准备: 微信小程序: WEB问题速查 小恐龙图标

1. 历届师兄师姐在今天课程中曾经问过的高频的经典的问题清单和解答: 小程序->首页->VUE->day01、day02

自己可能没有发现问题，但是想知道别的同学都问过哪些问题

2. 今日课程对应的重点案例的视频以及鄙视面试所需的扩展知识视频：小程序->在线->VUE-> day01、02

(学有余力!!!的同学或自己认为自己基础好的同学必须看!!!!!!)：

今日扩展视频：小程序->在线->VUE->day02

3. 双向绑定 v-model ... 防抖和节流——基础弱的同学不用看！

4. 双向绑定 v-model radio ... 选择飞机

5. 双向绑定 v-model select两级select级联选择省份和城市——基础弱的同学不用看！

6. 双向绑定 v-mode checkbox ... 多选 爱好——基础弱的同学不用看！

7. 双向绑定 v-model简写 监视函数 watch ——基础弱的同学不用看！

8. 绑定样式... 手机号验证 终极简写——基础弱的同学不用看！

作业: （扩展）高频笔试题 模拟VUE绑定原理

3. 今日课程中包含的英语单词的翻译：小程序->单词->VUE->day01、day02

英文不太好的同学，可以提前看一下第二天需要用到哪些英文单词。

约定: 2件事:

1. 第三阶段禁止上课跟着敲笔记和敲代码！谁抄笔记抄的勤，谁抄代码抄的勤，谁学的越烂！

2. 每个完整功能的案例/视频，至少写三遍: (如果需要画图，必须亲手画图)

(1). 连代码+注释抄一遍 —— 明白这个例子是要做什么事儿

(2). 保留注释，自己试着看着注释，将代码填回来 —— 复习函数和对象

(3). 删除所有代码和注释，自己试着用自己的话，把注释写回来，再翻译为代码！—— 只有自己可以写出人话的注释，这个例子才算自己会！

如果没有问题的同学: （享受vip一对一答疑的时间还剩4+3天，端午节不休息）

1. 上课时那句话没听懂，笔记中那句话看不懂，案例中那句话看不懂，立刻问！

2. 小程序问题清单中哪个问题或解答看不懂，立刻问！

3. 小程序视频中哪个视频看不懂，立刻问

4. 个人项目中哪个功能不会做，立刻问

5. 平时刷笔试题时，遇到不会的笔试题，立刻问

正课:

1. 指令:

2. 双向绑定:

3. 绑定样式:

4. 自定义指令:

一. 指令:

1. 根据数组内容，反复创建多个相同结构的元素: v-for

对应小程序视频: 小程序->在线->VUE->day01 4. v-for 遍历数组...

(1). <要反复创建的元素 v-for="(value, i ) of 数组/对象/字符串">

(2). 原理: 当new Vue()首次扫描到这里时，或依赖的数组内容发生变化时:

a. 自动遍历of后的数组或对象...

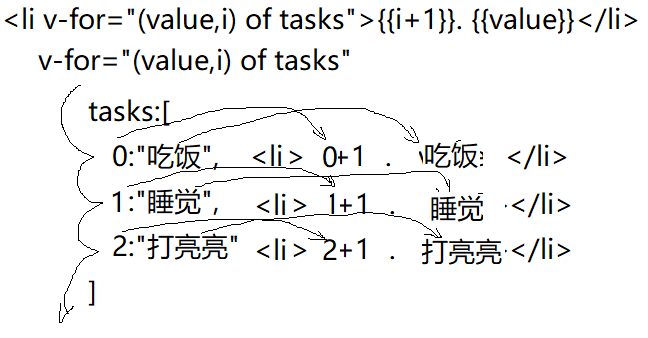
b. 每遍历一个成员，就

1). 将成员的值交给of前的value变量，

2) 将成员的下标位置交给of前的i变量

c. 反复创建v-for所在HTML元素，并将value和i的值替换到元素中可能发生变化的位置.

d. 在v-for所在的当前元素上以及当前元素的所有子元素中，都可以用value和i变量进行绑定。



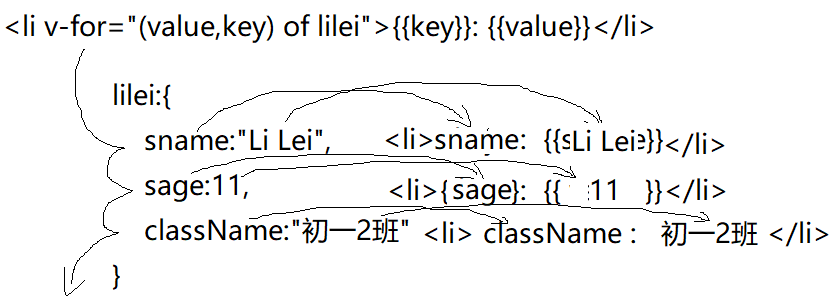
(3). 强调:

a. v-for必须写在要反复生成的元素上，而不能写在父元素上:

比如: 反复生成一个ul下的多个li元素，v-for应该写在子元素li上

b. of前的变量(value,i )可随意改名！但第一个变量始终接元素值，第二个变量始终接下标

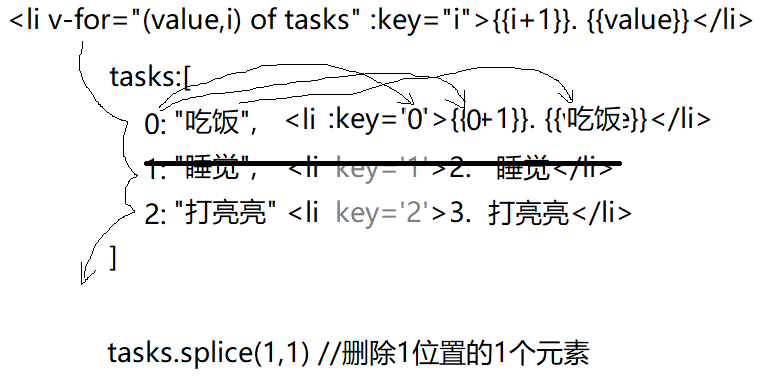
(4). 其实vue中的v-for of，统一了js中的for of和for in的功能！既可遍历数字下标，又可遍历自定义下标。



(5). 问题: 因为v-for反复生成的多个HTML元素，除了内容不同之外，元素本身毫无差别！导致，万一依赖的数组中某个元素发生变化，v-for无法精确的找到具体应该修改哪个HTML元素副本。所以v-for选择最笨的方法，将所有元素删掉，重新遍历数组，重新创建所有副本——效率低

(6). 解决: 凡是用v-for时，都必须同时绑定:key="下标"属性

结果: 每个HTML元素副本上都有一个key="下标"属性。当对数组中某一个元素执行操作时，vue会根据下标找到对应key="下标"的一个元素，只修改这一个元素，不影响其他元素。—— 效率高！



(7). 总结: 鄙视: 为什么v-for必须加:key？

a. 避免在修改某一个数组成员时，重建所有HTML元素副本

b. 如果加了:key，每次只需要修改一个数组元素对应的一个HTML元素副本即可——效率高

(8). 示例: 使用v-for遍历数组和对象

1\_v-for.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <script src="js/vue.js"></script>  </head>  <body>  <div id="app">  <ul>  <!--因为要根据数组tasks的内容反复生成多个相同结构的li元素，所以应该在li元素上用v-for-->  <li v-for="(value,i) of tasks" :key="i">  <!--因为反复生成的内容中需要动态获得每个元素值和下标，所以，可以用of前的变量value和i用于绑定语法中动态生成内容-->  {{i+1}}. {{value}}  </li>  </ul>  <!--想遍历出lilei对象中每个属性的属性名和属性值-->  <ul>  <li v-for="(value,key) of lilei" :key="key">{{key}}: {{value}}</li>  </ul>  </div>  <script>  new Vue({  el:'#app',  data:{  //有一个任务列表，希望展现到页面上  tasks:["吃饭","睡觉","打亮亮"],//.length=3  // 0 1 2  lilei:{  sname:"Li Lei",  sage:11,  className:"初一2班"  }  }  })  </script>  </body>  </html>  运行结果: |

(9). 其实: v-for还会数数: 仅根据一个数字，就可反复生成指定数量的HTML元素副本，且从1开始数数，一直数到给的数字为止

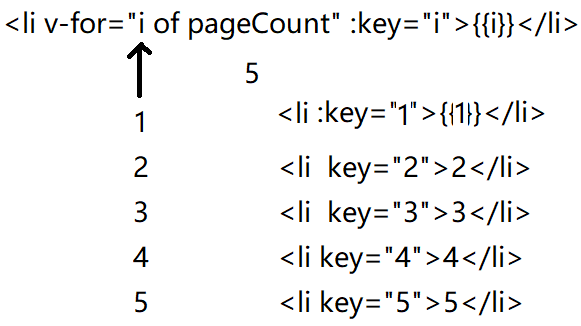
a. <要反复生成的元素 v-for="i of 数字">

b. 原理: 当new Vue()首次扫描到这里时，或依赖的数值发生变化时:

1). 数字是几，就会反复创建几个HTML元素副本

2). 每次创建副本时，都会将本次数到的数字保存到of前的变量中

3). 在当前元素及其子元素中变量i，可用于绑定。



(10). 示例: 使用v-for生成分页按钮

2\_v-for.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <script src="js/vue.js"></script>  <style>  #pages{ list-style: none; }  #pages>li{  float:left;  padding:5px 10px;  border:1px solid #aaa;  margin:0 5px;  }  </style>  </head>  <body>  <div id="app">  <!--希望总页数是几，就反复生成几个分页分页按钮-->  <ul id="pages">  <li v-for="i of pageCount" :key="i">{{i}}</li>  </ul>  </div>  <script>  new Vue({  el:"#app",  data:{  pageCount:5,//分页时都会有一个变量保存总页数  }  })  </script>  </body>  </html>  运行结果: |

2. 绑定事件: v-on

对应小程序视频: 小程序->在线->VUE->day02 1. 事件绑定v-on ...

