=
===

https://codeburst.io/getting-started-with-angular-7-and-bootstrap-4-styling-6011b206080

# Angular

A framework for building client applications in HTML, CSS and Javascript / TypeScript.

**Why we need Angular.**
We need a way to properly structure our application.

1. Clean Structure.
2. Includes a lot of reusable code.
3. Makes our applications more testable.

**Architecture of Angular Applications.**

1. **Frontend(UI)**
    a. HTML
    b. CSS
    c. Typescript
    d. Angular

    | Http Services / Apis (Endpoints that are accessible via http protocol.)
    |
    |

2. **Backend (Data and processing)**
    a. Data + APIs
    b. Business Logic
    c. Databases.

=
===

## First Angular App

1. **Installations**
    a. **Nodejs --latest version**
        i. **Node --version.**
            1. (Min version for Angular is - 6.9)

    b. **Angular CLI**
        i. **npm install -g @angular/cli**
            1. ng --version

2. **Create new Angular Project**
    a. **Angular CLI**
        i. **ng new ProjectName**
            1. This generate bunch of files and folders.

3. **Visual Studio Code.**
    a. **Open command pallet.**
        i. **Command - code**
            1. **This opens the terminal,**
            2. **Terminal opens, in the folder where we created our project.**
            3. **Type in terminal - code .**
                a. This Opens up our project in Visual Studio Code Editor.
            4. **Try running the application through the below command.**
                a. **ng serve**
                    i. **http://localhost:4200**
                    ii. **Angular CLI compiles our application, generates bundles of HTML, CSS, JS**
                    iii. **To Test**
                        1. **On Browser hit the above url.**

=
===

4. **Project Structure**
    a. **e2e**
    b. **node_modules**
    c. **src** - Source code of our application
        i. **app**
            1. **Modules**
            2. **Components**
        ii. **Assets**
            1. **Store image, text , icons here.**
        iii. **Environments** - where we store the configuration settings of different environments.
            1. **Production.ts -** (one file for production environment)
            2. **Development.ts** - (one file for development environment)
        iv. **favicon.ico** - {icon displayed in the browser.}
        v. **Index.html**
        vi. **main.ts -** { starting point of our application - kind of main method.}
            1. BootStrapping the main Module
                a. Angular loads this module.
        vii. **polyfills.ts**
            1. Basically imports some scripts required for running angular.
        viii. **style.css**
        ix. **Test.ts** - setting up our test environment.

    d. **.angular-cli.json**
        i. This is a configuration File for angular.

    e. **.editorconfig**
        i. If you are working in a team.
        ii. All developers use the same setting in the editor

    f. **.gitignore**
        i. Excluding certain files and folders from your git Repository.

    g. **karma.conf.js**
        i. It is a Configuration File

=
===

ii. It is a Test runner for javascript code.

**h. package.json**

i. This is standard file, every node project has.
ii. Name : , Version:,
iii. Dependencies : {
1. Delete whatever you don't need. Like Animations.
2. }

**i. protractor.conf.js**

i. Tool for running end to end test for angular.

**j. tsconfig.json**

i. Bunch of settings for type-script compiler.

**k. tslint.json**

i. Bunch of setting for tslint.
ii. **tslint** is static analysis code for type script code.

**5. Tiny Changes to the project.**

**a. App**

**i. App.component.ts**

```
import { Component } from '@angular/core';

@Component({
 selector: 'my-app',
 templateUrl: './app.component.html',
 styleUrls: [ './app.component.css' ]
})
export class AppComponent  {
 name = 'Angular Test';
}
```

1. name = ' app';
2. to
3. name = ' Angular app';

After saving this file. Without reloading the browser page. Changes can be seen there.

