vue第1天

今天目标：

- 熟练使用各种**指令**
  - 元素内容区域
  - 元素属性
  - 元素绑定事件
  - v-model双向数据绑定
  - 遍历指令
- 实现品牌列表展示
  - 列表展示
  - 添加品牌
  - v-for 和 :key 应用

vue-介绍

三大前端框架

Vue

Vue.js (view)是一套构建用户界面的前端框架技术

内部集成了许多基础技术，例如html、css、javascript、ajax、node等，当然还有vue本身高级技术体现，例如组件、过滤器、指令、路由、webpack等等

2012年出现，是中国人 尤雨溪 开发的，2016年3月 加入阿里巴巴公司(该事件助推了vue的发展)

jquery：库 侵入性弱 (工具 库)，项目 对其进行 安装卸载 非常方便

vue：框架 侵入性强 (框架)，项目 从内到外 都是vue，不可以随便拆卸

Vue只关注**视图**（页面）层的开发，文档非常丰富、易于上手，流行度高，拥抱经典的web技术、早期灵感来源于angular

vue.js兼具angular.js和react.js的优点，并剔除了它们的缺点

支持所有兼容ECMAScript 5的浏览器，IE9以上

vue是前端的主流框架之一，和**Angular**、**React** 一起，并成为前端三大主流框架！

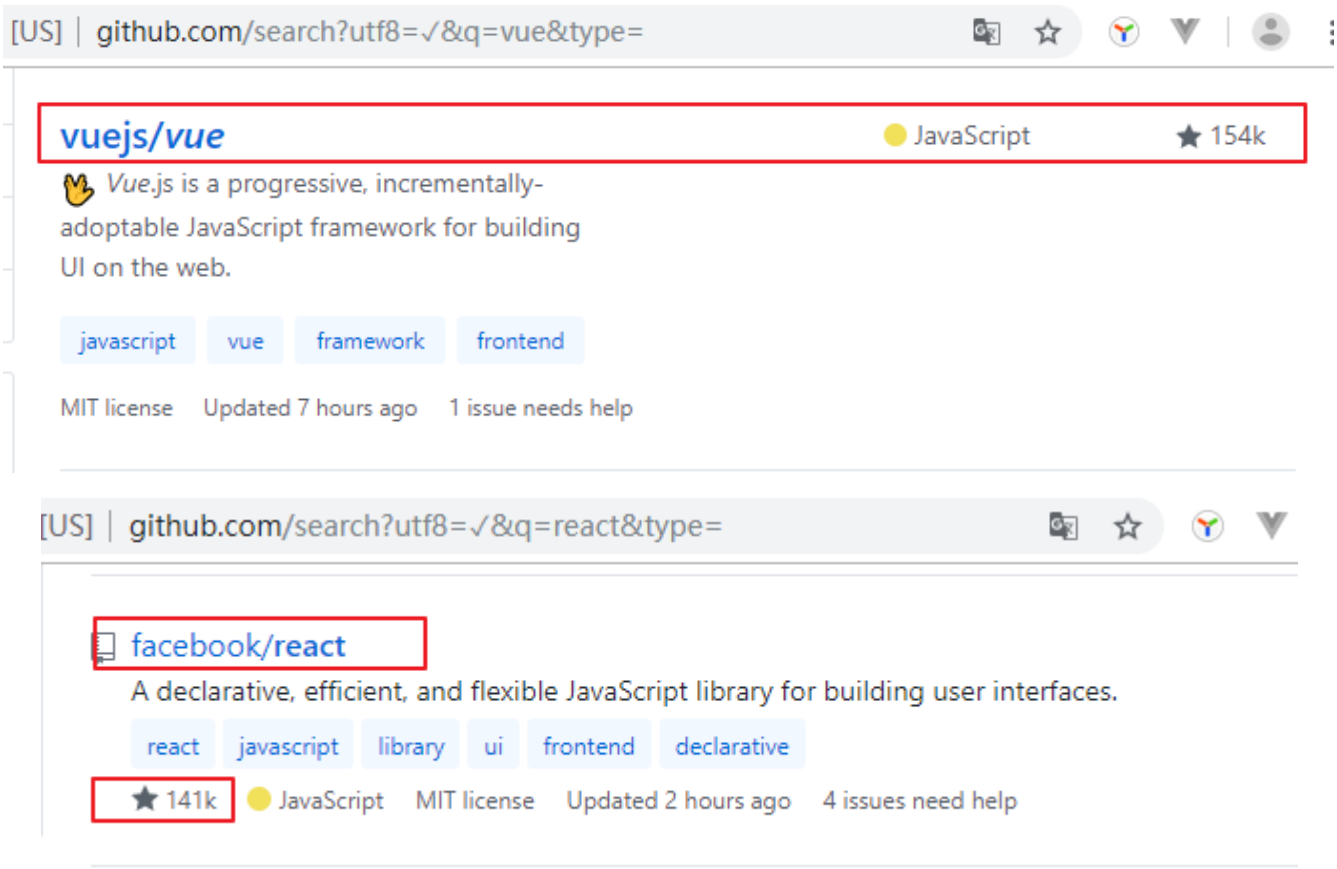
学习Vue前的技术准备：

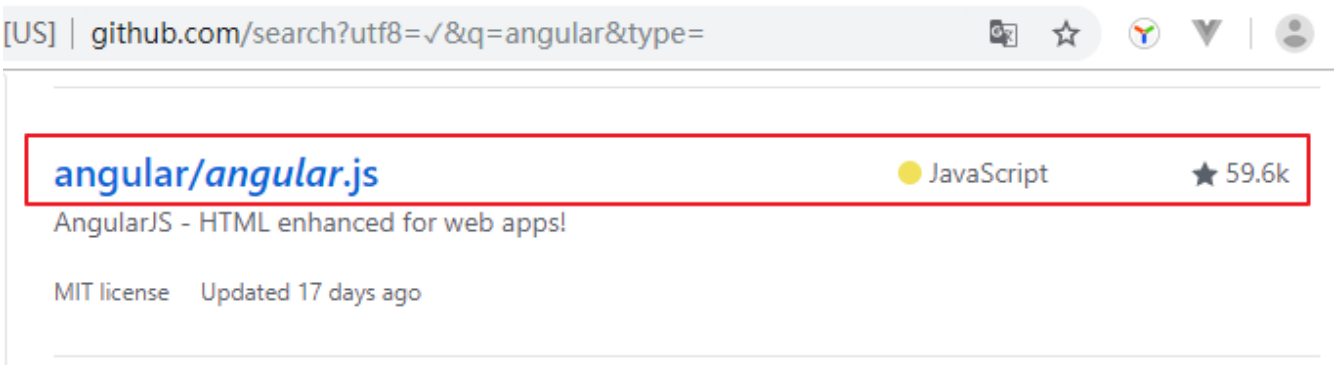
- 掌握 HTML + CSS + JS 基本网页制作能力
- 了解Node基础概念、包、模块化，会用 npm 维护项目中的包即可
- ES6/ES7 基础语法要会用，在Vue课程中我们也会补充更多的ES6

使用vue的公司

小米、阿里、百度、饿了么、**掘金**、苏宁易购、美团、天猫、Laravel、htmlBurger、哔哩哔哩直播、哔哩哔哩商城

github上vue框架的“点赞”次数，可见vue的流行程度高于react和angular





angular

2009年出现，**google**公司出品

Angular.js 出现最早的前端框架，曾经很火，但是现在已经 被边缘化了；也不好学；

Angular 1.x 学起来好麻烦；

Angular 2.x ~ 5.x 学习起来相对简单；

TypeScript(javascript超集)

新旧版本没有平滑升级

使用特点：强侵入，凡事都必须遵循angular的规则

React

2010年出现，**facebook**公司出品

React.js 是目前最流行的一个框架；是用的人最多的一个框架；但是，学习起来也比较难，因为在 React中，所有的功能，都要用 JavaScript 来实现；

JSX(JavaScript XML/Xhtml) & ES2015(es6) 一切都是javascript、包括html、css

针对初学者太笨拙，难以掌握，有些地方的代码看起来完全没有逻辑性，学习过程痛苦

其他框架

Ember

Knockout

Polymer

Riot

。。。

为什么要学习流行框架

- 最先进最前沿的开发模式(前后端分离)
- 提升项目开发效率(节约成本)
- 应用最前沿技术es6/es7
- 前端不是做辅助工作的，而是主力开发者、增加开发话语权、由被动变为主动、增强核心价值/核心竞争力

获得vue

网址：

https://cn.vuejs.org 官方地址(服务器在外国，访问速度慢)

https://vue.docschina.org/ 官方地址镜像(服务器在中国，访问速度快)

最新稳定版本：2.6.11

1) 直接下载

- 开发版本：https://cdn.jsdelivr.net/npm/vue/dist/vue.js
- 生产版本：https://cdn.jsdelivr.net/npm/vue

2) CDN(Content Delivery Network内容分布式部署)

在应用中通过script标签直接引入一个完整路径名的vue文件包

该方式要求具备上网环境

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

3) 使用 `npm` 下载（默认安装最新稳定版）

```
npm install vue
```

`注意`：

Vue.js 不支持 IE8 及其以下版本

vue-简单使用

`目标`：

通过vue输出城市和天气信息

`步骤`：

1. 创建div容器

```
<div id="app"> {{ city }}----{{weather}} </div>
```

2. 引入vue.js文件

```
<script src="./vue.js"></script>
```

3. 实例化Vue对象

```
var vm = new Vue({
  el: '#app',
  data:{
    city:'北京',
    weather:'snow'
    }
})
```

注意：

1. Vue需要有目标操作容器，可以是div、p、span等等标签，推荐div，所有被Vue处理的内容都放到该容器中
2. {{}} 是Vue内容，浏览器上看不到，最终要被解析掉
3. {{}}双花括号是Vue语法，后期详解
4. el、data 是Vue内部固定的标志名称
5. data内部可以声明一个或多个数据供使用
6. el:'#app' 是通过id="app"联系容器，也可以通过其他"选择器"联系
7. 模板中所有内容需要体现到div容器里边

示例代码：

```
 <!-- 2. 创建一个供vue操控的 标签容器(推荐是div) -->
  <div id="app">
    <!-- 输出vue实例的data数据 -->
    {{ city }}--------{{ weather }}
  </div>

  <!-- 1. 引入vue文件包，此时系统增加了一个全局变量，名称为Vue(类似引入jquery.js，系统增加$符号全局变量)
-->
  <script src="./vue.js"></script>

  <script>
    // 3. 实例化Vue对象
    var vm = new Vue({
      // el:'选择器' ,// el固定名称，理解为element，使得 Vue实例 与 标签容器 联系
      el:'#app',
      // data固定名称，给 Vue实例 声明数据，用于使用
      data:{
        city:'北京',
        weather:'sunshine'
      }
    })
```

```
</script>
```

