

Responding to User Events in Vue.js



Chad Campbell

INDEPENDENT SOFTWARE CONSULTANT

@chadcampbell <https://www.ecofic.com>



Overview



Listen for user-initiated events

Modify events



Using Event Handlers



Attaching to Events

v-on Directive

Listens for events in the HTML DOM



```
<button v-on:click="executeSearch">Search</button>
```

Attach to an Event

Using the `v-on` directive



Ignore the “on” prefix that you might’ve used in HTML



```
var growler = new Vue({  
  el: '#growler',  
  data: {  
    query: ''  
  },  
  methods: {  
    executeSearch:function() {  
      alert(this.query);  
    }  
  }  
})
```



In a method, the **this** keyword is automatically bound to the Vue instance

```
<div id="growler">  
  <form>  
    <input v-model="query" type="search"  
      placeholder="search...">  
    <button type="button"  
      v-on:click="executeSearch">  
      Search  
    </button>  
  </form>  
</div>
```



```
var growler = new Vue({
  el: '#growler',
  data: {
    query: ''
  },
  methods: {
    executeSearch:function(event) {
      alert('Query: "' + this.query +
        '" Button:"' + event.target.innerText +
        '"');
    }
  }
});
```



The last parameter in an event handler is the Event object that triggered the event.

```
<div id="growler">
  <form>
    <input v-model="query" type="search"
      placeholder="search...">
    <button type="button"
      v-on:click="executeSearch">
      Search
    </button>
  </form>
</div>
```



```
var growler = new Vue({
  el: '#growler',
  data: {
    query: ''
  },
  methods: {
    executeSearch: function(t, e) {
      var msg = 'Token: ' + t +
        ' Query: ' + this.query +
        ' Button: ' +
          event.target.innerText;
      alert(msg);
    }
  }
});
```

```
<div id="growler">
  <form>
    <input v-model="query" type="search"
      placeholder="search...">
    <button type="button"
      v-on:click="executeSearch('token',
$event)">
      Search
    </button>
  </form>
</div>
```



<https://github.com/ecofic/course-vue-getting-started>



\$event
Parameter

Reserved variable

Access to the HTML DOM Event



```
<button @click="executeSearch">Search</button>
```

v-on Shorthand Syntax

Uses the “@”



Altering Events



Altering Events

Explain event propagation

React to keyboard and mouse events

Consider special keys



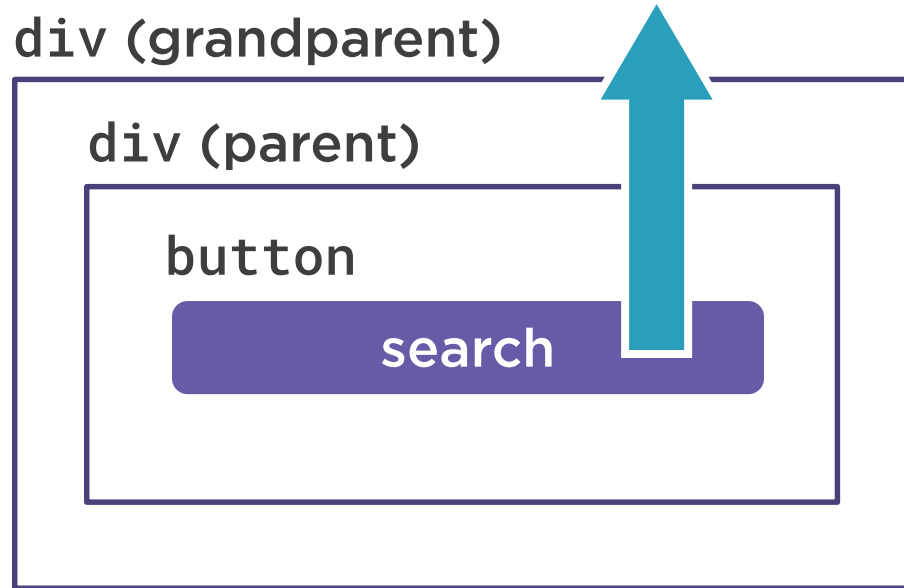
```
<div v-on:click="buttonGrandparentClick">  
  <div v-on:click="buttonParentClick">  
    <button v-on:click="executeSearch" type="button">Search</button>  
  </div>  
</div>
```

