Adventure Time

with

JavaScript & Single Page Apps

Carl Bergenhem - Web Unleashed 2013 - **@carlbergenhem**

Introductions

Carl Bergenhem

Manager, Solutions Consultants

Telerik

@carlbergenhem

Quick Notes

Covering a lot of ground today

High level overview

More code in tomorrow's workshop

9 AM - 12:30 PM @ Room 325 BC

What is a SPA?

Single Page Application

A web app that fits in a single page

Entire app is loaded at once

Built with a MV* Framework

Server provides the initial app

Section our HTML page in views

Get data via RESTful requests

Bind the data to our UI

Business logic in our MV* framework

MV* framework handles URL requests

Why SPA?

Create desktop-like apps in the web

Increase responsiveness

Reduce loadtime of pages

Less time spent on the server

Control user interactions on the client

Improved end-user experience

History

In the Beginning

Reign of the post back

Everything done on the server

Full page refreshes

We got wiser

Ajax & XHR Requests

Serve some content via server

Do asynchronous calls to server

Partial page refreshes

Focus on responsiveness

Introduce the SPA concept

Move all items over to the client

What makes up a SPA?

MV* Framework

Basic pieces of a MV*

- Model
- View
- *

A full SPA framework also has

Routing

Model

In charge of holding data

Communication with server

Work with server end-points

Define what data should be used

Controller

Control applications'

- State
- Logic
- Behavior

Custom business logic

Communicate between the View and Model

Provide validation

Bind data to our Views (UI)

View

Responsible for the UI of our app

Modifies and interacts with our DOM

Main point of interaction from users

Bound to the controller for data and events

Often works with a template engine

Router

Responds to URL changes via JavaScript

Changes between various views

Browser initiated routes

- Type in a URL
- Hyperlink

Client initiated routes

Change from one page or a view to another

Utilizes hash fragments

```
http://myurl.com/#/products/
```

JS rights to modify page URL

Anything after '#' is fair game

Just like the anchor tag

Examples

Wild SPA Apps

- Gmail
- Facebook
- Twitter
- Trello
- Asana

Popular MV* Frameworks

- Backbone
- Angular
- Kendo UI
- Ember
- Durandal
- Meteor

Today's Focus

Backbone

Angular

Kendo UI

Backbone

Light-weight library

The basic foundation

Not a ton of extra frills

Heavy lifting is up to you

Huge community for tooling

Offers a set of helpful classes

Each one can
- and will be -
extended

Backbone main classes

- Model
- Collection
- View
- Router

Model

Deals with all of our data

Bound to a view

Syncs back to our server

Backbone.Model

```
var myModel = new Backbone.Model({});
```

Extend Method

```
var Character = Backbone.Model.extend({
        defaults: {
            name: 'Blanks',
            species: 'Human',
            description: 'No description yet.'
        },
        initialize: function() {
            console.log('New character created');
        }
});

var jake = new Character({
        name: 'Jake the Dog',
        species: 'Dog',
        description: 'A magical dog. Best friend, and brother, with F
});
```

Use get() or set() to change data

```
var finn = new Character({ name: 'Finn the Human' });
finn.set({ species: 'Human' });
var name = finn.get('name'); //Finn the Human
```

Collections

Group related models together

Methods for adding/removing models

Simple array-like object

Backbone.Collection

```
var Character = Backbone.Model.extend({
        defaults: {
            name: 'Blanks',
            species: 'Human',
            description: 'No description yet.'
        },
        initialize: function() {
            console.log('New character created');
        }
});

var Cast = Backbone.Collection.extend({
        model: Character
});

var jake = new Character({
        name: 'Jake the Dog',
        species: 'Dog',
        description: 'A magical dog. Best friend, and brother, with F
});

var finn = new Character({
```

View

The link between models and the UI

Renders the models and their data

Views are bound to models or collections

Receives events from the Model and HTML document

The 'el' property

- Every view has an 'el' property
- If not defined, a blank div is created
- References the view's DOM object
- Pass in a HTML element to modify it
- Can be a jQuery object via $el

Backbone.View

```
var FinnView = Backbone.View.extend({
        tagName: 'div',
        id: 'finnDiv',
        className: 'finnClass',
        attributes: {
                'data-custom': 12345
        }
});

var quickView = new FinnView();

$('#placeholder').prepend(quickView.el);
```

Resulting HTML

```
<div data-custom="12345" id="sampleDiv" class="sampleClass"></div>
```

Default templates by Underscore

jsBin.com/lWewESE/8/

Router

Use of hash fragments

```
http://myurl.com/#/characters/
```

Parses URL and finds appropriate function

Uses HTML5 History API if possible

Backbone.Router

```
var SampleRouter = Backbone.Router.extend({
        routes: {
                'character/:name': 'character' //route pattern : hand
        },
        character: function(name) { //handler
            //...
        }
});
```

