What is Angular

- Angular is **a platform and framework for building single-page client applications using HTML and TypeScript**. Angular is written in TypeScript. It implements core and optional functionality as a set of JavaScript/TypeScript libraries that you import into your applications.
- Developed by google and the first version of Angular is **AngularJs**, which used javascript as scripting language.
- Latest Angular version is **Angular 12**.
- Typescript is superset of javascript which provides type safety.
- **Business Benefits of Angular Features**
- Effective Cross-Platform Development.
- High Quality of the Application.
- Improved Speed and Performance.
- Faster Development Process.
- Readable and Testable Code.
- More Lightweight Web Applications.

Angular Setup

Following development environment and editor are required to develop an Angular application:

- Node.js
- Node Package Manager(NPM)
- IDE (Integrated development environment) usually VS Code

Install Angular:

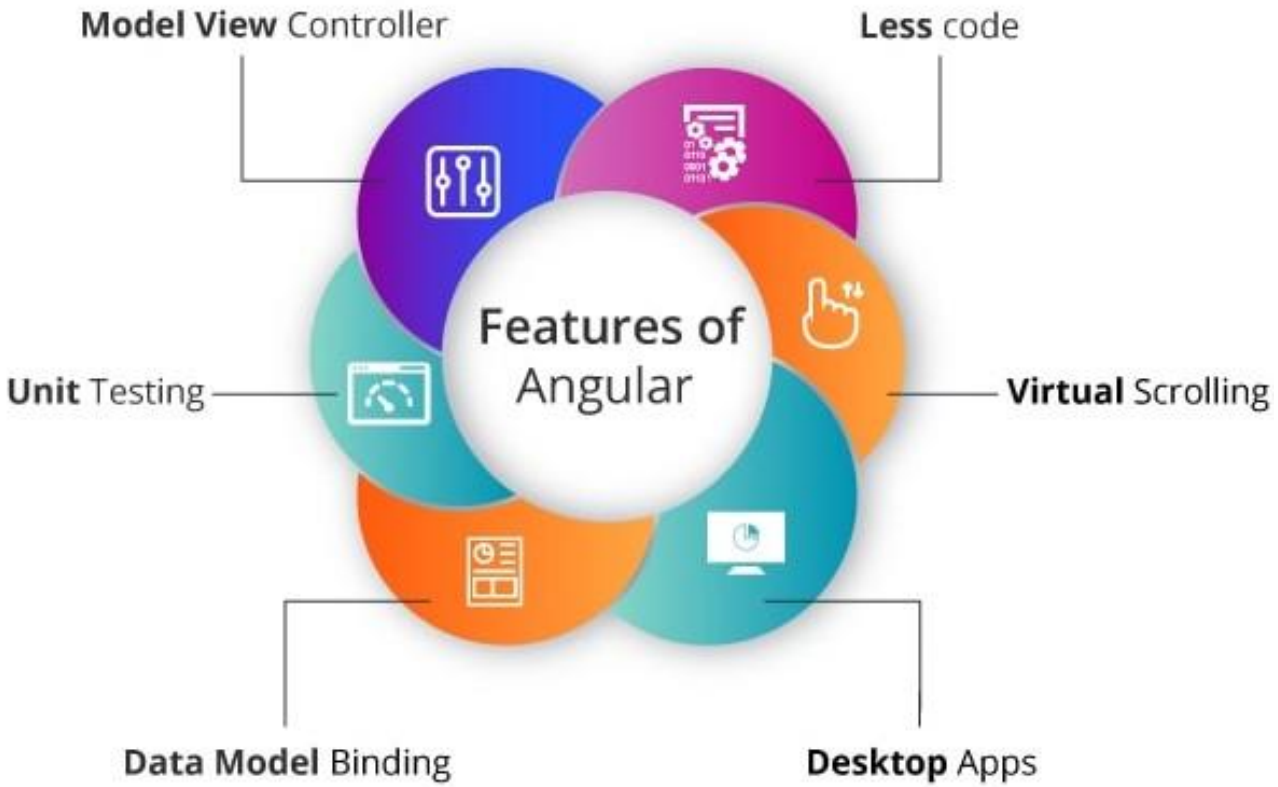
- npm install –g @angular/cli ---To install angular globally
- ng –version (or) ng -v --- To check angular version
- npm –version (or) npm -v ---- To check npm version
- npm install (or) npm i ----To download packages and its dependencies

