ANGULAR 2

One framework for mobile and desktop Apps

About Me

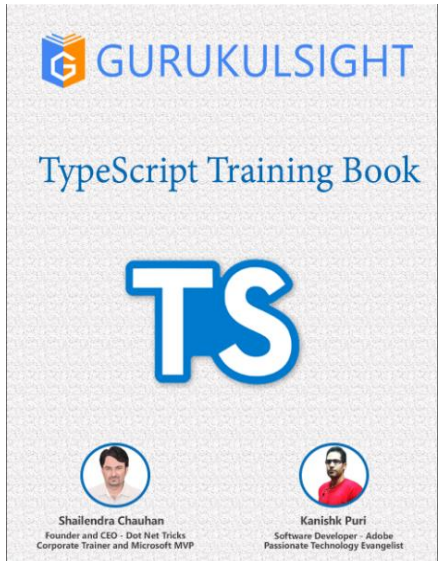
Hi, I'm Shailendra Chauhan

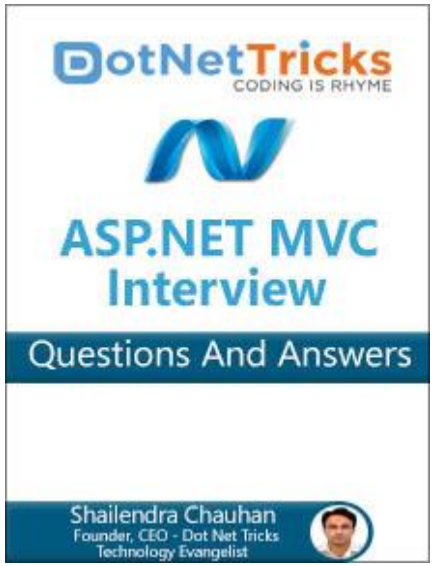
- Author
- Architect,
- Corporate Trainer
- Microsoft MVP
- Founder and CEO of Dot Net Tricks

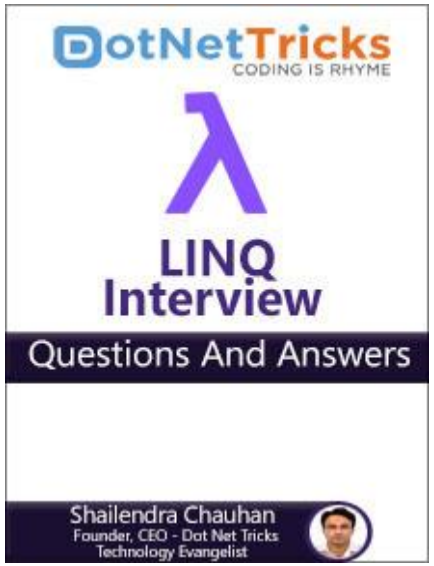
@proshailendra

@proshailendrachauhan

Author of Most Popular Free e-Books





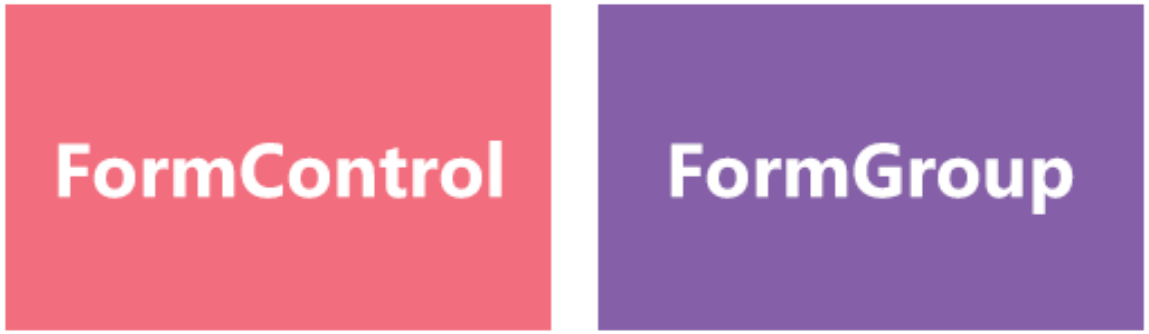


Agenda

- Angular Forms
- Angular Forms Built-In Validation
- Angular Forms and Form Fields States
- Angular Forms (Template-driven vs. Model-driven)
- Services
- Dependency Injection
- RxJs - Reactive Extensions Library
- Observables vs Promises
- Change Detection – Zone.js

