提前准备:

1. 今天的历届师兄师姐问过的经典的高频的问题清单都已经整理在小程序->首页->VUE->day01里

2. 今天的课程中所需的单词列表都已经整理在小程序->单词->VUE->day01里

3. 今天的重点视频、扩展高频鄙视题视频已经整理在小程序->在线-> VUE ->day01里

如果不知道问什么: 谁问的越多！谁培训费花的越值！

1. 笔记中或画图中那句话或者那个位置不明白，立刻问！

2. 案例中哪行代码或注释不明白，立刻问！

3. 课下自己做项目，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！——公众号: 前端大全 定期总结大厂笔试题

5. 看小程序问题清单时，遇到看不懂的问题，立刻问！

每个案例三遍:

1. 连注释+代码抄一遍

2. 保留注释，删掉代码，自己试着填回来

3. 删除注释和代码，自己使用用人话把注释写回来！——才算自己会！

正课:

1. 什么是VUE

2. 如何使用VUE

3. MVVM

4. 绑定语法

5. 指令

一. 什么是VUE:

1. 什么是: 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: 需要下载才能用

(2). 基于MVVM设计模式?

(3). 渐进式: 可以逐步在项目中使用VUE的各项技能。不必整个项目都改为VUE

(4). 纯前端js: 只靠浏览器就可执行VUE程序，无需后端技术。

(5). 框架: 原生DOM vs jquery函数库 vs 框架

a. 原生DOM: 浏览器自带的操作网页内容的唯一途径

1). 优点: 万能

2). 缺点: 繁琐！

b. jquery函数库: 第三方开发的基于DOM实现的操作网页内容的简化版函数

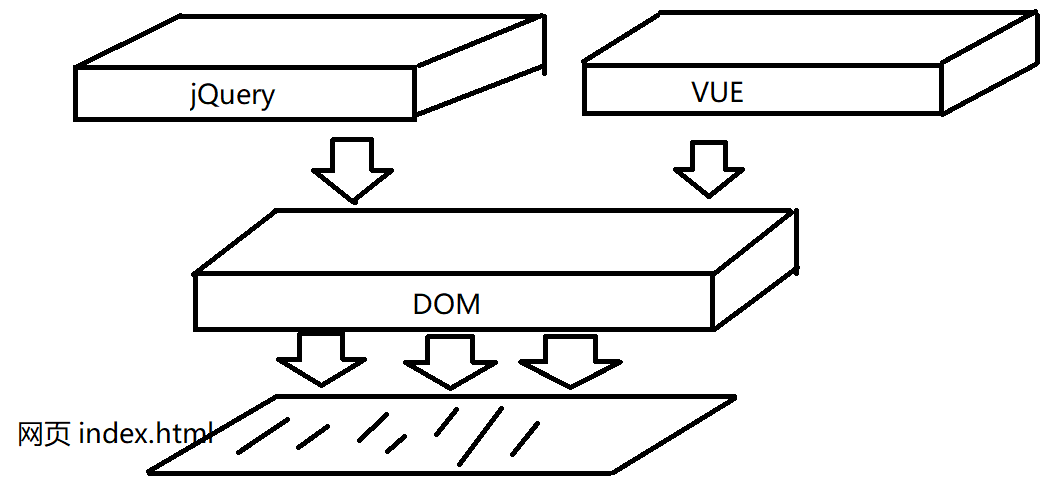
1). 优点: 简单

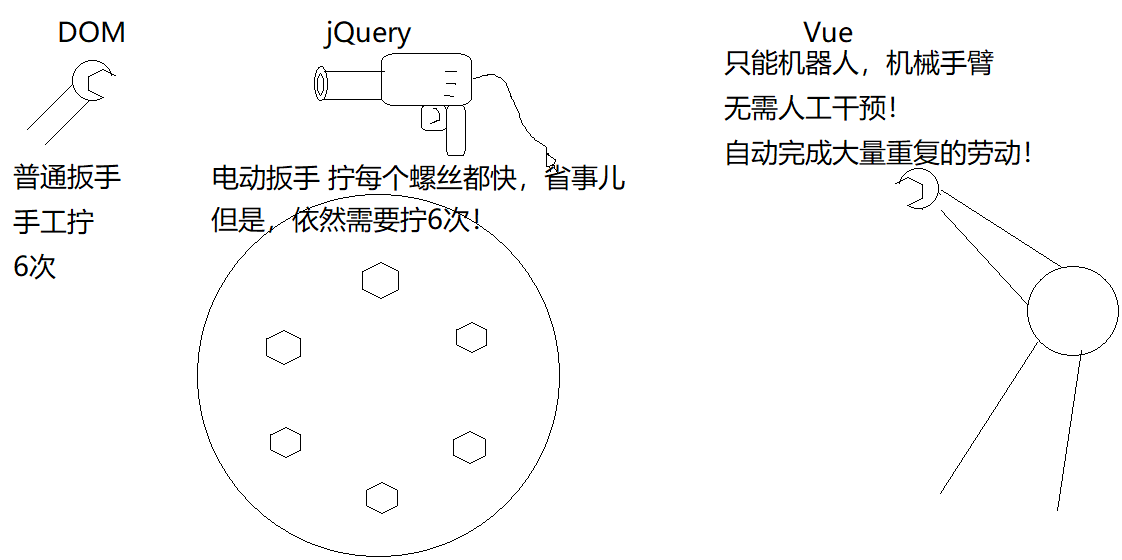
2). 缺点: 仅仅是对DOM的每一步操作进行的简化，并没有从根本上减少开发步骤！

c. VUE框架: 第三方开发的基于响应式编程方式的程序

1). 优点: 从根本上简化了开发步骤！

2). 缺点: 过于精简，初学者很难看懂！老程序员也不容易习惯！





2. 何时: 今后所有以数据操作(增删改查)为主的项目都可用vue开发

二. 如何使用VUE

1. 下载: 2种方式:

(1). 仅下载vue.js文件，引入网页中即可使用：初学者使用——前三天 2.5版本

a. 官网: cn.vuejs.org

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

b. 版本:

1). 开发版: ——仅适合于学习和开发项目时使用

i. 未压缩: 包含完备的注释、代码格式和见名知义的变量名

