Avinash Konduri

Full Stack Developer

Section 1: Introduction & Basics of Angular

Section 2: First App in Angular [Practical Starts Here]

Section 3: Data Binding,ngFor,Style Management, Pipes, Build-in Directives

Section 4: Modules & Services

Section 5: Rxjs & Rest API calls

Section 6: Authentication & Security

Section 7: Template Driven Forms

Section 8: Reactive Forms

Section 9: Component Communication

Avinash Konduri

Full Stack Developer

Section 10: Debugging and Auto-Deployment

Section 11: Life Cycle Hooks

Section 12: Pipes - Deep Dive

Section 13: Directives - Deep Dive

Section 14: Advanced Routing

Section 15: Animations

Section 16: Feature Modules

Section 17: Dynamic Components

Section 18: Unit Testing and Angular Zones

Introduction & Basics of Angular

Course Introduction

Introduction to Angular

Where to Start Practical

Goals of Angular

Code Compilation Process in Angular

Do’s and Don'ts of Angular

Building Blocks of Angular

Angular Architecture

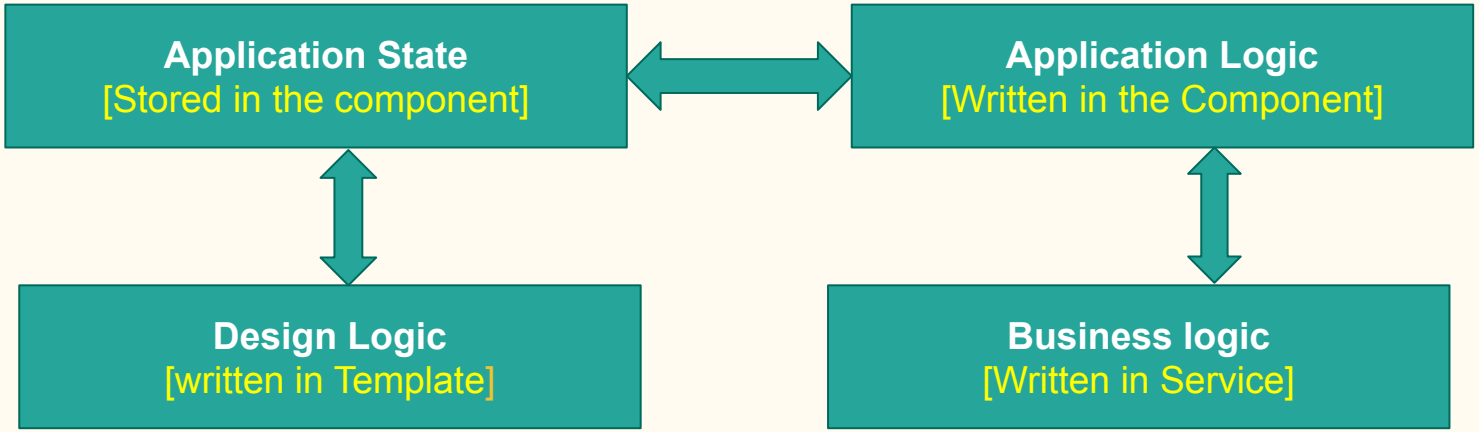
Overview of Angular packages

Angular Folder Structure

What is Angular ?

- Most Popular frontend framework to create maintainable web applications
- Completely rewrite of Angularjs
- Developed by Google
- Angular CLI Enables you to create things faster

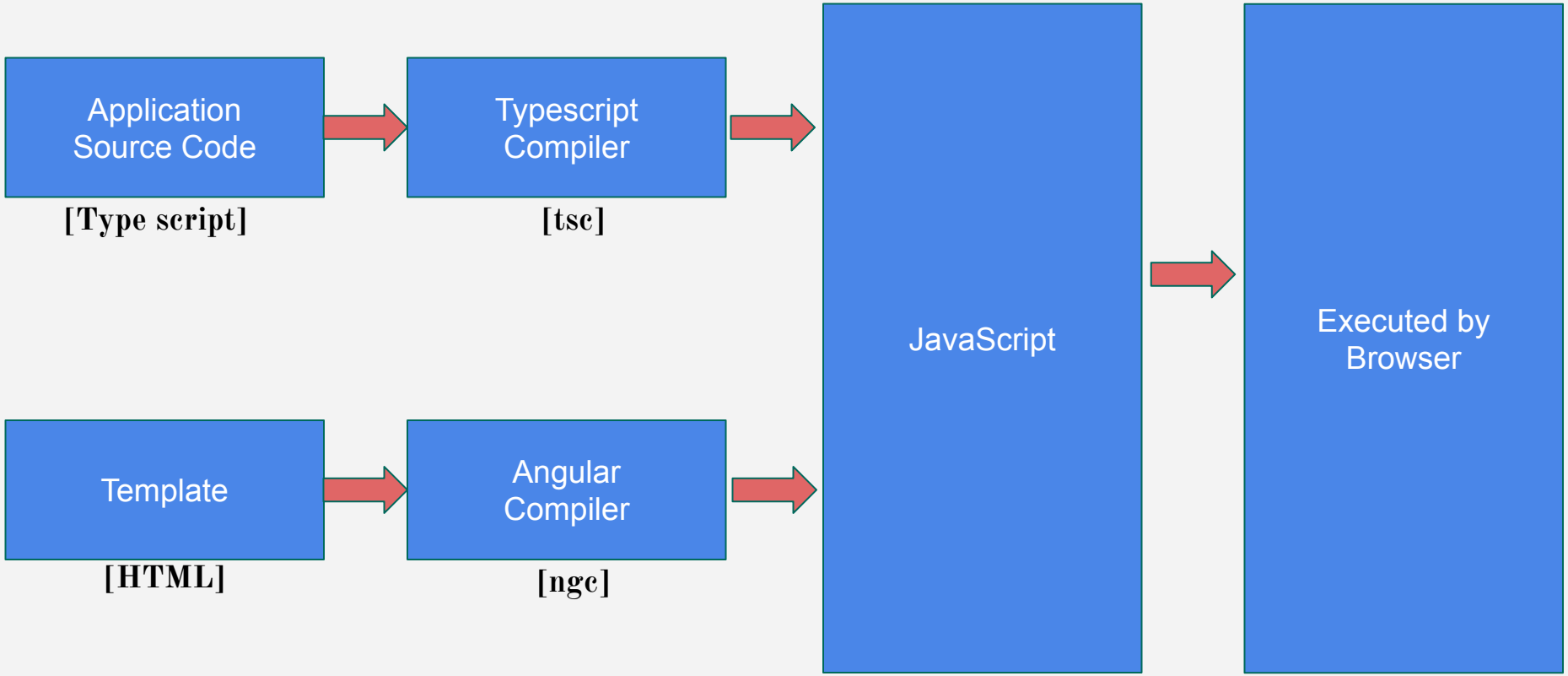
Features of Angular



Goals and Advantages of Angular

- Separation of DOM Manipulation Logic from Application Logic
- Separation of HTML Logic from Application Logic
- Separation of Business Logic from Application Logic

- Make the code Unit Testable
- Make the "Single-Page Application Development" easier
- Make the code Maintainable


Application Source Code
[Type script]
Typescript Compiler
[tsc]
Template
[HTML]
Angular Compiler
[ngc]
JavaScript
Executed by Browser

Do’s and Don'ts in Angular

Don’ts

- Never perform DOM manipulation directly in Angular Components to make the "Application Logic independent of "Design logic" and make it unit testable.
- Never write Javascript code in Angular Templates.
- Never write Business logic in Components.
- Avoid using jQuery to manipulate DOM elements.

