Visual Studio LIVE! | Las Vegas

EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

An Introduction to Getting Up and Running with Modern Angular

Allen Conway
Principal Software Engineer
Cognizant

Level: Introduction

#VSLIVE

Your Code Powers the World.
Our Training Powers You.

Allen Conway

Principal Software Engineer

cognizant

Blog: https://www.AllenConway.net
Twitter: @AllenConway
GitHub: https://github.com/AllenConway
Email: Allen.Conway@Cognizant.com

30th ANNIVERSARY VSLive! 1993 - 2023

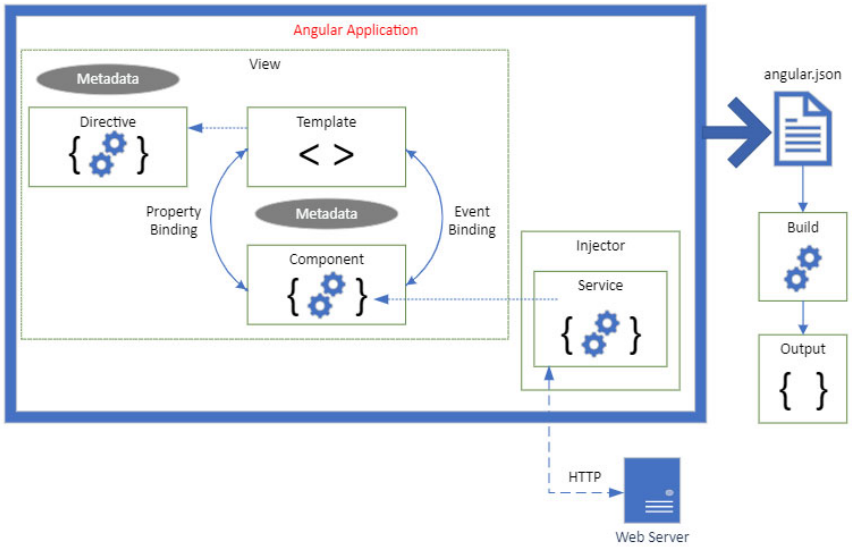
Agenda

- Welcome to Angular
- TypeScript
- Getting Started: CLI
- Building Blocks
  - Components (Standalone)
  - Directives
  - Services / HTTP
- Routing
- Responsive Design
  - Component Libraries
- RxJS
- Unit Testing
- Builds & Configurations
- Architecture

Angular

- Mature web client framework
  - 12+ years net traction
  - Currently at version 15
  - Completely rewritten in 2016
- Rooted in component-based architecture
- Feature rich
- Built with TypeScript
- Enterprise ready
- Rich ecosystem of OSS and developer experience

Architecture / Flow



TypeScript

- High level language like JavaScript yet offering familiar OO concepts and techniques
- Superset of JavaScript
- All valid JavaScript is valid TypeScript
  - Transpiled to JavaScript
  - Microsoft chose to build atop JavaScript
  - Big wins for TypeScript in Angular adoption
- Offers compile-time checking for type safety
  - #failfast
- Provides features ahead of ECMAScript Standards
- Pick your favorite editor or IDE
- Open Source
- Intellisense!

Angular CLI / Tooling

https://angular.io/cli

- Preferred way to initialize, develop, scaffold, build, and test
  - No hand rolling
- Leverages the command line interface
- `npm install -g @angular/cli`
- Leverage 'ng' commands

Angular CLI Core Commands

- Create an app
  - `ng new`
- Run the app
  - `ng serve`
- Test the app
  - `ng test`
  - `ng e2e` (requires config)
- Help
  - `ng help`

https://angular.io/cli

- Build the app
  - `ng build`
  - `ng build -c production`
  - `ng build -c` (custom env)
- Lint the app
  - `ng lint`
- i18n
  - `ng xi18n`
- Update the app
  - `ng update`

Angular CLI Generate Command

- https://angular.io/cli/generate
- `ng generate | ng g`
  - component ‘c’
  - module ‘m’
  - class ‘cl’
  - service ‘s’
  - directive ‘d’
- `ng generate | ng g`
  - interface ‘i’
  - enum ‘e’
  - pipe ‘p’
  - guard ‘g’
  - service-worker

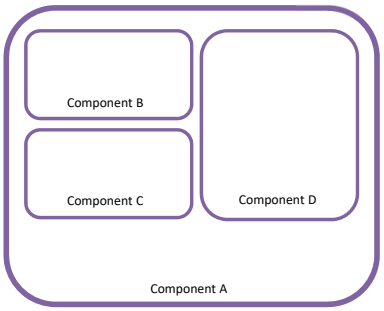
ANGULAR CLI
https://angular.io/cli

Building an Application

- Leverage the CLI
- Seasoned Angular dev/architect helps
- Solidify testing plan
- Add responsive framework/components
- State management
- Leverage modern Standalone APIs

Components

- Primary functions
  - Provide a section of the screen or view
  - Fundamental building block
  - Respond to a route change
  - Participate in a component hierarchy
  - Handle interaction
  - Fetch data for the template
- @Component *base* metadata
  - Selector
  - Template
  - Styles



