**1. Create basic route**

**App.module.ts:**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

import { Page1Component } from './page1/page1.component';

import { Page2Component } from './page2/page2.component';

import { RouterModule, Routes } from '@angular/router';

const appRoutes: Routes = [

  { path: '', redirectTo: '/page1', pathMatch: 'full' },

  { path: 'page1', component: Page1Component },

  { path: 'page2', component: Page2Component },

];

@NgModule({

  imports: [BrowserModule, FormsModule, RouterModule.forRoot(appRoutes)],

  declarations: [AppComponent, HelloComponent, Page1Component, Page2Component],

  bootstrap: [AppComponent],

})

export class AppModule {}

**app.component.ts:**

<ul>

      <li><a [routerLink] = "['/page1']">page1</a></li>

      <li><a [routerLink] = "['/page2']">page2</a></li>

   </ul>

   <router-outlet></router-outlet>

**2.Create a custom parameterized pipe to multiply with input argument.**

**Multiplier.pipe.ts:**

import {

   Pipe,

   PipeTransform

} from '@angular/core';

@Pipe ({

   name: 'Multiplier'

})

export class MultiplierPipe implements PipeTransform {

   transform(value: number, multiply: string): number {

      let mul = parseInt(multiply);

      return mul \* value

   }

}

**App.module.ts**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

import { HelloComponent } from './hello.component';

import { MultiplierPipe } from './multiplier.pipe';

//ts for custom pipeline

@NgModule({

  imports:      [ BrowserModule, FormsModule ],

  declarations: [ AppComponent, HelloComponent,MultiplierPipe ],

  bootstrap:    [ AppComponent ]

})

export class AppModule { }

**app.component.htm**

<hello name="{{ name }}"></hello>

<p>Multiplier: {{2 | Multiplier: 10}}</p>

**3.Create a directive to stylize the font color.**

**Ultapulta.directive.ts**

import { Directive, ElementRef, Renderer } from '@angular/core';

@Directive({

  selector: '[appUltapulta]'

})

export class UltapultaDirective {

  constructor(el: ElementRef, re: Renderer) {

     // Use use native element & renderer to render the element with styles

     re.setElementStyle(el.nativeElement, 'color', 'red');

   }

}

**App.module.ts:**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

import { UltapultaDirective } from './ultapulta.directive';

@NgModule({

  imports:      [ BrowserModule, FormsModule ],

  declarations: [ AppComponent, UltapultaDirective ],

  bootstrap:    [ AppComponent ]

})

export class AppModule { }

**app.component.html**

<h2 appUltapulta>

  Hello

</h2>

**4. Call an API using http direct in module and component and use it to show list of data.**

**App.module.ts**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';

@NgModule({

  imports:      [ BrowserModule, FormsModule, HttpClientModule ],

  declarations: [ AppComponent, HelloComponent ],

  bootstrap:    [ AppComponent ]

})

export class AppModule { }

**app.component.ts:**

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

import { HttpClient } from '@angular/common/http';

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: [ './app.component.css' ]

})

export class AppComponent  {

  constructor(private http: HttpClient){};

  private url\_search ="https://freeflowideas.in/myapp/apiservice/search.php";

  private url\_insert ="https://freeflowideas.in/myapp/apiservice/insert.php";

  data :any=[];

  getProfile(){

  this.http.get(this.url\_search)

    .subscribe(result => this.data = result.json());

  }

    id:string="";

    name:string="";

    age:string="";

    message:any="";

   postProfile1(){

   this.http.post(this.url\_insert,{id:this.id,name:this.name,age:this.age})

    .subscribe(result => this.message = result);

   }

}

5. **Call an API using service DI and httpClient module and component and use it to show list of data.**

**Data.service.ts**

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

import { Observable, of, throwError } from 'rxjs';

import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';

import { catchError, tap, map } from 'rxjs/operators';

import { Datamodal } from './datamodal';

const httpOptions = {

  headers: new HttpHeaders({'Content-Type': 'application/json'})

};

const apiUrl = "https://reqres.in/api/users?page=1";

const apiUrlID = "https://reqres.in/api/users";

@Injectable()

export class DataService {

  constructor(private http: HttpClient) { }

  getData(): Observable<Datamodal[]> {

  return this.http.get<Datamodal[]>(apiUrl)

    .pipe(

      tap(datas => console.log('HttpErrorResponse',datas)),

      catchError(this.handleError('getData', []))

    );

}

getDataID(x): Observable<Datamodal[]> {

  return this.http.get<Datamodal[]>(apiUrlID+'/'+x)

    .pipe(

      tap(datas => console.log('Fetched data')),

      catchError(this.handleError('getData', []))

    );

