Generic font-families in Chromium

Author: Frédéric Wang <fwang@igalia.com>

Last Updated: 2022/04/19

TL;DR CSS defines generic font families such as `serif` or `monospace` so that authors can specify system fonts that depend on user preferences, content language, platform settings, etc. For historical reasons, the current implementation of generic font families in chromium-based browsers is complex, inconsistent and incomplete. However, code refactoring from 2021 made it easier to introduce new generic font families. The present design document focuses on how to perform that task, with details of the different parts of the code involved.

Implementation status

Summary

The following table provides an overview of the implementation status for each generic family name, with references to the relevant part of the code base. Further details are provided in other sections below.

		Standard font family	`serif`	`sans-serif`	`cursive`	`fantasy`	`monospace`	`system-ui`	`emoji`	`math`	`fangsong,ui-`
CSS Fonts spec & tests	level 3	Corresponds to the `initial` and fallback fonts.	✅	✅	✅	✅	✅	❌	❌	❌	❌
	level 4		✅	✅	✅	✅	✅	✅	✅	✅	✅
	WPT tests	At least one test for each keyword thanks to font-family-keywords.js ; in general it may be difficult to write interoperable tests as browsers have specific font selection mechanisms. Internal tests are available to test font selection from settings but one may need a special API.									
Shipped?	runtime_enabled_features.json5	✅	✅	✅	✅	✅	✅	✅	❌	🚧 (Under `CSSFontFamilyMath`)	❌
CSS parsing & serialization	css_value_keywords.json5|font-family	The `initial` value is converted to a `FontDescription` with	✅	✅	✅	✅	✅	✅	❌	✅	❌
	computed_style_utils.cc|IdentifierForFamily										

		Standard font family	serif	sans-serif	cursive	fantasy	monospace	system-ui	emoji	math	fangsong,ui-
	css_parsing_utils.cc|ConsumeGenericFamily	kStandardFamily generic family type and a default standard font family, see FontBuilder's SetInitial, CreateInitialFont, InitialFamilyDescription, InitialGenericFamily, SetFamilyDescription and FontBuilder::SetFamilyDescription Additionally, a fallback standard generic font family is built FontFallbackList::GetFontData.									
Internal font data	style_builder_converter.cc|ConvertFontFamilyName		✅	✅	✅	✅	✅	✅	❌	✅	❌
	FontFamily::InferredTypeFor										
	font_family_names.json5										
Legacy GenericFamilyType	font_description.h|GenericFamilyType		❌	❌	❌	❌	❌	✅ These names are not part of GenericFamilyType.			
	FontBuilder::GenericFontFamilyName										
	style_builder_converter.cc|ConvertGenericFamily										
Font selection for generic families	FontSelector::FamilyNameFromSettings	✅	✅	✅	✅	✅	✅	❌	❌	✅	❌
	FontCache::GetGenericFamilyNameForScript	family_name is handled specially for serif and monospace. Other than that, this method relies really only on font_description.Locale().									
	alternate_font_family.h|GetFallbackFontFamily	This somewhat relies on GenericFamilyType for font fallback.									
	font_fallback_win.cc										
	font_matcher_mac.mm|MatchNSFontFamily	The generic name system-ui is handled specially.									
	FontCache::SystemFontPlatformData										
Blink's page settings	generic_font_family_settings.h	✅	✅	✅	✅	✅	✅	❌	❌	✅	❌
	generic_font_family_settings.cc										
	🧪generic_font_family_settings_test.cc										
	internal_settings.idl										

		Standard font family	serif	sans-serif	cursive	fantasy	monospace	system-ui	emoji	math	fangsong,ui-
	internal_settings.h										
	internal_settings.cc										
	🧪font-family-*.html										
	🧪generic-font-families.js										
Blink's webview settings	web_settings.h	✅	✅	✅	✅	✅	✅	❌	❌	✅	❌
	web_settings_impl.h										
	web_settings_impl.cc										
Blink's preferences	WebView::ApplyWebPreferences	✅	✅	✅	✅	✅	✅	❌	❌	✅	❌
	web_preferences.h										
	web_preferences.cc										
	web_preferences_mojom_traits.cc										
	web_preferences.mojom										
	🧪font_preferences_browsertest.cc										
Chromium's preferences	ChromeContentBrowserClient::OverrideWebkitPrefs	✅	✅	✅	✅	✅	✅	❌	❌	✅	❌
	locale_settings_chromiumos.grd										
	locale_settings_google_chromeos.grd										
	locale_settings_linux.grd										
	locale_settings_mac.grd										
	locale_settings_win.grd										
	startup_resources_mac.txt										
	startup_resources_win.txt										
	pref_font_webkit_names.h										

