Angular7

(官网地址：<https://www.angular.cn/tutorial/toh-pt0>)

（推荐地址：https://www.jianshu.com/p/f0f81a63cbcb）

# Angular 介绍

Angualr 是一款来自谷歌的开源的 web 前端框架，诞生于 2009 年，由 Misko Hevery 等 人创建，后为 Google 所收购。是一款优秀的前端 JS 框架，已经被用于 Google 的多款产品当中。

Angualr 基于 TypeScript 和 react、vue 相比, Angular 更适合中大型企业级项目。

目前 2018 年 11 月 25 日 angular 最新版本 angular7.x。根据官方介绍，Angular 每过几个月 就会更新一个版本。此教程同样适用于后期更新的 Angular8.x、Angular9.x

**学习 Angular 必备基础**

必备基础:html 、css 、js、es6、Typescript

# 2.Angular 环境搭建及创建项目

## 1.创建项目

npm install -g @angular/cli

ng-v

# 创建项目

ng new my-app

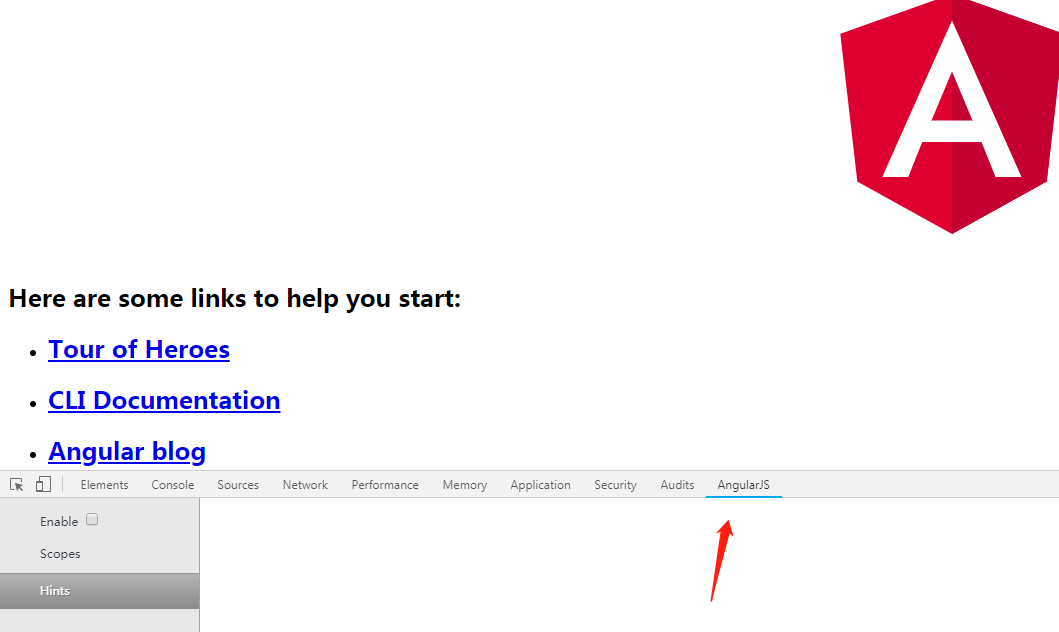
cd my-app

# 运行项目

ng serve --open

## 2.安装谷歌扩展





## 3项目的目录结构

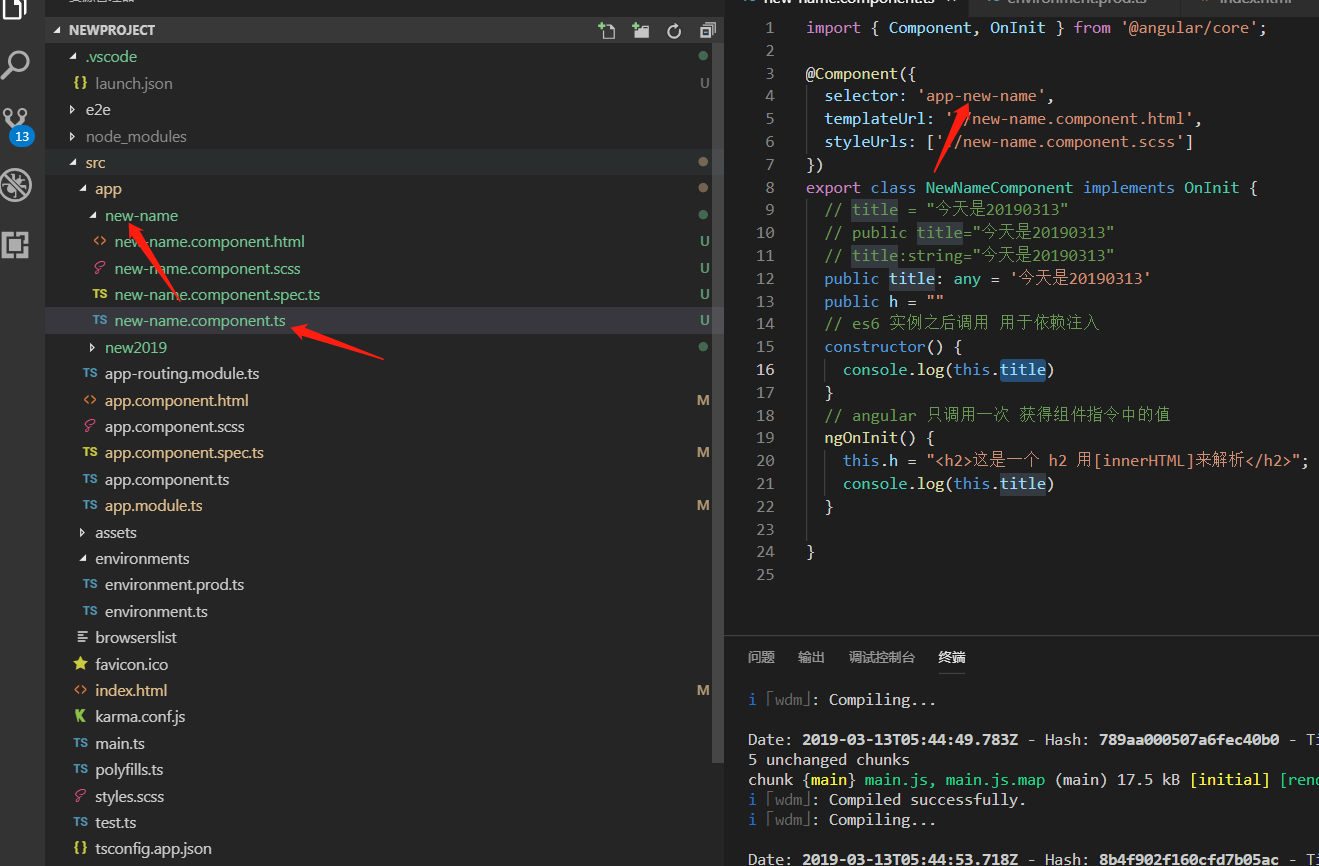


## 4.创建新组件

### 1.创建

ng g component new-name

### 2.引用



### 3.变量设置

Constructor es6属性 实例化调用 这里用来注入

