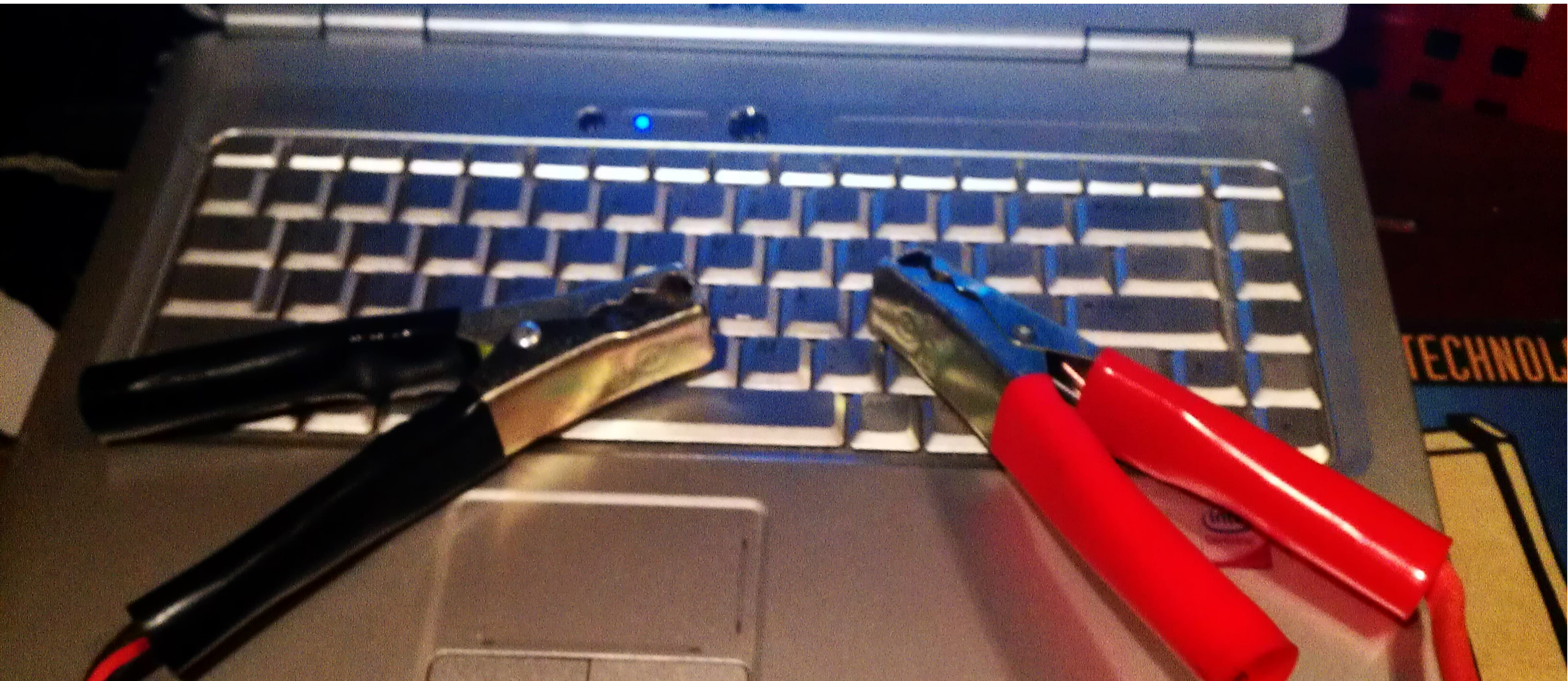
Backbone.js in a Php Environment

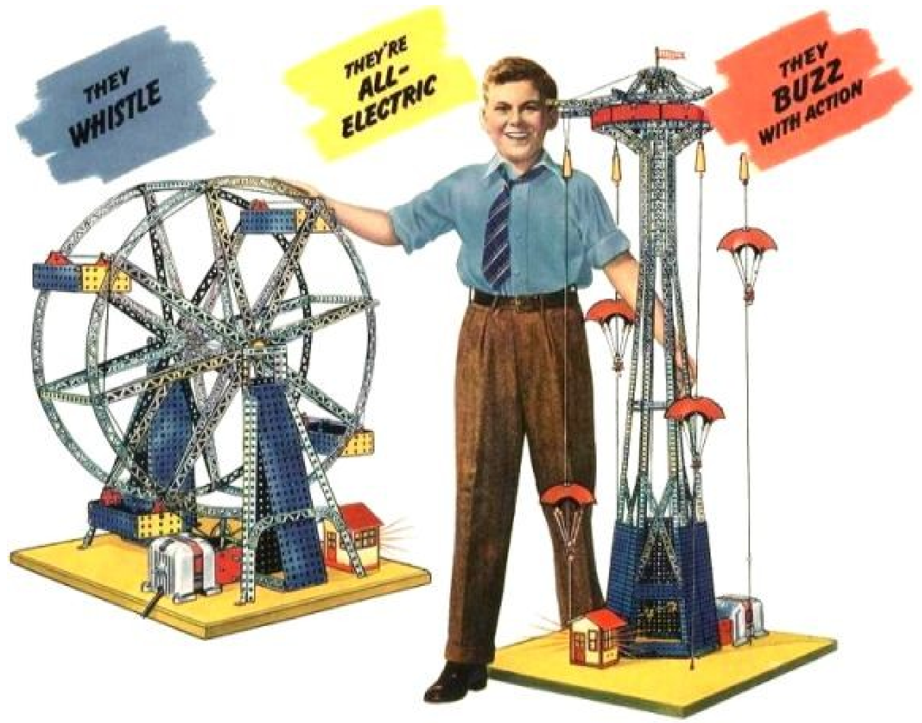
Midwest Php – March 2, 2013

Ken Harris – Sr. Developer, Telkonet.com
Milwaukee, WI



Trends in Web Apps

- Fatter Clients
- Desktop style apps
- Lots of Javascript
- Lots of CSS
- Requires structure



Backbone.js

- Framework for client side Javascript
- Organize and structure large JS applications
- Lightweight – only 4kb minified
- Becoming popular

