

More Models

- LEVEL 1 -



ReIntroducing 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

More Models



Reviewing how Model Data is fetched from Server

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

Fetch todo with id = 1

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

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

GET /todos/1

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



*RESTful Web Service
(Rails Flavor)*

More Models



Parsing non-standard JSON into your Models

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

A non-standard Response from Server

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

```
var TodoItem = Backbone.Model.extend({  
  parse: function(response){  
    return response;  
  }  
});
```

**Default Backbone
just returns response**

```
todoItem.attributes; --> { }
```



More Models




Server Breaking Conventions

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

Response from Server

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

```
var TodoItem = Backbone.Model.extend({  
  parse: function(response){  
    return response.todo;  
  }  
});
```



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

More Models



Instantiating Models doesn't call parse by Default

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



```
todoItem.attributes; --> { }
```



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



```
todoItem.attributes;
```

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

More Models



Changing Attribute Names

Response from Server

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

```
var TodoItem = Backbone.Model.extend({  
  parse: function(response){  
    response = response.todo;  
    response.description = response.desc;  
    delete response.desc;  
    return response;  
  }  
});
```



```
todoItem.attributes;
```

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

More Models



Sending JSON back to the Server

Update the todo

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

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

PUT /todos/1

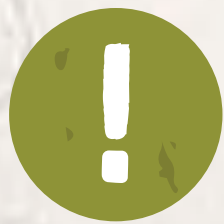
Sends to JSON() to server

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



Server expects JSON with root and desc

```
{todo:{id: 1, desc: 'Pick up cookies.', status: 'incomplete'}}
```



**So how do we modify the JSON
before we send it to the server?**

More Models



Overriding the toJSON function

```
var TodoItem = Backbone.Model.extend({  
  toJSON: function(){  
    return _.clone(this.attributes);  
  }  
});
```

default backbone



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

```
var TodoItem = Backbone.Model.extend({  
  toJSON: function(){  
    return { todo: _.clone(this.attributes) };  
  }  
});
```



```
{  
  todo: { id: 1, description: 'Pick up cookies.', status: 'incomplete' }  
}
```

More Models



Overriding the toJSON function

```
var TodoItem = Backbone.Model.extend({  
  toJSON: function(){  
    var attrs = _.clone(this.attributes);  
    attrs.desc = attrs.description;  
    attrs = _.pick(attrs, 'desc', 'status');  
    return { todo: attrs };  
  }  
});
```

*returns an object
with only the 'desc'
and 'status' properties*



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



```
{  
  todo: { desc: 'Pick up cookies.', status: 'incomplete' }  
}
```



**Now we
can't use
toJSON in
View**

More Models



Render View with Attributes

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

toJSON no longer works for templates



Use model.attributes instead

```
var TodoView = Backbone.View.extend({  
  render: function(){  
    this.$el.html(this.template(this.model.attributes));  
    return this;  
  }  
});
```



More Models



Unconventional ID Attribute

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

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

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

todoItem.id --> undefined



```
var TodoItem = Backbone.Model.extend({  
  idAttribute: '_id'  
});
```

todoItem.id --> 1



More Models



Customizing Collections

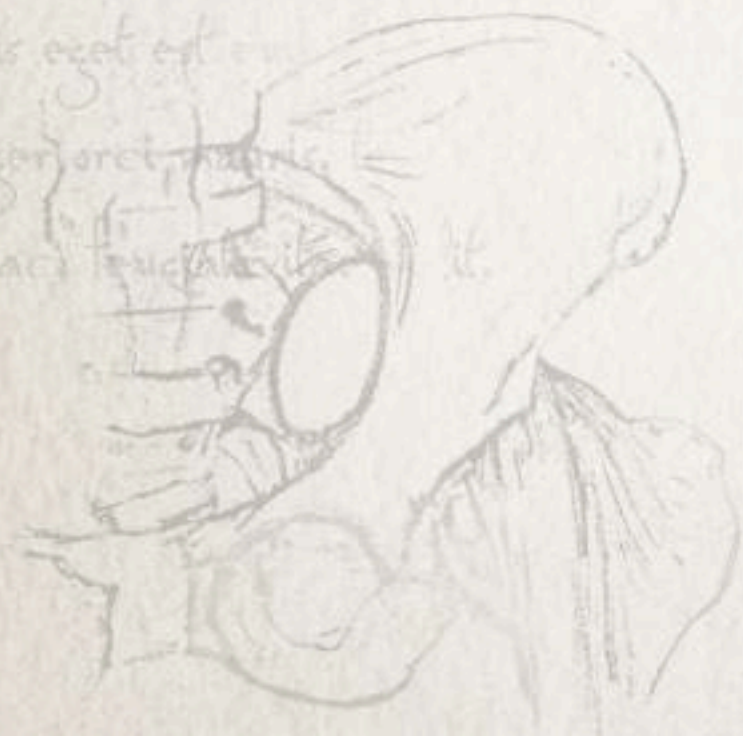
- LEVEL 2 -

amet dolor. Fusce at facilisis
velit. Maecenas lacinia turpis



non commodo enim.
lacinia lacus iaculis
Fusce at facilisis vel
turpis eget est.

Integer orci, diam
lacinia ac faucibus



quisque. Quisquam
quisque. Quisquam
quisque. Quisquam



ac a turpis in
lacinia lacus
lacinia lacus



Review fetching Data from the Server

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

*URL to get
JSON data from*

populate collection from server

`todoItems.fetch()` **GET /todos**



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

`todoItems.length` **---> 2**

Customizing Collections



Handling non-standard Response from Server

```
todoItems.fetch()
```

Response from Server

```
{  
  "total": 25, "per_page": 10, "page": 2,  
  "todos": [ {"id": 1, ... }, {"id": 2, ... } ]  
}
```

```
var TodoItems = Backbone.Collection.extend({  
  parse: function(response){  
    return response;  
  }  
});
```

```
todoItems.toJSON(); --> { }
```

Customizing Collections

Handling non-standard Response from Server

Response from Server

```
{  
  "total": 25, "per_page": 10, "page": 2,  
  "todos": [ {"id": 1}, {"id": 2} ]  
}
```

```
var TodoItems = Backbone.Collection.extend({  
  parse: function(response){  
    this.perPage = response.per_page;  
    this.page = response.page;  
    this.total = response.total;  
    return response.todos;  
  }  
});
```



todoItems.perPage --> 10

todoItems.page --> 2

todoItems.total --> 25

**These aren't
special properties**

todoItems.toJSON(); --> [{ id: 1, description: 'Pick up milk.' ... }]


Customizing Collections



How do we fetch with extra params?

