目录

[VueJS 1](#_Toc14811)

[复习 3](#_Toc12454)

[一、Vue 4](#_Toc6300)

[1.1事件 4](#_Toc3369)

[1.1.1事件修饰符 4](#_Toc13797)

[1.1.2常见修饰符 4](#_Toc17786)

[1.1.3鼠标修饰符 5](#_Toc25233)

[1.1.3系统键修饰符 5](#_Toc11319)

[1.1.4键盘事件修饰符 5](#_Toc7982)

[1.2类的绑定 6](#_Toc32382)

[1.3样式绑定 6](#_Toc35)

[下午复习 8](#_Toc1001)

[1.4模板指令 9](#_Toc26659)

[1.4.1条件模板指令 9](#_Toc4034)

[1.4.2显隐指令 9](#_Toc29489)

[1.4.3循环模板指令 9](#_Toc29809)

[1.4.4模板元素 10](#_Toc32443)

[1.5支付宝注册 11](#_Toc17324)

[1.6过渡 14](#_Toc24473)

[1.6.1进场动画 14](#_Toc18688)

[1.6.2动画事件 14](#_Toc7410)

[1.6.3多元素过渡 15](#_Toc22939)

[1.6.4动画模式 15](#_Toc32593)

[1.6.5列表过渡 16](#_Toc11147)

# 复习

MVVM M， V， VM

Vue M V VM

new Vue({ el, data }) $el, \_data, $data, 属性

数据绑定 特性

数据丢失 1 值类型 2 新成员 3 新属性 4 未初始化的

$set(target, key, value)

webpack ES 6, ES Module

resolve: { extensions, alias }

插值 {{}}

v-bind:key=”value” :key=”value”

v-text v-html v-once

过滤器 {{ data | filter(arg1, arg2) | filter2 }}

Vue.filter(name, fn(data, ...arg) {return })

计算属性数据 computed: { key() {} }

数据双向绑定 v-model

v-cloak [v-cloak] v-cloak

单选框的数据双向绑定 1 同一个 2 value 3 checked 4 默认值

多选框 1 不同 2 布尔值 :true-value, :false-value 3 checked 4 默认值

下拉框 select v-model options value multiple

数据监听 watch: {key(value, oldValue) {this}}

状态数据

# 一、Vue

## 1.1事件

通过v-on指令来绑定DOM事件

语法 v-on:click=”fn()”

参数集合可有可无

v-on指令的语法糖是@，所以可以用@绑定事件

@click=”fn()”

事件回调函数定义在methods属性中，

methods是专门为vue实例化对象定义方法的属性（不局限于事件回调函数）

定义方法，会添加给vue实例化对象自身因此，能够访问vue实例化对象，就可以方法这些方法。

事件回调函数的参数

没有添加参数集合，默认有一个参数，就是事件对象

添加了参数集合，默认没有参数，使用什么参数要传递什么数据，

使用事件对象传递$event

注意：事件对象是源生的事件对象，因此可以使用源生的DOM API操作它。

vue中的事件，没有实现事件委托，是将事件直接绑定给元素自身。

### 1.1.1事件修饰符

在jquery中，为了做某些事情（例如阻止冒泡，取消默认行为，识别按键，识别鼠标按钮等等），我们要处理事件对象，vue为了简化这个过程，提供了事件修饰符。

语法 v-on:click.修饰符=”fn()”

### 1.1.2常见修饰符

stop 阻止冒泡

prevent 取消默认行为

self 触发事件的元素与绑定事件的元素是同一个元素

once 单次触发。

注意：修饰符还可以组合使用，

|  |
| --- |
| <div id="app">  <!-- 绑定事件 -->  <button v-on:click="clickBtn">btn-1</button>  <button v-on:click="clickBtn(100, msg, true, $event)">btn-2</button>  <!-- @语法糖 -->  <button @click="clickBtn(100, msg, true, $event)">btn-3</button>  <!-- 事件修饰符 -->  <!-- <div @click="parent"> -->  <!-- 触发事件的元素与绑定事件的元素是同一个元素 -->  <div @click.self="parent">  hello  <button @click="child">btn-1</button>  <!-- 阻止冒泡 -->  <button @click.stop="child">btn-2</button>  <a href="" @click="child">btn-3</a>  <!-- 阻止默认行为 -->  <a href="" @click.prevent="child">btn-4</a>  <!-- 单次触发 -->  <button @click.once="child">btn-5</button>  <!-- 单次触发并且阻止冒泡 -->  <button @click.stop.once="child">btn-6</button>  <!-- 阻止默认行为，并且阻止冒泡 -->  <a href="" @click.prevent.stop="child">btn-7</a>  </div>  </div> |