Angular Forms

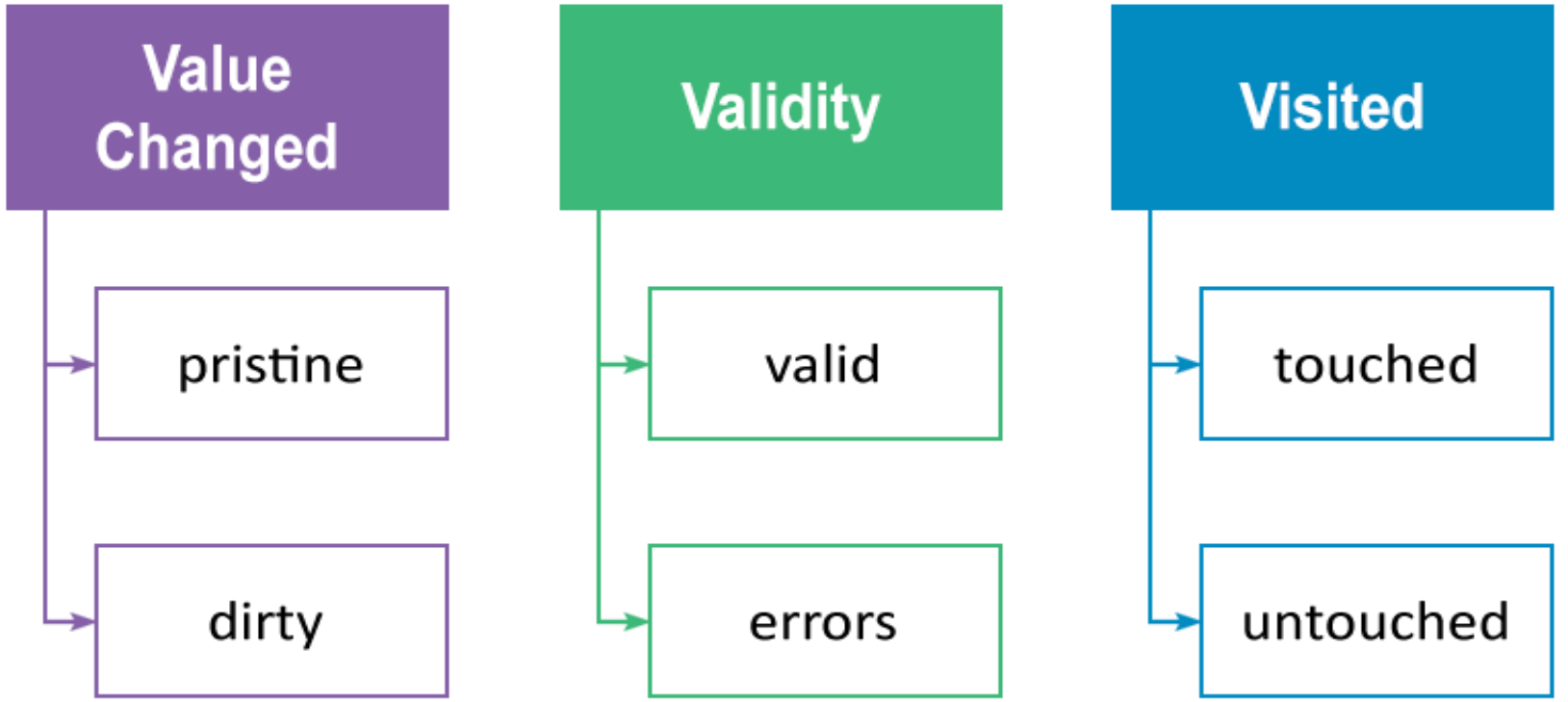
- Match the value of Forms model to data model
- Check form states – dirty, touched, valid
- Watch for changes and react
- Perform field validation