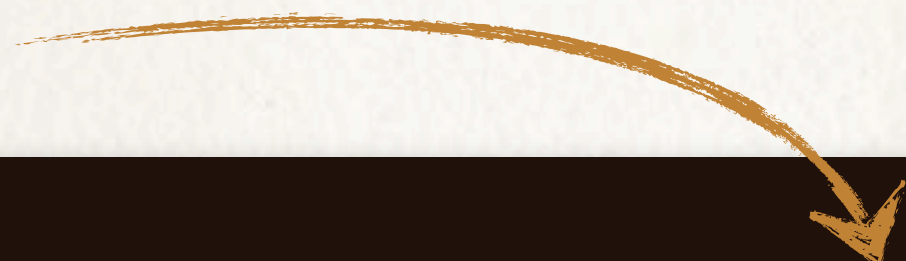

`todoItems.fetch()`

GET /todos



```
{  
  "total": 100, "per_page": 10, "page": 1,  
  "todos": [ {"id": 1}, {"id": 2} ]  
}
```

GET /todos?page=6




```
{  
  "total": 100, "per_page": 10, "page": 6,  
  "todos": [ {"id": 61}, {"id": 62} ]  
}
```

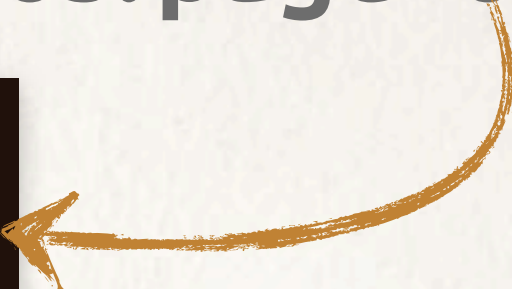

fetching with extra params

```
todoItems.fetch({data: { page: 6 }})
```

GET /todos?page=6



```
{  
  "total": 100, "per_page": 10, "page": 6,  
  "todos": [ {"id": 61}, {"id": 62} ]  
}
```




Get next page

```
todoItems.page ---> 1
```

```
todoItems.fetch({data: {page: todoItems.page + 1}});
```

GET /todos?page=2



```
{  
  "total": 100, "per_page": 10, "page": 2,  
  "todos": [ {"id": 11}, {"id": 21} ]  
}
```


Review our Collection View

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

How do we add a
“next” link?

Render the Next Link

```
var TodosView = Backbone.View.extend({
  template: _.template('<a href="#/todos/p<%= page %>">next page</a>'),
  initialize: function(){
    this.collection.on('reset', this.render, this);
  },
  render: function(){
    this.addAll();
    this.$el.append(this.template({page: this.collection.page + 1}));
    return this;
  },
  ...
});
```

Now all we have to create a route for “/todos/p:page”

Customizing Collections



Review our Router

```
var TodoApp = new (Backbone.Router.extend({
  routes: {
    "": "index"
  },
  initialize: function(){
    this.todoItems = new TodoItems();
    this.todosView = new TodosView({collection: this.todoItems});
    this.todosView.render();
    $('#app').append(this.todosView.el);
  },
  index: function(){
    this.todoItems.fetch();
  }
}));
```



Customizing Collections



Implementing the Page Route

```
var TodoApp = new (Backbone.Router.extend({
  routes: {
    "todos/p:page": "page",
    "": "index"
  },
  page: function(page){
    this.todoItems.fetch({data: {page: page}});
  },
  initialize: function(){
    this.todoItems = new TodoItems();
    this.todosView = new TodosView({collection: this.todoItems});
    this.todosView.render();
    $('#app').append(this.todosView.el);
  },
  index: function(){
    this.todoItems.fetch();
  }
}));
```

Customizing Collections



Sorting Collections by Comparator

Sort by Status

```
var TodoItems = Backbone.Collection.extend({  
  comparator: 'status'  
});
```

```
todoItems.fetch();
```



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


Sorting Collections by function

Passing a function in as a comparator

```
var TodoItems = Backbone.Collection.extend({  
  comparator: function(todo1, todo2) {  
    return todo1.get('status') < todo2.get('status')  
  }  
});
```

**Sort by status
in reverse order**

todoItems.fetch()



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

Customizing Collections



Aggregate Values

Display amount of completed todo items

```
var TodoItems = Backbone.Collection.extend({  
  completedCount: function() {  
    return this.where({status: 'complete'}).length;  
  }  
});
```

**Returns filtered
array of models**

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

`todoItems.completedCount()` -----> 1

Customizing Collections



Real Routes

- LEVEL 3 -



Optional Routes

Search route with query and optional page parameter

```
var TodoRouter = new (Backbone.Router.extend({
  routes: {
    'search/:query': 'search',
    'search/:query/p:page': 'search'
  },
  search: function(query, page) {
    page = page || 0;
    console.log(query);
    console.log(page);
  }
}));
```



**Duplicate
Routes**

TodoRouter.navigate('search/milk', {trigger: true}); --> "milk", 0


TodoRouter.navigate('search/milk/p2', {trigger: true}); --> "milk", 2

Real Routes



Optional Routes

```
var TodoRouter = new (Backbone.Router.extend({
  routes: {
    'search/:query(/p:page)': 'search',
  },
  search: function(query, page) {
    page = page || 0;
    console.log(query);
    console.log(page);
  }
}));
```



Optional
Route Part

TodoRouter.navigate('search/milk', {trigger: true}); --> "milk", 0

TodoRouter.navigate('search/milk/p2', {trigger: true}); --> "milk", 2


What if we want to match 'search/milk/p2/'?

