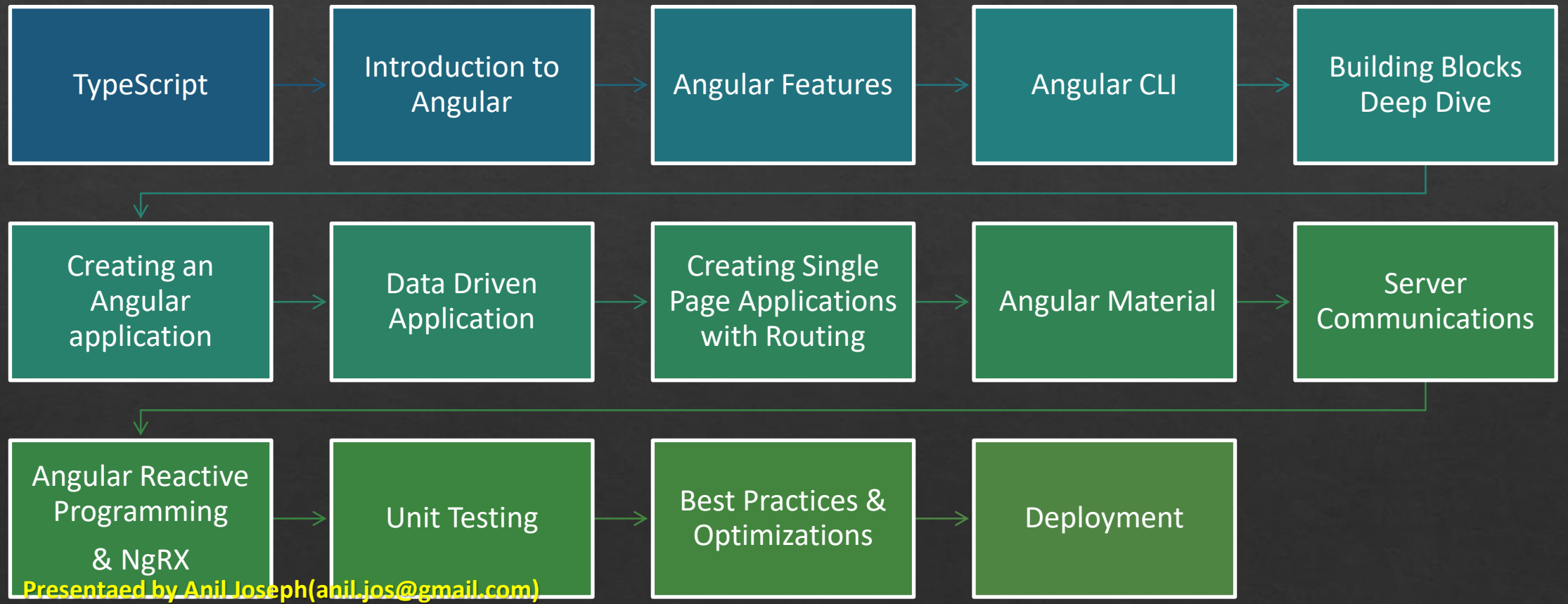
ANIL JOSEPH

Presentaed by Anil Joseph(anil.jos@gmail.com)

Agenda



Presentaed by Anil Joseph(anil.jos@gmail.com)

Introduction

Anil Joseph

- Over 20 years of experience in Training and Development
- Technologies
  - C++
  - Java
  - .NET and .NET Core
  - Node, Node Express and other frameworks
  - UI Technologies: React, Angular, ExtJS, jQuery, Knockout, Redux etc
  - Mobile: Native Android and React Native
- Worked on numerous projects
- Conducted training for corporates(700+)

Presentaed by Anil Joseph(anil.jos@gmail.com)

Software

NodeJs and npm

- Version 12 or higher

Angular CLI

- npm install @angular/cli -g

Visual Studio Code

Browsers

- Chrome, Firefox

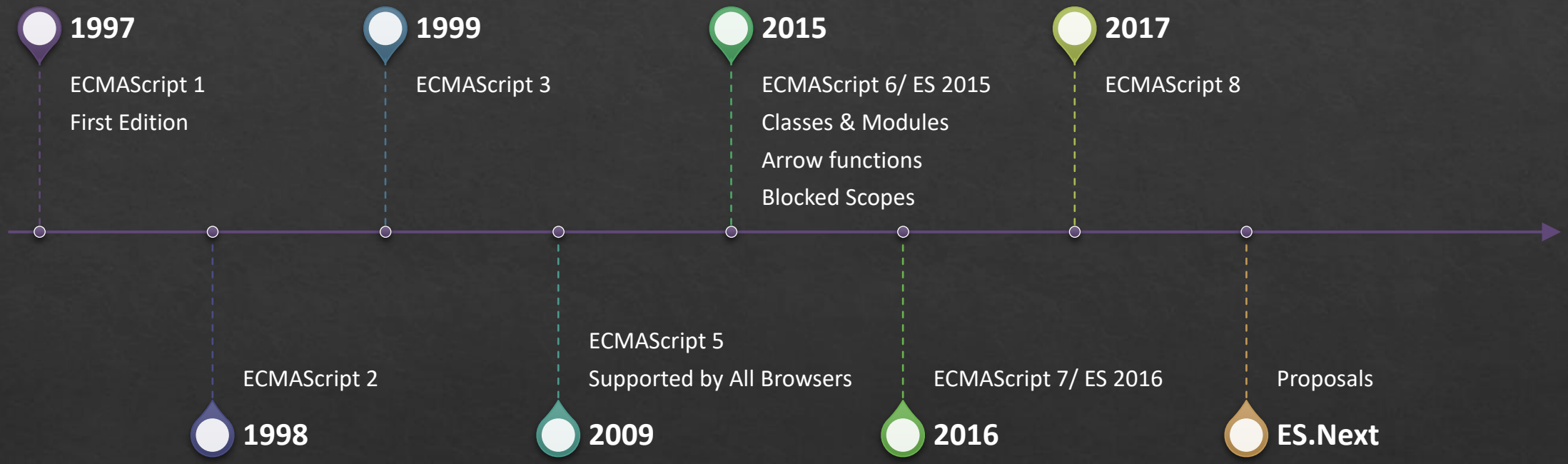
Presentaed by Anil Joseph(anil.jos@gmail.com)

JavaScript

- JavaScript (JS) is an interpreted programming language.
- It’s a dynamic language.
- Supports object-oriented programming.
- All Web browsers have a JavaScript Engine. In the browser JavaScript is used
  - For executing Client-side scripts to interact with the user.
  - Alter the document content that is displayed.
  - Control the browser.
  - Communicate asynchronously.
- Server-side JavaScript is available with Nodejs
- JavaScript was developed by Brendan Eich at Netscape.
- Released in September 1995

Presentaed by Anil Joseph(anil.jos@gmail.com)

ECMAScript Versions



Presentaed by Anil Joseph(anil.jos@gmail.com)

TypeScript

Presentaed by Anil Joseph(anil.jos@gmail.com)

TypeScript

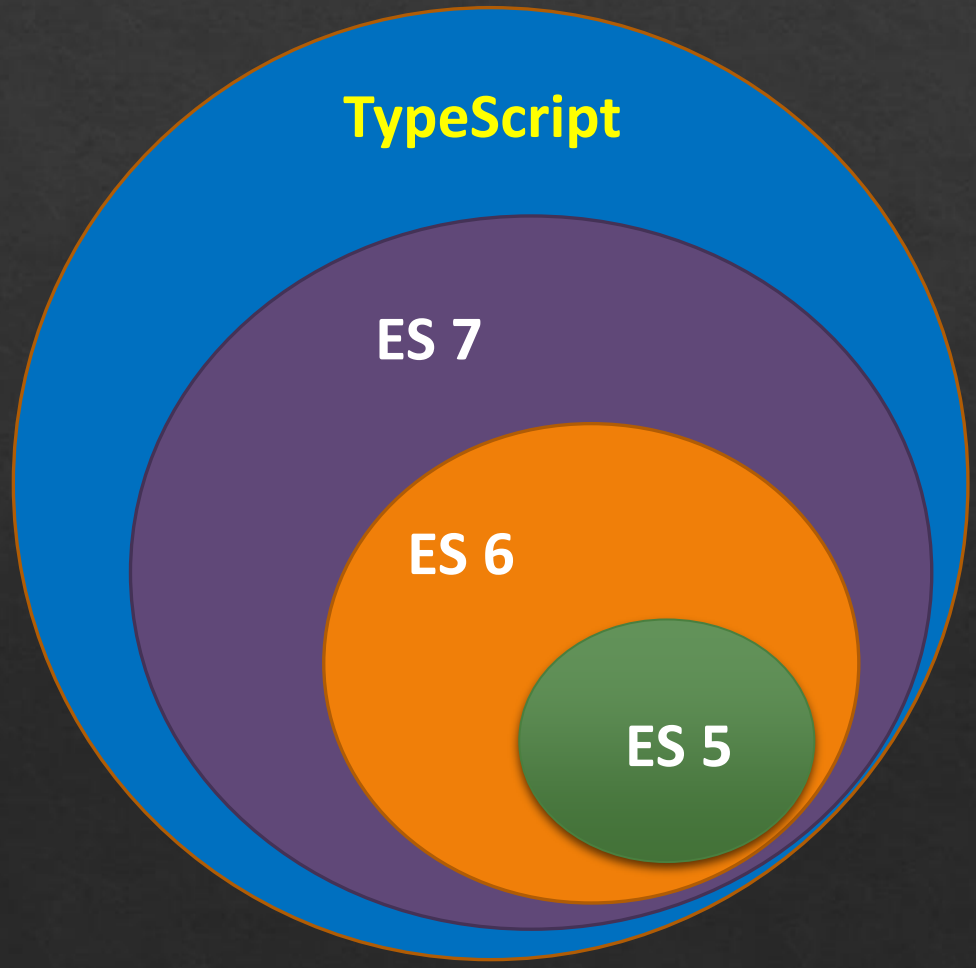
TypeScript is programming language developed and maintained by Microsoft.

TypeScript is a typed superset of JavaScript.

Transcompiles to JavaScript.

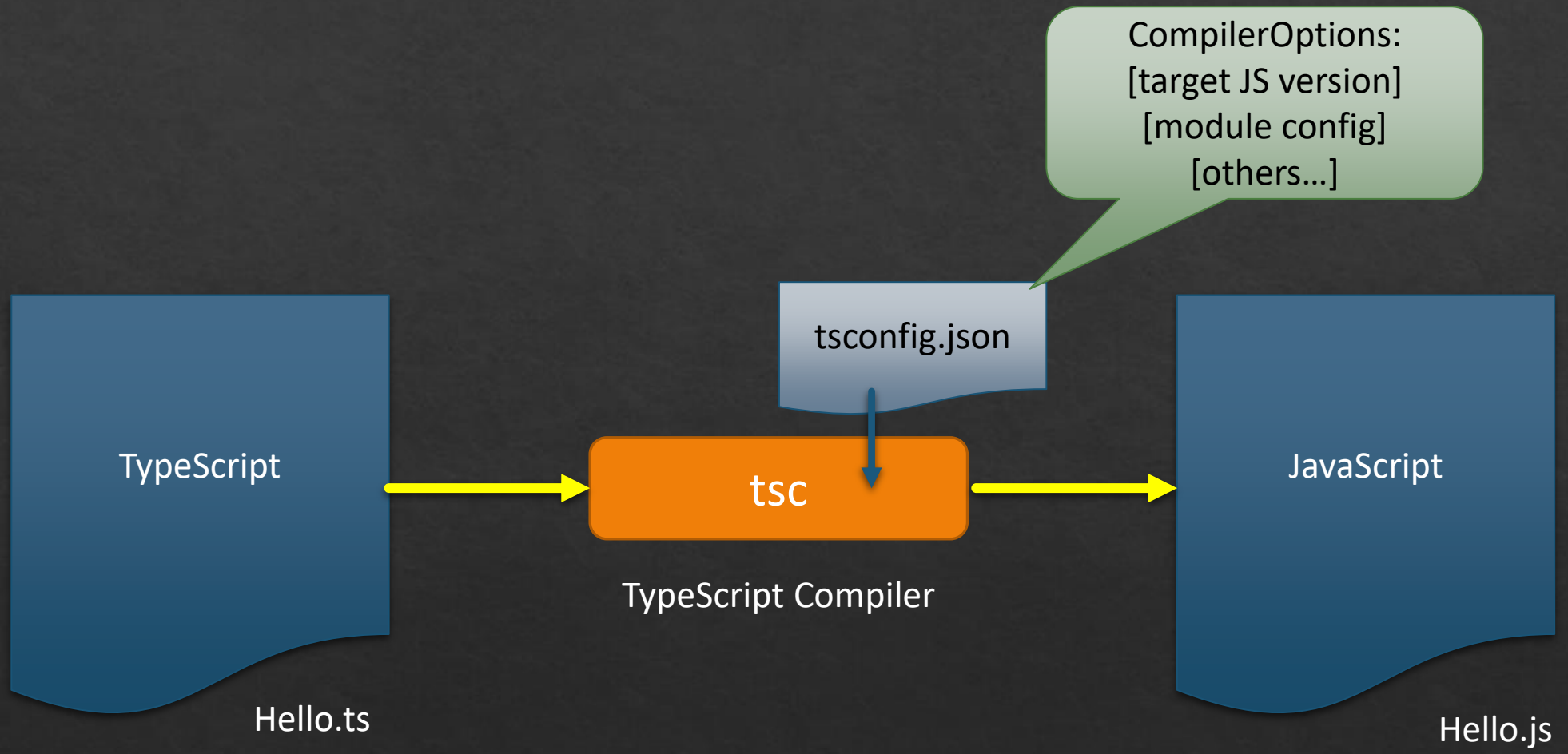
Designed for development of large applications.

