Angular File Structure

Package.json--> firstly loaded,project name,version,script,dependency,dev-dependency,external package
node modules-->all dependency and related file
src folder-->code write inside
environment--> production.ts,declare constant,url config
assets--> static file,images
index.html-->entry point of html page
main.js-->entry point of js (linked with html)
style.css--> global css
app-->root default module(routing,html,css,module,ts)
module--> combination of components
routing--> which page render at what url,config routes
editorconfig--> editor related setting
karma.config--> testing use
package.json.lock-->package.json related logs
tslint--> validation of what we are using
tsconfig.base--> editor control,auto suggest
tsconfig.spec.json-->testing purpose
tsconfig.app-->app config control
e2e folder-->end to end testing

Interpolation

Interpolation--> move the value of properties from ts file to html file
****
{{name}}- -properties
getName(){}--function
ts file--> want properties in function use this keyword
 name='chail';
 getName(){
  return this.name;
 }
 ****
 create object in ts file
 obj={
 name:'chail',
 age:18,
 }
 now in html--> {{obj}}==> [object object]
          -->{{obj.name}}--> chail
          -->{{obj.age}}--> 18
****
create array
a=10;
b=40;
siteurl=window.location.href;
arr=['first','second','third']
{{arr}}==> first,second,third
{{arr[0]}}==> first
{{arr.length}}==>3
{{10+10}}==>20
{{a=10}}==>error
{{a+b}}==>50

{{siteurl}}==>http://localhost:4200/
{{window.location.href}}==>error

Components and Modules

piece of code for reuse (header,navbar,footer etc)
component have 4 files
command - ng g c nameofcomponent(userList-->user-list.comp..)

module*********
mechanism to group component,directive,pipes,services that are related
command --> ng g m modulename(created 1 file-->users.module.ts)
--> ng g c modulename/componentname(4 file created)
now export component from users.module.ts(
exports:[LoginComponent]
and import in app.module.ts(import:[UserModule]
now in app.html--><app-login>

Function in Angular

Call A function in angular
step 1- define a function
step 2- make button and call functin on button click
step 3- pass param with function call
whenever we call button then both thing should be in same component
btn-html--> <button (click)="getName()">getmyname</button>
define-ts--> getName(){
alert("chail singh");

}
**pass param as a string
btn-html--> <button (click)="getName('chail')">getmyname</button>
define-ts--> getName(name){
alert(name);

}

**pass as variable

btn-html--> <button (click)="getName(yourName)">getmyname</button>
define-ts-->
youName="chail";

```
  getName(name){
    alert(name);

  }
```

Event in Angular

click,keyup,keyup with enter and space,keydown,blur,mouseover and mouse leave, get values on textbox

ts--> myEvent(evt){
console.log(evt)
}

html-->>

```
<button (click)="myEvent('click event')">click me</button>
<input type="text" (keyup)="myEvent('keyup event')">
<input type="text" #box (keyup)="myEvent(box.value)">
<input type="text" #box (keyup.space)="myEvent(box.value)">
<input type="text" #box (keydown.space)="myEvent(box.value)">
<input type="text" #box (keydown)="myEvent(box.value)">
<input type="text" #box (keydown.enter)="myEvent(box.value)">
<input type="text" #box (keydown.space)="myEvent(box.value)">
<input type="text" (blur)="myEvent(blur event)">

<div (mouseover)="myEvent('mouse in')"
style="background-color:red;">
Mouse Event
</div>

<div (mouseleave)="myEvent('mouse in')"
style="background-color:red;">
  Mouse Event
  </div>
```

can we use more than one event at a time in angular

Yes You Can Bind To The Same Event Several Times In Angular 2 Beta 17.

Get Textbox value

```
<input type="text" (keyup)="getVal($event.target.value)">
  getVal(val){
    console.log(val);
  }
  **

<input type="text" (keyup)="getVal($event.target.value)">
{{currentVal}}
currentVal=""
 myEvent(val){
    console.log(val);
    this.currentVal=val
  }
  **
```

get value with button click

```
<input type="text" #box>
<button (click)="getVal(box.value)">Get Value</button>
{{currentVal}}

  currentVal=""
  getVal(val:any){
    this.currentVal=val
  }
```

Property binding in angular

property binding****
name="chailsingh";
<input type="text" value="chai">
<input type="text" value={{name}}>
<input type="text" [value]=name>
note:above both gives same output so what's difference