Create Angular Project and Run:

- ng new ProjectName
- ng server ---- To run angular project in windows
- ng generate component ComponentName (or) ng g c ComponentName -- To generate/add new component
- ng generate service ServiceName (or) ng g s ServiceName -- To add a service in angular application

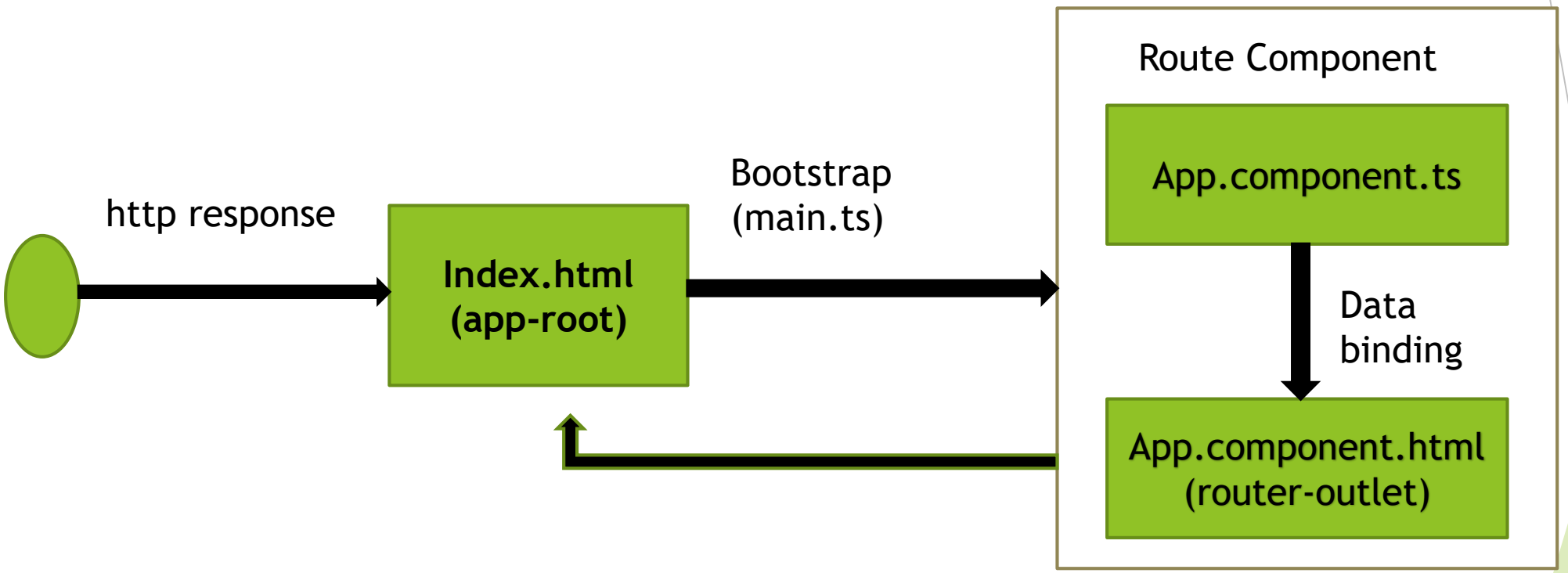
Lifecycle

ngOnChanges	Called after a bound input property changes
ngOnInit	Called once the component is initialized
ngDoCheck	Called during every change detection run
ngAfterContentInit	Called after content (ng-content) has been projected into view
ngAfterContentChecked	Called every time the projected content has been checked
ngAfterViewInit	Called after the component's view (and child views) has been initialized
ngAfterViewChecked	Called every time the view (and child views) have been checked
ngOnDestroy	Called once the component is about to be destroyed

