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

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

(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->day01

1.1 vscode部署vue网页模板code.snippet

作业: 为v-show和v-if添加过渡动画效果

作业: 使用vue实现标签页效果

高频笔试题: 观察者模式

基础比较弱！千万不要现在看！还是以课上内容为主。等到鄙视面试前再看！！

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

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5天）

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. 什么是VUE

2. 如何使用VUE

3. MVVM

4. 绑定语法

5. 指令

对应小程序视频: 小程序->在线->VUE->day01 1. vue和第一个vue程序

一. 什么是VUE:

1. 什么是: 是第三方开发的基于MVVM设计模式的渐进式的纯前端js框架

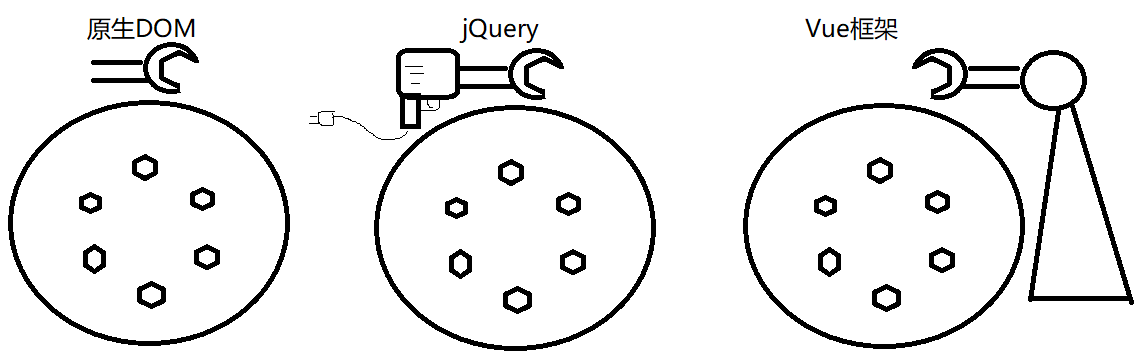
(1). 第三方开发: 下载才能用

(2). 基于MVVM: ?

(3). 渐进式: 可以在项目中逐步引入vue相关功能，很容易和其它技术混搭。

(4). 纯前端js: 不需要任何nodejs和后端的知识，单靠浏览器就可以运行和学习VUE

(5). 框架: 已经包含核心功能的半成品前端程序



2. 为什么: 简洁！避免大量重复的编码！

3. 何时: 今后只要以数据操作(增删改查)为主的项目，都可用vue开发

二. 如何使用VUE:

1. 2种:

(1). 将vue.js下载到项目本地引入网页中使用: —— 前3天 学vue基础知识和原理

a. 官网: cn.vuejs.org —— 中文！ —— 学有余力才能去看！

b. <script src="js/vue.js">

c. 问题: 因为前端项目越来越庞大，文件夹结构和代码量越来越复杂，导致不同的团队和公司组织文件和文件夹结构时，各不相同！——混乱

(2). 公司中都是用vue脚手架代码来开发项目: ——后2天 做学子商城项目

a. 什么是: 已经包含标准的文件夹结构和核心功能的半成品项目！

b. 优点: 标准！不同团队和不同公司发开出的项目结构几乎是完全相同的！

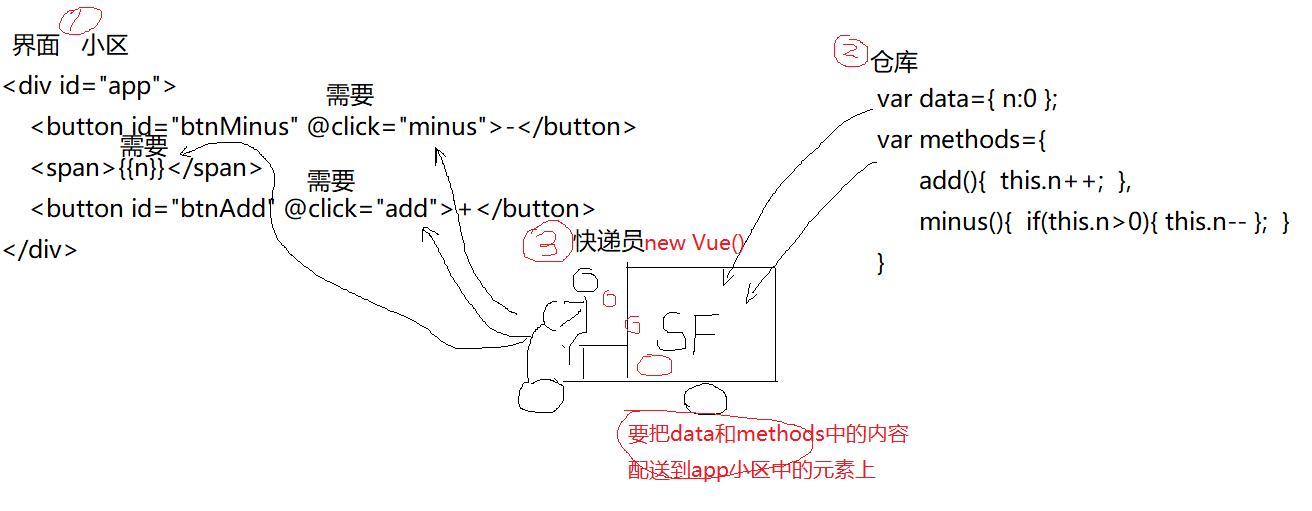
2. 示例: 我的第一个vue程序，分别用jquery和vue实现点击按钮修改数量功能

