Angular

Documentation : https://angular.io/docs

prerequisite :

- install node js
- install angular cli :
  - npm install -g @angular/cli (g - globaly)

- To allow the execution of PowerShell scripts, which is needed for npm global binaries
  - Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

Default port : 4200

This is a framework not library.
ng version : command to check the version of the angular
ng new <name> : create the new workspace and initialize the new angular app.

Step to create angular app :
1. ng new hello-angular
2. routing needed the yes
3. css type needed
   created.....

to run application :
ng serve
and open localhost:4200 on browser

the counter application. the working of function and injecting the class function inside html.

```
<div class="container">
  <header>
```

```
    <h1>Counterr App</h1>
  </header>
  <main>
    <h2>Current count is : {{count}}</h2>
  </main>
  <button (click)="handleDecrese()" >Decrement</button>** this ()="" is specified
inside the angular framework for injecting functiond. In angular all the fucntions are
click , blur , submit not onClick, onSubmit etc.
  <button (click)="handleReset()" >Reset</button>
  <button (click)="handleIncrese()">Increment</button>
</div>
```

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title:string = 'counter-app';
  count:number = 0;
  handleIncrese = () => {
    this.count += 1;
  }
  handleDecrese = () => {
    this.count -= 1;
  }
  handleReset = () => {
    this.count = 0;
  }
}
```

- injecting values from class to html via {{ }} called interpolation.
- (event)="function" that's called event binding.

```
<div *ngIf="countryS" class="card card-body">
//injection of checking block in angular
```

command to create new component :

ng generate componentName foldername/icon

command to create service :

ng generate service foldername/serviceName or shortcut : - ng g s foldername/serviceName

generate module and routing

ng generate module moduleName --routing or ng g moduleName module --routing

RxJS

- it solves the problem of handling asynchronous call with multiple events.
- using observable from rxjs : their are more then one way to create the observable : one of the way is :

```
import { fromEvent } from 'rxjs';
fromEvent(document, 'click').subscribe(() => {
  console.log('clicked');
});
```

using operators and pipe example :

```
import { fromEvent } from 'rxjs';
import { scan } from 'rxjs/operators';
//here best part is the count variable is integrated perfectly because the count is only accessable by pipe and subscriber so no one can modify the count from out side.
// we can use more then one pipe in one event listener
fromEvent(document, 'click')
  .pipe(scan((count) => count + 1, 0))
  .subscribe((count) => {
    console.log(`subscriber : ${count}`);
});
```

creating observable and observing:

```
import { Observable } from 'rxjs';
// creating observable
const observable = new Observable((subscriber) => {
  subscriber.next(1);
  subscriber.next(2);
  subscriber.next(3);
  setTimeout(() => {
    subscriber.next(4);
    subscriber.next(5);
    subscriber.complete();
  }, 2000);
});
console.log('About to Subscribe');
//observing the observable
observable.subscribe({
  next(x) {
    console.log('the return is : ', x);
  },
```

```
    console.log('the error is : ', err);
  },
  complete() {
    console.log('Completed');
  },
});
console.log('All Done');
```

TypeScript

```
const variablename : type = value
different types are :
string
number
boolean
null
undefined
any( when not sure about data , can any of the above )
```

if same as java script

for of gives values and for in gives index.

function declaration and uses same as JavaScript.

** Decorators