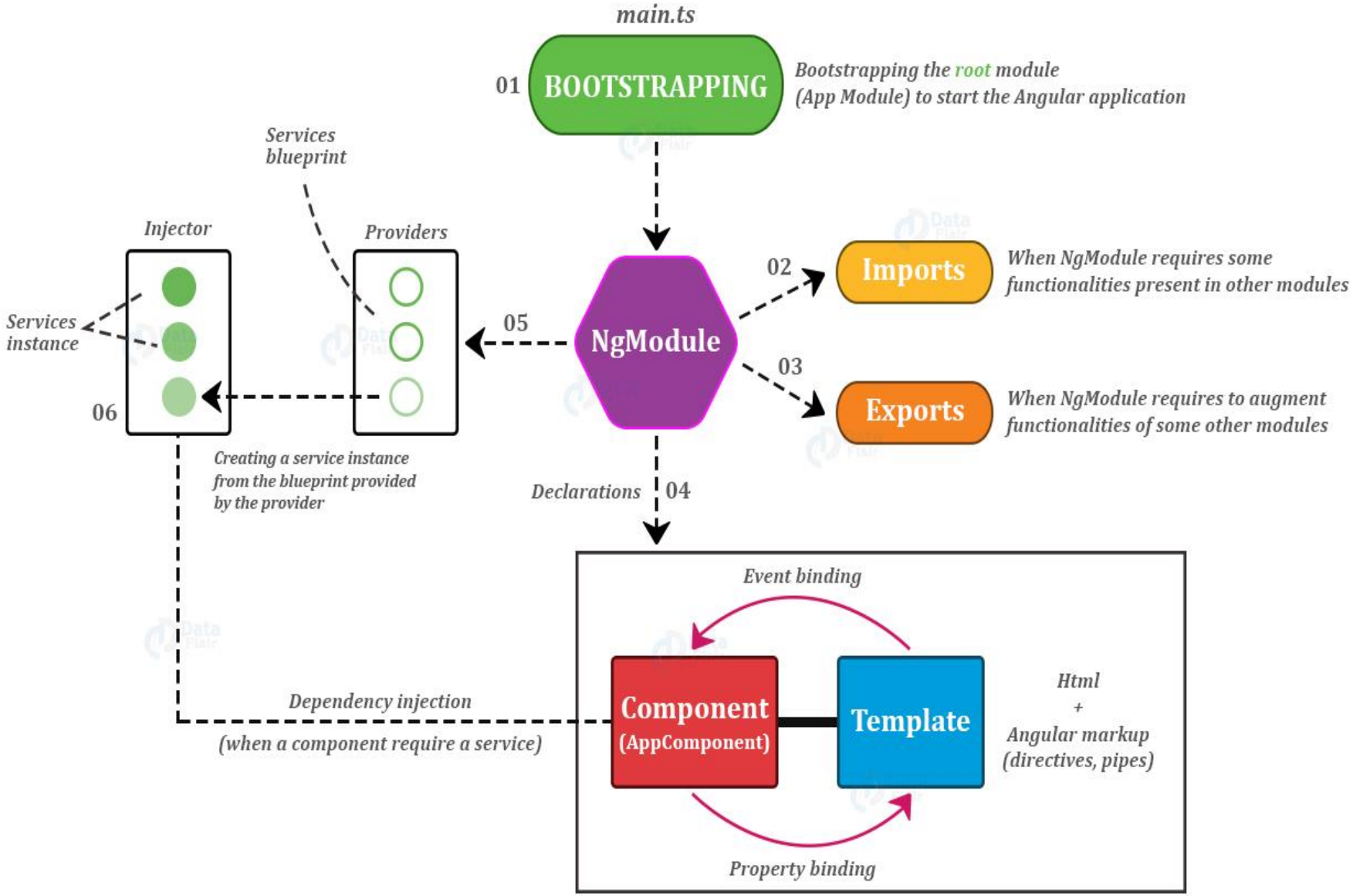


Features of Angular

main.ts file is a TypeScript file. It is the starting point of our application. Here, we can bootstrap (the process of initializing or loading) our main module using bootstrapModule

