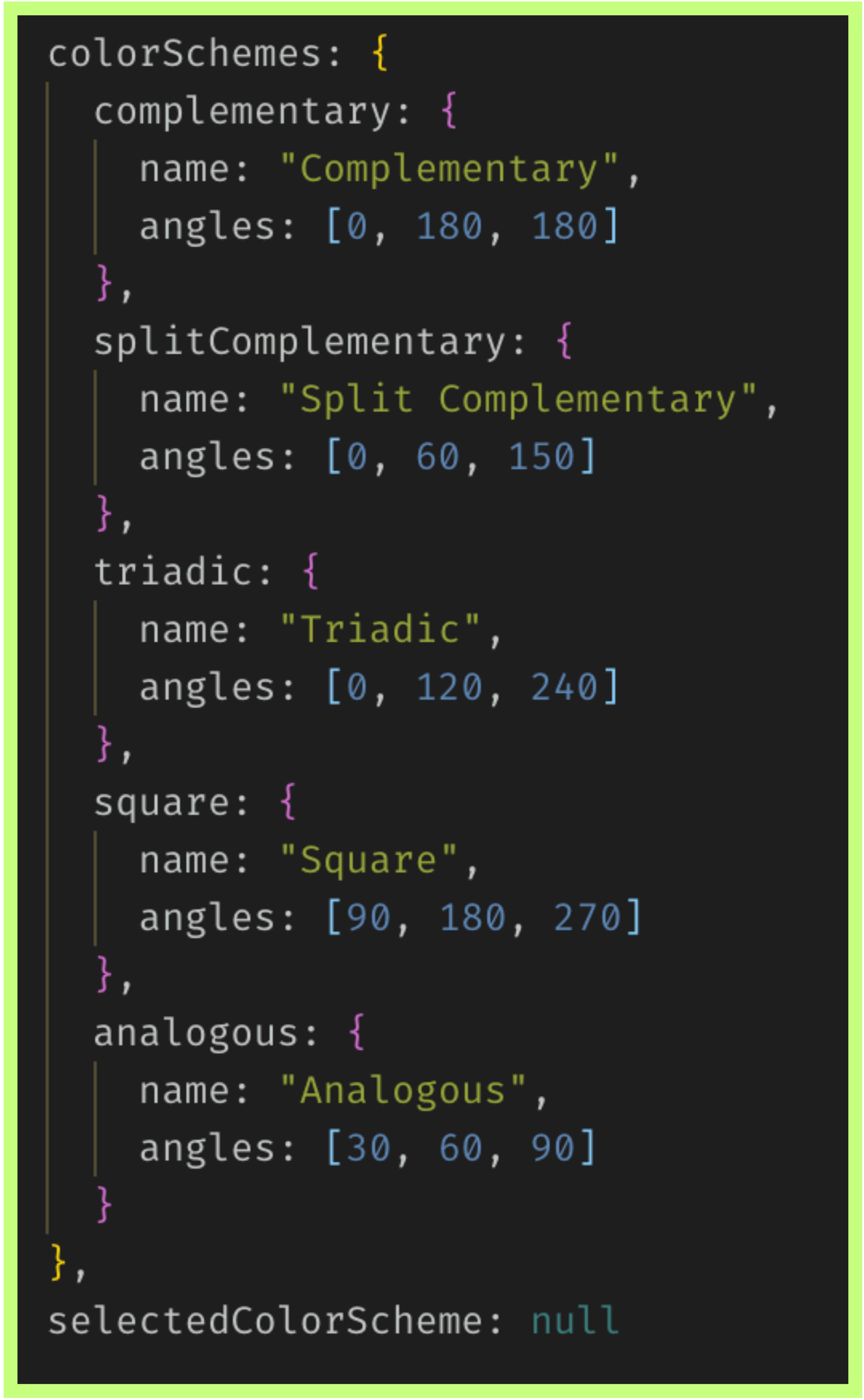
Analogous Colors
Complementary
Split Complementary
Triadic
Square
Analogous
Colours

```
<h2>{{ selectedColorScheme.name }} Colors</h2>
<ul class="color-scheme">
  <li v-for="n in 4" :key="n"></li>
</ul>

<input type="color" v-model="primaryBgColor" @change="updateColors" />

<button
  v-for="scheme in colorSchemes"
  @click="selectColorScheme(scheme)"
  :key="scheme.name"
>
  {{ scheme.name }}
</button>
```

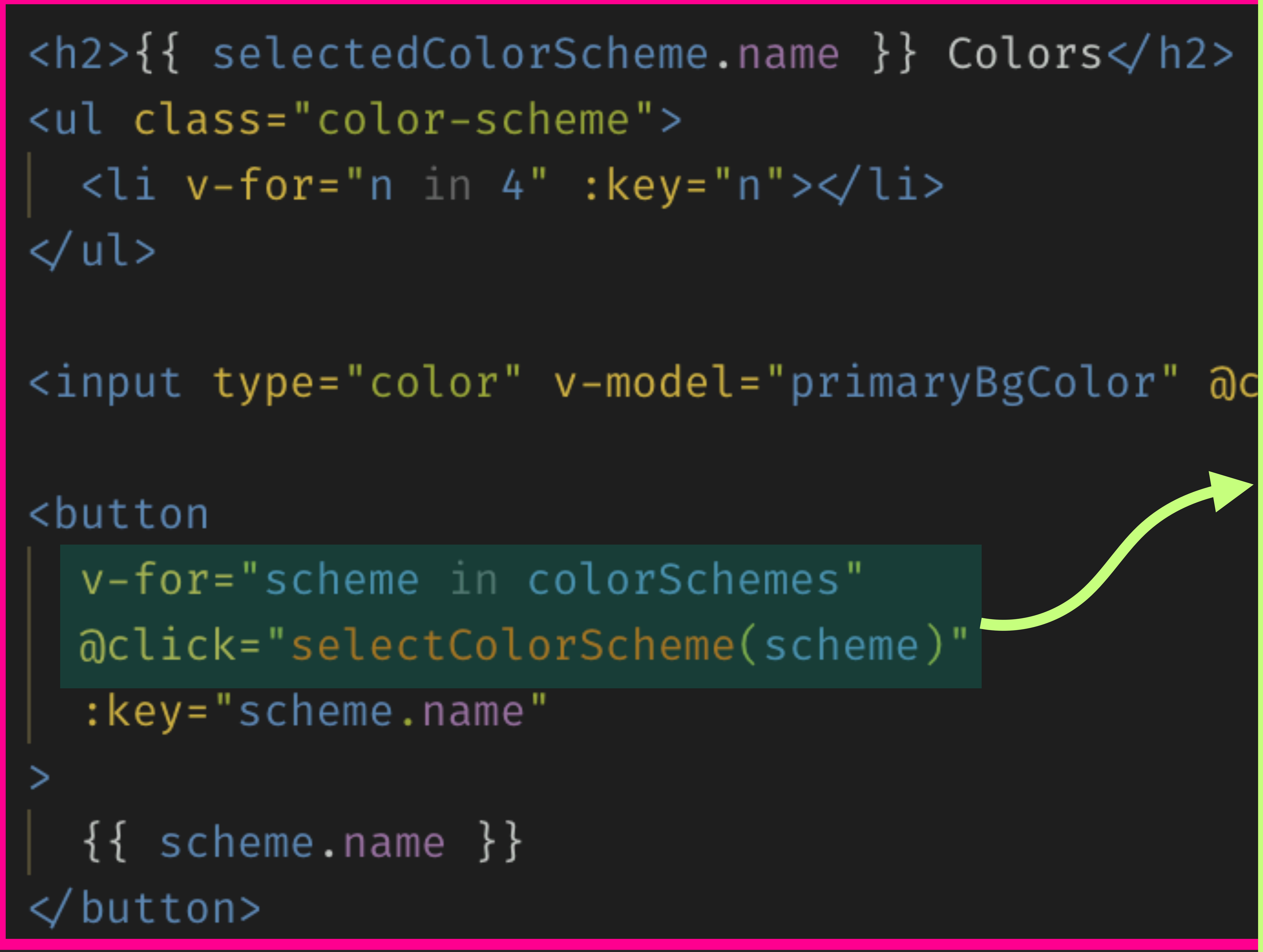
USING VUE TO SELECT THE COLOUR SCHEME



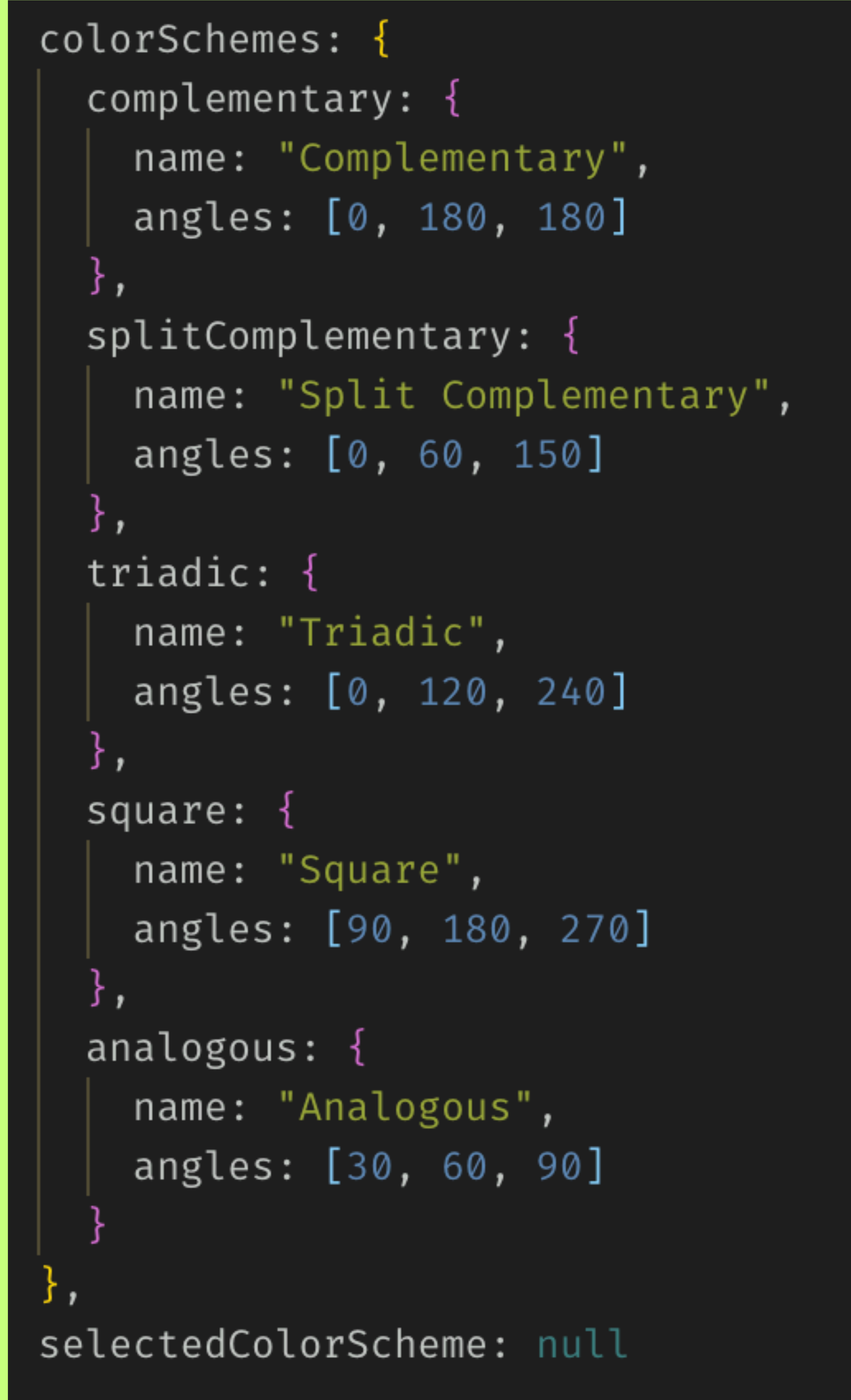
```
