Backbone - Day 1

Macy's Learning Spike

About me - Alain Chautard (or just Al)

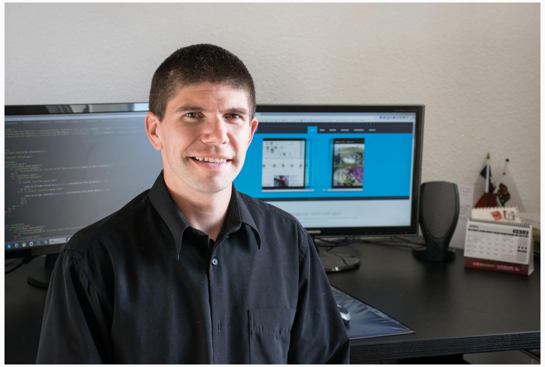
Google Developer Expert in Web technologies / Angular

Java developer since 2006

Angular JS addict since 2011

Web consultant (60%) / trainer (40% of the time)

Organizer of the Sacramento Angular Meetup group



Quick Poll

- How many of you are Java developers? C#, .Net?
- How many of you are developers? Full-stack? Back-end?
- Any experience with Javascript? TypeScript? Angular?
- jQuery?
- Any other Javascript framework?

How we're going to work

- Your questions are welcome, anytime!
- Being a web developer requires constant learning
- My goal is to give you the tools to work efficiently with web technologies - We’re going to practice a lot!
- As a result, we will be going through online docs very often

Useful links

- Repository for all labs code + solutions: https://github.com/alcfeoh/di-backbone-js
- Link to these slides: https://goo.gl/rGJiWv

Outline for today

Introduction to Backbone

Backbone Models

Backbone Views

Backbone Collections

Backbone with handlebars

Outline for today

Introduction to Backbone

Backbone Models

Backbone Views

Backbone Collections

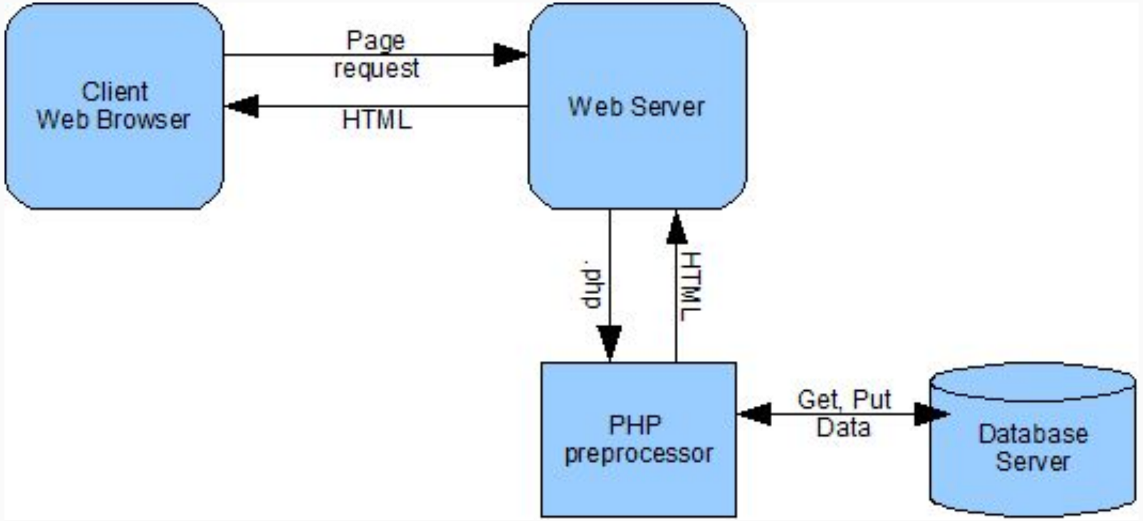
Backbone with handlebars

Introduction to Backbone

PHP / JSP / ASP

In the past, all of the front-end code (HTML, JS, CSS) was generated from the back-end

