Angular

张众磊 微信:teduzzl 邮箱:bjzhangzl@tedu.cn

# 推荐项目:

1、vue中很棒的组件库

https://github.com/vuejs/awesome-vue

和过滤器相关的：

https://github.com/freearhey/vue2-filters

2、angular

angular.cn

https://github.com/PatrickJS/awesome-angular

建议:

1、数据的操作

定义\读\写

2、处理问题策略

从数据着手--》显示--》交互

学习计划:

1、Angular以及生态圈

完成中大型SPA的开发

完成webApp的开发

2、React以及生态圈

完成React基于web的页面开发

完成原生开发

3、Python以及生态圈

完成python基础语法的学习

完成django全栈类框架的学习

编程建议:

①修复bug

查看具体错误消息

修正代码

②处理业务问题思路

找数据

渲染

交互

# 一、回顾Vue

0、vue的生态圈(技术栈)

vue核心库+vuex+vue-router

ui组件库(ElementUI/MintUi)+axios

vuex(weex--nativeapp vux--移动端ui组件库 。。。)

## 1、组件的创建和使用

组件?

是可被反复使用的，带有特定功能的视图！

Vue.component('',{

components:{

}

})

SFC: \*\*\*.vue

<my-header></my-header>

## 2、基础语法（指令、过滤器。。。）

指令??

v-for

v-if/v-else-if/v-else

v-on:click ---> @click 事件绑定

v-bind:src --> :src 属性绑定

v-model 双向数据绑定

view-->model 表单元素视图操作的结果 绑定数据

model-->view 数据绑定到视图

常见属性??

computed 计算属性

watch 监听属性

props 接收父组件调用组件通过属性传来的值

data/methods/filters/components/...

## 3、通信

3.1 组件间通信

props down:父--》子

events up: 子--》父

event bus: 兄弟组件

$refs $parent

3.2 与远程服务器端的通信

npm i axios

import Axios from 'axios'

Axios.get/post().then((result)=>{//result.data})

## 4、路由模块的使用 vue-router

路由的本质:建立起url和组件之间的映射关系

### 4.1 基本使用

①容器

<router-view></router-view>

②创建组件

③配置路由词典(由多个路由对象构成的数组)

var myRouter = new VueRouter({

routes:[

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

{path:'/login',component:LoginComponent}

]

})

new Vue({

router:myRouter

})

### 4.2 跳转

this.$router.push()

<router-link to=""></router-link>

### 4.3 传参

①配置接收方的路由地址

/detail ---> /detail/:id

②发送

this.$router.push('/detail/10')

③接收

this.$route.params.id

### 4.4 路由嵌套

一个spa，A组件需要嵌套B组件、C组件

①给A组件指定一个容器

router-view

②给A组件的路由对象，添加一个children

{

path:'/a',

component:A,

children:[

{path:'/b',component:B},

...

]

}

### 4.5 路由守卫

指定全局守卫：

this.$router.beforeEach((to,from,next)=>{

//to记录去哪

//from 来自哪里

//next 决定是否允许向前进

})

指定路由独享守卫:

{

path: '/foo',

component: Foo,

beforeEnter: (to, from, next) => {

// ...

}

}

## 5、vuex

集中式的管理数据，按照约定的方式完成数据的读(getters/state)写(mutation),异步操作(action)

state:

负责数据的定义

getters：

负责在state数据基础之上派生新数据

mutations：

完成state中数据的修改操作

actions:

异步处理

modules:

模块化

# 二、设计原则

1、YAGNI

You Aren't Gonna Need It 你不需要这个东西

2、KISS

Keep It Simple And Stupid 足够的简洁和简单

3、DRY

Don't Repeat Yourself 不要重复自己

4、SRP

Single Responsibity Principle 单一责任原则

不要在一个地方封装过多的功能

5、OCP

Open Closed Principle 开闭原则

对于工程的升级迭代工程，建议对于扩展开放，对于修改原有的代码

6、高内聚低耦合

耦合:不同部分之间他们之间的关联，应该越少越好(低耦合)

7、最少知识法则

尽可能减少与其他模块间的关系

# 三、Angular

Angular的生态圈:

核心模块

FormsModule

HttpClientModule

RouterModule

ui组件库:

NgZorro(pc\mobile)

MaterialDesign、MaterialDesignMobile

ionic

....

