Adding More Theme Props

This lesson discusses how we can add custom theme props to our themes.

WE'LL COVER THE FOLLOWING

Step 1: Add custom prop

Adding additional props to the theme definition is easy:

```
$ct-themes: (
  'default': (
    'brightness': 'light',

    'border-radius': 3px // We're adding our own defined theme prop here t
hat we'll use ourselves
  ),
  'default-dark': (
    'brightness': 'dark',

    'border-radius': 2px
  ),
);
```

In the above example, we added a `border-radius` prop to our themes. This allows us to change this `border-radius` value from theme to theme.

The next step is creating a dynamic CSS variable that points to this additional prop.

Step 2: Link to a CSS variable

> **Note:** A dynamic CSS variable in this context is one that changes its

> values depending on the active theme.

We do this by using a SCSS mixin defined in `css-theming`:

```
@include ct-themes-apply {
  --border-radius: #{map-get($theme, 'border-radius')};
}
```

> **Note**: A SCSS mixin simply defines some content that we can insert in multiple places.

The `ct-themes-apply` mixin allows you to define additional dynamic CSS variables tied with the theme. Inside, you'll have access to the resolved theme definition, in the `$theme` variable. So, all we have to do here is obtain the `border-radius` value from the theme using `map-get`:

```
map-get($theme, 'border-radius')
```

and create a new CSS variable from that:

```
--border-radius: #{SCSS EXPRESSION};
```

Now, we have a dynamic CSS variable that we can easily consume! This is the end result that we want to reach: a simple CSS variable that we can reference in our app.