

Mixing Colors

This lesson discusses the main problems that arise when dealing with colors in an application and how mixing colors in a certain way provides the solution.

WE'LL COVER THE FOLLOWING ^

- Problems
- Solutions
 - Using the alpha channel
 - Mixing colors
 - Dynamic mixing

Problems

Here's our conundrum when it comes to colors:

- Manually picking up the different variations of colors we need in our app is hard and prone to error.
- When different themes have different brightness levels, the color from one of them usually doesn't translate very well to the other.

Here's an example of what we're talking about. Let's create some `label` and use `blue` as a `background` and `color`, we'll use a lighter blue for the background:

Output
HTML
CSS (SCSS)
<div>This is a label: <code>css-theming</code></div>



Light mode

Now, what if we're in a dark theme:

Output

HTML

CSS (SCSS)

This is a label: `css-theming`



Dark mode

It becomes obvious that the light blue background is out of place in this dark theme. It's a *dark* theme, so why are we using a color that's very close to white as a background?

Solutions

Using the alpha channel

The alpha channel is the 4th component of an RGBA color. It controls the *opacity* of the color. A **0** alpha means the color is fully invisible, while **255** means it's fully opaque.

So, let's talk about a better way to color this `label` component. A straightforward solution is to use a blue color with an alpha channel.

Output

HTML

CSS (SCSS)

This is a label: `css-theming`



Light mode

Output

HTML

CSS (SCSS)

This is a label: `css-theming`



Dark mode

Note: Let's ignore for now that the text itself looks bad. We'll talk specifically about text colors and how to correct them in a later lesson.

Sure, that's a bit better than having a whitish color in a dark theme throwing our concentration off. But this has several problems; first, a side effect of using an alpha channel, the background color of the label will blend into the background color behind it. This is known as **Alpha compositing** and can create all sorts of visual nuisances, the result will vary depending on the background color as well:

Output

HTML

CSS (SCSS)

This is a label: `css-theming`



Dark mode



The result is almost unreadable, and certainly fails the [AA standard contrast ratio](#)! Try it with any other extreme color, just change the `background` property of the `body` element.

The problem here, as we stated, is that the transparent color we chose for the `label` is blending with the background color of the `body` behind it, thus creating a different color than what we intended. This is almost always bad for colors used as backgrounds.

And so, we reach the next solution.

Mixing colors

Let's imagine that we magically found the following 2 colors:

-  `#e3edff`: We were told to use this in light themes.
-  `#324055`: We were told to use this in dark themes. It might not look like it on a white background, but this is actually a blue color.

Let's try and see:

Output

HTML

CSS (SCSS)

This is a label: `css-theming`



Light mode

Output

HTML

CSS (SCSS)

This is a label: `css-theming`



Dark mode

This is exactly what we want! It even works in any background because `#e3edff` (background color of the `label`) is a fully opaque color:

Output

HTML

CSS (SCSS)

This is a label: `css-theming`

Dark mode

Note: Don't worry that the text color in the above examples was changed too, we'll talk about this extensively in the next lesson. For now, we're focusing on the background color.

Do note that the examples here are extremely simplified. In a real-world app, the problems mentioned will be more catastrophic, and updating your app to use the correct approach of mixing colors can certainly get you much closer to a good looking app in any brightness.

Dynamic mixing

Now, we imagined that the *correct* colors magically appeared in front of us. That's fine for this course, but how do we obtain those colors in a real app? Handpicking them? That would be a nightmare!

SCSS can help here once again. We'll take advantage of the `mix` function in SCSS:

```
mix(color1, color2, percentage)
```