## 1、概述

区别:

vue是渐进式的框架，采用组件化的编程方式

Angular是一个可以实现中大型SPA的框架，采用模块化的编程方式，提供了更多的类

what?

Angular 早在09年就已经由Google所推出的一个遵循MVVM设计模式的js框架

when?

中大型、超大型SPA(single page application)

why?

①强大的cli支持

②typescript(ts)是es6的超集

③基于angular(ng)的第三方工具和选择比较多

④采用了模块化的开发方式，可以应对更复杂的开发需求

how?

npm install -g @angular/cli

ng new my-app

cd my-app

ng serve --open

准备ng的项目运行环境:

①node版本

8.11.1

②压缩包

forStu.zip

**使用ng开发的流程:**

①封装组件来拆分视图

②封装模块来拆分应用的业务

③封装服务来完成数据和方法的共享

④封装指令和管道 来增强html功能

。。。

①准备一个文件夹

C:/xampp/htdocs/codes

②将压缩包拷贝到上述目录

forStu.zip

③将压缩包解压缩到当前目录

④拷贝C:\xampp\htdocs\codes\forStu\ng\Angular.rar到C:\xampp\htdocs\codes\ng目录中

链接: https://pan.baidu.com/s/1yqNF475\_VzhG6-CZ3V-gFw

提取码: 9xc5

forStu.zip下载链接:

链接: https://pan.baidu.com/s/1tlzOkHf\_qForOU0HnAqW-w 提取码: 7cp9

**常见错误:**

①启动

在当前的myngapp工程下面执行 npm start

启动开发服务器

②node版本

v8.11.1

E:/software/node\*\*\*.msi

③powershell

文件--》首选项--》设置--》

在右侧加上这句话:

"terminal.integrated.shell.windows": "C:\\Windows\\System32\\cmd.exe"

启动流程:

main.ts --> app.module.ts -->app.component.ts(app.component.html)

**两个概念:**

装饰器(decorator)：

装饰器是一个方法，是用来告诉angular这个是一个什么样的类

@NgModule 模块

@Component 组件

元数据(metaData)：

是用来告诉angular如何来处理当前这个类

@Component({

selector:'app-demo01'，//作为标签拿去

templateUrl:'',

styleUrls:[]

})

## 2、组件的创建和使用

准备工作:

将C:\xampp\htdocs\codes\ng\myngapp\node\_modules\.bin添加到了计算机的环境变量

注意事项：

设置过环境变量之后，关闭cmd窗口或者vscode，重新打开试下

创建一个组件：

ng g component demo01

(创建了4个文件，更新了app.module.ts--declarations)

调用组件:

每一个组件都有一个selector的属性:

<app-demo01></app-demo01>

## 3、基础语法(重点)

{{}}

<any>{{expression}}</any>

**循环指令**

<any \*ngFor="let tmp of myList"></any>

<any \*ngFor="let tmp of myList;let myIndex=index"></any>

**判断指令**

<any \*ngIf="expression"></any>

expression结果为真，渲染；否则从DOM中移除

<div [ngSwitch]="expression">

<p \*ngSwitchCase="情况1"></p>

<p \*ngSwitchCase="情况2"></p>

<p \*ngSwitchDefault></p>

</div>

<div [ngSwitch]="true">

<p \*ngSwitchCase="score>90"></p>

<p \*ngSwitchCase="score>80 && score<90"></p>

...

