More AngularJS

George Azmy | @grgzmy | grg@azmy.ca

Let’s recap

Angular teaches HTML new tricks.. cool tricks

Model(`$scope`) binds the View with Controller

Using automatic or manual `$digest` loops to *‘watch’* changes on `$scope`

Events

Allows controllers to talk to each other

You can dispatch events up the scope tree to parents, or down to children

Two $scopes without parent/child relationship can talk by dispatching down from $rootScope

Events: $emit vs $broadcast

```
$scope.$emit('pugWhoof', {state: 'hungry'})
//children of this $scope receive event


$scope.$broadcast('pugHappy', true)
//parents of this $scope, up to $rootScope,
receive event
```

Responding to events

```
function ParentCtrl($scope){
   $scope.$emit('pugHappy', true);
}
function ChildCtrl($scope){
   $scope.$on('pugHappy', function(event,
data){
   data ? alert('Pug Happy') : alert(':-(')
};}
```

Separation of Concerns

Controllers: Only deal with logic/state/flow

Directives: DOM manipulations; have access to $scope, so can respond to events

Services: This is where you data lives, and where the REST communication

Directives

Extend Angulars HTML compiler

Has access to $scope of HTML node

Only place you should ever manipulate DOM if you have to

```
<ng-pug > </ng-pug>

function PugCtrl($scope){
    $scope.pug = {name: 'Pablo', bestFriend: 'George'};
}

angular.module('PugApp').directive('ngPug', function(){
    return{
        templateUrl: 'pug.html'
    }
})

//pug.html
<span>{{pug.name}} is {{pug.bestFriend}}'s best friend!
</span>
```

Directive: restrict

Can be restricted to any combo of:

- A : Attribute //<div pug />
- E: Element //<pug />
- C: Class //

Services and REST

Hold the data, and abstract data operations form the Controller.

They are singletons; can implement factories

Controller agnostic of how it works..you can move stuff to backend, ctrl doesnt care

Services

Injected into controller

```
angular.controller('PugCtrl',
   ['$scope', '$pugs', function($scope,
$pugs){
      $scope.pug = {name: 'Pablo', age:2}
      $pugs.add($scope.pug);
}]
```

Singletons

MainCtrl($pugs), OtherCtrl($pugs) get the same $pugs service, which holds are the necessary data

One controller triggering a HTTP GET, will update the data for all other controllers

```
angular.module('PugApp')
.service($pugs, ['$http', function($http){
   var Pugs =  function(){
      this.list = [];
   }

   Pugs.prototype.get = function() {...}

   return new Pugs();
}]);
```

```
Pugs.prototype.get = function() {
   var that = this;

   $http.get('/pugs')
   .success(
   function(data, status, header, config){
      that.list = angular.copy(data);
   }).error(
      function(d, s, h, c){}
   );};
```

the $http service

Lets you perform AJAX calls

Async.. returns a promise

Enriched promise with regular `.then` and `.catch`, but also `.success` and `.error`

$http

Takes a config object

```
$http({
method: 'POST' //or GET, DELETE, PUT
url : 'api/pugs'
data: [pug1, pug2] //JSON payload
cache: true
}) //returns promise
```

Factories

Exactly the same as $service except not singleton

Returns a function that returns a service

```
angular.module('PugApp')
.service($pugs, ['$http', function($http){
   return function(country){
      var Pugs =  function(country){
         this.list = [];
      }
      Pugs.prototype.get = function() {...}
      return new Pugs(country);
      };
   };
]);
```

```
Pugs.prototype.get = function() {
   var that = this;

   $http.get(‘/pugs/’ + country )
   .success(
   function(data, status, header, config){
      that.list = angular.copy(data);
   }).error(
      function(d, s, h, c){}
   );};
```

Routing

HTML5 Routing

Angular has ng-route and it sucks

Being replaced for a state manager much like angulars most popular library **ui-router**

ui-router

Smart state manager

States have controllers, views, and parent/child

Many transition events

Can resolve services and inject into controller

```
myApp.config(function($stateProvider){
   $stateprovider
   .state('pug', {
      url: '/pug/:id',
      controller: 'PugCtrl',
      templateUrl:'/partials/main.pug.html',
      resolve:{$dogService: '$pug'}
      data: {friend: 'George'}
   });
}
```

States params and data

Inject **`$stateParams`** in Controller

Inject $stateData into controller

Get data of other states with **`$state.get('name').stateData`**

State transitions

```
$state.go('name', toParams) //from ctrl

$state.transitionTo //same as.go but low
level, returns prmomise

<div ui-sref="pug(id: 12)">

<a href= "/#/pug/12">Go</a>
```

State transitions

```
onEnter, onExit //callback in config obj

stateChangeStart, stateChangeSuccess

//events dispatched on state changes

stateChangeError, stateNotFound
```

Angular Testing

Frameworks: Jasmine + ng-mocks

Runner: Karma!

Karma runner: grunt!

Mocks

$httpBackend injected instead of $http in test runtime

lets you mock http calls, simulate response codes and data

lets you assert calls are made exactly as expected

```
describe("PugApp", function(){
   beforeEach(module('PugApp')
   var http, ctrl, pugs;
   beforeEach(inject(function($controller,
   $httpBackend, $pugs){
      pugs = $pugs; http = $httpBackend
      ctrl = function(){
         return $controller.get('pugCtrl')
      }
   }))})
```

```
it("should get pugs", function(){
   http.expectGET('/pugs')
   .respond(200, []);
   var c = ctrl();
   http.flush();

   expect(c.pugs).toBe([]);
   http.verifyNoOutstandingExpectation;

   http.verifyNoOutstandingRequest

})
```

```
describe("PugApp", function(){
   beforeEach(module('PugApp')
   var scope;
   beforeEach(inject(function($controller,
   $rootScope){
      scope = $rootScope.$new();
      ctrl = function(){
         return $controller.get('pugCtrl',{
            $scope: scope
         } )
      }})})}
```