

Angular ist ein TypeScript-basiertes Front-End-Webapplikationsframework. Es wird von einer Community aus Einzelpersonen und Unternehmen, angeführt durch Google, entwickelt und als Open-Source-Software publiziert.

### Ihr Nutzen

In diesem Seminar lernen Sie, wie sich mit Angular große und robuste Unternehmensanwendungen entwickeln lassen. Basierend auf einem vollständigen Projekt und vielen Beispielen werden zahlreiche Architekturansätze und Best Practices diskutiert. Am Ende liegt eine Vorlage für Ihre eigenen Projekte

### Preis pro Teilnehmer

EUR 2450,- exklusive der gesetzlichen MwSt.

### Seminardauer

4 Tag(e)/Day(s)

### Seminarinhalte

#### Tag 1

##### \* Standalone Components: Concepts & Migration

- Standalone Components vs Modules
- Creation, Bootstrapping
- Registering Providers & Dependency Injection
- Routing & Lazy Loading
- Migration to Standalone Components

##### \* Components & Forms Deep Dive

- Using & Migrating to Control Flow Syntax
- Deferred Loading
- Standalone Directives & Directives Composition Api
- Components and Required Inputs
- Content Projection
- Templates TemplateRef, \*ngTemplateOutlet
- Comparison: ng-template vs ng-content - pro / cons
- ViewChild, -Children, ContentChild, -Children
- HostBinding & HostListener
- Recap Reactive Forms Revisited (FormGroup, Form Builder, FormControl, FormArray)
- Dynamic Component Loading & DataBinding
- Untyped Forms vs Typed Forms
- Typed Forms Nullability, NonNullableFormBuilder, GetRawValue
- Partial Values, Optional Controls, Dynamic Groups and FormRecord
- Cascading Form Controls
- Implementing Custom Controls using ControlValueAccessor
- Typed Forms Validation & Custom Validators
- Handling FormErrors & ErrorStateMatcher

##### \* Mastering Reactive Programming using RxJS

- Imperative vs Functional Programming
- Immutability & Pure Functions
- Introduction to RxJS
- Observables, Observers & Use Cases
- Data- vs Action-Streams
- Mouse & DOM Events as Observables
- Implementing Side Effects using tap
- Base Operators: Mapping, Filtering, Merging, Scanning, ...
- Unsubscribing (takeUntil, DestroyRef, takeUntilDestroyed)
- Introduction to Signals
- Imperative vs Declarative Reactive Programming
- Signals vs Observables: Synchronous & Asynchronous Reactive Programming

### Voraussetzungen

Angular Basic~9757

oder dem entsprechende Kenntnisse

### Hinweise

Version: 2019

- Combination & Transformation Operators
- Retry & Error Handling Strategies
- Implementing & Testing Custom Observable Operators
- Communication between using Event Bus Pattern
- Stateful Services using Behavior Subjects and Signals

#### Tag 2

##### \* Advanced State Management using NgRx

- Overview State Management Patterns
- Introduction to the Redux Pattern & NgRx
- Feature State and ActionReducerMap
- Implementing NgRx Store, Reducers & Selectors using createFeature
- Actions & createActionGroup
- Debugging NgRx using Redux Dev Tools
- Effects, Facades, @ngrx/entity adapters
- Simplifying Data Access with @ngrx/data
- Implementing a reactive View Model
- NgRx Container Presenter Best Practices
- Using @ngrx/component-store
- Optimize Change Detection

##### \* Mastering Signals

- Introduction to Signals (Writable, Computed, Effects)
- Working with Arrays
- Signals vs Observables: Synchronous & Asynchronous Reactive Programming
- Signals & NgRx Interoperability
- Creating a Signals Store using @ngrx/signals
- Local Change Detection using Signals

##### \* Advanced Routing and App Initialization

- Dependency Injection in Depth: Resolution modifiers and Dependency providers
- Using Constructor vs inject for DI
- APP\_INITIALIZER, Injection & forwardRef
- Implementing Global Error Handling and Retry-Patterns
- Modules & Code Splitting
- Introduction to @ngrx/router-store
- Routing using NgRx Actions
- Binding Router-Params to Component Inputs
- Functional Route Guards & Interceptors

##### \* View Transition API

- Auxiliary Routes: Common use cases