Real Routes



Optional Trailing Slash

```
var TodoRouter = new (Backbone.Router.extend({  
  routes: {  
    'search/:query(/p:page)(/)': 'search',  
  },  
  search: function(query, page) {  
    page = page || 0;  
    console.log(query);  
    console.log(page);  
  }  
}));
```



Optional
trailing slash

```
TodoRouter.navigate('search/milk/p2/', {trigger: true}); --> "milk", 2
```

Real Routes



VRI with Spaces Gotcha

```
var TodoRouter = new (Backbone.Router.extend({
  routes: {
    'search/:query(/p:page)(/)': 'search',
  },
  search: function(query, page) {
    page = page || 0;
    console.log(query);
    console.log(page);
  }
}));
```

Encoded space

Search for "Hello World"

```
TodoRouter.navigate('search/Hello%20World/p2', {trigger: true})
```

"Hello%20World", 2



Not decoded

Real Routes



URI with Spaces Gotcha

```
var TodoRouter = new (Backbone.Router.extend({
  routes: {
    'search/:query(/p:page)(/)': 'search',
  },
  search: function(query, page) {
    page = page || 0;
    query = decodeURIComponent(query);
    console.log(query);
    console.log(page);
  }
}));
```

Encoded space

```
TodoRouter.navigate('search/Hello%20World/p2', {trigger: true})
```

"Hello World", 2



Decoded

Real Routes



Regex in Routes

Restrict parameter to numeric input

```
var TodoRouter = new (Backbone.Router.extend({
  routes: {
    'todos/:id': 'show'
  },
  show: function(id) {
    console.log("id = " + id);
  }
}));
```

`TodoRouter.navigate('todos/2', {trigger: true})` \dashrightarrow id = 2

`TodoRouter.navigate('todos/notanid', {trigger: true})`

\dashrightarrow id = notanid



Shouldn't trigger show

Real Routes



Regex in Routes

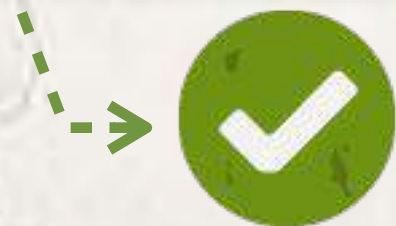
```
var TodoRouter = new (Backbone.Router.extend({  
  show: function(id) {  
    console.log("id = " + id);  
  }  
}));
```

Each Regex Capture
Group becomes a param

```
TodoRouter.route(/^todos\/(\d+)$/, 'show');
```

```
TodoRouter.navigate('todos/2', {trigger: true}) ---> id = 2
```

```
TodoRouter.navigate('todos/notanid', {trigger: true})
```



Doesn't trigger show

Real Routes



Regex in Routes Refactor

```
var TodoRouter = new (Backbone.Router.extend({
  initialize: function(){
    this.route(/^todos\/(\d+)$/, 'show');
  },
  show: function(id) {
    console.log("id = " + id);
  }
}));
```

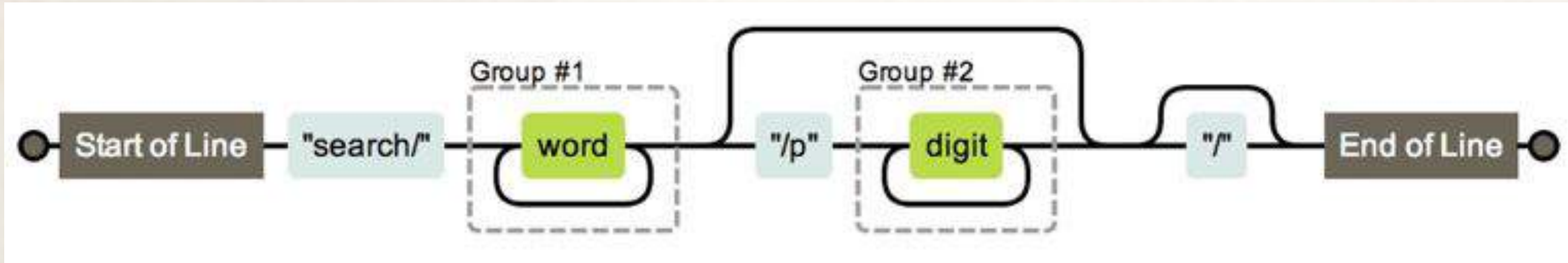
**Keep routes inside
the router**

Real Routes



Advanced Regex Routes

`/^search\/(\w+)(?:\/p(\d+))?\/?$/` --> `'search/:query(/p:page)'`



Matches

`'search/milk'`
`'search/milk/p2'`
`'search/milk/p2/'`

Doesn't Match

`'searches/milk'`
`'search/milk/p2/p1'`
`'search/milk/2'`

<http://www.regexper.com/>

Real Routes

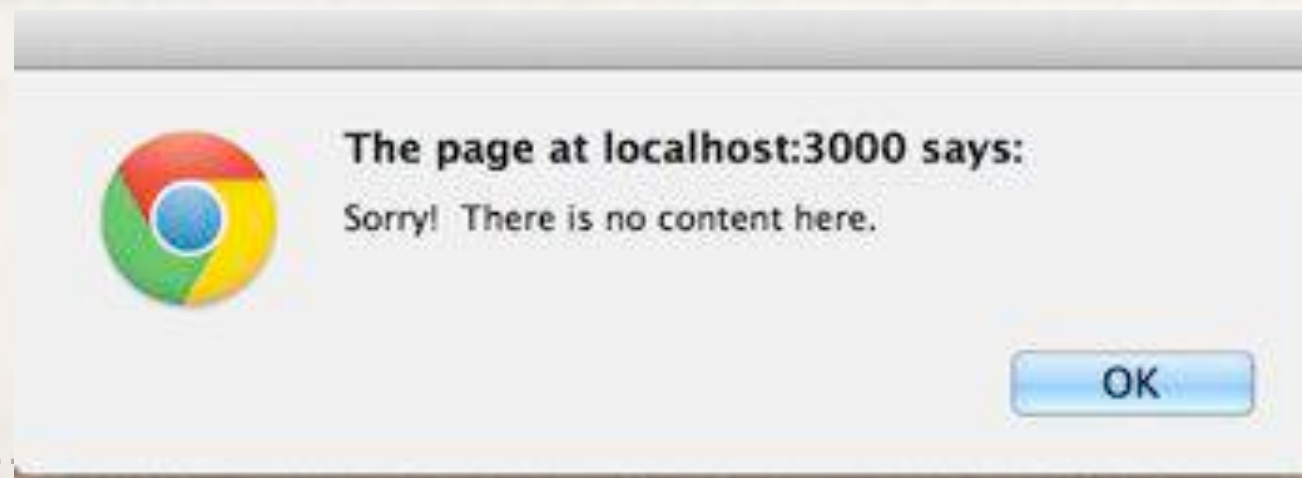


Catch-all Routes

Alert user when no route matches

```
var TodoRouter = Backbone.Router.extend({  
  routes: {  
    '*path': 'notFound'  
  },  
  notFound: function(path) {  
    alert('Sorry! There is no content here.');  }  
});
```

```
TodoRouter.navigate('nothinghere', {trigger: true})
```



