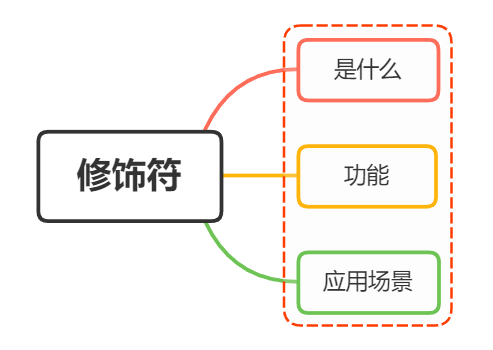
# 面试官：Vue常用的修饰符有哪些有什么应用场景



## 一、修饰符是什么

在程序世界里，修饰符是用于限定类型以及类型成员的声明的一种符号

在Vue中，修饰符处理了许多DOM事件的细节，让我们不再需要花大量的时间去处理这些烦恼的事情，而能有更多的精力专注于程序的逻辑处理

vue中修饰符分为以下五种：

* 表单修饰符
* 事件修饰符
* 鼠标按键修饰符
* 键值修饰符
* v-bind修饰符

## 二、修饰符的作用

### 表单修饰符

在我们填写表单的时候用得最多的是input标签，指令用得最多的是v-model

关于表单的修饰符有如下：

* lazy
* trim
* number

#### lazy

在我们填完信息，光标离开标签的时候，才会将值赋予给value，也就是在change事件之后再进行信息同步

<input type="text" v-model.lazy="value">  
<p>{{value}}</p>

#### trim

自动过滤用户输入的首空格字符，而中间的空格不会过滤

<input type="text" v-model.trim="value">

#### number

自动将用户的输入值转为数值类型，但如果这个值无法被parseFloat解析，则会返回原来的值

<input v-model.number="age" type="number">

### 事件修饰符

事件修饰符是对事件捕获以及目标进行了处理，有如下修饰符：

* stop
* prevent
* self
* once
* capture
* passive
* native

#### stop

阻止了事件冒泡，相当于调用了event.stopPropagation方法

<div @click="shout(2)">  
 <button @click.stop="shout(1)">ok</button>  
</div>  
//只输出1

#### prevent

阻止了事件的默认行为，相当于调用了event.preventDefault方法

<form v-on:submit.prevent="onSubmit"></form>

#### self

只当在 event.target 是当前元素自身时触发处理函数

<div v-on:click.self="doThat">...</div>

使用修饰符时，顺序很重要；相应的代码会以同样的顺序产生。因此，用 v-on:click.prevent.self 会阻止**所有的点击**，而 v-on:click.self.prevent 只会阻止对元素自身的点击

#### once

绑定了事件以后只能触发一次，第二次就不会触发

<button @click.once="shout(1)">ok</button>

#### capture

使事件触发从包含这个元素的顶层开始往下触发

<div @click.capture="shout(1)">  
 obj1  
<div @click.capture="shout(2)">  
 obj2  
<div @click="shout(3)">  
 obj3  
<div @click="shout(4)">  
 obj4  
</div>  
</div>  
</div>  
</div>  
// 输出结构: 1 2 4 3

#### passive

在移动端，当我们在监听元素滚动事件的时候，会一直触发onscroll事件会让我们的网页变卡，因此我们使用这个修饰符的时候，相当于给onscroll事件整了一个.lazy修饰符

<!-- 滚动事件的默认行为 (即滚动行为) 将会立即触发 -->  
<!-- 而不会等待 `onScroll` 完成 -->  
<!-- 这其中包含 `event.preventDefault()` 的情况 -->  
<div v-on:scroll.passive="onScroll">...</div>

不要把 .passive 和 .prevent 一起使用,因为 .prevent 将会被忽略，同时浏览器可能会向你展示一个警告。

passive 会告诉浏览器你不想阻止事件的默认行为

#### native

让组件变成像html内置标签那样监听根元素的原生事件，否则组件上使用 v-on 只会监听自定义事件

<my-component v-on:click.native="doSomething"></my-component>

使用.native修饰符来操作普通HTML标签是会令事件失效的

### 鼠标按钮修饰符

鼠标按钮修饰符针对的就是左键、右键、中键点击，有如下：

* left 左键点击
* right 右键点击
* middle 中键点击

<button @click.left="shout(1)">ok</button>  
<button @click.right="shout(1)">ok</button>  
<button @click.middle="shout(1)">ok</button>

### 键盘修饰符

键盘修饰符是用来修饰键盘事件（onkeyup，onkeydown）的，有如下：

keyCode存在很多，但vue为我们提供了别名，分为以下两种：

* 普通键（enter、tab、delete、space、esc、up...）
* 系统修饰键（ctrl、alt、meta、shift...）

// 只有按键为keyCode的时候才触发  
<input type="text" @keyup.keyCode="shout()">

还可以通过以下方式自定义一些全局的键盘码别名

Vue.config.keyCodes.f2 = 113

### v-bind修饰符

v-bind修饰符主要是为属性进行操作，用来分别有如下：

* async
* prop
* camel

#### async

能对props进行一个双向绑定

//父组件  
<comp :myMessage.sync="bar"></comp>   
//子组件  
this.$emit('update:myMessage',params);

以上这种方法相当于以下的简写

//父亲组件  
<comp :myMessage="bar" @update:myMessage="func"></comp>  
func(e){  
 this.bar = e;  
}  
//子组件js  
func2(){  
 this.$emit('update:myMessage',params);  
}

使用async需要注意以下两点：

* 使用sync的时候，子组件传递的事件名格式必须为update:value，其中value必须与子组件中props中声明的名称完全一致
* 注意带有 .sync 修饰符的 v-bind 不能和表达式一起使用
* 将 v-bind.sync 用在一个字面量的对象上，例如 v-bind.sync=”{ title: doc.title }”，是无法正常工作的

#### props

设置自定义标签属性，避免暴露数据，防止污染HTML结构

<input id="uid" title="title1" value="1" :index.prop="index">

#### camel

将命名变为驼峰命名法，如将view-Box属性名转换为 viewBox

<svg :viewBox="viewBox"></svg>

## 三、应用场景

根据每一个修饰符的功能，我们可以得到以下修饰符的应用场景：

* .stop：阻止事件冒泡
* .native：绑定原生事件
* .once：事件只执行一次
* .self ：将事件绑定在自身身上，相当于阻止事件冒泡
* .prevent：阻止默认事件
* .caption：用于事件捕获
* .once：只触发一次
* .keyCode：监听特定键盘按下
* .right：右键

## 参考文献

* https://segmentfault.com/a/1190000016786254
* https://vue3js.cn/docs/zh