Backbone.js

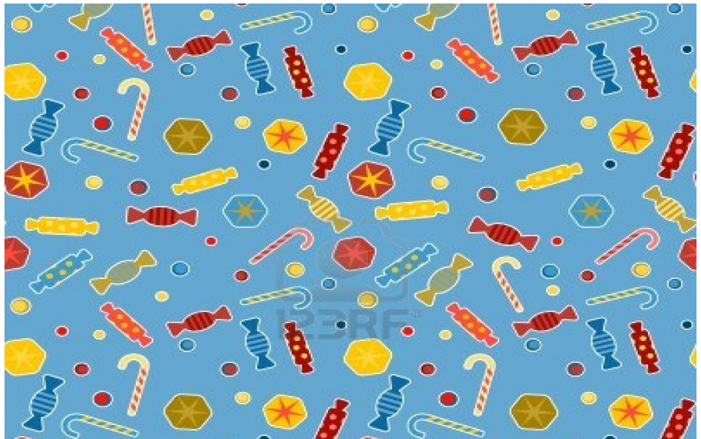
- Agnostic & very flexible
- Mixes well with other frameworks & modules
- Lots of ways to do things
- Two Backbone apps may look quite different
- Especially good for migrating an app from server side to client side rendering

Backbone components

- events
- models
- collections
- views
- routers

MVC Design Pattern?

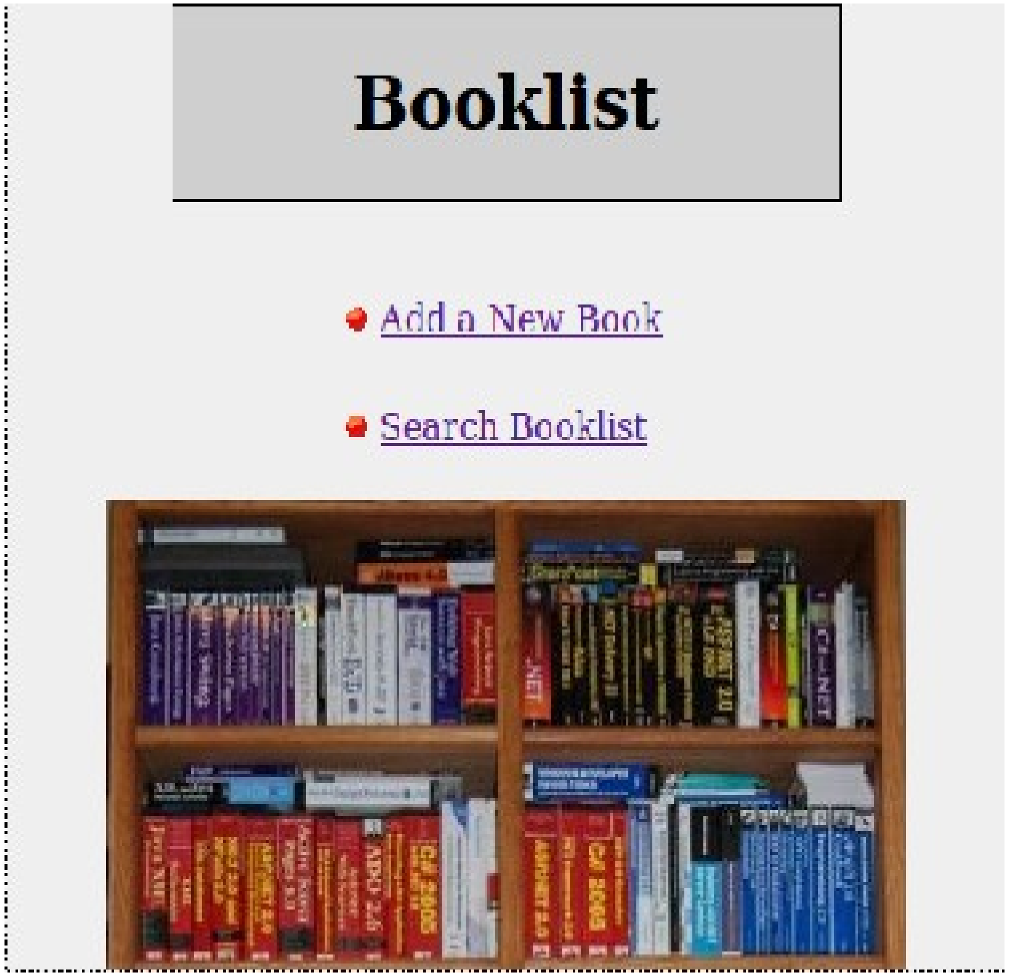
- Backbone is not classic MVC
- Model – Data objects for resources
- View – Like a view controller. Control and render templates.
- Templates – Correspond to MVC view. Can use template facility of choice. Get rendered by the view.
- Routers – Like Rails routes.



Requirements

- Underscore.js – support library
  - Especially for collections
- Jquery or Zepto
  - Uses jquery.ajax to access server DB
- Json2.js for RESTful persistence