Open Source.



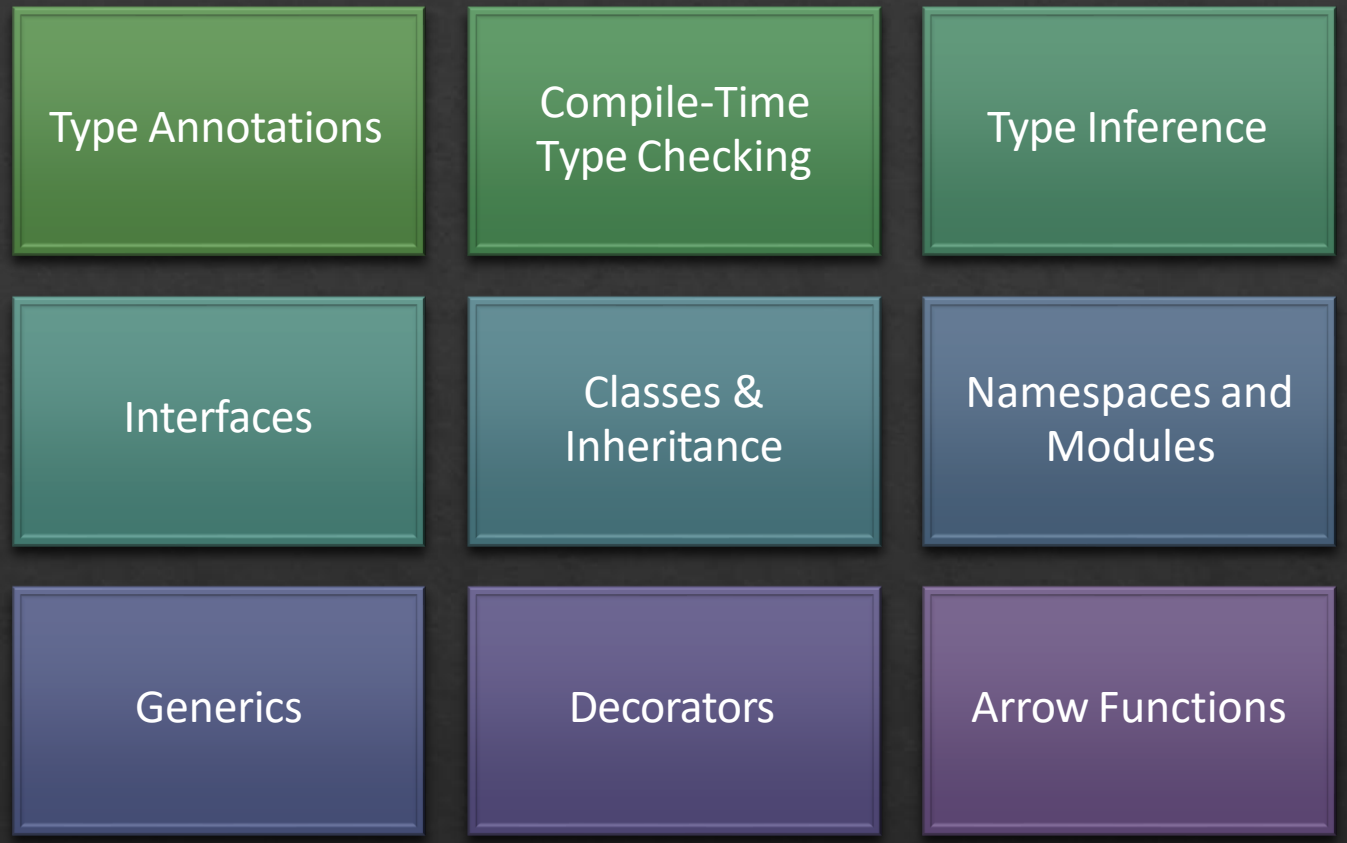
Presentaed by Anil Joseph(anil.jos@gmail.com)

TypeScript



Presentaed by Anil Joseph(anil.jos@gmail.com)

TypeScript Features



TypeScript Installation

Node.js

- npm install -g typescript

Available with

- Visual Studio 2015 & 2017
- Visual Studio Code

Plugins available with

- Eclipse
- Atom
- Sublime
- WebStrome
- Many more

TypeScript Types

Boolean

- **let** isAvailable: boolean = false;

Number

- **let** age: number = 16;
- **let** hex: number = 0xf00d;

String

- **let** name: string = “Anil";

Array

- **let** list: number[] = [1, 2, 3];
- **let** list: Array<number> = [1, 2, 3];

Presentaed by Anil Joseph(anil.jos@gmail.com)

TypeScript Types

Enum

- **enum** Color {Red, Green, Blue}
- **let** c: Color = Color.Green;

Any

- **let** x: any = 4;
- x = “hello”

Tuple

- let data: [number, string];
- data = [1, "Anil"];

Presentaed by Anil Joseph(anil.jos@gmail.com)

TypeScript Types

void

```
function foo(): void {
    console.log(“foo");
}
```

Null and Undefined

- var x: string = null;
- var y:string= undefined

Interfaces

- Interfaces are a powerful way of defining contracts.
- Example

```
interface Vehicle{

    name: string;
    speed: number;
    gear?: number;

    applyBrakes(decrement: number): void;
}
```

- Interfaces can extend interfaces
  - (keyword extends)
- Classes implements interfaces
  - (keyword implements)

Presentaed by Anil Joseph(anil.jos@gmail.com)

Classes

- Traditional JavaScript uses functions and prototype-based inheritance to build up reusable components.
- Starting with ECMAScript 2015, also known as ECMAScript 6, JavaScript introduces the object-oriented class-based approach.
- TypeScript supports classes that compile down to JavaScript
  - Works across all major browsers and platforms
  - Without having to wait for the next version of JavaScript.

Classes

- Example

```
class Car implements Vehicle{
    constructor(public name: string, public speed: number, public
    gear: number){

    }
    applyBrakes(decrement: number): void{

    }
}
```

- Modifiers supported
  - public, private, protected
- Can have constructors
- Supports Properties
- Supports static members
- Supports inheritance
- Classes and methods can be abstract

Arrow Functions

Represents a function expression

An arrow function expression has a shorter syntax than a function expression

They do not receive the implicit arguments “this” and “arguments”

Used widely for asynchronous and functional programming

TypeScript: Modules

- Starting with the ECMAScript 2015, JavaScript has a concept of modules.
- TypeScript shares this concept.
- Modules are executed within their own scope
  - Not in the global scope

Modules

- Use the import and export statements.

```
let foo = function(){
   //some code
}


export default foo;
```

one.js

```
import foo from ‘./one’;

foo();
```

two.js

```
import bar from ‘./one’;

bar();
```

three.js

Presentaed by Anil Joseph(anil.jos@gmail.com)

Modules

```
export let foo = function(){
   //some code
}


 export let bar = function(){
    //some code
 }
```

one.js

```
import {foo, bar} from ‘./one’;


 foo();
 bar();
```

two.js

Presentaed by Anil Joseph(anil.jos@gmail.com)

NPM(Node Package Manager)

- NPM is a package manager for the JavaScript programming language.
- Allows users to consume and distribute JavaScript modules that are available on the registry
- The default package manager for Node.
- Comprises of
  - Command line client, also called npm
  - An online database called the npm registry.
    - Public
    - Paid-for private packages
  - NPM website
- The package manager and the registry are managed by npm, Inc.

NPM Packages

- Packages are reusable code published in the npm registry.
- A directory with one or more files a metadata file called package.json.
- A package is a building block that solves a specific problem.
- An application generally depends on many packages.
- Packages can be used on
  - Server side. Example: Express, Request
  - Client Side. Example Angular, React
  - Command based. Typescript, Angular CLI, JSLint

Angular

A client-side JavaScript Framework for building Web Applications

A Google Product

Developed in 2009 by Miško Hevery and Adam Abrons(Google Employees).

Open source under the MIT license.

Presentaed by Anil Joseph(anil.jos@gmail.com)

Angular Core Features

- Client-side JavaScript Framework
- Declarative UI(Extends HTML)
- Data Binding
- Modular
- Loose coupling with Dependency Injection
- Designed for Single Page Application
- Develop using JavaScript, **Typescript** or Dart
- Tools available for development
- Create for Desktop and Mobile
- Supports server-side rendering

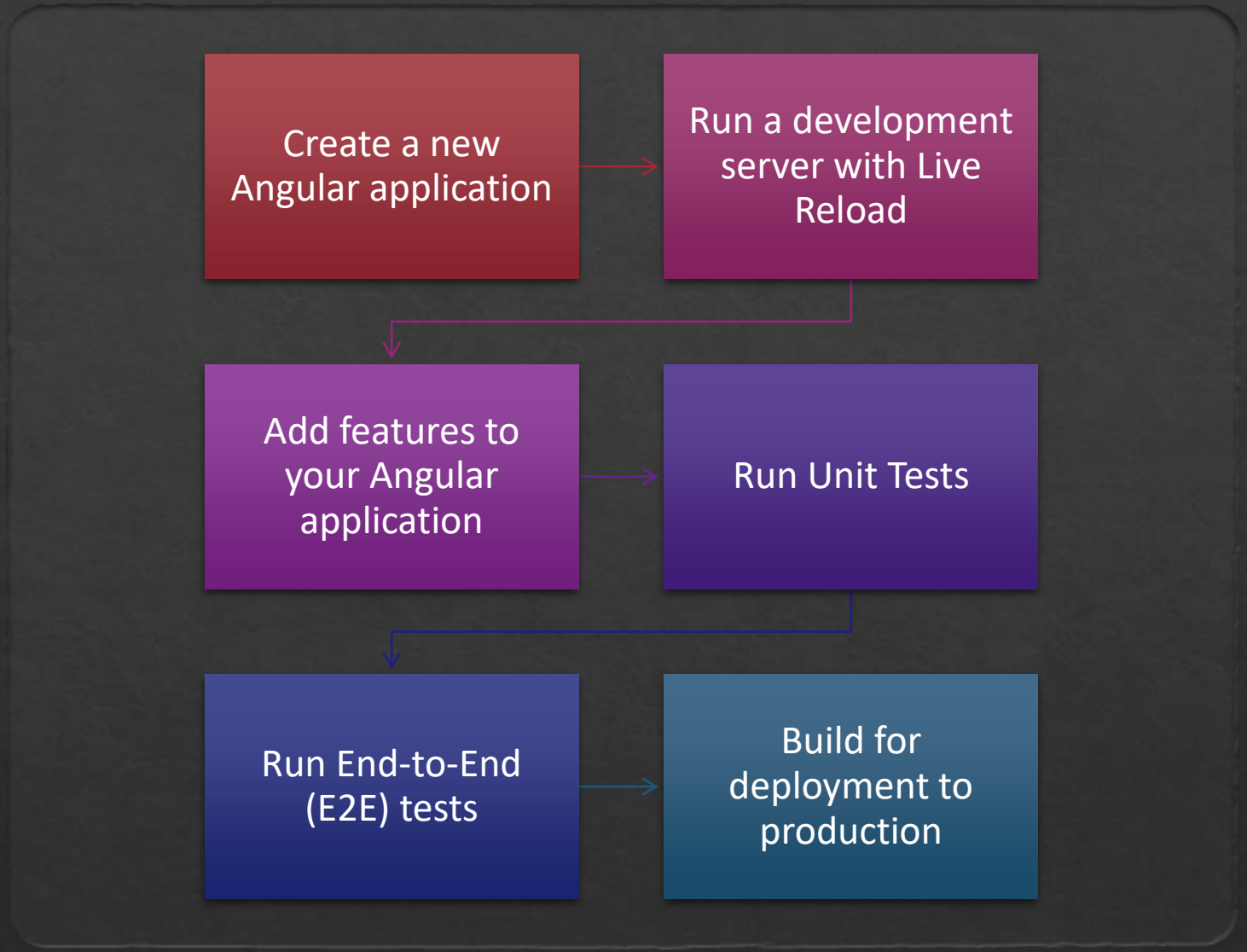
Presentaed by Anil Joseph(anil.jos@gmail.com)