<p \*ngSwitchDefault></p>

</div>

**绑定：**

属性绑定 []

事件绑定 ()

双向数据绑定

准备工作:

src/app/app.module.ts

import {FormsModule} from '@angular/forms'

@NgModule({

imports:[FormsModule]

})

使用:

<表单元素 [(ngModel)]=""></表单元素>

注意事项:

①结构性指令不能同时用在一个标签

\*ngFor \*ngIf \*ngSwitchCase \*ngSwitchDefault ；

可以使用：

<ng-container \*ngFor="let item of list;let index=index">

<li \*ngIf="index%2 === 0">

{{"index is " + index + " value is " + item}}

</li>

</ng-container>

②调用双向数据绑定和ngModelChange事件时要注意显示顺序

<input [(ngModel)]="uname" (ngModelChange)="handleChange()"/>

③指令在ng中分为两大类，一个是结构型指令，一个是属性型指令，影响 HTML的DOM结构，HTML模板的增强

## 4、管道(pipe/filter)

vue:

<any>{{expression | myFilter(1,2) | myFilter2}}</any>

管道：管道本质就是一个有参数 有返回值的方法;管道可以帮助我们对要显示的数据做一些筛选、过滤、格式化

**angular使用管道:**

语法:

<any>{{expression | myPipe}}</any>

<any>{{expression | myPipe:1:2}}</any>

<any>{{expression | myPipe:1:2 | myPipe2}}</any>

**①内置管道**

**number** '3.4-5'

**currency** 货币 参数可以是货币代码CNY EUR...

**json** 实现将对象或者数组转换为 一个json格式的字符串(json序列化)

**slice** 对数组或者字符串进行截取

slice:1 从小标为1的开始截取，一直到最后一个

slice:1:3 从1开始到3截止(不包含)

**percent**

**uppercase**

**lowercase**

**②自定义管道**

0/1 -->女/男

步骤1：ng g **pipe** sex

步骤2：具体实现

在transform方法，根据管道的要求对数据做处理，最后将处理的结果通过返回值 来指定管道被调用后最终要显示的结果

步骤3：和内置的管道一个用法

// 具体要求，将0,1转为女，男，必须要有返回值

transform(value: any, args?: any): any {

if(value==0){

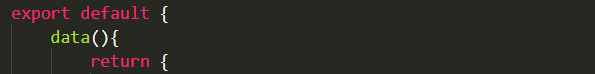
return "女"

}else if(value==1){

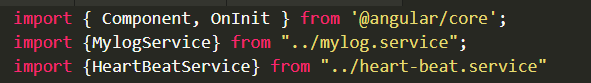
return "男"

}

## **5、通信(\*\*重点\*\*)**



如果有default 引入时，就不需要加{}



否则就需要加{}

### ①父-->子(Input)

props down

组件封装完成之后，被调用时需要动态的传值，让组件更灵活的来渲染数据

**Vue:**

步骤1: <son myName='zhangsan'></son>

步骤2: 接收

Vue.component('son',{

//this.myName {{myName}}

props:['myName']

})

**Angular:**

步骤1: <father myName="lucy"></father>

步骤2：接收

import {Input} from '@angular/core'

//①允许接收myName属性的值 ②将值存在myName变量

@Input() myName = ""

//this.myName {{myName}}

<son myName={{myName}}></son>

### ②子-->父 (Output,EventEmitter,$event)

events up

步骤1：绑定(父组件)

//$event是angular内部在处理事件时 ng内部传来的数据，在当前场景下其实就是事件触发是传来的数据

<son (myEvent)="handleEvent($event)">

</son>

handleEvent(msg){

//msg就是输入框的值

}

步骤2：触发事件(子组件)

import {Output,EventEmitter} from '@angular/core'

//定义一个可以触发'myEvent事件'的事件触发器：myEvent

@Output() myEvent = new EventEmiiter()

//触发

this.myEvent.emit(123)

