Engineer

Web browser

the program a person uses to open web pages

What it is

A web browser is the software that takes an address, fetches whatever the server sends back and turns it into a page someone can read, scroll and click; in everyday speech it is just the browser. Chrome, Safari, Firefox and Edge are the names on the icons. Underneath them sit far fewer programs than the names suggest.

The HTML standard rarely says “browser”. Its word is *user agent*, the software acting on the reader’s behalf, and most of its length is a list of what a user agent must do with a page. Three of those duties explain most of what a browser is for.

It parses. The standard sets out parsing rules for HTML documents “whether they are syntactically correct or not”, so a missing closing tag or a misplaced element does not stop the page. Every conforming browser repairs the same broken markup into the same tree of elements, which is why a sloppy page still appears and why its mistakes can sit unnoticed for years.

It runs an event loop. Clicks, key presses, timers, network replies, scripts and the drawing of the next frame queue up and take turns. A script that holds the loop for two seconds holds up clicks, typing and the next paint with it, which is what a person means when they say a page has hung.

It keeps sites apart. The standard defines an origin as a scheme, a host and a port, and a site as a scheme plus the registrable part of the domain. A script from one origin that reaches into the document of another gets a security error, not the contents. Most of what keeps a bank tab safe from a news tab is this rule, enforced by the browser on every access.

The part that does this work is the *engine*. Most browsers in use today are built on one of three: Blink inside Chrome, Edge and a long tail of others; WebKit inside Safari; and Gecko inside Firefox. A new browser name usually means a new shell around one of these, with its own address bar, settings and sync, but the same rules for reading and drawing a page.

Two neighbours get confused with it. A search engine is a website the browser opens when the address bar is given words instead of an address. A *webview* is a browser engine embedded inside another app, with the address bar and most of the controls taken away. Test tools add a *headless* browser, which draws pages no one looks at so that a script can check them.

Use the concept in a decision

Put “What is a web browser?” to work in a real decision. We can help relate the concept to the system you are building or buying. 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 →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Directed full-stack GIS map development — full-stack mapping, from database to browser

Directed full-stack GIS map development Get in touch →

Why it matters

A server can send exactly the right bytes and the reader can still get an unreadable page, because what a person sees is whatever their browser made of those bytes, with their own settings applied on top. The browser is where the work is judged.

Testing follows engines, not brands. A test plan with four browser names on it may be covering two engines twice and missing the third. The useful list names the engines and the platforms where the same engine behaves differently, such as a phone with a small screen and touch input, or a print dialog.

The reader’s settings are part of the page. A browser carries a preferred colour scheme, a request for reduced motion, a zoom level, a default text size and sometimes a translation layer. A page that ignores all of that is designed for one configuration of one person, usually whoever built it.

Features are asked for, not assumed. Engines ship new CSS and new interfaces at different times. A page that needs something only some engines do can ask the engine directly: CSS has `@supports`, and a script can test for a property before it uses it. The user-agent string is a poor substitute, because browsers have long written it partly to look like each other so that older sniffing code would let them in.

Rules in the browser are courtesies. Form validation, a disabled control or a hidden link all run inside a program that the reader owns and can change. They make the honest path easier and stop no one who opens the developer tools or skips the page entirely, so anything that protects money, data or access is checked again on the server.

What it looks like in practice

This site underlines its links with skip-ink, the setting that breaks the line where it would cross a descender. In Chromium the breaks looked right. Measured in a real Safari at eight times scale, the breaks were about six times narrower, and the cause turned out to be a WebKit defect in which a single font face declaring a non-Latin character range degraded skip-ink for the whole family. The fix was a renamed family in the font stack. The account of the Safari underline and the font subset that caused it ends on the lesson this article would put first: a skip-ink measurement is only true of the engine that produced it. The table, the diagnosis and the first fix were all Chromium, and the screenshot that reopened the case came from a person looking at the page.

Printing is a second place where engines part ways. The 637-line print stylesheet documents four engine behaviours as traps, each tied to a real defect. One engine sizes viewport units against the printed page box and another against the browser window, so a page printed from a wide window lost a third of every line in the second. Fixed-position elements paint on every sheet. Labels coloured for the dark default theme printed white on white for readers whose browser preferred dark. A flex or grid box that paged media cannot split produced a blank first sheet. The check that guards it runs six paper and orientation combinations under both colour-scheme preferences and counts the real pages, and one engine still ignores widow and orphan control.

The third case was found by readers rather than by a test. When the site went out as a Tor onion mirror on a readable address, its first readers were on a different browser engine, one that does not implement the scroll-driven

animation the footer relied on, and every one of them got the colophon painted across the front page. The defect was real on the ordinary site as well; it took an audience on another engine to show it.

Where it goes wrong

- **A page is checked in the browser on the developer's desk and declared done.** Every habit of that one engine becomes a hidden assumption. The remedy is a short matrix with one browser per engine, plus a phone, measured in each before a fix is called done.
- **Code reads the browser's name and guesses what it can do.** The guess is wrong for every browser that borrows another's label, and it goes stale with each release. Asking for the feature keeps the decision with the engine, and the rule outside the block is what every other engine gets, so it should be the safe state:

```
.footer {
    position: static;
}

@supports (animation-timeline: scroll()) {
    .footer {
        animation: settle linear both;
        animation-timeline: scroll();
    }
}
```

- **Unstyled elements inherit the browser's own defaults.** Headings, margins and especially form controls arrive with each engine's built-in style sheet, and those differ. A design that looked finished may have been relying on one engine's choices without anyone deciding them.
- **A rule enforced only in the page is treated as enforced.** A button greyed out by a script, a field limited to numbers, a link hidden from people without a plan: each holds only for readers who leave the page as it was sent. The server repeats every check that matters.
- **Links open inside someone else's app.** A link tapped in a chat or on social media often lands in an in-app browser or a webview, frequently with storage of its own, so a person signed in to their main browser arrives as a stranger. Flows that hand off to another app or rely on an existing session need testing from those entry points too.

Further reading

- NIST CSRC Glossary: web browser — two short definitions taken from NIST publications, one of them the plain “client software used to view Web content”.
- HTML Standard: Parsing HTML documents — the rules every browser follows to turn markup, broken or not, into the same tree.
- HTML Standard: Event loops — how input, scripts, network replies and rendering take turns on one loop.
- HTML Standard: Loading web pages — origins, sites and the rules that stop one page from reading another.
- WebKit Bugzilla: bug 255159 — the engine’s own record of the skip-ink defect described above.

Glossary

Choose for your actual constraints

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 →

Make the terminology practical

Ask how “What is a web browser?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

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/glossary/browser/