Angular

Superheroic framework

Training Agenda

- ❑ TypeScript : Introduction
- ❑ Angular : Introduction
- ❑ Angular Building Blocks
  - Component
  - Modules
  - Directives
  - Services
  - Pipes
- ❑ Data Binding
- ❑ Forms
- ❑ HTTP
- ❑ Routing
- ❑ Authentication / Authorization
- ❑ Deployment

Why Angular ?

- Angular is easy
- ES2015+ features
- Corporate Care-taker
- Performance and Mobile
- Project architecture and Maintenance
- Component based architecture

AngularJS vs. Angular

Description	AngularJS	Angular
Nested scopes ($scope, watches)	Used heavily	Gone
Directive vs controllers	Used together	Component directives
Component and service implementations	Functions	ES6 classes
Module system	Angular’s modules	ES6 module system
Transpilation	Not used	Typescript

What is Angular ?

01

Angular is a structural framework for dynamic (SPA) web apps.

02

Extension of standard HTML markups.

03

Keeps the front-end clean; separation of concerns.

Why Typescript ?

Microsoft extension for JS

Object oriented features

ES6+ features

Type definition

Angular itself programmed in TS

TypeScript features

- Classes & Inheritance
- Module system
- Arrow functions
- Template String
- Constants and block scopes
- Destructuring
- Spread & Rest operators
- Decorators
- Additional types

Understanding Angular Environment Setup

- Node
- TypeScript
- SystemJS / Webpack
- Angular Packages
- RxJS
- ZoneJS

Components

- A component controls a patch of screen real estate that we could call a view, and declares reusable UI building blocks for an application.
- Passing data into components
  - Property binding
  - Event binding
  - Two way data-binding
- Nested components
- Data projection
- Component types :
  - Smart components
  - Dump components

ngOnChanges	- called when an input binding value changes
ngOnInit	- after the first ngOnChanges
ngDoCheck	- after every run of change detection
ngAfterContentInit	- after component content initialized
ngAfterContentChecked	- after every check of component content
ngAfterViewInit	- after component's view(s) are initialized
ngAfterViewChecked	- after every check of a component's view(s)
ngOnDestroy	- just before the component is destroyed

Component Life Cycle

Directives

- A Directive modifies the DOM to change appearance, behavior or layout of DOM elements.
- Directive Types :
  - *Component Directive* : directive with template
  - *Attribute Directive* : directives that change the behavior of a component or element but don't affect the template
  - *Structural Directives* : directives that change the behavior of a component or element by affecting how the template is rendered

Pipes

- Pipes are used to filter/format data for template
- Built-in Pipes :
    - Currency
    - Date
    - Uppercase
    - Lowercase
    - Number
    - JSON
    - Percent
    - Async
- Custom pipes
    - Pure
    - Impure

Forms

Template Driven Forms

- Angular infers the Form Object from the DOM
- App logic resides inside the template

Model Driven Forms

- Form is created programmatically and sync with the DOM
- App logic resides inside the component
- Use of FormControl, FormGroup, FormBuilder

DI & Services

Angular's DI system is controlled through **@NgModule**.

Services implement DI concepts in an Angular App.

Services are simple ES6 classes.

Services are registered with Angular App using providers.

