Angular 2.0 for JEE

Lesson 03 : Components

Lesson Objectives

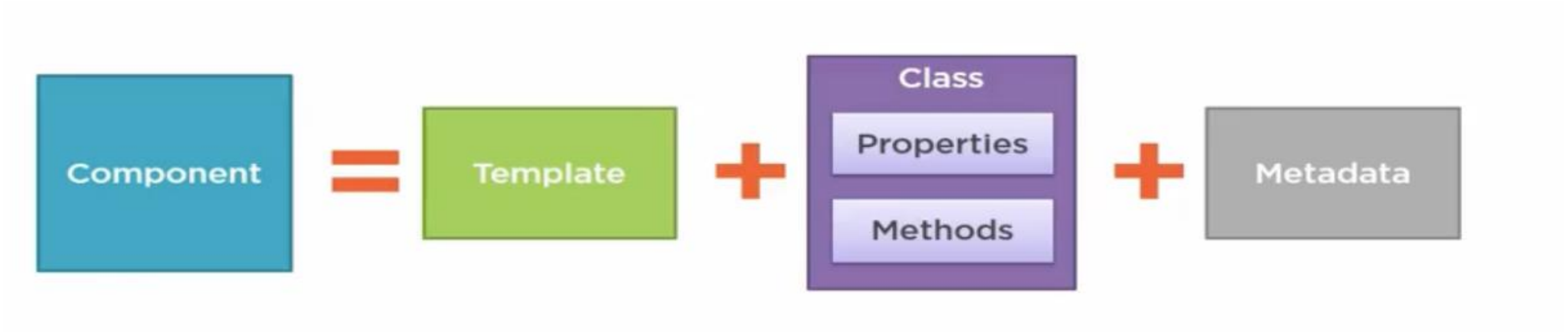
- Introduction of component
- Developing a simple component
- Templates for a component
- Component style
- Component lifecycle

Components

- A *component* controls a patch of screen called a *view*.
- A component's application logic—what it does to support the view—inside a class.
- Components are the main way to build and specify elements and logic on the page.
- In Angular 2, “everything is a component.”
- Component is comprised of a template, metadata and class.
  - Template provides HTML(View) for the user interface.
  - Class provides the code associated with the view.
  - Class contains the properties or data elements to be used in the view and methods to perform actions for the view.

Components

- Component also has metadata, which provides additional information about the component
  - Meta data that identifies the class as an angular component.



Components

- AppComponent

```
import { Component } from
'@angular/core';
@Component({
  selector: 'my-app',
  template: `<h1>Hello {{name}}</h1>`
})
export class AppComponent
 { name = 'Welcome Angular 2'; }
```

Template & metadata

Class

Components-Metadata

- Metadata tells Angular how to process a class.

  export **class AppComponent**

  { name = 'Welcome Angular 2'; }

- To tell Angular that AppComponent is a component, attach metadata to the class. In TypeScript, we can attach metadata by using a decorator,
- @Component decorator, which identifies the class.
- The metadata in the @Component tells Angular where to get the major building blocks .
- export keyword exports the class; thereby making it available for use by other components of the application.

Components-Metadata

- @Component configuration options:
  - selector: CSS selector that tells Angular to create and insert an instance of this component where it finds a <hero-list> tag in parent HTML.
  - template : This is the portion of our component that holds template. It is an integral part of the component as it allows to tie logic from component directly to a view. Its call inline
  - templateUrl: module-relative address of this component's HTML template, its call external
  - providers: array of dependency injection providers for services that the component requires.