***
disabledBox=true;
enableBox(){

  this.disabledBox=false;
}

<input type="text" disabled={{disabledBox}} value={{name}}>
<input type="text" [disabled]="disabledBox" [value]=name>
<button (click)="enableBox()">Enable button</button>

imp:property binding can handle disable but interpolation can't

If-else in angular

if**
```
  show=false;
<h1 *ngIf="show" >
  if block
</h1>
```

if-else**
```
show false;
<h1 *ngIf="show else elseBlock" >
  if block
</h1>
<ng-template #elseBlock>
  <h1>else block</h1>
</ng-template>
```

**if else with string
```
show="yes";
```

```
<h1 *ngIf="show=='yes' else elseBlock" >
  if block
</h1>
<ng-template #elseBlock>
  <h1>else block</h1>
</ng-template>

**
<h1 *ngIf="show=='yes'; then ifBlock else elseBlock" >

</h1>
<ng-template #ifBlock>
  <h1>if block</h1>
</ng-template>
<ng-template #elseBlock>
  <h1>else block</h1>
</ng-template>


 color="red";
<ng-template [ngIf]="color=='red'">
  <h1>Red Block</h1>
</ng-template>
```

Switch case

```
**switch case
 color="green";
<div [ngSwitch]="color">
<h2 *ngSwitchCase="'red'">
  Red color
</h2>
```

```
<h2 *ngSwitchCase="'green'">
  green color
</h2>
<h2 *ngSwitchCase="'blue'">
  blue color
</h2>
</div>
```

For loop

```
Normal for loop-->
data=['chail','mitesh','ashu']

<h4 *ngFor="let item of data">
  {{item}}
</h4>

**for with object array

data=[
  {
    name:'chail',
    age:20
  },
```

```
  {
    name:'mitesh',
    age:25
  }
]

<h4 *ngFor="let item of data">
  {{item.name}}
  {{item.age}}
</h4>

**for with table
data=[
  {
    name:'chail',
    age:20
  },
  {
    name:'mitesh',
    age:25
  },
  {
    name:'ashu',
    age:30
  }
]

<table border="1">
  <tr>
    <td>Name</td>
    <td>Age</td>
  </tr>
  <tr *ngFor="let item of data">
    <td>{{item.name}}</td>
    <td>{{item.age}}</td>
```

```
  </tr>
</table>
```

Form in angular

step 1 import formmodule

```
<form #simpleForm="ngForm"
(ngSubmit)="getFormValue(simpleForm.value)" >
  <input ngModel type="text" name="username" placeholder="enter
name"><br><br>
  <input ngModel type="password" name="password"
placeholder="enter password"><br><br>
  <input ngModel type="text" name="address" placeholder="enter
address"><br><br>
  <input ngModel type="text" name="age" placeholder="enter
age"><br><br>
  <button>Get form value</button><br>
</form>

getFormValue(value:any){
  console.log(value)
```

}

Header and Footer

ng g c header
ng g c footer
app.comp[onent.html-->
<app-header></app-header>
<h1>Header and footer in angular</h1>
<app-footer></app-footer>

styles.css
```
body{
    margin: 0px;
    padding: 0px;
}
```
header
<h3>Welcome to header</h3>
```
h3{
    background-color: skyblue;
     margin: 0;
    padding: 10px;
}
```

footer
<h5>copyright</h5>
h5{
    background-color: skyblue;
    position: absolute;
    bottom:0;
    width: 100%;
    margin: 0;
    padding: 10px;
}

app.html
<app-header></app-header>
<h1>Header and footer in angular</h1>
<app-footer></app-footer>

Style binding

difference between normal style and style binding
--normal style can't be use as a dynamically
***
color="orange"
<h1 style="color: red;">normal style</h1>
<h1 [style.color]="'green'">style binding</h1>
<h1 [style.color]="color">style binding dynamic </h1>

change color after click on button**
color="orange"

changeColor(){
    this.color="blue";
  }

<h1 [style.color]="color">style binding dynamic </h1>

<button (click)="changeColor()">change color</button>

conditional**
err=true;
color="orange";
<h1 [style.color]="err?'red':'blue'">conditional</h1>

Add Bootstrap in angular

command install bootstrap--> ng add @ng-bootstrap/ng-bootstrap
try to use different methods