}

//error handling

private handleError<T> (operation = 'operation', result?: T) {

  return (error: any): Observable<T> => {

    // TODO: send the error to remote logging infrastructure

    console.error(error); // log to console instead

    // Let the app keep running by returning an empty result.

    return of(result as T);

  };

}

}

**app.module.ts**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';

import { HelloComponent } from './hello.component';

import { DataService } from './data.service';

@NgModule({

  imports:      [ BrowserModule, FormsModule, HttpClientModule ],

  declarations: [ AppComponent, HelloComponent],

  bootstrap:    [ AppComponent ],

  providers: [DataService]

})

export class AppModule { }

**app.component.ts:**

import { Component, OnInit } from '@angular/core';

import { DataService } from './data.service';

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: [ './app.component.css' ]

})

export class AppComponent implements OnInit {

  constructor(private ds: DataService) { }

private queryset:any;

private querysetID:any;

  ngOnInit(){

    return this.ds.getData()

    .subscribe(

      data => {this.queryset = data,console.log("response data:-"+data)},

      err=> console.log(err)

    )

  }

  view(x){

    console.log("View initialized"+x)

    return this.ds.getDataID(x)

    .subscribe(

      data => {this.querysetID = data,console.log("response data:-"+data)},

      err=> console.log(err)

    )

  }

}

**App.component.html**

<hello name="{{ name }}"></hello>

<p>Status Code:{{errStatus | json}}</p>

<table>

  <tr>

    <th>ID</th>

    <th>Email</th>

    <th>Avatar</th>

    <th>Actions</th>

  </tr>

  <tr \*ngFor="let x of queryset?.data">

    <td>{{x.id}}</td>

    <td>{{x.email}}</td>

    <td><img src ="{{x.avatar}}"></td>

    <td>

      <button class="btn btn-info" (click)="view(x.id)" data-toggle="modal" data-target="#myModal">View</button>

      <button class="btn btn-warning" (click)="edit()" data-toggle="modal" data-target="#editModal">Edit</button>

      <button class="btn btn-danger" (click)="delete()">Delete</button>

    </td>

  </tr>

</table>

**6.Promise API call.**

**App.module.ts:**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';

import { HelloComponent } from './hello.component';

import { DataService } from './data.service';

@NgModule({

  imports: [BrowserModule, FormsModule, HttpClientModule],

  declarations: [AppComponent, HelloComponent],

  bootstrap: [AppComponent],

  providers: [DataService],

})

export class AppModule {}

**data.service.ts:**

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

import { HttpClient } from '@angular/common/http';

import 'rxjs/add/operator/map';

import 'rxjs/add/operator/catch';

import 'rxjs/add/operator/toPromise';

@Injectable()

export class DataService {

  constructor(private http: HttpClient) {}

  private apiRoot = 'https://jsonplaceholder.typicode.com/todos';

  getBooksWithPromise(){

    return this.http

      .get(this.apiRoot)

      .toPromise()

      .then((res) => res)  
      .catch(this.handleErrorPromise);

  }

  private handleErrorPromise(error: any) {

    console.error(error.message || error);

    return Promise.reject(error.message || error);

  }

}

**App.component.ts:**

import { Component, OnInit } from '@angular/core';

import { DataService } from './data.service';

import { Data } from './data';

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css'],

})

export class AppComponent implements OnInit {

  promiseData: Promise<Data[]>;

  data: Data[];

  errorMessage: String;

  constructor(private serviceObject: DataService) {}

  ngOnInit(): void {

    this.serviceObject.getBooksWithPromise()

    .then(

      (resp) => ((this.data = resp), console.log(this.data)),

      (error) => (this.errorMessage = <any>error)

    );

  }

}

**App.component.html:**

<h3>Book Details after "then" to Promise</h3>

<ul \*ngFor="let x of data">

  <li>ID: {{ x.id }}</li>

  <li>UserID: {{ x.userId }}</li>

  <li>Title: {{ x.title }}</li>

  <li>Completed: {{ x.completed }}</li>

</ul>

<div \*ngIf="errorMessage" style="color:red">{{ errorMessage }}</div>

**7.Template Drive Form and Validation**

**App.module.ts**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

import { HelloComponent } from './hello.component';

@NgModule({

  imports:      [ BrowserModule, FormsModule ],

  declarations: [ AppComponent, HelloComponent ],

  bootstrap:    [ AppComponent ]

})

export class AppModule { }

**App.component.html**

<div class="container">

  <h2>Registration form:</h2>

  <form #fb="ngForm" (ngSubmit)="send(fb.value)">

      <div class="form-group">

        <label>Username</label>

        <input type="text" id="uname" name="uname" class="form-control" required minlength="4" appForbiddenName="bobb" #uname="ngModel" ngModel>

        <!-- username validation-->

        <div \*ngIf="uname.invalid && (uname.dirty || uname.touched)" class="alert alert-danger">