(1). 标准: <元素 v-on:事件名="事件处理函数()">

(2). 等效于: DOM中的<元素 on事件名="事件处理函数()">

(3). 重要差别: VUE中事件处理函数中的this，~~不再指向当前触发事件的元素~~。而是指向当前整个new Vue()对象！

(4). 简写:

a. "v-on:"可用"@"代替: <元素 @事件名="事件处理函数()">

b. 如果事件处理函数没有任何实参值，则可以省略():<元素 @事件名="事件处理函数">

(5). 其实: vue中的事件处理函数可以传参！

a. <元素 @事件名="事件处理函数(实参值列表)">

methods:{

事件处理函数(形参列表){ ... }

}

b. 示例: 点哪个div，哪个div喊谁疼！

3\_@.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <script src="js/vue.js"></script>  <style>  #app>div{  width:300px;  height:100px;  }  #app>div:hover{  box-shadow:0 0 5px red  }  </style>  </head>  <body>  <div id="app">  <!--因为点击两个div都能喊疼，只不过div的名字不同，所以两个div可共用同一个事件处理函数，只不过传入不同的div名字作为实参即可！-->  <div id="d1" style="background-color:#aaf" @click="say('d1')"></div>  <div id="d2" style="background-color:#ffa" @click="say('d2')"></div>  </div>  <script>  new Vue({  el:"#app",  data:{},  //因为界面中两个事件共用同一个事件处理函数，所以methods中只需要一个事件处理函数即可  methods:{  //但是，因为界面上的事件处理函数绑定时传入了一个实参值，所以methods中的事件处理函数就必须用一个形参变量接住实参值！  say(id){  alert(`${id} 疼! `)  }  }  })  </script>  </body>  </html>  运行结果： |

(6). 其实: 在vue中也可获得事件对象: 2种

a. 无需传其它实参值，只希望获得事件对象时：

1). 同DOM: 事件对象总是默认作为事件处理函数的第一个实参值自动传入

2). <元素 @事件名="事件处理函数">

methods:{

事件处理函数(e){ ... }

}

3). 问题: 如果需要同时传入实参值和获得事件对象，实参值和事件对象e传入的位置就会撞车！

<元素 @事件名="事件处理函数(实参值)">

event

methods:{

事件处理函数(e){ ... }

}

4). 错误的解决1: 在methods中在事件处理函数中e之前多加一个形参变量

<元素 @事件名="事件处理函数(实参值)">

event

methods:{

事件处理函数(形参1, e){ ... }

}

因为event对象，默认只能传给第一个形参变量，不会给之后的其它形参

5) 错误解决2: methods中交换事件处理函数的两个形参变量的顺序

<元素 @事件名="事件处理函数(实参值)">

event

methods:{

事件处理函数(e, 形参1){ ... }

}

因为第一个实参值，不会聪明到自动给第二个形参变量，依然给第一个形参，依然会和event发生冲突

6). 示例: 鄙视题: vue中如何获得鼠标位置:

4\_@\_e.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <script src="js/vue.js"></script>  <style>  #app>div{  width:300px;  height:100px;  }  #app>div:hover{  box-shadow:0 0 5px red  }  </style>  </head>  <body>  <div id="app">  <!--点哪个div的某个位置，就喊某个为疼！-->  <div id="d1" style="background-color:#aaf" @click="say"></div>  <div id="d2" style="background-color:#ffa" @click="say"></div>  </div>  <script>  new Vue({  el:"#app",  data:{},  methods:{  //因为想获得鼠标点击的位置，必须加形参e  //event  // ↓  say(e){  alert(`${e.offsetX},${e.offsetY}位置 疼! `)  }  }  })  </script>  </body>  </html>  运行结果： |

b. 如果同时传入实参值和事件对象：借助于vue一个关键词$event

DOM event

先↓

<元素 @事件名="事件处理函数(实参值, $event)">

methods:{

事件处理函数(形参1, e){ ... }

}

说明: $event和实参值可以交换位置。$event无论在实参列表中第几个位置，都可先获得事件对象event。不受位置影响！

c. 示例: 鄙视题: 如果同时传入自定义实参值和事件对象:

