

Angular04

作业1: 轮播图

```
ng g c work01
```

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

@Component({
  selector: 'app-work01',
  templateUrl: './work01.component.html',
  styleUrls: ['./work01.component.css'],
})
export class work01Component implements OnInit {
  images = ['111.jpg', '222.jpg', '333.jpg', '444.jpg'];

  curP = 0; //当前页

  nextP() {
    this.curP++;
  }

  prevP() {
    this.curP--;
  }

  constructor() {}

  // 添加定时器都可以
  ngOnInit(): void {
    //组件开始初始化
    setInterval(() => {
      this.curP++;

      if (this.curP == this.images.length) this.curP = 0;
    }, 2500);
  }

  ngAfterViewInit(): void {
    //组件初始化完毕
  }
}
```

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

<div>
  <div class="box">
    
    <div class="points">
      <span
```

```

        [ngClass]="{ 'point-cur': curP == i }"
        class="point"
        *ngFor="let item of images; let i = index"
        (click)="curP = i"
    ></span>
</div>
</div>
<div class="pages">
    <span class="page" (click)="prevP()" *ngIf="curP > 0">上一页</span>
    <span class="page-disabled" *ngIf="curP <= 0">上一页</span>
    <!-- 动态样式: [ngClass]="{样式名: true/false}" -->
    <span
        class="page"
        [ngClass]="{ cur: curP == i }"
        *ngFor="let item of images; let i = index"
        (click)="curP = i"
        >{{ i + 1 }}</span>
    >
    <!-- 当前页 < 最大序号时, 可以点 -->
    <span class="page" (click)="nextP()" *ngIf="curP < images.length - 1"
        >下一页</span>
    >
    <!-- 当前页>=最大序号 则不能点 -->
    <span class="page-disabled" *ngIf="curP >= images.length - 1">下一页</span>
</div>
</div>

```

作业2: 代办事项

ng g c work02

```

<p>work02 works!</p>

<div>
    <input type="text" placeholder="请输入待办事项" [(ngModel)]="todo" />
    <button (click)="doAdd()" [disabled]="todo.trim() == ''">确定</button>
</div>
<ul>
    <li *ngFor="let item of items; let i = index">
        <span>{{ i + 1 }}. </span>
        <span>{{ item }}</span>
        <!--
            数组元素的删除: arr.splice(i, n[, args])
            代表删除数组的序号i开始 n个元素。 把参数3添加到i的位置
        -->
        <button (click)="items.splice(i, 1)">删除</button>
    </li>
</ul>

<h4 *ngIf="items.length == 0">暂无待办事项</h4>

```

```

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

@Component({
  selector: 'app-work02',
  templateUrl: './work02.component.html',
  styleUrls: ['./work02.component.css'],
})
export class Work02Component implements OnInit {
  items = ['吃饭', '睡觉', '打亮亮'];

  todo = '';

  doAdd() {
    this.items.push(this.todo);

    // 双向绑定:
    // 方向1: 属性传递到页面, 属性变页面变
    // 方向2: 修改页面时, 同步影响属性
    this.todo = '';
  }

  constructor() {}

  ngOnInit(): void {}
}

```

TypeScript

TypeScript是 JavaScript的超集: 具备了JS的所有功能 并 新增了更多的特性, 由 **微软公司** 开发并进行维护.

浏览器默认只能执行 HTML, CSS, JS; 浏览器不认识 TypeScript 语言, 只能通过翻译工具(编译软件)把TS 编译成 JS 之后 再交给浏览器.