colorSchemes: {
  complementary: {
    name: "Complementary",
    angles: [0, 180, 180]
  },
  splitComplementary: {
    name: "Split Complementary",
    angles: [0, 60, 150]
  },
  triadic: {
    name: "Triadic",
    angles: [0, 120, 240]
  },
  square: {
    name: "Square",
    angles: [90, 180, 270]
  },
  analogous: {
    name: "Analogous",
    angles: [30, 60, 90]
  }
},
selectedColorScheme: null
```

Analogous Colors
Complementary
Split Complementary
Triadic
Square
Analogous
Colours

USING VUE TO SELECT THE COLOUR SCHEME

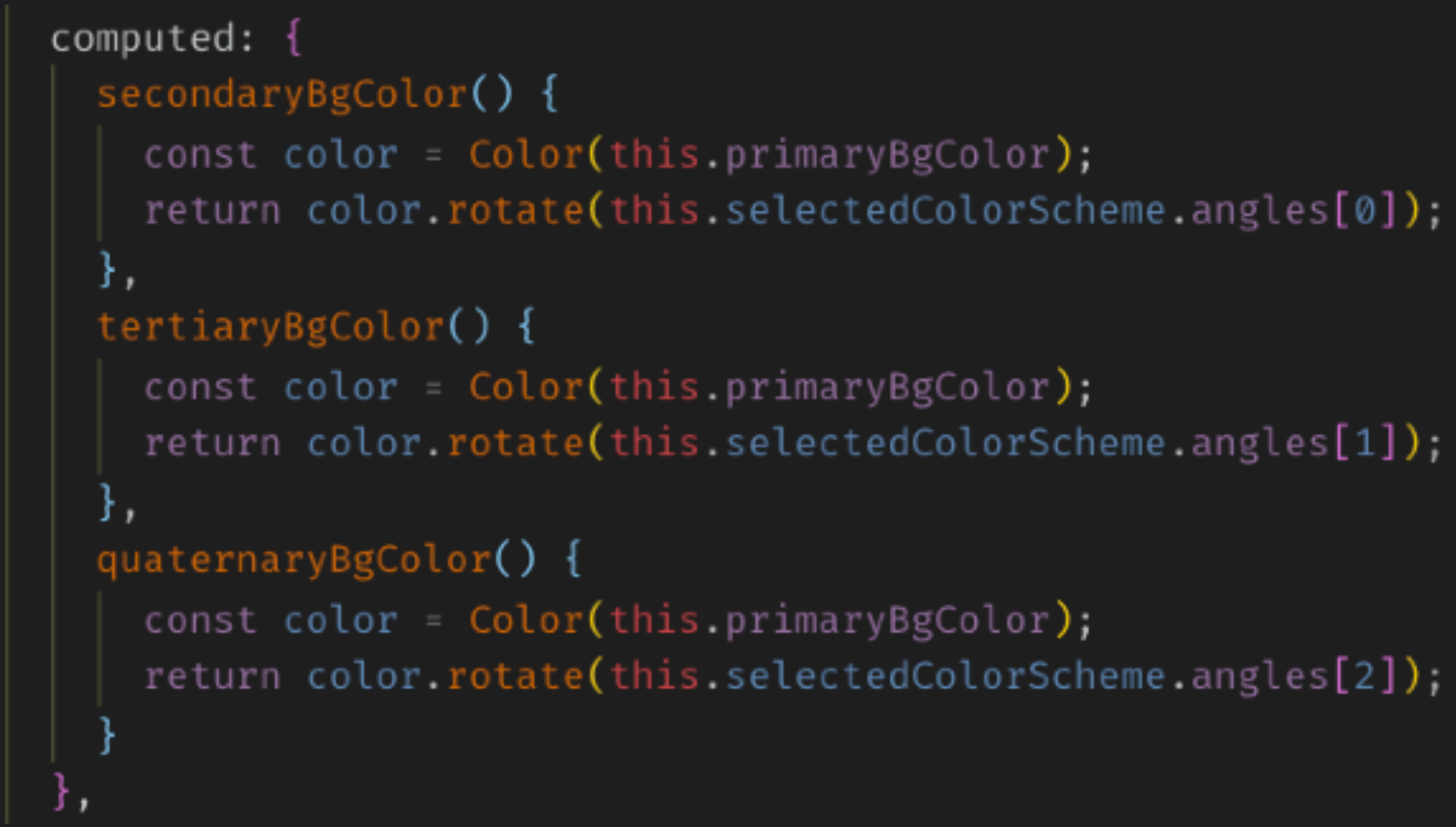


```
