Angular Hands-on Workshop

Ervin Suhanko
Front End Software Engineer,
Team Lead & Mentor

Prerequisite

- Basic understanding of HTML, CSS, and JavaScript
- Basic understanding of Programming
- Familiarity with ES6 and Typescript is helpful

5th – 6th – 7th – 8th – 9th Sep'22 | 9:30 – 10:30 AM EST

Agenda

Day 1	Day 2	Day 3	Day 4	Day 5
Angular Introduction and Framework Overview	Modules	Routing	RxJS Essentials	Tips, Tricks and Best Practices
Angular CLI (Command-line Interface tool)	Components and Data flow	Pipes and Directives	Services and HTTP	
Angular Project Structure		Reactive Forms		
TypeScript Essentials				

Component Driven Architecture

- Components are small, encapsulated pieces of software that can be reused in many different contexts
- Angular strongly encourages the component architecture by making it easy (and necessary) to build out every feature of an app as a component
- Angular components self encapsulated building blocks that contain their own templates, styles, and logic so that they can easily be ported elsewhere

Component Driven Architecture

- This helps us solve the problem of structure
- This helps us solve the problem of communication

Component Driven Architecture

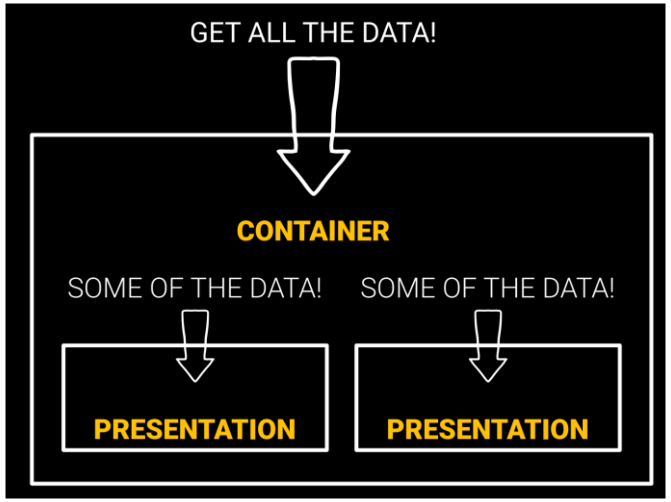
Container and presentational components

- Container components are connected to services
- Container components know how to load their own data, and how to persist changes
- Presentational components are fully defined by their bindings
- All the data goes in as inputs, and every change comes out as an output
- Create as few container components/many presentational components as possible

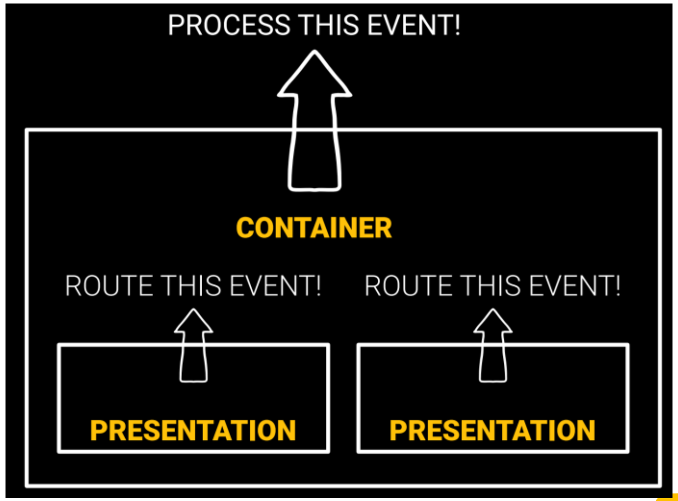
Component Driven Architecture

Container and presentational components

State Flow down



Events flows up



Demo

Component Driven Architecture

Component Driven Architecture

Use State Management

The biggest problem in the development and maintenance of large-scale software systems is complexity — large systems are hard to understand.

We believe that the major contributor to this complexity in many systems is the handling of state and the burden that this adds when trying to analyze and reason about the system. Other closely related contributors are code volume, and explicit concern with the flow of control through the system.

Component Driven Architecture

Test Your Code

- Small methods are easier to test
- Pure methods are easier to test
- Focus on testing just what that method does
- Don't use real services, faking and spying are both great options

Best Practices

Coding

- Single responsibility principle
- Symbol naming
- Preferring immutability
- Using small functions

Best Practices

File and Folder Structure

LIFT Principle

- **L**ocate code quickly
- **I**dentify code at a glance
- **F**lattest structure possible
- **T**ry to be DRY

Best Practices

File and Folder Structure

- Using the Angular CLI
- File Naming
- Folder Structure

Best Practices

Components

- Prefixing component selectors
- Using separate CSS and template files
- Decorating input and output properties
- Delegating complex logic to services
- Component member sequence
- Implementing lifecycle hook interfaces
  When to create components

Best Practices

Services

- Marking services as injectable
- Using services for data retrieval
- Working with the Angular injector

Best Practices

Performance

- Ahead-of-time compilation and the CLI
- Lazy loading feature modules
- Paying attention to bundle sizes
- Immutability and OnPush change detection
- Pure and impure pipes