Vue学习第2天

反馈

回顾

v-if,v-else-if,v-else指令

传送门

是否渲染当前的DOM

1. v-if="是否渲染当前的dom"
2. v-else-if="是否渲染当前的dom"
3. v-else 以上条件不满足时，渲染当前的dom

```
<div id="app">
  <button @click="awesome=!awesome">切换</button>
  <h1 v-if="awesome">Vue is awesome!</h1>
  <h1 v-else>very bad</h1>

  <h2>----------------------------</h2>
  <h2>老弟，你今年{{age}}岁了</h2>
  <input type="text" v-model="age">
  <!-- 不同年龄做的事情 -->
  <h2 v-if="age<18">去网吧</h2>
  <h2 v-else-if="age<25">来黑马学习</h2>
  <h2 v-else>晚婚了</h2>
</div>
<script src="./lib/vue.js"></script>
<script>
  const app = new Vue({
    el: "#app",
    data: {
```

```
      awesome: false,
      age: 18
    }
  });
</script>
```

v-show指令

传送门

`v-show` 只是简单地切换元素的 CSS 属性 `display`。

使用方法 v-show="是否展示当前dom"

1. v-if 隐藏元素会移除dom
2. v-show隐藏元素，用css display:none
3. 对于频繁切换元素的显示与隐藏，建议用v-show

```
<div id="app">
  <button @click="isShow=!isShow">切换显示与隐藏</button>
  <h2>使用v-if</h2>
  <p v-if="isShow">这是一个寂寞的天</p>
  <h2>使用v-show</h2>
  <p v-show="isShow">这是一个寂寞的天</p>
</div>
<script src="./lib/vue.js"></script>
<script>
  const app = new Vue({
    el: "#app",
    data: {
      isShow: true
    }
  });
</script>
```

v-cloak指令

传送门

这个指令保持在元素上直到关联实例结束编译。和 CSS 规则如 `[v-cloak] { display: none }` 一起用时，这个指令可以隐藏未编译的 Mustache 标签直到实例准备完毕。

1. 添加这个指令后，vue解析完后，会移除这个属性
2. 配合[v-cloak] { display: none }来隐藏胡子语法

注意点：

1. disable cache，每次都会去请求vue.js
2. slow 3g，让请求vue.js更慢

```
<!DOCTYPE html>
<html lang="en">

<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>
    [v-cloak] {
      display: none;
    }
  </style>
</head>

<body>
  <div id="app">
    <h1>吃了吗</h1>
    <h3 v-cloak>{{msg}}</h3>
  </div>
  <!-- <script src="./lib/vue.js"></script> -->
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  <script>
    const app = new Vue({
      el: "#app",
      data: {
        msg: '李晨又又又分手。。。'
      }
    });
```

```
  </script>
</body>

</html>
```

v-once（了解）

传送门

只渲染元素和组件**一次**。

```
<div id="app">
  <h2>{{message}}</h2>
  <h2 v-once>{{message}}</h2>
  <input type="text" v-model="message">
</div>
<script src="./lib/vue.js"></script>
<script>
  const app = new Vue({
    el: "#app",
    data: {
      message: '这是一个寂寞的天'
    }
  });
</script>
```

v-pre（了解）

Vue不解析，原样输出

```
<div id="app">
  <h2>{{message}}</h2>
  <h2 v-pre>{{message}}</h2>
</div>
<script src="./lib/vue.js"></script>
<script>
  const app = new Vue({
    el: "#app",
    data: {
      message: '这是一个寂寞的天'
    }
  });
</script>
```

