

Angular04

提前生成今日项目包

- `ng new ngpro`
- `ng s -o`

作业

新闻列表

- 开启网络模块: 网络模块默认不加载, 必须在 `app.module.ts` 中进行加载
- 生成组件: `ng g c work01`

	河中发现男性尸体被装在不锈钢笼中 警方: 排除他杀 2020-07-31 10:00:38
	抢劫运钞车嫌犯官至副局长 家属: 死者太阳穴有枪伤 2020-07-31 10:00:38
	31省区市新增确诊127例 其中新疆112例 辽宁11例 2020-07-31 10:00:38
	美议员提案美军可武力保卫台湾 台媒:又吃台湾豆腐 2020-07-31 10:00:38
	苹果证实了! 今年新款iPhone上市将比往年晚几周 2020-07-31 10:00:38

```
.cell {  
  display: flex;  
  width: 500px;  
  background-color: aliceblue;  
  border-bottom: 1px solid gray;  
  padding: 5px;  
}  
  
.cell > img {  
  border-radius: 4px;  
}  
  
.cell > div {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-around;  
  margin-left: 5px;  
  font-size: 19px;  
}  
  
.cell:hover {  
  background-color: orange;
```

```

    cursor: pointer;
    border-radius: 10px;
}

```

```
<p>work01 works!</p>
```

```
<!--
```

网络请求的异步性

* 网络请求与网速有关，网速慢会比较卡；

默认情况下 程序运行是排队执行的。如果某个环境可以预料到会卡顿，则为了程序的流畅性，应该把这个卡顿的程序放到其他 `线程` 执行

执行完毕之后，则需要重新刷新页面；让内容显示出来！

此处必须判断：网络请求内容有了之后 再使用！

```
-->
```

```

<div *ngIf="res">
  <div *ngFor="let item of res.result" class="cell">
    <img [src]="item.image" alt="" />
    <div>
      <span class="title">{{ item.title }}</span>
      <span class="time">{{ item.passtime }}</span>
    </div>
  </div>
</div>

```

```
<!--
```

例子：

亮亮上厕所：正常流程-> 执行任务->用手纸 -> 穿裤子

亮亮没带纸，让然然去买纸

之前报错：穿裤子之前没判断是否有纸，结果就报错了！

```
-->
```

```

import { Component, OnInit } from '@angular/core';
// * 引入http服务
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-work01',
  templateUrl: './work01.component.html',
  styleUrls: ['./work01.component.css'],
})
export class Work01Component implements OnInit {
  // 构造方法 public http: HttpClient 是语法糖：会自动生成下方的http属性
  // http: HttpClient;

  // 依赖注入机制：

```

```

// constructor 被称为构造方法：构造方法的参数 在类实例化时 必须传递才可以！ -- 来自于面向对象：东哥
// 依赖：声明依赖 -- 构造方法中的参数 必须声明类型 变量名:类型名
// 注入：系统在实例化当前组件时，就可以根据类型自动提供对应类型的对象
constructor(public http: HttpClient) {}

res: Res; //设定变量类型，vscode才能够给出对应的提示

// 生命周期：相当于 vue 的 mounted. 挂载时
ngOnInit(): void {
    // this.http : 代表当前对象中的http成员属性
    let url = 'https://api.apiopen.top/getwangyiNews';

    // subscribe(回调方法)：订阅请求的返回值.
    this.http.get(url).subscribe((res: Res) => {
        console.log(res);
        // res: 局部变量，只能在当前{}中使用
        // this.res: 成员属性，必须提前声明才能使用；特点：可以全局通用，在html中使用！
        // 相当于vue 中的 data 属性中的值！
        this.res = res;
    });
}
}

// 高质量的代码，需要前端书写时，能够有代码提示；则需要定制数据类型
interface Result {
    image: string;
    passtime: string;
    title: string;
    path: string;
}

// 类型名是自定义的，你写Abc都可以..但是最好有含义！
interface Res {
    code: number;
    message: string;
    result: Result[]; //数组类型 中的元素 都是 Result类型
}

