Vue DAY03

v-for 指令

v-for 指令用于循环输出元素。例如：

```
<p v-for="item in 10">一段话...</p>
```

上面的写法，类似于：

```
for(let i=0; i<10; i++){
   let item = i+1
   输出: <p>一段话...{{item}}</p>
}
```

案例：新建一个组件：VFor.vue，当访问： /vfor 时看到这个组件。

还可以遍历数组：

```
data() {
    return {
        hobby: ['玩单杠', '摊煎饼', '背麻袋', '健身', '游泳', '瑜伽']
    }
},
```

```
<span class="tag" v-for="(item, i) in hobby" :key="item">
    {{ item }}
</span>
```

上述写法，类似于：

```
for(let i=0; i<hobby.length; i++){
   let item = hobby[i]
   输出: `<span class="tag">${item}</span>`
}
```

还可以遍历对象数组：

```
foods: [
    {id:1001, name:'煎饼果子', price:4.5, num:2},
    {id:1002, name:'大榴莲', price:104.5, num:5},
    {id:1003, name:'鲱鱼罐头', price:34.5, num:1},
    {id:1004, name:'毛鸡蛋', price:2.5, num:2},
]
```

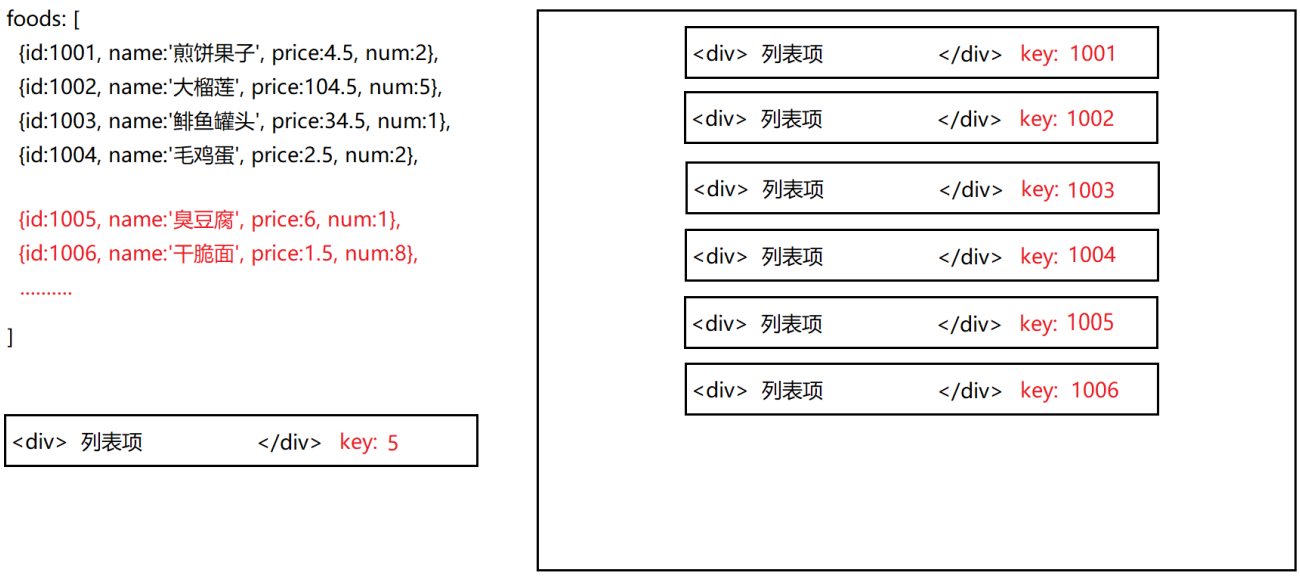
```
<div class="cart-item" v-for="(item, i) in foods" :key="item.id">
    商品ID: {{item.id}}    
    商品名称: {{item.name}}    
    商品价格: {{item.price}}    
    商品数量: {{item.num}}    
    小计: {{item.price * item.num}}
</div>
```