### 1.1.3鼠标修饰符

为了检测点击的鼠标按键，vue提供了鼠标修饰符。

left 点击左键

right 点击右键

middle 点击中间件

### 1.1.3系统键修饰符

vue为系统组合键提供了修饰符

shilft, ctrl, alt, meta

我们可以通过修饰符，使用这些组合键

在点击该按键的时候，再触发事件。

### 1.1.4键盘事件修饰符

当我们绑定键盘事件的时候（keydown，keyup，keypress），我们可以添加键盘事件修饰符，来快速的捕获某个键。

在过去，我们要获取事件对象，判断e.keyCode。

键盘事件修饰符包括：esc, tab, space, enter, delete, up, down, left, right, 以及所有字母。

|  |
| --- |
| <div id="app">  <button @click="clickBtn">btn-1</button>  <!-- 默认点击的就是左键 -->  <button @click.left="clickBtn">btn-1</button>  <button @click.right="clickBtn">btn-2</button>  <a href="" @click.prevent.right="clickBtn">btn-2</a>  <button @click.middle="clickBtn">btn-3</button>  <hr>  <!-- 系统键修饰符 -->  <button @click.ctrl="clickBtn">btn-1</button>  <button @click.shift="clickBtn">btn-2</button>  <button @click.alt="clickBtn">btn-3</button>  <button @click.meta="clickBtn">btn-4</button>  <hr>  <!-- 键盘事件修饰符 -->  <input type="text" @keyup.enter="show">  <input type="text" @keyup.delete="show">  <input type="text" @keyup.up="show">  <input type="text" @keyup.z="show">  <input type="text" @keypress="show">  </div> |

## 1.2类的绑定

为了动态的为元素添加类，我们可以通过v-bind指令，动态绑定类，有三种绑定方式

第一种 :class=”{}”

key 表示一组类的名称（包含空格）

value 是否保留这组类

true 保留类

false 删除类

第二种 :class=”[]”

每一个成员代表一组类（包含空格）

第三种 :class=”str”

代表一组类（包含空格）

我们可以使用字符串拼接形式。

|  |
| --- |
| 1. <!-- 绑定字符串 --> 2. <!-- <div :class="'blog ' + cls" @mouseover="showBlog" @mouseleave="hideBlog"> --> 3. <!-- 数组 --> 4. <!-- <div :class="['blog', cls]" @mouseover="showBlog" @mouseleave="hideBlog"> --> 5. <!-- 对象 --> 6. <div :class="{ 7. blog: true, 8. show: cls 9. }" @mouseover="showBlog" @mouseleave="hideBlog"> 10. <span>博客</span> 11. <ul> 12. <li>博客评论</li> 13. <li>未读提醒</li> 14. </ul> 15. </div> |

## 1.3样式绑定

样式的绑定也有三种方式

第一种方式 :style=”{}”

key 表示样式名称

建议使用驼峰式命名

value 样式的属性值

第二种方式 :style=”[]”

每一个成员代表一组样式对象

key 表示样式名称

建议使用驼峰式命名

value 样式的属性值

第三种方式 :style=”str”

行内式样式字符串，可以使用字符串拼接的技术。

|  |
| --- |
| 1. <div class="goods"> 2. <p> 3. <label for="">尺寸</label> 4. <span :style="{ borderColor: span[0] }" @click="choose(37)">37</span> 5. <span :style="{ borderColor: span[1] }" @click="choose(38)">38</span> 6. <span :style="{ borderColor: span[2] }" @click="choose(39)">39</span> 7. </p> 8. <p> 9. <label for="">颜色</label> 10. <img :class="img[0]" src="img/01.jpg" alt=""> 11. <img :class="img[1]" src="img/02.jpg" alt=""> 12. <img :class="img[2]" src="img/03.jpg" alt=""> 13. </p> 14. </div> |

## 下午复习

事件 v-on:click=”fn” @click

methods

没有参数集合 $event, ($event)

常见修饰符 stop, prevent, self, once

鼠标 left， right， middle

系统键 shift, alt, ctrl, meta

键盘事件 keyup, keydown, keypress,

esc, tab, space, enter, delete, up, down, left, right, 字母

绑定类 :class=”{}” :class=”[]” :class=”str”

