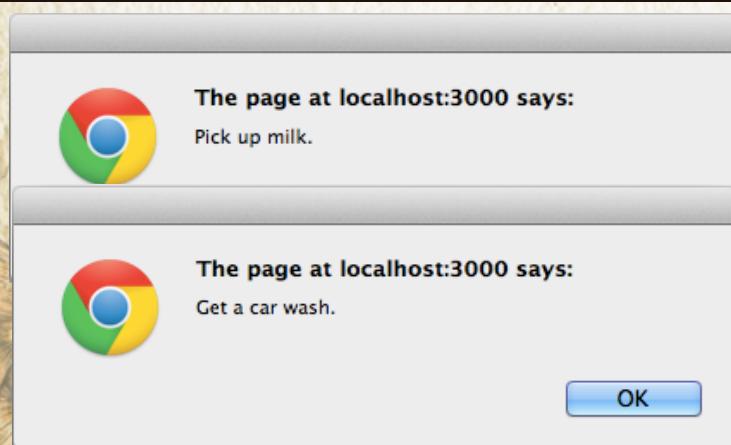
Iteration

Setup our collection

```
todoList.reset([
  {description: 'Pick up milk.', status: 'incomplete', id: 1},
  {description: 'Get a car wash.', status: 'complete', id: 2}
]);
```

Alert each model's description

```
todoList.forEach(function(todoItem){
  alert(todoItem.get('description'));
});
```



Iteration continued

Build an array of descriptions

```
todoList.map(function(todoItem){  
  return todoItem.get('description');  
});
```

----> ['Pick up milk.', 'Get a car wash']

Filter models by some criteria

```
todoList.filter(function(todoItem){  
  return todoItem.get('status') === "incomplete";  
});
```

Returns array of items
that are incomplete

Other Iteration functions

forEach	reduce	reduceRight
find	filter	reject
every	all	some
include	invoke	max
min	sortBy	groupBy
sortedIndex	shuffle	toArray
size	first	initial
rest	last	without
indexOf	lastIndexOf	isEmpty
chain		

