Engineer

The underline that crossed every letter

September 28, 2026

Our own site underlines every link, and lets the browser cut a gap where a letter's tail crosses the line — the `g`, the `p`, the comma. In Safari it did not. The line ran straight through them. It had been doing so on the English and the Danish pages for as long as the site had been multilingual.

The cause was three font declarations for an alphabet those pages never download.

Apply the engineering to your system

The engineering behind "The underline that crossed every letter" can inform your next change. We can investigate similar behaviour in your software, data or infrastructure. Interfaces that load quickly, work on a phone and behave the same in every browser your customers happen to use.

Explore our services Get in touch →

Follow the reasoning and its limits

Read the reasoning and its limits in this note. We can test whether the same diagnosis holds in your environment. Directed full-stack GIS map development — full-stack mapping, from database to browser

Directed full-stack GIS map development Get in touch →

What the browser is supposed to do

`text-decoration-skip-ink` is the property that lifts an underline around a descender. It is on by default, and for years the answer to "my underline looks wrong" has been to reach for `text-underline-offset` and push the line further down. That is the wrong reflex: moving the line changes the design, and the gap is what was actually missing.

We found two real causes before the interesting one, and both are worth knowing.

- **A `1px` underline gets a `1px`-sized gap.** The break the engine cuts is proportional to the thickness of the line doing the cutting. Our stylesheet pinned the thickness at one pixel, which left about 0.6px of air either side of a stem — technically a gap, visually a line straight through the letter. Setting `text-decoration-thickness: auto` scaled both together and took the gap from 3.5px to 12.4px without moving the underline a single pixel.
- **`from-font` is not the safe default it sounds like.** It reads the thickness the typeface itself declares, and Ubuntu declares 0.056em for Light, 0.020em for Medium and 0.120em for Bold — a family whose Medium rule is under half the weight of its Light one. At large sizes that is a six-pixel line beside a two-pixel line on the same page.

That fixed Chrome. Safari still drew the line through everything.

The part that took the longest

We measured it properly: drive a real Safari, recolour the underline, screenshot at eight times scale, and count the pixels where the line stops and starts.

	painted thickness	gap around a descender
Chrome	2.00px	5.8 / 12.5 / 6.3px
Safari	1.38px	1.8 / 1.0 / 1.0px

So Safari was cutting a gap. It was just six times narrower than Chrome's, and at reading size a one-pixel gap is not a gap. Worse, the usual lever made it worse: at a 3px thickness Safari dropped from three gaps to one, at 4px to none, because it does not scale the gap with the line. It pins it at roughly 0.07em at every size, where Chrome gives 0.6em.

We swept about twenty declarations against it — `skip-ink: all`, both legacy spellings of the property, every thickness from half a pixel to four, `text-underline-position`, `font-smoothing`, `text-rendering`, `paint-order`. Nothing widened it. The honest conclusion at that point was that WebKit simply capped the gap and there was nothing further to do from a stylesheet.

That conclusion was wrong, and the thing that broke it was a bug report against a font project: Safari ignores skip-ink when a typeface is loaded with a non-Latin subset.

The actual cause

The site serves Ubuntu in six pieces — three weights of Latin, three of Cyrillic — so a Ukrainian page is set in the same typeface as an English one instead of falling back to whatever the system offers. Each piece declares the range of characters it covers, and the browser downloads only the pieces a page actually needs.

All six were declared under the one family name, which is the obvious way to write it. And that is the trigger, recorded as WebKit bug 255159: **when one face in a family declares a non-Latin character range, Safari degrades skip-ink for every character in that family** — including Latin text on a page that never fetches the other file.

An English page was rendering its underlines badly because a Ukrainian font declaration existed somewhere in the stylesheet. Deleting those three lines at runtime, and changing nothing else, took the gaps from 1.8/1.0/1.0px to 4.5/11.0/5.0px.

The fix

Give the second script its own family name, and name it in the font stack after the first:

```
@font-face {
    font-family: UbuntuCyrillic;   /* was: Ubuntu */
    src: url(ubuntu_300_cyrillic.woff2) format('woff2');
    unicode-range: U+0400-045F, U+0490-0491;
}

:root {
    --body-font: Ubuntu, UbuntuCyrillic, system-ui, sans-serif;
}
```

A Latin letter is found in the first family and never reaches the second. A Cyrillic one falls past the first, which has no such glyph, and lands in the second rather than in a system face. The character ranges still decide what gets downloaded, so nothing about the loading changes: English and Danish pages fetch three Latin files and no Cyrillic ones, Ukrainian pages the reverse.

Three renamed declarations and one stack entry. Safari's gaps went to 4.5/11.0/5.0px, Chrome was untouched, the underline did not move, and Ukrainian still sets in Ubuntu at identical widths.

What we would tell the next person

- **A skip-ink measurement is only true of the engine that produced it.** We had a table of numbers, a correct diagnosis and a real fix, and it was all Chromium. The screenshot that reopened the case came from a person looking at the page.
- **Reach for the thickness before the offset.** Widening the gap keeps the design; moving the line down replaces it.
- **Splitting a typeface by script is right, but the split belongs in the family name too.** Sharing one name across scripts reads better and costs you skip-ink in Safari. Merging them back looks like tidying and is a regression.
- **The loading was never the problem.** Every instinct says a page rendering badly because of a Cyrillic font must be downloading something it should not. It was not. The declaration alone was enough.

Sources

- WebKit bug 255159 — `text-decoration-skip-ink` and font subsets
- Fontsource issue 1096, which named the trigger
- MDN: `text-decoration-skip-ink`
- MDN: `unicode-range`

All blog posts

Measure before you change

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Discuss your observations

Discuss “The underline that crossed every letter” in the context of your own project. Tell us what you observed and what a useful result would look like.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/notes/safari-underline-font-subset/