ii. 还包含完备的错误提示

iii. 缺点: 体积大，下载慢

2). 生产版: ——产品开发完上线时才使用

i. 压缩: 去掉所有注释和代码格式，极简化了变量名

ii. 还去掉了很多友好的错误提示

iii. 优点: 体积小，下载快！

(2). 下载并安装脚手架代码: 熟手以及今后公司中项目使用——后两天 4.x版本

(敬请期待...)

2. 我的第一个vue程序: 点击按钮增减数量

(1). 示例: jquery版本

|  |
| --- |
| <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>    <script src="js/jquery-1.11.3.js"></script>  </head>  <body>  <button>-</button><span>1</span><button>+</button>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中查找两个按钮  var $btnMinus=$("button:contains(-)");  var $btnAdd=$("button:contains(+)");  console.log($btnMinus);  console.log($btnAdd);  //2. 绑定事件处理函数  $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())    //数字+1    n++    //再放回span中    $span.html(n);  })  $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())    //数字+1    n--;    n==0&&(n=1);    //再放回span中    $span.html(n);  })  </script>  </body> |

(2). 问题: 因为HTML和CSS两种语言根本就不是真正的编程语言。缺少程序必须的基本功能，生活不能自理，一切都需要js帮助实现功能！比如:

a. HTML和css中不支持变量: 如果程序中的变量修改，页面上无法自动更新。只能靠js程序送到页面上。

b. HTML和css中不支持分支和循环: 如果希望根据条件显示不同的页面内容，或反复生成多个相同结构的页面内用，也只能依靠js程序先在内存中生成好，再更新到页面上去

(3). 结果: 导致每做一项功能，都要反复的查找元素，获取元素内容，修改后，再放回去，就产生了大量重复的劳动。

(4). 解决: 框架：采用响应式编程方式：程序中只负责修改数据即可，由专门的框架程序帮助我们将程序中的数据自动同步到页面上！再不用我们频繁查找元素、获取元素内容，修改后再放回去！减少了大量重复的代码！

