MVVM框架的组成：数据绑定/指令系统/组件式编程/路由和导航/状态保持/第三方组件库

2. Angular框架

Angular v1.x官网：<https://angularjs.org/>

Angular v2.x～v8.x官网：<https://angular.io/>

Angular中文镜像网站：<https://angular.cn/>

是google2009年开发的MVVM框架

Vue.js适合中小型项目, Angular适合中大型项目

Web worker/web socket是html5中最有价值的，表单是很low的

1. 开发方式

Vue.js开发方式：script引入vue.js，脚手架方式

Angular2.x开发方式：脚手架方式

1. 搭建angular开发环境

前提：要有node.js

查看当前默认的下载仓库地址 npm config get registry

修改NPM默认的下载仓库地址为国内镜像网站：

npm config set registry 新仓库地址

1. 下载安装脚手架

npm install -g @angular/cli

1. 运行脚手架创建空白项目

ng new myngapp01

项目可以选想装在哪里

1. 运行开发服务器

Cd myngapp01

Npm start

1. 客户端访问测试

https://127.0.0.1:port

1. Angular的组件不是直接放在最外层的而是各个模块中
2. Angular项目启动过程分析：
3. **Angular.json**: NG项目的配置

Index: ./src/index.html <app-root></app-root>

Main: ./src/main.ts

1. **Main.js** > AppComponent
2. **App.component.ts** > selector: ‘app-root’

* templateUrl: ‘app.component.html’

1. **app.component.html** > html片段
2. 一个组件由三个组成：templateurl，script，style
3. Angular核心概念之一：模块

Module：不同于Node.js或者ES6中的模块的模块，NG中的模块就是一个抽象的容器，用于对组件分组。

整个应用初始时有且只有一个主模块：AppModule

1. Angular核心概念之二：组件

组件：是一段可以反复使用的页面片段，如页头/轮播/手风琴等

组件（component）= 模版（template）+脚本（script）+样式（style）

提示：在NG中任何一个组件都需要声明/注册在一个模块中

1. 自定义组件步骤：
2. 创建组件class

@Component({

template: '<h2>我的组件C01</h2><hr>',

selector: '[myTitle]' /\*属性来用要加[]，当元素来用直接用没有[]，如果.xxx，当作class来用 \*/

})

export class MyC01Component {

}

1. 在某个模块中注册组件class

如果没有找到注册的文件app.module.ts的话，创建项目的时候要加一些指令

ng new my-app --no-standalone --routing --ssr=false

1. 使用组件

//app.component.html

<myc01></myc01>

练习：创建一个自定义组件myc02，创建class，注册，使用

练习：创建一个文件夹，myc03，其中在自定义组件myc03.component.ts

1. **Angular提供的创建组件的简化工具**

ng g[enerate] component 组件名

npx ng g[enerate] component 组件名

**\*\* .spec.ts是做单元测试的，不是一定要用**

1. Node.js官方安装的工具：

npm：第三方模块的维护工具

npx：第三方可执行文件的执行工具，Node Package Executor npx 可用于执行当前项目中node\_modules./bin目录下的可执行

1. Angular核心概念之三：数据绑定
2. Html绑定：{{NG表达式}}

测试：NG表达式中可以执行哪些代码？

算术运算？

比较运算：

逻辑运算：

三目运算：

调用函数：

创建对象：不可以new

<!-- <div>当前时间：{{new Date()}}</div> -->

Json序列化：不可以，因为NG表达式中JSON是undefined

<!-- <div>JSON字符串：{{JSON.stringify({})}}</div> -->

1. 属性绑定：Vue.js：v-bind

形式1：直接在属性上用{{}}绑定

<p title="{{uname}}">

形式2:（推荐）直接使用中括号[]

<p [title]="uname">

1. 事件绑定：Vue.js：v-on或者简写为@
2. 指令绑定
3. 双向数据绑定
4. 课后任务：

（1） 比较vue.js和angular中的表达式有何不同

（2） 自学手册中的“事件绑定”和“指令绑定”，实现购买计数器n

Day2

1. Angular中的数据绑定
2. HTML绑定：{{}}
3. 属性绑定：[]

注意：如果赋值是常量要加单引号

1. 事件绑定

(click)=”add()”