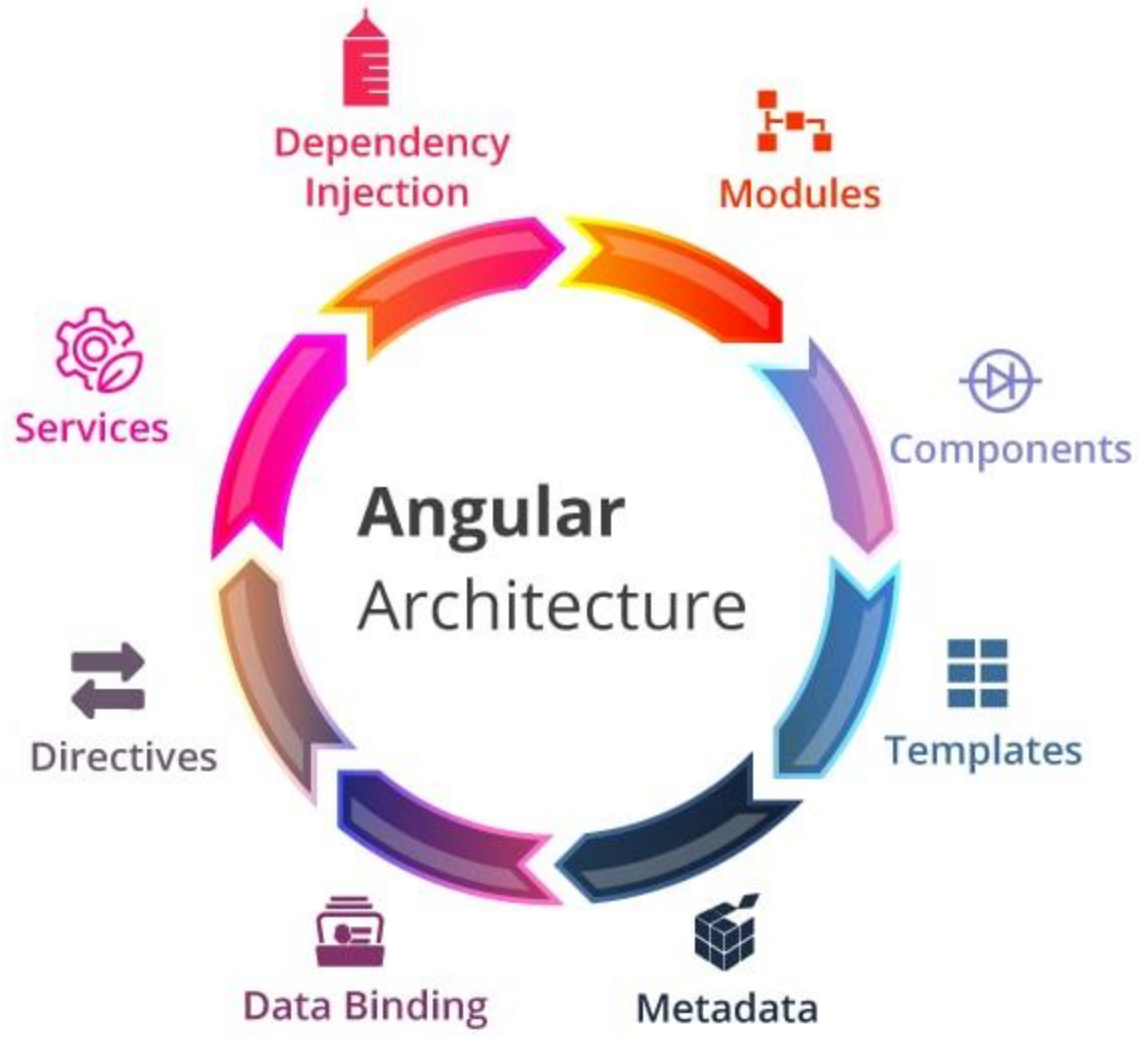


index.html file contains our Angular application. Here, you can not find any reference to CSS or other JS files. All other pages are dynamically inserted into the page.


main.ts
01
BOOTSTRAPPING
Bootstrapping the root module
(App Module) to start the Angular application
Services
blueprint
Injector
Providers
Services
instance
05
NgModule
02
Imports
When NgModule requires some
functionalities present in other modules
03
Exports
When NgModule requires to augment
functionalities of some other modules
06
Creating a service instance
from the blueprint provided
by the provider
Declarations
04
Event binding
Component
(AppComponent)
Template
Html
+
Angular markup
(directives, pipes)
Dependency injection
(when a component require a service)
Property binding

Basic Building Blocks of Angular Application

1. Module
2. Template
3. Component
4. Metadata
5. Data Binding
6. Services
7. Directives
8. Dependency Injection



Module

- Angular is a modular platform and it may contain one or more Angular Module or NgModules depending on the demand. It is the essential module that is always present is the Root module namely “AppModule” in the application.
- **NgModule** is a Decorator function that handles the compilation part of the application.
- **Imports**: When NgModule requires some functionalities, which are present in other modules, it inherits these functionalities using Import feature in the form of list.
- **Exports**: When the NgModule requires to augment functionalities of some other module, this is used.
- **Declarations**: When a view is displayed on the screen of any application, it is the mixture of Component and Template (Directives + Pipes), which are constituents of declarations.