(5). 示例: vue实现

|  |
| --- |
| <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>    <!--0. 引入vue.js-->    <script src="js/vue.js"></script>  </head>  <body>    <!--1. 先定义界面: 要求:    1.1 整个界面必须包裹在一个唯一的父元素内    1.2 界面中可能发生变化的位置用{{变量名}}标记出来    1.3 界面中可能触发事件的元素用@事件名="处理函数名"标记出来-->    <div id="app"><!--小区-->      <!--住户-->      <button @click="minus">-</button>      <span>{{n}}</span>      <button @click="add">+</button>    </div>    <script>      //2. 定义所有页面上所需的数据和函数,要求:      //2.1 页面中所需的变量和初始值，应该放在一个名为data的对象中      var data={ //仓库1: 专门存变量        //页面上需要几个变量，data中就定义几个变量        n:1      }      //2.2 页面中所需的事件处理函数，必须放在一个名为methods的对象中      var methods={ //仓库2: 咱们存函数        //页面上需要几个事件处理函数，methods中就要有几个方法        //methods中的方法只管操作data中的变量即可,不用关心变量将来放在那里，如何放！——信任        add(){ //复习ES6          //特别的要求: methods中的方法想操作data中的变量，必须加"this."——暂时不要问为什么!          this.n++;        },        minus(){          this.n--;          //如果n==0，就将n重置回1          this.n==0&&(this.n=1)        }      }      //3. 创建一个Vue类型的对象——快递员      new Vue({        el:"#app", //element:"选择器" 告诉这个快递员，负责哪个小区的业务        data,//data:data,        methods//methods:methods      })    </script>  </body> |

(6)优势: 因为快递员()承包了自动同步程序中的数据和方法到页面的工作，所以程序员不再需要任何重复的体力劳动，只要维护好数据即可。——从根本上优化了开发步骤。几乎避免了大量重复的代码。

三. MVVM设计模式:

1. 旧前端项目分为三部分:

(1). HTML 专门定义网页的内容

(2). CSS 专门定义网页的样式

(3). JS 专门为网页添加交互行为，负责修改HTML内容和CSS样式

2. 问题: 因为HTML和CSS缺少程序必须的要素(变量、分支和循环)，生活不能自理，所以一切变化都需要js来操作。导致js中有大量重复的编码。

3. 解决: MVVM设计模式（鄙视）: 对前端内容的重新划分: 3部分:

(1). 界面(View)：包括以前的HTML +CSS，且HTML还是增强版的。比如:

MVVM设计模式中的HTML支持变量:{{n}}

MVVM设计模式中的HTML中还可写分支和循环

(2). 数据模型(Model): 是专门保存页面所需的变量和函数 的对象，比如:

var data={ n:1 };

var methods={

add(){ ... },

minus(){ ... }

}

问题: 数据模型不会自己长腿跑到页面上

(3). 视图模型(ViewModel): 是专门自动保持页面与内存中数据同步的特殊对象(快递员)

4. Vue的绑定原理: Vue是如何实现MVVM设计模式:

(1). 一切都从new Vue()开始！将来任何vue程序要想运行，必须有一个new Vue()

(2). new Vue()的第一件事: 创建Vue类型的对象，并将模型对象的内容包裹进Vue类型的对象中托管！

a. 将原data隐姓埋名，并为data中每个变量自动请保镖(定义访问器属性)

访问器属性直接隶属于new Vue()

结果: 今后只要想操作data中的变量，必然要经过保镖才能使用。一旦经过保镖，我们就可以实时监控每个变量的变化

b. 打散引入的methods，让methods中的所有方法，都直接隶属于new Vue()

c. 结果methods中的方法和为data中变量请的保镖就平级了！所以，对象中，平级的方法，想使用平级的访问器属性，必须加"this."  
 (3). new Vue()做的第二件事: 创建虚拟DOM树（鄙视），并渲染页面内容

a. 什么是虚拟DOM树: Vue扫描原HTML页面获得的仅保存可能发生变化的元素的简化版DOM树

b. 何时创建: 在new Vue()创建完对象后，自动扫描el:"#app"所指向的页面区域。在扫描过程中仅找出可能发生变化的元素，保存在一个新创建的虚拟DOM树集合中。

c. 为什么使用虚拟DOM树: （好处）（鄙视）

1). 内容少: 仅包含可能发生变化的元素，其余元素都不包含

2). 遍历快: 可快速找到受影响的元素

3). 渲染效率高: 只更新受影响的元素，不受影响的元素保持不变！

4). 已经封装了DOM增删改查操作，避免了大量重复的编码！

(4). 当在new Vue()中任何情况下修改了变量:

a. 修改变量其实修改的是变量的访问器属性

b. 修改访问器属性就会触发set()