1. 指令绑定
   1. **循环绑定：\*ngFor**

<Any \*ngFor=”let 临时变量 of 数据”>

<Any \*ngFor=”let 临时变量 of 数据; let i=index;”>

<Any \*ngFor=”let 临时变量 of 数据; index as i””>

* 1. **选择绑定： \*ngif**

<Any \*ngif=”布尔表达式”>

说明：如果布尔表达式为false，则当前元素从DOM树上删除

<Any \*ngif=”布尔表达式； else ELSE块的编号”></Any>

<ng-template #ELSE块的编号>

<Any></Any>

</ng-template>

* 1. **样式绑定**

ngStyle/ngClass

说明：ngStyle绑定的值必须是一个对象！对象属性就是CSS样式名

ngClass

说明：ngClass绑定的值必须是一个对象！对象属性就是CSSclass名，属性值是true/false的话该class就出现，否则该class不出现

* 1. 了解：特殊的选择绑定：

<Any [ngSwitch]=’表达式’>

<Any \*ngSwitchCase=”值1”></Any>

<Any \*ngSwitchCase=”值2”></Any>

<Any \*ngSwitchDefault=”值3”></Any>

1. 双向数据绑定：[(ngModel)]-重点

方向一：Model => View, 模型变则视图变，用[]绑定

方向二：View => Model，视图（表单元素）变则模型变，用()绑定

<input/select/textarea [(ngModel)]=”uname”>

注意：如果想要监听模型数据改变，可以绑定ngModel指令不在CommonModule模块中，而在FormsModule中，使用之前必须在主模块中导入该模块；ngModel指令不在CommonModule模块中，而在FormsModule中，使用之前必须在主模块中导入该模块：

imports: [

BrowserModule,//if you use Angular to create a web application, you should have this module, including ngFor/ngIf, not including ngModel

AppRoutingModule,

FormsModule

],

1. 练习：注册新用户时，密码字段随着输入的进行，后面span中显示提示文字，可能为：

密码长度应该在6～12位之间

密码太短了，

密码长度合适

密码太长了

Angular中的指令分为3类，什么时候用\*什么时候用[]：

1. 组件指令：NG中Component继承自directive

Eg. ng g directive 指令名称

（2）结构型指令：会影响DOM树结构，必须使用\*开头，如\*ngFor, \*ngIf

（3）属性型指令：不会影响DOM树结构，只是影响元素外观或者行为，必须用[]括起来，如[ngClass],[ngStyle]

1. 练习：创建变量保存银行储户基本 userLevel = “normal”/”vip”/”blackgold”，根据用户级别不同，看到不同的内容，如果用户基本不在上面三个之一，呈现“欢迎您：游客”

Angular-5.1-复习

1. 练习：创建员工列表数据： empList=[{eid:101,ename:’亮亮’,sex1,zzmm:10, salary:5000}],把上述数据绑定在table中，最后添加一列<button>删除</button>，点击后删除该员工

Angular-5.5过滤器-01

Filter:过滤器，用于在View中呈现数据时显示为另一种格式；过滤器的本质是一个函数，接收原始数据转换为新的格式进行输出function（oldVal）{return newVal}

使用过滤器：{{e.salary | 过滤器名}}

Angular2.x中，过滤器更名为“管道（pipe）”

自定义管道的步骤：

1. 创建管道class，实现转换功能

@Pipe({

name: "sex" //过滤器/管道的名字

})

export class SexPipe {

//管道中执行过滤任务的是一个固定的函数

transform(val:number){

if(val==1) {

return '男'

}else if(val==0) {

return '女'

}else{

return '未知'

}

}

}

1. 在模块中注册管道

//app.module.ts

Declarations:[SexPipe]

