System and User Defined Themes

This lesson talks about the different types of themes in an app.

WE'LL COVER THE FOLLOWING

System defined themes

A theme that is defined by the developers of the app is called a **system defined theme**. In the simplest case, an app would have two defined themes: a *light* and a *dark* theme.

A system defined theme's props are defined by the designer of the app.

Example

Assume that we have a `border-radius` value we use in cards in the app. Right now, it's a constant value and remains the same no matter the currently applied theme. But let's say we want it to change per theme so that we can tweak it between different themes. Our theming platform will be able to support this.

User defined themes

We talked before about *system defined themes*, and how they're defined by the developers of the app. This means the app will have a fixed number of well-known themes that the user can choose from. Imagine that we want to allow the user to customize certain (or all) aspects of the theme; this is what we call *user defined themes*.

We learned in a previous lesson that we can access design values (CSS props)

programmatically:

```
const style = getComputedStyle(element);
const value = style.getPropertyValue(prop);
```

In a similar way, we can set those values:

```
const style = getComputedStyle(element);
const value = style.setPropertyValue(prop, value);
```

In theory, this allows the user to not only choose one of our system defined themes, but even create one of their own and customize the props of the theme to whatever they like.

Now that you have basic insights on the general theming concepts, let's begin using all the ideas we introduced by learning how they fit in the `css-theming` package.