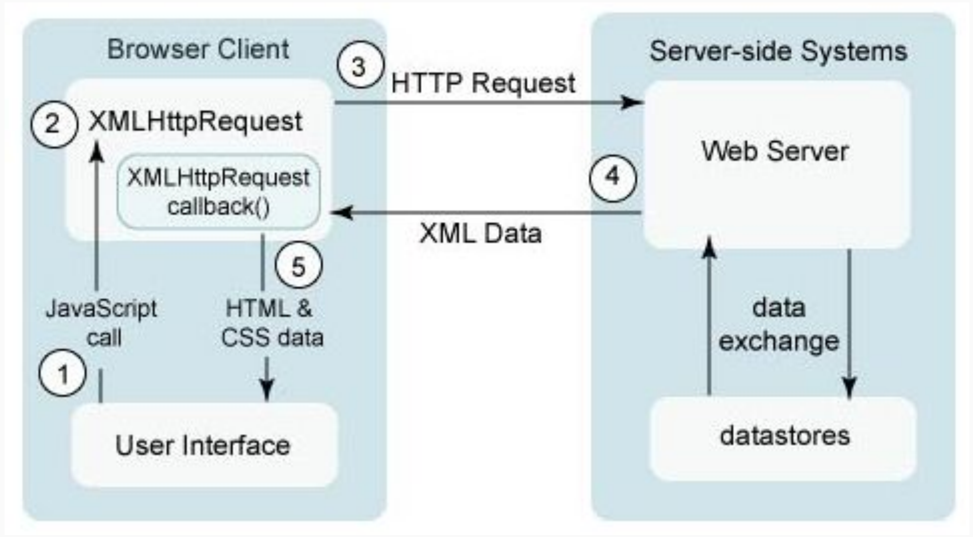
User interactions with the webpage often required a full-page refresh



Ajax

Then came into play AJAX and jQuery

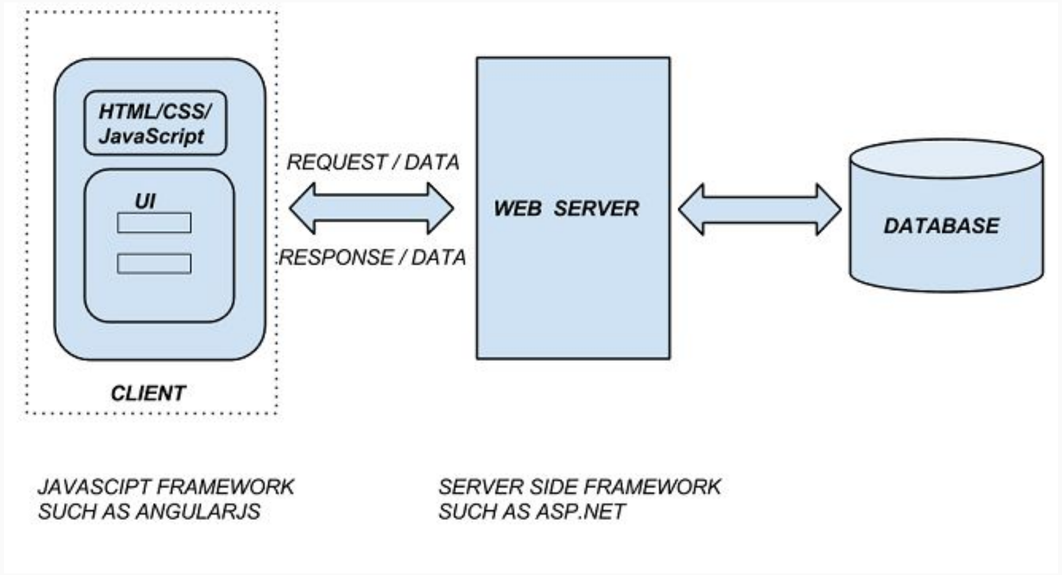
The main idea was to load content asynchronously in the background to refresh portions of a webpage



Backbone

With Backbone, the front-end code is now independent from the back-end: It’s a **Single Page Application**

The web server becomes a web-service that outputs JSON data, not dynamic HTML or CSS



What is Backbone?

- Backbone.JS is a framework that brings the MVC (**M**odel - **V**iew - **C**ontroller) pattern to **JavaScript**
- It is very lightweight (7.6kb) packed and zipped and requires few dependencies (only hard dependency is underscore.js)
- Official website: http://backbonejs.org

Why Backbone?

