How to Implement a Mouseover or Hover in Vue

In CSS it's pretty easy to change things on `hover`. We just use the `:hover` psuedo-class:

```
.item {
  background: blue;
}

.item:hover {
  background: green;
}
```

In Vue it gets a little trickier, because we don't have this functionality built in.

We have to implement most of this ourselves.

But don't worry, it's not **that** much work.

In this short chapter you'll learn:

- How to implement a **hover effect** in Vue
- How to **show an element** on mouseover
- How to dynamically **update classes** with a mouseover
- How to do this even on **custom Vue components**

Listening to the right events

So, which events do we need to listen to?

We want to know when the mouse is hovering over the element. This can be figured out by keeping track of when the mouse **enters** the element, and when the mouse **leaves** the element.

To keep track of when the mouse leaves, we'll use the mouseleave event.

Detecting when the mouse enters can be done with the corresponding mouseenter event, but we won't be using that one.

The reason is that there can be significant performance problems when using `mouseenter` on deep DOM trees. This is because `mouseenter` fires a unique event to the entered element, as well as every single ancestor element.

What will we be using instead, you ask?!?

Instead, we will use the mouseover event.

The `mouseover` event works pretty much the same as `mouseenter`. The main difference being that `mouseover` bubbles like most other DOM events.

Instead of creating a ton of unique events, there is only one — making it much faster!

Let's hook things up

To hook everything up we will use Vue events to listen for when the mouse enters and leaves, and update our state accordingly.

We will also be using the shorthand of `v-on`.

Instead of writing `v-on:event`, we can just write `@event`.

Here's how we hook it up:

```
<template>
  <div
    @mouseover="hover = true"
    @mouseleave="hover = false"
  />
</template>
```

```
export default {
  data() {
    return {
      hover: false,
    };
  }
}
```

Now the reactive property `hover` will always reflect if the mouse is hovering over the element or not!

Showing an element when hovering

If you wanted to show an element based on the hover state, you can pair this with a v-if directive:

```
<template>
  <div>
    <span
      @mouseover="hover = true"
      @mouseleave="hover = false"
    >
      Hover me to show the message!
    </span>
    <span v-if="hover">This is a secret message.</span>
  </div>
</template>
```

```
export default {
  data() {
    return {
      hover: false,
    };
  }
}
```

Whenever you put your mouse over `Hover me to show the message!`, the secret message will appear!

Toggling classes when hovering

You can also do a similar thing to toggle classes:

```
<template>
  <div>
    <span
      @mouseover="hover = true"
      @mouseleave="hover = false"
      :class="{ active: hover }"
    >
      Hover me to change the background!
    </span>
  </div>
</template>
```

```
export default {
  data() {
    return {
      hover: false,
    };
  }
}
```

```
.active {
  background: green;
}
```

Although that works, it's not the best solution.

For something like this it's better to just use CSS:

```
<template>
  <span>
    Hover me to change the background!
  </span>
</template>
```

```
span:hover {
  background: green;
}
```

As you can see, even though we can do it using Vue, we don't need it to achieve this effect.

Hovering over a Vue component

If you have a Vue component that you'd like to implement this behaviour with, you'll have to make a slight modification.

You can't listen to the `mouseover` and `mouseleave` events like we were doing before.

If your Vue component doesn't emit those events, then we can't listen to them.

Instead, we can add the `.native` event modifier to listen to DOM events directly on our custom Vue component:

```
<template>
  <my-custom-component
    @mouseover.native="hover = true"
    @mouseleave.native="hover = false"
  />
</template>
```

```
export default {
  data() {
    return {
      hover: false,
    };
  }
}
```

Using `.native`, we listen for native DOM events instead of the ones that are emitted from the Vue component.

If this part is a little confusing, check out the docs. They do a really great job of explaining how this works.

Conclusion

There you have it!

Achieving a mouseover effect in Vue JS requires only a little bit of code.

Now you can go and do all sorts of things when someone hovers over your Vue app.