TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Angular

EXPERIENTIAL
learning factory

Overview of Angular

- Angular is a javascript framework for developing single page applications

- Angular can be built using ES6 or Typescript

- Typescript uses
  - ✓ Class-based object oriented programming.
  - ✓ Static Typing.

History

- Angular JS 1.x
  - Uses Javascript.
- Angular 2
  - Totally rewritten in typescript.
  - Stable version released to market in 2016
- Angular 4
  - Http Client was introduced.
  - Router Guards were proposed.
  - Package size reduced by 40%.
- Angular 5
  - Support for progressive web apps.
- Angular 6
- Angular 7
- Angular 8
  - @ViewChild is added.
  - Faster re-build time

Note:

Every 6 months Google releases a new version of Angular. From Angular 4 onwards there were no major changes but came up with stability improvements

AngularJS vs. Angular

- Angular is a completely revived component-based framework in which an application is a tree of individual components.
- AngularJS
  - It is based on MVC architecture
  - This uses use JavaScript to build the application
  - Not a mobile friendly framework
  - Difficulty in SEO friendly application development
- Angular
  - This is based on Service/Controller
  - Introduced the typescript to write the application
  - This is a component based UI approach
  - Developed considering mobile platform
  - Ease to create SEO friendly applications

Angular CLI

- Angular CLI(Command Line Interface) is a command line interface to build angular apps using nodejs modules.
- You need to install using below npm command,

  “npm install @angular/cli@latest”
- Command to create a new Project

  “ng new project-name”

