###### vue指令大全

带有前缀v-的称为vue的指令

1. **v-text**

用来更新textContent,等同于JS的text属性。例如<p v-text=’msg’></p>等价于<p>{{msg}}</p>

1. **v-html**

输出html，等同于js的innerHtml属性

1. **v-pre**

v-pre主要用于跳过这个元素和它的子元素编译过程，可以用来显示原始的Mustache标签。跳过大量没有指令的节点加快编译。

1. **v-cloak**

当网络较慢，网页还在加载 Vue.js ，而导致 Vue 来不及渲染，这时页面就会显示出 Vue 源代码。我们可以使用 v-cloak 指令来解决这一问题。解决屏幕闪动的问题。用来保持在元素上直到关联实例结束时进行编译。

1. **v-once**

v-once关联的实例，只会渲染一次。之后的重新渲染，实例及其所有的子节点将被视为静态内容跳过，可以用于优化更新性能。

1. **v-if**

实现条件渲染，会根据表达式值的真假条件来渲染元素。

1. **v-else**

v-else是搭配v-if使用的，它必须紧跟在v-if或者v-else-if后面，否则不起作用。当v-if不满足时，显示v-else的内容。

1. **v-else-if**

v-else-if充当v-if的else-if块，可以链式的使用多次，可以更加方便的实现switch语句。

1. **v-show**

也是根据条件展示元素，和v-if不同的是，如果v-if的值为false，则这个元素被销毁，不在dom中。但是v-show的元素会始终被渲染并保存在dom中，它只是简单的切换css的display属性为none。

注意：v-if有更高的切换开销，v-show有更高的初始渲染开销。如果需要频繁切换，则优先使用v-show，如果初始条件不太可能改变，则用v-if更好。

1. **v-for**

遍历数组来进行渲染，有下面两种遍历形式

使用for...in : <div v-for=”(item,index) in items></div> //index是一个可选参数，表示当前项的索引;

使用for...of : <div v-for=”item of items”></div>

在v-for中，拥有对父作用域属性的完全访问权限

注意：当v-for和v-if同处于一个节点时，v-for的优先级比v-if更高。这意味着v-if将运行在每个v-for循环中。

1. **v-bind**

用于动态的绑定一个或多个特性，可简写为一个冒号 :name等价于v-bind:name

1. **属性绑定**

v-bind:name=”name” //引号内的属性定义在data中，注意是引号而不是花括号

1. **内联字符串拼接**

<a :herf=” ‘http://’+addr”>跳转到百度</a>

data:{

addr:”www.baidu.com”

}

1. **class绑定**
2. 对象语法：

a-1. v-bind:class="{prop1:isTrue,prop2:isActive}"，在data中要声明属性值(isTrue/isActive)为true或false，为true时才应用这个CSS class，为false时不应用;

a-2. 声明对象名，在data中声明属性并指定是否可用：v-bind:class=”classObj”,在data中声明classObj:{class1:true, class2:false}。则只应用class1;

1. 数组语法：

b-1. 声明数组名，在data中定义需要的数组元素；所有数组元素都会被应用；

b-2. 声明数组并定义其元素，在data中定义元素是否可用：

v-bind:class=”[class1,class2,class3]” data中{class1:’class1’, class2:’class2’,class3:false}则只会应用class1和class2这两个类。

b-3. 声明数组和条件表达式，在data中定义条件表达式的值

1. **绑定内联样式**
2. 对象语法：

在template中声明属性，在data中定义对应的属性值

v-bind:style=”{background:color, fontSize:fontSize+’px’}”

data:{color:’green’, fontSize:25}

1. 数组语法：

v-bind:style="[prop1,prop2]"

data:{ prop1:{background:'green'}, prop2:{fontSize:’25px’,fontWeight:’bold’}}

1. 多重值绑定：

可以为style绑定中的属性提供一个包含多个值的数组，常用于提供多个带前缀的值，如果浏览器支持不带浏览器前缀的 flexbox，那么就只会渲染 display: flex

v-bind:style=”{display: [ ‘-webkit-box’, ‘-ms-flexbox’, ‘flex’ ]}”

1. **v-model**

用于在表单上创建双向数据绑定

<input v-model=”name”></input> data:{ name:’Tom’}

**v-model修饰符**

<1> .lazy (v-model.lazy)

默认情况下，v-model同步更新，使用lazy修饰符后，转变为在change事件后再同步

<2> .number (v-model.number)

将用户输入值转化为数值类型

<3> .trim (v-model.trim)

自动过滤用户输入的首尾空格

1. **v-on**

用于监听dom事件，然后执行某些操作，简写为@，v-on:click等价于@click。表达式可以是一个方法名

事件修饰符

1. .stop 防止事件冒泡，相当于js中的event.stopPropagation();
2. .capture 与事件冒泡方向相反，事件捕获由外到内
3. .once 只会触发一次
4. .self 只会触发自己范围内的事件，不包含子元素
5. .prevent 拦截默认事件，相当于js中的event.preventDefault()。例如可以阻止表单提交是页面刷新等浏览器默认行为
6. .passive 执行默认方法。一般用在滚动监听，@scoll,@touchmove。因为在滚动过程中，移动每个像素都会产生一次事件，每次都使用内核线程查询prevent会使滑动卡顿，，我们可以通过passive将内核线程查询跳过，可以大大提高滑动的流畅度。

###### 为啥项目中data需要使用return返回数据呢？

不使用return包裹的数据会在项目的全局可见，会造成变量污染；使用return包裹后数据中变量只在当前组件中生效，不会影响其他组件。

###### 声明式渲染

1. 采用模板语法声明式地将数据渲染到Dom系统

<div id=”app”>{{message}}</div>

<script>  
 var app = new Vue({  
 el: '#app', //el表示Vue实例挂载的元素节点，值可以是CSS选择符，或实际的HTML元素，或返回html元素的函数  
 data: {  
 message: 'Hello Vue!'  
 }  
 })  
</script>

1. 使用v-bind来绑定属性