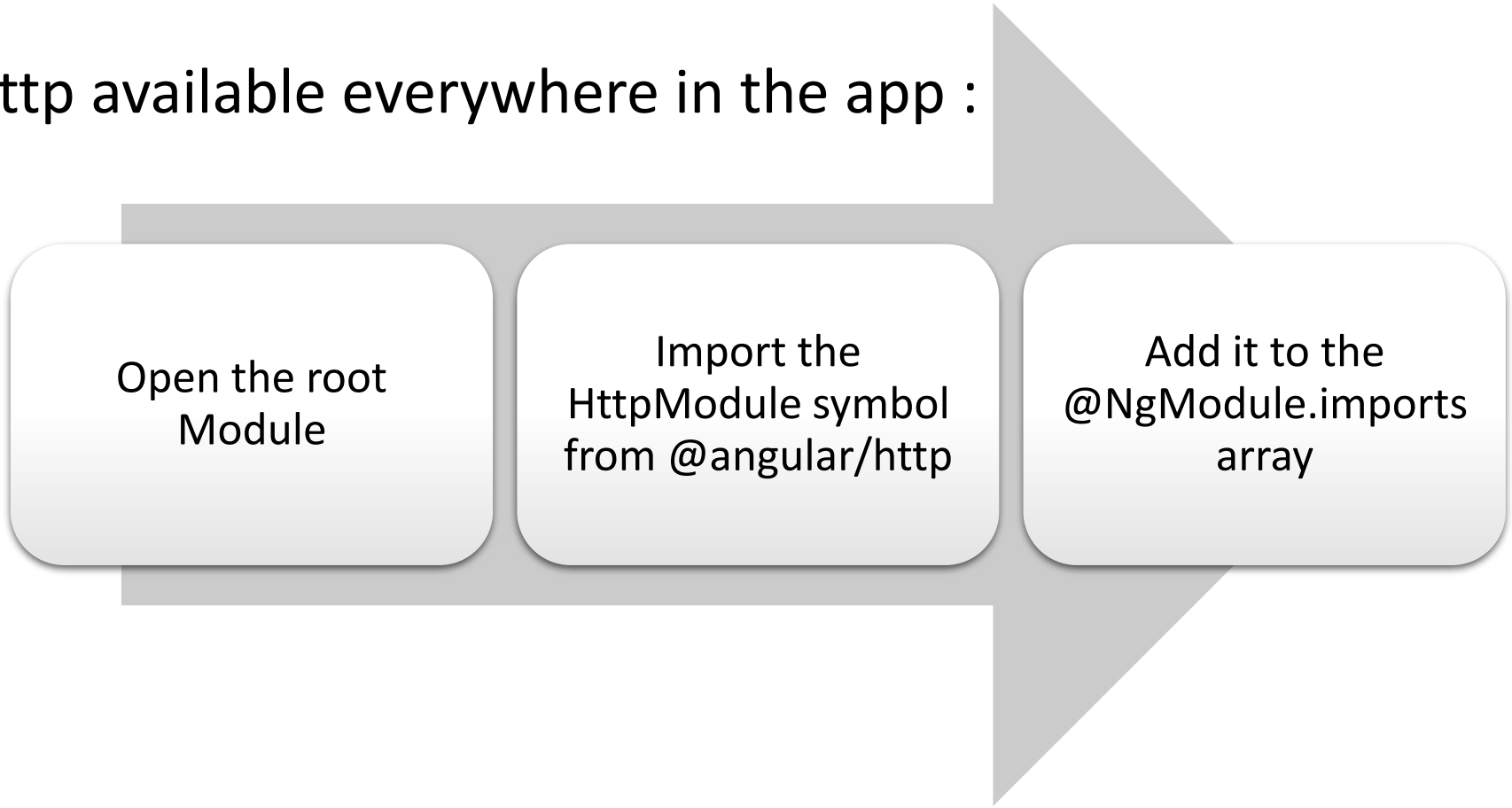
Services are Singleton.

