Module 4-1

VUE Nested Components

Working With Multiple Components

- So far we have dealt with a relatively simple project setup, with a single component doing all the hard work, then importing that component into the App.vue parent component.
- The whole idea of component based JS development is that we can take several components and integrate them together under a parent component.
  - This requires us to learn some additional techniques as individual components are not aware of each other.
- Namely, we will be covering how to transmit data from a parent component to a child component and vice versa.

Properties

- Properties can be thought of as characteristics of the component.

- We will be using properties to transmit data **from a parent component to a child component**.

Properties Defining

- Within a child component, properties are defined in JSON format within the script section.

- It is a peer of the other sections we've seen so far: data(), computed, methods.

```
<script>
export default {
    name: "product-review",
    props() {
      // your props go here
    },
    data() {
      ...
    },
    computed: {
      ...
    },
    methods: {
      ...
    }
}
</script>
```

Properties Defining

- Within a child component, properties are defined in JSON format within the script section.

- It is a peer of the other sections we’ve seen so far: data(), computed, methods.

```
<script>
export default {
    name: "product-review",
    props() {
      // your props go here
    },
    data() {
      ...
    },
    computed: {
      ...
    },
    methods: {
      ...
    }
}
</script>
```

Properties Example

Consider the following example, where we utilize props to send data from a parent component to a child component:

On App.VUE (or parent component)

```
<template>
  <div id="app">
...
    <product-review
      name="Toothbrush"
      description="To brush teeth"
       />
...
  </div>
</template>
```

Child Component (script)

```
  props: {
    name: String,
    description: String,
  },
```

The child component can now use these props, for instance having **{{ name }}** on the template would result in *Toothbrush* being rendered.

Let’s Define Some Props

Emissions

- To **transmit data from a child to a parent** we want to use an emission.

Defining an Emission

- We can create an emission by tying it to an event handler, or a method attached to an event handler.

```
methods: {
    handleFilter() {
        this.$emit('filter', this.searchText);
    }
}
```

- When the method handleFilter is called it does the following: ***Emit an event called filter which contains the value of this.searchText*** (this.searchText is a data element within the JSON object inside data().

Emission Example: part 1

Child Component

```
<template>
    <form v-on:submit.prevent="handleFilter">
        <input type="text" placeholder="Filter on..."
v-model="searchText"><button>Filter</button>
    </form>
</template>
<script>
export default {
    name: 'search-bar',
    data() {
        return {
            searchText: ''
        };
    },
    methods: {
        handleFilter() {
            this.$emit( 'filter', this.searchText);
        }
    }
}
</script>
```

1. The user types something, because the field is bound to the JSON data model, *searchText* takes on the value of what the user types.

2. We define an $emit passing the value of the searchText from the JSON data model.

3. On the parent component, we need an event handler to respond to the "filter" emission event..

4. In this case the action to take in response to this event is to call a method called *handleFilter*

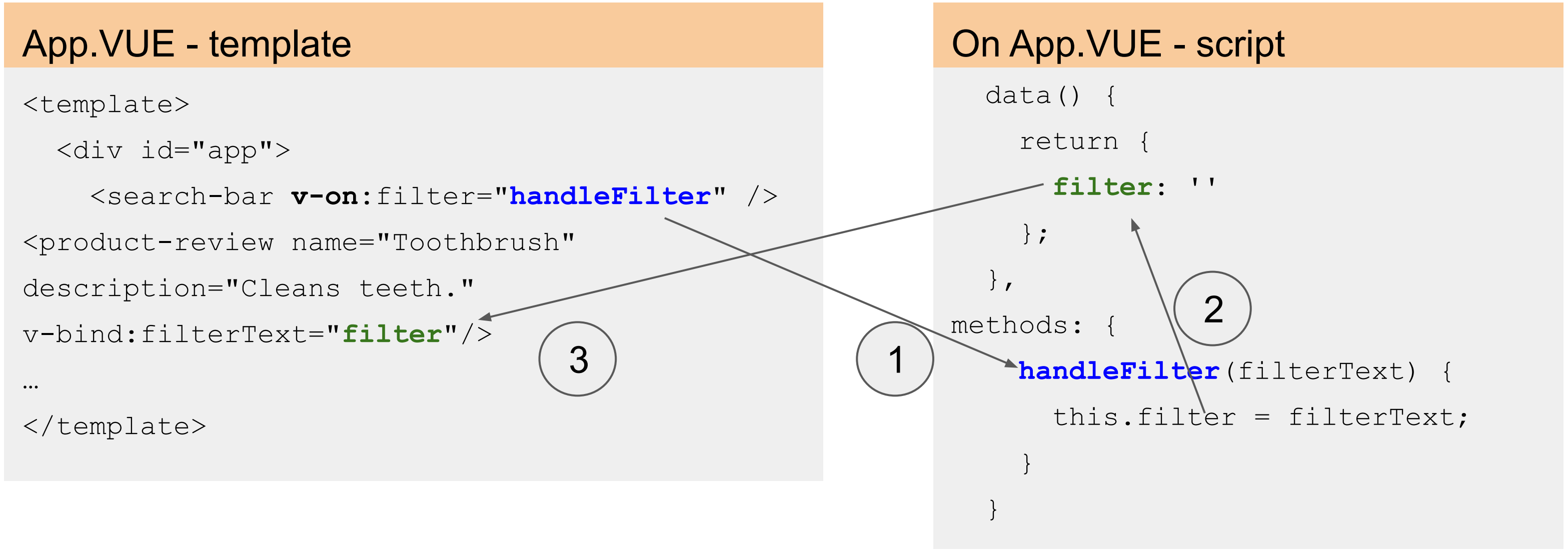
On App.VUE (or parent component)

```
<template>
  <div id="app">
    <search-bar v-on:filter="handleFilter" />
…
</template>
```