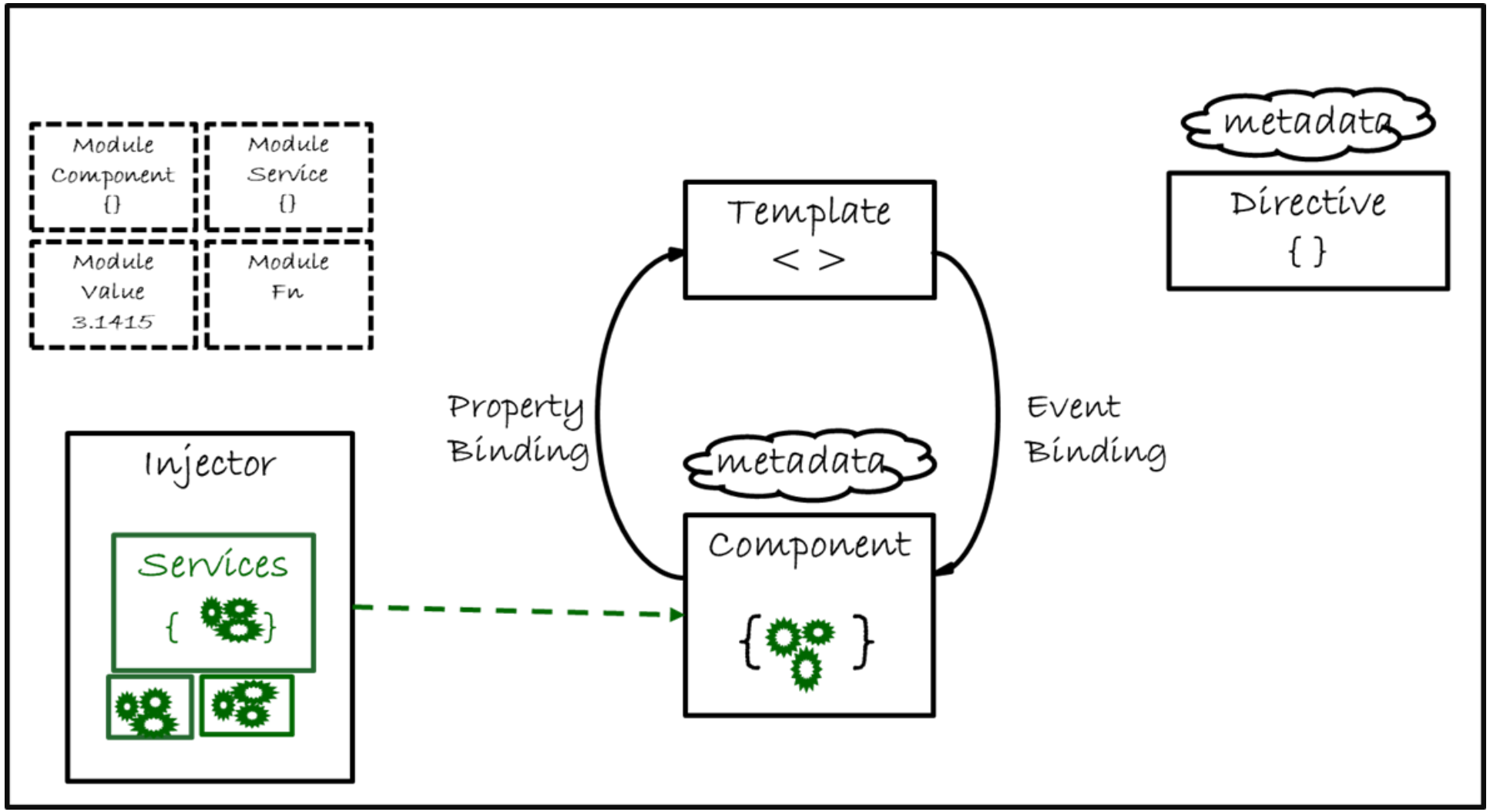
Angular Project Structure

- index.html : Entry point of angular app.
- main.ts : It is the global typescript file.
- styles.css : Global CSS file.
- Environments: Production and Development.
- package.json: Takes care of node related packages.
- polyfills.ts: For browser purpose.
- App: Root component.
- e2e: End to end testing purpose.

Key Components

- **Component:** These are the basic building blocks of angular application to control HTML views.
- **Modules:** An angular module is set of angular basic building blocks like component, directives, services etc. An application is divided into logical pieces and each piece of code is called as "module" which perform a single task.
- **Templates:** This represent the views of an Angular application.
- **Services:** It is used to create components which can be shared across the entire application.
- **Metadata:** This can be used to add more data to an Angular class.

Angular Architecture



Meta Data

- Metadata is used to decorate a class so that it can configure the expected behavior of the class.
- The metadata is represented by decorators.

1. Class decorators.

   e.g. @Component and @NgModule

2. Property decorators: Used for properties inside classes.

   e.g. @Input and @Output

3. Method decorators: Used for methods inside classes.

   e.g. @HostListener

4. Parameter decorators: Used for parameters inside class constructors.

   e.g. @Inject

Lifecycle Hooks

Angular application goes through an entire set of processes or has a lifecycle right from its initiation to the end of the application.

- **ngOnChanges:** When the value of a data bound property changes, then this method is called.
- **ngOnInit:** This is called whenever the initialization of the directive/component after Angular first displays the data-bound properties happens.
- **ngDoCheck:** This is for the detection to act on changes that Angular can't or won't detect on its own.
- **ngAfterContentInit:** This is called in response after Angular projects external content into the component's view.
- **ngAfterContentChecked:** This is called in response after Angular checks the content projected into the component.
- **ngAfterViewInit:** This is called in response after Angular initializes the component's views and child views.
- **ngAfterViewChecked:** This is called in response after Angular checks the component's views and child views.
- **ngOnDestroy:** This is the cleanup phase just before Angular destroys the directive/component.

Constructor vs. ngOnInit

- TypeScript classes has a default method called constructor which is normally used for the initialization purpose.
- Whereas ngOnInit method is specific to Angular, especially used to define Angular bindings.
- Even though constructor getting called first, it is preferred to move all of your Angular bindings to ngOnInit method.
- In order to use ngOnInit, you need to implement OnInit interface.

Interpolation

- When ever we need to communicate properties (variables, objects, arrays, etc.) from the component class to the template, we can make use of Interpolation.
- String Interpolation uses template expressions in double curly {{ }} braces to display data from the component, the special syntax {{ }}, also known as moustache syntax.
- The {{ }} contains JavaScript expression which can be run by Angular and the output will be inserted into the HTML.
- Say if we put {{ 5 + 5 }} in the template 10 will be inserted into the HTML.

Data Binding

Data binding is a core concept in Angular and allows to define communication between a component and the DOM, making it very easy to define interactive applications without worrying about pushing and pulling data.

- **One Way Data Binding**
  - Property Binding
  - Style Binding
  - Class Binding
  - Attribute Binding
  - Event Binding
- **Two Way Data Binding**
  - NgModel – Forms Module is required

One Way Data Binding

- **Property Binding**

  Property binding is the primary way of binding data in Angular. The square braces are used to bind data to a property of an element, the trick is to put the property onto the element wrapped in brackets: [property] .