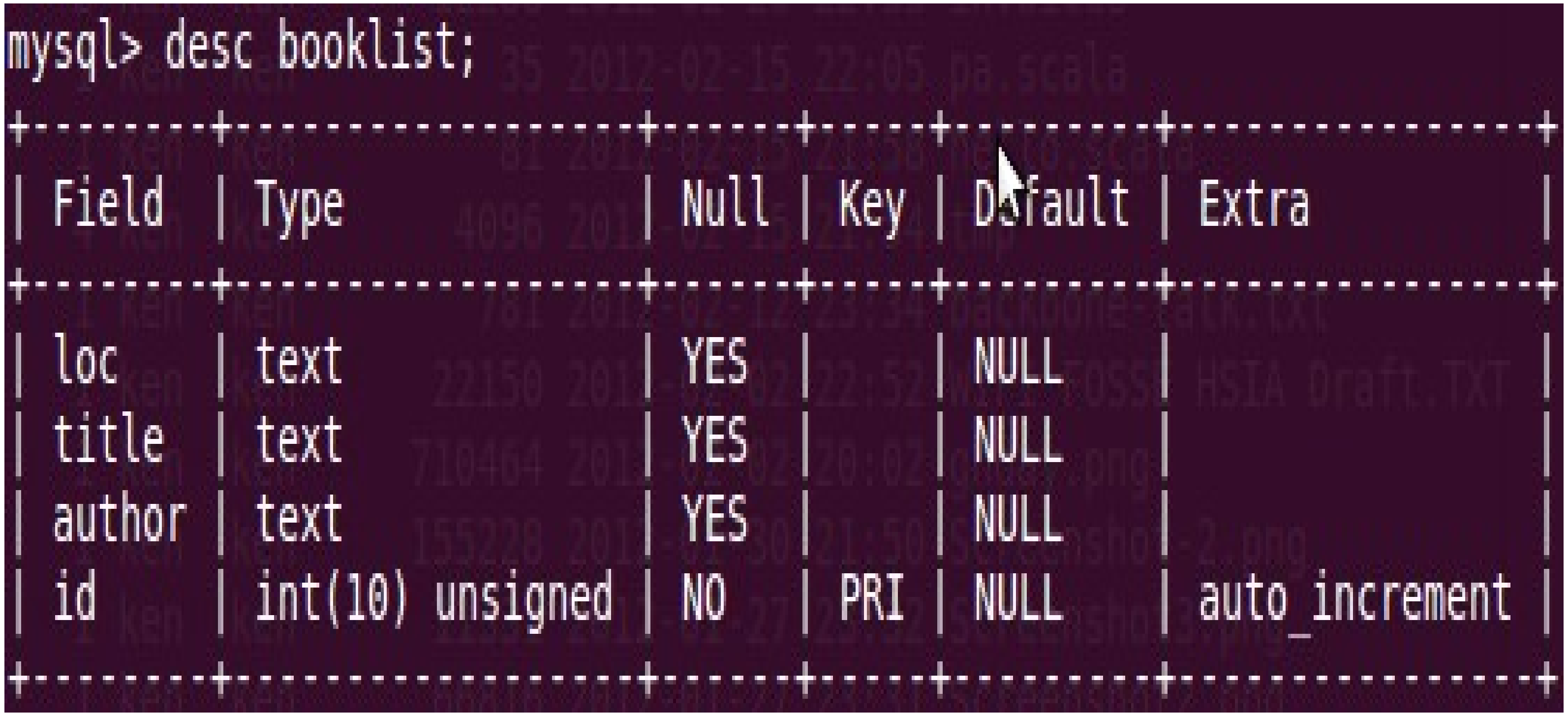
Booklist Example



Booklist

- Add a New Book
- Search Booklist

Booklist Example



```
mysql> desc booklist;
+--------+------------------+------+-----+---------+----------------+
| Field  | Type             | Null | Key | Default | Extra          |
+--------+------------------+------+-----+---------+----------------+
| loc    | text             | YES  |     | NULL    |                |
| title  | text             | YES  |     | NULL    |                |
| author | text             | YES  |     | NULL    |                |
| id     | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
+--------+------------------+------+-----+---------+----------------+
```

Backbone.Events

- Mixin module – can be added to any object
- Bind, trigger, and unbind
- Events are strings (eg. “change”)
- Trigger can pass arguments
- Changes to data models can trigger automatic refresh of views

Backbone.Events

```
var object = {};
_.extend(object, Backbone.Events);

object.bind(“poke”, function(msg) {
    alert(“Just got poked: “+msg); });

object.trigger(“poke”, “hello...”);
```

Backbone.Model

- Heart of the application
- Data + logic (conversions, validations, computed properties, access control, …)
- Create new model by extending Backbone.Model
- Create instance with new

Backbone.Model

```
var Book = Backbone.Model.extend({
            initialize: function() { // do something
                                    },
            defaults: { loc : '', title : '', author : ''},
            urlRoot: '/booklist/db.php/'
    });

var book = new Book;
```

Backbone.Model

- Properties
  - model.id – unique server id
  - model.cid – client id
  - model.defaults – default attribute values
- Methods
  - Initialize() - called on creation
  - get(attribute) - getter
  - set(attributes) - setter
  - validate() - validation

Backbone.Model

- More methods
  - fetch()
  - save()
  - destroy()
  - toJSON()
- Events
  - change, destroy, error
  - change event can be bound to render a view

Server Interaction

- Makes RESTful calls to server
- CRUD (create,read,update,delete)
- Create → POST server/collection/id
- Read → GET server/model/id
- Update → PUT
- Delete → DELETE
- Data passed using JSON encoding

Example Requests

- METHOD host/table/id
- GET http://myserver/customer/16
- GET http://myserver/salesclass
- POST http://myserver/payment
- DELETE http://myserver/appointment/2137

Backbone.Collection

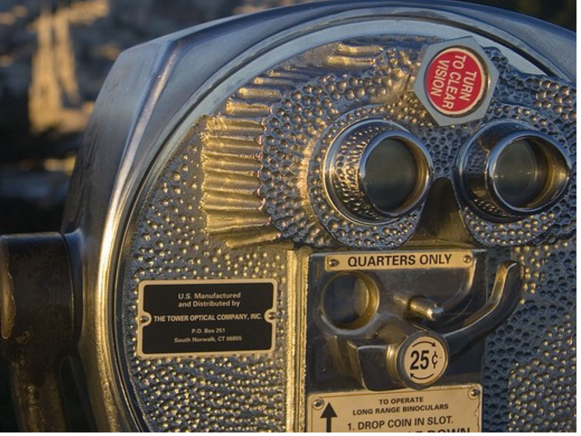
- Ordered set of models
- Can listen for events on any model
- Create by extending Backbone.Collection
- Create an instance using new

Backbone.Collection

```
var Booklist = Backbone.Collection.extend({
                model : Book,
                url : '/phpdemo/db.php'
          });

var booklist = new Booklist;
```

