我为大家准备了: 微信小程序——“WEB问题速查”——每天必须跟着看

1. 历届师兄师姐的问题清单: 小程序->首页->VUE->day02~day03 点击右侧列表中的问题，就可弹出窗口，查看问题的详情

2. 当天重点视频片段: 小程序->在线->VUE -> day02~day03

3. 当天的单词列表: 小程序->单词-> VUE -> day02~day03

如果tmooc上的视频有问题看不了，可以直接看小程序中的视频

小程序中内容到鄙视面试前看完就行！不用现在都看完！

关于提问: (剩3天)直接私信我是最快的解决途径——谁问的问题越多，谁的培训费交的越值

1. 上课时哪个问题没听明白，立刻问

2. 笔记中哪句话/哪个图不明白，立刻问

3. 小程序中问题清单里哪个问题看不懂，立刻问

4. 如果个人项目中功能不会做，立刻问

5. 刷笔试题时，遇到不会的笔试题，立刻问

约定: 每个重要的案例，至少抄三遍，直到自己可以写出注释才算自己会。（个别代码写不出来，都没关系！）

正课:

1. 双向绑定

2. 绑定样式

3. 自定义指令

4. 计算属性

5. 过滤器

一. 双向绑定:

1. 问题: 使用之前的12种指令，的确可以将new Vue()中的变量值更新到页面显示。但是都无法将页面中用户输入的新内容，自动反向更新会new Vue()中保存。

2. 单向绑定: 只能将new Vue()中模型对象中变量值的修改更新到页面上显示。如果页面中元素内容发生变化，无法自动将更改更新回new Vue()中的模型对象中保存

之前的12种指令，都只是单项绑定而已。

3. 解决: 今后只要希望自动获得用户在界面上输入或选择的新值，都只能用双向绑定

4. 双向绑定: 既能将new Vue()中模型对象中变量值的修改更新到页面上显示。又能将页面中元素内容发生的变化，自动更新回new Vue()中的模型对象中保存

5. 如何: <表单元素 v-model~~:可能变化的属性~~="变量">

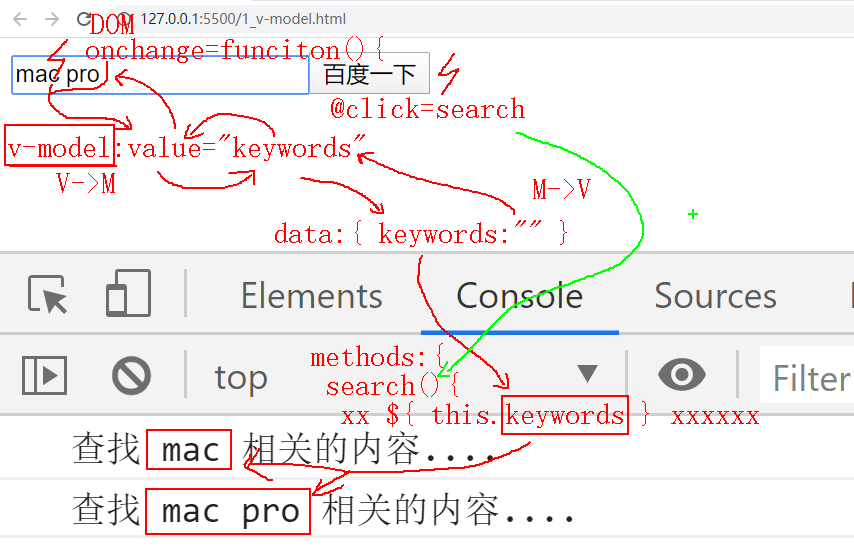
6. 强调:双向绑定几乎只用在表单元素上

7. 原理: v-model包含两部分原理

(1). 从Model->View的绑定与之前讲解的绑定元素一样: 访问器属性+虚拟DOM树

(2). 但是从View->Model的绑定是新的原理：new Vue()扫描到包含v-model的表单元素时，就自动为表单元素绑定onchange=function(){}事件处理函数。意为当表单元素的值发生改变时，就自动触发。然后，在onchange事件处理函数内，包含自动获取当前元素值并更新回Vue中指定变量中保存的语句。

8. 示例: 根据文本框中输入的关键词搜索相关内容



|  |
| --- |
| <!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>    <!--想点百度一下，可按关键词搜索-->    <!--1. 先做界面:      问题: 哪里可能发生变化      本例中: 当用户在文本框中输入新内容时，input的value可能发生变化，希望new Vue()中自动获得用户输入的新值，所以用该用v-model:绑定。      问题: 哪个元素会触发事件      本例中: 点击button执行查找操作            用户在文本框上按回车，也能执行和按钮的click一样的查找操作！-->    <div id="app">      <input type="text" @keyup="search2" v-model~~:value~~="keywords"><button @click="search">百度一下</button>    </div>    <script>      //2. 定义new Vue()对象      new Vue({        el:"#app",        //3. 定义模型对象        //问题: 页面中共需要几个变量        //本例中: 页面中只需要一个变量keywords，且如果用户未输入时，keywords的默认值为""        //问题: 页面中共需要几个事件处理函数        //本例中: 按钮需要search函数        //       文本框需要search2函数        data:{          keywords:""        },        methods:{          search(){            //如果用户输入的关键词去空格后不为空            //if(this.keywords.trim()!==""){              //才用关键词执行查找操作              console.log(`查找 ${this.keywords} 相关的内容....`)            //}          },          search2(e){            //只有点在回车上，才执行search操作            if(e.keyCode==13){              this.search();            }          }        }      })    </script>  </body>  </html>  运行效果： |

9. 问题: 不是所有表单元素修改时，更新的都是value属性。比如radio和checkbox元素更新时，更新的就不是value属性。而是value属性写死固定不变，而更新checked选中状态属性。

10. 解决: 用户修改表单元素时，哪个属性变化，就用v-model绑定哪个属性。比如: 如果value属性值可能发生变化，就用v-model:value="变量"。再比如: 如果checked属性可能发生变化，就用v-model:checked="变量"。

11. 其他类型的表单元素如何双向绑定:

