学习课程你需要的前置知识,,咱班的同学没问题.

1. HTML的基础知识，你需要达到中级水平，写前端页面的结构代码完全没有问题。
2. CSS的基础知识，最好做过半年以上的切图和布局，最好了解CSS3的知识。
3. Javascript的基础知识，对基本语法掌握，这个要求不高，因为遇到难的我会在视频中讲解。
4. node.js初级知识，只需要会npm的使用和项目初始化就可以了

第1节：走起我的Vue2.0

一、下载Vue2.0的两个版本：

官方网站：<http://vuejs.org/>

* 开发版本：包含完整的警告和调试模式
* 生产版本：删除了警告，进行了压缩

二、项目结构搭建

新建文件夹,webstorm中打开

创建assets文件夹, 包含css,js

创建index.html

三、live-server使用

npm install live-server -g

live-server

四、编写第一个HelloWorld代码：



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | *<!DOCTYPE html>*  <html lang="en">  <head>      <meta charset="UTF-8">      <script type="text/javascript" src="../assets/js/vue.js"></script>      <title>Helloworld</title>  </head>  <body>      <h1>Hello World</h1>      <hr>      <div id="app">          {{message}}      </div>        <script type="text/javascript">          var app=new Vue({              el:'#app',              data:{                  message:'hello Vue!'              }          })      </script>  </body>  </html> |

第2节：v-if  v-else  v-show 指令

通过上节课的学习，我们已经搭建好了开发环境，并且写出了一个简单的HelloWorld程序，我们今天主要学习v-if  v-else 和 v-show。

一、v-if:

v-if:是vue 的一个内部指令，指令用在我们的html中。

v-if用来判断是否加载html的DOM，比如我们模拟一个用户登录状态，在用户登录后现实用户名称。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | *<!DOCTYPE html>*  <html lang="en">  <head>      <meta charset="UTF-8">      <script type="text/javascript" src="../assets/js/vue.js"></script>      <title>v-if & v-show & v-else</title>  </head>  <body>      <h1>v-if 判断是否加载</h1>      <hr>      <div id="app">          <div v-if="isLogin">你好：changjiang</div>          <div v-else>请登录后操作</div>        </div>        <script type="text/javascript">          var app=new Vue({              el:'#app',              data:{                 isLogin:false              }          })      </script>  </body>  </html> |

这里我们在vue的data里定义了isLogin的值，当它为true时，网页就会显示：你好：changjiang，如果为false时，就显示请登录后操作。

2、v-show ：

调整css中display属性，DOM已经加载，只是CSS控制没有显示出来。

|  |  |
| --- | --- |
| 1 | <div v-show="isLogin">你好：changjiang</div> |

3、v-if 和v-show的区别：

* v-if： 判断是否加载，可以减轻服务器的压力，在需要时加载。
* v-show：调整css dispaly属性，可以使客户端操作更加流畅。

第3节：v-for指令 ：解决模板循环问题

v-for指令是循环渲染一组data中的数组，v-for 指令需要以 item in items 形式的特殊语法，items 是源数据数组并且item是数组元素迭代的别名。

一、基本用法：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | *<!DOCTYPE html>*  <html lang="en">  <head>      <meta charset="UTF-8">      <script type="text/javascript" src="../assets/js/vue.js"></script>      <title>V-for 案例</title>  </head>  <body>      <h1>v-for指令用法</h1>      <hr>      <div id="app">         <ul>             <li v-for="item in items">                  {{item}}             </li>         </ul>      </div>        <script type="text/javascript">          var app=new Vue({              el:'#app',              data:{                  items:[20,23,18,65,32,19,54,56,41]              }          })      </script>  </body>  </html> |