练习:(:- 15:25)

①准备两个组件demo09-list demo09-item

②指定组件的模板内容

demo09-list ul

demo09-item li(span+button)

③指定功能1

demo09-list 准备一个数组

myList=["商品1","商品2","商品3"]

调用demo09-item时将myList的值传递，让span显示

④指定功能2 (:- 16:20)

点击app-demo09-item的delete按钮时候，将数值1 传递给app-demo09-list

子组件:demo09-item

父组件:demo09-list

目标：子--1--》父

手段: 子组件触发一个事件传值，父组件在事件的处理函数中接收数值

步骤1:父组件中写代码($event)

绑定事件 delEvent

步骤2:子组件中写代码 (Output,EventEmitter)

触发事件

### ③与远程服务器端通信

HttpClient

总结:

父--》子 Input

子--》父 Output EventEmitter $event

## 6、服务 service

服务是一个类，封装一些数据和方法，方便不同的组件来调用

服务的使用场景:

日志服务、心跳服务。。。。

①创建

ng g **service** mylog

②在服务类中封装数据或者方法

③调用

组件是服务的最大的消费者

//引入

import {MylogService} from '../mylog.service'

//实例化

constructor(private myService:MylogService){}

//调用实例中所对应的方法或者数据

this.myService.showLog('----')

练习:(:- 17:52)

①在demo10-test准备两个按钮，一个开始，一个结束

②在当前工程中，准备一个服务 heart-beat

ng g service heart-beat

在此服务中，封装两个方法

startBeat：启动定时器，每隔0.5s，输出一句话

stopBeat：结束定时器的运行

③在demo10-test引入心跳服务，准备两个方法给按钮使用，在方法中分别调用服务中所封装方法

总结:

①pipe：筛选、过滤、格式化 本质：有参数有返回值的方法

②props down (Input)

父--》子

③events up (Output,EventEmitter,$event)

子--》父

④service

创建，用的步骤：

①引入

②实例化

③调用

## 7、路由

路由的本质：建立起url和组件间的映射关系

### 7.1 路由的基本用法

准备工作:

完成一个自定义模块(包含路由配置)的创建和调用

ng g module my-module --routing

app.module.ts/引入/imports

①**准备一个容器**

<router-outlet></router-outlet>

forChild-->forRoot

②创建组件

Login/Register/Cart...

ng g component demo13-login

ng g component demo13-register

③配置路由词典

引入组件

各自指定url地址

总结：

①完成自定义路由模块的创建和调用

②router-outlet

③创建组件

④路由词典：是一个多个路由对象构成的数组

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

### 7.2 跳转和传参

7.2.1 跳转(Router)

vue:

this.$router.push('')

router-link to="/login"

Angular:

**方式1：编程式导航**

import {Router} from '@angular/router'

constructor(private myRouter:Router){}

this.myRouter.navigateByUrl()

**方式2：routerLink**

<button routerLink="/register">注册</button>

练习:(:- 15:50)

①完成demo14-list demo14-detail两个组件的创建

②各自分配一个路由地址 list/detail

③demo14-list

准备一个字符串数组，将数组中数据显示在ul，点击li导航到详情页（使用Router类来导航）

navigateByUrl

### 7.2.2 传参(ActivatedRoute)

①配置接收方的路由地址

detail --> detail/:id

②发送

this.myRouter.navigateByUrl('detail/3')

<any routerLink="detail/3"></any>

③接收

vue: this.$route.params.\*\*

import {ActivatedRoute} from '@angular/router'

constructor(private myRoute:ActivatedRoute){}

this.myRoute.params.subscribe((result)=>{

//result.id

})

注意事项：

① 路由地址设置detail/:id 和 接收参数result.id 两个属性名字要保持一致

② 动态传值

myNum = 10

routerLink="/detail/myNum"