```

练习

学子商场 产品接口

写好之后到浏览器中测试

<http://101.96.128.94:9999/data/product/list.php?pno=1>

- 仿造之前的作业, 新建 myc01 组件, 完成此练习
- 注意通过 自定义数据类型, 实现html中的代码提示功能
- 这是一个 get请求
- 图片地址: 请求返回的是典型的相对路径, 需要人为拼接前缀才能显示
 - 前缀域名: <http://101.96.128.94:9999/>

```

import { Component, OnInit } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';

@Component({
  selector: 'app-myc01',
  templateUrl: './myc01.component.html',
  styleUrls: ['./myc01.component.css'],
})
export class Myc01Component implements OnInit {
  // 声明但未赋值：默认值是 undefined
  res: Result; //声明类型 html中才会有对应的提示

  constructor(public http: HttpClient) {}

  ngOnInit(): void {
    // post请求的发送
    // 请求发送有两种方式 具体与服务器有关：此处的接口服务器做过处理 兼容post 和 get 两种方式

    let url = 'http://101.96.128.94:9999/data/product/list.php';
    let body = 'pno=1';

    // 有些post请求可能要设定header，默认有一个 content-type 可能需要设置
    let options = {
      headers: new HttpHeaders({
        // 因为默认值就是此设置，所以options 写不写没有影响：这里只是一个演示，如何设置
        'content-type': 'application/x-www-form-urlencoded',
      }),
    };

    // post请求主要指的是参数的传递方式不同。 post需要把 url?参数 的格式拆开
    // post的参数3: options 是可选的，具体写不写要看服务器对接口的设定！
    this.http.post(url, body, options).subscribe((res: Result) => {
      console.log(res);

      this.res = res;
    });
  }

  getData() {
    let url = 'http://101.96.128.94:9999/data/product/list.php?pno=1';
    this.http.get(url).subscribe((res: Result) => {
      console.log(res);
      // 声明属性来保存请求下来的返回值： 因为返回值res是局部变量，不能全局使用！
      this.res = res;
    });
  }
}

// 自定义返回值的数据类型：为了vscode在html中给代码提示！
// 类 和 interface 是同级别，不能包含！
// 类的封装：封装变量和函数
interface Data {
  // 程序员大法：复制 粘贴 改一改...
  is_onsale: string;
  lid: string;
  pic: string;
}

```

```

    price: string;
    sold_count: string;
    title: string;
  }

  interface Result {
    data: Data[];
    pageCount: number;
    pageSize: number;
    pno: number;
    recordCount: number;
  }

  /**
   * undefined 和 null 的差别?
   *
   * 此处两个场景:
   * 场景1: 然然在 23岁之前 没有过女朋友
   * 场景2: 然然在 24岁的时候 和女朋友分手了
   *
   * 请问 场景1 然然的女友 == undefined
   *       场景2 然然的女友 == null
   */

```

```

<p>myc01 works!</p>

<div *ngIf="res">
  <!-- Cannot read property 'data' of undefined -->
  <!-- 不能够 对 undefined 读取 属性 'data' -->
  <!-- 即 res 是 undefined, 未定义 -->
  <!-- 网络请求之后才给 res 赋值 -->
  <!-- 使用时应该先判断: res 有值再使用! -->
  <div *ngFor="let item of res.data">
    <img [src]='http://101.96.128.94:9999/' + item.pic" alt="" />
  <div>
    <span>{{ item.title }}</span>
    <span>{{ item.price }}</span>
  </div>
</div>
</div>

```

父子传参

组件: myc02

父组件.html

```
<app-myc02
  title="父子传参"
  name="亮亮"
  [age]="18"
  [hobby]="['吃饭', '睡觉']"
  [boss]="{ name: '文华', age: 33 }"
></app-myc02>
```

子组件.ts

```
import { Component, OnInit, Input } from '@angular/core';

@Component({
  selector: 'app-myc02',
  templateUrl: './myc02.component.html',
  styleUrls: ['./myc02.component.css'],
})
export class Myc02Component implements OnInit {
  // 固定写法：标注接收传递的属性
  // input: 输入
  @Input() title: string;
  @Input() name: string;
  @Input() age: number;
  @Input() hobby: string[];
  // any是偷懒写法，不会报错 但是也没代码提示!
  // @Input() boss: any;
  @Input() boss: Boss;

  constructor() {}

  ngOnInit(): void {}
}

// 自定义类型必须主动声明 才能有代码提示
interface Boss {
  name: string;
  age: number;
}
```