(1). jQuery版:

1\_first\_jquery.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>  </head>  <body>  <button id="btnMinus">-</button><span>0</span><button id="btnAdd">+</button>  <script src="js/jquery-1.11.3.js"></script>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 点两个按钮修改数量  var $btnMinus=$("#btnMinus");  var $btnAdd=$("#btnAdd");  //2. 绑定事件处理函数  //本例中: 单击两个按钮  $btnAdd.click(function(){  //3. 查找要修改的元素  //本例中: 要修改span  var $span=$("span");  //4. 修改元素  //先取出span中的内容，转为整数  var n=parseInt($span.html())  //再将span的内容+1  n++;  //最后将新值放回span内容中  $span.html(n);  });  //2. 绑定事件处理函数  //本例中: 单击两个按钮  $btnMinus.click(function(){  //3. 查找要修改的元素  //本例中: 要修改span  var $span=$("span");  //4. 修改元素  //先取出span中的内容，转为整数  var n=parseInt($span.html())  //如果数量>0时，才将span的内容-1  if(n>0){ n-- }  //最后将新值放回span内容中  $span.html(n);  });  </script>  </body>  </html> |

(2). VUE版



2\_first\_vue.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. 编写界面:  要求: 整个界面必须包裹在一个唯一的父元素内,习惯上: <div id="app"></div>  额外: 2件事:  1.1 找到界面中将来可能发生变化的位置，用{{自定义变量名}}临时占位  本例中: span的内容，将来随点击按钮而变化，所以<span>{{n}}</span>  1.2 找到界面中可能触发事件的元素，用@事件名="自定义处理函数名"标记出来  本例中: 单击btnMinus时触发减法操作，单击btnAdd时触发加法操作，所以  -->  <div id="app">  <button id="btnMinus" @click="minus">-</button>  <span>{{n}}</span>  <button id="btnAdd" @click="add">+</button>  </div>  <!-- <div id="#app2">  </div> -->  <script>  //2. 定义两个仓库对象: 仓库名必须叫data和methods  //2.1 一个仓库叫data，专门保存界面上所需的所有变量和初始值  //本例中: 界面上一共需要一个变量n，初始值为0  var data={ n:0 };  //2.2 另一个仓库叫methods, 专门保存界面上所需的所有事件处理函数  //本例中: 界面上共需要两个事件处理函数,分别是add和minus  var methods={  //复习小程序->在线->JSCORE->day07 作业:ES6关于对象中属性和方法的简写  add(){  //虽然现在事件处理函数和data中的n分着写，但是将来data对象和methods对象会被vue合并为一个对象，事件处理函数data中的变量最终会放在一个大的对象中保存，所以,methdos中的函数可通过this.xx方式操作data中的变量  //本例中: 点btnAdd，触发add函数，只想给data中的数量n+1  this.n++;  //不用自己放回去！  },  minus(){  //本例中: 点btnMinus，触发minus函数，如果data中的数量n>0，才将数量n-1  if(this.n>0){ this.n-- };  }  }  //3. 创建一个VUE对象充当快递员，自动送货  new Vue({  el:"#app", //选择器: 告诉快递员，请将货物送到id为app的小区中的各个住户手中！——这个快递元素所负责的小区  //告诉快递员，需要配送的变量和函数都放在哪些仓库里。  //复习小程序->在线->JSCORE->day07 作业:ES6关于对象中属性和方法的简写  data, //data:data,  methods //methods:methods  });  // new Vue({  // el:"#app2" //这个快递员专门负责app2小区的自动送货任务  // })  </script>  </body>  </html>  运行结果： |

3. 总结: Vue开发一个功能的基本步骤: 3步

(1). 定义界面:

a. 要求: 整个界面所有元素必须放在一个唯有的父元素下包裹

习惯上: <div id="app"> ... </div>

b. 找到界面内将来可能发生变化的位置，用{{自定义变量名}}临时占位

c. 找到界面中所有触发事件的元素，用@事件名="自定义处理函数名"标记

(2). 定义仓库对象: 2个 data和methods

a. data: 专门保存界面中所需的所有变量及其初始值

b. methods: 专门保存界面中所需的所有事件处理函数

因为将来data对象和methods对象会被new Vue()合并为一个对象，所以methods中的事件处理函数，和data中的变量最终会保存在同一个对象中。所以，methods中的事件处理函数可以通过this.变量名操作data中的变量

(3). 创建Vue对象，充当快递员:

new Vue({

el:"#app", //选择器: 告诉vue对象要将变量和函数送到页面中哪个大块区域中的元素上。

//告诉vue对象界面所需的一切变量和函数都保存在哪里——仓库位置

data, //data:data,

methods, //methods:methods,

})

4. 简写: 因为data对象和methods对象迟早都要被装进new Vue()对象中，所以实际开发中，我们不单独定义data和methods对象。而是直接在new Vue()中data属性和methods属性值对象中直接添加页面所需变量和事件处理函数:

new Vue({

el:"#app", //选择器: 告诉vue对象要将变量和函数送到页面中哪个大块区域中的元素上。

//告诉vue对象界面所需的一切变量和函数都保存在哪里——仓库位置

data:{

变量:初始值,

... : ...

},

methods:{

处理函数名(){

this.变量

},

...