### 7.3 路由守卫(canActivate)

守卫：在一个spa中，完成路由词典设置之后，可以通过路由地址来访问任何的一个页面，但是有些页面，比如后台管理界面、需要权限校验的页面，在正式显示之前做一些相关的判断处理

在angular中路由是可以通过封装服务来实现

步骤1：创建服务的这个文件

ng g service my-guard

步骤2：在服务类中 实现了canActivate

import {CanActivate} from '@angular/router'

export class MyGuardService implements CanActivate{

canActivate(){

//在此来做一些功能处理 来决定是否可以访问你要保护的页面

return true/false

}

}

步骤3：调用守卫服务来保护一个页面

import {MyGuardService} from '../my-guard.service'

[

{

path:'admin',

component:AdminComponent,

canActivate:[MyGuardService]

}

]

### 7.4 路由嵌套

angular的路由嵌套和vue中一模一样

举例：mail需要嵌套inbox、outbox

步骤1：

在mail的指定一个容器

<router-outlet></router-outlet>

步骤2:

在mail的路由对象中指定一个属性:

{

path:'mail',

component:MailComponent,

children:[

{},

{}

]

}

注意事项：

访问页面时

http://localhost:4200/mail/outbox

准备工作:①启动xampp ②通过node启动http服务器

综合练习:(:- 10:20)

①完成demo16-cart demo16-submit两个组件的创建

②各自指定一个路由地址 cart submit

③demo16-cart

http://localhost:8080/product/list

页面初始化完成，发请求 拿数据保存(data所对应的数组)，显示在视图(ul/li显示商品的title) 点击li 跳转(routerLink)到submit页面，将商品的lid发送过去

①拿数据 ②保存并显示 ③处理交互

④demo16-submit

接收传来的参数，保存，显示在视图

考察：

①网络通讯

HttpClient

②跳转

Router

RouterLink

③传参

配置接收方路由 submit/:id

发送

this.myRouter.navigateByUrl('submit/'+lid)

routerLink="/submit/{{tmp.lid}}"

接收

ActivatedRoute

常用类:

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

import {HttpClient} from '@angular/common/http' //网络请求

import {FormsModule} from '@angular/forms'

import {Input} from '@angular/core' //父-》子

import {Output,EventEmitter} from '@angular/core' //子--》父

import {Router} from '@angular/router' //跳转

import {ActivatedRoute} from '@angular/router' //接收跳转时传来参数

今天目标:

①完成路由的核心知识的学习

路由嵌套！！！

②ionic常见的组件

# 四、Ionic （UI组件库）

npm i ionic@3.20.0

移动端的3种常见开发形式：

WebApp/NativeApp/HybridApp

WebApp：使用前端技术来实现的 运行在浏览器上的移动端的网页！！ (m.jd.com 跨平台)

NativeApp:使用java/kotlin/swift/oc来调用由apple或者google所提供的开发包来完成可以运行在手机操作系统的应用 （原生应用开发-比如Android/iOS 高性能）

HybridApp:使用前端技术和原生开发技术进行混合编程，最终生成的在手机操作系统运行的app

cli: command line inerface

gui: graphical user interface

## 1、概述

what?

Ionic是一个移动端的ui组件库，主要是给Angular进行使用

Ionic = angular+ IonicModule(集成了很多的好看的ui组件)+ionIcons + cordova

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

where?

Ionic可以实现webApp 还可以实现HybridApp

why?

①ionic基于angular

②强大的cli支持

③ionic内置了丰富的移动端的ui组件库，

在不同的操作系统运行时会遵循对应的系统风格（android-->MaterialDesign iOS->扁平化）

④维护了自己的一套图标体系

。。。

how?

