

Vue学习第2天

反馈

回顾

1. vue基本使用
2. vue指令

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <title>Document</title>
8     <style>
9       #app > div {
10         width: 100px;
11         height: 100px;
12         border: 1px solid #666;
13         margin: 10px;
14       }
15     </style>
16   </head>
17   <body>
18     <div id="wrapper">
19       <h2>{{ message }}</h2>
20       <!-- <h2 v-text="message"></h2> -->
21       <h2 v-html="alink"></h2>
22       <input type="button" value="点我呀" @click="sayHello" />
23       <div :class="{ 'red': isRed }" @click="isRed=!isRed"></div>
24       <input type="text" v-model="message" />
25       <input type="button" value="改变data" @click="changeData" />
26       <ul>
27         <li v-for="(song,index) in songList">
28           {{ song }}--{{ index }}
29         </li>
30       </ul>
31       <p v-if="isShow">这是一个寂寞的天，下着有些伤心的雨</p>
32       <p v-show="isShow">这是一个寂寞的天，下着有些伤心的雨</p>
33     </div>
34     <!-- 引入vue.js -->
35     <script src="./lib/vue.js"></script>
36     <!-- 实例化vue -->
37     <script>
38       /*
39       Vue的指令是vue提供给HTML标签的属性
```

```

40      */
41      new Vue({
42        el: "#wrapper",
43        data: {
44          message: "还是吃得太饱",
45          alink: '<a href="http://www.baidu.com">百度</a>',
46          isRed: true,
47          songList: ["两只蝴蝶", "终于等到你", "可惜不是你"],
48          isShow: true
49        },
50        methods: {
51          sayHello() {
52            alert("ok");
53          },
54          changeData() {
55            this.message = "李晨又又分手了";
56          }
57        }
58      });
59    </script>
60  </body>
61 </html>

```

v-cloak指令

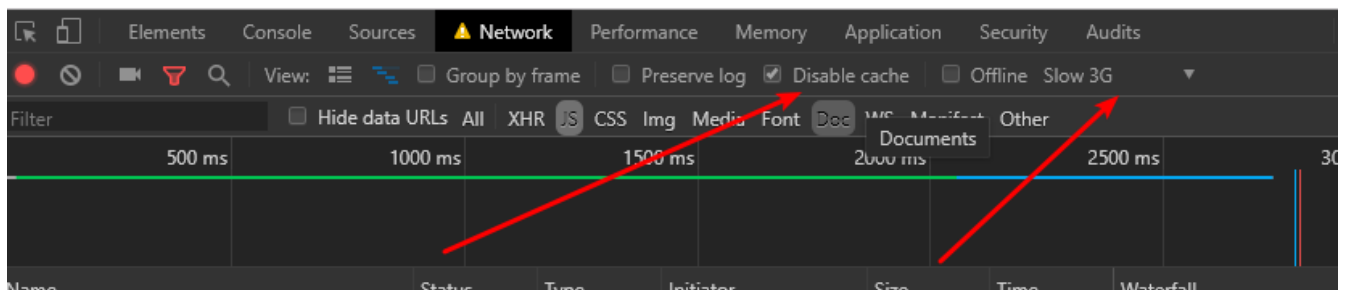
传送门

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. 添加了这个指令之后, Vue解析完后, 会移除该指令
2. v-cloak和 display:none结合使用, 可以隐藏未编译的`{}`语法

注意:

1. disable cache 请求的资源不会在浏览器缓存，下一次请求同一资源还会请求服务器
2. slow 3G 模拟低速网络



```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />

```

```

7   <title>Document</title>
8   <style>
9     [v-cloak]{
10      display: none;
11    }
12  </style>
13 </head>
14 <body>
15   <div id="app">
16     <h2 v-cloak>{{ message }}</h2>
17   </div>
18   <!-- 开发环境版本，包含了有帮助的命令行警告 -->
19   <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
20   <!-- <script src="./lib/vue.js"></script> -->
21   <script>
22     new Vue({
23       el: "#app",
24       data: {
25         message: "Hello world"
26       }
27     });
28   </script>
29 </body>
30 </html>

```

v-once (了解)

[传送门](#)

只渲染元素和组件一次。

```

1   <div id="app">
2     <h2>{{message}}</h2>
3     <h2 v-once>{{message}}</h2>
4     <input type="text" v-model="message">
5   </div>
6   <script src="./lib/vue.js"></script>
7   <script>
8     const app = new Vue({
9       el: "#app",
10      data: {
11        message: '是兄弟，就一起玩贪玩蓝月'
12      }
13    });
14  </script>

```

v-pre (了解)

跳过这个元素和它的子元素的编译过程。

