**Prop / Event Fallthrough & Binding All Props**

There are two advanced concepts you also should have heard about:

* **Prop Fallthrough**
* **Binding All Props on a Component**

Prop Fallthrough

You can set props (and listen to events) on a component which you haven't registered inside of that component.

For example:

**BaseButton.vue**

1. <template>
2. <button>
3. <slot></slot>
4. </button>
5. </template>
7. <script>export default {}</script>

This button component (which might exist to set up a button with some default styling) **has** **no special props** that would be registered.

Yet, you can use it like this:

1. <base-button type="submit" @click="doSomething">Click me</base-button>

Neither the type prop nor a custom click event are defined or used in the BaseButton component.

**Yet, this code would work.**

Because Vue has built-in support for **prop (and event) "fallthrough"**.

Props and events added on a custom component tag **automatically fall through** to the **root component** in the template of that component. In the above example, type and @click get added to the <button> in the BaseButton component.

You can get access to these fallthrough props on a built-in $attrs property (e.g. this.$attrs).

This can be handy to build "utility" or pure presentational components where you don't want to define all props and events individually.

**You'll see this in action** the component course project ("Learning Resources App") later.

You can learn more about this behavior here: <https://v3.vuejs.org/guide/component-attrs.html>

Binding all Props

There is another built-in feature/ behavior related to props.

If you have this component:

**UserData.vue**

1. <template>
2. <h2>{‌{ firstname }} {‌{ lastname }}</h2>
3. </template>
5. <script>
6. export default {
7. props: ['firstname', 'lastname']
8. }
9. </script>

You **could** use it like this:

1. <template>
2. <user-data :firstname="person.firstname" :lastname="person.lastname"></user-data>
3. </template>
5. <script>
6. export default {
7. data() {
8. return {
9. person: { firstname: 'Max', lastname: 'Schwarz' }
10. };
11. }
12. }
13. </script>

But if you have an object which holds the props you want to set as properties, you can also **shorten the code a bit**:

1. <template>
2. <user-data v-bind="person"></user-data>
3. </template>
5. <script>
6. export default {
7. data() {
8. return {
9. person: { firstname: 'Max', lastname: 'Schwarz' }
10. };
11. }
12. }
13. </script>

With v-bind="person" you **pass all key-value pairs** inside of person **as props** to the component. That of course requires person to be a **JavaScript object**.

This is **purely optional** but it's a little convenience feature that could be helpful.

Resources for this lecture

* adv-props.pdf