Angular Forms Built-In Validation

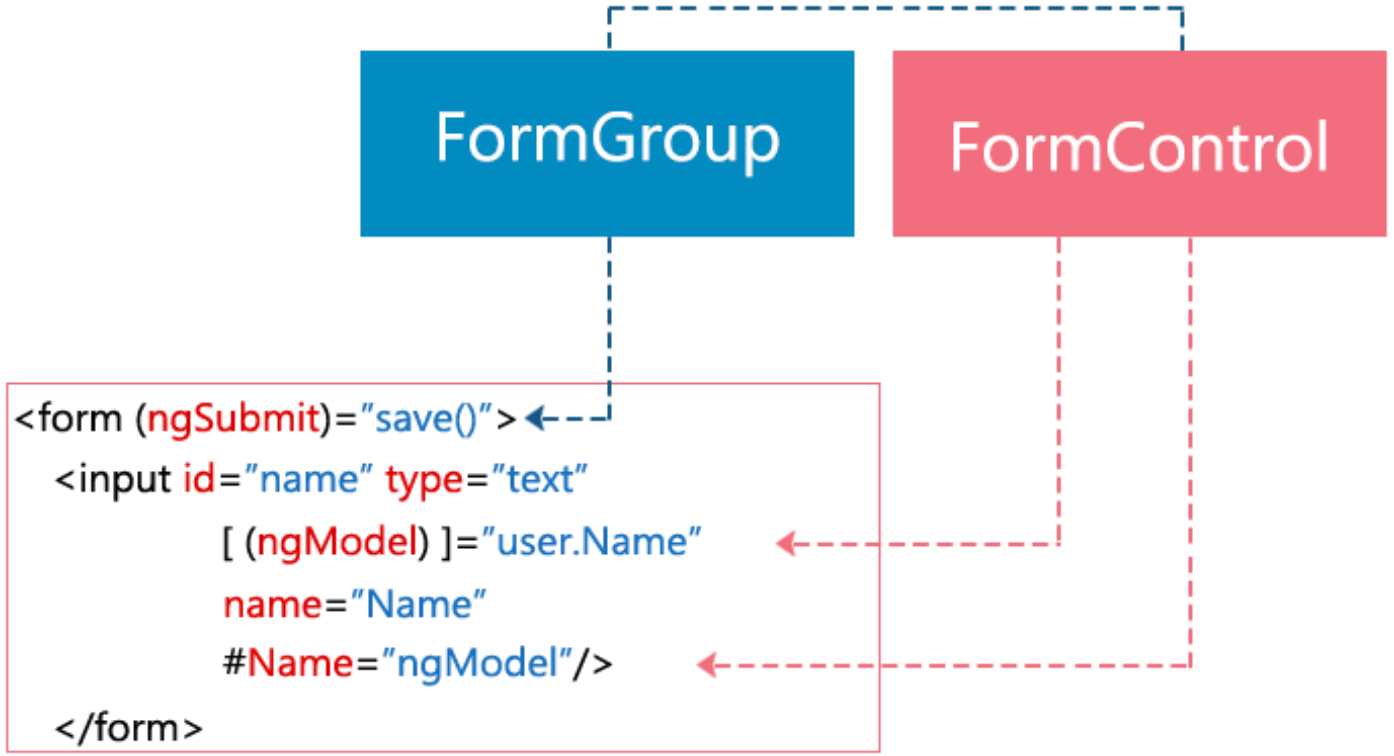
- required
- minlength
- maxlength
- pattern
- email
- min
- max

Angular Forms and Form Fields States



Template Driven Forms

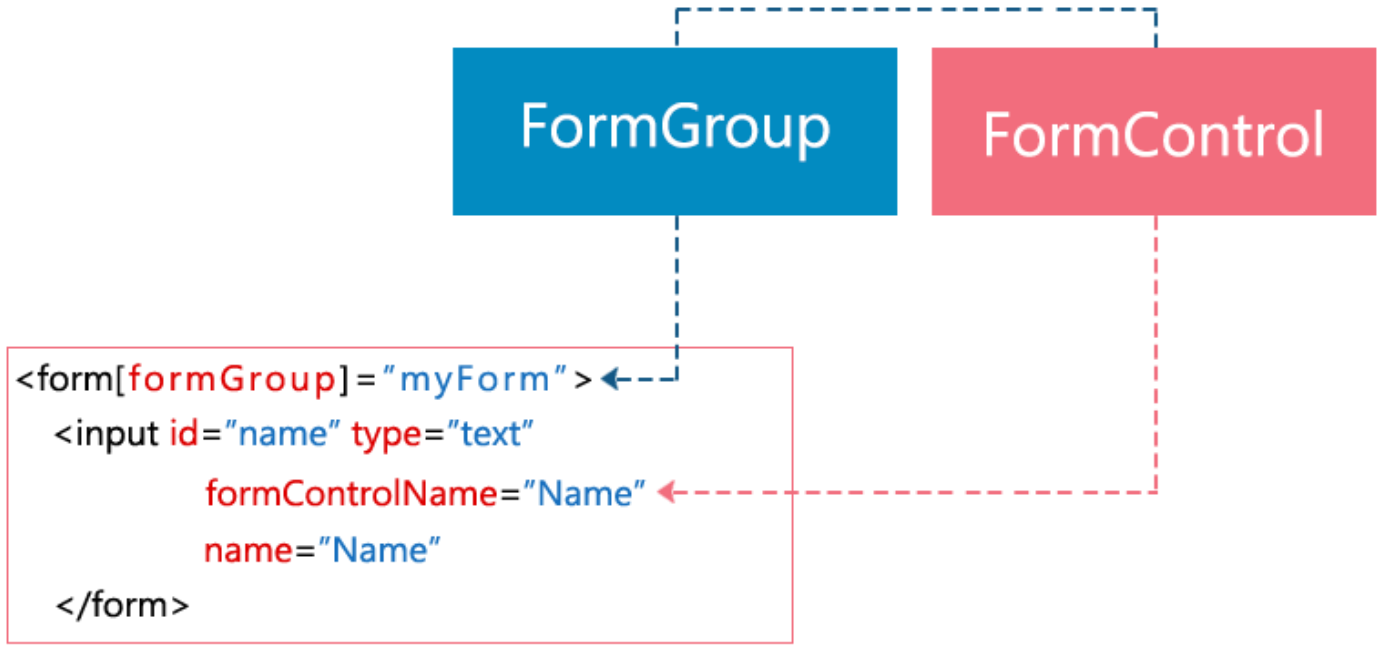
- ngModel
- ngFrom
- ngSubmit
- #refName