Create an Angular Project

Presentaed by Anil Joseph(anil.jos@gmail.com)

Angular CLI

A command-line interface.



Presentaed by Anil Joseph(anil.jos@gmail.com)

Getting Started: CLI

1

Install Node.js and npm

- node 10 or higher

2

Install the Angular CLI globally.

- npm install –g @angular/cli@10

3

Create a new project

- ng new awesome-app

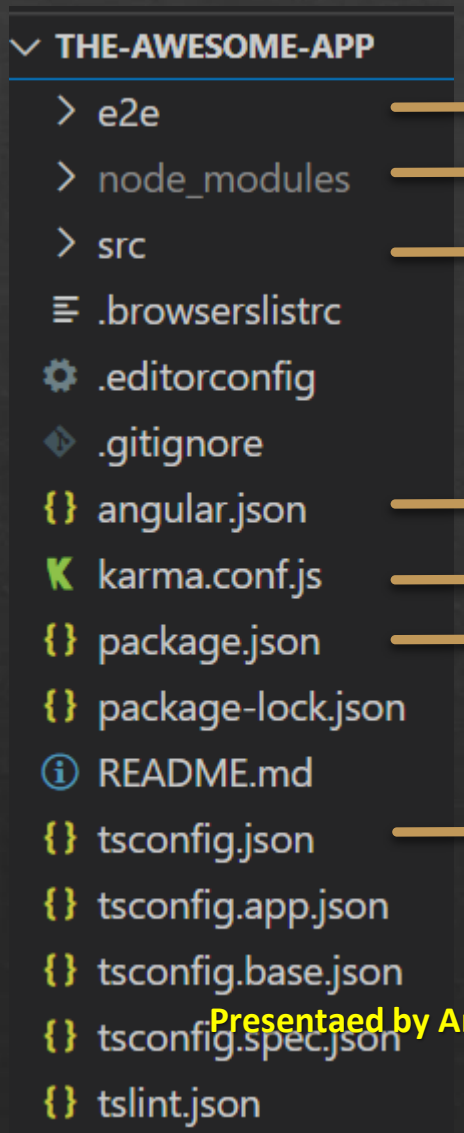
4

Serve the application

- cd awesome-app
- ng serve --open

Presentaed by Anil Joseph(anil.jos@gmail.com)

Project Files



e2e configuration

Folder contains downloaded dependencies

Folder contains all the source code

Angular CLI configuration file

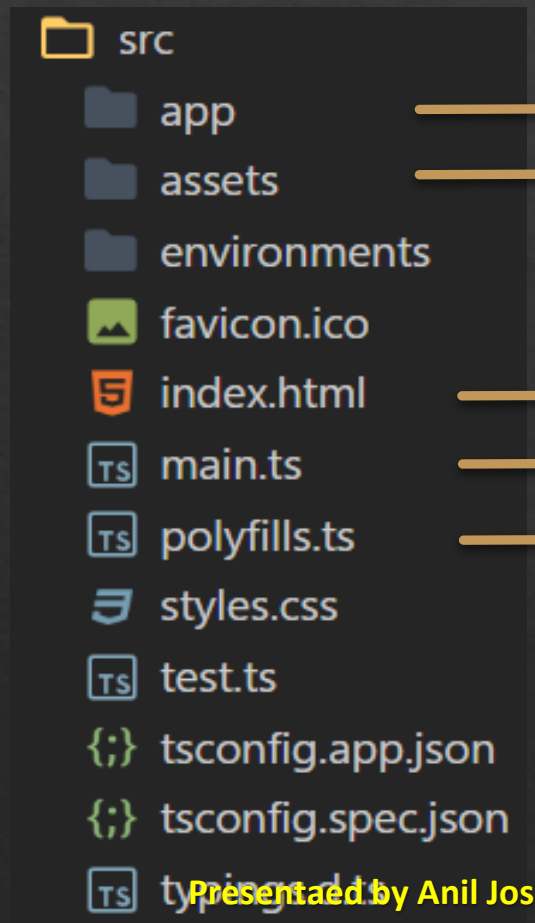
Karma configuration file, the test-runner

NPM configuration file, contains the dependencies and scripts

Typescript configuration file, contains typescript compiler options

Presentaed by Anil Joseph(anil.jos@gmail.com)

Project Files



Folder contains all the code

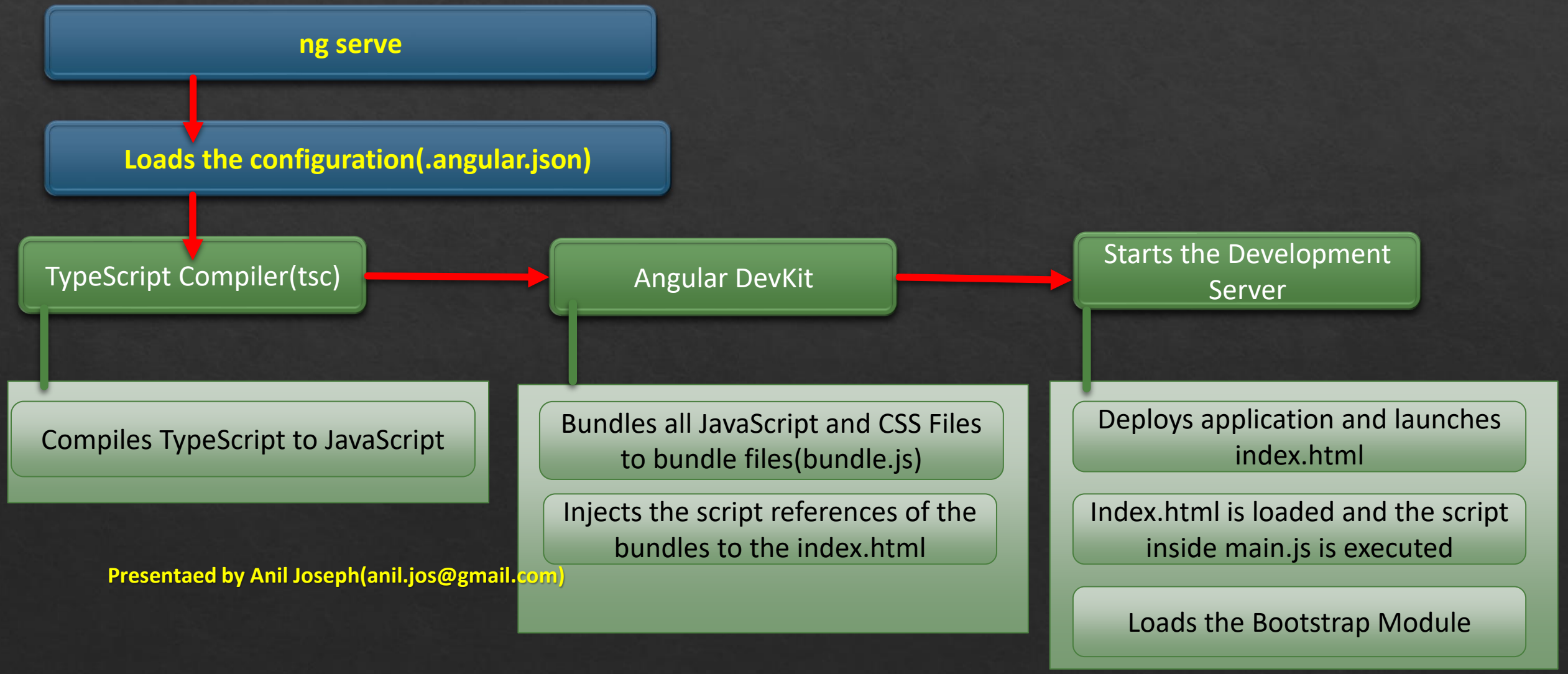
Folder contains all static contents

The page displayed on the browser(In SPA it's the only page)

The bootstrapping JavaScript

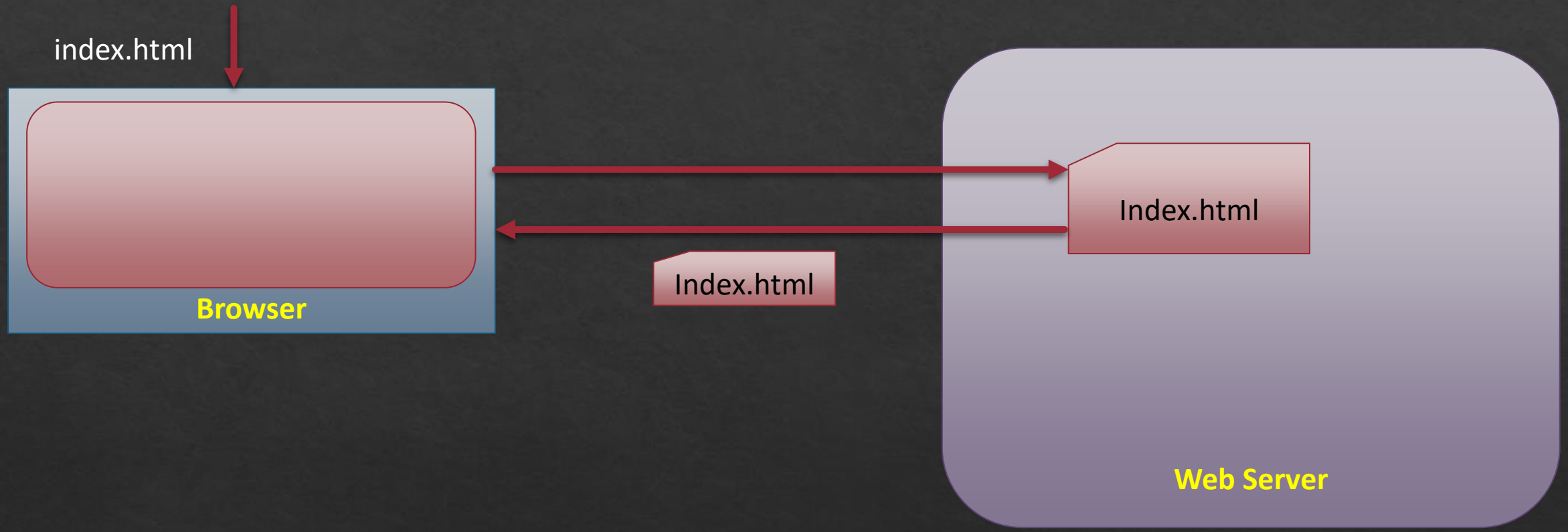
Config file for polyfills(support older browsers and IE)

