3、3模板语法

v-text，v-html以及差值表达式{{“”}}的区别

<div>{{ name }}</div>

<div v-text=”name”></div>

<div v-html=”name”></div>

Js

new Vue({

el: ‘#app’,

data: {

name: “<h1>del</h1>”

}

})

V-text: name元素的值标签元素会给转义，也就是标签会以字符串的形式输出，没有css样式，而v-html里的标签会给识别出来，同时显示相应的css样式；差值表达式的结果跟v-text一样。

3、4计算属性，方法，监听器

<div id="app">

{{ fullname }}

</div>

Js

new Vue({

el: '#app',

data: {

firstname: "dell",

lastname: "ldb"

},

computed: { //计算属性

fullname: function() {

Console.log(“计算一次”)

return this.firstname + " " + this.lastname

}

}

})

computed具有缓存的特性，他依赖两个变量，firstname与lastname，当依赖的变量没有改变时就不用重新计算了，直接用上一次的计算结果，当你在控制台输入vm.firstname=”Li”时，他又会重新计算一遍，你可以看到控制台输出了“计算一次”。

以上代码为例，在data属性里增加一个fullname变量，将computed计算属性改变成watch 监听属性代码如下:

data: {

firstname: "dell",

lastname: "ldb"

Fullname: ”dell ldb”

},

watch: {

firstname : function() {

return this.fullname = this.firstname + " " + this.lastname

},

lastname : function() {

return this.fullname = this.firstname + " " + this.lastname

}

}

watch的作用是，当监听到的firsname或者lastname变量发生改变时，它会实时更新fullname的值，watch比较复杂，当watch与computed都可以使用时一般采用computed。

3、5计算属性的setter与getter

<div id="app">

{{ fullname }}

</div>

Js

var app = new Vue({

el: '#app',

data: {

firstname: "dell",

lastname: "ldb"

},

computed: {

fullname: {

get: function() {

return this.firstname + " " + this.lastname

},

set: function(value) { //接受一个修改的参数

var arr = value.split(" ");//以将字符串空格分割成一个数组

this.firstname = arr[0];

this.lastname = arr[1];

}

}

}

})

页面默认值是dell ldb

打开控制台，输入app.fullname=”Li Ming”,同时页面会渲染成LI MIng，此时set函数里面接受的value就是”Li Ming”,而这时firstname与lastname的值也跟着发生改变，所以computed

计算属性会重新计算一遍。同时同步更新页面的值。

3、6条件渲染

V-if与v-show的区别

<div id="app">

<div v-if="show">{{ message }}</div>

<div v-show="show">{{ message }}</div>

</div>

Js

var app = new Vue({

el: '#app',

data: {

show: true,

message: "hello world"

}

})

当变量show为true时，两个都会显示hello world。当为false时，两个都不会显示，v-if是直接把所在节点从DOM上删除了，v-show则是在该节点上加了display: none该属性，实际上是隐藏了。

<div v-if="show">

用户名：<input />

</div>

<div v-else="show">

邮箱：<input />

</div>

V-else要紧跟着v-if来使用.

Vue在重新渲染页面的时候，会尽量去复用页面上已经存在的DOM，例如以上代码为例，

当在用户名输入框输入值时，然后在切回邮箱输入框时，邮箱输入框里会保留用户名输入框的值，解决该问题的方法是在input节点上各加入一个key值，如key=”username”、key=”password”，这样Vue在重新渲染的时候知道该节点时唯一的，就不会再去复用其他节点了。(虚拟DOM

3、9 Set方法

2、3代码

<ul>

<li v-for=”item in list”>{{ item }}</li>

</ul>

<script>

Var example = new Vue({

el: ‘#example’,

data: {

list: [

{‘abac’ },

{‘mnbacac’ }

]

}

})

当你想要在页面上增加list的项时，打开控制台,输入Vue.set(example.list,”123456”),这样页面就会同时更新，或者是这样

example.$set(example.list,”123456”).如果是这样

example.list[2]=“123”.不会显示”123”,要想数组变化，页面也跟着变的响应式效果，

只能通过Vue定义的方法:pop,push,shift(删除第一项),unshift(往第一项加内容),splice(数组的截取),reverse(取反),sort(排序).

Vue.set(example.list,1,”123”);

改变list下标为1的项，变成123.

example.list.splice(1,1,”123”),从下标为1的地方开始，删除一项，并增加一项”123”.

也可以通过引用的方式添加项

Example.list=[{

{‘abac’ },

{‘mnbacac’ },

{‘123’}

}]