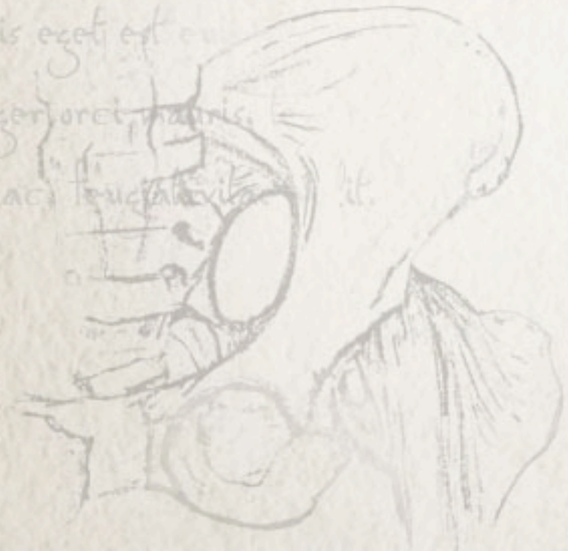
Customizing Backbone

- 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

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" }
```

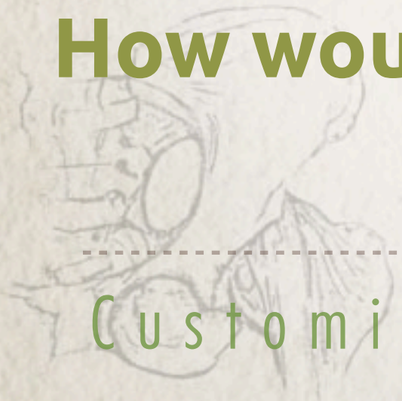
undercore.js template

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

Mustache.js template

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

How would you render an array of names?



Mustache.js Sections

Render array of names

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

undercore.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>
<% }); %>
```

undercore.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?
```

View

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

Template

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

Output

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

Inverted

View

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

Template

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

Output

```
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?

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" }
```

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"}

Customizing Backbone

- LEVEL 7 -