Hierarchical Injector

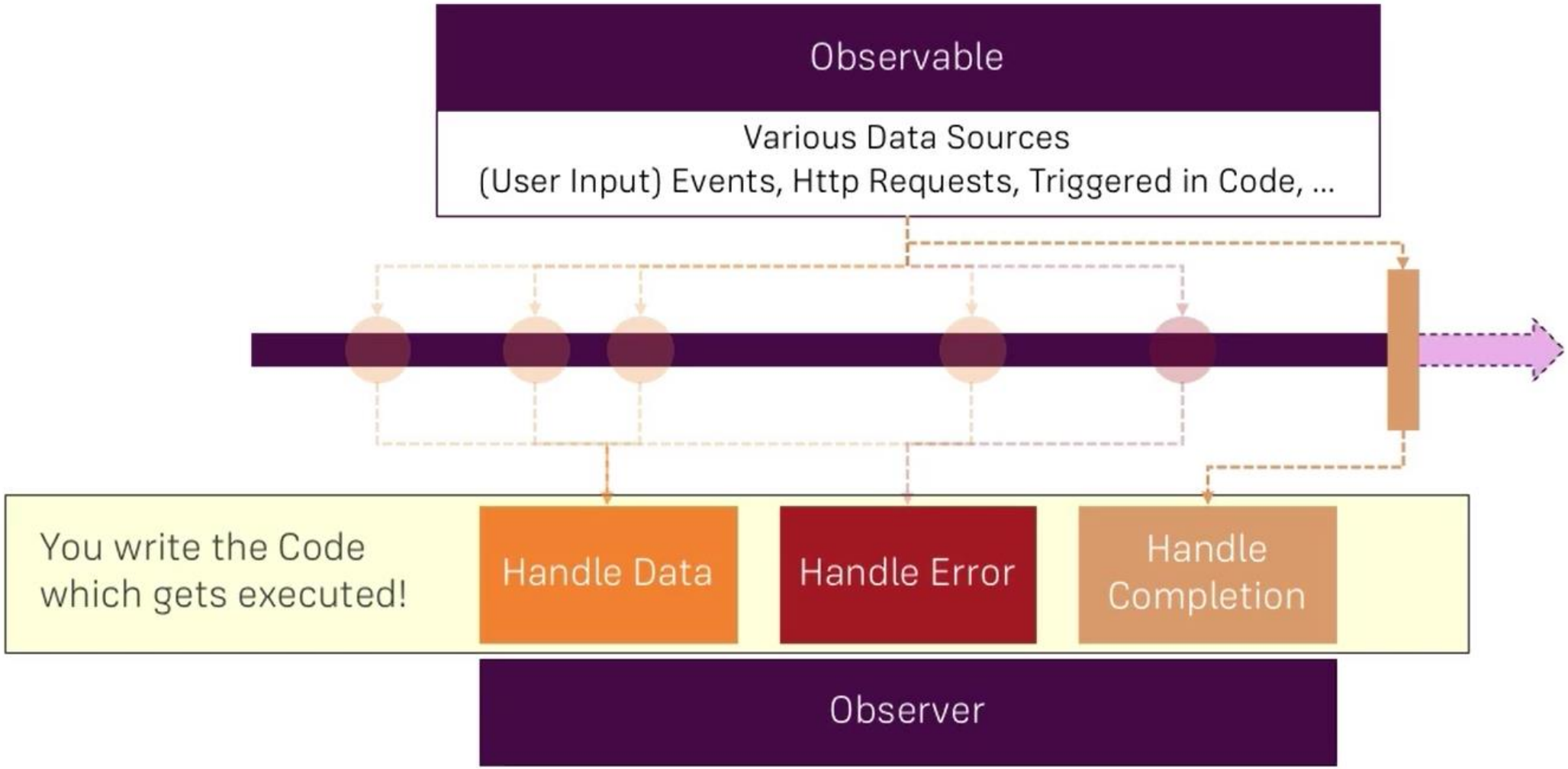
Root Module	Same instance of service is available Application-wide
Root Component	Same instance of service is available for all components (but not for other services)
Other Component	Same instance of service is available for the component and it’s own child components

HTTP

- Http is Angular's mechanism for communicating with a remote server over HTTP.
- To make Http available everywhere in the app :