Project Execution



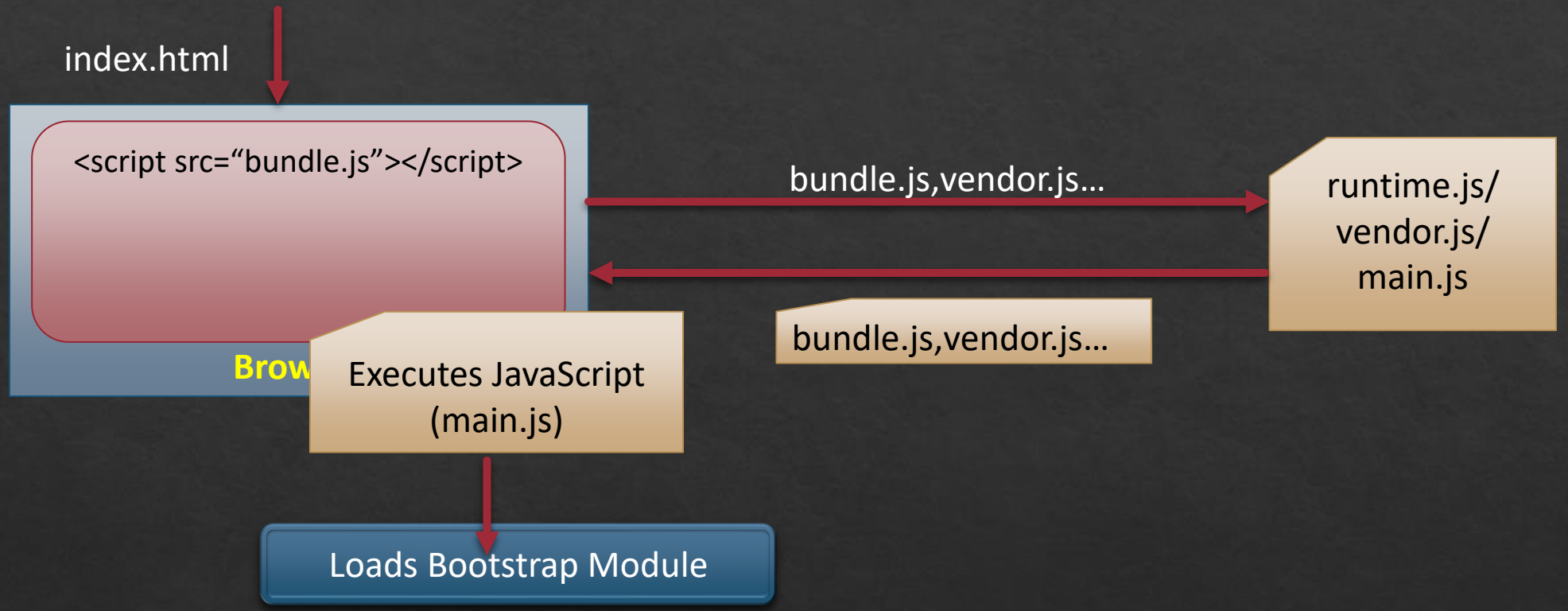
Presentaed by Anil Joseph(anil.jos@gmail.com)

Application Startup



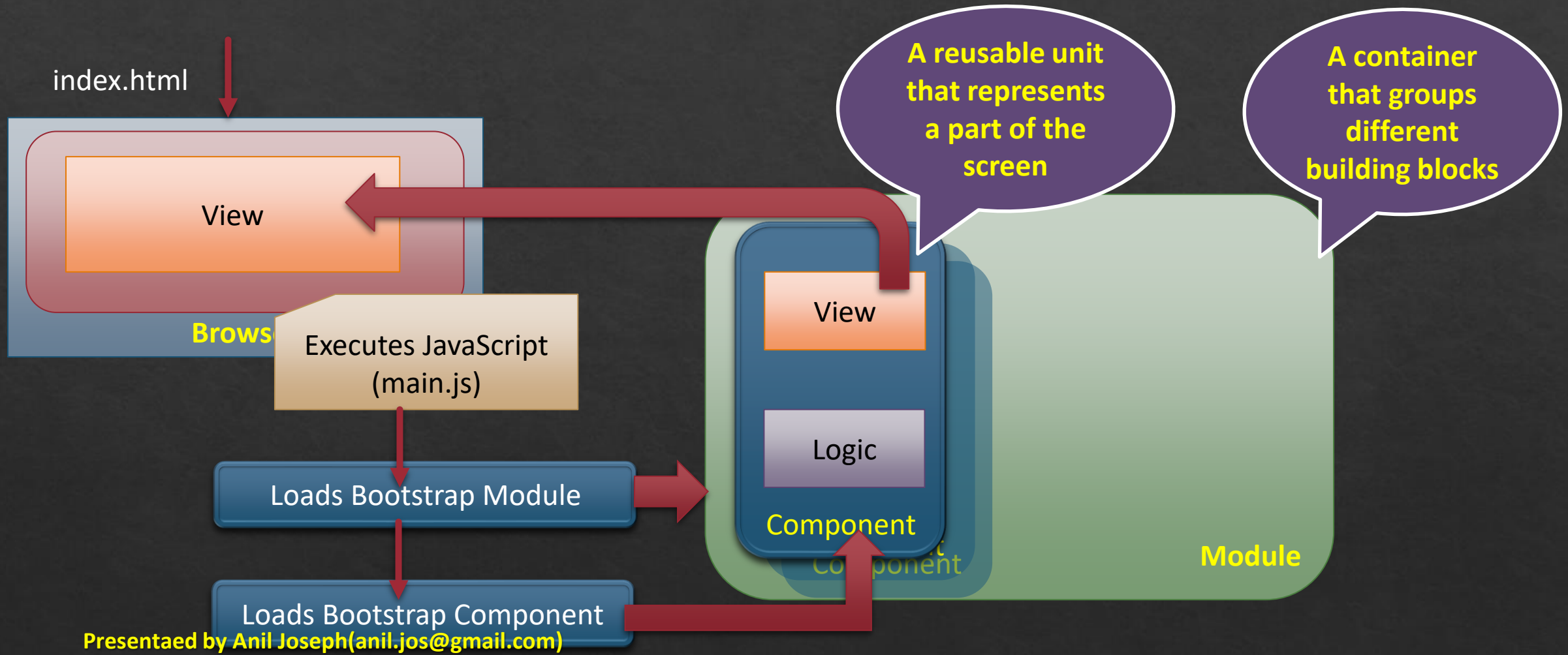
Presentaed by Anil Joseph(anil.jos@gmail.com)

Application Startup



Presentaed by Anil Joseph(anil.jos@gmail.com)

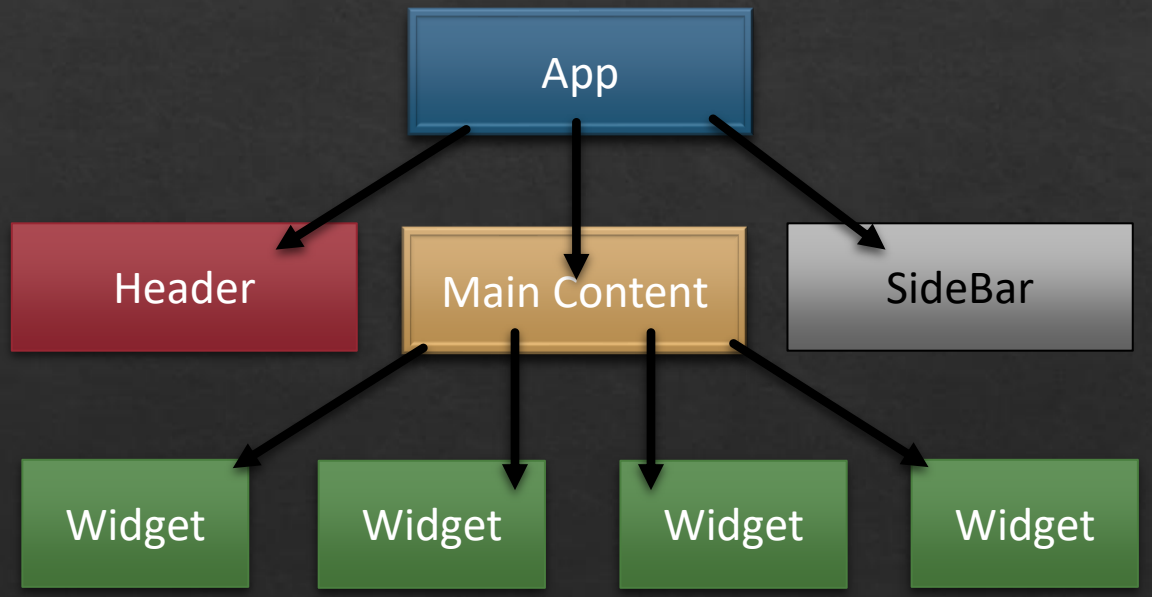
Application Startup



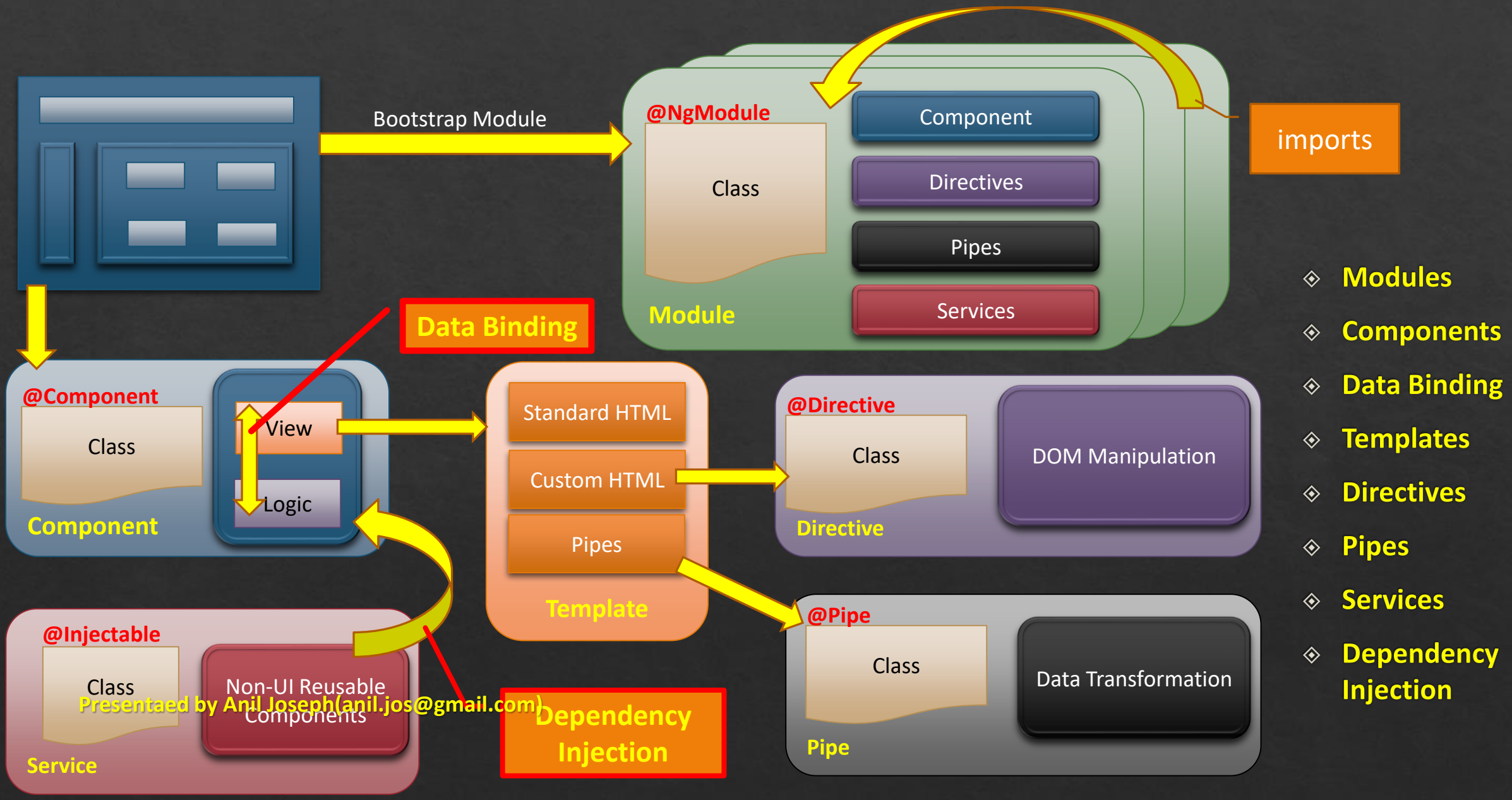
Presentaed by Anil Joseph(anil.jos@gmail.com)

Components

- Components are the **core building blocks of Angular applications**
- An Angular app can be depicted as a **component tree**



Angular Building Blocks



Angular Building Blocks

Modules

- A module is a set of similar utilities that perform a similar task.
- Contains components, services, directives etc.

