复习：

见思维导图

1.高阶话题：服务对象的作用范围

声明服务提供者的方式：

**方式1：在根模块中提供服务对象——在整个应用中服务是单例**

@Injectable( {providedIn: 'root'} )

export class TimerService{ }

**方式2：在组件中提供服务对象——在每个组件实例中服务都有一个实例的**

@Injectable( )

export class TimerService{ }

---------------------------------------

@Component({

....

providers: [ TimerService ]

})

export class LoginComponent{ }

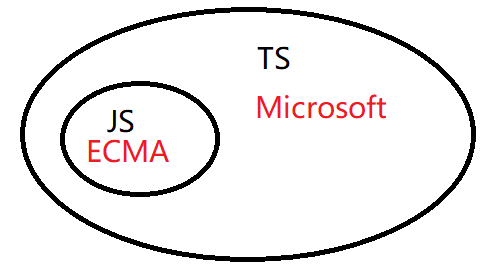
注意：项目中只要服务对象中有属性，只能用方式2；否则推荐使用方式1

2.面试严重加分项：TypeScript

Angular从V2.x开始使用TS编写；Vue.js从V3开始使用TS编写；

中文网：https://www.tslang.cn/

TS是由微软编写的一门基于JS又高于JS的强类型编程语言



浏览器不支持TS，TS的源码需要编译为JS才能在浏览器中使用；使用步骤：

①下载TS编译器

npm i -g typescript

②编写.ts文件，使用编译器转化为.js

tsc hello.ts

//得到hello.js

|  |
| --- |
| TS学习手册：  <https://www.tslang.cn/docs/handbook/basic-types.html> |

TS的扩展特性：

**①TS是强类型语言 ——**  JS是弱类型语言

强类型语言的特点：变量、形参、函数都可以声明类型

uname: string

f1( n1:number ):boolean {

return true

}

常用的数据类型关键字：string / number / boolean / any / number[ ] / Object / Date /void

**②TS对属性和方法定义了三种访问修饰符**：

private：私有的，私有成员只能在本类内部使用

protected：被保护的，被保护的成员只能在本类内部及子类内部使用

public：公共的，公共成员可以在本类内部以及外部使用

提示：一般情况下，class内属性不应该让外界随便访问，通常设置为private；方法一般允许调用的，通常设置为public

**③访问修饰符的特殊用法**

下述两种写法的作用相同

|  |  |
| --- | --- |
| class Emp{  private age:number  constructor( age ){  this.age = age  }  } | class Emp{  constructor( private age:number ){  }  }  注意：此种简写只能出现在构造方法中 |

**④面向对象编程核心概念——class(类)和interface(接口)**

interface：接口，是一种特殊的类，规范“要求某个class必须具备XXX方法”，如管道类必须提供transform方法

interface Runnable{

start( ) //接口中的方法没有主体！

stop( )

}

--------------------------------------------------

class Car implements Runnable {

start(){ } //接口的实现者必须提供指定的方法，要有主体

stop(){ }

.....

}

**⑤装饰器**

见手册<https://www.tslang.cn/docs/handbook/decorators.html>

面试题：前端体系中有哪些发起异步请求的工具？各自的利弊？

|  |  |  |
| --- | --- | --- |
| **工具名** | **本质** | **优劣势** |
| 原生XHR | let xhr = new XMLHttpRequest() | 浏览器支持的原生技术；  基于回调方式处理响应； |
| jQuery.ajax() | 也是XHR，只是进一步的封装而已 | 比原生要简单；  基于回调方式处理响应； |
| Axios | 也是XHR，只是进一步的封装而已 | 比原生要简单；  基于Promise处理响应；  可以排队、并发、撤销； |
| NG HttpClient | 也是XHR，只是进一步的封装而已 | 比原生要简单；  基于“观察者模式”处理响应；  可以排队、并发、撤销； |
| Fetch | 不再是XHR，是W3C提出的新技术，有望取代XHR | 比XHR从根本上就更加先进；天然基于Promise；  目前浏览器还有兼容性问题； |

|  |
| --- |
| "马走日象走田" ——> "双炮枪"组合连招  JS类/接口/原型 ——> 设计模式(23+1种) |
| 23种设计模式之一：观察者模式的原理：  **Observable：**可被关注的对象，在未来某个不确定的时间点可能抛出特定的事件  **Subscribe：**订阅，建立与可被关注的对象之间的契约，一旦其发布新消息订阅者立即会被通知到 |

3.Angular官方提供的服务对象 —— HttpClient

HttpClient服务：是Angular提供的用于发起异步XHR请求的对象

使用步骤：

①在主模块中导入HttpClientModule模块

//app.module.ts

imports: [ BrowserModule, HttpClientModule ]

②在组件中声明依赖于HttpClient服务对象，就会被自动注入进来

constructor( private http: HttpClient ){ }

③调用HttpClient实例实现异步请求

this.http.get( url ).subscribe( (res)=>{ } )

4.面试题：组件的生命周期钩子函数

手册地址：<https://angular.cn/guide/lifecycle-hooks>

Angular中的组件的声明周期钩子函数调用顺序：

(0) constructor() 组件对象被创建了

(1) ngOnChanges() 组件绑定的属性值发生改变

(2) ngOnInit() 组件初始化完毕 —— 等同于Vue.js的mounted

(3) ngDoCheck() 组件检查到了系统对自己的影响

(4) ngAfterContentInit() 组件的内容初始化完成

(5) ngAfterContentChecked() 组件的内容发生变化需要检查

<myc01><p>{{uname}}</p></myc01>

(6) ngAfterViewInit() 组件的视图初始化完成

(7) ngAfterViewChecked() 组件的视图发生变化需要检查

template: '<p>{{uname}}</p>'

(8) ngOnDestroy() 组件即将被从DOM树上卸载，适合执行一些资源释放性语句，例如：定时器销毁...

\*ngIf="false"

课后任务：

(1)比较Vue.js和NG中生命周期钩子函数

(2)复习：Vue.js中的父子组件间数据传递 —— 难点

(3)基于“商品列表”练习，表格最后添加一列“查看详情”，点击后在表格下方呈现出该商品的详情

提示：

商品列表API：<http://www.codeboy.com/data/product/list.php?pno=2>

商品详情API：<http://www.codeboy.com/data/product/details.php?lid=3>

注意：需要显示出该商品的详情图片！

<img [src]=" 'http://www.codeboy.com/' + p.pic ">