Output

JavaScript

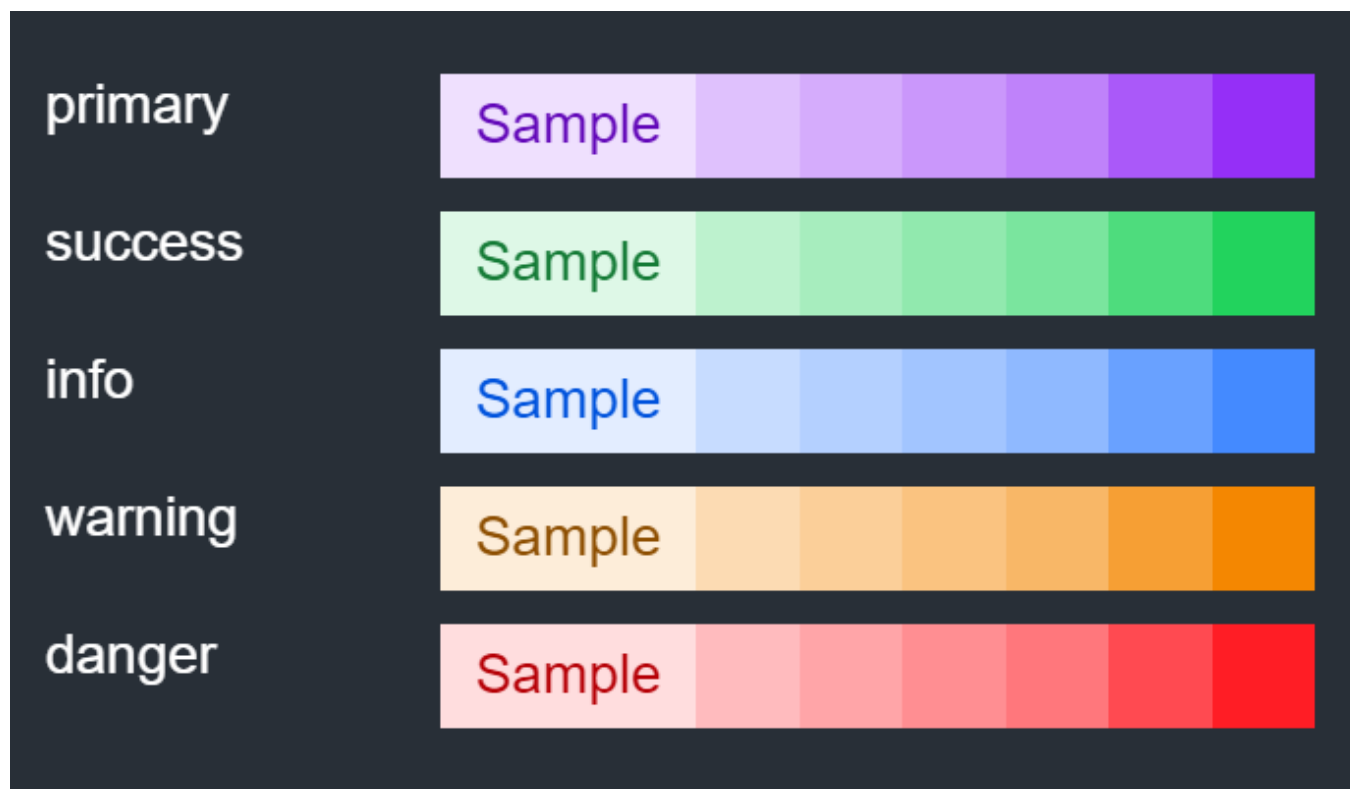
HTML

CSS (SCSS)

Light mode

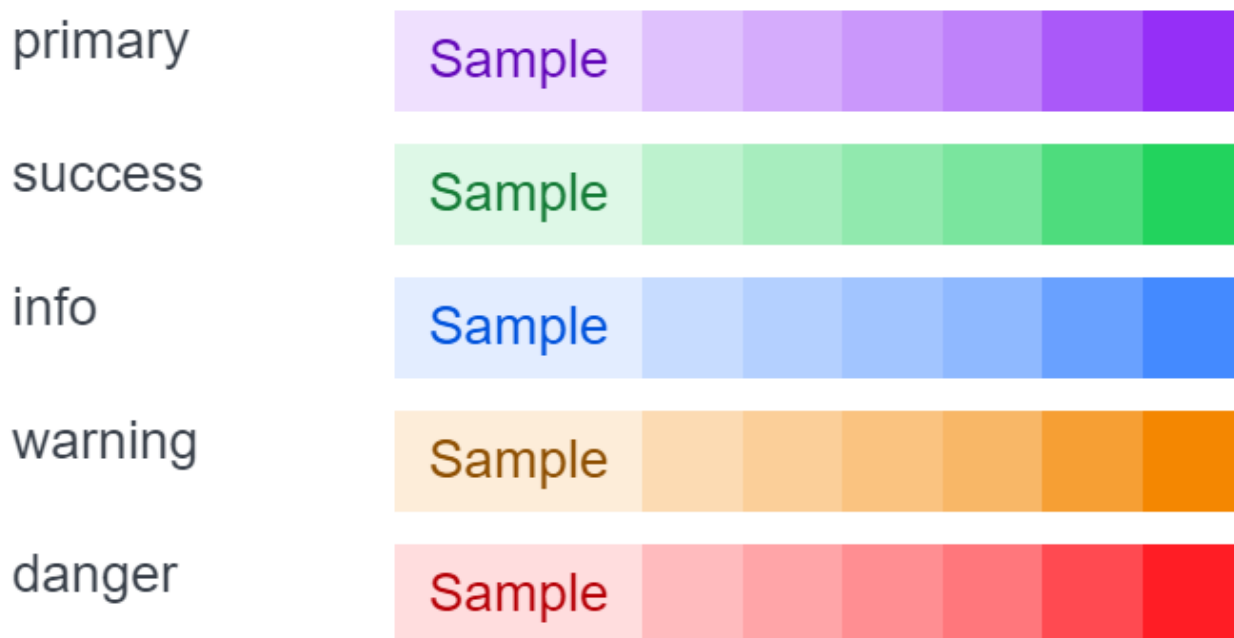
In the above example, we were able to easily recreate fully opaque colors that could act as backgrounds for light and dark themes. Using this pattern, we can recreate the whole palette and create versions that are appropriate for their respective theme brightness.

