**note**

一、angular

1、路由 app.module中 const routes:Routes[ 数组内添加

path: '/hhh'.component :Homecomponent ] 实现路由到组件

2、app.module中的imports:[ 数组中添加根路由模块

RouterModule.forroot(routes)]

3、app.module中 添加模块到imports 添加组件(指令）到declaration，添加服务到providers

4、子路由

{path: '', component: HomeComponent

childen:[ {path: '' , cmponent ZxxComponent} ] }

<a [routerLink] = "[ ' ./Zxx ' ]" >

5、typescript里操作路由 使用route.navigate([' /product '])

6、 使用返回值为promise的函数

this.heroService.getHeroes().then(heroes => this.heroes = heroes);

7、debug的时候 ，.then(（res） => {}, 可在后面加一个（err） => {console.log(err);})

8、使用回调函数的时候，this指针会指向函数内，解决办法两种，一是传递fn\_this 参数，使得 fn\_this指向外层类的this，另一种是使用 ( )=> { } 代替 function () {}

9、 组件内传递数据的四种方式 一、变量名前加@input装饰器 html中可以使用 [ 变量名 ] 导入 二、html（click）等事件 绑定到div上 触发作为函数参数传递 三、新建一个服务，方法写在服务里，依赖注入这个服务即可调用服务中方法 四、直接依赖注入要使用的组件

10、所有表单元素可以用 [(ngModel)]双向绑定 value 的值，但是只能传递简单的数据类型，不能传对象。value如不指定默认值是标签的text

11、angular中使用jquery编译时 需要在开头declare $:any ; 插件也一样 declare swiper:any

12、ngOnInit(){ }中不能直接异步调用，要把异步函数和语句封装在一个异步函数里

async\_func(){await this.get\_types(); }中，然后在ngOnInit(){ this.async\_func(); }

13、img标签路径错误时 显示另一张图片 使用onerror方法

<img src={{show.images[0].path}} onerror="onerror = null; src ='./assets/hsb.png'">

**14、声明数组时一定要先赋一个空值 product: Array<any> = []；**

声明二维数组时 product\_array :Array = [ [], [], [], [], [], [], [] ]; 否则无法使用push方法

15、获取ngfor时的key（下标），\*ngfor let product of products; index as i

**16、监听一个事件 写法 比如触摸屏幕计数器清零**

window.addEventListener('touchstart', (event) => {

console.log(event);

this.time\_counter = 0;

});

17、[class.green] = " is\_green " 可以让dom在不同情况下添加或删除css 同理也可以用布尔值判断绑定的样式和dom属性

18、遇到可能需要运行时间的函数 如从售货机里读取数据 就要使用异步

async func() {

await this.kiosk\_data.get\_products.then(); }

在wait语句执行结束前不往下执行

19、在environments中配置生产环境和开发环境

开发环境

**export const** environment = {

production: **false**,

hostname: '192.168.70.254'

};

生产环境 新建 eviroment.pro.ts

**export const** environment = {

production: **true**,

hostname: 'localhost'

};

**ng build 或 npm start 通过添加 --env=prod 来使用生产环境**

20、使用parseInt 配合slice 把字符串转成数字

square = parseInt(**this**.layout\_list[i][j].name.slice(4, 6)) - parseInt(**this**.layout\_list[i][j].name.slice(1, 3)) + 1;

**使用something.toFixed(2)保留两位小数**

21、与服务器通讯 配置从哪个端口接收 新建proxy.conf.json文件

｛

“/api":{

"target":"http:.//localhost:8000"

}

｝

在 package.json中 ”start“: ”ng-server --proxy--config proxy.conf.json“ 引入

22、Observable类型是一个流 （stream），基于Rxjs , 可以不刷新的状态下动态获取服务器更新过的数据 需要使用subscribe来订阅，订阅时候需要写出异步或使用settimeout来确保拿到数据。

23、有较少（但比较重要）的情况 angular不能检测到变量的变化，导致延迟很久才显示，

可以 import {ChangeDetectorRef} 依赖注入 constructor(private ref ChangeDetectorRef){}

使用 this.ref.detectChanges(); 强制变化检测

24、模板式表单 调用对象#myForm = "ngForm" 提交事件(ngSubmit) = “a()”