这是一个最基础的循环，先在js里定义了items数组，然后在模板中用v-for循环出来，需要注意的是，你需要那个html标签循环，v-for就写在那个上边。

二、排序

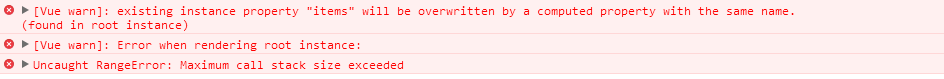
我们已经顺利的输出了我们定义的数组，但是我需要在输出之前给数组排个序，那我们就用到了Vue的computed:属性。



|  |  |
| --- | --- |
| 1  2  3  4  5 | computed:{      sortItems:function(){            return this.items.sort();      }  } |

我们在computed里新声明了一个对象sortItems，如果不重新声明会污染原来的数据源，这是Vue不允许的，所以你要重新声明一个对象。

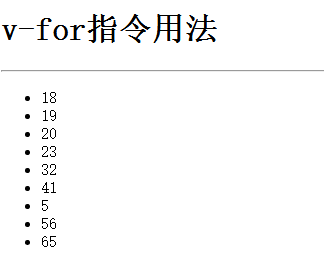
如果不重新声明报错：



如果一切顺利的话，你已经看到了结果，但是这个小程序还是有个小Bug的，现在我把数组修改成这样。

|  |  |
| --- | --- |
| 1 | items:[20,23,18,65,32,19,5,56,41] |

我们把其中的54修改成了5，我们再看一下结果，发现排序结果并不是我们想要的。



我们可以自己编写一个方法sortNumber，然后传给我们的sort函数解决这个Bug。

|  |  |
| --- | --- |
| 1  2  3 | function sortNumber(a,b){              return a-b    } |

用法

|  |  |
| --- | --- |
| 1  2  3  4  5 | computed:{      sortItems:function(){      return this.items.sort(sortNumber);      }  } |

经过一番折腾，我们终于实现了真正的数字排序，这是在工作中非常常用的，一定要学好，记住,以后用到了, 直接拿代码走就成.

三、对象循环输出

我们上边循环的都是数组，那我们来看一个对象类型的循环是如何输出的。

我们先定义个数组，数组里边是对象数据

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | students:[    {name:'changjiang',age:32},    {name:'Panda',age:30},    {name:'PanPaN',age:21},    {name:'King',age:45}  ] |

在模板中输出

|  |  |
| --- | --- |
| 1  2  3  4  5 | <ul>     <li v-for="student in students">         {{student.name}} - {{student.age}}     </li>  </ul> |

加入索引序号：

|  |  |
| --- | --- |
| 1  2  3  4  5 | <ul>    <li v-for="(student,index) in students">      {{index}}：{{student.name}} - {{student.age}}    </li>  </ul> |

排序，我们先加一个原生的对象形式的数组排序方法：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | //数组对象方法排序:  function sortByKey(array,key){      return array.sort(function(a,b){        var x=a[key];        var y=b[key];        return ((x<y)?-1:((x>y)?1:0));     });  } |

有了数组的排序方法，在computed中进行调用排序

|  |  |
| --- | --- |
| 1  2  3 | sortStudent:function(){       return sortByKey(this.students,'age');  } |

第4节：v-text & v-html

我们已经会在html中输出data中的值了，我们已经用的是{{xxx}},这种情况是有弊端的，就是当我们网速很慢或者javascript出错时，会暴露我们的{{xxx}}。Vue给我们提供的v-text,就是解决这个问题的。我们来看代码：

|  |  |
| --- | --- |
| 1 | <span>{{ message }}</span>=<span v-text="message"></span><br/> |