5\_@\_$event.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <script src="js/vue.js"></script>  <style>  #app>div{  width:300px;  height:100px;  }  #app>div:hover{  box-shadow:0 0 5px red  }  </style>  </head>  <body>  <div id="app">  <!--如果既需要传入实参值，又需要获得事件对象-->  <!-- DOM event-->  <!-- 先 ↓ -->  <div id="d1" @click="say($event,'d1')" style="background-color:#aaf" ></div>  <!-- DOM event-->  <!-- 先 ↓ -->  <div id="d2" @click="say($event,'d2')" style="background-color:#ffa" ></div>  </div>  <script>  new Vue({  el:"#app",  data:{},  methods:{  //因为界面上调用事件处理函数时，传入了两个实参，所以methods中定义事件处理函数时，也必须定义两个形参对应  say(e, id){  alert(`${id}的 ${e.offsetX},${e.offsetY}位置 疼! `)  }  }  })  </script>  </body>  </html>  运行结果: |

对应小程序视频: 小程序->在线->VUE->day02 2. 其它指令 绑定HTML内容

3. 防止用户短暂看到{{}}:

4. 绑定HTML内容:

5. 只在首次加载时绑定一次:

6. 防止内容中{{}}被编译:

总结:

1. MVVM: 界面View+模型对象Model+视图模型ViewModel

2. Vue的绑定原理: 访问器属性+虚拟DOM树

3. 虚拟DOM树的优点:

(1). 小，仅包含可能变化的元素

(2). 遍历查找快

(3). 更新效率高，只更新受影响的元素

(4). 避免重复编码，已封装了DOM的增删改查操作

总结: 今后所有VUE功能三步:

1. 创建界面，界面中标记变量和事件处理函数

2. 创建new Vue()对象，用el:"选择器"，找到自己负责的页面区域的唯一父元素

3. 在new Vue()中添加data和methods，分别保存界面所需的所有变量和事件处理函数

总结:

1. 绑定语法和指令:

(1). 只要元素的内容随变量自动变化: {{}}

(2). 只要元素的属性值随变量自动变化: :

(3). 控制一个元素显示隐藏: v-show （display:none）

(4). 控制两个元素二选一显示: v-if v-else （删除元素）

(5). 控制多个元素多选一显示: v-if v-else-if v-else

(6). 反复创建多个相同结果的元素: v-for :key="下标"

(7). 绑定事件: @ $event

总结: this 7种:

1. obj.fun() this->obj

2. new Fun() this->new正在创建的子对象

3. 类型名.prototype.共有方法=function(){ ... } this->将来调用这个共有方法的.前的子对象

4. fun() 和 (function(){ })() 和回调函数中的this->window

5. 事件处理函数中的this->当前正在触发事件的这个元素对象

6. $jquery对象.click(function(){ this->将来点击的那个DOM元素对象})

7. new Vue()中的几乎所有this，都指向当前new Vue()对象。（回调函数用箭头函数）

今日对应小程序视频:

小程序->在线->VUE->day01 4. v-for 遍历数组...

小程序->在线->VUE->day02 1. 事件绑定v-on ...

2. 其它指令 绑定HTML内容

3. 双向绑定v-model 绑定文本框

4. 双向绑定v-model radio

5. 双向绑定v-model select

6. 双向绑定 v-model checkbox

7. 双向绑定简写

8. 绑定样式 style...

9. 自定义指令vue directive

作业:

1. 复习今日小程序问题清单(小程序->首页-> VUE ->day01、day02)，预习明日小程序问题清单(小程序->首页->VUE->day03)

2. 看小程序视频预习计算属性:

小程序->在线->VUE->day02 10. 计算属性 computed...

3. 看小程序视频预习过滤器用法

小程序->在线->VUE->day03 3. 过滤器

4. 看小程序视频预习axios发送ajax请求

小程序->在线->VUE->day03 4. axios 发送ajax请求

5. 看小程序视频完成用vue实现学子商城首页和详情页部分功能:

小程序->在线->VUE->day01 作业: 使用vue绑定学子商城首页 ...

下载网盘中public\_start文件夹

将今日上课素材day01中js文件夹里vue.js，拷贝到public\_start下的js文件夹下

在public\_start/js文件夹下新建index.js文件

在public\_start/js文件夹下新建details.js文件

然后, 在index.html页面body内，底部，引入vue.js和index.js

然后, 在product\_details.html页面body内，底部，引入vue.js和details.js

然后，看视频，在index.js和details.js中添加自定义代码

使用ajax请求的接口改为: http://xzserver.applinzi.com/index

http://xzserver.applinzi.com/details

网盘中public压缩包是最终结果，如果发现错误，可以参考