<http://documentcloud.github.com/backbone/#Collection-Underscore-Methods>

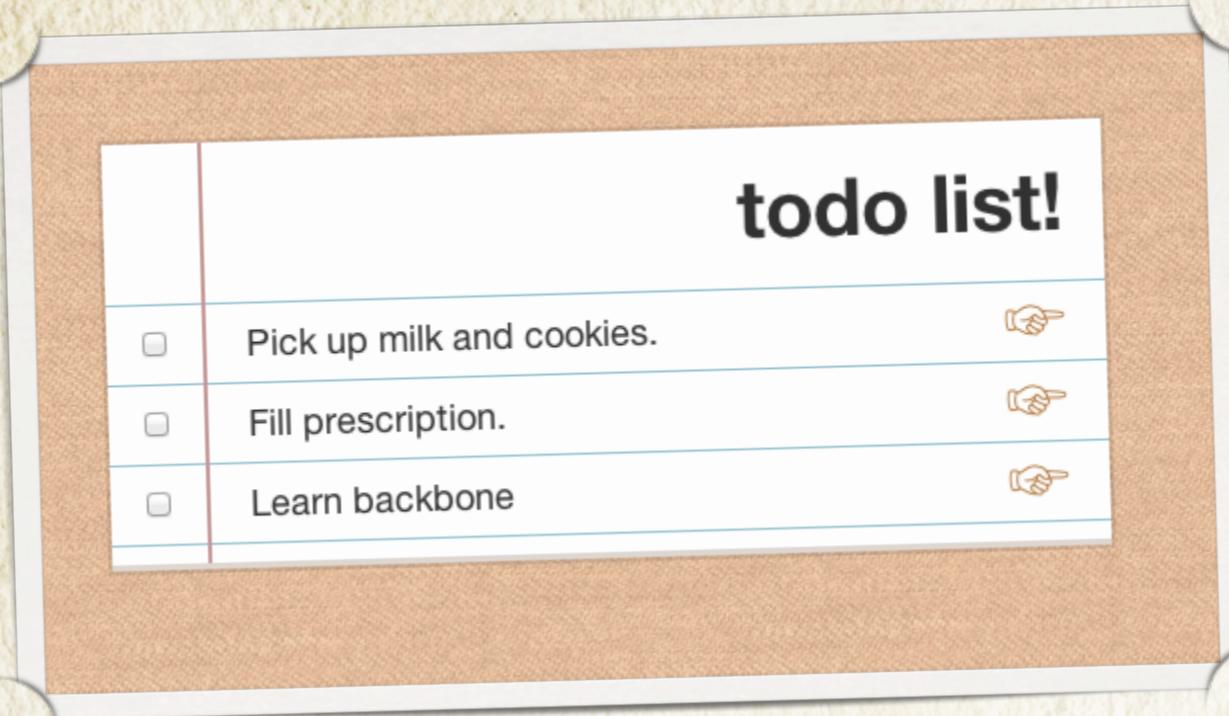
Sed sed orci. Nulla
Donec a turpis in

Collections & Views

- LEVEL 6 -

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

Todo List!

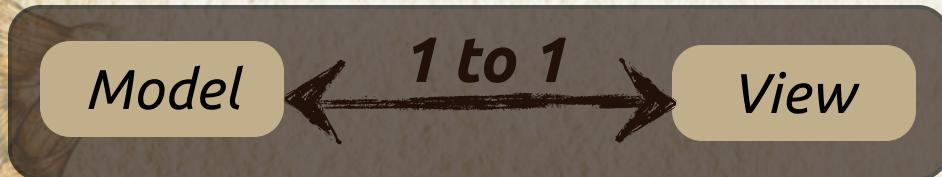


Collection + View == Collection View!

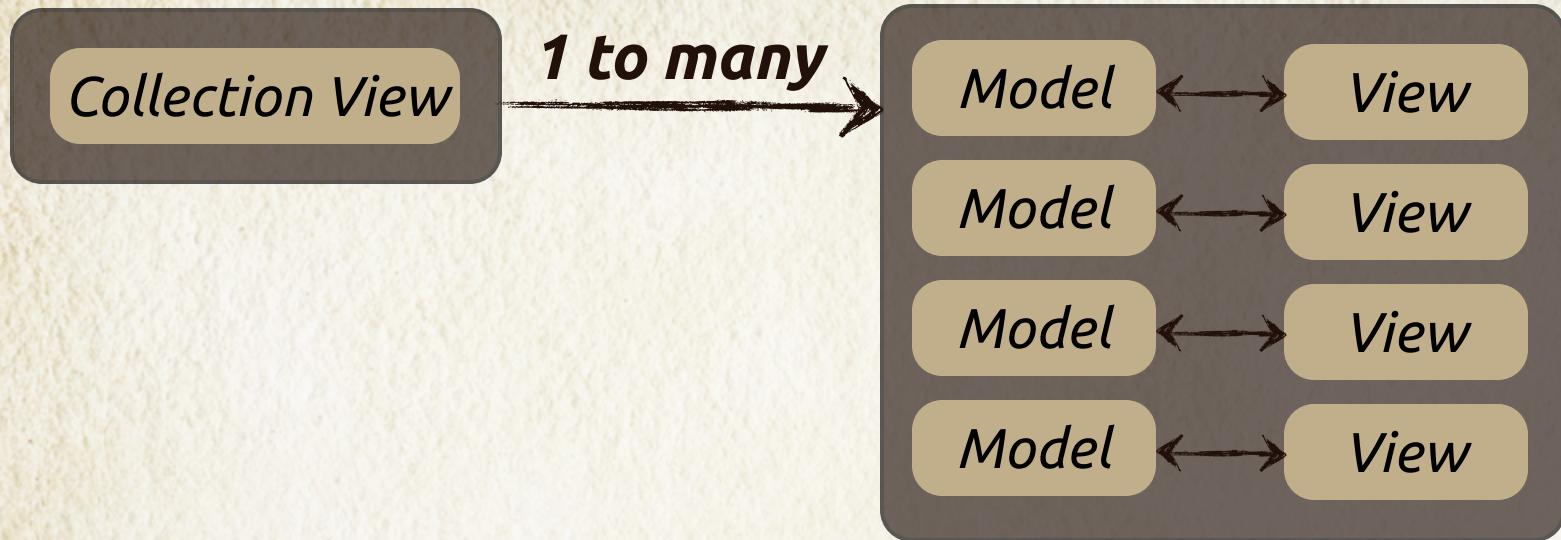
Review our Model View

```
var TodoView = Backbone.View.extend({  
  render: function(){  
    this.$el.html(this.template(this.model.toJSON()));  
    return this;  
  }  
  ...  
});  
  
var todoItem = new TodoItem();  
var todoView = new TodoView({model: todoItem});  
console.log(todoView.render().el);
```

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



Collection Views



A Collection View doesn't render any of it's own HTML.
It delegates that responsibility to the model views.