- Backbone.JS is unopiniated: There are different ways to solve any problem
- Its learning curve is very short (we can do it in two days!)
- It does not require a lot of set-up (unlike Angular for instance)

BACKBONE.JS

What we just learnt

Backbone is a Javascript framework that brings structure to Single Page Applications

Backbone is lightweight and unopiniated

Its official website is http://backbonejs.org

Its learning curve is very short

Outline for today

Introduction to Backbone

Backbone Models

Backbone Views

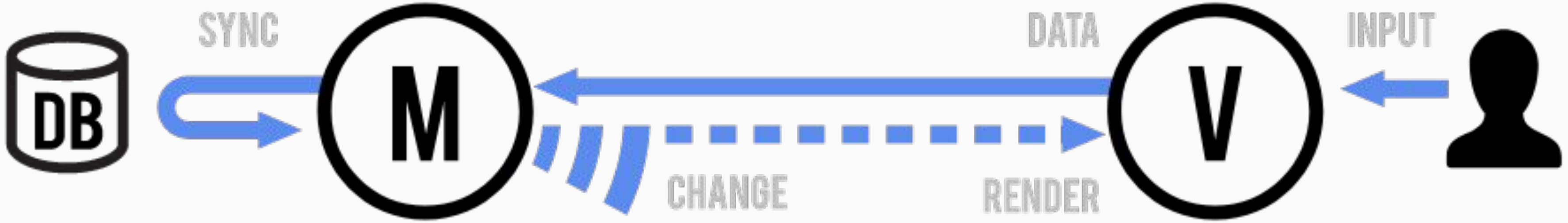
Backbone Collections

Backbone with handlebars

Backbone Models

What are Backbone Models?

- **Models** represent the data of your application
- In Backbone, a model is a set of keys and values, an internal table of data attributes
- Models handle syncing data with a persistence layer (usually a REST API)



Simple Model

We create the definition of our model using **`Backbone.Model.extend`**

Models can have several attributes or functions detailed here: http://backbonejs.org/#Model

```
// Here we define the structure
// of our data model
var Todo = Backbone.Model.extend({
    // Default values when a new
    // instance is created
   defaults: {
       title: '',
       completed: false
   }
});
```

Instance of a Model

Once our Model definition is done, we can create an instance of it

The constructor function can be used to pass the value of the data model

Simple getters / setters can then be used to read / write the model

```
// Create object with attributes
var todo = new Todo({title: 'Learn
Backbone', completed: false});

todo.get('title'); // "Learn Backbone"
todo.get('completed'); // false
todo.get('created_at'); // undefined

// Setting a value
todo.set('created_at', Date());
todo.get('created_at'); // "Wed Sep 12
2012 12:51:17 GMT-0400 (EDT)"
```