上述写法，类似于：

```
for(let i=0; i<foods.length; i++){
    let item = foods[i]
    // 输出div，在div中使用item变量，来引用当前元素的属性
    `<div class="cart-item">${item.id}.......</div>`
}
```

:key

`v-for` 指令一般都需要配一个 `v-bind:key`（简写 `:key`）指令属性来为当前输出的列表项设置一个唯一的标识符。目的是为了提高列表更新时的页面渲染性能。如果当前列表中已经渲染了很多列表项了，当列表有后续变化重新渲染列表时，将会通过 `:key` 来检测当前需要渲染的列表项是否已经渲染过，如果已经渲染过，则不会创建新的 `DOM` 元素。直接更新旧元素即可。



`:key=""` 一般设置什么值?

```
<div v-for="(item,i) in list"  :key="i"></div>
<div v-for="(item,i) in list"  :key="item.id"></div> id等唯一属性（首选）
```

如果列表是静态列表，没有更新的业务需求，`:key`写不写都可以。

`Vue`中的计算属性

`vue`提供了一种特殊的属性：计算属性。这种属性本质上是一个函数，返回数据运算之后的结果。在`template`中可以当做**普通属性**来进行访问。如果计算属性的值由于参数的变化而发生变化，将立即影响`DOM`。

```
export default {
    data(){
        return {
            price: 5,
            num: 2
        }
    },
    // 声明计算属性
    computed: {
        total(){
            return this.price * this.num
        }
    }
}
```

在`template`中可以将计算属性直接当做普通属性来使用：

```
<span>{{total}}</span>
```

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="请输入账号" v-model="form.name">
```