```
1 <div id="app">
2   <span v-pre>{{ 这是一个寂寞的天 }}</span>
3 </div>
4 <script src="./lib/vue.js"></script>
5 <script>
6   const app = new Vue({
7     el: '#app',
8     data: {}
9   })
10 </script>
```

Demo-天知道

实现步骤

1. 输入城市，显示搜索城市
 1. 获取输入的城市 v-model.trim
 2. 显示城市 {{city}}
2. 回车，请求数据
 1. enter键点击响应@keyup.enter
 2. 请求天气 \$.ajax(url,success (获取数据))
 3. 接口 http://wthrcdn.etouch.cn/weather_mini?city=深圳
3. 展示天气搜索结果
 1. 获取天气信息 forecastList[]
 2. v-for展示天气
4. emoji表情的展示
 1. 判断item.type里面是否包含 云 雨 v-if

注意点

1. function会绑定this到function所在的对象，箭头函数不会绑定this
2. a.indexOf(b) 查找b在a字符串中的索引位置，如果没有的话返回-1
3. includes 字符串中是否包含字符

Demo-聊天机器人

实现步骤

1. 展示消息
 1. 很多条消息，得有一个消息数组messageList[]
 2. 姐姐的消息和我的消息是不一样的，所以消息得有两个属性，一个是消息体，一个是所有者（属性谁的消息）。所以一条消息必须是一个对象

```

1  messageList:
2  [
3    {
4      content: '你好',
5      isme: true
6    },
7    {
8      content: '好呀',
9      isme: false
10   },
11   {
12     content: '吃饭了吗',
13     isme: true
14   },
15   {
16     content: '滚!',
17     isme: false
18   }
19 ]

```

3. 展示消息v-for

4. v-bind: 绑定属性，根据是我的消息还是姐姐的消息，显示不同的样式

2. 我输入消息，回车，或者点击发送，添加我的消息

1. 获取输入消息 v-model.trim:inputVal

2. 回车/点击发送 @keyup.enter/@click: chat

3. 添加我的消息

```

1  messageList.push({
2    content: inputVal,
3    //标志是我的消息
4    isme: true
5  })

```

3. 姐姐回复消息

1. 添加完我的消息后，请求接口获取姐姐的消息

2. 请求接口获取姐姐的消息 \$.ajax()

```

1  请求地址：http://www.tuling123.com/openapi/api
2  请求方法：post
3  请求参数：key,info
4  2162602fd87240a8b7bba7431ffd379b
5  a618e456f0744066840ceafb6a249d9d
6  d7c82ebd8b304abeacc73b366e42b9ed
7  7b1cf467c0394dd5b3e49f32663f8b29
8  9fbb98effab142c9bb324f804be542ba

```

3. 添加姐姐的消息

```
1 messageList.push({
2   // 标志不是我的消息
3   isme: false,
4 })
```

注意点

1. 消息数组元素为一个对象，对象有isme来区分是我的消息，还是姐姐的消息
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
4. 留有问题：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是包裹元素，功能上类似于div
2. 但是并不渲染template标签

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。

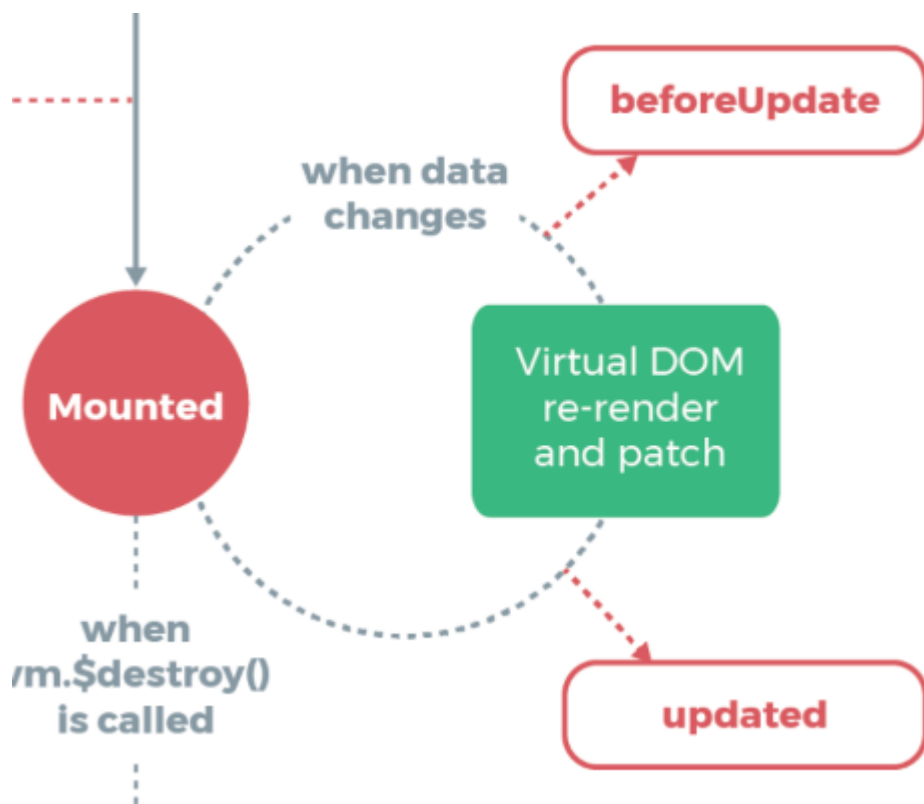
Vue会把数据的改变，缓冲起来，批量更新DOM

使用定时器强制让滚动在DOM更新完后执行

