Fallthrough Attributes

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

Attribute Inheritance

A "fallthrough attribute" is an attribute or `v-on` event listener that is passed to a component, but is not explicitly declared in the receiving component's props or emits. Common examples of this include `class`, `style`, and `id` attributes.

When a component renders a single root element, fallthrough attributes will be automatically added to the root element's attributes. For example, given a `<MyButton>` component with the following template:

```template
<!-- template of <MyButton> -->
<button>Click Me</button>
```

And a parent using this component with:

```template
<MyButton class="large" />
```

The final rendered DOM would be:

```html
<button class="large">Click Me</button>
```

Here, `<MyButton>` did not declare `class` as an accepted prop. Therefore, `class` is treated as a fallthrough attribute and automatically added to `<MyButton>`'s root element.

`class` and `style` Merging

we change the template of `<MyButton>` in the previous example to:

```template
<!-- template of <MyButton> -->
<button class="btn">Click Me</button>
```

Then the final rendered DOM would now become:

```html
<button class="btn large">Click Me</button>
```

`v-on` Listener Inheritance

The same rule applies to `v-on` event listeners:

```template
<MyButton @click="onClick" />
```

The `click` listener will be added to the root element of `<MyButton>`, i.e. the native `<button>` element. When the native `<button>` is clicked, it will trigger the `onClick` method of the parent component. If the native `<button>` already has a `click` listener bound with `v-on`, then both listeners will trigger.

Nested Component Inheritance

If a component renders another component as its root node, for example, we refactored `<MyButton>` to render a `<BaseButton>` as its root:

```template
<!-- template of <MyButton/> that simply renders another component -->
<BaseButton />
```

Then the fallthrough attributes received by `<MyButton>` will be automatically forwarded to `<BaseButton>`.

Note that:

1. Forwarded attributes do not include any attributes that are declared as props, or `v-on` listeners of declared events by `<MyButton>` - in other words, the declared props and listeners have been "consumed" by `<MyButton>`.

2. Forwarded attributes may be accepted as props by `<BaseButton>`, if declared by it.

Disabling Attribute Inheritance

If you do **not** want a component to automatically inherit attributes, you can set `inheritAttrs: false` in the component's options.

Since 3.3 you can also use `defineOptions` directly in `<script setup>`:

```vue
<script setup>
defineOptions({
  inheritAttrs: false
})
// ...setup logic
</script>
```

The common scenario for disabling attribute inheritance is when attributes need to be applied to other elements besides the root node. By setting the `inheritAttrs` option to `false`, you can take full control over where the fallthrough attributes should be applied.

These fallthrough attributes can be accessed directly in template expressions as `$attrs`:

```template
<span>Fallthrough attributes: {{ $attrs }}</span>
```

The `$attrs` object includes all attributes that are not declared by the component's `props` or `emits` options (e.g., `class`, `style`, `v-on` listeners, etc.).

Some notes:

- Unlike props, fallthrough attributes preserve their original casing in JavaScript, so an attribute like `foo-bar` needs to be accessed as `$attrs['foo-bar']`.
- A `v-on` event listener like `@click` will be exposed on the object as a function under `$attrs.onClick`.

Using our `<MyButton>` component example from the previous section - sometimes we may need to wrap the actual `<button>` element with an extra `<div>` for styling purposes:

```template
<div class="btn-wrapper">
  <button class="btn">Click Me</button>
</div>
```

We want all fallthrough attributes like `class` and `v-on` listeners to be applied to the inner `<button>`, not the outer `<div>`. We can achieve this with `inheritAttrs: false` and `v-bind="$attrs"`:

```
</div>
```

Remember that `v-bind` without an argument binds all the properties of an object as attributes of the target element.

Attribute Inheritance on Multiple Root Nodes

Unlike components with a single root node, components with multiple root nodes do not have an automatic attribute fallthrough behavior. If `$attrs` are not bound explicitly, a runtime warning will be issued.

```template
<CustomLayout id="custom-layout" @click="changeValue" />
```

If `<CustomLayout>` has the following multi-root template, there will be a warning because Vue cannot be sure where to apply the fallthrough attributes:

```template
<header>...</header>
<main>...</main>
<footer>...</footer>
```

The warning will be suppressed if `$attrs` is explicitly bound:

```template
<header>...</header>
<main v-bind="$attrs">...</main>
<footer>...</footer>
```

Accessing Fallthrough Attributes in JavaScript

If needed, you can access a component's fallthrough attributes in `<script setup>` using the `useAttrs()` API:

```vue
<script setup>
import { useAttrs } from 'vue'

const attrs = useAttrs()
</script>
```

```js
export default {
  setup(props, ctx) {
    // fallthrough attributes are exposed as ctx.attrs
    console.log(ctx.attrs)
  }
}
```

Note that although the `attrs` object here always reflects the latest fallthrough attributes, it isn't reactive (for performance reasons). You cannot use watchers to observe its changes. If you need reactivity, use a prop. Alternatively, you can use `onUpdated()` to perform side effects with the latest `attrs` on each update.

Edit this page on GitHub

Previous
Component v-model

Next
Slots