Ngoninit angular 只调用一次

变量命名规范   
 public titlt=”这是一个标题”

titlt =”这是一个标题”

title:string=’ 这是一个标题’

public title:any=’这是一个标题’

### 4绑定方式

title =”aaaa”

<div>{{title}}</div>

this.title=”<h1>wdwdw</h1>”

<div [innerHtml]=’ title’></div>

//绑定属性

name=’id123’

<div [id]=’ name ‘></div>

* public 共有（默认） 可以在类里外使用
* protected 保护类型 只能在当前类和子类中使用
* private 私有类型 只能在当期类使用

### 5循环\*ngFor,条件\*ngIf

**\*ngFor,**

arrObj=[{name:'张三',age:12},{name:'李四',age:151}]

<ul>

<li \*ngFor="let item of arrObj">{{item.name}}+{{item.age}}</li>

</ul>

**\*ngIf**

<ul \*ngIf="arr.length>0">

<li \*ngFor="let item of arr">{{item}}</li>

</ul>

### 6.\*ngSwitch

<ul [ngSwitch]=’5’>

<li \*ngSwitchCase=1>1</li>

<li \*ngSwitchCase=2>2</li>

<li \*ngSwitchDefault>无效</li>

</ul>

### 7.(click)

（click）=”clickme”

