基础：

vue.js代码：<https://unpkg.com/vue>

引用vue.js： <script src="https://unpkg.com/vue"></script>

运用模式：

var app5 = new Vue({

el: '#app-5', //对应html元素ID、也可以对应class

data: {

'message': '123456789' //变量名

},

methods: {

reverseMessage: function(){ //方法名

this.message = this.message.split('').reverse().join('')

}

}

})

Html：

直接引用变量：{{message}}

v-bind:title=”message” 绑定属性

v-if=”seen” 判断

v-for=”todo in todos” 循环

v-on:click=”functionName” 绑定事件

v-model=”message” 表单输入和应用状态之间的双向绑定

计算属性：computed:

var vm = new Vue({

el: '#example',

data: {

message: 'Hello'

},

computed: {

// 计算属性的 getter

reversedMessage: function () {

// `this` 指向 vm 实例

return this.message.split('').reverse().join('')

}

}

})

****计算属性是基于它们的依赖进行缓存的****

计算属性默认只有 getter ，不过在需要时你也可以提供一个 setter

这也同样意味着下面的计算属性将不再更新，因为 Date.now() 不是响应式依赖：

|  |
| --- |
| computed: {  now: function () {  return Date.now()  }  } |

****侦听属性：****watch

|  |
| --- |
| <div id="demo">{{ fullName }}</div> |
| Js： |

|  |
| --- |
| var vm = new Vue({  el: '#demo',  data: {  firstName: 'Foo',  lastName: 'Bar',  fullName: 'Foo Bar'  },  watch: {  firstName: function (val) {  this.fullName = val + ' ' + this.lastName  },  lastName: function (val) {  this.fullName = this.firstName + ' ' + val  }  }  }) |

当需要在数据变化时执行异步或开销较大的操作时，这个方式是最有用的