Practical Course: Web Development

Angular JS – Part I
Winter Semester 2016/17

Juliane Franze
Today’s Agenda

• What is Angular?
  – Origin
  – Architecture
  – Best Practices

• ... In between... Hands on 1\textsuperscript{st} Angular App
Angular is..

• A JavaScript Framework
  – Built for SinglePage application (SPA)
  – For big scale applications
  – Distributed as a JS-File
  – Frontend part of MEAN Stack

• Created by
  – Miško Hevery (former Google Employee)
  – Started in 2007
  – V1.0 released in 2009
  – V2.0 announced in 2014
    - final since September 2016

• Used by

https://www.madewithangular.com
Concept of Angular

• 2-way data binding
  • Eases data handling by tight coupling between input and values
  • Automatic synchronization of models and views
  • Taken from OO languages to JS world

• Dirty Checking
  • Checks for changes in model and updates view

• Decouple DOM Manipulation from application logic
  • Safe boiler plate code
  • Increase testability and performance

• Dependency injection
  • Brings traditional server-side services to client

• Scopes
  • Glues data between view and controller

• Follow several Design Patterns
  • Singleton
  • MVC
  • MVVM
Advantages

• Open Source framework
• Maintained by Google and a big community
• The Code:
  – Angular analyses the DOM directly
  – Builds bindings based on angular attributes or elements
  – Less boilerplate code thanks to 2-way data binding
  – Therefore code is cleaner, easier to understand, less error prone
  – Extended features: dependency injection, deep-routing, build in filter
  – Good Error Feedback from Angular via browser console
Disadvantages

• **Angular is big**
  – Multiple ways to do one thing
  – Hard to tell which way is the best for a particular task
  – Different people take different coding styles
  – Might complicate group coding → try to stick with one!

• **Black boxes**
  – You don’t know at the beginning how angular works internally
  – Once your application grows and your skills improve it is likely that you change used approaches
  – If you reach more than 3.000 watchers your app will lag in responsivness
  – Third Party modules might be programmed sloppy
Structure of an application
Setting up an Angular Project

1. Create index.html
   - Contain placeholder for view elements
   - Add first dummy data /structure

2. Create folder structure
   - 2 ways possible semantic or syntactic
   - Just a matter over easier understanding and maintenance

3. Include Angular & Add Modules
   - Decide on structure
   - Create a folder hierarchy

4. Create bi-directional data binding
   - Add $scope
Let’s get started!
You should have

• Node
  – Check: node –v

• Bower for Frontend Libraries
  – npm install –g bower
Create Angular via npm in Webstorm

- **File | New Project** – select „Angular JS“ – name project
  - This is a skeleton only
  - Need for installing components

- **Open Terminal**
  - Install npm locally
    - `npm install`
    - `/bower` will be installed with it
    - `angular` will be installed as well

  - See if angular works...

  - Install Bootstrap css locally
    - `bower install bootstrap-css-only`
    - Add css links to index.html

  - Update all packages locally
    - `bower update`
2. Create Folder Structure

• Start with:
  – The flattest structure that makes sense
  – Design for what you know so far
  – This does not paralyze to make the wrong choice
  – You can adjust as needed

• Create ONE feature per file
  – Each controller, service, factory, view has its own file

• 2 options
  – Structure and name Folder by type (css, images, controllers)
  – Structure and name Folder by content (dashboard, loginpage,...)
Structure

Per Type

App/
app.module.js
app.config.js
app.routes.js
directives.js
controllers/
topnav.js
content.js
dashboard.js
views/
tovnap.html
content.html
dashboard.html
factories/
localstorage.js
rest-requests.js

Per Content

App/
app.module.js
app.config.js
app.routes.js
directives.js
topnav/
tovnap.html
tovnap.controller.js
dashboard/
content.html
dashboard.html
content.controller.js
dashboard.controller.js
misc/
localstorage.factory.js
rest-requests.factory.js
LIFT Guidelines

L  Locating your code is easy
   • Find the code you fast want is super important

I  Identify code at a glance
   • When you look at a file you should know what it contains
   • No files with multiple controllers of even mixed code

F  Flat structure as long as you can
   • Try to make the way as short ar possible (no 7 levels)

T  Try(!) to stay DRY (Don’t repeat yourself)
   • Important but not crucial
   • Try to find a good way for you to avoid redundant information but keep the code readable
2. Naming Conventions

• Name modules as precise as possible!
• CONSISTENCY within a project and a team is important → provides efficiency and more maintainable code
• Write conventions down or at least talk about it & remind yourself and your team mates
  – Dash vs. camelCase
  – German vs. English
  – Abbreviation vs. Full words
  – Dots for separation of category of features vs. Nothing like that
2. Naming Convention

• In Angular there are 2 names for most assets:
  1. The file name:
  2. The asset name with Angular

• The main module is always named: app.js

• All other dependents are named what they represent:
  – Admin.module.js
  – Admin.controller.js
Angular Modules

• Every Angular Application has at least ONE Module
  – File: App.js
  – Start module: ng-app

• Additionally there are
  – Ng-controller
  – Ng-model
  – Ng-view
  – Filter
  – Services
  – Directives
Structure + Modules

Structure

Add Module

• Add new file: controller.js
• Add script-Tag for controller.js into index.html
• Fill controller.js with life

• Add attribute to html
Controller

- A JS constructor function
- Used to augment the Angular Scope
- When controller attached:
  - Angular instatiates a new Controller object
  - A new child scope is created
- Any object assigned to the scope become model properties
  - Any methods can be invoked via angular expression "ng"
- Should not do too much
- Keep them slim
  - Encapsulate work that does not belong to controllers into services
  - Use these services in controllers via dependency injection
$scope

• Concept of $scope is crucial in Angular
• $scope is a simple JS object
• Available for view and controller
• Allows to exchange information
• Inherit from their parent scopes – all up until root scope
• Root Scope is visible in entire application
• Do not use rootScope for communication among scopes
$scopes

Template

```
<html ng-app="demoApp">
  <body>
    <div ng-controller="contactController">
      <tr ng-repeat="person in persons">
        <td>{{person.name}}</td>
      </tr>
    </div>
  </body>
</html>
```

Model

Root Scope

contactController Scope

Repeater Scope

Scope inheritance
Implicit Scope Declaration
UI-router

• Install UI router in Webstorm
  – Bower install angular-ui-router
• Add script tag to index
• Create new ui-view in html
• Add links with ui-sref directive
• Add Templates(html) that will plug into the ui-view
• Create config & set up states

→ Advantage of UI-Router – views can be nested!
MEAN Stack

• helps in the creation of a complete JavaScript web apps
• 2 options
  – http://mean.io
  – https://meanjs.org/
• Differences:
  https://medium.com/@chrisbateson80/mean-js-vs-mean-io-723123051d14#.aqa0p32l7
• Challenges:
  – Dependency Problems – only tested on node 0.10 extensively
1. Install mean-cli package & modules
   – npm install <module> [options]
   – npm install –g mean-cli bower gulp
2. Init
   – mean init myFirstApp
   – → Clones mean –repo from github
   – Follow the instructions given in terminal
3. Install npm in project folder
   – go into project folder
   – npm install
Learn more..

- W3C School intro: http://www.w3schools.com/angular/
- Code a project: https://docs.angularjs.org/tutorial
- Help for Webstorm
- Check out code from other websites built with Angular
  - e.g. https://www2.sixt.jobs/de/en/
Next week...

• More about $scopes
• Start with nesting controllers
• Extend controllers
• Information exchange among controllers
• Own filters
• Answer your questions that you may sent me via slac (until Thursday 5pm CET)