<button class="button" (click)="getData()"> 点击按钮触发事件

</button>

### 8.表单元素（keyup）

<input

type="text"

(keyup)="keyUpFn($event)"/>

<input type="text" (keyup)="keyUpFn($event)"/>

keyUpFn(e){

console.log(e)

}

### 9.双向绑定 [(ngModel)]

App.module.ts文件引入 FormModule模块

<input type="text" [(ngModel)]='iptval'>

<input type="text" [(ngModel)]='iptval'>

### 10.管道处理时间格式

datenow=new Date()

<div>{{datenow|date:'yyyy-MM-dd hh:mm:ss'}}</div>

### 11.大小写转换

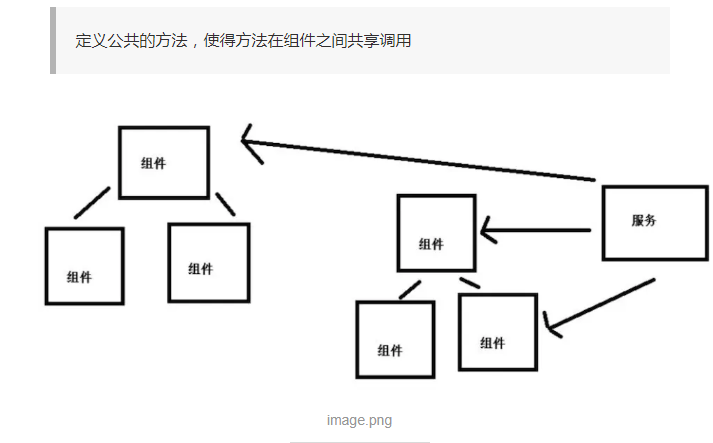
<!-- 大小写转换 -->

<p>{{ddnum|uppercase}}</p>

<p>{{ddnum|lowercase}}</p>

# 3.服务创建引用

## 1.服务原理



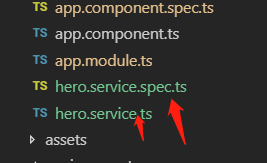
## 2.创建

ng g service hero

指定路径

ng g service services/user

创建之后会多出两个文件service



Hero文件内容

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

@Injectable({

providedIn: 'root'

})

export class HeroService {

constructor() {

}

}

## 组件内部引入服务

hero.service文件

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

@Injectable({

providedIn: 'root'

})

export class HeroService {

newServer="http://wwww.baidu.com"

add(){

alert('我是服务里面的事件')

}

constructor() {

}

}

import { HeroService} from '../hero.service';

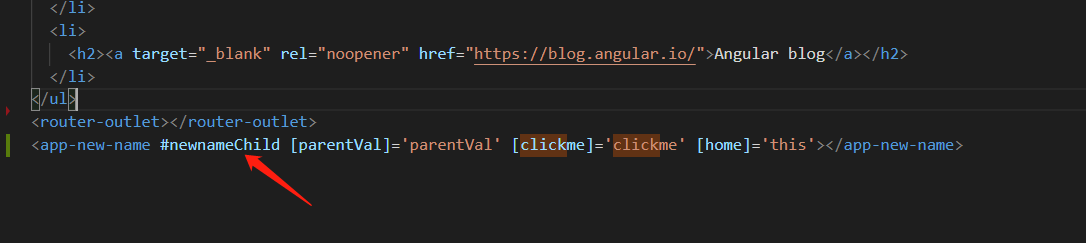
constructor(private storage: HeroService) {

console.log(this.storage.newServer)

