小程序: WEB问题速查

1. 问题清单: 历届师兄师姐在今天课程中集中问过的高频的重点的问题！

那些自己发现不了问题，又迫切想知道其它同学都问了什么问题，的同学

小程序->首页->VUE->day02、day03

当老师在回答其它同学的问题时，你不要闲着！你可以打开问题清单浏览别人曾经问过的经典问题！

2. 高清无码小视频: 小程序->在线->VUE->day02、day03

2部分: 1. 课上重点知识和案例的回放！2. 重点高频笔试面试题讲解扩展视频

万一家里或中心网络断了！暂时看不了直播了！

因为某些事儿请假了，错过了当天的直播！

课下复习时，觉得TMOOC视频太大，浏览不方便或TMOOC不能看了！

——打开小程序中"在线"，就可下载观看当前所讲内容和当天所讲全部内容的回放。

基础比较好的，确实学有余力的同学，想拿高薪的同学，在我答疑时，不要闲着:

要看今天扩展视频:

小程序->在线->VUE->day02

3. 双向绑定 v-model 绑定文本框...边输入边搜索，按回车搜索，防抖和节流

4. 双向绑定 v-mode radio ... 选择飞机

5. 双向绑定 v-model select ... 级联选择省份和城市

6. 双向绑定 v-model checkbox ... 多选 爱好

8. 绑定样式 style ... 手机号验证 终极简写

10. 计算属性 computed 作业 vue 购物车 仅包含客户端

作业:(扩展)高频笔试题 模拟VUE绑定原理

3. 单词列表: 小程序->单词->VUE->day02、day03

第三阶段上课约定:

1. 上课千万不要跟着记笔记和写代码！

2. 凡是拼写错误，自己懒得找！我可以帮你找，但是找到后，错误的单词你要手抄20遍，并拍照！

出错: 先开f12，console，看错误提示！

3. 每个带注释的案例至少写三遍:

(1). 注释+代码抄一遍

(2). 保留注释，删除代码，自己试着把代码翻译回来！

(3). 删除注释和代码，自己试着用自己的话把注释写回来！

一个例子只有自己可以把人话的注释写回来！才算自己会！

问题: 一听就会一做就废！原因: 你自己无法用人话精确描述自己想干什么！

如何提问: （vip待遇的剩余时间3+2天）

1. 上课时，案例中，笔记中那句话哪行代码看不懂，立刻问！

2. 小程序中问题清单里哪个问题和回答看不懂！立刻问！

3. 个人项目中哪个功能不会做！立刻问！

4. 刷笔试题时，遇到不会的笔试题！立刻问！

谁问的问题越多！谁的培训费交的越值！

正课:

1. 指令

2. 双向绑定

3. 绑定样式

4. 自定义指令

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

一. 指令:

1. **v-html**：

(1). 什么是: 专门**绑定HTML代码片段**内容的指令

(2). 问题: {{}}和v-text在绑定元素内容时，即使要绑定的值是HTML片段，也不会交给浏览器解析，而是尽量保持原样显示。——类似于DOM中textContent的特点

(3). 解决: 今后只要要**绑定**的内容是一段**HTML片段时**，都**用v-html代替v-text和{{}}**。

(4). 原理: v-html绑定的HTML片段内容会先交给浏览器解析后，再显示到页面上给人看。

(5). 示例: 使用v-html绑定一段HTML代码片段:

1\_v-html.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">      <h3>{{msg}}</h3>      <h3 v-text="msg"></h3>  **<h3 v-html="msg"></h3>**    </div>    <script>      new Vue({        el:"#app",        data:{          msg:`来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息`        }      })    </script>  </body>  </html>  运行结果:  **来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息**  **来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息**  **来自**[**<<新华社>>**](javascript:;)**的消息** |

2. **v-once:**

(1). 什么是: 只在首次加载时，绑定一次。之后即使变量发生变化，也不自动更新界面

(2). 问题: 每次程序中变量更新时，vue都会重新扫描虚拟DOM树。虚拟DOM树中保存的元素越多，扫描一遍用时越长。虚拟DOM树中保存的元素越少，扫描一遍用时越短。

(3). 优化: 如果**一个元素的内容只需要**在**首次**加载时**动态显示**，在**之后**的操作过程中**不会发生变化**，就应该用**v-once标记。**

(4). 原理: v-once标记的元素，只在首次加载页面时动态加载内容。因为之后不再需要动态更新，所以并**不加入到虚拟DOM树中**，减少了虚拟DOM树中的元素个数。

(5). 何时: 只有一个值确实只在首次动态加载，之后不再发生变化时，才用v-once。

(6). 示例: 绑定用户上线时间和当前系统时间

2\_v-once.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>    <!--1. 做界面-->    <div id="app">      <!--想显示用户上线的时间点(首次动态加载，之后不再改变)-->      <h3 v-once>用户上线时间: {{time}}</h3><!--不在虚拟DOM树-->      <!--想显示当前系统时间(随系统时间频繁变化)-->      <h3>当前系统时间: {{time}}</h3><!--在虚拟DOM树-->    </div>    <script>      //2. 创建new Vue()对象，监视id为app的区域      var vm=new Vue({        el:"#app",        //3. 创建模型对象        data:{ //只在首次创建new Vue()时执行一次，不会反复执行               //当前系统时间当地字符串格式          time:new Date().toLocaleString()        }      });      //每隔1秒      setInterval(function(){        //修改new Vue()对象中的time属性为新的当前系统时间        vm.time=new Date().toLocaleString()      },1000)    </script>  </body>  </html>  运行结果: 用户上线时间: 2020/7/22 上午9:48:54当前系统时间: 2020/7/22 上午9:49:30 |