1. 在模版视图中使用管道

{{e.empSex|sex}}

也可以用到属性里

<p title=”empSex|sex”>

说明：管道的transform方法除了val还可以接受其他参数，调用管道时冒号为这些参数赋值，例如：{{e.empSex|sex:’en’}}

创建管道的工具：ng g pipe zzmm

1. 创建一个政治面貌转换过滤器/管道ZzmmPipe，可以接收一个可选参数，format，转换规则如下：

如果format 值为‘short’，则转换规则为10-党员，20-团员，30-群众

如果format值为‘long’，则转换规则为10-中国共产党员，20-中国共青团员，30-中国人民群众

Vue.js中没有预定义管道，Angular有

1. Lowercase：转换成小写

<td>{{e.ename|lowercase}}</td>

1. Uppercase：转换成大写

<td>{{e.ename|uppercase}}</td>

1. Titlecase：转换成首字母大写

<td>{{e.ename|titlecase}}</td>

Note: 两个word要分开才能识别两个分别首字母大写

1. Slice：只显示字符串的一部分

<td>{{e.ename|slice:0:3}}</td>

1. Json：把js对象序列化成json字符串，在调试的时候很有用
2. Number：把数字转换为具有指定整数位和小数位的字符串

{{num|number}}

{{num|number:’4.1-4’}}

1. Currency：把数字转换成货币格式字符串：货币符号+三位一逗号+二位小数位

{{num|currency}}

{{num|currency:’¥’}}

1. Date：把数字转换成日期字符串

{{num|date:’yyyy-MM-dd HH:mm:ss’}}

1. 创建对象的两种方式

方式一：手工创建式 – 自己创建：let c2 = new Car()

方式二：依赖注入式 – 无需自己new，只需要声明依赖；服务提供者就会创建被依赖的对象，注入给服务需要者

1. Angular核心概念之六 – 服务和依赖注入 – 抽象但重点！

Service：服务，跟用户有关的放在组件（跟用户交互的一种对象）里，其中的内容都应该与用户操作有关系的，而跟用户操作无关的都应该剥离出去，放在“服务对象”中，为组件服务；例如：日志记录，计时统计，数据服务器的访问。

但vue.js不一样，他没有服务的概念，都是放在组件里，所以vue.js是中小型框架，angular是大型框架。

创建服务对象的步骤：

1. 创建服务对象并指定服务提供者

@Injectable({providedIn:’root’})

Export class LogService{}

1. 在组件中声明依赖，服务提供者就会自动注入进来，组件直接使用服务对象即可

Constructor(log: LogService){

//此处的log变量就会被注入为LogService实例

}



依赖注入

Dependency injection

组件

服务提供者

创建

Instance

声明依赖

Dependency

服务

面试题：前端有哪些异步请求工具，各自的利弊：

1. 原生XHR
2. Jquery.ajax()
3. Axios
4. Angular HttpClient
5. Fetch
6. 使用Angular官方提供的服务对象-HttpClient Service

HttpClient服务对象用于向指定的URL发起异步请求，使用步骤：

1. 在主模块中导入HttpClient服务所在的模块

//app.module.ts

Imports:[BrowserModule, FormsModule, HttpClientModule]

1. 在需要用到异步请求的组件中声明依赖于HttpClient服务对象，就可以使用该对象发起异步请求了。

http = null

constructor(http: HttpClient){

this.http = http

}



<https://www.tslang.cn> Typescript

TS的拓展特性：

1. TS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/函数都要声明类型

Uname:string

F1(n1:number):Boolean{

Return true

}

常用的数据类型关键词：string/number/Boolean/any/number[]/object/date/void

1. TS对属性和方法成员定义了三种访问修饰符

Private：只能在本类内部使用

Protected：被保护的成员只能在本类中或者子类内部使用

Public：在本类中以及外部使用

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，设置为public

1. 访问修饰符的特殊用法

下边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){}  } |

1. 面向对象编程，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，如所有的管道类必须提供transform方法

Interface Runnable{

Start()

Stop()