Metadata

- Metadata tells Angular how to process a class.
- Also known as decorators(TypeScript)

Components

- Components are the **basic building blocks** of an Angular 2 application.
- It assembles a screen, ui-element or a route in the application.
- Comprises of the view and application logic

Presentaed by Anil Joseph(anil.jos@gmail.com)

Angular Building Blocks

Templates

- We define a component's view with its companion template.
- A template looks like regular HTML

Data Binding

- A mechanism for coordinating parts of a template with parts of a component

Directives

- Angular renders templates(it transforms the DOM) according to the instructions given by directives.

Pipes

- Pipes transform displayed values within a template.

Presentaed by Anil Joseph(anil.jos@gmail.com)

Angular Building Blocks

Services

- *Service* is a broad category encompassing any value, function, or feature that your application needs.
- Examples: Logging, DataServices

Dependency Injection

- *Dependency injection* is a way to supply a new instance of a class with the fully-formed dependencies it requires.

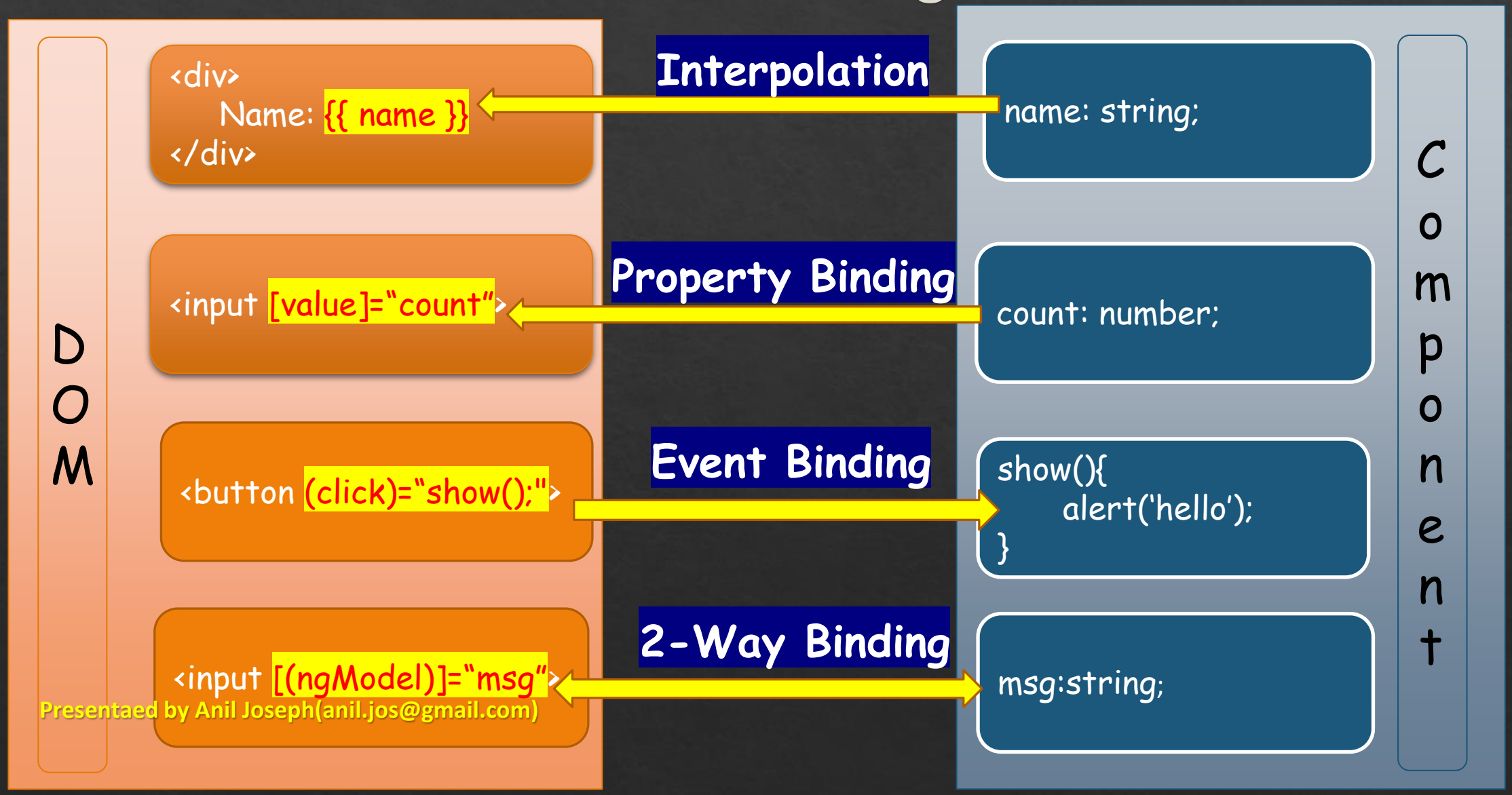
Presentaed by Anil Joseph(anil.jos@gmail.com)

Components

- A *component* controls a patch of screen called a *view*.
- Defined as class with the Component decorator
- @Component properties
  - **selector** - css selector that identifies this component in a template
  - **styleUrls** - list of urls to stylesheets to be applied to this component's view
  - **styles** - inline-defined styles to be applied to this component's view
  - **template** - inline-defined template for the view
  - **templateUrl** - url to an external file containing a template for the view

Presentaed by Anil Joseph(anil.jos@gmail.com)

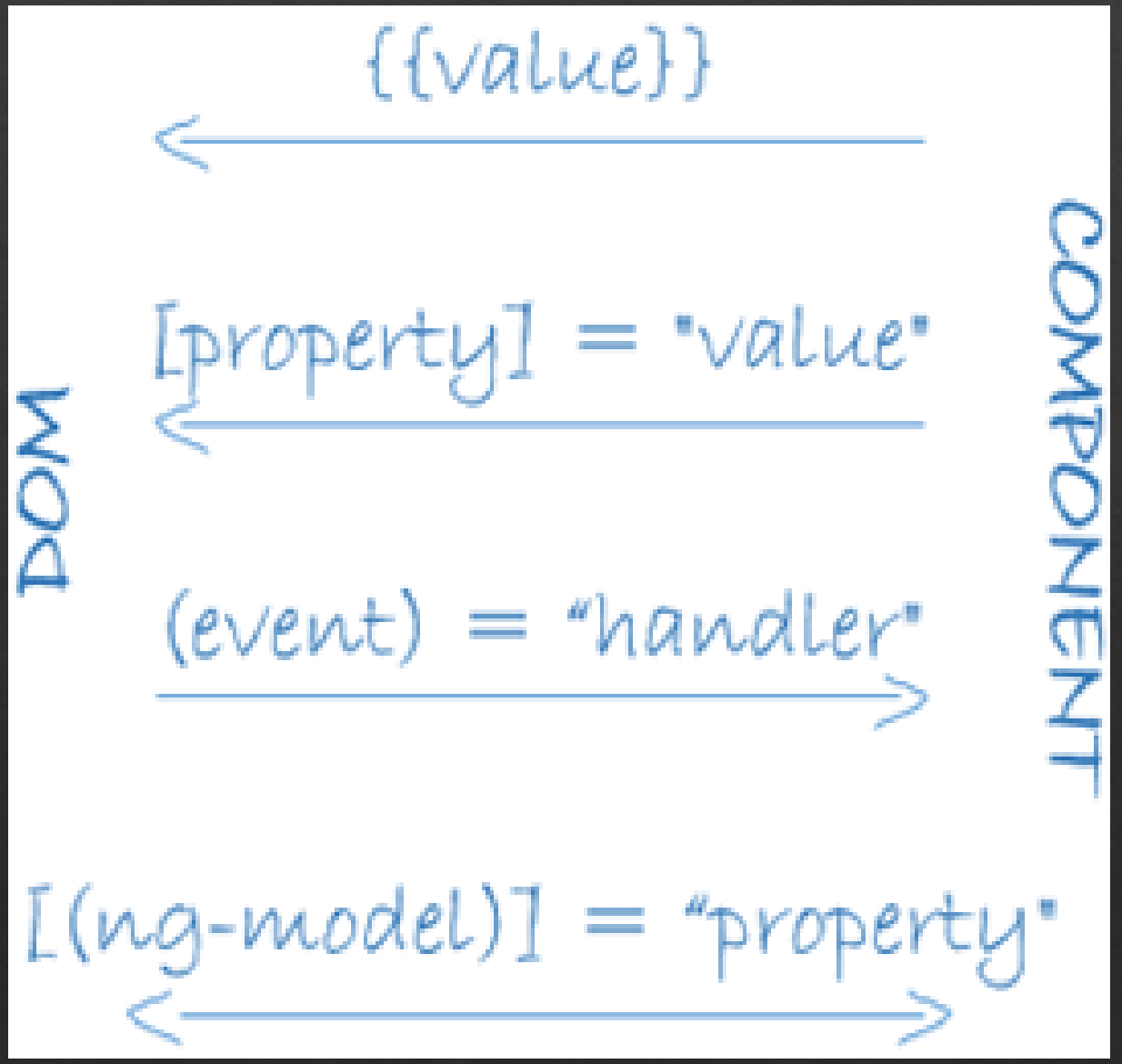
Data Binding



Presentaed by Anil Joseph(anil.jos@gmail.com)

Data Binding

- Interpolation
  - {{ message }}
- Property Binding
  - [value] = “message”
- Event Binding
  - (click) = “addData()”
- Two Way Binding
  - [(ngModel)]=“message”



Presentaed by Anil Joseph(anil.jos@gmail.com)

Modules

- Modules consolidate components, directives, services and pipes into cohesive blocks of functionality.
- Modules can be loaded eagerly when the application starts.
- They can also be *lazy loaded* asynchronously by the router.
- Every application will have a module called the root module. (This is bootstrapped from the main.js file)
- Decorate a class with @NgModule to define a module.

Presentaed by Anil Joseph(anil.jos@gmail.com)

Directives

Directives allows to manipulate the HTML DOM.

The extend HTML with custom markup.

Angular 1.x shipped with over 60 directives

Angular 2 has a smaller list of directives.

The new binding syntax eliminates the need for many directives.

Presentaed by Anil Joseph(anil.jos@gmail.com)

Directive Types

Components

- Angular 2 components are *actually* just directives under the hood
- Directives with a template.

Structural directives

- Change the DOM layout by adding and removing DOM elements.

Attribute directives

- Change the appearance or behavior of an element, component, or another directive.

Presentaed by Anil Joseph(anil.jos@gmail.com)

Structural Directives

Structural directives are responsible for HTML layout.

Typically used for adding, removing, or manipulating elements.

An asterisk (*) precedes the directive attribute name.

