Adding fonts to theme.json for WordPress 6.x

Starting in WordPress 6.1 fonts can be fully defined in `theme.json` including adding variable fonts to the list of available fonts.

The `fontFace` syntax in the `theme.json` file is very similar to what you would use when defining a font using the @font-face at-rule, as described in the @font-face entry in MDN and explained in this CSS Tricks article.

System Fonts

The first example uses fonts that are known to be in the system. These are the default fonts across operating systems and the browser will pick the first font it supports.

The `font-family` attribute contais a list of fonts for this family, the `slug` attribute the name attribute is the human readable name.

You should only use this system if you can be 100% sure that at least one of the fonts in the stack you choose will be available. Otherwise, the default font (`serif`, `sans-serif`, `cursive` or `monospace`) can have unpredictable results based on what fonts the user has available on their system.

```
{
  "typography": {
    "fontFamilies": [{
      "fontFamily": "system-ui,
      \"Segoe UI\",
      Roboto,
      Oxygen-Sans,
      Ubuntu,
      Cantarell,
      \"Helvetica Neue\",
      sans-serif",
      "name": "System Font",
```

```
      "System Font""slug": "system-font"
    }]
  }
}
```

Multiple instances of the same font

The most common web font usage is to provide different font files for each style of font.

`fontWeight` 400 defines the regular font weight.

`fontWeight` 700 defines the bold font weight.

`fontStyle` defines if the font is in italics or not

With a combination of these three attributes, we define four instances for the font:

- Regular (weight 400)
- Italics (weight 400)
- Bold (weight 700)
- Bold-Italics (weight 700)

These are the four basic weights. Depending on the project needs we may define additional combinations that register different weights, styles and stretch values.

```
{
  "typography": {
    "fontFamilies": [{
        "fontFace": [{
            "fontFamily": "DM Sans",
            "DM Sans""fontStretch": "normal",
            "fontStyle": "normal",
            "fontWeight": "400",
            "src"         "file:./assets/fonts/dm-sans/DMSans-Regular.woff
            ]
          },
```

```
          {
            "fontFami"file:./assets/fonts/dm-sans/DMSans-Regular.woff2""fo
        ": ""fontStyle"             "file:./assets/fonts/dm-sans/DMSans-Reg
            ]
          },
          {
            "fontFamily": "DM Sans",
            "fontStretch": "normal",
      "file:./assets/fonts/dm-sans/DMSans-Regular-Italic.woff2""fontFamily"
          "src""fontStretch": "normal",
            "fontStyle": "normal",
            "fontWeight": "700",
            "src": "DM Sans",
            "fontStretch": "normal",
            "fontStyle": "italic",
            "fontWeight": "700","DM Sans""fontFamily"
              "file:./assets/fonts/dm-sans/DMSans-Bold-Italic.woff2"
    "file:./assets/fonts/dm-sans/DMSans-Bold-Italic.woff2": "\"DM Sans\", s
        "\"DM Sans\", sans-serif"   "slug": "dm-sans"
": ""slug""name""src": [
              "file:./assets/fonts/dm-sans/DMSans-Bold-Italic.woff2"
            ]
          }
        ],
        "fontFamily": "\"DM Sans\", sans-serif",
        "name": "DM Sans",
        "slug": "dm-sans"
      }
    ]
  }
}
```


Variable Fonts

The final example uses a single variable font with both slant/italics and weight axes.

I've modified the `fontStyle` attribute to better reflect the values supported in

the font.

Note how the weight is now a range from 200 to 900. When used in CSS this range of values also allow us to use all intermediate values (455, 680 and so on) rather than just multiples of 100 like we could before variable fonts.

This doesn't mean that all instances of the font will have a -10 degree oblique axis but that we are giving the font an option to match when the text calls for italics/oblique text.

```
{
  "typography": {
    "fontFamilies": [{
      "fontFace": [{
        "fontFamily": "Inter",
        "Inter""fontStretch": "auto",
        "fontStyle": "auto",
        "fontWeight": "200 900",
        "src"       "file:./assets/fonts/inter/Inter-VariableFont_slnt,wg
        ]
      }],
      "fontFa"file:./assets/fonts/inter/Inter-VariableFont_slnt,wght.ttf"
      "name": "Inter",
      "slug": "inter"
    }]
  }
}
```

Some variable fonts have broken the font into two separate file, one for regular styles and one for italics.

Using Source Serif Pro as an example we see to `fontFace` declarations where the only difference is that one is set for a normal, non-italics, font and the other one for italics, both referencing different files.

```
{
  "typography": {
    "fontFamilies": [{
```

```
      "fontFace": [{
          "fontFamily": "Source Serif Pro",
          "Source Serif Pro""fontStretch"    "fontStyle": "norm": ""fontSt
            "file:./assets/fonts/source-serif-pro/SourceSerif4Variable-Ron
          ]
        },
        {
          "fontFamily": "Source Seri"file:./assets/fonts/source-serif-pro/
        ": ""fontStretch"      "file:./assets/fonts/source-serif-pro/Sourc
       "file:./assets/fonts/source-serif-pro/SourceSerif4Variable-Italic.t
      "slug": "source-serif-pro"
    }]
  }
}
": ""name""fontFamily""fontFamily": "\"Source Serif Pro\", serif",
      "name": "Source Serif Pro",
      "slug": "source-serif-pro"
    }]
  }
}
```

Drawbacks

Using `theme.json` to declare fonts works fine if all you're doing is adding basic attributes.

However I'm not sure if the style engine work with all attributes available with the `@font-face` at-rule or whether all values of the supported properties will work or not.

Another reason to say with CSS, at least for advanced typography work, is that the current WordPress style engine doesn't support font-variant or its constituent properties:

- font-variant-alternates
- font-variant-caps
- font-variant-east-asian
- font-variant-ligatures

- font-variant-numeric

Other unsupported OpenType-related features are: font-variant-alternates, font-variant-position or the lower-level font-feature-settings

So there is currently no way to really implement these features in WordPress' style engine so I have to default to using CSS if I want to leverage a font's OpenType features.