Controller Methods

We can add our own methods to any Model definition

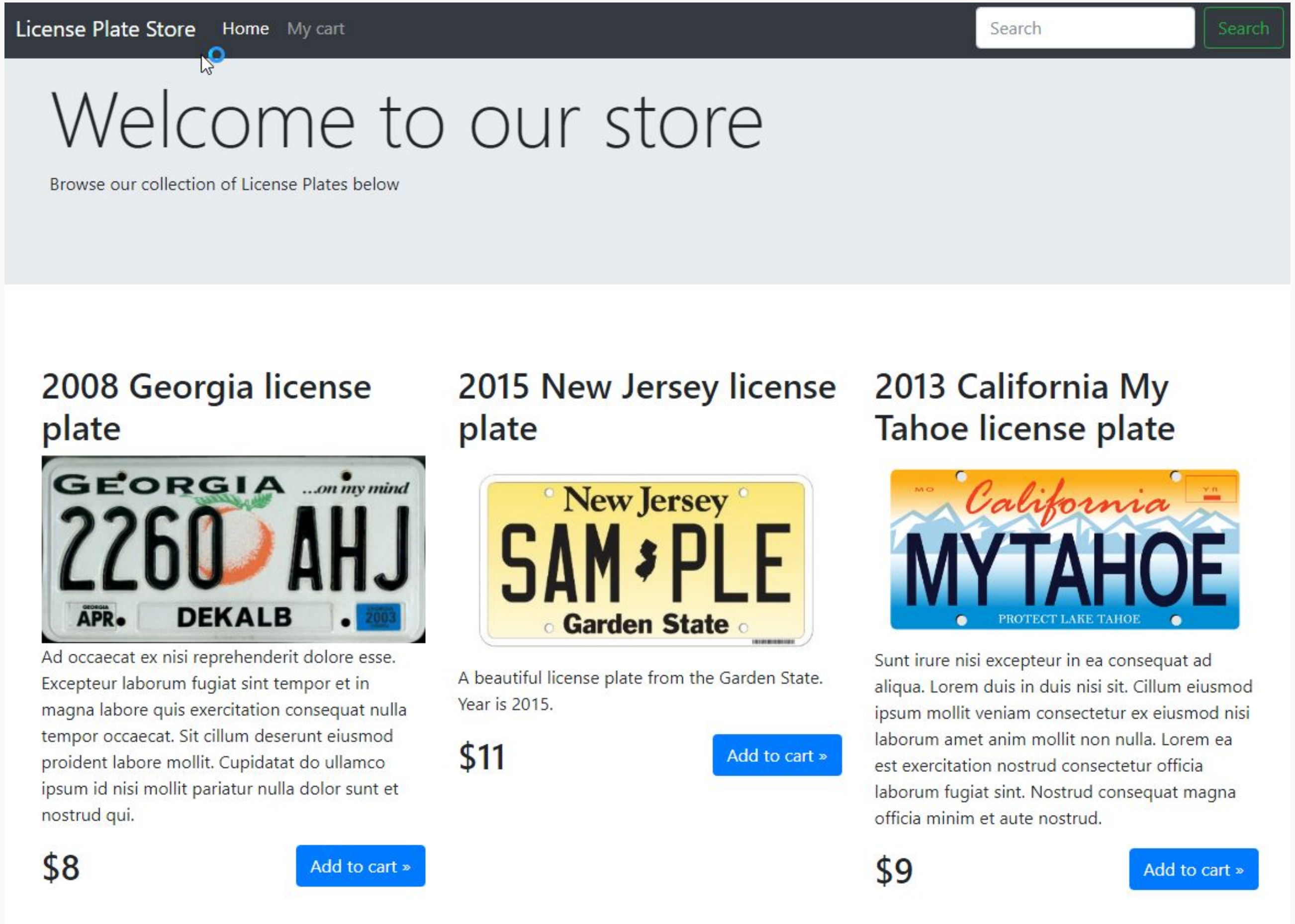
These methods play the role of the Controller in the MVC pattern

```
var Todo = Backbone.Model.extend({
    defaults: {
        title: '',
        completed: false
    },
    // Our own model method
    completeTodo: function(){
        this.set('completed', true);
    }
});
```

Lab 1 - Hello Backbone

- In this lab, we’re going to write a simple Backbone Model definition that just says **“Hello Backbone World!”**
- **Your mission:** Start from the file **`1-hello-backbone.html`**. Create a model definition in that file for a **HelloWorld** model.
- That Model should have a **`helloWorld()`** method that alerts **“Hello message!”** and the default message should be **“Backbone World”**
- A custom message can be passed to the model constructor.
- Use the browser console to get / set values of your model and call its **`helloWorld()`** method

We’re going to build an online License Plate Store!



License Plate Store Home My cart Search Search

Welcome to our store

Browse our collection of License Plates below

2008 Georgia license plate

Ad occaecat ex nisi reprehenderit dolore esse. Excepteur laborum fugiat sint tempor et in magna labore quis exercitation consequat nulla tempor occaecat. Sit cillum deserunt eiusmod proident labore mollit. Cupidatat do ullamco ipsum id nisi mollit pariatur nulla dolor sunt et nostrud qui.

$8 Add to cart »

2015 New Jersey license plate

A beautiful license plate from the Garden State. Year is 2015.

$11 Add to cart »

2013 California My Tahoe license plate

Sunt irure nisi excepteur in ea consequat ad aliqua. Lorem duis in duis nisi sit. Cillum eiusmod ipsum mollit veniam consectetur ex eiusmod nisi laborum amet anim mollit non nulla. Lorem ea est exercitation nostrud consectetur officia laborum fugiat sint. Nostrud consequat magna officia minim et aute nostrud.