Understanding Event Propagation

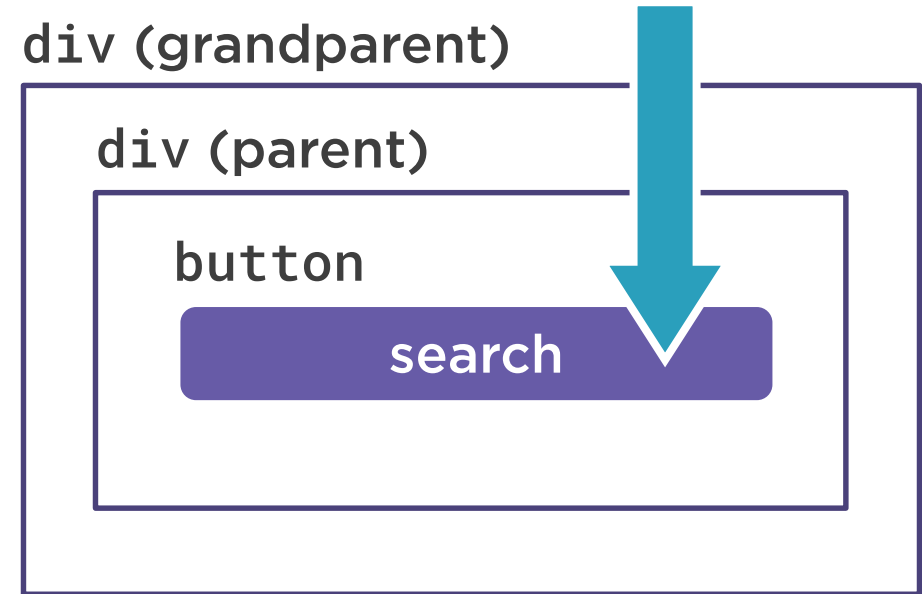
Sample snippet for event propagation



Event Propagation



Event Bubbling



Event Capturing



Bubble up, Capture down



Event Capturing

Introduced in the early days of the web
Supported by the W3C specification
Applied via capture modifier



```
<div v-on:click.capture="grandparentClick" ❶  
  <div v-on:click.capture="parentClick" ❷  
    <button v-on:click.capture="executeSearch" ❸  
      type="button">Search</button>  
  </div>  
</div>
```



Using the capture Modifier

Propagating events from top-to-bottom




```
<div v-on:click.capture="grandparentClick" ❶  
  <div v-on:click="parentClick" ❸  
    <button v-on:click="executeSearch" ❷  
      type="button">Search</button>  
  </div>  
</div>
```

Mixing Capturing and Bubbling

Events propagate in two directions



Avoid using event capturing



prevent
Modifier

Bypasses default event behavior

**Representation of JavaScript's
preventDefault method**



```
<form action="#" method="GET">  
  <input v-model="query" id="query" name="query" />  
  <button type="submit"  
    v-on:click="executeSearch">Search</button>  
</form>
```

Submitting a Search



```
var growler = new Vue({
  el: '#growler',
  data: {
    query: ''
  },
  methods: {
    executeSearch: function() {
      if (this.query) {
        document.forms[0].submit();
      } else {
        alert('Please enter a query');
      }
    }
  }
});
```

```
<div id="growler">
  <form action="#" method="GET">
    <input v-model="query" id="query"
name="query" />
    <button type="submit"
      v-on:click.prevent="executeSearch">
      Search
    </button>
  </form>
</div>
```



stop Modifier

Ceases event propagation

Hook to JavaScript's stopPropagation method



```
<div v-on:click="grandparentClick">  
  <div v-on:click="parentClick">  
    <button v-on:click.stop="executeSearch"  
      type="button">Search</button>  
  </div>  
</div>
```

Using the stop Modifier

Ceasing event propagation



```
<div v-on:click="grandparentClick">  
  <div v-on:click.stop="parentClick">  
    <button v-on:click="executeSearch"  
      type="button">Search</button>  
  </div>  
</div>
```

Using the stop Modifier

Ceasing event propagation at an ancestor



Stopping Propagation



Less code gets executed



More predictable code

self Modifier

Triggers event if element is originator
Only concerned with the target



```
<div v-on:click="grandparentClick">  
  <div v-on:click="parentClick">  
    <button v-on:click.self="executeSearch"  
      type="button">Search</button>  
  </div>  
</div>
```

Using the self Modifier

Ignoring event propagation for simplification



```
<div v-on:click="grandparentClick">  
  <div v-on:click.self="parentClick">  
    <button v-on:click="executeSearch"  
      type="button">Search</button>  
  </div>  
</div>
```

Using the self Modifier

Ignoring event propagation for simplification



once Modifier

Run an event one, and only one, time

Removes the event handler after the event is fired



```
var growler = new Vue({
  el: '#growler',
  data: {
    query: '',
    isRunning: false
  },
  methods: {
    executeSearch:function() {
      this.isRunning = true;
      document.forms[0].submit();
    }
  }
});
```

```
<div id="growler">
  <form action="someUrl" method="POST">
    <input type="search"
      v-model="query"
      v-bind:disabled="isRunning" />
    <button type="button"
      v-on:click.once="executeSearch"
      v-bind:disabled="isRunning">
      Search
    </button>
  </form>
</div>
```



