ANGULAR

Angular is a javaScript framework that helps you build reactive single page applications.

# Angular Versioning

* Angular JS
* Angular 2-2016.Completely rewritten
* Angular 4
* Angular 5
* Angular 6
* Angular 7
* Angular 8
* Angular 9
* Angular 10
* Angular 11

New version is released every 6 months.

### App Root

If we inspect the html page, there are no written codes but only app root written in it which is the selector for app component.

## TypeScript

Typescript is the super set of javascript with some additional features in it. Strong typing is the most important feature of TS. Unlike JS where you can assign a number to a string variable, this is not possible in TS. TS is compiled to JS to run it in the browser.

## Bootstrap

Npm install –save bootstrap installs bootstrap locally in the project.

To make angular aware of boot strapping we need to add it in the angular.json file. Go to architect-> build -> styles.

**“node\_modules/bootstrap/dist/css/bootstrap.min.css”**

# How Angular get loaded and started?

Index.html is the page which is served by the server. In the body of it we have the selector for app component <app-root> tht’s why App component template is loaded when we run the application.

When we give ng serve it will create javascript bundles and automatically add script imports in the index.html file. The first code to be executed is the code wewrite in main.ts file. When we start main.ts file, we bootstrap an angular application and pass AppModule as an argument. In this module, we tell angular about the component it must be familiar with. So it can interpret the app-root in the index.html as that from app component.

# Component

Angular application is composed of various components. The root component is the app component.

Components may have different templates and different application logic. It can be usable codes.

@Component() tells the angular that the class is a component. It is present in angular/core. It has some meta data in it, selector which is used as the html tag, the template points to the file given.

Modules are used to bundle files into packages. To use the created component it must be imported in the AppModule. We have to register it in the NgModule in the declaration array. By adding it in declaration,angular knows it exist but to let type script know we must give the import statement at the top. We donot give extension it path, it will be later added by webpack.

NgModule: It has four properties, declaration, imports, providers and bootstrap.

We can use template instead of templatesUrl for inline template. ` back ticks are used to give multi line commands`

Selector can be given as element, attribute or class.

# Data Binding

Data binding=Communication.

String interpolation: {{data}}, property binding : [property]=”data”

Template

(HTML)

Typescript code

(Business logic)

Output data

React to (User) Events

Event Binding: (event)=”expression”

Combination of both- two way data binding

[(ngModel)]=”data”

**String Interpolation{{}}**

Used to output data from typescript code to HTML template(view).

**Property Binding []**

We bind property of a DOM element to a field which is a defined property in our component typescript code. Angular internally converts string interpolation into property binding.Helps set values for properties of HTML elements or directives.

String Interpolation v/s Property Binding

If we simply want to output some data, use interpolation and if we want to change some properties, use property binding.

**Event Binding ()**

It is used to handle the events raised from the DOM likebutton click,mouse move,etc. When this DOM event happens , it calls the specified method in the component.

**$event**

**Two way data binding**

Update data from component to view and viceversa.

[(ngModel)]=”[property of your component]”

For Two-Way-Binding to work, you need to enable the ngModel  directive. This is done by adding the FormsModule  to the imports[]  array in the AppModule.

You then also need to add the import from @angular/forms  in the app.module.ts file:

import { FormsModule } from '@angular/forms';

# Directives

Directives are instructions in the DOM. They are used to manipulate the DOM elements. By using directives, we can change appearance, behaviour or layout of DOM elements.

The different types of Angular directives are as follows:

1. [Components](https://angular.io/guide/component-overview)—directives with a template. This type of directive is the most common directive type.
2. [Attribute directives](https://angular.io/guide/built-in-directives#built-in-attribute-directives)—directives that change the appearance or behavior of an element, component, or another directive.
3. [Structural directives](https://angular.io/guide/built-in-directives#built-in-structural-directives)—directives that change the DOM layout by adding and removing DOM elements.

**Attribute directives**

Attribute directives listen to and modify the behavior of other HTML elements, attributes, properties, and components.

Many NgModules such as the [RouterModule](https://angular.io/guide/router" \o "Routing and Navigation) and the [FormsModule](https://angular.io/guide/forms" \o "Forms) define their own attribute directives. The most common attribute directives are as follows:

* [NgClass](https://angular.io/guide/built-in-directives#ngClass)—adds and removes a set of CSS classes.
* [NgStyle](https://angular.io/guide/built-in-directives#ngstyle)—adds and removes a set of HTML styles.
* [NgModel](https://angular.io/guide/built-in-directives#ngModel)—adds two-way data binding to an HTML form element.

**Structural Directives**

Structural directives are responsible for HTML layout. They shape or reshape the DOM's structure, typically by adding, removing, and manipulating the host elements to which they are attached.

This section introduces the most common built-in structural directives:

* [NgIf](https://angular.io/guide/built-in-directives#ngIf)—conditionally creates or disposes of subviews from the template.
* [NgFor](https://angular.io/guide/built-in-directives#ngFor)—repeat a node for each item in a list.
* [NgSwitch](https://angular.io/guide/built-in-directives#ngSwitch)—a set of directives that switch among alternative views.

# Component Interaction

**To send data from parent to child using @Input()**

* Import Input from @angular/core in child class.
* Declare variable in child class as @Input() child\_variable.
* In the parent class define the variable whose value should be assigned to child \_variable.
* Do property binding of the declared variable inside the parent template and assin the declared variable in parent to it.

*If we give @Input(‘child’)child\_variable , @*[*Input*](https://angular.io/api/core/Input)*aliases the child component property name child\_variable as child.*

**To send data from child to parent**

* **De**clare variable in parent class
* Import Output and EventEmitter from @angular/core in child class.
* Declare @Output() outpit\_value:EventEmitter<any>=new EventEmitter();
* Define the value for this variable.
* Create a method with $event as argument this.output\_value.emit(this.declaredvalue).Emit this data using the function e.g., on a button click.
* In the parent class create a method to receive the emitted event.
* In the parent HTML,do event binding on the output value defined in child class and assign it to the method in parent class.