Theming with CSS Variables

One of the best things about CSS variables is that they allow you to create themes for your content.

Reviewing CSS Variables

CSS variables or, more precisely, CSS Custom properties allow you to define reusable values for elements across your stylesheet. This is particularly good for consistency and ease of change, we only need to make one change and all the places where we use the variable will be changed automatically.

Variables can be global, those defined in the `:root` pseudo-element (equivalent to the HTML document) and those specific to elements in the page.

Defining The Core Theme

The example below shows a set of variables defined globally, values that we can use regardless of the element we're working with.

The second block defines attributes for messages with different levels of severity (`info`, `warning` and `danger`). One block has attributes that are common to all messages, the other is specific to each level and, for now, adds a background color specific to each type of message we're working with.

```
:root {
  /* generic margin values*/
  --margin-small: 0.5em;
  --margin-normal: 1em;
  --margin-large: 2em;

  /* generic padding values */
  --padding-small: 0.5em;
  --padding-normal: 1em;
  --padding-large: 2em;

  /* MESSAGES */
```

```
  /* message common attributes */
  --message-bordercolor: rgba(0, 0, 0, 1);
  --message-bordertype: solid;
  --message-borderthickness: 1px;
  --message-borderradius: 10px;
  /* info background color */
  --message-info--backgroundcolor: rgba(176, 216, 230, 1);
  /* warning background color */
  --message-warning--backgroundcolor: rgba(255, 255, 224, 1);
  /* danger background color */
  --message-danger--backgroundcolor: rgba(205, 92, 92, 1);
}
```

Using the stylesheet we defined above, we can create new elements using the variables defined in the stylesheet. You can use shorthand properties such as border, shown below.

```
.message {
  border: var(--message-borderthickness) var(--message-bordertype) var(--m
  border-radius: var(--message-borderradius);
  margin: var(--margin-normal) auto;
  padding: var(--padding-small);
}

.info {
  background-color: var(--mesage-info--backgroundcolor);
}

.warning {
  background-color: var(--message-warning--backgroundcolor);
}

.danger {
  background-color: var(--message-danger--backgroundcolor);
}
```

We can also code defensively and use multiple values to cover browsers that don't support variables. In the example below, the CSS parser will work through the

different values and ignore those it doesn't support, so it'll go through RGB, RGBA and then use custom variables to assign the background color, if it doesn't understand a value it will ignore the rule so we can rely on having only one rule that the browser will understand and it'll use the last rule added to the page. I've made the assumption that if the browser supports variables it also supports RGBA, which I've used to define the color.

```
.info {
  background-color: rgb(176, 216, 230);
  background-color: rgba(176, 216, 230, 1);
  background-color: var(--message-info--backgroundcolor);
}
```

I've worked this example into a Codepen Demo that may be easier to understand.