Loading Multiple Versions of The Same Font

For most of my web work I use Font Face Observer to handle checking that the fonts have loaded.

Using the following `@fontface` declaractions:

```
@font-face {
  font-family: 'Roboto';
'Roboto'url('../../fonts/Roboto-min-VF.woff2') format('woff2');
  font-weight: normal;
  font-style: normal;
  font-display: swap;
}
```

I can use the following script to make sure the font loaded and provide a fallback when it doesn't.

Assuming that `fontfaceobserver.js` is already loaded I use the following script to add classes based on whether the font loaded

```
    const roboto = new FontFaceObserver('Roboto');

    let html = document.documentElement;

    html.classList.add('fonts-loading');

    Promise.all([
      roboto.load(),
    ]).then(() => {
      html.classList.remove('fonts-loading');
      html.classList.add('fonts-loaded');
      console.log('All fonts have loaded.');
    }).catch(() => {
      html.classList.remove('fonts-loading');
```

```
    html.classList.add('fonts-failed');
    console.log('One or more fonts failed to load');
  });
```

When I use multiple fonts I add new `FontFaceObserver` objects as variablles and to the `Promise.all` array.

But what happens when you load variants of the same font, like so:

```
@font-face {
  font-family: 'Work Sans'Work Sans'url'../../fonts/WorkSans-Regular.woff2
  font-weight: normal;
  font-style: normal;
  font-display: swap;
}

@font-face {
  font-family: '') format('@font-facel('../../fonts/WorkSans-Bold.woff2')
  font-weight:') format('url('../../fonts/WorkSans-Bold.woff2') format('tr
  font-weight: bold;
  font-style: normal;
  font-display: swap;
}
```

Until recently I had not realized that there was a second parameter that lists the attributes of the font that we want to download.

In the example below, the `workBold` definition includes the second parameter with the weight of the font we're using in the second declaration.

The second parameter is an object with one or more of `weight`, `style`, and `stretch` and it must match one of the font declarations you use to load the fonts.

```
  const work = new FontFaceObserver('Work Sans');
  const workBold = new FontFaceObserver('Work Sans', {
    weight: 'bold'
  });
```

```
let html = document.documentElement;

html.classList.add('fonts-loading');

Promise.all([
  work.load(),
  workBold.load(),
]).then(() => {
  html.classList.remove('fonts-loading');
  html.classList.add('fonts-loaded');
  console.log('All fonts have loaded.');
}).catch(() => {
  html.classList.remove('fonts-loading');
  html.classList.add('fonts-failed');
  console.log('One or more fonts failed to load');
});
```

Using this technique you can use Font Face Observer to load multiple instances of the same font without having to give them different names.

Links

- Web Font Loading Patterns
- Faster Font Loading with Font Events
- Delivering Web Fonts
- How We Load Web Fonts Progressively