ANGULAR 2

One framework for web, mobile and desktop Apps

About Me

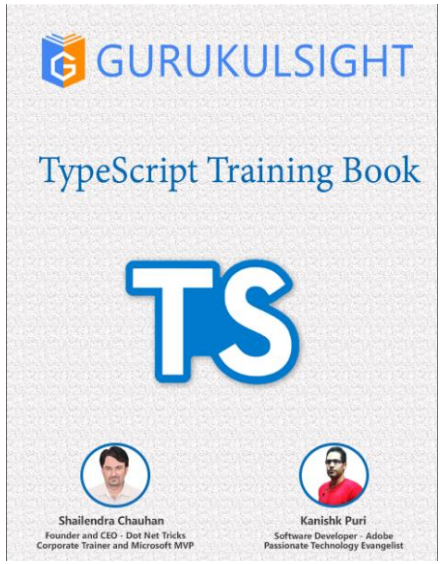
Hi, I'm Shailendra Chauhan

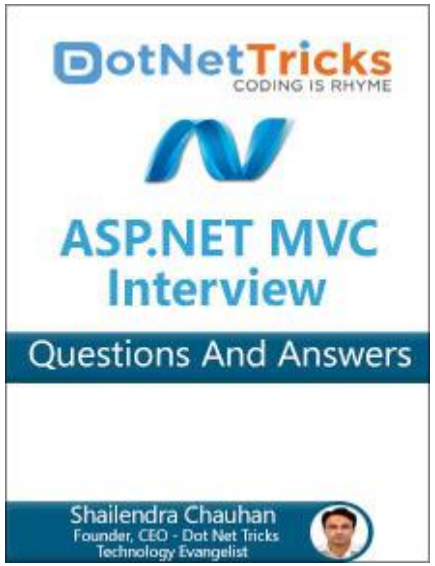
- Author
- Architect,
- Corporate Trainer
- Microsoft MVP
- Founder and CEO of Dot Net Tricks

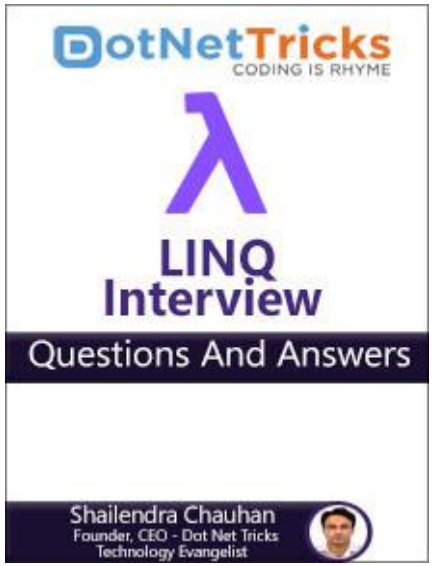
@proshailendra

@proshailendrachauhan

Author of Most Popular Free e-Books







Agenda

- Angular History
- Introduction to Angular2
- Angular2.x vs. Angular1.x
- Building Blocks of Angular 2
- Angular CLI
- Module
- Component
- Metadata
- Decorators
- Data Binding

Angular History

- AngularJS was originally developed in 2009 by Misko Hevery and Adam Abrons at Brat Tech
- Misko Hevery started to work for Google in 2009
- AngularJS version 1.0 was released in 2012 by Google
- Angular version 2 was released in September 2016 after 2 years development

Introduction to Angular2

- Modern, faster and highly scalable
- One framework for web, mobile and desktop apps
- Not an update to Angular1.x, but a complete rewrite
- Written in Typescript
- Simple and Expressive
- Web components based architecture
- Hierarchical Dependency Injection

Angular2.x vs. Angular1.x

- Based on Components
- Improved DI
- Supports web components
- Mobile first
- ES5, ES6, TypeScript or Dart
- Angular CLI
- Class is only way to define service
- Run on client-side & server-side

- Based on Controller, Scope
- Supports DI
- Doesn’t support web components
- Not built with Mobile first
- ES5, ES6 and Dart
- Doesn’t have CLI
- Service is defined by using factory, service, provider, value, constant
- Run on client-side only

Angular2.x vs. Angular1.x

- bootstrapModule function is used to initialize
- Supports Pipes
- Supports camelCase syntax ngModel, ngClass
- Uses HTML DOM element properties and events
- Uses () for events and [] for properties