Model Driven (Reactive) Forms

- formGroup
- formControlName

- FormGroup
- FormBuilder
- Validators



Angular Forms (Template-driven vs. Model-driven)

- Form is setup and configured in HTML Code
- Easy to use and similar to Angular 1
- Two-way data binding
- Automatically track form and input element state
- Form data is passed via ngSubmit()
- Complex Unit Testing

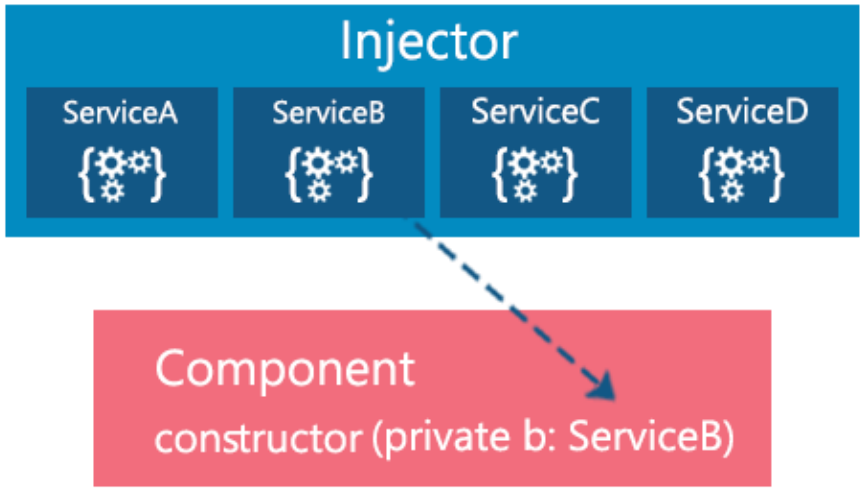
- Form is setup and configured in Component Class using FormBuilder
- Flexible & suitable for complex scenarios
- Immutable Data Model
- Reactive Transformation
- Form data can be used through class without passing it via ngSubmit()
- Easier Unit Testing

