Vue学习第三天

反馈

回顾

1. 计算属性
    1. data属性，如果不想原样输出，那么可以使用计算属性
    2. 计算属性是`computed`属性里面的一个方法
    3. 方法必须有返回值
    4. 方法名可以像data的属性一样使用
    5. 方法名的值就是方法的返回值
    6. 方法所依赖的data属性有改变时，方法会重新计算
2. 方法
    1. 事件处理方法都应该被声明在`methods`
3. updated
    1. Vue生命周期钩子函数，Vue从创建到销毁过程中会经历8个重要的时间节点，在每个节点上，会以回调函数的形式通知我们，在回调函数里面我们可以自定义逻辑。
    2. updated当数据改动，对应的dom更新后，会触发updated
    3. 生命周期钩子函数和el、data是平级的

```
<div id="app">
  <h2>{{date}}</h2>
  <h2>日期格式化{{formatDate}}</h2>
  <input type="text" v-model="date">
</div>
<script src="./lib/vue.js"></script>
<script src="./lib/moment.js"></script>
<script>
  const app = new Vue({
    el: "#app",
    data: {
      date: '2019-8-22'
    },
    computed: {
```

```
      formatDate() {
        return moment(this.date).format('YYYY-MM-DD HH:mm:ss a')
      }
    }
  });
</script>
```

```
<div id="app">
  <button @click="sayHello">鹏鹏你好</button>
</div>
<script src="./lib/vue.js"></script>
<script>
  const app = new Vue({
    el: "#app",
    data: {},
    methods: {
      sayHello() {
        alert('bye bye ...')
      }
    },
  });
</script>
```

```
<div id="app">
  <button @click="isRed=!isRed">切换显示</button>
  <h2 v-show="isRed">这是一个寂寞的天</h2>
  <h2>{{ message }}</h2>
  <input type="text" v-model="message" />
  <br><br>
</div>
<script src="./lib/vue.js"></script>
<script>
  const app = new Vue({
    el: '#app',
    data: {
      message: '李晨又分手了',
      isRed: true
    },
    updated() {
      console.log('数据改变了')
    }
```

```
  })
</script>
```

滚动底部-Vue.nextTick

传送门

在下次 DOM 更新循环结束之后执行延迟回调。在修改数据之后立即使用这个方法，获取更新后的 DOM。

使用方法:

```
// 数据变更后
Vue.nextTick(()=>{
  //姐姐消息添加的数据改变，并且对应的dom更新完成后
  $('.content').scrollTop(9999999)
})

this.$nextTick(()=>{
  $('.content').scrollTop(999999)
})
```

updated: data中的数据变更，DOM更新完成后，就会触发

nextTick: 一次数据改后变，DOM更新完成后，就会触发

网络请求库axios

传送门

1. 专注于发请求
2. axios能独立使用，也能够结合vue和react等其他框架使用
3. 使用
    1. 导包 undefined
    2. 用法

```
// Make a request for a user with a given ID
```

```
axios.get('/user?ID=12345')
  .then(function (response) {
    // 成功可以在这里拿到数据
    console.log(response);
  })
  .catch(function (error) {
    // 处理失败逻辑
    console.log(error);
  })
  .finally(function () {
    // 成功或失败都需要做的处理
  });

axios.post('/user', {
    firstName: 'Fred',
    lastName: 'Flintstone'
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

注意点

1. .then的回调里面，成功了可以在这里拿到数据
2. .catch的回调里面，处理失败
3. .finally的回调，成功或者失败都会进入这里
4. 回调函数里面，如果用function,this就是window,推荐用箭头函数

```
<!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>
  </head>
  <body>
```

```
    <button onclick="search()">查询天气</button>
    <div id="app">
      <input type="text" v-model="city" @keyup.enter="queryWeather" />
    </div>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <script src="./lib/vue.js"></script>
    <script>
      function search() {
        axios
          .get('http://wthrcdn.etouch.cn/weather_mini?city=深圳')
          .then(function(res) {
            // 成功可以在这里拿到数据
            console.log(res.data.data.forecast)
          })
          .catch(function(error) {
            // 处理失败逻辑
            console.log(error)
          })
          .finally(function() {
            // 成功或失败都需要做的处理
          })
      }