Instead of this palette that isn't really appropriate in a dark theme:

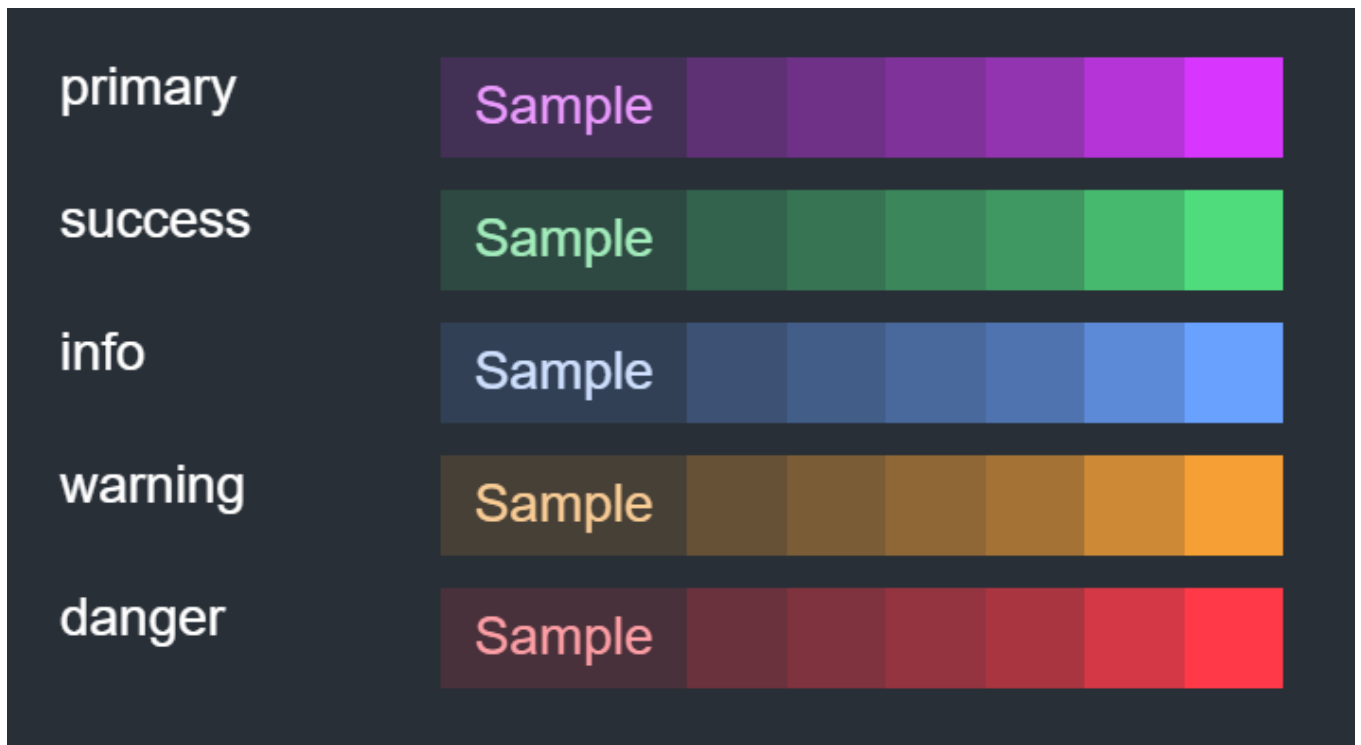


The result ends up being the following:

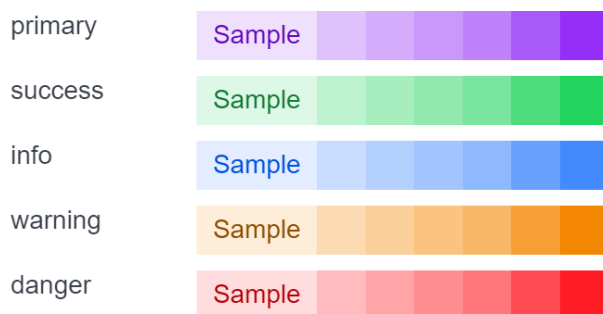
- Light:



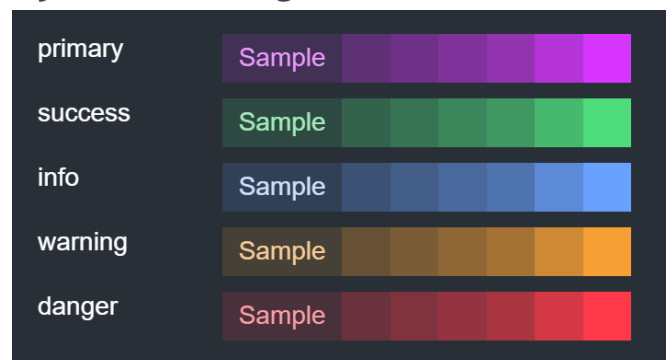
- Dark:



Dynamic mixing in a light theme:



Dynamic mixing in a dark theme:



That's how we can properly mix colors for use in backgrounds in any theme. But, we still encountered problems with the text. The next lesson focuses on text colors.