Emission Example: part 2

Now we can pass the emitted value to other components!

1. We run the handleFilter method in response to the filter emitted event. 2. The handleFilter method sets the filter attribute in the JSON data model. 3. When the product-review component is defined we specify that the filter property is passed to it.



App.VUE - template

```
<template>
  <div id="app">
    <search-bar v-on:filter="handleFilter" />
<product-review name="Toothbrush"
description="Cleans teeth."
v-bind:filterText="filter"/>
…
</template>
```

On App.VUE - script

```
  data() {
    return {
      filter: ''
    };
  },
methods: {
    handleFilter(filterText) {
      this.filter = filterText;
    }
  }
```

Emission Example: part 3

Finally, the filter value is stored within the prop on the product-review component.

App.VUE - template

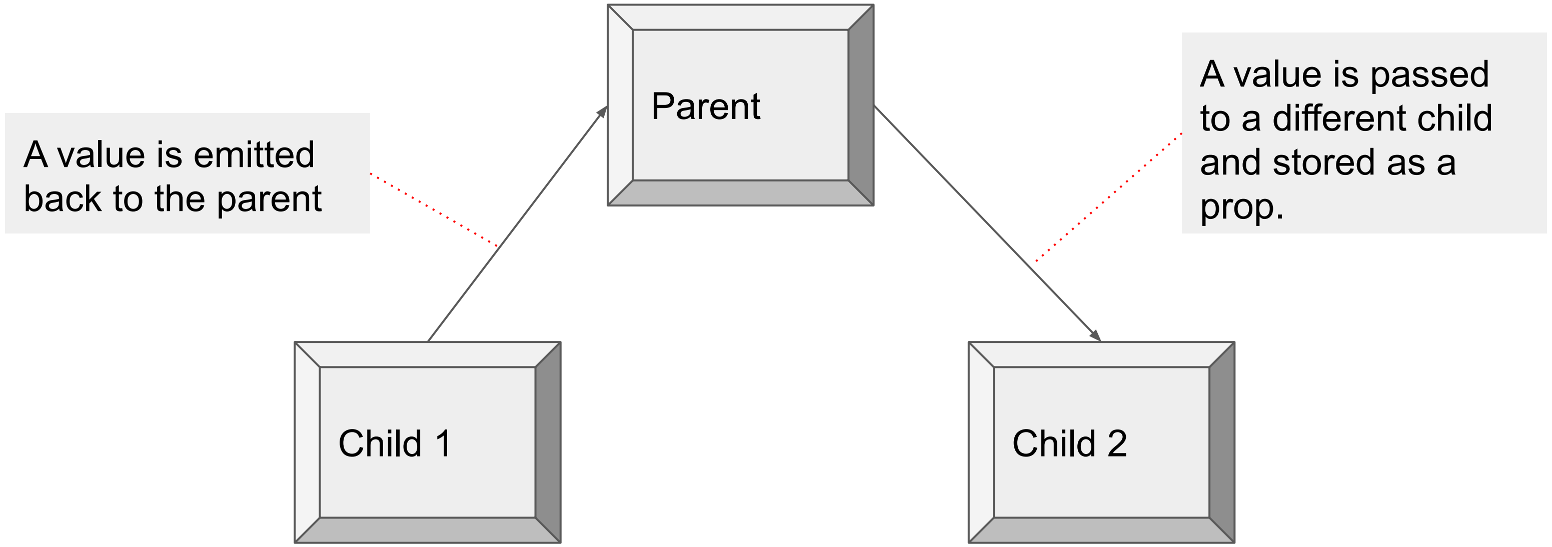
```
<template>
  <div id="app">
    <search-bar v-on:filter="handleFilter" />
<product-review name="Toothbrush"
description="Cleans teeth."
v-bind:filterText="filter"/>
…
</template>
```

Second Child Component

```
<script>
export default {
  name: "product-review",
  props: {
    name: String,
    description: String,
    filterText: String
  },
```

Emission Example: Summary

Here is a summary of how data was transmitted from a child to a parent, then from a parent to a different child


Parent
A value is emitted back to the parent
A value is passed to a different child and stored as a prop.
Child 1
Child 2

Let’s Implement This Logic