Angular Material UI

add material ui in angular--> ng add @angular/material
let's use components-->
1)use button and slider
go to module.ts--> import {MatButtonModule} from '@angular/material/button';
-->import {MatSliderModule} from '@angular/material/slider';

go to app.html-->
for button--><button mat-raised-button color="warn" >angular button</button>

for slider-->
<mat-slider min="1" max="5" step="0.5" value="1.5"></mat-slider>
<mat-slider
  thumbLabel
  [displayWith]="formatLabel"
  tickInterval="1000"
  step="1000"
  min="0"
  max="100000"

```
  aria-label="units"></mat-slider>

ts file-->
  formatLabel(value: number) {
    if (value >= 1000) {
      return Math.round(value / 1000) + 'k';
    }
return value;}
```

Parent to child data pass

steps to be followed-->
make users component
make it child of app component
pass data to app to child component
display data in child component
****
ng g c users --> create users component
put app-users in app.html file
define variable data="chail" in app.ts
go to app.html--> <app-users [hero]="data"></app-users>
go to users.ts --> import input
@input() hero
go to users.html <h3>{{hero}}</h3>

**pass object
1)change in app.ts

```
  data={
    name:"chail",
    age:18
  }
```

2) change in users.html

```
<h3>{{hero.name}}</h3>
 <h3>{{hero.age}}</h3>
```

Reuse of component

make user component
use it inside app componen as child
pass data from p to c
use for loop and result child component
app.ts→

```
 users=[
   {
     name:'chail',
     age:20
   },
   {
     name:'sahil',
     age:30
   }
 ]
```

app.html→

```
<div *ngFor="let data of users">
  <app-users [hero]="data"></app-users>
</div>
```

users.ts→

```
 @Input() hero:any;
```

users.html→

```
<h1>{{hero.name}}</h1>
<h1>{{hero.age}}</h1>
```

make user component
use it inside app componen as child
pass data from p to c
use for loop and result child component

app.ts→

```
  users=[
    {
      name:'chail',
      age:20
    },
    {
      name:'sahil',
      age:30


}
  ]
```

app.html-->

```
<div *ngFor="let data of users">
  <app-users [hero]="data"></app-users>
</div>
```

users.ts-->

```
 @Input() hero:any;
```

users.html-->

```
<h1>{{hero.name}}</h1>
<h1>{{hero.age}}</h1>
```

Send Data Child To Parents Component

make user component
use it inside app component as child
pass data from child to parent with EventEmitter

app.component.ts-->

```
  parentComponent(data:any){
    console.log(data);
  }
```

app.html-->

```
<app-users
(parentFunction)="parentComponent($event)"></app-users>
```

users.ts-->

```
 @Output() parentFunction:EventEmitter<any>=new EventEmitter()
   constructor() { }

  ngOnInit(): void {
    this.parentFunction.emit("heelo");
        this.parentFunction.emit({name:'chail',age:30})

  }
```

send data using button
changes in
user.html-->

```
<button (click)="sendData()">SendData</button>
```

user.ts

```
  sendData(){
  let item={name:'chail',age:30}
    this.parentFunction.emit(item)
  }
```

PIPES IN ANGULAR

pipe--change the format
pipe with string**
{{name | uppercase }}
{{name | lowercase }}
{{name | titlecase }}
pipe with date**
{{today | date:'fullDate'}}
{{today | date}}
pipe with slice**
{{str | slice:2:5}}
pipe with currency***
{{money | currency:'USD'}}

Routing Basics

make 2 component
add routing in app-routing file
write code in html for making routing link
test routing
Note: if you did not select routing option on the project make time
then run this command to enable routing-->
ng generate module app-routing --flat --module=app
Now make two component
ng g c user
ng gc admin
go to app-routing file and make array of object in routes

```
const routes: Routes = [
  {
    path:'user',
    component:'userComponent'
  },
  {
    path:'admin',
    component:'adminComponent'
  }
];
```

now go to app.html and

```
<a routerLink="user">user</a>
<a routerLink="admin">admin</a>
```

now add router-outlet tag in app.html

```
<router-outlet></router-outlet>
```

404 Page Not Found

make a component
use it as 404 page with wildcard routing
step 1- ng g c page-not-found
step 2-

```
 {
   path:'**',
   component:PageNotFoundComponent

 }
```

step 3-

```
 <a routerLink="about">about</a>
 <router-outlet></router-outlet>
```

since we don't have about page so it will go to the page not found component. ** shows wildcard component