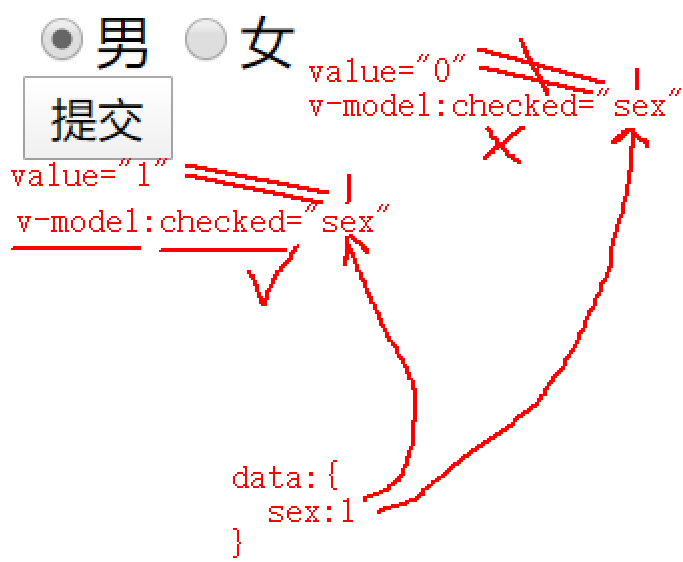
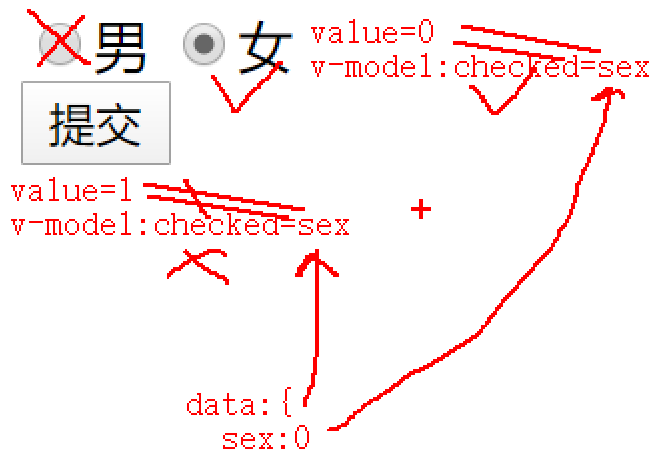
(1). 单选按钮: <input type="radio">

a. 当用户选中与不选中时，修改的是checked属性，所以如果双向绑定时应该用v-model:checked="变量"。

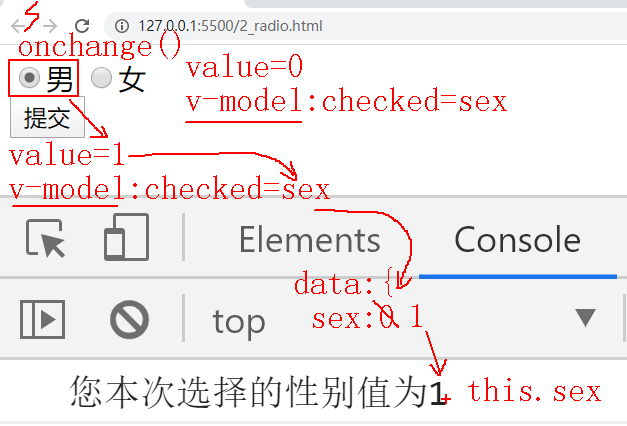
b. 如何: <input type="radio" value="写死的值" v-model~~:checked~~="变量">

c. 原理:

1). 从Model->View绑定时，v-model先自动用变量的值和写死的value值做比较。如果变量值==value值，则当前radio选中。否则如果变量值!=value值，则当前radio不选中。

2). 从View->Model绑定时，当用户切换选中状态时，自动触发onchange事件，onchange事件中只将选中的radio的值自动更新回v-model绑定的变量中。



d. 示例: 切换选择性别

|  |
| --- |
| <!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>    <!--请用户选择性别-->    <!--1. 先做界面      问题: 界面中哪里可能发生变化      本例中: 随着用户切换选择性别，两个input的checked属性可能发生变化又因为是表单元素      所以: 两个input都用v-model:checked="变量"      问题: 哪个元素可能触发事件      本例中: 点按钮输出用户选择的性别值-->    <div id="app">      <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>      <br/>      <button @click="submit">提交</button>    </div>    <script>      //2. 创建new Vue()对象      new Vue({        el:"#app",        //3. 定义模型对象        //问题: 界面上一共需要几个变量        //本例中: 界面上只需要一个变量sex        //问题: 界面上共需要几个事件处理函数        //本例中: 只需要一个事件处理函数submit        data:{          sex:0 //假设初始时选择0        },        methods:{          submit(){            console.log(`您本次选择的性别值为${this.sex}`)          }        }      })    </script>  </body>  </html>  运行效果: |

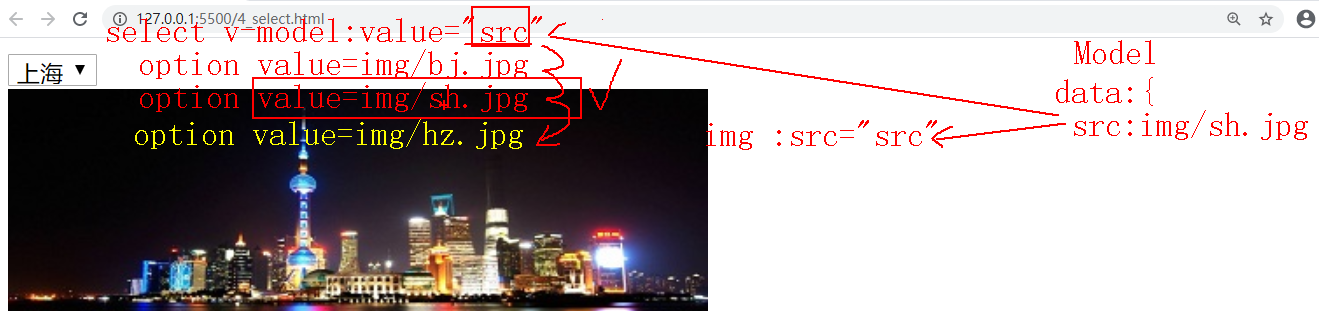
(2). 下拉选择框: <select>...</select>

a. 当用户在select内选择不同的option时，其实改的是整个select的value属性。

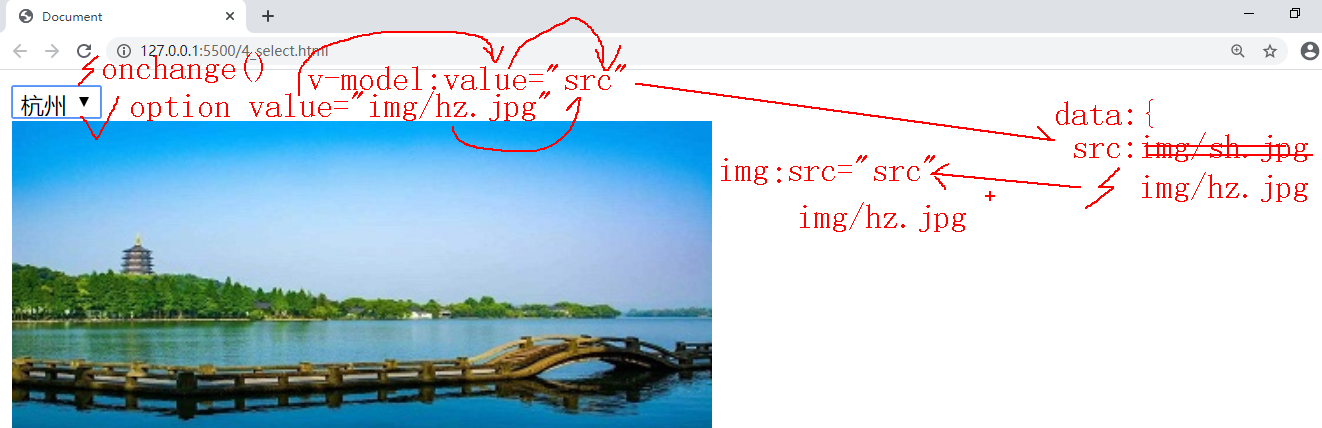
b. 如何: <select v-model~~:value~~="变量">

c. 原理:

1). 从Model->View绑定时，v-model将变量值自动和每个option的value比较。哪个option的value值与变量值相等，就自动选中这个option



2). 从View->Model绑定时，当用户选中一个option时，option的value先交给select，成为整个select的value，然后v-model再将select的新value，自动更新回data中的src变量中保存。



d. 示例: 选择城市，查看城市图片

|  |
| --- |
| <!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>    <!--用户选择城市，切换不同的城市图片-->    <!--1. 做界面      问题: 整个界面中共有几处可能发生变化？      本例中: select元素的value属性值和img的src属性值，两处可能发生变化      又因为: select元素是由用户手动选择的，而img的src是根据select的选择被动接受的。所以select元素采用双向绑定，img的src采用单项绑定。      规律: select元素选中的value值，就是img的src值，所以，select的value和img的src可共用一个变量-->    <div id="app">      <select v-model~~:value~~="src">        <option value="img/bj.jpg">北京</option>        <option value="img/sh.jpg">上海</option>        <option value="img/hz.jpg">杭州</option>      </select><br/>      <img :src="src">    </div>    <script>      //2. 创建new Vue()对象      new Vue({        el:"#app",        //3. 创建模型对象:        //问题: 界面上共需要几个变量        //本例中只需要一个变量就可同时控制select和img。        data:{          src:"img/sh.jpg",//开局默认选中上海        }      })    </script>  </body>  </html>  运行效果: |

(3). 复选框单独使用: <input type="checkbox">

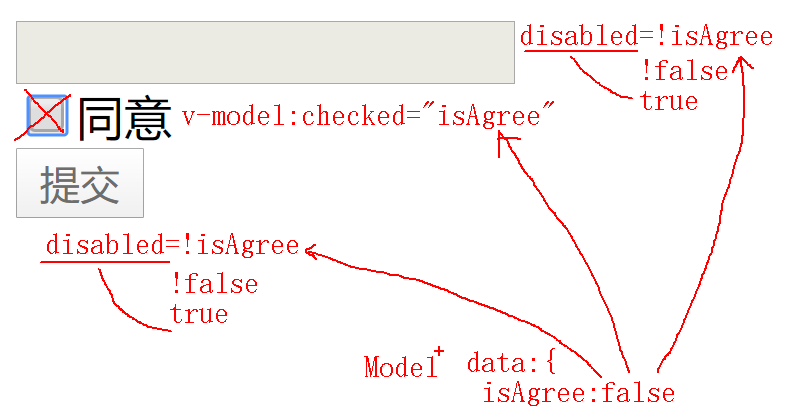
a. 用户选中或不选中checkbox时，更改的是checked属性

b. 如何: <input type="checkbox" v-model~~:checked~~="变量">

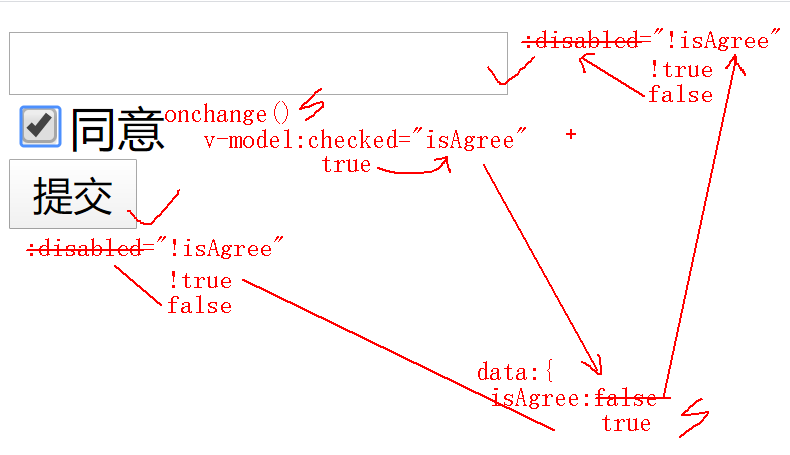
c. 强调: 不需要写value，且checked值应该是一个bool值！true就选中，false就不选中

d. 原理:

1). 从Model->View绑定，变量值为true，就选中；变量值为false，就不选中



2). 从View->Model绑定，用户选中checkbox，就将true传到vue中变量里保存。如果用户未选中checkbox，就将false传到vue中变量里保存。



e. 示例: 点同意，启用表单元素

|  |
| --- |
| <!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>    <!--点同意才启用其他表单元素-->    <!--1. 先做界面      问题: 点击checkbox时，界面中共有几处变化      本例中: checkbox自己的选中状态会改变，input和button的disabled属性值会改变      因为: checkbox是由用户手动切换选中状态，应该用双向绑定      但是: input和button的disabled，是被动接受启用和禁用状态。用户并不是直接修改他们。应该用单向绑定      规律: 如果checkbox选中，则input和button都启用，disabled为false才是启用。如果checkbox未选中，则input和button都禁用，disabled为true才是禁用。所以，checkbox的checked属性值应该和disabled属性值取反即可！所以，只需要1个变量即可控制三个元素的状态-->    <div id="app">      <!--              未 同意时才是禁用-->      <input :disabled="!isAgree"><br/>      <label><input type="checkbox" v-model~~:checked~~="isAgree">同意</label><br/>      <!--              未 同意时才是禁用-->      <button :disabled="!isAgree">提交</button>    </div>    <script>      //2. 创建new Vue()      new Vue({        el:"#app",        //3. 创建模型对象        //问题:整个页面只需要1个变量就可控制所有元素状态        data:{          isAgree:false //开局默认暂时是不同意。        }      })    </script>  </body>  </html>  运行效果: |

12. 简写: 其实以上例子中v-model后的~~:value和:checked全都可以省略~~！

今后双向绑定只写<表单元素 v-model="变量">

因为v-model会自定区分该绑定哪个元素！不用人为区分！

说明: select和checkbox都可以多选，但是多选非常麻烦，基础好的同学可以看小程序中扩展视频学习。——有难度，量力而为！

二. 绑定样式：

1. 绑定内联样式(style属性):

(1). 何时：只修改个别css属性值时，首选修改内联样式:

(2). 比如: 做游戏时，控制主角的方向: left top等

控制一个元素的显示隐藏: display:none

(3). 如何:

a. 不好的方法: 因为style属性值也是一个巨大的字符串，所以完全可以当做一个普通的字符串属性来绑定:

1). HTML中:<元素 :style="变量">

2). new Vue()中:

data:{

变量: "css属性1: 值1; css属性2:值2;..."

}

3). 问题: 因为多个css属性混在一个字符串中，无法对一个css属性值随意加减。不便于修改style中某一个css属性。

b. 好的做法: 将整个style属性，看作一个对象来绑定:

1). HTML中: <元素 :style="变量">

"css属性:值; css属性:值;..."

自动翻译为

2). new Vue({

data:{

变量:{

Css属性:值,

Css属性:值,

... : ...

