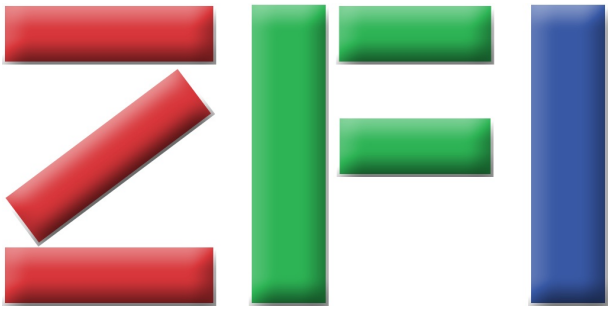
Kurs-Dokumentation



Zentrum für Informatik ZFI AG

Angular 4+ ? Der komplette Leitfaden (JANG)

http://www.zfi.ch/JANG

Weitere Infos finden Sie unter www.zfi.ch oder via Adresse:

Zentrum für Informatik ZFI AG
Zentralsekretariat
Rütistrasse 28
CH-8952 Zürich-Schlieren
Telefon: 044 732 40 00
Telefax: 041 530 31 68

Zürich, Basel, Bern, ZÃ¼rich, Schweiz

Titel	**Angular 4+ ? Der komplette Leitfaden**
Untertitel	**Alles was für die moderne Applikationsentwicklung mit Angular 4+ benötigt wird.**
Einleitung	**AngularJS ? oft einfach als Angular bezeichnet ? ist ein clientseitiges JavaScript-Webframework zur Erstellung von Single-page-Webanwendungen nach einem Model-View-ViewModel-Muster. Die Softwareentwicklung und das Komponententesten können damit vereinfacht werden. Es wird als Open-Source-Framework vom US-amerikanischen Unternehmen Google Inc. entwickelt.** **Angular basiert auf der clientseitigen Generierung von HTML-Ansichten und Erweiterungen des Vokabulars von HTML. Hierdurch kann Funktionalität im Rahmen der View abgebildet werden, ohne auf DOM-Manipulation via jQuery zurückgreifen zu müssen. Angular behandelt die Datenvalidierung im Rahmen von Eingabeformularen als Funktionalität der View. Hierbei wird der Gedanke der HTML5-Form-Validation fortgesetzt und durch AngularJS in eine Webbrowser-unabhängige Javascript-Variante implementiert. Diese Vorgehensweise besitzt keine Funktionalitäten zur Abbildung von Fachkonzept-Daten (Model) in Form einer clientseitigen Entitätenverwaltung. In anderen SPA-Frameworks stellt dies einen typischen Bereich zur Datenvalidierung dar. Stattdessen werden die Fachkonzeptdaten häufig direkt in einem ViewModel abgelegt. In dem offenen Vorschlag ?Data Persistence in Angular 4+? wird über eine Erweiterung hinsichtlich dieses Funktionsbereichs nachgedacht.** **Die Strukturierung eines Angular-Webclients erfolgt auf Basis von Modulen, View-Templates, Controllern, Scopes, Filtern und Providern (Factory, Service). Für die Zusammenführung dieser Elemente ist der Dependency-Container von Angular verantwortlich. Hierdurch entsteht eine lose gekoppelte Anwendung, welche aus wiederverwendbaren Teilkomponenten besteht. Unter Berücksichtigung der ISO/IEC 25010 können so wartbare Anwendungen realisiert werden. Angular besitzt einen Mechanismus zur Datenbindung nach dem MVVM-Muster. Hierdurch kann Programmcode zur Synchronisation zwischen View und Anwendungslogik eingespart werden. Durch deklarative Beschreibungen von Datenbindungen innerhalb der View leitet Angular eine bidirektionale Änderungsverfolgung der Werte ein. Dies kann auch starke Auswirkungen auf die Performance einer SPA haben. Technisch gesehen befindet sich Angular hierzu in einer Eventschleife um jede Änderung abzufangen, auszuwerten und ggf. eine Aktualisierung der View zu initiieren. Nicht editierbare Daten können mittels Angulars One-Time-Binding von weiteren Aktualisierungen ausgeschlossen werden.** **(Text: Wikipedia)** **Das Team hinter dem Projekt bezeichnet Angular 4+ als Framework «für das Web von morgen». Als solches nutzt es konsequent aufstrebende Webtechniken ? allen voran Web Components. Dabei handelt es sich um einen Ansatz, der auf mehreren beim W3C eingebrachten Vorschlägen fußt, und das Schaffen wiederverwendbarer JavaScript-Komponenten erlaubt. Webanwendungen können heute schon den aktuellen Entwicklungsstand dieser Vorschläge nutzen, zumal einige Browser sie**