子父传参

组件: myc03

点击按钮时: 触发的代码过程



子.ts

```
import { Component, OnInit, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-myc03',
  templateUrl: './myc03.component.html',
  styleUrls: ['./myc03.component.css'],
})
export class Myc03Component implements OnInit {
  // output: 输出
  // EventEmitter(): 事件触发器; 用于保存外部传入的函数
  @Output() doMsg = new EventEmitter();

  constructor() {}

  ngOnInit(): void {}
}
```

```
<p>myc03 works!</p>

<button (click)="doMsg.emit('我是东东')">东东</button>
<button (click)="doMsg.emit('我是亮亮')">亮亮</button>
```

父组件

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
```

```

    title = 'ngpro';

    // 1.父准备一个方法，给子
    // msg是参数，是子元素传递的信息
    showMsg(msg) {
        console.log(msg);
    }
}

```

```

<!-- 2.传递方法给子：showMsg是函数,用事件方式传递 -->
<!-- $event：是固定写法 -->
<app-myc03 (doMsg)="showMsg($event)"></app-myc03>

```

掌控子元素

父元素拥有对子元素的控制权

组件: myc04

```

import { Component, ViewChild } from '@angular/core';
import { Myc04Component } from './myc04/myc04.component';

@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css'],
})
export class AppComponent {
    // @ViewChild('m4')：代表 查找到 #m4 的元素
    // mm4 是自定义变量，名字随意；会自动关联到查找到的元素
    @ViewChild('m4') mm4: Myc04Component;

    changeM4() {
        console.log(this.mm4);

        // 12行 书写了类型声明，此处才能有代码提示！
        this.mm4.name = '哈哈';
        this.mm4.show();
    }

    title = 'ngpro';

    // 1.父准备一个方法，给子
    // msg是参数，是子元素传递的信息
    showMsg(msg) {
        console.log(msg);
    }
}

```



```

<!-- <h2>Hello world!</h2> -->

<!-- 作业 -->
<!-- <app-work01></app-work01> -->

<!-- 练习: -->
<!-- <app-myc01></app-myc01> -->

<!-- 父子组件传参 -->
<!--
vue中: <HelloWorld title="xxx" />
组件中: props =['title'] 代表接收title属性
-->
<!-- <app-myc02
    title="父子传参"
    name="亮亮"
    [age]="18"
    [hobby]="['吃饭', '睡觉']"
    [boss]="{ name: '文华', age: 33 }"
></app-myc02> -->

<!-- 子父传参 -->
<!--
    在程序的世界中，等级非常严格；
    父元素拥有子元素的使用权，但是子元素不能与父元素直接交流！

    子若要传递信息给父，必须是 父提供一种途径，子通过此途径进行传递。
    所以 子父传参非两步：
    1. 父传递一个方法给子
    2. 子通过父给的方法 来传递值给父
-->

<!-- 2.传递方法给子：showMsg是函数,用事件方式传递 -->
<!-- $event: 是固定写法 -->
<!-- <app-myc03 (doMsg)="showMsg($event)"></app-myc03> -->

<!-- 掌控子元素：先设置唯一标识 -->
<!-- # 代替了 id -->
<app-myc04 #m4></app-myc04>
<app-myc04 #m3></app-myc04>

<button (click)="changeM4()">修改m4</button>

```

ionic

ionic相较于angular地位, 相当于 mint-ui 和 vue 的关系:

ionic就是一套手机端的UI组件库, 基于angular. 可以用来开发手机端App

官方: <https://ionicframework.com/>

- 安装ionic脚手架

```
npm i -g @ionic/cli
```

- 带 `-g` 说明是全局安装
- 需要npm是中国镜像 :参考day01文档
- 安装过程中的报错 :参考day01文档

```
+ @ionic/cli@6.11.0
added 226 packages from 156 contributors in 26.335s
```

查看ionic版本

```
ionic -v
```

• 生成项目包

- * 你的命令行需要在 你希望生成项目的目录下打开
- * `ionic start` 项目名
- * 例如: `ionic start ionicApp`
- * 生成过程中 除了下方图片中 需要选择**blank** 其它询问都 回车 使用默认值即可!

