Defining custom properties from Javascript

When I was writing a previous post, I wanted to create custom properties in Javascript that I could use in CSS based on the result of running color.js to convert colors and then test if the browser supports a given colorspace.

How does it work?

This example, as posted in Using color.js as a bridge between color spaces, takes a color defined as a new color.js `Color` object, converts it to an sRGB string as a default.

The script then tests if the browser supports different color spaces (LCH and Display-P3) using the css.supports() method. If the color space is supported then it converts the color to a string in the specified color space

```
import Color from "colorjs.io";

let rp = new Color("#663399");

"#663399"// Initialize cssColor
let cssColor;

if (CSS.supports("color", myColor)) {
  cssColor = myColor.toString();
}

if (CSS.supports("color", "color(display-p3 0 0 0)")) {
  cssColor = myColor.to("p3").toString();
}

if (CSS.supports("color", "lch(0% 0 0)")) {
  cssColor = myColor.to("lch").toString();
}
```

We can then use the value of CSS color as the value of a custom property.

Assuming the following code is part of the same script, we can do a naive first pass with code like this:

```
const root = document.documentElement

root.style.setProperty('--color-name', `${cssColor}`);
```

There are two issues with the code we've written so far.

- There is no way to specify the name of the property that we want to create. If you have more than one property then this will be a problem.
- The code is not DRY. As written the code would work for one color variable overwrite the variable each time we run it with different values.

Creating a function

The `processColor` function takes two parameters:

- **name**: the name of the color property we're creating
- **color**: the value for the color we want to create. This can be in any color space supported by color.js

Inside the function we define two vaiables that we'll use throughout the function:

- **myColor**: creates a new color.js `Color` object from the `color` parameters
- **cssColor**: empty for now, it will contain the final color we'll use

```
function processColor(name, color) {
  let myColor = new Color(color);
  let cssColor;
```

Testing for supported color spaces and setting the `cssColor` variable apporpriately.

I've added the initial test to convert the color to the `srgb` equivalent.

I've also changed the tests to test against a color of the appropriate color

space.

```
  if (CSS.supports("color", myColor)) {
    cssColor = myColor.toString();
  }

  if (CSS.supports("color", "color(display-p3 0 0 0)")) {
    cssColor = myColor.to("p3").toString();
  }

  if (CSS.supports("color", "lch(0% 0 0)")) {
    cssColor = myColor.to("lch").toString();
  }
```

I decided to use custom properties, as defined in CSS Custom Properties for Cascading Variables Module Level 1.

> We create two Javascript variables
>
> - One to hold the path to the root element in Javascript (`document.documentElement`)
> - One to hold the name of the variable we want to create converted to a string

With the two variables in place, we create a variable in the root element with the colorName variable as the first parameter and the `cssColor` variable as the second parameter.

```
  let root = document.documentElement;
  let colorName = ('--color-' + name).toString();

  let registeredColor = root.style.setProperty(colorName, cssColor);

  return registeredColor
}
```

Note that this will add the variables as inline styles for the HTML element.

> To test the code, we call it twice with two different colors defined as the six-

digit hexadecimal code.

```
processColor('rebeccapurple', '#663399')
processColor('magenta', '#ff00ff')
```

The calls to processColor will return undefined but the variables will be available to use in the CSS code.

I used the code in this gist as a proof of concept.