Vue DAY03

`:key`的作用

`v-for`指令一般都需要配一个`v-bind:key`(`:key`)来为当前每一个列表项设置一个唯一标识符（简单理解为一个名字），目的是为了提高列表更新时的DOM渲染性能。

如果当前列表中已经包含很多列表项了，后续由于列表数据的变化需要重新渲染列表时，将会通过:key绑定的值来检测当前需要渲染的列表项是否原来已经渲染过（比较列表项的key在原始列表中是否已经存在），如果已经存在则不再重新创建DOM，仅需更新即可。

一般将什么值设置为key?

要求每个列表项必须唯一，数据类型要么number、要么用string。实在没有，可以用下标，但是用下标有坑。

```
<div v-for="item,i  in  list"  :key="item.id">列表项</div>
<div v-for="item,i  in  list"  :key="i">列表项</div>
```

案例：编写一个购物车

1. 下载Cart.vue组件，配置路由，访问：/cart可以看到该购物车静态页面。

如果希望在末尾计算购物车所有商品总价格时，发现计算算法较为复杂，无法在页面中直接用`{{}}`来表示，所以可以如下实现：

```
methods: {
    /** 获取购物车总价格，将每一件商品单价*数量，再累加即可 */
    getTotal() {
        let total = 0
        this.cartInfo.forEach(item=>{
            total += item.count * item.price
        })
        return total
    }
},
```

```
<div style="width:300px; font-size:1.5em;">
    总计: ¥ {{ getTotal() }}
</div>
```

如上解决方案即可显示复杂运算的结果，并且当运算所需要的变量有变化时，页面也会及时更新。

Vue针对这一类需求（显示复杂运算的结果），也提供了一个更好的解决方案：**计算属性**。

Vue中的计算属性

Vue提供了一种特殊的属性：计算属性。它本质上就是一个函数，返回复杂运算之后的结果。在template中可以使用访问属性的语法来访问它。

定义计算属性：

```
// computed选项用于定义计算属性，计算属性的本质实际上是函数
computed: {
    // 定义一个计算属性(函数)，函数名就是计算属性名
    totalPrice() {
        let total = 0
        this.cartInfo.forEach(item=>{
            total += item.count * item.price
        })
        return total
    }
},
```

```
<div style="width:300px; font-size:1.5em;">
    总计: ¥ {{totalPrice}}
</div>
```

计算属性与方法本质上都是使用函数来定义，都需要进行运算之后才可以得到结果。不一样的是方法每次调用每次都要算，而计算属性第一次计算完毕后将会把结果进行缓存，后续需要使用时，直接拿来用。如果计算属性计算时涉及到的变量有变化，再重新运算。

表单元素的双向数据绑定指令 `v-model`

假如有如下输入框：

```
<input type="text" placeholder="" v-model="username">
<button>提交</button>
```

```
data(){
    return {
        username: ''
    }
}
```

如上写法即可完成输入框的value值与data中声明的username变量的**双向数据绑定。**

1. 如果用户在输入框中输入内容，则data中username的值会立即更新。
2. 如果通过程序修改了data中的username值，则将会立即更新界面。

案例：新建Form.vue，访问：/form时，看到该组件。该组件包含一个表单，收集用户数据。

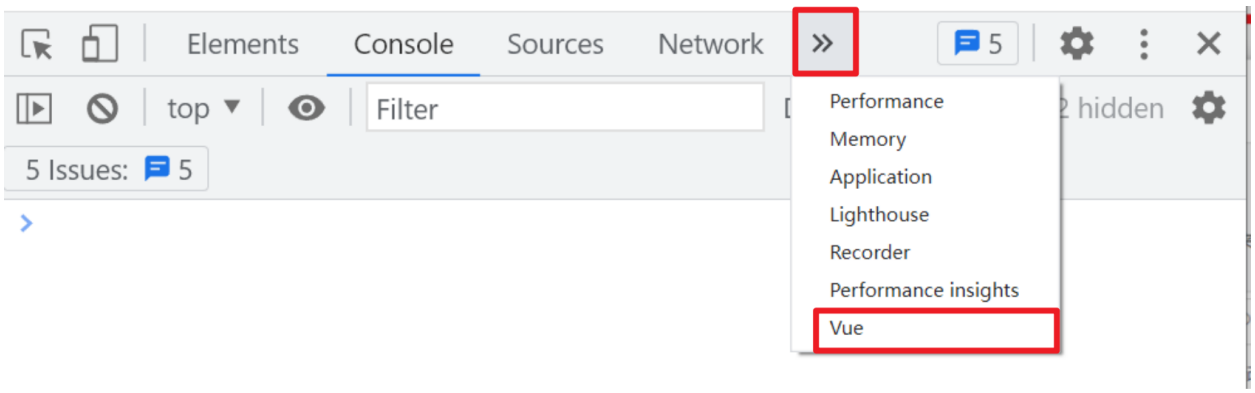
安装基于Chrome浏览器的vue扩展插件：

```
Vue.jsDevtools-v5.3.3.crx
```

