Angular

Lecture 1

Agenda

- Angular building blocks : Components.
- Templates and styles
- Reusable components
- Component lifecycle.
- Data binding
- Directives
- Questions!

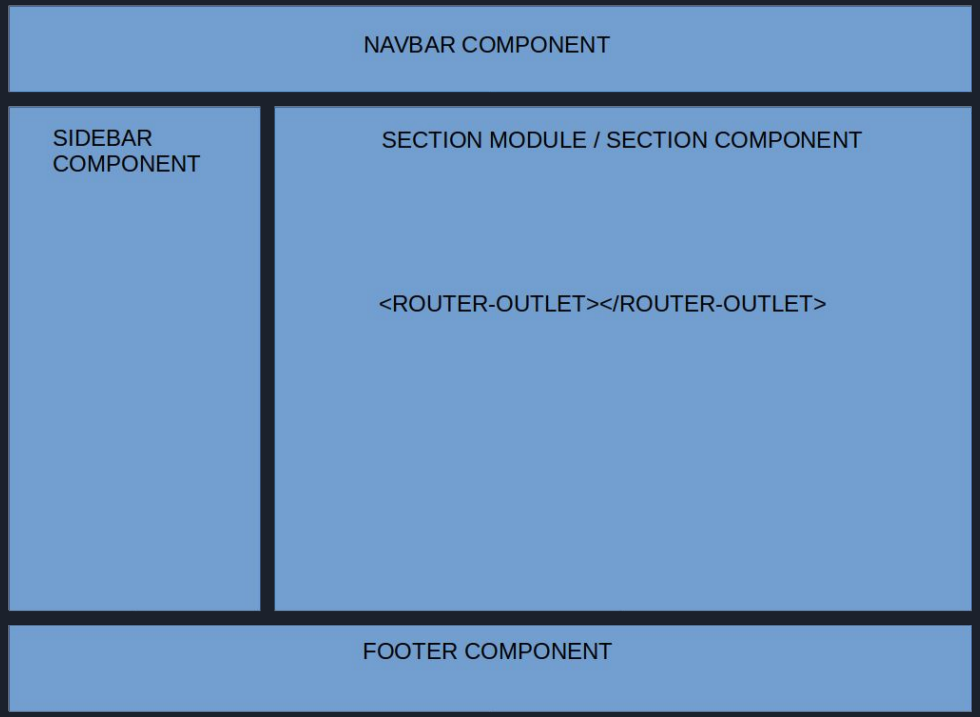
Components

Components

Components are the main building block for Angular applications , Components are composable, we can build larger Components from smaller ones. Each component consists of:

- An HTML template that declares what renders on the page
- A Typescript class that defines behavior
- A CSS selector that defines how the component is used in a template

Components


NAVBAR COMPONENT
SIDEBAR COMPONENT
SECTION MODULE / SECTION COMPONENT
<ROUTER-OUTLET></ROUTER-OUTLET>
FOOTER COMPONENT

Components

- **Create new component :** ng generate component navbar
- Discover new generated component class.
- Create inline template and styles .
- External template and styles

Reusable components

- To use component in any other place you can use it by selector name mentioned in @component between HTML tags.
- Create once , use multiple times.
- Syntax : **<app-navbar></app-navbar>**

Component lifecycle

- A component instance has a lifecycle that starts when Angular instantiates the component class and renders the component view along with its child views.
- You don't have to implement all (or any) of the lifecycle hooks, just the ones you need.
- After your application instantiates a component or directive by calling its constructor, Angular calls the hook methods you have implemented at the appropriate point in the lifecycle of that instance.

Component lifecycle

ngOnInit()

Initialize the directive or component after Angular first displays the data-bound properties and sets the directive or component input properties.

ngDoCheck()

Detect and act upon changes that Angular can't or won't detect on its own.

Component lifecycle

ngOnDestroy()

Cleanup just before Angular destroys the directive or component. Unsubscribe Observables and detach event handlers to avoid memory leaks.

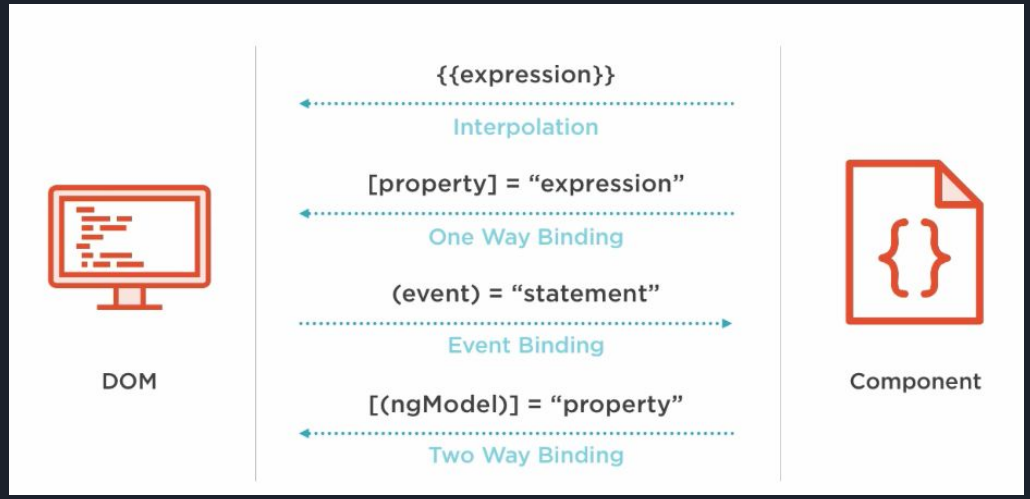
ngAfterViewInit()

Respond after Angular initializes the component's views and child views, or the view that contains the directive.