```
data(){
    return {
        form: {
            name: '',   // name将与input组件完成双向数据绑定
        }
    }
}
```

如上写法，即可实现`form.name`与**文本框输入的值**之间的双向数据绑定：

1. 如果用户在输入框中输入内容，则`data`中的变量`form.name`将会更新。

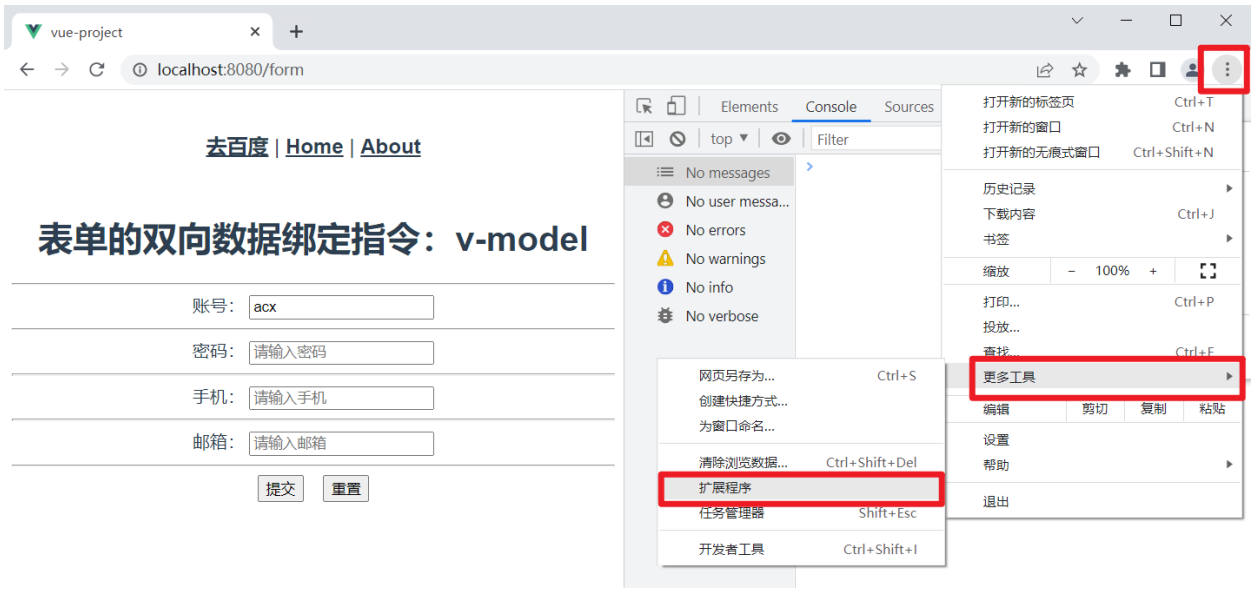
2. 如果通过代码将 `form.name` 的值进行了修改（ `this.form.name='zs'` ），那么输入框中的内容也会改变。

案例：新建 `Form.vue` ，访问 `/form` 时，看到该组件。

安装基于 `chrome` 谷歌浏览器的 `vue` 扩展插件：

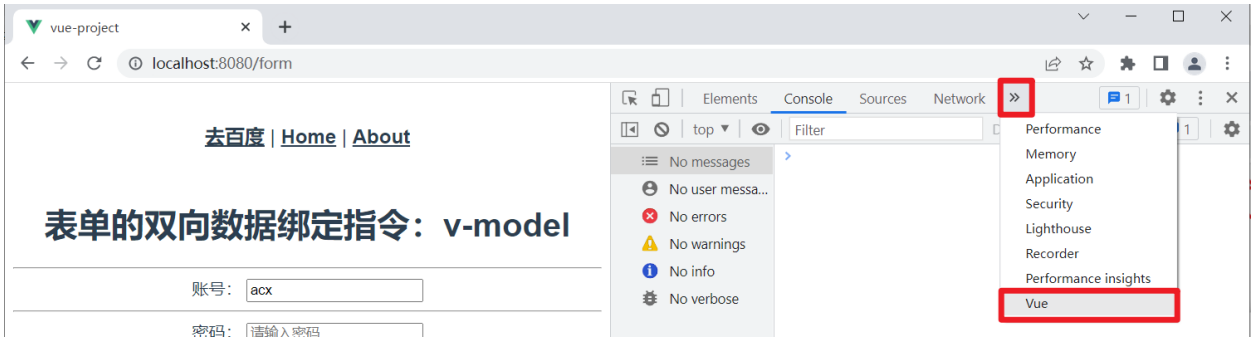
```
vue.jsDevtools-vxx.x.x.crx
```

1. 打开浏览器的扩展程序页面：



开启开发者模式后，将安装包拖拽进入浏览器即可。

在 `f12` 控制台，找到 `vue` 插件，并打开插件界面：



点击提交按钮时，实现表单验证，点击重置按钮时，重置表单

1. 当点击提交按钮时，获取四个输入框中的值。每个值都需要进行验证（非空）。
2. 将表单验证的结果通过 `alert` 进行提示。
3. 点击重置按钮时，修改 `form` 对象。

实现实时监听输入框中的值，做输入框内容的验证

`vue` 提供了数据**监听器**可以监听 `data` 中声明的变量的变化。 一旦 `data` 中变量有变化，将会触发监听，执行相应的监听方法。

```
data(){
    return {
        name: '',
        pwd: ''
    }
},
watch : {
    // 该方法用于监听data中声明的name变量的变化，一旦有变化则立即执行该监听方法
    // 并且自动传入两个参数：  newval(改动的新值)  oldval(以前的旧值)
    name (newval, oldval){

    }
}
```

案例：监听form.pwd, form.phone, form.email的文本内容是否符合要求，不符合要求也给设置一个红色边框。

```
form.pwd      必须是6位数字
form.phone    必须符合手机号规则
form.email    必须符合邮箱规则
```

其他表单组件的双向数据绑定方式

了解到了input输入框的双向数据绑定写法后，还有很多其他常用的表单组件也需要完成双向数据绑定：单选按钮、多选框、下拉列表框等。

单选按钮：

```
性别：
<input type="radio" value="m" v-model="form.sex"> 男
<input type="radio" value="f" v-model="form.sex"> 女
```

多选框：

```
感兴趣的行业：
<input type="checkbox" value="jy" v-model="form.hy"> 教育
<input type="checkbox" value="jr" v-model="form.hy"> 金融
<input type="checkbox" value="hlw" v-model="form.hy"> 互联网
<input type="checkbox" value="yl" v-model="form.hy"> 医疗
```

下拉列表框：