}

}

})

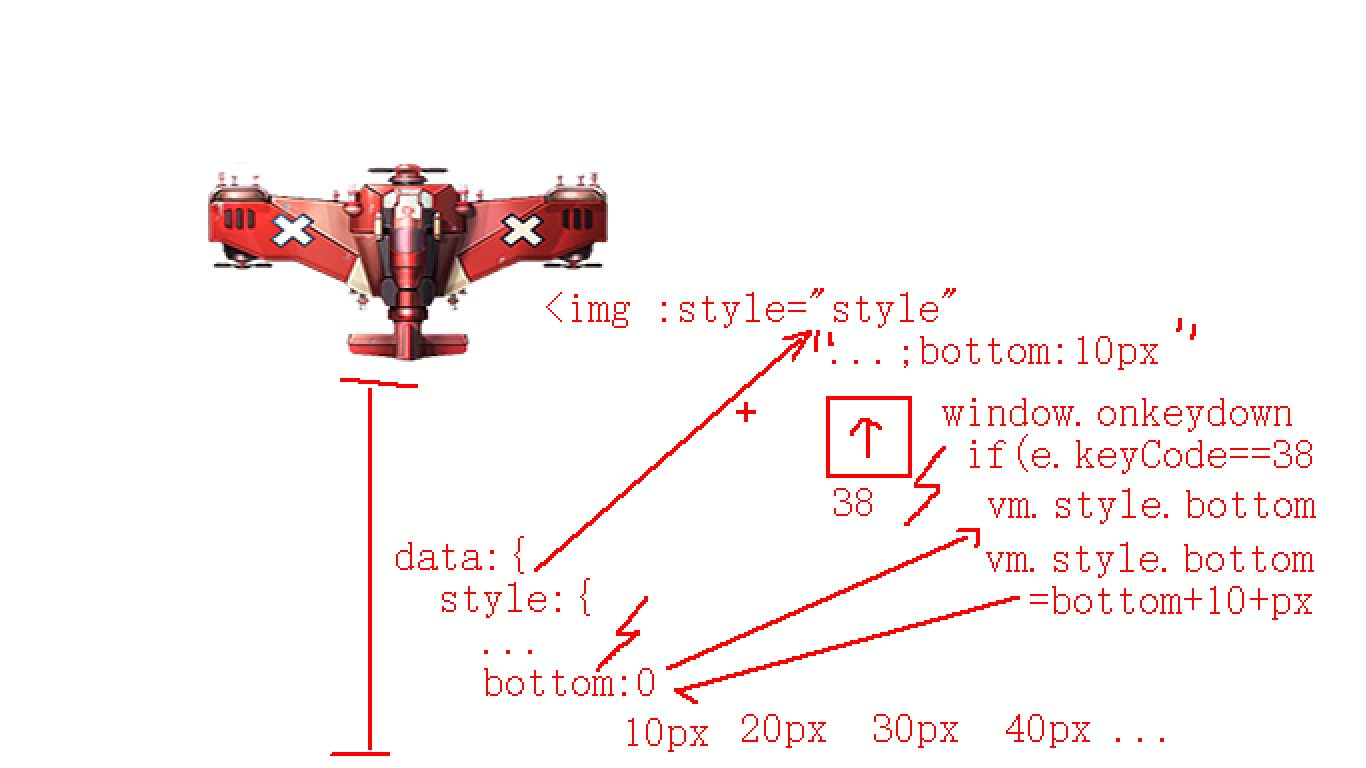
3). 优点: 非常便于只修改其中某一个css属性！

vm.变量.css属性值="xxx" //new Vue()之外修改时

或

this.变量.css属性值="xxx" //new Vue()内部修改时

(4). 示例: 绑定style控制飞机飞行



|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>    <style>      img{        position:fixed;        left:50%;        bottom:0;      }    </style>    <script src="js/vue.js"></script>  </head>  <body>    <div id="app" >      <img :style="style" src="img/p3.png" alt="">    </div>    <script>      var vm=new Vue({        el:"#app",        data:{          // 不好: style:"margin-left:-100px;bottom:0"          //                   ↑          style:{//可自动翻译为 |            marginLeft:"-100px",//必须带单位！            bottom:0 //只有0特殊，可以不加单位          }        }      })      //当在窗口中按下键盘时      window.onkeydown=function(e){        //如果按的是上键,则bottom+10        if(e.keyCode==38){          //vm.style //{ marginLeft:"-100px", bottom:0 }          //只取出vm对象中style属性中bottom的值，转为整数          var bottom=parseInt(vm.style.bottom);          //将bottom+10，再保存回vm.style.bottom中，记得加单位          vm.style.bottom=bottom+10+"px";        }else if(e.keyCode==40){//否则如果按的是下键,则bottom-10          //只取出vm对象中style属性中bottom值，转为整数          var bottom=parseInt(vm.style.bottom);          //将bottom-10，再保存回vm.style.bottom中，记得加单位          vm.style.bottom=bottom-10+"px";        }else if(e.keyCode==37){//否则如果按的是左键,则marginLeft-10          //只取出vm对象中style属性中marginLeft值，转为整数          var marginLeft=parseInt(vm.style.marginLeft);          //将marginLeft-10，再保存回vm.style.marginLeft中，记得加单位          vm.style.marginLeft=marginLeft-10+"px";        }else if(e.keyCode==39){//否则如果按的是右键,则marginLeft+10          //只取出vm对象中style属性中marginLeft值，转为整数          var marginLeft=parseInt(vm.style.marginLeft);          //将marginLeft+10，再保存回vm.style.marginLeft中，记得加单位          vm.style.marginLeft=marginLeft+10+"px";        }      }    </script>  </body>  </html>  运行效果： |

2. 绑定class:

(1). 何时: 批量修改多个css属性时，首选修改class属性

(2). 比如: 带样式的表单验证，提示信息在验证通过样式和验证失败样式之间来回切换，因为每种样式都包含多个css属性，所以用class批量修改是最划算的。

(3). 如何:

a. 不好的做法:

1). 将class属性也当做普通的字符串来绑定。

2). 如何:

i. HTML中: <元素 :class="变量">

ii. new Vue({中

data:{

变量: "class1 class2 class3 ..."

}

})

3). 缺点: 不便于只操作众多class中的一个

b. 好的做法:

1). 将class属性看作一个对象来绑定:

2). 如何:

i. HTML中: <元素 :class="变量">

"class1 ..."

ii. new Vue({中

data:{

变量:{//自动翻译为(只有值为true的class，才会翻译进最终的class字符串)

class1: true,

class2: false,

... : ...

}

}

})

c. 如果一个元素上既有固定不变的class，又有个别变化的class

1). HTML: <元素 class="固定不变的class" :class="变量">

2). new Vue({中

data:{

变量: {

可能变化的class: true或false,

... : ...