Do’s

- Always use Routing and Modules
- prefer Routing Guards and JWT for Authentication & Security
- Always manipulate "application data" by writing the "application Logic" in the components
- Always place global css "styles.css" file and local css style in "component.css" file
- Always write "REST-API calls" and Business Logic in Services only and return Observable from Services
- prefer to use Bootstrap [or equivalent].
- Always user css-pre processor, such as scss.

Building Blocks of Angular

Building Blocks of Angular

Modules
[Collection of Components]

DataBinding
[Mediates between components and templates]

Components
[App Data + Event Handler]

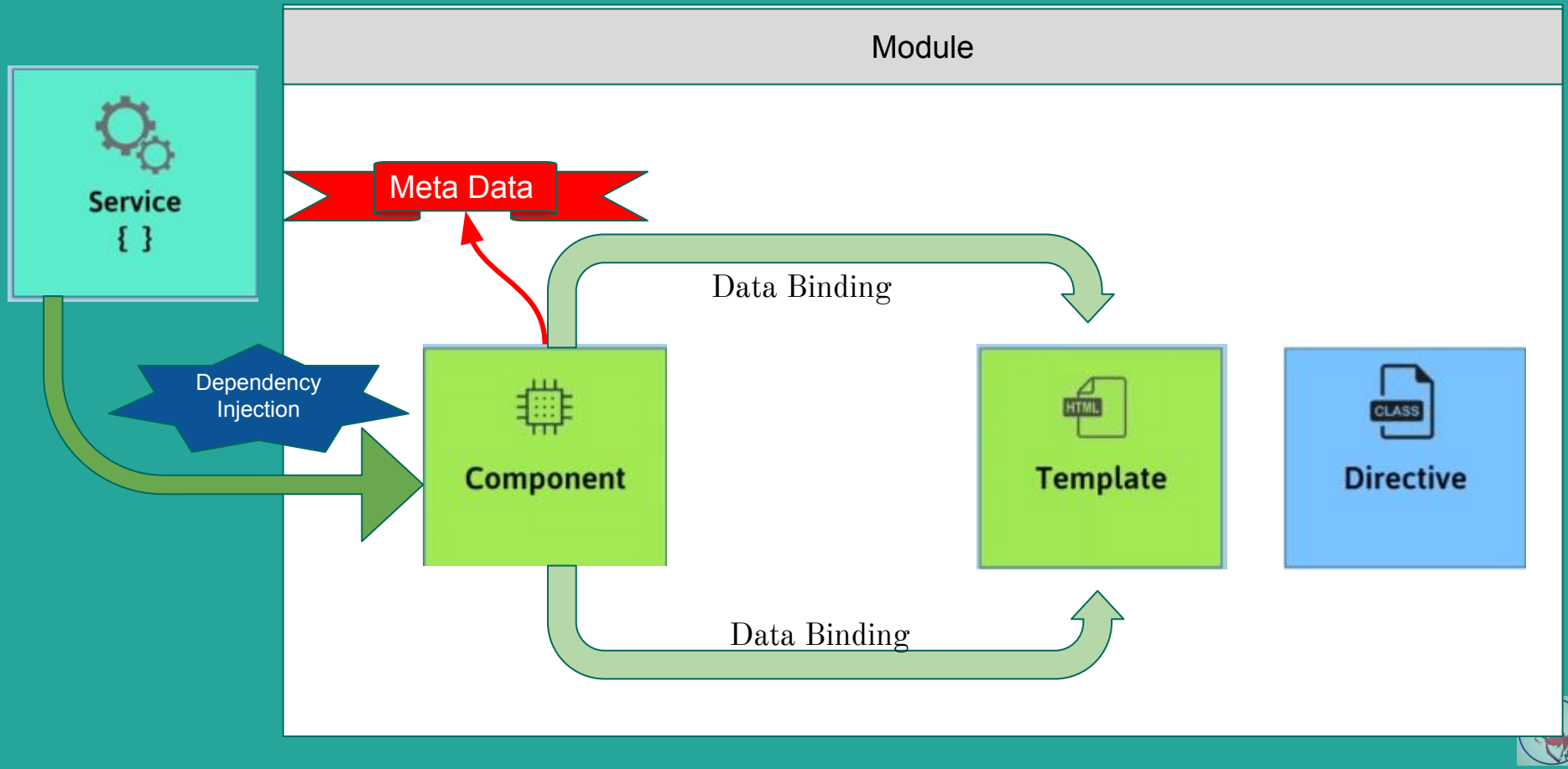
Templates
[Design Logic]

Dependency Injection
[Loads service objects in to components]

Services
[Business Logic + REST API calls]

Directives
[Direct DOM Manipulation]

Angular Architecture



Angular Packages

1.@angular/core

- Provides essential pre-defined decorators,classes,interface and modules that are needed to run every angular application
- EX: @component,@NgModule, @pipe,@Directive, @Injection, @inject,Ngzone,Onchange,OnInt,ApplicationModule etc

2.@angular/common

- Provides essential pre-defined Provides built-in directive that are useful for most of the real-time applications
- Ex: ngIf,ngSwitch,ngClass,ngFor etc

3.@angular/compiler

- Compiles "templates"(html code) into "javascript code"

4.@angular/platform-browser-dynamic

- Invokes the angular compiler(JIT compilation) and specifies the startup module and also starts executing the application

Angular Packages

5.@angular/platform-browser

- Provides a set of pre-defined classes that are related to DOM and browser interaction.
- Ex:BrowserModule

6.@angular/forms

- Provides necessary pre-defined classes that are related to a are needed to create and execute angular forms
- Ex: FormsModule, ReactiveFormsModule, Validators, ngModel,ngForm etc.

7. @angular/router

- Provides necessary pre-defined classes that are needed to create and execute angular routes
- Ex: RouterModule,Routes, ActivatedRoutes,canActive, routerLink ect.

8.@angular/animations

- Provides necessary pre-defined classes that are needed to create and execute angular animations
- Ex: BrowserAnimationsModule, animate, state, style, transition etc.