Real Routes



File Path Route

Accept a file path and get its parts

```
var TodoRouter = new (Backbone.Router.extend({
  routes: {
    'file/*path': 'file'
  },
  file: function(path) {
    var parts = path.split("/");
    console.log(parts);
  }
}));
```

```
TodoRouter.navigate("file/this/is/a/file.txt", {trigger: true});
```



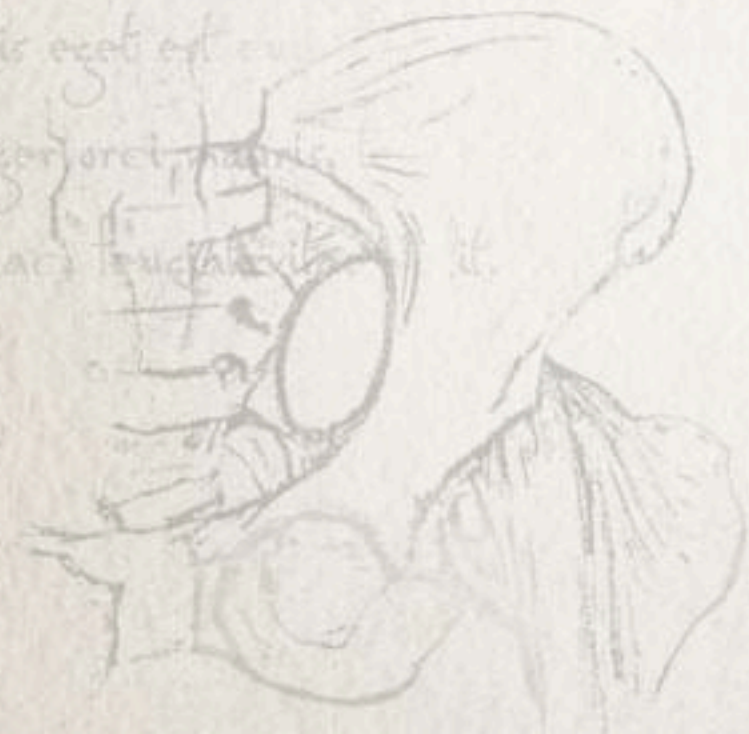
```
["this", "is", "a", "file.txt"]
```

Real Routes



Varying Views


- LEVEL 4 -



View Initialization Options Review


Pass in the model

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

 `todoView.model == todoItem`

Or the collection

```
var todoView = new TodoView({  
  collection: todoItems  
});
```

 `todoView.collection == todoItems`

**What other options
can you pass into View
initialization?**

Varying Views



Using Existing DOM Elements

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

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

```
<div class="todo">
</div>
```

```
$('.todo').html(todoView.render().el);
```



```
<div class="todo">
  <div>Pickup Milk.</div>
</div>
```

Varying Views

Using Existing DOM Elements

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

```
var todoView = new TodoView({model: todoItem, el: $('#.todo')});
```

```
<div class="todo">  
</div>
```

```
todoView.render()
```



```
<div class="todo">  
  Pickup Milk.  
</div>
```

Varying Views

Custom Initialization Options

Pass in an extra option

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

A dashed green arrow points from the `TodoView` constructor call in the code block above to the `todoView.options` property access. Another dashed green arrow points from `todoView.options` to the options object `{ "user": currentUser }`.

Access extra options in initialize

```
var TodoView = Backbone.View.extend({  
  initialize: function(options){  
    this.user = options.user;  
  }  
});
```

Varying Views



Escaping User Content

Rendering user supplied strings can lead to XSS attack

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

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

```
todoItem.set('description', "<script src='attack.js' />")
```

```
todoView.render().el
```

```
<div>
  <script src='attack.js' />
</div>
```



Varying Views

Escaping User Content

Rendering user supplied strings can lead to XSS attack

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

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

```
todoItem.set('description', "<script src='attack.js' />")
```

```
todoView.render().el
```



```
<div>
  &lt;script src=&#x27;attack.js&#x27; /&gt;
</div>
```



Varying Views



Passing extra options to event handlers

```
var TodoView = Backbone.View.extend({
  template: _.template('<span><%= description %></span>'),
  initialize: function(){
    this.model.on('change:description', this.change, this);
  },
  render: function(){
    this.$el.html(this.template(this.model.attributes));
  },
  change: function(model, value){
    this.$('span').html(value);
    this.$el.effect("highlight", {}, 1000);
  },
});
```

todoItem.set({description: "Pickup Kids"}); ---> 'change:description'

How do make a change without firing off events?

Varying Views



Passing extra options to event handlers

```
var TodoView = Backbone.View.extend({
  template: _.template('<span><%= description %></span>'),
  initialize: function(){
    this.model.on('change:description', this.change, this);
  },
  render: function(){
    this.$el.html(this.template(this.model.attributes));
  },
  change: function(model, value){
    this.$('span').html(value);
    this.$el.effect("highlight", {}, 1000);
  },
});
```

```
todoItem.set({description: "Pickup Kids"}, {silent: true});
```

How do we stop just the highlight from happening?