绑定样式 :style=”{}” :style=”[]” :style=”str”

## 1.4模板指令

模板指令就是用来控制元素创建的指令。

### 1.4.1条件模板指令

模拟了js中的if语句。

通过v-if来定义。控制元素是否创建。

属性值是true 创建该元素

属性值是false 删除该元素

是真正的创建和删除，不是显示和隐藏

与v-if相关的组合指令有：v-else， v-else-if

### 1.4.2显隐指令

通过v-show指令，控制元素的显示和隐藏，

不是模板指令，没有删除或者创建元素。

实现原理：通过切换样式：display: none实现的。  
 与v-if的区别

1 实现原理

v-if：创建和删除元素

v-show：显示和隐藏元素，没有创建或者删除元素。

2 组合指令

v-if可以与v-else以及v-else-if一起使用

v-show不能

注意：由于显隐指令v-show性能更高，所以工作中更常用。

|  |
| --- |
| 1. <div id="app"> 2. <!-- <h1 v-if="isShow === true">爱创课堂</h1> 3. <h1 v-else-if="isShow === 'hello'">hello</h1> 4. <h1 v-else>专业前端培训学校</h1> --> 5. <h1 v-show="isShow">爱创课堂</h1> 6. <!-- 不能与v-else一起使用 --> 7. <!-- <h1 v-else>专业前端培训学校</h1> --> 8. </div> |

### 1.4.3循环模板指令

vue模仿for in循环，提供了循环模板指令。

循环模板指令有两种语法。

v-for=”item in data”

v-for=”(item, index) in data”

in 表示关键字

item 表示成员值

index 表示索引值

data 表示在vue中定义的数据，有三类数据：

第一类，data是数组。

item表示成员值

index表示索引值

第二类，data是对象，

item表示属性值 value

index表示属性名称 key

第三类，data是数字，表示循环data次

item从1计数

index从0计数

注意：在v-for循环中，我们要为元素设置key属性，并且属性值是唯一的。

我们通常可以设置索引值（index）或者数据的唯一属性，如id。

后面我们使用vue cli的时候，不设置key会抛出错误

### 1.4.4模板元素

想同时控制多个兄弟元素，我们可以将指令委托给父元素，但是这么做会引入多余的元素，

为了避免引入多余的元素，vue建议我们使用模板元素。

在html中，定义模板有两种方式

第一种方式：script模板标签

<script type="text/template"></script>

第二种方式：html5提供了模板元素：template，

vue建议我们使用template元素定义模板。

模板元素特点：跟普通元素一样，可以添加属性，指令，子元素等等，但就是不会渲染出来。

注意：在模板元素中，使用v-for指令，不要设置key属性。

|  |
| --- |
|  |

## 1.5支付宝注册

<https://memberprod.alipay.com/account/reg/enterpriseIndex.htm>

<div id="app">

<!-- 创建新闻 -->

<ul>

<!-- <li v-for="item in news">{{item}}</li> -->

<!-- 数组 -->

<!-- <li v-for="(item, index) in news" :key="index">{{item}}--{{index}}</li> -->

<!-- 数字 -->

<!-- <li v-for="(item, index) in num">{{item}}--{{index}}</li> -->

<!-- 对象 -->

<!-- <li v-for="(item, index) in obj">{{item}} -- {{index}}</li> -->

<!-- 斑马线 -->

<!-- <li v-for="(item, index) in news" :key="index" :style="{

backgroundColor: index % 2 ? 'skyblue' : '#efefef'

}">{{item}}--{{index}}</li> -->

<!-- <div v-for="(item, index) in news" :key="index">

<li>{{item}}--{{index}}</li>

<hr>

</div> -->

<!-- 避免引入新的元素，使用模板元素 -->

<!-- 删除key属性 -->

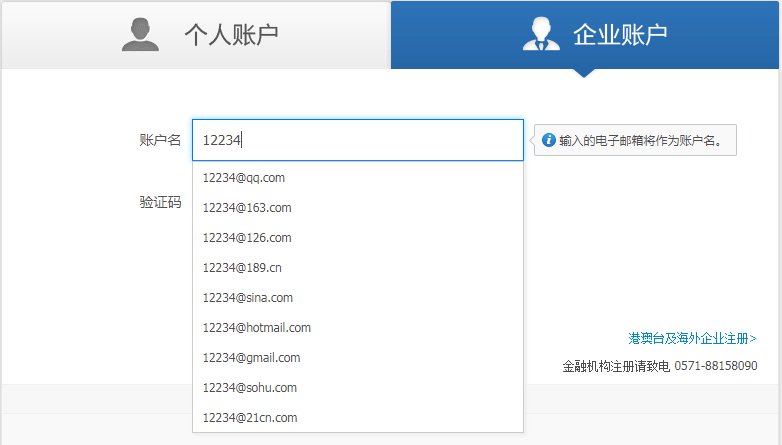
<template v-for="(item, index) in news">