$9 Add to cart »

Lab 2 - Creating a Model for a license plate

- In this lab, we're going to write a simple Backbone Model definition for a license plate of our store.
- **Your mission:** Start from the file **`2-license-plate-model.html`**. Use the given Javascript object as the data model for an instance of a **`LicensePlate`**.
- Once your model is created, add a line of Javascript to do the following:

```
alert("License plate created: " + plate.get('title'));
```

What we just learnt

Backbone Models are a way to represent the data of our application and interact with it

Constructor functions allow to pass actual data to our model

A **`defaults`** object can also be created with default values

Getters / setters allow us to read / write to our data model

Outline for today

Introduction to Backbone

Backbone Models

Backbone Views

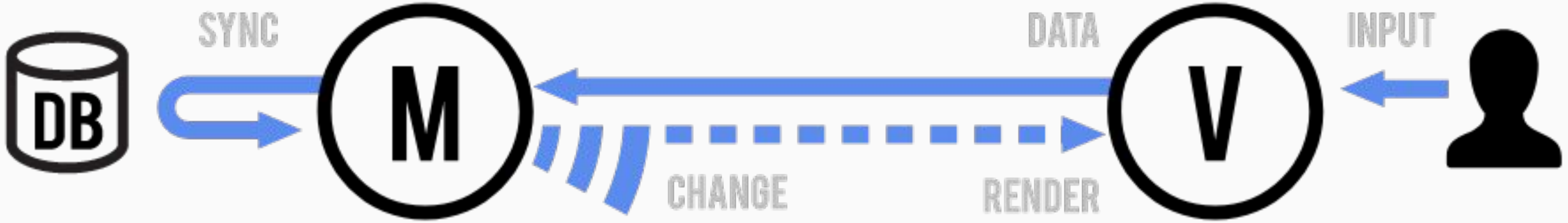
Backbone Collections

Backbone with handlebars

Backbone Views

What are Backbone Views?

- **Views** are atomic chunks of user interface
- They render data from a specific model
- Views listen to the model "change" events, and react or re-render themselves appropriately



Basic View

A view needs to be hooked to an HTML element (**el**) so that Backbone knows where to render it.

We use **Backbone.View.extend** to create the view definition

initialize is the first function called when a view gets instantiated

```
var DocumentRow = Backbone.View.extend({
    // HTML tag this view is going to generate
    tagName: "li",
    // CSS class applied to the view tag
     className: "document-row",
     // Where the view will be rendered
     el: "#container",

    // Init function for the view
    initialize: function() {
        this.listenTo(this.model, "change",
                                    this.render);
    },

    render: function() {
        // Do some DOM manipulation
        // to render things here
    }
});
```

$el

`$el` is a jQuery object that references the element where the view should be rendered (in our case, **#container**)

```
var DocumentRow = Backbone.View.extend({
    // HTML tag this view is going to generate
    tagName: "li",

     // Where the view will be rendered
     el: "#container",

    // ...

    render: function() {
        // A jQuery object to render things
        this.$el.html("View rendering text");
    }
});
```

Instantiating a View

Once a view is defined, we can create an instance of it and pass a **model** to its constructor function

Other view attributes, such as **`el`**, can be passed to the constructor function as well

```
var row = new DocumentRow({
    model: doc,
    // Now the view is dynamic based
    // on the model ID
    el: "document-row-" + doc.id
});
```

Lab 3 - Hello Backbone View

- In this lab, we’re going to write a simple Backbone View that just renders **“Hello Backbone World!”** using our **HelloWorld** model.
- **Your mission:** Start from the file **`3-hello-backbone-view.html`**. Create a view definition in that file that uses the **HelloWorld** model to render the message.
- The view should be rendered on the HTML **body** of the document
- Don’t forget to add **jQuery** as a dependency in your HTML scripts section
- Instantiate the view and make sure it renders as expected