ES6语法 就无法被低版本的浏览器识别, 所以利用了 babel 工具进行编译, 转为兼容的JS代码 再传递给浏览器

安装编译工具

```
npm i -g typescript
```

```
+ typescript@4.0.3
updated 1 package in 3.371s
```

```
C:\Users\web>
```

如果安装出现报错, 可能是已经安装过. 具体参考Day01 笔记中的解决办法

通过 `tsc -v` 可以检查是否安装

```
C:\Users\web>tsc -v
Version 4.0.3
```

```
// 01.ts
```

```
// TypeScript 相较于 JavaScript 最明显的特征：静态类型分析
// 静态类型分析：在大多数语言中都存在 Java, Php, C, Python...
// 代码中明文设定类型名，编程工具就可以在未运行的情况下 判断出代码中的错误

// 变量名:类型名
// 告诉编程工具，即 vscode，name是字符串类型
function show(name: string) {
    name.splice(1, 1);
}

// 静态类型分析：代码未运行的情况下 就有报错提示
```

```
// 数据类型有哪些

class Emp {
    a = 6; //面向对象的赋值 对象写法{a:6}

    name: string; //字符串

    age: number; //数字类型

    married: boolean; //布尔类型

    //任意类型
    abc: any;

    // 指定，|代表或
    xyz: number | string;

    // 数组类型：必须声明数组中存放的值 是什么类型的
    names: Array<string>; //Array<元素类型名>

    // 语法糖写法：
    nums: number[]; //等价于 Array<number>

    show() {
        this.nums = [123, 2, 3, 54, 56];

        this.names = ["lily", "lucy", "123"];
        // this.names.push(123);

        // 赋值类型不匹配，则会有报错提示！
        // this.name = 123;

        // 任意类型 可以赋值 任意类型的值
        this.abc = 123;
        this.abc = true;

        this.xyz = "dongdong";
        this.xyz = 33;
        // this.xyz = true;

        console.log("我是show方法");
    }
}
```

```
}

let e = new Emp();
e.show();

// ts文件的执行： 不能直接执行，需要转化成 JS 文件之后再执行
// 编译命令： tsc xxx.ts
// 就会把 ts 文件 转化成 js 文件

// tsc：就是按照一定的规则 把ts中的语法 转为兼容性的 JS语法。
```

```
// 自定义对象类型

// 假设有类型
let boss = {
  name: "文华",
  age: 28,
  sex: "男",
};

// 希望其他人 按照上方的格式生成更多的对象类型
let ranran: Emp = {
  name: "然然",
  age: 33,
  sex: "男",
};

// 制定自定义的对象类型结构
// 新的关键词 interface 和 function class 相同的，都是关键词
// function函数 class类 interface类型
interface Emp {
  name: string;
  age: number;
  sex: string;
}

// 新的类型：女婿
interface NvXu {
  che: string;
  fang: string;
  money: number;
}

// 思源
let siyuan: NvXu = {
  che: "娃娃车",
  fang: "17平米大平房",
  money: 500,
};
```

```

/**
 * 访问控制词：在 java c php python... 都存在， js没有!!!
 *
 * public    公开的
 * protected 保护的
 * private   私有的
 */

class Demo {
  public name = "思源"; //公开的
  protected money = "思源的存款：500元"; //保护：对外
  private avi = "思源珍藏的1T 种子"; //私有的，只能自己看

  show() {
    // 自身可以 阅读使用所有权限
    this.name;
    this.money;
    this.avi;
  }
}

// 继承 extends
class Son extends Demo {
  // 子类
  show() {
    this.name;
    this.money;
    this.avi; //子类无法访问私有属性！
  }
}

// 类外访问 Demo类的属性
Demo.name; //类外可以访问 公开属性

// 类外无法访问私有和保护
// Demo.money;
// Demo.avi;

```

父子传参

vue中的父子传参

子组件 `<Son name="然然" age="18" />`

在 Son 组件中，必须声明接收参数的属性

```
props: ['name', 'age']
```

angular中

新的组件： `ng g c myc01`

```

<!-- 父子传参 -->
<!-- 通过属性方式传递参数 -->
<app-myc01
  name="然然"
  [age]="18"
  [boss]="{ name: '文华', age: 33, phone: '10086' }"
></app-myc01>