Vue-MVVM设计模式

目标：

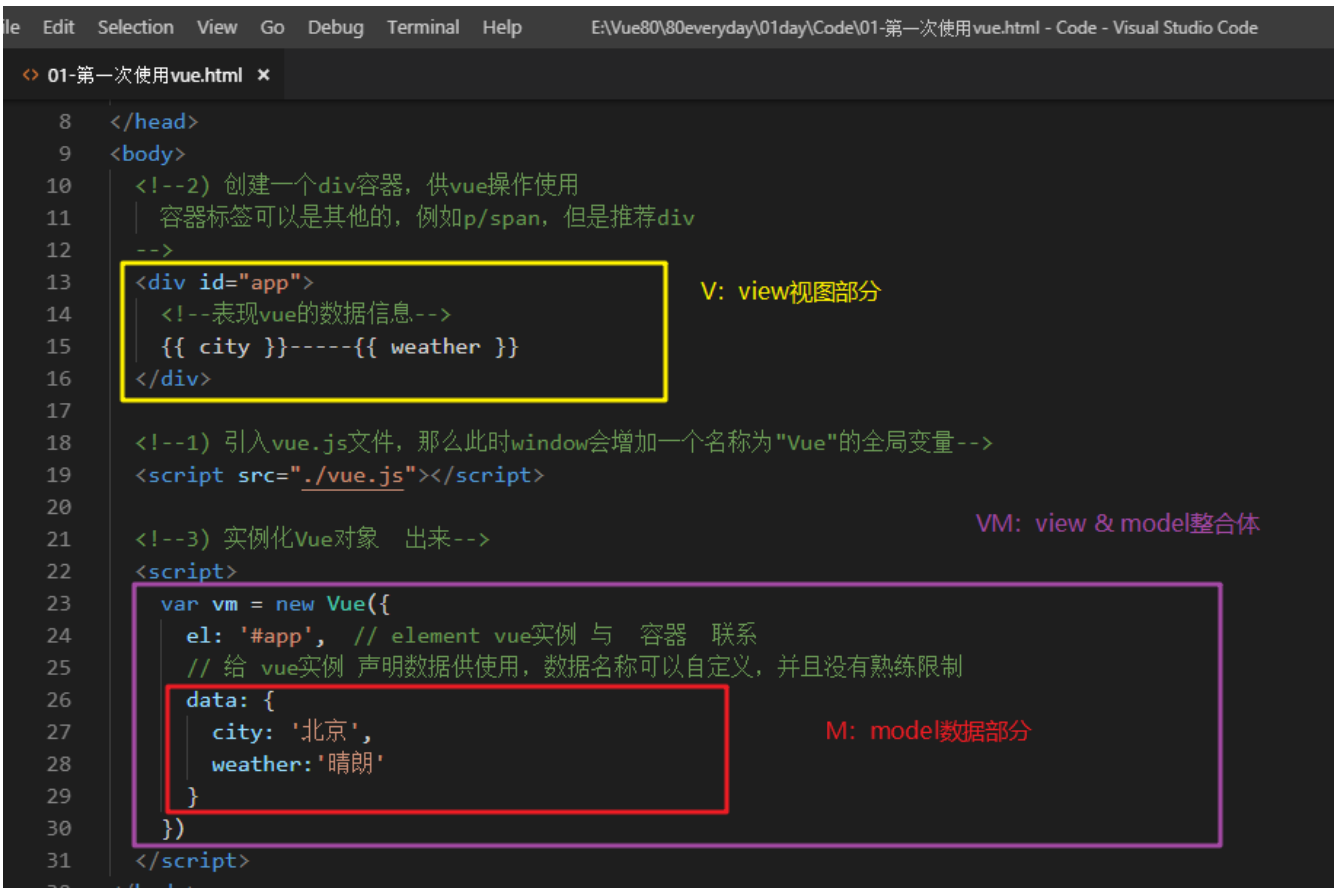
了解MVVM各部分含义 和 对应代码

mvvm设计模式可以解读为如下：

m: model 数据部分 data

v：view 视图部分 div容器

vm： view & model 视图和数据 的 结合体



vue指令

插值表达式

目标：

熟练使用插值表达式

vue中如果需要在html标签的“**内容区域**”中表现数据，就可以使用**{{}}双花括号**，这个技术称为**插值表达式**

语法：

```
<标签> {{ 表达式 }} </标签>
```

> 表达式：变量、常量、算术运算符、比较运算符、逻辑运算符、三元运算符等等

使用示例：

```
<div id="app">
  <p>{{ msg }}</p>                                    <!--变量-->
  <p>{{ score }}</p>                                  <!--变量-->
  <p>{{ 500 }}</p>                                    <!--常量-->
  <p>{{ score+10 }}</p>                               <!--算术运算-->
  <p>{{ score>10 }}</p>                               <!--比较运算-->
  <p>{{ score>80 && age>18 }}</p>            <!--逻辑运算-->
  <p>{{ age>18 ? '成年' :'少年' }}</p>       <!--三元运算-->
</div>
```

如果{{}}使用中有冲突，想更换为其他标记，可以这样：

```
delimiters:['${', '}$']  // 标记符号变为${  }$了
```

使用要点：

1. 在插值表达式中 只能设置**简单**的javascript表达式，不能设置**复杂**表达式(例如for循环)
2. 在data值大小不改变的前提下，可以进行一般的 **算术**运算、**比较**运算、**逻辑**运算、**三元**操作符 等运算使用，也可以通过**常量**进行数据体现
3. 插值表达式只能用在html标签的**内容区域**；不能用在其他地方
4. {{}}花括号与变量之间为了美观可以有适当的**空格**，数量不限制，例如{{ msg}}、{{msg }}、{{ msg }}等都可以，为了美观，表达式左右**各一个**空格即可

示例代码：

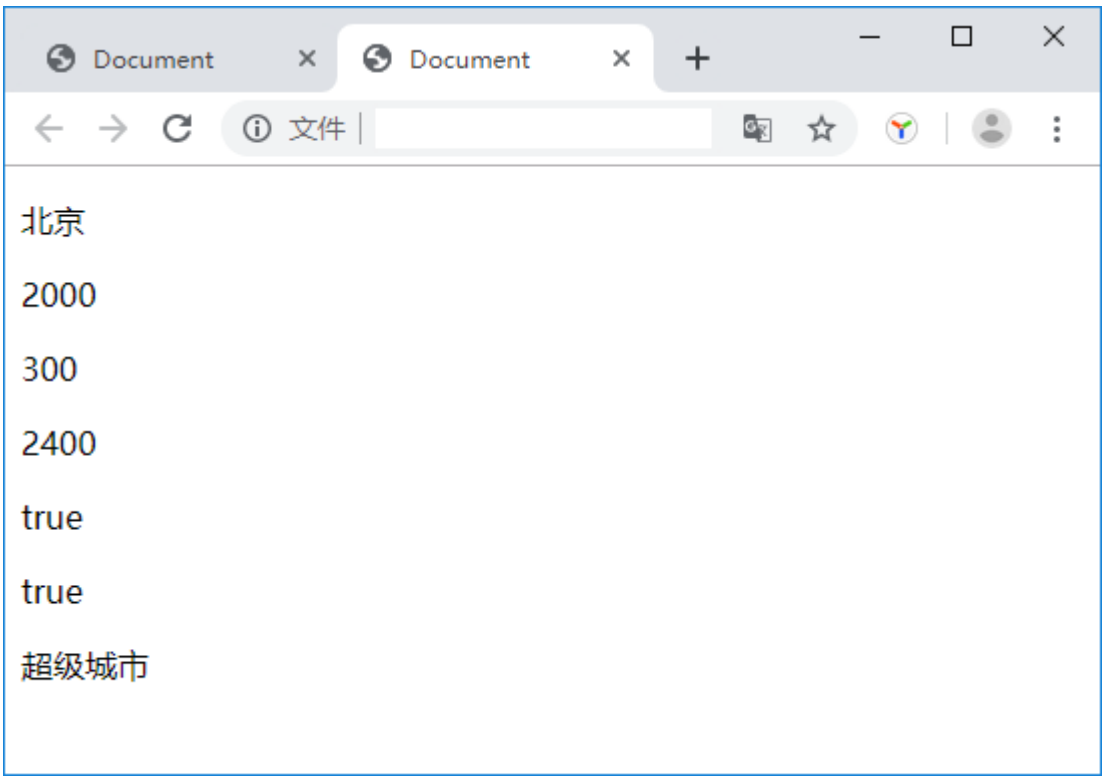
```
<div id="app">
   <p class="{{city}}">{{ city }}</p>
   <p>{{ people }}</p>  <!--变量-->
   <p>{{ 300 }}</p>  <!--常量-->
   <p>{{ people+400 }}</p>  <!--算术-->
   <p>{{ people>1000 }}</p>  <!--比较-->
   <p>{{ people>1000 && city==='北京' }}</p>  <!--逻辑-->
   <p>{{ people>1000 ? '超级城市' : '大城市' }}</p>  <!--三元运算-->
```

```
</div>
<script src="./vue.js"></script>

<script>
  var vm = new Vue({
    // delimiters:['$','#'],
    el:'#app',
    data:{
      city: '北京',
      people: 2000
    }
  })

</script>
```

效果：



练习：

在data中声明 score1 和 score2两个变量，分别赋予一定的数字

两个变量做算术和，判断总数大小并显示相关信息，大于600显示'清华'，否则显示'北大'

```
<p>{{ city }}</p>

<p>{{ js表达式 }}</p>

在javascript中，有实在信息返回的语句就是表达式
var a = 10
var b = 20
var c = a+b
var d = a/b
var d = a>b
var d = a>5 && b>9
// 以上=等号 右边的都称为表达式

// 有返回实在信息，就是表达式
console.log(10)
console.log(a>5 && b>9)

// 没有返回实在信息(undefined)，就是语句
console.log(  var d = a/b  )
console.log(  alert(123)  )
```

v-text

目标：

熟练使用 v-text 指令

v-text与{{}}的作用是一样的，都是操控 标签的**内容区域**信息

语法：

```
<标签 v-text="表达式"> </标签>
```

注意：

1. v-text 是通过标签的**属性**形式呈现
2. 如果 标签 内容区域 有默认信息，则会被v-text覆盖掉
3. v-text 可以进行 变量、常量、算术符号、比较符号、逻辑符号、三元运算符号等运算