Define and Render

```
var TodoListView = Backbone.View.extend({});  
var todoListView = new TodoListView({collection: todoList});
```

first crack at render

```
render: function(){  
  this.collection.forEach(function(todoItem){  
    var todoView = new TodoView({model: todoItem});  
    this.$el.append(todoView.render().el);  
  });  
}
```



forEach changes context

Rendering

addOne

```
render: function(){  
  this.collection.forEach(this.addOne, this);  
}  
  
addOne: function(todoItem){  
  var todoView = new TodoView({model: todoItem});  
  this.$el.append(todoView.render().el);  
}
```



forEach saves context

Render continued

```
var todoListView = new TodoListView({collection: todoList});  
todoListView.render();  
console.log(todoListView.el);
```

```
<div>  
  <h3 class="incomplete">  
    <input type=checkbox />  
    Pick up milk.  
  </h3>  
  
  <h3 class="complete">  
    <input type=checkbox checked/>  
    Learn backbone.  
  </h3>  
</div>
```



Adding new Models

```
var TodoListView = Backbone.View.extend({
  addOne: function(todoItem){
    var todoView = new TodoView({model: todoItem});
    this.$el.append(todoView.render().el);
  },
  render: function(){
    this.collection.forEach(this.addOne, this);
  }
});
```

```
var newItem = new TodoItem({
  description: 'Take out trash.',
  status: 'incomplete'
});
todoList.add(newItem);
```



newTodoItem not in DOM

Collections & Views

Backbone.js

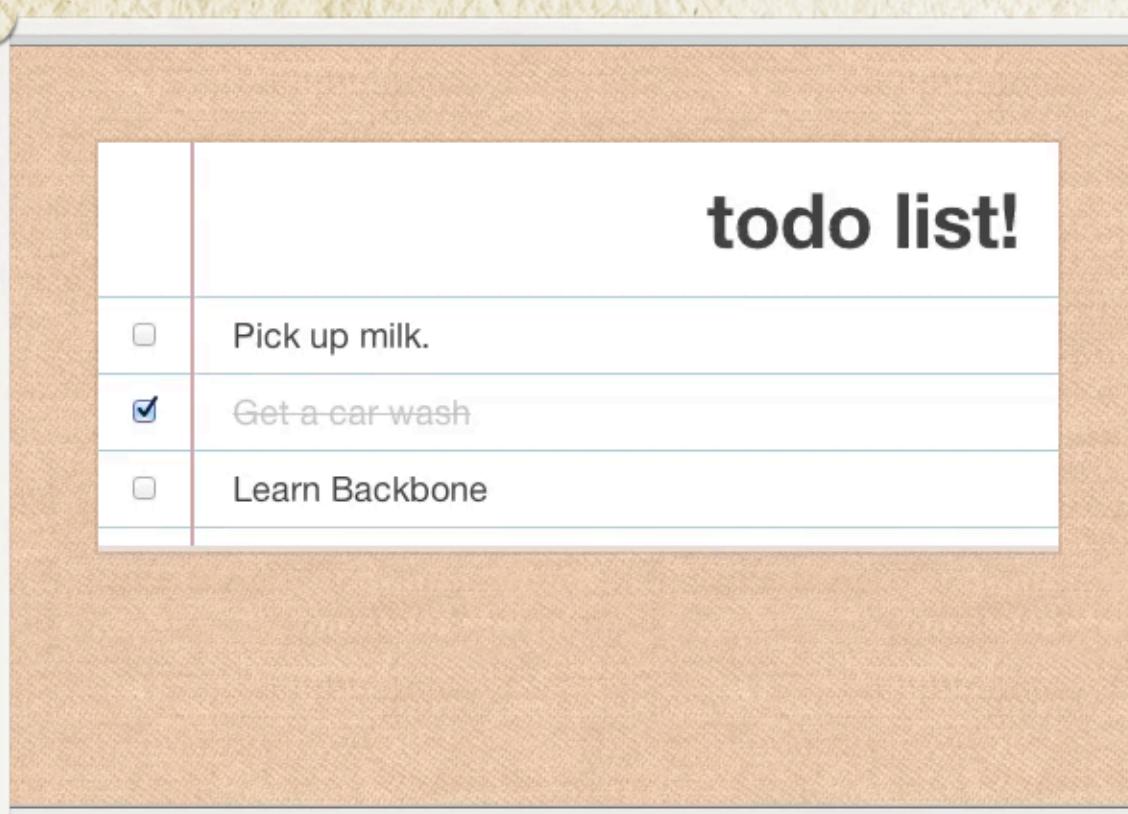
Listen to the add Event

```
var TodoListView = Backbone.View.extend({
  initialize: function(){
    this.collection.on('add', this.addOne, this);
  },
  addOne: function(todoItem){
    var todoView = new TodoView({model: todoItem});
    this.$el.append(todoView.render().el);
  },
  render: function(){
    this.collection.forEach(this.addOne, this);
  }
});
```

```
var newItem = new TodoItem({
  description: 'Take out trash.',
  status: 'incomplete'
});
todoList.add(newItem);
```