- <div *ngIf="hero" >{{hero.name}}

Built-in structural directives

- NgIf
- NgFor
- NgSwitch

Use <ng-container> to group elements when there is no suitable host element for the directive.

Presentaed by Anil Joseph(anil.jos@gmail.com)

Attribute Directives

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

They are usually applied to elements as if they were HTML attributes

Built-in attribute directives

- NgClass
- NgStyle
- NgModel

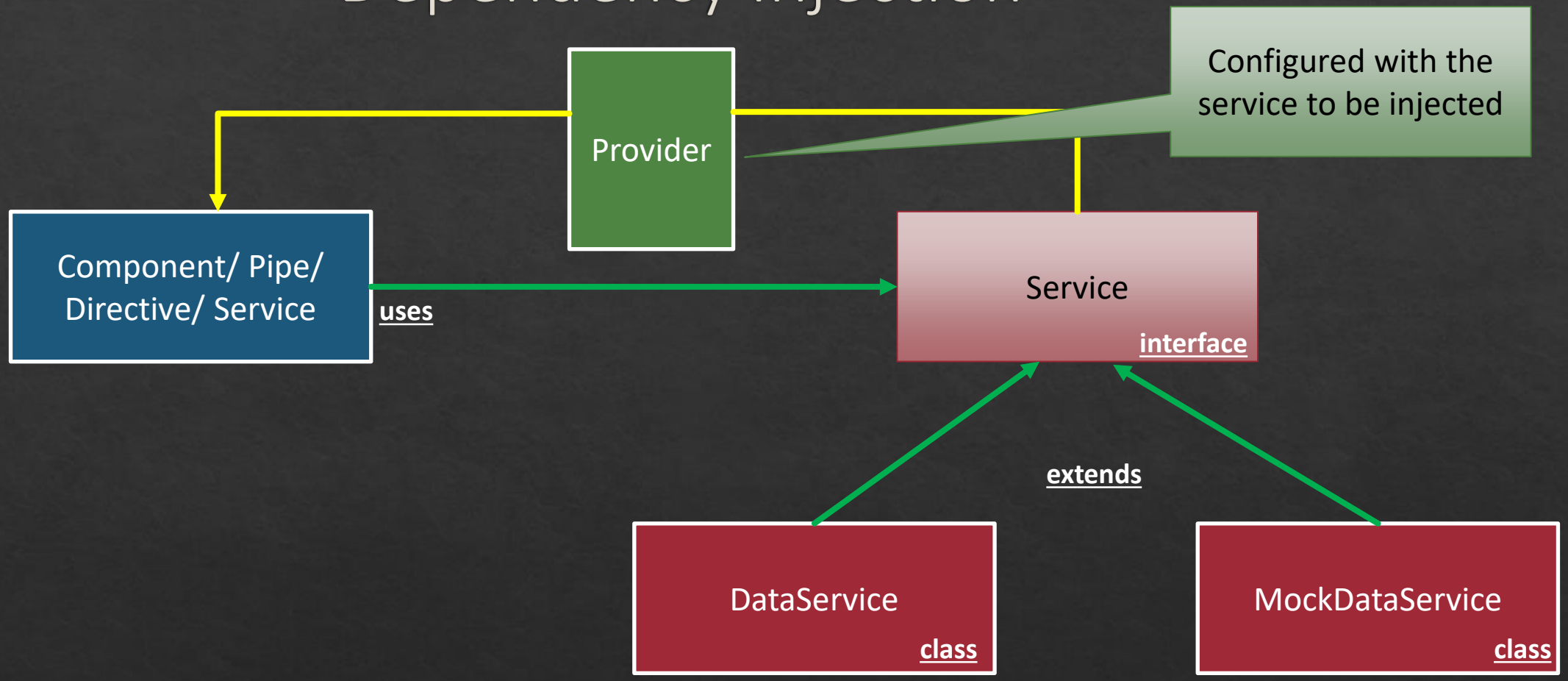
Presentaed by Anil Joseph(anil.jos@gmail.com)

Services

- Services are reusable functionalities.
- They can be injected into components, services, pipes and directives
- Implemented as Simple Classes in Angular
- Services can be configured as singletons
- They can be used to share data between components

Presentaed by Anil Joseph(anil.jos@gmail.com)

Dependency Injection



Presentaed by Anil Joseph(anil.jos@gmail.com)

Pipes

- Pipes transform displayed values within a template.
- Used inside the interpolation expression with the pipe operator (|)
  - {{ dateOfBirth | date}}
- Built-in Pipes
  - uppercase
  - lowercase
  - currency
  - percent
  - json
- Pipes can be chained togather
  - {{ dateOfBirth | date | uppercase }}
- Custom pipes.

Presentaed by Anil Joseph(anil.jos@gmail.com)

Pipes

{{ message | uppercase }}

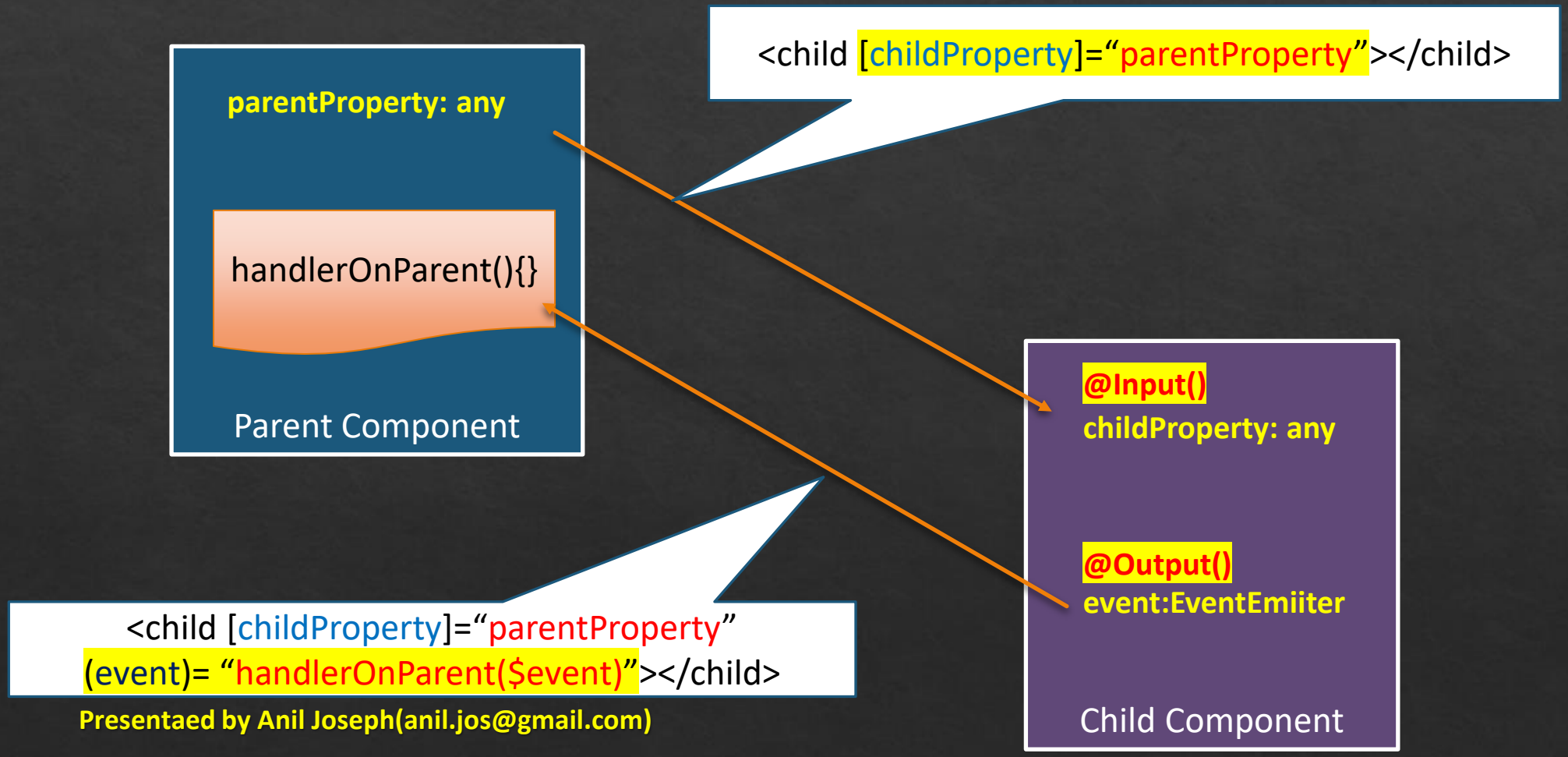
```
transform(input: String): String{
    let output;
    // some code

    return output;
}
```

Presentaed by Anil Joseph(anil.jos@gmail.com)

{{ dateOfBirth | date: 'mdy' }}

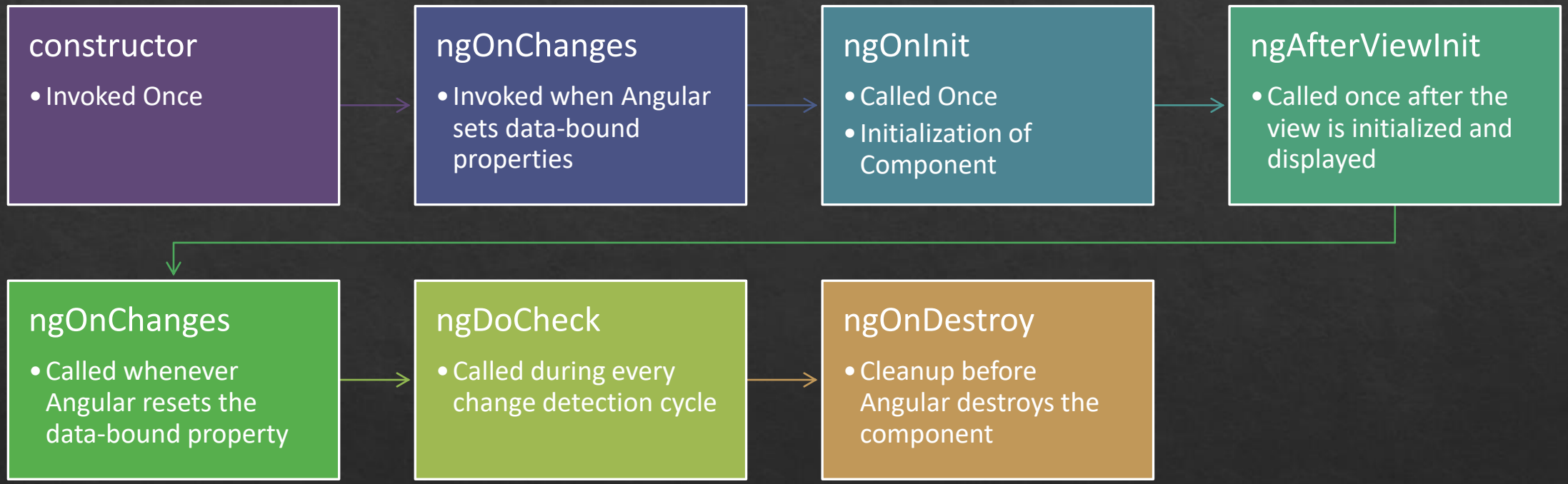
```
transform(input: Date, format: String): Date{
    let output;
    // some code

    return output;
}
```

Component Interaction(Parent-Child)