}

Class Car implements Runnable{

Start(){ } //接口实现必须提供指定的方法要有主体

Stop(){ }

}

1. 装饰器

见手册

1. Angular官方提供的服务对象 – httpClient

使用步骤：

1. 在HttpClientModule模块

//app.module.ts

Imports:[BrowserModule, HttpClientModule]

1. 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){ }

1. 调用HttpClient实例实现异步请求

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



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

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；基于毁掉方式处理响应 |
| 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 | 比原生简单；基于Promise处理响应；可以排队并发。 |
| 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 | 比XHR从根本上就更加先进，天然基于promise |
|  |  |  |
|  |  |  |

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跑出特定的事件

Subscribe：订阅，订阅可被关注的对象的新消息，一旦得到新消息立即会被通知到

TS是强类型语言

1. 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：

1. Constructor() 组件对象被创建了
2. NgOnchanges() 组件绑定的属性值发生改变

<myc01 [title]=”mytitle”>

1. ngOnInit() 组件初始化完毕 – 等同于Vue.js的mounted (\*\*\*)
2. ngDoCheck() 组件检查到了系统对自己的影响
3. ngAfterContentInit() 组件的内容初始化完成
4. ngAfterContentChecked() 组件的那日发生变化需要检查

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

1. ngAfterViewInit() 组件的视图初始化完成
2. ngAfterViewChecked() 组件的视图发生变化需要检查

template:’<p>{{uname}}</p>’

1. ngOnDestroy() 组件即将被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”

1. 面试题：Angular模块中可以声明哪些种类的对象？

组件/指令/管道/服务(providers)/其他模块(imports)/fn/value

1. 父子间组件传递—重点&难点

Vue.js和Angular中的父子间消息传递原理一样，都可以用口诀：“Props Down, Events Up”

方向一：父 => 子

父组件通过“子组件的自定义属性”向下传递数据给子组件

1. Parent.ts: 父组件创建数据

userName=”Carmen”

1. Parent.html: 父组件将自己的数据绑定给子组件的属性

<child2 [child2Name]=”userName”></child2>

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

1. Child2.ts：子组件定义扩展属性

@input() //声明为输入型属性

Child2Name = “”

1. Child2.html：子组件使用自定义属性

<p>{{child2Name}}</p>

方向二：子 => 父

子组件通过触发特定的事件（其中携带着数据），把数据传递给父组件（父组件提供事件处理方法）

1. Child.ts：自定义事件发射器 – 输出型属性

@output()

Private cryEvent = new EventEmitter()

1. Child1.ts：在某个时刻发射事件

This.cryEvent.emit(子组件想穿给父组件的数据)

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

1. Parent.html：监听子组件的事件

<child1 (cryEvent)=”doCry($event)”></child1>

1. Parent.ts：接收子组件传递的数据

doCry(e){e就是子组件想传给父组件的数据}

1. 总结：组件的模版中可以出现的内容

<div>

<p \*ngIf=“isMarried”></p>

<myc01 #c01></myc01>

<li \*ngFor=”let e of empList”>{{e}}</li>

<button (click)=”doClick($event)”></button>

<div #myDiv></div>

</div>

1. HTML标准元素 – 如div/id…
2. 组件对应的元素 – 如 myc01
3. Angular指令 – 如 ngIf/ngFor
4. 模型数据/模型方法-Model/empList
5. 事件对象/子组件传递的数据 --$event

<button (click)=”doClick($event)”></button>

1. 模版变量

<li \*ngFor=”let e of empList”>{{e}}</li>

1. NG元素识别符

<div #myDiv></div>

父子组件传递数据的简便方法：父亲直接使用子组件的引用 #xxx

1. 路由和导航

多页面应用：一个项目中有多个完整的HTML文件，使用超链接跳转-销毁一颗dom树，同步请求留一颗，得到之后再重建新的DOM树，不足：DOM树要反复重建，间隔中客户端一片空白

单页面应用：SPA，single page application，整个项目中有且只有一个“完整”的HTML文件，其他的“页面”只是DIV片段，需要哪个就异步请求下来，插入到完整的html文件中