- **Providers**: When you need to access external features in an application like routing data to a server, it is not done using Components rather than this Services comes into action, which is a special concept fabricated to do all these external tasks. Providers contain the list of Services present in the module for external work. They do not have the actual service, but the Blueprint of Services are present.
- **Bootstrapping**: It is the initial process that is responsible for the application to start. This process starts in main.ts (TypeScript) file which is then compiled into main.js (JavaScript) file. It is one of the essential processes in the working of an application. Bootstrapping is set only by the root module.

```
@NgModule({
  declarations: [
    AppComponent,
    NavMenuComponent,
    HomeComponent
  ],

  imports: [
    BrowserModule,
    HttpClientModule,
    FormsModule,

    RouterModule.forRoot([
      { path: '', component: HomeComponent},
      { path: 'grid', component: gridComponent },
      { path: 'fetch-data', component: FetchDataComponent },
    ]),
  ],

  providers: [FetchService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Component

Modules

Routing

Services & Root Component

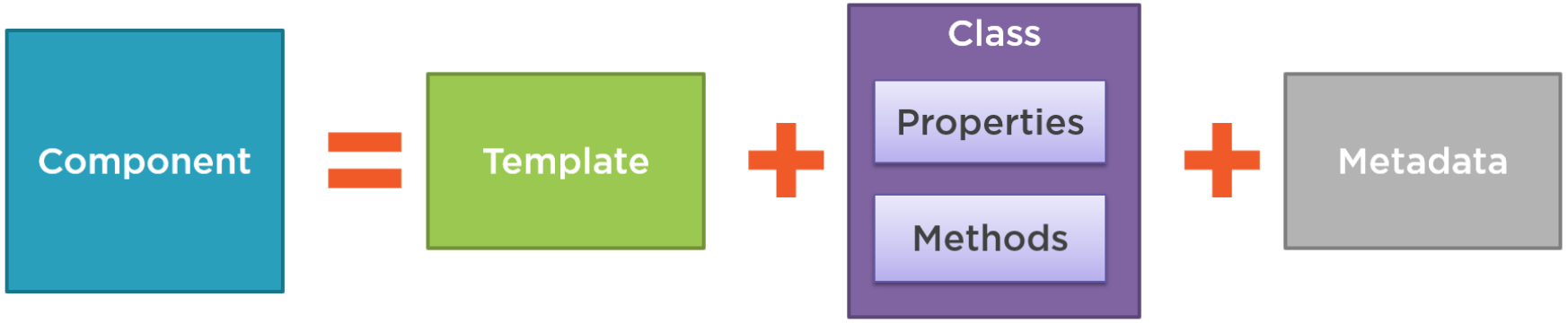
Components

- The view of the screen is controlled by the Component. At least one component is always present in every angular project, i.e. the root component. The work of the root component is to connect the Document Object Model (DOM) of the page with the component hierarchy.
- Each Component has a class that contains data and the logic of the application, and it is also linked with the HTML template that defines the view which is to be displayed at the target app. There is @component decorator present that provides the template and metadata

Template

