Engineer

Viewport

the part of a page a window actually shows

What it is

A viewport is the rectangle a browser lays a document out in and paints it into: on a screen, the area inside the window's own toolbars; on paper, the page area of each sheet. CSS measures against it constantly. The initial containing block, the box the root element is laid out in, takes its size, a `width` or `height` media query asks about it, and the `vw` and `vh` units are hundredths of it. On a phone the word, and its plural viewports, needs a qualifier, and the specifications supply several: layout, visual, small, large and dynamic.

On a desktop the viewport and the window are nearly the same thing. On a phone they part company. Because so many documents were designed for desktop windows, mobile browsers lay out a page that gives no instructions at a fixed width that mimics a desktop window, typically 980 to 1,024 CSS pixels according to the CSS Viewport specification, and scale the result down to fit the glass. A page opts out with one element in its head, which accepts `width`, `height`, three scale limits, `user-scalable` and `interactive-widget`:

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

Zoom splits the viewport in two. The layout viewport is the one CSS lays the page out against, and pinch-zoom leaves it alone, so line breaks stay put. The visual viewport is the part of the page on the glass at that moment: it shrinks as the reader zooms in and slides as they pan. By default an on-screen keyboard shrinks only the visual viewport; `interactive-widget=resizes-content` asks the browser to resize the layout as well, and `overlays-content` asks it to resize neither.

Phones add browser chrome that slides away. The address bar takes space when a page loads and retracts as the reader scrolls, so the visible height changes under a page that has not moved. CSS Values 4 names three sizes for this: the large viewport assumes every retractable toolbar is retracted, the small one assumes every one is shown, and the dynamic one follows whichever is true. The units carry the prefixes, `lvh`, `svh` and `dvh`, with matching widths, and plain `vh` and `vw` are defined against the large viewport. An interface that overlays content without shifting it, typically the on-screen keyboard, changes none of these lengths.

Use the concept in a decision

Put “What is a viewport?” 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

Responsive design is built on the viewport. A width breakpoint in a stylesheet is a question about its width, so a page that omits the `meta` element is laid out at desktop width on a phone and shrunk until its text is unreadable, however careful its media queries are.

Height is the hard axis. Width can change with rotation or resizing; height can also change when the toolbar slides, when the keyboard opens and when the phone turns, so anything sized to the screen’s height has to say which height it means. The small viewport allows content to fit with retractable toolbars shown and leaves a strip of empty space when those toolbars retract. The large one fills the screen and puts content behind the toolbars while they show. The dynamic one accounts for changes in the toolbar state, and the specification itself warns that it can resize content while the reader scrolls, which moves everything below.

Screens also have corners. A phone with rounded corners or a camera cutout is not a rectangle, and with the default, `auto`, the whole page stays viewable on it. Adding `viewport-fit=cover` asks for the viewport to be the rectangle drawn

around the whole physical screen, and hands the page the job of keeping text clear of the cutout with the `env(safe-area-inset-*)` variables. Those insets must be zero on a rectangular display, so one padding rule serves both kinds.

Fixed positioning hangs on the viewport too. A fixed box is placed against the layout viewport, which CSS Positioned Layout sizes to match the dynamic viewport, so cookie bars, modal backdrops and full-screen menus hold still while the page scrolls. Each assumes no ancestor has taken that role, and `transform`, `will-change` and `contain` can.

Print is the viewport a screen-only stylesheet never meets. On paper the viewport units measure the page area, a `width` media query asks about the page box, and a fixed element repeats on every page.

What it looks like in practice

The pages on this site declare `width=device-width, initial-scale=1, viewport-fit=cover`, so they lay out at the phone's width and run edge to edge beneath a notch. The stylesheet pays for the second half: side padding is `max(1rem, env(safe-area-inset-left))` and its right-hand twin, so text keeps a rem from the edge and moves further in only where a cutout is. On short landscape screens the floor rises to `max(2rem, 7vw, env(safe-area-inset-left))`, since the insets are zero on a phone without a cutout and one rem sat too close to the edge.

Full-screen sections take their height from a shared variable set to `100dvh`, with `100vh` as the fallback, and the comment beside it records why: `100vh` can be taller than what shows while the address bar is out, and it had let the next section peek in under the hero. Two elements deliberately ignore it. The masthead mark is sized in static `vh`, because a mark keyed to the dynamic viewport would resize as the browser chrome slid away and reflow the hero in mid-scroll. The fixed background on touch screens is `100lvh` tall, so it never resizes as the address bar comes and goes, and it sits on its own compositing layer, because touch browsers tore it into light and dark bands during momentum scrolling.

An earlier home-page layout gave each service block a minimum height of `100dvh`, so it occupied one screen while its content was shorter than the screen. The stylesheet comments checked in November 2026 record a test at 852 by 393

pixels: a block's content ran to 412, and the blocks butted together. In that layout, padding inside the `100dvh`, through `box-sizing: border-box`, cost nothing on a tall screen and opened a gap only once the content overflowed.

At the documented August 2026 quality-gate implementation stage, testing followed the media queries. The site's browser-driven quality gate widened its checks to a shared table of twenty-two window shapes, taken from every media query in the stylesheet, and that took the layout check alone from 95 seconds to 405.

Print exposed the engines. The print stylesheet records viewport units resolving to the page box in one engine and to the window in another, so a page printed from a wide window lost a third of every line in the second. The masthead's own stylesheet records the media-query side of the same gap: print kept the window's orientation, height and width, so sheets printed from a wide and a narrow window came out with mastheads 47 pixels apart, until `screen and` at the front of the query confined the rule to windows.

The containing-block trap is on record as well. The reduced-motion work found that five different properties each make an element the containing block for anything positioned against the viewport, and that this had silently shrunk a dismiss layer to a fraction of the screen.

Where it goes wrong

The phone shows a tiny desktop page. Without the viewport `meta` element the browser lays the page out at a desktop width and scales it down, and the media queries see that width. Add `width=device-width, initial-scale=1`, and leave zoom on: `user-scalable=no` takes away the reader's own way to enlarge small text.

A full-screen section runs past the bottom of the screen. `100vh` is the large viewport, so while the address bar is out the bottom of the section, and any button placed there, sits behind it. Use `svh` to keep an element clear of expanded browser toolbars, and `dvh` only where resizing mid-scroll is acceptable.

A full-screen overlay shrinks to a box inside the page. An ancestor with a `transform`, a `will-change` naming one, or certain containment settings can become the containing block for `position: fixed`, and the overlay sizes itself to that ancestor. Render overlays outside any transformed element.

A printed page loses the end of every line. Viewport units and window-sized media queries leak into print, and an engine that resolves them against the window lays the sheet out for a screen. Keep viewport units out of anything that prints, and put `screen and` in front of queries meant for windows.

Further reading

- CSS Viewport Module Level 1 — the viewport `meta` element and the three `interactive-widget` behaviours.
- CSS Values and Units Level 4: viewport-percentage lengths — the small, large and dynamic viewport sizes and every unit built on them.
- CSS Positioned Layout Level 3 — fixed positioning, the properties that capture it and what it does on paper.
- CSS Environment Variables Level 1 — the safe-area insets and the rule that makes them zero on a rectangular screen.
- Media Queries Level 4 — `width`, `height` and `orientation`, measured on the viewport on screen and on the page box in print.

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 viewport?” 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/viewport/