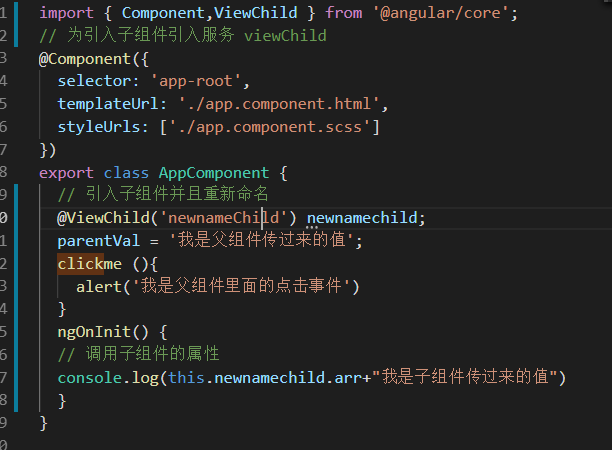
}

# 4.父子组件传参

## 1.子级传父级

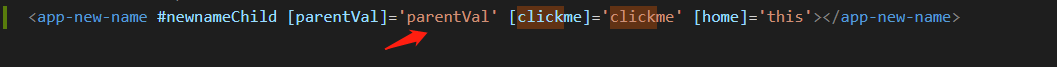
界面引入子级需要做一个唯一的#开头的名称  


父组件引入ViewChild,引用名称并重新命名，在ngOnInit直接调用



## 2.父级传子级

引入时需要将你已经在父组件里面定义的变量，事件引入 如下图



子组件需要引入input



同样在ngOnInit里调用即可，如果父组件引入子组件的时候标签值为this 就是把父元素的整个组件全部引入，父组件里面定义的值变量都可使用

## 3.非父子组件通讯

* 1. 公共的服务
* 2. Localstorage (推荐)
* 3. Cookie

# 5.对dom操作

直接在组建中引入新的方法ngAfterViewInit

// 对原生直接操作的方法

ngAfterViewInit() {

var docm = document.getElementById("docm")

docm.style.color = "red"

}

# 6.生命周期

 生命周期函数通俗的讲就是组件创建、组件更新、组件销毁的时候会触发的一系列的方法。

 当 Angular 使用构造函数新建一个组件或指令后，就会按下面的顺序在特定时刻调用这些 生命周期钩子方法。

 每个接口都有唯一的一个钩子方法，它们的名字是由接口名再加上ng前缀构成的，比如OnInit接口的钩子方法叫做ngOnInit.

## 1.生命周期钩子分类（基于指令与组件的区别）

指令与组件共有的钩子

NgOnChanges

ngOnInit

ngDoCheck

ngOnDestory

组件特有的钩子

ngAfterContentInit

ngAfterContentChecked

ngAfterViewInit

ngAfterViewChecked

## 2.生命周期钩子的作用及调用顺序

1.ngOnchages 当数据绑定输入属性的值发生变化时调用

2.ngOnInit 在第一次ngOnchanges后调用

3.ngDoCheck 自定义的方法，用来检测和处理值得改变

4.ngAfterContentInit 组件内容初始化后调用

5.ngAfterContentChecked 组件每次检查内容时调用

6.ngAfterViewInit 组件响应的视图初始化后调用

7.ngAfterViewChecked 组件每次检查视图调用

8.ngOnDestory 指令销毁前调用

## **3**. 首次加载生命周期顺序



点击事件或者是model改变 checked结尾的事件都会调用

# 7.RXjs异步数据流编程

## 1.RXjs介绍

RxJS 是 ReactiveX 编程理念的 JavaScript 版本。ReactiveX 来自微软，它是一种针对异步数据 流的编程。简单来说，它将一切数据，包括 HTTP 请求，DOM 事件或者普通数据等包装成流的形式，然后用强大丰富的操作符对流进行处理，使你能以同步编程的方式处理异步数据，并组合不同的操作符来轻松优雅的实现你所需要的功能。

 RxJS 是一种针对异步数据流编程工具，或者叫响应式扩展编程;可不管如何解释 RxJS 其目 标就是异步编程，Angular 引入 RxJS 为了就是让异步可控、更简单。

 RxJS 里面提供了很多模块。这里我们主要给大家讲 RxJS 里面最常用的Observable 和 fromEven

**目前常见的异步编程的几种方法:**

* 回调函数
* 事件监听/发布订阅
* Promise
* Rxjs

2.RXjs与Promise的对比

getPromiseData() {

return new Promise(resolve = >{

setTimeout(() = >{

resolve('---promise timeout---');

},

2000);

});

// RxJS 处理异步:

getRxjsData() {

return new Observable(observer = >{

setTimeout(() = >{

observer.next('observable timeout');

},

2000);

});

}

从上面列子可以看到 RxJS 和 Promise的基本用法非常类似，除了一些关键词不同。Promise 里面用的是 then() 和 resolve()，而 RxJS里面用的是 next() 和 subscribe()