Backbone.Collection

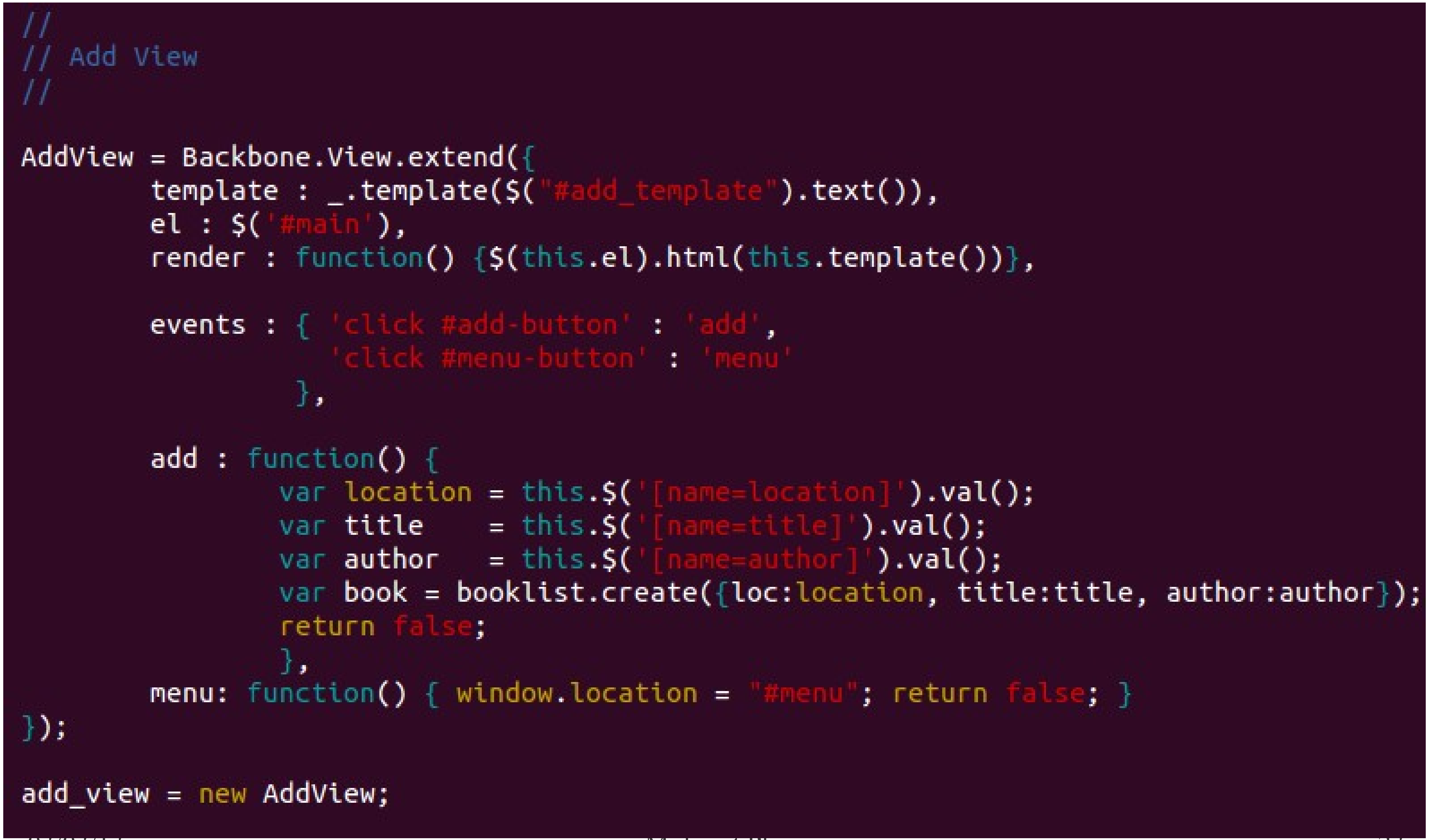
- Methods
  - add()
  - remove()
  - get(id)
  - getByCid(cid)
  - Comparator – ordering function
  - Uses underscore.js for iteration methods



Backbone.View

- Control piece that renders a view
- Extend Backbone.View to create new view
- Use new to create an instance
- View.el – associated DOM element ($el)
- initialize() function
- render() function
- $(selector) – locally scoped jQuery

Backbone.View

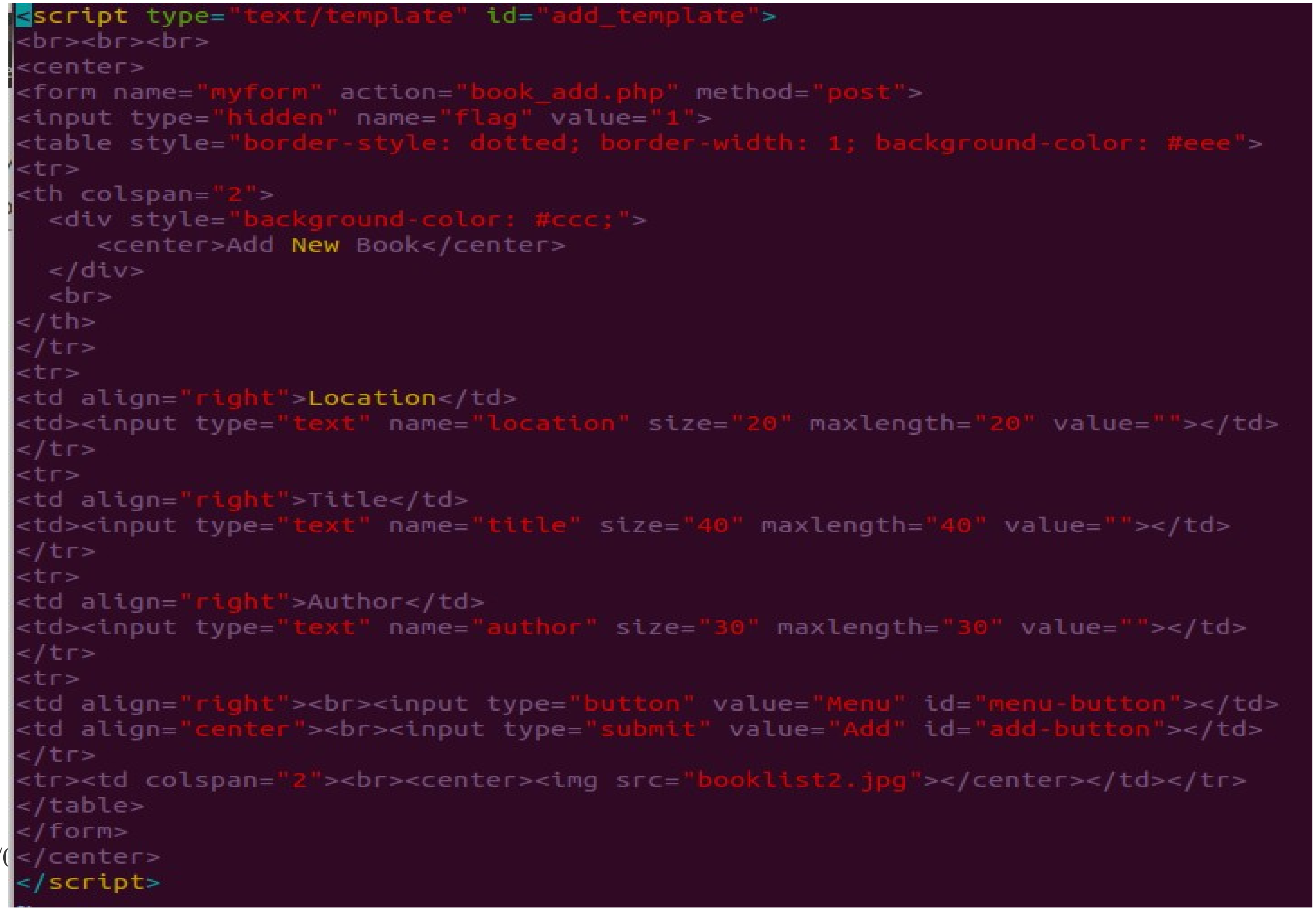


```
//
// Add View
//

AddView = Backbone.View.extend({
        template : _.template($("#add_template").text()),
        el : $('#main'),
        render : function() {$(this.el).html(this.template())},

        events : { 'click #add-button' : 'add',
                   'click #menu-button' : 'menu'
                 },

        add : function() {
                var location = this.$('[name=location]').val();
                var title    = this.$('[name=title]').val();
                var author   = this.$('[name=author]').val();
                var book = booklist.create({loc:location, title:title, author:author});
                return false;
                },
        menu: function() { window.location = "#menu"; return false; }
});

add_view = new AddView;
```