Varying Views



Passing extra options to event handlers

```
var TodoView = Backbone.View.extend({
  template: _.template('<span><%= description %></span>'),
  initialize: function(){
    this.model.on('change:description', this.change, this);
  },
  render: function(){
    this.$el.html(this.template(this.model.attributes));
  },
  change: function(model, value, options){
    this.$('span').html(value);
    if (options.highlight !== false){
      this.$el.effect("highlight", {}, 1000);
    }
  },
});
```

```
todoItem.set({description: "Pickup Kids"}, {highlight: false});
```



Varying Views



View Event Cleanup

```
var TodoView = Backbone.View.extend({
  initialize: function() {
    this.model.on('change', this.render, this);
  }
});
```

```
todoView.remove();
```



Model still holds reference to view

```
var TodoView = Backbone.View.extend({
  remove: function() {
    Backbone.View.prototype.remove.apply(this, arguments);
    this.model.off(null, null, this);
  }
});
```

Old Backbone

Varying Views



View Event Cleanup

```
var TodoView = Backbone.View.extend({  
  initialize: function() {  
    this.listenTo(this.model, 'change', this.render);  
  }  
});
```

View “listens to” a model



Stop all listeners for this view

```
todoView.stopListening();
```

Will automatically call stopListening()

```
todoView.remove();
```

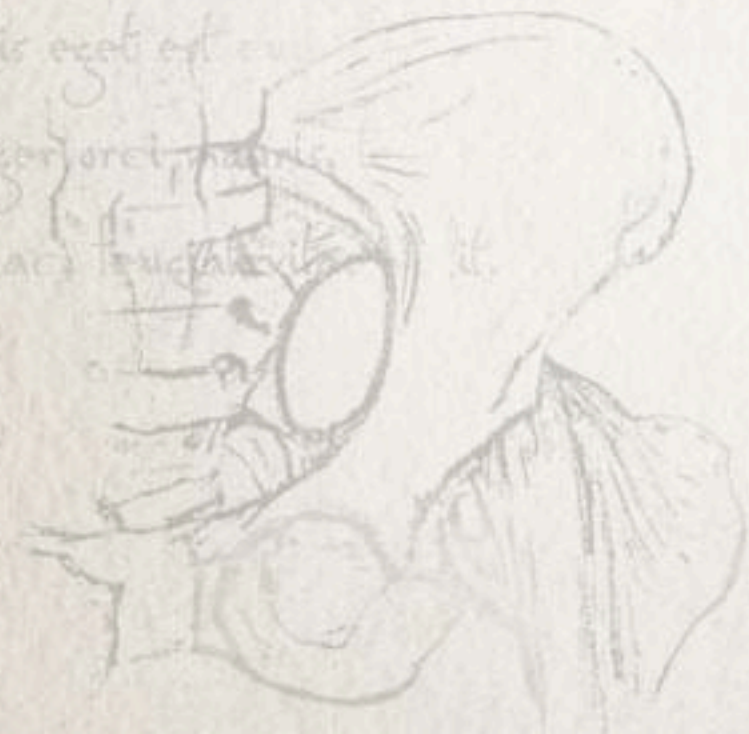
Added in Backbone 0.9.9

Varying Views



Working with forms

- LEVEL 5 -



Create an Ajax form for creating new Todos

```
<form action="/todos" method="POST">  
  <input name="description" value="What do you need to do?" />  
  <button>Save</button>  
</form>
```

Default description

Use plain jQuery

```
$('#button').click(function(e){  
  e.preventDefault();  
  var uri = $('#form').attr('action');  
  var description = $('#input[name=description]').val();  
  $.post(uri, {description: description});  
});
```

**Wait, shouldn't
we be creating Todos
using Backbone?**

 **POST /todos** { description: 'Pickup Kids.' }

Working with Forms



Convert to using Backbone

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

Default description

```
var todoItem = new TodoItem({description: "What do you need to do?"});  
todoItem.save({description: 'Pickup Kids.'});
```

 **POST /todos** { description: 'Pickup Kids.' }

What about the form?

Working with Forms



Build form with Backbone View

```
var TodoForm = Backbone.View.extend({
  template: _.template('<form>' +
    '<input name=description value="<%= description %>" />' +
    '<button>Save</button></form>'),
  render: function(){
    this.$el.html(this.template(this.model.attributes));
    return this;
  }
});
```

```
var todoItem = new TodoItem({description: "What do you need to do?"});
var todoForm = new TodoForm({model: todoItem});
$('#app').html(todoForm.render().el);
```

→ `<form><input name=description value="What do you need to do?" />
<button>Save</button></form>`

Working with Forms



Capture Button Click to Save Model

```
var TodoForm = Backbone.View.extend({
  template: _.template('<form>' +
    '<input name=description value="<%= description %>" />' +
    '<button>Save</button></form>'),
  events: {
    'click button': 'save'
  },
  save: function(e) {
    e.preventDefault();
    var newDescription = this.$('input[name=description]').val();
    this.model.save({description: newDescription});
  }
});
```

POST /todos { description: 'Pickup Kids.' }

How do we save when pressing enter?

Working with Forms



Capture Return Key to Save Model

```
var TodoForm = Backbone.View.extend({
  template: _.template('<form>' +
    '<input name=description value="<%= description %>" />' +
    '<button>Save</button></form>'),
  events: {
    submit: 'save'
  },
  save: function(e) {
    e.preventDefault();
    var newDescription = this.$('input[name=description]').val();
    this.model.save({description: newDescription});
  }
});
```

Will call save on either click or pressing Enter/Return

Working with Forms



Reusing form to Edit existing TodoItem

Get existing TodoItem from already fetched collection

```
var todoItem = todoItems.get(1);
```

Pass in existing model

```
var editTodoForm = new TodoForm({model: todoItem});
```

Replace #app with the HTML of the form

```
$('#app').html(editTodoForm.render().el);
```

A screenshot of a web form. It features a single text input field containing the text "Pickup Kids from School". To the right of the input field is a button labeled "Save". The form is set against a light orange background.