Add Event in Action



```
x Elements Resources Network » Search Console
> newTodoItem = new TodoItem({description: 'Learn Rails', status: 'incomplete'})
> child
>
```

Collections & Views

Backbone.js

Reset Event

```
var TodoListView = Backbone.View.extend({  
  initialize: function(){  
    this.collection.on('add', this.addOne, this);  
  },  
  addOne: function(todoItem){  
    var todoView = new TodoView({model: todoItem});  
    this.$el.append(todoView.render().el);  
  },  
  render: function(){  
    this.collection.forEach(this.addOne, this);  
  }  
});
```

```
var todoList = new TodoList();  
var todoListView = new TodoListView({  
  collection: todoList  
});  
todoList.fetch();
```

todoList

reset

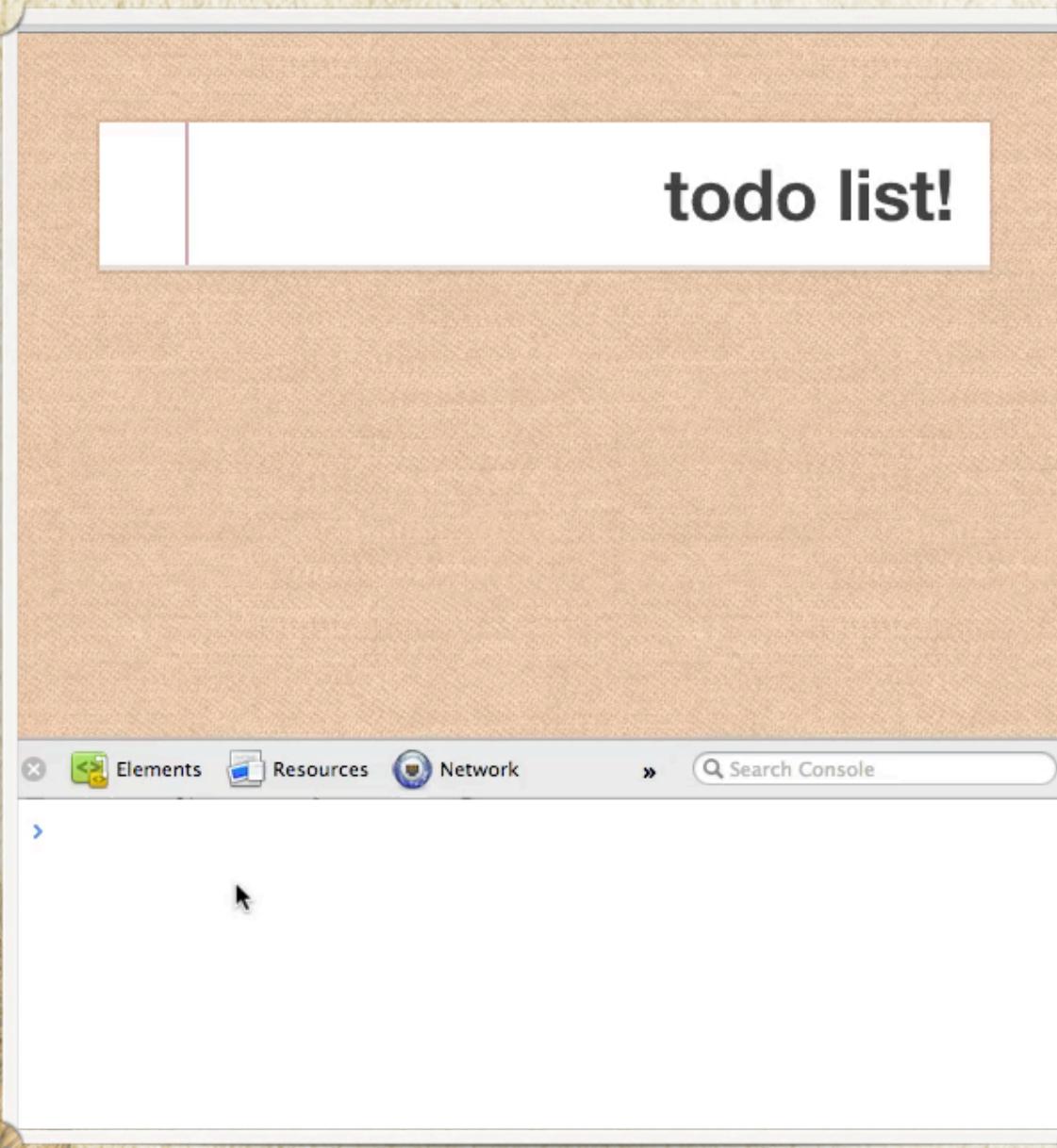
Reset Event

```
var TodoListView = Backbone.View.extend({
  initialize: function(){
    this.collection.on('add', this.addOne, this);
    this.collection.on('reset', this.addAll, this);
  },
  addOne: function(todoItem){
    var todoView = new TodoView({model: todoItem});
    this.$el.append(todoView.render().el);
  },
  addAll: function(){
    this.collection.forEach(this.addOne, this);
  },
  render: function(){
    this.addAll();
  }
});

todoList.fetch();
```



