

Vue学习第一天

课程安排

1. 总共的课程27天
2. 三个阶段 Vue基础、PC端项目和移动端项目
3. 基础课程7天

1. 首先呢，我们来看一下课程的安排
2. 课程天数从上一期的17天增加到27天
3. 27天的课程分为3个阶段，vue基础、pc端项目和移动端项目
4. Vue的基础课程，总共7天，会讲到Vue语法，指令，还有一些小型的项目来把前面的知识串起来

Vue.js介绍

1. [官方文档](#)
2. Vue是一个js框架，对比jquery, **不需要操作DOM，精力集中在数据上。**
3. 大家就业后更有可能就是写Vue.js

1. 首先来了解一下我们要学的这个vue.js到底是个啥东东呢？好，来它的官网看看，是怎么介绍它自己的。
2. 带着大家找一个官方文档
3. 建议大家把Vue官方网站收藏了。官方文档整体翻译质量高，还有文档本身很详细,后面的学习会经常参考到官网
4. 渐进式Javascript框架，我们学过哪些js框架呢？回顾一下jquery.js的使用。相比jquery,Vue.js基本上不用操作数据
5. 接着我们来看一下，Vue在行业内的地位。大家毕业后，很有可能就是写Vue.js的。

Vue.js的HelloWorld

[传送门](#)

1. Vue.js的学习会参考Vue官方文档，以项目驱动学习。
2. vue读音,两种都要了解
 1. /vju:/
 2. /uju:/
3. 渐进式: vue能做小的项目也能做大的项目

1. 那么vue.js怎么用呢？在文档的这里有一个起步，这个就是入门教程。接下来呢，我们就参考这份文档，把vue.js运行起来
2. 我说一下我是怎么学习vue.js。分期乐会员的前端架构是我搭的，基于Vue，但是在此之前我都没接触过Vue，我是在入职前参考这份文档，从前到尾看了一遍，重点的代码敲了，然后实际工作中遇到问题，再回头看看这份文档。妥妥的。
3. 那么接下来Vue的学习，会以文档的知识点为主线，并不会覆盖文档各个方面，重要的地方或实际工作中用得多的会重点练习。以项目驱动，在实际的应用场景里面会重点补一下知识点和复习已经学过知识点。
4. vue的读音
5. 渐进式解释
6. 学习Vue.js的知识储备
7. copy helloworld代码，并对比vue.js开发和生产版本
8. 再下一代代码结构，我们猜想，运行结果会是什么。好，我们运行起来看看，Hello Vue.js。下一节我们再来解释为什么页面上会显示Hello vue.js

Vue基本使用解析

使用vue.js

1. 引入vue.js
2. dom结构，这个Vue管理的容器
3. 实例化Vue
 1. el:'id选择器' 关联Vue管理的容器
 2. data是一个对象，里面的属性，在Vue管理的容器中用{}渲染

```
1  <!-- {{message}} -->
2  <div id="container" class="app">
3    {{ message }}
4    <h2>{{info}}</h2>
5  </div>
6  <!-- 开发环境版本，包含了有帮助的命令行警告 jquery.js-->
7  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
8  <!-- 生产环境版本，优化了尺寸和速度 jquery.min.js-->
9  <!-- <script src="https://cdn.jsdelivr.net/npm/vue"></script> -->
10 <script>
11   var app = new Vue({
12     // el: '#container',
13     // el: '.app',
14     el: 'div',
15     data: {
16       // message: 'Hello vue!'
17       message: 'Hello world!',
18       info: '今天天气如何?'
19     }
20   })
21 </script>
```

1. Vue是一个构造函数，谁引入的呢。Vue构造函数是vue.js引入的
2. el:'#app'猜想选择器关联到dom结构，改一下id验证猜想。el的作用是关联Vue管理的容器

3. dom容器外，并不能被渲染。
4. 既然是选择器，我们还学过哪些选择器呢？class选择器和tag选择器。推荐用id选择器。
5. 画图结合模板语法的猜想data数据通过{{}}渲染，修改数据，验证猜想
6. 总结

js表达式

[传送门](#)