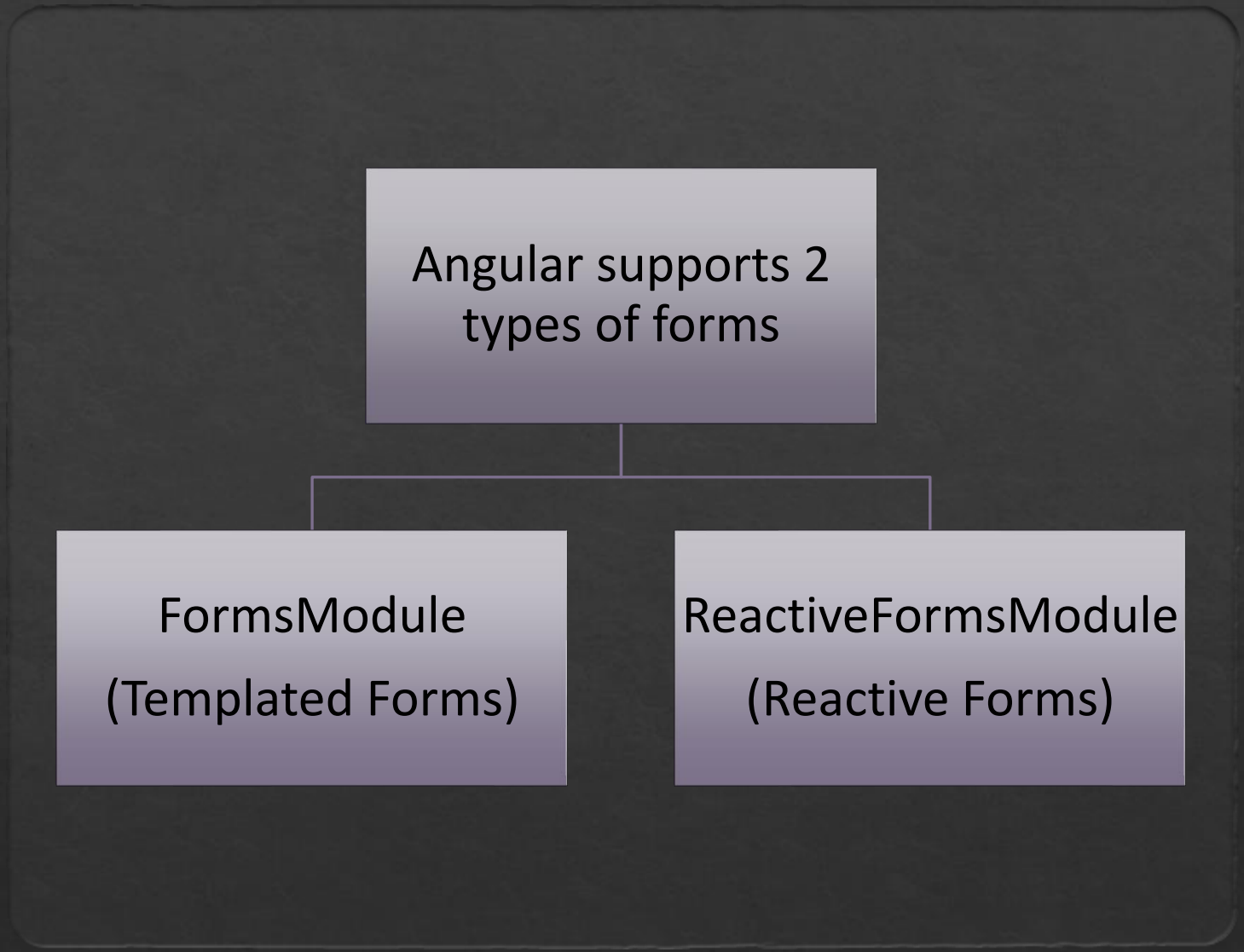
Presentaed by Anil Joseph(anil.jos@gmail.com)

Component Lifecycle



Presentaed by Anil Joseph(anil.jos@gmail.com)

Forms

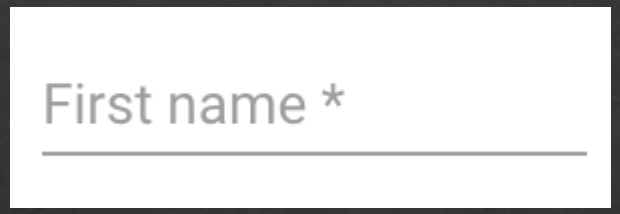


Presentaed by Anil Joseph(anil.jos@gmail.com)

Forms: Building Blocks

Presentaed by Anil Joseph(anil.jos@gmail.com)

Forms: FormControl

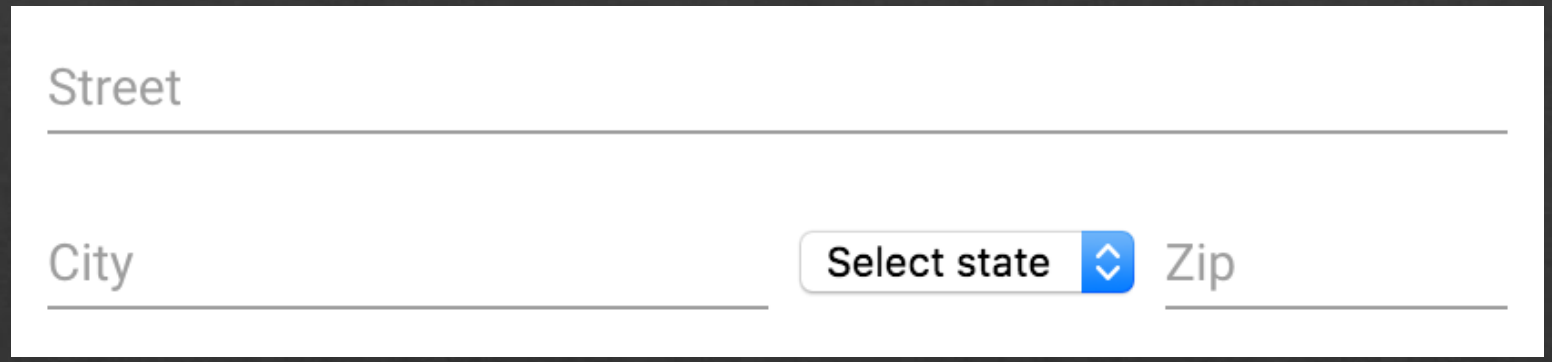


FormControl

- Value
- Validation status
- User interactions
- Events

Presentaed by Anil Joseph(anil.jos@gmail.com)

Forms: FormGroup



FormGroup

Presentaed by Anil Joseph(anil.jos@gmail.com)

Forms

Reactive Forms Module

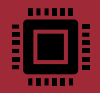
- Explicit creation of FormControls
- Source of truth: component class
- Reactive Programming Support

Forms Module

- Implicit creation of FormControl by directives
- Source of truth: template

Presentaed by Anil Joseph(anil.jos@gmail.com)

Reactive Programming



Reactive programming is a declarative programming paradigm concerned with data streams and the propagation of change

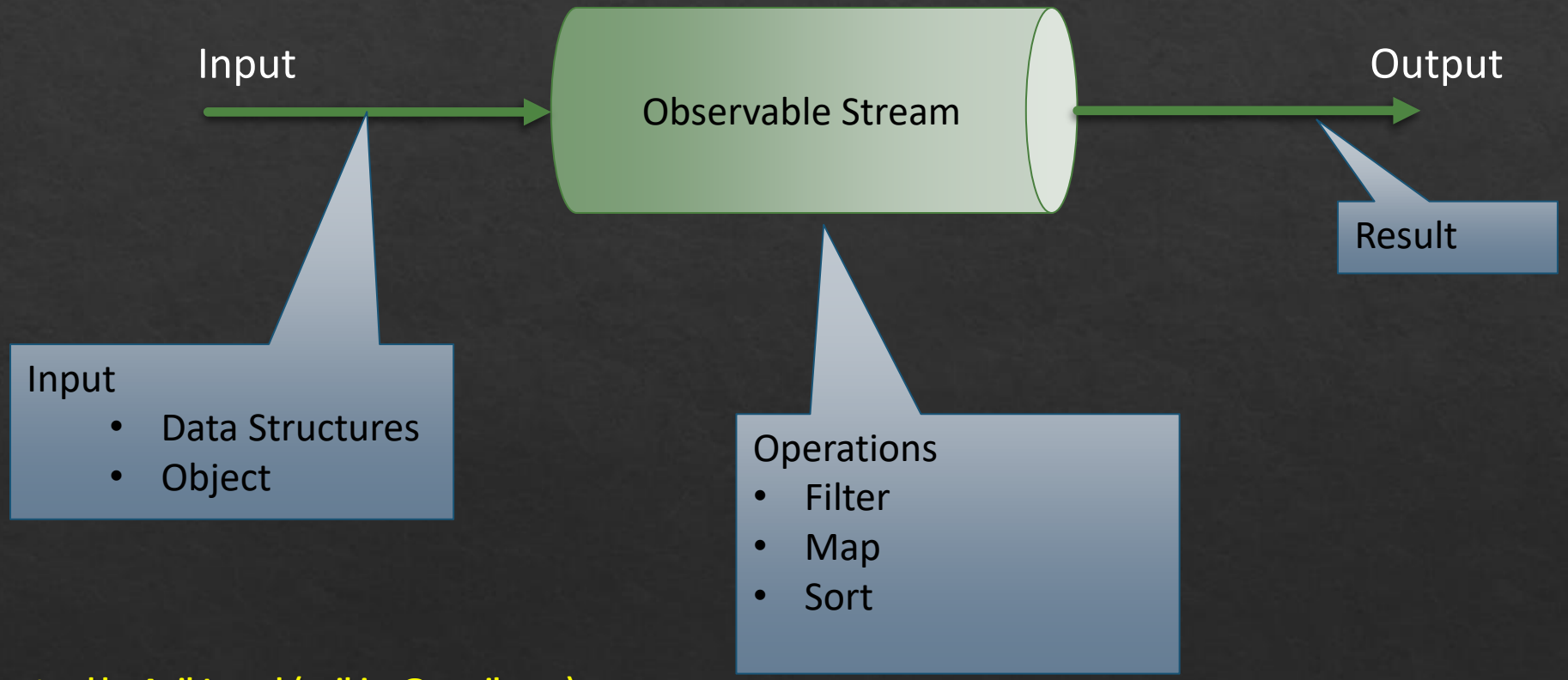
ReactiveX(http://reactivex.io/) is an API for asynchronous programming with observable streams

RxJS is a JavaScript library for reactive programming using Observables

In Angular, RxJS is used with Reactive Forms, Http, Routers

Presentaed by Anil Joseph(anil.jos@gmail.com)

Reactive Programming(Streams)



Presentaed by Anil Joseph(anil.jos@gmail.com)

Http Client

- Angular exposes a service to send AJAX request to the server.
- The service is a wrapper around the XMLHttpReqeust object.
- Angular 4
  - HttpModule
  - Http(Service)
- Angular 5
  - HttpClientModule
  - HttpClient

Presentaed by Anil Joseph(anil.jos@gmail.com)

Http: Methods

request

Performs any type of http request.

A RequestOptions object allows to configure the call

get

Performs a request with http method.

A RequestOptions object allows to configure the call

post

Performs a request with http method.

A RequestOptions object allows to configure the call

put

Performs a request with http method.

A RequestOptions object allows to configure the call

delete

Performs a request with http method.

A RequestOptions object allows to configure the call

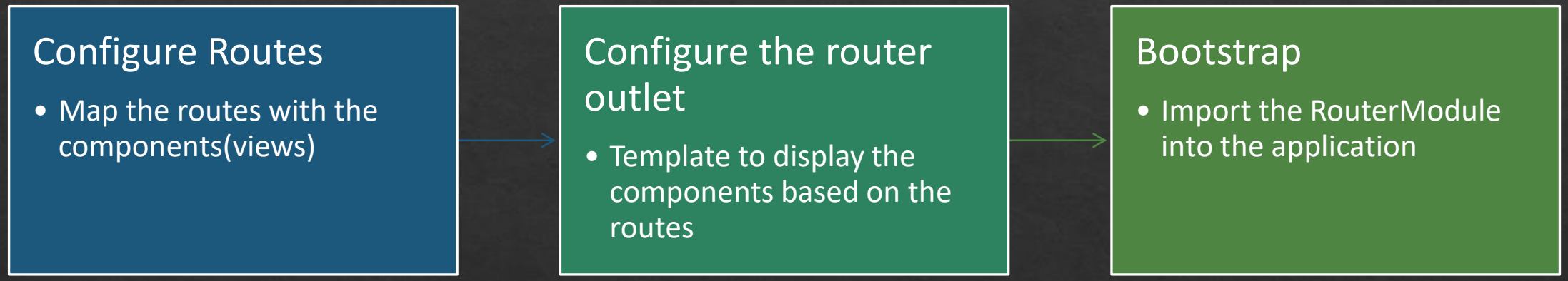
Presentaed by Anil Joseph(anil.jos@gmail.com)

Router

- The Angular Router enables navigation from one view to the next as users perform application tasks.
- Location Stratergies
  - Hash Location Eg: #/home
  - Path Location
    - Requires <base href="../">
    - Eg: /home

Presentaed by Anil Joseph(anil.jos@gmail.com)

Router: Setup



Presentaed by Anil Joseph(anil.jos@gmail.com)

Router: Navigation

- Using regular links with hash
  - #/home
  - /home
- Using routerLink directive
  - <a [routerLink]=“/home”>Home
- Programmatically
  - router.navigate(['users', 10])
  - router.navigateByUrl('/users/10')

Presentaed by Anil Joseph(anil.jos@gmail.com)

Ivy Compiler

Ivy is Angular's next-generation compilation and rendering pipeline.