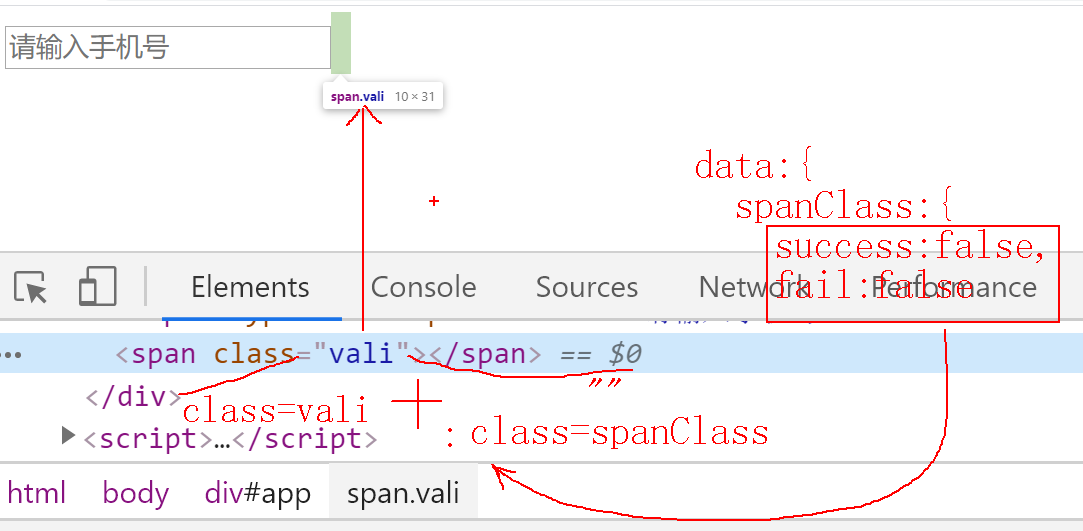
}

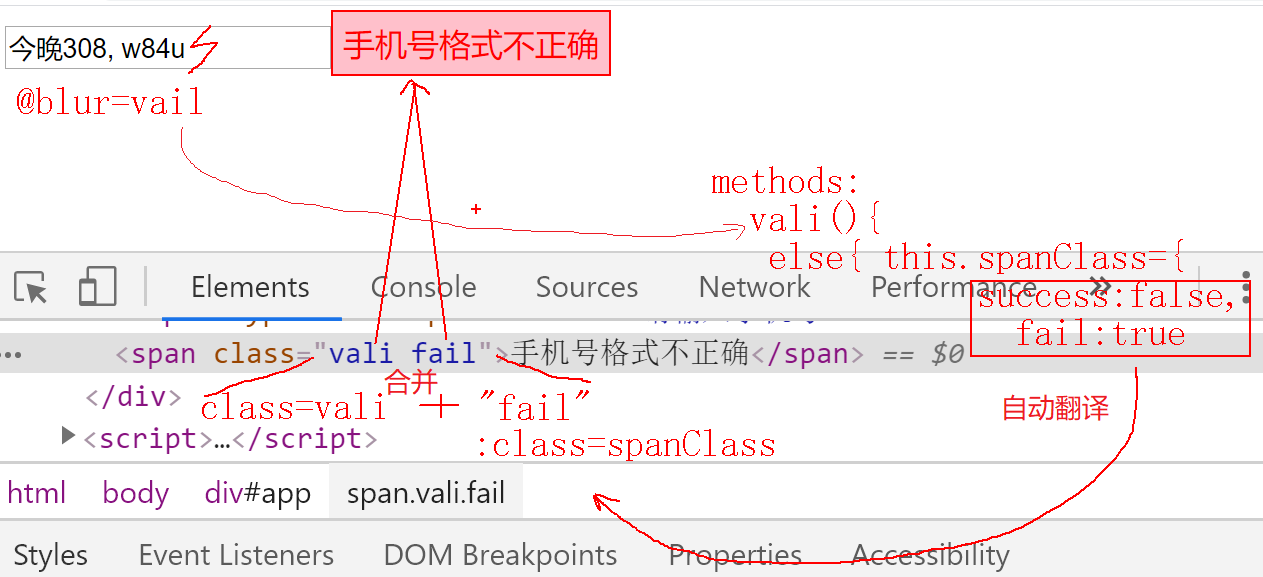
}

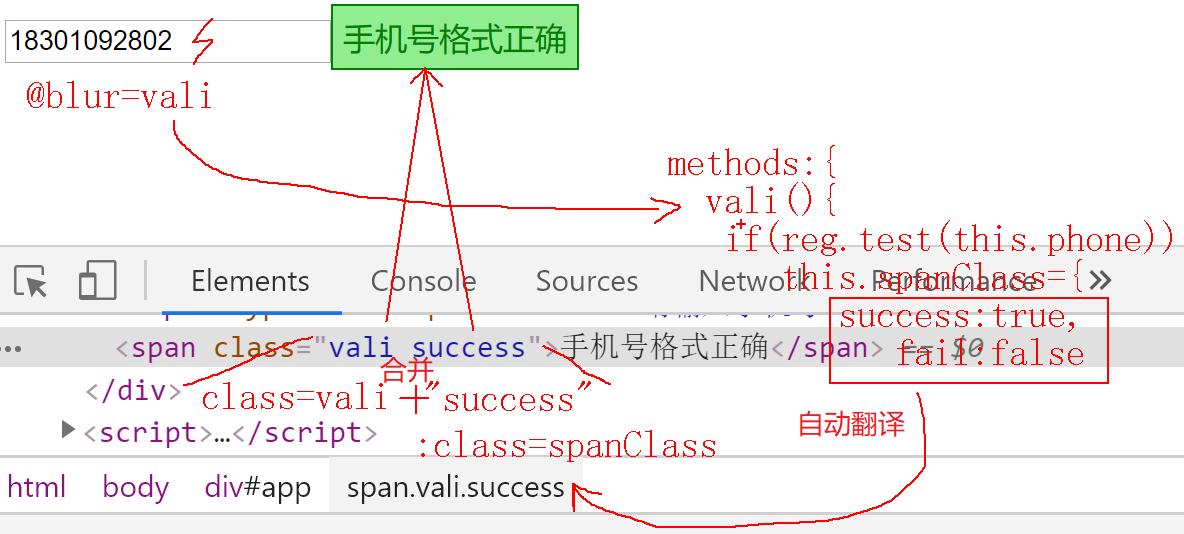
})

3). 结果: :class中值为true的class先编译为一个class字符串，然后再和不带:的class中固定的class字符串拼接在一起，形成最终的一个class，共同修饰这个元素。

(4). 示例: 带样式的手机号格式验证







|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>    <style>      .vali{/\*始终在span上保持不变\*/        padding:5px      }      .success{        border:1px solid green;        background-color:lightGreen;        color:green;      }      .fail{        border:1px solid red;        background-color:pink;        color:red;      }    </style>    <script src="js/vue.js"></script>  </head>  <body>    <!--文本框失去焦点时验证手机号格式是否正确-->    <!--1. 先做界面:      问题: 界面中共有几处可能发生变化      本例中:3处        (1). input文本框由用户手动输入内容        (2). span的class属性会随失去焦点验证的结果而改变class        (3). span的内容会随失去焦点验证结果而改变提示内容      所以: 界面中需要哪些变量:        (1). 因为input是用户手动输入，所以用v-model绑定        (2). 因为span的class属性是被动接受验证结果所以用:绑定。        (3). span的内容，是被动接受验证结果所以用{{}}      所以: 共需要三个变量:        phone用来接住用户输入的手机号        spanClass用来控制span的class属性        msg用来控制span的内容      问题: 界面中哪里可能触发事件      本例中: input失去焦点时会触发验证和变化-->    <div id="app">      <input type="text" placeholder="请输入手机号" v-model="phone" @blur="vali"><span class="vali" :class="spanClass">{{msg}}</span>    </div>    <script>      //2. 创建new Vue()对象      new Vue({        el:"#app",        //3. 创建模型对象：        //问题: 页面上共需要几个变量        //本例中: 需要三个变量        //问题: 页面上共需要几个事件处理函数        //本例中: 一个vali        data:{          phone: "", //开局用户未输入手机号时暂时为空          spanClass:{ //开局两个class应该都false,表示一个class都不用            success:false,            fail:false          },          msg:"", //开局用户未输入手机号时暂时为空        },        methods:{          vali(){            //定义验证手机号的正则表达式            var reg=/^1[3-9]\d{9}$/;            //如果phone接到的手机号格式正确            //因为test()返回值本来就是bool，所以不需要再写==true            if(reg.test(this.phone)){              //就修改spanClass中的success为true，fail为false              this.spanClass={success:true, fail:false};              //同时修改msg为"手机号格式正确"              this.msg="手机号格式正确";            }else{//否则phone接到的手机号格式不正确              //就修改spanClass中的success为false，fail为true              this.spanClass={success:false, fail:true};              //同时修改msg为"手机号格式不正确"              this.msg="手机号格式不正确";            }          }        }      })    </script>  </body>  </html>  运行结果： |

