Introduction to Angular

Shailendra Chauhan

Microsoft MVP, Technical Consultant and Corporate Trainer

DotNetTricks

Introduction to Angular

- A framework for building application using web technologies like html, css and js
- Empowers developers to build applications for browsers, mobiles, or desktop

Angular History

- Developed in 2009 by Misko Hevery and Adam Abrons at Brat Tech
- Misko Hevery started to work for Google in 2009
- Angular version 1.0 (AngularJS) was released in 2012 by Google
- Angular version 2.0 was released in September 2016
- Angular 4.0 was released in March 2017
- Angular 5.0 was released in Nov 2017
- Angular 6.0 was released in May 2018
- Angular 7.0 was released in Oct 2018
- Angular 8.0 was released in May 2019
- Angular 9.0 was released in Feb 2020

Angular CLI

- A powerful to create, build, compile and serve Angular2 App
- Used to generate new components, routes, services and pipes
- Installing Angular CLI
  - *npm install -g @angular/cli*
- Generating and serving Angular app
  - *ng new proj_name --skip-install*
  - *cd proj_name*
  - *npm install*
  - *ng serve*

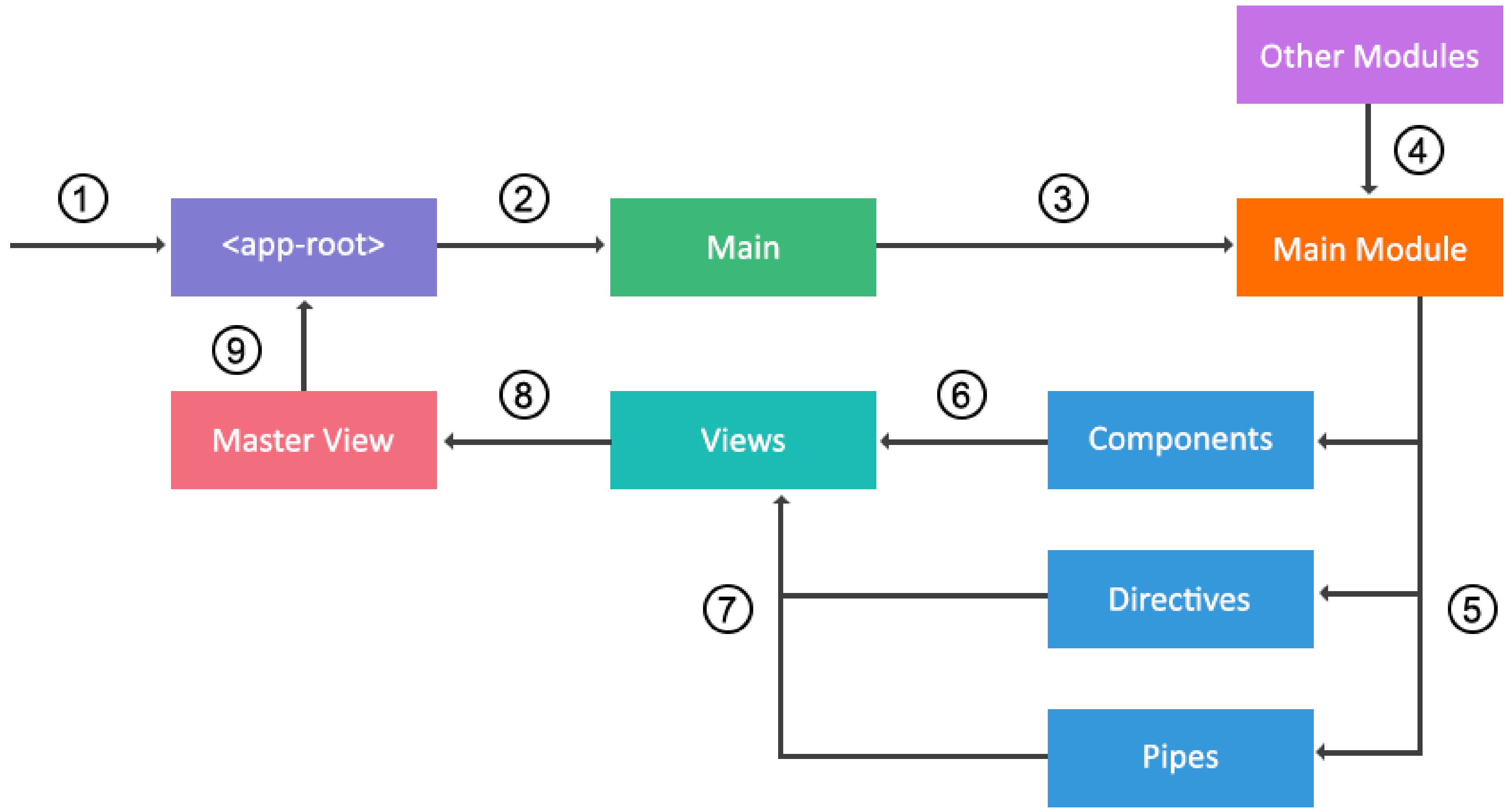
Angular CLI Options

Options	Usage
Help	ng --help
Build	ng build --*env*
Build and Run	ng serve
Testing	ng test
End-End Testing	ng e2e

Angular CLI Commands

Scaffold	Usage	In Short
Module	ng generate module *my-module*	ng g m *my-module*
Component	ng generate component *my-component*	ng g c *my-component*
Directive	ng generate directive *my-directive*	ng g d *my-directive*
Pipe	ng generate pipe *my-pipe*	ng g p *my-pipe*
Service	ng generate service *my-service*	ng g s *my-service*
Guard	ng generate guard *my-guard*	ng g g *my-guard*
Class	ng generate class *my-class*	ng g cl *my-class*
Interface	ng generate interface *my-interface*	ng g i *my-interface*
Enum	ng generate enum *my-enum*	ng g e *my-enum*