在ngForm的 input中使用 ngModel不用[],但需要加name属性 ngForm.value 就是input的值，name为该对象的属性名

require 必填项

pattern = “[a-zA-Z0-9]+”

响应式表单<form [formGroup] = "abc" > html属性为对象或变量时 使用绑定和[] 属性为字符串时不要加[]

25、写有可能失败的函数 （如网络不通很多函数就会失败） 要加try{ }catch(e){

alart(); }

26、正则表达式

两个反斜线/ / 之间的内容 /[bdfs]uck/ 可以匹配buck duck fuck suck

转义符号为\ 如果要输出/ + ? . \*等关键字，前面要加\ （所以会经常出现 \ / \ /）

|  |  |
| --- | --- |
| [a-z] | 字符范围。匹配指定范围内的任意字符。例如，“[a-z]”可以匹配“a”到“z”范围内的任意小写字母字符。 注意:只有连字符在字符组内部时,并且出现在两个字符之间时,才能表示字符的范围; 如果出字符组的开头,则只能表示连字符本身. |
| [^a-z] | 负值字符范围。匹配任何不在指定范围内的任意字符。例如，“[^a-z]”可以匹配任何不在“a”到“z”范围内的任意字符。 |
| + | 匹配前面的子表达式一次或多次(大于等于1次）。例如，“zo+”能匹配“zo”以及“zoo”，但不能匹配“z”。+等价于{1,}。 |

27、获取一个对象或json中的键值

for(var i in Object){

alert(i);//“键名”

alert(Object[i]);//“键值'

}

也可以使用get方法获取value Object.get('属性名/方法名/子对象名') 获取这个属性/方法/子对象的值

28、如果类里面有类或json格式的，使用类的嵌套解决，如定义下面的结构

this.sessionContent = {

amount: {

paid: 1,

total: 1,

discount: 0,

real\_paid: 1

},

order\_no: '',

}

先 class Amout{

paid: number;

total:number;

discount: number;

real\_paid:numvber;

}

然后再 class Session{

amout: Amout;

order\_no:string;

}

29、如果通过格式比较复杂，建一个类再new一个对象很麻烦，则可以直接在组件的构造函数里写

**constructor**() {

**this**.session\_content = {

payment: {

method: '',

no: ''

},

amount: {

paid: 1,

total: 1,

discount: 0,

real\_paid: 1

},

order\_no: '',

timestamp: 0,

detail: [],

quantity: {

total: 1,

fetched: 0

},

state: 'FINISHED',

kiosk: {

name: '',

id: ''

}

};

}

30、引入外部库尽量在index.html里引入，使用npm引入可能造成库冲突，引入时一定要先引入依赖库，再引入方法库，有顺序要求 比如

<script src="./assets/ice/Ice.js"></script>

<script src="./assets/ice/exception.js"></script>

<script src="./assets/ice/data.js"></script>

<script src="./assets/ice/mpay.js"></script>

<script src="./assets/ice/robot.js"></script>

41、javascript中 每一个对象都自动生成prototype函数集，可以调用里面的方法 javascript中 this 的指向是可以改变的， 使用call()或apply()改变

二、node.js

1、nodejs 使用nodemon 服务器可以实时更新

import \* as express from "express"

2、构建 ng build 部署 编译好的文件放在服务器文件夹/client 中

nodejs 添加 import \* as path from "path"

服务器get方法改写成 app.ues('/',express.static(path.join(\_dirname,'..','client'))

app.module中 provides[]加入 {provide：LocationStrategy,useClass:HashLocationStrategy}

3、var buffer = new Buffer(8) 为分配8个字节的内存 buffer.write()写入内容 转化为16进制

buffer.toString 转换回字符 buffer.copy() 拿副本出来

4、do\_it().on( 'data',function(chunk)｛ ｝ ) 监听函数的一般写法 data为数据传输事件发生时，读取的数据为chunk，

同理监听end事件 on('end' ,function(){} ) ，

关闭事件 .on（close, function(){}）

错误事件.on('error' ,function(e){ })

5、流的类型：readable writable Duplex Transform，fs.createReadStream()代替 fs.readFile

后者是度二进制文件放入buffer

6、创建可读流对象实例

var Read\_Able = require('stream').Readable()

var read\_stream = new Read\_Able（）

流获取的数据块为chuck

7、pipe是集合了很多功能的管道，使用 request库（var requst = require('request') 注意request请求和require需要没有关系 ） 直接在creatServer的function(req,res){ request( 'URL' ).pipe(res) } 实现边下载 边pipe pipe只处理流

