4-22

07

<body>

<div id="app">

<p>{{ msg }}</p>

</div>

<script type="text/javascript">

var vm = new Vue({ //new一个vue实例控制id为app的区域

el:'#app',//el==element

data:{ //mvvm中的m层（数据）

msg:'欢迎学习vue'

}

})

</script>

</body>

08

<div v-text=”msg2”>事实上这段文字会被msg2替代 </div>

<div v-html="msg2"></div> //直接显示msg2中的html内容

<script type="text/javascript">

var vm = new Vue({ el:'#app',//el==element

data:{

msg2:'这是msg2'

}

})

</script>

09

<body>

<div id="app">

<input type="button" value="按钮" v-bind:title="mytitle = ‘加上字符串也是合法的’"> //v-bind:提供绑定属性的功能，使得原属性与新增属性mytitle绑定。

可以省略v-bind 直接 :title

</div>

<script type="text/javascript">

var vm = new Vue({

el:'#app',

data:{

mytitle:'这是一个自己定义的title'

}

})

</script>

</body>

10

<body>

<div id="app">

<p>{{ msg }}</p>

<div v-html="msg2"></div>

<input type="button" value="按钮2" v-on:click="show"> //v-on事件绑定 可用@来替代v-on: 鼠标点击

<input type="button" value="按钮3" v-on:mouseover="show"> //鼠标覆盖

</div>

<script type="text/javascript">

var vm = new Vue({

el:'#app',

data:{

mytitle:'这是一个自己定义的title'

},

methods:{//属性中定义了当前vue实例所有可用的方法

show:function(){

alert('hello')

}

}

})

</script>

</body>

11

<body>

<div id="app">

<input type="button" value="动起来" @click='lang'>

<input type="button" value="停止" >

<h4>{{ msg }}</h4>

</div>

<script type="text/javascript">

var vm = new Vue({

el:'#app',

data:{

msg:'随便写点什么'

},

methods:{

lang(){

setInterval(() => {//箭头函数 内部this指向实例vue，因为如果不用的话，setInterval中的this是指向windows

// console.log(this.msg)//vm实例中要货取data中数据 要加this修饰

var start = this.msg.substring(0,1)//获取第一个字符

var end = this.msg.substring(1)//获取后面所有字符

this.msg = end + start//重新拼接并赋值给msg

},400)

}

}

})

</script>

</body>

12

<body>

<div id="app">

<input type="button" value="动起来" @click='lang'>

<input type="button" value="停止" @click='stop'>

<h4>{{ msg }}</h4>

</div>

<script type="text/javascript">

var vm = new Vue({

el:'#app',

data:{

msg:'随便写点什么',

intervalId: null //计时器的值

},

methods:{

lang(){

if(this.intervalId != null) return;//判断intervalId是否为null，不然重复点击就会开始多个计时器

this.intervalId = setInterval(() => {//箭头函数 内部t

var start = this.msg.substring(0,1)//获取第一个字符

var end = this.msg.substring(1)//获取后面所有字符

this.msg = end + start//重新拼接并赋值给msg

},400)

},

stop(){

clearInterval(this.intervalId)

this.intervalId = null;//按下停止后将intervalId重新赋值为null

}

}

})

</script>

</body>

13

事件修饰符

冒泡机制，触发内层后外层也会被触发。

<div id="app">

<div class="inner" @click="divHandler">

<input type="button" value="按钮" @click.stop="btnHandler"//阻止冒泡机制

</div>

</div>

阻止默认事件

<a href="http://www.baidu.com" @click.prevent="linkClick">BAIDU</a> //阻止默认事件，即百度的页面不会弹出。

使用捕获机制添加监听

<div class="inner" @click.capture="divHandler">

<input type="button" value="按钮" @click="btnHandler">

</div> //捕获机制，先触发外层div再触发内层button

只有点击当前元素才会触发事件

<div id="app">

<div class="inner" @click.self="divHandler">

<input type="button" value="按钮" @click="btnHandler">

</div>//不会被被动触发，即不会被捕获事件和冒泡事件所影响，只有在真正点击时触发。

只触发一次事件

<a href="http://www.baidu.com" @click.prevent.once="linkClick">BAIDU</a>

14

双向数据绑定v-model

<div id="app">

<h4>{{msg}}</h4>

<input type="text" style='width: 100%' v-model='msg'>

</div> //修改input中的内容也会改变h4中的内容，即将msg真正的修改了，实现双向绑定，只能运用在表单元素中

16

Vue中使用样式

<style>

.red{

color: red;

}

.thin{

font-weight: 200;

}

.italic{

font-style: italic;

}

.active{

letter-spacing:0.5em;

}

</style>

<div id="app">

<h1 :class="['thin','red']">这是h1，很大</h1>//绑定样式类名，用数组中括号，类名要加单引号，不然会当做属性来在data里面找

</div>

18

v-for循环

<div id="app">

<h1 v-for="item in list">{{item}}</h1>

</div>

<script type="text/javascript">

var vm = new Vue({

el:'#app',

data:{

list:[1,2,3,4,5]

},

methods:{

}

});

</script>

20

隐藏或显示

v-if 每次都会重新删除或创建元素，性能消耗较高

v-show 修改display：none样式，不会删除或创建，

<input type="button" value="按钮" @click="flag = !flag">

<h3 v-if="flag">v-if控制的元素</h3>

<h3 v-show="flag">v-show控制的元素</h3>

…

data:{

flag:true

},