Services

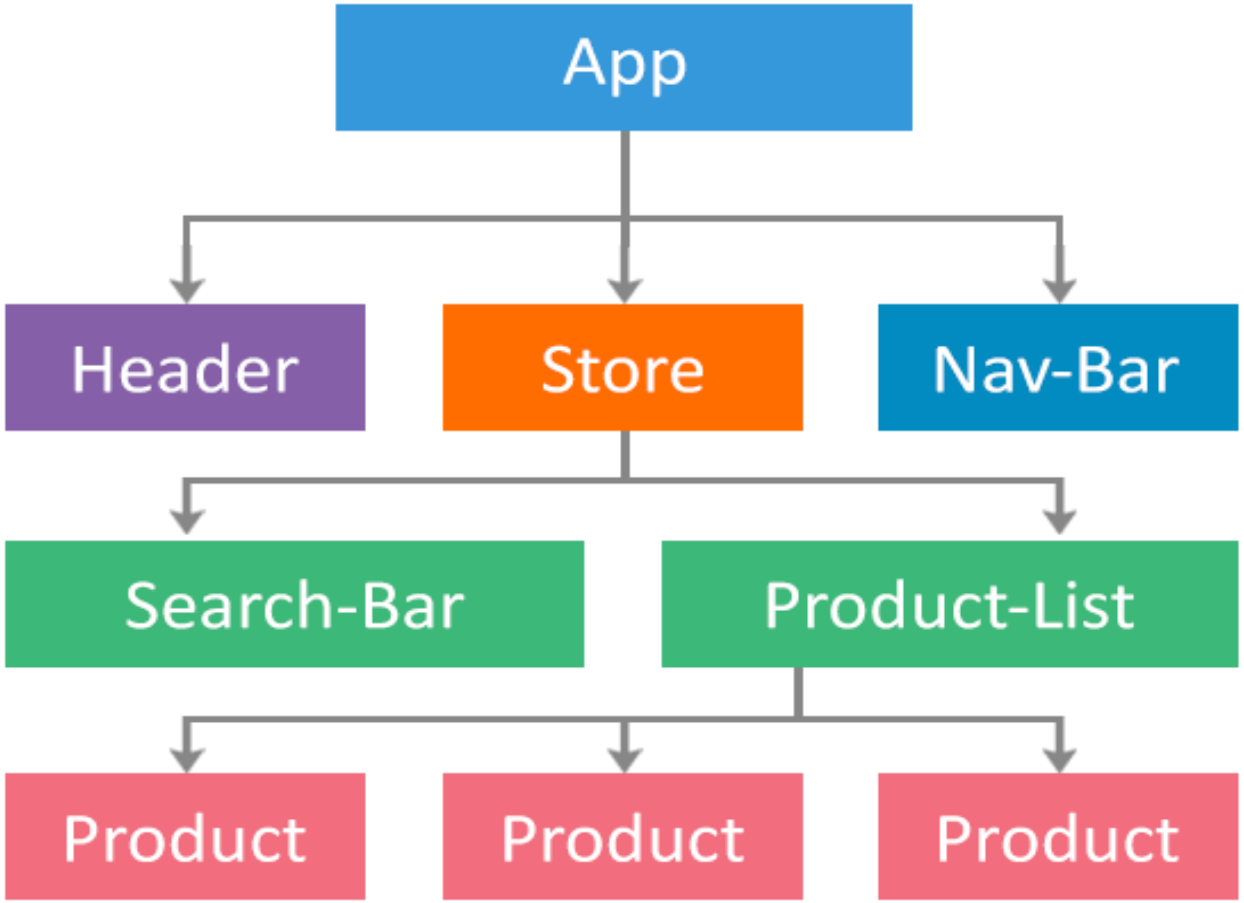
- A service is used to share functionalities among components
- A services is injected using Dependency Injection mechanism
- For example:
  - Logging service
  - Data service
  - Application configuration

Dependency Injection

- A way to supply a new instance of a class with the fully-formed dependencies it requires
- Most commonly used dependencies are services
- Angular uses dependency injection to provide new components with the services they need
- Ng2 has a Hierarchical Dependency Injection system



Dependency Injection



RxJs - Reactive Extensions Library

- Angular comes with RxJS library to deal with async data streams
- Makes easier to compose asynchronous or callback-based code
- A observable function emits the data over time to a subscriber
- Supports multiple operators to transform the stream's data
- Angular uses Observable with HTTP service and event system

Observables (RxJs)

- An Observable object (producer) can :
  - Stream elements
  - Throw an error
  - Send a signal that the streaming is over
- The data streaming starts when you invokes subscribe() on the observable

Observers (RxJs)

- An Observer object (consumer) provides :
  - A function to handle the next object from the stream
  - A function for error handling
  - A function for end-of-stream handling

Observables Operator (RxJs)

- Transforms an observable object into another observable object
- Observable operators are map, filter etc.

Observables vs Promises

- Observables are cancellable while Promises not
- Observables can be retried by using retry and retryWhen operator. On the other hand, Promises require the caller to have access to the original function that returned the promise in order to have a retry

Change Detection – Zone.js

- Zone.js monitors all async events
- Every component has its own change detector
- Change Detection is unidirectional and makes a single pass
- The changes can come only from a component
- Change detection runs from top to bottom of the component tree
- Angular 2 supports two change detection strategies :
  - Default - The change detector's mode will be set to CheckAlways during hydration
  - OnPush - The change detector's mode will be set to CheckOnce during hydration

Q&A


THANK
YOU!