Class 5: Responding to State Changes

1. Review: Vue.js Events & Reactive Variables
2. Conditional CSS Classes
3. Custom Events
4. Watchers

Review: Components

`components/StarRating.vue`:

```
<script setup>
</script>

<template>
  <div>
    <button>☆</button>
    <button>☆</button>
    <button>☆</button>
    <button>☆</button>
    <button>☆</button>
  </div>
</template>

<style scoped>
</style>
```

`App.vue`:

```
<script setup>
import StarRating from './components/StarRating.vue'
</script>

<template>
  <StarRating />
</template>
```

Review Props

`components/StarRating.vue`:

```
<script setup>
const props = defineProps({
  name: String
})
</script>

<template>
  <div>
    <span>{{ name }}</span>
    <button>☆</button>
    <button>☆</button>
...
```

`App.vue`:

```
<script setup>
import StarRating from './components/StarRating.vue'
</script>

<template>
  <StarRating name="Movie Rating"/>
</template>
```

Review: Prop Binding

Literal values:

```
<StarRating name="Movie Rating"/>
```

Variables, Reactive Variables, Expressions:

```
<script setup>
import { ref } from 'vue'
const movieRating = ref('Movie Rating')
</script>

<template>
  <StarRating v-bind:name="movieRating"/>
</template>
```

```
<script setup>
import { ref } from 'vue'
const movieRating = ref('Movie Rating')
</script>

<template>
  <StarRating :name="movieRating"/>
</template>
```

Review: Conditional Rendering

`components/StarRating.vue`:

```
<script setup>
defineProps({
  name: String
})
</script>

<template>
  <div>
    <span v-if="!!name">{{ name }}</span>
    <button>☆</button>
...
```

Review: Rendering Lists – `v-for` Directive

`components/StarRating.vue`:

```
<script setup>
defineProps({
  name: String
})
const stars = [1, 2, 3, 4, 5]
</script>

<template>
  <div>
    <span>{{ name }}</span>
    <button v-for="num in stars" :key="num">☆</button>
  </div>
</template>
```

Conditional CSS Classes

React: Conditional CSS Classes

```
export default function StarRating({ variant = "filled" }) {
  return (
    <div>
      {[1, 2, 3, 4, 5].map(num => (
        <button key={num}
          className={variant === "filled" ? "solid" : "outline"}>☆</button>
      ))}
    </div>
  )
}
```

Vue.js: Conditional Class Binding

```
<script setup>
defineProps({
  variant: {
    type: String,
    default: 'filled'
  }
})
</script>

<template>
  <div>
    <button v-for="num in 5" :key="num"
      :class="variant === 'filled' ? 'solid' : 'outline'">
      ☆</button>
  </div>
</template>
```

Activity: Conditional Class Binding

Working with a peer, complete the following tasks:

1. Open the activity repository.
2. Conditionally render the `playing` CSS class on the `<button>` when the `SoundButton` component its currently playing.

Custom Events

React: Custom Events

```
export default function StarRating({ onRate }) {
  return (
    <div>
      {[1, 2, 3, 4, 5].map(num => (
        <button key={num} onClick={() => onRate(num)}>☆</button>
      ))}
    </div>
  )
}
```

```
export default function App() {
  return (
    <StarRating onRate={(num) => console.log(`Rated ${num} stars!`)}/>
  )
}
```

Vue.js: Custom Events

```
<script setup>
const emit = defineEmits(['rate'])
</script>

<template>
  <div>
    <button v-for="num in 5" :key="num" @click="$emit('rate', num)">☆</button>
  </div>
</template>
```

```
<template>
  <StarRating v-on:rate="(num) => console.log(`Rated ${num} stars!`)"/>
</template>
```

Activity: Custom Events

Working with a peer, complete the **activity handout**.

```
<script setup>
const emit = defineEmits(['rate'])
</script>

<template>
  <div>
    <button v-for="num in 5" :key="num" @click="$emit('rate', num)">☆</button>
  </div>
</template>
```

```
<template>
  <StarRating v-on:rate="(num) => console.log(`Rated ${num} stars!`)"/>
</template>
```

Activity: Play and Stop Events

In the activity repository, emit `play` and `stop` events from the `SoundButton` component. The parent component should listen for these events and update the `currentSounds` reactive variable. **Hint:** `removeCurrentSound`

```
<script setup>
const emit = defineEmits(['rate'])
</script>

<template>
  <div>
    <button v-for="num in 5" :key="num" @click="$emit('rate', num)">☆</button>
  </div>
</template>
```

```
<template>
  <StarRating v-on:rate="(num) => console.log(`Rated ${num} stars!`)"/>
</template>
```

Watchers

Vue.js: Watchers

Definition: A watcher is a function that observes changes to a reactive variable and executes a callback function when the variable changes.

```
<script setup>
import { ref, watch } from 'vue'
const rating = ref(0)

watch(rating, (newVal, oldVal) => {
  console.log(`Rated ${newVal} stars!`)
})
</script>

<template>
  <StarRating v-on:rate="(num) => {rating = num}"/>
</template>
```

Activity: Watchers

In the activity repository, add a watcher for the `currentSounds` reactive variable. When the variable changes, update `document.title` with the names of the currently playing sounds.

```
<script setup>
import { ref, watch } from 'vue'
const rating = ref(0)

watch(rating, (newVal, oldVal) => {
  console.log(`Rated ${newVal} stars!`)
})
</script>

<template>
  <StarRating v-on:rate="(num) => {rating = num}"/>
</template>
```

Gotcha: Deep Watching

When watching an object or array, by default, the watcher will only trigger when the reference to the object or array changes. To watch for changes within the object or array, you need to use the `deep` option.

```
<script setup>
import { ref, watch } from 'vue'
const players = ref([13, 22, 32, 48])

watch(players, (newVal) => {
  console.log(`Players changed: ${newVal}`)
},
{ deep: true })
</script>
```

What's Next

Homework 2 (Drum Simulator) released today.