New color features in CSS

There are a lot of interesting color features in CSS that are either under development or have recently been implemented in browsers.

If I've written about these features before, I will provide an update as to where they are in terms of browser support. Otherwise I will document the feature.

New color spaces

CSS Color Level 4 specification introduces many new color spaces into the CSS world.

Some of these, like `display-p3` are already supported in Safari while others are just becoming available in Chromium browsers

hwb()

HWB stands for hue, whiteness, and blackness with an optional fourth value separated by a slash representing alpha transparency.

```
hwb(194 0% 0%) /* #00c3ff */
hwb(194 0% 0% / .5) /* #00c3ff with 50% opacity */
```

Note how the hwb function **does not** use commas to separate the values and how the alpha value is separated with w a slash.

`hwb()` produces colors from the sRGB space, the same as HSL and RGB. It just makes it easier for humans to understand and work with.

Other color spaces

The way colors are represented is done with a color space. Each color space offers various features and trade-offs for working with color. Some may pack all the bright colors together; some may line them up first based on their lightness.

The CSS Color Module Level 4 just became a candidate recommendation specification (the step before it's finally adopted as a specification).

It introduces ten new color and predefined color spaces to work with and new functions to manipulate colors.

Some of these colors are already supported in some browsers and operating systems combinations (like display-p3 support in Safari for macOS) and others like lch, oklch, lab, oklab will become available in the future, hopefully with all browsers adding support close to each other.

Perhaps the one I find most instriguing is LCH, a better way to describe colors than RGB and other colors spaces based on it.

LCH uses three parameters to describe colors:

- **Lightness**: a percentage
- **Chroma**: a number with a minimum of 0 and a maximum value that is theoretically unbounded but in practice seldom goes above 230
- **Hue**: an angle value in degrees. There are some special things about the use of hue in LCH colors, I'm still trying to figure them out
- **alpha**: optional fourth parameter represting the transparency as either a value between 0 and 1 or a percentage

These values are not separated by commas and the alpha value is separated by a slash (`/`).

Lea Verou does a much better and throrough work in explaining what LCH is and why should we use it in her post: LCH colors in CSS: what, why, and how?. She also provides an LCH color picker as a tool to learn about and use the LCH color space.

Note: As of this writing (early July, 2022), LCH is only supported in Safari but it's under development in Chromium browsers and under discussion in Firefox

color-mix()

Color mix is exciting. It's a CSS native way to mix colors where before we would rely on preprocessors or Javascript to do it.

THe idea is that you specify a color space then a base color, a percentage of the color to mix and the color that you want to mix.

- **colorspace**: The color space that we're working with. There is no default so

you must specify a value. The available color spaces are:

- srgb
- srgb-linear
- lab
- oklab
- xyz
- xyz-d50
- xyz-d65
- hsl
- hwb
- lch
- oklch

- **color**: Any valid color in the color space we're working with
- **percentage**: The percentage of that color to mix

We can use `color-mix()` to mix two colors, evenly (both at 50%) or unevenly (one at 40% and one at 60%).

The following custom properties show examples of those mixes between red and blue at different percentages for each color.

```
:root {
  --color1: color-mix(in srgb, blue 50%, red);
  --color2: color-mix(in srgb, blue 40%, red 60%);
  --color3: color-mix(in srgb, blue 60%, red 40%);
  --color4: color-mix(in srgb, blue 30%, red 70%);
}
```

You can also use this to lighten and darken colors using white (to lighten) and black (to darken) as the second color

The following `color-mix()` examples darken a color nine steps to black. We first create a custom property for each step that we want to darken:

```
:root {
  --color-mint: #98ecc3;

  --dark-text1: color-mix(in srgb, var(--color-mint) 10%, black);
```

```
  --dark-text2: color-mix(in srgb, var(--color-mint) 20%, black);
  --dark-text3: color-mix(in srgb, var(--color-mint) 30%, black);
  --dark-text4: color-mix(in srgb, var(--color-mint) 40%, black);
  --dark-text5: color-mix(in srgb, var(--color-mint) 50%, black);
  --dark-text6: color-mix(in srgb, var(--color-mint) 60%, black);
  --dark-text7: color-mix(in srgb, var(--color-mint) 70%, black);
  --dark-text8: color-mix(in srgb, var(--color-mint) 80%, black);
  --dark-text9: color-mix(in srgb, var(--color-mint) 90%, black);
}
```

We can then use the custom properties any place we can use a color declaration. In this case I used them as background colors so you can see the differences between steps.

```
.mint {
  background-color: var(--color-mint);
  color: black;
}

.dark-text1 {
  background-color: var(--dark-text1);
}

.dark-text2 {
  background-color: var(--dark-text2);
}

.dark-text3 {
  background-color: var(--dark-text3);
}

.dark-text4 {
  background-color: var(--dark-text4);
}

.dark-text5 {
  background-color: var(--dark-text5);
```

```
}

.dark-text6 {
  background-color: var(--dark-text6);
}

.dark-text7 {
  background-color: var(--dark-text7);
}

.dark-text8 {
  background-color: var(--dark-text8);
}

.dark-text9 {
  background-color: var(--dark-text9);
}
```

The full example is shown below:

A similar example using `color-mix()` to lighten colors is show in the Codepen example below:

`color-mix()` is available behind a flag in Safari TP and Firefox. It is under development in Chromium (see the Chromestatus entry for more information)

accent-color

Accent colors give you a way to simplify your color palette by giving you a single place to add the acccent color for accent color for user-interface controls generated by the element.

Possible values are:

- **auto**: A UA-chosen color, which should match the accent color of the platform, if any
- <color>: The specific color to use

Browsers that support accent-color currently apply it to the following HTML elements:

- <input type="checkbox">
- <input type="radio">
- <input type="range">
- <progress>

One possible way to use `accent-color` is to create a special class for it and then add the class to the elements we want to use it on.

The CSS indicates both the default accent color and the accent color to use in our custom class.

```
input {
  accent-color: auto;
  display: block;
  width: 30px;
  height: 30px;
}

.custom {
  accent-color: rebeccapurple;
}
```

We can then apply the custom class to the input elements that we want to appear different.

> Check caniuse.comfor a list of browsers that support `accent-color()`.