This repository Search Pull requests Issues Gist

TamsynM / webfontloader
forked from typekit/webfontloader

Unwatch 1 Star 0 Fork 447

Code Pull requests 0 Pulse Graphs Settings

Web Font Loader gives you added control when using linked fonts via **@font-face**. — Edit

1,131 commits **2** branches **102** releases **38** contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is even with typekit:master. Pull request Compare

bramstein 1.6.26 Latest commit `33bec9d` 26 days ago

bin	bind demos to localhost because that's what some services require	6 years ago
lib	Release 1.6.26	26 days ago
spec	Release 1.6.17	7 months ago
src	Fix Google activating before latin-ext load	27 days ago
tools	Fix auto dependency loading.	6 months ago
.gitignore	add gem packaging tools	6 years ago
.travis.yml	Try the strategy outlined in: https://mediocre.com/forum/topics/phant…	a year ago
CHANGELOG	Release 1.6.26	26 days ago
CONTRIBUTING.md	Add ability to specify modules when compiling webfont.js. Also update…	3 years ago
Gemfile	Use secure Rubygems.	a year ago
LICENSE	Remove appendix from LICENSE and add Copyright and License section to…	3 years ago
README.md	Merge pull request #320 from parterburn/patch-1	27 days ago
Rakefile	Update Rakefile to automatically update package.json and publish to npm.	a year ago
bower.json	Simplify bower.json.	5 months ago
browsers.json	Increase timeout for IE7.	3 years ago
externs.js	Add UMD wrapper.	a year ago
package.json	1.6.26	26 days ago
webfontloader.gemspec	Release 1.6.26	26 days ago
webfontloader.js	Release 1.6.26	26 days ago

README.md

Web Font Loader

Web Font Loader gives you added control when using linked fonts via `@font-face`. It provides a common interface to loading fonts regardless of the source, then adds a standard set of events you may use to control the loading experience. The Web Font Loader is able to load fonts from Google Fonts, Typekit, Fonts.com, and Fontdeck, as well as self-hosted web fonts. It is co-developed by Google and Typekit.

build passing

Contents

- Get started
- Configuration

Get Started

To use the Web Font Loader library, just include it in your page and tell it which fonts to load. For example, you could load fonts from Google Fonts using the Web Font Loader hosted on Google Hosted Libraries using the following code.

```
<script src="https://ajax.googleapis.com/ajax/libs/webfont/1.6.16/webfont.js"></script>
<script>
  WebFont.load({
    google: {
      families: ['Droid Sans', 'Droid Serif']
    }
  });
</script>
```

Alternatively, you can link to the latest `1.x` version of the Web Font Loader by using `https://ajax.googleapis.com/ajax/libs/webfont/1/webfont.js` as the `script` source. Note that the version in this url is less specific. It will always load the latest `1.x` version, but it also has a shorter cache time to ensure that your page gets updates in a timely manner. For performance reasons, we recommend using an explicit version number (such as `1.6.16`) in urls when using the Web Font Loader in production. You can manually update the Web Font Loader version number in the url when you want to adopt a new version.

Web Font Loader is also available on the jsDelivr & CDNJS CDNs.

It is also possible to use the Web Font Loader asynchronously. For example, to load Typekit fonts asynchronously, you could use the following code.

```
<script>
   WebFontConfig = {
      typekit: { id: 'xxxxxx' }
   };

   (function(d) {
      var wf = d.createElement('script'), s = d.scripts[0];
      wf.src = 'https://ajax.googleapis.com/ajax/libs/webfont/1.6.16/webfont.js';
      s.parentNode.insertBefore(wf, s);
   })(document);
</script>
```

Using the Web Font Loader asynchronously avoids blocking your page while loading the JavaScript. Be aware that if the script is used asynchronously, the rest of the page might render before the Web Font Loader is loaded and executed, which can cause a Flash of Unstyled Text (FOUT).

The FOUT can be more easily avoided when loading the Web Font Loader synchronously, as it will automatically set the `wf-loading` class on the HTML element as soon as `Webfont.load` has been called. The browser will wait for the script to load before continuing to load the rest of the content, FOUT is avoided.

Web Font Loader is also available on npm as a CommonJS module. Just `npm install webfontloader` and then require it in your code.

```
  var WebFont = require('webfontloader');
```

```
WebFont.load({
  google: {
    families: ['Droid Sans', 'Droid Serif']
  }
});
```

Configuration

The Web Font Loader configuration is defined by a global variable named `WebFontConfig`, or passed directly to the `WebFont.load` method. It defines which fonts to load from each web font provider and gives you the option to specify callbacks for certain events. When using the asynchronous approach, you must define the global variable `WebFontConfig` before the code that loads the Web Font Loader (as in the example above).

Events

Web Font Loader provides an event system that developers can hook into. It gives you notifications of the font loading sequence in both CSS and JavaScript.

