@NgModule & @Component

Siddharth Ajmera **@SiddAjmera**

Content

- Decorators
- Angular Module
- Application Bootstrap
- Angular Components
- Data/Property/Event Bindings
- Inter-Component Communication
- Change Detection in Angular with ZoneJS, NgZone

“Decorators are to Classes what Dragons are to Daenerys”

HBO

Decorators

- Decorators are a stage 2 proposal for JavaScript and are available as an experimental feature of TypeScript.
- Decorators are essentially functions and are used with @ sign prefixed.
- Angular classes and properties in them are defined by the decorators that sit on them.
- In Angular, almost everything is essentially a class. But these decorators transform them into a module, component, service, pipe or a directive.
- Following are a few Decorators used in Angular

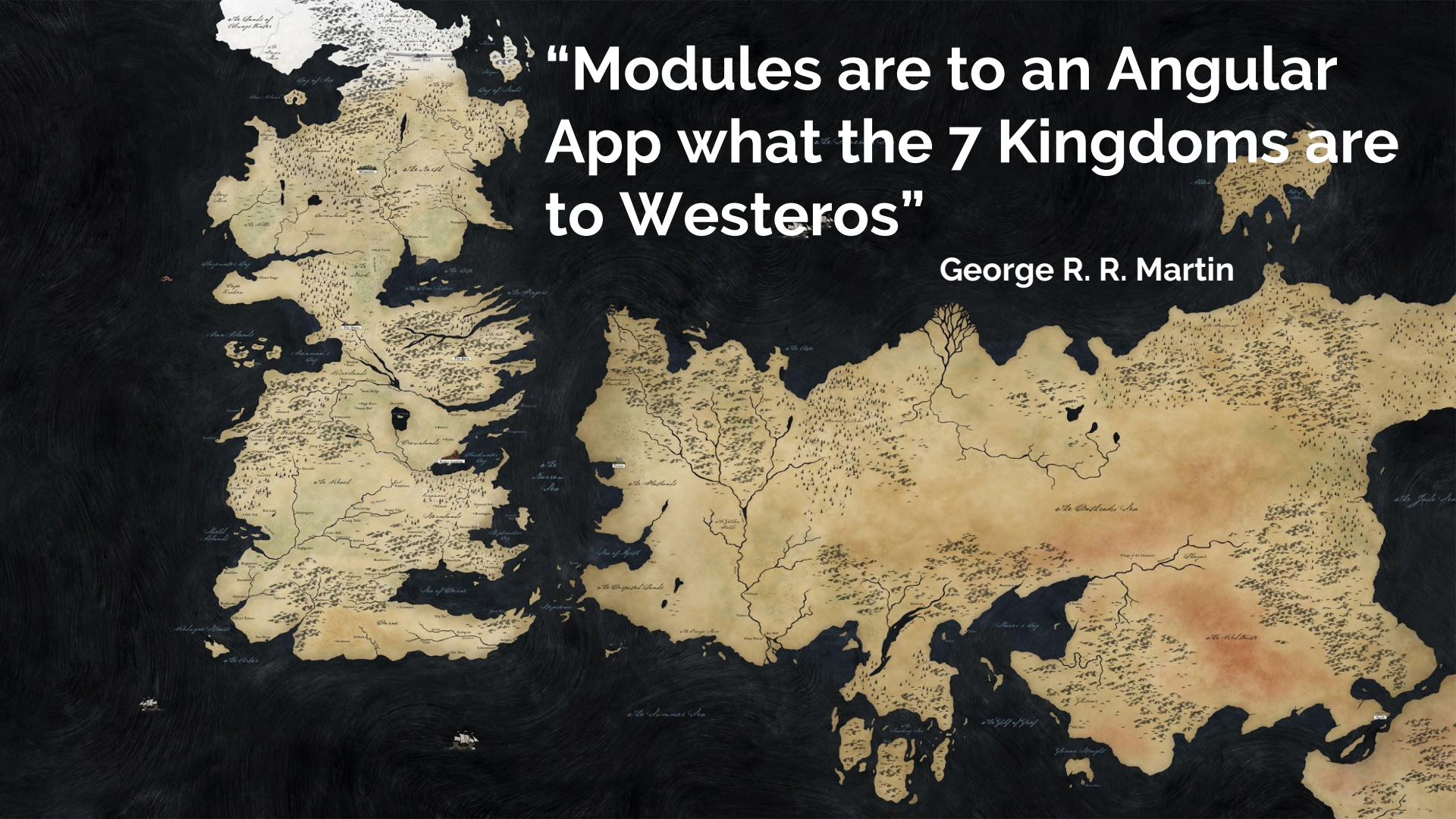
@NgModule
@Pipe
@Input

@Component
@Injectable
@Output

@Directive
@Inject
@HostListener

Before We start coding...

- Every property, method, member must have an access modifier
- Every property, method, member must have a type definition
- Keep files under 200 lines of code
- Keep lines under 100 cols in length (wrap longer lines)
- Keep number of function parameters under 4 (wherever possible)
- For best practices, refer to the Angular Style Guide.



“Modules are to an Angular App what the 7 Kingdoms are to Westeros”

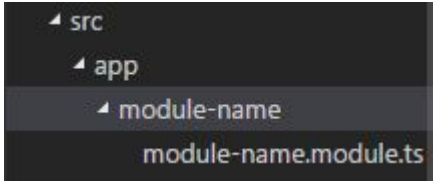
George R. R. Martin

Angular Module

- Angular apps are modular and Angular has its own modularity system called NgModules.
- Every Angular app has at least one NgModule class, the root module, conventionally named AppModule.
- A Module can be defined as a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities.
- Modules are a great way to organize an application and extend it with capabilities from external libraries.
- Many Angular libraries are modules (such as FormsModule, HttpModule, and RouterModule). Many third-party libraries are available as NgModules (such as Material Design, Ionic, AngularFire2).

Creating a Module

- We recommend creating anything with Angular CLI.
- To create a new Angular Module, type in ng g m module-name
- This will create a file named module-name.module.ts inside module-name folder in the app folder. Something like this:



- You'll also receive a message.

```
installing module