Note

The once modifier only detaches from the event handler of the containing element.























```
var growler = new Vue({
  el: '#growler',
  data: {
    query: '',
  },
  methods: {
    executeSearch: function() {
      alert('Search Clicked');
    },
    parentClick: function() {
      alert('Parent Clicked');
    }
  }
});
```

```
<div id="growler">
  <form action="someUrl" method="POST">
    <input type="search"
      v-model="query"
      v-bind:disabled="isRunning" />
    <div v-on:click="parentClick">
      <button type="button"
        v-on:click.once="executeSearch"
        v-bind:disabled="isRunning">
        Search
      </button>
    </div>
  </form>
</div>
```



Modifier Impact

	capture	prevent	stop	self	once
Passes Event to Ancestors					
Passes Event to Descendents					
One Time Use?					
Triggered by Lineage					



You can chain
modifiers together




```
<div v-on:click="grandparentClick">  
  <div v-on:click.stop.self="parentClick">  
    <button v-on:click="executeSearch">Search</button>  
  </div>  
</div>
```

Bypassing Event Propagation

Chaining modifiers together to prevent event propagation



Reacting to Keyed Events



```
var growler = new Vue({
  el: '#growler',
  data: { ... },
  methods: { ... }
});

function checkForEnter(e) {
  var c = e.keyCode ? e.keyCode:e.which;
  if (c == 13) {
    growler.executeSearch();
    return false;
  }
  return true;
}
```

```
<div id="growler">
  <form onkeypress="return
    event.keyCode != 13;">
    <input type="search"
      v-model="query" />
    <button type="button"
      v-on:click.once="executeSearch">
      Search</button>
    </form>
</div>
```



Common Key Events

Key	Key Code	Modifier
Enter	13	enter
Tab	9	tab
Delete	8	delete
Escape	18	esc
Space	32	space
Up	38	up
Down	40	down
Left	37	left
Right	39	right



```
var growler = new Vue({
  el: '#growler',
  data: { ... },
  methods: { ... }
});

function checkForEnter(e) {
  var c = e.keyCode ? e.keyCode:e.which;
  if (c == 13) {
    growler.executeSearch();
    return false;
  }
  return true;
}
```

```
<div id="growler">
  <form onkeypress="return
    event.keyCode != 13;">
    <input type="search"
      v-model="query" />
    <button type="button"
      v-on:click.once="executeSearch">
      Search</button>
    </form>
</div>
```



```
var growler = new Vue({  
  el: '#growler',  
  data: { ... },  
  methods: { ... }  
});
```

```
<form v-on:submit.prevent  
  action="..." method="GET">  
  <input type="search"  
    v-model="query"  
    v-on:keyup="evaluateKey"  
    v-on:keyup.enter="executeSearch" />  
  <button type="button"  
    v-on:click.once="executeSearch">  
    Search  
  </button>  
</form>
```



Creating Key Modifiers

Used for key codes used less often

Useful for shortcut keys



```
Vue.config.keyCodes = {  
  f1: 112  
};
```

Defining Custom Key Modifiers

Defining keyCodes in the Vue config




```
<div id="growler" v-on:keydown.f1="openInfo">  
  ...  
</div>
```

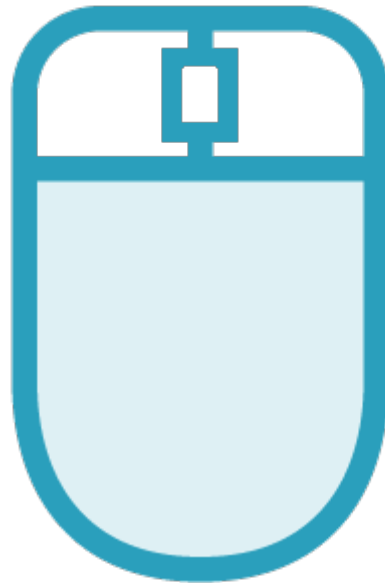
Using Custom Key Modifiers

Using the f1 modifier



Reacting to Mouse Buttons

Button	Modifier
Left	<code>left</code>
Middle	<code>middle</code>
Right	<code>right</code>



Left modifier

Empowers you to respond to a user interacting with the left mouse button



```
<div v-on:mousedown.left="onBlockClick">
```

Click in this area with the left mouse button.

```
</div>
```

The left Modifier

Considering a left mouse button down



```
<div v-on:click.left="onBlockClick">
```

Click in this area with the left mouse button.

```
</div>
```

The left modifier

Considering a left mouse button click



```
<div v-on:click="onBlockClick">
```

Click in this area with the left mouse button.



```
</div>
```

The left modifier

Considering a left mouse button click



Event Support of Left Modifier

	Supported?
<code>onclick</code>	
<code>oncontextmenu</code>	
<code>ondblclick</code>	
<code>onmousedown</code>	
<code>onmouseenter</code>	
<code>onmouseleave</code>	
<code>onmousemove</code>	
<code>onmouseover</code>	
<code>onmouseout</code>	
<code>onmouseup</code>	



Middle modifier






Helps you respond to a user interacting with the middle mouse button



Only works with the `onmousedown` and `onmouseup` events.