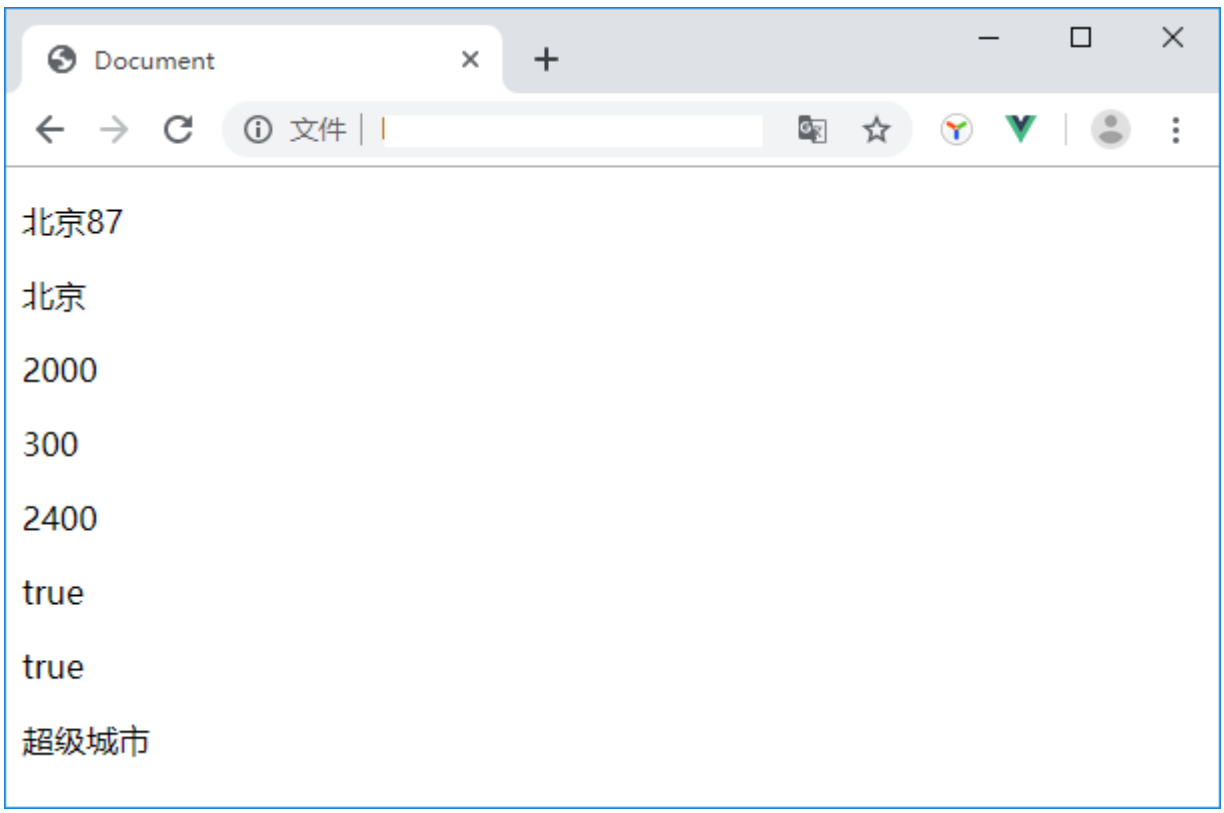
示例代码：

```
<div id="app">
  <p>{{ city }}87</p>
  <p v-text="city">87</p>
  <p v-text="people">{{city}}</p>  <!--变量-->
  <p v-text="300"></p>  <!--常量-->
  <p v-text="people+400"></p>  <!--算术-->
  <p v-text="people>1000"></p>  <!--比较-->
  <p v-text="people>1000 && city==='北京'"></p>  <!--逻辑-->
  <p v-text="people>1000 ? '超级城市' : '大城市'"></p>  <!--三元运算-->
</div>
<script src="./vue.js"></script>

<script>
  var vm = new Vue({
    // delimiters:['$','#'],
    el:'#app',
    data:{
      city: '北京',
      people: 2000
    }
  })

</script>
```

效果：



v-text-闪烁

目标：

了解什么是闪烁 和 出现条件

v-text与{{}} 都可以操控 标签 内容区域，但是他们有一点 区别

网速非常慢的时候，{{}} 有**闪烁**问题，而 v-text没有(属性 本身就是不会显示出来的)

什么是闪烁：

网速非常慢时，{{ }}花括号 等原生内容 在 Vue编译期间 在浏览器**短时**显示的现象就是闪烁

> 闪烁是负面的内容，想方设法不要让其出现

出现闪烁的条件：

1. http协议方式打开应用程序文件(给vscode编辑器 安装 live server)
2. 调整网速为 慢3G(firebug-->network-->slow 3G)

解决闪烁：

1. 使用v-text
2. vue.js在div容器上边引入

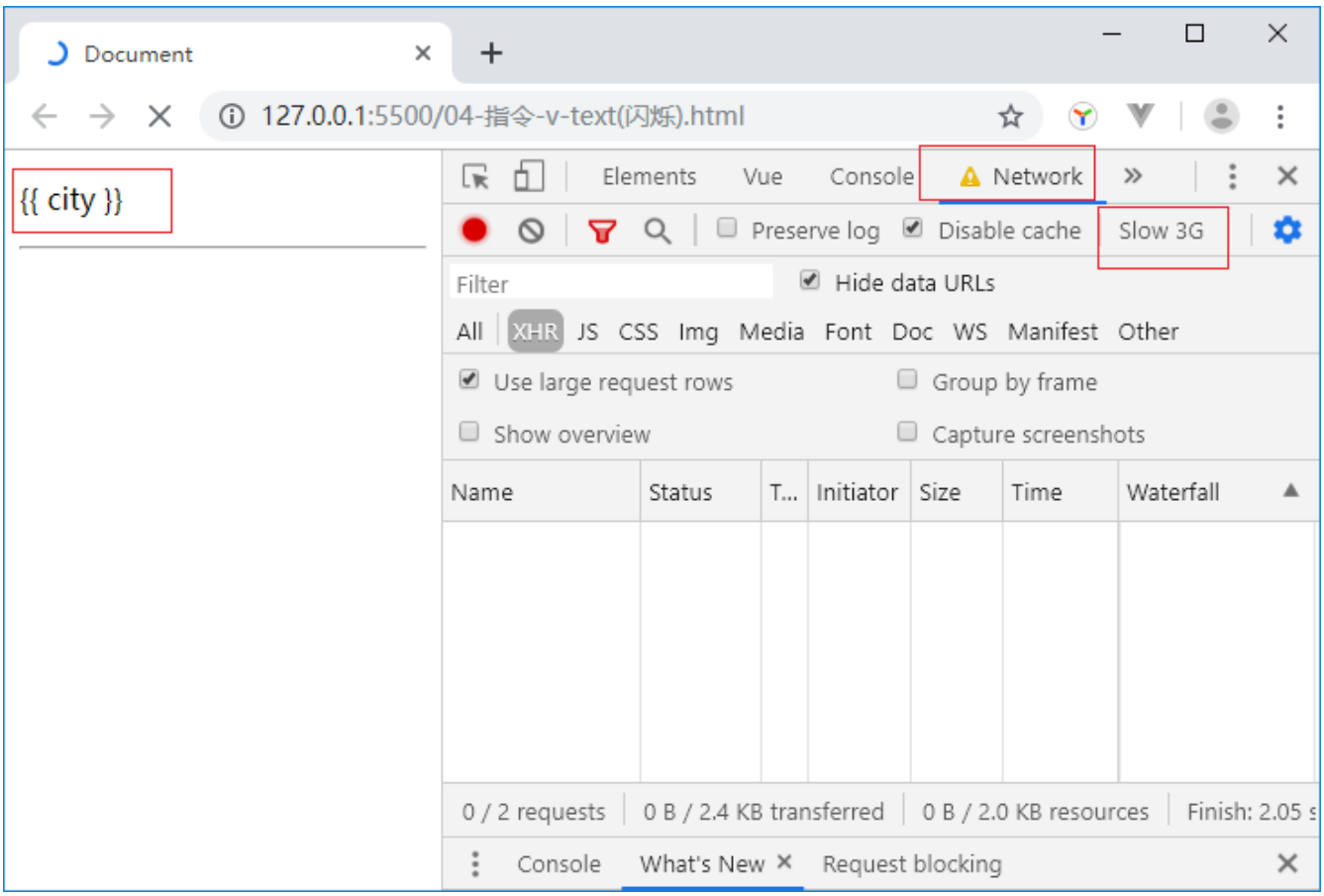
代码：

```
<div id="app">
  <p>{{ city }}</p><!--有闪烁-->
  <hr />
  <p v-text="city"></p><!--没有闪烁-->
</div>

<!--vue.js在div之后引入 是 闪烁出现的必要条件-->
<script src="./vue.js"></script>
<script>
  var vm = new Vue({
    // delimiters:['$','#'],
    el:'#app',
    data:{
      city: '北京',
      people: 2000
    }
  })

</script>
```

效果：



注意：

vue文件包需要在div容器**之后**引入

v-html

目标：

熟练使用v-html指令

v-html 与 v-text、{{ }} 的作用一样，都是操控 标签的**内容区域**

语法：

```
<标签 v-html="表达式"> </标签>
```

`v-html、v-text、{{ }}的异同`：

1. v-html对 **html标签** 和 **普通文本** 内容都可以设置显示

2. v-text、{{ }} 只针对 **字符串** 起作用，如果数据中有html标签，那么标签的箭头符号要做字符实体转换，进而使得浏览器上直接"显示箭头"等标签内容，等同于不解析html标签内容

   > < 变为 < // less than
   >
   > > 变为 > // great than

   字符实体详细说明： https://www.w3school.com.cn/html/html_entities.asp

3. v-html和v-text没有闪烁问题
4. 它们都可以进行 **变量、常量、算术运算、比较运算、逻辑运算、三元运算**等技术应用

应用场景：

如果 服务器返回的数据中，包含有HTML标签(例如富文本编辑器内容)，就可以使用 v-html 来渲染(v-text和 {{}}都不行)

默认内容

```
<p>{{score}}默认内容</p>
<p v-text="score">默认内容</p>
<p v-html="score">默认内容</p>
```