  create src\app\module-name\module-name.module.ts
  WARNING Module is generated but not provided, it must be provided to be used
```

Essential Metadata: Module

- imports: This is an array that contains a list of all the modules, features of which your Angular Module is going to use. It can contain internal as well as external modules. Only NgModule classes go in the imports array.
- declarations: This is an array that contains a list of declarable, that your module contains and uses. Only declarables(components, directives and pipes) belong in the declarations array.
- exports: This is an array that contains a list of directives/pipes/modules that your module will expose, for the other modules that import it, to use.
- providers: This is an array that contains a list of Services that are exposed inside your module and that the components and other services in your module will use.
- bootstrap: This is an array that you'll generally see only in the app.module.ts file. This is an array that contains a list of Components that will get bootstrapped once your module gets bootstrapped.

Application Bootstrap

- Many ways to bootstrap an App depending on how you want to compile it and where you want to run it.
- For this tutorial, we'll compile the application dynamically with the JIT compiler and run it in a browser. Find the other way to compile your App here.
- Our App has an entry point “main” and an “index” that we define in our .angular-cli.json config file.
- We need a browser platform for our dynamic JIT Compilation and an execution environment for our App to run on. That’s what the code in main.ts does.


“Components are Lego Blocks, just in code.”
JON SNOW

Angular Component

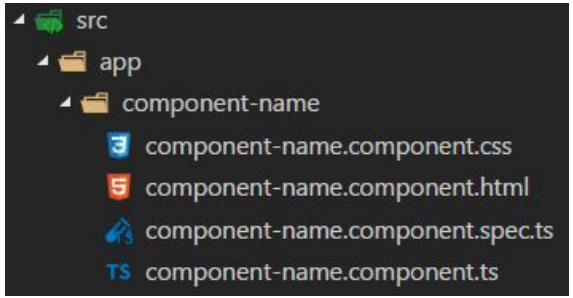
- Most basic building block of a UI in an Angular App which is a tree of Angular components.
- Angular components are a subset of directives. Only one component can be instantiated per element in a template.
- @Component marks a class as an Angular component and provides additional metadata to determine how the component should be processed, instantiated and used at runtime.
- Must belong to an NgModule in order for it to be usable by another component or application i.e should be listed in the declarations field of that NgModule.
- Control their runtime behavior by implementing various Life-Cycle hooks.

