AngularJS application architecture

Gabriele Falace
workshop 11.03.2015

Separation of Concerns

Rule of 1

- each component has 1 role
- 1 component per file
- each component has a single purpose

Separation of Concerns - example

Classifying concerns:

- Cross-Cutting and generic
  - logging
  - exception handling
- Cross-Cutting and feature-specific
  - a service fetching Customer data
- Features
  - Customer Controller
  - Customer Address widget

Consistent syntax

Many things can be done with different styles.

It is strongly recommended to stick with one, especially when working in team!

Consistent syntax - example

It is often needed to create an alias for the this reference.

WHY: in JavaScript a scope is created by a function definition (and not by {} blocks) → while nesting function definitions, the reference to the “original” this is lost.

Consistent syntax - example

```
function foo(){
    this.x = 5;
    // other code
    function bar(){
        // can’t access x through “this”
        // this “this” belongs to bar
        this.x = 6;
    }
}
```

Consistent syntax - example

```
function foo(){
    var self = this;
    self.x = 5;
    // other code
    function bar(){
        // can access “external” x through “self”
        this.x = 6;
        console.log(self.x);
    }
}
```

Consistent syntax - example

widely used aliases

- in controllers: `var vm = this; // view model`
- in general js: `var self = this;`

Consistent syntax - services

```
angular.module('globalServices')
     .service('paymentService', PaymentService);    // a service is a SINGLETON

/* @ngInject */
function PaymentService($http, tokenService){
     var service = {
       validateCardNumber: validateCardNumber,
       properCardNumberLength: properCardNumberLength
     };
     return service;

     // implementation
}
```

Organizing the App

can be by type of by feature

by type can be ok for small-medium

sized apps.

When the size grows this gets confusing.

larger app → by feature, then by type

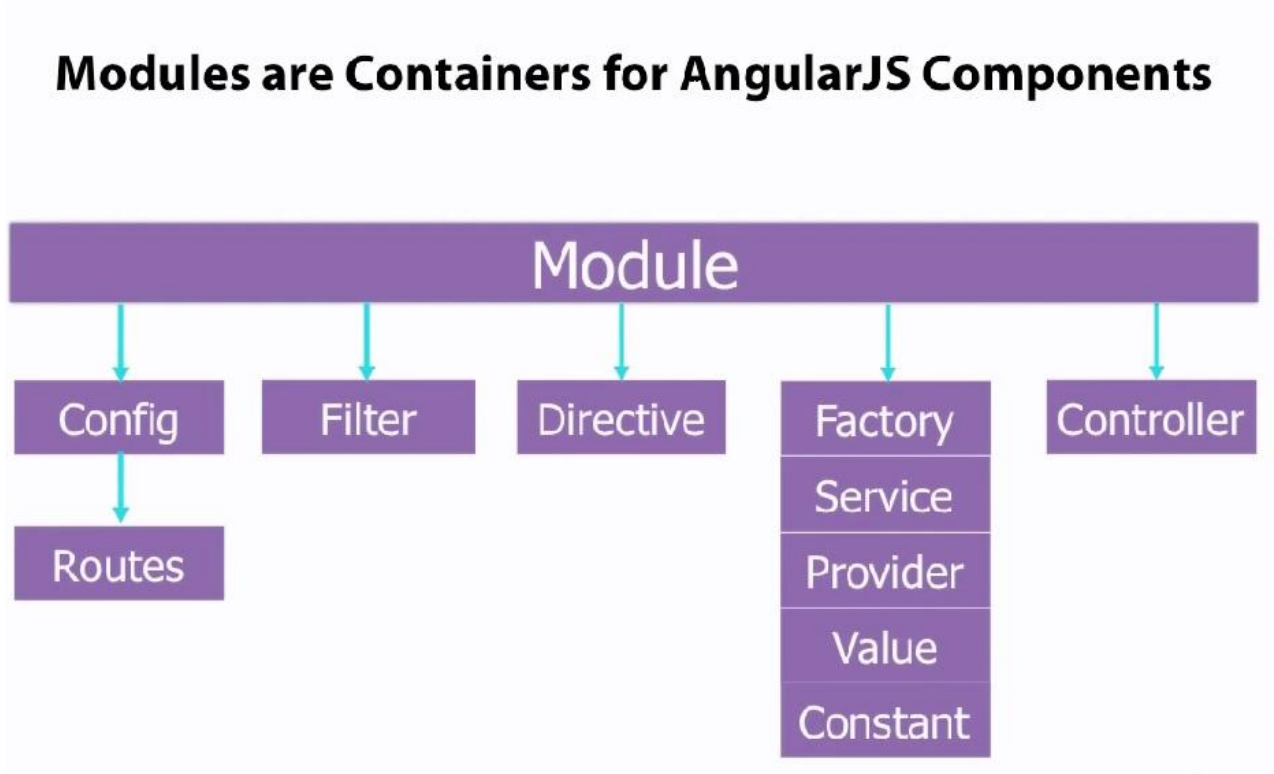
naming conventions

Controllers: avoid the suffix “Controller” use uppercase for the function name (it’s a constructor)