		Standard font family	serif	sans-serif	cursive	fantasy	monospace	system-ui	emoji	math	fangsong,ui-
	pref_names.h										
	pref_names.cc										
	prefs_tab_helper.cc										
	🧪prefs_tab_helper.cc										
Blink inspector / Chrome DevTools Protocol	InspectorOverlayAgent::LoadOverlayPageResource	✅	✅	✅	✅	✅	✅	❌	❌	✅	❌
	InspectorPageAgent::setFontFamilies										
	browser_protocol.pdl										
	Devtools-frontend (automatically generated files)										
	🧪set-font-families-generic-names.js										
chrome.fontSettings API for browser extensions	font_settings_json|GenericFamily	✅	✅	✅	✅	✅	✅	❌	❌	✅	❌
	🧪api_test/font_settings										
Android WebView settings API	ContentSettingsAdapter.java	✅	🚧 They can be set but actually have no effect.					❌	❌	❌	❌
	aw_settings.cc										
	AwSettings.java										
	🧪AwSettingsTest.java		❌	❌	❌	❌	❌	❌	❌	❌	❌
Preference User Interface chrome://settings/fonts	appearance_fonts_page.html	✅	✅	✅	❌	❌	✅	❌	❌	✅	❌
	SettingsAppearanceFontsPageElement										
	GetAllowlistedKeys										
	settings_utils_mac.mm										
	settings_localized_strings_provider.cc										
	settings_strings.grdp										

		Standard font family	serif	sans-serif	cursive	fantasy	monospace	system-ui	emoji	math	fangsong,ui-
	🧪appearance_fonts_page_test.ts										

Explanation

The CSS fonts module defines generic font families. The names from level 3 (`serif`, `sans-serif`, `cursive`, `fantasy`) have been implemented for a long time. A "standard font family" was also implemented and is still being used for initial/fallback fonts as well as for the proprietary keyword `-webkit-body`. The `system-ui` keyword was implemented in browsers in 2016, with the goal to replace proprietary names such as `BlinkMacSystemFont` or `-apple-system`. Finally, the math keyword is being implemented in Chromium under the Under `CSSFontFamilyMath` runtime flag and at the same time as MathML Core, so that appropriate default user-configurable font is used for math layout. As of April 2022, other generic family keywords are not implemented and are treated as any non-generic font-family names.

Blink's internal representation

The first implementation step is about supporting CSS parsing and serialization for the keywords corresponding to these generic family names with conversion to/from internal data structures. This is more or less straightforward although it's worth noting the following points:

- Internally, the `font-family` property is represented by a `FontDescription` structure with an associated linked list of font families. Based on the original WebKit implementation, the *last* generic font family of the list is converted to an enum `GenericFamilyType` stored on the `FontDescription`. We probably want to get rid of this enum and make it use the first generic family instead.
- For the initial font, we essentially have a `FontDescription` with a `GenericFamilyType` set to `kStandardFamily` with a linked family list whose only item has its font name read from the standard font family preference. Additionally when calculating fallback font a fake family list is created with a single generic family of name `-webkit-standard`. That way, it can be resolved later to the standard font family.
- Proprietary keywords `BlinkMacSystemFont` (for macOS only) and `-webkit-body` are supported and handled specially. The former is mapped to `system-ui` when building the linked list while the latter is stored like any generic family keyword and is instead handled during font resolution. Eventually, we would like to get rid of these proprietary keywords.

Blink's font selection for generic families

The main place where generic font families are resolved to concrete system names is in FontSelector::FamilyNameFromSettings. The following cases are accepted:

- `FontDescription`'s `GenericFamilyType` is `kStandardFamily`, corresponding to the `initial` font case.
- `FontDescription`'s `GenericFamilyType` is `kWebkitBodyFamily`, corresponding to the proprietary `-webkit-body` keyword.
- `FontFamily` is generic, corresponding to the CSS keywords ; as well as the internal `-webkit-standard` used for the fallback case.

On non-Android builds, the font selection of `FontSelector::FamilyNameFromSettings` is easy to describe: the CSS keywords will return the corresponding font value from `Settings` while the `initial`, `-webkit-body` and fallback case will return the standard font setting.