```
? Starter template: (Use arrow keys)
> tabs 下方带有 标签栏切换 的项目包
sidemenu 带有侧边栏目录的项目包
blank 空项目包 ← 选这个!!!!
list A blank starter project
my-first-app A starting project with a list
conference A starting project with a simple tabbed interface
A kitchen-sink application that shows off all Ionic has
Move up and down to reveal more choices)
```

生成完毕时

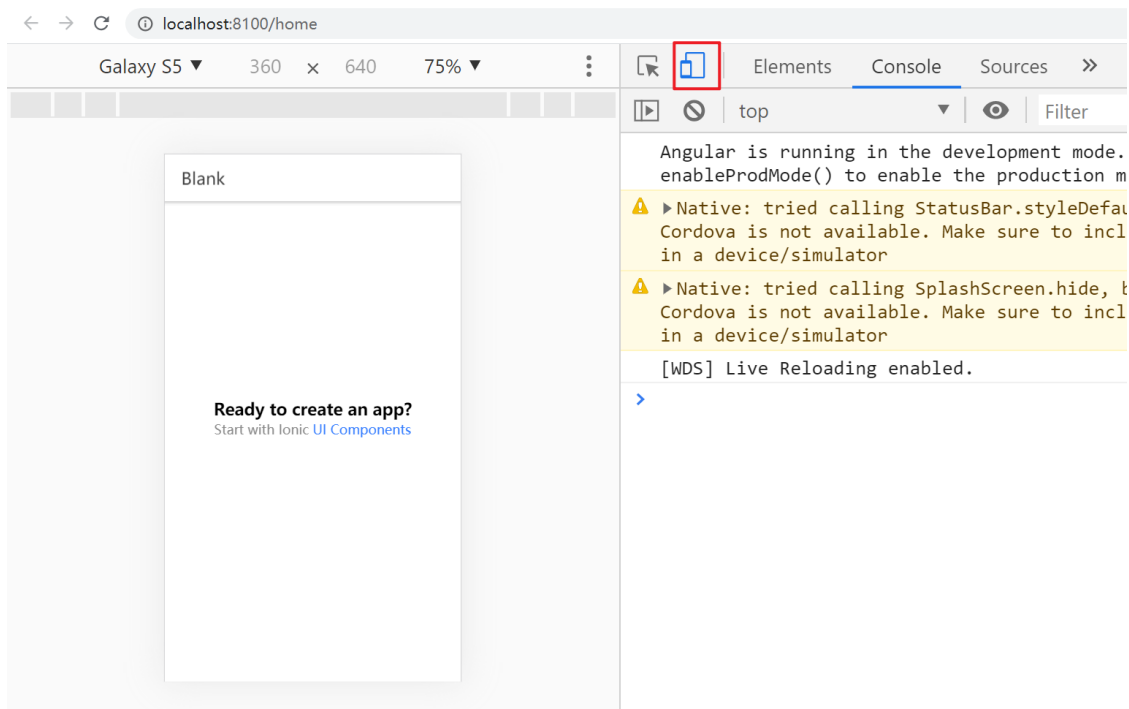
Your Ionic app is ready! Follow these next steps:

- Go to your new project: `cd .\ionicApp`
- Run `ionic serve` within the app directory to see your app in the browser
- Run `ionic capacitor add` to add a native iOS or Android project using Capacitor
- Generate your app icon and splash screens using `cordova-res --skip-config --copy`
- Explore the Ionic docs for components, tutorials, and more:
<https://ion.link/docs>
- Building an enterprise app? Ionic has Enterprise Support and Features:
<https://ion.link/enterprise-edition>

D:\WEB2003\Angular>

• 运行项目

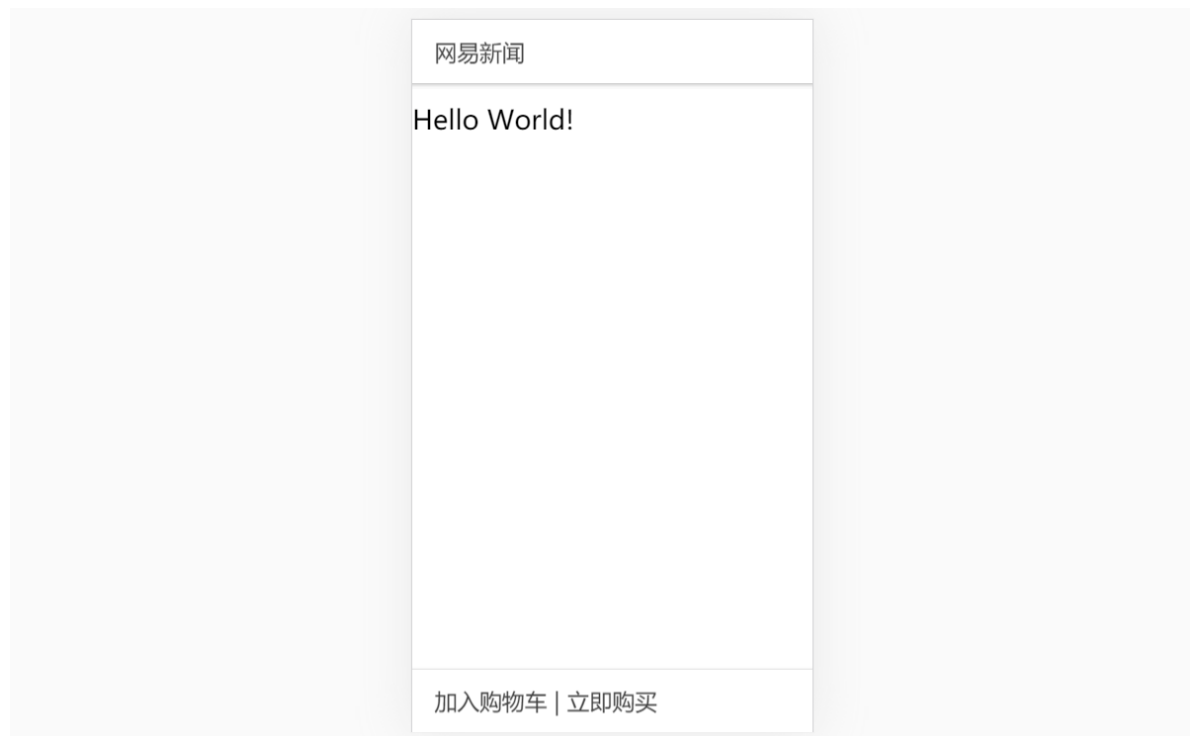
- * 必须到项目目录下执行
- * `ionic s`
- * 浏览器 `F12` 切换为手机模式进行查看



组件官方列表

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

基础容器结构



<!-- 插件的提示: i- 开头 -->

<!-- i-app -->

<ion-app>

<!-- i-header 头部栏-->

<ion-header>