Component Data Binding

- Display data by binding controls in an HTML template to properties of an Angular component
- Bind the property name through *interpolation*
  - One-way data binding
    - {{value}}
    - [ngModel]="value"
  - Two-way data binding (AKA 'Banana in a Box')
    - [(myInputValue)]
- Event Binding

```
<button (click)="onSave()">Save</button>
```

target event name — template statement

Component Syntax

- Expression syntax
  - <h1>Hello {{firstName + lastName}}</h1>
  - Less advisable
    - Prefer expression logic in imperative code in TS
    - Angular executes template expressions after every change detection cycle
- Pipes syntax
  - <!-- Default format: output 'Jun 15, 2015'-->
  - Today is {{today | date}}
  - Built in and custom pipes

Standalone Components

- Streamline Angular project composition
- Replaces the need for ng Modules
- New in Angular 15
  - Continue to work seamlessly with existing code and modules if required
- Reduces the ‘complexity’ black eye

Angular Change Detection

- Process which checks for application state changes and updates the DOM
  - Change detection cycles triggered manually or asynchronously
    - Promise resolution
    - HTTP results
    - Timers
    - Key presses and mouse moves
  - Expressions should finish quickly
    - User experience may drag, especially on slower devices
    - Cache values when computation is expensive.

Directives

- Adds addition behavior to elements in the application
- 3 types of directives
  - Component
    - Contains code, view, and styles
  - Structural
    - Responsible for HTML layout
    - Shape or reshape the DOM's structure
    - `*ngIf, *ngFor`
  - Attribute
    - Listen to and modify the behavior of other HTML elements
    - Built in attribute directives: `[ngClass], [ngStyle], [ngModel]`
    - Custom: `<label my-directive></label>`

Services

- A service is typically a class with a narrow, well-defined purpose
  - Should do something specific and do it well
  - This is a general OO/encapsulation concept no different than C#
- Separate a component's view-related functionality from other kinds of processing
  - Strive for thin Components
  - Don't let domain logic slip into the component
  - Angular will not enforce these principals
- Uses @Injectable decorator to notate DI
  - Angular injector is responsible for creating service instances and injecting them into classes
  - Use `providedIn` to create a provider for the service
    - Use 'root' unless only required for a particular module
    - Allows for tree-shakable providers

Http Client

- Offers a simplified client HTTP API for Angular apps
  - rests on the XMLHttpRequest interface exposed by browsers
- Features
  - Typed request/response objects
  - Testability
  - Observable APIs
  - Streamlined error handling
- Import HttpClientModule into root standalone component
- These types of calls should not be in a Component
  - Should exist in a service with post-processing logic

Routing

- Uses provided URL to navigate to specified view
- Router logs activity in the browser's history
- Route has 2 main properties
  - Path
  - Component
- `<router-outlet>` informs router where to display views
- Consider giving each logical feature its own route configuration file
- Keep default and wildcard routes at the root

Routing – Lazy Loading

- The Achilles heel of SPAs: the initial bundle size
- Asynchronous routing loads feature modules lazily
  - Load on demand via user's interaction
  - Speed up load times for users only using a portion of the app
  - Expand lazy loaded modules without increasing initial load bundle
- Easy to configure
- Standalone components allow lazy loading

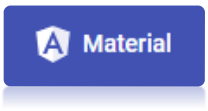
Responsive Design

- Further elevate your code :: eyeball ratio
- Foot in the door for mobile devices
- Invest in your base template
- Core technique: CSS Media Queries (CSS3)
- Tools for testing
- Responsive Frameworks
- Responsive Component Libraries

```
@media screen and (max-width: 400px) {
   .site-header {
      padding: 0.5em 0 0.5em 0.5em;
   }

   .left-view {
      padding-left: 5px;
      padding-right: 5px;
   }
}
```

Angular Material



- Material Design components for Angular
- Variety of responsive components
- Implement common interaction pattern
  - Consistent API
- Component categories
  - Form controls
  - Navigation
  - Layout
  - Buttons & Indicators
  - Popups and Modals
  - Data table
- https://material.angular.io/components/categories

RxJS and Observables

- RxJS (Reactive extensions for JavaScript)
  - library for reactive programming using observables
  - Included with Angular installation using CLI
- Both Promises and Observables allow us to deal with abstractions to deal with asynchronous code
  - Promises execute once and then are done.
  - Observable continue to be observed after the event occurs
  - Observables are easier to compose async or callback-based code
- An Observable is wrapper around a stream of data
- RxJS provides Observable implementations which are required until browsers natively support it
- RxJS provides a slew of utility functions for working with observable

Unit Testing

- Integral part of our front-end code development
- Angular CLI configuration
  - Jasmine unit testing framework
  - Karma test runner
- Flexibility to use other frameworks and runners
- Angular Test Bed
- Code coverage metrics and reporting

Builds & Configurations

- angular.json is the window into builds and configurations
- Façade over the lower level Webpack
- Create environment specific configurations
  - 2 default configurations
    - Development
    - Production

Useful References

- GitHub Repo
  - https://github.com/AllenConway/AngularMMApp
- Angular docs
  - https://angular.io/docs
- Angular CLI
  - https://angular.io/cli
- TypeScript docs
  - https://www.typescriptlang.org/docs

Thank you! Q&A

http://c1.staticflickr.com/1/28/65098350_b7bd96f38_b.jpg

Allen Conway

- **Blog**: http://www.AllenConway.net
- **Twitter**: @AllenConway
- **GitHub**: https://github.com/AllenConway
- **Email**: Allen.Conway@Cognizant.com

Thank you for attending Visual Studio LIVE! Las Vegas