Data Binding

Data binding

You could translate data binding with communication. Communication between your TypeScript code of your component (your business logic) and the template .



Data binding

- String Interpolation

  allows you to incorporate dynamic string values into your HTML templates ,

  And used like this {{expression}}

- Property Binding

  Property binding in Angular helps you set values for properties of HTML elements or directives.

  Example : <img [src]="itemImageUrl">

Data binding

- Event Binding

  Event binding allows you to listen for and respond to user actions such as keystrokes, mouse movements, clicks, and touches.

  Example : <button (click)="onSave()">Save</button>

- Two way binding

  Two-way binding combines property binding with event binding for example to two way binding [(ngModel)]

  Note : need to import import { FormsModule } from '@angular/forms' for two way binding in forms in app module to work

Directives

Directives

Directives are kind of instructions to the DOM , Directives are components without a view. They are components without a template. Or to put it another way, components are directives with a view.

There’s different types of directives :

- Component Directives
- Structural Directives
- Attribute Directives

Directives

Directives are kind of instructions to the DOM.

There’s different types of directives :

- Component Directives - 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.

Directives

Structural directives :

Structural directives are responsible for HTML layout. They shape or reshape the DOM's structure, typically by adding, removing, or manipulating elements.

- NgIf
- NgFor
- NgSwitch [Self Study]

Directives

NgFor

NgFor is a structural directive, meaning that it changes the structure of the DOM.

It's point is to repeat a given HTML template once for each value in an array, each time passing it the array value as context for string interpolation or binding.

```
 <li *ngFor="let person of people; let i = index"> (1)

    {{ i + 1 }} - {{ person.name }} (2)

  </li>

list.map((item) => ())
```

Directives

NgIf

The NgIf directive is used when you want to display or remove an element based on a condition.

If the condition is false the element the directive is attached to will be removed from the DOM.

```
<li *ngIf="age < 30">

   {{name}}

</li>
```

Directives

Attribute directives

- **ngClass :** ngClass directive changes the class attribute that is bound to the component or element it’s attached to.

  ```
  <div [ngClass]="{'text-success': country === 'UK'}"> </div>
  ```

- **ngStyle :** ngStyle is mainly used to modify or change the element’s style attribute. This attribute directive is quite similar to using style metadata in the component class.

  ```
  <div [ngStyle]="{'background-color': country === 'UK' ? 'green' : 'red' }"> </<div>
  ```

Thank you

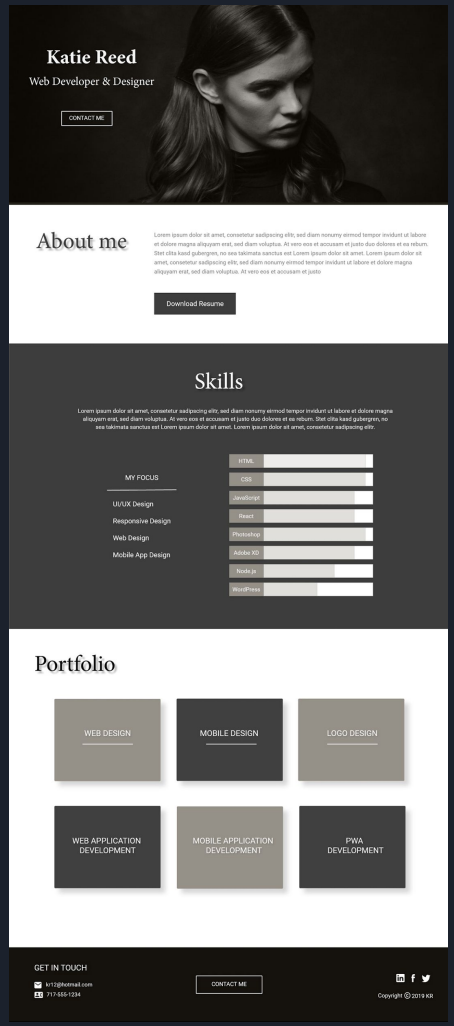
Portfolio

Replace code found in app.component.html and start create your portfolio page using Bootstrap.

Portfolio will contain the following sections :

- Hero section (name and job title).
- Bio and about me (education and experiences) section with button to download CV .
- Skills section with progress bar for each skill
- Portfolio and projects section
- Footer contains contact us section with email and social media links (facebook , github , linkedin) [Will use fontawesome]

Note : Installing ng-bootstrap and use one of its component (Bonus)



Portfolio

Notes:

- Each section is a separate component.
- Use String interpolation to bind your name and job title form TS to html.
- Create Array in TS file for skills and use directives to iterate over it
- Same for Portfolio to set array in TS
- Use the style directives to change the background of portfolio cards same as shown in the design