}

})

5. 示例: 简写以上案例:

3\_first\_vue2.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. 编写界面: -->  <div id="app">  <button id="btnMinus" @click="minus">-</button>  <span>{{n}}</span>  <button id="btnAdd" @click="add">+</button>  </div>  <script>  //2. 创建一个VUE对象充当快递员，自动送货  new Vue({  el:"#app",  //3. 在Vue对象内添加data属性和methods属性，分别保存页面所需的所有变量和事件处理函数  data:{  n:0  },  methods:{  add(){  this.n++;  },  minus(){  if(this.n>0){ this.n-- };  }  }  });  </script>  </body>  </html>  运行效果: 同上 |

6. 简写后的步骤: 3步

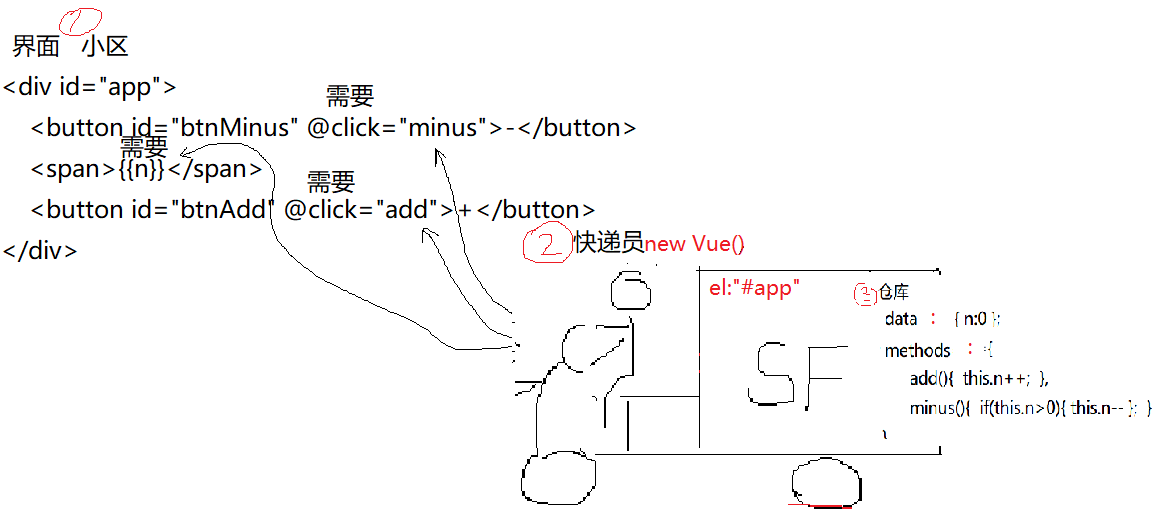
(1). 定义界面：

(2). 先定义new Vue()对象，用选择器找到自己负责的页面区域

(3). 在new Vue()对象内，添加data属性和methods属性，其值都是对象。

a. data属性对象中专门保存页面所需的所有变量,

b. methods属性对象中专门保存页面所需的所有函数。



7. 结果: 无论任何原因，只要data中的变量值发生变化！new Vue()快递员都会自动将新值重新送货到界面中对应的元素上。new Vue()会始终保持界面显示的内容和data中变量值同步！

三. MVVM: (Model, View, ViewModel)

对应小程序视频: 小程序->在线->VUE->day01 2. MVVM vue绑定原理 虚拟DOM树

1. 什么是MVVM设计模式: 对前端代码的重新划分

2. 旧前端代码，分为三部分:

(1). HTML: 专门保存网页的内容和结构

(2). CSS: 专门为网页中的元素添加样式

(3). JS: 专门为网页中的元素添加交互效果

3. 问题:

(1). 因为HTML和CSS都是静态的语言。缺少标准的程序语言必须的要素

a. 比如HTML和CSS中没有变量，就无法实现一次修改，多处变化

b. 比如HTML和CSS中缺少分支和循环，就无法实现基本的判断和反复生成的功能

总结: HTML和CSS生活不能自理！

(2). 页面上所有的琐碎的修改都要依赖js来实现。导致js代码极其冗余，且包含大量重复劳动！

4. 新MVVM，也将代码分为3部分:

(1). 界面（View）: 包括以前的HTML+CSS，而且还增强了HTML的功能

a. 给HTML添加了变量的功能: <span>{{n}}</span> 如果n变化，则span自动变化，无需任何js代码。

b. 为HTML添加了分支和循环功能: v-if v-else 以及v-for

(2). 模型对象(Model): 专门集中保存页面所需的变量和函数的对象

比如: data和methods都是模型对象

问题: 模型对象中的变量和方法不会自己长腿跑到界面中的元素上

(3). 视图模型(ViewModel): 快递员

a. 什么是视图模型: 自动将data中的变量和methods中的函数送到界面中指定元素上，并能自动保持界面显示与data中变量值同步 一种特殊的对象

b. 比如: new Vue()就充当了视图模型自动配送的角色。

5. new Vue()如何实现MVVM设计模式: Vue的绑定原理: 访问器属性+虚拟DOM树

(1). 访问器属性: 复习： 小程序->在线->JSCORE->day05 5. 访问器属性 get set

a. 回顾: 访问器属性:

1). 专门提供对一个普通属性的保护。

2). 每个访问器属性都包含一对儿, get和set函数

i. 当外界试图获取访问器属性的值时，自动调用访问器属性中的get

ii. 当外界试图修改访问器属性的值时，自动调用访问器属性中的set

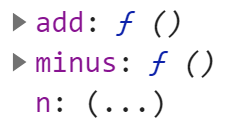
b. new Vue()如何利用访问器属性:

1). data对象和methods对象进入new Vue()后会立刻被打散。data中的成员变量和methods中的成员函数，最终会直接隶属于new Vue()对象。且data中的变量和methods中的函数处于平级。所以methods中的函数才能用this.变量名方式访问data中的变量

所以，new Vue()中的几乎所有this，都指new Vue()对象！

Vue{

... ...



