事件的基本使用

v-on：XXX

1、使用 v-on:xxx 后者 @xxx 绑定事件，其中 xxx 就是事件名。

2、事件的回调需要配置在 `mehtods` 对象中 ,最终会在 `vm` 上显示。

3、`methods` 中配置的函数，不需要用箭头函数！否则this指向就不是 `vm` 而是 `window`。

4、`methods` 中配置的函数，都是被 Vue 所管理的函数，`this` 指向 是 vm 后者 组件实例对象。

5、`@click='show1'` 和 `@click='show1($event) '` 效果一致，但是后者可以传参。

```
    <div id="a">
        <button v-on:click="bt">按钮</button>
    </div>
    <script>
        const vm = new Vue({
            el:'#a',
            data:{
                name:'123'
            },
            methods: {
                bt(){
                    alert('弹窗')
                }
            }
        })
    </script>
```

`@click='show1($event) '` 演示

```
    <div id="a">
        <button v-on:click="bt">按钮</button>
    </div>
    <script>
        const vm = new Vue({
            el:'#a',
            data:{
                name:'123'
            },
            methods: {
                bt(event){
                    console.log(event.layerX);
                    //evnet为事件，layerX是事件中的一个属性
                }
            }
        })
    </script>
```

汇总代码

```
    <div id="root">
        <h1>欢迎来到{{name}}进行学习</h1>
        <!-- v-on的使用方法 -->
        <button v-on:click="show">点我提示信息</button>
        <!-- v-on简写形式  @xxx绑定事件,    $event 传参 -->
        <button @click="show2($event,66)">点我提示信息2</button>
    </div>

    <script>
        const vm =  new Vue({
            el:'#root',
            data:{
                name:'家里蹲'
            },
            methods: {
                show(){
                    alert('测试');
                },
                show2(event,number){
                    console.log(event,number);
                    alert('测试!!!!!');
                }
            },
        })
    </script>
```

事件修饰符

1. `prevent`：阻止默认事件(常用)

2. `stop`：阻止事件冒泡(常用)

3. `once`：事件只触发一次(常用)

//4,5,6为不常用事件修饰符

4. `capture`：使用事件的捕获方式

5. `self`：只有 `event.target` 是当前操作的元素才是触发事件

6. `passive`：事件的默认行为立即执行，无需等待时间回调执行完毕;

prevent：阻止默认事件(常用)

给事件后面添加`prevent`阻止默认事件

```
<body>
    <div id="a">
        <a href="http://www.baidu.com" @click.prevent="tc">链接</a>
    </div>
    <script>
        const vm = new Vue({
            el:'#a',
            methods: {
                tc(){
                    alert('会有弹窗，但是我们需要使用 @click.prevent 阻止超链接的跳转')
                }
            },
        })
    </script>
</body>
```

stop：阻止事件冒泡(常用)

阻止事件冒泡(常用)，给停止到的层级加`stop`

`CSS`

```
.demo1{
    border: 1px solid black;
}
```

`HTML`

```
<body>
    <div id="a">
        <div class="demo1" @click="tc">
            <button @click.stop="tc">冒泡按钮，父亲div也有点击事件</button>
        </div>
    </div>

    <script>
        const vm = new Vue({
            el:"#a",
            methods: {
                tc(){
                    alert('.stop阻止冒泡');
                }
            },
        })
    </script>
</body>
```

once：事件只触发一次(常用)

```
<div id="a">
    <button @click.once="tc">使用once，让事件只触发一次</button>
</div>

<script>
    const vm = new Vue({
        el:"#a",
        methods: {
            tc(){
                alert('once，让事件只触发一次');
            }
        },
    })
</script>
```

capture： 使用事件的捕获方式

`@click.capture` 中的 `capture`：使用事件的捕获方式，因为浏览器默认是冒泡事件，`capture`修饰符添加到事件捕获的对应父元素上

`css`

```
        .box1{
            background-color: skyblue;
            padding: 4px;
        }
        .box2{
            background-color: yellow;
            padding: 10px;
        }
```

`HTML`

```
<div id="a">
    <div class="box1" @click.capture="mes(1)">
        box1
        <div class="box2" @click="mes(2)">box2</div>
    </div>
</div>

<script>
    const vm = new Vue({
        el:"#a",
        methods: {
            mes(msg){
                console.log(msg);
            }
        },
```

```
        })
    </script>
```

self：只有 event.target 是当前操作的元素才是触发事件

`self`：只有 `event.target` 是当前操作的元素才是触发事件,类似像阻止冒泡事件

CSS

```
        .demo1{
            border: 1px solid black;
        }
```

HTML

```
<body>
    <div id="a">
        <div class="demo1" @click.self="selfT">
            <button @click="selfT">点我提示信息</button>
        </div>
    </div>

    <script>
        const vm = new Vue({
            el:"#a",
            methods: {
                selfT(e){
                    alert('弹窗');
                    console.log(e.target);
                },
            },
        })
    </script>
</body>
```

passive： 事件的默认行为立即执行，无需等待时间回调执行完毕

CSS

```
        .list{
            width: 200px;
            height: 200px;
            background-color: aqua;
            overflow: auto;
        }
        li{
            height: 100px;
        }
```

HTML

```
<div id="a">
    <!-- wheel js事件，鼠标滚轮滚动一次触发一次 -->
    <ul  class="list" @wheel.passive="demo">
        <li>1</li>
        <li>2</li>
        <li>3</li>
    </ul>
</div>

<script>
    const vm = new Vue({
        el:"#a",
        methods: {
            demo(){
                for(let i = 0;i<100000;i++){
                    console.log('#');
                }
                console.log('十万次执行完毕');
            }
        },
    })
</script>
```

修饰符可以连续写

`@click.prevent.stop="show"`

```
<div id="a">
    <div class="class" @click="show">
        <!-- 阻止超链接的跳转并且阻止事件冒泡 -->
        <a href="http://www.baidu.com" @click.prevent.stop="show">百度链接</a>
    </div>
</div>
<script>
    const vm = new Vue({
        el:'#a',
        methods: {
            show(){
                alert('会有弹窗，但是我们需要使用 @click.prevent 阻止超链接的跳转');
            },
        },
    })
</script>
```

键盘事件

Vue 中常用的按键别名

功能	按键别名
enter	回车
delete (捕获'删除'和'退格'键)	删除
esc	退出
space	空格
tab (tab在浏览器中，会切换到下一个元素，所以 keyup不适合用， 建议搭配keydown使用)	换行
up	上
down	下
left	左
right	右

enter 回车

```
<div id="a">
        <!-- keyup  js事件，鼠标按下 -->
        <input type="text" placeholder="按下回车提示输入" @keyup.enter="showInfo">
    </div>
    <script>
        const vm = new Vue({
            el:'#a',
            methods: {
                showInfo(e){
                    console.log('vue中的enter使用');
                }
            },
        })
    </script>
```

Vue未提供别名的按键 (不常用)

Vue未提供别名的按键，可以使用按键原始的`key`值去绑定，但要注意转为`kebab-case`(短横线命名)，不常用

```
    <div id="a">
        <!-- keyup  js事件，鼠标按下 -->
        <input type="text" placeholder="按下回车提示输入" @keyup.caps-
lock="showInfo">
    </div>
```

```
    <script>
        const vm = new Vue({
            el:'#a',
            methods: {
                showInfo(e){
                    // e.key 按键名字      e.keyCode键码值
                    alert(e.key+"-"+e.keyCode);
                }
            },
        })
    </script>
```

系统修饰键(用法特殊)

系统修饰键(用法特殊)：`ctrl`、`alt`、`shift`、`meta(win/commod)`

(1)需要配合 `keyup` 使用：按下修饰键的同时，再按下 其他键，随后释放其他键，事件才被触发

(2)配合`keydown`使用：正常触发事件

```
    <div id="a">
        <!-- keydown  js事件 -->
        <input type="text" placeholder="按下回车提示输入" @keydown.tab="showInfo">
    </div>

    <script>
        const vm = new Vue({
            el:'#a',
            methods: {
                showInfo(e){
                    console.log(e.key);
                }
            },
        })
    </script>
```

也可以使用keyCode 键码值 去指定具体的按键(不推荐)

Vue.config.keyCodes.自定义键名 = 键码（不推荐）

可以去定制按键名。(不太推荐，基本默认的一般都够用了)

```
    <div id="a">
        <!-- keydown  js事件 -->
        <input type="text" placeholder="按下回车提示输入"
@keydown.huiche="showInfo">
    </div>
```

```
<script>
    Vue.config.keyCodes.huiche = 13;      //定义了一个按键别名
    const vm = new Vue({
        el:'#a',
        methods: {
            showInfo(e){
                console.log(e.key);
            }
        },
    })
</script>
```