Observables : An Overview

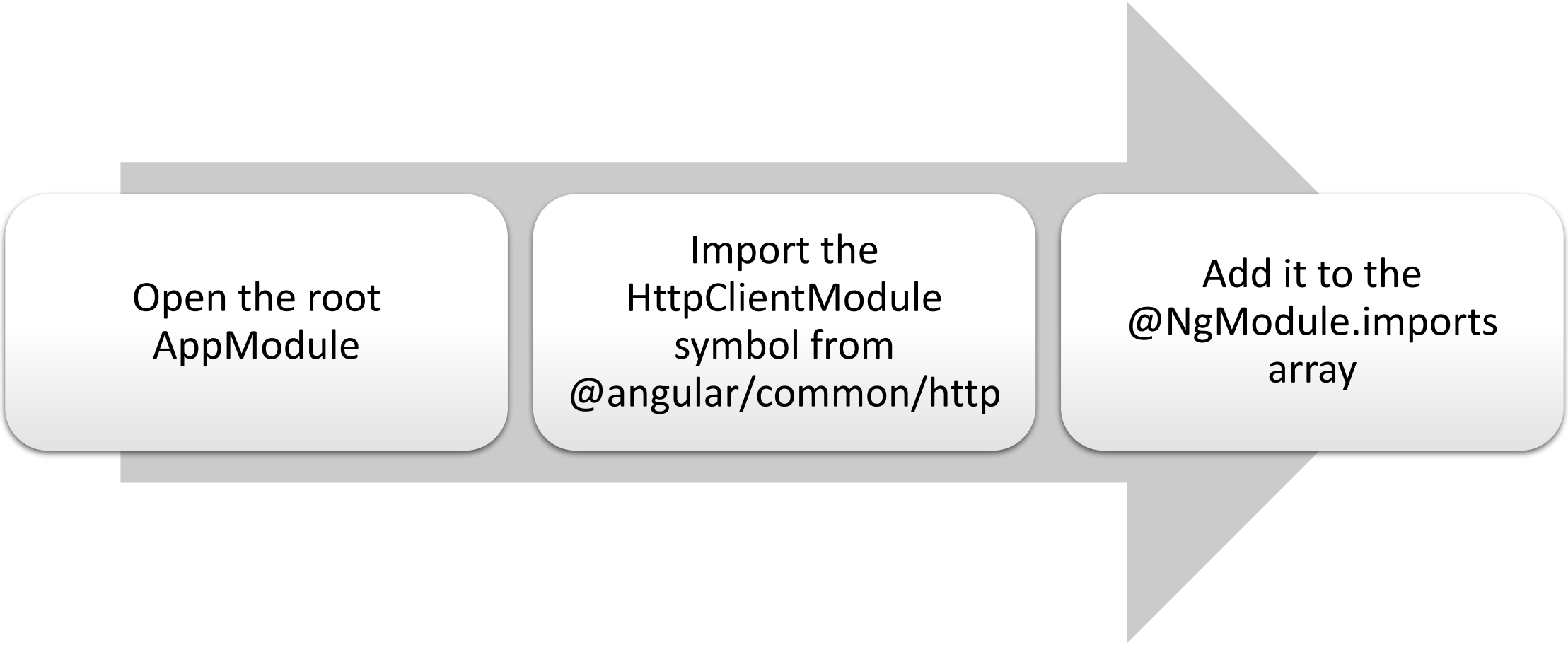


HttpClient

The HttpClient in @angular/common/http offers a simplified client HTTP API for Angular applications that rests on the XMLHttpRequest interface exposed by browsers.

Benefits of HttpClient:	
	Typed request and response objects
	Request and response interception
	Observable APIs
	Streamlined error handling.

HttpClient : Unlocking



Routing allows to:

- Maintain the state of the application
- Implement modular applications
- Implement the application based on the roles (certain roles have access to certain URLs)

5 steps routing:

- Checking the base href tag in index file
- Configuring routes with components
- Tell angular about routing app
- Setting up the routing links
- Provide space on template to load the component