js表达式：结果是一个值的js语句

1. 在Vue的基本使用里面，我们用{{}}渲染数据对吧，message和info都是字符串，其实{{}}不仅仅可以渲染字符串，它可以包含一个js表达式。什么是js表达式呢，看传送门。
2. 那么什么是js表达式呢，看代码
3. 回顾使用Vue使用三个步骤
4. 渲染message
5. {{num+1}} {{isRed?'red':''}} {{const PI=3.141592653}} 运算和三元运算是可以的，但是const不行
6. 我的理解是：js表达式就是结果是一个值的js语句
7. 那么{{1+1}} 可以吗？{{if}}能吗？
8. {{ }} 胡子语法（画图），插值语法和mustache语法是同一个意思。文档里面胡子语法和插值语法混着用，看文档的时候要理解

```
1 <!-- dom结构 -->
2 <div id="container">
3   <h2>{{ message }}</h2>
4   <h2>{{ num + 100 }}</h2>
5   <h2>{{ isRed ? 'red' : '' }}</h2>
6   <!-- <h2>{{const PI = 3.141592653}}</h2> -->
7   <h2>{{ 1 + 1 }}</h2>
8 </div>
9 <!-- 引入vue.js -->
10 <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
11
12 <!-- 实例化Vue.js -->
13 <script>
14   new Vue({
15     el: '#container',
16     data: {
17       message: '还是吃得太饱了',
18       num: 100,
19       isRed: true
20     }
21   })
22 </script>
```

Vue指令

传送门

指令 (Directives) 是带有 `v-` 前缀的特殊特性。

vue提供给HTML标签新增的属性。

用法和HTML属性一样。 `v-text="js表达式"`

```
1 | 
```

1. Vue并不只是用来渲染数据，接下来介绍Vue的一个重要特性。--vue指令
2. 文档对指令的定义
3. 翻译成我的理解 vue提供给HTML标签的新增属性
4. HTML img的属性引出指令的语法 `v-text="js表达式"`
5. 那么接下来，我们会花一天多的时间介绍一系列Vue指令

v-text指令(textContent)

传送门

1. `v-text`把值作为**文本插入**到所在的标签之间
2. Vue内部是`textContent`实现的
3. 会覆盖掉标签之间的文本
4. 简写`{{}}`，简写用得更多
5. 不能解析html

```
1 | <div id="app">
2 |   <h2>{{ message }}</h2>
3 |   <h2 v-text="message"></h2>
4 |   <h2 v-text="message">标签之间的文本</h2>
5 |   <h2>之前的文本--{{ message }}--之后的文本</h2>
6 |   {{ alink }}
7 | </div>
8 | <script src="./lib/vue.js"></script>
9 | <script>
10 |   /*
11 |     v-text:把值作为文本插入到它所在的标签之间 textContent，会覆盖掉标签之间的文案
12 |     简写就是{{}}，推荐使用{{}}
13 |     v-text不能解析html
14 |   */
15 |   const app = new Vue({
16 |     el: '#app',
17 |     data: {
18 |       message: '今天天气如何',
19 |       alink: '<a href="http://www.baidu.com">百度</a>'
20 |     }
21 |   })
22 | </script>
```

1. 接下来我们来学习第一个指令，v-text，text是什么意思呢。文本，那肯定跟文本有关对吗？
2. 看代码，还记得使用vue的三个步骤吗？好，我这弄一个自定义代码段，一键生成基本结构。
3. vue-tpl生成vue模板。不需要再回顾使用Vue的三个步骤，建议同学们多敲。自定义代码片段明天再给。
4. vue.js的引入放在本地。浏览器运行的时候，读取的是本地文件，vue.js在本地，加载速度快。
5. v-text=""的值同样为js表达式，试试message.结果一样，为什么呢。{}是v-text简写
6. v-text的作用，内部实现textContent
7. 对于所有指令，都推荐简写。如果原始写法更简单，为什么要弄个一个简写呢。
8. {}简写有v-text没有的特性, 不会覆盖
9. 不能解析html
10. 工作中几乎全用{}

v-html指令(innerHTML)

[传送门](#)