Reset Event in Action



removing from collection

Collections & Views

Backbone.js

~~Fixing remove with Custom Events~~

```
todoList.remove(todoItem);
```



todoList
remove

TodoList Collection

```
initialize: function(){
  this.on('remove', this.hideModel);
},
hideModel: function(model){
  model.trigger('hide');
}
```

TodoItem View

```
initialize: function(){
  this.model.on('hide', this.remove, this);
}
```



todoItem
hide

Bringing it all together

A screenshot of a web browser window. The main content area displays a "todo list!" with three items:

- Pick up milk and cookies.
- Fill prescription.
- Learn backbone

The browser's developer tools are open at the bottom, showing the "Elements", "Resources", and "Network" tabs. A search bar in the tools is set to "Search Console".

Collections & Views

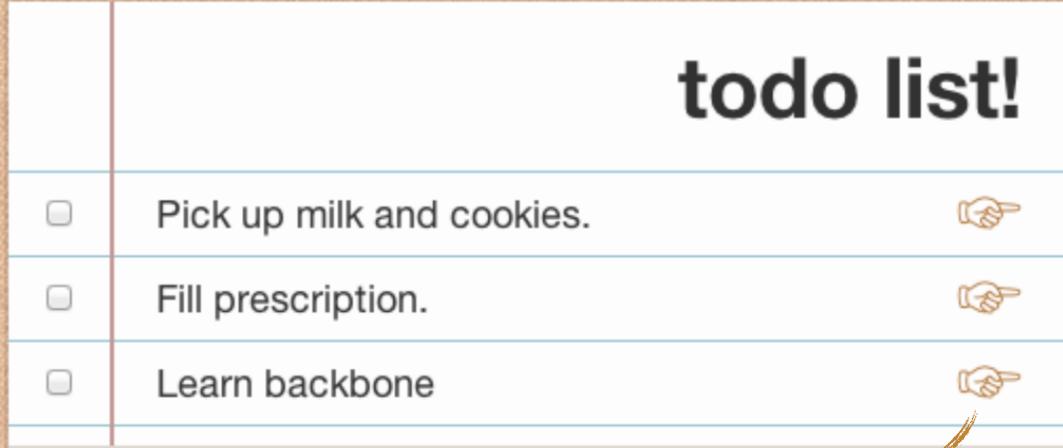
Backbone.js

Router and History

- LEVEL 7 -

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

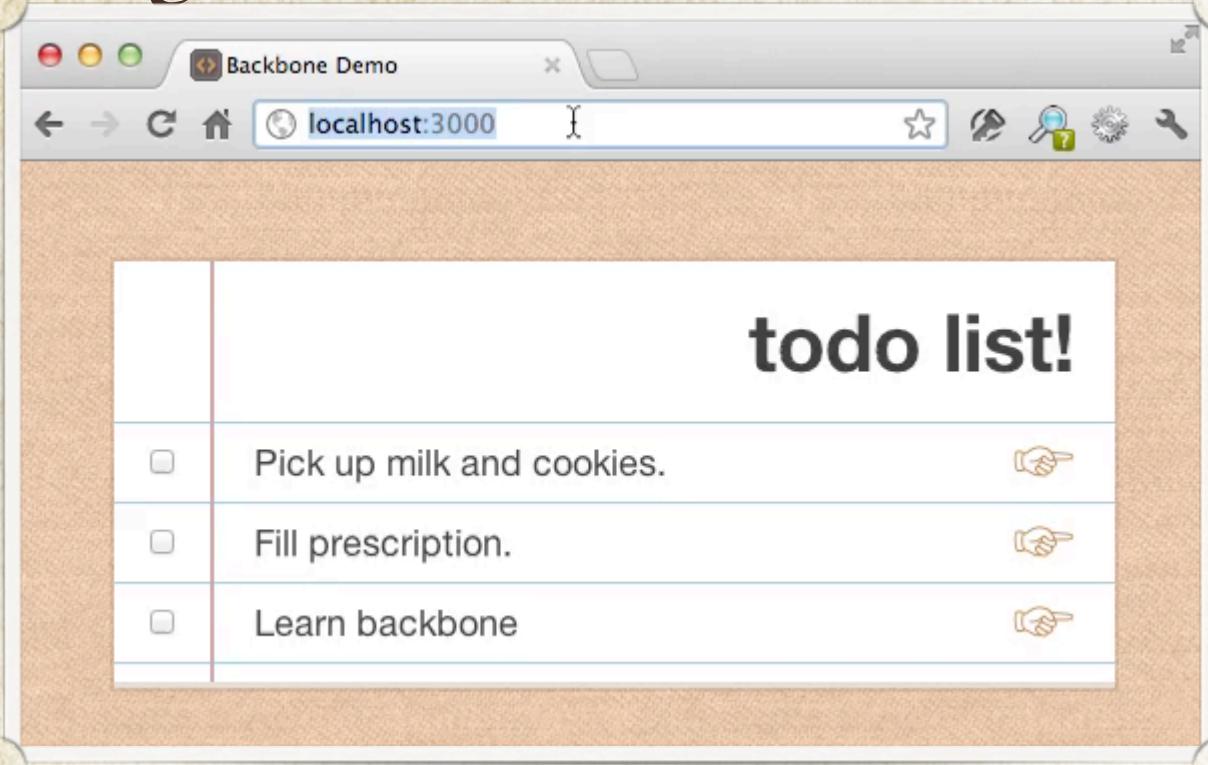
The Problem



```
<a href="#" class='todo'></a>
```

```
$( 'a.todo' ).click( function(e){  
  e.preventDefault();  
  // show single todo  
})
```

The Problem



Router & History to the rescue!

Router & History

Backbone.js

The Router

Router's map URLs to actions

```
var router = new Backbone.Router({  
  routes: { "todos": 'index',  
  
    index: function(){  
      ...  
    }  
  });
```

when the url path is

/todos or #todos



index: function(){ ... }

The Router

Routes match parameter parts

```
var router = new Backbone.Router({  
  routes: { "todos/:id": 'show' }  
  
  show: function(id){ ... }  
})
```

Matches

URL	params
/todos/1	<i>id = 1</i>
/todos/2	<i>id = 2</i>
/todos/hello	<i>id = 'hello'</i>
/todos/foo-bar	<i>id = 'foo-bar'</i>

More Route Matchers

matcher	URL	params
search/:query	search/ruby	<i>query = 'ruby'</i>
search/:query/p:page	search/ruby/p2	<i>query = 'ruby', page = 2</i>
folder/:name-:mode	folder/foo-r	<i>name = 'foo', mode = 'r'</i>
file/*path	file/hello/world.txt	<i>path = 'hello/world.txt'</i>