- ng-app and angular.bootstrap function are used to initialize
- Supports Filters
- Supports spinal case syntax like ng-model, ng-class
- Uses its own directives like ng-click, ng-show, ng-src etc.
- Don’t supports () and [] based syntax for events and properties

Building Blocks of Angular 2

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

Angular CLI

- A powerful to create, build, compile and serve Angular2 App
- Used to generate new components, routes, services and pipes

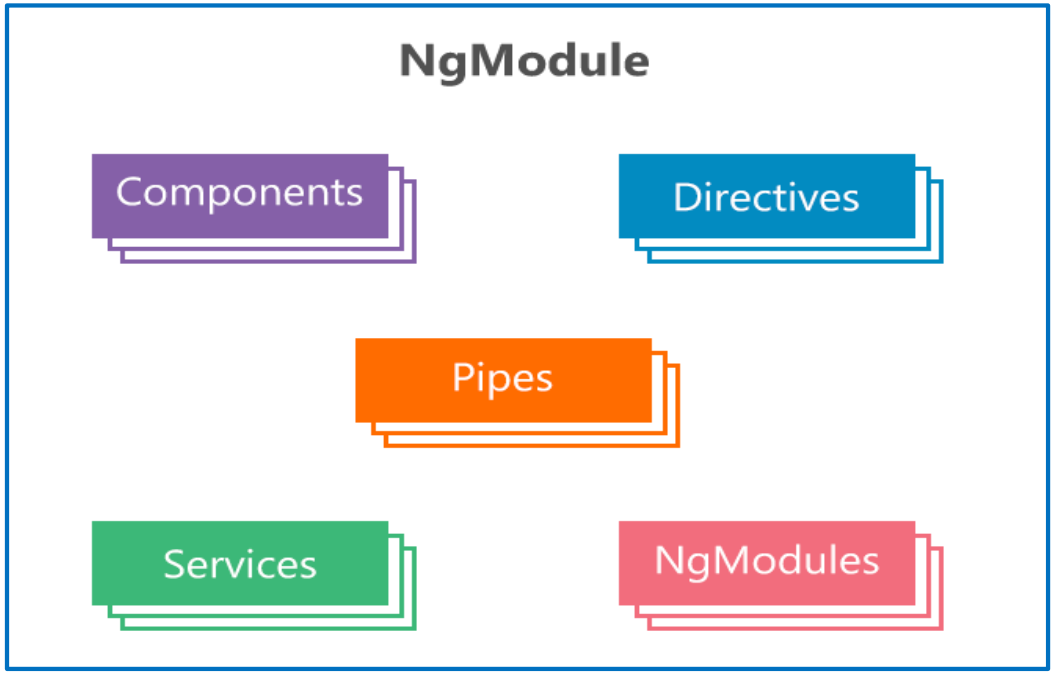
Generating and serving an Angular2 project

```
ng new PROJECT_NAME
cd PROJECT_NAME
ng serve
```

Scaffold	Usage
Component	`ng g component my-new-component`
Directive	`ng g directive my-new-directive`
Pipe	`ng g pipe my-new-pipe`
Service	`ng g service my-new-service`
Class	`ng g class my-new-class`
Interface	`ng g interface my-new-interface`
Enum	`ng g enum my-new-enum`
Module	`ng g module my-module`

Module

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



NgModule Decorator Main Properties

- **imports** – Specify the other dependent modules whose exported classes are required by component templates declared in the module
- **declarations** – Specify the view classes - 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 component templates of other modules. A root module has no reason to export anything because other components don't need to import the root module
- **providers** – Specify the main app services, accessible across the app

Built-In Modules

- Angular also has built-In library modules starting with the @angular as prefix. e.g. NgModule, RouterModule, FormsModule, HttpModule etc.
- Built-In library can be installed using npm manager

@angular/core

@angular/router

@angular/forms

@angular/http

Component

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


```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-mycomponent',
  template: `<p>{{name}}</p>`,
  styles: []
})
export class MyComponent {
  name: string = 'Shailendra Chauhan';
  constructor() { }
}
```

Controller to Component

```
<div ng-controller="myController">
    {{name}}
</div>
<script>
  angular
     .module("app",[])
     .controller("myController",function($scope){
          $scope.name="Shailendra";
     });
</script>
```