Angular Packages

9.@angular/cli

- Provides necessay pre-defined commands that are needed to create, compile, build, add items in angular applications
- Ex: ng new, ng serve, ng build, ng test etc

10.rxjs

- Provides necessary pre-defined classes for creating Observables, which are needed to represent the response of REST_API calls of AJAX
- Ex: Observable, Observer, Subject etc

11.zone.js

- Provides necessary pre-defined classes for executing "change detection processes",while executing angular app

Angular File Folder Structure

- e2e : Contains "end-to-end" test cases
- src : Contains source code of the application
  - app
    - app.component.scss : Contains CSS styles of AppComponent.
    - app.component.html : Contains templates of AppComponent
    - app.component.spec.ts : Contains unit test cases of AppComponent
    - app.component.ts : Contains AppComponent.
    - app.module.ts : Contains AppModule
    - app-routing.module.ts : Contains Routing Configuration
  - assets : Contains static files such as images
  - favicon.ico : Contains browser icon
  - index.html : Default page / startup page
  - main.ts : Defines Startup Module
  - polyfills.ts : Defines polyfills (additional scripts) needed to load & run app
  - style.scss : Contains global CSS styles of entire app
- angular.json : Contains Angular CLI configuration
- package.json : Defines current app (package) details and its dependencies
- tsconfig.json : Contains TypeScript Compiler configuration settings

Section 2: First App in Angular [Practical Starts Here]

Creating First app in Angular

Adding bootstrap

Adding Bootstrap navBar

Creating & Nesting Components

Creating Basic Routing

1. How to Specify Port Number in "ng serve" command

-pt 5200
--port 5200
--port=5200
Both B & C

2. What is the default location of global css file "styles.scss" in Angular App?

app\styles.scss
src\styles.scss
app\src\styles.scss
src\assets\styles.scss

3. Which tag is used for navigation bar in Bootstrap in Angular App?

<navigation>
<menu>
<nav>

Assignment: Wizard Assignment: Creating Simple Wizard using Angular Routing

Create a set of components and create wizard navigation among them, using Angular routing. The user can navigate forth and back using the buttons.

Assignment: Creating Wizard using Routing

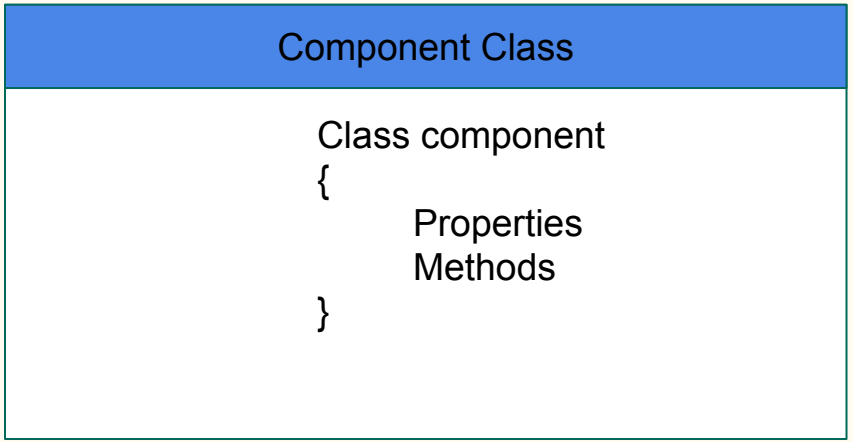
Task:

Create PersonalDetailsComponent, ContactsDetailsComponent, SkillsComponent and

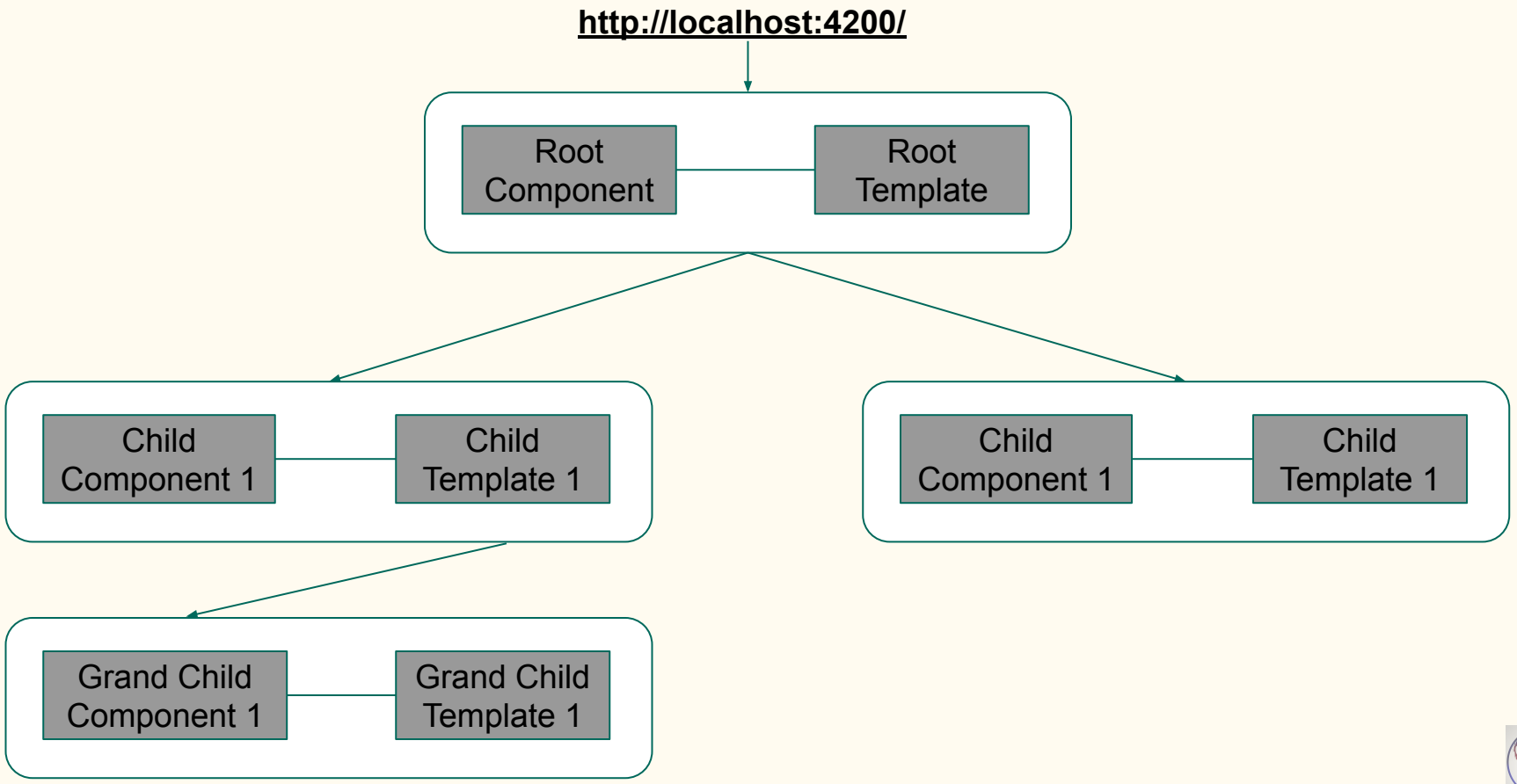
WorkExperienceComponent and WizardFinised and enable navigation using routerLinks, in a wizard

