Engineer

HTML

the language a page's structure is written in

What it is

HTML, short for HyperText Markup Language and still widely called HTML5, is the plain-text format every web page arrives in: content wrapped in tags that say what each piece is, so the browser can tell a heading from a paragraph, a link from a list item and a form field from its label.

A browser reads that text from top to bottom and builds a tree of elements from it, the Document Object Model. That tree is what stylesheets style, what scripts change and what screen readers announce. The division of labour on the web is old and still holds: HTML says what something is, CSS says how it looks, and JavaScript says what it does when someone touches it. An element such as `<nav>`, `<button>` or `<h2>` carries meaning every one of those readers can act on. A real `<button>` gets keyboard focus, a role that assistive technology announces, and activation by Enter or Space, all without a line of extra code.

The language began as a way to describe scientific documents semantically, and it grew into the layer underneath almost every application with a web interface. Its first five years, 1990–1995, were spent at CERN and then at the IETF; HTML 3.2 was completed at the W3C in 1997, and HTML4 followed soon after. For years the W3C and the WHATWG kept diverging texts, until in 2019 they signed an agreement to work on one: the HTML Living Standard, which the WHATWG maintains and every major browser engine implements. It carries no version number. Asked “is this HTML5?”, the standard answers with a short yes and calls the term a buzzword for the modern web platform rather than a release.

Where does it sit beside the things it is compared with? Markdown is a shorthand that a build tool turns into HTML. PDF fixes the layout to a printed page, where HTML reflows to whatever screen it lands on. XHTML is the same vocabulary written under the stricter rules of XML. JSX, Jinja and Hugo templates are ways

of producing HTML, and what reaches the browser from all of them is still HTML. A team weighing a hand-built site against a website builder is choosing who writes the markup, never whether there is any.

Use the concept in a decision

Put “What is HTML?” 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

Few teams choose HTML on purpose. It arrives as whatever their framework, page builder or content system happens to emit. The choice they are really making is whether that output is a deliverable someone inspects or a by-product nobody reads, and the consequences land in three places.

Who can use the page comes first. Assistive technology reads the tree, not the design. A screen-reader user moves through a page by its headings and landmarks; a keyboard user tabs from one link or control to the next. A styled `<div>` with a click handler looks like a button to a sighted mouse user and is plain text to both of the others. Accessibility requirements such as WCAG are written largely against what the markup exposes, so meeting them after launch tends to mean rewriting templates rather than adjusting colours.

How long the content lasts comes second. Browsers still open pages written in the 1990s, because backward compatibility is one of the principles the standard is maintained under. Frameworks turn over every few years. Content held in clean, valid HTML can be moved to another system, archived, converted to other formats or given a new design without being rewritten. Content buried in a page builder’s nested wrappers usually has to be copied out by hand.

Speed and reach come third. A page the server sends complete can be painted as the bytes arrive. An application that sends a near-blank shell and builds the document in JavaScript makes every reader wait for the script, and anything that reads the page without running it, from a link preview to many crawlers, sees

only the shell. For a product behind a login, where interaction is the point, that trade can be worth it. For marketing pages, documentation and anything meant to be found by search, complete HTML is the cheaper default.

The cost of all three is the same and small: someone on the team who reads HTML as carefully as they read code. Semantic markup costs almost nothing when it is written first and a great deal when it is retrofitted.

What it looks like in practice

This site is the plainest example to hand. Its credits page lists what reaches a visitor's browser: HTML, one stylesheet, local webfonts, images, with no executable JavaScript. Every page is written out in full as a file on disk before anyone asks for it, rather than assembled per request from a template. The site's quality gate validates that HTML with vnu and treats every diagnostic as an error, and the same page links the W3C Markup Validator, which runs that engine against the living standard, so a reader can check the claim instead of trusting it.

Earlier client work came at HTML from the other direction. Across thirty websites built from finished visual designs, the Photoshop files were turned into HTML by hand. The standard had two sides: faithful to the spacing, type and detail the designer intended, and flexible enough that a site kept working the first time someone edited its content. Matching a mockup is half of that job; markup that survives the next edit is the other half.

Keeping HTML as the canonical form also leaves room to publish other formats beside it. When the site was mirrored to Gemini and Gopher, the same build produced 777 Gemini documents and 777 Gopher menus from the same source at the recorded implementation stage. A separate build comparison measured zero HTML bytes changed when those formats were added; that result does not describe later editorial or visual changes.

Where it goes wrong

The layout shifts in ways no stylesheet explains. A page that starts without `<!DOCTYPE html>`, or with an old and malformed one, puts the browser into quirks mode, a rendering mode that imitates legacy browsers. Percentage heights, table cells and line heights then follow old behaviour that departs from the CSS specifications, and the result differs from the same page in standards mode. The fix is one line at the top of every template.

Broken markup renders, and the tree comes out different. The HTML parser recovers from every error in a defined way, identically in every browser, so mistakes rarely show on screen. They surface later, when a selector or a script looks for an element somewhere other than where the parser put it. A block element inside a paragraph is the classic case:

```
<p>Opening line
  <div>A block placed inside it</div>
</p>
```

The parser closes the paragraph before the `<div>`, then turns the stray `</p>` into a second paragraph with nothing in it. Run a validator in the build and fail on its errors; the browser's tolerance is a recovery mechanism, and it reviews nothing.

Every control turns into a `div`. Clickable `div` and `span` elements, headings picked for their size, a subscript used to make text smaller: each one looks right and tells assistive technology the wrong thing. Keyboard users pass controls that never receive focus, and screen-reader users lose the headings and landmarks they navigate by. A page without a `<main>` element or an `<h1>` leaves them reading from the top every time. Use the element that names the job, such as `button`, `a`, `h2`, `main` or `nav`, and let CSS change its look.

The page stays blank until its script runs. A client-rendered site sends little more than `<div id="app"></div>` and builds the document in the browser. On a slow phone the reader waits; when the script errors or a filter blocks it, the page stays blank for good. Render the document on the server or at build time, and add interaction on top of HTML that already makes sense on its own.

Further reading

- HTML Living Standard — the specification browsers implement, maintained by the WHATWG and updated continuously.
- Introduction to the HTML standard — the history from CERN to the 2019 agreement, and the standard's own answer to the HTML5 question.
- Parsing HTML documents — the error recovery every browser shares, written out rule by rule.
- The HTML syntax — the doctype, elements and attributes, and what makes a document conforming.

- The Nu Html Checker — the engine behind vnu and the W3C's markup validator, with its command-line use.
- DOM Standard — the tree the parser builds and the interfaces scripts use to change it.

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 HTML?" 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/html/