> 以上三者，v-text和v-html标签有默认内容，但是都会被覆盖掉，而 插值表达式 不会覆盖

注意：

1. v-html尽量避免使用(除非完全掌控)，否则会带来危险(XSS攻击 跨站脚本攻击)
2. 标签的默认内容会被v-html覆盖

使用示例：

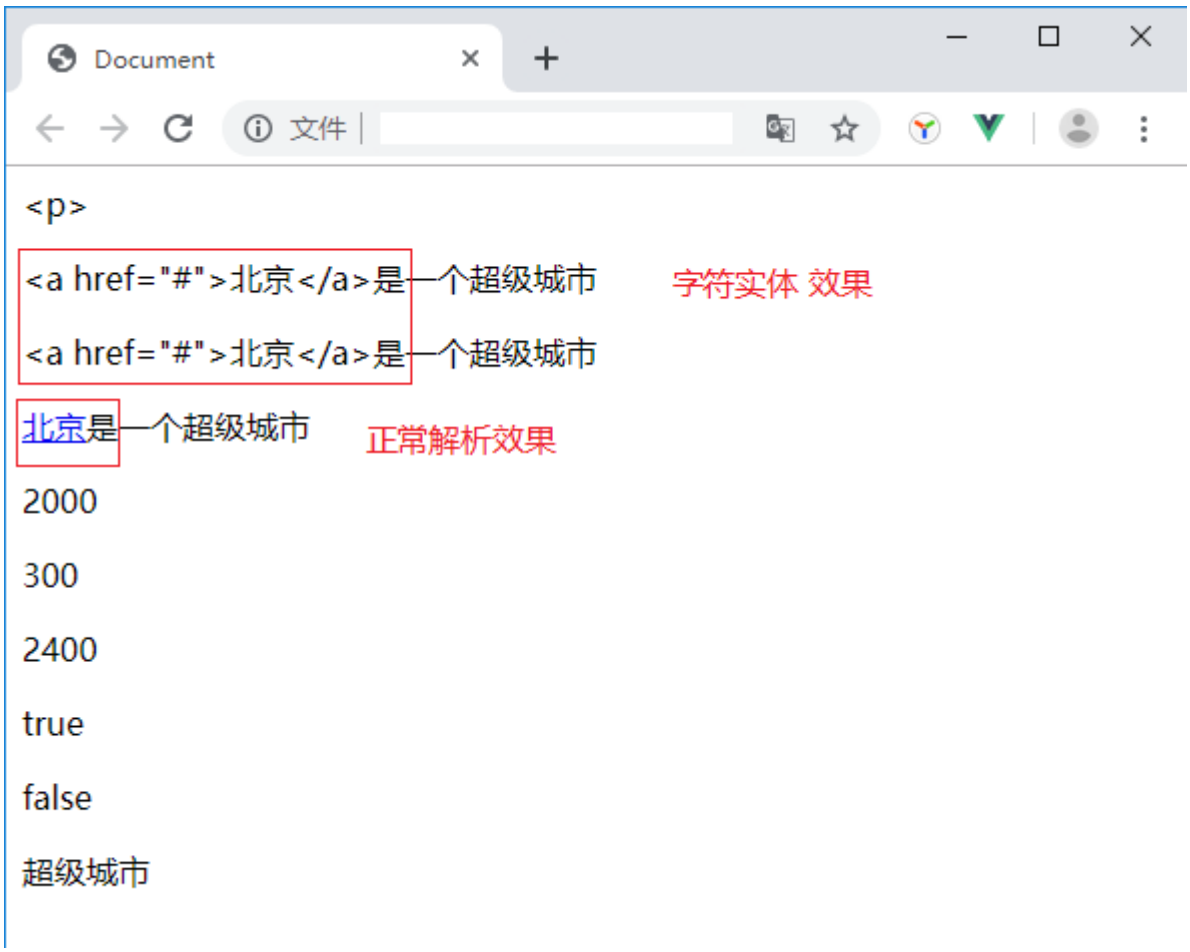
```
<div id="app">
  &lt;p&gt;
  <p>{{ city }}</p>
  <p v-text="city"></p>
  <p v-html="city"></p>
  <p v-html="people"></p>  <!--变量-->
  <p v-html="300"></p>  <!--常量-->
  <p v-html="people+400"></p>  <!--算术-->
  <p v-html="people>1000"></p>  <!--比较-->
  <p v-html="people>1000 && city==='北京'"></p>  <!--逻辑-->
  <p v-html="people>1000 ? '超级城市' : '大城市'"></p>  <!--三元运算-->
</div>
<script src="./vue.js"></script>

<script>
  var vm = new Vue({
```

```
    // delimiters:['$','#'],
    el:'#app',
    data:{
      city: '<a href="#">北京</a>是一个超级城市',
      people: 2000
    }
  })

</script>
```

效果:



v-bind(冒号)

属性绑定-简单使用

目标：

熟练使用 v-bind指令 对标签属性进行绑定

{{ }}、v-text、v-html可以对标签的**内容区域**进行操作，操作标签的**属性**需要通过 v-bind: 指令

语法：

```
<标签 v-bind:属性名称="表达式" ></标签>
<标签 :属性名称="表达式"></标签>  // 简易方式设置，推荐使用
```

注意：

1. 冒号 的简易设置推荐使用
2. 如果有需要，绑定的内容可以进行 **变量**、**常量**、**算术运算**、**比较运算**、**逻辑运算**、**三元运算**等技术应用

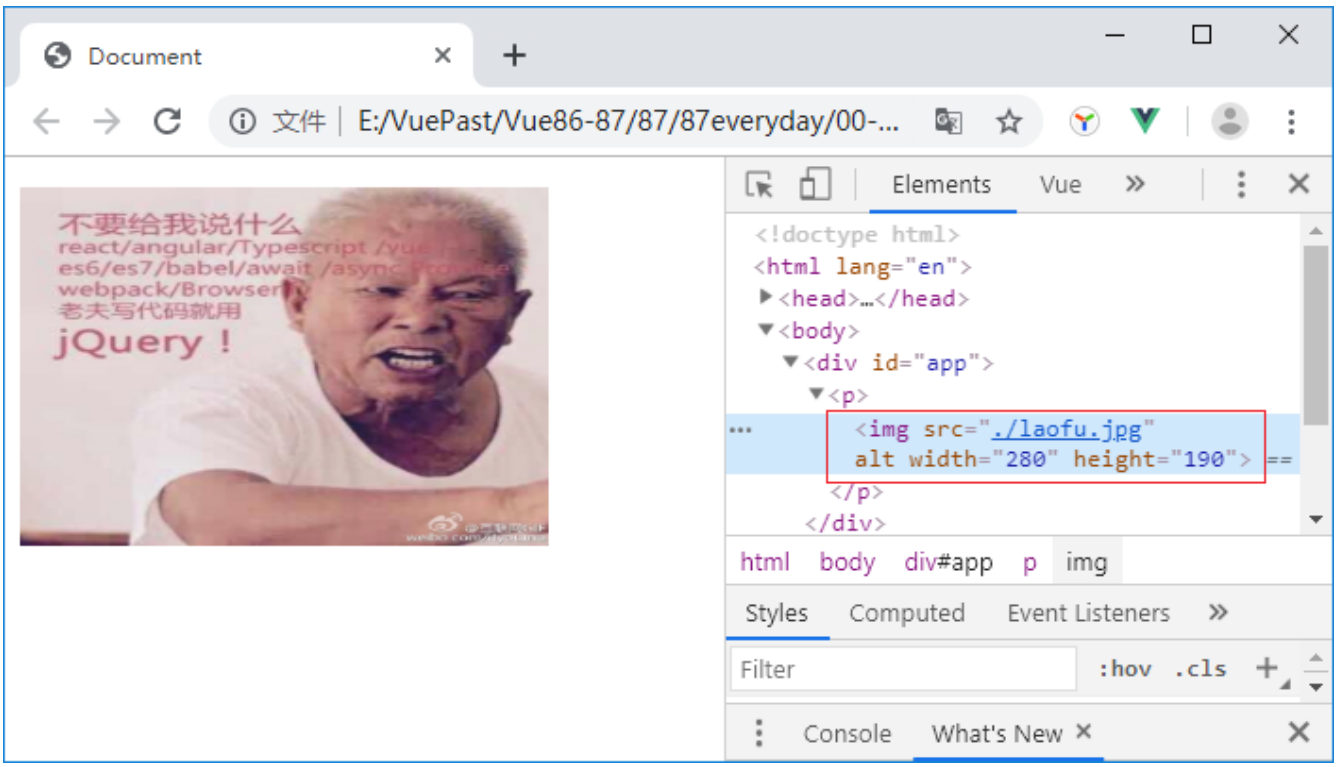
案例：

通过v-bind对img标签的src、width、height属性进行绑定

```
<div id="app">
  <p>
    <img v-bind:src="mysrc" alt="" :width="wh" :height="ht" />
  </p>
</div>
<script src="./vue.js"></script>

<script>
  var vm = new Vue({
    el:'#app',
    data:{
      mysrc:'./laofu.jpg',
      wh:280,
      ht:190
    }
  })
</script>
```

效果：



注意：

在项目根目录提前准备一个名称为 laofu.jpg 的图片

属性绑定-class属性

目标：

熟练使用 v-bind 对标签的class属性进行绑定

class属性较比其他属性不一样的地方是，其既可以接收一个值，也可以通过空格分隔接收多个值

例如：

```
<tag class="apple"></tag>                                <!--接收一个值-->
<tag class="apple pear orange"></tag>        <!--接收多个值-->
```

class属性绑定的语法：

```
1）对象方式
    <tag :class="{xx:true, xx:false}"></tag>
    <!--true: 设置    false:不设置-->

2）数组方式
    <tag :class="['xx','yy','zz']"></tag>
    <!--数组元素值如果不是data成员 就要通过单引号圈选，代表是普通字符串-->
```

提示：

通过数组方式给class绑定多个值，值如果不是 Vue 的 data成员，就需要通过**单引号**圈选，表明其是普通字符串

案例：

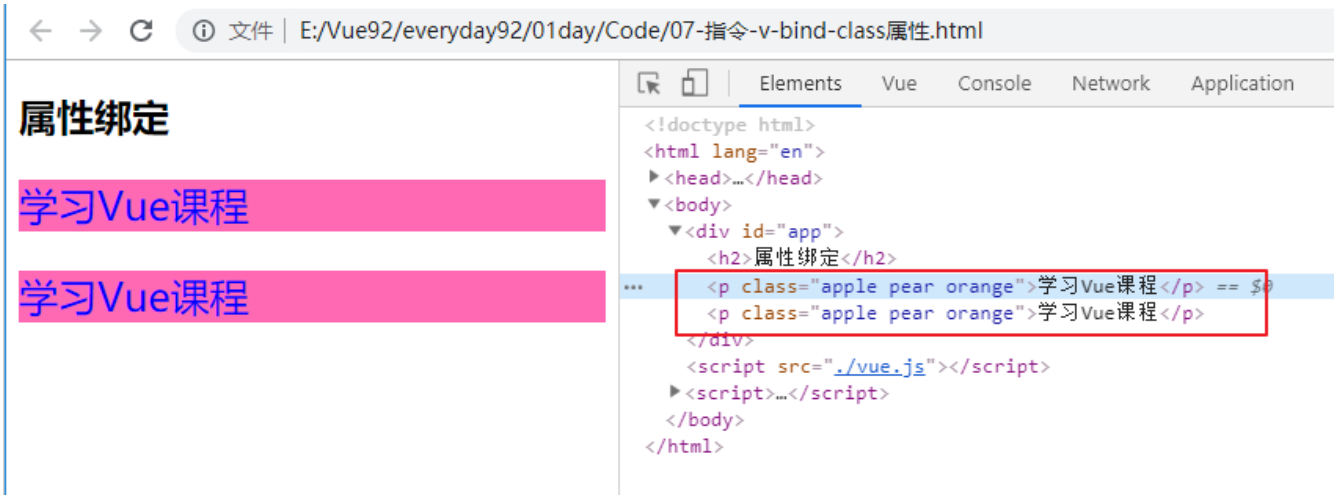
通过数组、对象两种方式给p标签的class属性绑定 apple、pear、orange 多个值

```
  <style>
    .apple{color:blue;}
    .pear{font-size:25px;}
    .orange{background-color:hotpink;}
  </style>
</head>
<body>

  <div id="app">
    <h2>属性绑定</h2>
    <!-- <p class="apple pear orange">学习Vue课程</p> -->
    <!-- 通过vue方式给class绑定如上3个信息 -->
    <!-- 1. 对象方式 -->
    <p :class=" {apple:true, pear:true, orange:true, banana:false} ">学习Vue课程</p>
    <!-- 2. 数组方式 -->
    <p :class=" ['apple','pear','orange'] ">学习Vue课程</p>
  </div>

  <script src="./vue.js"></script>
  <script>
    var vm = new Vue({
      el:'#app',
      data:{
        // apple:'pingguo'
      }
    })
  </script>
```

