Vue学习第2天

反馈

回顾

1. vue基本使用
2. vue指令

```
<!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>
      #app > div {
        width: 100px;
        height: 100px;
        border: 1px solid #666;
        margin: 10px;
      }
    </style>
  </head>
  <body>
    <div id="wrapper">
      <h2>{{ message }}</h2>
      <!-- <h2 v-text="message"></h2> -->
      <h2 v-html="alink"></h2>
      <input type="button" value="点我呀" @click="sayHello" />
      <div :class="{'red':isRed}" @click="isRed=!isRed"></div>
      <input type="text" v-model="message" />
      <input type="button" value="改变data" @click="changeData" />
      <ul>
        <li v-for="(song,index) in songList">
            {{ song }}--{{ index }}
        </li>
      </ul>
      <p v-if="isShow">这是一个寂寞的天，下着有些伤心的雨</p>
      <p v-show="isShow">这是一个寂寞的天，下着有些伤心的雨</p>
    </div>
    <!-- 引入vue.js -->
    <script src="./lib/vue.js"></script>
    <!-- 实例化vue -->
    <script>
      /*
        vue的指令是vue提供给HTML标签的属性
```

```
      */
      new Vue({
        el: "#wrapper",
        data: {
          message: "还是吃得太饱",
          alink: '<a href="http://www.baidu.com">百度</a>',
          isRed: true,
          songList: ["两只蝴蝶", "终于等到你", "可惜不是你"],
         isShow:true
        },
        methods: {
          sayHello() {
            alert("ok");
          },
          changeData() {
            this.message = "李晨又又分手了";
          }
        }
      });
    </script>
  </body>
</html
```

v-cloak指令

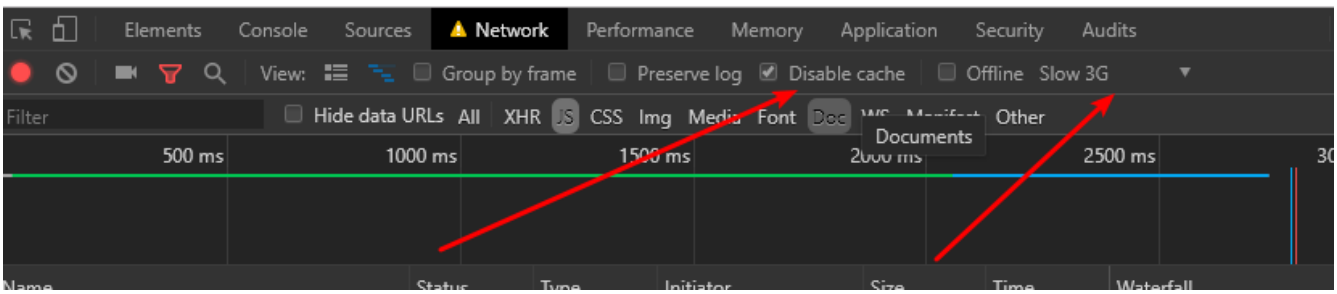
传送门

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




```
<!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">
      <h2 v-cloak>{{ message }}</h2>
    </div>
    <!-- 开发环境版本，包含了有帮助的命令行警告 -->
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <!-- <script src="./lib/vue.js"></script> -->
    <script>
      new Vue({
        el: "#app",
        data: {
          message: "Hello World"
        }
      });
    </script>
  </body>
</html>
```

1. 看效果，一闪而过。vue CDN及降网速。明显了。
2. 原因剖析 从上往上执行
3. 查看文档, 解决方案
4. 画图说明解释，如何隐藏胡子语法

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:'是兄弟，就一起玩贪玩蓝月'
    }
  });
</script>
```

v-pre（了解）

跳过这个元素和它的子元素的编译过程。

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

> 1. 看效果
> 2. 思路分析
> 3. HTML结构分析
> 4. 思路分析到（数组,ajax）
> 5. 实现
> 6. 总结

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

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

    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

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

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()

```
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({
  // 标志不是我的消息
  isme:false,
})
```

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.

> 1. 看效果，然后查看HTML，分析思路（先把消息展示出来）
> 2. 先把数据渲染出来，很多条消息，用数组。 一条消息至少有两个属性 文本，所有者 是一个对象
> 3. 显示消息 v-bind指令绑定样式
> 4. 输入消息，回车，添加我的消息
> 5. 请求添加姐姐的消息

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

> 1. v-bind:表达式里面的逻辑重复，在div包裹，并在div上用v-if判断更好
> 2. div可能有副作用，用template更好
> 3. 查看template文档，解释，用template替换div，总结

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