1. v-html是把值作为html插入到所在的标签之间，内部实现是innerHTML。会覆盖标签之间的html
2. 没有简写

```
1 <div id="app">
2   <h2>{{ a link }}</h2>
3   <h2 v-html="a link"></h2>
4   <h2 v-html="a link">标签之间的文本</h2>
5 </div>
6 <script src="./lib/vue.js"></script>
7 <script>
8   /*
9       v-html把值作为html插入到它所在的标签之间。底层实现 innerHTML
10      没有简写
11   */
12   new Vue({
13     el: '#app',
14     data: {
15       a link: '<a href="http://www.baidu.com">百度</a>'
16     }
17   })
18 </script>
```

1. 刚才不是说v-text指令不能解析html吗？那来一个能解析html的指令v-html
2. {{html str}}不能解析，换v-html引出v-html定义
3. 内部实现是innerHTML
4. 会覆盖标签之间的文本
5. 没有简写，用得不是很频繁

v-on指令

[基本使用](#)

1. v-on:事件名="事件处理方法"
2. 事件处理方法声明在methods对象里边，methods和el、data并列
3. 简写@事件名="事件处理方法" 当然推荐用简写
4. 事件名和原生的事件名一致，可以是dblclick,click,mouseover,keyup,keyenter....
5. methods是一个对象，里面方法声明有三种，推荐用ES6简洁写法

```
1 <div id="app">
2   <input type="button" value="点我呀" v-on:click="sayHello">
3   <input type="button" value="简写" @click="sayHello">
4   <input type="button" value="双击" @dblclick="sayHello">
5 </div>
6 <script src="../lib/vue.js"></script>
7 <script>
8   /*
9     v-on:事件名="事件处理方法"
10    事件处理方法声明在methods对象里面,methods和el、data并列
11    简写 @事件名="事件处理方法" 当然推荐用简写
12    事件名和原生相同,事件名可以是dblclick,click,mouseover,focus,blur,keyup,keyenter...
13   */
14   const app = new Vue({
15     el: "#app",
16     data: {
17
18     },
19     methods: {
20       // 方法的简洁写法
21       //sayHello(){
22       //  alert('hello')
23       // },
24       // sayHello:function(){
25       //  alert('ok')
26       // },
27       sayHello: () => {
28         alert('ok')
29       },
30     },
31   });
32 </script>
```

1. 接着我们学一个让页面有交互的指令，v-on指令，看一下官方文档
2. 注册事件，回顾原来HTML注册事件的方法
3. 引出v-on注册指令。方法声明在methods里面。方法声明methods对象里面。
4. 简写
5. 事件名和原生的事件名一致，演示dblclick
6. methods是一个对象，里面方法声明有三种，推荐ES6的简洁写法。

事件参数

1. 默认事件参数是event对象

2. 事件处理方法的调用如果给括号的话，无法接受Vue传过来的event对象
3. 方法传参和原来HTML注册事件方法没什么分别

```
1 <div id="app">
2   <button @click="sayHello">点我呀</button>
3   <button @click="callMethod()">有括号</button>
4   <button @click="bigger(99)">逼格</button>
5 </div>
6 <script src="./lib/vue.js"></script>
7 <script>
8   /*
9     事件处理方法，默认接受一个事件对象，少部分情况下会用到的
10    传参和原来HTML注册事件没什么不同
11   */
12   const app = new Vue({
13     el: '#app',
14     data: {},
15     methods: {
16       sayHello(event) {
17         // alert('ok')
18         console.log(event)
19       },
20       callMethod(event) {
21         console.log(event) //undefined
22       },
23       bigger(num) {
24         alert(99 * 77)
25       }
26     }
27   })
28 </script>
```

1. 回顾事件注册
2. 方法里面可以接受一个事件对象，Vue帮我们传过来.有些情况下会用到的。
3. 事件处理方法的调用给括号的情况下，无法接受到Vue传过来的event对象
4. 传参和原来HTML注册事件没什么分别

事件修饰符

[传送门](#)

使用方法：v-on:事件名.修饰符 = "事件处理方法"

```
1 <!-- 阻止单击事件继续传播 -->
2 <a v-on:click.stop="doThis"></a>
3
4 <!-- 提交事件不再重载页面 -->
5 <form v-on:submit.prevent="onSubmit"></form>
6
7 <!-- 修饰符可以串联 -->
```

```

8 <a v-on:click.stop.prevent="doThat"></a>
9
10 <!-- 只当在 event.target 是当前元素自身时触发处理函数 -->
11 <!-- 即事件不是从内部元素触发的 -->
12 <div v-on:click.self="doThat">...</div>
13
14 <!-- 只有在 `key` 是 `Enter` 时调用 `vm.submit()` -->
15 <input v-on:keyup.enter="submit">