优势：整个项目中客户端只需要下载一个html页面，创建一个完整的DOM树，页面跳转都是一个DIV替换另一个DIV而已—能够实现过场动画

单页应用的不足：不利于SEO访问优化

1. Angular中使用“单页应用”的步骤：
2. 创建整个应用需要的路由组件
3. 定义“路由词典”—[{URL-组件},{ URL-组件}]

//app.module.ts 为每个路由组件分配一个路由地址, path:’’,component:

let routes = [

{path:'index',component:IndexComponent},

{path:'plist',component:ProductListComponent},

{path:'pdetail',component:ProductDetailComponent},

{path:'ucenter',component:UserCenterComponent}

]

1. 注册“路由词典”

//app.module.ts

imports: [

BrowserModule,

AppRoutingModule,

FormsModule,

RouterModule.forRoot(routes)

],

1. 创建路由组件挂载点—称为“路由出口”

//app.component.html

<router-outlet></router-outlet>

1. 访问测试

<http://127.0.0.1:4200/ucenter>

注意：

路由词典中的路由地址不能以/开头或者结尾，但中间可以包含/

路由词典中可以指定一个默认首页地址：

{path:'',component:UserCenterComponent}

路由词典中每个路由中要么指定component（由哪个组件提供内容），要么指定redirectTo（重定向到另一个路由地址）

路由词典中可以指定一个匹配任意地址的地址：\*\*，注意改地址只能用于整个路由词典的最后一个！！

1. 路由跳转/导航：从一个路由地址跳转到另一个

实现方案：

方式一：使用模板方法

<any routerLink=”/ucenter”></any>

注意：（1）可以用于任意标签（2）跳转地址应该怡/开头，防止以相对方式跳转

方式二：使用脚本方法

this.$route this.$router

注意：Router类是RouterModule提供的

constructor(private router:Router){

}

jump(){

//跳转到商品列表页-需要“路由器”服务

this.router.navigateByUrl('/plist')

}

Vue.js中的路由跳转的机制有哪些？

1. hash法

<http://127.0.0.1/index/html#/ucenter>

1. history法：需要修改window.history对象，从而支持浏览器自带的后退按钮

<http://127.0.0.1/ucenter>

Angular中的路由跳转的机制有哪些？

History法：需要修改window.history对象，从而支持浏览器自带的后退按钮

1. 练习：在首页添加一个超链接，可以跳转到商品列表；在商品列表页面下创建三个超链接，表示三个商品，都可以跳转到商品详情页
2. 路由参数

在路由词典中定义路由地址时，其中可以包含可变的参数：

{path:’product/detail/:lid’,component:…}

在路由跳转时可以为路由参数提供具体的参数值：

<a routerLink=”product/pdetail/5”>

<a routerLink=”product/pdetail/13”>

道理目标路由组件，可以读取“当前路由地址”中的路由参数：

Constructor(private route:ActivatedRoute){}

ngOnInit(){

//读取路由参数

This.route.params.subscribe((data)=>{

Data.lid就是路由参数})

}

1. 补充知识点：路由嵌套

一级路由：

Index：首页

User/center：用户中心

二级路由：

User/center/avatar：用户中心 > 更改头像

User/center/info：用户中心 > 个人信息

User/center/security：用户中心 > 安全管理

路由嵌套修改路由词典：

const routes = [

{path:'',component:IndexComponent},

{path:'index',component:IndexComponent},

{

path:'user/center',

component:UserCenterComponent,

children:[

{path:'info',component:InfoComponent},

{path:'avatar',component:AvatarComponent},

{path:'security',component:SecurityComponent}

]

},

{path:'\*\*',component:NotFoundComponent}

]