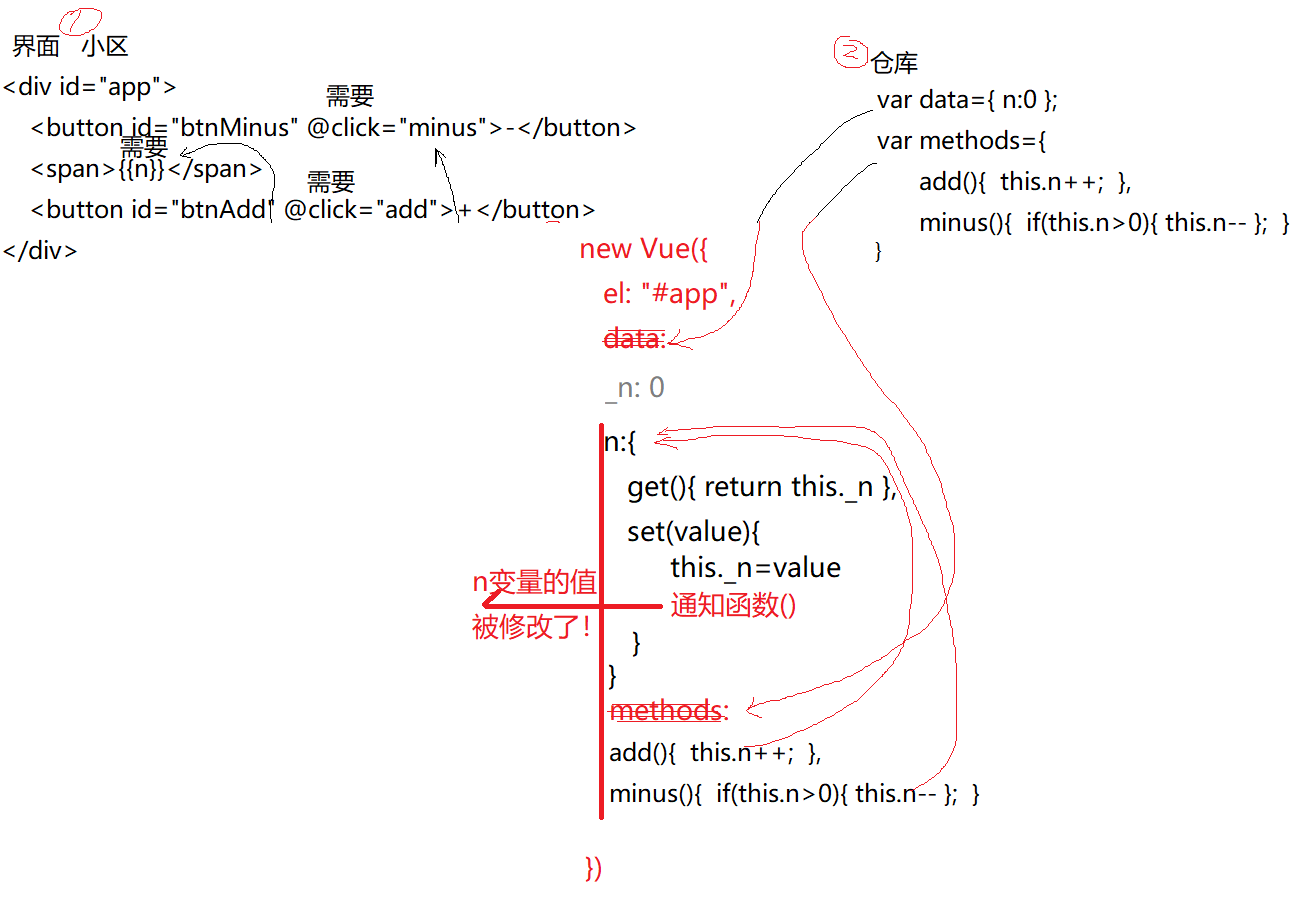
... ...

}

2). new Vue()隐藏data中原变量。然后为data中每个变量分别请保镖(访问器属性)。

结果: 任何情况下操作data中的变量n，早就不是普通的变量了。实际操作的都是访问器属性。且，只要试图修改data中的变量n，都会自动调用访问器属性n的set函数！

3). new Vue()在每个访问器属性的set函数中悄悄内置了一个通知函数。只要试图修改变量，都会调用访问器属性的set，都会制定向外发出通知: "xx变量的值变了！快来拿新值！"