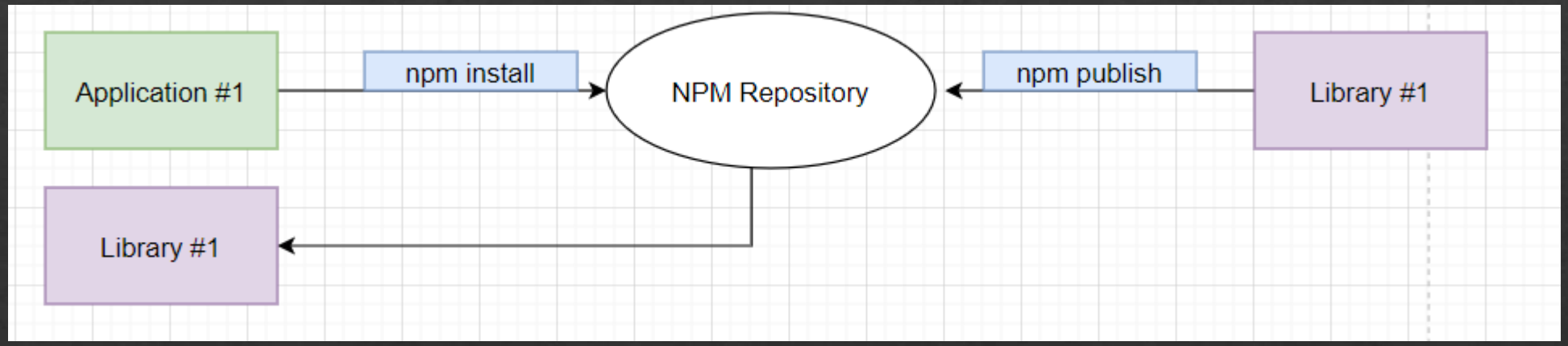
Creates smaller bundles and faster compilation

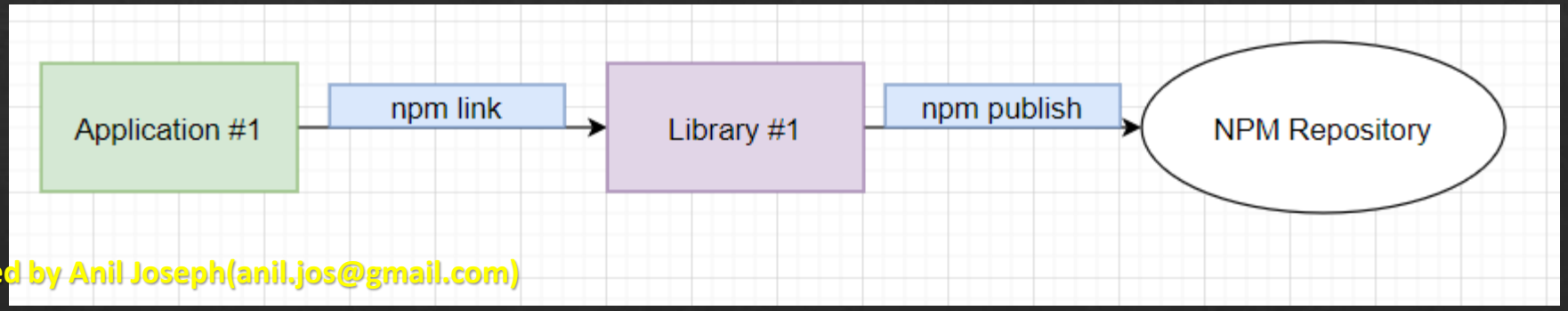
Better debugging

Dynamic loading of modules

Presentaed by Anil Joseph(anil.jos@gmail.com)

Angular Library





Presentaed by Anil Joseph(anil.jos@gmail.com)

Angular Library

An Angular library is an Angular project that differs from an app in that it cannot run on its own.

A library must be imported and used in an app.

Libraries extend Angular's base functionality.

Creating a library

- ng generate library my-lib

Build a library

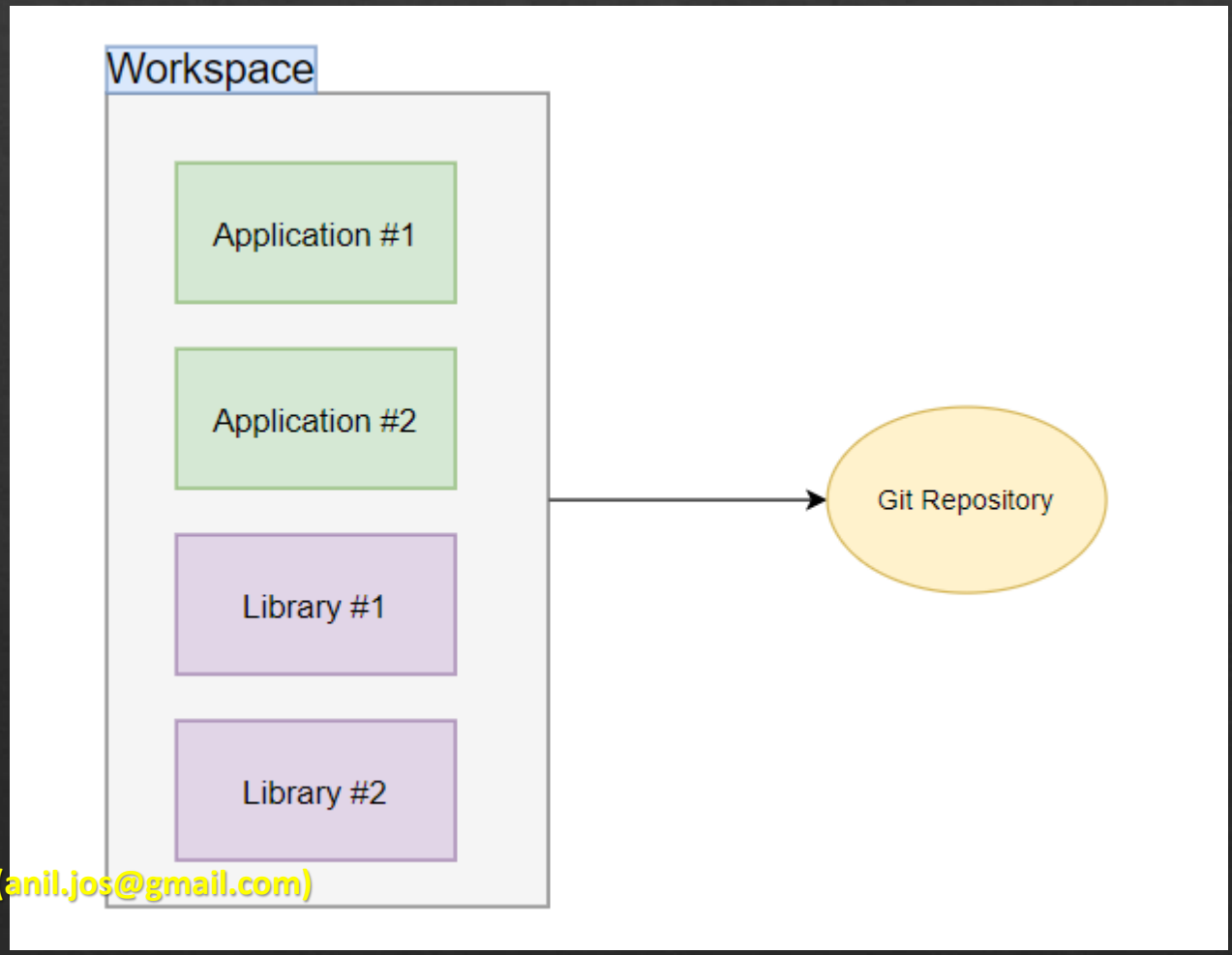
- ng build my-lib

Publish a library

- npm publish

Presentaed by Anil Joseph(anil.jos@gmail.com)

Angular Workspace(Mono Repo)



Presentaed by Anil Joseph(anil.jos@gmail.com)

Create a workspace

```
ng new my-workspace –create-application=“false”
```

Create an application in a workspace

```
cd my-workspace
ng generate application my-first-app
```

Create a library in a workspace

```
cd my-workspace
ng generate library my-lib
```

Angular Workspace(Mono Repo)

Presentaed by Anil Joseph(anil.jos@gmail.com)

Testing

Tools

Jasmine

- Library for Unit Tests

Karma

- TestRunner

Protractor

- Library for End to End Tests

Jasmine

- Jasmine is a behavior-driven development framework for testing JavaScript code.
- It does not depend on any other JavaScript frameworks.
- It does not require a DOM.

Jasmine

- Suites
  - Describe your tests
  - A test suite begins with a call to the global Jasmine function ***describe***
- Specs
  - The tests
  - Specs are defined by calling the global Jasmine function ***it***
  - A spec contains one or more expectations.
- Expectations
  - An expectation in Jasmine is an assertion that is either true or false.
  - Expectations are built with the function ***expect.***
- Matchers
  - A matcher implements a Boolean comparison between the actual value and the expected value.
  - Jasmine has a rich set of matchers included.

Jasmine

- beforeEach
  - called once before each spec in the describe in which it is called,
- afterEach
  - The afterEach function is called once after each spec.
- beforeAll
  - The beforeAll function is called only once before all the specs in describe are run
- afterAll
  - The afterAll function is called after all specs finish.

Angular Testing Utilities

- TestBed
  - TestBed is the first and most important of the Angular testing utilities.
  - It creates an Angular testing module
    - an @NgModule class
  - Use the configureTestingModule method to create the module.
  - The configureTestingModule method takes an @NgModule-like metadata object.
- ComponentFixture
  - A handle on the test environment surrounding the created component.
  - The fixture provides access to the component instance itself and to the DebugElement.
- DebugElement
  - A handle on the component's DOM element.

Deployment

Start the build

- ng build
- ng build –prod

Set the <base href="/> to point to the server subfolder if any

- ng build --base-href=/myapp/

Copy *everything* within the output folder (dist/ by default) to a folder on the server.

Configure the server to redirect requests for missing files to index.html

Presentaed by Anil Joseph(anil.jos@gmail.com)

Build for Production

ng build --prod

Ahead-of-Time (AOT) Compilation	• Pre-compiles Angular component templates.
Production mode	• Deploys the production environment which enables production mode.
Bundling	• Concatenates your many application and library files into a few bundles.
Minification	• Removes excess whitespace, comments, and optional tokens.
Uglification	• Rewrites code to use short, cryptic variable and function names.
Dead code elimination	• Removes unreferenced modules and much unused code.

Presentaed by Anil Joseph(anil.jos@gmail.com)

Resources

- Angular
  - https://angular.io
- Books
  - https://angular.io/resources?category=education

Presentaed by Anil Joseph(anil.jos@gmail.com)

Thank You

Email: anil.jos@gmail.com
WhatsApp: 9833169784

ANIL JOSEPH