      //实例化Vue
      new Vue({
        el: '#app',
        data: {
          city: '武汉',
          forecastList:[]
        },
        methods: {
          queryWeather() {
            axios
              .get(`http://wthrcdn.etouch.cn/weather_mini?
city=${this.city}`)
              // .then(function(res) {
                .then(res=>{
                // 成功可以在这里拿到数据
                console.log(res.data.data.forecast)
                console.log(this)// 如果用function这里是window
                this.forecastList = res.data.data.forecast
              })
```

```
                .catch(function(error) {
                  // 处理失败逻辑
                  console.log(error)
                })
                .finally(function() {
                  // 成功或失败都需要做的处理
                  console.log('总是被执行')
                })
            }
          }
        })
      </script>
    </body>
  </html>

```

Vue.js-DevTools

Vue调试工具，一个chrome的插件, 有一个常用功能可以查看Vue实例的数据

1. 安装
    1. 推荐chrome商店安装 https://chrome.google.com/webstore/category/extensions?hl=zh-CN
    2. .crx文件拖拽安装 需要打开开发者模
2. 设置 **允许文件网址**，浏览器打开本地文件时也能够生效
3. 使用
    1. 图标变亮且有Vue标签栏
    2.
4. 异常情况
    1. 浏览器访问的页面必须使用开发者版本的vue.js
    2. 要设置允许文件网址
    3. 尽量安装在chrome商店安装
    4. 重启

v-bind 使用补充

如果标签的属性不是写死的，就应该用v-bind来绑定属性

1. v-bind与class的对象方式
    1. :class="{类名:是否添加类名}"
        1. 如果true，就添加类名
        2. 如果为false，就不添加类名
    2. :class和原有的class不冲突，:class是追加类名

```
    <div class="box" :class="{red:isRed}" @click="isRed=!isRed"></div>
```

2. v-bind:style
    1. v-bind:style="{样式的属性名:属性值, 属性名2：属性值。。。。。}"
    2. 属性名如果是-分隔的话，换成驼峰命名或者是加字符串

```
<div :style="{backgroundColor: changeRed?'red':'','margin-top':'100px'}" @click="changeRed=!changeRed"></div>
```

```
<!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 green;
        margin:10px;
      }
      .red{
        background-color: red;
      }
    </style>
  </head>
  <body>
    <div id="app">
```

```
      <div class="red" :class="{red:isRed,green:true}" :class="
{green:true}" @click="isRed=!isRed"></div>
      <div :style="{backgroundColor: changeRed?'red':'','margin-
top':'100px'}" @click="changeRed=!changeRed"></div>
    </div>
    <script src="./lib/vue.js"></script>
    <script>
      let style={
        // 'background-color': 'red'
        backgroundColor: 'red'
      }