Event Support of Middle Modifier

	Supported?
<code>onclick</code>	
<code>oncontextmenu</code>	
<code>ondblclick</code>	
<code>onmousedown</code>	
<code>onmouseenter</code>	
<code>onmouseleave</code>	
<code>onmousemove</code>	
<code>onmouseover</code>	
<code>onmouseout</code>	
<code>onmouseup</code>	



```
<div v-on:mousedown.middle="onBlockClick">
```

Click in this area with the middle mouse button.

```
</div>
```

The `middle` modifier

Considering a middle mouse button click



Right modifier

Helps you respond to a user interacting with the right mouse button



```
<div v-on:mousedown.right="onBlockClick">
```

Click in this area with the right mouse button.

```
</div>
```

The right modifier

Considering a right mouse button click



```
var growler = new Vue({
  el: '#growler',
  data: {
    showContextMenu: false,
    top: '0px',
    left: '0px'
  },
  methods: {
    ...
  }
});
```

```
<div id="growler">
  <div v-on:
    contextmenu.prevent="onBlockClick">
    Click in this area with the right
mouse button.
  </div>

  <ul id="myContextMenu"
    v-if="showContextMenu"
    v-on:blur="closeContextMenu"
    v-bind:style="{top:top, left:left}">
    <li><a href="#"
      v-on:click="onCopyClick">
      Copy
    </a></li>
    <li><a href="#"
      v-on:click="onPasteClick">
      Paste
    </a></li>
  </ul>
</div>
```



```
var growler = new Vue({
  el: '#growler',
  data: {...},
  methods: {
    onBlockClick: function(e) {
      if (e.button === 2) {
        this.showContextMenu = true;
        this.top = e.y + 'px';
        this.left = e.x + 'px';
      }
    },
    ...
  });
```

```
<div id="growler">
  <div v-on:
    contextmenu.prevent="onBlockClick">
    Click in this area with the right
mouse button.
  </div>

  <ul id="myContextMenu"
    v-if="showContextMenu"
    v-on:blur="closeContextMenu"
    v-bind:style="{top:top, left:left}">
    <li><a href="#"
      v-on:click="onCopyClick">
      Copy
    </a></li>
    <li><a href="#"
      v-on:click="onPasteClick">
      Paste
    </a></li>
  </ul>
</div>
```

































```
var growler = new Vue({
  ...
  methods: {
    ...
    closeContextMenu: function() {
      this.showContextMenu = false;
    },
    onCopyClick: function() {
      alert('copy something');
      this.closeContextMenu();
    },
    onPasteClick: function() {
      alert('paste something');
      this.closeContextMenu();
    }
  }
});
```

```
<div id="growler">
  <div v-on:
    contextmenu.prevent="onBlockClick">
    Click in this area with the right
mouse button.
  </div>

  <ul id="myContextMenu"
    v-if="showContextMenu"
    v-on:blur="closeContextMenu"
    v-bind:style="{top:top, left:left}">
    <li><a href="#"
      v-on:click="onCopyClick">
      Copy
    </a></li>
    <li><a href="#"
      v-on:click="onPasteClick">
      Paste
    </a></li>
  </ul>
</div>
```



Mouse Event Modifications

	left	middle	right
<code>onclick</code>			
<code>oncontextmenu</code>			
<code>ondblclick</code>			
<code>onmousedown</code>			
<code>onmouseenter</code>			
<code>onmouseleave</code>			
<code>onmousemove</code>			
<code>onmouseover</code>			
<code>onmouseout</code>			
<code>onmouseup</code>			



**Add keyboard
shortcuts**

Reacting to Special Keys



Special Key Modifiers

Special Key	Key Code	Modifier
Alt	18	alt
Ctrl	17	ctrl
Meta	91	meta
Shift	16	shift



The Meta key is the “Windows” key or Command key



```
<input type="search"  
  v-model.trim="query"  
  v-on:keypress.enter.prevent="executeNewSearch"  
  v-on:keyup.ctrl.enter="executeSearchInNewWindow"  
  placeholder="please enter your query here." />
```

Keyboard Shortcut

Using modifier key modifiers



<https://github.com/ecofic/course-vue-getting-started>



Modifiers Help You

Deliver powerful features in your apps

Deliver a high-performing app

Enforce a separation between the UI and the logic



Summary



Using event handlers

Alter events with modifiers