underscore templates

- _.template(“text”) returns a template function which merges properties into the text
- Can embed template in dummy script tag
- <script type=”text/template” id=”add_template”> </script>
- <%= value %>   or   <% javascript %>
- template1 = _.template(“Hi <%= name %>”);
- result = template1({ name : “bob”});

add.tmpl

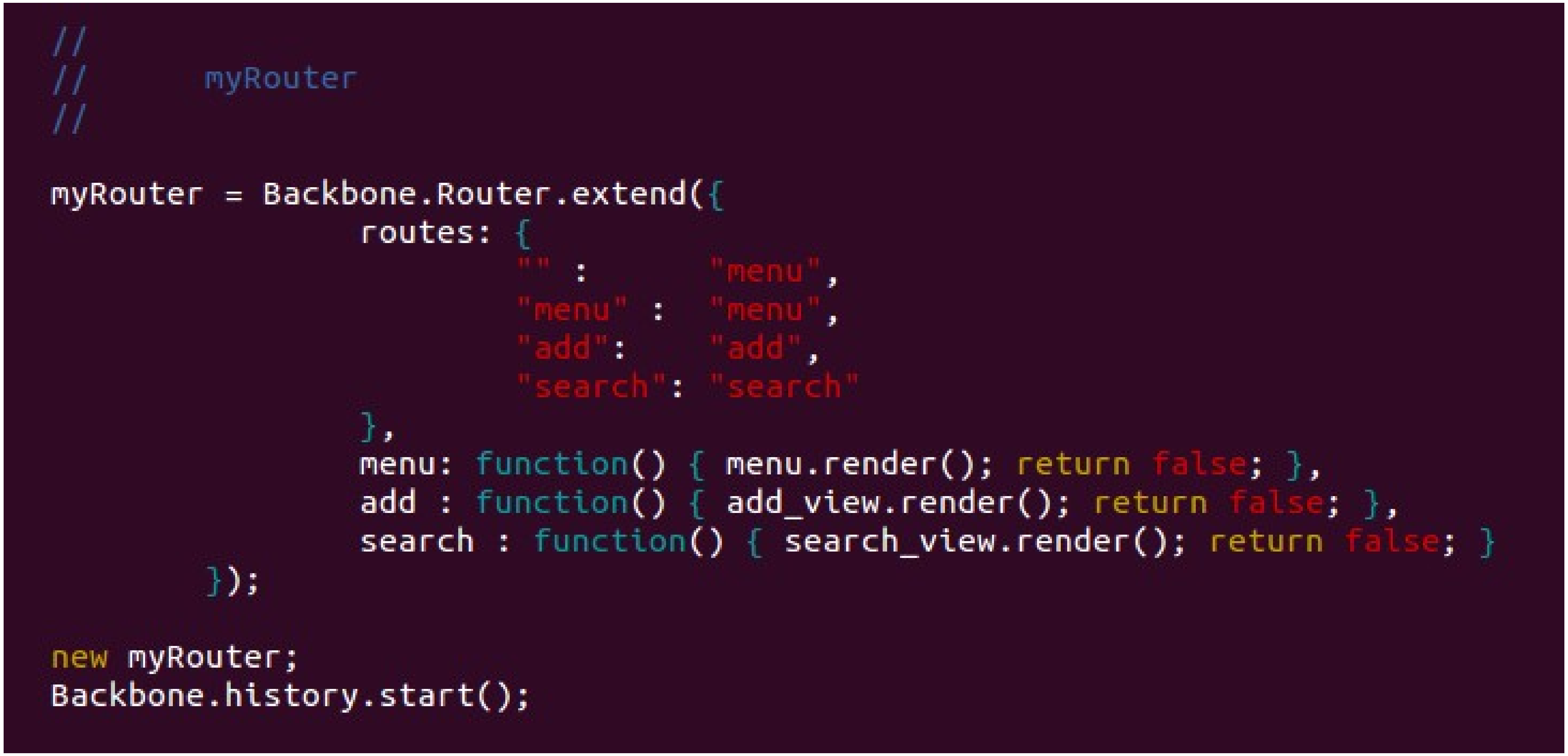


```
<script type="text/template" id="add_template">
<br><br><br>
<center>
<form name="myform" action="book_add.php" method="post">
<input type="hidden" name="flag" value="1">
<table style="border-style: dotted; border-width: 1; background-color: #eee">
<tr>
<th colspan="2">
  <div style="background-color: #ccc;">
     <center>Add New Book</center>
  </div>
  <br>
</th>
</tr>
<tr>
<td align="right">Location</td>
<td><input type="text" name="location" size="20" maxlength="20" value=""></td>
</tr>
<tr>
<td align="right">Title</td>
<td><input type="text" name="title" size="40" maxlength="40" value=""></td>
</tr>
<tr>
<td align="right">Author</td>
<td><input type="text" name="author" size="30" maxlength="30" value=""></td>
</tr>
<tr>
<td align="right"><br><input type="button" value="Menu" id="menu-button"></td>
<td align="center"><br><input type="submit" value="Add" id="add-button"></td>
</tr>
<tr><td colspan="2"><br><center><img src="booklist2.jpg"></center></td></tr>
</table>
</form>
</center>
</script>
```