```

子接收

```

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

@Component({
  selector: 'app-myc01',
  templateUrl: './myc01.component.html',
  styleUrls: ['./myc01.component.css'],
})
export class Myc01Component implements OnInit {
  // 声明接收通过属性传入的值
  @Input() name: string;
  // input: 输入
  // @Input() 代表此变量是外部输入的

  @Input() age: number;

  // object 代表 {} 空对象
  @Input() boss: Boss;

  constructor() {}

  ngOnInit(): void {}
}

// 对象类型需要自己声明
// 类型名 是 大写
interface Boss {
  name: string;
  age: number;
  phone: string;
}

```

子父传参

定时器练习:

制作一个 定时器组件

```
<app-clock [duration]="30" />
```

页面上就会出现 30 ，然后开始倒数 直到 0
当到0时，弹出提示框：倒计时结束

父ts

```
export class AppComponent {  
  title = 'ng-app';  
  
  // 方法，会交给子，让子通过此方法来联系  
  // 倒计时完毕  
  clockComplete(msg) {  
    console.log(msg);  
  }  
}
```

父html

```
<!-- 父子传参 -->  
<!-- 传递方法给子，方法传递用() 属性是[] -->  
<app-clock [duration]="30" (msgEvent)="clockComplete($event)"></app-clock>  
<!-- $event: 固定的 -->
```

子ts

```
import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';  
  
@Component({  
  selector: 'app-clock',  
  templateUrl: './clock.component.html',  
  styleUrls: ['./clock.component.css'],  
})  
export class ClockComponent implements OnInit {  
  // 声明专门接收外部传入函数的 属性  
  // output: 向外输出  
  @Output() msgEvent = new EventEmitter();  
  // 固定写法: @Output() xxx = new EventEmitter();  
  
  @Input()  
  duration: number;  
  
  timer: any;  
  
  doStart() {  
    if (this.timer) return;  
  
    this.timer = setInterval(() => {  
      this.duration--;  
    }, 1000);  
  }  
}
```



```

    if (this.duration == 0) {
        //关闭定时器
        clearInterval(this.timer);

        // 通过 msgEvent中 存储的 父组件传入的方法 来传递消息
        this.msgEvent.emit('倒计时结束!');
    }
}, 1000);
}

constructor() {}

ngOnInit(): void {}
}

/**
 * 子父传参
 *
 * 父组件 安排 子组件做某个任务， 子组件完毕时， 要通知父组件。
 *
 * 子通知父： 必须依赖父传入的方法
 *
 * 实际的生活例子：
 *
 * 老师给了思源一个手机号， 和思源说： 面试完给我打电话
 *
 * 代码中：
 * 父制作一个方法， 把方法通过子的属性传递给子， 子中调用属性即可触发父传入的方法
 */

```

掌控子元素

父元素 可以通过绑定变量 给子元素, 然后就可以利用变量来动态控制子元素的内容.

```
ng g c myc02
```

```

import { Component, ViewChild } from '@angular/core';
import { Myc02Component } from './myc02/myc02.component';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  @ViewChild('abc') abc: Myc02Component;
  // 固定写法: @ViewChild('id') xxx;
  // 查询到 #id 的组件 关联到 xxx 变量上

  nextYear() {
    console.log(this.abc);
    this.abc.age++;
    this.abc.show();
  }
}

```

```

title = 'ng-app';

// 方法，会交给子，让子通过此方法来联系
// 倒计时完毕
clockComplete(msg) {
  console.log(msg);
}
}

```

```

<!-- 掌控子元素 -->
<!-- #相当于id，所以#abc 相当于id='adc' -->
<!-- 代表唯一标识 -->
<app-myc02 #abc></app-myc02>

<button (click)="nextYear()">过了一年</button>

```

子

```

export class Myc02Component implements OnInit {
  name = '东东';
  age = 18;

  show() {
    alert('我是 myc02 的show');
  }

  constructor() {}

  ngOnInit(): void {}
}