```

```

1 <div id="app">
2   <input type="text" value="这是一个寂寞的天" @keyup="keyupHandler" />
3   <input type="text" value="下着有些伤心的雨" @keyup.enter="enterHandler" />
4 </div>
5 <script src="../lib/vue.js"></script>
6 <script>
7   const app = new Vue({
8     el: '#app',
9     data: {},
10    methods: {
11      // enter键抬起的时候, 打印input value
12      keyupHandler(event) {
13        //判断是否是enter键
14        if (event.keyCode === 13) {
15          console.log(event.target.value)
16        }
17      },
18      enterHandler(event){
19        console.log(event.target.value)
20      }
21    }
22  })
23 </script>

```

1. 监听事件还有一些高级的用法，事件修饰符，看文档了解到.stop替代 event.stopPropagation()
2. 举例@keyup.enter替代event.keyCode===13逻辑
3. 修饰符的语法
4. 各个修饰符的作用

vue方法中的this

[传送门](#)

方法中的 `this` 自动绑定为 Vue 实例。

1. 方法里面的this就是Vue实例
2. 方法里面的this可以访问到data的属性和methods的方法. 用法this.
3. Vue里面属性值改变，对应视图将会"响应"

1. 在事件处理方法是可写this，功能强大，使用灵活
2. 方法中的this是什么呢？举例打印app及this，看似相等。全等试试，结果全等。

3. 重点看一下打印的app, app.data.message没有, 直接就有app.message. 添加两个属性, 发现app依然可以访问到。所以data和methods里面的属性直接设置到vue实例
4. 那么this可访问到data及methods的属性, 不需要this.data.
5. this.message打印, this.message赋值, 页面动态改变, 引出属性值改变, 对应视图会响应。
6. 方法声明箭头函数里面的this是window, 建议用es6对象里方法的简洁写法

```
1 <div id="app">
2   <h2>{{message}}</h2>
3   <input type="button" value="修改data" @click="changeData">
4 </div>
5 <script src="../lib/vue.js"></script>
6 <script>
7   const app = new Vue({
8     el: "#app",
9     data: {
10       message: '中分还带波浪',
11       person: {
12         name: 'Joven',
13         age: 18
14       }
15     },
16     methods: {
17       // changeData(){
18       //   // console.log(this===app)
19       //   console.log(this.message)
20       //   this.message = '李晨又分手了'
21       // },
22       // 箭头函数的地方式, 方法里面的this是window
23       changeData: ()=>{
24         console.log(this)
25       },
26       run(){
27         console.log('biu biu biu ...')
28       }
29     },
30   });
31   console.log(app)
32 </script>
```

js表达式的作用域

[传送门](#)

表达式会在所属 Vue 实例的数据作用域下作为 JavaScript 被解析

js表达式里面的变量是Vue实例的属性

1. data和methods里面的属性在表达式里面可以直接使用, 不要加this

1. js表达式的作用域是什么意思呢。我们看文档, copy文档说明。解释。

2. copy上一节课的例子，提到sayHello里面就一句话，能否写在行内。原来HTML注册事件也要以写行内。copy,点行内没有效果。
3. 看文档js表达式里面的作用域是Vue实例。data和methods里面的属性在表达式里面可以直接使用，不要加this
4. 行内放console.log没效果。能理解吗？需要写在方法里面。
5. 回顾一下，我们之写的一些页面，js表达式里面写的变量都是Vue实例的。

```
1 <div id="app">
2   <h2>{{msg}}</h2>
3   <input type="button" value="点我呀" @click="sayHello">
4   <input type="button" value="行内" @click="msg='中分还带波浪。。。'">
5   <input type="button" value="打印" @click="print">
6 </div>
7
8 <script src="./lib/vue.js"></script>
9 <script>
10   const app = new Vue({
11     el: "#app",
12     data: {
13       msg: '这是一个寂寞的天。。。'
14     },
15     methods: {
16       sayHello(){
17         this.msg='中分还带波浪。。。'
18       },
19       print(){
20         console.log('结婚证')
21       }
22     },
23   });
24 </script>
```

v-bind指令

[传送门](#)

