Angular Training Notes

[Tooling in Angular 1](#_Toc92017284)

[CLI Capabilities 1](#_Toc92017285)

[Server-side rendering 2](#_Toc92017286)

[Benefits: Server-side rendering 2](#_Toc92017287)

[Testing Tools 3](#_Toc92017288)

[Package.json 4](#_Toc92017289)

[Angular Forms 4](#_Toc92017290)

[Forms properties 4](#_Toc92017291)

[Angular Forms Agenda 4](#_Toc92017292)

[Angular Reactive Forms 4](#_Toc92017293)

[Forms basics 5](#_Toc92017294)

[Data Binding 5](#_Toc92017295)

[Form Validations 5](#_Toc92017296)

[HTTP Posting 5](#_Toc92017297)

[Third-party Controls 5](#_Toc92017298)

[Template-driven Forms 6](#_Toc92017299)

# Tooling in Angular

## CLI Capabilities

* Create new applications with correct set-up
* Make new component, services, pipes & other pieces of application
* **Serving up the Application:** Act as the web server development
* **Linting:** lint the code
* **Testing:** Test the code using Unit Test & Cypress using e2e tests
* **Building:** build the code for production

# Server-side rendering

* It’s the simple concept but implementation is complex

## Benefits: Server-side rendering

* Performance
  + Initial download size
  + Reduces the Render time
* **Search Engine Optimization**
* **Mode of Operations**
  + Full Pre-Render
    - Involves running a development time process that will take site & create HTML for each of the views
    - Then it takes an HTML, load it onto a CDN. When people hit the site, they get fast, quick initial responses
    - It will make the switch on the client side to regular, angular app
  + **Dynamic Pre-Render**
    - When request received by a Web server, the requested page which built by Angular Universal and the and rendered HTML & CSS is sent to the browser, and then the switch happens
    - Currently Node and ASP.NET Core are supported in this mode
    - Once the browser has the pre-rendered HTML, Universal kicks in, downloads your full Angular App, including Angular Framework, then in a hidden div, its boots the app, renders the current page, replays events, which done by the user to maintain the current state and then swap the display inside of the hidden div to the main page

# Testing Tools

Angular team built **Two Testing Tools**

* **Karma**
  + Is unit testing tool
  + It’s installed locally and will run unit tests against multiple browsers
  + Tt’s very fast and convenient, and popular
* **Protractor**
  + Its web automation testing tool
  + Used to running e2e (end-to-end tests
  + It’s beneficial because rather than just testing a single piece of code, we can test entire application is working
* **Alternatives**
  + Jest
  + Intern
  + Cypress.io
* **Angular Testing Utilities**
  + **TestBed**
    - Helps for to construct components in a live environment
    - Gives you handles to wrappers around the component and the DOM of it’s template
  + **Async and fakeAsync utiliies**

# Package.json

**Dependencies**

* Packages required for development & deployment

**devDependencies**

* Packages only required for development

# Angular Forms

## Forms properties

1. ng-touched [touched]
2. ng-untouched [untouched]
3. ng-pristine [pristine]
4. ng-dirty [dirty]
5. ng-valid [valid]
6. ng-invalid [invalid]

## Angular Forms Agenda

[To create functional and great looking forms for Angular Application]

1. Building a form from scratch
2. Data Binding
3. Validating our form data
4. Sending a form data to a server
5. Third-party form controls

# Angular Reactive Forms

2-types of Angular form Technologies

* **Template-Driven Forms**
  + Use a component’s Template
  + Unit Test Against DOM
* **Reactive Forms**
  + Use a component’s Template
  + Create a Form Model in TypeScript [Must by in sync with the template]
  + Unit Test Against Form Model
  + Validation in Form Model

# Forms basics

* Create simple user settings form with wide verity of controls i.e. checkboxes, radio buttons, Textbox, submit button, Drop Down

# Data Binding

* Store data somewhere in Angular. We store in regular JavaScript object or TS class using Interface
* Interface gives types information about different fields on the form & store data in JavaScript object
* We are using 2-way data binding to object

# Form Validations

* HTML 5 Attributes
  + required
  + pattern
  + minlength
  + maxlength
  + min
  + max

# HTTP Posting

* Form data is posted to a server & store in sort of data store such as database
* Here we are validating the data, which we need to posting properly to the server is an asynchronous event, for that we need to work with RXJS observables in handling the posting and showing error when needed

# Third-party Controls

* Do with the forms
* Quickly look at **PrimeNG**, a good source of form controls
* **Angular material**, which has a set of form controls,
* **ngx-bootstrap**

# Template-driven Forms

* **Template-Driven Forms**
  + Use a component’s Template
  + Unit Test Against DOM
  + Things done quickly if we use HTML-5 type forms Angular Process using Template Driven form
* **Reactive Forms**
  + Use a component’s Template
  + Create a Form Model in TypeScript [Must by in sync with the template]
  + Unit Test Against Form Model
  + Validation in Form Model
  + Need to include form model, if we change template, then need to change form-model and we need to keep two in Sync, which cause extra work
  + As big benefit is that we can write Unit Test Against form model, which create very fast Unit Test
  + For Unit Testing, Reactive form is the better choice