	bereits nativ umsetzen. Für andere liegen entsprechende Polyfills vor. Bei der Entwicklung von Angular 4+ setzt das Produkt-Team auf die bei Microsoft entstandene Sprache TypeScript, die eine Obermenge von ECMAScript in Version 6 darstellt und künftig auch Sprachmerkmale anbieten wird, die für das darauf folgende Release geplant sind. Daneben bietet TypeScript ein optionales statisches Typsystem.
Ihr Nutzen	
Vorausetzungen	**- Gute Kenntnisse in HTML** **- Gute Kenntnisse in CSS** **- Gute Kenntnisse in JavaScript** **- Erfahrungen mit OOP**
Teilnehmerkreis	
Unterlagen	
Folgekurse	**Ionic 2 Mobile Applikationen mit Webtechnologien (ZFI-Kurscode ION2)**
Inhalt	**Welcome** **- Introduction of course participants & teacher** **- Discuss expectation and suggested procedure of course** **Getting started** **- Review the state of knowledge** **- Installing required software** **- Examine a getting started application** **- Advantages & disadvantages of angular 4+** **- When to use angular 4+** **Writing a skeleton application (for all further exercises)** **- An easy getting started clone with detailed explanation** **- Taking a look at npm** **- Taking a look at tsc**

- **Taking a look at Angular 4+ decorators**
- **Taking a look at typings**
- **Taking a look at gulp / webpack**
- **How Angular 4+ bootstraps the app**

Typescript a quick reference

- **What is typescript?**
- **Why using typescript?**
- **Classes**
- **Built in types**
- **Typescript the angular 4+ way**
- **Inheritance**

Sass a quick reference

- **What is sass?**
- **Why using sass?**
- **How to use it?**

Angular 4+ demystify the syntax

- **Directives ? a quick overview**
- **Components**
- **Decorators**
- **Interpolation**
- **Template expressions / statements**

- **Property binding**
- **Angular 4+ lifecycle**

Angular 4+ writing a component & databinding

- **Angular 4+ guidelines**
- **Project: Getting started with a component (TaskComponent)**
- **Typescript create a task model**
- **One-way binding**
- **Two-way property binding**
- **Styling our component**
- **Delete a task**
- **Create a new angular 4+ event**

Router

- **How Angular 4+ routing works**
- **Basic router structure**
- **Project: creating a CRUD router**
- **Securing our route. (General talk about security in Angular 4+)**
- **Introduction to AuthGuard**
- **Project: Creating and AuthGuard**

Dependency injection

- **What is dependency injection?**
- **Advantages to dependency injection?**

- **How to use it in Angular 4+?**
- **Angular 4+ Providers & Injectables**
- **Visibility & search tree / Component visibility**
- **Project: Creating an injectable class**

Services & HTTP

- **How to use Angular4+/HTTP**
- **What are observables?**
- **How to use observables?**
- **What are promises?**
- **How to use promises?**
- **Project: Writing a weather forecast service**

NodeJS a quick reference

- **How to write an API**
- **Cloning the NodeJS project**
- **Project: Create a restfull task-API**
- **Project: Create restfull user-API**

Setting up a database

- **Installing software**
- **Configure our database connection**

Forms (Template driven)

- **Including modules**
- **What are validators**
- **Project: Creating a custom validator**
- **How forms works in angular4+**
- **Project: Writing a ?CreateTaskComponent?**

Project 1

- **Creating a user model**
- **Creating a user**
- **Create a login**
- **Create a task application**
- **Components / Service / Views / Routes / Authentication**

Forms (Reactive driven)

- **Including modules**
- **How to optimize our ?CreateTaskComponent?**
- **Advantages of reactive driven forms**
- **Project: Optimizing our 1. Project with reactive forms**

Dependency Injection advanced techniques

- **Using factories**
- **Analysing common patterns**
- **Circular dependencies**
- **Injecting a new service**

- **Injector**

Pipes

- **What are pipes**
- **Stateless vs purse pipes**
- **Project: Creating a pipe**

Directives advanced

- **Three for one: Structural pipe, Components, Attribute directives**
- **Project: Creating Structure & Attribute directive**

Animations

- **Quick introduction to animations**
- **Why use animations?**
- **Project: Implementing a delete animation in Project 1**

Security

- **HTML Binding**
- **Trusting safe values**
- **Protection of CSRF or XSRF**

Testing

- **A quick introduction to angular 4+ testing (Jasmine)**

- **Writing mocks**
- **Project: Adding a couple of tests to our Project 1**

Custom Input form type

- **What are custom input types**
- **Why using custom input types**
- **Project: Create an input type switch**
- **Project create an input type counter**

Project 2:

- **Create a simple inventory tool**

Form advanced techniques

- **Dynamic form validation**
- **Async form validation**
- **Project: Create a ?is? username taken validator?**

Review & further information

- **What?s next**
- **Greate blog posts & communities**
- **Advanced Angular 4+ techniques**

Beitrag	**Der Teilnehmerbeitrag versteht sich rein netto. Das ZFI ist (gemäss MwSt-Gesetz) nicht Mehrwertsteuerpflichtig und erhebt somit keine MwSt. Bei länger als einen Monat dauernden Lehrgängen ist die Zahlung des Teilnehmerbeitrages in mehreren Raten möglich (pro rata temporis).**