What is a Component

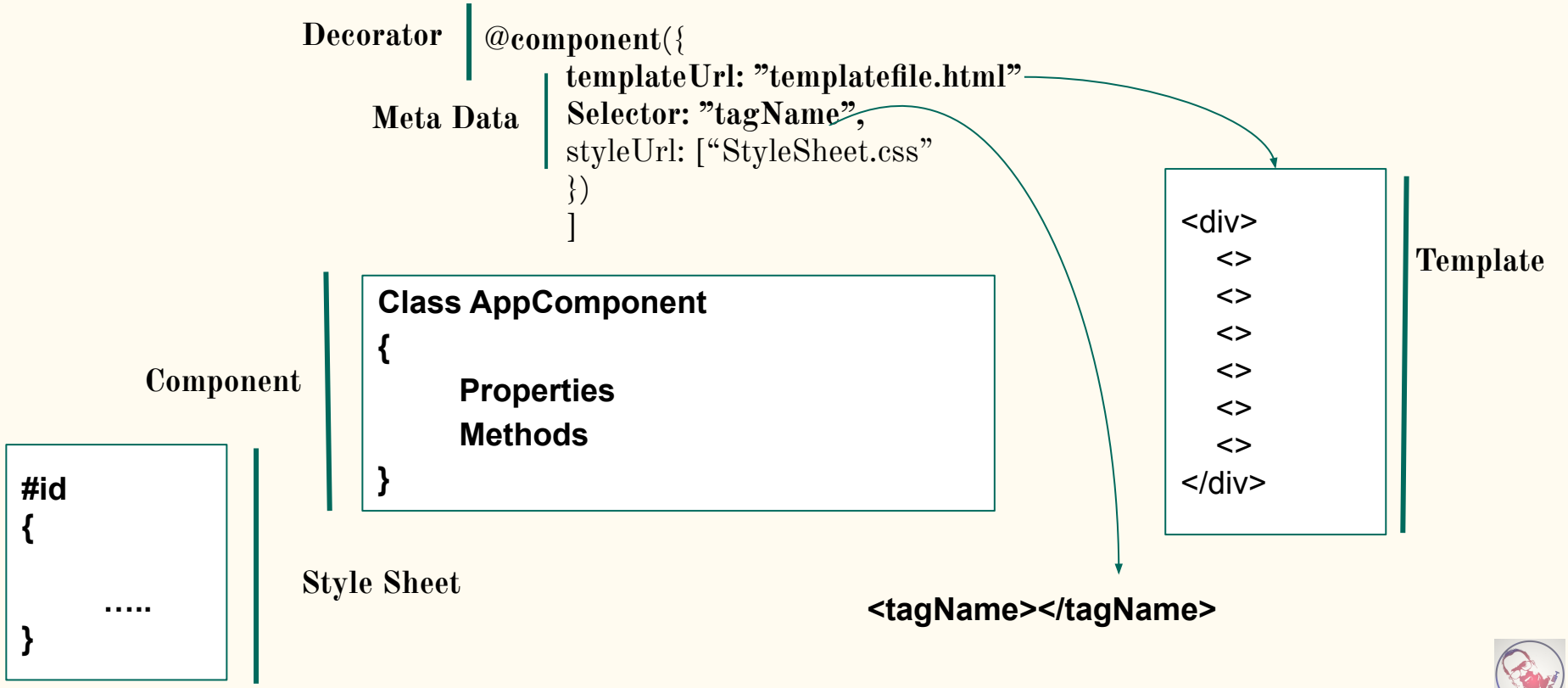
- Component class contains "programming logic" of the application
- COmponent class contains "application data" + "event handler methods"
- Component class is responsible to supply data to the template



Component Hierarchy

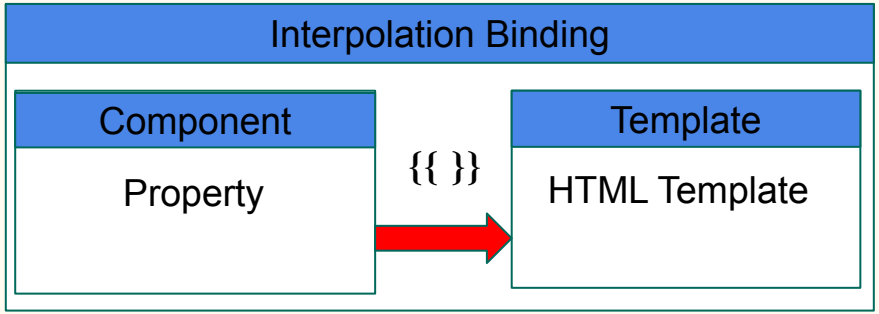


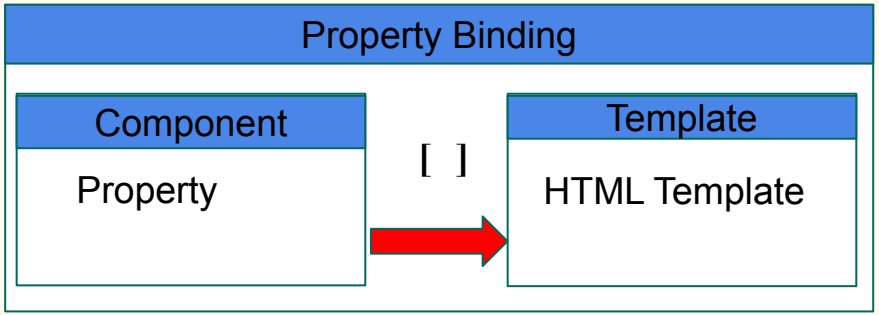
Component Metadata

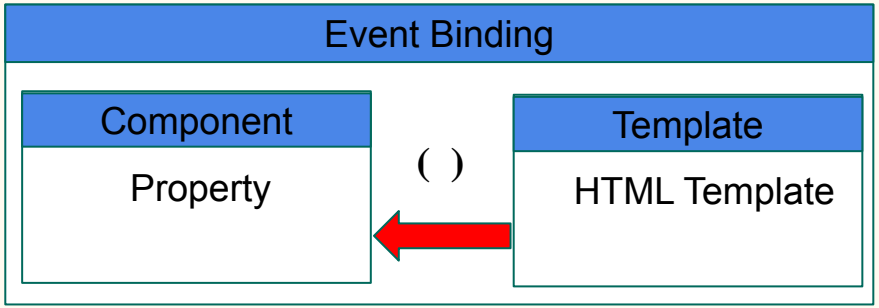


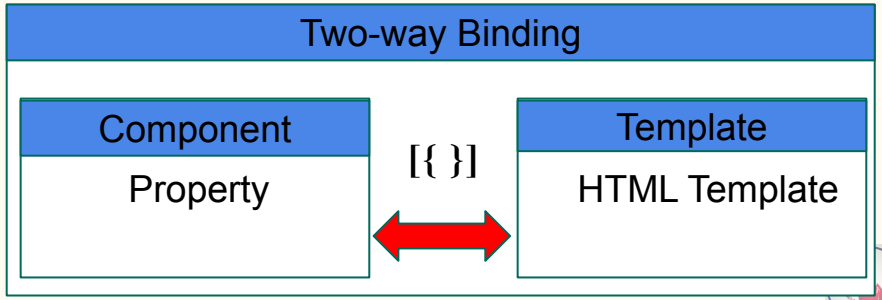
Data Binding

- Relationship between "Template's HTML Element" and "Component's Property"
- The value of "Template's HTML Element" will be automatically updated in to "Component's Property" and vice versa









Data Binding

- [style.property] = "value"
- [style.property] = "(condition)? truevalue : falsevalue"

