Angular02

复习

前端三大框架：

- angular
  - Google
  - 手机端App 和 网页Web
  - 适用于 大型项目
- vue
  - 开源社区
  - 手机端App 和 网页Web
  - 中小型项目
- react
  - Facebook
  - 手机端App 和 网页Web
  - 中小型项目

> 面试相关：
>
> Vue占比：50%； 需求量大,会的人多. 竞争非常强.. 面试难度高!
>
> React占比：30%; 需求量略大, 会的人少. 面试难度低, 比vue容易入职
>
> Angualr占比：20%; 需求量略大, 会的人少. 面试难度低, 比vue容易入职

- 脚手架

  与 vue 相同, 需要使用脚手架生成项目包

  - 安装命令

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

  - 生成命令

    ```
    ng new 包名
    简写：ng n 包名
    过程中的选项, 都回车即可
    ```

  - 启动命令

    ```
    ng s -o

    全称：ng serve --open
    ```

  - 生成组件

    ```
    ng generate component 组件名
    ng g c 组件名
    ```

- 语法

- `{{}}` ：同vue的． 用来展示 JS 中的变量．其中可以进行 数学运算，逻辑允许，比较运算，三目运算符．
- 属性： `[属性名]="值"`
- 事件： `(事件名)="值"`
- html： `[innerHTML]=""`
- 双向数据绑定： `[(ngModel)]=""`
  - 默认不可用，必须到配置文件中手动加载 **FormsModule** 才可以
- 动态样式：
  - **[ngStyle]="样式名：值"**
  - **[ngClass]="样式类名：true/false"** ：通过true和false切换是否可用

指令

条件渲染

> 在 vue 中， `v-if`

创建组件： `ng g c myc01`

```
<p>myc01 works!</p>

<div>
  <button (click)="married = true">结婚</button>
  <button (click)="married = false">单身</button>
</div>
<!-- 利用 if 判断，根据条件显示不同的UI -->
<!-- 在 vue 中  v-if -->
<!-- 在 ng 中：*ngIf -->
<p *ngIf="married">已婚</p>
<p *ngIf="!married">未婚</p>

<!--
  在 vue 中，有 v-if 和 v-else
  在 ng 中，也有 if 和 else，  不过因为格式较为麻烦，通常不用
 -->
<!-- ng-if-else -->
<ng-container *ngIf="married; else abc">
  <p>已婚</p>
</ng-container>
<!-- 语法糖写法：#xxx  类似于 id="xxx";  用于声明唯一标识 -->
<ng-template #abc>
  <p>未婚</p>
</ng-template>

```

列表渲染

> 在 vue 中: v-for="(item, index) in items" :key="index"

组件： `ng g c myc02`

```
<p>myc02 works!</p>

<!--
  vue中:
  <li v-for="(item,index) in items" :key="index">

  v-for="item in items"
 -->

<ul>
  <!-- ng-for -->
  <!-- angular中不需要 key -->
  <li *ngFor="let item of names">
    <span>{{ item }}</span>
  </li>
</ul>

<ul>
  <!-- ng-for-index -->
  <li *ngFor="let item of names; let i = index">
    <span>{{ i }}. </span>
    <span>{{ item }}</span>
  </li>
</ul>

<table border="1">
  <tr>
    <td>序号</td>
    <td>姓名</td>
    <td>年龄</td>
    <td>手机号</td>
  </tr>
  <tr *ngFor="let item of emps; let i = index">
    <td>{{ i + 1 }}</td>
    <td>{{ item.name }}</td>
    <td>{{ item.age }}</td>
    <td>{{ item.phone }}</td>
  </tr>
</table>

<!--
  vue 循环, 支持遍历数字
  v-for="item in 4"    则 item 的值是 1 2 3 4
 -->
<ul>
  <!-- angular不支持数字的循环遍历, 只支持 数组 -->
  <li *ngFor="let item of range(4)">{{ item }}</li>
</ul>
```

```
49
```