Templates

Most views are complex and require the use of a HTML template

Such templates can be defined in our HTML using a **script** tag

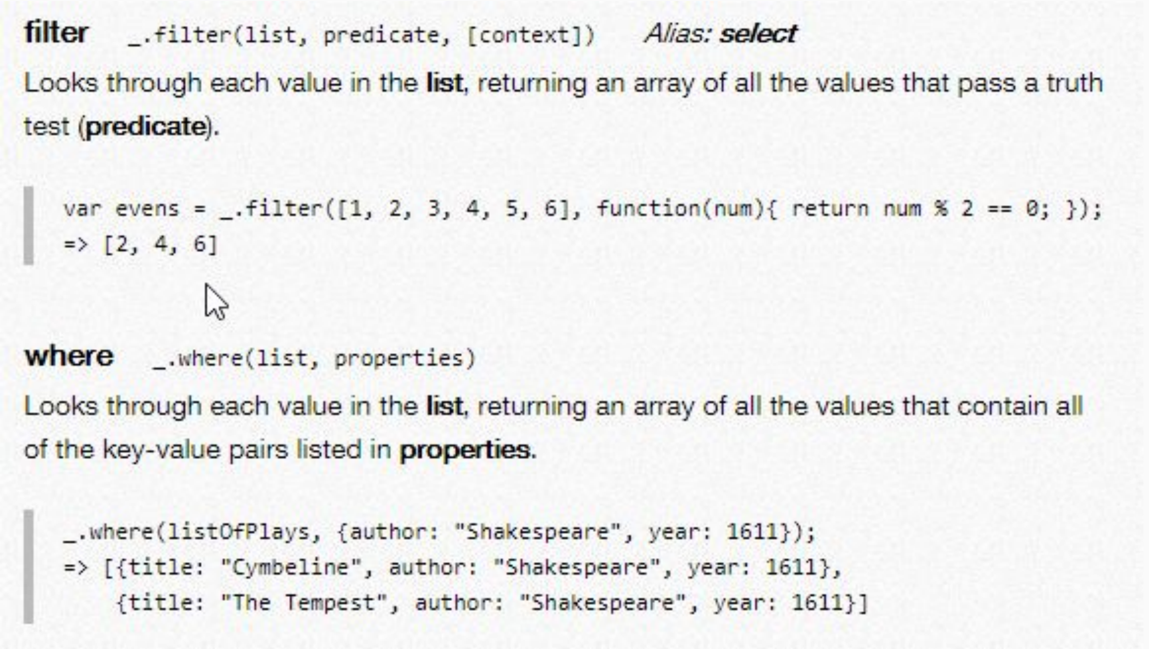
We're using the **underscore** library to load the template and render it

```
// HTML template
<script type="text/template" id="item">
    <div class="view">
        <input class="toggle" type="checkbox">
        <label><%- title %></label>
    </div>
</script>

var TodoView = Backbone.View.extend({
    tagName: 'li',
    template: _.template($('#item').html()),

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

Underscore.js

Underscore provides over 100 functions: map, filter, invoke, function binding, javascript templating, creating quick indexes, deep equality testing, and so on.

Official website: http://underscorejs.org



filter `_.filter(list, predicate, [context])` *Alias: **select***

Looks through each value in the **list**, returning an array of all the values that pass a truth test (**predicate**).

```
var evens = _.filter([1, 2, 3, 4, 5, 6], function(num){ return num % 2 == 0; });
=> [2, 4, 6]
```

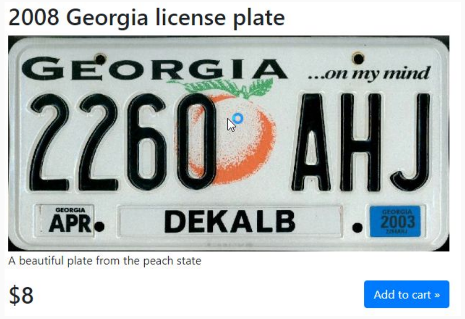
where `_.where(list, properties)`

Looks through each value in the **list**, returning an array of all the values that contain all of the key-value pairs listed in **properties**.