Factories/Services: same as file name

Directives: same as file name + short and consistent prefix (e.g. “eb”)

Modules


Modules are Containers for AngularJS Components
Module
Config
Filter
Directive
Factory
Controller
Routes
Service
Provider
Value
Constant

Modules

use angular.module() for both declaring and fetching modules.

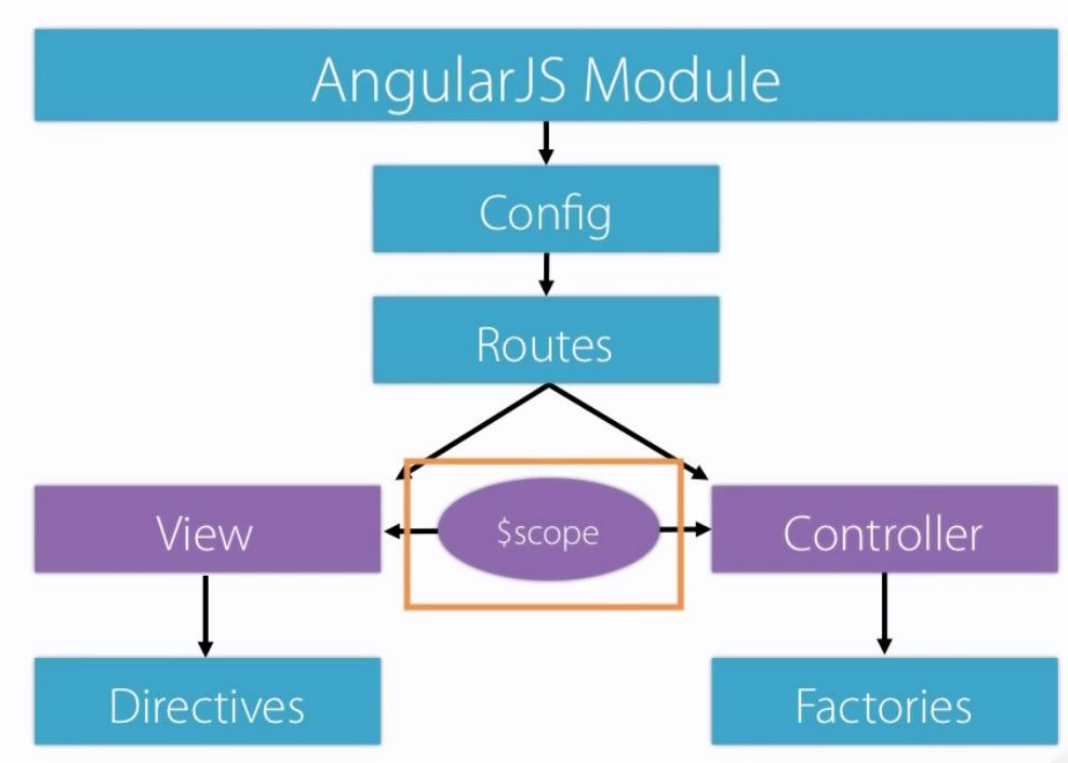
```
angular.module(‘app’, []); // CREATES
angular.module(‘app’);  // FETCHES

var app = angular.module(‘app’, []); //not recommended
```

Modules

Aggregate dependencies, in order to avoid an overly long list of dependencies in our modules.

Use a root “app.core” module to aggregate 3rd party and other generic modules.