Backbone.Router

- Route client side pages and connect them to actions and events
- Interfaces with hashchange events
- Reacts to history navigation
- Need to be aware of possible server interactions

Backbone.Router

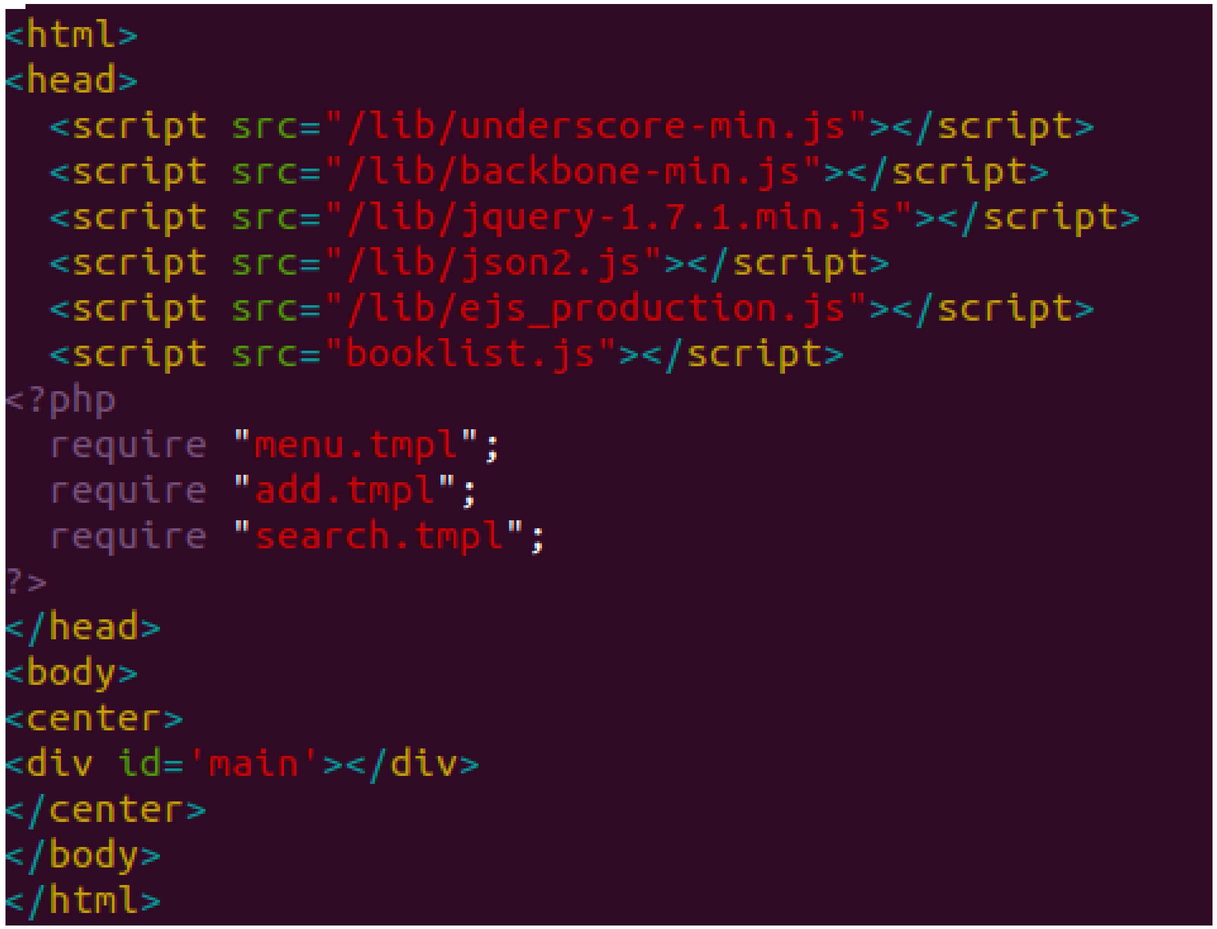


```
//
//      myRouter
//

myRouter = Backbone.Router.extend({
                routes: {
                        "" :      "menu",
                        "menu" :  "menu",
                        "add":    "add",
                        "search": "search"
                },
                menu: function() { menu.render(); return false; },
                add : function() { add_view.render(); return false; },
                search : function() { search_view.render(); return false; }
        });

