问题：

1.angular-cli版本已经升级至7

2.更新项目后，重新安装一下依赖

复习：

1.变量、常量

```
let a: string;
let b: number;
let c: boolean;
let d: string[];
let e = 'e';
const PI = 3.1415926;

//元组
const tuple: [string, number];
tuple = ['tom', 20];

//任意类型
const result: any;
const foo: any[];

//void
function fn():void{}

//枚举enum
enum Color {Red,Yellow,Blue}
Color.Red;
```

TypeScript

1. 接口: 约束类型结构
2. 类 class
    - 基本用法
    - 继承

        ```
        class Animal {}
        class Dog extends Animal {}
        ```

    - 修饰符：public 公共、private 私有、protected受保护

        - public：成员默认是public，表示成员可以自由访问
        - private：表示成员只能在当前类内部使用
        - protected：表示成员能够在当前类和子类中使用
        - readonly: 只读，当属性只能读取，不能设置
        - 参数属性：给构造函数参数加上修饰符，能够同时定义并初始化成员属性
    - 静态成员：通过static关键字修饰属性、方法，将来通过类名直接访问

```
class Grid {
  // origin原点是所有网格都会用到的属性
  static origin = {x: 0, y: 0};

  distance(point: Point) {
    const xDist = point.x - Grid.origin.x;
    const yDist = point.y - Grid.origin.y;
    return Math.sqrt(xDist * xDist + yDist * yDist);
  }
}

const grid = new Grid();
console.log(Grid.origin.x, Grid.origin.y);
console.log(grid.distance({x: 3, y: 4}));
```

    - 存取器：当获取和设置属性时有额外逻辑时可以使用存储器（getter、setter)

```
class Employee {
  private _fullName: string;
  get fullName(): string {
    return this._fullName;
  }

  set fullName(value: string) {
    console.log('管理员修改了雇员名称');
    this._fullName = value;
  }
}
```

3. 函数
    - ts中函数的参数是必须的
    - 函数的可选参数
    - 函数参数的默认值

```
// first是必要参数，last可选，last有默认值
function buildName(first: string = 'James', last?: string = 'Harden') {
  return first + last;
}

buildName('tom', 'jerry');
buildName('tom'); // 可选参 last?
buildName(); // 默认值
```

4. 泛型：可以使用泛型Generic来创建可重用组件，一个组件可以支持多种类型的数据

```
interface Lengthwise {
  length: number;
}

// 使用泛型及泛型约束：
// T称为类型变量，它是一种特殊的变量，只用于表示类型而不是值，保证参数和返回值的类型
// 泛型约束保证T中含有length属性
function useGeneric<T extends Lengthwise>(arg: T): T {
  console.log(arg.length);
  return arg;
}

// 用法1：完整语法
useGeneric<string>('myString'); // myString
// 用法2：利用类型推论省略<number>
useGeneric({length: 1, other: 'bla'}); // 1
```

- 泛型接口

```
interface Result<T, U> {
  success: boolean;
  data?: T;
  message?: U;
}

interface User {
  id: number;
  name: string;
}

const r: Result<User> = {
  success: true,
  data: {id: 1, name: 'tom'}
};
```

- 泛型类

```
class Result<T> {
  constructor(public success: boolean, public data: T) {
  }
}

const r2: Result<User> = new Result<User>(true, {id: 1, name: 'tom'});
console.log(r2.success);
console.log(r2.data);
```

5. 模块
    - 导出
        - 具名导出

```
export const HOST = 'http://localhost:4200';

export function add(a, b) {
  return a + b;
}

export class Foo {
  bar: string;
}
```

        - 导出语句

```
// 导出语句
const Bar = 'bar';
const abc = 'abc';
export {Bar, abc as ooxx};
```

        - 默认导出

```
// 默认导出
export default class Student {
  name: string;
}
```

    - 导入
        - 具名导入

```
import {HOST as host, add, Foo, Bar, ooxx} from './myModule';
```

        - 默认导入

```
import Student from './myModule';
```

6. 装饰器 decorator

   @NgModule({})

学生端：用户登录、注册

0.创建组件：ng g c compName

1.登录组件 login：ng g c login

2.注册组件register：ng g c register

3.组件注册，AppModule

```
declarations: [// 声明组件，才能正常使用组件
  AppComponent,
  LoginComponent,
  RegisterComponent
]
```

3.路由: 用于在不同视图之间导航

- 声明路由：

  const routes: Routes = [{path,component},....]

- 导入路由模块：

  imports:[

   RouterModule.forRoot(routes)

  ]

- 放入路由插座
- 路由链接

  ```
  <a routerLink="login" routerLinkActive="active">登录</a>
  ```

作业

1.复习

2.预习angular路由、表单使用及验证、HTTP请求使用

3.尝试将登录和注册组件提取至独立特性模块