3. **v-pre**:

(1). 什么是: 可以保持元素内容中的**{{}}不被vue编译**的特殊指令

(2). 何时: 只要刚巧在元素的**正文中包含{{}}**，但是又**不想被vue编译**，就用**v-pre保护**

(3). 示例: 使用v-pre保护内容中{{}}不被vue编译

3\_v-pre.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">      <h3 v-pre>Vue框架中用{{变量名}}语法来标记页面中可能变化的位置</h3>    </div>    <script>      new Vue({        el:"#app"      })    </script>  </body>  </html>  运行结果: Vue框架中用{{变量名}}语法来标记页面中可能变化的位置 |

二. **双向绑定:**

对应小程序视频: 小程序->在线->VUE->day02 3. 双向绑定 **v-model** 绑定文本框...

1. 问题: 使用:value="变量名"，无法将页面上用户输入的新内容，自动更新回程序中new Vue()对象里的变量中。

2. 原因: 之前学的绑定语法以及12种指令，都是单向绑定:

(1).只能将程序中变量的修改自动更新到界面上显示(只能Model->View)

(2). 无法将界面上的修改反向更新回程序中的变量中(无法View->Model)

3. 解决: 今后只要希望动态获得界面上用户修改的新值，都要用双向绑定:

(1). 既能将程序中变量的修改自动更新到界面上显示(既能Model->View)

(2). 又能将界面上的修改反向更新回程序中的变量中(又能View->Model)

4. 何时: **只有绑定表单元素时**，才需要用到**双向绑定**。因为只有表单元素的值，用户才能修改。除表单元素之外的其它普通元素(div, h3, p,...)，都不需要双向绑定。因为这些元素用户改不了！

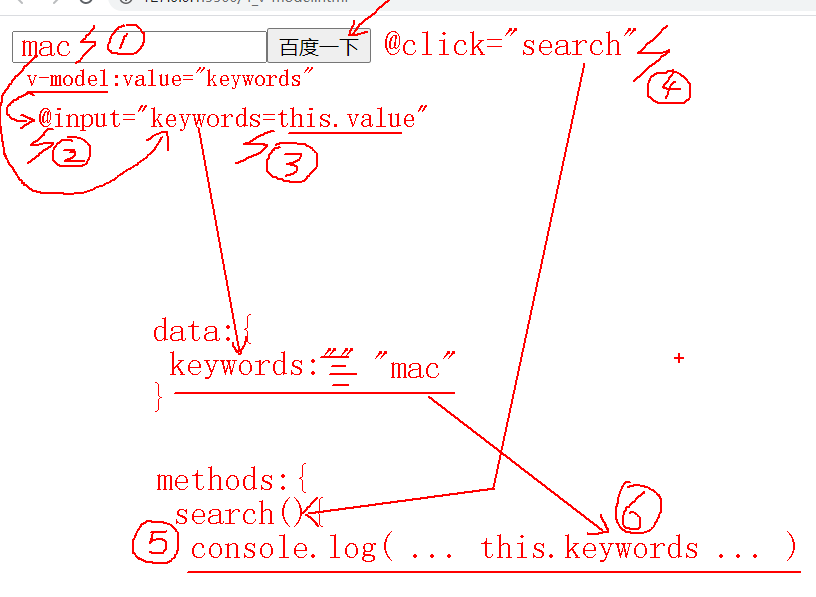
5. 如何: 标准: **<表单元素 v-model:value="变量名">**

6. 示例: 点按钮，获得文本框中输入的内容，执行查找操作

4\_v-model.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>    <!--1. 做界面-->    <div id="app">      <!--在文本框中输入关键词，点查询按钮执行用关键词查找的操作        当用户在文本框中输入关键词时，input元素的value属性会发生变化，所以需要定义变量keywords来绑定input的value属性,因为希望将input元素的新内容反向更新回程序中的变量中，所以不能用:绑定，而应该用v-model:绑定value属性        用户点按钮执行查找操作-->      <input type="text" v-model~~:value~~="keywords"><button @click="search">百度一下</button>    </div>    <script>      //2. 创建newVue()对象，监视id为app的区域      var vm=new Vue({        el:"#app",        //3. 创建模型对象保存界面所需的所有变量和事件处理函数        data:{          keywords:""        },        methods:{          search(){            //               JS高级第2天            //如果keywords去掉开头和结尾的所有空字符后不等于""            //if(this.keywords.trim()!==""){              //才执行查找操作              console.log(`查找 ${this.keywords} 相关的内容....`)            //}          }        }      })    </script>  </body>  </html>  运行结果: |

6. 原理: 当new Vue()扫描到带有v-model的元素时，都会自动给当前元素绑定DOM中的@**input**或@**change**事件。只要用户**输入**或**修改**了表单元素的值，都会自动触发事件。在事件处理函数中，提前保存了修改对应变量的语句。