Demo-天知道

实现步骤

1. 输入城市，显示城市
    1. 获取用输入的城市名 v-model:city
    2. 显示 {{city}}
2. 回车，发出请求
    1. enter键抬起的事件 @keyup.enter:queryWeather
    2. 发出请求 $.ajax(url, success:function(获取数据))
    3. 接口：http://wthrcdn.etouch.cn/weather_mini?city=深圳，是get请求
3. 展示天气列表
    1. 天气数组 forecastList = res.data.forecast
    2. v-for 渲染列表
4. 展示emoji表情
    1. 如果item.type有云的话，展示云表情
        1. indexOf判断字符串包含字符串
    2. emoji表情window7不展示、
5. 隐藏胡子语法

注意点

1. indexOf的用法 a.indexOf(b), b在a中的下标，否则-1。 是否包含！-1
2. a.includes(b)返回true/false
3. $.ajax({success:function(res){这里的this不是Vue实例}}) 因为function会绑定this到当前调用的对象。

   所以最好用箭头函数，箭头函数默认不会绑定this

Demo-聊天机器人

实现步骤

1. 展示聊天消息
   1. 很多条消息，得有一个消息数组messageList[]
   2. 一条消息有两个属性，一个是消息体本身，还有一个消息的所属者。所以消息必须是一个对象。

      用isme来标识是我的消息还是姐姐的消息
   3. vfor遍历数组，渲染li
   4. v-bind:isme,来渲染不同的样式

```
messageList:[
   {
     msg:'你好',
     isme:true
   },
   {
     msg:'好呀',
     isme:false
   },
   {
     msg:'吃饭了吗?',
     isme:true
   },
   {
     msg:'滚！',
     isme:false
   }
]
```

2. 我输入消息，回车或者点击发送的时候，添加我的消息
    1. 获取我输入的消息 v-model.trim:inputVal,
    2. 回车或者发送 @keyup.enter/@click:chat
    3. 添加我的消息

```
messageList.push({
  content:inputVal,
  //标志是我的消息
  isme:true
})
```

3. 姐姐根据我的消息，请求接口获取消息，添加姐姐的消息
    1. 根据inputVal请求接口
    2. $.ajax success:()=>{}

        - 请求地址：http://www.tuling123.com/openapi/api
        - 请求方法：post
        - 请求参数：key,info
        -
        - 2162602fd87240a8b7bba7431ffd379b
        - a618e456f0744066840ceafb6a249d9d
        - d7c82ebd8b304abeacc73b366e42b9ed
        - 7b1cf467c0394dd5b3e49f32663f8b29
        - 9fbb98effab142c9bb324f804be542ba

    3. 添加姐姐的消息

```
messageList.push({
  content:'',
  // 标志不是我的消息
  isme:false
})
```

    4. v-cloak解决胡子语法的问题

注意点

1. 消息格式，得有一个属性区分是姐姐的消息还是我的消息，所以用了一个对象。isme来区分。

2. v-bind:src和v-bind:class配合isme来区分姐姐和我的消息样式
3. v-cloak vue解析完后移除这个属性，一般结合display:none样式，在解析前隐藏元素

留有问题

1. 添加姐姐的消息后，需要手动滚动滚动条.

template结合v-if

传送门

把一个 `<template>` 元素当做不可见的包裹元素，并在上面使用 `v-if`。最终的渲染结果将不包含 `<template>` 元素。

1. template是Vue提供的标签，有包裹元素的功能，和div
2. 最终不会渲染
3. template雷锋

滚动底部-Vue异步更新

异步更新

Vue 在更新 DOM 时是**异步**执行的

Vue会把数据的更新，缓冲起来，批量地进行更新DOM

用定时器，强制让姐姐消息添加的DOM更新后，再执行滚动到底部。

```
setTimeout(()=>{
  $('.content').scrollTop(88888)
},0)
```

课后学习

`js事件循环`

Vue生命周期钩子函数

传送门

每个 Vue 实例在被创建时都要经过一系列的初始化过程——例如，需要设置数据监听、编译模板、将实例挂载到 DOM 并在数据变化时更新 DOM 等。同时在这个过程中也会运行一些叫做**生命周期钩子**的函数，这给了用户在不同阶段添加自己的代码的机会。