- `loading` - This event is triggered when all fonts have been requested.
- `active` - This event is triggered when the fonts have rendered.
- `inactive` - This event is triggered when the browser does not support linked fonts *or* if none of the fonts could be loaded.
- `fontloading` - This event is triggered once for each font that's loaded.
- `fontactive` - This event is triggered once for each font that renders.
- `fontinactive` - This event is triggered if the font can't be loaded.

CSS events are implemented as classes on the `html` element. The following classes are set on the `html` element:

```
.wf-loading
.wf-active
.wf-inactive
.wf-<familyname>-<fvd>-loading
.wf-<familyname>-<fvd>-active
.wf-<familyname>-<fvd>-inactive
```

The `<familyname>` placeholder will be replaced by a sanitized version of the name of each font family. Spaces and underscores are removed from the name, and all characters are converted to lower case. For example, `Droid Sans` becomes `droidsans`. The `<fvd>` placeholder is a Font Variation Description. Put simply, it's a shorthand for describing the style and weight of a particular font. Here are a few examples:

```
/* n4 */
@font-face { font-style: normal; font-weight: normal; }

/* i7 */
@font-face { font-style: italic; font-weight: bold; }
```

Keep in mind that `font-weight: normal` maps to `font-weight: 400` and `font-weight: bold` maps to `font-weight: 700`. If no style/weight is specified, the default `n4` (`font-style: normal; font-weight: normal;`) will be used.

If fonts are loaded multiple times on a single page, the CSS classes continue to update to reflect the current state of the page. The global `wf-loading` class is applied whenever fonts are being requested (even if other fonts are already active or inactive). The `wf-inactive` class is applied only if none of the fonts on the page have rendered. Otherwise, the `wf-active` class is applied (even if some fonts are inactive).

JavaScript events are implemented as callback functions on the `WebFontConfig` configuration object.

```
WebFontConfig = {
  loading: function() {},
  active: function() {},
  inactive: function() {},
  fontloading: function(familyName, fvd) {},
  fontactive: function(familyName, fvd) {},
  fontinactive: function(familyName, fvd) {}
};
```

The `fontloading`, `fontactive` and `fontinactive` callbacks are passed the family name and font variation description of the font that concerns the event.

It is possible to disable setting classes on the HTML element by setting the `classes` configuration parameter to `false` (defaults to `true`).

```
WebFontConfig = {
  classes: false
};
```

You can also disable font events (callbacks) by setting the `events` parameter to `false` (defaults to `true`).

```
WebFontConfig = {
  events: false
};
```

If both events and classes are disabled, the Web Font Loader does not perform font watching and only acts as a way to insert @font-face rules in the document.

Timeouts

Since the Internet is not 100% reliable, it's possible that a font will fail to load. The `fontinactive` event will be triggered after 5 seconds if the font fails to render. If *at least* one font successfully renders, the `active` event will be triggered, else the `inactive` event will be triggered.

You can change the default timeout by using the `timeout` option on the `WebFontConfig` object.

```
WebFontConfig = {
  google: {
    families: ['Droid Sans']
  },
  timeout: 2000 // Set the timeout to two seconds
};
```

The timeout value should be in milliseconds, and defaults to 3000 milliseconds (3 seconds) if not supplied.

Iframes

Usually, it's easiest to include a copy of Web Font Loader in every window where fonts are needed, so that each window manages its own fonts. However, if you need to have a single window manage fonts for multiple same-origin child windows or iframes that are built up using JavaScript, Web Font Loader supports that as well. Just use the optional `context` configuration option and give it a reference to the target window for loading:

```
WebFontConfig = {
  google: {
    families: ['Droid Sans']
  },
  context: frames['my-child']
};
```

This is an advanced configuration option that isn't needed for most use cases.

Modules

Web Font Loader provides a module system so that any web font provider can contribute code that allows their fonts to be loaded. This makes it possible to use multiple web font providers at the same time. The specifics of each provider currently supported by the library are documented here.

Adobe Edge Web Fonts

When using Adobe Edge Web Fonts, you can use the `typekit` module by passing in a catenated list of fonts in the `id` parameter and set the `api` parameter to point to the Edge Web Fonts URL.

```
WebFontConfig = {
  typekit: {
    id: 'adamina;advent-pro',
    api: '//use.edgefonts.net'
  }
};
```

Custom

To load fonts from any external stylesheet, use the `custom` module. Here you'll need to specify the font family names you're trying to load, and optionally the url of the stylesheet that provides the `@font-face` declarations for those fonts.