- [ngClass] ="value"
- [ngClass] = "(condition)? truevalue : falsevalue"

ngIf

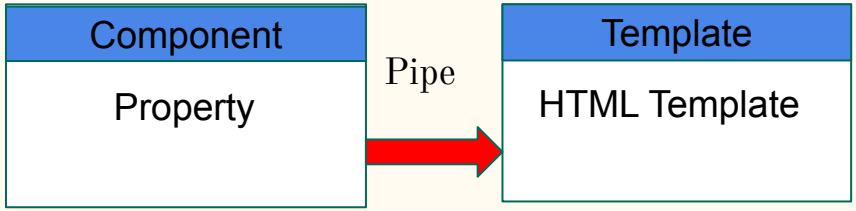
Simple ngIf

```
<tag *ngIf=”condition”>
</tag>
```

ngIf and Else

```
<tag *ngIf="condition;then TrueTemplate; else
FasleTemplate">
</tag>
```

Built-in Pipe



{{property | uppercase}}	Converts string to Uppercase
{{property | lowercase}}	Converts string to Lowercase
{{property | slice:startIndex:endIndex}}	Gets Part of string, between startIndex and endIndex
{{property | number:.2}}	Provides digit grouping and controls decimal places
{{property | currency:"USD"}}	Provides currency symbol
{{property | percent}}	Converts the number to percent
{{property | json}}	Converts the "JavaScript Object" to :"json"
{{property | date}}	specifies date format

Date Formats of Date Pipe

shortDate	31/12/2019
mediumDate	Dec 31, 2019
longDate	December 32,2019
fullDate	Monday, December 31,2019
shortTime	11:59 AM
mediumTime	11:59:59 AM
short	31/12/2019,11:59 AM
medium	Dec 31, 2019, 11:59:59 AM
d/m/y	31/12/2019

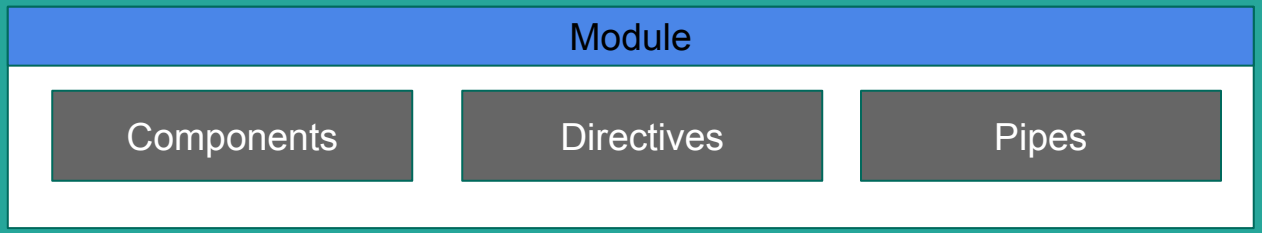
y-m-d	201-12-31
h:m:s	11:59:59
a	AM
H:m	23:59
EEE	Tue
EEEE	Tuesday
MMM	Dec
MMMM	December
z	+0530

What is Angular's Module

- ❖ Module is collection of Components,directives, pipes
- ❖ Mainly used to organize the components and others (directives and pipes)

Goals of Modules:

- Consolidate components, directives, pipes into cohesive of functionality
- Make some of the components, directives, pipes public; so that, other modules component templates can use them
- Import components, directives, pipes from other modules, that are required by current module's component templates
- Provides services that other components can use



What is Angular's Module

- Angular Modules are called NgModules
- An NgModule is a class that is decorated with "NgModule" decorator, that contains the following metadata
- **Module metadata**
- declarations:list of components, directives and pipes, that are part of current module
- exports:list of components, directives, and pipes, that are public that can be accessible in other modules, that are importing the current module
- imports:list of modules, that the current module imports; so, the current module can use components, directives,pipes that are already exported by that particular module
- providers:List of services that can be used by the components, directives and pipes of current module

Simple ngIf

```
@NgModule({
declarations:[..., ..., ..],
exports:[..., ..., ...],
imports:[..., ..., ...],
providers:[..., ..., ..,],
})
class ModuleName
{
}
```

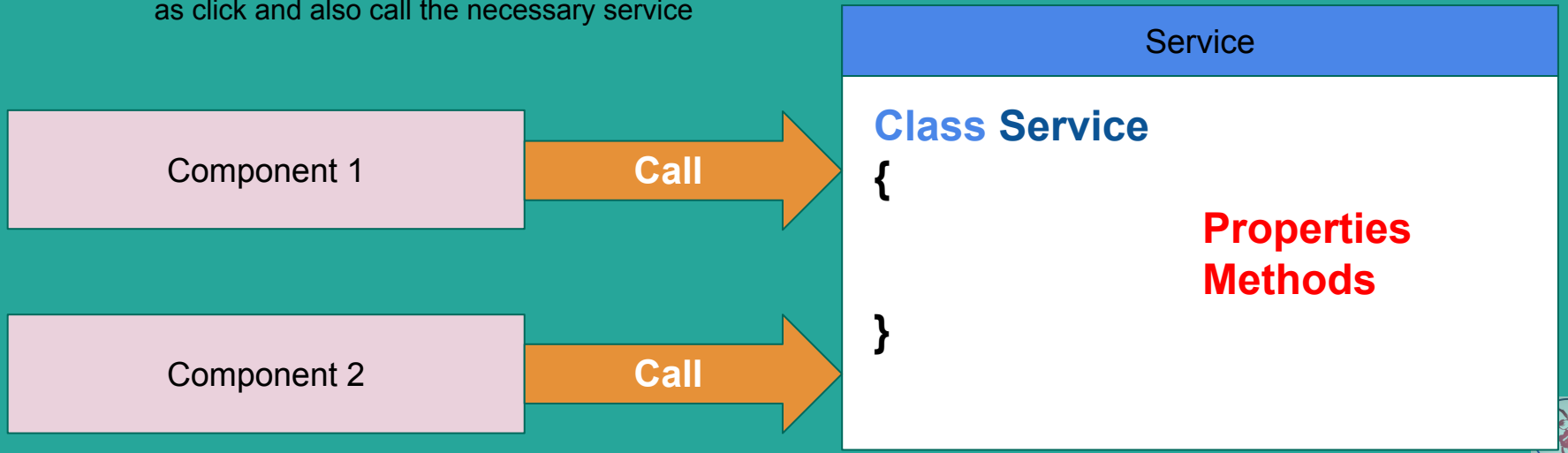
NgSwitch

NgSwitch

```
<tag
*ngSwitchCase="value1">Content here</tag>
<tag
*ngSwitchCase="value2">Content here</tag>
<tag
*ngSwitchCase="value3">Content here</tag>
<tag *ngSwitchDefault>Content
here</tag>
</tag>
```

