recursive variable font... a deep dive

Recursive is a variable font under development by Arrowtype that caught my attention by the posibilities it provides.

Recursive provides two custom axes: One that moves from monospaced to sans-serif and another one that ranges from standard/linear to more casual/ playful styles.

It also provides three standard axes: Weight, Slant and Italics.

The combination of these 5 axes in 64 pre-defined instance variables make for a very expressive font that can serve many duties in a site or application without adding font files and impacting performance.

In a previous post I looked at using Recursive in the context of Material Design typography and how to add features that are specific to a font rather than to the design system created with Material Design.

But what would it take to use the font outside a Material Design environment.

The first thing to do is to download the font from Github release page. This will give you the WOFF2 file.

I will use Wakamaifondue to generate the CSS that we'll use throughout the document. Load the font (currently Beta 25) on Wakamaifondue and save the generated CSS file to your computer. This generated file coontains all the classes to enable OpenType features and font named instances.

One thing the font doesn't have is the `@font-face` at-rule to actually load the font. The code to actually load the font looks like this.

```
@font-face {
  font-family: 'Recursive'Recursive'   url('../fonts/recursive.woff2') fo
  font-weight: 300 1000;
  font-display: swap;
}
```

There are a few differences to account for the variable font. Instead of `woff2` we use `woff2-variations` to indicate that it's a variable font compressed using WOFF2.

The `font-weight` attribute for variable fonts takes two values, representing the minimum and maximum values for the weight. It geos withou saying that these values should match those available to the font.

Now we're ready to start exploring the font.

OpenType features and Predefined Instances

What makes Wakamaifondue so attractive is that it it does most of the work for you. It creates variables for each OpenType feature the font makes available and for each predefined instance.

You can combine these classes to get the effects that you want. In the example below we have a monospaced font and the Slashed Zero OpenType feature.

```
<pre class="recursive-b025-zero
            recursive-b025-mono-linear">
  All the text in this box is monospaced.

  All the 0 have a diagonal dash on them
  to distinguish them from O
</pre>
```

All the work Wakamaifondue did up front means that developers and content authors can use existing structures adding classes to existing elements or wrapping content in the semantically neutral `div` and `span` elements to accomplish our objectives.

Using variables

Because Recursive uses custom axes, two of the default axes may conflict with each other and there is an issue with inheritance for `font-variation-settings` as documented in Boiling eggs and fixing the variable font inheritance problem we would have to use CSS variables or their Houdini equivalents to control individual axes and them merge them together. You can get more information about how to write CSS using Variable Fonts in Variable Fonts: What web authors need to know issue from Jason Pamental's Responsive Typography Newsletter.

The first block of CSS sets up the default values for each axis and then uses `font-weight` and `font-variation-settings` to configure the default values for the font.

```
:root {
  --vf-mono: 0;
  --vf-casl: 0;
  --vf-wght: 400;
  --vf-slnt: 0;
  --vf-ital: 0.5;
  font-weight: var(--vf-wght);
  /*
    This will define the values for
    the entire document
  */
  font-variation-settings: "MONO" var(--vf-mono)
                           "CASL" var(--vf-casl)
                           "slnt" var(--vf-slnt)
                           "ital" var(--vf-ital);
}
"MONO"
```

in later elements we update only the values that we want to change. For example, if we want element to use the monospaced font we could do something like this:

```
pre, code, pre code {
  --vf-mono: 1;
  font-variation-settings: "MONO" var(--vf-mono)
                           "CASL" var(--vf-casl)
                           "slnt" var(--vf-slnt)
                           "ital" var(--vf-ital);
}
```

Note that since we defined font variation settings on the `:root` element with the default values, this technique forces you to define the values and `font-variation-settings` on all elements that don't use the default.

If we were not using variables then we'd have to use the bare metal approach which we'll discuss below.

Bare Metal

Using variables address the cascade issues with `font-variation-settings` but there may be times when we don't care about the extra work that we need to do.

Let's say, for example that we want to tweak `strong` and b to make it slightly less bold than the default value, we could redefine it like this:

```css
strong,
b {
  font-weight: 600;
  font-variation-settings: "MONO" 0,
    "CASL" 0,
    "slnt" 0,
    "ital" 0.5;
}
```

This technique also allows for customizing the classes and how they use the font. Expanding on the previous example we can use `font-variation-settings` to combine slant and italics axes for the same text rather than using a single axis.

```css
em,
i {
  font-weight: 400;
  font-variation-settings: "MONO" 0,
    "CASL" 0,
    "slnt" -15,
    "ital" 1;
}
```

The block below, from Prism.js default stylesheet incorporates Recursive as the default monospaced font for all code fenced blocks that Prism handles.

```
code[class*="language-"],
pre[class*="language-"] {
  font-family: Recursive, Consolas, Monaco,
    'Andale Mono', 'Ubuntu Mono', monospace;
  font-s'Andale Mono'/* Solarized light base00 */
  color: #657b83;
  text-align: left;
  white-space: pre;
  word-spacing: normal;
  word-break: normal;
  word-wrap: normal;

  /* Variable font settings */
  font-weight: 400;
  font-variation-settings: "MONO" 1,
    "CASL" 0,
    "slnt" 0,
    "ital" 0.5;

  line-height: 1.5;

  "MONO"/* Change tab size to 2 */
  tab-size: 2;

  /* Control hyphenation */
  hyphens: none;
}
```

These are a few examples of what you can do with the font. You can also combine the different approaches to create an even more flexible solution.

Recursive also presents some interesting characteristics that make it fun to play and experiment with. Let’s see how far we can push the technology.