Working with Forms

Review of our App's Router

```
var TodoApp = new (Backbone.Router.extend({
  routes: {
    "": "index"
  },
  initialize: function(){
    this.todoItems = new TodoItems();
    this.todosView = new TodosView({collection: this.todoItems});
  },
  index: function(){
    this.todoItems.fetch();
    $('#app').html(this.todosView.render().el);
  }
}));
```

**Let's add a route to render the
TodoForm with an existing TodoItem**

Working with Forms



Add Route to render Edit form

```
var TodoApp = new (Backbone.Router.extend({
  routes: {
    "": "index",
    "todos/:id/edit": "edit"
  },
  initialize: function(){
    this.todoItems = new TodoItems();
    this.todosView = new TodosView({collection: this.todoItems});
  },
  index: function(){
    this.todoItems.fetch();
    $('#app').html(this.todosView.render().el);
  }
  edit: function(id){
    var todoForm = new TodoForm({model: this.todoItems.get(id) });
    $('#app').html(todoForm.render().el);
  }
}));
```

Working with Forms



Add Route to render New Form

```
var TodoApp = new (Backbone.Router.extend({
  routes: {
    "": "index",
    "todos/:id/edit": "edit",
    "todos/new": "newTodo"
  },
  ...

  newTodo: function(){
    var todoItem = new TodoItem({description: "What do you have to do?"});
    var todoForm = new TodoForm({model: todoItem});
    $('#app').append(todoForm.render().el);
  },

  edit: function(id){
    var todoForm = new TodoForm({model: this.todoItems.get(id) });
    $('#app').html(todoForm.render().el);
  }
}));
```

Working with Forms



Get Back to the List after saving

```
var TodoForm = Backbone.View.extend({  
  ...  
  save: function(e) {  
    e.preventDefault();  
    var newDescription = this.$('input[name=description]').val();  
    this.model.save({description: newDescription});  
    Backbone.history.navigate('', { trigger: true });  
  }  
});
```



**We should only call this if
the model saved successfully**

Get Back to the List after saving success

```
var TodoForm = Backbone.View.extend({
  ...
  save: function(e) {
    e.preventDefault();
    var newDescription = this.$('input[name=description]').val();
    this.model.save({description: newDescription}, {
      success: function(model, response, options){
        Backbone.history.navigate('', { trigger: true });
      }
    });
  }
});
```



What happens if the Todo doesn't save?

Alert User to save error

```
var TodoForm = Backbone.View.extend({
  ...
  save: function(e) {
    e.preventDefault();
    var newDescription = this.$('input[name=description]').val();
    this.model.save({description: newDescription}, {
      success: function(model, response, options){
        Backbone.history.navigate('', { trigger: true });
      }, error: function(model, xhr, options){
        var errors = JSON.parse(xhr.responseText).errors;
        alert('Oops, something went wrong with saving the TodoItem: ' + errors);
      }
    });
  }
});
```



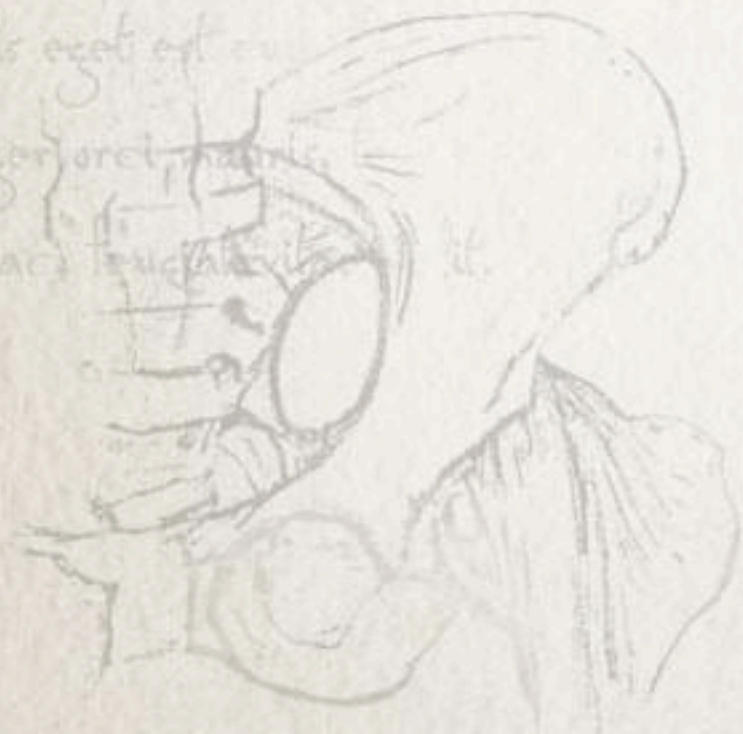
App Organization

- LEVEL 6 -

amet dolor. Fusce at facilisis
velit. Maecenas lacinia turpis



non commodo enim.
tum lacus iaculis
usce at facilisis veli
turpis eget est eu



Integer orci, nunc
tum ac tunc

quisque congue odio porta. Integer or
quisque congue odio porta. Integer or



ac a turpis in
Nam a lacus
curtus



quisque congue odio porta. Integer or
quisque congue odio porta. Integer or

Class naming

Everything in the Global Scope

```
var TodoItem = Backbone.Model.extend({});  
var TodoItemView = Backbone.View.extend({});  
var TodoItems = Backbone.Collection.extend({});  
var TodoItemsView = Backbone.View.extend({});  
var TodoRouter = Backbone.Router.extend({});
```



- Leads to naming collisions
- Need to put “what kind of object” it is in the name e.g. “TodoItemView”
- Maintainability doesn’t scale with large applications

Use a Global Object for Namespace

```
var App = {  
  Models: {},  
  Views: {},  
  Collections: {}  
}
```

*Create a single global object
where everything is stored*

```
App.Models.TodoItem = Backbone.Model.extend({});  
App.Views.TodoItem = Backbone.View.extend({});  
App.Collections.TodoItems = Backbone.Collection.extend({});  
App.Views.TodoItems = Backbone.View.extend({});  
App.TodoRouter = Backbone.Router.extend({});
```

Reference classes with the namespace

```
var todoItem = new App.Models.TodoItem({...})
```

Store one-off objects on App

LEVEL TITLE



Handle Links Outside of Backbone Views

```
<ul>
  <li><a href="/completed">Show Completed</a></li>
  <li><a href="/support">Support</a></li>
</ul>
```

Using plain jQuery

```
$('a').click(function(e){
  e.preventDefault();
  Backbone.history.navigate(e.target.pathname, {trigger: true});
});
```



What if we wanted backbone to handle this?