<h2>{{ selectedColorScheme.name }} Colors</h2>
<ul class="color-scheme">
  <li v-for="n in 4" :key="n"></li>
</ul>

<input type="color" v-model="primaryBgColor" @c

<button
  v-for="scheme in colorSchemes"
  @click="selectColorScheme(scheme)"
  :key="scheme.name"
>
  {{ scheme.name }}
</button>
```



```
colorSchemes: {
  complementary: {
    name: "Complementary",
    angles: [0, 180, 180]
  },
  splitComplementary: {
    name: "Split Complementary",
    angles: [0, 60, 150]
  },
  triadic: {
    name: "Triadic",
    angles: [0, 120, 240]
  },
  square: {
    name: "Square",
    angles: [90, 180, 270]
  },
  analogous: {
    name: "Analogous",
    angles: [30, 60, 90]
  }
},
selectedColorScheme: null
```

ALLOWING THE USER TO SELECT THE COLOUR SCHEME



```
computed: {
  secondaryBgColor() {
    const color = Color(this.primaryBgColor);
    return color.rotate(this.selectedColorScheme.angles[0]);
  },
  tertiaryBgColor() {
    const color = Color(this.primaryBgColor);
    return color.rotate(this.selectedColorScheme.angles[1]);
  },
  quaternaryBgColor() {
    const color = Color(this.primaryBgColor);
    return color.rotate(this.selectedColorScheme.angles[2]);
  }
},
```