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

Booklist App

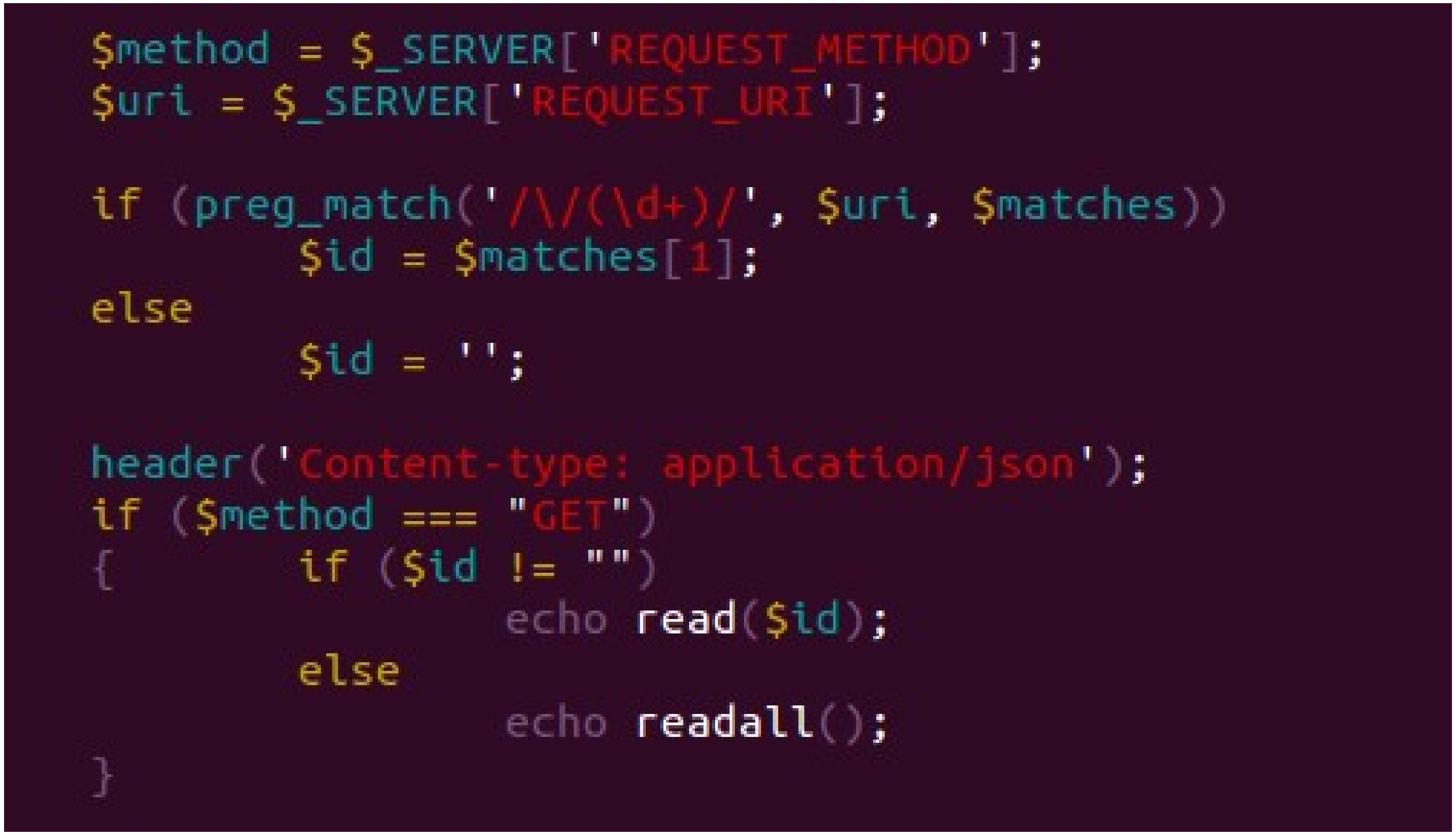
- server
  - index.php
  - db.php
- client-side
  - booklist.js
  - templates

index.php



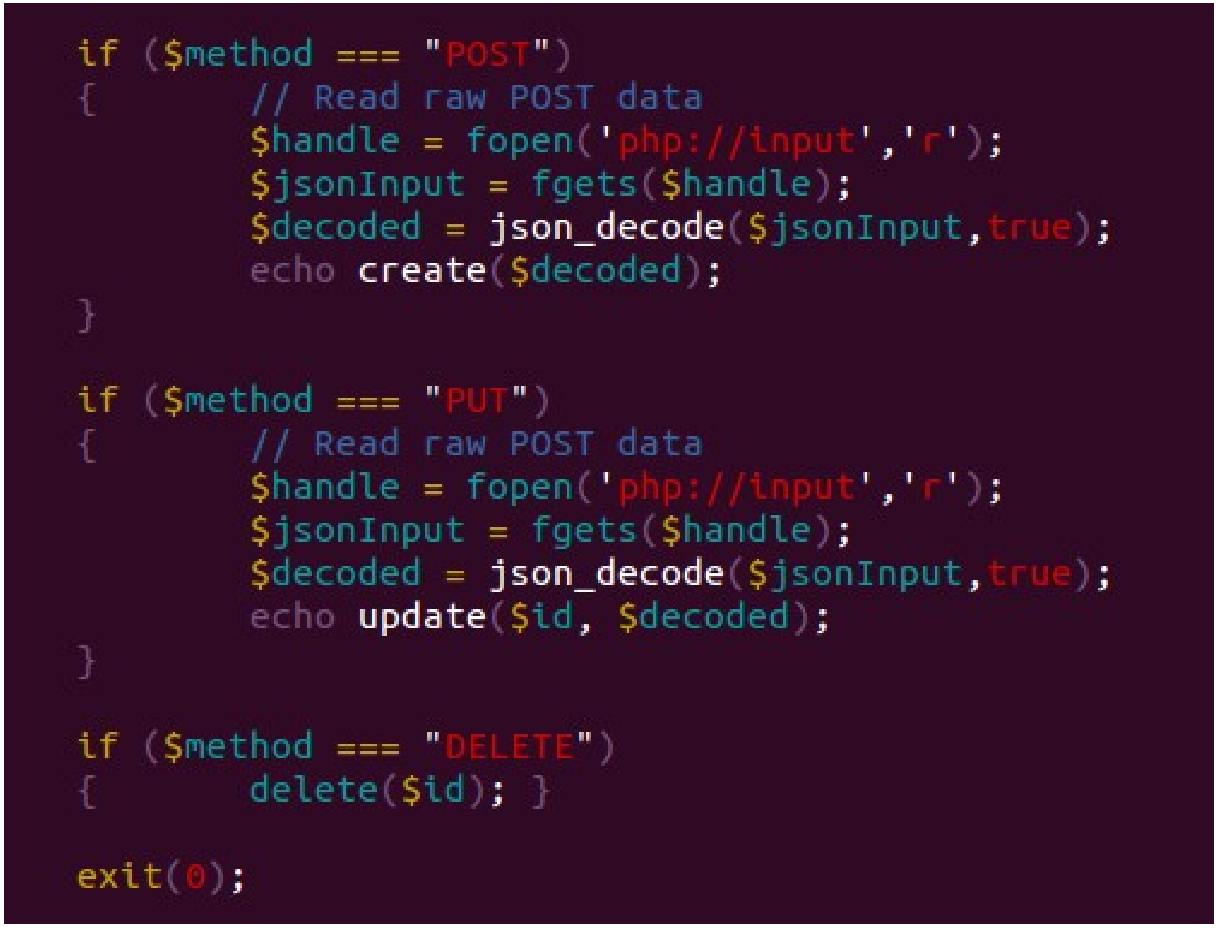
```
<html>
<head>
  <script src="/lib/underscore-min.js"></script>
  <script src="/lib/backbone-min.js"></script>
  <script src="/lib/jquery-1.7.1.min.js"></script>
  <script src="/lib/json2.js"></script>
  <script src="/lib/ejs_production.js"></script>
  <script src="booklist.js"></script>
<?php
  require "menu.tmpl";
  require "add.tmpl";
  require "search.tmpl";
?>
</head>
<body>
<center>
<div id='main'></div>
</center>
</body>
</html>
```

db.php



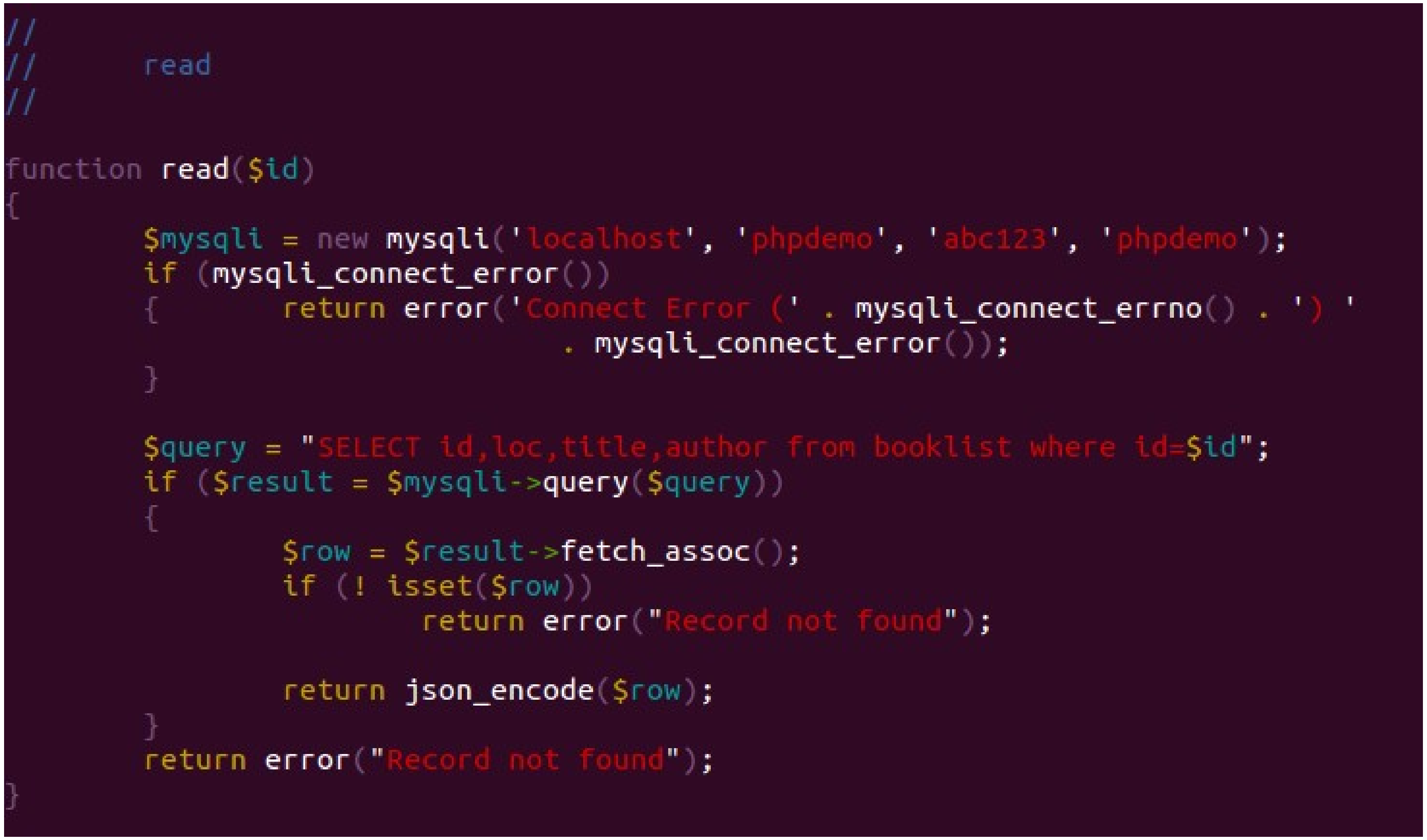
```php
$method = $_SERVER['REQUEST_METHOD'];
$uri = $_SERVER['REQUEST_URI'];

