# 小兴的VUE笔记

## 起步：

1. <!-- 开发环境版本，包含了有帮助的命令行警告 -->

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

1. 或者是

<!-- 生产环境版本，优化了尺寸和速度 -->

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

开发环境版本：相当于是拥有很多提示，可以更好快速的开发（建议开发时使用）

生产环境版本：代码全部被压缩过、不存在代码提示功能，更简洁（建议项目开发完后使用）

<div id="app">

{{ message }}

</div>

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

<script>

var app = new Vue({

el: '#app',

data: {

message: 'Hello Vue!'

}

</script>

})

## Vue对象类型的使用：

<script>

    var app=new Vue({

        el:"#app",

        data:{

            message:"你好 小兴！！",//基本类型

            school:{ //对象

                name:"程序员",

                mobile:"182-1637-2691"

            },

            hobby:["下象棋","跑步","打王者"]//数组

        }

    })

</script>

## Vue标签的使用：

### v-text标签：

V-text指令的作用是：设置标签的内容（textContent）

默认写法会替换全部内容，使用差值表达式{{}}可以替换指定内容

  <body>

      <!-- 如果使用标签将会替换标签内所有的内容，如果使用{{}}括号的话将不会替换 -->

    <div id="app">

        <h2 v-text="message+'!'">放上这句话并不会生效！！！！</h2>

        <h2 v-text="info+'加上字'">{{message+'aa'}} 这样放也不会生效！！！！</h2>

        <h2>{{message+'加上字'}} 可用''单引号添加内容！</h2>

    </div>

  </body>

  <!-- 开发环境版本，包含了有帮助的命令行警告 -->

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

<script>

  var app=new Vue({

        el: "#app",

        data:{

          message:" 小兴学vue！！！",

          info:"前端与移动教研院！"

        }

      })

</script>

### v-html标签：

v-html指令的作用是：设置元素的innerHTML

内容中有html结构会被解析为标签

V-text指令无论内容是什么，只会解析为文本

  <body>

    <div id="app">

        {{ message }}

        <p v-html="content"></p>

        <p v-text="content"></p>

    </div>

  </body>

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

<script>

  var app=new Vue({

        el: "#app",

        data:{

          content:"<a href='https://www.baidu.com'>程序员</a>"

        }

      })

</script>

### v-on标签：

v-on指令的作用是：为元素绑定事件

事件名不需要写on

指令可以简写为@

绑定的方法定义在methods属性中

方法的内部通过this关键字可以访问定义在data中数据

  <body>

    <div id="app">

       <input type="button" value="v-on指令" v-on:click="doIt"/>

       <input type="button" value="v-on指令简写" @click="doIs"/>

       <input type="button" value="双击事件" @dblclick="doIs"/>

       <h2 @click="changeFood">{{ food }}</h2>

    </div>

  </body>

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

<script>

  var app=new Vue({

        el: "#app",

        data:{

          food:"西兰花炒蛋"

        },

        methods: {// methods这个单词好像是固定的，不能写错！

          doIt:function(){

            alert("微IT");

          },

          doIs:function(){

            alert("微Java");

          },

          changeFood:function(){

            console.log(this.food);

            this.food+="好好吃呀！"

          }

        }

      })

</script>

### V-show指令：

v-show指令的作用是：根据真假切换元素的显示状态

原理是修改元素的display，实现显示隐藏

指令后面的内容，最总都会解析成为布尔值

值为true元素显示，值为false元素隐藏

<body>

    <div id="app">

        <button type="button" @click="changeIsShow">切换显示状态</button>

        <div v-show="age>=18">显示</div>

        <button type="button" @click="add">累加年龄</button>

    </div>

  </body>

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

<script>

  var app=new Vue({

        el: "#app",

        data:{

          isShow: false,

          age: 17

        },

        methods:{

          changeIsShow:function(){

            this.isShow= !this.isShow;

          },

          add:function(){

              console.log(this.age);

              this.age++;

          }

        }

      })

</script>

### V-if指令：

v-if指令的作用是：根据表达式的真假切换元素的显示状态

本质是通过dom元素来切换显示状态

表达式的值为true，元素存在与dom树中，为false时从dom树中移除

频繁移除dom树会对显示有一定压力

频繁的切换使用v-show，反之使用v-if前者切换消耗小

**同v-show差不多：不添加案例**

### V-bind指令：

v-bind指令的作用是：为元素绑定属性

完整的写法是v-bind:属性名

简写可以直接省略v-bind，只保留**:属性名**

表达式可以绑定样式：

:class="{active:isActive}" vue简写  样式在前、参数在后

还可以使用三元运算符的方式 :class="isActive?'active' : ' ' "

 <style>

        .active{

            border: 1px solid red;

        }

    </style>

</head>

<body>

    <div id="app">

        <img v-bind:src="imgSrc" alt="" /></br>

        <!-- 三元运算符方式

        <img :src="imgSrc" alt="简写" :title="imgTitle+'!!!'"

         :class="isActive?'active' : ''" @click="toggleActive"/> -->

        <!-- vue简写  样式在前、参数在后-->

         <img :src="imgSrc" alt="简写" :title="imgTitle+'!!!'"

         :class="{active:isActive}" @click="toggleActive"/>

    </div>

</body>

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

<script>

    var app = new Vue({

        el: "#app",

        data: {

            imgSrc: "http://www.itheima.com/images/logo.png",

            imgTitle: "简写",

            isActive:false

        },

        methods:{

            toggleActive:function(){

                this.isActive=!this.isActive;

            }

        }

    })

</script>

## 计数器demo实战：

巩固知识：

<body>

    <!-- 计数器功能区域 -->

    <div id="app">

        <div class="input-num">

            <button @click="sub">

                -

            </button>

            <span>{{ num }}</span>

            <button @click="add">

                +

            </button>

        </div>

    </div>

</body>

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

<script>

    var app = new Vue({

        el: "#app",

        data: {

            num: 1

        },

        methods: {

            add: function () {

                console.log('add');

                if (this.num < 10) {

                    this.num++;

                }else{

                    alert("最大10")

                }

            },

            sub: function () {

                console.log('sub');

                if (this.num >1) {

                    this.num--;

                }else{

                  alert("不能再减了！")

                }

            }

        }

    })

</script>