Custom Directive

what is Directive--to manupulate in dom and repeat dom (if else etc,some hide some show)
default directive--ngFor loop,ngIf,ngSwitch,binding
how to make custom directive--
run command ng g directive--ng g directive customStyle
use it with html-->

step 1 ng g directive customStyle
step 2 copy selector from customstyle directive and paste into html
step 3 <h2 appCustomStyle>custom directive</h2>
step 4 go to custom.directive and import ElementRef and use it in constructor as a instance and style-->

```
import {ElementRef} from '@angular/core';
  constructor(private el:ElementRef) {
    el.nativeElement.style.color="red"
   }
```

Service-Basics

what is serice in angular
how to make service
how to use service
example
service used to share data among the component
service is neither depend on module nor component dependent
you can use one service anywhere
ng g s servicename
ng g s userData

step 1 ng g s userData
step2 service.ts-->

```
  getData(){
    return {
      name:'komal',
      age:20,
      id:1
    }
  }
```

step 3 app.ts

```
name="";
  constructor(private user:UserDataService){
    console.warn(this.user.getData());
    let data=this.user.getData();
    this.name=data.name;
  }
```

step 4 app.html => {{name}}

Calling A simple API

what is api
how to make service
how to fetch api data in service
import in component
get data in component

step 1 ng g service airline
step 2 import HttpClient in service and import hHttpClientModule in app.module.ts
step 3 user instance of http in service-constructor and create method

```
  constructor(private _http:HttpClient) { }
  getFakeData(){
    let
url="https://api.instantwebtools.net/v1/passenger?page=0&size=10"
;
    return this._http.get(url);
  }
```

step 4 go to component
app.ts

```
  constructor(private fake:FakeServiceService){
       this.fake.getFakeData().subscribe(data=>{
        console.warn(data);
       })
  }
```

API-Data List in Table

continue from 24 file
make table in html file
use for loop for render data with table

step 1 service.ts-->

```
  constructor(private _http:HttpClient) { }
   getAirline(){
   return
this._http.get("https://api.instantwebtools.net/v1/passenger?page=0
&size=10");

   }
```

step 2 component.ts

```
 data:any=[];
  constructor(private fake:FakeServiceService){
       this.fake.getFakeData().subscribe(data=>{
        console.warn(data);
       this.data=data;
       })
  }
```

component.html

```
<table >
  <tr>
    <td>Id</td>
    <td>userId</td>
    <td>Title</td>
  </tr>
  <tr *ngFor="let item of data">
    <td>{{item.Id}}</td>
    <td>{{item.userId}}</td>
    <td>{{item.Title}}</td>
  </tr>
</table>
```

Model in Angular

what is model--it define the data structure and validate the data
{
name:'chail',
id:20
}
model is a part of ts ,it is not a part of angular

how to make and use it

******
step 1 go to app.component.ts and
create an
interface dataType{
name:string,
id:number,
indian:boolean,
address:any
}
create an function
getData(){
const data:dataType={
name:'chail',
id:100,
indian:true,
address:"giftcity"
}
return data;
}

use model with service file*****
create an interface dataType{
name:string,
id:number,

indian:boolean,
address:any
}
create an function
getData(){
const data:dataType={
name:'chail',
id:100,
indian:true,
address:"giftcity"
}
return data;
}

***best way
create a file model.ts
export interface dataType{
name:string,
id:number,
indian:boolean,
address:any
}
Now import wherever you want to use it
let's use in userService
import{dataType} from****
getData(){
const data:dataType={
name:string,
id:number,
indian:boolean,
address:any
}
}

with model with other file

use model with component file

what is model--it define the data structure and validate the data
{
name:'chail',
id:20
}
model is a part of ts ,it is not a part of angular

how to make and use it

******
step 1 go to app.component.ts and
create an
interface dataType{
name:string,
id:number,
indian:boolean,
address:any
}
create an function
getData(){
const data:dataType={
name:'chail',
id:100,
indian:true,
address:"giftcity"
}
return data;
}

use model with service file*****
create an interface dataType{
name:string,

id:number,
indian:boolean,
address:any
}
create an function
getData(){
const data:dataType={
name:'chail',
id:100,
indian:true,
address:"giftcity"
}
return data;
}

