Explainer: Animating Font-Palette

Authors: Munira Tursunova , Dominik Röttsches
Contributors: Anders Hartvoll Ruud
Tracking Bug: crbug.com/1400620
Last update: 2023-05-14

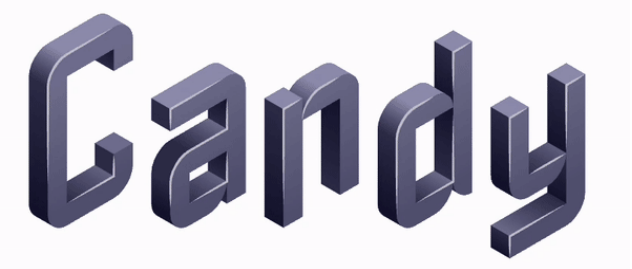
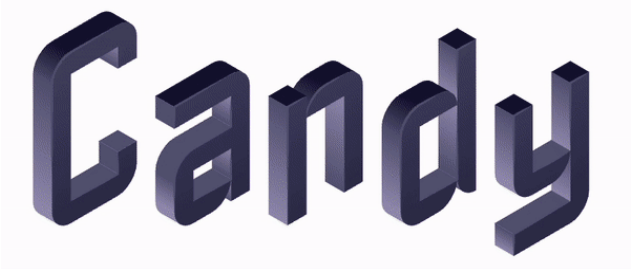
Summary

The CSS font-palette property allows selection of a specific palette used to render a font. For this property, CSS Fonts 4 defines the animation behavior of this property as discrete, which is insufficient to achieve a smooth transition between two selected palettes. Similar to other color transitions in CSS animating the font-palette property should happen by interpolating each of the color record values between start and end palette.

In this explainer, we propose the smooth animation of the font-palette property using declarative CSS. This makes the interpolation of the palette's color values and thus the the use of color fonts on the web more versatile and expressive. See section Examples for more details.

Motivation

Color fonts have vastly improved typographic expressiveness on the web. They provide web authors with a variety of new features, like defining the palette of the font, that will allow them to style glyph appearance flexibly, on top of what's pre-defined by the font. Currently the animation type of the font-palette property is discrete, meaning that there is no smooth transition between the different color values of the font. Animating font palette will allow users to enable smooth transition between the different font palette values, as seen in this example:

Without interpolation:	**With interpolation:**
	

Description	Description
This is how animation of font-palette property works now, it is a discrete jump between base-palette 1 and 3.	This is how animation of font-palette property will work when the feature is implemented, each of the base-palette colors will be interpolated.

We observe positive signals from web-developers on animating font-palette:

- In his article on CSS Tricks, Ollie Williams expressed his interest, describing animatable palettes as a dream coming true
- Scott Kellum (of typetura.com) has also been suggesting it as a useful feature for the web (origin: a Twitter thread and email conversation, Scott in the meantime deleted their Twitter account).

Animating the font-palette property manually is rather complicated: in order for animation to work, web authors need to retrieve information about color records from the font and compute font-palette values for each frame. This workaround solution is demonstrated below for reference.

Defining animation behavior for the CSS font-palette property solves this by enabling a declarative CSS way of achieving a smooth transition. This feature will provide web authors with an easy way to have a smooth transition between font-palette values.

Defining the Transition

The font palette CSS property can be interpolated either

- by base-palette value (i.e. interpolate between all colors of base-palette A and base-palette B that differ)
- or override-colors value (i.e. assuming the base-palette value is identical only the overridden colors need to be interpolated) or
- by the combination of them both.

To compute interpolated values for specific frames, we first need to gather all color records values for the endpoint palettes. The interpolated value for the property is computed by interpolating each color from the palette the same way other color CSS properties are interpolated, compare section 12 of CSS Color Level 4.

Mixing Palettes: the ‘palette-mix()’ Function

We represent mix values during the animation/transition with the following mix function value to represent interpolated font-palette values during the animation progress:

```
palette-mix() = palette-mix(<color-interpolation-method>, [ [normal | light | dark | <palette-identifier>] && <percentage [0,100]>? ]#{2})
```

Example

```
palette-mix(in oklab, --p1 (100% - progress), --p2 progress)
```

Interpolates each color between --p1 and --p2 palettes as it is done for other color properties. The function is used for interpolation, to represent font-palette interpolated value at time *progress* between the palettes “--p1” and “--p2”.

Combination of palettes

We propose font-palette to not be additive. That means in additive animations, animation-composition would always have a replacement behavior, i.e the following animation:

None
```
@keyframes foo {
  from { font-palette: light; animation-composition: add; }
  to { font-palette: dark; animation-composition: accumulate; }
}
```

Would gave the same result as:

None
```
@keyframes foo {
  from { font-palette: light; animation-composition: replace; }
  to { font-palette: dark; animation-composition: replace; }
}
```

Considerations for edge cases

Constraint for the font-family

In order for interpolation to work, both, the `@font-palette-values` rule of the start and end of the animation need to be applicable to the same font (defined in `font-family` descriptor).