7. 示例: 使用@input事件模拟实现v-model

5\_v-model\_oninput.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>    <!--1. 做界面-->    <div id="app">      <!--在文本框中输入关键词，点查询按钮执行用关键词查找的操作        当用户在文本框中输入关键词时，能自动修改vm对象中的keywords值        用户点按钮执行查找操作-->      <input id="txtKw" type="text" :value="keywords" @input="myChange"><button @click="search">百度一下</button>    </div>    <script>      //2. 创建newVue()对象，监视id为app的区域      var vm=new Vue({        el:"#app",        //3. 创建模型对象保存界面所需的所有变量和事件处理函数        data:{          keywords:""        },        methods:{          myChange(e){ //当前文本框的内容            this.keywords=e.target.value;          },          search(){            //               JS高级第2天            //如果keywords去掉开头和结尾的所有空字符后不等于""            //if(this.keywords.trim()!==""){              //才执行查找操作              console.log(`查找 ${this.keywords} 相关的内容....`)            //}          }        }      })    </script>  </body>  </html>  运行结果: |

8. 不同类型的表单元素，v-model绑定原理不同:

(1). **文本框(input type=text)和文本域(textarea)**

a. 特点: 用户输入新内容，新内容自动保存到当前元素的value属性中

b. 都可以: **v-model:value="变量名"**

(2). **单选按钮(input type=radio)：**

对应小程序视频: 小程序->在线->VUE->day02 4. 双向绑定 v-mode radio ...

a. 特点:

1). value属性值是写死的备选值，用户切换选项时，每个选项的value值保持不变！所以，不能用v-model:value。

2). 当用户切换选项时，其实改变的是radio元素的checked属性值，也就是选中状态。所以**v-model:checked="变量名"**

b. 如何: **<input type="radio" name="分组名" value="写死的备选值" v-model:checked="变量名">**

c. 原理:

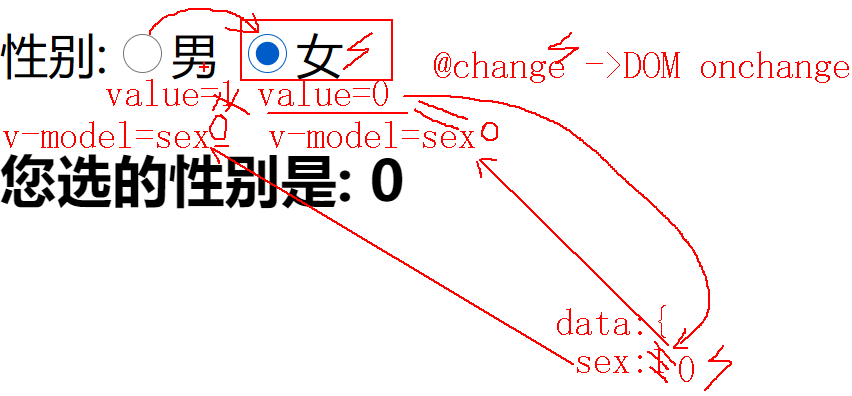
1). 首次加载时： 当new Vue()扫描到input type=radio中的v-model时，会用变量名和当前radio元素的value值做比较！

i. 如果v-model绑定的变量值等于value属性值，则当前radio就选中

ii. 如果v-model绑定的变量值不等于value属性值，则当前radio就不选中



2). 当选中项发生改变时: v-model会自动触发**onchange**事件，将当前选中的元素value值自动更新回程序中data中的变量上。



d. 示例: 选择性别:

6\_v-model\_radio.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>    <!--1. 做界面-->    <div id="app">      <!--想选择性别          无论选男，还是选女，其实都是选择性别属性一个值！所以，只需要一个变量sex就能保存选中的性别-->      性别:<label><input type="radio" name="sex" value="1" v-model~~:checked~~="sex">男</label>      <label><input type="radio" name="sex" value="0" v-model~~:checked~~="sex">女</label>      <!--上方选择后，下方显示选择了哪种性别-->      <h3>您选的性别是: {{sex}}</h3>    </div>    <script>      //2.创建new Vue()对象，监视id为app的区域      new Vue({        el:"#app",        //3. 创建模型对象，保存页面所需的所有变量        data:{          sex:1        }      })    </script>  </body>  </html>  运行结果: |

(3). **select**元素

对应小程序视频: 小程序->在线->VUE->day02 5. 双向绑定 v-model select ...

a. 特点:

1). 每个选项的value值不是写在select元素上，而是写在select下的每个option元素上，且这些option的value也都是写死的固定不变的备选值。