Component: Metadata

- selector: css selector that identifies this component in a template
- template: inline-defined template for the view
- styles: inline-defined styles to be applied to this component's view
- templateUrl: url to an external file containing a template for the view
- styleUrls: list of urls to stylesheets to be applied to this component's view
- providers: - list of providers available to this component and its children
- animations: list of animations of this component
- encapsulation: style encapsulation strategy used by this component
- changeDetection: change detection strategy used by this component

Creating a Component

- To create a new Angular Component, type in ng g c component-name
- This will create a folder named component-name with 4 files in it.Something like this:

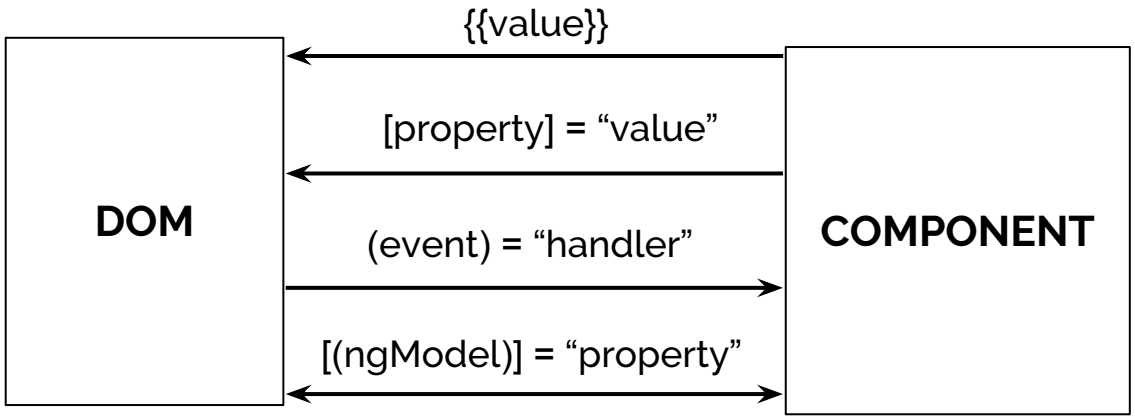


- You'll also receive a message.

```
C:\Angular\AngularApp>ng g c componentName
installing component
  create src\app\component-name\component-name.component.css
  create src\app\component-name\component-name.component.html
  create src\app\component-name\component-name.component.spec.ts
  create src\app\component-name\component-name.component.ts
  identical src\app\app.module.ts

C:\Angular\AngularApp>
```

Component Bindings



View Queries

- Angular provides the decorators @ViewChild, @ViewChildren, @ContentChild, @ContentChildren to get element references
- ViewChild can be used to capture elements in the component template
- ContentChild can be used to capture elements present in the opening and closing tags of a component, i.e. elements that are projected into the components.
- Angular allows us to create template references by adding a local variable #name to the HTML element
- Template references can be used with ViewChild and ContentChild in order to get the element reference (ElementRef) in component
- We can access and modify the native element properties through the element reference provided by ViewChild or ContentChild

Component Interaction

- Pass data from parent to child with @Input binding. Demo.
- Parent listens for child event using @Output binding. Demo.
- Parent interacts with child via local variable. Demo.
- Parent calls an @ViewChild(). Demo.