(2). 虚拟DOM树:

a. 什么是: vue通过扫描真实DOM树，只提取出个别可能发生变化的元素，组成的一个棵简化版的新的DOM树。

b. 为什么: 优点：

1). 小, 只包含可能受影响的元素，不受影响的元素是不包含在内的

2). 遍历和查找快！

3). 更新效率高，因为每次只更新受影响的元素，不受影响的元素，不变的！

4). 避免重复代码。虚拟DOM树中已经提前封装了DOM增删改查操作！不用我们自己写！

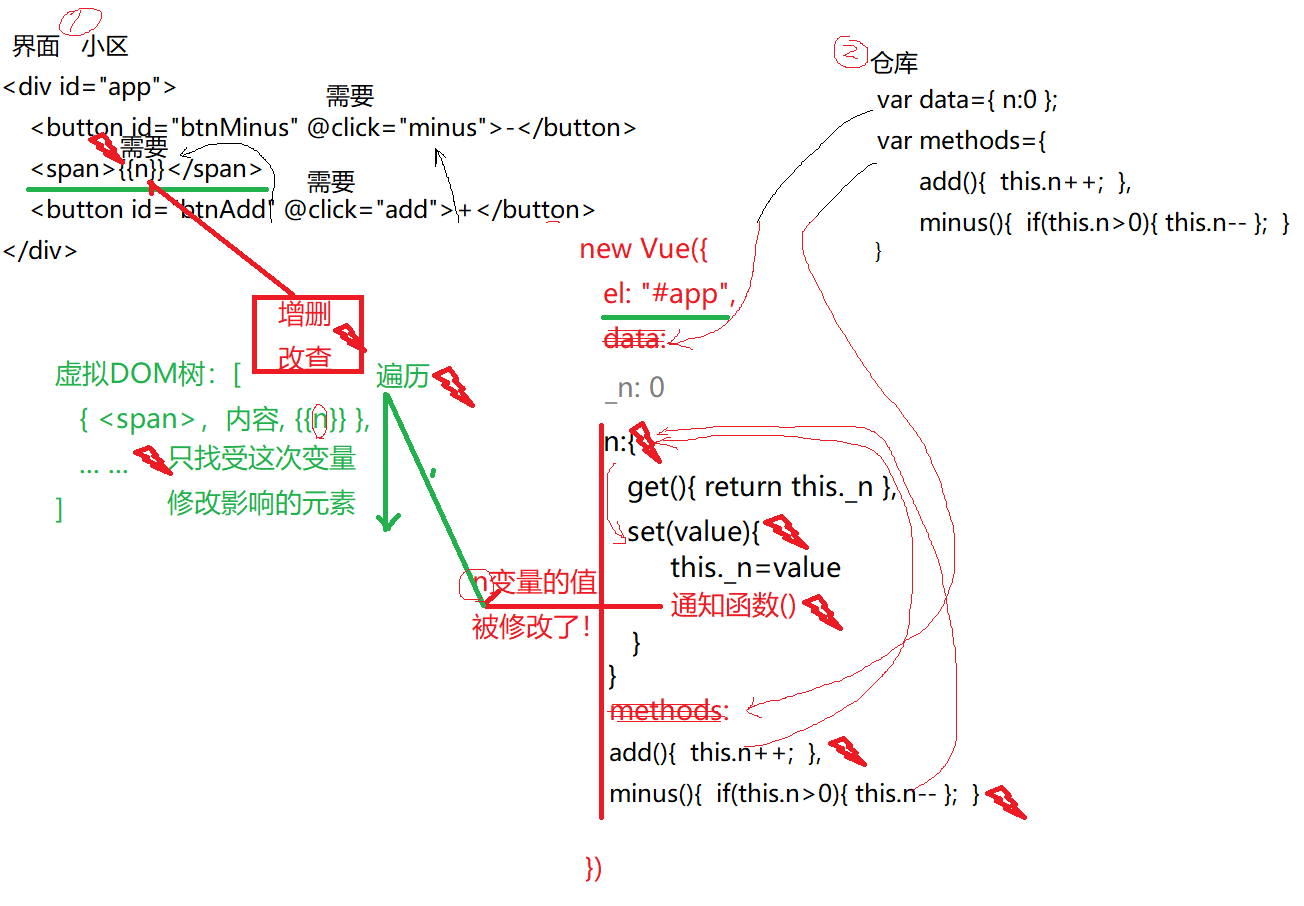
c. 虚拟DOM树在绑定过程中如何发挥作用：

1). new Vue中任何位置修改了data中的变量，其实修改的都是访问器属性

2). 都会自动触发这个访问器属性的set，都会自动发出通知

3). new Vue接到通知遍历虚拟DOM树，只找受本次变量修改影响的个别DOM元素

4). 虚拟DOM树利用已经封装好的增删改查操作，仅更新页面上受影响的个别元素。而不是大范围替换界面元素。—— 效率远高于jQuery！



对应小程序视频: 小程序->在线->VUE->day01 3. 绑定语法 {{}} 指令 v-bind... ...

四. 绑定语法: 学名插值语法, Interpolation

1. 什么是绑定语法: vue给HTML中新增的一种可以让HTML直接使用js中的变量 的特殊语法.