搭建环境:

npm install -g ionic

ionic start myApp tabs

cd myApp

ionic serve

使用ionic:

ionic提供的组件

ionic封装的类

ionic中封装的导航工具

准备工作之模板项目：

①找到C:\xampp\htdocs\codes\forStu\ng\ionic\_资料.zip

②拷贝压缩包 粘贴到

C:\xampp\htdocs\codes\ionic；

将ionic\_资料.zip解压缩到当前文件夹

③拷贝C:\xampp\htdocs\codes\ionic\ionic模板\myProject.zip，粘贴到C:\xampp\htdocs\codes\ionic

④将C:\xampp\htdocs\codes\ionic\myProject.zip解压缩到当前文件夹

⑤通过vscode打开C:\xampp\htdocs\codes\ionic\myProject

⑥运行开发服务器

npm start

在浏览器的地址栏中测试localhost:8100

准备工作之ionic不可以使用:

①将C:\xampp\htdocs\codes\ionic\ionic模板\ionic\_cmd.zip解压缩到ionic\_cmd目录

②确认下C:\xampp\htdocs\codes\ionic\ionic模板\ionic\_cmd\node\_modules\.bin中有没有一个ionic.cmd的执行文件

③将C:\xampp\htdocs\codes\ionic\ionic模板\ionic\_cmd\node\_modules\.bin添加到环境变量中

我的电脑--》右键属性--》高级系统设置--》环境变量--》Path--》编辑

## 2、ionic常见的ui组件：

### 2.1 page的创建和使用

在ionic中将一个组件称之为一个page

创建一个page：

ionic g page demo01

注册：

到根模块中注册组件类 app.module.ts

import {Demo01Page} from '../pages/demo01/demo01'

@NgModule({

declarations:[Demo01Page],

entryComponents:[Demo01Page]

})

调用一个page:

app.component.ts

import {Demo01Page} from '../pages/demo01/demo01'

rootPage:any = Demo01Page

npm i ionic@3.20.0

### 2.2 button

color-->src/theme/variables.scss

<button ion-button outline clear color='' block large small></button>

按钮支持设置icon：

<button ion-button icon-left>

<ion-icon name="home"></ion-icon>

主页

</button>

<button ion-button icon-right>

设置

<ion-icon name="settings"></ion-icon>

</button>

练习:(:- 16:45)

①完成demo02-lianxi的page的创建和调用(根组件中设置为rootPage)

ionic g page demo02-lianxi

②在demo02-lianxi的组件类中 准备一个数组，遍历数组在ul中创建多个li，li指定一个按钮

### 2.3 list

**普通文本列表**

<ion-list>

<ion-item>123</ion-item>

</ion-list>

**icon列表**

<ion-list>

<ion-item>

<ion-icon name=""></ion-icon>

23

</ion-item>

</ion-list>

**联系人列表**

<ion-list>

<ion-item>

<ion-avatar item-start/end>

</ion-avatar>

</ion-item>

....

</ion-list>

**缩略图列表**

<ion-list>

<ion-item>

<ion-thumbnail item-start/end>

<img src=”” />

</ion-thumbnail>

</ion-item>

....

</ion-list>

**侧滑动**

<ion-list>

<ion-item-sliding>

<ion-item></ion-item>

<!-- side隐藏选项按钮出现的方向-->

<ion-item-options side="left">

<button></button>

</ion-item-options>

</ion-item-sliding>

</ion-list>

下拉刷新

上拉加载更多

。。。

练习:

①完成页面类的创建、注册、调用

demo04-lianxi

②在demo04-lianxi中 准备一个字符串数组

通过ionList来显示，列表项支持右侧滑动，滑动之后有一个删除的红色的 按钮，点击按钮时，将列表项从数组中移除splice

### 2.4 grid(自定义布局)

grid是基于flexbox来实现的,是可以帮助大家方便的实现自定义布局

flexbox:

direction 主轴方向，默认row

align-items 设置沿着交叉轴对齐

justify-content 沿着主轴对齐

①基础用法

<ion-grid>

<ion-row>

<ion-col></ion-col>

<ion-col></ion-col>

</ion-row>

<ion-row></ion-row>

</ion-grid>

②常见设置

纵向对齐的设置:

align-items-start/center/end 一行中所有列的纵向对齐

align-self-start/center/end 一列中内容的纵向对齐

横向对齐处理：

justify-content-start/center/end 一行中列的横向对齐的方式

某一列内容的横向对齐方式 text-align：''

列宽:col-\* (1~12)

偏移量的设置：offset-\*

右推/左拉： push/pull-\*

....