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

@Component({
  selector: 'app-myc02',
  templateUrl: './myc02.component.html',
  styleUrls: ['./myc02.component.css'],
})
export class Myc02Component implements OnInit {
  names = ['html5', 'axios', 'ajax', 'jsonp', 'cors', 'proxy'];

  emps = [
    { name: '东东', age: 33, phone: '13877889988' },
    { name: '然然', age: 34, phone: '13877779988' },
    { name: '亮亮', age: 29, phone: '13877239988' },
    { name: '小新', age: 28, phone: '13877549988' },
  ];

  // 自制方法: 4 -> [1,2,3,4]
  range(num) {
    let arr = [];

    for (let i = 1; i <= num; i++) {
      arr.push(i);
    }

    return arr;
  }

  constructor() {}

  ngOnInit(): void {}
}

```

管道 pipe

> 在 vue 中, 称为 过滤器 filter
>
> 用法: {{ 值 | 过滤器 }}

创建组件: `ng g c myc03`

官方提供的管道

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

<!--
  与 vue 不同: vue 必须自定义过滤器之后 才能使用.

```

```
  angular 官方贴心的 提供了一些 常用的管道，可以直接使用
  当然 也可以像 vue 一样自定义
 -->
<ul>
  <li>全大写：{{ "nice to meet you" | uppercase }}</li>
  <li>全小写：{{ "HELLO WORLD!" | lowercase }}</li>
  <li>首字母大写：{{ "nice to meet you" | titlecase }}</li>
  <li></li>
  <li>百分数：{{ 0.55555 | percent }}</li>
  <!-- {{ 值 | percent: 'n.m' }}  最少n位整数，不足补0.  小数m位,不足补0 -->
  <li>百分数：{{ 0.55555 | percent: "2.2" }}</li>
  <li>百分数：{{ 0.055 | percent: "2.2" }}</li>
  <li></li>
  <li>钱：{{ 123456.789 | currency }}</li>
  <li>钱：{{ 123456.789 | currency: "¥" }}</li>
  <li></li>
  <!-- 时间戳的转换：当前时间距离 1970.1.1 的秒数 -->
  <!-- 此处要求单位：毫秒  1秒 = 1000毫秒 -->
  <li>日期：{{ 1609126768000 | date }}</li>
  <!--
    year   年   y
    month  月   M
    day    日   d
    hour   小时 h12 H24
    minute 分钟 m
    second 秒   s
   -->
  <!-- mm：保证两位，不足补0 -->
  <li>日期：{{ 160912625000 | date: "yyyy-MM-dd HH:mm:ss" }}</li>
  <li>日期：{{ 160912625000 | date: "y-M-d H:m:s" }}</li>
</ul>

```

自定义管道

生成组件：`ng g c my04`

管道生成命令：

```
ng generate pipe 管道名

简写：ng g p 管道名
```

```
D:\webtn2008\04_Angular\Day02\ngpro>ng g p gender
CREATE src/app/gender.pipe.spec.ts (187 bytes)
CREATE src/app/gender.pipe.ts (217 bytes)
UPDATE src/app/app.module.ts (844 bytes)
```

spec.ts 是单元测试文件，对开发没有作用

自动注册生成的管道 到全局

```
<p>myc04 works!</p>

<!-- 自定义管道 -->

<!-- 练习：性别 在服务器存储的值 往往是数字：0女 1男 2未知 ...代号 -->

<!-- 生成管道的命令：ng generate pipe 管道名  例如：ng g p gender -->

```

```
<!-- 生成管道后，必须重启服务器才生效 -->
<ul>
  <!-- 显示 女 -->
  <li>{{ 0 | gender }}</li>
  <!-- 显示 男  -->
  <li>{{ 1 | gender }}</li>
  <!-- 显示 保密 -->
  <li>{{ 2 | gender }}</li>
</ul>

<!-- 练习：pf  求平方 -->
<ul>
  <li>{{ 3 | pf }} 9</li>
  <li>{{ 9 | pf }} 81</li>
</ul>

<!-- 带参数的写法 -->
<ul>
  <li>{{ 2 | pow: 10 }} 2^10=1024</li>
  <li>{{ 5 | pow: 4 }} 5^4 = 5xx</li>
</ul>