LEVEL TITLE



Handle Links Outside of Backbone Views

```
var App = {  
  Models: {},  
  Views: {},  
  Collections: {},  
  start: function(){  
    $('a').click(function(e){  
      e.preventDefault();  
      Backbone.history.navigate(e.target.pathname, {trigger: true});  
    });  
  
    Backbone.history.start({pushState: true});  
  }  
}
```

```
$(function(){ App.start(); })
```

! Outside of Backbone Views

LEVEL TITLE



Handle Links Outside of Backbone Views

```
var App = Backbone.View.extend({  
  Models: {},  
  Views: {},  
  Collections: {},  
  events: {  
    'click a': function(e){  
      e.preventDefault();  
      Backbone.history.navigate(e.target.pathname, {trigger: true});  
    }  
  },  
  start: function(){  
    Backbone.history.start({pushState: true});  
  }  
});
```

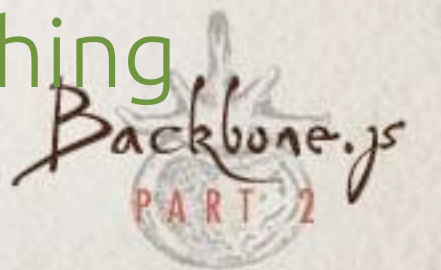
App is now a View Class

Use View event handling for clicks

```
var app = new App({el: document.body});
```

Using body will capture everything

LEVEL TITLE



Handle Links Outside of Backbone Views

```
var App = new (Backbone.View.extend({
  ...
  events: {
    'click a': function(e){
      e.preventDefault();
      Backbone.history.navigate(e.target.pathname, {trigger: true});
    }
  },
  start: function(){
    Backbone.history.start({pushState: true});
  }
}))({el: document.body});
```



Create instance without
creating a class

```
$(function(){ App.start(); })
```

LEVEL TITLE



Skip links

```
<ul>
  <li><a href="/completed">Show Completed</a></li>
  <li><a href="/support">Support</a></li>
</ul>
```

Event Handler

```
events: {
  'click a': function(e){
    e.preventDefault();
    Backbone.history.navigate(e.target.pathname, {trigger: true});
  }
}
```



Captures all link clicks

LEVEL TITLE




Skip links

```
<ul>
  <li><a href="/completed" data-internal="true">Show Completed</a></li>
  <li><a href="/support">Support</a></li>
</ul>
```

Event Handler

```
events: {
  'click a[data-internal]': function(e){
    e.preventDefault();
    Backbone.history.navigate(e.target.pathname, {trigger: true});
  }
}
```



Captures only the links we want it to

Build Initial HTML

```
var App = new (Backbone.View.extend({  
  ...  
  template: _.template('<h1>ToDo List!</h1>' +  
    '<div id="app"></div>'),  
  render: function(){  
    this.$el.html(this.template());  
  }  
}))( {el: document.body});
```



```
$(function(){  
  App.render();  
  App.start();  
})
```

```
<body>  
  <h1>ToDo List!</h1>  
  <div id="app"></div>  
</body>
```

*Add Initial HTML
to the body*

LEVEL TITLE



Object Initialization

```
var App = new (Backbone.View.extend({
  ...
  start: function(){
    var todos = new App.Collections.TodoItems();
    var todosView = new App.Views.TodoItems({collection: todos});
    this.$el.append(todosView.render().el);
    todos.fetch();
  }
}))(document.body);
```



Ajax slows down initial render

```
$(function(){ App.start(); })
```


Bootstrap Model Data

```
var App = new (Backbone.View.extend({
  start: function(bootstrap){
    var todos = new App.Collections.TodoItems(bootstrap.todos);
    var todosView = new App.Views.TodoItems({collection: todos});
    this.$el.append(todosView.render().el);
  }
}))({el: document.body});
```

```
var bootstrap = {
  todos: [
    {id: 1, description: "Pickup Milk.", status: "complete"},
    {id: 2, description: "Pickup Kids.", status: "incomplete"},
  ]
}
```

Bootstrap data
comes from
rendered HTML page

```
$(function(){ App.start(bootstrap); })
```

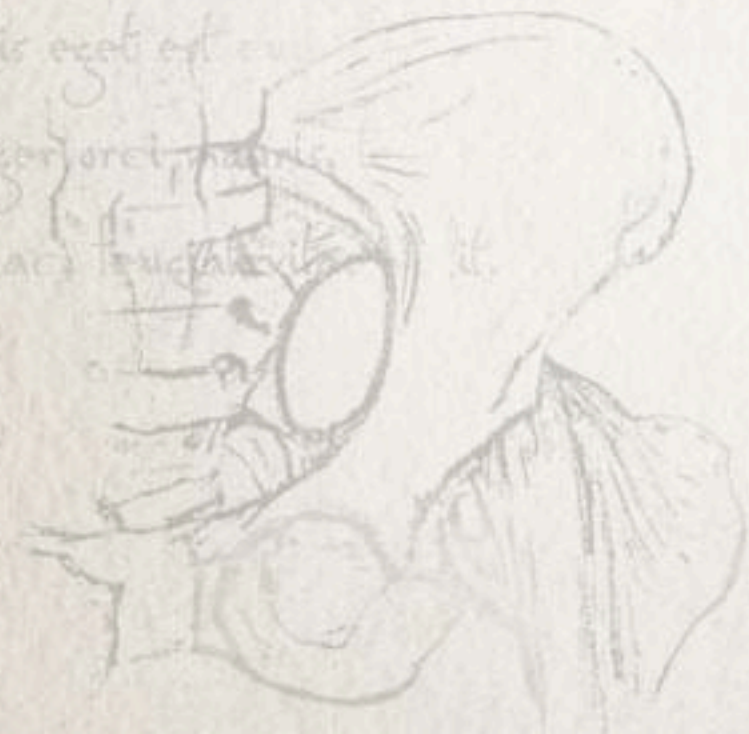
Pass in data to start

LEVEL TITLE



Customizing Backbone

- LEVEL 7 -



Using other template engines

Views don't care how you use templates

```
var TodoItemView = Backbone.View.extend({
  banana: _.template("<span><%= description %></span>" +
    "<em><%= assigned_to %></em>"),

  render: function(){
    this.$el.html(this.banana(this.model.attributes));
  }
});
```

! Very easy to use other template libraries

Mustache.js

handlebars

EJS

Google Clojure Templates

Customizing Backbone



Using Mustache.js

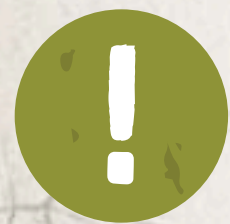
Underscore template

```
_.template("<span><%= description %></span>" +  
  "<em><%= assigned_to %></em>")
```

{{mustache template}}



```
Mustache.compile("<span>{{ description }}</span>" +  
  "<em>{{ assigned_to }}</em>")
```



Mustache doesn't allow arbitrary js

Using Mustache.js in Backbone View

```
var TodoItemView = Backbone.View.extend({
  template: Mustache.compile("<span>{{ description }}</span>" +
    "<em>{{ assigned_to }}</em>"),
  render: function(){
    this.$el.html(this.template(this.model.toJSON()));
  }
});
```

Mustache.compile() returns a function just like `_.template()`

Mustache.js templates

View

```
{ "name": "Eric" }
```

underscore.js template

```
<li><%= name %></li>
```

Mustache.js template

```
<li>{{name}}</li>
```

How would you render an array of names?