<li>{{item}}--{{index}}</li>

<hr>

</template>

</ul>

</div>

|  |
| --- |
| <div id="app"><div class="alipay"><label for="username">账户名</label><inputtype="text"id="username"autocomplete="off"v-model="msg"@keydown.up.prevent="choosePrevLi"@keyup.down="chooseNextLi"@keyup.enter="chooseCurrentLi"><!-- ul的显隐与msg有关,还与显示的邮箱个数有关 --><ul v-show="msg && dealEmail.length"><liv-for="(item, index) in dealEmail":key="index"@click="clickChooseLi">{{dealMsg}}@{{item}}.<template v-if="item == '189'">cn</template><template v-else>com</template></li></ul></div></div>  1. // 实例化 2. new Vue({ 3. // 容器 4. el: '#app', 5. // 数据 6. data: { 7. // 邮箱后缀 8. email: ['qq', '163', '126', '189', 'sina', 'hotmail', 'gmail', 'sohu', '21cn'], 9. // 输入的内容 10. msg: '', 11. // 当前li的索引值 12. num: 0, 13. // 是否是第一次点击向下箭头 14. isFirstChooseNextLi: true 15. }, 16. // 计算属性数据 17. computed: { 18. // 处理msg 19. dealMsg(v) { 20. // console.log(this.msg.slice(0, this.msg.indexOf('@'))) 21. // 获取@符号位置 22. let index = this.msg.indexOf('@') 23. // 如果@存在 24. if (index >= 0) { 25. // 截取msg 26. return this.msg.slice(0, index); 27. } 28. // else { 29. // 返回msg 30. return this.msg 31. // } 32. }, 33. // 对邮箱做处理 34. dealEmail() { 35. // 获取@符号位置 36. let index = this.msg.indexOf('@'); 37. // 如果@存在 38. if (index >= 0) { 39. // 获取@后面的字符串 40. let str = this.msg.slice(index + 1); 41. // console.log(str) 42. return this.email.filter(item => { 43. // 邮箱名少了后缀名称 44. let emailName = item + (item == 189 ? '.c' : '.co'); 45. // 返回值就是过滤的条件 46. // 邮箱是以str开头的 47. return emailName.indexOf(str) === 0; 48. }) 49. } 50. // 没有输入@，显示所有邮箱 51. return this.email; 52. } 53. }, 54. // 方法 55. methods: { 56. // 点击li，选中li 57. clickChooseLi(e) { 58. // 获取元素 59. let li = e.target; 60. // 获取内容 61. let text = li.innerHTML; 62. // 存储数据 63. this.msg = text; 64. }, 65. // 根据长度获取当前索引值的方法 66. getIndex(len) { 67. // 根据li的长度，因此num的值，获取真正的索引值（0-len之间） 68. // 不论num多么大，对len取余： 就是 -len到len之间的数 69. // let num = this.num % len; 70. // 此时再加上len，就是 0到2\*len之间的数 71. // num = num + len; 72. // 此时再对len取余，就是 0-len之间的数。 73. // num = num % len; 74. // 合并一句话 75. return (this.num % len + len) % len 76. }, 77. // 改变li的样式 78. changeLisStyle() { 79. // 获取所有的li 80. let lis = document.getElementsByTagName('li'); 81. // 获取长度 82. let len = lis.length 83. // console.log(len) 84. // 获取当前的真实的索引值 85. let index = this.getIndex(len); 86. // console.log(index) 87. // 排它法添加类 88. // 清空所有li的类 89. // for () {} 90. // 转换成数组 91. // Array.from(lis) 92. // 使用数组迭代器方法 93. // .forEach(li => li.className = '') 94. // console.log(lis) 95. // 设置特殊li的类 96. // lis[index].className = 'hover' 97. // 合并语句 98. Array.from(lis) 99. // 找到特殊li添加类，其它的删除类 100. .forEach((li, i) => li.className = i === index ? 'hover' : '') 101. }, 102. // 选中上一个li 103. choosePrevLi(e) { 104. // 更新索引值 105. this.num--; 106. // 更改li的样式 107. this.changeLisStyle(); 108. // 点击过向上箭头，也算是点击向下箭头了（索引值已经更新过来了） 109. this.isFirstChooseNextLi = false; 110. }, 111. // 选中下一个li 112. chooseNextLi() { 113. // 如果不是第一次点击向下箭头，应该更新num 114. if (!this.isFirstChooseNextLi) { 115. // 更新索引值 116. this.num++; 117. } 118. // 更改li的样式 119. this.changeLisStyle(); 120. // 到这里，已经点击完向下箭头了 121. this.isFirstChooseNextLi = false; 122. }, 123. // 选中当前的li 124. chooseCurrentLi() { 125. // 获取所有的li 126. let lis = document.getElementsByTagName('li'); 127. // 获取长度 128. let len = lis.length; 129. // 获取当前的li的索引值 130. let index = this.getIndex(len); 131. // 更新内容 132. this.msg = lis[index].innerHTML; 133. } 134. } 135. }) |

