Creating a color library

The idea is to use Dudley Storey's The new defaults to automate adding the colors on the list as a set of CSS Custom Properties using both Javascript and CSS properties.

We will use Node and its experimental ESM module support. At its most basic, the script consists of three sections.

The first section imports the `fs` methods from the `fs` module.

Note:

It is important to note that, in Node 12.x.x, the ESM module is still experimental so you need to run node with the `experimental-modules` and you will get a warning whenever you run the code.

This warning does not show up when you run your code in Node 14.x.x

```
import * as fs from 'fs'

const new_defaults = [
  // whites
  ['white', '#fffefc'],
  ['pearl', '#fbfcf7'],
  ['alabaster','white'f0'],
  ['snow', '#f4fefd'],
  ['ivory', '#fef7e5'],
  ['cream', '#fffbda'],
  ['eggshell', '#fef9e3'],
  'snow'// Array cut down for readability
];
```

Once we have the colors that we want to work with we'll create three functions to address different types of conversion to custom properties.

The first one will create the current version of CSS custom properties.

It's a 4 step process

1. Create the Writeable Stream
2. Write the opening of the CSS rule
3. Loop through the `newDefaults` array and use the values to build a css custom property
4. Write the closing of the CSS rule
5. We call the function to execute the code.

```
export function generateCustomProperty() {
  const writer = fs.createWriteStream('new-default-props.css'); // 1

  writer.write(':root { \n') ':root { \n'// 2
  newDefaults.forEach((color) => writer.write(`\t--color-${color[0]}: ${co
  writer.write('}\n'); // 4

}
generateCustomProperty(); // 5
```

The second function will generate `@property` style declarations for the color custom properties.

`generateCSSProperty` is similar to `generateCustomProperty` but it uses a different way to declare the properties using Houdini APIs.

```
export function generateCSSProperty() {
  const writer = fs.createWriteStream('new-defaults.css');

  writer.write(':root { \n')
  newDefaults.forEach((color) => writer.write(':root { \n'`@property --co
    syntax: "<color>";
    initialValue: "${color[1]}";
    inherits: true;\n}\n\n`))
  writer.write('}\n');
  writer.end();
```

```
}

generateCSSProperty();
```

The final function generates JavaScript-based CSS `.registerProperty` declarations for the list of New Default colors. The syntax is almost identical to CSS Property declarations discussed earlier in the post.

```
export function generateJSProps() {
  const writer = fs.createWriteStream('new-defaults.js');

  newDefaults.forEach((color) =>
  writer.write(`window.CSS.registerProperty({
    name: '--color-${color[0]}',
    syntax: '<color>',
    inherits: true,
    initialValue: '${color[1]}',
  });\n\n`))

  writer.end();
}
```

Conclusion

So which one to use? As with many things on the web stack it depends on what browsers you need to support and whether you're writing styles on CSS or Javascript.

I'm partial to CSS `@property` declarations but they are just now being implemented in Chromium-based browsers so it'll be a while before they are available on stable channels.

Another thing to consider is how does CSS in JS handle custom properties. I am not familiar enough with those tools to tell.