```
1 // 让消息列表滚动底部
2 setTimeout(()=>{
3   $('content').scrollTop(99999999)
4 },100)
```

Vue生命周期钩子函数

[传送门](#)



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. Vue的生命周期钩子函数是从创建到销毁，有个8个重要节点，8个节点事件发生的时候，Vue以回调函数的形式通知我们
2. 钩子函数和data，el、methods是并列的
3. 生命周期钩子的 `this` 上下文指向调用它的 Vue 实例。用methods里面的方法是一样的
4. `updated` 在数据改变，对应的视图已经更新完后，会触发updated方法.

日期格式化库 moment.js

[传送门](#)

```
1 <script>
2   //当前的时间，默认的格式化
3   document.write(moment().format('YYYY-MM-DD HH:mm:ss a'))
4 </script>
```

计算属性

[传送门](#)

任何复杂逻辑，你都应当使用**计算属性**。

1. computed和el、data同级。计算属性作为computed里面的一个方法，必须return
2. 使用的时候和data里面的属性一样
3. 计算属性所依赖的属性有变化的时候，计算属性会重新计算

```

1  <div id="app">
2    <h2>成为京东会员的日期 : {{ date }}</h2>
3    <input type="text" v-model="date" />
4    <h2>格式化后的日期 {{ formatDate }}</h2>
5    <h2>您已经京东会员{{ vipDays }}天</h2>
6  </div>
7  <script src="./lib/vue.js"></script>
8  <script src="./lib/moment.js"></script>
9  <script>
10   const app = new Vue({
11     el: '#app',
12     data: {
13       date: '2019-8-8'
14     },
15     computed: {
16       formatDate() {
17         return moment(this.date).format('YYYY-MM-DD HH:mm:ss a')
18       },
19       vipDays() {
20         console.log('重新计算')
21         return Math.ceil((Date.now() - new Date(this.date.replace(/\-/g,
22         '/' )).getTime()) / (24 * 60 * 60 * 1000))
23       }
24     })
25   </script>

```

Demo-品牌管理

手机品牌管理

品牌列表				新增品牌
编号	品牌名称	创立时间	操作	
1	红米	2019-07-24 11:48:37 am	删除	
2	oppo	2019-07-24 11:48:37 am	删除	
3	vivo	2019-07-24 11:48:37 am	删除	
4	坚果	2019-07-24 11:48:37 am	删除	
5	华为	2019-07-24 11:49:14 am	删除	
6	小米	2019-07-24 11:49:54 am	删除	

实现步骤

1. 展示列表

1. 品牌列表数组


```

1  brandList:[
2      {
3          name:'小米',
4          time:'2019-07-26 10:36:38 am'
5      },
6      {
7          name:'红米',
8          time:'2019-07-24 10:36:38 am'
9      }
10 ]

```

2. 列表的展示 v-for tr

2. 删除一项

1. 点击事件 @click:delBrand(index)
2. 数组移除元素 brandList.splice(从哪一个元素开始删除, 删除多少个)

3. 新增品牌

1. 弹层的显示与隐藏

1. 弹层 v-show="isShow"
2. 点击新增品牌, 显示 @click isShow=true
3. 添加与取消 隐藏 @click isShow=false

2. 弹层输入框, 输入品牌, 回车或者点击添加, 新增品牌

1. 获取用户输入的品牌 v-model.trim:inputVal
2. 回车或者点击添加 @keyup.enter/@click addBrand
3. 新增品牌, 给数组添加一项

```

1  brandList.push({
2      name:'商品名称',
3      time: 当前的时间
4  })

```

4. 输入关键字, 展示包含关键字的品牌名称项

1. 获取用户的输入 v-model.trim:keyword
2. 列表展示的内容是过滤后的数组, 需要用到计算属性 filterBrandList
 1. 取出brandList里面的每一项, 每一项的品牌名称是否包含keyword, 如果包含就展示这一项

注意点

1. 品牌列表是一个过滤的数组, 过滤条件是品牌名称包含搜索的关键词
2. 计算属性也是属性, 可以用在for 里面
3. 字符的非空判断建议用 `if(!str)` 等价于 `if(str=='')`
4. Array的filter返回一个符合条件的所有元素的新数组
 1. 取出数据的每一项, 根据回调函数的返回来确定是否添加元素。
 1. 更细一点, 可以举例直接返回true/false的效果
 2. 在方法里面打印, 看调用情况

3. item>50 看filter的结果

```
1 let arr =[4,6,67,78,234,345,33,543,53,34,34,534,53,534,5,345]
2
3 let filterArr = arr.filter(function(item){
4   console.log(item)
5   return item>50
6 })
7 console.log(arr,filterArr)
```

总结

回顾

练习

1. todoMVC作业
2. 其他资料中的练习案例