=

===

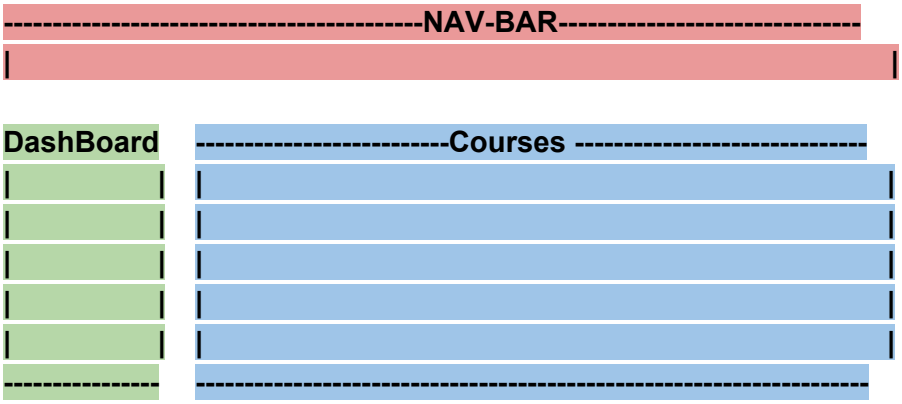
**History**

1. **AngularJs 1.x**
2. **Angular 2 - 2.0 2.1 2.2 2.3**
3. **Angular 4**

## Building Blocks of Angular Apps

1. **Component**
    a. **Data**
    b. **Html Template**
    c. **Logic for view.**

**Ex -**

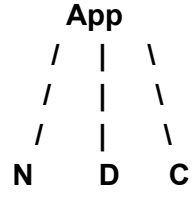



**Application can have one Component or many.**

**Three Different components.**

1. **NAV - BAR**
2. **DashBoard**
3. **Course**

**App - component or root component**




**N - Navigation Bar**

**D - Dashboard**

=
===

**C - Courses**

2. **Modules**
    a. **Example - UDemy**
        i. **Courses**
        ii. **Messages**
        iii. **Instructor**
        iv. **Admin**

## Creating Components

### 3 Steps

1. Create a Component
2. Register it in a module.
3. Add an element in html markup.

1. **Create a Component**
    a. app> new File - name = component-name.component.ts
        i. Ex → test.component.ts

```
//Import the decorator
import { Component } from '@angular/core';

//Decorator Function
@Component({
        selector : 'test ',
        template :  '<h2> Test Page </h2>'
        // <div class="test">".test"
        // <div id="test">"#test"
})
export class TestComponent{
}
```

```
import { Component} from '@angular/core';

@Component({
 selector : 'test',
 template : '<h1> Hello Form Test Component </h1>'
})
```

=

===

```
export class Test{

}
```

2. **Register it in a module.**
    a. app> app.module.ts

```
import { TestComponent } from './test.component';

@NgModule({
 imports:      [ BrowserModule, FormsModule ],
 declarations: [ AppComponent, TestComponent ],
 bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

**Extensions - Install Auto Import.**

3. **Add an element in html markup.**
    a. app > app.component.html

```
<p>
 Start editing to see some magic happen :)
</p>
<test><test>
```

Here test tag is selector. Written in test.component.ts

=
===

## Generate Components

**ng g c component-name**

This command create component-name directory, and all files associated with it. .css, .html, .ts etc.
Also It registers the component in app.module.ts

**Ex - ng g c login**

Login (folder)

- login.component.css
- login.component.html
- login.component.spec.ts
- login.component.ts

update app.module.ts

Inside the
declarations : [
AppComponent
LoginComponent,
]

=
===

Name - variable , and getName method  in component.ts- example

```
import {Component} from '@angular/core';

@Component({
 selector : 'test',
 template : '<h1> Hello Form Test Component  {{"name:"+name}}  {{"by method:"+getName()}}</h1>'
})
export class TestComponent{
 name = "Neo";

 getName(){
   return this.name;
 }
}
```

```
 template : '<h1> Hello Form Test Component  {{"name:"+name}}  {{"by method:"+getName()}}</h1>'
```

```
For multiple line use ``
 template : `<h1> Hello Form Test Component  {{"name:"+name}}  {{"by method:"+getName()}}</h1>`
```

=
===

For Loop - example

```
import {Component} from '@angular/core';

@Component({
 selector : 'test',
 template : `<h1> Hello Form Test Component  {{"name:"+name}}
              {{"by method:"+getName()}}</h1>

              <ul>
              <li *ngFor="let x of languages">
                {{x}}
              </li>
              </ul>
              `
})
export class TestComponent{
 name = "Neo";

 languages = ["Java", "C" ,"C#"];

 getName(){
   return this.name;
 }
}
```

=
===

# Service

test.service.ts

```
export class TestService{
	getSubjects(){
		return [“java”,”C”, “C#”,”HTML”];
	}
}
```

test.component.ts

```
export class TestComponent{
	subjects;
	constructor(){
		let service = new TestService();
		this.subjects = service.getSubjects();
	}

}
```

----------or----------

```
export class TestComponent{
	subjects;
	constructor(service: TestService){
		this.subjects = service.getSubjects();
	}

}
```

## Dependency Injection

**-Injecting or providing the dependencies of the class into its constructor.**

**=**
**===**

**In app.module.ts**