Rxjs相比Promise要强大很多。 比如 Rxjs 中可以中途撤回、Rxjs 可以发射多个值、Rxjs 提供了多种工具函数等等

## 2.MAP filter

import {Observable} from 'rxjs';

import {map,filter} from 'rxjs/operators'

this.rxjs.getRxjs().pipe(filter(val => val % 2 == 0),map(value=> value\*value)).subscribe(data => { console.log(data) })

# 8.http请求

Angular5.x 以后 get、post 和和服务器交互使用的是 HttpClientModule 模块

**在 app.module.ts 中引入 HttpClientModule 并注入，再用到的模块同样需要引入**

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

imports: [

BrowserModule,

HttpClientModule

]

Get 请求

var api = "http://a.itying.com/api/productlist";

this.http.get(api).subscribe(response => {

console.log(response); });

post请求 **在用到的地方引入 HttpClient、HttpHeaders 并在构造函数声明 HttpClient**

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

var api = "http://127.0.0.1:3000/doLogin";

// 手动设置请求类型

const httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json'

})

this.http.post(api, {

username: '张三',

age: '20'

},

httpOptions).subscribe(response = >{

console.log(response);

});

Jsonp请求

**在 app.module.ts 中引入 HttpClientModule、HttpClientJsonpModule 并注入**

**组件中在用到的地方引入 HttpClient 并在构造函数声明**

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

constructor(public http:HttpClient) { }

var api = "http://a.itying.com/api/productlist";

this.http.jsonp(api,'callback').subscribe(response => {

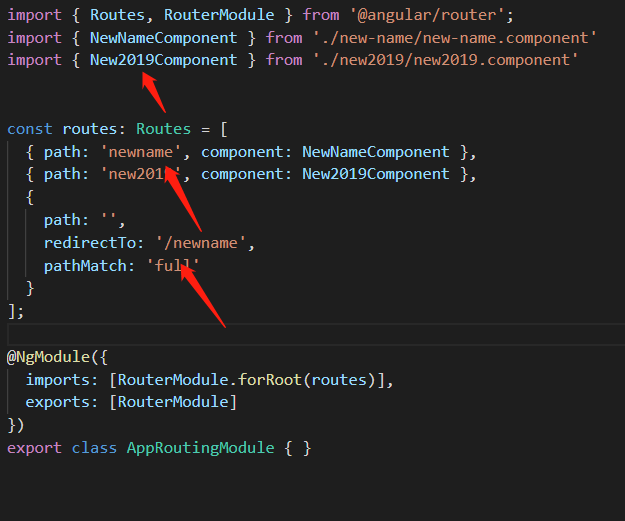
console.log(response); });

同时也可以引入axios

# 9 路由

## 1.路由配置

找到app-routing.modules.ts配置路由



引入模块，设置好对应的组件，然后在配置一个默认的路径，两种引入方式 href ,routerLink



## 2.路由传参

1.第一种方式 路由里面配置 a标签点击的时候进行传参





组件里面进行接收

首先引入ActivatedRoute并注入

import { ActivatedRoute} from '@angular/router'

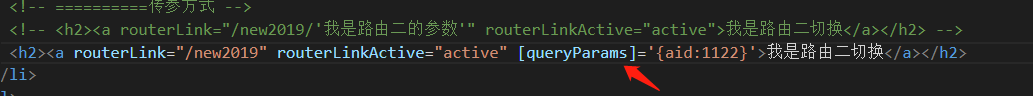
constructor(private route: ActivatedRoute) { }

this.route.params.subscribe(data=>{

console.log(data)

})

2.通过queryParams传参



this.route.queryParams.subscribe(data=>{

console.log(data)

})

动态路由跳转及传参

首先引入Router

this.router.navigate(['new2019',{name:'张三'}])

第一个参数是路由路径 第二个是参数

接收的时候同样引入 ActivatedRoute注入

this.route.params.subscribe(data => {

console.log(data)

})

Data就是传过来的参数

父子传参

1. **创建组件引入组件**

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

2. **配置路由**

{ path: 'newname', component: NewNameComponent },

// { path: 'new2019/:id', component: New2019Component },

{

path: 'new2019',

component: New2019Component,

children:[{

path:'child',

component: ChildComponent

}]

},

{

path: '',

redirectTo: '/newname',

pathMatch: 'full'

}

3.给父组件设置

<router-outlet></router-outlet>