@font-face font-display property

There are times when the current solutions are not enough. Most of my non-Wordpress web content uses fontfaceobserver to handle web fonts and how they display on the page.

We've already used `@font-face` to load the fonts using CSS. The script below creates one instance of Fontfaceobserver for each of the fonts we've loaded using CSS.

Each FontFaceObserver instance has several methods that we'll use later in the script.

We create a shortcut to `document.documentElement` to save ourselves typing.

The last command in this section will add a `fonts-loading` class to the page's HTML element.

```
const mono = new FontFaceObserver('notomono-regular');
const sans = new FontFaceObserver('notosans-regular');
const italic = new FontFaceObserver('notosans-italics');
const bold = new FontFaceObserver('notosans-bold');
const bolditalic = new FontFaceObserver('notosans-bolditalic');

let html = document.documentElement;

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

We use the load method of FontFaceObserver to create a promise for each of the fonts we are using and defined in the previous block. We then use `Promise.all()` to create an array of promises.

If all the promises resolve then we remove the `fonts-loading` CSS class and replace it with `fonts-loaded`.

If any of the promises rejects then `Promise.all()` will reject as well. In that

case we replace `fonts-loaded` with `fonts-failed`.

```
Promise.all([ mono.load(), sans.load(), italic.load(),
bolditalic.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')
});
```

The classes we added in Javascript are also used in CSS to style the content. The first class is the default and uses system fonts so we don't have to wait for web fonts to download.

The second class will only match when web fonts fail to load. It uses the same stack as the default.

The third class will match when web fonts load successfully and use the web fonts we just verified.

```
/* Default body style */
body {
  font-family: Verdana, sans-serif;
  font-size: 16px;
  line-height: 1.375;
}

/* This will match if the fonts failed to load. It is identical to the de
.fonts-failed body {
  font-family: Verdana, sans-serif;
  font-size: 16px;
  line-height: 1.375;
}

/* This will match when fonts load successfully */
```

```
.fonts-loaded body {
  font-family: notosans-regular, verdana, sans-serif;
  font-size: 16px;
  line-height: 1.375;
}
```

The `font-display` property provides predefined behaviors for web font downloading. It can take one of four possible values:

- **auto**: The font display strategy is defined by the user agent. This means that it will do whatever it does normally
- **block**: Gives the font face a short block period and an infinite swap period
- **swap**: Gives the font face no block period and an infinite swap period
- **fallback**: Gives the font face an extremely small block period and a short swap period
- **optional**: gives the font face an extremely small block period and no swap period

We need to further define what we mean by block, swap and failure periods of font loading and what's their sequence:

1. The first period is the **block period**. During this period, if the font face is not loaded, any element attempting to use it must instead render with an invisible fallback font face causing a Flash Of Invisible Text. If the font face successfully loads during the block period, the font face is then used normally.
2. The **swap period** occurs immediately after the block period. During this period, if the font face is not loaded, any element attempting to use it must instead render with a fallback font face. If the font face successfully loads during the swap period, the font face is then used normally.
3. The **failure period** occurs immediately after the swap period. If the font face is not yet loaded when this period starts, it's marked as a failed load, causing normal font fallback. Otherwise, the font face is used normally

Understanding these periods means you can use font-display to decide how your font should render depending on whether or when it was downloaded. There may be situations where performance is you most important concern so a more agressive strategy is necessary but there may be other instances where we want the content to display with our web fonts and performances is a secondary concern.

If all browsers worked the same way we wouldn't have any problems and we could use `font-display` with no major issue. But not all browsers work the same; The table below shows how browsers behave ragarding fonts download.

Browser	Timeout	Fallback	Swap
Chrome 35+	3 seconds	Yes	Yes
Opera	3 seconds	Yes	Yes
Firefox	3 seconds	Yes	Yes
Internet Explorer	0 seconds	Yes	Yes
Edge	0 seconds	Yes	Yes
Safari	No timeout	N/A	N/A

- Chrome and Firefox have a three second timeout after which the text is shown with the fallback font. If the font manages to download, then eventually a swap occurs and the text is re-rendered with the intended font.
- Internet Explorer has a zero second timeout which results in immediate text rendering. If the requested font is not yet available, a fallback is used, and text is re-rendered later once the requested font becomes available.
- I've assumed that Edge works the same as IE. I've also asked on Twitter to see if this assumption is correct.
- Safari has no timeout behavior (or at least nothing beyond a baseline network timeout).

So, which one do we use?

Until browsers smooth out their differences or developers can take tighter control I'd say that solutions like Font Face Observer are still the best solution out there.

Links and resources

- CSS Tricks
- Google Developers
- MDN