如果在javascript中写有html标签，用v-text是输出不出来的，这时候我们就需要用v-html标签了。



|  |  |
| --- | --- |
| 1 | <span v-html="msgHtml"></span> |

双大括号会将数据解释为纯文本，而非HTML。为了输出真正的HTML，你就需要使用v-html 指令。

需要注意的是：在生产环境中动态渲染HTML是非常危险的，因为容易导致XSS攻击。所以只能在可信的内容上使用v-html，永远不要在用户提交和可操作的网页上使用。

完整代码：

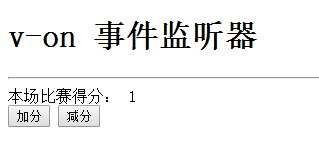
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | *<!DOCTYPE html>*  <html lang="en">  <head>      <meta charset="UTF-8">      <script type="text/javascript" src="../assets/js/vue.js"></script>      <title>v-text & v-html 案例</title>  </head>  <body>      <h1>v-text & v-html 案例</h1>      <hr>      <div id="app">          <span>{{ message }}</span>=<span v-text="message"></span><br/>          <span v-html="msgHtml"></span>      </div>        <script type="text/javascript">          var app=new Vue({              el:'#app',              data:{                  message:'hello Vue!',                  msgHtml:'<h2>hello Vue!</h2>'              }          })      </script>  </body>  </html>  个人习惯, {{}}插值表达式写法简单 |

第5节：v-on：绑定事件监听器

v-on 就是监听事件，可以用v-on指令监听DOM事件来触发一些javascript代码。

一、使用绑定事件监听器，编写一个加分减分的程序。

效果如图



程序代码



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | *<!DOCTYPE html>*  <html lang="en">  <head>      <meta charset="UTF-8">      <script type="text/javascript" src="../assets/js/vue.js"></script>      <title>v-on事件监听器</title>  </head>  <body>      <h1>v-on 事件监听器</h1>      <hr>      <div id="app">         本场比赛得分： {{count}}<br/>         <button v-on:click="jiafen">加分</button>         <button v-on:click="jianfen">减分</button>        </div>        <script type="text/javascript">          var app=new Vue({              el:'#app',              data:{                  count:1              },              methods:{                  jiafen:function(){                      this.count++;                  },                  jianfen:function(){                      this.count--;                  }              }          })      </script>  </body>  </html> |

我们的v-on 还有一种简单的写法，就是用@代替。

|  |  |
| --- | --- |
| 1 | <button @click="jianfen">减分</button> |

我们除了绑定click之外，我们还可以绑定其它事件，比如键盘回车事件v-on:keyup.enter,现在我们增加一个输入框，然后绑定回车事件，回车后把文本框里的值加到我们的count上。

绑定事件写法：

|  |  |
| --- | --- |
| 1 | <input type="text" v-on:keyup.enter="onEnter" v-model="secondCount"> |

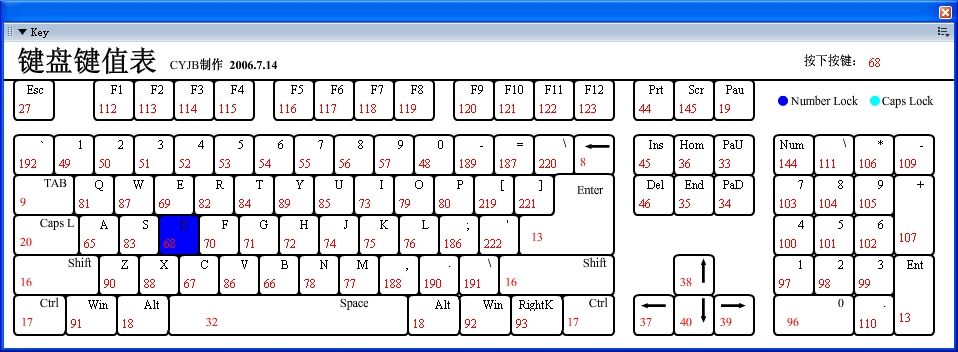
javascript代码：



|  |  |
| --- | --- |
| 1  2  3 | onEnter:function(){       this.count=this.count+parseInt(this.secondCount);  } |

因为文本框的数字会默认转变成字符串，所以我们需要用parseInt()函数进行整数转换。

你也可以根据键值表来定义键盘事件：



第6节：v-model指令

v-model指令，我理解为绑定数据源。就是把数据绑定在特定的表单元素上，可以很容易的实现双向数据绑定。

一、我们来看一个最简单的双向数据绑定代码：

html文件：

|  |  |
| --- | --- |
| 1  2  3  4  5 | <div id="app">      <p>原始文本信息：{{message}}</p>      <h3>文本框</h3>      <p>v-model:<input type="text" v-model="message"></p>  </div> |

javascript代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var app=new Vue({    el:'#app',    data:{         message:'hello Vue!'    }  }) |