练习:(:- 14:42)

①完成demo07-lianxi的创建、注册、调用

②在demo07-lianxi组件类中 准备一个数组

["商品0",'商品1',"商品2",'商品3',"商品4",'商品5',"商品6",'商品7',"商品8",'商品9']

③到视图中通过ionGrid来显示，一行两列

（建议：可以先实现一行只显示一列 列的下标和2的余数为0）

\*ngFor

\*ngIf

### 2.5 ionSlides(轮播图、新手教程)

实现轮播图、新手教程

<ion-slides>

<ion-slide>123</ion-slide>

<ion-slide>456</ion-slide>

<ion-slide>789</ion-slide>

</ion-slides>

autoplay='2000' loop

effect="cube/flip/slide/fade"

speed='1500'

pager paginationType='bullets/fraction/progress'

direction='horizontal/vertical'

### 2.6 ionFAB（悬浮式操作按钮）

Floating Action Button 悬浮式操作按钮

<ion-fab left/right/middle top/bottom/center>

<button ion-fab>编辑</button>

<ion-fab-list side='top/left/right/bottom'>

<button ion-fab>\*\*</button>

</ion-fab-list>

</ion-fab>

### 2.7 ionCard（卡片化风格显示数据）

<ion-card>

<ion-card-header>

Card Header

</ion-card-header>

<ion-card-content>

<!-- Add card content here! -->

</ion-card-content>

</ion-card>

### 2.8 **交互窗口**

**loading: 一个活动正在进行**

使用步骤:

①引入

import {LoadingController} from 'ionic-angular'

②实例化

constructor(private loadingCtrl:LoadingController){}

③创建窗口

var myLoading = this.loadingCtrl.create()

④显示窗口

myLoading.present()

配置参数

this.loadingCtrl.create({

content:'',

duration:1000

})

myLoading.present()

myLoading.dismiss()

myLoading.setContent()

**toast:一个操作完成 通过通知来进行反馈,给一个结果**

ToastController

this.toastCtrl.create({

message:'',

duration:1500,

position:'top/middle/bottom',

showCloseButton:true,

closeButtonText:''

})

**actionSheet:让用户从多个选项中做选择**

ActionSheetController

{

title:"",

buttons:[

{

text:"",

handler:()=>{},

role:'cancel/destructive'

}

]

}

**alert:实现警告、确认、输入提示**

AlertController

this.alertCtrl.create({

title:'',

message:'',

inputs:[],

buttons:[]

})

**modal: 自定义模态窗口**

ModalController

this.modalCtrl.create(MyLoginPage)

进阶用法:

关闭模态窗口？

import {ViewController} from 'ionic-angular'

constructor(private viewCtrl:ViewController){}

this.viewCtrl.dismiss()

模态窗口在关闭时 如何完成数据的传递和接收?

this.viewCtrl.dismiss({id:1,age:20,name:'test'})

myModal.onDidDismiss((data)=>{

//data就是dismiss方法调用时 传递来的数据

})

练习:

①完成demo10-lianxi的页面的创建、注册、调用

②在demo10-lianxi放上一个按钮：‘验证手机号’

③点击按钮，通过AlertController来实现一个输入提示窗口（输入框，确定按钮，取消按钮），

点击确定按钮时，显示toast：

‘验证码已经成功的发送到123456’