Customizing Backbone



Mustache.js Sections

Render array of names

```
{ "names": ["Eric", "Nate", "Jacob"] }
```

underscore.js template

```
<% _.each(names, function(name) { %>  
  <li><%= name %></li>  
<% }); %>
```

Strange and verbose

Mustache.js template

```
{{#names}}  
  <li>{{.}}</li>  
{{/names}}
```

“.” refers to each string
in the array

Mustache.js templates cont.

```
{  
  "people": [  
    { name: "Eric", hairColor: "brown" },  
    { name: "Nate", hairColor: "blond" },  
    { name: "Jacob", hairColor: "blue" },  
  ]  
}
```

```
<% _.each(people, function(person) { %>  
  <li><%= person.name %> has <%= person.hairColor %> hair</li>  
<% }); %>
```

underscore.js

```
{{#people}}  
  <li>{{ name }} has {{ hairColor }} hair</li>  
{{/people}}
```

Mustache.js

More Mustache.js Sections

View

```
{ "completed": false }
```

Template

```
Are you done?  
{{#completed}}  
  <em>Done!</em>  
{{/completed}}
```

Output

```
Are you done?
```



```
{ "names": [] }
```

```
<ul>  
  {{#names}}  
    <li>{{.}}</li>  
  {{/names}}  
</ul>
```

```
<ul>  
</ul>
```



Inverted

```
{ "completed": false }
```

```
Are you done?  
{{^completed}}  
  <em>Nope!</em>  
{{/completed}}
```

```
Are you done?  
<em>Nope!</em>
```



Mustache.js function Sections

View

```
{  
  name: "Eric",  
  header: function(){  
    return function(text, render){  
      return "<h1>" + render(text) + "</h1>";  
    }  
  }  
}
```

Hello {{name}}.

Template

```
{{#header}}  
Hello {{name}}.  
{{/header}}
```

Output

<h1>Hello Eric.</h1>

Default ~~RESTful~~ persistence Strategy

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

Read

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

GET /todos/1

Update

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

PUT /todos/1

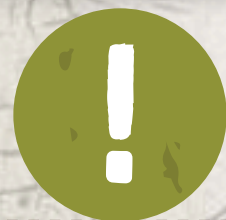
Delete

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

DELETE /todos/1

Create

```
(new TodoItem({description: "Pickup Kids"})).save()
```

POST /todos

How do we make TodoItem read-only?

Customizing Backbone



Make Read-Only Model

```
var TodoItem = Backbone.Model.extend({
  sync: function(method, model, options){
    if (method === "read"){
      Backbone.sync(method, model, options);
    }else{
      console.error("You can not " + method + " the TodoItem model");
    }
  }
});
```

`method` = `"read"` , `"create"` , `"update"` , or `"delete"`

`todoItem.fetch();`



`todoItem.save();`

You can not update the TodoItem model

Completely Replace Persistence Strategy

```
var TodoItem = Backbone.Model.extend({
  sync: function(method, model, options){
    options || (options = {});

    switch(method){
      case 'create':
        break;
      case 'read':
        break;
      case 'update':
        break;
      case 'delete':
        break;
    }
  }
});
```

**How would we replace
a server with localStorage?**

Persistent Key/Value Store for the Web

```
localStorage.setItem(<key>, <value>)
```

```
localStorage.setItem("animal", "Dog")
```

```
localStorage.getItem("animal") ---➤ "Dog"
```

```
localStorage.removeItem("animal")
```

```
localStorage.getItem("animal") ---➤ undefined
```

Object Syntax

```
localStorage["animal"] = "Cat"
```

```
localStorage["animal"] ---➤ "Cat"
```

Browser Support

IE 8.0+

Firefox 3.5+

Safari 4.0+

Chrome 4.0+

Opera 10.5+

iPhone 2.0+

Android 2.0+

Implement the Create Method

```
case 'create':  
  var key = "TodoItem-" + model.id;  
  localStorage.setItem(key, JSON.stringify(model));  
break;
```

TodoItem-1

Automatically calls
toJSON on model

```
(new TodoItem({id: 1, description: "Pickup Kids"})).save()
```



Key	Value
TodoItem-1	{"id":1,"description":"Pickup Kids"}

Implement the Read Method

```
case 'read':  
  var key = "TodoItem-" + model.id;  
  var result = localStorage.getItem(key);  
  if (result){  
    result = JSON.parse(result);  
    options.success && options.success(result);  
  }  
  break;
```

Pass result to success
callback

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

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

```
todoItem.attributes; ---> { "id": 1, "description": "Pickup Kids" }
```

Customizing Backbone



Implement the Read Method

```
case 'read':  
  var key = "TodoItem-" + model.id;  
  var result = localStorage.getItem(key);  
  if (result){  
    result = JSON.parse(result);  
    options.success && options.success(result);  
  } else if (options.error){  
    options.error("Couldn't find TodoItem id=" + model.id);  
  }  
  break;
```

```
var todoItem = new TodoItem({id: 2})
```

```
todoItem.fetch({error: function(m){ alert(m) }});
```

Pass message to
error callback

Backbone.localStorage

```
<script src="backbone-localstorage.js" />
```

 <https://github.com/jeromegn/Backbone.localStorage>

```
var TodoItems = Backbone.Collection.extend({  
  model: TodoItem,  
  localStorage: new Backbone.LocalStorage("TodoItems")  
});
```

Key	Value
TodoItems	1,2
TodoItems-1	{"id":1,"description":"Pickup Milk.,"status":"incomplete"}
TodoItems-2	{"id":2,"description":"Pickup Kids.,"status":"incomplete"}