Angular Initialization Process



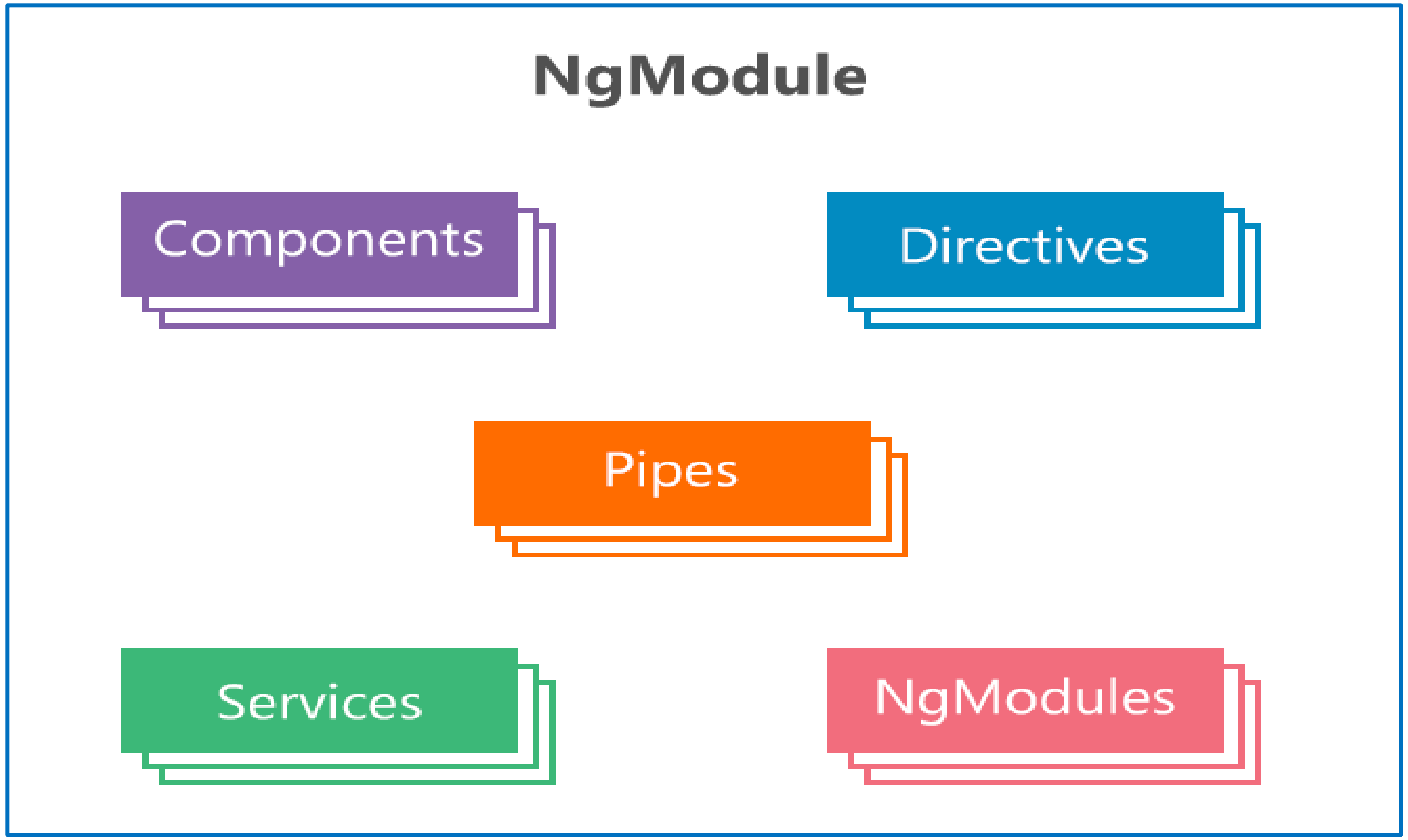
Angular Building Blocks

- Modules
- Components
- Templates
- Metadata
- Data binding
- Directives
- Pipes
- Routing
- Forms
- Services
- Dependency injection

Modules

- A module organize an application into unified blocks of functionality
- An Angular module is a class with an *@NgModule* decorator
- Accepts a single metadata object whose properties describe the module
- Each Angular app must have at least one module, known as root module

Modules


NgModule
Components
Directives
Pipes
Services
NgModules

NgModule Metadata Main Properties

- **imports** – Specify other dependent modules whose classes are required by the component templates declared in the module
- **declarations** – Specify the components, directives, and pipes that belong to the module
- **bootstrap** – Specify the main app view i.e root component. Only the **root module** can have this bootstrap property
- **exports** – A subset of declarations that will be visible and usable in the other modules. A root module doesn’t have export option.
- **providers** – Specify the services, accessible across the app