***best way
create a file model.ts
export interface dataType{
name:string,
id:number,
indian:boolean,
address:any
}
Now import wherever you want to use it
let's use in userService
import{dataType} from****
getData(){
const data:dataType={
name:string,
id:number,
indian:boolean,
address:any
}
}

Module + Routing

ng g m users --> create module with name users
ng g c users/login -->create component with name login in users module
step 1 go to users.module.ts
exports:[
LoginComponent
]
step 2 go to app.module.ts
imports:[
UsersModule
]
step 3 go to app.html
<app-login></app-login>

***routing
go to app.routing
step 1

```
const routes: Routes = [
  {
    path:'login',
    component:LoginComponent
  },
  {
    path:'signup',
    component:SignupComponent
  }
];
```

step 2 go to app.html

```
<ul>
    <li><a routerLink="login">Login</a></li>
   <li><a routerLink="signup">Signup</a></li>
</ul>
<router-outlet></router-outlet>
```

Routing Module

Note : till date we were defining all routes of all module in single file but now we can make a different routing file for different module and configure routes for that particular module only to avoid complexity and we don't want all routes to be load at a time.

step 1
ng g m admin --routing
ng g c admin/login
ng g c admin/list
step 2 go to admin.module.ts and import adminModule
step3 go to admin-routing.module.ts and define routes
const routes: Routes=[
{path:'login',component:LoginComponent},{path:'list',component:List Component}]
step 4 go to app.html

```
<ul>
    <li><a routerLink="login">Login</a></li>
    <li><a routerLink="list">List</a></li>
</ul>
<router-outlet></router-outlet>
```

Group Routing

if we have same component name in different module then which component will load in that case we do group routing

step 1 ng g m admin --routing
step 2 ng g c admin/login
step 3 ng g c admin/list
step 4 ng g m users --routing
step 5 ng g c users/login
step 6 ng g c users/list
step 7 go to app.html→

```
<h1>Admin</h1>
<ul>
    <li><a routerLink="admin/login">Login</a></li>
    <li><a routerLink="admin/list">List</a></li>
</ul>
<h1>users</h1>
<ul>
    <li><a routerLink="users/login">Login</a></li>
    <li><a routerLink="users/list">List</a></li>
</ul>
<router-outlet></router-outlet>
```

step 7 go to app.module.ts

```
  imports: [
    AdminModule,
    UsersModule
  ]
```

step 8 go to users-routing

```
const routes: Routes = [
```

```
  {
  path:'users',children:[
  {path:' login',component:LoginComponent},
  {path:' list',component:ListComponent}
  ]
}
];
```

step 9 go to admin-routing

```
const routes: Routes = [
  {
  path:'admin',children:[
  {path:' login',component:LoginComponent},
  {path:' list',component:ListComponent}
  ]
}
];
```

Angular Lazy Loading

Normal loading--> in this all routes loads at a time so app slow
Lazy Loading --> only clickable routes load
step 1 make a module--> ng g m admin --routing
step 2 make two component--> ng g c admin/login
--> ng g c admin/list
step 3 app.html-->

```
<ul>
  <li><a routerLink="admin/login">login</a></li>
  <li><a routerLink="admin/list">list</a></li>
</ul>
<router-outlet></router-outlet>
```

step 4 admin.routing

```
const routes: Routes = [
  {
    path:'login',
    component:LoginComponent
  },
  {
    path:'list',
    component:ListComponent
  }
];
```

Note: for lazy loading we don't import module directly into app.module.ts rather we use loadchildren in app-routing
use module in lazy loading way
step 5 app-routing.ts

```
const routes: Routes = [
  {path:'admin',loadChildren:()=>import('./admin/admin.module')
  .then(mod=>mod.AdminModule)}];
```

cross check:=>
console.log("admin module) into admin.ts file
this will only load when admin component will laod

Lazy Loading in Component

step 1 ng g c userlist
step 2 ng g c adminlist
step 3 app.html

```
<button (click)="loadAdmin()">Load Admin List</button>
<button (click)="loadUser()">Load User List</button>
```

step 5 app.component.ts

```
  constructor(private _vc:ViewContainerRef,
    private _cfr:ComponentFactoryResolver){
  }
 async loadAdmin(){
    this._vc.clear();
    const {AdminlistComponent}= await
import('./adminlist/adminlist.component')
    this._vc.createComponent(
      this._cfr.resolveComponentFactory(AdminlistComponent)
    )
  }
  async loadUser(){
    this._vc.clear();
    const {UserlistComponent}= await
import('./userlist/userlist.component')
    this._vc.createComponent(
      this._cfr.resolveComponentFactory(UserlistComponent)
    )
  }
```