三. 自定义指令

1. 如果vue中13种指令不够你用的，你就可以给vue家添加自定义指令

2. 何时: 如果希望在首次加载时，对元素自动执行某些操作时，就可以用自定义指令：

比如: 让某个元素在页面加载时就自动获得焦点！

3. 如何: 2步

(1).向vue家添加一个新的指令，并起一个指令名

Vue.directive("指令名",{ //强调: 定义指令时，指令名不要加v-前缀！

//将来当带有这个指令的DOM元素被渲染到页面上之后自动执行inserted回调函数

inserted(dom元素){

//形参dom元素会自动获得当前拥有该指令的DOM元素对象

//对当前拥有该指令的DOM元素执行原生的DOM操作

}

})

(2). 在HTML界面中某个元素上使用自定义指令:

<元素 v-自定义指令名>

强调: 使用指令时，必须加v-前缀

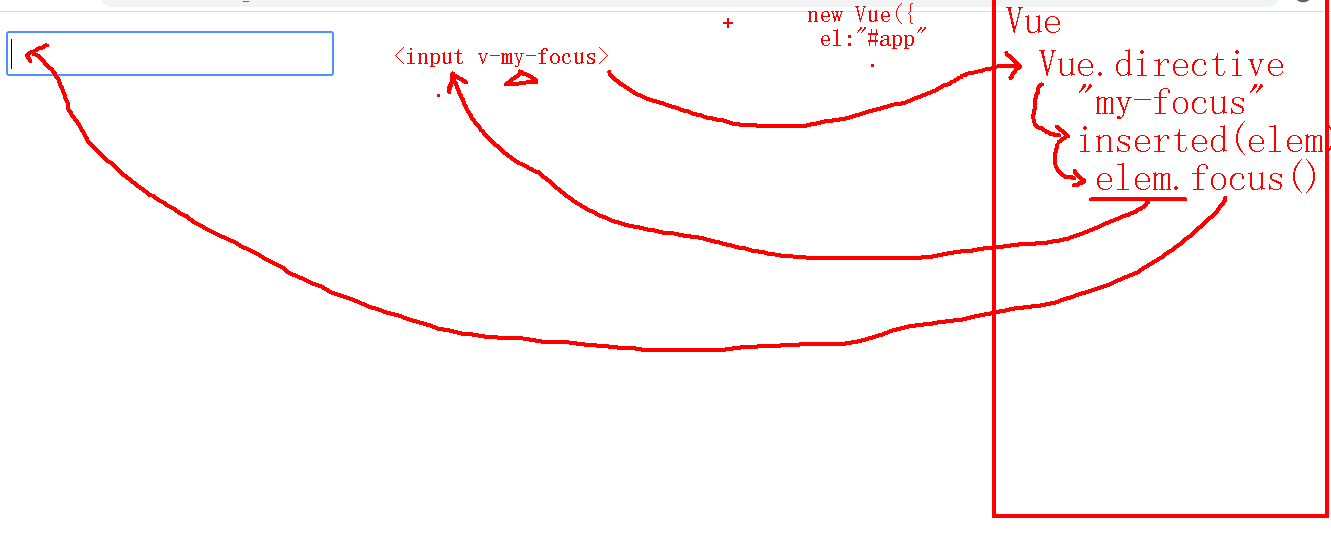
4. 原理:

(1). new Vue()扫描界面时，发现v-前缀的指令，都回去Vue家找，有没有对应的指令对象。

(2). 如果找到有对应名称的指令对象，就自动调用指令对象中的inserted()函数

(3). Inserted()函数中包含了对当前元素执行的初始化DOM操作！

5. 示例: 让元素在页面加载时就自动获得焦点



|  |
| --- |
| <!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>      //给Vue家添加一个新指令my-focus      Vue.directive("my-focus",{        //当拥有该指令的元素被渲染到页面上显示之后，自动执行inserted()函数        inserted(elem){          //每次: elem都自动获得当前带有这个指令的DOM元素对象          //本例中: 希望当前元素自动获得焦点          elem.focus();//原生DOM中的函数，复习DOM第三天        }      })    </script>  </head>  <body>    <!--想让一个文本框\*在页面加载时自动\*获得焦点-->    <div id="app">      <input v-my-focus>    </div>    <script>      //vue所有指令的执行，全靠new Vue()在扫描过程中发现。      //如果没有new Vue()，单靠浏览器是不认识任何指令的！就什么也不干      //所以即使new Vue()中只有一个el，也必须保留！不能删除！      new Vue({//整个vue程序的起点！        el:"#app"      })    </script>  </body>  </html>  运行效果： |

四. 计算属性

1. 什么是: 自己不保存属性值，需要根据其他属性值经过复杂的计算得到自己的属性值。

2. 为什么: 因为通常需要结果复杂计算得出的结果，都不会保存在数据库中，自然后端也不会给我们提供现成的值直接使用。只能靠我们在前端，利用其它属性的值计算获得。

比如: 购物车的总价: 计算过程复杂，又频繁发生变化的值，通常数据库里都不存。而是完全依靠客户端动态计算获得！

3. 何时: 只要界面上想要的值，后端没有直接提供，但是可以根据其他值计算获得，就用计算属性。

4. 如何: 2步

(1). 定义一个计算属性:

new Vue({

el:"#app",

data:{

可直接使用的属性

},

methods:{

事件处理函数

},

computed:{

计算属性名(){ //计算属性虽然用法和属性一样，但是本质却是一个函数！

//根据data中其他属性值，经过复杂的计算获得一个最终的属性值

return 计算结果

}

}

})

(2). 在HTML中使用计算属性:

比如: <元素>{{计算属性名}}</元素>

强调: 计算属性虽然本质是函数，但是使用时不要加()，和data中的普通属性完全一样

5. 结果: 运行时new Vue()会自动计算计算属性的结果，代替{{计算属性名}}的位置

6. computed vs methods：

(1). 用法:

a. methods中的函数在HTML中使用时必须加()调用

b. computed中的计算属性在HTML中使用时不用加()