动态地绑定一个或多个特性

v-bind指令动态绑定HTML标签的属性。当属性值不是一个写死的值时，都需要用动态绑定v-bind

1. v-bind指令的使用 v-bind:属性名="js表达式"
2. v-bind: src="imgUrl", src属性绑定到imgUrl, 改变imgUrl, 会动态改变src的值
3. v-bind : 可以简写为 :属性名 = "js表达式" 当然推荐简写
4. v-bind:class="需要添加的class"
5. v-bind:disabled="是否添加disabled属性"

1. 看文档v-bind的定义，翻译成自己的理解：当属性值不是一个写死的值时，都需要用动态绑定v-bind
2. 显示图片的需求，定义变量，{{}}显示URL的改变，猜想 src="{{}}"这种语法，其实行不通, 引入v-bind
3. 查看运行后的html呢。

4. 解释v-bind:src="imgUrl"的作用，当imgUrl变了，用了v-bind的地方也会重新解析。
5. 简写的方式
6. 第二小节讲v-bind:class 用得也很多，先讲解v-bind:class 绑定样式。 查看HTML结构生成的样式。
7. v-bind:class的值为isRed:true确定。
 7. v-bind:class 三元运算 点击切换样式
8. 忘记写:会怎么样
9. disabled属性的介绍，绑定disabled的html结构，button切换disabled的值。v-bind:disabled="是否添加disabled属性"

```
1 <div id="app">
2   <h2>{{ imgUrl }}</h2>
3   
4   
5 </div>
6 <script src="./lib/vue.js"></script>
7 <script>
8   const app = new Vue({
9     el: '#app',
10    data: {
11      imgUrl: './img/01.png',
12      imgUrl2: './img/cat.gif'
13    },
14    methods: {
15      changeImg() {
16        this.imgUrl = './img/03.png'
17      }
18    }
19  })
20 </script>
```

```
1 <div id="app">
2   <div :class="bgRed"></div>
3   <div :class="isRed?'red':''" @click="isRed=!isRed"></div>
4 </div>
5 <script src="./lib/vue.js"></script>
6 <script>
7   /*
8     v-bind: class="js表达式"
9   */
10  new Vue({
11    el: "#app",
12    data: {
13      bgRed: 'red',
14      isRed: true
15    }
16  });
17 </script>
```

```

1 <div id="app">
2   <button @click="isDisabled=!isDisabled">缴械</button>
3   <input type="text" :disabled="isDisabled" />
4 </div>
5 <script src="./lib/vue.js"></script>
6 <script>
7   const app = new Vue({
8     el: '#app',
9     data: {
10       isDisabled: true
11     }
12   })
13 </script>

```

Demo-计数器



实现步骤

1. 数字的显示 1.声明数字 data.num 2.显示数字 {{num}}
2. + 按钮功能实现 1.+注册点击事件 @click:add num++ 2.-注册点击事件 @click:sub num--
3. 数字范围是0-10，边界问题的处理
 1. 数字为0的时候
 1. -按钮添加disabled属性
 1. v-bind:disabled="num===0"
 2. -按钮添加'disabled'样式
 1. v-bind:class="num==0?'disabled':""
 2. 数字为10的时候
 1. +按钮添加disabled属性
 1. v-bind:disabled="num===10"
 2. +按钮添加'disabled'样式
 1. v-bind:class="num==10?'disabled':""

注意点

1. v-bind:disabled="是否添加disabled属性" 值为true的时候，按钮disabled; 否则不disabled
2. v-bind:class="添加的样式" 三元运算的值为className

1. 打开页面，查看页面上的行为
2. 检查HTML，disabled类名和disabled属性
3. 思路分析

Demo-图片切换

图片切换



上一张

下一张

实现步骤

1. 图片的展示

1. 图片数组装四张图片imgList:[img1,img2,img3,img4] 下标0,1,2,3
2. 数组下标index:0
3. 取出数组中一张图片 imgList[index]
4. 显示图片v-bind:src="imgList[index]"

2. 上一张和下一张功能

1. 上一张 @click:pre index--
2. 下一张 @click:next index++

3. 图片循环展示

1. 点下一张的时候，如果当前显示最后一张，下一张是第一张
2. 点上一张的时候，如果当前显示是第一张，上一张是最后一张

注意点

1. 图片的展示 v-bind:src="imgList[index]"