```
_.where(listOfPlays, {author: "Shakespeare", year: 1611});
=> [{title: "Cymbeline", author: "Shakespeare", year: 1611},
    {title: "The Tempest", author: "Shakespeare", year: 1611}]
```

Lab 4 - License Plate View

- In this lab, we're going to write a simple Backbone View that renders one license plate for our store.
- **Your mission:** Start from the file **`4-plate-view-and-model.html`**. Create a view definition in that file that uses the **LicensePlate** model to render it.
- The view template is almost ready for you to use: **`#plate-template`**
- Complete the template, hook it up to your view and render it!
- It should look like this once ready:



2008 Georgia license plate

A beautiful plate from the peach state

$8

What we just learnt

Views are Backbone’s way of rendering data models

Views listen to model updates to refresh their HTML

Views can use templates to render their HTML

Underscore.js is a library of utility functions used by Backbone.js

Outline for today

Introduction to Backbone

Backbone Models

Backbone Views

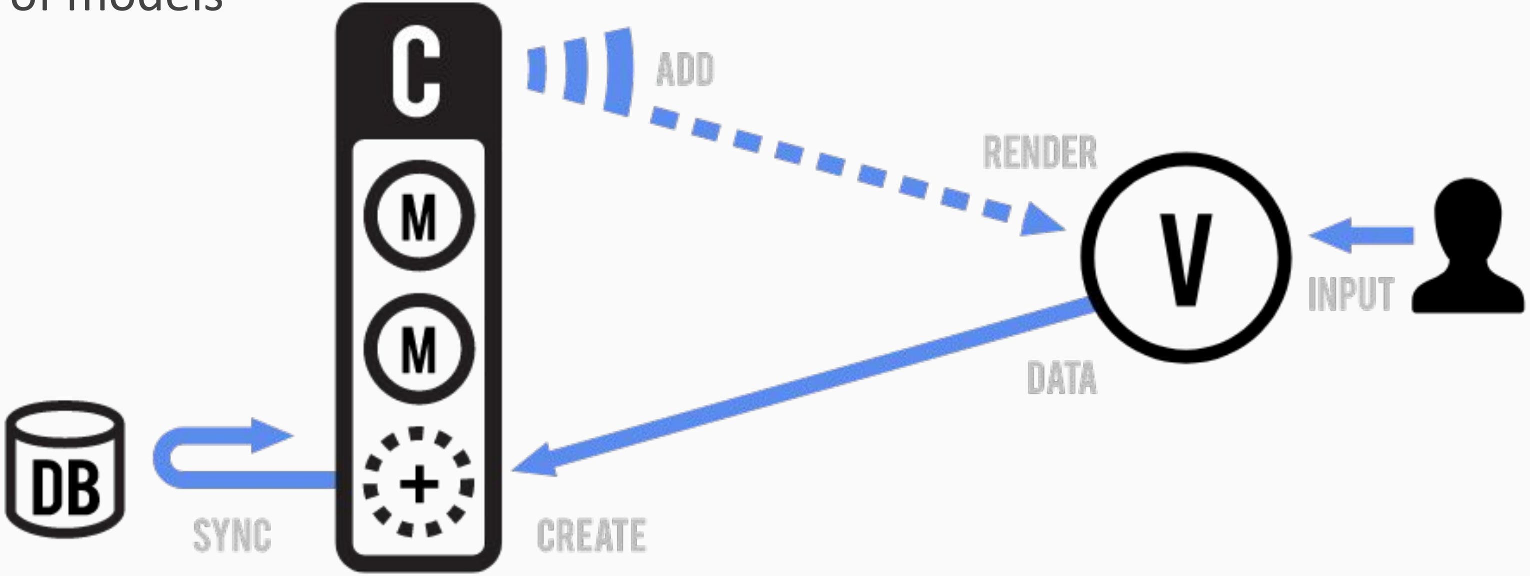
Backbone Collections

Backbone with handlebars

Backbone collections

What are Backbone Collections?

- **Collections** help deal with a group of related models
- Provide helper functions for performing aggregations or computations against a list of models



Collection

Collections are based on a specific model

They can sync with a REST backend when a URL is passed to them

```
var TodoList =
 Backbone.Collection.extend({
   model: Todo,
   // Collections can sync with REST WS
   url: "/todos"
});

// Instance of the Collection
var todoList = new TodoList();
```

Collection

Collections are based on a specific model

They can sync with a REST backend when a URL is passed to them