Demo

- Component Demo

Template

- HTML is the language of the Angular template
- Template are mostly HTML which is used to tell Angular how to render the component.
- Template for a component can be created using
  - Inline template (Embedded template string)
  - Linked template (Template provided in external html file)
- Interpolation ({{...}})-use interpolation to weave calculated strings into the text between HTML element tags and within attribute assignments. Example
  - `<h1>Hello {{name}}</h1>
  - <h1>Hello world {{10 + 20 + 30}}</h1>
  - <h3> {{title}} <img src="{{heroImageUrl}}" style="height:30px"></h3>

Demo

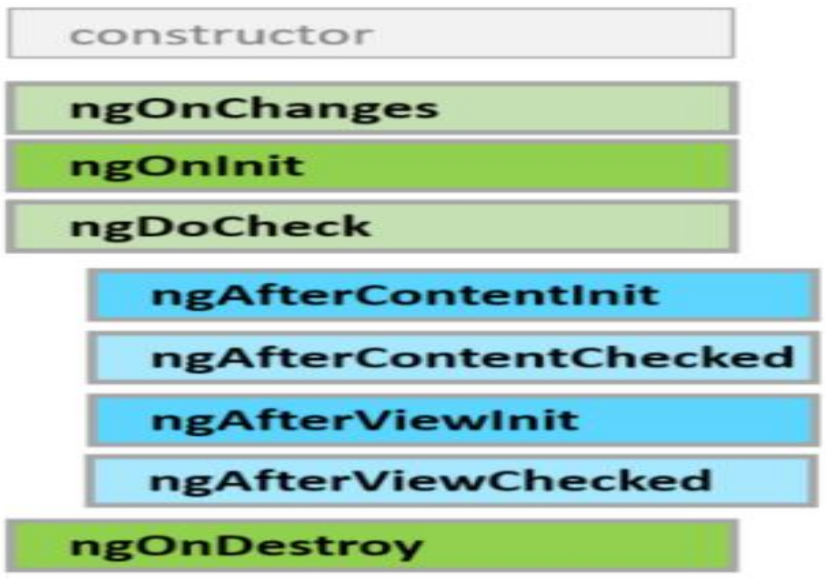
- Component Demo Inline Template
- Component Demo External Template

Component Lifecycle

- Each Angular application goes through a lifecycle.
- If we want to access the value of an input - to load additional data from the server for example - you have to use a lifecycle phase.
- The constructor of the component class is called before any other component lifecycle hook.
- For best practice inputs of a component should not be accessed via constructor.
- To access the value of an input for instance to load data from server component’s life cycle phase should be used.

Component Lifecycle (Contd…)

- A component has a lifecycle managed by Angular.
- Angular creates it, renders it, creates and renders its children, checks it when its data-bound properties change, and destroys it before removing it from the DOM.
- Angular offers **lifecycle hooks** that provide visibility into these key life moments and the ability to act when they occur.



Component Lifecycle (Contd…)

- *After* creating a component by calling its constructor, Angular calls the lifecycle hook methods in the following sequence at specific moments:

Hooks	Purpose and Timing
ngOnChanges()	Respond when Angular (re)sets data-bound input properties. The method receives a SimpleChanges object of current and previous property values. Called before ngOnInit() and whenever one or more data-bound input properties change.
ngOnInit()	Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties. Called once, after the first ngOnChanges().
ngDoCheck()	Detect and act upon changes that Angular can't or won't detect on its own. Called during every change detection run, immediately after ngOnChanges() and ngOnInit().

Component Lifecycle (Contd…)

Hooks	Purpose and Timing
ngAfterContentInit()	Respond after Angular projects external content into the component's view. Called once after the first ngDoCheck(). A component-only hook.
ngAfterViewInit()	Respond after Angular initializes the component's views and child views. Called once after the first ngAfterContentChecked(). A component-only hook.
ngAfterViewChecked()	Respond after Angular checks the component's views and child views. Called after the ngAfterViewInit and every subsequent ngAfterContentChecked(). A component-only hook.
ngOnDestroy()	Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks. Called *just before* Angular destroys the directive/component.

Demo

- Component Life Cycle

Summary

- Every component must be declared in some NgModule and a component can belong to one and only one NgModule
- exports key is nothing but the list of public components for NgModule.
- Angular2 Application is a tree of components and the top level component is nothing but the application.
- Components are Composable.
- Template for a component can be created using InlineTemplate and LinkedTemplate using template and templateUrl respectively.