```

服务

在vue中的技术名称: Vuex

作用: 进行全局状态管理. 主要在多个组件间共享数据和方法.

```

生成组件:
ng g c myc03
ng g c myc04

```

生成服务命令

ng generate service 服务名

简写: ng g s 服务名

例如: ng g s name

```
import { Component, OnInit } from '@angular/core';
import { NameService } from '../name.service';

@Component({
  selector: 'app-myc03',
  templateUrl: './myc03.component.html',
  styleUrls: ['./myc03.component.css'],
})
export class Myc03Component implements OnInit {
  // 通过属性保存服务到本地
  names: NameService; //变量名随意写，但是最好有含义

  // 服务：是系统提供的；我们使用时只需要声明 依赖即可！
  // 变量名随意写，但是最好有含义
  constructor(names: NameService) {
    // names 局部变量：只能在当前{}中用。 要想全局使用则必须保存到属性
    this.names = names;
  }

  ngOnInit(): void {
    console.log(this.names.names);
  }
}

/**
 * 依赖注入机制：
 *
 * 实际生活中此机制广泛使用：
 * 1. 思源说：提供代写项目服务，只需要给我介绍女朋友
 *    某某某：经过精心打扮之后 ...得到了项目。
 *
 * 2. 超市门口的摇摇椅：1元/次 --声明依赖
 *    注入：投币，可以玩..
 *
 * 3. 代码中同样存在
 */
class Demo {
  // 构造方法：就是初始化时调用的方法 new Demo()
  // 声明参数：要想初始化我，必须传递 字符串类型的值 --- 声明依赖
  constructor(name: string) {}
}

//注入
new Demo('东东');
```

服务

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

@Injectable({
  providedIn: 'root',
})
export class NameService {
  // 在这里写的数据 和 方法 就可以在组件间通用
  names = ['东东', '然然', '亮亮', '小新', '华哥'];
  // 这里就类似于 vuex 的state

  constructor() {}
}
```

语法糖

```
import { Component, OnInit } from '@angular/core';
import { NameService } from '../name.service';

@Component({
  selector: 'app-myc04',
  templateUrl: './myc04.component.html',
  styleUrls: ['./myc04.component.css'],
})
export class Myc04Component implements OnInit {
  // ns: NameService;

  // 服务遵循依赖注入机制： 我们只需要在构造方法中声明 组件的出生依赖什么
  // constructor(ns: NameService) {
  //   this.ns = ns;
  // }

  // 语法糖写法： 简短 但是不易理解
  // 固定格式： 权限词 变量名:服务的类型
  // 权限词： public private protected 此处随意， 没有差别
  constructor(protected ns: NameService) {}

  ngOnInit(): void {}
}

/**
 * 语法糖： 利用一些简化写法 简化代码 --- 懒
 *
 * 1. if (条件) { xxx } 如果{}只有一行， if (条件) xxx;
 * 2. ()=>{ return xxx; } 则 ()=> xxx
 * 3. (name)=>{} 只有一个参数 name => {}
 * 4. let obj = {name:'', age:18}; let name=obj.name; let age=obj.age;
 *    let {name, age} = obj;
 *
 */
```

```

<p>myc04 works!</p>

<ul>
  <li *ngFor="let item of ns.names">{{ item }}</li>
</ul>

<!-- 04中修改数组，03组件也会变化 -->
<!-- 说明是共享数据 -->
<button (click)="ns.names.push('燕子')">添加</button>

```

网络服务

官方提供了很多服务, 其中网络服务非常常用!

vue中使用第三方的axios, angular自带网络服务, 不需要axios

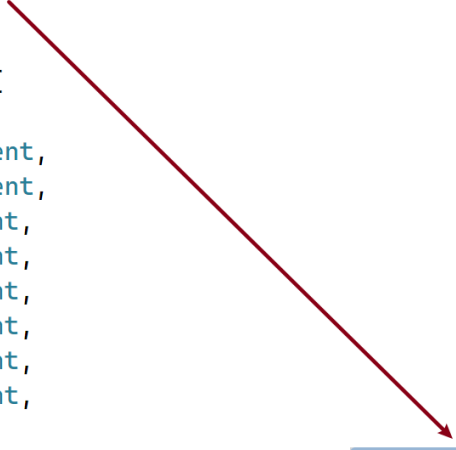
```
ng g c myc05
```