Built-In Modules

- Angular has built-In library modules starting with the @angular as prefix

@angular/core

@angular/router

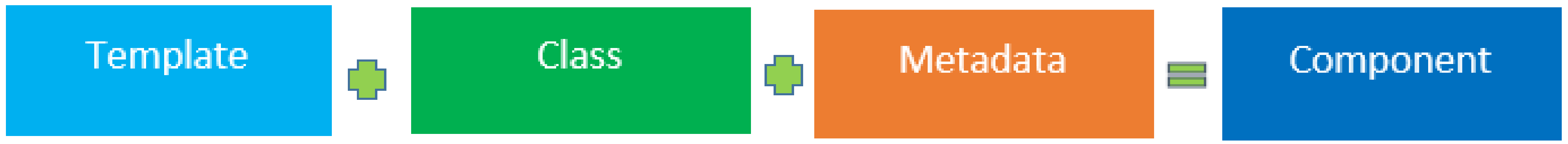
@angular/forms

@angular/http

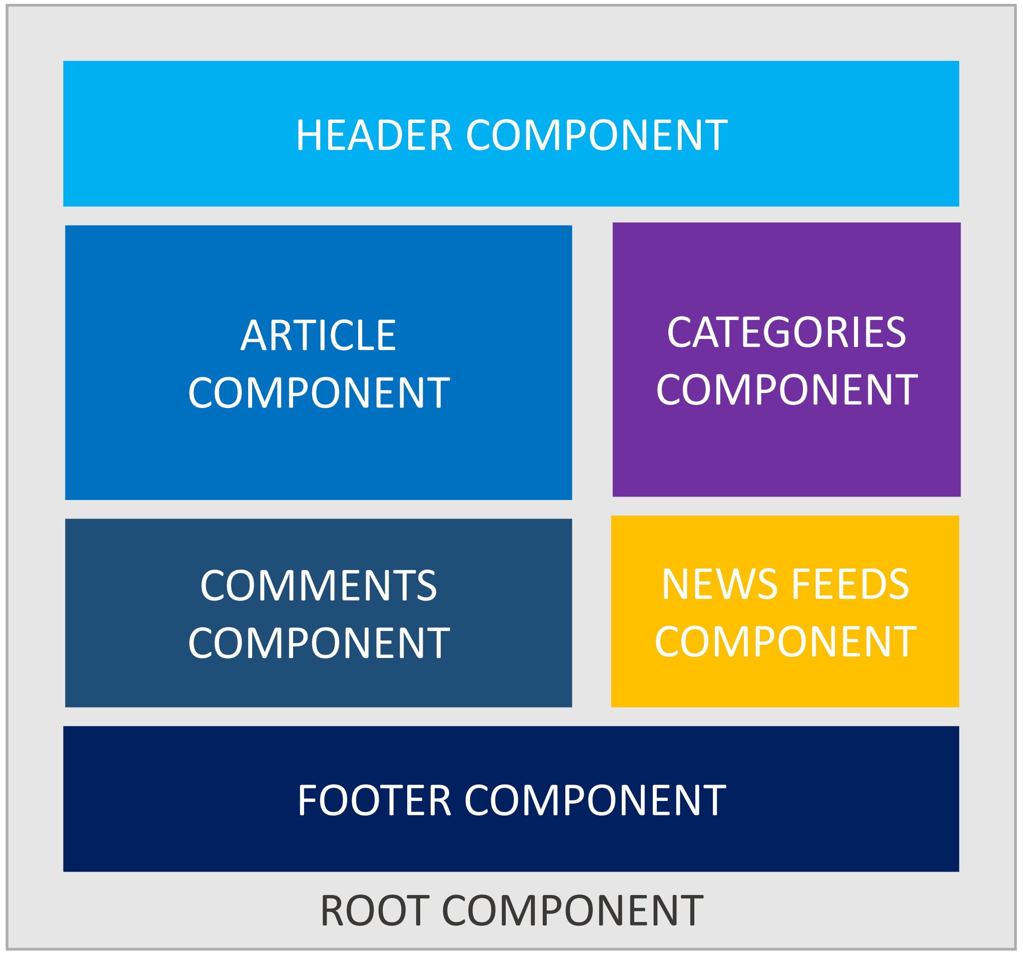
- Built-In library & third part modules can be installed using npm manager
- Built-In modules, components, services, directives etc. can be imported by using built-In library modules

Component

- A type of directives with template, styles and logic for user interaction
- Exported as a custom HTML tag like as:
  - <my-component></my-component>
- Initialized by Angular Dependency Injection engine



Angular Components Page View



Component Example

```
import { Component} from '@angular/core';

@Component({
  selector: 'my-component',
  template: `<h3>Interpolation</h3>
             <p>Name : {{name}}</p>
             <p><input type="text" value="{{name}}" /></p>`,
  styles: []
})
export class MyComponent {
  name: string = 'Shailendra';
  constructor() { }
}
```