                  <div \*ngIf="uname.errors.required">Name is required.</div>

                  <div \*ngIf="uname.errors.minlength">Name must be at least 4 characters long.</div>

                  <div \*ngIf="uname.errors.forbiddenName">Name cannot be Bobb.</div>

        </div>

      </div>

      <div class="form-group">

        <label>Email</label>

        <input type="email" email="true" id="email" name="email" class="form-control" required #email="ngModel" ngModel>

        <!-- email validation-->

        <div \*ngIf="email.invalid && (email.dirty || email.touched)" class="alert alert-danger">

                    <div \*ngIf="email.errors.required">Email is required.</div>

                    <div \*ngIf="email.errors.email">Put a valid email id.</div>

        </div>

      </div>

    <button type="reset" class="btn btn-warning">Reset</button>

    <button type="submit" class="btn btn-primary" [disabled]="fb.invalid">Submit</button>

  </form>

</div>

**App.component.ts**

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

export class AppComponent {

  send(formob: any): void {

    console.log(formob);

  }

}

**8.Reactive Forms / Model Driven Forms and Validation**

**App.module.ts:**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

import { HelloComponent } from './hello.component';

@NgModule({

  imports:      [ BrowserModule, FormsModule, ReactiveFormsModule ],

  declarations: [ AppComponent, HelloComponent ],

  bootstrap:    [ AppComponent ]

})

export class AppModule { }

**app.component.ts:**

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

import { FormGroup, FormControl, Validators } from '@angular/forms';

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css'],

})

export class AppComponent {

  myform: FormGroup;

  constructor() {

    this.myform = new FormGroup({

      userName: new FormControl('', [

        Validators.required,

        Validators.minLength(5),

      ]),

    });

  }

  ngOnInit() {}

  get userName() {

    return this.myform.get('userName');

  }

  fun(myform: any) {

    console.log(myform.value);

  }

}

**app.component.html:**

<div class="container">

    <h2>Registration form:</h2>

    <form [formGroup]="myform" (ngSubmit)="fun(myform)">

        <div class="form-group">

          <input type="text"  formControlName="userName" class="form-control">

          <!-- validations -->

          <div \*ngIf="userName.invalid && (userName.dirty || userName.touched)" class="alert alert-danger">

              <div \*ngIf="userName.errors.required">Name is required.</div>

              <div \*ngIf="userName.errors.minlength">Name must be at least 4 characters long.</div>

          </div>

        </div>

      <button type="submit" class="btn btn-primary" [disabled]="myform.invalid">Submit</button>

    </form>

  </div>

<div>

  <pre>{{myform.value | json}}</pre>

</div>

**9.Data Transfer - @Input**

**App.component.html (Parent)**

<div style="background-color:skyblue;padding:20px">

    Parent Component

    <input type="text" #ptext>

    <button class="btn-danger" (click)="setdata(ptext.value)">Send to child</button>

    <hr>

    <!-- child component injected -->

    <app-child [parentData]="testdata"></app-child>

</div>

**App.component.ts (Parent)**

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

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: [ './app.component.css' ]

})

export class AppComponent  {

  name = 'Angular';

   public childData:string;

   public testdata:any;

   setdata(val){

      this.testdata = val;

   }

}

**Child.component.html (Child)**

<div style="background-color:#eee;padding:10px">

    <p>child component</p>

    <b>Data Recieved from parent:</b><b><i style="color:red">{{parentData}}</i></b>

</div>

**10. Data Transfer - @Output**

**Child.component.html (Child)**

<p>child works!</p>

{{parentData}}

<input type="text" #childtext (keyup)="onChange(childtext.value)">

**Child.component.ts (Child)**

import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';

@Component({

  selector: 'app-child',

  templateUrl: './child.component.html',

  styleUrls: ['./child.component.css'],

  inputs:['parentData'],

  outputs:['childEvent']

})

export class ChildComponent implements OnInit {

  constructor() { }

  ngOnInit() {

  }

  public parentData:string;

  childEvent = new EventEmitter<string>();

  onChange(value:string){

      this.childEvent.emit(value);

  }

}

**Parent.component.ts (Parent)**

import { Component, OnInit } from '@angular/core';

import {ChildComponent} from '../child/child.component';

@Component({

  selector: 'app-parent',

  templateUrl: './parent.component.html',

  styleUrls: ['./parent.component.css'],

})

export class ParentComponent implements OnInit {

  constructor() { }

  ngOnInit() {

  }

  public childData:string;

}

**Parent.component.html (Parent)**

<p>parent works!</p>

{{ childData }}

<input type="text" #ptext (keyup)="(0)" />

<app-child

  (childEvent)="childData = $event"

  [parentData]="ptext.value"

></app-child>