Routing

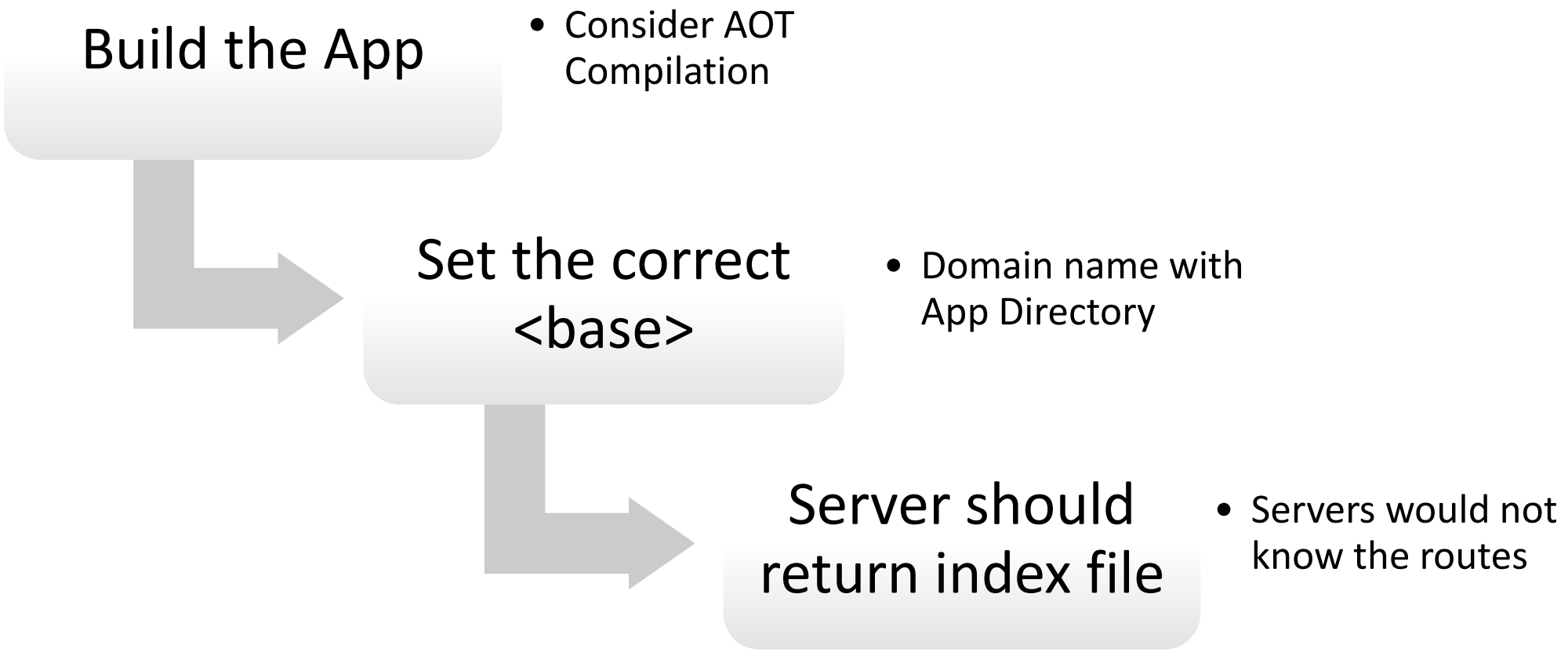
Routing (Cntd..)

- Programmatic navigation
- Child routing
- Routes with parameters
- Route guard (Authentication)
- Query Parameters

Modules

- A module is a mechanism to group components, directives, pipes and services that are related
- Module Types -
  - Root Module : one per application
  - Feature Module : depends on application features
- Modules can be instantiate lazily

Deployment Steps :



Debugging Angular Apps

- Prevent Bugs with TypeScript
- Using Debugger Statements to Stop JavaScript Execution
- Inspect Data with the JSON pipe
- Console Debugging
- Augury Chrome Plugin
- Debugging RxJS Observables using ‘do’ operator

Optimizing Angular App Performance

- Using onPush change detection strategy
- Using trackBy function
- Avoid computing values in templates
- Using lazy loading
- Disable change detection (if required)

References

Books

- Rangle's Angular2 Training Book
- Ngbook2

Web

- http://angular.io
- http://rangle.io
- http://www.egghead.io
- http://www.youtube.com