      const app = new Vue({
        el: "#app",
        data: {
          isRed:true,
          changeRed:true
        }
      });
      </script>
  </body>
</html>
```

Demo-天知道

实现步骤

1. 输入城市，回车或者点搜索，请求数据，展示天气信息列表
    1. 获取城市名 v-model:city
    2. 回车或者点搜索 @keyup.enter/@click: queryWeather
    3. 请求 axios.get(url)
    4. 接口：http://wthrcdn.etouch.cn/weather_mini?city=深圳
    5. .then(res=>获取数据)
    6. vfor遍历数组，生成天气信息的列表:forecastList
    7. item.type.includes 结合v-if展示图标
2. loading状态的问题
    1. .input_sub这个按钮是否添加 loading这个样式

    2. 是否添加loading样式 :class="{loading:是否添加样式isLoading}"
        1. isLoading为true,就添加loading样式
        2. isLoading为false，不添加loading样式
    3. 发请求前，isLoading=true
    4. 请求完成后.finally isLoading:false
3. 点击北上广深,输入的城市改变且查询点击的城市天气
    1. 点击城市名 @click:clickSearch(城市名)
    2. 输入城市改变 city=城市名
    3. 查询点击城市天气 queryWeather

注意点

1. 加载效果是由isLoading来控制是否添加loading类名
2. v-bind:class的对象方式语法，class="{类名:是否添加类名}"
3. axios().finally统一关闭loading
4. 热门城市的展示用数组渲染，避免写重复的html

Vue动画-单个元素过渡动画

传送门

元素在显示与隐藏时，Vue会在恰当的时机添加/删除6个类名，在类名里让我们自定义动画。

使用方法：

1. v-if或者v-show控制元素的显示与隐藏
2. 标签包裹显示与隐藏的元素
3. transtion name属性的值和类名首个单词一致
4. 在进入/离开的过渡中，会有 6 个 class 切换
    1. v-enter-active 整个动画出现过程中都有
    2. v-enter 动画出现前
    3. v-enter-to 完全出现

```
<!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>
    img {
      width: 200px;
      height: 200px;
      /* transition: all 2s; */
    }

    /* img:hover {
      width: 400px;
    }
    */

    .list-enter-active {
      transition: all 2s;
    }

    .list-enter-to {
      width: 400px;
    }
  </style>
</head>

<body>
  <div id="app">
    <button @click="isShow = !isShow">切换显示</button>
    <br>
    <br>
    <transition name="list">
      <!-- <img v-show="isShow" src="./img/baby.png" alt=""> -->
      <img v-if="isShow" src="./img/baby.png" alt="">
    </transition>
  </div>
  <script src="./lib/vue.js"></script>
  <script>
    const app = new Vue({
      el: "#app",
```

```
      data: {
        isShow: true
      }
    });
  </script>
</body>

</html>
```

Vue动画-结合animate.css

传送门

animate.css是比较流行的动画库, 包括一些流行的css动画效果

使用: 挑选合适的css动画样式，copy进代码，替换掉原来的动画

工作中的动画并不会很复杂。

```
<!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>
    img {
      width: 200px;
      height: 200px;
    }

    @keyframes slideInLeft {
      from {
        -webkit-transform: translate3d(-100%, 0, 0);
        transform: translate3d(-100%, 0, 0);
        visibility: visible;
      }
```

```
21
22        to {
23          -webkit-transform: translate3d(0, 0, 0);
24          transform: translate3d(0, 0, 0);
25        }
26      }
27
28      @keyframes zoomOut {
29        from {
30          opacity: 1;
31        }
32
33        50% {
34          opacity: 0;
35          -webkit-transform: scale3d(0.3, 0.3, 0.3);
36          transform: scale3d(0.3, 0.3, 0.3);
37        }
38
39        to {
40          opacity: 0;
41        }
42      }
43
44      @keyframes rotate {
45        0% {
46          transform: rotate(0deg);
47        }
48
49        100% {
50          transform: rotate(360deg);
51        }
52      }
53
54      .list-enter-active {
55        /* animation: rotate 6.25s infinite linear; */
56        /* animation: zoomOut 2s; */
57        animation: slideInLeft 2s;
58      }
59    </style>
60  </head>
61
62  <body>
```

```
  <div id="app">
    <button @click="isShow = !isShow">切换显示</button>
    <br>
    <br>
    <transition name="list">
      <img v-if="isShow" src="./img/baby.png" alt="">
    </transition>
  </div>
  <script src="./lib/vue.js"></script>
  <script>
    const app = new Vue({
      el: "#app",
      data: {
        isShow: true
      }
    });
  </script>
</body>

</html>
```

Vue动画-列表过渡

传送门

列表动画，列表元素的增加和删除时，Vue会在恰当的时机给元素增加和移除一些类名，在类名里面我们可以自定义样式。

使用：

1. v-for的标签必须包裹在
2. v-for必须给Key
3. transtion-group的tag会自动生成一个标签
4. 添加和移除的类名和单元素动画一致
5. 改变数组本身并不会触发动画