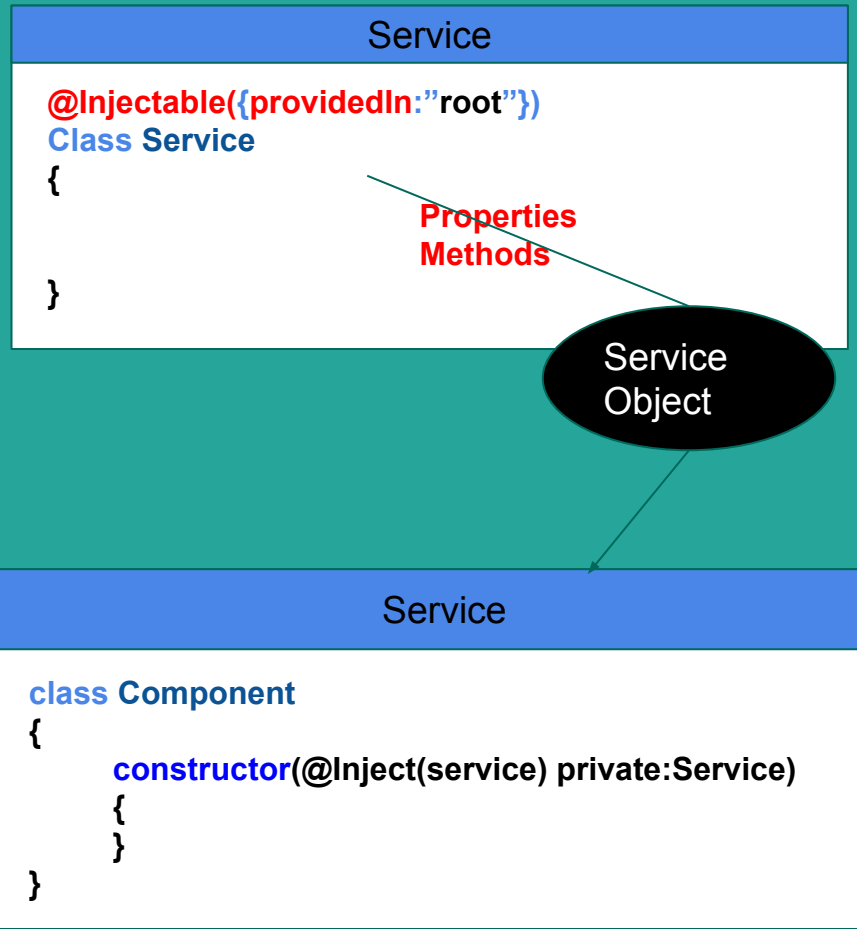
What is Service in Angular?

- Service is a class, which is a collection of properties & methods, which contains re-usable programming logic, which mainly contains “business logic” and also performs “data source interaction”
- Service can be accessible in components
- Goals of Service:
  - To separate business logic and data access logic from component
  - Makes components contain code for only supplying the data to the template and respond to the user actions such as click and also call the necessary service



Steps to work with Angular Services

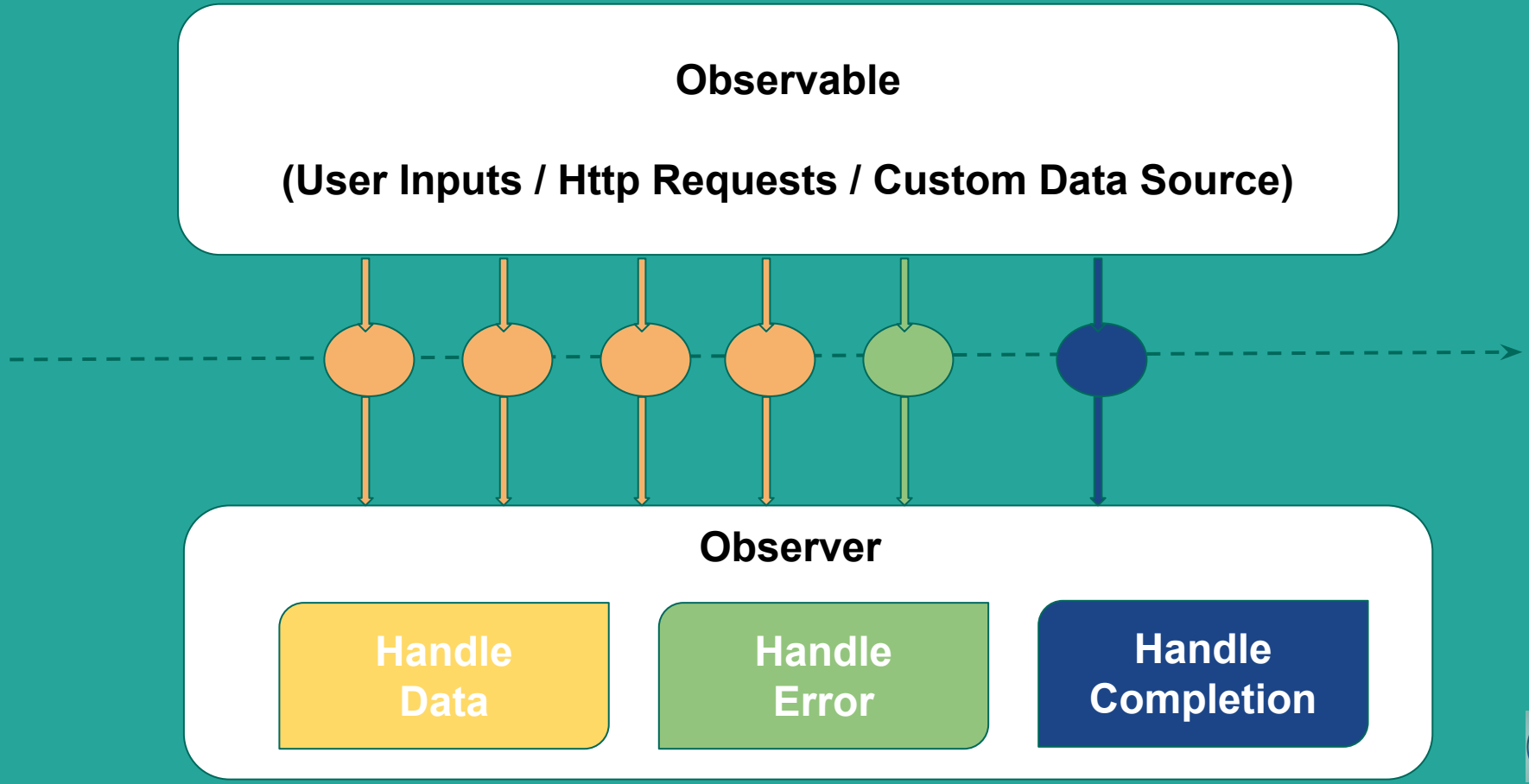
1. Create service class
   Create a class with one or more properties and methods that contains business logic and data access logic.
2. Make ready the service for Dependency Injection:
   Add @Injectable() decorator above the service class
3. Provide the service Globally / Locally:
   Add providedIn: “root” option in @Injectable( ) Decorator.[or]
   Add Providers: [Service] in AppModule’s metadata.[or]
   Add providers:[service] in any other module’s metadata.[or]
   Add providers: [Service] in any other component’s metadata
4. Inject the service into actual component
   Add @Inject(Service) private referenceVariable: Service in any component’s constructor
   Add private referenceVariable : Service in any component’s constructor



What is Observable and Observer?

- "Observable and Observer" is a pattern of "message passing" from "publisher" to "subscriber"
- Flow of functionality:
  - Observable is created.
  - Observer subscribes to the Observable.
  - Observable can pass messages (notifications) to the Observer
  - Each time, when the Observable passes a notification, it is received by Observer
- Real-time usage of Observables and Observer:
  - while receiving response from AJAX/API
  - while performing large tasks in client (browser).
- Observables execute only when the observer subscribes to it.

How “Observable and Observer” works?