效果：



注意：

1. 通过class绑定多个值，每个值都可以设置css样式
2. 表达式：就是要按照javascript的语法规则设置相关内容

属性绑定-style属性

目标：

熟练使用v-bind指令操控style属性

style样式属性较比普通属性也比较特殊，其也可以接收多个值

例如：

```
<p style="color:red; font-size:25px; background-color:lightgreen;"></p>
```

你

具体绑定语法：

```
1）对象方式
    <tag :style="{xx:yy,xx:yy.....}"></tag>
    <!--xx:样式名称，yy:样式的值-->
2）数组方式
    <tag :style="[{xx:yy},{xx:yy.....}]"></tag>
    <!--根据需要，数组的每个元素都是一个对象，每个对象包含一对或多对css样式-->
```

提示：

1. 数组方式绑定style属性，每个数组元素可以包含**一个**或**多个**css样式对
2. 复合属性带中横线(例如background-color)的需要通过**单引号**圈选，或变为**驼峰**名称

案例：

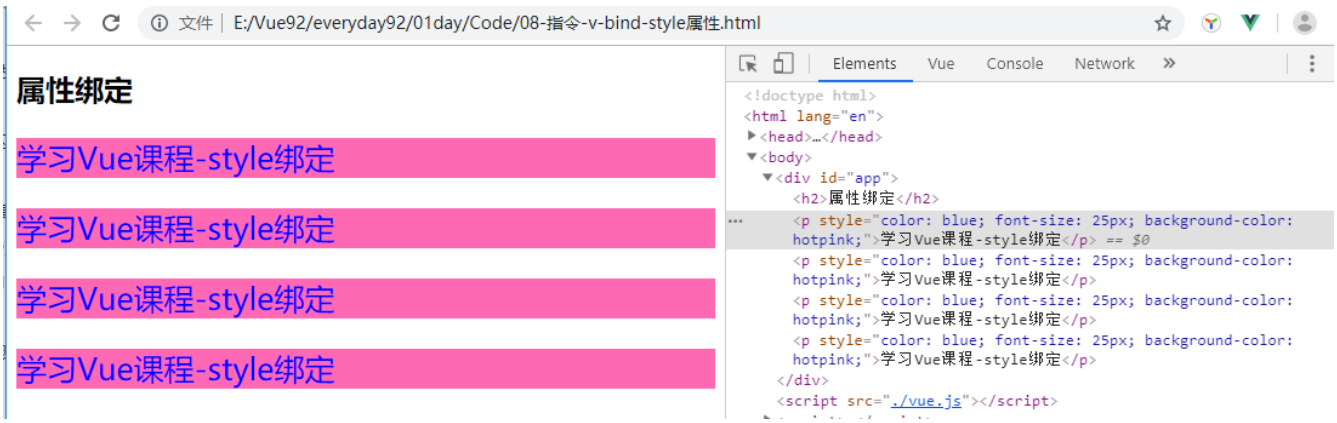
通过数组、对象两种style属性绑定方式，实现如下标签效果

```
<p style="color:blue;font-size:25px;background-color:hotpink;">学习Vue课程-style绑定</p>
```

```
  <div id="app">
    <h2>属性绑定</h2>
    <!-- <p style="color:blue;font-size:25px;background-color:hotpink;">学习Vue课程-style绑定
</p> -->
    <!-- 通过vue实现给style绑定多个css样式信息 -->
    <!-- 1. 对象方式 -->
    <p :style=" {'color':'blue','fontSize':'25px','background-color':'hotpink'} ">学习Vue课程-
style绑定</p>
    <!-- 2. 数组方式 -->
    <p :style=" [{'color':'blue','fontSize':'25px','background-color':'hotpink'}] ">学习Vue课
程-style绑定</p>
    <p :style=" [{'color':'blue'},{'fontSize':'25px','background-color':'hotpink'}] ">学习Vue
课程-style绑定</p>
    <p :style=" [{'color':'blue'},{'fontSize':'25px'},{'background-color':'hotpink'}] ">学习
Vue课程-style绑定</p>
  </div>

  <script src="./vue.js"></script>
  <script>
    var vm = new Vue({
      el:'#app',
      data:{
        // apple:'pingguo'
      }
    })
  </script>
```

效果：



注意：

通过传统方式也可以操作class或style属性，但是Vue的操控会更加**灵活** 和 **精准**，可以针对**某一个值**进行设置

v-on(@)

事件绑定-简单使用

目标：

熟练地给标签绑定事件

web端网页应用程序开发，事件是一个不不可或缺的技术

在vue中给元素进行**事件**绑定，需要通过v-on:指令实现，也使用@符号，@是v-on:的简写，使用更方便

事件类型：

> 鼠标事件：onclick ondblclick onmouseenter onmouseleave onmouseover onmousedown等等
>
> 键盘事件：onkeyup onkeydown onkeypress 等等

语法：

```
<tag v-on:事件类型="事件处理驱动"></tag>
<tag @事件类型="事件处理驱动"></tag>  <!-- @符号 简便用法，推荐使用-->
<div @click="事件处理驱动"></tag>

<script>
var vm new Vue({
  el:xx
  data:xx,
  // 给当前vue实例 声明方法，以供事件调用
  methods:{
    名称: function(){}
  }
})
</script>
```

注意：

1. 事件处理驱动 需要通过methods定义
2. 被绑定的事件类型可以是 click、dblclick、keyup、keydown等等，**不要**设置on标志了

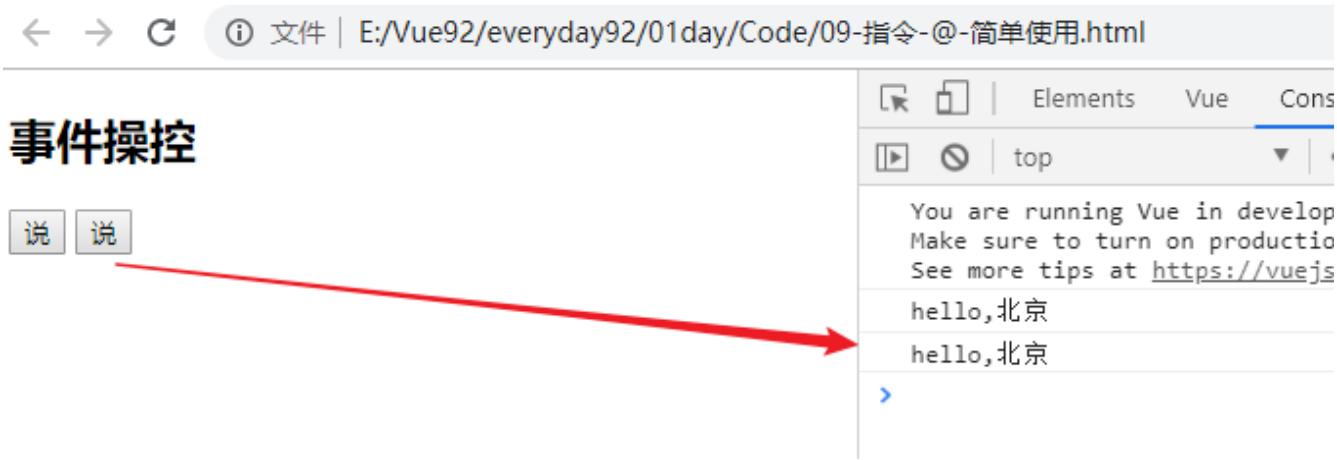
案例：

定义一个按钮，使得单击后 控制台 可以输出 “hello，你好呀”

```
<div id="app">
  <h2>事件操控</h2>
  <!-- <button v-on:click="方法名称">说</button> -->
  <!-- <button @click="方法名称">说</button> @是 v-on: 的简写，推荐使用，记住-->
  <button v-on:click="say()">说</button>
  <button @click="say()">说</button>
</div>

<script src="./vue.js"></script>
<script>
  var vm = new Vue({
    el:'#app',
    data:{
      // apple:'pingguo'
    },
    // 给Vue实例 声明方法，该方法可以给事件使用
    methods:{
      // 名称:function(){}  // es6简易设置为  名称(){}   :function被清除了
      say(){
        console.log('hello,北京')
      }
    }
  })
</script>
```

效果：



事件绑定-传递参数

目标：

清楚事件中被传递参数的意思

vue“单击”事件参数的3种类型：

1. 有声明()，也有传递实参，形参就代表被传递的实参
2. 有声明(),但是没有传递实参，形参就是**undefined**
3. 没有声明()，第一个形参就是**事件对象**

```
<div id="app">
  <h2>事件操控</h2>
  <p><button @click="delA(301)">删除A</button></p>
  <p><button @click="delB()">删除B</button></p>
  <p><button @click="delC">删除C</button></p>
</div>

<script src="./vue.js"></script>
<script>
  var vm = new Vue({
    el:'#app',
    // 给Vue实例 声明方法，该方法可以给事件使用
    methods:{
      delA(id){
        console.log('id为'+id+'的商品被删除了')  // 301
      },
      delB(id){
        console.log('id为'+id+'的商品被删除了')  // undefined
      },
      delC(id){
        console.log('id为'+id+'的商品被删除了')  // [object MouseEvent/鼠标事件对象]
      }
    }
  })
</script>
```

千万注意：

没有()括号情形，由于**事件类型**不一样，参数所代表的 事件对象 也会不同，请灵活使用

事件绑定-访问data成员

目标：

知道事件中如何访问data成员

根据业务需要，事件在执行过程中需要对Vue实例的data数据进行操作，通过this关键字实现，

this代表Vue实例对象，并且针对data或methods成员都可以直接进行调用

案例：

给data声明city成员，值为“铁岭”

设置按钮，使得单击后控制台可以输出“铁岭是一个大城市”

```
<button @click="getInfo()" >获取数据</button>

<script>
  var vm = new Vue({
    el:'#app',
    data:{
      address:'铁岭'
    },
    methods:{
      getInfo:function(){
        // 通过 this关键字 获得 data区域的数据信息
        console.log(this.address+'是一个大城市');
      }
    }
  })
</script>
```