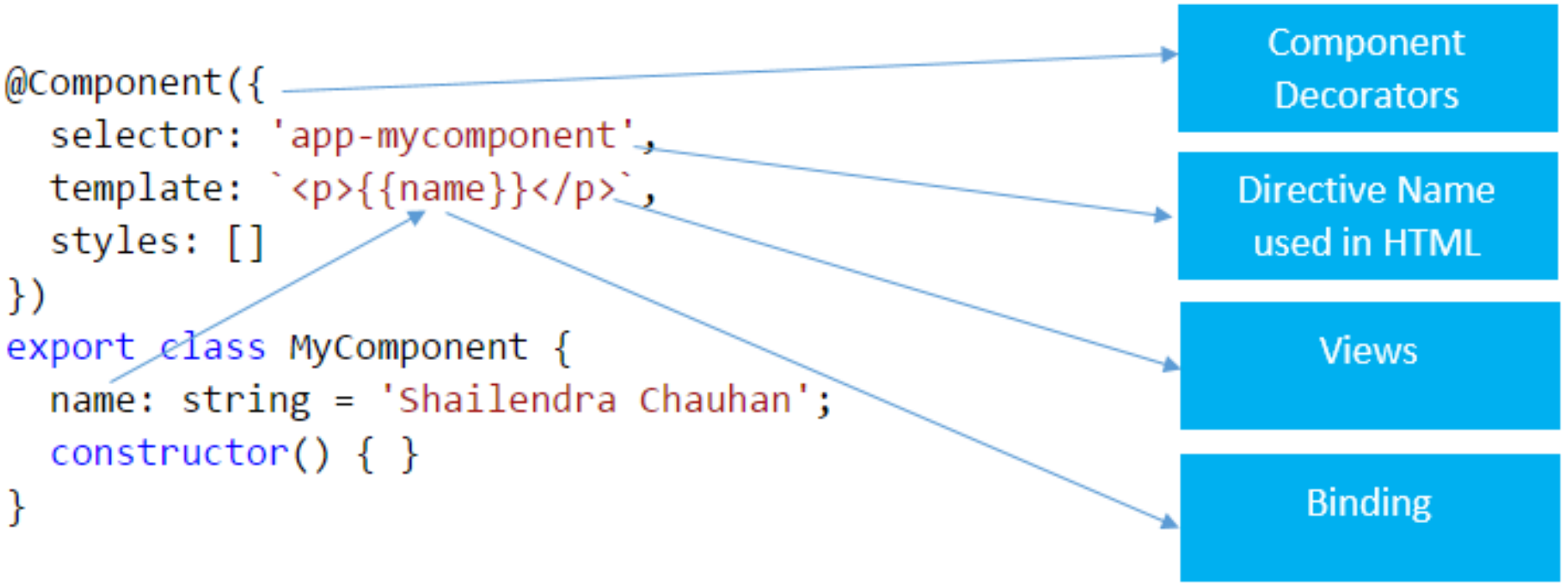
Angular1

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-mycomponent',
  template: `<p>{{name}}</p>`,
  styles: []
})
export class MyComponent {
  name: string = 'Shailendra Chauhan';
  constructor() { }
}
```

Angular2

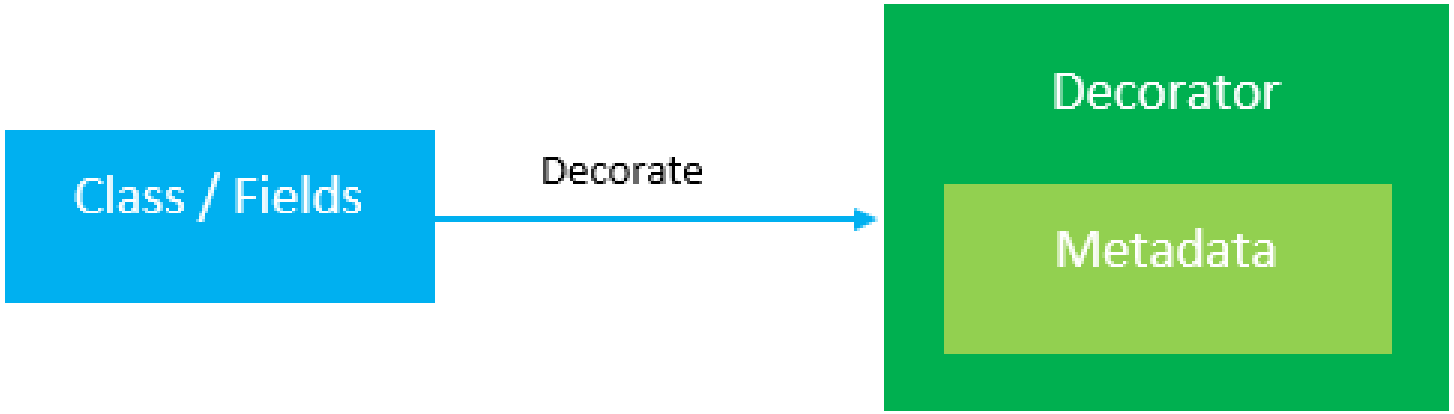
Metadata

- Tells Angular how to process a class
- Template, Metadata, and Component together specify a view