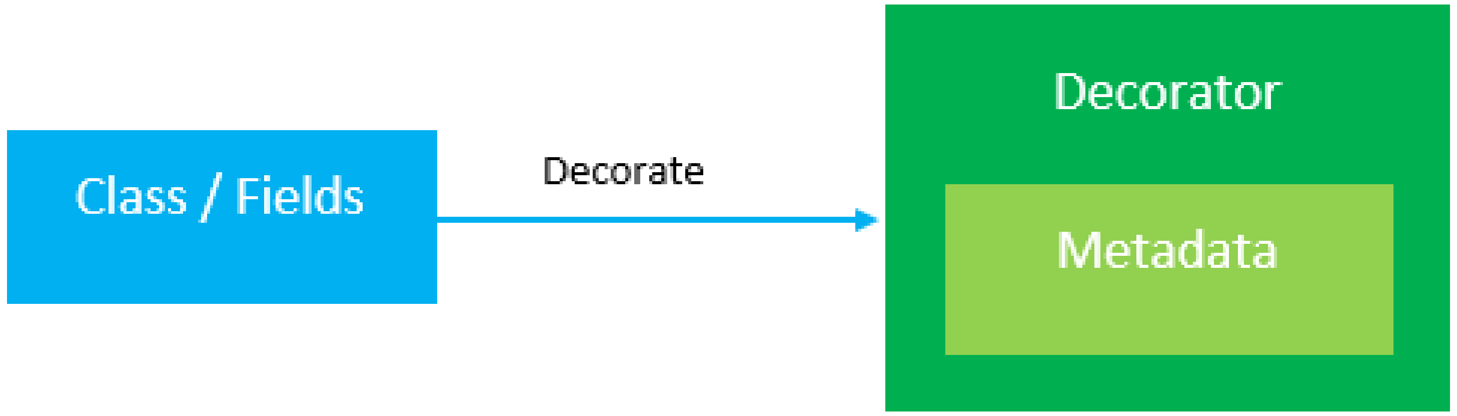
Template

- Define the view of a component
- Contains Html markup and angular directives, attributes etc.
- Describe how a component is rendered on the page

```
<h3>Interpolation</h3>
<p>Name : {{name}}</p>
<p>
  <input type="text" value="{{name}}" />
</p>
```

Decorators

- A function that adds metadata to a class, class members
- These are prefix with @ symbol
- Angular has built-In decorators like - @Component, @NgModule, @Directive, @Pipe etc.

