RenderText and Chrome UI text drawing

Most text in Chrome's UI is rendered either through gfx::Canvas text drawing facilities or by using gfx::RenderText directly. Indeed, gfx::Canvas itself uses gfx::RenderText for drawing and measuring text, so that nearly all UI text drawn in Chrome Windows and ChromeOS ends up going through gfx::RenderText.

gfx::RenderText has platform-specific subclasses for shaping text runs using platform APIs: Uniscribe on Windows, Pango on Linux + ChromeOS and CoreText on the Mac. Although the shaping/layout code is platform-specific, drawing the text uses a common path, using Skia. The gfx::RenderText subclasses also share common implementations of the various effects that can be applied to the text such as fading and shadows, as well as the more rudimentary settings such as text alignment.

In addition to text layout, each gfx::RenderText subclass also performs any necessary selection of fallback fonts needed to render particular runs of text. While this is mostly delegated to Pango on Linux and CoreText on the Mac, the Windows implementation has to do this work itself through the font linking information from the registry.

gfx::RenderText is a stateful API - an instance of a gfx::RenderText subclass will cache its layout information between draw calls. Because of this, it is often more efficient to use the gfx::RenderText API directly instead of using a state-less abstraction such as the gfx::Canvas drawing calls. In particular, for text that changes rarely but that may be drawn multiple times, it is more efficient to keep an instance of gfx::RenderText around, so that the text layout would be performed only when the text is updated and not on every draw operation. Prior to the introduction of gfx::RenderText, this pattern was not possible, so you may see existing code still doing its text drawing through gfx::Canvas text drawing APIs (which in the past were not based on gfx::RenderText).

gfx::RenderText is also used to render footers and headers in printed page. This works because Skia has a PDF backend so that the glyph drawing calls done by gfx::RenderText are appropriately transformed to PDF output. This is currently the case on Mac OS X and Windows, but not yet Linux or ChromeOS because of sandboxing issues.

Comments

You do not have permission to add comments.