```

// 注意代码的书写技巧, 善用代码提示!
// imp -> @co -> /h
// hm
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    Work01Component,
    Work02Component,
    Myc01Component,
    ClockComponent,
    Myc02Component,
    Myc03Component,
    Myc04Component,
    Myc05Component,
  ],
  imports: [BrowserModule, FormsModule, HttpClientModule],
})

```



```

<p>myc05 works!</p>

<!-- 判断: res 有值 再使用! -->
<div style="width: 700px; margin: 0 auto" *ngIf="res">
  <!--
    res: 通过网络请求获取的值
    网络请求是异步的: 先发 -> 获取值, 在没有获取值之前, res就是undefined 所以使用就报错!
  -->
  <img
    *ngFor="let item of res.result"
    [src]="item.img"
    alt=""
    style="width: 150px; height: 200px; border-radius: 3px; margin: 5px"
  />
</div>

```

<!--

遗留问题：后台各种爆红，html中代码没有提示 --- 下节课解决

Cannot read property 'result' of undefined

不能对 undefined 读取result 属性

一些404 报错，是因为免费接口 内容不全； --- 忍着

-->

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

@Component({
  selector: 'app-myc05',
  templateUrl: './myc05.component.html',
  styleUrls: ['./myc05.component.css'],
})
export class Myc05Component implements OnInit {
  /**
   * 网络服务 与 Form表单一样：默认模块不加载。必须手动加载
   * 到 app.module.ts 中加载 网络模块 并重启服务器！
   */

  // 告诉系统：此处需要网络服务
  constructor(public http: HttpClient) {}

  res: Result;

  ngOnInit(): void {
    // 相当于mounted 周期：挂载时， 通常发请求
    // axios: axios.get(url).then(res=>{})
    // http: this.http.get(url).subscribe(res=>{})
    // 变化：把 then(然后) 换成了 subscribe(订阅)
    let url = 'https://api.apiopen.top/getImages';
    this.http.get(url).subscribe((res: Result) => {
      console.log(res);
      // 属性 = 局部变量； 属性才能全局使用，局部变量只能在当前{}中用
      this.res = res;
    });
  }
}

// 返回值是对象类型，使用时出现代码提示，则必须声明类型
interface Result {
  code: number;
  message: string;
  result: ResultData[]; //数组类型：其中的元素都是 ResultData 类型
}

interface ResultData {
  id: number;
  img: string;
  time: string;
}
```

ionic

第五阶段重点: App开发

ionic框架 就是基于Angular, 然后增加了一套 手机端样式的组件.

官方网站: <https://ionicframework.com/>

Installation Guide

01

Install Ionic

Install [Node.js](#), then install the latest Ionic command-line tools in your terminal. Follow the [Android](#) and [iOS](#) platform guides to install their required tools.

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

New to command line? Read our [Terminal tutorial](#) →

02

Start an app

Create an app using one of our ready-made app templates, or a blank one to start fresh. Check out the [Market](#) for more designs.

```
ionic start myApp tabs
```

03

Run your app

Much of your app can be built in the browser with **ionic serve**. We suggest starting with this workflow. When you're ready, check out our [Deploying](#) guide.

```
cd myApp
ionic serve
```

安装脚手架:

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

* 安装脚手架是为了生成项目包, 如果无法安装则可以使用别人生成的包--- 重装系统可以解决无法安装的问题.

```
+ @ionic/cli@6.11.9
added 5 packages from 5 contributors, removed 13 packages and updated 29 packages in 47s
```