2. 为什么: 旧HTML不能使用js中的变量，哪怕很小的修改都要js先取出旧值，再修改后，最后放回来，才能更新界面——繁琐！

3. 何时: 今后只要界面中一个元素的内容，需要随js中变量自动变化，就要用绑定语法

4. 如何: <元素>{{自定义变量名}}</元素>

5. 后续: 如果界面中定义了一个{{自定义变量名}}，则new Vue()的data中就必须添加一个同名的变量，来支持这个界面位置的变化！

6. 其实: {{}}中除了可以放单个变量之外，还能放任何有返回值的合法的js表达式！——同ES6模板字符串${}的规则

(1). 可以放: 变量，运算，三目，有返回值的函数调用，创建对象，访问数组元素

(2). 不可以放: ~~程序结构~~(分支和循环)以及~~没有返回值的~~函数调用。

7. 示例: 在网页中用各种语法绑定变量

4\_{{}}.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">  <!--复习： 小程序->在线->JSCORE->day07 1. 模板字符串-->  <h3>Welcome {{uname}}</h3>  <h3>性别: {{sex==1?"男":"女"}}</h3>  <h3>小计: ¥{{(price\*count).toFixed(2)}}</h3>  <h3>下单时间: {{new Date(orderTime).toLocaleString()}}</h3>  <h3>今天星期{{week[new Date().getDay()]}}</h3>  </div>  <script>  new Vue({  el:"#app",  data:{  uname:"dingding",  sex:1,  price:12.5,  count:5,  orderTime:1592549839941, //new Date().getTime()  week:["日","一","二","三","四","五","六"]  // 0 1 2 3 4 5 6  }  })  </script>  </body>  </html>  运行结果: |

8. 问题: 如果元素的属性值想根据js中的变量不断变化，不能用{{}}语法，报错

- src="{{pm25<100?'img/1.png':pm25<200?'img/2.png':pm25<300?'img/3.png':'img/4.png'}}":

Interpolation inside attributes has been removed.

属性中的插值语法已经被移除了

Use v-bind or the colon shorthand instead. For example,

请用v-bind或:简写代替. 例如，

instead of <div id="{{ val }}">, use <div :id="val">.

用<div :id="val"> 代替<div id="{{val}}">

解决: 指令:

五. 指令:

1. 什么是: 一批vue官方提供的，为HTML添加新功能的特殊自定义扩展属性

2. 为什么: 因为{{}}只支持元素内容随变量自动变化，不支持属性自动变化，甚至无法实现分支和循环等复杂结构

3. 何时: 今后只要元素的属性值想随变量自动变化或想给HTML添加更多新功能时，都用指令！

4. 包括:

(1). 如果元素的属性值可能随变量自动变化: v-bind

a. <元素 v-bind:属性名="js变量或表达式">

b. 原理: 当new VUe()扫描到这里时，或者依赖的变量发生变化时，new Vue都会自动重新计算v-bind后的变量值或表达式结果，用变量值或表达式结果代替该属性的值！

c. 强调: 如果前边加了v-bind:，则""中千万不要再加{{}}，此时""就扮演了{{}}的角色。

d. 简写: 今后v-bind: 都省略 v-bind，只写:

<元素 :属性名="js变量或表达式">

e. 示例: 根据pm25数值改变表情图片

5\_v-bind.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>  <!--需求: 希望img的src属性值随data中的pm25数值自动变化  pm25<100，src属性值为img/1.png  pm25<200, src属性值为img/2.png  pm25<300, src属性值为img/3.png  其余img/4.png  -->  <div id="app">  <h3>{{pm25<100?'img/1.png':pm25<200?'img/2.png':pm25<300?'img/3.png':'img/4.png'}}</h3>  <img :src="pm25<100?'img/1.png':pm25<200?'img/2.png':pm25<300?'img/3.png':'img/4.png'" alt="">  </div>  <script>  new Vue({  el:"#app",  data:{  pm25:350  }  })  </script>  </body>  </html>  运行结果： |