二、修饰符

* .lazy：取代 imput 监听 change 事件。
* .number：输入字符串转为数字。
* .trim：输入去掉首尾空格。

三、文本区域加入数据绑定

|  |  |
| --- | --- |
| 1 | <textarea  cols="30" rows="10" v-model="message"></textarea> |

四、多选按钮绑定一个值

|  |  |
| --- | --- |
| 1  2  3 | <h3>多选按钮绑定一个值</h3>  <input type="checkbox" id="isTrue" v-model="isTrue">  <label for='isTrue'>{{isTrue}}</label> |

五、多选绑定一个数组

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <h3>多选绑定一个数组</h3>         <p>              <input type="checkbox" id="Changjiang" value="Changjiang" v-model="web\_Names">              <label for="Changjiang">Changjiang</label><br/>              <input type="checkbox" id="Panda" value="Panda" v-model="web\_Names">              <label for="Changjiang">Panda</label><br/>              <input type="checkbox" id="PanPan" value="PanPan" v-model="web\_Names">              <label for="Changjiang">PanPan</label>              <p>{{web\_Names}}</p>         </p> |

六、单选按钮绑定数据

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <h3>单选按钮绑定</h3>  <input type="radio" id="one" value="男" v-model="sex">  <label for="one">男</label>  <input type="radio" id="two" value="女" v-model="sex">  <label for="one">女</label>  <p>{{sex}}</p> |

第7节：v-bind 指令

v-bind是处理HTML中的标签属性的，例如<div></div>就是一个标签，<img>也是一个标签，我们绑定<img>上的src进行动态赋值。

html文件：

|  |  |
| --- | --- |
| 1  2  3 | <div id="app">      <img v-bind:src="imgSrc"  width="200px">  </div> |

在html中我们用v-bind:src=”imgSrc”的动态绑定了src的值，这个值是在vue构造器里的data属性中找到的。

js文件：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var app=new Vue({      el:'#app',      data:{            imgSrc:'http://baidu.com/wp-content/uploads/2017/02/vue01-2.jpg'       }  }) |

我们在data对象在中增加了imgSrc属性来供html调用。

v-bind 缩写

|  |  |
| --- | --- |
| 1  2  3  4 | <!-- 完整语法 -->  <a v-bind:href="url"></a>  <!-- 缩写 -->  <a :href="url"></a> |

绑定CSS样式

在工作中我们经常使用v-bind来绑定css样式：

在绑定CSS样式是，绑定的值必须在vue中的data属性中进行声明。

1、直接绑定class样式

2、绑定classA并进行判断，在isOK为true时显示样式，在isOk为false时不显示样式。

3、绑定class中的数组

4、绑定class中使用三元表达式判断

5、绑定style

6、用对象绑定style样式

第8节：其他内部指令(v-pre & v-cloak & v-once)

v-pre指令

在模板中跳过vue的编译，直接输出原始值。就是在标签中加入v-pre就不会输出vue中的data值了。

|  |  |
| --- | --- |
| 1 | <div v-pre>{{message}}</div> |

这时并不会输出我们的message值，而是直接在网页中显示{{message}}

v-cloak指令

在vue渲染完指定的整个DOM后才进行显示。它必须和CSS样式一起使用，

|  |  |
| --- | --- |
| 1  2  3 | [v-cloak] {    display: none;  } |

|  |  |
| --- | --- |
| 1  2  3 | <div v-cloak>    {{ message }}  </div> |

v-once指令

在第一次DOM时进行渲染，渲染完成后视为静态内容，跳出以后的渲染过程。

|  |  |
| --- | --- |
| 1  2 | <div v-once>第一次绑定的值：{{message}}</div>  <div><input type="text" v-model="message"></div> |