2). 当选中select下某一个option时，这个option的value，会成为整个select元素的value

b. 如何: 所以, 应该用v-model绑定select元素的value属性

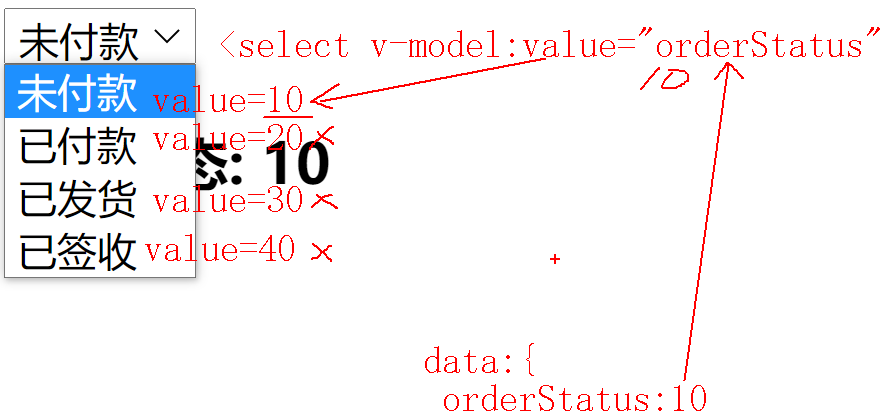
**<select v-model:value="变量名">**

**<option value="备选值">**

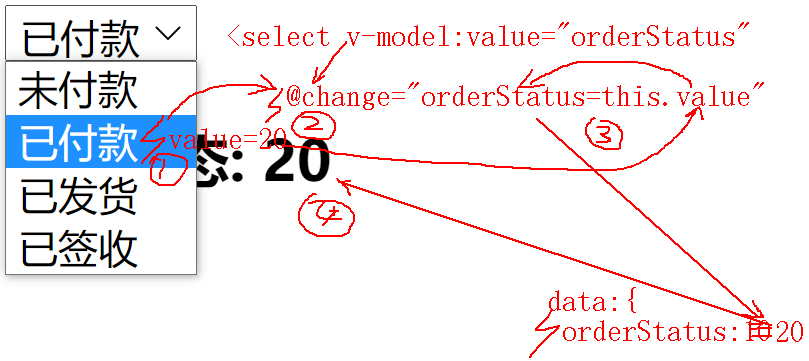
**... ...**

c. 原理:

1). 当首次加载页面时: v-model会用绑定的变量值和每个option中的备选值value做比较。哪个option的value值与变量的值相同，就选中哪个option



2). 当选项发生改变时: v-model会获得当前选中的option的value值，自动更新回v-model绑定的变量中。



d. 示例: 选择订单状态

7\_v-model\_select.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>    <!--1. 做界面-->    <div id="app">      <!--想用select选择订单状态          希望选中select下的某个option，自动更改程序中的orderStatus变量-->      <select v-model~~:value~~="orderStatus">        <option value="10">未付款</option>        <option value="20">已付款</option>        <option value="30">已发货</option>        <option value="40">已签收</option>      </select>      <!--下边显示选中的订单状态-->      <h3>订单状态: {{orderStatus}}</h3>    </div>    <script>      //2. 创建new Vue()对象监控id为app的区域      new Vue({        el:"#app",        //3. 创建模型对象保存界面所需的变量        data:{          orderStatus:10        }      })    </script>  </body>  </html>  运行结果: |

(4). **checkbox**: 只讨论单独使用的情况

a. 特点: 用户选中与不选中checkbox改变的不是value属性，而是checked属性，且**checked属性的值是bool类型**

b. 如何:**<input type="checkbox" v-model:checked="bool类型变量">**

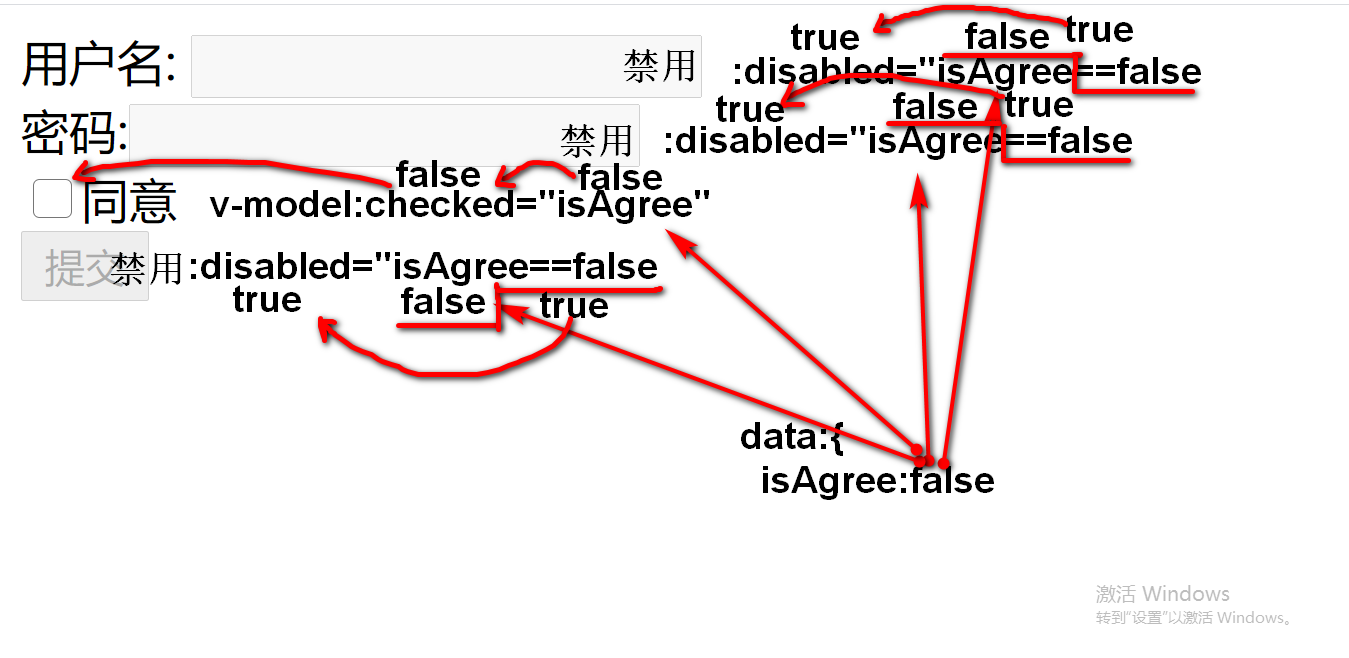
c. 原理:

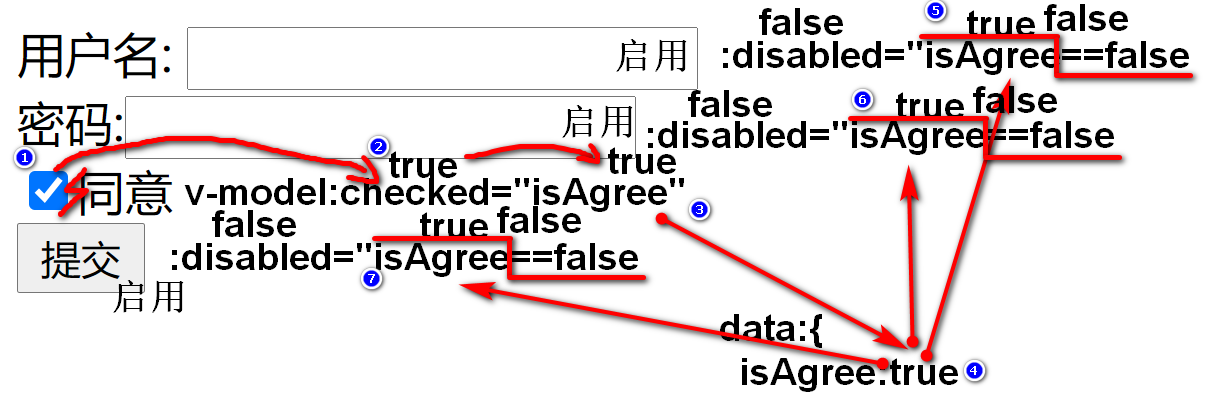
1). 首次加载时:

i. 如果v-model绑定的变量值为true，则当前checkbox选中

ii. 如果v-model绑定的变量值为false，则当前checkbox不选中

2). 在修改checkbox选中状态时: v-model会将checkbox最新的checked属性值，也就是checkbox的选中状态，自动更新回程序中的变量上，且值也是bool类型





d. 示例: 点同意，启用其它表单元素；不同意，禁用其它表单元素

8\_v-model\_checkbox.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>    <!--1. 做界面-->    <div id="app">      <!--选中同意，则启用其它表单元素。如果不选中同意，则其它表单元素禁用        用户点击同意可修改是否同意的状态        其它三个表单元素是否启用也取决于是否同意这个状态        所以，整个界面中只需要一个变量就够了:        isAgree 表示是否同意        希望其余三个表单元素的disabled属性随变量isAgree自动变化，应该用:绑定。  **又因为disabled的属性值是bool**，所以-->      用户名: <input :disabled="isAgree==false"></br>      密码:<input type="password" :disabled="isAgree==false"><br>      <!--用户点击checkbox修改的是checkbox的checked属性，当修改checked属性时，就将是否同意保存到变量isAgree中-->      <input type="checkbox" v-model~~:checked~~="isAgree">同意<br>      <input type="submit" :disabled="isAgree==false">    </div>    <script>      //2. 创建new Vue()对象，监视id为app的区域      new Vue({        el:"#app",        //3. 创建模型对象保存界面中所需的所有变量        data:{          isAgree:false //开局，isAgree=false，表示默认不同意        }      })    </script>  </body>  </html>  运行结果： |

9. 简写: 其实以上所有**v-model后的":属性名"都可省略**，v-model会自动根据自己所在的表单元素不同，自动判断该绑定哪个属性，所以，今后只要希望修改表单元素，同时自动修改程序中的变量时，只用v-model即可！

三. 绑定样式: 2种

1. 如果**内联样式**（style）需要根据程序中的变量自动变化

(1). 不好: 将style属性作为一个普通的字符串类型的属性来绑定

a. <元素 :style="变量">

data:{  
 变量:"css属性:值; css属性:值; ..."

b. 缺点: 极其不便于修改其中某一个css属性值

(2). 好的: vue中的style支持以对象语法绑定样式

a. 使用**匿名对象**的方式: js 语法

1). 如何: **<元素 :style="{ css属性名: 变量名1, css属性名: 变量名2}"**

**data:{**

**变量名1: 属性值1,**

**变量名2: 属性值2,**

**... : ...  
 }**

2). 原理: 运行时，new Vue()先用data中的变量值替换style对象中的属性后的变量名，最后将style后的对象，翻译为style字符串语法。

3). 问题: 如果多个变量都需要修改style中的内联样式，刚巧都需要修改同一种css属性值，很可能发生变量冲突

4). 解决: 用有名称的对象方式

b. 使用**有名称对象**的方式:

1). 如何: **<元素 :style="变量名">**

//再替换界面中:style的位置

**data:{**

**变量名:{** //运行时，会先将变量名后的对象内容编译为style字符串

**css属性名: 属性值,**

**css属性名: 属性值,**

**... : ...**

**}**

**}**

2). 好处: 能够避免多个元素之间要修改的style变量发生冲突

(3). 问题: 如果有些内联样式是固定不变的！而另一些内联样式是动态变化的

(4). 其实: 一个元素上**不带:的写死的style**，可以和**带:动态变化的style并存**！最后在运行时，带:的style和不带:的style会合并为一个style！

**<元素 style="固定不变的样式" :style="对象(包含变化的css属性)">**

(5). 示例: 使用内联样式控制飞机飞行

a. 用匿名对象方式绑定:

9\_style.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>      img{        position:fixed;      }    </style>  </head>  <body>    <!--1. 先做界面-->    <div id="app">      <!--希望img的top和left随程序中的变量自动变化        所以我们可以分别定义两个变量top和left分别控制垂直和水平方向的位移        又因为:style支持以对象语法方式绑定样式: -->      <img src="img/p3.png" style="position:fixed;" :style="{top:top, left:left}">    </div>    <script>      //2. 创建new Vue()对象监视id为app的位置      var vm=new Vue({        el:"#app",        //3. 创建模型对象保存界面所需的变量        data:{          top:"50px",          left:"100px"        }      });      //用键盘方向键控制飞机飞行      //当键盘按下时      window.onkeydown=function(e){        //37  左  38  上  39  右  40  下        if(e.keyCode==37){ //左移动          //取出vm中left变量值，转为整数          var left=parseInt(vm.left);          //left-10          left-=10;          //再修改回vm的left变量中保存，一定要拼上单位          vm.left=left+"px";        }else if(e.keyCode==38){//上          //取出vm中top变量值，转为整数          var top=parseInt(vm.top);          //top-10          top-=10;          //再修改回vm的top变量中保存，一定要拼上单位          vm.top=top+"px";        }else if(e.keyCode==39){ //右移动          //取出vm中left变量值，转为整数          var left=parseInt(vm.left);          //left+10          left+=10;          //再修改回vm的left变量中保存，一定要拼上单位          vm.left=left+"px";        }else if(e.keyCode==40){ //下移动          //取出vm中top变量值，转为整数          var top=parseInt(vm.top);          //top+10          top+=10;          //再修改回vm的top变量中保存，一定要拼上单位          vm.top=top+"px";        }      }    </script>  </body>  </html>  运行结果: |

b. 用有名称的对象方式绑定:

10\_style2.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>    <!--1. 先做界面-->    <div id="app">      <!--希望img的top和left随程序中的变量自动变化        但是别的元素也可能修改top和left属性        为了避免和别的元素的top和left属性冲突，可以将top和left属性装到一个有名称的对象内，再绑定-->      <img src="img/p3.png" style="position:fixed;" :style="imgStyle">    </div>    <script>      //2. 创建new Vue()对象监视id为app的位置      var vm=new Vue({        el:"#app",        //3. 创建模型对象保存界面所需的变量        data:{          imgStyle:{            top:"50px",            left:"100px"          }        }      });      //用键盘方向键控制飞机飞行      //当键盘按下时      window.onkeydown=function(e){        //37  左  38  上  39  右  40  下        if(e.keyCode==37){ //左移动          //取出vm中left变量值，转为整数          var left=parseInt(vm.imgStyle.left);          //left-10          left-=10;          //再修改回vm的left变量中保存，一定要拼上单位          vm.imgStyle.left=left+"px";        }else if(e.keyCode==38){//上          //取出vm中top变量值，转为整数          var top=parseInt(vm.imgStyle.top);          //top-10          top-=10;          //再修改回vm的top变量中保存，一定要拼上单位          vm.imgStyle.top=top+"px";        }else if(e.keyCode==39){ //右移动          //取出vm中left变量值，转为整数          var left=parseInt(vm.imgStyle.left);          //left+10          left+=10;          //再修改回vm的left变量中保存，一定要拼上单位          vm.imgStyle.left=left+"px";        }else if(e.keyCode==40){ //下移动          //取出vm中top变量值，转为整数          var top=parseInt(vm.imgStyle.top);          //top+10          top+=10;          //再修改回vm的top变量中保存，一定要拼上单位          vm.imgStyle.top=top+"px";        }      }    </script>  </body>  </html>  运行结果: |

(6). 问题: 网页中很多效果需要同时修改多个css属性，如果用style一个属性一个属性去修改，代码会很繁琐。

(7). 总结:

a. 如果确实需要精确修改某一个css属性值时才用style  
 b. 如果需要批量修改一个元素的多个css属性值时最好采用class

2. 如果class需要根据程序中的变量自动变化

(1). 不好的做法: 将整个class字符串看做一个变量来绑定

(2). 问题: 极其不便于只修改其中某一个class！

(3). 好的做法: 用对象语法: 2种:

a. 为每个class都分别指定一个变量:

1). 如何: **<元素 :class="{ class名1:变量名1, class名2:变量名2, ... }"**

**data:{**

**变量名1: 必须bool值true或false,**

**变量名2: 必须bool值true或false,**

**... : ...**

**}**

2). 强调: 绑定语法中class后的变量值，必须是bool类型的true或false

i. 如果值为true，则该class才会出现在最终的元素上，起作用

ii. 如果值为false，则该class不会出现在最终的元素上，不起作用

3). 问题: 如果多个元素都需要用success和fail作为验证结果的class名，则变量之间一定会产生冲突

4). 解决: 将一个元素用的所有class集中保存在一个变量内

b. 为整个class属性只指定一个**统一**的变量

1). 如何: **<元素 :class="变量名">**

**data:{ "class名1 ..."  
 变量名:{**

**class名1:true,**

**class名2: false**

**... ...**

**}**

**}**

2). 好处: 避免多个元素使用相同class名时发生变量冲突

(4). 问题: 如果一个元素上有些class是不变的，有些class是动态变化的，怎么办？

(5). 其实: class也支持**不带:写死的class与带:动态绑定的class在一个元素中混搭**。最终运行时不带：的class和带:的class会合并为一个class作用于元素上.

**<元素 class="固定不变的class" :class="动态变化的class">**

(6). 示例: 使用class实现手机号格式验证

a. 为每个class分别起一个变量名:

11\_class.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>      .errMsg{        padding:5px 10px;      }      .success{        background-color:lightGreen;        border:1px solid green;        color:green;      }      .fail{        background-color:pink;        border:1px solid red;        color:red;      }    </style>  </head>  <body>    <!--1. 先做界面-->    <div id="app">      <!--希望一边输入一边验证手机号的格式是否正确，根据验证结果修改span的class      errMsg作为基础样式的class常驻span元素上,必须要变量      success和fail两个class根据验证结果不同动态变化,所以需要两个变量success和fail      用户在文本框中输入手机号时，程序中能随时接住用户输入的手机号！所以需要一个双向绑定的变量phone      用户一边输入，一边验证，所以需要绑定input的@input事件处理函数名为vali      span的内容也可能根据验证结果的不同动态变化，所以也需要一个变量msg-->      <!-- <input v-model:value="phone"> -->      <input v-model="phone" @input="vali">      <span class="errMsg" :class="{success:success, fail:fail}">{{msg}}</span>    </div>    <script>      //2. 创建new Vue()对象监视id为app的区域      new Vue({        el:"#app",        //3. 创建模型对象保存界面所需的所有变量和事件处理函数        //因为页面上需要4个变量+1个函数        data:{          phone:"", //准备接页面上传入的手机号          success:false, //开局没验证时既不成功也不失败          fail:false,          msg:"" //保存将来可能的错误提示信息，但是开局，未验证之前，没有任何提示        },        methods:{          vali(){            //定义正则表达式:            //      JS高级第一天            var reg=/^1[3-9]\d{9}$/;            //**用正则验证phone变量中的手机号**            var result=reg.test(this.phone);            //如果验证通过就修改span的class为启用sucess，禁用fail，同时修改span的内容msg为手机号格式正确            if(result==true){              this.success=true;              this.fail=false;              this.msg="手机号格式正确"            }else{              //否则如果验证未通过就修改span的class为禁用success启用fail，同时修改span的内容为手机号格式不正确              this.success=false;              this.fail=true;              this.msg="手机号格式不正确"            }          }        }      })    </script>  </body>  </html>  运行结果: |

b. 为一个元素的整个class属性只起一个变量名:

12\_class2.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>      .errMsg{        padding:5px 10px;      }      .success{        background-color:lightGreen;        border:1px solid green;        color:green;      }      .fail{        background-color:pink;        border:1px solid red;        color:red;      }    </style>  </head>  <body>    <!--1. 先做界面-->    <div id="app">      <!--希望一边输入一边验证手机号的格式是否正确，根据验证结果修改span的class      errMsg作为基础样式的class常驻span元素上,必须要变量      success和fail两个class根据验证结果不同动态变化,但是考虑到其它元素也可能使用success和fail当做样式类名字动态变化，为了避免样式冲突，应该将success和fail两个变量放在一个统一的变量内。      用户在文本框中输入手机号时，程序中能随时接住用户输入的手机号！所以需要一个双向绑定的变量phone      用户一边输入，一边验证，所以需要绑定input的@input事件处理函数名为vali      span的内容也可能根据验证结果的不同动态变化，所以也需要一个变量msg-->      <!-- <input v-model:value="phone"> -->      <input v-model="phone" @input="vali">      <span class="errMsg" :class="spanClass">{{msg}}</span>    </div>    <script>      //2. 创建new Vue()对象监视id为app的区域      new Vue({        el:"#app",        //3. 创建模型对象保存界面所需的所有变量和事件处理函数        //虽然页面上需要3个变量+1个函数        //但是spanClass变量中却包含两个class名的定义        data:{          phone:"", //准备接页面上传入的手机号          spanClass:{            success:false,            fail:false          },          msg:"" //保存将来可能的错误提示信息，但是开局，未验证之前，没有任何提示        },        methods:{          vali(){            //定义正则表达式:            //      JS高级第一天            var reg=/^1[3-9]\d{9}$/;            //用正则验证phone变量中的手机号            var result=reg.test(this.phone);            //如果验证通过就修改span的class为启用sucess，禁用fail，同时修改span的内容msg为手机号格式正确            if(result==true){              // this.spanClass.success=true;              // this.spanClass.fail=false;              this.spanClass={                success:true,                fail:false              }              this.msg="手机号格式正确"            }else{              //否则如果验证未通过就修改span的class为禁用success启用fail，同时修改span的内容为手机号格式不正确              this.spanClass={                success:false,                fail:true              }              this.msg="手机号格式不正确"            }          }        }      })    </script>  </body>  </html>  运行结果: |

