常见的修饰符

修饰符是一个语法糖，能够保证方法只有纯粹的数据逻辑

✧ 事件修饰符

.stop
.prevent
.capture
.self
.once
.passive

```
<!-- 阻止单击事件继续传播 -->
<a v-on:click.stop="doThis"></a>

<!-- 提交事件不再重载页面 -->
<form v-on:submit.prevent="onSubmit"></form>

<!-- 修饰符可以串联 -->
<a v-on:click.stop.prevent="doThat"></a>

<!-- 只有修饰符 -->
<form v-on:submit.prevent></form>

<!-- 添加事件监听器时使用事件捕获模式 -->
<!-- 即内部元素触发的事件先在此处理，然后才交由内部元素进行处
理 -->
<div v-on:click.capture="doThis">...</div>

<!-- 只当在 event.target 是当前元素自身时触发处理函数 -->
<!-- 即事件不是从内部元素触发的 -->
<div v-on:click.self="doThat">...</div>
```

✧ 表单修饰符

.lazy

默认情况下，v-model 会在每次 input 事件后更新数据 (IME 拼字阶段的状态例外)。你可以添加 lazy 修饰符来改为在每次 change 事件后更新数据

```
<!-- 在 "change" 事件后同步更新而不是 "input" -->
<input v-model.lazy="msg" />
```

.number
如果你想让用户输入自动转换为数字，你可以在 v-model 后添加 .number 修饰符来管理输入。
注意：如果该值无法被 parseFloat() 处理，那么将返回原始值。
number 修饰符会在输入框有 type="number" 时自动启用。

```
<input v-model.number="age" />
```

.trim
如果你想要默认自动去除用户输入内容中两端的空格，你可以在 v-model 后添加 .trim 修饰符

```
<input v-model.trim="msg" />
```

✧ 按键修饰符
.enter
.tab
.delete (捕获“Delete”和“Backspace”两个按键)
.esc
.space
.up
.down
.left
.right
在监听键盘事件时，我们经常需要检查特定的按键。Vue 允许在 v-on 或 @ 监听按键事件时添加按键修饰符。

```
<!-- 仅在 `key` 为 `Enter` 时调用 `submit` -->
<input @keyup.enter="submit" />
```

你可以直接使用 KeyboardEvent.key 暴露的按键名称作为修饰符，但需要转为 kebab-case 形式。
在上面的例子中，仅会在 $event.key 为 'PageDown' 时调用事件处理。

```
<input @keyup.page-down="onPageDown" />
```

✧ .sync修饰符

在有些情况下，我们可能需要对一个 prop 进行“双向绑定”。不幸的是，真正的双向绑定会带来维护上的问题，因为子组件可以变更父组件，且在父组件和子组件两侧都没有明显的变更来源。

这也是为什么我们推荐以 update:myPropName 的模式触发事件取而代之。举个例子，在一个包含 title prop 的假设的组件中，我们可以用以下方法表达对其赋新值的意图：

```
<text-document v-bind:title.sync="doc.title"></text-document>
<!-- 等价于 -->
<text-document
  v-bind:title="doc.title"
  v-on:update:title="doc.title = $event"
></text-document>
```