Change Detection

- Change detection is the process of capturing the internal state of a program and its changes, and projecting it to the user
- Change detection can be triggered by async tasks:
  - Events (Action on the dom, or custom events)
  - XHR (Http requests to fetch data)
  - Timers (timeouts and intervals)
- Angular uses Zones to implement change detection
- Angular comes with its own zone called NgZone
- Each component has its own change detector
- Angular exposes an API for CD called ChangeDetectorRef

A little more on Zones

- A Zone is an execution context that persists across async tasks

```
Zone.current.fork({}).run(function () {
    Zone.current.inTheZone = true;

    setTimeout(function () {
        console.log('in the zone: ' + !!Zone.current.inTheZone);
    }, 0);
});
```

- Zone monkey-patches all methods which cause async tasks to run in a zone

```
function zoneAwareAddEventListener() {...}
function zoneAwareRemoveEventListener() {...}
function zoneAwarePromise() {...}
function patchTimeout() {...}
window.prototype.addEventListener = zoneAwareAddEventListener;
window.prototype.removeEventListener = zoneAwareRemoveEventListener;
window.prototype.promise = zoneAwarePromise;
window.prototype.setTimeout = patchTimeout;
```

Continued...

- Zones can be created, forked, and extended
- The forked zone contains the methods:
  - onZoneCreated - Runs when zone is forked
  - beforeTask - Runs before a function called with zone.run is executed
  - afterTask - Runs after a function in the zone runs
  - onError - Runs when a function passed to zone.run will throw an error
- We can extend the Zone to include methods we need.

NgZone

- NgZone is a forked zone that extends the zone API
- NgZone can be imported from @angular/core
- NgZone adds the following custom events that we can subscribe to:
  - onUnstable() - Notifies when code entered Angular Zone
  - onMicrotaskEmpty() - Notifies when there are no more microtasks
  - onStable() - Notifies when the last onMicrotaskEmpty has run
  - onError() - Notifies that an error has occurred
- It contains a method runOutsideAngular() to execute code outside Angular's zone which will not trigger change detection.
- You can re-enter the Angular zone by invoking the NgZone.run() method

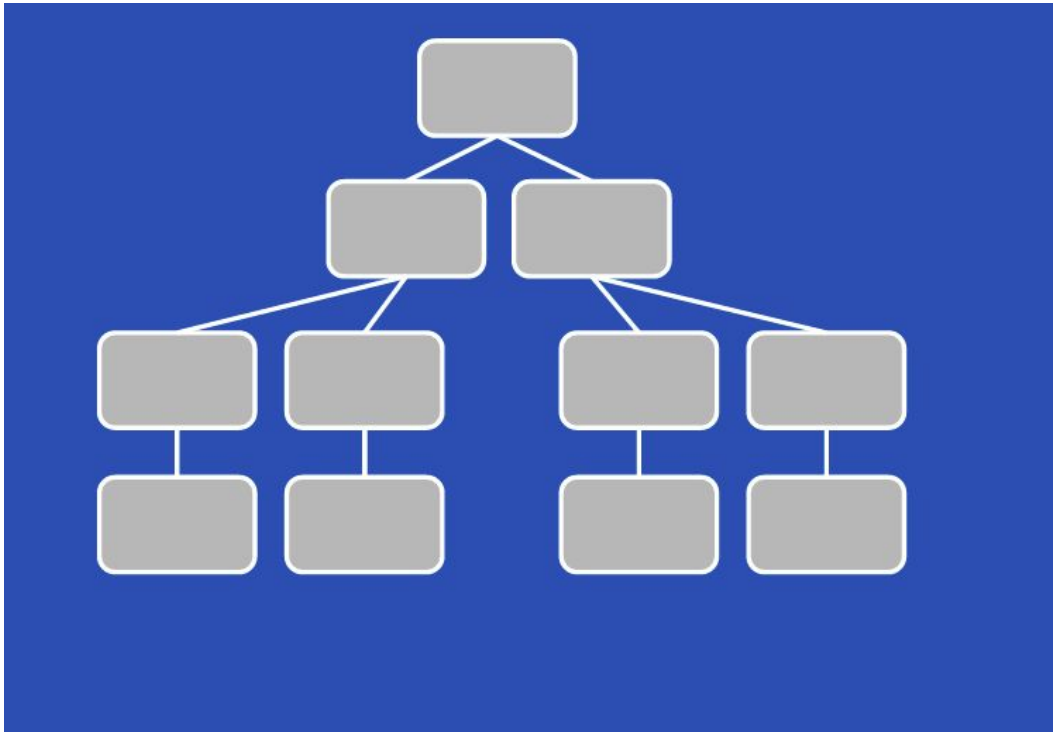
Change Detection in Ng

- Since Angular uses Zones, change detection is triggered through monkey-patched native methods and does not require anything more
- Angular internally contains an ApplicationRef which controls CD
- Whenever an onMicrotaskEmpty() event is fired, Angular executes a tick() function which initiates change detection for all change detectors.
- Each component has its own change detector.
- Change detection is performed top to bottom, starting from the root component and flowing down the change detector tree
- By default, Angular detects changes for all components when event is fired
- We can use ChangeDetectionStrategy to prevent CD where required

