

# Icons and Svgs

This lesson discusses how we'll make icons and svgs play nice with our theme.

## WE'LL COVER THE FOLLOWING ^

- Font icons
- Svgs

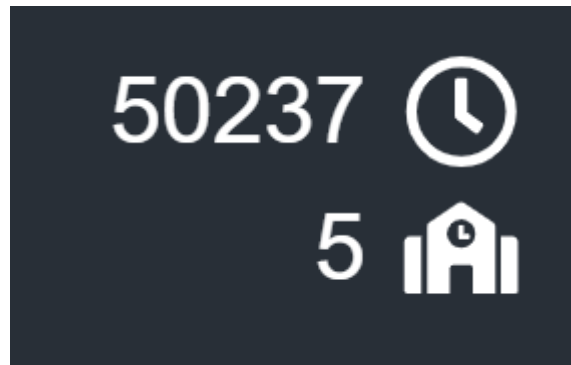
## Font icons #

Font icons are the same as any text, so we don't have much to talk about here. Simply changing the color of the text will change the color of the icons:

- Light theme

50237   
5 

- Dark






## Svgs #

An **svg** is another type of HTML element that helps us to manipulate and interact with different animated graphics and shapes.




That is why they need some special attention. Unlike normal HTML elements, svgs aren't effected by `background` and `color` CSS properties. We use `fill`, `stroke`, and `stroke-width` to control the color. Let's quickly show what those properties control.

Here's an svg that has a fill:

Here's an svg that has a fill.

Output
HTML
CSS (SCSS)

<div></div>

Here's an svg that has a stroke:

Output
HTML
CSS (SCSS)

<div></div>

Here's an svg that has both a fill and a stroke:

Output
HTML
CSS (SCSS)



Usually, we'll be using ready-made svgs we find on the web or ones that are given to us by a designer. For most of those svgs they'll only have a `fill`. We want simple svgs to act the same as text, meaning we want to color them as if they were text. We can do the following:

```
svg {  
  fill: currentColor;  
}
```

We're setting the `fill` property to `currentColor`, instructing the `fill` color to get its value from the currently inherited `color` value.

**Note:** This way of styling the svgs only works with [inline svgs](#).

Since we're styling the svg, we can do whatever we want if it has a stroke too:

```
svg {  
  // You could also use currentColor for the stroke and something else for  
  // the fill if it has a fill.  
  stroke: var(--primary);  
}
```

---

That's it for icons and svgs. In the next lesson, you'll learn how to get programmatic access to design values.