Controllers



Controllers

- only **presentation** logic in here
- don’t use anonymous functions
- use the “Controller As” syntax with “this” instead of scope
- separate registration, injection, declaration

```
angular.module(‘app’).controller(‘foo’, Foo);
Foo.$inject = [‘$http’, ‘$scope’];
function Foo($http, $scope){ … }
```

Controllers

Declare the “Controller As” in ngRoutes, whenever possible, also using **resolve** for bootstrapping logic

```
controller: ‘Controller’
controllerAs: ‘ctrl’
resolve: {
    message: [‘customerService’, function(customerService) {
        return customerService.getGreetingMessage();
    }]
}
```

Controllers

The controller can be injected with “message” before the controller is instantiated → good mechanism for bootstrapping logic, better than “app.run” when logic should be specific to the controller

```
Controller.$inject = [‘message’];
function Controller(message){ … }
```

AJAX - sequential requests

```
$http.get(PRICELIST_URL)
 .then(function (data) {
       pricelist = data['data'];
       console.log('AFTER CALL 1 __ ' + pricelist + '(pricelist)');
       return $http.get(CURRENCY_URL);
 })
 .then(function (data) {
       currency = data['data'];

       var actualUrl = TARIFFS_URL.replace('{priceListId}', pricelist);
       return $http.get(actualUrl);
 })
 .then(function (data){
       console.log('AFTER CALL 3 __ (tariffs) \n' + JSON.stringify(data['data']));
 });
```

AJAX - parallel requests

```
$q.all([fn1(), fn2(),]);
```

array of functions each returning a Promise

```
function fn1(){
   return $http.get(SERVICE_URL);
}
```

Tips

using value/constant to keep everything in the scope of the application. constants and values can be injected.

```
angular.module('application').constant('origin', 'Sydney');

service('simpleService', ['origin', function (origin) {
     // implementation ...
}
```

Tips

It’s usually nice to consider …

- using ngAnnotate (e.g. `/* @ngInject */`)
- using jshint (to impose some coding style)
- using JavaScript
  - especially iterators over arrays:
    - `someArray.forEach(fn)`
    - `someArray.filter(fn)`
    - `someArray.some(fn)`
    - `someArray.every(fn)`
    - `someArray.map(fn)`

Unit Testing - installation

- choose a framework (e.g. Jasmine)
- choose a test runner (e.g. Karma)
- setup:

```
npm install -g karma
npm install -g karma-cli
```

in the project folder run the following

```
karma init
```

Unit Testing - configuring

- in the project root a karma.conf.js file specifies settings for the tests
  - make sure that the file property in the file references an array of file path where the JS app file and the test are located!
- Typically, most important things to test are Services, as they’re the components which hold business logic

Unit Testing - example

```
describe('testPaymentService', function () {
  var paymentService, $httpBackend;
  beforeEach(module('globalServices'));

  beforeEach(function () {
        inject(function($injector){
                $httpBackend = $injector.get('$httpBackend');
                paymentService = $injector.get('paymentService');
        });
  });

  afterEach(function() {
        $httpBackend.verifyNoOutstandingExpectation();
        $httpBackend.verifyNoOutstandingRequest();
  });

  it('should check mastercard', function () {
        $httpBackend
            .when('GET', REST_API_URL + '/services/payment/validateCreditCard/5105105105105100')
            .respond('MASTERCARD');
        var res = paymentService.validateCardNumber('5105105105105100');
        $httpBackend.flush();
        expect(res.$$state.value.data).toEqual('MASTERCARD');
  });
});
```

E2E Testing - installation

- choose a framework (e.g. Jasmine)
- choose a test runner (e.g. Protractor)
- setup:

```
npm install -g protractor
```

in the project folder run the following, to start the Selenium WebDriver

```
webdriver-manager start
```

Configuring

create the configuration file. Copy the following into conf.js:

```
exports.config = {
  seleniumAddress: 'http://localhost:4444/wd/hub',
  specs: ['todo-spec.js']
};
```

This configuration tells Protractor where your test files (specs) are, and where to talk to your Selenium Server (seleniumAddress).

Write a test

A protractor test, manipulates the actual elements in the app launching the browser and picking up stuff in the page through matchers. Run it with **`protractor conf.js`**

```
describe('Ebuero CSW number fields validation', function () {
        beforeEach(function () {
                browser.sleep(2000);
                browser.get(URL);
                ...
        });
        it('should format, validate and copy the phone, mobile and fax number', function () {
                companyName.sendKeys('000TEST000 ebuero AG');
                ...
                expect(faxNumber.getAttribute('value')).toBe(FORMATTED_PHONE_NUMBER);
                element(by.cssContainingText('option', 'AG')).click();
                nextButton1.click();
                expect(createErrorBox.isDisplayed()).toBeFalsy();
        });
});
```

References

- John Papa's course on PluralSight
- Style guide: https://github.com/johnpapa/angular-styleguide
- More on Services, Providers: http://slides.wesalvaro.com/20121113/#/2/5
- Possible JSON security threat: http://haacked.com/archive/2008/11/20/anatomy-of-a-subtle-json-vulnerability.aspx/