Change Detector Ref

- ChangeDetectorRef is responsible for performing change detection
- It consists of the methods:
  - markForCheck()
  - detach()
  - detectChanges()
  - checkNoChanges()
  - reattach()
- We can import ChangeDetectorRef() from @angular/core
- Can be used to force change detection
- Can be used to exclude components from change detection
- Can be used to override ChangeDetectionStrategy.

Change Detection Tree




"~~Winter~~ Assignment 1 is here."
NIGHT KING

Assignment 1

- Generate an App with Angular CLI.
- Create a Parent Component and a Child Component.
- In Parent Component, implement:
  - String Interpolation
  - Property Binding
  - Event Binding
  - Two Way Binding
  - Usage of Child Component with Input Data passed to it.
- In Child Component, implement:
  - An @Input Property
  - Usage of the Data provided to it by the Parent Component

H2 # One Way Data Binding

p This is a placeholder text for String Interpolation

This is a placeholder text for Property Binding

I'm an Example for Event Binding. Click Me!

button with btn btn-primary

H2 # Two Way Data Binding

This is a placeholder text for Two Way Data Binding

p This is a placeholder text for Two Way Data Binding

hr

H2 # Child Element

- First Element
- Second Element

li with list-group-item

Content

- SystemJS and Webpack
- Change Detection in Angular with ZoneJS, NgZone
- Coding Guidelines
- Decorators
- Angular Module
- Application Bootstrap
- Angular Components
- Data/Property/Event Bindings
- Inter-Component Communication

SystemJS

- SystemJS is a Configurable module loader
- Is required to load Typescript/ES6 modules
- Supports Lazy Loading
- Supports plugins for loading files of different types
- Does not have built-in support for bundling, packaging, minification, and other build tasks.
- Requires integration with gulp or SystemJS builder
- Easy to configure and suitable for small projects
- Configuration file: systemjs.config.js

Webpack

- Webpack is a module bundler
- Uses a dependency graph to load and bundle all modules
- Supports plugins for loading files of different types
- Supports build tasks such as linting, bundling, minification, and more with the use of loaders/plugins
- Supports Lazy Loading in Angular with ng-route-loader
- Comes with a dev server with hot reloading
- Used by angular-cli under the hood
- Configuration file: webpack.config.js

Before We start coding...

- Naming conventions:
  "Clean code is simple and direct. Clean code reads like well-written prose".
  - Grady Booch
- Proper Indentation and Linting:
  "We spend most of our time staring into the abyss rather than power typing"
  - Douglas Crockford
- Single responsibility principle:
  Good, clean code, does one thing, it does it perfectly, and it does it only.
- A few rules of thumb:
  - Every property, method, member must have an access modifier
  - Every property, method, member must have a type definition
  - Keep files under 200 lines of code
  - Keep lines under 100 cols in length (wrap longer lines)
  - Keep number of function parameters under 4 (wherever possible)

Change Detection in Ng

- Since Angular uses Zones, change detection is triggered through monkey-patched native methods and does not require anything more
- Angular internally contains an ApplicationRef which controls CD
- Whenever an onMicrotaskEmpty() event is fired, Angular executes a tick() function which initiates change detection for all change detectors.
- Each component has its own change detector.
- Change detection is performed top to bottom, starting from the root component and flowing down the change detector tree
- By default, Angular detects changes for all components when event is fired
- We can use ChangeDetectionStrategy to prevent CD where required
- ChangeDetector classes are monomorphic and highly optimized, which speeds up change detection