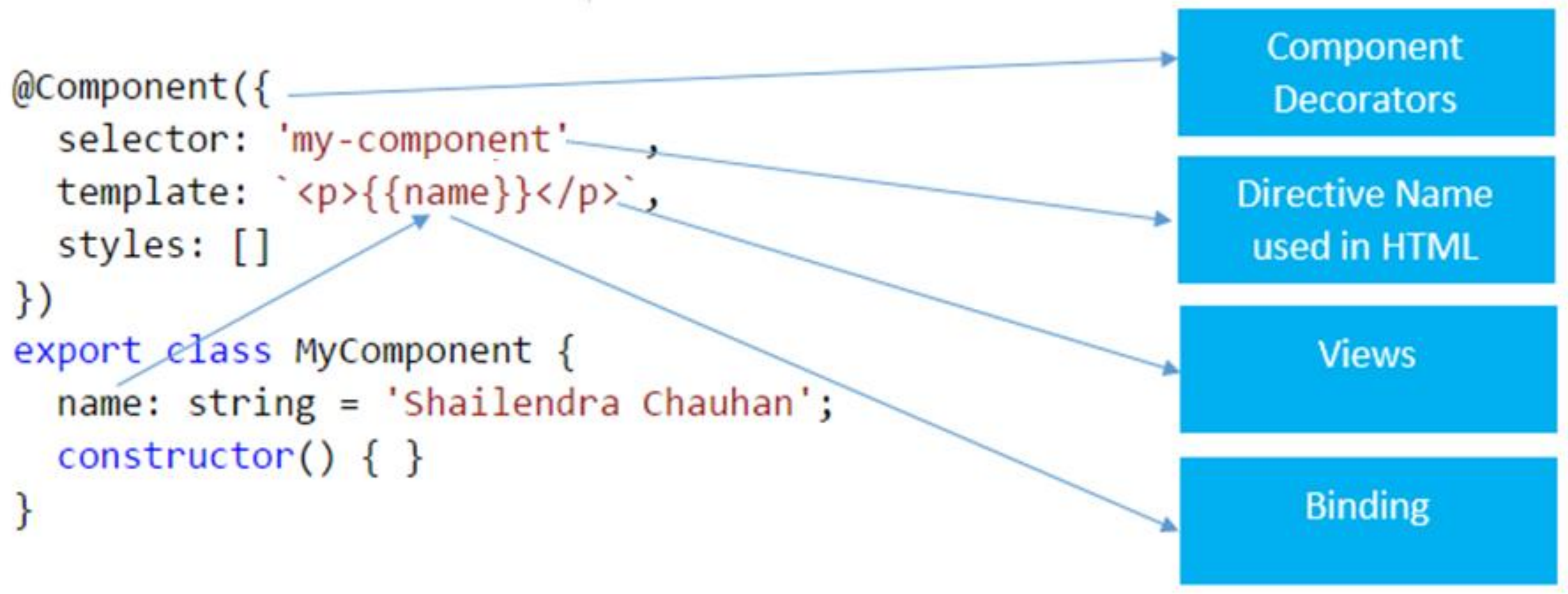


Types of Decorators

- Class decorators
  - @NgModule – Used for defining a module
  - @Component – Used for defining a component
  - @Directive – Used for defining a directive
  - @Injectable – Used for injecting dependencies
  - @Pipe – Used for defining a pipe
- Class field decorators
  - @Input – Used for receiving data (input) from parent to child component
  - @Output – Used for passing data (events) from child to parent component

Metadata

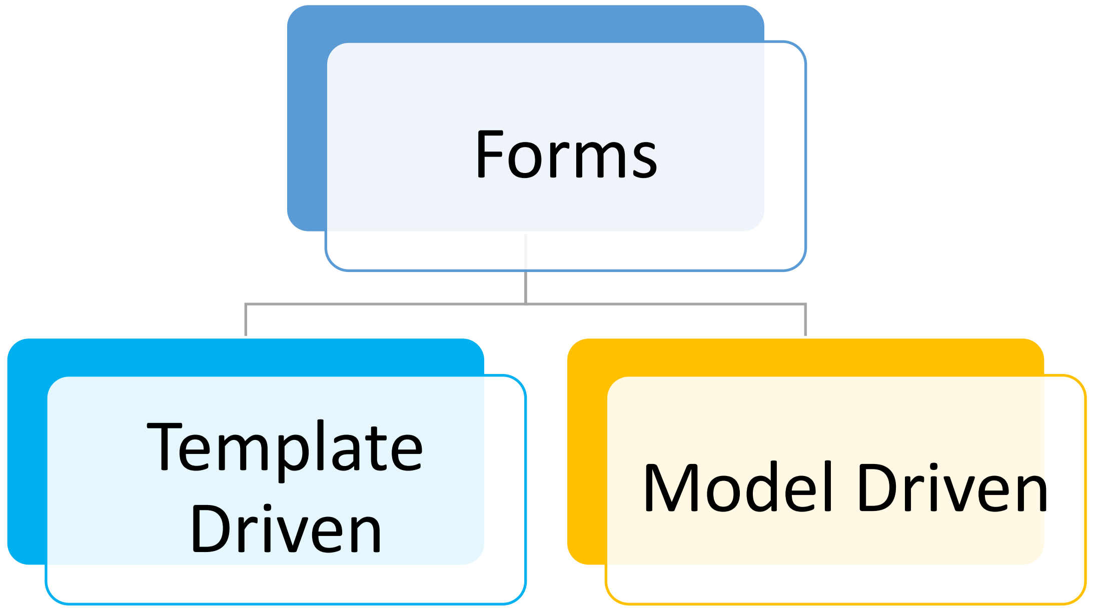
- Tells Angular how to process a class
- Decorators are used to attach metadata to a class



