Engineer

Lazy loading

loading something only when it is first needed

What it is

Lazy loading means putting off the fetching of something until the moment it is needed, instead of loading everything up front. On a web page it usually means images and embedded frames further down, which are fetched only as the reader scrolls towards them. The same idea applies to code that runs only after a click, and to the related records an application reads from a database.

The HTML Standard builds it into the markup. Images and iframes take a `loading` attribute with two values. `eager`, the default, fetches the resource straight away. `lazy` is, in the standard's words, "used to defer fetching a resource until some conditions are met." The standard defines the same attribute on `audio` and `video` as well. A missing or misspelled value counts as `eager`, so a typo leaves the page loading the way it always did.

The condition is that the element comes close to the viewport. Each document has an intersection observer of its own for this job, and when it reports a lazy element about to come into view, the browser resumes loading it. That observer looks a little beyond the visible area, so that an image can arrive while the reader is still scrolling towards it, and the standard leaves the size of that margin to the browser.

Pages can use the same mechanism directly. The W3C's Intersection Observer specification lets a script ask to be told, asynchronously, when an element starts or stops intersecting the viewport or another element, and it names custom deferred loading of content and data among the practices it was written to support. It covers what the attribute does not: a background image set in CSS, a map that needs its own script, a list that fetches its next page when the reader reaches the end.

For code, the unit is a module. The Next.js documentation presents lazy loading as a way to cut the JavaScript needed to render a route: client components and libraries are left out of the first bundle and fetched when they are needed, its example being a dialog that loads only once someone clicks to open it. Its `next/dynamic` combines React's `lazy()` and `Suspense`, and a plain `import()` call does the same for a library.

The term is older than any of this. In Martin Fowler's catalogue of enterprise application patterns, Lazy Load leaves a marker where related data would go and fetches that data only if something uses it, so that reading one object from a database does not pull in a large web of other objects nobody looks at. An ORM works this way when it fetches a related record the first time code touches it. Django does so by default: a field that was not part of the original query is read from the database when it is first accessed, which its documentation describes without using the name.

Lazy loading is not caching, which keeps a copy of something already fetched so it can be used again. It is the opposite of preloading: a `<link rel="preload">` tells the browser to fetch a resource ahead of time, because the page is very likely to need it.

Use the concept in a decision

Put "What is lazy loading?" 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 page pays for every byte it asks for, whether or not anyone sees it. A long article with forty photographs, a footer with an embedded map and a product page with a video near the bottom all spend bandwidth, battery and connection time on content that some readers never scroll to. With lazy loading, those requests are made only for the readers who get that far, and what the first screen needs does not have to share the connection with them.

How early is early enough is left to the browser. The standard suggests that the margin take account of how fast the reader is scrolling and how good the network is, with a wider margin on slow or high-latency connections, so that images are in place by the time they scroll into view. It also requires that the margin not leak information that could be used to fingerprint the reader.

There is a privacy rule built in as well. Browsers lazy-load only when scripting is enabled. Otherwise, the standard explains, a site could scatter images through its markup and work out roughly how far each reader scrolled from which images the server is asked for, and when. With scripting off, `loading="lazy"` quietly becomes an ordinary load.

For code, Next.js puts the gain in terms of the first load: less JavaScript needed before a route can render. A charting library used on one tab, or a rich text editor behind an Edit button, need not slow down every other page.

In an ORM the gain is in what is never read. Fowler’s point is that loading an object together with everything related to it is convenient until it means loading a huge number of objects when only a few are wanted, and Lazy Load breaks that chain.

What it looks like in practice

For images and frames, the attribute does the work. The image and the map below sit far down a page, so both can wait:

```
<img
  src="/photos/harbour-800.jpg"
  srcset="/photos/harbour-800.jpg 800w, /photos/harbour-1600.jpg 1600w"
  sizes="auto, 100vw"
  width="1600"
  height="900"
  alt="Fishing boats in the harbour at dawn"
  loading="lazy"
/>

<iframe
  src="/embeds/route-map.html"
  title="Map of the walking route"
  width="640"
```

```
  height="360"
  loading="lazy"
></iframe>
```

`width` and `height` give the image its proportions before any of it has arrived: the HTML rendering rules map the two numbers to an aspect ratio, so the browser can hold a box of the right shape. `sizes="auto"` is allowed only on a lazy image. It tells the browser to choose from `srcset` by the width the image is actually laid out at, which a lazy image has by the time it is fetched, and the `100vw` after it is the fallback when that width cannot be worked out.