On Android builds, the summary is that the method does not rely on user settings and in most cases just returns the CSS keyword, except for non-monospace and non-serif Asian scripts. In details:

1. `FontSelector::FamilyNameFromSettings` delegates to GetGenericFamilyNameForScript which in addition to the `FontDescription` parameter accepts a family name (corresponding either to the CSS keyword or `-webkit-standard` for the `initial`, `-webkit-body` and fallback cases) and a generic family name fallback (corresponding either to the CSS keyword or to the one obtained by GetFallbackFontFamily for the `initial`, -webkit-body and fallback cases).
2. `GetGenericFamilyNameForScript` just returns the family name if it's `monospace` or `serif`. For some Asian scripts used by `FontDescription` locale, it will rely on Skia to find a font containing special asian characters. Otherwise, it will just return the specified generic family name fallback.

Besides `FontSelector::FamilyNameFromSettings`, there are other places involved in the font selection mechanism. The present document focuses on `FontSelector::FamilyNameFromSettings` which is relevant for the new generic keywords. However one can also mention:

- Some stuff like the `initial` font or `BlinkMacSystemFont` are handled when building the internal font data structure, see the previous section.
- `system-ui` is treated differently than the other generic keyword and preference UI/API is not needed.
- The font fallback mechanism on Windows relies on the `GenericFamilyType` in some places.
- FontCache::GetLastResortFallbackFont performs more attempts to find a system font.

Preferences and public APIs

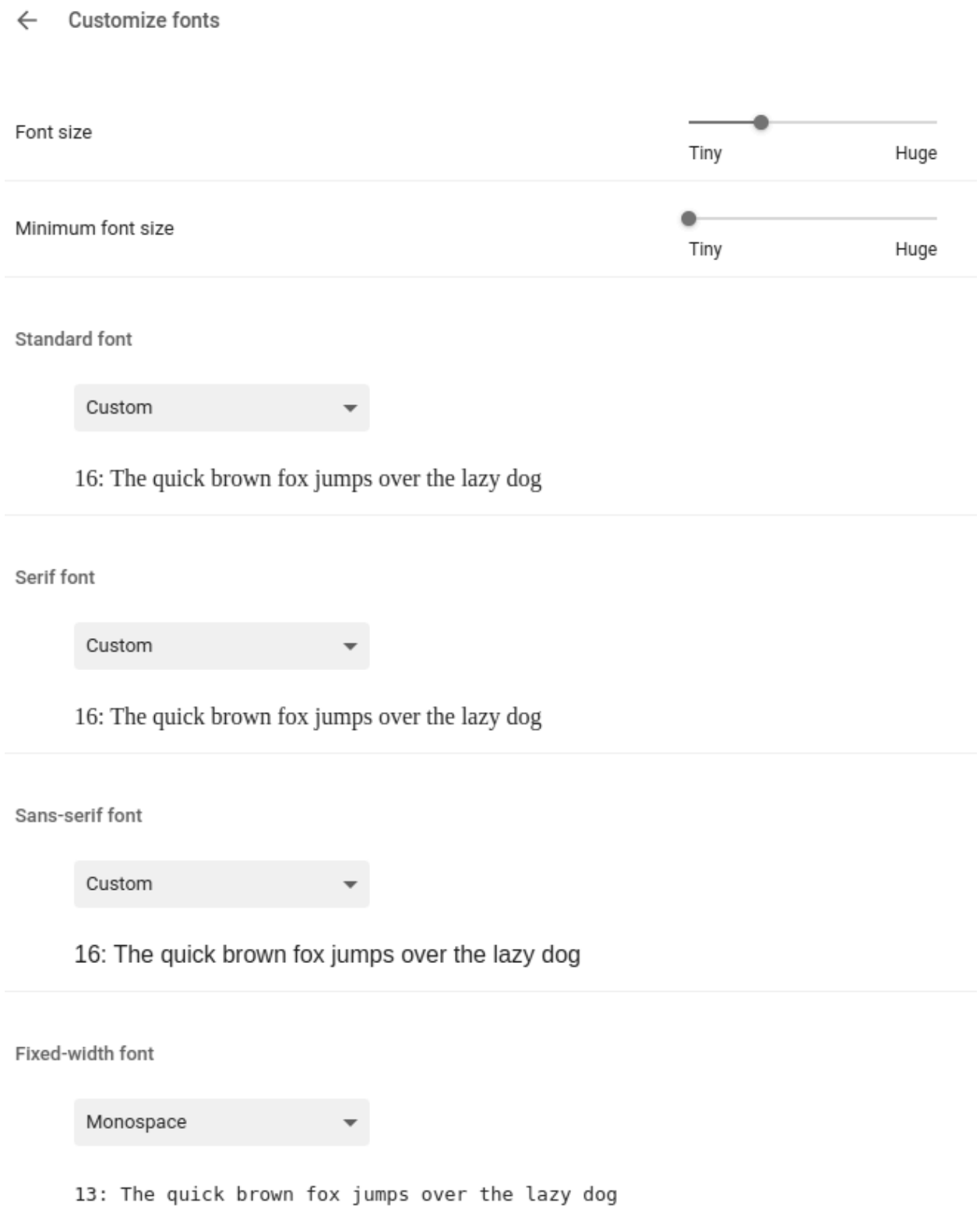
Because it is important to provide specific fonts for the different platforms and make them customizable by users, there are internal preferences to configure generic names. For each generic name and for each unicode script (including the common `Zyyy` script), one can specify a concrete family name as a string. This family name will be used for font selection as described in the previous section. There are several layers:

- An internal `GenericFontFamilySettings` in `platform/fonts` doing the mapping described above. It is used by Blink's `Settings` and corresponding setters are exposed to internal Web tests via the internal settings IDL.
- The WebSettings class on top of `Settings`, which exposes setters to the `WebView`.
- Blink's `WebPreferences` structure, transferable between processes via a Mojo API. These preferences can also be applied to a `WebView` (via `WebSettings`).
- Chromium's preferences (managed by `PrefService/PrefRegistry)` on top of `WebPreferences`. These are stored in a chromium profile. They are not registered on Android, see http://crbug.com/308033.

These preferences are also exposed by the public APIs below. Note that these APIs apply to different levels of the layers previously described.

- Page.setFontFamilies for inspector/devtools provide a way to set generic families of Blink's `Settings` for the frame.
- chrome.fontSettings for chrome extensions provide a way to get and set generic families of Chrome's preferences.
- WebView settings API for Android, which provides a way to get and set generic families of Blink's `WebPreferences`.

User interface



Finally, the `chrome://settings/fonts` user interface provides a way for users to customize

some of the Chromium's preferences, namely the generic family `standard`, `serif`, `sans-serif` and `monospace` for the common `Zyyy` script. For each of these generic names, a dropdown menu is provided to select a font installed on the system, together with a text preview rendered with the selected font family.

Implementation proposals

Test coverage 🧪

When this design document was initially written, there were only tests for a few of the preference APIs mentioned in the previous sessions and a subset of the generic family names. Consequently, a preliminary step for the implementation of font families has been to extend test coverage:

- Blink's `GenericFontFamilySettings`: CL 3581719.
- Blink's `Settings`: CL 3578679, CL 3582581, CL 3578971.
- ~~Blink's `WebSettings`: CL 3582228~~. Not done, since it's very close to the `Settings` and `WebPreferences` ones anyway.
- Blink's `WebPreferences`: CL 3582001.
- Chrome's preferences: CL 3581724.
- Devtools `Page.setFontFamilies`: CL 3578960.
- `chrome.fontSettings` API for extensions: CL 3576395.
- Android `WebView` settings API: The standard preference is covered, others have no effect. See explanation below.
- User interface for font appearance : CL 3570548.

In addition to testing setter/getter for the standard preference (see `AwSettingsTestHelper`'s `ensureSettingHasValue` method), `AwSettingsStandardFontFamilyTestHelper` also verifies that if the standard preference is set to `cursive` or `sans-serif` then the same value is returned by

```
getComputedStyle(document.body).getPropertyValue('font-family')
```

where the body uses the `initial` font-family. This trick works because of how the initial value is handled. Other font preferences are not considered for font selection on Android platform and are consequently not testable.

The `fantasy` and cursive font families

Implementation for these font families is essentially complete. The only thing missing is integrating some menus in `chrome://settings/fonts` to make them configurable by users. This can be done using the same interface as e.g. `sans-serif`.

The `math` font family ✅

The `math` font family is intended for displaying mathematical expressions. Such expressions not only require special symbols but also extra data provided by the OpenType MATH table such as dedicated layout parameters or stretchy constructions. Except for very simple cases, mathematical formulas will render poorly without such a math font.

Because MathML Core is currently being implemented, the need for `font-family: math` becomes instrumental to ensure that mathematical expressions are rendered with a math font by default, rather than the inherited text font. This task is tracked by issue 1228189.

The implementation can be divided into several intermediary steps, covering different parts of the code with their own review processes.