2. 数组的边界的问题

1. 下一页：数组元素最后一项->第一项
2. 上一页：第一项->数组最后一项

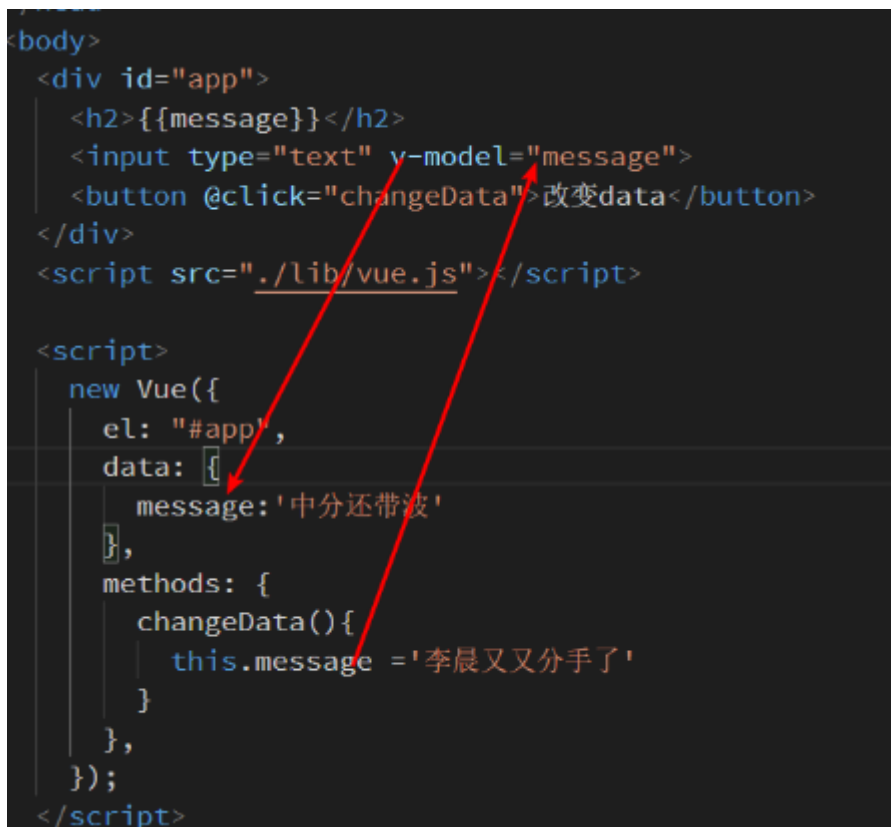
画图分析有四张图片分析 图片数组及下标：src

表单输入绑定 v-model

传送门

你可以用 `v-model` 指令在表单 `<input>`、`<textarea>` 及 `<select>` 元素上创建双向数据绑定。

1. v-model只能用在表单标签上, input, textarea, select
2. 双向数据绑定：
 1. 表单用户输入改变引起data.message改变
 2. data.message改变引起表单用户输入值的改变
3. v-model是用来获取用户输入的



```
<body>
  <div id="app">
    <h2>{{message}}</h2>
    <input type="text" v-model="message">
    <button @click="changeData">改变data</button>
  </div>
  <script src="./lib/vue.js"></script>

  <script>
    new Vue({
      el: "#app",
      data: {
        message: '中分还带波'
      },
      methods: {
        changeData(){
          this.message = '李晨又又分手了'
        }
      }
    });
  </script>
```

```
1 <div id="app">
2   <h2>{{message}}</h2>
3   <input type="text" v-model="message">
4   <button @click="changeData">改变data</button>
5 </div>
6 <script src="./lib/vue.js"></script>
7
8 <script>
9   new Vue({
10     el: "#app",
11     data: {
12       message: '中分还带波'
13     },
14     methods: {
15       changeData(){
```

```

16         this.message = '李晨又又分手了'
17     }
18 },
19 });
20 </script>

```

1. 查看文档
2. v-model只能用在表单元素
3. 举例说明联动效果
4. 改变数据输入值改变，双向数据绑定。画图并打字解释
5. v-model获取用户输入

v-for指令

[传送门](#)