```
@NgModule({
providers: [ TestService
],
})
```

**Generating Services using Angular CLI**
**Command - ng g s servicename**
**Ex - ng g s test**

1. **Creates 2 files**
    a. **test.service.spec.ts**
    b. **test.service.ts**

**@Injectable() -> constructor (anotherService : )**

=
===

## Angular Directives

-ngFor
-ngIf
-ngSwitchCase
-ngClass
-ngStyle

## Two Way Binding

## Building Form For Angular Apps.

-Implement Form with different kinds of input fields.
-Display Validation errors
-Disable the submit button

Building Bootstrap form

1. Create component. ng g c contact-form

Adding Bootstrap in Angular Project
Add the following.
In angular.json:
"styles": [
"node_modules/bootstrap/dist/css/bootstrap.min.css",
"styles.css"
],
Install the dependencies.

=

===

## Basic form

```
<form>
 <div class ="form-group">
    <label for ="firstName">First Name</label>
    <input id="firstName" type ="text" class="form-control">
 </div>

 <div class ="form-group">
    <label for="comment">Comment </label>
    <textarea id="comment" cols ="30" rows="10" class="form-control">
    </textarea>
 </div>

<button class="btn btn-primary"> Submit </button>

</form>
```

## Applying Validation to above form

**property - touched , valid**

**input - field**

1. **required**
2. **ngModel**
3. **#anyName=”ngModel”**
    a. **Ex -firstName**

**Make division which will show the error**

1. **div class =”alert alert-danger” (bootstrap)**
2. ***ngIf=”firstName.touched && !firstName.valid”**

**Render the above division when input field is touched.**

=

===

```
 <div class ="form-group">
    <label for ="firstName">First Name</label>
    <input required ngModel name="firstName" type ="text" #firstName="ngModel"
class="form-control">
    <div class="alert alert-danger"  *ngIf="firstName.touched && !firstName.valid"
>First Name is required.</div>
 </div>
```

**Multiple Validations**

**property - errors**

1. **required**
2. **minlength = ”3”**
3. **maxlength = “10”**
4. **pattern = “banana”**

**Separate div for each validation error**

**errors**

- **required**
- **minlength**
  - **Actual length = 2**
  - **Required length = 3**

```
<div class ="form-group">
    <label for ="firstName">First Name</label>
    <input  required minlength="3" maxlength="10"  pattern="banana" ngModel
name="firstName" type ="text" #firstName="ngModel"  class="form-control">
    <div class="alert alert-danger"  *ngIf="firstName.touched && !firstName.valid" >
        <div *ngIf="firstName.errors.required" >First Name is required.</div>
        <div *ngIf="firstName.errors.minlength">Min Length = 3 </div>
        <div *ngIf="firstName.errors.pattern">First name doesn't match the pattern</div>
    </div>
<div>
```

**Render  min length dynamically . not hardcoded**

**Change the 1. To 2.**

```
1.     <div *ngIf="firstName.errors.minlength">Min Length = 3 </div>
```

=

===

```
 2.     <div *ngIf="firstName.errors.minlength">Min Length
{{firstName.errors.minlength.requiredlength}}</div>
```

**Highlight the input field.**
**class=”form-control ng-invalid ng-dirty ng-touched”**
**The division which contains the error.**

**contact-form.component.css**
**.form-control.ng-touched.ng-invalid{**
**border : 2px solid red;**
**}**

```
.form-control.ng-touched.ng-invalid{
 border : 2px solid red;
}
```

=
===

**Material in Angular -skipped**
**Animations in Angular -skipped**

## Routing and Navigation in Angular

- **Configuring Routes**
- **Implementing single page application (SPAs)**
- **Working with route and query parameters**
- **programmatic navigation**

**Modules**

- **Forms**
- **ReactiveForms**
- **Http**
- **Router**

**3 Steps to implement Navigation**

1. **Configure the routes.**
   Route is mapping of a pass to a component.
2. **Add router outlet.**
3. **Add links.**

Ex-**Best thing  The Page is not reloading.**

1. Create 2 different components
    1. Product
    2. Members

    ----------app.c.html----------
    <app-product>
    </app-product>

    <app-member>
    </app-member>

=

===

----------OUTPUT---------

product works!

member works!

2. In app.module.ts

A. `import { HttpModule } from '@angular/http';`

B. Add HttpModule in imports and

```
@NgModule({
 imports:      [ BrowserModule, FormsModule,HttpModule ],
 declarations: [ AppComponent, HelloComponent ,TestComponent, LoginComponent,
RegisterComponent, ContactFormComponent, ProductComponent, MemberComponent],
 bootstrap:    [ AppComponent ]
})
```

C. `import { RouterModule } from '@angular/router';`

D. In the imports section add one more thing.

a. Name of the view in path.

```
imports:      [ BrowserModule, FormsModule,HttpModule,
   RouterModule.forRoot(
      [
        {
          path : 'member',
          component : MemberComponent
        },
        {
          path : 'product',
          component : ProductComponent
        }
      ])
  ],
```

3. In app.component.html

<a routerLink = “/path”> TextLink </a>

=

===

```
<a routerLink ="/member"> Member </a>
<br>
<a routerLink ="/product"> Product </a>
```

4. Define an area where we want to show the data of the components.
   In app.component.html
   <router-outlet> </router-outlet>

**Service - ng g s service-name**

1. In app.module
   a. import { service-name } from ‘./service-name.service’
   b. Do entry inprovide
      i. ServiceName

Ex -

```
import { Injectable } from '@angular/core';

@Injectable()
export class MyDataService {

 constructor() { }

 obj = {

   id : "1",
   name : "Neeraj",
   Age :  "21"
 }

 success(){
   return "Successful";
 }

}
```

In order to use this service in another component
Let say

=

===

app.component

1. import this service in app.component.ts
2. 

class AppComponent{

constructor (private newService : MyDataService){ }

ngOnInit(){

console.log(this.newService.success())

}

}

```
import { Component } from '@angular/core';
import { MyDataService } from './my-data.service';

@Component({
 selector: 'my-app',
 templateUrl: './app.component.html',
 styleUrls: [ './app.component.css' ]
})
export class AppComponent  {
 name = 'Angular  by Neeraj';

 constructor( private newService: MyDataService){}

 ngOnInit(){
   console.log(this.newService.success())
   console.log(this.newService.obj)
this.newService.obj.name="Deepak";
   console.log(this.newService.obj)

 }
}
```

=
===

**HTTP Service**

Want to fetch the data from external source.
Call specific API and fetch the data in JSON Format.

1. Create one json file in a folder
    a. Data foldername
    b. students.json

```
[

 { "id" : "101",
   "name" : "Neeraj"
 },


 { "id" : "102",
   "name" : "Deepak"
 },

 { "id" : "103",
   "name" : "Manish"
  },
]
```

    c. In app.module.ts
        i. import HttpModule library
        ii. Add in imports array.

    d. MyDataService.service
        i. Import http

=

===

```
import(Http) from '@angular/http';
```

ii. Add constructor

```
constructor(private http:Http) { }
```

iii. Add fetchDataMethod.

```
fetchData(){
  //path of the student.json file
  this.http.get("src/data/students.json").subscribe(
    (data) => console.log(data)
  )
 }
```

iv. Add map in this.http.get().map().subscribe

1. ```
   import 'rxjs/add/operator/map';
   ```
2. ```
   fetchData(){
     //path of the student.json file
     return this.http.get('src/assets/students.json').map(
       (response) => response.json()
     ).subscribe(
       (data) => console.log(data)
     )
    }
   ```

v. Service - Add fetchData in it.

1. .
   ```
   fetchData(){
     //path of the student.json file
     return this.http.get('/assets/students.json')
    }
   ```

e. In app.component.ts call the fetchData method of that service.

i. class AppComponent{

1. ```
   import 'rxjs/add/operator/map';
   ```

2. ```
   ngOnInit{
   this.newService.fetchData().map(
     (response) => response.json()
   ```

=
===

```
    ).subscribe(
      data => {console.log(data)}
    )
  }
```

}

HTTPClient Get and Post.

https://github.com/Neo2007/JsonServerGit/blob/master/db.json

```
{
  "profiles":[


  { "id" : 101, "name" : "Neeraj","age" : 21},


  { "id" : 102, "name" : "Deepak","age" : 25},


  { "id" : 103, "name" : "Manish","age" : 26}


  ]


}
```

https://my-json-server.typicode.com/neo2007/JsonServerGit/profiles

=
===

Angular with Firebase

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/6.2.4/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#config-web-app -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyCyH0EgunsjdBrIz6nXswvusmtf8kZrLhk",
    authDomain: "crud-with-angular.firebaseapp.com",
    databaseURL: "https://crud-with-angular.firebaseio.com",
    projectId: "crud-with-angular",
    storageBucket: "",
    messagingSenderId: "782860383401",
    appId: "1:782860383401:web:5349fda89da7907a"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

=

===

## Angular Material Components

https://material.angular.io/