if (preg_match('/\/(\d+)/', $uri, $matches))
        $id = $matches[1];
else
        $id = '';

header('Content-type: application/json');
if ($method === "GET")
{       if ($id != "")
                echo read($id);
        else
                echo readall();
}
```

db.php - continued



```php
if ($method === "POST")
{       // Read raw POST data
        $handle = fopen('php://input','r');
        $jsonInput = fgets($handle);
        $decoded = json_decode($jsonInput,true);
        echo create($decoded);
}

if ($method === "PUT")
{       // Read raw POST data
        $handle = fopen('php://input','r');
        $jsonInput = fgets($handle);
        $decoded = json_decode($jsonInput,true);
        echo update($id, $decoded);
}

if ($method === "DELETE")
{       delete($id); }

exit(0);
```

db.php - continued



```
//
//      read
//

function read($id)
{
        $mysqli = new mysqli('localhost', 'phpdemo', 'abc123', 'phpdemo');
        if (mysqli_connect_error())
        {       return error('Connect Error (' . mysqli_connect_errno() . ') '
                                . mysqli_connect_error());
        }

        $query = "SELECT id,loc,title,author from booklist where id=$id";
        if ($result = $mysqli->query($query))
        {
                $row = $result->fetch_assoc();
                if (! isset($row))
                        return error("Record not found");

                return json_encode($row);
        }
        return error("Record not found");
}
```

References

- Backbone.js Applications (Early Access), Addy Osmani, O'Reilly (2012)
- Backbone.js: Basics & Beyond, Sarah Mei, Fluent2012 video, O'Reilly (2012)
- Javascript Master Class, Douglas Crockford, O'Reilly Media (2009)
- Javascript The Good Parts, Douglas Crockford, O'Reilly (2008)
- Javascript Web Applications, Alex MacCaw, O'Reilly (2011)
- Recipes With Backbone, Nick Gauthier and Chris Strom, eee_c (2012)
- Secrets of Javascript Ninjas, John Resig and Bear Bibeault, Manning (2012)

The End

Ken Harris
harris@itech-mke.com
@harris901