

Vue

vue.js导入

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

html代码块

```
<div id="myApp">
  {{ message }}
</div>
```

javascript脚本部分 (对html代码块的一个功能性的明述)

```
<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      message: 'Hello Vue.js world!'
    }
  })
</script>
```

v-if

条件判断式，根据表达式的真伪进行页面处理

```
<!-- Virtual DOM 虚拟的，若为假则直接不加载在页面中DOM（服务端渲染） -->
<div v-if="seen">2017最新发卖</div>
```

v-for

处理数组循环，将数据循环显示到页面时

```
<div id="myApp">
  <h3>游戏列表</h3>
  <!-- Virtual DOM -->
  <div v-if="seen">2017最新发卖</div>
  <ol>
    <li v-for="game in games">{{ game.title }} / {{ game.price }}元</li>
  </ol>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      seen: true,
      games: [{
        title: '勇者斗恶龙',
```

```
        price: 400
      },
      {
        title: '超级马里奥',
        price: 380
      },
      {
        title: '我的世界',
        price: 99
      }
    ],
  })
</script>
```

v-model

为页面输入框进行数据绑定（绑定一个模型一个变量，让变量自动反应用户输入的内容），例如：

```
input
select
textarea
components（组件）
```

```
<div id="myApp">
  <p>你最喜欢的游戏是: {{ mygame }}</p>
  <input v-model = "mygame">
</div>
<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      mygame: '超级马里奥'
    }
  })
</script>
```

游戏介绍 - Vue.js

Document

文件 | D:/Desktop/vue/index.html

应用 扩展程序 YouTube Google 翻译 前端

游戏列表

2017最新发卖

1. 勇者斗恶龙 / 400元
2. 超级马里奥 / 380元
3. 我的世界 / 99元

你最喜欢的游戏是: 造梦西游

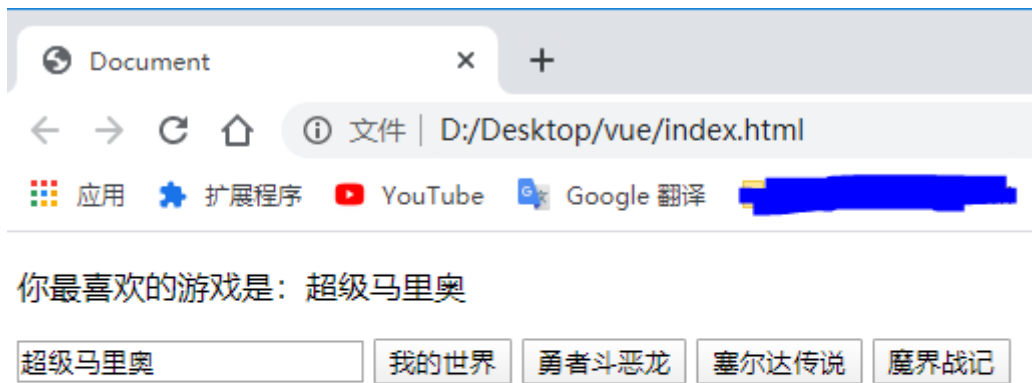
按钮事件

v-on

为页面元素绑定各种事件: keydown, keyup, click, dbclick, load, etd

```
<div id="myApp">
  <p>你最喜欢的游戏是: {{ mygame }}</p>
  <input v-model = "mygame">
  <button v-on:click = "btnClick('我的世界')">我的世界</button>
  <button v-on:click = "btnClick('勇者斗恶龙')">勇者斗恶龙</button>
  <button v-on:click = "btnClick('塞尔达传说')">塞尔达传说</button>
  <button @click = "btnClick('魔界战记')">魔界战记</button>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      mygame: '超级马里奥',
    },
    methods: {
      btnClick: function(pname) {
        this.mygame = pname;
      }
    },
  })
</script>
```



组件（一个页面，好多的组件好多小的功能区域块组成的）

component

定义页面的局部区域块，完成单独的页面小功能。示例：

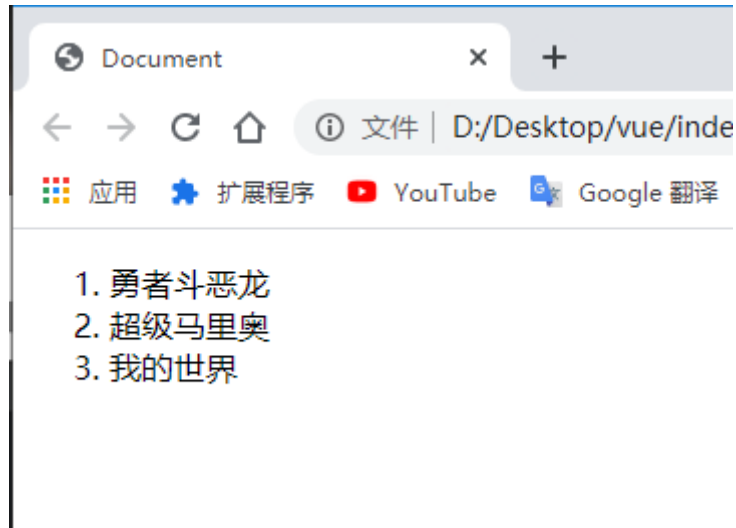
```
<div id="myApp">
  <ol>
    <game-item v-for="item in games" v-bind:game="item"></game-item>
  </ol>
</div>
<script>
  /*组件定义: game-item 定义了一个html标签，拓展了标准的html语言 */
  Vue.component('game-item', {
    props: ['game'],
    template: '<li>{{ game.title }}</li>'
  });

  var myApp = new Vue({
    el: '#myApp',
    data: {
      games: [{
        title: '勇者斗恶龙',
        price: 400
      },
      {
        title: '超级马里奥',
        price: 380
      },
      {
        title: '我的世界',
        price: 99
      }
    ],
  },
})
</script>
```

标签中穿了一个 props: ['game']属性，v-bind:game绑定上game属性

game-item标签还被定义了一个模板，这个非标准的html标签在网页渲染时应该渲染成模板中所显示的内容（template里面的内容）即：

- {{ game.title }}



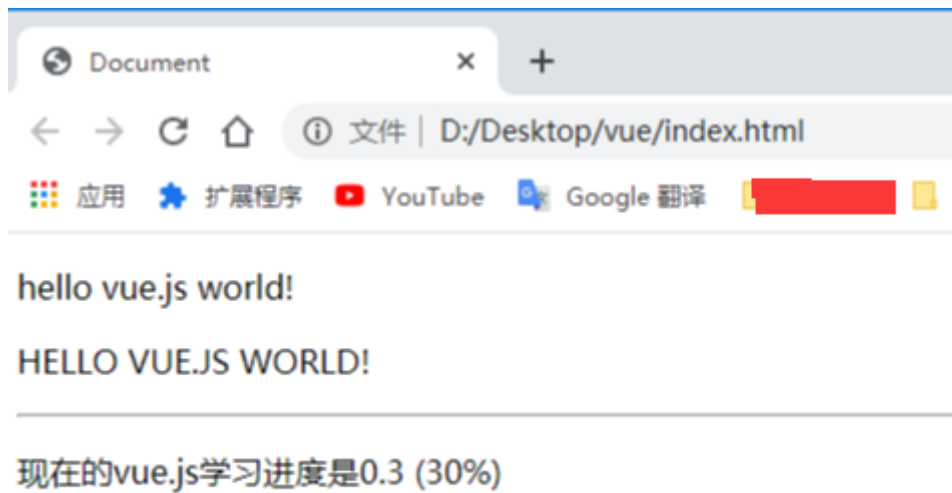
过滤器

filters

格式化变量内容的输出。（日期格式化，字母大小写，数字再计算等）

```
<div id="myApp">
  <p>{{ message }}</p>
  <p>{{ message | toupper }}</p>
  <hr>
  <p>现在的vue.js学习进度是{{ num }} ({{ num | topercenter }})</p>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      message: 'hello vue.js world!',
      num: 0.3
    },
    filters: {
      toupper: function(value) {
        return value.toUpperCase();
      },
      topercenter: function(value) {
        return value * 100 + '%';
      }
    }
  })
</script>
```



计算属性

computed

处理元数据（从数据库中取出的数据），便于进行二次利用（比如：消费税自动计算功能）

```
<div id="myApp">
  原价格: {{ price }}含税价: {{ priceInTax }}折合人民币{{ priceChinaRMB }}
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      price: 2998
    },
    computed: {
      priceInTax: function() {
        return this.price * 0.8;
      },
      priceChinaRMB: function() {
        return Math.round(this.priceInTax / 16.75);
      }
    }
  });
</script>
```



观察属性

\$watch

与computed属性类似，用于观察变量的变化。然后进行相应的处理。

```
<div id="myApp">
  原价格: {{ price }}含税价: {{ priceInTax }}折合人民币{{ priceChinaRMB }}
  <hr>
  <button @click = "btnClick(1000)">加价1000</button>
</div>

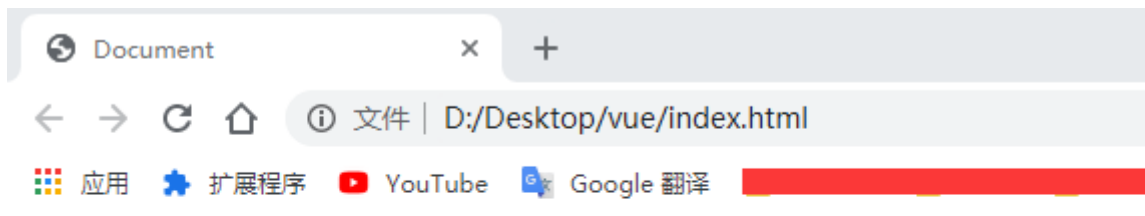
<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      price: 0,
      priceInTax: 0,
      priceChinaRMB: 0,
    },
    watch: {
      price: function(newVal, oldVal) {
        console.log(newVal, oldVal);
        this.priceInTax = Math.round(this.price * 1.08);
        this.priceChinaRMB = Math.round(this.priceInTax / 16.75);
      },
    },
    methods: {
      btnClick: function(newPrice) {
        this.price += newPrice;
      },
    },
  });
</script>
```

与计算属性区分：计算属性中priceInTax、priceChinaRMB为methods中的两个方法，而在观察属性中作为整个vue实例中的两个成员的一个属性（在计算属性作为计算属性，在watch的实现方式里需定义成data的属性）

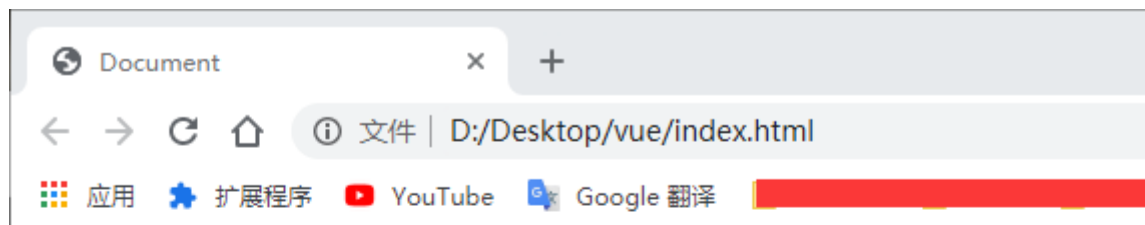
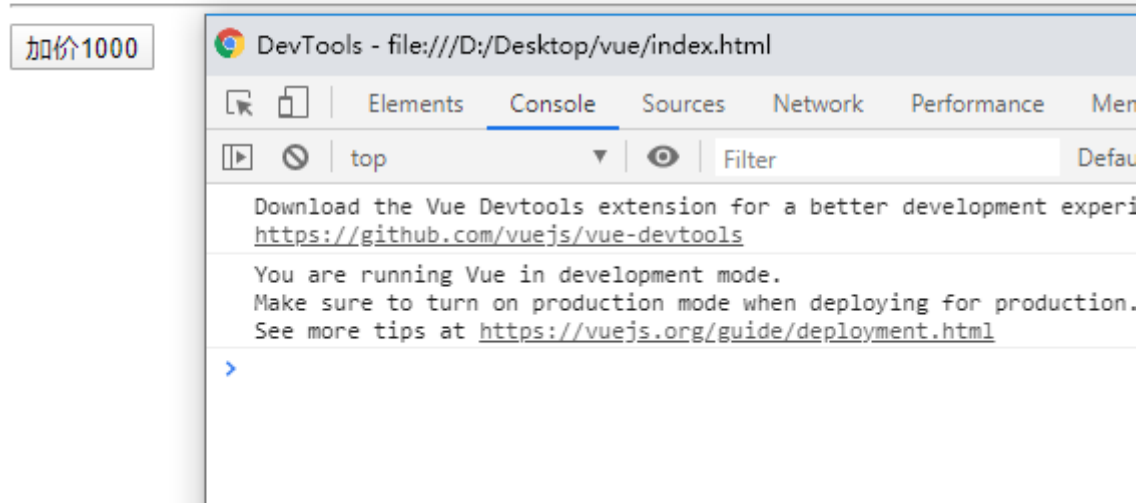
实现：

```
watch: {
  price: function(newVal, oldVal) {
    console.log(newVal, oldVal);
    this.priceInTax = Math.round(this.price * 1.08);
    this.priceChinaRMB = Math.round(this.priceInTax / 16.75);
  },
},
```

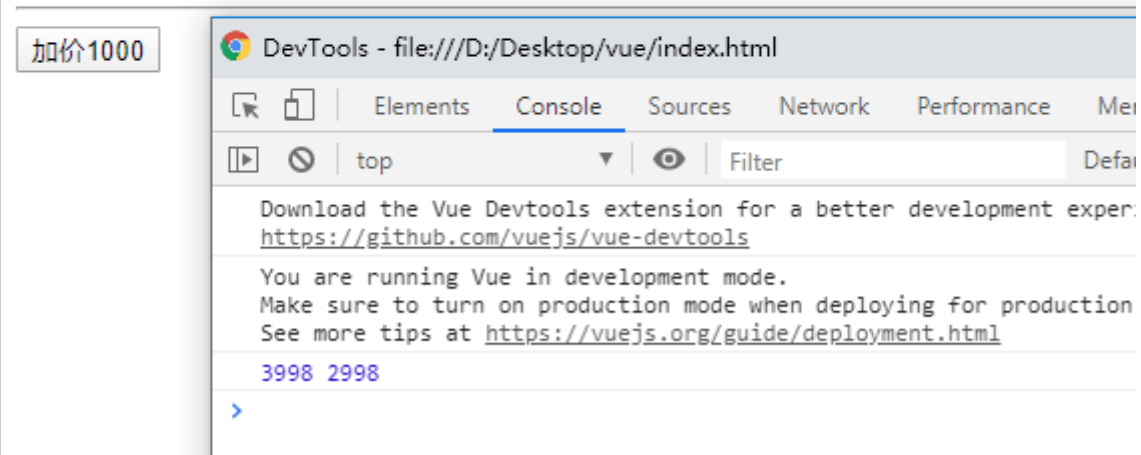
watch盯住price变量，当price发生变化时调用函数：变量名price需与属性名一致



原价格: 2998含税价: 0折合人民币0



原价格: 3998含税价: 4318折合人民币258



注意到，初始化的时候，含税价与人民币都为0，在进行一次加价后，含税价与人民币才变化，与代码实现逻辑一致，要解决含税价与人民币初始价格为：在实例化完对象后再进行赋值操作

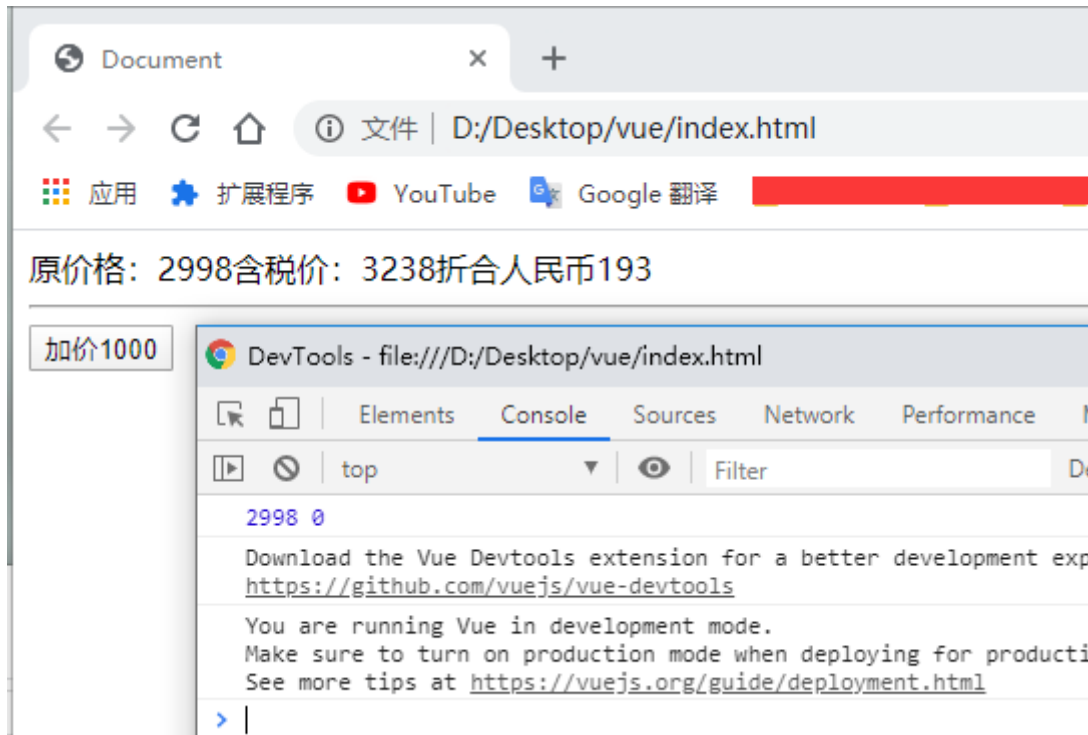
```
<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      price: 0,
      priceInTax: 0,
      priceChinaRMB: 0,
    },
    watch: {
      price: function(newVal, oldVal) {
```



```

        console.log(newVal, oldVal);
        this.priceInTax = Math.round(this.price * 1.08);
        this.priceChinaRMB = Math.round(this.priceInTax / 16.75);
    },
    },
    methods: {
        btnClick: function(newPrice) {
            this.price += newPrice;
        },
    },
    },
});
myApp.price = 2998;
</script>

```



设定计算属性

setter

设置计算属性，同步更新元数据的值 => (反推数据)

```

<div id="myApp">
  原价格: {{ price }}含税价: {{ priceInTax }}折合人民币{{ priceChinaRMB }}
  <hr>
  <button @click = "btnClick(10800)">含税价设为10800</button>
</div>

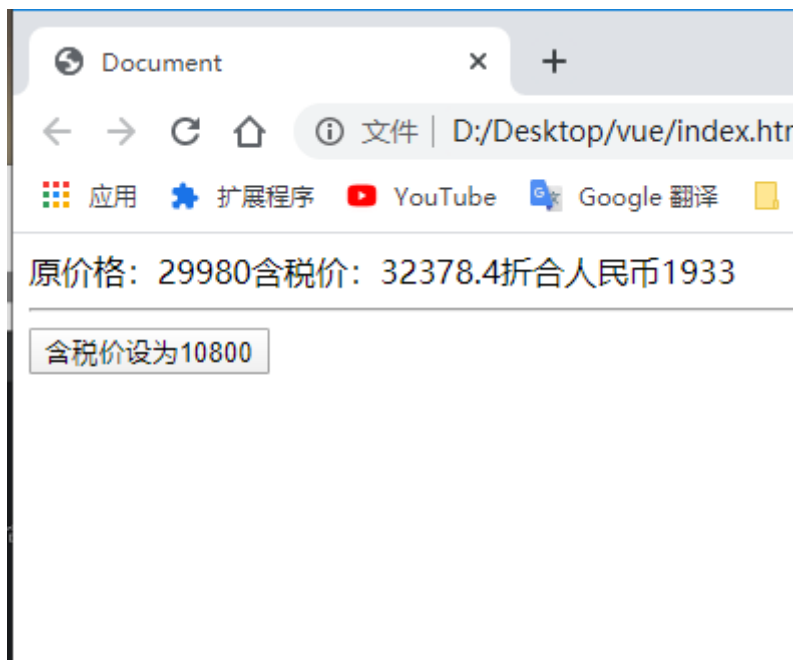
<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      price: 29980,
    },
    computed: {
      priceInTax: {
        get: function() {
          return this.price * 1.08;

```

```

    },
    set: function(value) {
      this.price = value / 1.08;
    }
  },
  priceChinaRMB: function() {
    return Math.round(this.priceInTax / 16.75);
  },
},
methods: {
  btnClick: function(newPrice) {
    this.priceInTax = newPrice;
  },
},
});
</script>

```



点击后



Class属性绑定

v-bind:class

为html标记绑定样式单class属性

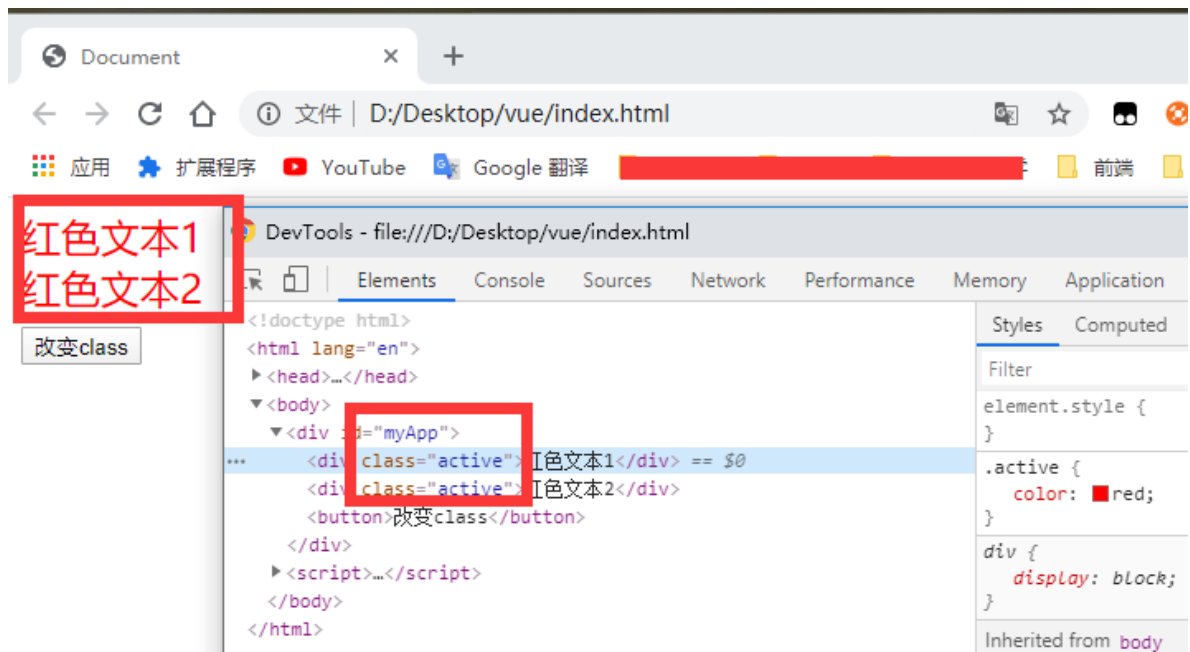
```
<style>
```

```

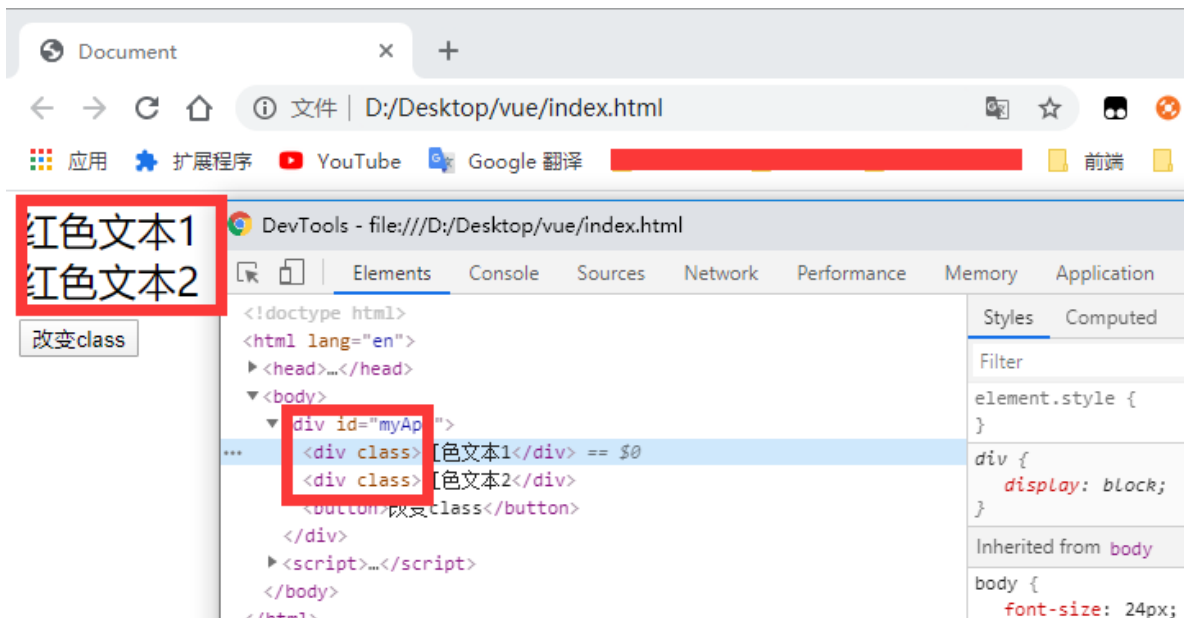
body {
  font-size: 24px;
}
.active {
  color: red;
}
</style>
<div id="myApp">
  <div v-bind:class = "{ active:isActive }">红色文本1</div>
  <div :class = "{ active:isActive }">红色文本2</div>
  <button @click = "btnClick">改变class</button>
</div>

<script>
var myApp = new Vue({
  el: '#myApp',
  data: {
    isActive: true,
  },
  methods: {
    btnClick: function() {
      this.isActive = false;
    },
  },
});
</script>

```



点击按钮后



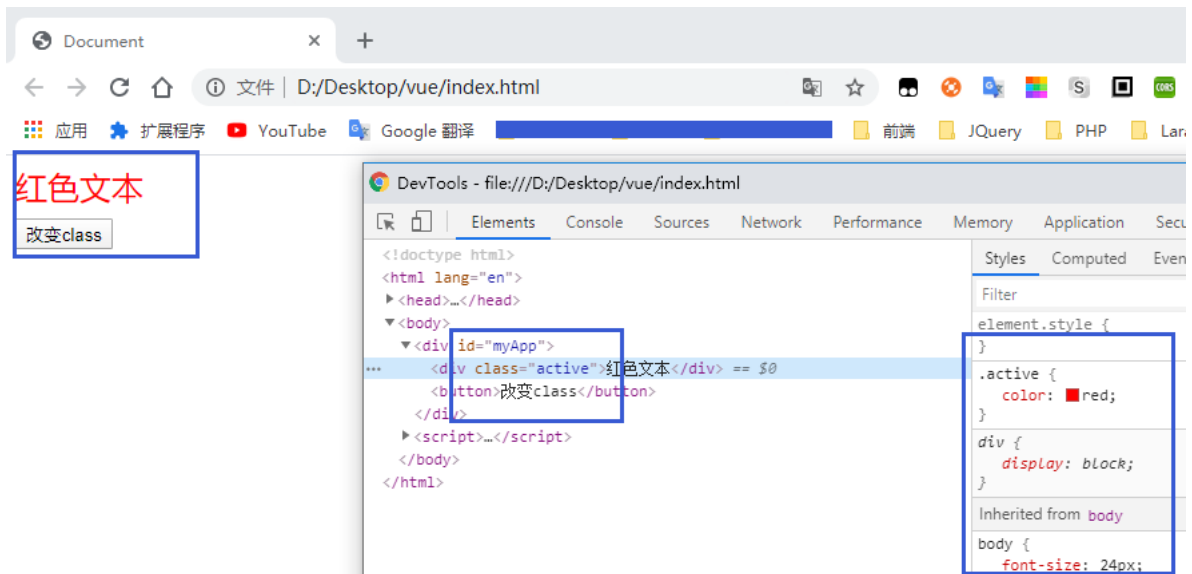
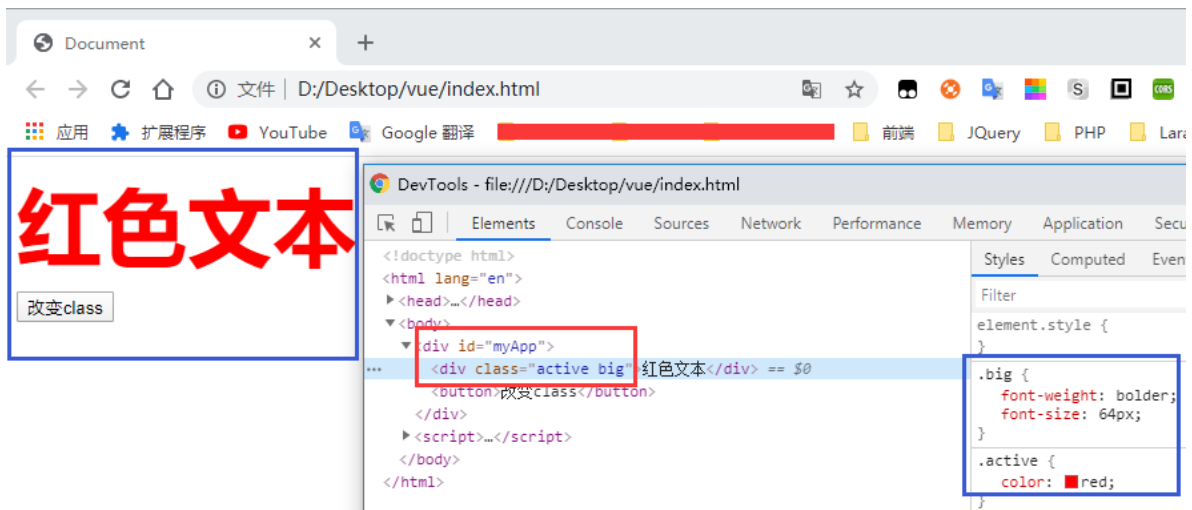
Class对象绑定

v-bind:class

为html标记绑定样式单class对象

```
<style>
  body {
    font-size: 24px;
  }
  .active {
    color: red;
  }
  .big {
    font-weight: bolder;
    font-size: 64px;
  }
</style>
<div id="myApp">
  <div :class = "myClass">红色文本</div>
  <button @click = "btnClick">改变class</button>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      myClass: {
        active: true,
        big: true,
      },
    },
    methods: {
      btnClick: function() {
        this.myClass.big = !this.myClass.big; /*可循环点击*/
      },
    },
  });
</script>
```



条件渲染

v-if v-else-if v-else

判断vue.js的变量的值，然后执行页面渲染逻辑 (if-elseif-else)

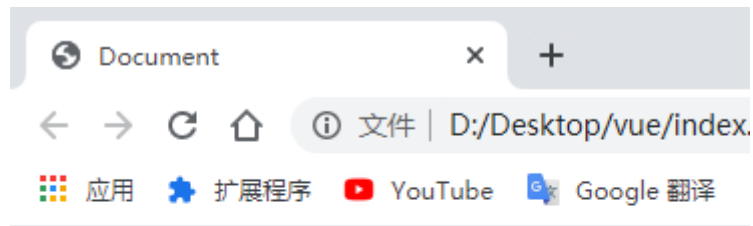
```

<div id="myApp">
  <h1 v-if = "result == 0">成绩未公布</h1>
  <h1 v-else-if = "result < 60">{{ result }}分 --考核未通过</h1>
  <h1 v-else-if = "result < 80">{{ result }}分 --考核通过</h1>
  <h1 v-else>{{ result }}分 --考核不错</h1>
  <button @click = "btnClick">考核成绩</button>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      result: 0,
    },
    methods: {
      btnClick: function() {
        this.result = Math.round(Math.random() * 100);
      },
    },
  });

```

</script>



成绩未公布

考核成绩

点击按钮后



元素显示

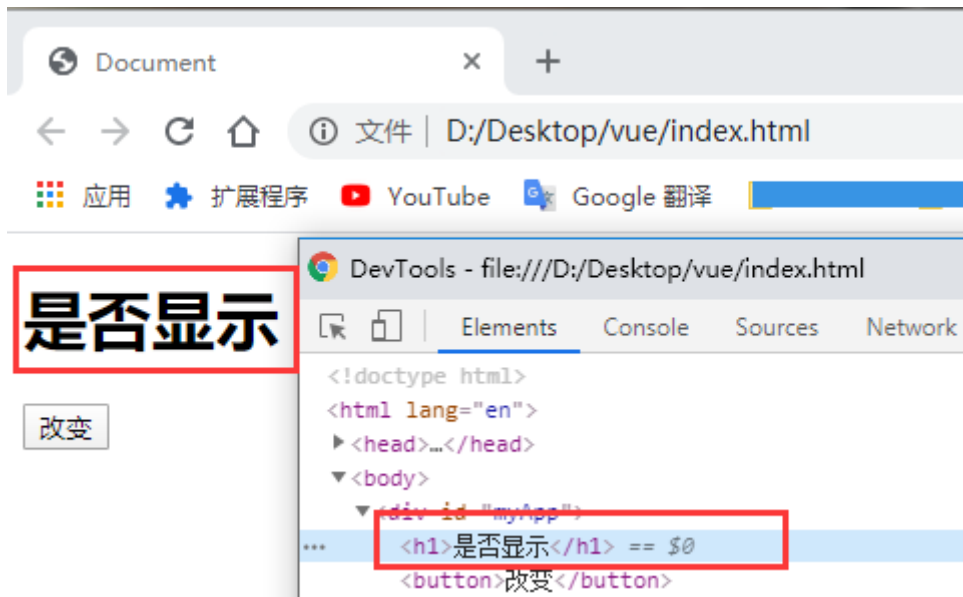
v-show

标记是否显示html元素 (注意: v-show设置的标记在html DOM树中不会消失)

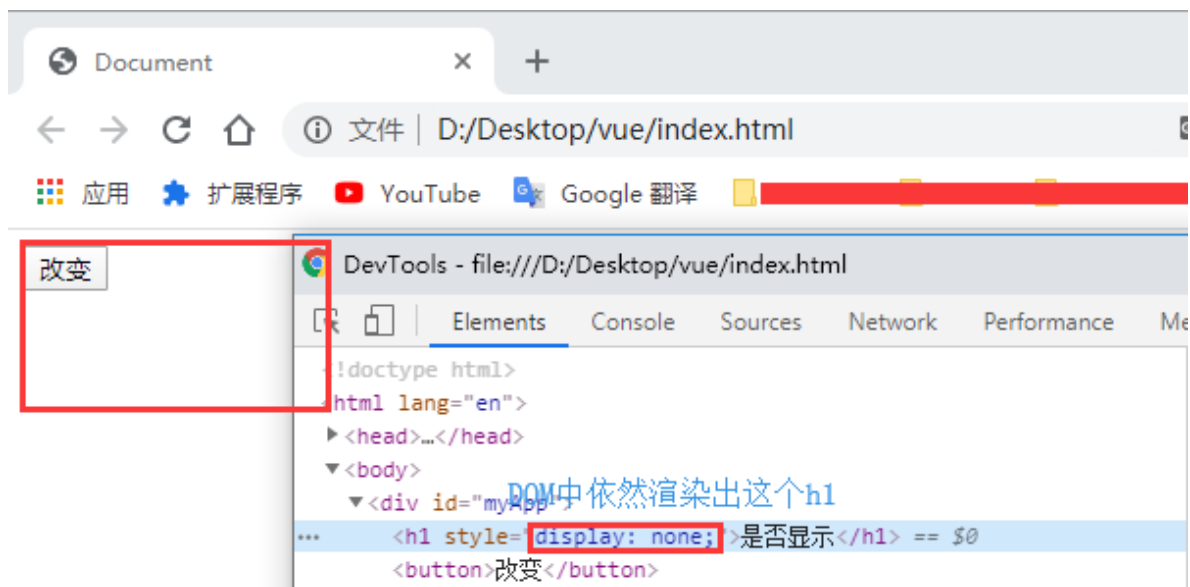
```
<div id="myApp">
  <h1 v-show = "result">是否显示</h1>
  <button @click = "btnClick">改变</button>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      result: true,
    },
    methods: {
      btnClick: function() {
        this.result = !this.result;
      },
    },
  });
```

</script>



点击按钮后



列表渲染

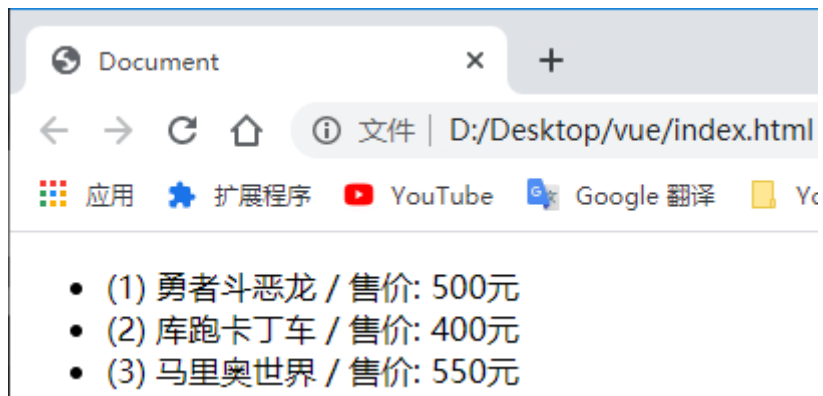
v-for

循环数组元素(对象、数列)整理内容显示到页面上

```
<div id="myApp">
  <ul>
    <li v-for = "(game, index) in games">({{ index + 1}}) {{ game.title
}} / 售价: {{ game.price }}元</li>
  </ul>
</div>

<script>
var myApp = new Vue({
  el: '#myApp',
  data: {
    games: [
      {title: "勇者斗恶龙", price: 500},
```

```
        {title: "库跑卡丁车", price: 400},
        {title: "马里奥世界", price: 550},
    ],
  },
});
</script>
```



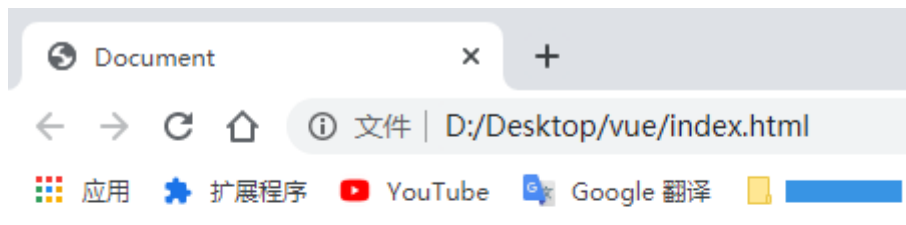
JS对象迭代

v-for

循环JS对象，把对象内容循环显示到页面上(打印属性、属性值，可供调试代码)

```
<div id="myApp">
  <h1>JS对象迭代</h1>
  <ul>
    <li v-for = "(value, key) in mygames">
      {{ key }} : {{ value }}
    </li>
  </ul>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      mygames: {
        title: "马里奥",
        price: 550,
        agerange: "全年齡",
      },
    },
  });
</script>
```

JS对象迭代

- title : 马里奥
- price : 550
- agerange : 全年齡

事件处理器

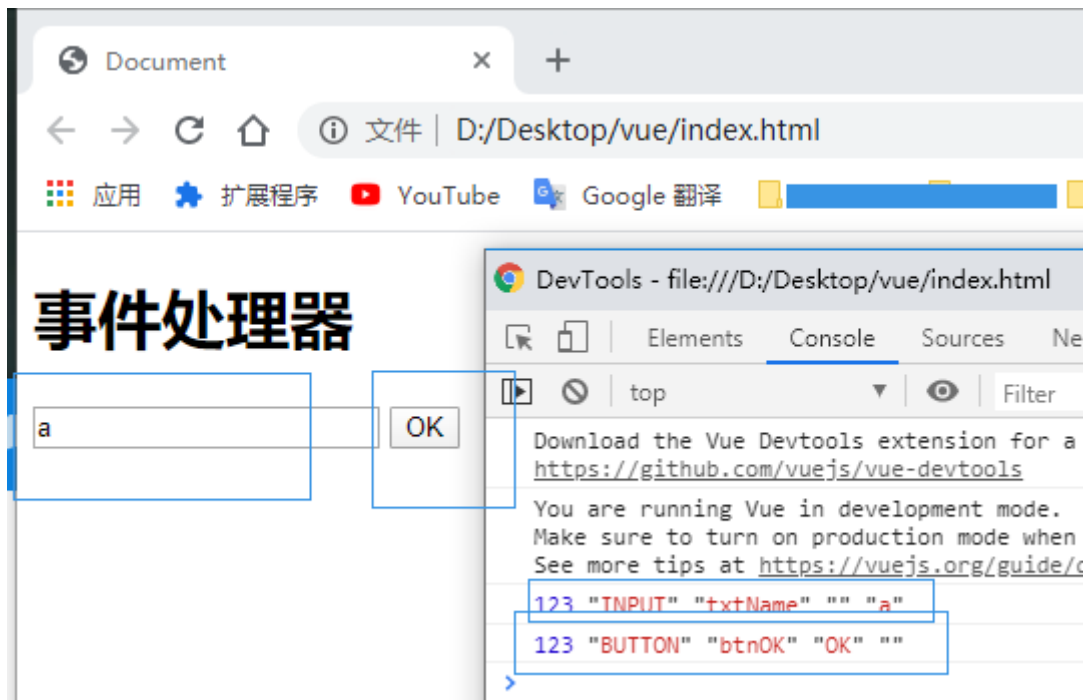
v-on:(event)\@(event)

页面元素的事件绑定 (click、keyup、load等)

当用户与页面交互时，获得用户信息的例子

```
<div id="myApp">
  <h1>事件处理器</h1>
  <input type="text" id="txtName" v-on:keyup = "txtKeyUp($event)">
  <button id="btnOK" v-on:click = "btnClick($event)">OK</button>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
    },
    methods: {
      txtKeyUp: function(event) {
        this.debugLog(event);
      },
      btnClick: function(event) {
        this.debugLog(event);
      },
      debugLog: function(event) {
        console.log(
          123,
          event.srcElement.tagName,
          event.srcElement.id,
          event.srcElement.innerHTML,
          event.key?event.key:""
        )
      }
    }
  });
</script>
```



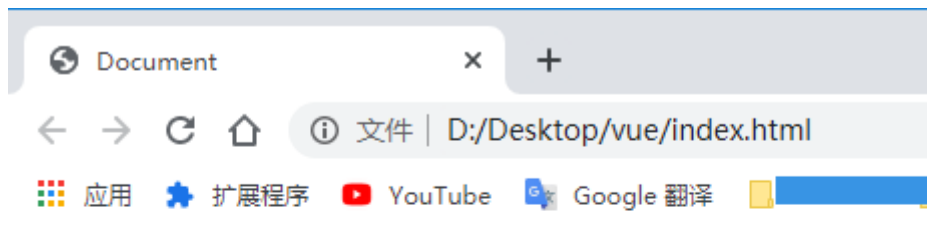
表单控件绑定

`v-model => input[type="text"]`

为表单控件元素创建双向数据绑定(将JS变量的值“快速”设定到控件上,然后将用户输入的内容“快速”设置回JS变量)

```
<div id="myApp">
  <h1>表单控件绑定</h1>
  <input type="text" v-model = "message" placeholder="编辑我!!! ">
  <p>Message is: {{ message }}</p>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      message: "马里奥",
    },
    methods: {
    },
  });
</script>
```



表单控件绑定

Message is: 马里奥

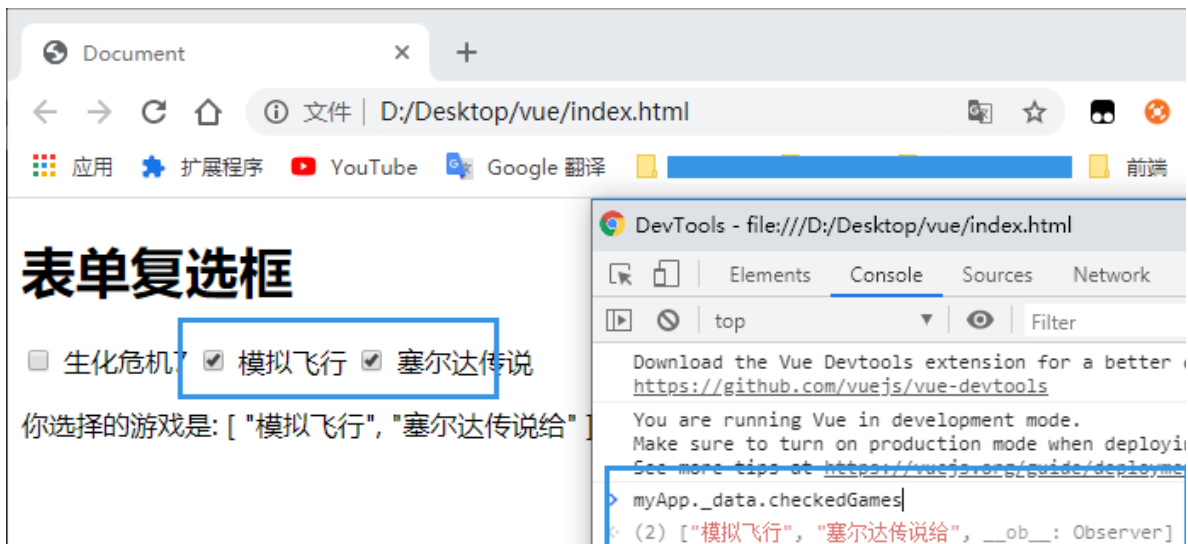
表单复选框

v-model => input[type = "checkbox"]

表单复选框绑定

```
<div id="myApp">
  <h1>表单复选框</h1>
  <input type="checkbox" name="shwj7" value="生化危机7" v-
model="checkedGames">
  <label for="shwj7">生化危机7</label>
  <input type="checkbox" id = "mnfx" value="模拟飞行" v-model =
"checkedGames">
  <label for="mnfx">模拟飞行</label>
  <input type="checkbox" id = "sedcs" value="塞尔达传说给" v-model =
"checkedGames">
  <label for="sedcs">塞尔达传说</label>
  <br>
  <p>你选择的的游戏是: {{ checkedGames }}</p>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      checkedGames: [],
    },
    methods: {
    },
  });
</script>
```



表单复选框

生化危机 模拟飞行 塞尔达传说

你选择的的游戏是: ["模拟飞行", "塞尔达传说给"]

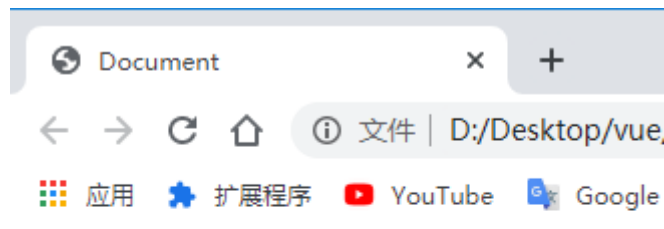
表单单选按钮

`v-model => input[type="radio"]`

表单单选按钮绑定

```
<div id="myApp">
  <h1>表单单选按钮</h1>
  <input type="radio" id="male" value="男" v-model="pickedSex">
  <label for="male">男</label>
  <input type="radio" id="female" value="女" v-model="pickedSex">
  <label for="female">女</label>
  <h3>选择爱好</h3>
  <input type="radio" id="game" value="玩游戏" v-model="pickedHobby">
  <label for="game">玩游戏</label>
  <input type="radio" id="movie" value="看电影" v-model="pickedHobby">
  <label for="movie">看电影</label>
  <br>
  <h3>选择结果</h3>
  <p>你选择的性别是: {{ pickedSex }}</p>
  <p>你选择的爱好是: {{ pickedHobby }}</p>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      pickedSex: '',
      pickedHobby: '',
    },
    methods: {
    },
  });
</script>
```



表单单选按钮

男 女

选择爱好

玩游戏 看电影

选择结果

你选择的性别是: 男

你选择的爱好是: 看电影

表单下拉框

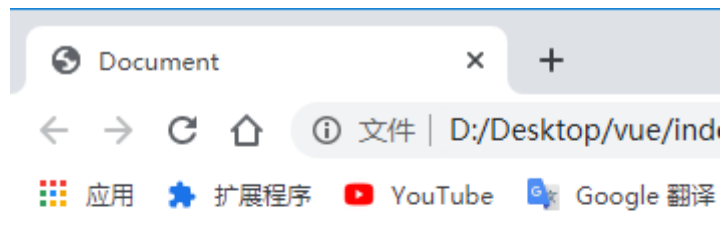
v-model => select

表单下拉框绑定

```
<div id="myApp">
  <h1>你最新喜欢的NBA球星</h1>
  <select v-model="likedNBAStar" style="width: 200px;">
    <option value="科比">科比</option>
    <option value="詹姆斯">詹姆斯</option>
    <option value="库里">库里</option>
  </select>
  <h3>我的全明星</h3>
  <select v-model="likedNBASTars" multiple style="width: 210px;height:
210px;">
    <option value="阿德托昆博">阿德托昆博</option>
    <option value="格里芬">格里芬</option>
    <option value="巴特勒">巴特勒</option>
    <option value="保罗">保罗</option>
  </select>
  <br>
  <h3>选择结果</h3>
  <p>你选择的性别是: {{ likedNBAStar }}</p>
  <p>你选择的爱好是: {{ likedNBASTars }}</p>
</div>

<script>
  var myApp = new Vue({
    el: '#myApp',
    data: {
      likedNBAStar: "",
```

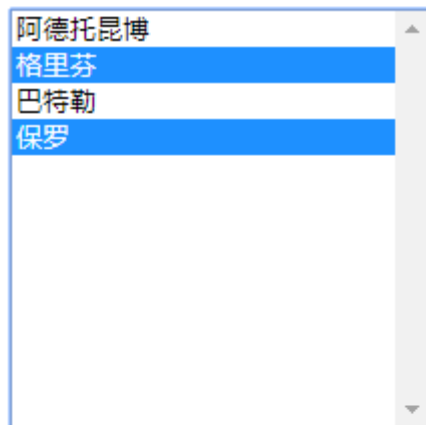
```
likedNBAStars: [],
},
methods: {},
});
</script>
```



你最喜欢的NBA球星

詹姆斯

我的全明星



选择结果

你选择的性别是: 詹姆斯

你选择的爱好是: ["格里芬", "保罗"]

表单修饰符

`/.lazy/.number/.trim`

用户输入内容时不做绑定数据的更新处理, 在控件的onchang实践中更新绑定的变量(提高页面处理效率)

```
<input type="text" id="username" v-model.lazy="username">
```

将用户输入的内容转换为数值类型, 如果用户输入非数值的时候, 返回NaN

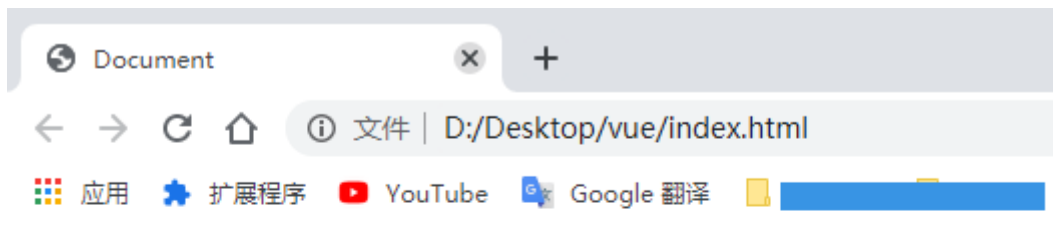
```
<input type="text" id="age" v-model.number="age">
```

自动去掉用户输入内容两端的空格

```
<input name="content" id="content" v-model.trim="content"></input>
```

```
<div id="myApp">
  <h1>用户注册</h1>
  <div>
    <label for="username">用户: </label>
    <input type="text" id="username" v-model.lazy="username"
@change="checkUsername">
    <span v-if="checkUsernameOK">可注册</span>
  </div>
  <div>
    <label for="age">年龄: </label>
    <input type="text" id="age" v-model.number="age">
  </div>
  <div>
    <label for="content">个人见解: </label>
    <textarea name="content" id="content" v-model.trim="content"
cols="55" rows="8"></textarea>
  </div>
  <h4>信息区</h4>
  <p>{{ username }}</p>
  <p>{{ age }}</p>
  <p><pre>{{ content }}</pre></p>
</div>

<script>
var myApp = new Vue({
  el: '#myApp',
  data: {
    username: '',
    checkUsernameOK: false,
    age: '',
    content: '',
  },
  methods: {
    checkUsername: function() {
      if(this.username.length > 0) {
        this.checkUsernameOK = true;
      }else {
        this.checkUsernameOK = false;
      }
    }
  },
});
</script>
```



用户注册

用户: 可注册

年龄:

个人简介:

测试
测试

信息区

fmujie

18

测试
测试

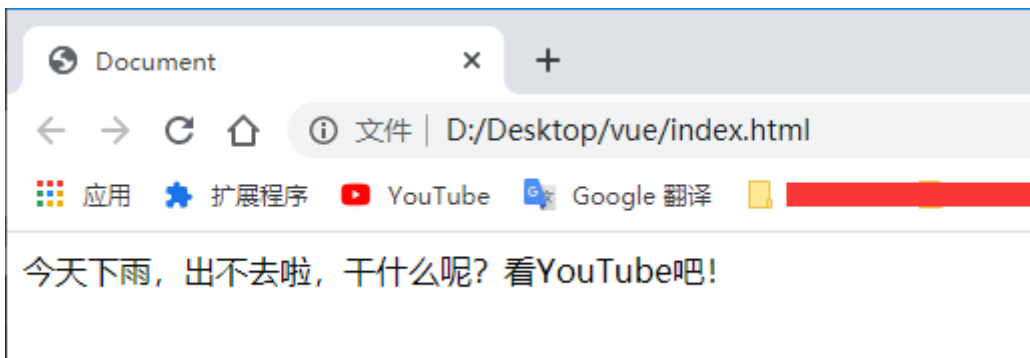
组件：基础的基础

- 组件 (Component, portlet)

组件就是页面上的一小块区域内容，完成一个小的页面功能

```
<div id="myApp">
  <today-weather></today-weather>
</div>

<script>
  Vue.component('today-weather', {
    template: '<div>今天下雨，出不去啦，干什么呢？看YouTube吧！ </div>'
  });
  var myApp = new Vue({
    el: '#myApp',
  });
</script>
```

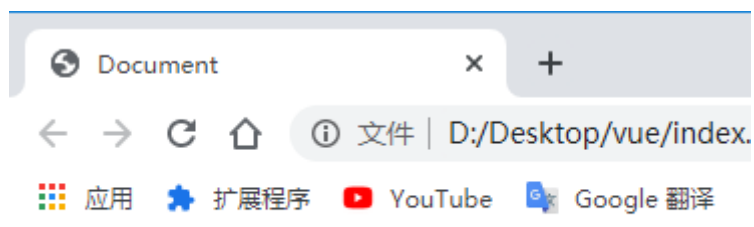
组件：局部的组件

- 组件的局部注册

Vue.js的组件不仅可以单独声明注册使用，还可以在Vue的实例中进行局部注册使用

```
<div id="myApp">
  <my-weather></my-weather>
</div>

<script>
  var WeatherComponent = {
    template: '<div>今天下雨，随他去吧！</div>'
  };
  var myApp = new Vue({
    el: '#myApp',
    components: {
      'my-weather': WeatherComponent
    },
  });
</script>
```



组件：表行组件

- 制作表行组件

为自己的页面表格编写表行组件

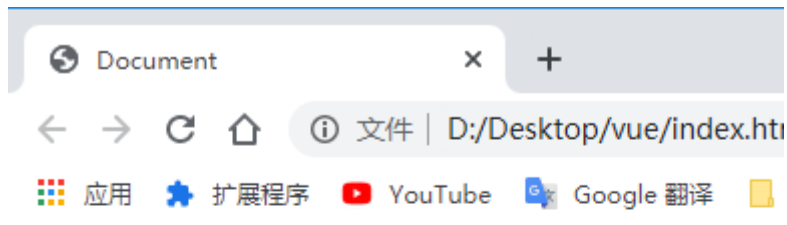
```
<div id="myApp">
  <h1>2017年最期待的游戏: </h1>
  <table border="1px">
    <tr>
      <td>编号</td>
      <td>游戏名称</td>
    </tr>
    <tr is="my-row1"></tr>
    <tr is="my-row2"></tr>
```

```

<tr is="my-row3"></tr>
<!-- <my-row1></my-row1>
<my-row2></my-row2>
<my-row3></my-row3> 注释部分自己实验，解决问题方法用is="" HTML样式问题-->
</table>
</div>

<script>
Vue.component('my-row1', {
  template: '<tr><td>(1)</td><td>塞尔达传说</td></tr>'
});
Vue.component('my-row2', {
  template: '<tr><td>(2)</td><td>新马里奥赛车</td></tr>'
});
Vue.component('my-row3', {
  template: '<tr><td>(3)</td><td>喷射乌贼娘二代</td></tr>'
});
var myApp = new Vue({
  el: '#myApp',
});
</script>

```



2017年最期待的游戏：

编号	游戏名称
(1)	塞尔达传说
(2)	新马里奥赛车
(3)	喷射乌贼娘二代

组件：数据

- 组件中的数据函数

为Vue.js组件添加数据，使组件可以动态显示各种数据，注意是数据函数，不是数据属性

```

<div id="myApp">
  <div>今天的天气是： <today-weather></today-weather>
</div>
</div>

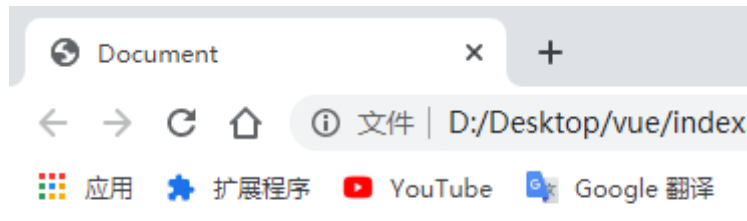
<script>
Vue.component('today-weather', {
  template: '<strong>{{ todayweather }}</strong>',
  // data: {
  //   todayweather: '雨夹雪', //自行实验注释部分，并查看报错信息，需是函数
  // },

```

```

    data: function () {
      return {
        todayweather: '雨夹雪',
      };
    },
  });
  var myApp = new Vue({
    el: '#myApp',
    data: {
    },
    methods: {
    },
  });
</script>

```



今天的天气是：雨夹雪

组件：传递数据

- 为组件传递数据

制作可接受参数的组件(交互)

```

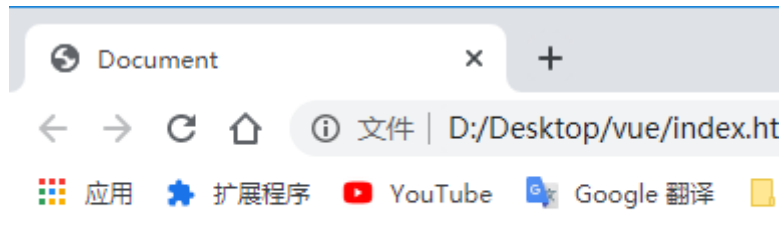
<div id="myApp">
  <h1>成绩评价</h1>
  <test-result :score = "50"></test-result>
  <test-result :score = "65"></test-result>
  <test-result :score = "90"></test-result>
  <test-result :score = "100"></test-result>

</div>

<script>
  Vue.component('test-result', {
    props: ['score'],
    template: '<div><strong>{{ score }}分, {{ testResult }}</strong>
</div>',
    computed: {
      testResult: function() {
        var strResult = "";
        if (this.score < 60) {
          strResult = "不及格";
        } else if (this.score < 90 ) {
          strResult = "中等生";
        } else if (this.score >= 90) {
          strResult = "优等生";
        }
        return strResult;
      }
    },
  },

```

```
});  
var myApp = new Vue({  
  el: '#myApp',  
});  
</script>
```



成绩评价

50分, 不及格
65分, 中等生
90分, 优等生
100分, 优等生

组件：传递变量

- 为组件传递变量数据

制作可接收变量参数的组件(更加啊灵活)

```
<div id="myApp">  
  <div>  
    请输入你的名字: <input type="text" v-model = "myname">  
  </div>  
  <hr>  
  <say-hello :pname = "myname"></say-hello>  
</div>  
  
<script>  
  Vue.component('say-hello', {  
    props: ['pname'],  
    template: '<div>你好, <strong>{{ pname }}!</strong></div>',  
  });  
  var myApp = new Vue({  
    el: '#myApp',  
    data: {  
      myname: 'fmujie',  
    },  
  });  
</script>
```



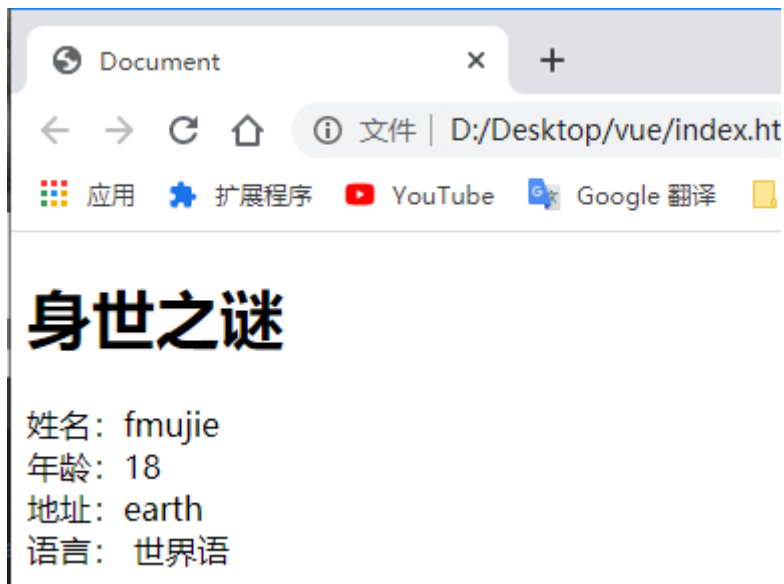
组件：参数验证

- props: 组件参数验证语法

为组件中接收到的变量进行逻辑验证

```
<div id="myApp">
  <h1>身世之谜</h1>
  <show-member-info name="fmujie" :age="18" :detail="{address:'earth',
language:'世界语'}"></show-member-info>
</div>

<script>
Vue.component('show-member-info', {
  props: {
    name: {
      type: String,
      required: true,
    },
    age: {
      type: Number,
      validator: function (value) {
        return value >= 0 && value <= 130;
      }
    },
    detail: {
      type: Object,
      default: function () {
        return {
          address: 'US',
          language: 'English',
        };
      }
    }
  },
  template: '<div>姓名: {{ this.name }}</br>' +
    '年龄: {{ this.age }}</br>' +
    '地址: {{ this.detail.address }}</br>' +
    '语言: {{ this.detail.language }}</div>',
});
var myApp = new Vue({
  el: '#myApp',
  data: {
    myname: 'fmujie',
  },
  methods: {
  },
});
</script>
```



组件：事件传递

- v-on

侦听组件事件，当组件触发事件后进行事件处理

- \$emit

触发事件，并将数据提交给事件侦听器

```
<div id="myApp">
  <h1>人生加法</h1>
  <add-method :a="6" :b="12" v-on:add_event="getAddResult"></add-method>
  <!-- 首先调用子组件，传两个参数a, b为侦听计算后的结果，
       在父组件当中定义了一个侦听子组件add_event的方法叫getAddResult,
  -->
  <hr>
  <h3>{{ result }}</h3>
</div>

<script>
Vue.component('add-method', {
  props: ['a', 'b'],
  template: '<div><button v-on:click="add">加吧</button></div>',
  //在子组件中定义了一个按钮，当按钮按下时会调用子组件add方法，
  //add方法会把相加结果以事件发射的方式传回给父组件
  methods: {
    add: function() {
      var value = 0;
      value = this.a + this.b;
      this.$emit('add_event', {
        result:value,
      });
    }
  }
});
var myApp = new Vue({
  el: '#myApp',
  data: {
    result: 0,
  },
});
```

```

    methods: {
      getAddResult: function(pval) {
        this.result = pval.result;
        //父组件通过getAddResult方法接收到子组件传过来的参数pval
        //并将传过来的result赋给自身的result
      }
    },
  });
</script>

```



组件: slot插槽

- slot

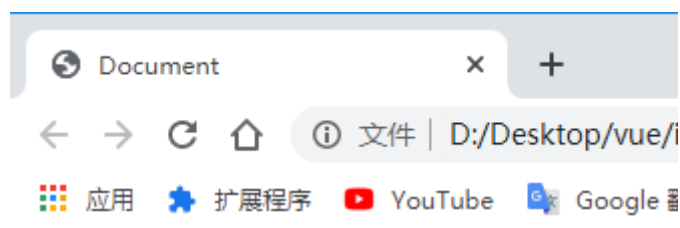
slot是父组件与子组件的通讯方式, 可以将父组件的内容显示在子组件当中(相当于一个内容显示的标记)

```

<div id="myApp">
  <say-to pname="fm">
    你的视频做的太差了
  </say-to>
  <say-to pname="john">
    千万不要学fm
  </say-to>
  <say-to pname="bily">
    你教教他们两个吧
  </say-to>
</div>

<script>
  Vue.component('say-to', {
    props: ['pname'],
    template: '<div>' +
      '你好, <strong> {{ pname }} </strong>!' +
      '<slot></slot>' + //将本行注释掉看看是否显示标签中的内容
      '</div>',
  });
  var myApp = new Vue({
    el: '#myApp',
  });
</script>

```



你好, **fm**! 你的视频做的太差了
 你好, **john**! 千万不要学fm
 你好, **bily**! 你教教他们两个吧

定义子组件时, 不是传属性(pname)而是传标记内容(文字信息), 标记内容通过插槽(slot)嵌入到子组件的模板当中

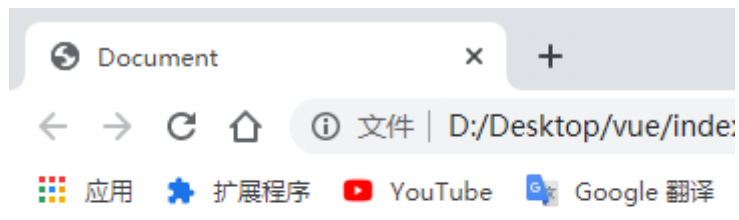
组件：组合slot

- slot命名

在子组件中通过为多个slot进行命名，来接收父组件的不同内容的数据

```
<div id="myApp">
  <nba-all-stars c="奥尼尔" pf="加内特">
    <span slot="sf">皮尔斯</span>
    <span slot="sg">雷阿伦</span>
    <span slot="pg">隆多</span>
  </nba-all-stars>
</div>

<script>
  Vue.component('nba-all-stars', {
    props: ['c', 'pf'],
    template: '<div>' +
      '<div>中锋: {{ c }}</div>' +
      '<div>大前: {{ pf }}</div>' +
      '<div>小前: <slot name="sf"></slot></div>' +
      '<div>分卫: <slot name="sg"></slot></div>' +
      '<div>控卫: <slot name="pg"></slot></div>' +
      '</div>',
  });
  var myApp = new Vue({
    el: '#myApp',
    data: {},
    methods: {},
  });
</script>
```



中锋：奥尼尔
大前：加内特
小前：皮尔斯
分卫：雷阿伦
控卫：隆多