1. 打开浏览器的**扩展程序**界面：

2. 打开开发者模式，将插件拖拽进入chrome浏览器即可：

3. 找到控制台vue选项：



案例：针对手机号码输入框，实现表单验证

需求：只要在phone输入框中输入内容，那么就立即执行验证：

先判断输入的内容是否是11位字符串，如果是，则验证手机号码格式是否符合要求，如果不符合要求，则在后面提示红色的消息。

Vue的监听器

Vue为了更好的监控所管理的变量的变化，提供了数据监听器：watch。通过监听器就可以监听数据（data中声明的变量）的动态更新。只要data中所管理的变量有变化，就会触发相应的监听器，执行监听方法：

```
data(){
    return {
        name: ''
    }
}
watch: {
    // 监听器形式上也是一个方法，方法名与监听的属性名保持一致
    // 这样如果变量有变化，就会立即执行该方法，并且vue将会自动传入两个额参数
    // (newValue 新值, oldValue 旧值)
    name(newValue, oldValue){

    }
}
```

其他表单组件的双向数据绑定方式

```
<template>
  <div>
    <h1>Form.vue,测试表单与v-model</h1>
    <hr>
    用户账号: <input v-model="form.name" type="text">
    <hr>
    用户密码: <input v-model="form.pwd" type="password">
    <hr>
    确认密码: <input v-model="form.pwd2" type="password">
    <hr>
    用户手机: <input v-model="form.phone" type="text">
    <hr>
    证件类型:
    <input type="radio" value="sfz" v-model="form.card"> 身份证
    <input type="radio" value="jgz" v-model="form.card"> 军官证
    <input type="radio" value="sbk" v-model="form.card"> 社保卡
    <input type="radio" value="jz" v-model="form.card"> 驾照

    <hr>
    选择行业:
    <input type="checkbox" value="jy" v-model="form.hy"> 教育
    <input type="checkbox" value="yl" v-model="form.hy"> 医疗
    <input type="checkbox" value="jr" v-model="form.hy"> 金融
    <input type="checkbox" value="zmt" v-model="form.hy"> 自媒体

    <hr>
    选择籍贯:
    <select v-model="form.jg">
      <option value="hb">河北省</option>
```

```
      <option value="hn">河南省</option>
      <option value="sd">山东省</option>
      <option value="sx">山西省</option>
    </select>

    <hr>
    <button @click="submit">提交注册信息</button>

  </div>
</template>

<script>
  export default {
    data() {
      return {
        form: {  // 收集整个表单的数据
          name: '',
          pwd: '',
          pwd2: '',
          phone: '',
          card: 'sfz',   // 单选按钮的默认值
          hy: ['jy', 'zmt'],  // 多选框的默认值
          jg: 'hb'  // 下拉列表的默认选中
        }
      }
    },
    methods: {
      /** 点击按钮后执行该方法，提交表单收集的数据 */
      submit() {
        console.log(this.form)
        // 做表单验证，基于正则表达式  验证账号：[3-10]的单词字符
        let exp = /^\w{3,10}$/;
        if(exp.test(this.form.name)){ // 验证通过
          alert('账号验证通过')
        }else {
          alert('账号格式不正确，要求[3-10]单词字符')
        }
      }
    },

    /** watch监听器 */
    watch: {
      // 监听form.phone，一旦该属性有变化，则执行该监听方法
      // 传入的参数：newValue新值，oldValue旧值
      'form.phone': (newValue, oldValue) => {
        console.log(`form.phone从${oldValue}变成了${newValue}`)
        // 验证phone的值是否符合要求：
        if(newValue.length==11){
          let exp = /^1[3-9]\d{9}$/;
          if(exp.test(newValue)){
            console.log('手机号格式正确')
          }else {
            console.error('手机号格式错误')
```

```
          }
        }
      }
    },
  }
</script>

<style lang="scss" scoped>

</style>
```

Vue的自定义指令

Vue官方提供了很多的指令来对页面中国的元素进行特殊处理。例如:v-html v-if等。这些指令本质都是一段程序，用于处理当前dom元素。当vue在加载页面template时，一旦发现有元素身上包含v-开头的属性，就会当做是指令来看待，将会寻找指令处理函数来操作当前DOM元素。

vue提供了自定义指令的语法，可以让开发者自己设计指令（自定义指令的名称、指令的功能），以更加方便的完成特殊的DOM操作需求。

```
<span v-red>删除</span>
```

案例：访问：/direct 看到 views/Direct.vue。在该组件中测试自定义指令的声明与使用。

```
directives: {
    // 自定义一个v-red指令
    red: {
        // 当vue检测到元素上包含有v-red指令时，
        // 并且当绑定了v-red指令的dom元素被插入dom树后
        // 将会自动执行inserted方法，传入dom对象
        inserted: (el)=>{
            el.style.color = "red"
        }
    }
}
```

Axios

axios是一个网络通信库，封装了原生了ajax。提供了一些简单的API辅助程序员方便的发送http、https请求。底层基于Promise进行封装。

在脚手架中安装axios

找到项目根目录，打开cmd，执行命令：

```
npm  install  axios
```

安装成功后，将会在package.json中生成axios的依赖项。

基于axios发送请求

```
import axios from 'axios'
let instance = axios.create()
instance({
    url: '请求资源路径',
    method: 'GET',
    params: {name:'zs', pwd:'1234'}
}).then(res=>{
    res就是发送请求后，axios封装的响应数据
})
```

测试接口：

```
https://web.codeboy.com/bmdapi/movie-infos?page=1&pagesize=20
```