```
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>
```

通过上述表单组件中声明的`v-model`将用户填写的数据更新到`this.form`对象中。

```
form: {
    name: "",
    pwd: "",
    phone: "",
    email: "",
    sex: 'm',
    hy: ['jy'], // 通过数组来收集复选框选中项的value
    jg: 'hb'  // 下拉列表框，默认选中的是hb
},
```

`Vue`自定义指令

`vue`官方提供了很多的指令来对相应的页面元素进行特殊处理。例如：v-text可以将变量的值输出；v-html可以将变量的值当做html代码解析输出；v-for将会遍历输出元素等。

当`vue`在加载页面模板时，如果检测到元素有`v-`为前缀的属性的话，就会把这些属性当做指令来进行处理。每一个指令背后都是一段程序，用于对当前元素进行特殊处理。

`vue`还提供了自定义指令的语法，可以让开发者自己设计vue指令（自定义vue指令的名称、功能），更加方便的完成项目中的需求。

```
<p v-red>xxxxxxxxxx</p>
<p v-blue>xxxxxxxxxxxxxx</p>
<p v-color="#f68">xxxxxxxxxx</p>
```

代码实现：

```
<template>
  <div>
    <h3 v-blue>测试自定义指令</h3>
    <p v-red>专家说，再过俩仨月疫情就会进入第二波，反应比第一波弱很多。</p>
    <p v-color="`#2fd356`">专家说，再过俩仨月疫情就会进入第二波</p>
  </div>
</template>

<script>
export default {
    directives: {  // 在此处定义  自定义指令
      color : {
        inserted(el, binding){
          // el: 绑定该指令的DOM对象
          // binding: 指令参数对象
          console.log(binding)
          el.style.color = binding.value
        }
      },

      // red用于定义指令的名字    <p v-red></p>
      red : {
        // inserted方法由vue自动调用，当绑定该指令的元素被插入到DOM树后执行
        // vue将传入一个参数el：表示绑定了该指令的DOM对象
        inserted: function(el){
          el.style.color = '#f00'  //为当前DOM元素修改文本颜色
        }
      },
      blue : {
        inserted: function(el){
          el.style.color = '#00f'
        }
      },

    }
};
</script>

<style lang="scss" scoped></style>
```

axios

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

https://www.axios-http.cn/

在脚手架中安装`axios`

在项目的根目录下，通过`cmd`，执行命令，安装`axios`：

```
npm install axios
npm i axios
```

基于axios发送`GET`请求：

```
import axios from 'axios'
let instance = axios.create()   // 创建axios对象，通过instance发送请求

instance({
    url: 'https://web.codeboy.com/bmdapi/movie-infos',    // 请求地址
    method: 'GET',   // 请求方式
    params: {page:1, pagesize:20}   // 请求参数
}).then(res=>{
    res就是响应数据
}).catch(err=>{
    err就是请求失败信息
})
```

案例：安装`axios`，新建：`views/Http.vue`。当访问：`/http`时看到它。

经典错误集合

```
[Vue warn]: Property or method "clickMe" is not defined on the instance but referenced during render. Make sure that this property is reactive, either in the data option, or for class-based components, by initializing the property. See: https://v2.vuejs.org/v2/guide/reactivity.html#Declaring-Reactive-Properties.

(found in <Root>)                                                                                    vue.js:5108
[Vue warn]: Invalid handler for event "click": got undefined

(found in <Root>)                                                                                    vue.js:5108
```

属性或方法`clickMe`并没有在vue对象中定义。要么在data段中声明，要么定义成组件的成员。

`Module not found: Error: Can't resolve './views/Basic.vue' in 'D:\code2209\01_Vue\day02\demo\vue-project\src\router`

模块找不到错误。好好检查代码中编写的这个路径是否可以找到对应的组件。当前这个错误就是找不到这个组件（`'./views/Basic.vue'`）而出的错误。