(2). 效率: 当反复使用这个函数或计算属性时

a. methods中的函数，每加一次()，就重复执行一次！——没必要

b. computed中的计算属性结果，只在首次使用时计算一次，然后就被vue缓存起来。从第二次使用开始无需重复计算，就可直接使用缓存中的中。——效率更高！

7. 总结: 何时使用函数？何时使用计算属性？

(1). 如果更侧重于使用函数的执行结果数值时，首选计算属性

(2). 如果不关心执行结果，反而更侧重于执行一项操作（比如: 发送ajax请求），首选methods中的函数。

8. 示例: 计算购物车总价:

|  |
| --- |
| <!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>    <!--计算购物车总价-->    <div id="app">      <h3>总价: ¥{{total.toFixed(2)}}</h3>      <ul>        <!--小计: 因为计算简单所以不用计算属性-->        <li v-for="(p,i) of cart" :key="i">          {{i+1}} | {{p.pname}} | ¥{{p.price.toFixed(2)}} | {{p.count}} | 小计:¥{{(p.price\*p.count).toFixed(2)}}        </li>      </ul>      <!--总价因为计算过程复杂，所以应该用计算属性-->      <h3>总价: ¥{{total.toFixed(2)}}</h3>    </div>    <script>      new Vue({        el:"#app",        data:{          cart:[            {pname:"华为", price:5588, count:2},            {pname:"苹果", price:8588, count:1},            {pname:"小米", price:3588, count:3}          ]        },        methods:{          },        computed:{          //定义计算属性total，根据data中cart数组中的内容，计算出总价          total(){            console.log(`计算了一次总价`)            var result=0;            for(var p of this.cart){              result+=p.price\*p.count;            }            return result;          }        }      })    </script>  </body>  </html>  运行效果: |

五. 过滤器

1. 什么是过滤器: 专门对变量的原始值进行加工后再显示出来的一种特殊函数

2. 为什么: 有些时候，变量的原始值，不能直接给人看！人看不懂！所以需要先将原始值加工成人能看懂的内容，再显示出来

比如: 性别: 1和0，要翻译为男和女

再比如: 时间:毫秒数，要翻译为年/月/日 时:分:秒 人才能看懂

3. 何时: 今后要显示的变量原始值不能直接给人看，需要加工时，就用过滤器。

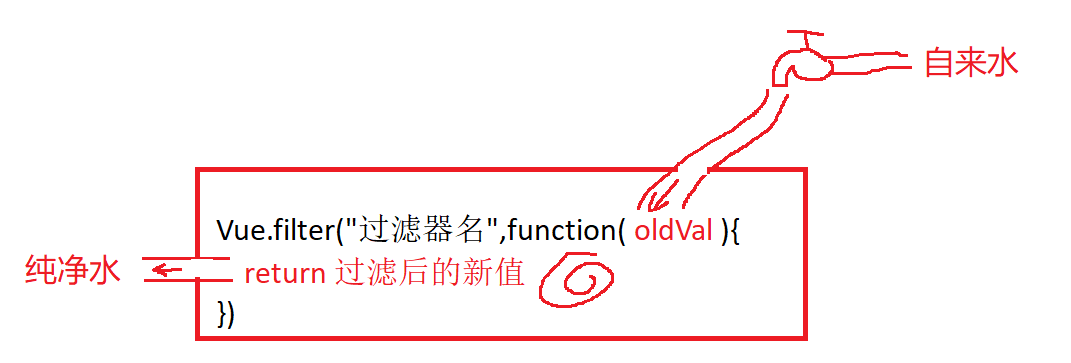
4. 如何: 2步

(1). 定义过滤器:

Vue.filter("过滤器名",function( oldVal ){

return 过滤后的新值

})



(2). 在页面上使用过滤器:

比如: 在绑定语法的{{}}中使用:

<元素>{{变量 | 过滤器名}}</元素>

连接

5. 结果:

(1). new Vue()扫描到绑定语法中的|，就会去Vue家找有没有指定名称的过滤器函数

(2). 如果找到指定名称的过滤器函数，则自动调用过滤器函数:

a. 将|左边的变量的原始值自动传给过滤器函数的oldVal形参

b. 将过滤器执行的结果返回给new Vue()

(3). new Vue()会用过滤器返回的加工后的结果，代替当前{{}}语法位置，显示给用户看。

6. 过滤器可以传参:

(1). 定义过滤器: 预留

Vue.filter("过滤器名",function( oldVal , 自定义形参){

return 根据不同的自定义形参，过滤出不同的新值

})

(2). 在页面上使用过滤器:

比如: 在绑定语法的{{}}中使用:

<元素>{{变量 | 过滤器名(自定义实参值)}}</元素>

连接

7. 多个过滤器可以连用：

<元素>{{变量 | 过滤器1 | 过滤器2 | ...}}</元素>

强调: 过滤器连用时，每个过滤器既可能接住变量的原始值，又可能接住的是上一个过滤器加工后的半成品。

8. 练习: 使用过滤器过滤性别

|  |
| --- |
| <!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>      //定义过滤器sexFilter可将性别的1和0过滤为男和女      //还可传入语种，翻译为不同语言的男和女      Vue.filter("sexFilter",function(oldVal,langage="cn"){        //如果{{sex | sexFilter }}        //则oldVal有2种可能: 0和1        //langage只考虑两种可能: cn 中文, en 英文，默认中文cn        if(langage=="cn"){          return oldVal==1?"男":"女"        }else{          return oldVal==1?"Male":"Female"        }      })      //再定义一个过滤器，为性别末尾追加一个性别图标      Vue.filter("sexIcon",function(oldVal){        if(oldVal==1||oldVal==0){          return oldVal==1?"♂":"♀"        }else if(oldVal=="男"||oldVal=="女"){          return oldVal=="男"?oldVal+"♂":oldVal+"♀"        }else{          return oldVal=="Male"?oldVal+"♂":oldVal+"♀"        }      })    </script>  </head>  <body>    <div id="app">      <h1>{{sex1 | sexFilter}}</h1>      <h1>{{sex2 | sexFilter}}</h1>      <h1>{{sex1 | sexFilter("en")}}</h1>      <h1>{{sex2 | sexFilter("en")}}</h1>      <h1>{{sex1 | sexIcon}}</h1>      <h1>{{sex2 | sexIcon}}</h1>      <h1>{{sex1 | sexFilter | sexIcon}}</h1>      <h1>{{sex2 | sexFilter | sexIcon}}</h1>      <h1>{{sex1 | sexFilter("en") | sexIcon}}</h1>      <h1>{{sex2 | sexFilter("en") | sexIcon}}</h1>    </div>    <script>      new Vue({        el:"#app",        data:{          sex1:1,          sex2:0        }      })    </script>  </body>  </html>  运行效果： |

总结:

1. MVVM:

(1). 界面(View) 增强版，支持变量、分支、循环等功能

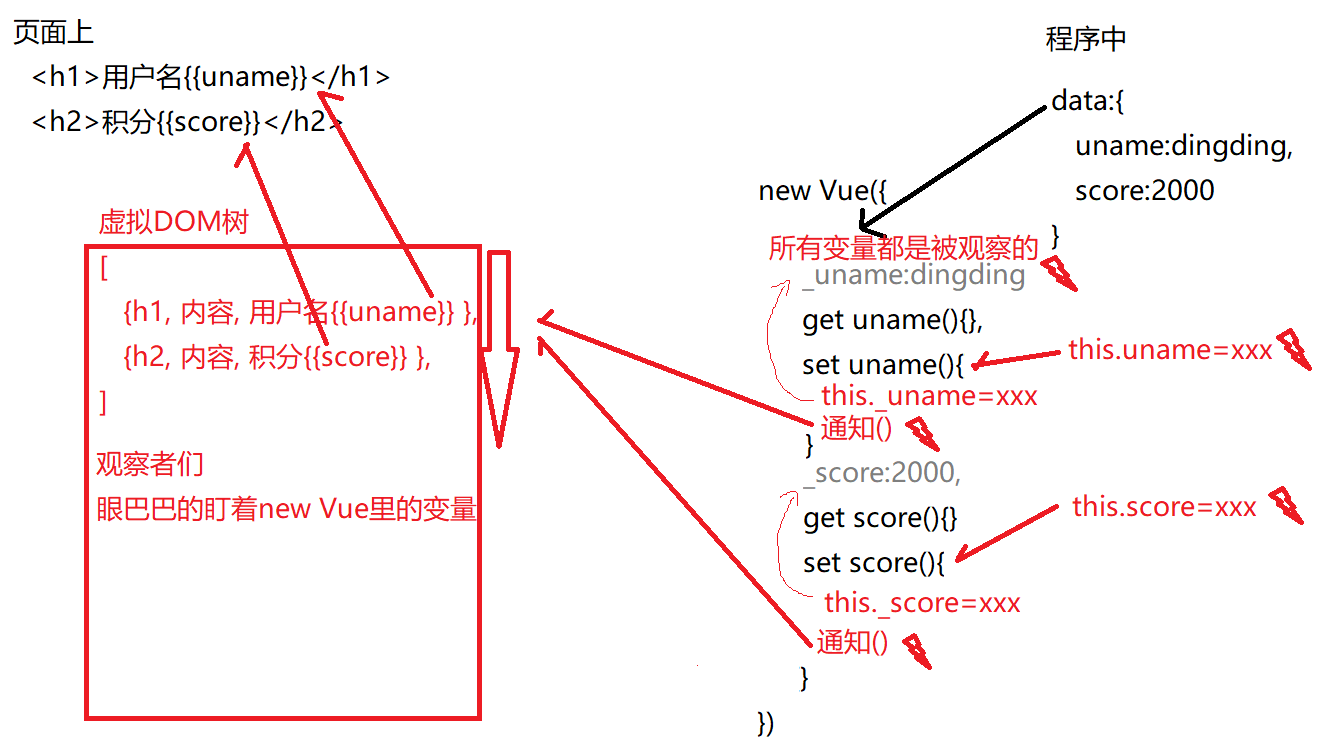
(2). 模型对象(Model) 保存界面所需的一切变量和函数:

(3). 视图模型(ViewModel) 快递员模型对象中的变量值和函数自动送到界面中所需的位置。并始终保持同步！

2. Vue绑定原理:

(1). 简单: 访问器属性+虚拟DOM树

(2). 完整: 访问器属性+观察者模式+新DOM树



3. 虚拟DOM树:

(1). 什么是: 仅保存可能变化的DOM元素的简化版DOM树  
 (2). 优点: 4个

a. 小: 仅保存可能变化的DOM元素

b. 快: 因为小，所以遍历快

c. 减少重复代码: 自动对受影响的元素执行增删改查操作

d. 更新效率高: 只更新受影响的元素。不受影响的元素，保持不变的！

4. Vue功能: 3步——招数

(1). 编写界面: 要求:

a. 整个界面必须包裹在一个唯一的父元素内，且父元素习惯上id="app"

b. 所有可能发生变化的地方，用{{变量名}}或指令方式标记

c. 所有触发事件的元素，用@事件名="函数名"标记

(2). 先创建new Vue()对象：

new Vue({

el:"#app", //选择器

... ...

})

(3). 为new Vue()添加data和methods模型对象

a. 界面中所需的所有变量都放在data对象中

b. 界面中所需的所有函数都放在methods对象中

c. methods中的函数，想操作data中的变量，必须加this.

5. 绑定语法和指令: ——招数

(1). 如果元素的内容可能随变量自动变化:

<元素>{{变量名}}</元素> 或 <元素 v-text="变量或js表达式"></元素>

(2).只要元素的属性值可能随变量自动变化: :属性名="js表达式"

(3). 控制一个元素显示隐藏: v-show="返回bool值的js表达式"

(4). 控制两个元素二选一显示: <元素1 v-if="返回bool值的js表达式">

<元素2 v-else>

(5). 控制多个元素多选一显示：<元素1 v-if="返回bool值的js表达式">

<元素2 v-else-if="返回bool值的js表达式">

... ...

<元素n v-else>

(6). 反复生成多个相同结构的元素：

<要反复生成的元素 v-for="(value,i) of 数组/对象/整数" :key="i">

(7). 绑定事件: <元素 @事件名="处理函数($event, 实参值)">

(8). 绑定原始HTML代码内容: <元素 v-html="变量"></元素>

(9). 防止用户短暂看到{{}}:

<元素 v-cloak>...</元素> 或 <元素 v-text="变量或js表达式"></元素>

(10). 今后只要一个绑定的值，在首次加载之后不再发生变化:

<元素 v-once>{{变量或js表达式}}</元素>

(11). 防止元素内容中的{{}}被new Vue()编译，保持{{}}原样显示:

<元素 v-pre>{{xxx}}</元素>

(12). 只要想获得用户在表单元素中输入的新值:

<元素 v-model ="变量名">

6. 绑定样式:

(1). 绑定内联样式:

<元素 style="固定不变的css属性" :style="变量">

new Vue({

data:{

变量: {

Css属性:"属性值",

... : ...

}

}

})

(2). 绑定class

<元素 class="固定不变的class" :class="变量">

new Vue({

data:{

变量: {

Class1: true或false,

... : ...

}

}

})

作业:

1. 复习今日问题清单(VUE->day02~day03)，预习明日问题清单(VUE->day03~day04)

2. 看小程序视频预习axios和生命周期

小程序->在线->VUE->day03 4. axios vue 发送ajax请求

5. 生命周期 钩子函数

3. 看小程序视频学习将绑定语法和指令应用到学子商城项目中:

小程序->在线->VUE->day01 作业: 使用VUE绑定学子商城首页和详情页部分内容...

下载public\_start文件夹，照着视频的做

视频中所有http://localhost:5050地址，全部改为我的新浪云服务器地址: http://xzserver.applinzi.com/接口名

更正视频中遗漏部分:

index.js中

... ...

new Vue({

... ...,

data:{

p1:{},

p2:{},

p3:{},

pOthers:result.slice(-3) //视频中缺少这句话

}

})

4. 看小程序视频学习如何用双向绑定选择飞机

小程序->在线->VUE->day02 4. 双向绑定v-model radio sex 选择性别 选择飞机 plans

5. (学有余力)看小程序视频学习vue如何实现多选

小程序->在线->VUE->day02 6. 双向绑定v-model checkbox 同意 启用禁用 多选 爱好