```

TS的静态类型分析

```
// typescript 是 JavaScript 的升级版本：  把很多 Java 的语言特性 添加到了 JavaScript 中
// 由微软公司进行维护

// 最重要的特征：静态类型分析特征

// 语言排行榜 TIOBE 中，除了JS 都有 静态类型分析功能；
// 变量名:类型名
// :类型名  是写给 IDE 开发软件看的，即 vscode
function show(abc: string) {
  // vscode：读代码时  就知道 变量abc 是个 string 类型
  return abc.toUpperCase();
}

show(123);

/**
 * 静态类型分析功能：  就是对程序员友好的特性
 * 1. 变量使用有提示
 * 2. 预警：代码不用运行 也能预判错误
 *
 * 有了这两个特征：有效降低 bug 的出现
 */

```

自定义指令

> 在 vue 中：directive

生成组件：`ng g c myc05`

> 指令的生成命令：
>
> ng generate directive 指令名
>
> 简化：ng g d 指令名

```
<p>myc05 works!</p>

<!--
  两个特殊
  {{ 值 | 管道}}  管道主要修改 双标签的值

  <tag 指令 />   指令主要修改 标签的DOM属性
 -->
<ul>
  <li>亮亮</li>
  <!-- app-hide：自定义指令，用于操作 style.display 属性，影响元素 -->
  <!-- 生成指令的命令行：ng generate directive 指令名，简写 ng g d 指令名 -->

  <!-- 重启服务 -->

  <!-- 例如：  ng g d hide-->
  <li>然然</li>
  <li appHide>东东</li>
</ul>

```

```
import { Directive, ElementRef } from '@angular/core';

@Directive({
  selector: '[appHide]',
})
export class HideDirective {
  // 指令初始化时，参数固定为 所在的 元素
  // 构造函数：对象初始化时触发；
  constructor(e: ElementRef) {
    console.log(e);

    e.nativeElement.style.display = 'none';
  }
}