Backbone Summary

Barebones tools to build our app

Emphasis on data and Models

Not a ton of tools for the View

.extend({}) everything

Backbone main classes

- Model
- Collection
- View
- Router

Model deals with all of our data

A collection has multiple Models

View needs a Model
Works with Model and DOM

Router works with URL to serve correct View

Angular

Created by Google

Believes HTML can be more powerful

Extends HTML vocabulary

Tries to follow a pure HTML path

Angular components

- Expressions
- Controllers
- Directives
- Routing

Expressions

Allows us to bring data to the view

Uses a simple syntax

{{ model.FieldName }}

One-way binding

```
<div ng-controller="SampleController">
        <span>{{ character.name }}</span>
</div>
```

Two-way binding

```
<select ng-model="selectedCharacter">
        <option value="0">Finn</option>
        <option value="1">Jake</option>
        <option value="2">BMO</option>
</select>
```

Controllers

Central component of Angular apps

Contains data, logic, and states

Creates a scope for the HTML element

$scope variable

Sample Controller

```
var sampleAppControllers = angular.module('sampleAppControllers', []);

sampleAppControllers.controller('ShowController', ['$scope',
    function ShowController($scope) {
        $scope.sampleVariable = 1;
        $scope.sampleArray = [{
                fieldOne: 0,
                fieldTwo: "Test"
            },
            {
                fieldOne: 1,
                fieldTwo: "Sample"
            }]
        $scope.sampleFunction: function() {
            //Function logic
        }
    }]);
```

$scope only available within the HTML element it is defined in

Directives

HTML-like attributes

Declared in view code

Helps with setting up

- Angular App
- Controllers and binding
- Attributes
- Conditional display
- Events

Sample Directives

- ng-app="appName"
- ng-controller="SampleCtrl"
- ng-model="ScopeVar"
- ng-src="{{scopeUrlVar}}"
- ng-click="eventHandler()"

Routing

Works with URL like you'd think

Single main view

Placeholder element for content

Load other views via partials

Just HTML pages

Simple structure for routing

```
var sampleApp = angular.module('sampleApp', [
    'ngRoute',
    'myControllers'
]);;

sampleApp.config(['$routeProvider',
    function($routeProvider) {
        $routeProvider.when('/test', { templateUrl: 'partials/test.htr
        $routeProvider.when('/sample:id', { templateUrl: 'partials/sar
        $routeProvider.otherwise({ redirectTo: '/' });
    }]);
```

Angular Summary

Extension of HTML

Empowering your regular HTML code

Controllers rule all

Data can be bound one- or two-way

DOM manipulation via controller

Drop that jQuery ;)

Directives are extremely helpful

Wire upp events and such

Routers use partials of HTML

Kendo UI

Part of Kendo UI Web

Combine SPA and MVVM Frameworks

Additional helpful items

- Template engine
- Effects and animations
- UI widgets
- Charts and gauges

Tooling across all parts of your SPA

Kendo UI key pieces

- Observable
- View
- Layout
- Router

Observable

Part of the Kendo UI MVVM framework

All fields are observable

All changes will propogate

kendo.observable

```
var finnModel = kendo.observable({
        name: 'Finn the Human',
        species: 'Human',
        description: 'The last human in the Land of Ooo. Hangs out wi
});
```

Use get() or set() with fields

```
var finnModel = kendo.observable({
        name: 'Finn the Human',
        species: 'Dog',
        description: 'The last human in the Land of Ooo. Hangs out wi
});

finnModel.set('species', 'Human');

var species = finnModel.get('species'); //Human
```

View

Can be defined with simple HTML string

Works with Kendo UI templates

Tie in with our observables to create data bound UI

kendo.View()

```
var index = new kendo.View('<span>Adventure Time!</span>');

index.render('#placeholder');
```

With Kendo UI Templates and Observables

jsbin.com/imAnimo/5/

Layout

Provides a layout for views to adhere to

MasterPages for you .NET folks

Easily display views in content section

Can also be nested

kendo.Layout

```
var view = new Kendo.View('<span>Finn the Human</span>');

var layout = new kendo.Layout('<header>Header</header><div id="content"></div><footer>Footer<

layout.render($('#app'));

layout.showIn('#content', view);
```

Router

Responsible for application state

Navigate between views

Use of hash fragments

```
http://myurl.com/#/characters/
```

Can programmatically navigate

Use with Layout to show new content

kendo.Router()

```
var router = new kendo.Router();
router.route('/', function () {
        layout.showIn('#content', homeView);
});

router.route('/character', function () {
        layout.showIn('#content', characterView);
});

$(function() {
        router.start(); //very important

        router.navigate('/character');
});
```

Kendo UI Summary

Tools across the board

Classes for dealing with data

Helpful UI and View classes

Large set of UI widgets

Kendo UI SPA key items

- Observable
- View
- Layout
- Router

The End

Thanks for attending!

@carlbergenhem

github.com/bergenhem/talks