```

<!-- toolbar : 工具栏 -->
<ion-toolbar>
  <!-- title: 标题 -->
  <ion-title>网易新闻</ion-title>
</ion-toolbar>
</ion-header>

<!-- i-content -->
<ion-content>
  <h2>Hello world!</h2>
</ion-content>

<!-- i-footer 脚部栏-->
<ion-footer>
  <ion-toolbar>
    <ion-title>加入购物车 | 立即购买</ion-title>
  </ion-toolbar>
</ion-footer>
</ion-app>

```

徽章组件



```

<!-- https://ionicframework.com/docs/api/badge -->
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>徽章组件 badge</ion-title>
    </ion-toolbar>
  </ion-header>
</ion-app>

<ion-content>
  <ion-badge>42</ion-badge>

  <!-- 强制样式: 样式默认会随设备自动变化, 但是可以强制 -->
  <ion-badge mode="ios">ios</ion-badge>
  <ion-badge mode="md">md</ion-badge>

  <!-- 颜色属性: 官方提供了固定的风格名, 具体参考 theme/variables.scss -->
  <ion-badge color="dark">dark</ion-badge>
  <ion-badge color="success">success</ion-badge>
  <ion-badge color="warning">warning</ion-badge>
  <ion-badge color="danger">danger</ion-badge>
  <ion-badge color="primary">primary</ion-badge>
</ion-content>

```

按钮



```
<!-- https://ionicframework.com/docs/api/button -->
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>按钮</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <h2>size: 大小</h2>
    <ion-button size="small">small</ion-button>
    <ion-button size="default">default</ion-button>
    <ion-button size="large" mode="ios" color="success">large</ion-button>