8、绝大部分函数使用异步 声明函数时在前面 加async 调用函数前加await 表示等这个函数/语句 执行完并获取到数据之后再进行下一步

三、PHP + MySQL

1、创造关联数组 （类似json的键值对） $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

遍历数组 foreach($arr as $x => $x\_value)

x为key x\_value为值

print\_r（）可以显示数组全部值

2、数组排序

* sort() - 对数组进行升序排列
* rsort() - 对数组进行降序排列
* asort() - 根据关联数组的值，对数组进行升序排列
* ksort() - 根据关联数组的键，对数组进行升序排列
* arsort() - 根据关联数组的值，对数组进行降序排列
* krsort() - 根据关联数组的键，对数组进行降序排列

3、namespace Chromie 定义一个命名空间

在其他文件里使用using namespace Chromie之后，使用Chromie中的函数可以直接调用，不需要Chromie::func() -- php Chromie.func() --java\js php中 使用双冒号::来调用静态方法、属性，类的方法是静态方法，类的实现的方法是动态方法

4、var\_dump() 判断数据类型

字符串使用双引号 ，会解析里面的变量，如果使用单引号则原样输出

5、function fuck\_it(girl1 = ' lin',girl='zhang ',gilr3='lv' ){ }可以这样设置参数默认值，当传的参数不够、甚至不传参的时候可以用默认值而正确调用函数

6、function more\_arg(){ 这样定义可传无限参数的函数

$args = func\_get\_args();

}

以函数为参数（回调函数时） 写函数名中字符串的部分 如 $fuck\_it() 做参数时 写fuck\_it

call\_user\_func\_array( 'fuck',array[arg1,arg2]){ 其中arg1 和arg2 是$fuck(){ }函数的参数

}

7、timestamp 时间戳 返回1970年1月1号至今的毫秒数， 1970应该可以改

8、数据类型有ENUM 枚举 ：比如性别只有男、女、保姆可选

SET（）集合 ，可以从中取任意数量的选项 比如ABCD中选 BD

9、INSERT tabel123(col1,col6) VALUES(a,x); 可插入部分值

10、AUTO\_INCREAMENT 自增1，要和PRIMARY KEY结合使用

UNIQUE KEY也可以保证唯一性 且可以有多个

11、建表时 DEFAULT 设置某一列默认值

12、外键 FOREIGN KEY (col\_name) REFERENCES fubiao\_name (fubiao\_col\_name)

13、列操作 ALTER TABLE table\_name ADD/DROP/MODIFY

三、布局

1、display:flex 多行多列布局

2、父div 使用position: relative 子div使用position:absolute 可方便调整位置

3、使用overflow属性解决 内容超出父容器的情况

4、弹性布局 父盒子 ｛display: -webkit-flex;display: flex;｝ ，子盒子｛flex: 1;flex: 2;flex: 3;｝

5、当Swiper 滑动时触发事件 用回调函数 slideChange

var mySwiper = new Swiper('.swiper-container',{ onTransitionEnd: () => {alert('过渡结束');}, })

四、其他

1、spu是一个产品单元，sku是spu的自己（不同颜色、规格等）

2、使用npm rm --save xxxxxx 可以彻底删除一个使用npm install 安装的包

3、调用整个方法集

get\_kiosk\_data(): Promise<any> {

return ThybotKioskService.DataService.DataInterfacePrx.checkedCast(this.proxy\_data).then((result: any) => result);

}

调用方法集里面的方法

get\_product(): Promise<any> {

return this.get\_kiosk\_data().then((result: any) => result.getSkus().then((sku: any) => { /\*console.log(sku);\*/ return sku; } ));

}

一些常用方法

get\_Skus() 获取商品列表

onload() 返回新库存到机器 下架时quantity参数取负值

getSkuQuantityOnRack()获取库存

getSlotLayout()获取的布局是一个字典 \_table中的2维数组代表着机器的列和行 使用

result.keys()获取列 ，result.get(int key)方法获取一行中的若干对象

getCategoryWithSkuOnRack() 获取分类