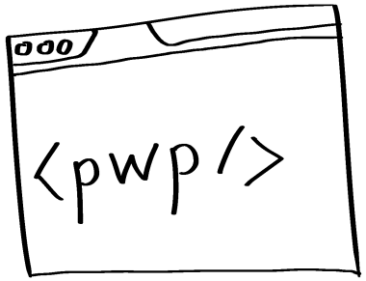
Practical Course: Web Development

Angular JS – Part III

Winter Semester 2016/17

Juliane Franze



Today's Agenda

- Lessons learned from Homework
- Advanced Angular Things
  - Data Binding & Watchers
  - Factory / Services
  - Inject
  - Controller As
- Testing
- Homework

Lessons learned from Homework

Have a look at your group members code.

What do you like and what would you do differently?

- Controller
  - How is it structured?
  - What tasks are conducted within one controller? Should they be moved?
  - Are all modules named and integrated properly?
- Structure
  - How do you like the current code structure?
  - How would you structure your final group project?
  - How and where would you create your HTML layout?
- Routing
  - How can you guarantee that I all routes lead to a valid page?

Let‘s dive into some Advanced Angular!

Data Binding & Watchers

- Data binding
  - Uses watchers ($watch API)
  - Watchers observe changes and model mutations on scope
  - Watchers are registered through directives
  - Each change triggers a digest cycle that automatically updates the DOM
  - Seen in ng-model=„test“
  - This may lead to performance issues if high amount of watchers reached

- Count Watchers to be aware of them
  - Plugin in Chrome
  - „Angular watchers“
  - https://chrome.google.com/webstore/search/angular%20watchers?hl=de

One Time Binding

One-time expressions will stop recalculating once they are stable, which happens after the first digest…

- Available since Angular 1.3
- New syntax: starting an expression with ::
- Works for all typical Angular expressions
  - `<h2> von: {{::todo.user}}</h2>`

Test it yourself

```
<input ng-model="test"></input>
<div>{{test}}</div>
```

- What happens when you add one time binding?

Factories / Services

- Defer logic in a controller by delegating to services and factories.
  - Logic may be reused by multiple controllers
  - Logic in a service can more easily be isolated in a unit test
  - Hides implementation details from the controller
  - Keeps the controller slim, trim, and focused
  - Factories and services are singleton

```
angular.module('pwp.factories.getData', [])
    .factory('getDataFactory', GetDataFactory);


function GetDataFactory() {
    console.log("Factory is called");
```

```
angular
    .module('pwp.controller.todo', [])
    .controller('TodoController',  TodoController);

TodoController.$inject = ['$scope', 'getDataFactory', 'userFactory'];

function TodoController($scope, getDataFactory, userFactory){

    console.log("todoController is called");
```

Injection to minify code

- Dependency injection is used everywhere in Angular
- Use „$inject“ to manually identify your dependencies
  - ControllerName.$inject = [what controller depends on]
  - Don‘t forget to put items in ‘ ‘
- This safeguards your dependencies from being vulnerable to minification issues

- Code:

```
TodoController.$inject = ['$scope', 'getDataFactory'];
```

Controller As

- $scope can be replaced –e.g with this - since Anguar 1.2
  - Controller as syntax does not give controller a new name – but the instance of the controller
  - In controller:

```
var ctrl = this;
ctrl.todo = ...
```

  - In HTML:

```
<div ng-controller="TodoController as vm“>
<h2> von: {{::vm.todo.user}}</h2>
```

- Or use it within the StateProvider
  - Then it wont show up in html Code
  - ** **

```
.state('state1.list', {  //The nested view in the view "state1"
    url: "/list",
    templateUrl: "views/state1.list.html",
    controller: "ListController",
    controllerAs: "vm"
})
```

- https://angularjs.de/artikel/controller-as-syntax

Now it‘s time for Testing.... ☺

Why testing?

- It is good practice ☺
- JS comes with almost no help from compiler
- Best way to prevent software defects
- If features are added or removed potential side effects can be seen
- You will have a good feature documentation
- Angular
  - Is written with testability in mind
  - Dependency injection makes testing components easier

Karma

- Command line tool
  - Results are listed in command line as well
- Tests several browsers
  - Good to know that application runs in all browsers
- A NodeJS application
- A direct product of Angular team
- http://karma-runner.github.io/0.12/intro/installation.html

Jasmine

- Popular JS unit testing framework
- Not tied to a particular framework
  - But popular for testing Angular applications
- Tests synchronous and asynchronous JS code
- Used in BDD (behavior-driven development)
  - focus on business value not on technical details

- 2 important terms
  - Suite & spec

Suite and Spec

```
//This is test suite
describe("Test Suite", function() {
    //.....
});
```

Suite

- A group of (related) test cases
- Used to test a specific behavior of JS code (function)
- Starts with call of Jasmine global function:
  - „describe“
  - with 2 parameters (<title of suite>, function implementing test suite)

```
//This is test suite
describe("Test Suite", function() {
    it("test spec", function() {
        expect( expression ).toEqual(true);
    });
});
```

Spec

- Represents an individual test case
- Begins with Jasmine global function:
  - „it“
  - With 2 parameters (<title>, function implementing test case)
- Contains one or more expectations
- Expections
  - Represent an assertion that can be true or false
  - To pass a spec: all expactions inside the spec have to be true
  - If one or more expectations are false → the spec fails
- There are pre-defined matchers

Test

- Load application module
- Load a special test module to overwrite setting (configuration) in tests with a mock version
  https://docs.angularjs.org/guide/module

- Use underscore notation
  - For variable names in tests: „_$rootScope_“
  - It is an Angular convention
  - $injector strips them out if they apply at start and end with exactly one underscore

Homework

- Extend your current homework
  - Write an own filter for your user-overview app
  - Write a test that tests the filter and the request
- Have a looke at Grunt and Gulp
  - Discuss advantages and Disadvantages for your final app
  - Decide within the team what you want to use
- Use Bluemix for Deployment or sth similar... ☺
- Have a look at this:
  - https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md#modules

Next year...

- Present second version of your application
- Coding review if wanted in first week

For boring evenings....

- Angular Best Practice: https://github.com/johnpapa/angular-styleguide
- Code a project: https://docs.angularjs.org/tutorial