* Wildcard matches everything
after file/

Triggering Routes

Using `navigate`

```
router.navigate("todos/1", {  
  trigger: true  
});
```

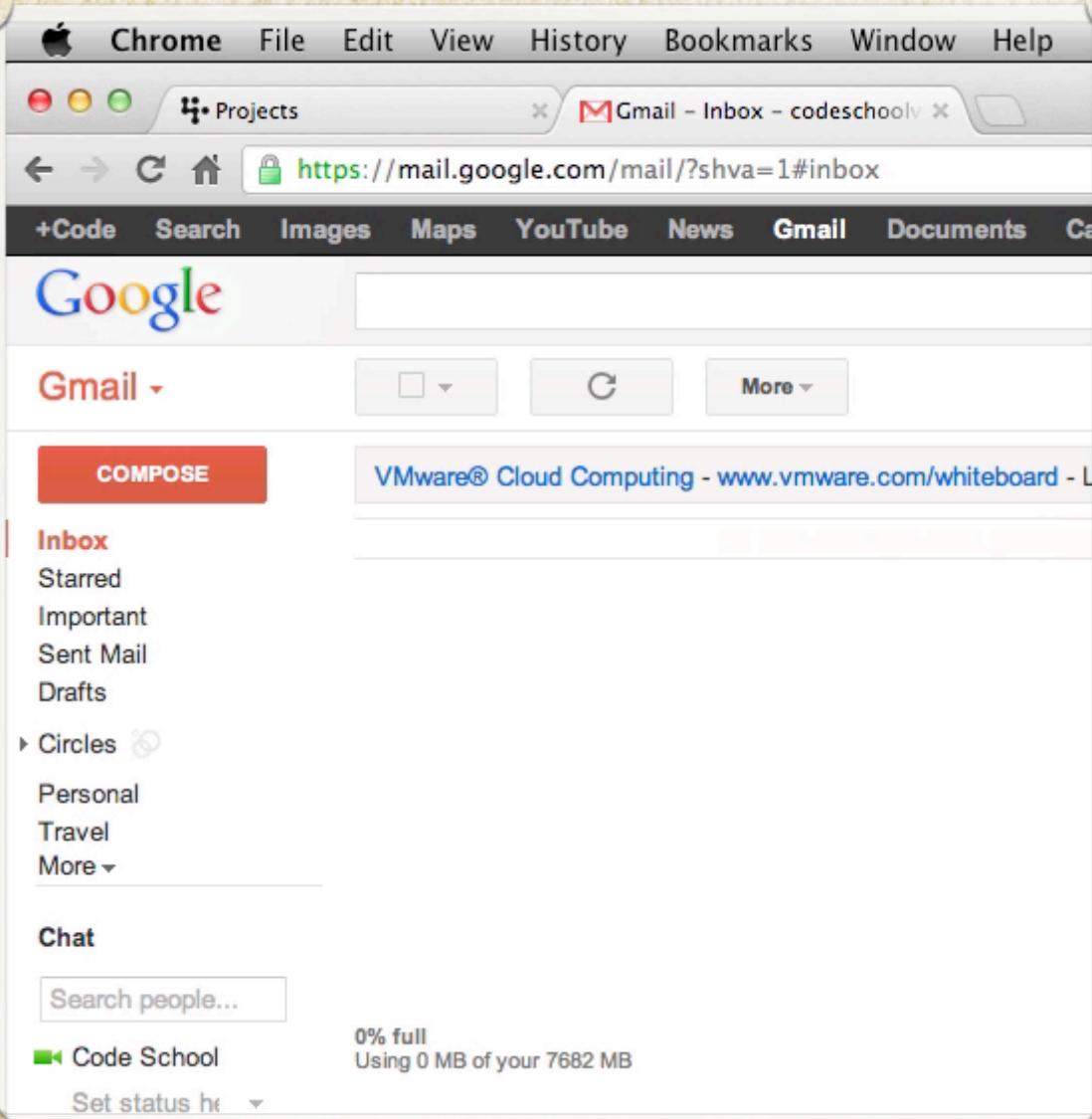
Using `links`

```
<a href="#todos/1"> ↗ </a>
```



These won't work yet

History



Router & History

Hashbangs (#)



HTML5 pushState

Backbone.js

Backbone.history

pushState off

```
Backbone.history.start();
```



```
router.navigate("todos/1") ----> #todos/1
```

pushState on

```
Backbone.history.start({pushState: true});
```



```
router.navigate("todos/1") ----> /todos/1
```

Show Action

Define router class

```
var TodoRouter = Backbone.Router.extend({  
  routes: { "todos/:id": "show" },  
  show: function(id){  
    this.todoList.focusOnTodoItem(id);  
  },  
  initialize: function(options){  
    this.todoList = options.todoList;  
  }  
});
```

