Vue.js Certification CYBER MONDAY Get Official Certification for 60% OFF Get Certified ENDS IN 06h:35m

Component Events

> This page assumes you've already read the Components Basics. Read that first if you are new to components.

Emitting and Listening to Events

A component can emit custom events directly in template expressions (e.g. in a `v-on` handler) using the built-in `$emit` method:

template
```
<!-- MyComponent -->
<button @click="$emit('someEvent')">Click Me</button>
```

The parent can then listen to it using `v-on`:

template
```
<MyComponent @some-event="callback" />
```

The `.once` modifier is also supported on component event listeners:

template
```
<MyComponent @some-event.once="callback" />
```

Like components and props, event names provide an automatic case transformation. Notice we emitted a camelCase event, but can listen for it using a kebab-cased listener in the parent. As with props casing, we recommend using kebab-cased event listeners in templates.

> ⓘ TIP
>
> Unlike native DOM events, component emitted events do **not** bubble. You can only listen to the events emitted by a direct child component. If there is a need to communicate between sibling or deeply nested components, use an external event bus or a global state management solution.

Vue.js Certification CYBER MONDAY Get Official Certification for 60% OFF Get Certified ENDS IN 06h:35m

Event Arguments

It's sometimes useful to emit a specific value with an event. For example, we may want the `<BlogPost>` component to be in charge of how much to enlarge the text by. In those cases, we can pass extra arguments to `$emit` to provide this value:

```template
<button @click="$emit('increaseBy', 1)">
  Increase by 1
</button>
```

Then, when we listen to the event in the parent, we can use an inline arrow function as the listener, which allows us to access the event argument:

```template
<MyButton @increase-by="(n) => count += n" />
```

Or, if the event handler is a method:

```template
<MyButton @increase-by="increaseCount" />
```

Then the value will be passed as the first parameter of that method:

```js
function increaseCount(n) {
  count.value += n
}
```

> ⓘ TIP
>
> All extra arguments passed to `$emit()` after the event name will be forwarded to the listener. For example, with `$emit('foo', 1, 2, 3)` the listener function will receive three arguments.

Declaring Emitted Events

A component can explicitly declare the events it will emit using the `defineEmits()` macro:

```vue
<script setup>
defineEmits(['inFocus', 'submit'])
</script>
```

Vue.js Certification CYBER MONDAY Get Official Certification for 60% OFF Get Certified ENDS IN 06h:35m

function that we can use instead:

```vue
<script setup>
const emit = defineEmits(['inFocus', 'submit'])

function buttonClick() {
  emit('submit')
}
</script>
```

The `defineEmits()` macro **cannot** be used inside a function, it must be placed directly within `<script setup>`, as in the example above.

If you're using an explicit `setup` function instead of `<script setup>`, events should be declared using the `emits` option, and the `emit` function is exposed on the `setup()` context:

```js
export default {
  emits: ['inFocus', 'submit'],
  setup(props, ctx) {
    ctx.emit('submit')
  }
}
```

As with other properties of the `setup()` context, `emit` can safely be destructured:

```js
export default {
  emits: ['inFocus', 'submit'],
  setup(props, { emit }) {
    emit('submit')
  }
}
```

The `emits` option and `defineEmits()` macro also support an object syntax. If using TypeScript you can type arguments, which allows us to perform runtime validation of the payload of the emitted events:

```vue
<script setup lang="ts">
const emit = defineEmits({
  submit(payload: { email: string, password: string }) {
    // return `true` or `false` to indicate
    // validation pass / fail
  }
})
</script>
```

Vue.js Certification CYBER MONDAY Get Official Certification for 60% OFF Get Certified ENDS IN 06h:35m

```vue
<script setup lang="ts">
const emit = defineEmits<{
  (e: 'change', id: number): void
  (e: 'update', value: string): void
}>()
</script>
```

More details: Typing Component Emits TS

Although optional, it is recommended to define all emitted events in order to better document how a component should work. It also allows Vue to exclude known listeners from fallthrough attributes, avoiding edge cases caused by DOM events manually dispatched by 3rd party code.

> ⓘ TIP
>
> If a native event (e.g., `click`) is defined in the `emits` option, the listener will now only listen to component-emitted `click` events and no longer respond to native `click` events.

Events Validation

Similar to prop type validation, an emitted event can be validated if it is defined with the object syntax instead of the array syntax.

To add validation, the event is assigned a function that receives the arguments passed to the `emit` call and returns a boolean to indicate whether the event is valid or not.

```vue
<script setup>
const emit = defineEmits({
  // No validation
  click: null,

  // Validate submit event
  submit: ({ email, password }) => {
    if (email && password) {
      return true
    } else {
      console.warn('Invalid submit event payload!')
      return false
    }
  }
})
```

Vue.js Certification CYBER MONDAY Get Official Certification for 60% OFF Get Certified ENDS IN 06h:35m

```
}
</script>
```

Edit this page on GitHub

Previous
Props

Next
Component v-model