Note: ViewContainerRef-->it create a div like container in which the dynamic component will load

ComponentFactoryResolver--> it will convert the dynamic code into component

Form Introduction

Form Use-->Login & create Account,save feedback,submit data
Type of forms--> 1 reactive,template driven
reactive--> control data in component.ts file
template driven form --> control and handle data in component.html file

Workflow of data=>
Form->ts file->service->server

Template Driven Form

import form module in app.module file
write html form in component.html
get data in component.ts file

step 1 import formModule in app.module
step 2 app.html

```
<form #userForm="ngForm"
(ngSubmit)="onSubmit(userForm.value)">
    <input ngModel type="email" name="email" placeholder="enter email"><br>
     <input ngModel type="password" name="password" placeholder="enter password"><br>
     <br>
     <button type="submit">login</button>
</form>
```

step 3 app.component.ts

```
  onSubmit(data:any){
    console.log(data);
  }
```

Validation in Template driven form

step 1 import formModule in module.ts
step 2 html file

```
<div class="col-md-6">
  <form #userForm="ngForm"
(ngSubmit)="onSubmit(userForm.value)">
    <div class="form-group">
      <label for="exampleInputEmail">Enter Email</label>
      <input required #email="ngModel" ngModel name="useremail"
type="email" class="form-control"  >
    </div>
    <span *ngIf="email.invalid && email.touched" class="error">email
required</span>
    <div class="form-group">
      <label for="exampleInputPassword">Enter Password</label>
      <input #password="ngModel" required ngModel
name="password"  type="password" class="form-control"  >
    </div>
    <button type="submit">submit</button>
  </form>
</div>
```

step 3 css file

```
input.ng-valid{
    border:1px solid green;
}
input.ng-invalid{
    border:1px solid red;
}
.error{color: red}
```

step 4 ts file

```
  onSubmit(data:any){
    console.log(data);
  }
```

Reactive Form in Angular

step 1 module.ts
import ReactiveFormModules
step 2 html file

```
<form [formGroup]="loginForm" (ngSubmit)="collectData()">
  <input type="text" name="username"
formControlName="username">
  <input type="password" name="password"
formControlName="password">
   <button type="submit">submit</button>
</form>
```

step 3 component.ts file
import FormControl and FormGrooup

```
loginForm=new FormGroup({
  username:new FormControl('default name'),
  password:new FormControl('')
})
collectData(){
console.log(this.loginForm.value)
}
```

Validation in reactive form

Angular Reactive form validation
-->
import reactive Form module
make html

define form group
get form value
apply value

step 1 app.module.ts-import ReactiveFormModule
step 2 html.file

```
<div class="col-md-6">
   <form [formGroup]="loginForm" >
      <div class="form-group">
         <label for="exampleInputEmail">Email</label>
         <input type="email" class="form-control"
formControlName="email"  >
      </div>
      <span class="red-error" *ngIf="email.invalid &&
email.touched">Email Required</span>
      <div class="form-group">
         <label for="exampleInputPassword">Password</label>
         <input type="password" class="form-control"
formControlName="password" >
      </div>
      <button type="submit" class="btn
btn-primary">submit</button>
   </form>
</div>
```

step 3 app.component.ts
import FormControl and FormGroupName,Validators

```
loginForm=new FormGroup(
 {
 email:new FormGroup('',Validators.required),
 password:new FormControl('')

 })

 get email(){
```

return this.loginForm.get('email')
}
step 4 app.css
input.ng-invalid{
border:1px solid red;
}
input.ng-valid{
border:1px solid green;
}
.red-error{
color:red;
}

Pre Filled Form

step 1 -> import formmodule in module.ts
step 2 - html file
<form #userForm="ngForm" (ngSubmit)="userForm.value">
  <input type="email" name="email"
[ngModel]="userData.email">
  <input type="password" name="password"
[ngModel]="userData.password">]
  <button type="submit">save</button>
</form>
step 3 component.ts
userData={
email:"test@email.com",
password:"123@abc"
}

bind form with ngModel

define data

set form value