Setup

This lesson shows how to do a basic setup of css-theming in your app.

WE'LL COVER THE FOLLOWING

Following are three things you have to do to setup your project to work with `css-theming`:

- Optionally define themes in your variables file.
- Import the main `css-theming` SCSS file from your app scss file.
- Call `initializeTheming` from javascript/typescript in the startup of your app.

Defining system themes

By default, `css-theming` gives you two themes, one *light* theme called `default`, and another *dark* theme called `default-dark`. You're free to completely override this to define your own system themes.

You override this in your **variables** file. A **variables** file is a SCSS file you use that doesn't emit any real CSS. It contains all the definitions (SCSS variables, mixins, functions, etc.,) of your app. You usually import this file in all your concrete SCSS files; those would be the main app SCSS file, and SCSS files for components if your app is a SPA.

The following is an example **variables** file that defines three themes:

```
// Override the system defined themes
```

```
$ct-themes: (
  'default': (
    'brightness': 'light',
  ),
  'default-dark': (
    'brightness': 'dark',
  ),
  'high-contrast': (
    'brightness': 'light',
    ...
  )
);

@import '../../node_modules/css-theming/src/scss/pure';
```

We import the **pure** SCSS file from `css-theming` after we override the variables we want, as shown in the above snippet. This **pure** file is what contains all the definitions of `css-theming`, including all SCSS variables it defines (themes, colors, swatches, etc.).

> **Note:** You can think of this **pure** file as the **variables** file of `css-theming`.

What we're doing here is almost exactly the same as what you'd do when using something like bootstrap for example. You would import bootstrap's **variables** file in all your SCSS files, but you only import bootstrap's main file in your main app SCSS file.

Importing the main `css-theming` SCSS file

In your main app SCSS file:

```
@import '../../node_modules/css-theming/src/scss/css-theming';
```

Calling `initializeTheming`

Call `initializeTheming` as soon as you can in your app. This can be the `created` lifecycle hook of the main app component in a Vue.js app, the constructor of the main app component in an angular app, etc.

```
import {
```

```
  initializeTheming,
} from 'css-theming';

// Initializes the default category themes
initializeTheming(/* theme */);
```

In this basic example, `initializeTheming` can take the theme that you want to initialize at first. Let's assume we always want to load `default-dark` whenever the application loads:

```
import {
  initializeTheming,
  getTheme,
} from 'css-theming';

// Initializes the default category themes
initializeTheming(getTheme('default-dark'));
```

Of course, instead of hard coding the initial theme you load, you can easily get this value from local storage or from your backend if you save the user's preference there.

> **Note:** If you register additional category themes you'll call `initializeTheming` once for each category, as we'll explain in a later lesson.

Setting a new theme

You tell `css-theming` to change the current theme by calling `setTheme`:

```
import {
  setTheme,
} from 'css-theming';

// Initializes the default category themes
setTheme('[theme-name]');
```

This of course can be in response to a button click for example, that the user clicks on to choose what theme they want to activate.

After we've successfully setup our app to use `css-theming`, we can learn about

After we've successfully setup our app to use `css-theming`, we can learn about the different ways we can customize it. Please move to the next lesson to see how it's done.