(2). 根据js中的变量值控制一个元素的显示隐藏: v-show

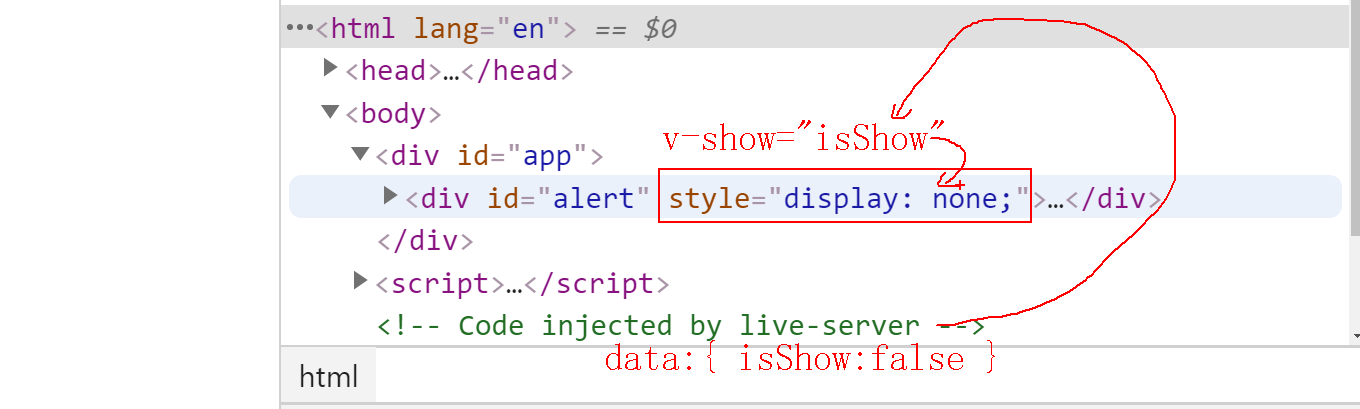
a. <元素 v-show="bool类型的变量或js条件表达式">

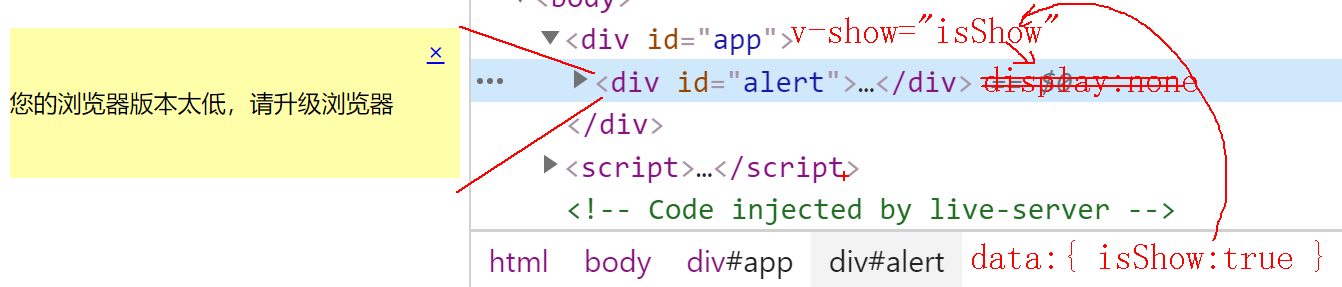
b. 原理: 当new Vue首次扫描到这里或依赖的js变量发生变化时，new Vue都会自动计算""中的变量值或js表达式值。

1). 如果变量值或表达式值为false，则new Vue()自动给元素添加display:none，当前元素隐藏

2). 如果变量值或表达式值为true，则new Vue()自动去掉元素上的display:none，当前元素就显示

c. 示例: 打开和关闭对话框(极其常用)





6\_v-show.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>  #alert{  width:300px;  height:100px;  background-color:#ffa;  position:fixed;  top:50%;  left:50%;  margin-left:-150px;  margin-top:-50px;  line-height:100px;  }  #alert>a{  float:right;  line-height:30px;  margin-right:10px;  }  </style>  </head>  <body>  <div id="app">  <!--希望div#alert随变量isShow的值不同而显示隐藏  希望点click me，让div#alert显示（将isShow变量值改为true）  希望点×，让div#alert隐藏（将isShow变量值改为false）-->  <button @click="pop">click me</button>  <div v-show="isShow" id="alert">  <a href="javascript:;" @click="close">×</a>  您的浏览器版本太低，请升级浏览器  </div>  </div>  <script>  new Vue({  el:"#app",  //因为页面上需要一个isShow变量，所以  data:{  isShow:false//定义一个变量控制对话框的显示和隐藏，开局默认隐藏  },  //因为页面上需要？个事件处理函数: pop和close  methods:{  pop(){  this.isShow=true;  },  close(){  this.isShow=false;  }  }  })  </script>  </body>  </html>  运行结果： |

(3). 根据js中的变量值，控制两个元素二选一显示隐藏: v-if v-else

a. <元素1 v-if="bool变量或js条件表达式">

<元素2 v-else >

b. 原理: 当new Vue首次扫描到这里或依赖的js变量发生变化时，new Vue都会自动计算v-if后""中的变量值或js表达式值。

1). 如果v-if后的值为true，则保留v-if所在元素，删除v-else所在元素

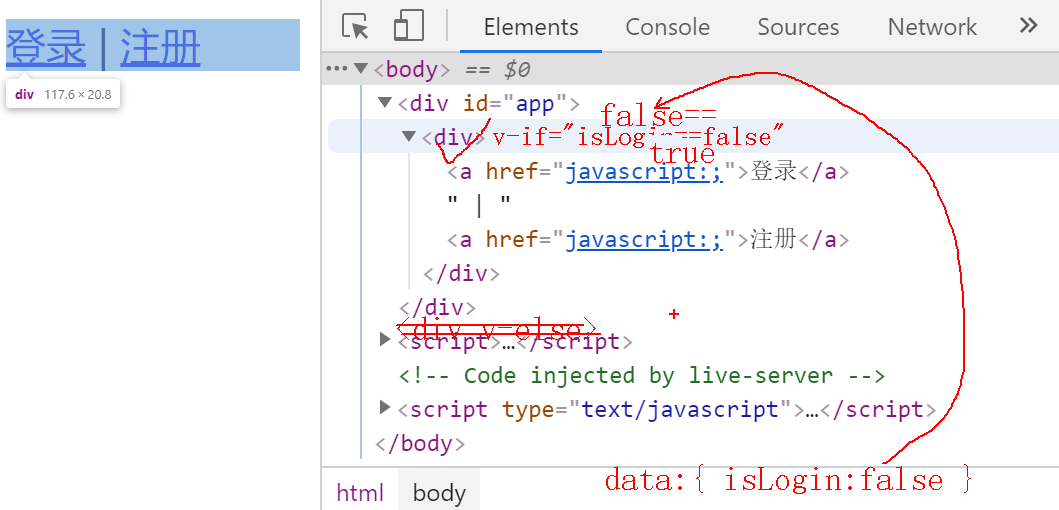
2). 如果v-if后的值为false，则删除v-if所在元素，保留v-else所在元素