- **Class Binding**

  The Angular Class binding is used to add or remove classes to and from the HTML elements. You can add CSS Classes conditionally to an element, hence creating a dynamically styled element.

- **Style Binding**

  Style binding is used to set a style of a view element. We can set inline styles with style binding. Like with class and attribute binding, style binding syntax is like property binding.

- **Attribute Binding**

  Attribute binding is useful where we don't have any property in DOM respected to an HTML element attribute.

  syntax: [attr.propertyname]=[propertyvalue]

View Encapsulation

View encapsulation defines whether the template and styles defined within the component can affect the whole application or vice versa. Angular provides three encapsulation strategies:

- Emulated (default) - styles from main HTML propagate to the component. Styles defined in this component's @Component decorator are scoped to this component only.
- Native - styles from main HTML do not propagate to the component. Styles defined in this component's @Component decorator are scoped to this component only.
- None - styles from the component propagate back to the main HTML and therefore are visible to all components on the page. Be careful with apps that have None and Native components in the application. All components with None encapsulation will have their styles duplicated in all components with Native encapsulation.

@Input and @Output

- The idea of Input and Output is to exchange data between components.
- They are a mechanism to send/receive data from one component to another.
- Input is used to receive data in whereas Output is used to send data out.
- Output sends data out by exposing event producers, usually EventEmitter objects.
- @Input

  To get data from parent to child.

  Makes use of property binding.
- @output

  To emit data from child to parent.

  This makes use of Event Emitters and event binding.

Directives

There are three kinds of directives in Angular:

- Component directives—directives with a template.
- Structural directives—change the DOM layout by adding and removing DOM elements.

  *ngFor

  *ngIf

  *ngSwitch
- Attribute directives—change the appearance or behavior of an element, component, or another directive.

  ngClass

  ngStyle

Forms

- Template Driven Forms
  - ngForm
  - ngModel
  - Cannot use Custom Validations
- Reactive Forms
  - FormGroup
  - FormControl
  - Custom Validations can be used

Observables

- Observables provide support for passing messages between publishers and subscribers in your application. Observables offer significant benefits over other techniques for event handling, asynchronous programming, and handling multiple values.
- Observables are declarative—that is, you define a function for publishing values, but it is not executed until a consumer subscribes to it. The subscribed consumer then receives notifications until the function completes, or until they unsubscribe.
- An Observable instance begins publishing values only when someone subscribes to it. You subscribe by calling the subscribe()method of the instance, passing an observer object to receive the notifications.
- Once a Observable is subscribed, it will be subscribing the data until it is unsubscribed. Usually unsubscribe() is called in the ngOnDestroy().

Services and Dependency Injection

- A service is used when a common functionality needs to be provided to various modules. Services allow for greater separation of concerns for your application and better modularity by allowing you to extract common functionality out of components.
- Dependency injection (DI), is an important application design pattern in which a class asks for dependencies from external sources rather than creating them itself. Angular comes with its own dependency injection framework for resolving dependencies(services or objects that a class needs to perform its function).So you can have your services depend on other services throughout your application.

Http Client Module

- HttpClient will help us fetch external data, post to it, etc.
- To use HttpClient we have to import HttpClientModule in app.module.ts.
- Using this we can make http requests for fetching the data, Posting the data, updation and deletion of data.
- Every http method returns Observable type.
- So we can subscribe the data from the Http requests we make.
- Usually we make use of HttpClient in the services and the data subscription is done in the components.

Pipes

- A pipe takes in data as input and transforms it to a desired output.
- A pipe can accept any number of optional parameters to fine-tune its output.
- The parameterized pipe can be created by declaring the pipe name with a colon (:) and then the parameter value.
- If the pipe accepts multiple parameters, separate the values with colons.
- You can chain pipes together in potentially useful combinations as per the needs.

  examples: uppercase, lowercase, titlecase, currency, date etc.

TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Thank You !!!

No.01, 3rd Cross Basappa Layout, Gavipuram Extension,
Kempegowda Nagar, Bengaluru, Karnataka 560019

praveen.d@testyantra.com

www.testyantra.com

EXPERIENTIAL
learning factory