You can specify a specific font variation or set of variations to load and watch by appending the variations separated by commas to the family name separated by a colon. Variations are specified using FVD notation.

```
WebFontConfig = {
  custom: {
    families: ['My Font', 'My Other Font:n4,i4,n7'],
    urls: ['/fonts.css']
  }
};
```

In this example, the `fonts.css` file might look something like this:

```
@font-face {
  font-family: 'My Font';
  src: ...;
}
@font-face {
  font-family: 'My Other Font';
  font-style: normal;
  font-weight: normal; /* or 400 */
  src: ...;
}
@font-face {
  font-family: 'My Other Font';
  font-style: italic;
  font-weight: normal; /* or 400 */
  src: ...;
}
@font-face {
  font-family: 'My Other Font';
  font-style: normal;
  font-weight: bold; /* or 700 */
  src: ...;
}
```

If your fonts are already included in another stylesheet you can also leave out the `urls` array and just specify font family names to start font loading. As long as the names match those that are declared in the `families` array, the proper loading classes will be applied to the html element.

```
<script src="https://ajax.googleapis.com/ajax/libs/webfont/1.5.10/webfont.js"></script>
<script>
  WebFont.load({
    custom: {
      families: ['My Font']
    }
  });
</script>

<style type="text/css">
  @font-face {
    font-family:"My Font";
    src:url("assets/fonts/my_font.woff") format("woff");
  }
</style>
```

The custom module also supports customizing the test strings that are used to determine whether or not a font has loaded. This can be used to load fonts with custom subsets or glyphs in the private use unicode area.

```
WebFontConfig = {
  custom: {
    families: ['My Font'],
    testStrings: {
      'My Font': '\uE003\uE005'
    }
  }
};
```

Tests strings should be specified on a per font basis and contain at least one character. If not specified the default test string (`BESbswy`) is used.

Fontdeck

To use the Fontdeck module, specify the ID of your website. You can find this ID on the website page within your account settings.

```
WebFontConfig = {
  fontdeck: {
    id: 'xxxxx'
  }
};
```

Fonts.com

When using Fonts.com web fonts specify your Project ID.

```
WebFontConfig = {
  monotype: {
    projectId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx',
    version: 12345, // (optional, flushes the CDN cache)
    loadAllFonts: true //(optional, loads all project fonts)
  }
};
```

The Fonts.com module has an optional `version` option which acts as a cache-buster, optional `loadAllFonts` loads all project fonts. By default, Fonts.com module loads only fonts used on the page.

Google

Using Google's Font API, name the font families you'd like to load. You can use the same syntax as in the Font API to specify styles. Please note that the Google module does not support the FVD syntax that is used in the custom module.

```
WebFontConfig = {
  google: {
    families: ['Droid Sans', 'Droid Serif:bold']
  }
};
```

Sometimes the font you requested doesn't come in the default weight (e.g. 400) and you need to explicitly request the variation you want for font events to work (e.g. `300`, `700`, etc.):

```
WebFontConfig = {
  google: {
    families: ['Open Sans Condensed:300,700']
  }
};
```

If you need to specify character subsets other than the default (e.g.: greek script in addition to latin), you must append the subset string to the requested family string after a colon. The subset string should follow the Google documentation (subset names separated by commas):

```
WebFontConfig = {
  google: {
    families: ['Open Sans Condensed:300,700:latin,greek']
  }
};
```

You can also supply the `text` parameter to perform character subsetting:

```
WebFontConfig = {
  google: {
    families: ['Droid Sans', 'Droid Serif'],
    text: 'abcdefghijklmnopqrstuvwxyz!'
  }
};
```

The `text` subsetting functionality is only available for the Google module.

Typekit

When using Typekit, specify the Kit to retrieve by its ID. You can find the Kit ID within Typekit's Kit Editor interface.

```
WebFontConfig = {
  typekit: {
    id: 'xxxxxx'
  }
};
```

FYI: Typekit's own JavaScript is built using the Web Font Loader library and already provides all of the same font event functionality. If you're using Typekit, you should use their embed codes directly unless you also need to load web fonts from other providers on the same page.

Browser Support

Every web browser has varying levels of support for fonts linked via `@font-face`. Web Font Loader determines support for web fonts is using the browser's user agent string. The user agent string may claim to support a web font format when it in fact does not. This is especially noticeable on mobile browsers with a "Desktop" mode, which usually identify as Chrome on Linux. In this case a web font provider may decide to send WOFF fonts to the device because the real desktop Chrome supports it, while the mobile browser does not. The Web Font Loader is not designed to handle these cases and it defaults to believing what's in the user agent string. Web font providers can build on top of the basic Web Font Loader functionality to handle these special cases individually.

If Web Font Loader determines that the current browser does not support `@font-face`, the `inactive` event will be triggered.

When loading fonts from multiple providers, each provider may or may not support a given browser. If Web Font Loader determines that the current browser can support `@font-face`, and *at least* one provider is able to serve fonts, the fonts from that provider will be loaded. When finished, the `active` event will be triggered.

For fonts loaded from supported providers, the `fontactive` event will be triggered. For fonts loaded from a provider that *does not* support the current browser, the `fontinactive` event will be triggered.

For example:

```
WebFontConfig = {
  providerA: 'Family1',
  providerB: 'Family2'
};
```

If `providerA` can serve fonts to a browser, but `providerB` cannot, The `fontinactive` event will be triggered for `Family2`. The `fontactive` event will be triggered for `Family1` once it loads, as will the `active` event.

Copyright and License

Web Font Loader Copyright (c) 2010 Adobe Systems Incorporated, Google Incorporated.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at