Decorators

- A function that adds metadata to a class, class members
- These are prefix with @ symbol
- Angular has built-In decorators like - @Component for defining components and @Injectable for injecting dependencies

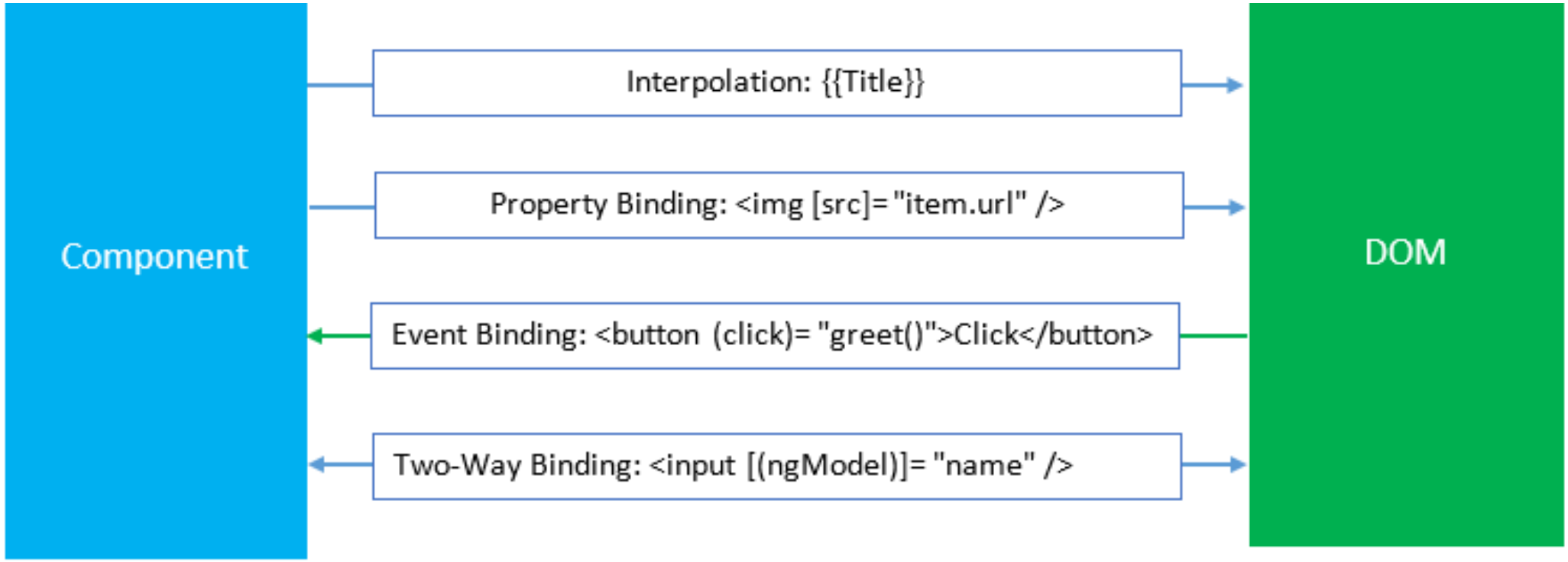


Types of Decorators

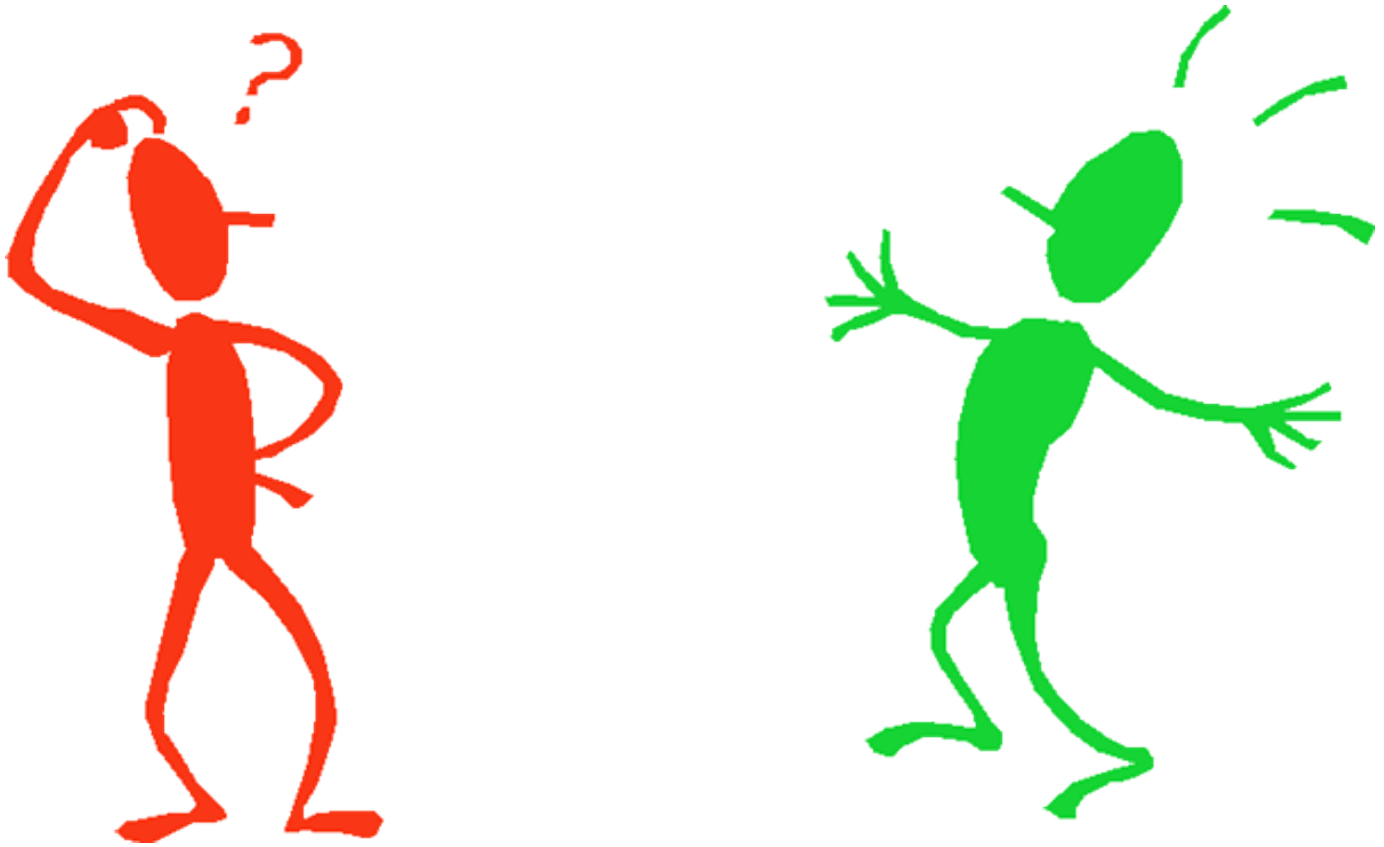
- Class decorators
  - @Component – Used for defining components
  - @Directive – Used for defining a directive
  - @Injectable – Used for injecting dependencies
  - @Pipe – Used for defining a pipe
- Class field decorators
  - @Input – Used for receiving data (input) from parent component to child component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

Data Binding

- A mechanism for binding data values to HTML elements and turning user activities into actions and data value updates



Q&A




THANK
YOU !