    <h2>fill: 背景填充</h2>
    <ion-button fill="solid">solid</ion-button>
    <ion-button fill="outline">outline</ion-button>
    <ion-button fill="clear">clear</ion-button>

    <h2>expand: 按钮的扩展</h2>
    <ion-button expand="full">full</ion-button>
    <ion-button expand="block">block</ion-button>
  </ion-content>
</ion-app>
```

icon

图标组件, 官方默认提供了很多常用的小图标, 可以直接使用

<https://ionicons.com/>

icon小图标



```
<!-- https://ionicons.com/ -->
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>icon小图标</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <!-- 图标属于字体，可以利用css来进行调整 -->
    <ion-icon
      name="checkbox-outline"
      style="font-size: 30px; color: aqua;"
    ></ion-icon>

    <!-- 找到以下图标并显示：齿轮(设置) 实心心 空心心 房子 人 星星 -->
    <ion-icon name="star-outline"></ion-icon>
    <ion-icon name="star"></ion-icon>
    <ion-icon name="settings-outline"></ion-icon>
    <ion-icon name="heart-outline"></ion-icon>
    <ion-icon name="heart"></ion-icon>

    <ion-icon name="logo-android"></ion-icon>
  </ion-content>
</ion-app>
```

card卡片组件

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

card卡片



小樱, 佐助, 鸣人....

火影忍者: 第七班

十多年前一只恐怖的尾兽“九尾妖狐”袭击了木叶隐村，当时的第四代火影拼尽全力，以自己的生命为代价将“九尾妖狐”封印在了刚出生的鸣人身

```

<!-- https://ionicframework.com/docs/api/card -->
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>card卡片</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <!-- i-card-full -->
    <ion-card>
      <!-- img图片组件 -->
      <ion-img
        src="https://tse4-mm.cn.bing.net/th/id/OIP.Ax06xu9hqFx81tnv_htP9QHaEo?
pid=Api&rs=1"
      ></ion-img>
      <ion-card-header>
        <!-- subtitle 子标题 -->
        <ion-card-subtitle>小樱，佐助，鸣人....</ion-card-subtitle>
        <!-- title: 标题 -->
        <ion-card-title>火影忍者：第七班</ion-card-title>
      </ion-card-header>
      <ion-card-content>
        十多年前一只恐怖的尾兽“九尾妖狐”袭击了木叶隐村，当时的第四代火影拼尽全力，以自己的生命
        为代价将“九尾妖狐”封印在了刚出生的鸣人身
      </ion-card-content>
    </ion-card>
  </ion-content>
</ion-app>

```

item组件

item

亮亮



Pizza

Pizza



Pizza



成亮老师

明天我们将吃饭, 睡觉, 打然然!

99+

具有点击效果



A

阿呆

阿花



缩略图

```
<!-- https://ionicframework.com/docs/api/item -->
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>item</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <ion-item>
      <ion-label>亮亮</ion-label>
    </ion-item>

    <ion-item>
      <!-- slot: 插槽 -->
      <ion-icon name="pizza" slot="start"></ion-icon>
      <ion-label>Pizza</ion-label>
    </ion-item>

    <ion-item>
      <!-- slot: 插槽 -->
      <ion-icon name="pizza" slot="end"></ion-icon>
      <ion-label>Pizza</ion-label>
    </ion-item>

    <ion-item>
```



```

<!-- slot: 插槽 -->
<ion-icon name="pizza"></ion-icon>
<ion-label>Pizza</ion-label>
</ion-item>

<!-- avatar: 头像 -->
<ion-item>
  <!-- avatar: 自动把内容图片变为圆形 -->
  <ion-avatar slot="start">
    
  </ion-avatar>
  <ion-label>
    <h2>成亮老师</h2>
    <p>明天我们将吃饭，睡觉，打然然!</p>
  </ion-label>
  <ion-badge color="danger" slot="end">99+</ion-badge>
</ion-item>

<!-- button属性: 点击效果! -->
<!-- detail: 右边的 右箭头 -->
<ion-item button detail>
  <ion-label>具有点击效果</ion-label>
</ion-item>

<!-- 通讯录效果 -->
<!-- i-item-group -->
<ion-item-group>
  <ion-item-divider>
    <ion-label>A</ion-label>
  </ion-item-divider>