Some lazy loading needs a script. Here a map library is fetched only when its container is about to scroll into view:

```
const box = document.querySelector("#route-map");

const observer = new IntersectionObserver(
  async (entries) => {
    if (!entries.some((entry) => entry.isIntersecting)) return;
    observer.disconnect();
    const { drawMap } = await import("./route-map.js");
    drawMap(box);
  },
  { rootMargin: "200px 0px" },
);

observer.observe(box);
```

The `rootMargin` grows the viewport by 200 pixels at the top and bottom for this check, so the module starts loading shortly before the box is visible. The callback runs asynchronously and can report several entries at once, and it checks `isIntersecting` because the observer also reports elements that are outside the area. Disconnecting first means the callback does its work once, however often the reader scrolls past. Until then, `route-map.js` is not part of the page at all.

Fonts can be split so that a browser fetches only the parts a page uses, with no script involved. A note on this site about a Safari underline bug records how the site's Ubuntu typeface was then served as six files, three weights for Latin text and three for Cyrillic, each of them stating the range of characters it covers. The browser routes Latin prose to the Latin cuts and Cyrillic prose to the Cyrillic ones. A Cyrillic language label can therefore need a Cyrillic cut on an otherwise English

or Danish page. The note does not call this lazy loading, but the same need-driven choice keeps an unused cut out of the page's font requests unless it is explicitly preloaded.

The files every page needs at once call for the opposite decision. A portfolio entry on cutting the stylesheet bundle explains why a font embedded in the stylesheet as base64 was taken out. It had been put there to speed up the first paint, but a separate file that is preloaded gives the same benefit, while the embedded copy added its weight to every page, needed there or not. At that stage, three weights of the font were preloaded instead, and the entry notes that field data, not a sense of completeness, put the third one on that list. Lazy loading and preloading are two sides of one judgement: what the first screen needs comes early, and the rest waits its turn.

Where it goes wrong

The first image arrives last. A lazy image waits for more than the network. Its fetch resumes only when the browser's observer reports it near the viewport, and observers are updated as part of rendering a frame, after style and layout. For a photograph far down the page that wait costs nothing. For the large image at the top, it delays the one picture every visitor sees. Leave `loading` off, or set it to `eager`, for anything that can appear in the first screen on a phone or a desktop, and keep `lazy` for what comes below.

The page jumps as images land. An image without `width` and `height` has no known shape until its file arrives, and then pushes everything below it down. With lazy loading that happens while the reader is scrolling, often in the middle of a sentence. Give every image its dimensions in the markup or in CSS. For `sizes="auto"` the HTML Standard is explicit: without stated dimensions such an image will likely be drawn at 300 by 150 pixels, so it strongly encourages giving them.

Images that need a script to exist. A hand-made version puts the real address in a `data-src` attribute and lets a script copy it into `src` when the image comes near. If that script does not load, throws an error or is blocked, the image never appears, and anything that reads the markup without running scripts sees an image with no source. The built-in attribute fails the other way: with scripting disabled, the standard has the browser load the image straight away. Use `loading="lazy"` for images and frames, and keep scripts for what the attribute does not cover.

Deferring what every visit needs. Lazy loading pays off when much of what is deferred is never used. When everything will be used, it only splits one fetch into many. A loop that reads a related record for each item in a list sends a separate query for each: Django's documentation counts one query for the list plus one per item in its default mode, and calls the result the N+1 queries problem. A page whose components each request their own data as they appear does the same over the network. Load what is certain to be needed together with the list, in one query or one request, and save lazy loading for what is likely to be skipped.

Further reading

- HTML Standard: Lazy loading attributes — the `lazy` and `eager` keywords, the browser's own intersection observer and the rule that switches lazy loading off without scripting.
- Intersection Observer — the W3C working draft behind the observer: roots, margins, thresholds and how callbacks are queued.
- How to lazy load Client Components and libraries — `next/dynamic`, `React.lazy()` with `Suspense`, skipping server rendering and loading a library with `import()`.
- Lazy Load — Martin Fowler's pattern from Patterns of Enterprise Application Architecture, with its four varieties.
- Fetch modes — how Django fetches fields that were not loaded with the original query, and the modes that batch or block those fetches.

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 lazy loading?” 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/lazy-loading/