angular

- framework (collection of JS, html and css files) used to design client side applications
- the angular application will be executed by browser
- used to develop SPA (single page application) type application
    - the angular application will load only one page at the begining
    - then it will keep on refreshes only part(s) of pages
    - benefits
        - best performance like native application
        - because the entire UI gets loaded at the begining, application works in case of no internet connection (provided the application is already loaded in browser)
    - disadvantages
        - the application takes more time to start
- typescript will be used for developing applications using angular
    - the typescript code will get transpiled to javascript
    - the transpiled JS code will get passed to the browser
- alternatives to angular
    - react
    - vue.js

configuration

- documentation: angular.io

- to install anuglar framework use npm install command

  ```
  sudo npm install -g @angular/cli
  ```

- @angular/cli installs a utility named **ng** (angular)

ng

- used to create/build/test angular application
- **ng new**
    - used to create a new angular application
    - e.g.

  ```
  > ng new app1
  ```

    - it downloads all the packages required to develop and run angular application
- **ng server**
    - used to run the application

    - e.g.

    ```
    # visit http://localhost:4200
    > ng serve

    # visit http://localhost:9090
    > ng serve --port 9090

    # visit http://localhost:4200
    # --host will accept incoming connections from networking
    > ng serve --host '0.0.0.0'
    ```

- **ng generate**
  - used to generate different type of classes
  - **component**
    - used to create a component
    - creates files with .html, .css, .ts and .spec.ts extensions
    - declares the component in the AppModule
    - e.g.

      ```
      # create a component
      > ng generate component <name>

      # create a component
      > ng g c <name>
      ```

angular project hierarchy

- **e2e**
  - end to end testing
  - used to test the application's functionality
  - angular uses jasmin to test the application
- **node_modules**
  - directory which contains all the dependency modules
  - e.g.
    - @angular/animations: used to add animation support in the application
    - @angular/cli: used for managing the application
    - @angular/core: used to provide fundamental components to the application
    - @angular/forms: used to add the forms support (used to get input from user)
    - @angular/router: used to add rounting facility

        - @angular/commong/http: used to provide HttpClient which can be used to connect the angular application with backend
- **src**
    - **app**
        - contains the application source code
    - **assets**
        - contains the application assets
        - like images, audio or video files
    - **environments**
        - used to separate the configuration logically
        - e.g.
            - environment.ts : represents the dev environment
            - environment.prod.ts : represents the production (cloud server) environment
    - **favicon.ico**
        - used to display the shortcut icon
        - the one which will be displayed on the tab
    - **index.html**
        - the only html file which has the header and body
        - the application starts by loading this file
    - **main.ts**
        - angular's entry point
        - when application starts, the application modules get bootstrapped using main.ts
    - **polyfill.ts**
        - used to fill the gap between the older JS version with the latest ES7/ES8 changes
    - **styles.css**
        - used to add global styles
        - the styles which can be shared among multiple components
    - **test.ts**
        - used for testing the application
- **.editorconfig**
    - used to configure the editors
- **.gitignore**
    - used to ignore the files/folders while committing the changes to git repositories
- **angular.json**
    - used to configure the application
- **karma.conf.js**
    - configuration used by jasmin
- **package.json**
    - configuration on your application
    - e.g.
        - application dpenedencies
        - application basic information
- **README.md**
    - used to configure the readme for your application
- **tsconfig.app.json**
- **tsconfig.json**

- **tslint.json**
  - used to configure the typescript
- **tsconfig.spec.json**
  - used for configuring the test cases

NgModule

- every angular application is a modular application
- every anuglar application requires at least one NgModule
- NgModule represents a module which brings all the application parts together
- NgModule is different than node module
- to create a agular module, call @NgModule() decorator

```
@NgModule({
  declarations: [],
  imports: [],
  providers: [],
  bootstrap: []
})
export class AppModule { }
```

- @NgModule is called with metadata which contains
  - declarations
    - list of components, pipes etc. in the application
  - imports
    - list of modules required to run the current module
  - providers
    - list of service classes used in the application
  - bootstrap
    - the component(s) need to be loaded by default when the module gets loaded
  - exports
    - list of components, pipes etc. exported from current module

Component

- in angular, component represents a screen (page) or part of page
- angular application development is a component oriented development
- contains files
  - .html: used for desiging
  - .css: used for adding styles
  - .ts: used to adding logic
  - .spec.ts: used to add test cases
- to create a component

  - create a class and call a decorator @Component()

      - decorator accepts a metatadata
        - selector
          - used to load the component in a parent component
          - use it as a tag
          - e.g. <app-root></app-root>
        - templateUrl
          - used to attache the view (html) which is the screen design
          - use it for designing the component's UI
        - styleUrls:
          - used to attach the styles on the html loaded in the component

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

  - declare the component in AppModule's declarations array

binding

- attaching one part to another
- types

  - **property binding**

    - **string interpolation**

      - way to get value from a class member

      - wrap the class member with {{}}

      - e.g.

```
  <!--- (string interpolation) getting value of
firstName -->
  <div>First name - {{firstName}} </div>
```

```
  export class FirstComponent {
    // class member
    firstName = 'steve'
  }
```

```
```

- **attribute binding**
  - used to bind value of a variable with an attribute
  - wrap the attribute in [] for binding the value with variable
  - e.g.

```html
<div [style.color]="color">Color: {{color}}</div>

<div
    [style.width]="size"
    [style.height]="size"
    [style.background-color]="color"
    class="box"></div>
```

```typescript


export class SecondComponent implements OnInit {
  color = 'green'
  size = '50px'
}
```

- ## class binding

- **event binding**

directive

- feature which directs the code to process the data
- types
  - built-in directives
    - attribute directive
    - structural directive
      - syntax
        *<directive>=""
      - **Ng For**
        - used to iterate over an array inside the html
        - e.g.

```html
  <div class="employee" *ngFor="let employee of
employees">
    <div>Name: {{employee['name']}}</div>
```

```
          <div>Id: {{employee['id']}}</div>
          <div>Department: {{employee['department']}}</div>
          <div>Role: {{employee['role']}}</div>
          <div>Salary: {{employee['salary']}}</div>
        </div>
```

- **Ng If**

- **Ng Switch**

- custom directives