```
<!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>
    .list-enter-active,
    .list-leave-active {
      transition: all 2s;
    }

    .list-enter,
    .list-leave-to {
      transform: translateX(30px)
    }
  </style>
</head>

<body>
  <div id="app">
    <button @click="add">添加</button>
    <button @click="remove">删除</button>
    <!-- <ul> -->
    <transition-group name="list" tag="ul">
      <li v-for="(num, index) in arr" :key="num">{{num}}</li>
    </transition-group>
    <!-- </ul> -->
  </div>
  <script src="./lib/vue.js"></script>
  <script>
    const app = new Vue({
      el: "#app",
      data: {
        arr: [1, 2, 3, 4, 5, 6, 7, 8, 9]
      },
      methods: {
        add() {
          this.arr.push(parseInt(Math.random() * 1000))
        },
        remove() {
          this.arr.splice(0, 1)
        }
```

```
      },
    });
  </script>
</body>

</html>
```

Demo-天知道-动画整合

1. 天气信息列表添加列表过渡效果
    1. transition-group包裹vfor所在的标签
        1. vfor所在的标签一定给key值
        2. name:list
        3. tag:ul
    2. 样式 用官网例子里面的样式，改改
    3. 在每一个次天气信息查询之前先清空天气信息数组，让动画重新生效
    4. 用tranisition-delay让过渡延迟，那么天气信息列表元素就会有依次出现的效果
        1. 最好在行内用v-bind:style动态设置每个一个li的transition-delay

Vue动画钩子

传送门

Vue动画钩子就是元素出现和隐藏的动画过程中，Vue在会恰当的时机以回调函数的形式通知我们。在回调函数里可以自定义一些逻辑。

1. 元素出现动画 前，中，后，还有取消
2. 元素消失动画的前，中后还有取消

```
<transition
  v-on:before-enter="beforeEnter"
  v-on:enter="enter"
  v-on:after-enter="afterEnter"
  v-on:enter-cancelled="enterCancelled"

  v-on:before-leave="beforeLeave"
  v-on:leave="leave"
  v-on:after-leave="afterLeave"
  v-on:leave-cancelled="leaveCancelled"
>
```

使用方法

1. 动画钩函数是注册在transition,transition-grup上
2. 钩子函数的声明是放在methods，默认接受一个el参数，就是动画元素的DOM

天知道-结合Vue动画钩子

实现步骤

1. 在transition-group上注册after-enter
2. after-enter事件处理方法里面重置transition-delay

vfor key 待总结

传送门

1. **重用和重新排序现有元素**, 及transition-group里面vfor, key值要唯一
2. 数组元素下标不变的话，不添加key值和key=index效果一样。
3. vfor的key只能为string和number

```
<!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>
</head>

<body>
  <div id="app">
    <button @click="reverseArr">反转</button>
    <ul>
      <li v-for='(brand,index) in brandList' :key="brand.id">
        {{brand.name}}
        <input type="text">
      </li>
    </ul>
  </div>
  <script src="./lib/vue.js"></script>
  <script>
    const app = new Vue({
      el: "#app",
      data: {
        brandList: [{
          name: '华为',
          id: 1234
        }, {
          name: '小米',
          id: 2345
        }]
      },
      methods: {
        reverseArr() {
          this.brandList.reverse()
        }
      },
    });
  </script>
</body>

</html>
```

补充前端兼容性

1. 移动端只用关注android chrome和ios safari，兼容性还可以
2. shim提供ES6语法 https://github.com/aFarkas/html5shiv
3. PC端一般是官网或者管理端，兼容不好主要考虑的是IE，现在IE分额也越来越小的。提示升级浏览器也是可以 的
4. 新的API才是未来，掌握新的API对自己成长更有利
5. https://caniuse.com/ 查看一些WebAPI的兼容性

总结