- The template is must to format and augment the application before it is displayed. Template combines Angular Markup and HTML, which can mutate and modify the elements before displaying. It also provides program logic and connects your application data and DOM using binding markup. There are two types of data binding present:
- **a. Event Binding:** It is used when the user inputs data and your app needs to respond to it by updating data of your application.
  b. Property Binding: This data binding property comes in handy when your application computes some values based on the given data and you need to reflect that into the HTML.

What Is a Component?



- View layout
- Created with HTML
- Includes binding and directives

- Code supporting the view
- Created with TypeScript
- Properties: data
- Methods: logic

- Extra data for Angular
- Defined with a decorator

Metadata

- Metadata is commonly is the Decorator. We use it so that it can configure the class according to its expected behavior. Metadata is attached to the typescript using the decorator. For example, @component decorator. The component decorator is defined just after the AppComponent which enables Angular to comprehend that it is a component.
- The class identified by the component is the Component Class. The decorator takes information which is necessary for angular to create and present the components and its view along with the required confirmation object.
  Some useful configuration options of @component:
- **templateUrl:** Contains the module relative address of the components HTML template.
- **Providers:** Contains the dependency injection providers array which is required by the services of the component.
- **Selector:** It is a CSS selector which tells the angular to instantiate the component when it finds the <app-root> tag, i.e., the parent component