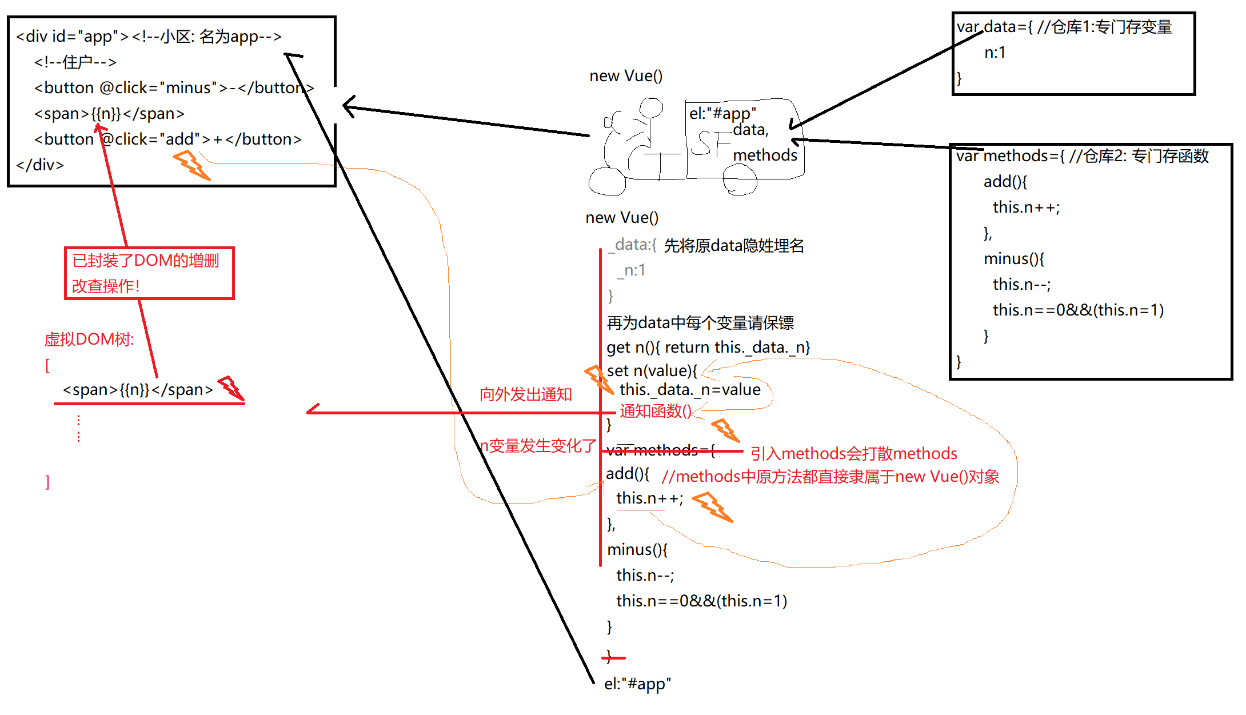
c. 触发set()，就会自动执行set中的通知函数()

d. 通知发给虚拟DOM树，告知哪个变量的值发生了变化

e. 虚拟DOM树遍历自己内部的元素，只找到受本次变化影响的元素

f. 虚拟DOM树用已经封装好的DOM操作，只更新页面中受影响的元素.

(5). 总结: VUE绑定原理的实现其实就靠: 访问器属性 + 虚拟DOM树



四: 绑定语法: {{}} 学名叫插值(Interpolation)语法，也称为大胡子语法

1. 什么是绑定语法:在界面中标记一处元素内容变化的位置的特殊语法

2. 为什么: 希望vue对象可以将我们想要的变量值自动送到这个指定的位置上

3. 何时: 今后只要元素的内容可能随变量变化，都要用绑定语法:

4. 如何: <元素>{{变量或js表达式}}</元素>

(1). 其实vue中的{{xxx}}语法等效于ES6的模板字符串中的${}语法

(2). Vue的{{xxx}}中和ES6的模板字符串的${xxx}中

a. 可以放什么: 变量、有返回值的函数调用、运算和js表达式、创建对象、访问数组的元素

b. 不可以放什么: 分支和循环等程序结构，以及没有返回值的函数调用或js表达式

5. 结果: 只要new Vue程序中，{{}}中所需的变量发生变化，vue对象就会将新的变量值自动送到这个位置来显示在页面上

6. 示例: 在页面上绑定显示各种类型各种需求的数据

|  |
| --- |
| <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"><!--小区-->      <h1>用户名:{{uname}}</h1>      <h2>性别:{{sex==1?"男":"女"}}</h2>      <h3>下单时间:{{new Date(orderTime).toLocaleString()}}</h3>      <h3>小计:¥{{(price\*count).toFixed(2)}}</h3>      <h3>今天星期{{week[new Date().getDay()]}}</h3>    </div>    <script>      new Vue({//快递员        el:"#app",        data:{          uname:"dingding",          sex:0,          orderTime:1585722499077,          price:12.5,          count:5,          week:["日","一","二","三","四","五","六"]          //     0    1    2    3    4    5   6        }      })    </script>  </body> |

五. 指令:

1. 什么是: 用来增强html元素的功能的特殊元素属性

2. 为什么: 因为旧的html，生活不能自理，所以才需要vue指令来增强HTML中元素的功能。比如指令可以让元素属性也支持变量，指令可以控制元素的显示隐藏，可以反复生成多个相同结构的元素等

3. 官网案例: https://cn.vuejs.org/v2/api/#%E6%8C%87%E4%BB%A4

4. 包括: 13种:

(1). 如果希望元素的一个属性值随程序中变量自动变化时: v-bind

a. 问题: 使用{{}}可以让元素的内容随程序中的变量自动变化，但是{{}}不可以让元素的属性值随程序中变量自动变化!

b. 解决: 用v-bind指令代替{{}}

c. 如何: <元素 v-bind:要变化的属性="变量或js表达式">

强调: 一旦使用v-bind: ，就不能写{{}}了！""就承担了{{}}的作用。之前{{}}中可以写什么，v-bind:后的""中就能写什么！

d. 结果: 如果v-bind""中的变量在程序中发生变化，则new Vue()会自动更新这个元素的这个属性值！保持与程序中的变量一致

e. 简写: <元素 ~~v-bind~~:要变化的属性="变量或js表达式">

f. 示例: 根据pm25值的变化，自动修改<img>的src属性，显示不同的图片

|  |
| --- |
| <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"><!--小区-->      <!--如果pm25<100,则src为img/1.png          否则如果pm25<200,则src为img/2.png          否则如果pm25<300,则src为img/3.png          否则src为img/4.png-->      <h3>src:{{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'">    </div>    <script>      new Vue({//快递员        el:"#app", //快递员负责的小区        data:{ //仓库1: 专门保存小区中所有住户所需的变量          pm25:Math.random()\*400        }      })    </script>  </body> |

(2). 控制一个元素的显示隐藏: v-show

a. 如何: <元素 v-show="bool类型的变量或条件表达式">

b. 结果:

(1). 当v-show所依赖的程序中的变量发生了变化，则重新计算并渲染v-show所在的元素

(2). 如果v-show后的变量或条件表达式值为true，则v-show什么也不干，该元素就默认显示

(3). 如果v-show后的变量或条件表达式值为false，则v-show自动为当前元素添加display:none属性，则该元素隐藏！

c. 示例: vue中实现打开和关闭对话框

|  |
| --- |
| <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>    <style>      \*{margin:0; padding:0}      #pop{        width:300px;        height:100px;        text-align:center;        background-color:#afa;        position:fixed;        top:50%;        left:50%;        margin-left:-150px;        margin-top:-50px;      }      #pop>p{        text-align:right;      }    </style>    <script src="js/vue.js"></script>  </head>  <body>    <div id="app">      <button @click="showIt">click me</button>      <div id="pop" v-show="show">        <p><a href="javascript:;" @click="hideIt">×</a></p>        您的浏览器版本过低,请升级浏览器      </div>    </div>    <script>      new Vue({        el:"#app",        data:{          show:false //记录对话框是显示还是隐藏即可        },        methods:{          showIt(){            this.show=true          },          hideIt(){            this.show=false          }        }      })    </script>  </body> |