事件绑定-this是谁

目标：

知道Vue实例中this是谁

在Vue实例内部包括不限于methods方法中，**this关键字** 是Vue实例化对象，具体与 **new Vue()** 是一样的

并且其可以对 **data** 和 **methods**成员进行直接访问

> 可以理解为this和vm是同一个对象的两个不同名字，this === vm

```
  <div id="app">
    <h2>事件操控</h2>
    <p><button @click="say()">说</button></p>
  </div>

  <script src="./vue.js"></script>
  <script>
    var vm = new Vue({
      el:'#app',
      data:{
        city:'北京',
        weather:'snow'
      },
      // 给Vue实例 声明方法，该方法可以给事件使用
      methods:{
        say(){
          console.log(this) // 是对象，具体是new Vue()的对象，构造器是Vue
          // 外部的vm也是Vue实例化对象
          // this 和 vm  是同一个Vue实例化对象的两部同步名字而已，它们全等于
```

```
        console.log(this === vm) // true
        // (实例化对象的构造函数 new xxx,xxx就是当前实例化出来对象的构造器)
        console.log(this.city)
        console.log(this.weather)
        console.log(this.say2())
      },
      say2(){return 1234}
    }
  })
</script>
```

注意：

this 与 vm 的指引完全一致，它们可以通过===判断等于(是同一个实体对象的两个不同名字而已)

一般this指向：

1. this就是window对象

```
var age = 20
function getInfo(){
  console.log(this)  // window
  console.log(this.age)
}
getInfo()  // 20
```

2. this代表调用该方法的当前对象

```
const tiger = {
  name:'东北虎',
  act:'猛虎扑食',
  say(){
    console.log('我的名字是'+this.name+', 我的招牌动作是'+this.act)
    // this代表tiger对象
  }
}
tiger.say()
```

3. this代表元素节点对象

```
<button onclick="this.style.color='red'" />确定</button>
```

注意：

this在不同情况下代表不同对象，不用强记，通过console.log输出查看便知

上午总结：

熟练使用各种**指令**

- 元素内容区域
  - {{ }}
  - v-text
  - v-html
- 元素属性
  - <标签 :属性名称=“表达式”>
  - <标签 :class=“对象、数组”>
  - <标签 :style=“对象、数组”>
- 元素绑定事件
  - <标签 @事件类型="方法">
  - methods
  - 访问data： this.xxx
  - this: Vue实例，可以直接访问data或methods等成员，与vm是全等于关系

事件绑定-方法简易设置

目标：

对methods的各个成员进行简易设置

具体设置：

根据es6标准，可以给methods各个方法做简易设置如下：

```
<script>
  var vm = new Vue({
    el:'#app',
    // 给Vue实例 声明方法，该方法可以给事件使用
    methods:{
      // 对象成员方法简易设置   方法名称:function(){}  简易设置为:    方法名称(){}
      delA(id){
        console.log('id为'+id+'的商品被删除了')  // 301
      },
      delB(id){
        console.log('id为'+id+'的商品被删除了')  // undefined
      },
      delC(id){
        console.log('id为'+id+'的商品被删除了')  // [object MouseEvent]
      }
    }
  })
```

```
  </script>
// delA:function(){}   简易设置为  delA(){}
// delB:function(){}   简易设置为  delB(){}
// delC:function(){}   简易设置为  delC(){}
```

> 以后在其他应用中，对象中成员方法都可以把 ":function" 去除，做简易设置，是es6的标准规范

原理：

es6标准里边规定，**对象成员名称** 与 **值的变量名称** 保持一致，就可以做简易设置

```
// 对象属性简易设置如下:
var name = "xiaoqiang"
var height = 176
var run = function(){
  console.log('在跑步')
}

var person = {name:name,height:height,run:run}       // 正常创建对象
var person = {name, height, run}                     // 简便方式创建对象
```

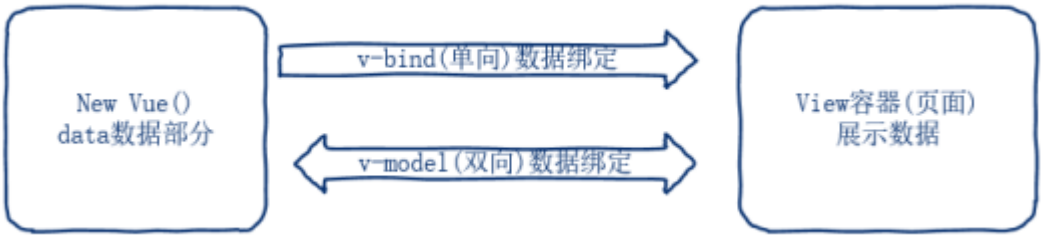
```
// 对象成员方法简易设置如下:
var obj = {
  // walk:function(){
  //   console.log('走直线')
  // }
  //变形
  // walk:function walk(){
  //   console.log('走直线')
  // }
  // 简易设置为如下:
  walk(){
    console.log('走直线')
  }
}
```

> 给一个对象直接声明一个成员方法，也可以简写为如上形式，去除": function"字样

v-model**

在Vue中有一个很重要的指令，名称为v-model，其被称为**双向数据绑定**指令，就是Vue实例对数据进行修改，页面会立即感知，相反页面对数据进行修改，Vue内部也会立即感知

v-model是vue中 唯一实现 双向数据绑定 指令



v-bind单向数据绑定，Vue实例修改数据，页面会感知到，相反页面修改数据Vue实例**不能**感知

v-model双向数据绑定，页面修改数据，Vue实例会感知到，Vue实例修改数据，页面也会感知到

简单使用

目标：

了解v-model的特性

语法：

```
<标签 v-model="data成员"></标签>
```

注意：

1. v-model是vue中**唯一**的双向数据绑定指令
2. v-model只针对**value属性**可以绑定，因此经常用在form表单标签中，例如input(输入框、单选按钮、复选框)/select(下拉列表)/textarea(文本域)，相反div、p标签不能用
3. v-model只能绑定**data成员**，不能设置其他表达式，否则错误

   > 例如
   >
   > v-model="score+100" 错误
   >
   > v-model="120" 错误
   >
   > v-model="score" 正确的

4. v-model绑定的成员需提前在data中声明好

示例代码：

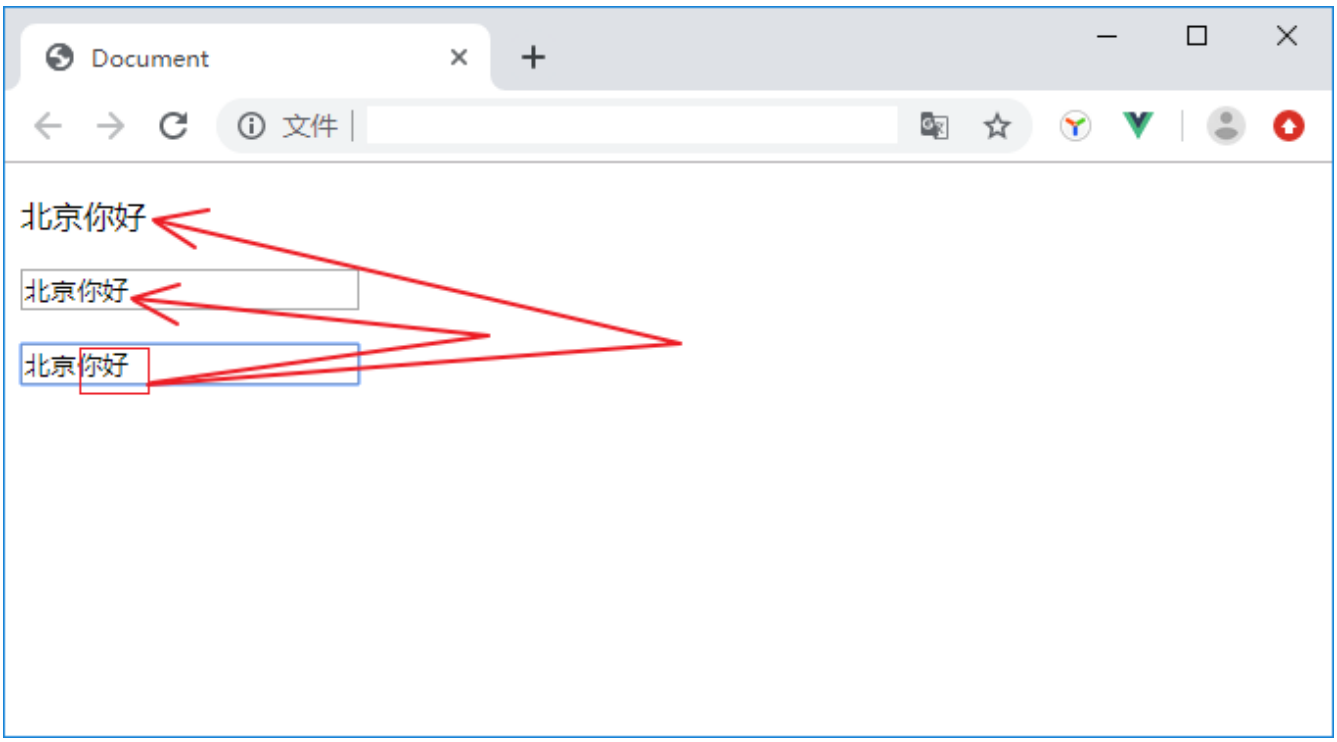
```
<div id="app">
  <p>{{ city }}</p>
  <p><input type="text" :value="city"></p>
  <p><input type="text" v-model="city"></p>
</div>
<script src="./vue.js"></script>

<script>
```

```
  var vm = new Vue({
    el:'#app',
    data:{
      city:'北京'
    },
  })
</script>
```

> v-model对应的city发生变化后，其他的{{ }} 和 :value的值也会发生变化

效果：



{{**表达式**}}

v-text="表达式"

v-html="表达式"

v-bind:src="表达式"