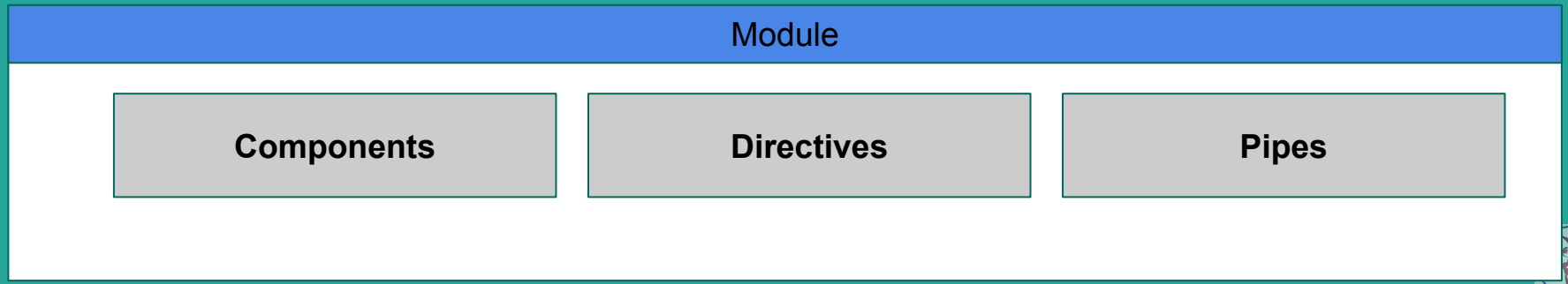
What is Angular Module

Module is a collection of Components, directives, pipes.

Mainly used to organize the components and others (directives and pipes)

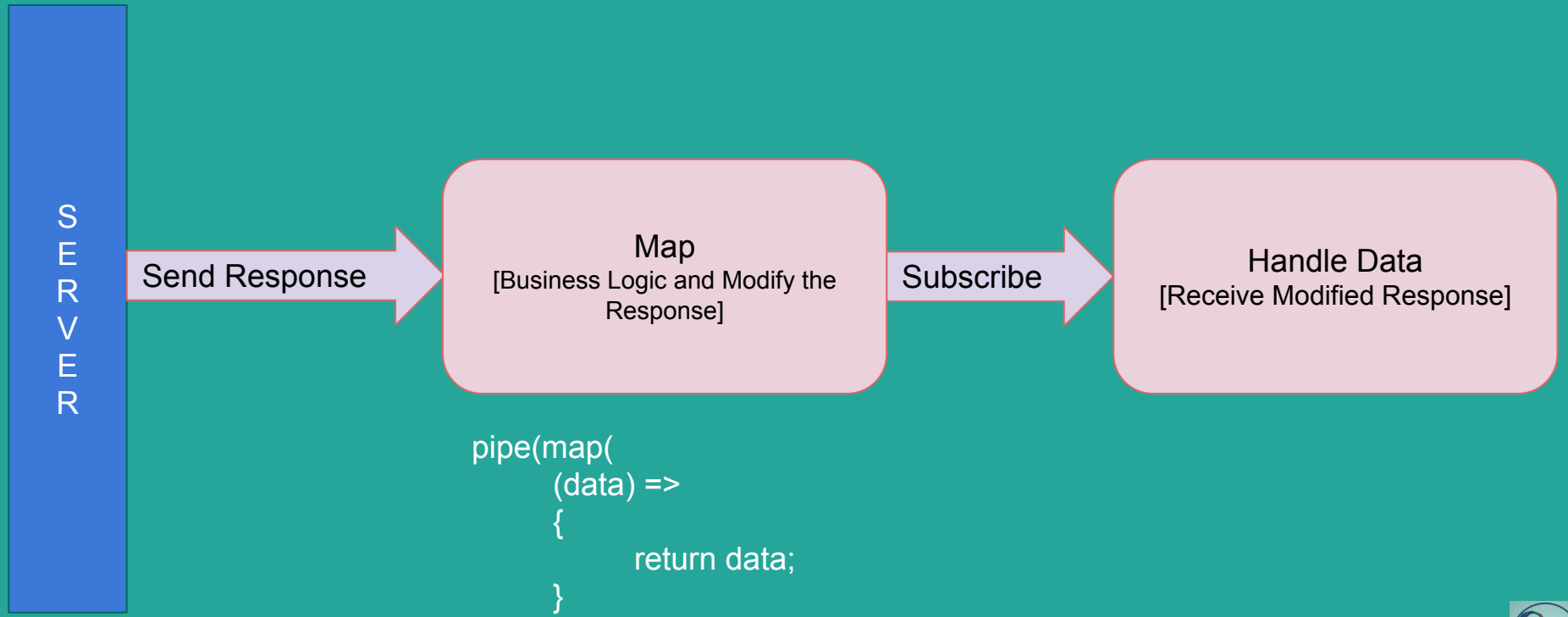
Goals of Modules:

- Consolidate components, directives, pipes into cohesive of functionality.
- Make some of the components, directives, pipes public; so that, other module's component template can use them.
- Import components,directives, pipes from other modules, that required by current module's component templates
- Provide services that the other modules can use



RxJs - Map

This "Map" is an RXJS Operator, which executes a function after receiving response from the server




```
pipe(map(
        (data) =>
        {
                return data;
        }
        ))
```

Authentication with ASP.Net Core

Models\Project.cs

Model class for Project table

ViewModels\LoginViewModel.cs

Model class for Login View

ServiceContracts\IUserService.cs

Interface for Users Service

Services\UserService.cs

Service for user authentication

Controllers\AccontController.cs

controller for user authentication

Controllers\ProjectsController.cs

Controller for CRUD operations of "Projects" table

Controllers\HomeController.cs

Controller for serving home/index page at startup

Identity\ApplicationUser.cs

Extends "IdentityUser" class, and acts as model class for users information

Identity\ApplicationRole.cs

Extends "IdentityRole" class, and acts as a model class for users roles information

Identity\ApplicationUser Store.cs

Extends "UserStore" class, and provides methods for storing users information

Identity\ApplicationRoleStore.cs

Extends "RoleStore" class, and provides methods for storing user roles information

Identity\ApplicationUserManager.cs

Extends "UserManager" class, and provides methods for manipulating users information

Identity\ApplicationRoleManager.cs

Extends "RoleManager" class, and provides methods for manipulating user roles information

Identity\ApplicationSignInManager.cs

Extends "SignInManager" class, and provides methods for user login

Identity\ApplicationDbContext.cs

Extends "IdentityDbContext" class, and contains other DbSet's for all the database tables, that you want to interact with.

