Controlling Font Performance with font-display

By Rob Dodson

Rob is a contributor to Web**Fundamentals**

Deciding the behavior for a web font as it is loading can be an important performance tuning technique. The new font-display descriptor for `@font-face` lets developers decide how their web fonts will render (or fallback), depending on how long it takes for them to load.

Differences in font rendering today

Web Fonts give developers the ability to incorporate rich typography into their projects with the tradeoff that if the user does not already posses a typeface the browser must spend some time downloading it. Because networks can be flaky, this download time has the potential to adversely affect the user's experience. After all, no one's going to care how pretty your text is if it takes an inordinate amount of time to display!

To mitigate some of the risk of a slow font download, most browsers implement a timeout after which a fallback font will be used. This is a useful technique but unfortunately browsers differ on the actual implementation.

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
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.
- Safari has no timeout behavior (or at least nothing beyond a baseline network timeout).

To make matters worse, developers have limited control in deciding how these rules will affect their application. A performance minded developer may prefer to have a faster initial experience that uses a fallback font, and only leverage the nicer web font on subsequent visits after it has had a chance to download. Using tools like the Font Loading API, it may be possible to override some of the default browser behaviors and achieve performance gains, but it comes at the cost of needing to write non-trivial amounts of JavaScript which must then be inlined into the page or requested from an external file, incurring additional HTTP latency.

To help remedy this situation the CSS Working Group has proposed a new `@font-face` descriptor, `font-display`, and a corresponding property for controlling how a downloadable font renders before it is fully loaded.

Font download timelines

Similar to the existing font timeout behaviors that some browsers implement today, `font-display` segments the lifetime of a font download into three major periods.

1. The first period is the **font 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. If the font face successfully loads during the block period, the font face is then used normally.
2. The **font swap period** occurs immediately after the font 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 **font failure period** occurs immediately after the font 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.

Which font-display is right for you?

To work with the `font-display` descriptor, add it your `@font-face` at-rules:

```
@font-face {
  font-family: 'Arvo';
  font-display: auto;
  src: local('Arvo'), url(https://fonts.gstatic.com/s/arvo/v9/rC7kKhY-eUDY-ucISTI
}
```

`font-display` currently supports the following range of values `auto | block | swap | fallback | optional`.

auto

auto uses whatever font display strategy the user-agent uses. Most browsers currently have a default strategy similar to **block**.

block

block gives the font face a short block period (3s is recommended in most cases) and an infinite swap period. In other words, the browser draws "invisible" text at first if the font is not loaded, but swaps the font face in as soon as it loads. To do this the browser creates an anonymous font face with metrics similar to the selected font but with all glyphs containing no "ink." This value should only be used if rendering text in a particular typeface is required for the page to be useable.

swap

swap gives the font face a zero second block period and an infinite swap period. This means the browser draws text immediately with a fallback if the font face isn’t loaded, but swaps the font face in as soon as it loads. Similar to **block**, this value should only be used when rendering text in a particular font is important for the page, but rendering in any font will still get a correct message across. Logo text is a good candidate for **swap** since displaying a company’s name using a reasonable fallback will get the message across but you’d eventually use the official typeface.

fallback

fallback gives the font face an extremely small block period (100ms or less is recommended in most cases) and a short swap period (three seconds is recommended in most cases). In other words, the font face is rendered with a fallback at first if it's not loaded, but the font is swapped as soon as it loads. However, if too much time passes, the fallback will be used for the rest of the page's lifetime. **fallback** is a good candidate for things like body text where you'd like the user to start reading as soon as possible and don't want to disturb their experience by shifting text around as a new font loads in.

optional

optional gives the font face an extremely small block period (100ms or less is recommended in most cases) and a zero second swap period. Similar to **fallback**, this is a good choice for when the downloading font is more of a "nice to have" but not critical to the experience. The **optional** value leaves it up to the browser to decide whether to initiate the font download, which it may choose not to do or it may do it as a low priority depending on what it thinks would be best for the user. This can be beneficial in situations where the user is on a weak connection and pulling down a font may not be the best use of resources.

Browser support

`font-display` is currently behind the Experimental Web Platform Features flag in desktop Chrome 49, and is shipping in Opera and Opera for Android.

Demo

Check out the sample to give `font-display` a shot. For performance minded developers it can be one more useful tool in your toolbelt!

Last updated July 2, 2018.