注意：用户中心的二级路由挂载点/路由出口应该放在UserCenter.Component

1. 路由守卫

商业项目中，有些路由只能在特定条件下才能访问，比如：

用户中心，只能在登陆后才能访问

TMOOC视频播放，只能在学校内播放（IP地址限制）

VIP学员视频播放，只能在13:00-22:00时间内播放（时间限制）

商城后台所有页面，只能登陆管理员之后才能使用

Angular提供了“路由守卫（Guard）”实现访问路由组件之前检查功能

客户端 -> 登录守卫 -> 身份守卫 -> 时间守卫 -> 组件

使用路由守卫的步骤：

1. 创建路由守卫class

import { Injectable } from "@angular/core"

@Injectable({ //路由守卫都是可注入的服务对象

providedIn:'root'

})

export class loginGuard {

isLogin = true

canActivate(){

if (this.isLogin) {

return true

} else {

return false

}

}

}

1. 在路由词典中使用路由守卫

{

path:'user/center',

component:UserCenterComponent,

//当前路由组件可以被激活吗？-- 应用路由守卫

canActivate: [loginGuard]

}

1. 创建移动App可用的技术

|  |  |  |  |
| --- | --- | --- | --- |
| 技术名称 | 典型代表 | 优势 | 劣势 |
| 原生开发  Native Code | Android：Java或者Kotlin；iOS：Object-C或者Swift | 运行速度快/功能丰富 | 两种设备互不兼容，开发速度慢 |
| WebView开发（内核是浏览器）  H5/CSS/JS | Vue.js + Mint-UI | 开发速度快 | 运行速度慢  功能有限 |
| 混编开发  H5+原生 | Phonegap/Cordova  Angular + lonic组件库 | 开发速度快  功能丰富 | 运行速度慢 |
| JS Bridge  编码是JS,运行是原生代码 | React + ReactNative | 开发速度快  运行速度快  功能丰富 | 原生组件在两种平台下效果不一致 |
| GPU绘图 | Google Flutter/Dart | 运行速度快/  功能丰富/  两种平台效果一致 | 目前还有不足（例如绘图库代码过大） |
|  |  |  |  |
|  |  |  |  |

关于vue.js的资料在github上搜awesome vue

关于angular的资料在github上搜awesome angular

关于react的资料在github上搜awesome react

1. Angular UI 组件库 -ionic

官网：<https://ionicframework.com/>

概述：是一个移动端UI组件库，可以和Angular/Vue/React组合使用，也可以单独使用（script直接引入）

使用步骤：

1. 下载并安装全局的脚手架工具

npm i -g ionic

1. 运行脚手架工具，创建一个空白项目

Ionic start 项目名

1. 进入空白项目，启动开发服务器

npm start

1. 使用客户端访问测试

Ionic的九种主题色：

1. Primary：蓝+白色
2. Secondary：青+白色
3. Tertiary：紫+白色
4. Success：绿+白色
5. Warning：黄+白色
6. Danger：红+白色
7. Dark：黑+白色
8. Light：白+黑色
9. Medium：灰+白色