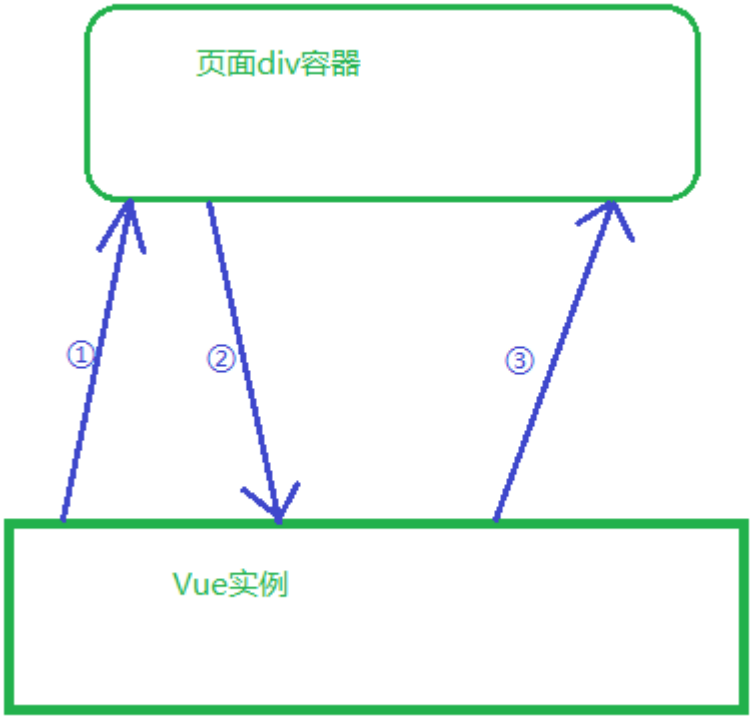
表达式此情此景有两种意思：

1. 有具体的返回信息
2. 按照javascript的正确语法规则设置相关内容

响应式

目标：

知道什么是响应式



v-model数据双向绑定步骤：

1. ①页面初始加载，vue的数据渲染给div容器
2. ②通过v-model修改数据，修改的信息会直接反馈给vue内部
3. ③vue的数据发生变化，页面上(也包括Vue实例本身)用到该数据的地方会重新编译渲染

> 2和3 步骤在条件满足情况下会**重复**执行

响应式：

vue实例的data数据如果发生变化，那么页面上(或Vue实例内部其他场合)用到的地方会重新编译执行，这样就把更新后的内容显示出来了，这个过程就是“响应式”，即上图的步骤3

> 如果Vue实例内部对变化的数据有使用，也会同步更新编译执行

注意：

响应式 是Vue中非常重要的机制，请留意

加法计算器

案例：

利用 **v-model** 和 **事件绑定** 实现 加法计算器案例效果：

```
<div id="app">
  <input type="text" v-model="num1" />
  <span>+</span>
  <input type="text" v-model="num2" />
  <button @click="add">=</button>
  <span>{{ result }}</span>
</div>
<script src="./vue.js"></script>

<script>
  var vm = new Vue({
    el:'#app',
    data:{
      num1:null,
      num2:null,
      result:null
    },
    methods:{
      add(){
        this.result = Number(this.num1) + Number(this.num2)
      }
    }
  })
</script>
```

提示：

加法计算器用到的技术点有：v-model + 事件绑定 + 响应式 + this + 数据类型转换为数值的

简易原理

目标：

了解简易v-model实现原理

给input输入框中定义oninput事件，在该事件中把用户输入的信息都给随时获得到，并对data成员进行赋值

data成员变化了，页面上用到的地方就重新渲染，达成简易双向绑定的效果

> oninput：是事件，可以随时感知输入框输入的信息

具体设置：

```
<div id="app">
  <h2>v-model简易原理</h2>
  <p>{{city}}</p>
  <p><input type="text" :value="city"></p>
```

```
    <hr />
    <!-- oninput:是一个键盘事件，可以感知输入框输入信息状态 -->
    <!-- 事件@xxx="方法名称/语句" -->
    <!-- $event:在vue的事件被绑定元素的行内部，代表事件对象 -->
    <p><input type="text" :value="city" @input="city = $event.target.value"></p>
    <p><input type="text" :value="city" @input="feel"></p>
  </div>

  <script src="./vue.js"></script>
  <script>
    var vm = new Vue({
      el:'#app',
      data:{
        city:'北京'
      },
      // 给Vue实例 声明方法，该方法可以给事件使用
      methods:{
        feel(evt){
          // console.log(evt)  // InputEvent输入键盘事件对象
          // evt.target: 触发当前事件的元素节点dom对象(类似document.getElementById()的返回结果)
          // console.dir(evt.target)
          // evt.target.value // 获得输入框当前输入的信息
          // console.log(evt.target.value)
          // 把输入框已经输入的信息赋予给city
          this.city = evt.target.value
        }
      }
    })
  </script>
```

`结论`：

1. 事件声明没有小括号()，**第一个形参**就是 事件对象
2. 在元素行内事件驱动中可以直接使用**$event**,其也是事件对象

其他表单域使用(自学)

绑定多行文本框

```
<textarea v-model="name"></textarea>
<div>{{ name }}</div>
```

> 注意：
>
> 多行文本框中不要使用`{{ name }}`的方式绑定，如果这样就没有双向绑定效果了

绑定复选框

- 绑定一个复选框

```
遵守协议：<input type="checkbox" v-model="xieyi">
<div>{{ xieyi }}</div>    <!-- 体现 true 或  false信息-->
<script>
    var vm = new Vue({
       el: '#app',
        data: {
            xieyi: true
        }
    })
</script>
```

- 绑定多个复选框

```
吃饭：<input type="checkbox" value="eat" v-model="hobby"><br>
睡觉：<input type="checkbox" value="sleep" v-model="hobby"><br>
敲代码：<input type="checkbox" value="qiao" v-model="hobby"><br>
{{ hobby }}    <!-- hobby体现为数组信息  ['eat','sleep','qiao']-->
<script>
    var vm = new Vue({
       el: '#app',
        data: {
            hobby: []
        }
    });
</script>
```

> 此种方式需要input标签提供value属性

绑定单选框

```
男<input type="radio" name="xingbie" value="男" v-model="sex">
女<input type="radio" name="xingbie" value="女" v-model="sex">
<p>{{sex}}</p>  <!-- 体现 男 或 女 的字符串信息-->
<script>
    var vm = new Vue({
       el: '#app',
       data: {
         sex: ''
       }
    });
</script>
```

绑定下拉列表

```
<div id="app">
  居住城市：
```

```
  <select v-model="mycity">
    <option disabled value="">请选择</option>
    <option value="B">北京</option>
    <option value="S">上海</option>
    <option value="G">广州</option>
  </select>
  <span>Selected: {{ mycity }}</span> <!--mycity: 体现为 B、S、G等字符串信息-->
</div>
<script>
    var vm = new Vue({
       el: '#app',
        data: {
            mycity: 'S'
        }
    });
</script>
```

v-for

遍历数组

目标：

使用v-for指令，熟练遍历数组信息

语法：

```
<标签 v-for="成员值 in 数组"></标签>
<标签 v-for="(成员值,下标) in 数组"></标签>
```

示例：

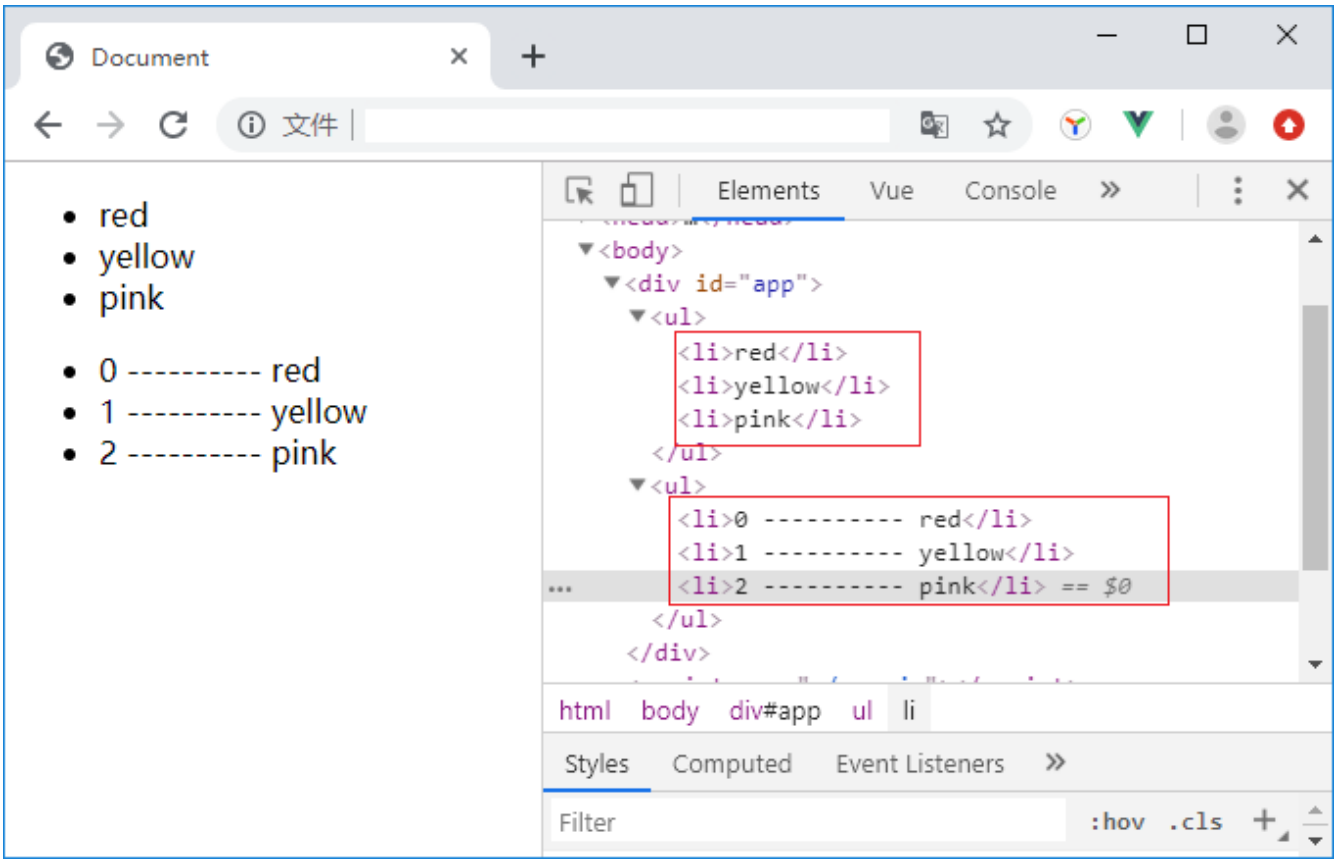
```
<div id="app">
  <ul>
    <li v-for="item in color">{{item}}</li>
  </ul>
  <ul>
    <li v-for="(item,k) in color">{{k}}-----{{item}}</li>
  </ul>
</div>

<script src="./vue.js"></script>

<script>
  var vm = new Vue({
    el:'#app',
    data:{
      color:['red','yellow','pink']
```

```
    },
    methods:{
    }
  })
</script>
```

效果：



注意：

使用v-for指令的html标签，由于遍历机制，本身标签会被**创建多份**出来

品牌管理案例

列表展示

目标：

通过v-for把vue实例定义的data数组对象集 品牌信息列表展示出来

步骤：

1. 复制现成的模板代码文件、引入vue.js文件

```
<script src="./vue.js"></script>
```

2. 实例化Vue对象，并通过data声明 brandsList 品牌列表信息

```
  var vm = new Vue({
    el:'#app',
    data:{
      // 品牌数组对象集
      brandList:[
        {id:10, name:'宝马', ctime:new Date()},
        {id:11, name:'奔驰', ctime:new Date()},
        {id:12, name:'奥迪', ctime:new Date()},
      ],
    },
  }
```