According to the CSS Fonts 4 spec, we match all fonts from the `font-family` descriptor of `@font-palette-values` rule against the current matched font. For example, in the code below, if `family-3` is the matched font for the div element then the `--custom` font-palette will be

applied to this element. If `family-4` was matched for the div element then the normal font-palette would be applied to this div element, since `font-family` descriptor for `--custom` font-palette does not contain matched font.

```
None
@font-palette-values --custom {
      font-family: "family-1", "family-2", "family-3",
      base-palette: 3;
}
div {
      font-family: "family-4", "family-3";
      font-palette: --custom;
}
```

Taking all of the above into account, if font-palettes from both of the endpoints of the animation do not have a common font-family, then at least one of them does not contain the matched font-family for the element. For example in the following scenario:

```
None
@font-palette-values --p1 {
      font-family: "family-1", "family-2",
      base-palette: 3;
}
@font-palette-values --p2 {
      font-family: "family-3", "family-4",
      base-palette: 3;
}
@keyframes anim {
      from { font-palette: --p1; }
      to { font-palette: --p2; }
}
div {
      font-family: "family-2", "family-3", "family-5";
      animation: anim 1s infinite;
}
```

If the current matched `font-family` for the div element is equal to:

- `family-2`, then the animation would be between `--p1` and `normal` palettes

- `family-3`, then the animation would be between `normal` and `--p2` palettes
- `family-5`, then the animation would be between `normal` and `normal` palettes

In theory it is possible that the matching font would change after the animation / transition has already started, for such cases we need to stop the animation / transition.

Constraint for the palette

Another constraint of the font-palette animation is that both, starting and ending palettes should contain the same amount of color records. But since we use the same font in both the start and the end of the animation / transition and since every palette in the font's CPAL table has the same amount of color records, this constraint will always be addressed unless the font is broken.

Examples

Animating within one palette

None

```
@font-palette-values --paletteFrom {
      font-family: "Nabla";
      base-palette: 3;
      override-colors: 6 #5e4fa2;
}
@font-palette-values --paletteTo {
      font-family: "Nabla";
      base-palette: 3;
      override-colors: 6 #f79459;
}
@keyframes anim {
  from { font-palette: --paletteFrom; }
  to { font-palette: --paletteTo; }
}
div {
  animation: anim 1s infinite;
}
```

For the following palettes, the color for the record at the index 6 should be changing smoothly.

This is how animation within one palette looks like now, it is discrete jump between the values:

And this how it will looks like when the interpolation for font-palette is implemented:

Animating between palettes

None

```
@font-palette-values --paletteFrom {
      font-family: "Nabla";
      base-palette: 3;
}
@font-palette-values --paletteTo {
      font-family: "Nabla";
      base-palette: 1;
}
@keyframes anim {
  from { font-palette: --paletteFrom; }
  to { font-palette: --paletteTo; }
}
div {
  animation: anim 1s infinite;
}
```

This is how animation between two palette looks like now, it is discrete jump between the palette color records' values:

And this how it will look like when the interpolation for font-palette is implemented:

The Nabla color font is used to illustrate examples: Nabla font at Google Fonts

Imperative workaround in JavaScript

The following example can be found online on Glitch:

- Live: https://animating-font-palette.glitch.me
- Code: https://glitch.com/edit/#!/animating-font-palette.

Currently animating the font palette is possible only by computing palette values for each frame in js script. For that to work, developers would first have to look inside the font to retrieve the color records for the first and the last frame of the animation then compute interpolated values for example using the D3.js library.

Below is an example of computing interpolated values for the Animating between palettes case:

```
JavaScript
var colors = new Array();
```

```
var from_colors = ['#5A5A78', '#141432', '#464664', '#323250',
'#5A5A78','#787896', '#5A5A78', '#787896', '#9696B4','#C8C8D2'];

var to_colors = ['#FF1471', '#780082', '#BE14B4', '#9B1EAF',
'#FF1471', '#FF6B8B', '#FF1471', '#FF6B8B', '#FF9CC2',
'#FFFFFF'];

var steps = 100;

var color_records_cnt = start_colors.length;

for(let i = 0; i < color_records_cnt; i++) {

     var color_interpolator = d3.interpolateHcl(from_colors[i],
to_colors[i]);

     var colors_array = d3.range(0, (1 + 1 / steps), 1 / (steps -
1)).map(function(d) { return d3.rgb(color_interpolator(d)).hex();
});

     colors.push(colors_array);

}
```

In this example, the color array stores the interpolated color records values of the palette on each step. `Start_colors` and `finish_colors` are the color records from the Nabla font's third and first palette respectively.

After computing interpolated values for each step, we set new values for each frame for example by defining them in the override-colors descriptor :

JavaScript

```
function setFontPalette(color) {

     const styleId = 'font-palette-sheet';

     var fontStyleSheet = document.getElementById(styleId);

     var newFontStyleSheet = document.createElement("style");
```

```
      newFontStyleSheet.id = styleId;

      var overrides = [...Array(color.length).keys()].map(i => (i
!= 0) ? (i + ` ` + color[i]) : (i + ` ` + color[i])).join();

      newFontStyleSheet.textContent = `@font-palette-values
--myPalette {

                                        font-family: "Nabla";

                                        base-palette: 3;

                                        override-colors:`

                                        + overrides + `; }`;

      document.head.appendChild(newFontStyleSheet);

      if (fontStyleSheet) {

            fontStyleSheet.parent.removeChild(fontStyleSheet);

      }

}
```