  <ion-item>
    <ion-label>阿杲</ion-label>
  </ion-item>
  <ion-item>
    <ion-label>阿花</ion-label>
  </ion-item>
</ion-item-group>

<!-- 缩略图 -->
<ion-item>
  <!-- thumbnail: 缩略图 -->
  <ion-thumbnail slot="start">
    
  </ion-thumbnail>
  <ion-label>缩略图</ion-label>
</ion-item>
</ion-content>
</ion-app>

```

输入框

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

输入框
Awesome Input
用户名:
密码:
E-Mail

```
<!-- https://ionicframework.com/docs/api/input -->
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>输入框</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <ion-input type="text" placeholder="Awesome Input"></ion-input>

    <!-- 输入框默认没有边框，通常配合 item 来使用 -->
    <ion-item>
      <ion-label>用户名:</ion-label>
      <ion-input></ion-input>
    </ion-item>

    <ion-item>
      <ion-label>密码:</ion-label>
      <ion-input type="password"></ion-input>
    </ion-item>

    <!-- -->
    <ion-item>
      <!-- position="floating" 浮动效果 -->
      <ion-label position="floating">E-Mail</ion-label>
      <ion-input type="text"></ion-input>
    </ion-item>
  </ion-content>
</ion-app>
```

搜索栏

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

搜索栏

🔍 请输入商品名 或 商家名

```
<!-- https://ionicframework.com/docs/api/searchbar -->
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>搜索栏</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <!-- i-searchbar -->
    <ion-searchbar
      placeholder="请输入商品名 或 商家名"
      (ionChange)="onSearchChange($event)"
    ></ion-searchbar>
  </ion-content>
</ion-app>
```

```
import { Component } from "@angular/core";

import { Platform } from "@ionic/angular";
import { SplashScreen } from "@ionic-native/splash-screen/ngx";
import { StatusBar } from "@ionic-native/status-bar/ngx";

@Component({
  selector: "app-root",
  templateUrl: "app.component.html",
  styleUrls: ["app.component.scss"],
})
export class AppComponent {
  constructor(
    private platform: Platform,
    private splashScreen: SplashScreen,
    private statusBar: StatusBar
  ) {
    this.initializeApp();
  }

  initializeApp() {
    this.platform.ready().then(() => {
      this.statusBar.styleDefault();
      this.splashScreen.hide();
    });
  }
}
```

```


}

onSearchChange(msg) {
  // console.log(msg);
  let val = msg.detail.value;
  console.log("搜索的内容:" + val);

  // 实际项目:则发送网络请求 获取对应的数据进行显示!
}
}

```

ionic插件



Ionic Snippets 2.2.2
 Ionic snippets for VS Code
 fivethree

129K ★ 4.5
 ✓ Enabled

作业

作业1: 新闻列表, 带有分页操作

ng项目, 非ionic

接口地址: http://101.96.128.94:9999/mfresh/data/news_select.php?pageNum=1

参数: pageNum 代表页数, 1就是第一页

请求方式: GET

难点:

页数 1 2 3 4 的显示;
 返回值有一个 pageCount:4
 * ng的for循环不支持循环数字, 只能循环数组; 所以要制作一个函数 把 4 转为 [1,2,3,4]
 * 例如: range(4); 就可以返回 [1,2,3,4] ; 然后循环此数组即可

▶ 1空气净化器要逆天? “玫瑰金” “土豪金” 齐上阵	2016-10-8
▶ 2净美仕新风净化系统 助力校园新风行动	2016-10-8
▶ 3全国新风行动全面启动 助推净美仕战略升级	2016-10-8
▶ 4智能空气净化器翻盘: 净美仕能否领衔?	2016-10-8
▶ 5空气净化器要逆天? “玫瑰金” “土豪金” 齐上阵	2016-10-8
▶ 6净美仕新风净化系统 助力校园新风行动	2016-10-8

选做

ionic项目

接口: <https://api.apiopen.top/getImages>

发送请求 并把图片 循环显示到页面上: 使用 `ion-card` 组件

提示:

```
initializeApp() {  
  this.platform.ready().then(() => {  
    this.statusBar.styleDefault();  
    this.splashScreen.hide();  
    这里发请求  
  });  
}
```