使用: `ionic -v` 可以查看版本

```
C:\Users\web>ionic -v
6.11.9
```

生成包

包会生成在 命令行执行的目录下

```
ionic start 包名 blank
```

例如: `ionic start ionicApp blank`

安装过程的所有询问, 都直接回车

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>

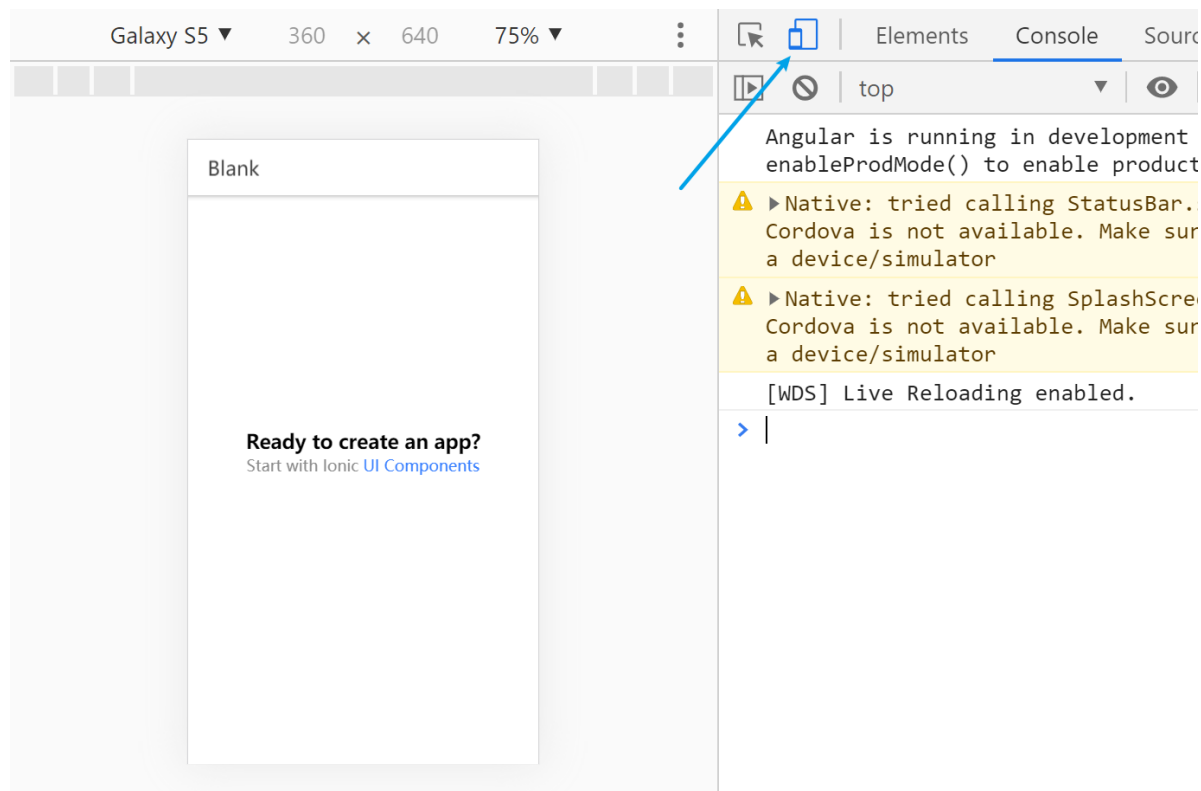
```
D:\WEBTN2005\18_Angular>
```

启动项目

在项目下执行:

```
ng s -o 可以 -- 端口号是4200
ionic s 可以 -- 端口号8100
实际效果没有任何差异
```

脚手架无法安装的, 命令前要加 `npx`



作业

接口地址:

```
http://101.96.128.94:9999/mfresh/data/news_select.php?pageNum=1
```

- * 参数: `pageNum` 代表页数
- * 返回值中的: `pageCount` 是总页数

难点提示: 中间的 1 2 3 4 页 的生成问题

- vue的循环可以直接循环数字, 但是angular不可以. angular只能循环数组

必须制作一个函数, 能够把数字转为数组, 例如 `range(4)` 可以得到 `[1,2,3,4]`

▶ 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