Ionic组件在iOS和Android的效果是不一样的：

Ios风格：

Md风格：google提供的material design风格

二者标题栏不同，弹出框不同，按钮不同，图片显示效果不同

APP页面结构

<ion-app>

<ion-header>

<ion-toolbar>

<ion-title>Main page</ion-title>

</ion-toolbar>

</ion-header>

<ion-content padding>

</ion-content>

<ion-footer>

<ion-toolbar>

<ion-title>Copyright</ion-title>

</ion-toolbar>

</ion-footer>

</ion-app>

1. Ionic中的响应式网络布局系统

<https://ionicframework.com/docs/api/grid>

ionic提供了一套类似bootstrap的响应式栅格布局系统

<ion-grid>

<ion-row>

<ion-col>

<div class="box">1</div>

</ion-col>

</ion-row>

</ion-grid>

特点：

列可以不指定宽度占比，在一行中的多个列的宽度会平均分配，一行中列的数量取决于屏幕宽度

可以使用size属性来指定一列的宽度占比-总分为12；6就是6/12，4就是4/12

可以使用offset属性指定一列向右偏移指定的宽度—底层是用margin-left实现的，会影响当前列或者后续的列

可以使用push和pull属性调整一列的出现位置-底层使用 绝对定为实现的，故不会影响同一行的其他列

1. Ionic常用的UI组件

<https://ionicframework.com/docs/components>

1. Badge

<ion-badge color="primary">11</ion-badge> 九种主题色之一

1. Icon

<ion-icon name="person" color="success"></ion-icon>

很多icon可以参考手册

1. 自学ionic常用组件：button，input，searchbar，card，toobar，slices
2. 使用上述组件完成首页组件内容
3. 按钮

<ion-button color=“主题色” size=”small/large” expand=”block/full”shape=”round”></ion-button>

1. 输入框

<ion-input type=”type” placeholder=”” maxlength=””></ion-input>

带边框和标签文字的输入框

<ion-item>

<ion-label position=”fixed/stacked/floating”>标签文字</ion-label>

<ion-input [(ngModel)]=”userInput”></ion-input>

</ion-item>

1. 列表项（ion-item）
2. 搜索框

<ion-searchbar color="dark" showCancelButton="focus" (ngModelChange)="doChange()"></ion-searchbar>

1. 卡片（ion-card）

<ion-card>

<img>

<ion-card-header>

<ion-card-title></ion-card-title>

<ion-card-subtitle></ion-card-subtitle >

</ion-card-header>

<ion-card-content></ion-card-content>

</ion-card >

1. 轮播广告（ion-slides）
2. 列表项（ion-item）

只要见到了有下划线的组件，就想到了列表项

<ion-item detail=”true”href=”#” color=””>

<ion-avatar/thumbnail slot=”start/end”>img</ion-avatar/ thumbnail >

<ion-label>文本</ion-label>

<ion-button slot=”end”>文字图标</ion-button>

</ion-item>

1. 列表list

<ion-list>

<ion-item>

</ion-item>

</ion-list>

1. 无限滚动（ion-infinite-scroll）

在滚动到底部指定的临界点（threshold）的时候，立刻触发特定时间，显示出底部的加载中，提示符号和文字，等数据加载完了（complete），再次隐藏加载中提示信息，等待用户继续滚动

<ion-infinite-scroll threshold=”50px”(ioninfinite)=”loadMore()”>

<ion-infinite-scroll-content loadingText=”加载中”loadingSpinner=”bubbles”></ion-infinite-scroll>

</ion-infinite-scroll>

loadMore(e){

…异步加载更多的数据

e.target.complete()

}

1. Actionsheet/alert/modal，都是弹出式对话框组件

提示：这三个组件都没有对应的元素，都是通过XXXcontroller动态创建出来的

Constructor（private controller：alertcontroller）{}

doXXX(){

this.controller.create({

header:”标题字”,

message:”提示信息”，

buttons:[“按钮1”，“按钮2”]

}).then((dialog)=>{

Dialog.present() //异步创建出来的对话框

})

}

|  |  |  |
| --- | --- | --- |
|  | Angular路由系统 | Ionic路由系统 |
| 创建路由词典 | //app.module.ts  Const routes = [{path:””,component:…}] | 一样的 |
| 注册路由词典 | //app.module.ts  Imports:[BrowserModule, RouterModule.forRoot(routes)] | 一样的 |
| 路由出口 | <router-outlet></router-outlet > | <ion-router-outlet></ion-router-outlet > |
| 路由跳转 |  |  |
|  |  |  |

Ionic组件不会自动注册

2. 创建空白的Ionic项目模版
3. 创建项目必需的组件

ng g component main 陈放页签组件的页面

ng g component index 首页

ng g component product-list 商品列表页

ng g component product-detail 商品详情页

ng g component cart 购物车页

ng g component user-login 登录页

ng g component not-found 404页

确保每个组件都在app.module.ts中进行声明：declarations

1. 修改根组件模版app.component.html, 使用MainComponent(其中要创建页签组件)

<ion-app>

<ion-router-outlet></ ion-router-outlet>

<app-main></app-main>

</ion-app>

1. 编辑main.component.html, 声明页签组件