(3). 控制两个元素二选一显示隐藏: v-if v-else

a. 如何:

<元素1 v-if="条件"></元素1>

<元素2 v-else></元素2>

强调:

1). 和程序中的else一样！v-else后也不写条件

2). V-if和v-else两个元素之间禁止插入任何其他元素！v-if和v-else两个元素必须紧紧相连！

b. 原理:

(1). 当v-if后依赖的变量发生变化时，new Vue()会重新渲染这两个元素，重新决定谁显示谁不显示

(2). 如果v-if后的条件值为true，则保留v-if的元素，删除v-else的元素

(3). 如果v-if后的条件值为false，则删除v-if的元素，保留v-else的元素

c. v-show vs v-if 区别： (高频笔试题)

(1). V-show采用display:none方式隐藏元素，而不删除元素，不影响DOM树——效率高！

(2). V-if采用删除元素方式隐藏元素，会改变DOM树——效率低

d. 示例: 使用vue实现登录和注销切换

|  |
| --- |
| <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">      <h1 v-if="isLogin==false"><a href="javascript:;" @click="login">登录</a> | <a href="javascript:;">注册</a></h1>      <h1 v-else>Welcome dingding, <a href="javascript:;" @click="logout">注销</a></h1>    </div>    <script>      new Vue({        el:"#app",        data:{          isLogin:false //记录是否登录        },        methods:{          login(){            this.isLogin=true;          },          logout(){            this.isLogin=false;          }        }      })    </script>  </body> |

(4). 多个元素多选一显示: v-if v-else-if v-else

a. 如何:

<元素1 v-if="条件1"></元素1>

<元素2 v-else-if="条件2"></元素2>

... ...

<元素n v-else></元素n>

强调: V-if、v-else-if和v-else多个元素之间禁止插入任何其他元素！

b. 原理:

(1). 当v-if或v-else-if后依赖的变量发生变化时，new Vue()会重新渲染这几个元素，重新决定谁显示谁不显示

(2). 如果v-if后的条件值为true，则保留v-if的元素，删除其余v-else-if和v-else的元素

(3). 如果v-if后的条件值为false，则继续判断v-else-if后的条件，哪个元素的v-else-if条件满足，就只保留哪个元素，并删除其余v-if、v-else-if和v-else元素

(4). 如果所有条件都不满足，则删除所有v-if和v-else-if的元素，只保留v-else元素

c. 示例: 根据pm25数值，显示不同的表情

|  |
| --- |
| <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <meta http-equiv="X-UA-Compatible" content="ie=edge">    <title>Document</title>    <script src="js/vue.js"></script>  </head>  <body>  <div id="app">    <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>  var vm=new Vue({    el:"#app",    data:{      pm25:Math.random()\*400    }  })  </script>  </body> |

(5). 反复生成多个相同结构的元素: v-for

a. 如何:

<要反复生成的元素 v-for="(elem, i) of 数组">

... {{elem}} 或 {{i}} ...

</要反复生成的元素>

b. 强调:

1). elem和i只是变量名，可以随便改，只要你自己知道什么意思举行，但是前后顺序不能换：逗号前获得的是元素值，逗号后获得的是下标i

2).

c. 原理:

1). 通常要反复生成多个元素时，都会遍历程序中一个数组中每个元素。遍历到几个元素，就反复生成几个相同结构的元素副本.

2). 每遍历一个元素of都会取出数组中当前元素值elem和当前下标位置i，被of取出的元素值elem和i，可继续用于当前这个元素内容中的绑定

d. 其实v-for of统一了程序中的for of和for in，vue中的v-for of即可遍历数字下标的索引数组、类数组对象和字符串，还可遍历自定义下标名称的对象和关联数组