### 2.9 ionScroll

实现一个支持滚动的容器，支持横向滚动、纵向滚动。。。

**<!-- 支持纵向滚动 -->**

<ion-scroll style="height:200px" scrollY>

<ion-card \*ngFor="let tmp of myList">

<ion-card-content>

<h2>{{tmp}}</h2>

</ion-card-content>

</ion-card>

</ion-scroll>

**<!-- 横向滚动 -->**

<ion-scroll style="height:200px;white-space:nowrap" scrollX>

<ion-card style="display:inline-block" \*ngFor="let tmp of myList">

<ion-card-content>

<h2>{{tmp}}</h2>

</ion-card-content>

</ion-card>

</ion-scroll>

### 2.10 表单组件

ionic中表单元素 是靠ionList和ionItem来进行管理的

<ion-list>

<ion-item>

<ion-label></ion-label>

<ion-input></ion-input>

<ion-range></ion-range>

</ion-item>

</ion-list>

<ion-label></ion-label>

<ion-input></ion-input>

<ion-range></ion-range>

<ion-checkbox></ion-checkbox>

<ion-radio></ion-radio>

### 2.11 ionTabs

tabs在ionic中实现一个固定在页脚位置的**底部工具条/导航条/选项卡**，可以方便的在移动端实现导航效果

步骤1:引入每一个tab被选中时要显示的页面类，并保存在类的内部成员

import {Demo12Page} from '\*\*\*'

export class Demo13TabsPage{

tabHome = Demo12Page

}

步骤2:调用组件ionTabs并设置root

<ion-tabs>

<ion-tab tabTitle="" tabIcon="" [root]="tabHome"></ion-tab>

<ion-tab></ion-tab>

<ion-tab></ion-tab>

</ion-tabs>

**注意事项:**

①在一个页面中如果需要设置ionTabs，无需保留之前ionHeader/ionContent

②将每一个tab被选中时要显示的页面类保存在类中之后，绑定给ion-tab的root属性

## 3、导航系统

ionic基于angular，angular自带路由模块，ionic当前这个版本自己实现了一个更简洁的导航方法

### 3.1 跳转

①编程式导航

import {Demo14MainPage} from '\*\*\*'

this.navCtrl.push(Demo14MainPage)

②属性指定

import {Demo14MainPage} from '\*\*\*'

demo14 = Demo14MainPage

<any [navPush]="demo14"></any>

练习:(:- 15:40)

①完成demo15-list demo15-detail的页面类的创建、注册，在根组件中调用demo15-list

②在demo15-list准备一个字符串数组，将数组通过ionList来显示，ionItem可以通过navPush导航到demo15-detail

返回:

this.navCtrl.pop()

### 3.2 传参

①发送

方案1：

this.navCtrl

.push(Demo15DetailPage,{id:1})

方案2:

<any [navPush]="detailPage" [navParams]='{id:1}'></any>

②接收

this.navParams.get('id')

## 4、lifecycle

使用框架所提供的组件的生命周期的钩子函数，在对应的阶段来完成初始化、清理工作、数据变化做逻辑处理

ng:

ngOnInit 初始化

ngOnChanges 在数据变化时做处理

ngOnDestroy 清理

ionic:

ionViewDidLoad 初始化（执行一次）

ionViewWillEnter 初始化（执行很多次）

ionViewWillUnload 清理工作(执行一次)

ionViewCanEnter 能不能进来 **返回true/false来决定是否能进入**

ionViewCanLeave 能不能离开 **返回true/false来决定是否能离开**

5、综合练习(:- 17:25)

①完成两个页面类的创建、注册

demo16-login (在根组件中调用登录页面)

demo16-main

②在demo16-login中 实现一个表单，两个输入框，分别输入用户名、密码，还有一个按钮， 点击按钮，跳转到demo16-main,同时将用户名传递

③demo16-main

接收通过路由跳转所传来的数据，将数据保存，显示在h2

课外：以NavPush的方式尝试跳转传参！

6、项目