Making MessageList a slot based Component

In this lesson, we will be making MessageList a slot based component.

WE'LL COVER THE FOLLOWING

- Make MessageList Slot Based

Slots are the way to make content distribution happen in the web components world. They make the structure of the components much more flexible, moving the responsibility of managing the state to the parent component. For example, we can have a `List` component, and different kind of item components, such `ListItem` and `ListItemImage`. They'll be used like:

```
<template>
  <List>
    <ListItem :someProp="someValue" />
    <ListItem :someProp="someValue" />
    <ListItemImage :image="imageUrl" :someProp="someValue" />
  </List>
</template>
```

The inner content of `List` is the slot itself, and it's accessible via the `<slot>` tag. So the `List` implementation looks like:

```
<template>
  <ul>
    <!-- slot here will equal to what's inside <List> -->
    <slot></slot>
  </ul>
</template>
```

And, say that the `ListItem` component looks like:

```
<template>
  <li> {{ someProp }} </li>
</template>
```

Then, the final result rendered by Vue.js would be:

```
<ul>
  <li> someValue </li>
  <li> someValue </li>
  <li> someValue </li> <!-- assume the same implementation for ListItemImage -->
</ul>
```

Make MessageList Slot Based

Let’s take a look at the `MessageList.vue` component:

MessageList.vue

```
<template>
  <ul>
    <Message
      @message-clicked="handleMessageClick"
      :message="message"
      v-for="message in messages"
      :key="message"/>
  </ul>
</template>
```

MessageList has “hardcoded” the Message component inside. In a way, it’s more automated, but on the other, it is not flexible at all. What if you want to have different types of Message components? What about changing its structure or styling? That’s where slots come in handy.

Let’s change `Message.vue` to use slots. First, move the `<Message...` part to the `App.vue` component, as well as the `handleMessageClick` method, so that it’s used externally:

Message.vue

```
<template>
  <div id="app">
    <MessageList>
      <Message
        @message-clicked="handleMessageClick"
        :message="message"
        v-for="message in messages"
        :key="message"/>
    </MessageList>
  </div>
</template>

<script>
```

```
import MessageList from "./components/MessageList";
import Message from "./components/Message";


export default {
  name: "app",
  data: () => ({ messages: ["Hey John", "Howdy Paco"] }),
  methods: {
    handleMessageClick(message) {
      console.log(message);
    }
  },
  components: {
    MessageList,
    Message
  }
};
</script>
```

Don't forget to import the Message component and add it to the `components` option in `App.vue`.

Then, in `MessageList.vue`, we can remove the references to `Message`, looking like:

MessageList.vue

```
<template>
  <ul class="list-messages">
    <slot></slot>
  </ul>
</template>

<script>
export default {
  name: "MessageList"
};
</script>
```

In the next lesson, we'll be learning more about `$children` and `$slots.`