1. 钩子函数，就是回调函数
2. Vue生命周期就是Vue实例创建到销毁过程中，会经历一些重要的阶段
3. vue生命周期钩子函数，就是Vue实例从创建到销毁过程中，有8个重要时间点，在这个时间点，Vue提供回调函数通知我们。回调函数里面可以自定义逻辑。
4. 生命周期钩子的 `this` 上下文指向调用它的 Vue 实例
5. 生命周期钩子函数和data、el、methods是平级的
6. updated是数据更新，而且对应的dom更新完成后，触发updated

日期格式化库 moment.js

传送门

```
// 猜想是当前的时间，默认的格式化
document.write(moment().format('YYYY-MM-DD HH:mm:ss a'))
```

计算属性

传送门

对于任何复杂逻辑，你都应当使用**计算属性。**

对于data里面的属性，如果不想原样的输出，都应该使用计算属性

使用

1. 计算属性是作为computed的一个方法
2. computed是一个对象，和data、el、methods、updated是平级的
3. 计算属性这个方法必须返回一个值，这个值就是计算属性
4. 计算属性这个方法所依赖的属性有改变的时候，会重新计算
5. 计算属性方法名和data里面的属性是一样的使用

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

```
  <h2>{{message}}</h2>
  <input type="text" v-model="message">
  <h2>字符的长度{{messageLen}}</h2>

</div>
<script src="./lib/vue.js"></script>
<script>
  const app = new Vue({
    el: "#app",
    data: {
      message: '这是一个寂寞的天，'
    },
    computed: {
      //message的长度
      messageLen() {
        console.log('我执行了。。。')
        return this.message.length
      }
    },
  });
</script>
```

```
<div id="app">
  <h3>你成为京东plus会员的日期{{date}}</h3>
  <h3>成为会员{{vipDays}}天</h3>
  <input type="text" v-model="date">

</div>
<script src="./lib/vue.js"></script>
<script>
  const app = new Vue({
    el: "#app",
    data: {
      date: '2019-8-21'
    },
    computed: {
      vipDays() {
        // (当前时间的毫秒数-成为会员日期的毫秒数)/(每天的毫秒数)
        //'2019/8/21
```

```
        return Math.ceil((Date.now() - new Date(this.date.replace(/\-/g, '/')).getTime()) / (24 * 60 * 60 * 1000))
      }
    }
  });
</script>
```

Demo-品牌管理

实现步骤

1. 展示列表
    1. 列表数组 brandList:[]
    2. 列表展示v-for 渲染tr
2. 点击删除链接，删除对应项
    1. 点击删除 @click.prevent:delBrand(index)
    2. index来自于vfor
    3. 操作数组 brandList.splice(从哪一项开始删除,1)
3. 新增品牌
    1. 控制弹层的显示与隐藏
        1. 弹层的显示与隐藏 v-show="isShow"
        2. 点击新增品牌@click="isShow=true"
        3. 添加或者取消 隐藏弹层 @click isShow=false
    2. 弹层输入框,输入品牌，点添加按钮，新增品牌
        1. 获取用户输入的品牌 v-model.trim:inputVal
        2. 添加按钮 @click: addBrand
        3. 新增品牌，给数组添加一项

```
brandLsit.push({
  name:'商品名称',
  time:当前时间
})
```

4. 用户输入搜索关键字，展示品牌名称包含搜索结果的品牌列表
    2. 用户输入的关键字 v-model.trim:keyword

3. 搜索结果的品牌列表是一个过滤后的数组, 需要用到计算属性filterBrandList
    1. 遍历brandList里面的每一项，如果某一项的品牌名称包含keywords,这一项就应该添加到新数组里面

注意点

1. 搜索结果列表背后数据是原列表和关键字过滤后的数组，是计算属性
2. 弹层的显示与隐藏 v-show

总结

回顾

练习

1. todoMVC作业
2. 其他资料中的练习案例