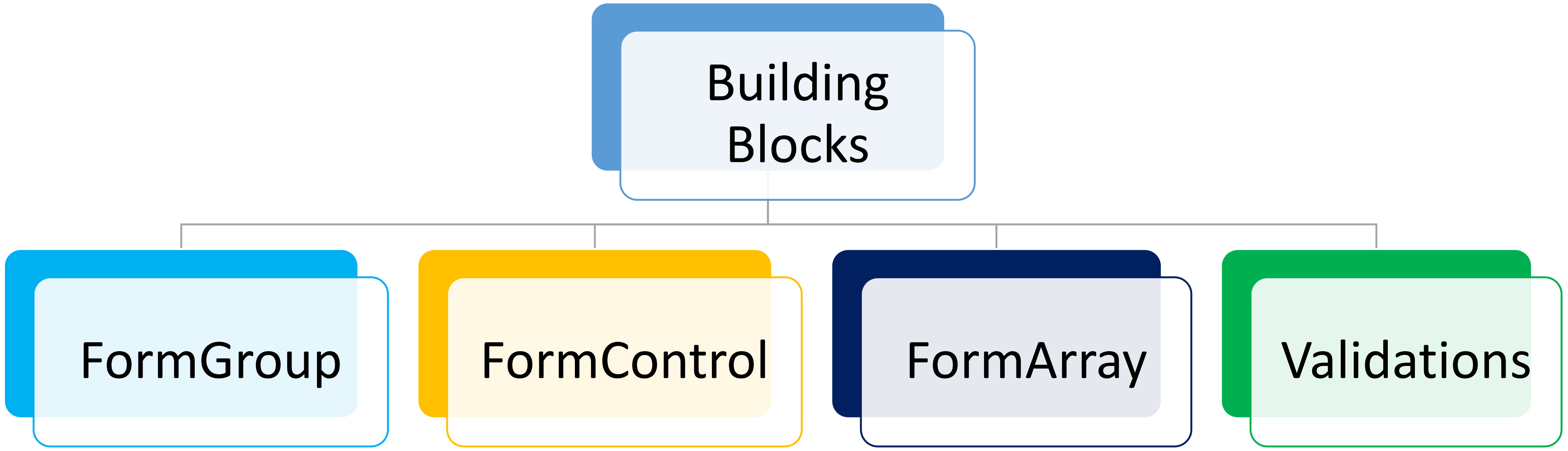
```
@Component({
  selector: 'my-component',
  template: `<p>{{name}}</p>`,
  styles: []
})
export class MyComponent {
  name: string = 'Shailendra Chauhan';
  constructor() { }
}
```

Angular Forms

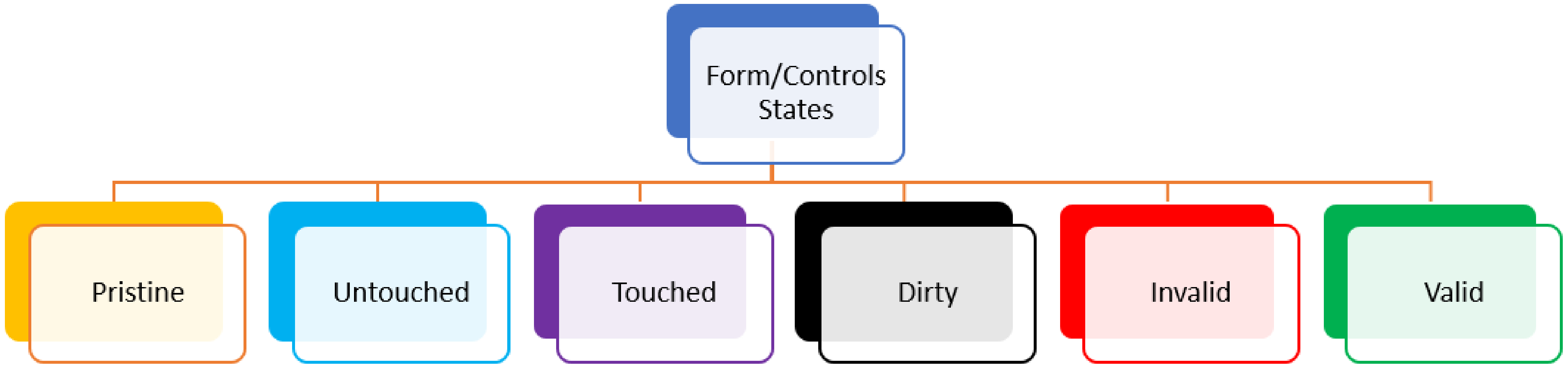
- HTML forms are an essential part of a web application
- Angular provides two ways to create form – Template Driven and Model Driven



Angular Form Building Blocks



Angular Form and Form Controls States


Form/Controls States
Pristine
Untouched
Touched
Dirty
Invalid
Valid