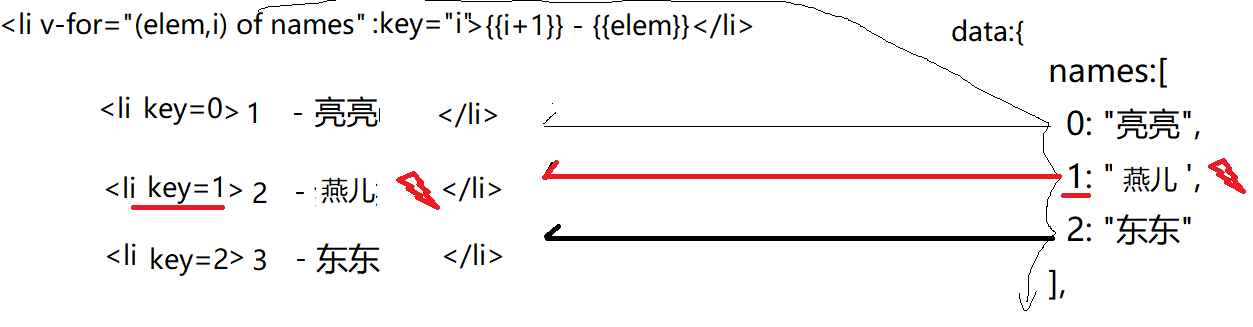
e. v-for of还可以数数！

f. 问题: (高频笔试题) 默认情况下，如果原数组内容中某个元素值改变，v-for会将所有元素重新生成一遍！——效率低

原因: v-for默认生成的元素除内容之外几乎都是一模一样的。当修改数组中某一个元素时，v-for是不知道该修改哪个对应元素副本的。

解决: 今后，凡是使用v-for，都要同时绑定一个:key="i"，且为:key属性绑定的值应该是不重复的下标：

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



好处: 从此每个反复生成的元素副本上都有一个唯一的标识key属性。如果元素中某个位置的元素发生改变，则v-for完全可以根据改变的元素的下标，找到对应key属性值的一个元素副本，只更新一个元素即可！不必重新生成所有元素副本。

g. 其实v-for为了适应多数人的习惯，即可写成v-for of，也可以写成v-for in。但是其实两者完全相同！没有任何差别！

h. 示例: 反复生成多个相同结构的元素

|  |
| --- |
| <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <meta http-equiv="X-UA-Compatible" content="ie=edge">    <title>Document</title>    <style>      .pages{        list-style:none;      }      .pages>li{        float:left;        padding:5px 10px;        border:1px solid #555;      }    </style>    <script src="js/vue.js"></script>  </head>  <body>  <div id="app">    <ul>      <!--先只保留一个要反复生成的元素当模板-->      <li v-for="(elem,i) of names" :key="i">{{i+1}} - {{elem}}</li>    </ul>    <h3>lilei的个人资料：</h3>    <ul>      <li v-for="(value,key) of lilei" :key="key">{{key}} : {{value}}</li>    </ul>    <ul class="pages">      <li v-for="i of pageCount" :key="i">{{i}}</li>    </ul>  </div>  <script>  var vm=new Vue({    el:"#app",    data:{      names:["亮亮","然然","东东"],      lilei:{        sname:"Li Lei",        sage:11      },      pageCount:10 //共10页数据    }  })  </script>  </body> |

总结:

1. 如果一个元素的内容可能发生变化: {{xxx}}

2. 如果一个元素的属性值可能发生变化: :属性名="xxx"

3. 如果控制一个元素显示隐藏: v-show="bool变量或条件表达式"

4. 如果两个元素二选一显示: v-if="条件" v-else

5. 如果多个元素多选一显示: v-if="条件1" v-else-if="条件2" ... v-else

6. 反复生成多个相同结构不同内容的元素:

v-for="(elem,i) of 数组/类数组对象/对象/数字" :key="i"

作业:

1. (必须)看小程序视频学习高频笔试题: 观察者模式

小程序->在线->vue->day01: 高频笔试题: 观察者模式

2. (扩展)看小程序视频学习使用vue绑定学子商城首页

开局可使用: 小程序->在线->jquery->day04:项目附件: 学子商城项目——首页index+服务器端cors跨域源代码 中 public压缩包。

学有余力的同学想看服务器端源代码的同学再下xzserver压缩包，如果基础一般能力有限，则不必关心服务器端xzserver代码。只用public+东哥新浪云做实验即可！

视频中所有http://localhost:5050/xxx 地址都改为 http://xzserver.applinzi.com/xxx

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 1

作业: 使用VUE绑定学子商城首页和详情页部分内容 2