```

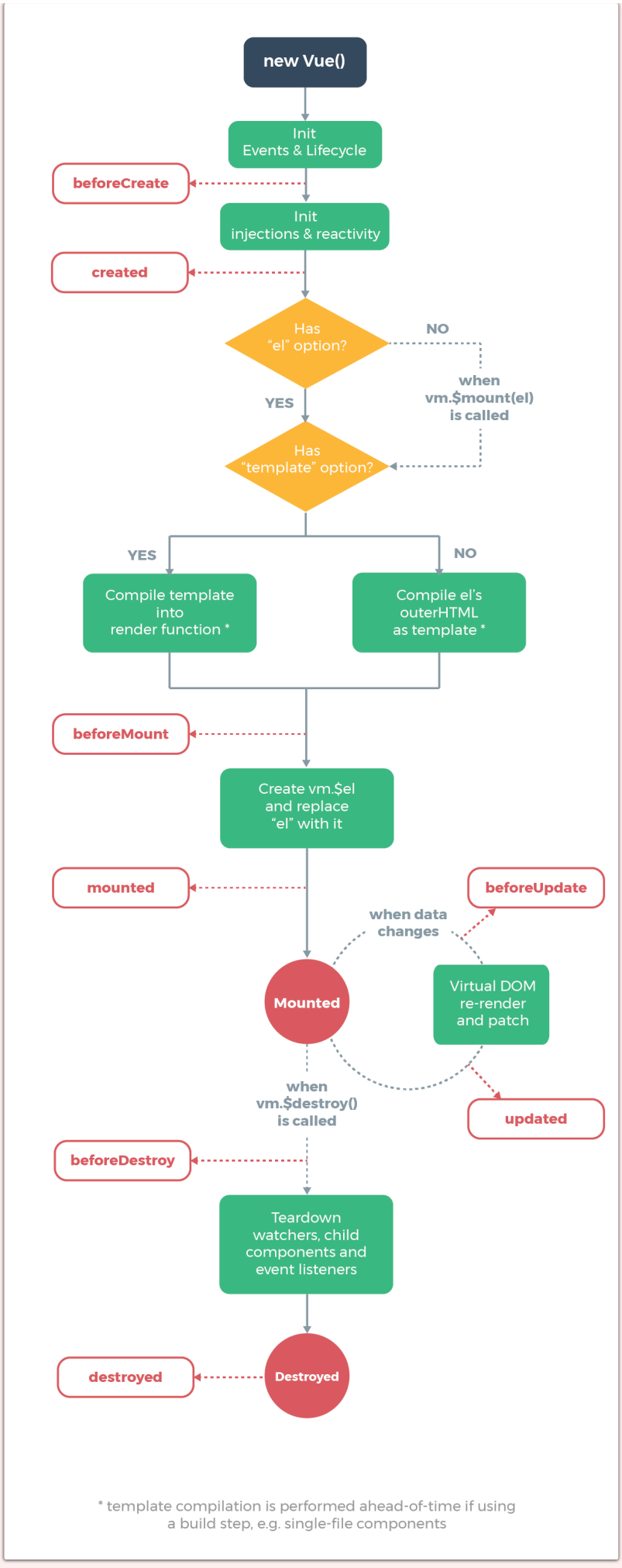
生命周期

生命周期：组件拟人化。 从出生 到 死亡 的过程中的 重要节点

备孕 -> 怀孕 -> 准备生 => 出生 => 学知识 => 学会 => 准备死 => 死了

vue的生命周期：

准备创建 => 创建出来 => 准备挂载 => 挂载完毕 => 准备更新 => 更新完 => 准备销毁 =>销毁



创建组件：`ng g c myc06`

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

@Component({
  selector: 'app-myc06',
  templateUrl: './myc06.component.html',
  styleUrls: ['./myc06.component.css'],
})
export class Myc06Component implements OnInit {
  count = 1;

  constructor() {
    // 构造方法：严格说不算生命周期.
    // 面向对象中，对象实例化时 触发
    console.log('constructor：构造方法');
  }

  ngOnInit(): void {
    // 相当于 vue 的 created
    console.log('ngOnInit：开始创建');
  }

  // 更新 分为：UI的更新 和 数据的更新
  ngAfterContentInit(): void {
    // 相当于 mounted 挂载
    console.log('ngAfterContentInit：数据初始化完毕后');
  }

  ngAfterViewInit(): void {
    // 相当于 mounted 挂载
    console.log('ngAfterViewInit：UI初始化完毕后');
  }

  ngAfterContentChecked(): void {
    console.log('ngAfterContentChecked：数据变更');
  }

  ngAfterViewChecked(): void {
    console.log('ngAfterViewChecked：UI变更后');
  }

  ngOnDestroy(): void {
    console.log('ngOnDestroy：将要销毁');
  }
}
```

TypeScript

微软公司 在 JavaScript 的基础上，添加了 Java 的很多特征，是一个升级版的语言.

官方网站： https://www.tslang.cn/

Typescript 无法直接在浏览器运行， 因为浏览器只识别 JavaScript;

此时需要编译器，来进行编译操作：

```
npm i -g typescript
```

```
+ typescript@4.1.3
added 1 package from 1 contributor in 2.211s
```

检测版本： `tsc -v`

> 凡是安装之后，提示 tsc **非内部或把外部命令**：
>
> 一定是 你的 npm 文件夹没有配置到 环境变量里

```
// 特征： 静态类型分析

// 参数名:类型
// :类型 是给 vscode 看。 vscode 就知道参数的类型，进而提供两大功能：
// 1. 代码提示
// 2. 报错预警
function show(name: string) {
  // 代码提示
  console.log(name.toUpperCase());
}

//报错预警
show("mike");

/**
 * 想要运行，则必须转化成 JS 文件
 *
 * 转化方式：
 * 1. 安装成功：tsc 01.ts
 * 2. 安装失败：npx tsc 01.ts
 *    npx是指临时下载 tsc 使用
 */

```

```
// 类型声明 详细介绍

class Demo {
  // 完整格式： 变量名：类型名 = 值;

  name: string = "lucy"; //字符串
  age: number; //数字
  married: boolean; // 布尔类型 true / false

  xyz: any; //any 任意类型。为默认值

  // 自定义混合类型
  abc: number | boolean; // 数字 或 布尔

  // 数组：两种写法，都代表 数组中 都是 string 类型元素
  names: Array<string>;
  items: string[];