```
// 让消息列表滚动底部
setTimeout(()=>{
 $('.content').scrollTop(999999999)
},100)
```

> 1. 消息聊天记录的滚动问题，用console里面用 jquery强制滚动生效
> 2. 代码里姐姐聊天记录添加后强制滚动，实际的效果滚动不彻底
> 3. 解释原因是dom更新是异步的。
> 4. setTimeout延时，能解决问题

Vue生命周期钩子函数

传送门

 1563939779294

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.

> 1. 看文档，解释钩子函数就是回调函数
> 2. 解释生命周期 一个人从生到死，比如18岁，28岁，比如学前端前和学前端后
> 3. Vue的生命周期 创建到销毁，重要的节点挂载数据更新。
> 4. 从创建到销毁有8个节点，8个事件发现的时候，Vue允许我们用回调函数的形式通知我们。
> 5. 看过一下，提一下Vue生命周期图，重点看updated
> 6. 解释updated，然后拿updated解决问题
> 7. 总结

日期格式化库 moment.js

传送门

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

> 1. 查看文档
> 2. html导入moment.js，对照文档说明，输出日期格式

计算属性

传送门

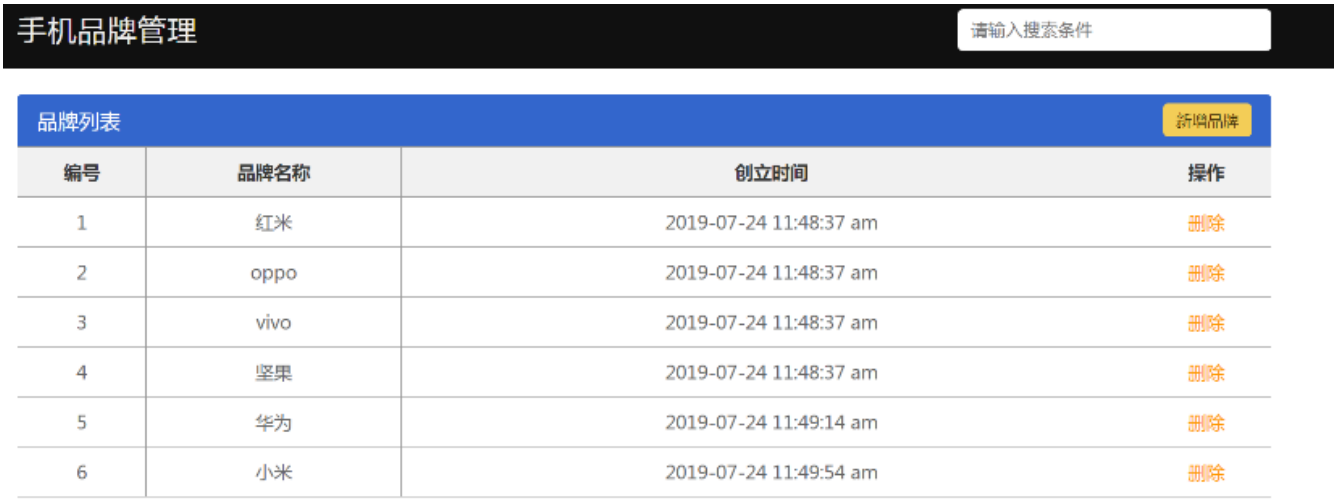
任何复杂逻辑，你都应当使用**计算属性。**

1. computed和el、data平级。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

```
<div id="app">
  <h2>成为京东会员的日期：{{ date }}</h2>
  <input type="text" v-model="date" />
  <h2>格式化后的日期 {{ formatDate }}</h2>
  <h2>您已经京东会员{{ vipDays }}天</h2>
</div>
<script src="./lib/vue.js"></script>
<script src="./lib/moment.js"></script>
<script>
  const app = new Vue({
    el: '#app',
    data: {
      date: '2019-8-8'
    },
    computed: {
      formatDate() {
        return moment(this.date).format('YYYY-MM-DD HH:mm:ss a')
      },
      vipDays() {
        console.log('重新计算')
        return Math.ceil((Date.now() - new Date(this.date.replace(/\-/g,
'/')).getTime()) / (24 * 60 * 60 * 1000))
      }
    }
  })
</script>
```

> 1. 展示一个日期，但是想按指定格式展示，不是原样输出。需要计算，放在方法里面。引出computed
> 2. 重点解释计算属性的return的值就是展示计算属性的值，
> 3. 且计算属性会重新计算，加console.log看看方法是否真的被调用了。
> 4. 总结
> 5. 再用以上的代码来一个复杂一些的你例子，说明复杂的逻辑是{{}}放不下的，适合放计算属性里面

Demo-品牌管理



实现步骤

1. 展示列表
    1. 品牌列表数组

```
brandList:[
            {
              name:'小米',
              time:'2019-07-26 10:36:38 am'
            },
            {
              name:'红米',
              time:'2019-07-24 10:36:38 am'
            }
          ]
```

    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

```
brandList.push({
  name:'商品名称',
  time: 当前的时间
})
```

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-mode.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

```
let arr =[4,6,67,78,234,345,33,543,53,34,34,534,53,534,5,345]

let filterArr = arr.filter(function(item){
  console.log(item)
  return item>50
})
console.log(arr,filterArr)
```

> 先讲正常的过滤数组的逻辑，再引出array filter语法

总结

回顾

练习

1. todoMVC作业
2. 其他资料中的练习案例