四. **自定义指令**

对应小程序视频: 小程序->在线->VUE->day02 9. 自定义指令vue directive

1. 如果vue官方提供的13种指令不够用，也可以自己添加自定义指令

2. 何时: 如果希望在首次加载时对一个元素执行一些默认的DOM操作

3. 如何向vue中添加自定义指令:

//给vue大家庭中添加一个新的自定义指令

**Vue.directive("指令名",{**

//希望凡是带有这个指令的元素，在被加载到页面时能自动执行一个操作，就会自动调用inserted函数。

**inserted(domElem){** //回调函数

//放在inserted中的代码，注定会在元素被加载到页面上之后自动执行

//形参变量domElem，自动接收当前带有自定义执行的DOM元素对象

//就可以对当前带有自定义指令的元素，名正言顺的执行DOM操作

**}**

**})**

强调: **定义**自定义指令时，指令名不**要加v-**前缀！

4. 如何**使用**自定义指令:

**<元素 v-自定义指令名>**

强调: 在使用自定义指令时，必须**加v-**前缀!

5. 结果:

(1). 当new VUe()扫描到带有v-自定义指令的元素后，就会去vue大家庭中找是否包含该名称的自定义指令。

(2). 只要找到当前名称对应的指令，就自动调用指令对象内的inserted函数，并自动传入当前DOM元素对象作为实参值

(3). 在inserted回调函数内可以对当前DOM元素执行DOM操作

6. 示例: 页面加载时，让一个元素自动获得焦点:

13\_directive.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>    <script>      //先定义一个可让当前元素自动获得焦点的自定义指令，名为myfocus      Vue.directive("myfocus",{        inserted(domElem){          //只要让当前元素对象自动获得焦点          domElem.focus();//DOM第三天 HTML DOM form        }      })    </script>  </head>  <body>    <div id="app">      <!--希望在页面加载时，当前文本框自动获得焦点-->      <input v-myfocus>    </div>    <script>      new Vue({        el:"#app"      })    </script>  </body>  </html>  运行结果: |

总结:

1. 绑定语法+13种指令

(1). 如果元素的内容需要随变量自动变化: {{}}

(2). 如果元素的属性值需要随变量自动变化: :

(3). 控制一个元素显示隐藏: v-show

(4). 控制两个元素二选一显示: v-if v-else

(5). 多个元素多选一显示: v-if v-else-if v-else

(6). 只要反复生成多个相同结构的元素组成列表时: v-for :key="唯一标识"

(7). 只要绑定事件: @ $event

(8). 防止用户短暂看到{{}}: v-cloak和v-text

(9). 只要绑定原始HTML代码片段内容: v-html

(10). 如果元素的内容只在首次加载时绑定一次，之后都不会改变: v-once

(11). 保护内容中的{{}}不被编译: v-pre

(12). 今后只要想获得表单元素的值或状态: v-model

2. 绑定样式:

(1). 需要精确修改某一个css属性，就绑定style:

a. <元素 style="固定样式" :style="{css属性:变量名, ...}"

data:{  
 变量名:css属性值

... : ...

}

b. <元素 style="固定样式" :style="变量名"

data:{  
 变量名:{

css属性名: 属性值,

... : ...

}

}

(2). 只要批量修改一个元素的多个css属性就绑定class

a. <元素 class="固定class" :class="{class名:变量名, ...}"

data:{  
 变量名:true或false,

... : ...

}  
b. <元素 class="固定class" :class="变量名"

data:{  
 变量名:{  
 class名:true或false,

... : ...

}

}

3. 只要希望在页面加载时自动对元素执行一些初始化操作时就用自定义指令:

(1). 添加自定义指令:

Vue.directive("自定义指令名",{

inserted(domElem){  
 对domElem执行DOM操作

}

})

(2). 使用自定义指令:

<元素 v-自定义指令名>

今日对应小程序视频:

小程序->在线->VUE->day02

2. 其它指令 绑定HTML内容 v-html

3. 双向绑定 v-model 绑定文本框...

4. 双向绑定 v-mode radio ...

5. 双向绑定 v-model select ...

6. 双向绑定 v-model checkbox ...

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->day02、day03

2. 预习:

小程序->在线->VUE->day02 10 计算属性

小程序->在线->VUE->day03 3. 过滤器

4. axios

5. 生命周期

6.1 创建单个组件

6.2 vue组件化开发

7. 组件间传参

下程序->在线->vue->day04 1. SPA单页面应用