Views\Home\index.cshtml

Startup view

appSettings.json

Contains application configuration settings such as connection strings, secret keys etc

Startup.cs

Contains "ConfigureServices" and "Configure" methods to add necessary services to the application

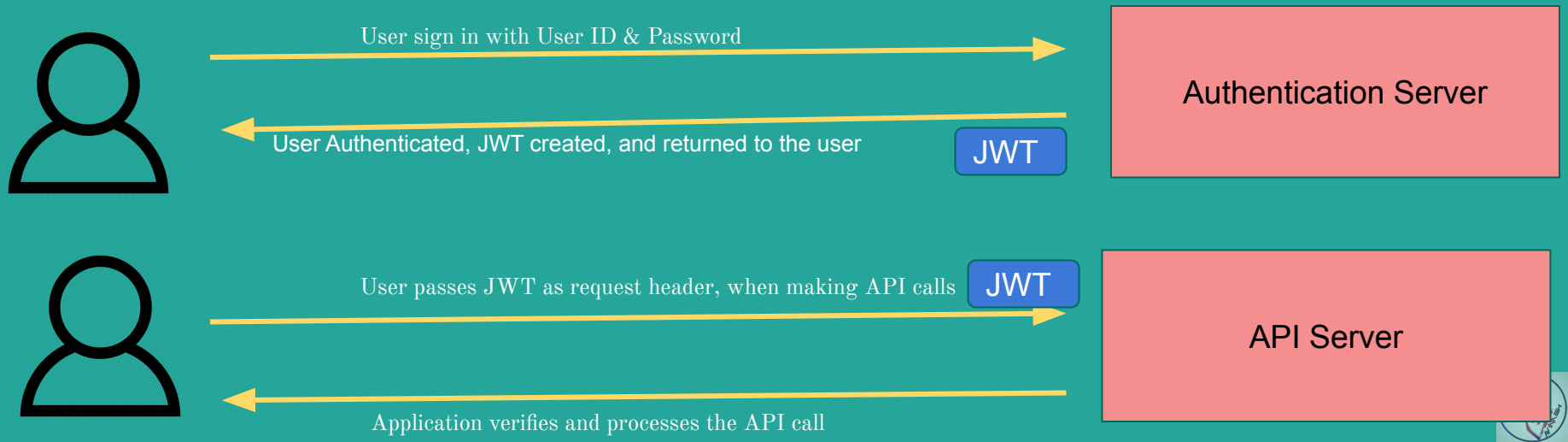
What is JSON Web Token(JWT)

A JSON Web Token (JWT) is a JSON object that is defined in RFC 7519 as a safe way to represent a set of information between two parties

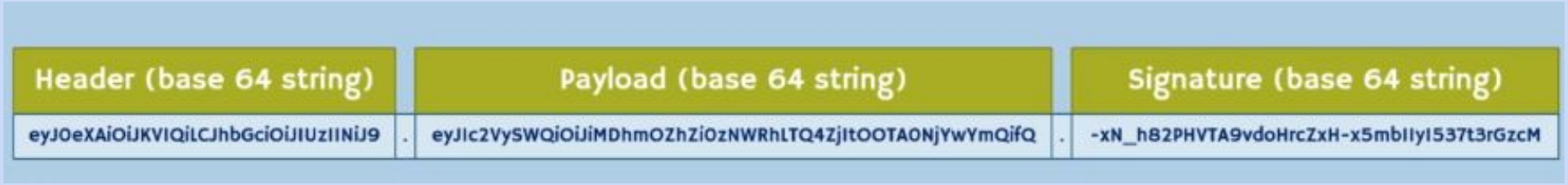
RFC (Request For Comments) is a formal document released by IEFT(Internet Engineering Task Force),Who releases Internet Standards

How JSON web Token Works

- The most common scenario of JWT is authentication.
- Once the user logs-in, JWT (an encrypted token) will be sent to the client.
- Each subsequent request includes JWT sent to the server, then the server validates JWT and provides response only when the JWT token is valid.
- The signature is generated based on the header and payload, so that receiver can verify the content hasn't been tampered with



Contents of JWT Web Token



JWT Token Generation Algorithm

Inputs	Header	PayLoad	SecretKey

	{
	"typ":"JWT",
	"alg":""HS256"
	}
		{
		"userId":"705f24034b7745a680c66704518de31a"
		}
			MySecret

Algorithm

Hashing Algorithm

- HMAC + SHA256
- RSASSA + SHA256
- ECDSA + SHA256

```
data = base64Encode(header)+"."+base64Encode(payload)

hashedData = hasg(data, secret)

signature = base64encode(hashedData)

jwtToken = data+"."+signature
```

Example

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9	eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ	SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

How JWT is verified (in subsequent request)

Header (base64 string)		Payload (base64 string)		Signature (base64 string)
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9	.	eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ	.	SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

Verification Algorithm

data = receivedHeader+"."+receivedPayload

hashedData = hash(data, secret)

signatureForVerification = base64encode(hashedData)

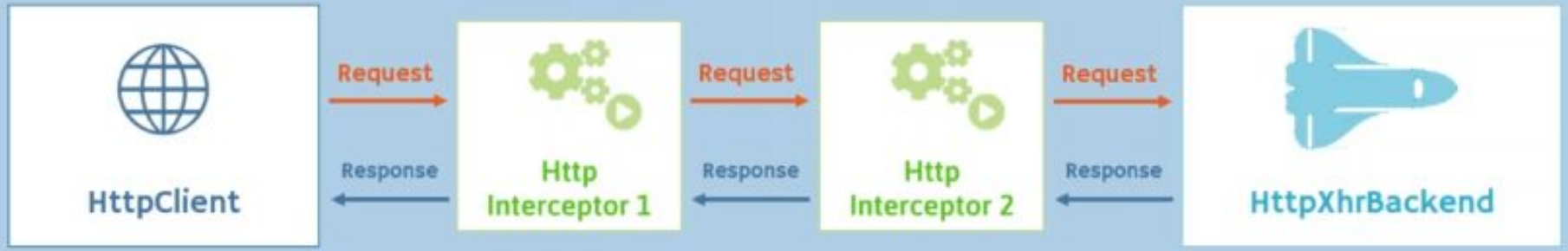
signatureForVerification == receivedSignature

Best Practices for JWT Authentication

- Don't include sensitive user information, such as Password, PIN etc., in the Payload; User Id and Timestamp is recommended for Payload
- Don't include too much information in the Payload, as it increases length of the JWT token
- Since JWT are signed and encoded only, and since JWT are not encrypted, JWT do not guarantee any security for sensitive data
    - If you require, you can encrypt the header and payload, before starting the JWT algorithm
- Always enable HTTPS, while using JWT, because HTTPS provides end-to-end encryption
- It is not recommended to store JWT token in cookies, since cookies are accessible by attackers

HTTP Interceptors

- Http Interceptor is a class, that executes as middleware, between the HttpClient and HttpXhrBackend.



HttpClient is a Service, used to create http requests.

HttpXhrBackend is a low-level Angular API, which creates and send actual XMLHttpRequest.

Class to use while working with Interceptors

- **HttpInterceptor** Interface
  - Represents Interceptor that handles HttpRequest and HttpResponse.
- **HttpRequest** Class
  - Represents a full outgoing request, with request headers, body, parameters etc.
- **HttpResponse** Class
  - Represents a full incoming response (200), with response headers, body, statusCode, statusText etc.
- **HttpErrorResponse** Class
  - Represents a full incoming error response (400), with response headers, body, statusCode, statusText etc.
- **HttpHandler** Class
  - Transforms a HttpRequest into a stream of HttpEvent's.
- **HttpEvent** Alias Name
  - Represents different events that occur after sending request, such as HttpSentEvent, HttpHeaderResponse, HttpResponse, HttpProgressEvent

What is Guard

- Guard is a service, which can tell the royter whether the current user can navigate to a specific route, or not
- Guards automatically execute before entering to a route and before leaving the route

The user is in Current Route
Ex:/dashboard

Check whether the user can leave the current route or not; and return true or false

Check whether the user can navigate to the requested route or not; and return true or false