## 1.6过渡

我们可以通过v-show指令，显示或者隐藏元素

我们还可以通过v-if指令，创建或者删除元素，

vue允许我们在显示与隐藏，或者创建与删除过程中，添加动画

这里的动画是通过css3实现的，因此要定义样式。

我们通过transition元素（组件）为内部的元素添加过渡动画。

通过name定义过渡动画类的名称

例如 name=”ickt”,此时在显示与隐藏，或者创建与删除过程中会出现六个类

显示过程中：.ickt-enter, .ickt-enter-to, .ickt-enter-active

隐藏过程中：.ickt-leave, .ickt-leave-to, .ickt-leave-active

注意：

显示过程中：类包含enter

元素由隐藏的状态变成显示的状态

隐藏过程中：类包含leave

元素由显示的状态变成隐藏的状态

我们可以基于这些类，为元素添加过度动画



### 1.6.1进场动画

我们在刷新页面的时候（第一次渲染的时候）希望添加动画，

为transition添加appear属性即可。

### 1.6.2动画事件

在js中，监听css动画可以通过事件：

webkitTransitionStart, webkitTransitionEnd

webkitAnimationStart, webkitAnimationEnd

vue允许我们为transtion绑定事件，监听动画。

显示过程中

before-enter 显示动画开始

enter 显示动画进行中

after-enter 显示动画结束

隐藏过程中

before-leave 隐藏动画开始

leave 隐藏动画进行中

after-leave 隐藏动画结束

我们通过@语法糖绑定这些事件。

|  |
| --- |
| 1. <div id="app"> 2. <button @click="toggle()">切换</button> 3. <!-- transion添加过渡 --> 4. <transition 5. name="demo" 6. appear 7. @before-enter="beforeEnter" 8. @enter="enter" 9. @after-enter="afterEnter" 10. @before-leave="beforeLeave" 11. @leave="leave" 12. @after-leave="afterLeave" 13. > 14. <!-- <div class="demo" v-show="isShow"></div> --> 15. <div class="demo" v-if="isShow"></div> 16. </transition> 18. </div> |

### 1.6.3多元素过渡

在对多个元素添加过渡的时候，要设置key属性，并且属性值是唯一的。

### 1.6.4动画模式

通过mode属性定义动画模式

in-out 新元素动画先执行，再执行当前元素动画

out-in 当前元素动画先执行，再执行新元素动画。

|  |
| --- |
| 1. <div id="app"> 2. <button @click="toggle()">切换</button> 3. <!-- transion添加过渡 --> 4. <!-- mode="out-in" --> 5. <transition 6. name="demo" 7. mode="in-out" 8. > 9. <div class="demo" v-if="isShow" key="1"></div> 10. <div class="demo2" v-else key="2"></div> 11. </transition> 13. </div> |

### 1.6.5列表过渡

当我们对v-for指令创建的列表添加过渡的时候，我们要使用transition-group元素（组件）。

它最终会渲染成真实的元素，默认是span。通过tag属性可以定义渲染元素的类型。

此时渲染列表的时候，一定要设置key属性，并且属性值是唯一的。

|  |
| --- |
| 1. <div id="app"> 2. <button @click="add">增加元素</button> 3. <button @click="itemRandom">打乱顺序</button> 4. <transition-group tag="div" name="ickt" class="list"> 5. <!-- 保证key的属性值是不变的 --> 6. <span v-for="item in nums" :key="item">{{item}}</span> 7. </transition-group> 8. </div> |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |