**引用**

<!-- 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@2/dist/vue.js"></script>

<!-- 生产环境版本，优化了尺寸和速度 -->

<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

**基本用法**

**html处**

<div id="app">

    {{ message }}

  </div>

**js处**

<script>

    new Vue({

      el: '#app',

      data: {

        message: 'Hello Vue.js!!'

      }

    })

  </script>

**el挂载点 注意：**

el 可以用其他选择器(class div...)来引用，但是尽量用id

el找到的元素内部将被改变，包括元素内部的子元素

el不能用于html和body标签

**复杂data的情况**

**html**

<div id="app">

    <p>{{ message }}</p>

    <h2>

      {{me.name}}

      {{me.phone}}

      {{housing[1]}}

    </h2>

  </div>

**js**

<script>

    new Vue({

      el: '#app',

      data: {

        message: 'Hello Vue.js!!',

        me: {

          name: "nanfei",

          phone: "2919",

        },

        housing: ["quzhou", 'hangzhou'],

      }

    })

  </script>

**vue指令**

**1.内容绑定、事件绑定**

**v-text（**设置标签的内容）

**html部分**

<div id="app2">

    <h2 v-text="message+'!'">111</h2><!-- 不会显示111 -->

<h2 v-text="info">111</h2><!-- 不会显示111 -->

<h2>{{message+'!'}}111</h2><!-- 会显示11 1-->

</div>

**js部分**

new Vue({

      el: "#app2",

      data: {

        message: "Nanfei",

        info: "2919"

      }

    })

**v-html（改变内容 如果是html内容会被解析）**

**html部分**

<div id="app2">

    <!-- v-html -->

    <h2 v-html="message"></h2><!-- 与v-text一样 -->

    <h2 v-html="htm"></h2><!-- 会解析html语言 -->

  </div>

**js部分**

new Vue({

      el: "#app2",

      data: {

        message: "Nanfei",

        info: "2919",

        htm: "<a href='https://www.baidu.com'>123</a>",

      }

    })

**computed数据**

**v-on基础（为元素绑定事件）**

**html部分**

<div id="app3">

    <input type="button" value="单击事件" v-on:click="click\_function">

    <input type="button" value="v-on简写" @click="click\_function">

    <input type="button" value="双击事件" @dblclick="click\_function">

 </div>

**js部分**

new Vue({

      el: "#app3",

      methods: {

        click\_function: function () {

          alert("click");

        }

      }

    })

**通过添加事件来改变DOM元素**

**html部分**

<div id="app3">

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

 </div>

**js部分**

new Vue({

      el: "#app3",

      data: {

        food: "eggs",

      },

      methods: {

        changeFood: function() {

          console.log(this.food)

          this.food += "good!"

        },

      }

    })

通过this指针来访问data里面的数据 从而改变DOM元素

**注意：如果用箭头函数会导致this指针指向不明确**

**2.显示切换、属性绑定**

**v-show（切换元素的显示和隐藏）**

**html部分**

<div id="app4">

    <!-- v-show -->

    <p v-show="true" v-html="age"></p>

    <p v-show="isShow" v-html="age"></p>

    <p v-show="age>18" v-html="age"></p><!-- 通过表达式真假来控制 -->

    <button @click="changeIsShow">切换</button><!-- 通过函数控制 -->

 </div>

**js部分**

new Vue({

      el: "#app4",

      data: {

        isShow: true,

        age: 18,

      },

      methods:{

        changeIsShow:function(){

          this.isShow=!this.isShow;

        }

      }

    })

**v-if（根据表达式的真假来切换元素的显示和隐藏，操纵DOM元素）**

**html部分**

<div id="app4">

    <!-- v-if -->

    <p v-if="true">P1</p>

    <p v-if="age>1">P1</p>

    <p v-if="isShow">P2</p>

 </div>

**js部分**

new Vue({

      el: "#app4",

      data: {

        isShow: false,

        age: 18,

      },

      methods: {

        changeIsShow: function () {

          this.isShow = !this.isShow;

        }

      }

  })

**与v-show的区别：**

v-if直接把元素删除，而v-show更改元素的样式

一般需要频繁更改用v-show 反之用v-if

**v-else-if和v-else**

<div v-if = "num === 1">我</div>

<div v-else-if = "num === 2">是</div>

<div v-else-if = "num === 3">Nan</div>

<div v-else = "num === 4">Fei</div>

**v-bind（设置元素的属性）**

**html部分**

<div id="app5">

    <img v-bind:src="url" alt="">

    <p v-bind:class="isActive?'active':'1'">1</p><!-- 三元表达式的应用 -->

    <p :class="!isActive?'active':'2'">2</p><!-- 简写 -->

    <p :class="{active:isActive}">3</p><!-- 如果isActive为true 则active反之空 -->

  </div>

**js部分**

new Vue({

      el: "#app5",

      data: {

        url: "https://i0.hdslb.com/bfs/archive/771bd6d35c782a31565675f6d04f00fa1c48d0b9.png",

        t: "233",

        isActive: false

      }

    })

**3.列表循环、表单绑定**

**v-for（根据数据生成列表结构）**

**html部分**

<div id="app6">

    <ul>

      <button @click="add">+</button>

      <button @click="sub">-</button>

      <li v-for="item in arr">你好</li><!-- item可以换成其他名字 -->

      <li v-for="(item,index) in arr" :title="item">第{{index+1}}个元素：{{item}}</li>

      <li v-for="(item,index) in objArr">

        {{item.name}}

      </li>

    </ul>

  </div>

**js部分**

new Vue({

      el: "#app6",

      data: {

        arr: [5, 4, 3, 2, 1],

        objArr: [

          { "name": "Jack" },

          { "name": "Rose" },

        ]

      },

      methods: {

        add: function () {

          this.arr.push(6);

        },

        sub: function () {

          this.arr.shift();

        }

      }

    })

**v-on补充（自定义参数，事件修饰符）**

**html部分**

<div id="app7">

    <button @click="dolt(1,2)">dolt</button>

    <input @keyup.enter="sayHi" type="text"><!-- .enter为事件修饰符 只有按下回车才能触发 -->

  </div>

**js部分**

new Vue({

      el: "#app7",

      methods: {

        dolt: function (p1, p2) {

          alert(p1 + p2);

        },

        sayHi: function () {

          alert("Hi");

        },

      }

    })

**v-modol（获取和设置表单元素的值，双向数据绑定）**

双向数据绑定：更改两边的任何一边会同步给对方

**html部分**

<div id="app8">

    <input type="button" value="重置message" @click="setM()">

    <input type="text" v-model="message" @keyup.enter="getM()">

    <h2>{{message}}</h2>

  </div>

**js部分**

new Vue({

      el: "#app8",

      data: {

        message: "Minami",

      },

      methods: {

        getM: function () {

          alert(this.message);

        },

        setM: function () {

          this.message = "Minami";

        }

      },

    })