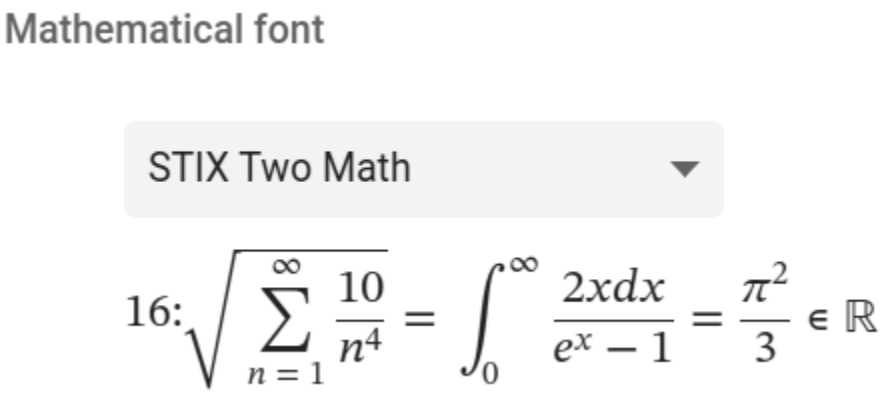
1. Implement the whole logic in Blink, with a hardcoded string instead of a preference for now – CL 2119534
2. Implement the internal preferences – CL 3578971 and CL 3587189
3. Extend Chromium's User Interface – CL 3568407
4. Extend Chrome DevTools API – CL 3586810
5. Extend `chrome.fontSettings` API – CL 3581733
6. Extend Android WebView Settings API – Not done, Android only considers the standard font preference.

Latin Modern Math is an open source font that is included in LaTeX distributions and consequently available on many platforms. It provides the default “Computer Modern” style most math users are often familiar with and willing to select. It is thus a good choice for the default math font, except that it is not generally installed by default. When available on a platform, a better option may thus be a pre-installed math font (e.g. Cambria Math on Windows and STIX Two on macOS). Implementing internal preferences will allow to make the default font more easily configurable per platform, will enable work on follow-up tasks and give the opportunity for users to override it according to their preferences.

As a consequence, it seems important to provide a user interface to allow customization of the math font. The most straightforward approach is to use the same UI as the one available for e.g. serif fonts: a dropdown menu, configurable font size and preview. However, existing text

previews are not really helpful for math fonts: not only they don't show special math symbols but also they don't allow checking the features mentioned above for math formulas.

Instead, this can be replaced with a MathML formula demoing relevant features like stretchy square roots, large integral and summation symbols, fraction spacing, math symbols, etc. The screenshot below shows an example with a formula based on the Riemann zeta function:



Finally, the developer APIs of steps 4-6 could be interesting to offer to developers (and would make things more consistent with other generic families) but they are not critical. Extending the Android API can be ignored until the preference is actually used. The `chrome.fontSettings` API could however be useful to create extensions helping with math font configuration. For example, an enhanced user interface that gives a more complex MathML preview or relies on font enumeration API to list relevant math fonts installed on the system.

Because this is work in progress, it is important to guard the feature under a flag until it is shipped. The Blink implementation at step 1 is guarded under the `CSSFontFamilyMath` flag (which is itself implied by the `MathMLCore` flag) so that they are not web-exposed during development. The User Interface will be protected with a dom-if template conditioned by the `base::Fe` flag, which implies `MathMLCore` and `CSSFontFamilyMath` ; this `dom-if` element will be removed when MathML is enabled by default. Finally, internal preferences and developer APIs are less likely to cause troubles, so they are not conditionally enabled.

The `emoji` font family

The `emoji` font family allows users to select a preferred font for Emojis. It could be interesting to implement it in Chromium and provide a user preference for this, following the same approach as for `font-family: math`. This task is tracked by issue 1240119.

Regarding the user interface, one should likely consider a previewed text containing emojis characters instead of the default text preview.

(Note: Before the Blink fork, Apple introduced a proprietary `-webkit-pictograph` family name that served similar purpose as `emoji` but it was never exposed to user interface or public developer APIs. Web usage was very low so it was unshipped in 2021.)

Other ideas

The CSS spec also defines `fangsong`, `ui-serif`, `ui-sans-serif`, `ui-monospace` and `ui-rounded` which could be considered in a future version of this design document. Like for `system-ui`, it's likely that the `ui-*` ones won't require any preference UI/API. External projects related to Chromium may also need to be updated to handle new generic names (e.g. Android, devtools or Fuchsia).