  // 规定数组中每个值的类型 及 数量
  emp: [string, number, boolean];
```

```
  show() {
    this.emp = ["dongdong", 33, true];
    this.emp = ["dongdong", 33];
    this.emp = ["dongdong", true];

    // this.names = ["mike", "lucy", 123, true];
    // this.items = ["mike", "lucy", 123, true];

    this.abc = 123;
    this.abc = true;
    // this.abc = "mike";

    this.xyz = 12;
    this.xyz = "mike";
  }
}

```

```
// 对象类型

let boss = {
  name: "wenhua",
  age: 33,
  gender: 1,
  married: true,
};

// 声明对象类型的范式:
// interface: 接口.  新的关键词 与 class function 一样.  用于声明类型
interface Boss {
  // 模板中规定:  要哪些属性, 属性要什么类型
  name: string;
  age: number;
  gender: number;
  married: boolean;
}

// 一个对象 是 Boss 类型:  系统会自动检测 对象是否符合 模板要求
let dong: Boss = {
  name: "东东",
  age: 44,
  gender: 1,
  married: true,
};

// 雇员IT  ---  模板
interface IT_Employee {
  name: string;
  age: number;
  skills: string[]; //数组类型, 其中都是字符串
}

let youyu: IT_Employee = {
```

```
  name: "鱿鱼须",
  age: 26,
  skills: ["html", "css", "js", "vue", "react"],
};

```

```
// 面向对象的 访问控制词：  可以让对象的属性 更加安全

/**
 * public: 公开的 --  类外 类内 子类
 * protected: 保护的 -- 类内 子类
 * private: 私有的 -- 类内
 */

class RanGe {
  // 默认不写关键词, 则是 public 代表公开: 类外可以
  public name = "然然";
  // protected: 保护的.  类外无法访问
  protected money = "999元";
  protected wifes = ["貂蝉", "小乔", "贾玲"];
  // 私有: 只有类内可以使用,  子类也不行
  private avi = "然哥珍藏多年的 小电影";

  show() {
    this.avi; //类内可以访问私有
  }
}

// 然哥的儿子
class Son extends RanGe {
  show() {
    // this.avi;
  }
}

// 在 类外 访问类中的属性
let rg = new RanGe();
// rg.money; //类外无法访问 保护属性

```

作业

待办事项

请输入待办事项 确定

* 吃饭 删除
* 睡觉 删除
* 打亮亮 删除

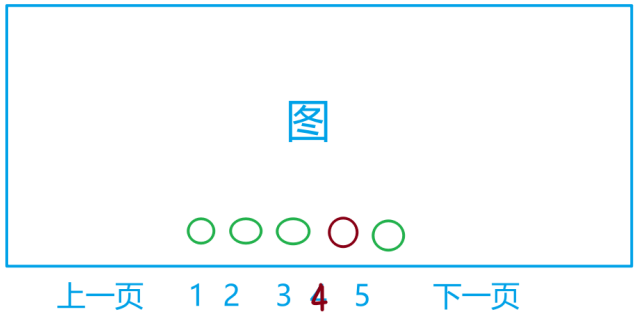
暂无待办事项

要求:
1. 输入框没有值时, 确定 按钮不可点击
2. 点击确定按钮后, 输入框的值 显示在下方列表同时清空输入框
3. 点击 删除 按钮, 删除对应项目
4. 所有待办事项删除后, 则显示 暂无待办事项

轮播图练习

提示：需要一个变量保存当前页的序号．然后通过改变这个序号 来改变图片

- 准备一个图片数组，本地图 和 网络图都可以． 数组中存放图片名称 或 路径url
- 每隔2.5s 滚动一次，要实现循环滚动
- 小圆点会 随着当前页发生样式变更
- 上一页 和 下一页 及 页数 使用 span 标签制作
- 当达到极限页数时：首页 和 末页． 则对应的 上一页 和 下一页 不可点击， 变为浅灰色．
- 1 2 3 4 5 这些页数 也可以点击 实现变化



管道

制作一个管道，实现 绝对值：正数

```
{{ 9 | abs }}  结果9
{{ -9 | abs }}  结果9
```

自定义指令

为标签添加红色背景色，白色文字

```
<p appDanger>Hello yy</p>
```