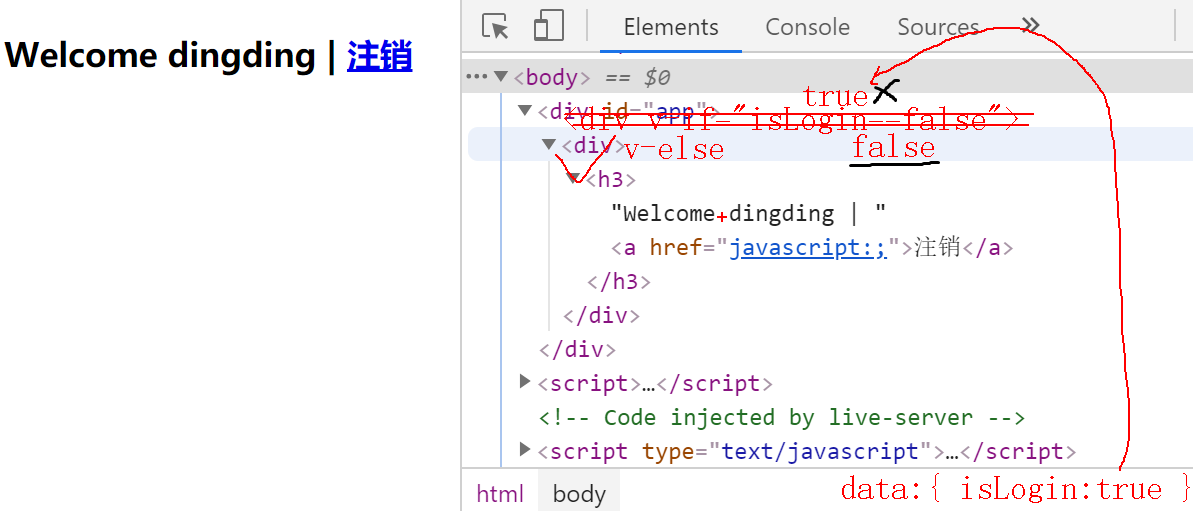
c. 强调:

1). 和程序中的if else一样，v-else后不要写="xxx"

2). v-if和v-else两个元素必须紧挨着，中间不能插入别的元素！

d. 示例: 切换登录和注销状态





7\_v-if\_v-else.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">  <!--根据变量值不同，显示不同的登录状态-->  <!--希望点击登录，切换到已登录状态  希望点击注销，切换回未登录状态-->  <!--希望未登录时只显示第一个div-->  <div v-if="isLogin==false">  <a href="javascript:;" @click="login">登录</a> | <a href="javascript:;">注册</a>  </div>  <!--已登录时只显示第二个div-->  <div v-else>  <h3>Welcome dingding | <a href="javascript:;" @click="logout">注销</a></h3>  </div>  </div>  <script>  new Vue({  el:"#app",  //因为页面上需要根据一个变量值显示不同的登录状态，所以data中要提供一个变量表示是否登录  data:{  isLogin:false,//开局，默认未登录  },  //因为页面上需要两个事件处理函数，所以methods中就要准备2个事件处理函数: login logout  methods:{  login(){  this.isLogin=true;  },  logout(){  this.isLogin=false;  }  }  })  </script>  </body>  </html>  运行结果： |

e. 鄙视: v-show 和 v-if的区别？

1). v-show通过删除元素 display:none 实现隐藏 ——不改变DOM树结构，效率高

2). v-if 通过删除元素 实现隐藏 —— 改变DOM树，效率略低

(4). 多个元素多选一显示： v-if v-else-if v-else

a. <元素1 v-if="bool变量或js条件表达式">

<元素2 v-else-if="另一个bool变量或js条件表达式">

... ...

<元素n v-else>

b. 原理: 当new Vue首次扫描到这里或依赖的js变量发生变化时，new Vue都会自动计算v-if后""中的变量值或js表达式值。

1). 如果v-if后的值为true，则保留v-if所在元素，删除其余所有v-else-if和v-else的元素

2). 如果v-if后的值为false，则删除v-if的元素，继续向后判断每个v-else-if后的条件。哪个v-else-if后的条件为true，就只保留这个v-else-if所在的元素。删除其余元素

3). 如果所有v-if和v-else-if中的条件都为false，则删除所有v-if和v-else-if，只保留v-else所在元素

c. 强调:

1). 和程序中的if else一样，v-else后不要写="条件"

2). v-if和v-else-if和v-else多个元素必须紧挨着，中间不能插入别的元素！

d. 示例: 根据PM25数值显示不同的表情

8\_v-else-if.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">  <!--希望根据pm25的数值，修改页面上的图片:  pm25<100，显示img/1.png  pm25<200, 显示img/2.png  pm25<300, 显示img/3.png  其余img/4.png  -->  <img v-if="pm25<100" src="img/1.png" alt="">  <img v-else-if="pm25<200" src="img/2.png" alt="">  <img v-else-if="pm25<300" src="img/3.png" alt="">  <img v-else src="img/4.png" alt="">  </div>  <script>  new Vue({  el:"#app",  //因为页面上需要根据pm25的数值控制显示哪张图片，所以:  data:{  pm25:360  }  })  </script>  </body>  </html>  运行结果: |

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

(4). 控制两个元素二选一显示: v-if v-else

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

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

1. vue和第一个vue程序

2. MVVM Vue绑定原理 虚拟DOM树

3. 绑定语法 {{}} 指令 v-bind v-show ...

作业:

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

2. 看小程序视频预习v-for用法

小程序->在线->VUE->day01 4. v-for遍历数组 ...

3. 看小程序视频学习如何用vue实现学子商城首页:

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

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

然后，看视频，在index.js中添加固定代码

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