```
@Component({
  selector: 'app-nav-menu',
  templateUrl: './nav-menu.component.html',
  styleUrls: ['./nav-menu.component.css']
})
```

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

@Component
({
 selector: 'app-root',
 templateUrl: './app.component.html',
 //template: ` <div>{{title}}</div>
})

export class AppComponent {
  title:string = 'SunilOS';
}
```

Import component

Metadata

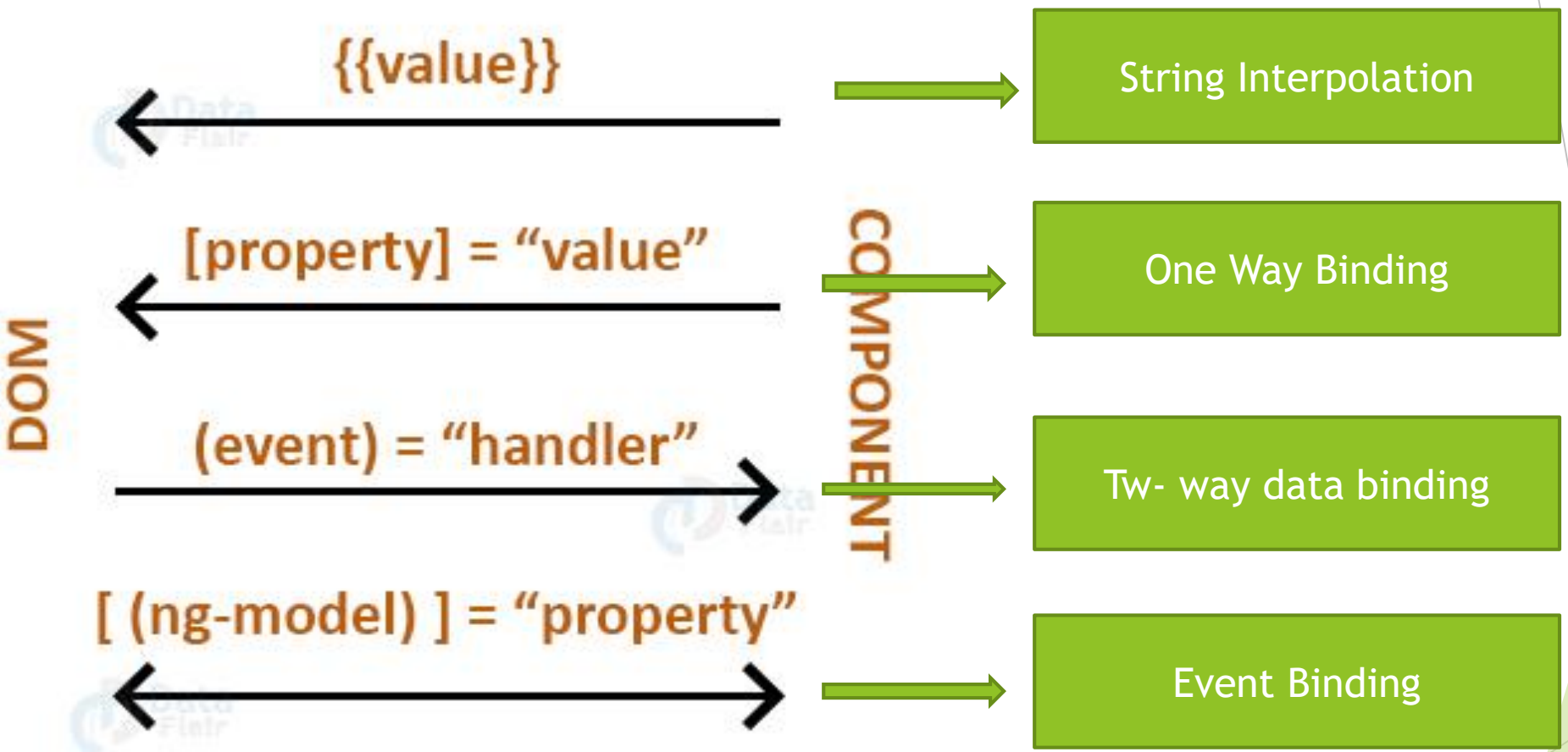
View html

Template

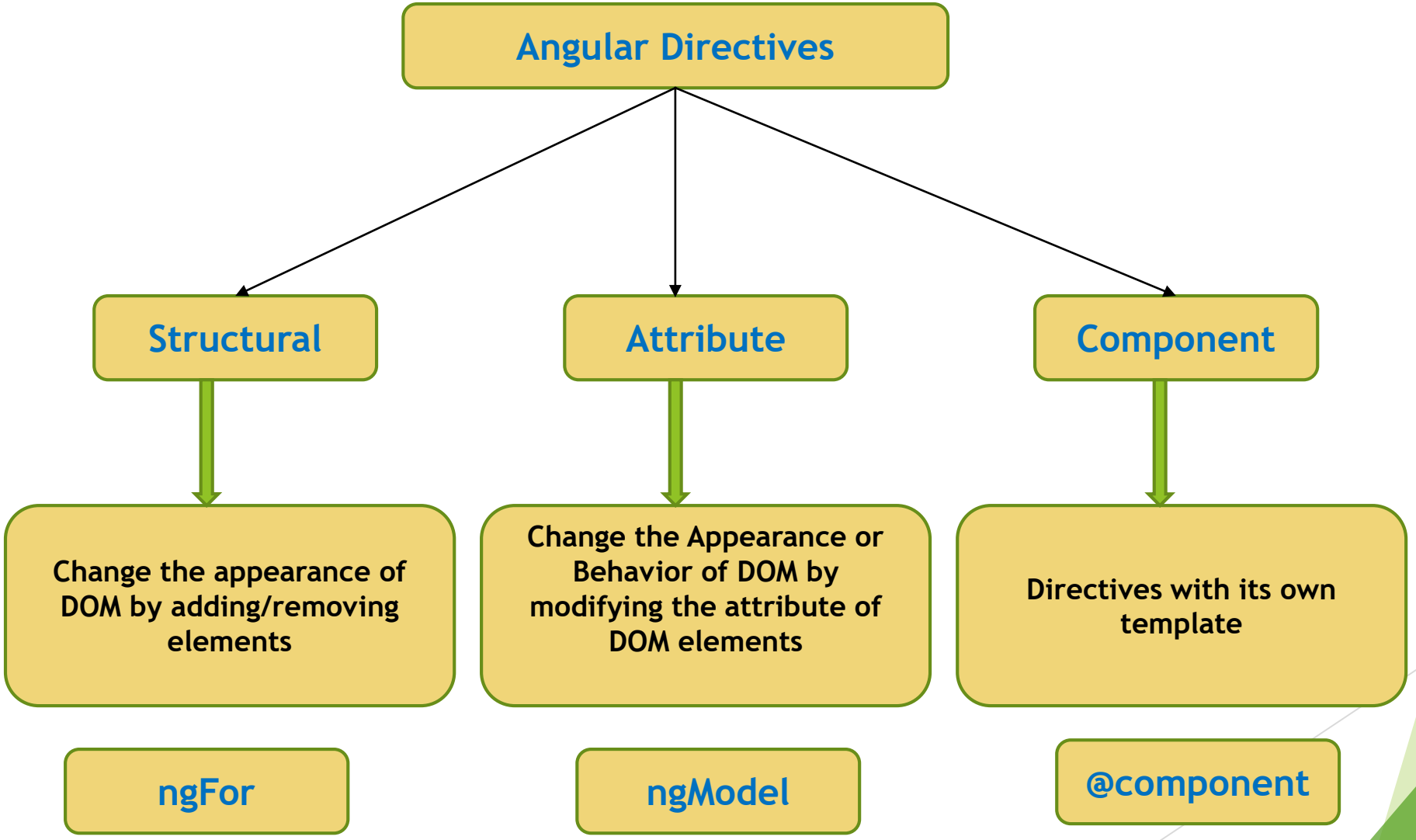
Class

Attribute

Data Binding



Directives:


Angular Directives
Structural
Attribute
Component
Change the appearance of DOM by adding/removing elements
Change the Appearance or Behavior of DOM by modifying the attribute of DOM elements
Directives with its own template
ngFor
ngModel
@component

Services, Routing & Dependency Injection

Services:

- Angular services are singleton objects that get instantiated only once during the lifetime of an application. They contain methods that maintain data throughout the life of an application, i.e., data does not get refreshed and is available all the time. The main objective of a service is to organize and share business logic, models, or data and functions with different components of an Angular application.

Routing:

- Routing allows you **to display specific views of your application** depending on the URL path. To add this functionality to your sample application, you need to update the app. ... You import this module from @angular/router

Dependency Injection:

- DI, is a design pattern in which a class requests dependencies from external sources rather than creating them.
- The client service will not create the dependent object itself rather it will be created and injected by an Angular injector

```
export class FetchDataComponent implements OnInit {
  public forecasts: WeatherForecast[];
  User:string;

  constructor(private _fetch:FetchService, private router:Router) {  }
  ngOnInit(){
    this.FetchData();
    if (this.User=="Admin"){
      this.router.navigateByUrl('/Home');
    }
  }
```

Service Injection in contorller

Router Injection

Navigate to Home page

Deployement:

- Command : **ng build --prod**