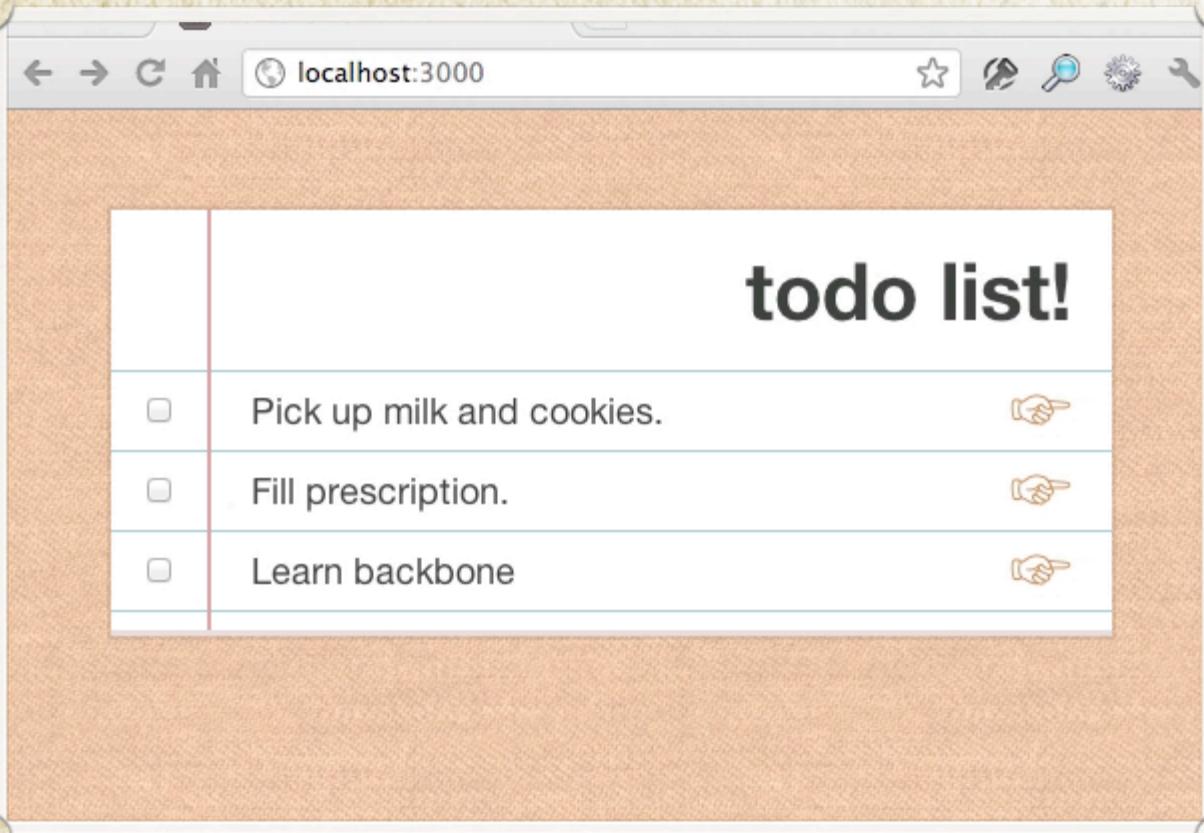
Instantiate router instance

```
var todoList = new TodoList();  
var TodoApp = new TodoRouter({todoList: todoList});
```

Index Action

```
var TodoRouter = Backbone.Router.extend({  
  routes: { "" : "index",  
            "todos/:id": "show" },  
  
  index: function(){  
    this.todoList.fetch();  
  },  
  show: function(id){  
    this.todoList.focusOnTodoItem(id);  
  },  
  initialize: function(options){  
    this.todoList = options.todoList;  
  }  
});
```

In Action



App Organization

```
var TodoApp = new Backbone.Router.extend({
  routes: { "" : "index", "todos/:id": "show" },
  initialize: function(){
    this.todoList = new TodoList();
    this.todosView = new TodoListView({collection: this.todoList});
    $('#app').append(this.todosView.el);
  },
  start: function(){
    Backbone.history.start({pushState: true});
  },
  index: function(){
    this.todoList.fetch();
  },
  show: function(id){
    this.todoList.focusOnTodoItem(id);
  }
});  
$(function(){ TodoApp.start() })
```