3. 通过v-for给tr标签遍历展示品牌信息

```
<tr v-for="(item,k) in brandsList">
  <td><input type="checkbox"></td>
  <td>{{ item.id }}</td>
  <td>{{ item.name }}</td>
  <td>{{ item.ctime }}</td>
  <td><button>删除</button></td>
</tr>
```

提示：

可以在现有的brand.html模板基础上做应用

效果：

品牌案例管理

添加 | 请输入关键字

	序号	名称	创建时间	操作
☐	10	宝马	Mon Dec 16 2019 15:52:27 GMT+0800 (中国标准时间)	删除
☐	11	奔驰	Mon Dec 16 2019 15:52:27 GMT+0800 (中国标准时间)	删除
☐	12	奥迪	Mon Dec 16 2019 15:52:27 GMT+0800 (中国标准时间)	删除

添加品牌

目标：

实现品牌添加操作

步骤：

1. 给新品牌输入框设置v-model="newbrand" 双向绑定
2. 在data中声明newbrand成员
3. 给“添加”按钮设置@click="add"事件
4. 在methods中声明add方法，实现添加逻辑，用到了数据方法**unshift**,在数组的前边添加元素
5. 在data中另行声明一个名称为**xu**的成员，记录已经存在品牌的最大id值，以便自增时给新品牌使用

模板代码：

```
<table>
  <tr>
    <td>
      <input type="text" v-model="newbrand" />
      <button @click="add">添加</button>
    </td>
    <td><input type="text" placeholder="请输入关键字"></td>
  </tr>
</table>
```

Vue实例代码：

```
  <script>
    var vm = new Vue({
      el:'#app',
      data:{
        newbrand:'',  // 被添加的新品牌
        xu:13, // 已有品牌最大序号信息
        // 定义品牌列表信息,数组对象集
        brandList:[
          {id:10, name:'奔驰', ctime:new Date()},
          {id:11, name:'宝马', ctime:new Date()},
          {id:12, name:'奥迪', ctime:new Date()}
        ]
      },
      methods:{
        // 添加品牌
        add(){
          // 判断如果没有输入品牌这停止添加
          if(this.newbrand.trim().length===0){return false}

          // 把新品牌做成是对象格式
          var obj = {id:this.xu++, name:this.newbrand, ctime:new Date()}
          // 把obj通过数组元素形式添加给brandList大数组
          // 数组.push(元素)后缀追加  数组.unshift(元素)前置追加
          this.brandList.unshift(obj)

          // 清除添加好的品牌
          this.newbrand = ''
```

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

注意：

添加新品牌的**xu序号**，是临时设置的，真实项目中不用维护，数据库会自动生成

vue指令-v-for-:key介入

目标：

理解:key的作用

在vue中v-for做遍历应用时，都需要与:key一并进行使用

在2.2.0+版本里边，v-for使用的同时必须使用:key，以便vue能准确跟踪每个节点，从而重用和重新排序现有元素，你需要为每个数据项提供一个唯一的、代表当前项目的属性值赋予给key

语法：

```
<标签 v-for="" :key="可以代表每个项目的唯一的值"></标签>
```

应用：

给品牌列表的v-for设置:key使用

```
<tr v-for="(item,k) in brandsList" :key="item.id">
```

> 项目应用中，每个v-for必须结合:key一并使用

注意：

1. :key不设置，也是存在的，默认绑定每个项目下标序号信息
2. key是一个普通属性，前边设置**:冒号**，代表属性绑定

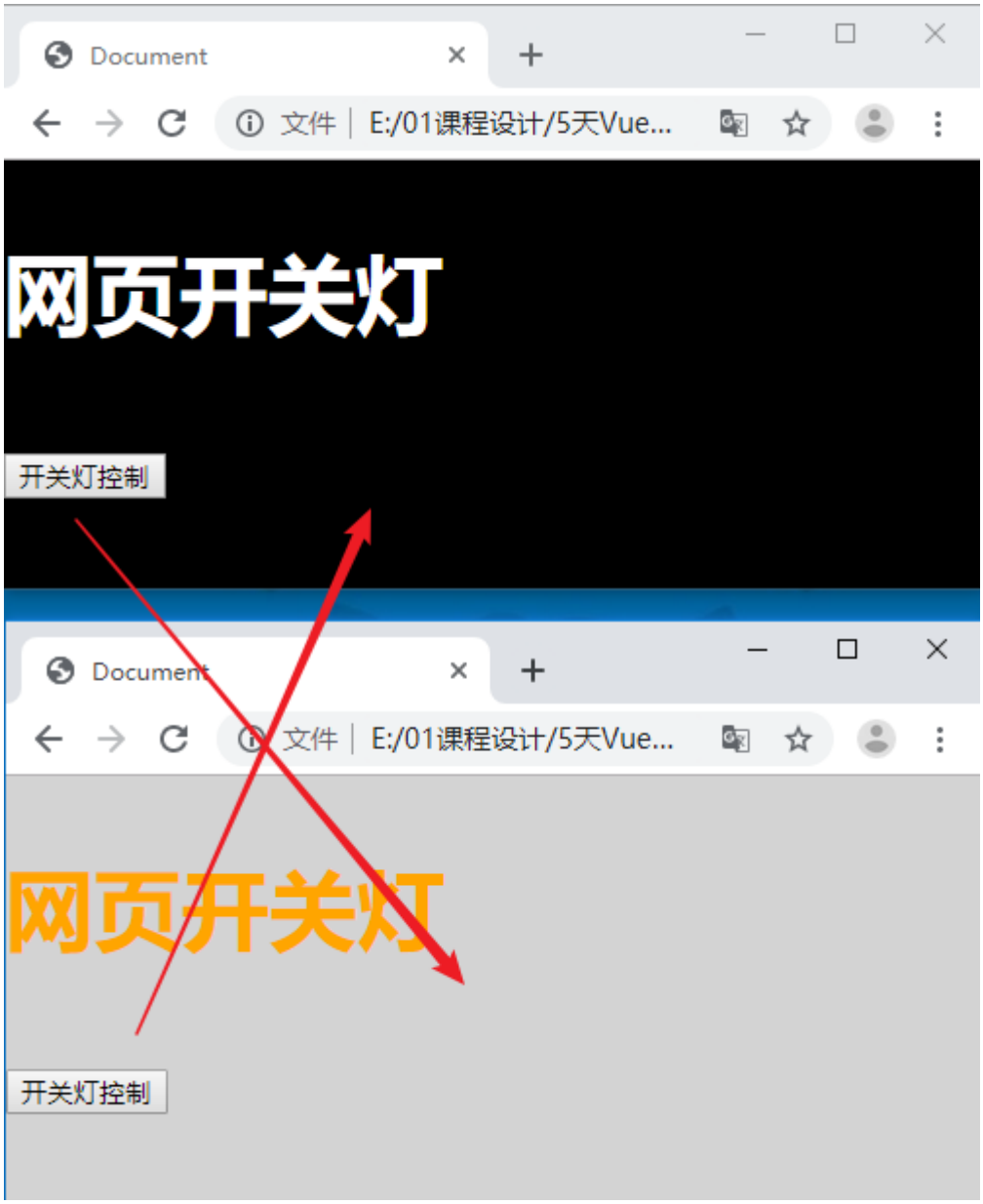
下午总结：

- v-model双向数据绑定
  - value
  - 绑定data成员

        - 响应式
    - 遍历指令
        - v-for="(成员名,下标名) in 数组"
        - 重复创建标签
- 实现品牌案例管理
    - 列表展示
        - 引入vue.js
        - 实例Vue对象
        - data生成品牌列表数据brandList
        - v-for遍历展示
    - 添加品牌
        - v-model="newbrand"
        - @click=add
        - data：newbrand xu
        - methods:add(){ 非空、创建品牌对象、unshift、清空 }
    - v-for 和 :key 应用

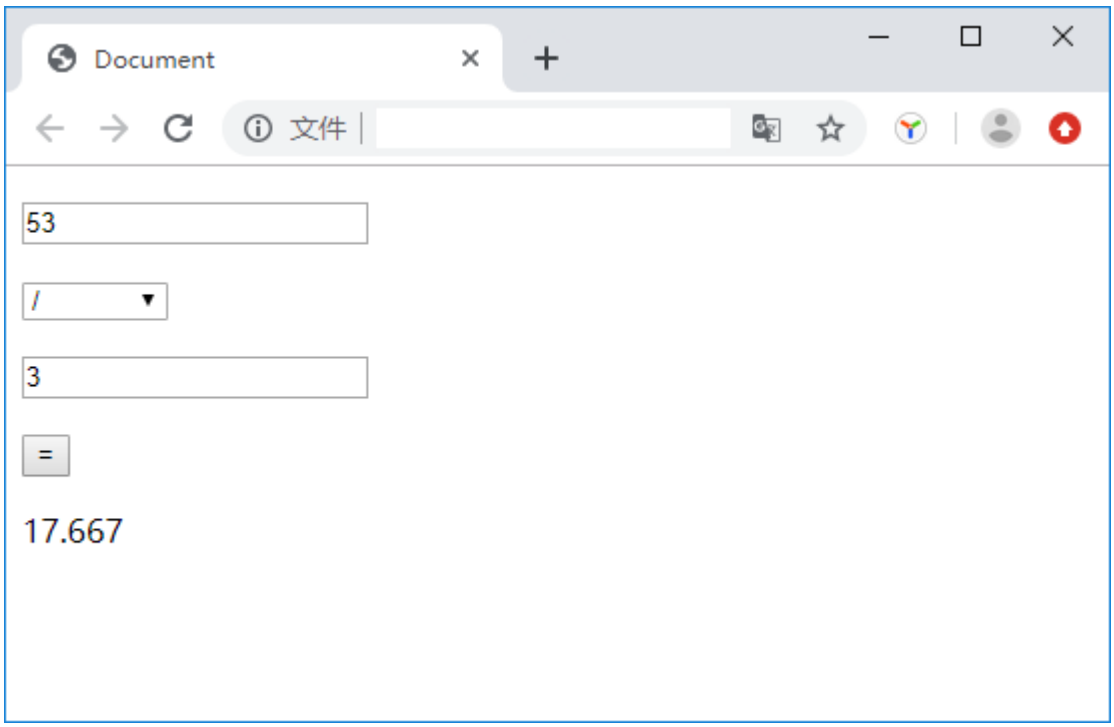
练习作业：

1. 利用 **事件绑定** + **style属性绑定** 实现网页开关灯效果



2. 同步完成 品牌管理 “列表展示 + 添加品牌” 功能

3. 下拉列表 + v-model 实现 加、减、乘、除 计算器效果


Document
文件
53
/
3
=
17.667