```
var TodoList =
 Backbone.Collection.extend({
   model: Todo,
   // Collections can sync with REST WS
   url: "/todos"
});

// Instance of the Collection
var todoList = new TodoList();
```

App view

The main application view uses a collection to iterate through models and render them

This is a common pattern in Backbone applications

```
var AppView = Backbone.View.extend({
  el: '#todoapp',
  //...
  addOne: function(todo){
     // Append every todo to the list
     var view = new TodoView({model: todo});
     this.$el.append(view.render().el);
  },
  render: function(){
     // Iterate through the collection
     todoList.each(this.addOne, this);
  }
});

var appView = AppView();
```

Lab 5 - A collection of License Plates

- In this lab, we’re going to write a simple Backbone app that uses a collection to render a list of license plates for our store.
- **Your mission:** Start from the directory **`5-collection-plates`**. Our code is now split in two files: **`index.html`** and **`app.js.`**
- Create a collection object and an application view to render it, using the provided **`plates`** array.
- The collection of plates should be renderer in the **`#container`** div
- **Hint:** The code you have to write is very similar to the one we saw in the previous slides

What we just learnt

Collections are Backbone’s way of helping with the rendering of several data models of the same type

Collections are lists of data models

A view’s render function will typically use a collection to iterate through model items and render them

Views can sync with a REST backend server

Outline for today

Introduction to Backbone

Backbone Models

Backbone Views

Backbone Collections

Backbone with handlebars

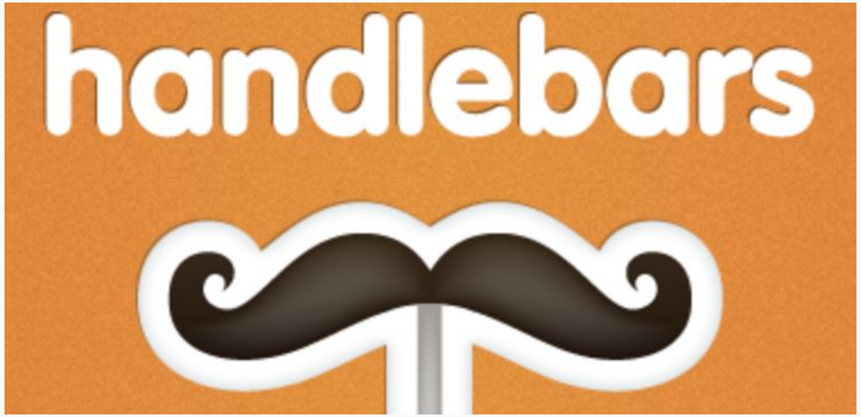
Backbone with Handlebars

Handlebars.js

Handlebars is a simple templating library that you can use instead of Underscore.js

Backbone is open to any templating engine

Official website: http://handlebarsjs.com/



```
<div class="entry">
    {{#if author}}
    <h1>{{author.firstName}}
        {{author.lastName}}
    </h1>
    {{/if}}
</div>
```

Handlebars with Backbone

All we have to do to use Handlebars is use its templating feature in the **render** function

```
var SearchView = Backbone.View.extend({
   initialize: function(){
       this.render();
   },

   render: function(){

     // Compile the template with Handlebars
     var src = $('#template').html();
     var template = Handlebars.compile(src);

     // Pass the data model to get the HTML
     var html = template(this.model.toJSON());

     // Load the HTML into the Backbone "$el"
     this.$el.html(html);
   }
});
```

Lab 6 - Using Handlebars templates

- In this lab, we're going to update our store app to use Handlebar templates.
- **Your mission:** Start from the directory **`6-handlebar-template`**. First update **`index.html`** to turn our template into a Handlebars template
- Then update **`app.js`** to make the **`LicensePlateView`** use Handlebars
- **Hint:** The code you have to write is very similar to the one we saw in the previous slides - also feel free to browse http://handlebarsjs.com/ for more information on how to use Handlebars

What we just learnt

Handlebars can be used In Backbone to handle template rendering

Backbone allows any kind of template rendering engine as long as we use it in the **render** function

Thanks for your attention

I need your feedback before you leave:
http://bit.ly/lsbackbone11-30-17

BACKBONE.JS

See you tomorrow for day 2