用 `v-for` 指令基于一个数组来渲染一个列表

1. v-for用来遍历数组并生成多个标签
2. v-for作用在需要重复的元素上
3. 语法 v-for="元素别名 in 数组" 或者 v-for="(元素别名,下标别名) in 数组"
4. 在v-for作用的标签之间就可以用使用元素别名和下标别名
5. in是固定的，不能变

```

1 <div id="app">
2   <ul>
3     <li v-for="(movie,index) in movieList">
4       {{movie}} --{{index}}
5     </li>
6   </ul>
7 </div>
8 <script src="./lib/vue.js"></script>
9 <script>
10   new Vue({
11     el: "#app",
12     data: {
13       // 电影列表
14       movieList: [
15         '战狼1',
16         '战狼2',
17         '战狼3'
18       ]
19     }
20   });
21 </script>

```

1. 看文档，详细说明官方的例子。v-for用来遍历数组生成多个标签，v-for作用在需要重复的元素上
2. 举例说明，检查页面生成的li元素

3. 元素别名可以换，in不能改

Demo-记事本

1561886193732

实现步骤

1. 展示任务列表
 1. todoList:['睡觉觉','吃饭饭','打豆豆']
 2. v-for="元素别名 in 数组" li就可以使用元素别名
2. 用户输入内容，回车，添加任务
 1. 获取用户输入 v-model:inputVal
 2. enter键抬起时候触发事件 @keyup.enter="addTodo"
 3. 添加任务 todoList.push(inputVal)
 4. 清空输入
3. 双击删除任务
 1. 双击事件 @dblclick:delTodo(index)
 2. index来自v-for
 3. todoList.splice(从哪个下标开始删除，删除多少项)

注意点

1. v-for="(元素别名,下标别名) in 数组"
2. v-model修饰符
 - 修饰符：
 - `.lazy` - 取代 input 监听 change 事件
 - `.number` - 输入字符串转为有效的数字
 - `.trim` - 输入首尾空格过滤
3. array.splice(从哪个下标开始删除，删除元素的个数)

没啥好说的，就是要讲得细

最后优化说明v-model.trim

v-if,v-else-if,v-else指令

传送门

`v-if` 指令用于条件性地渲染一块内容。这块内容只会在指令的表达式返回 truthy 值的时候被渲染。

1. v-if="js表达式"和v-else-if="js表达式"，如果表达式值为true，才会渲染该元素
2. v-else 以上的条件不满足时，添加所有元素

```
1 <div id="app">
```



```

2   <h2>老弟，你今年{{ age }}岁</h2>
3   <input type="text" v-model="age" />
4   <h2 v-if="age<18">偷偷摸摸去网吧！</h2>
5   <h2 v-else-if="age<25">光明正大去网吧</h2>
6   <h2 v-else>你到了晚婚的年龄了</h2>
7 </div>
8 <script src="./lib/vue.js"></script>
9 <script>
10   const app = new Vue({
11     el: "#app",
12     data: {
13       age: 18
14     }
15   });
16 </script>

```

1. 看官方文档，把文档的例子敲一下。
2. 然后举例年龄
3. 提一下else-if的用法

v-show指令

[传送门](#)

`v-show` 只是简单地切换元素的 CSS 属性 `display`。

1. v-show 隐藏元素通过改变元素css属性display:none
2. v-if 隐藏元素直接移除dom
3. 对于频繁切换隐藏和显示的元素，就用v-show

```

1   <div id="app">
2     <button @click="isShow=!isShow">切换显示</button>
3     <h2>使用v-if</h2>
4     <h2 v-if="isShow">中分还带波浪，是不是很萌呢</h2>
5     <h2>使用v-show</h2>
6     <h2 v-show="isShow">中分还带波浪，是不是很萌呢</h2>
7   </div>
8   <script src="./lib/vue.js"></script>
9   <script>
10     new Vue({
11       el: '#app',
12       data: {
13         isShow: true
14       }
15     })
16   </script>

```

1. 查看官方文档，简要说明v-show的功能和使用
2. 举例对比v-if和v-show

总结

插件推荐

Vetur 让vscode提高对vue的支持，比如高亮，比如图标...

Vue 2 Snippets vue关键语法的提示

用户代码片段
