Engineer

Lazy loading

at hente noget først, når der er brug for det

Hvad det er

Lazy loading betyder, at man venter med at hente noget, til det rent faktisk skal bruges, i stedet for at indlæse det hele fra starten. På en webside drejer det sig som regel om billeder og indlejrede rammer længere nede, som først hentes, når læseren scroller hen mod dem. Samme idé gælder for kode, der kun skal køre efter et klik, og for de relaterede poster, et program læser fra en database.

HTML-standarden har bygget det ind i selve markuppen. Billeder og iframes har en `loading` -attribut med to værdier. `eager` , som er standardværdien, henter ressourcen med det samme. `lazy` bruges ifølge standarden »til at udskyde hentningen af en ressource, indtil visse betingelser er opfyldt«. Standarden definerer også attributten på `audio` og `video` . En manglende eller forkert stavet værdi tæller som `eager` , så en slåfejl blot får siden til at indlæse, som den altid har gjort.

Betingelsen er, at elementet nærmer sig viewporten. Hvert dokument har sin egen intersection observer til netop det formål, og når den melder, at et element med `lazy` er ved at komme til syne, genoptager browseren indlæsningen. Observeren kigger lidt ud over det synlige område, så et billede kan nå frem, mens læseren stadig er på vej ned mod det, og standarden overlader det til browseren at bestemme, hvor bred den margen skal være.

Sider kan også bruge mekanismen direkte. W3C's specifikation Intersection Observer lader et script få besked, asynkront, når et element begynder eller holder op med at overlappe viewporten eller et andet element, og den nævner selvbygget udskudt indlæsning af indhold og data blandt de formål, den er skrevet til. Den dækker det, attributten ikke dækker: et baggrundsbillede sat i CSS, et kort, der kræver sit eget script, en liste, der henter næste side, når læseren når bunden.

For kode er enheden et modul. Next.js-dokumentationen beskriver lazy loading som en måde at skære ned på den JavaScript, der skal til for at rendere en route: klientkomponenter og biblioteker holdes uden for den første bundle og hentes, når der bliver brug for dem, og eksemplet er en dialogboks, der først indlæses, når nogen klikker for at åbne den. `next/dynamic` kombinerer Reacts `lazy()` og `Suspense`, og et almindeligt `import()`-kald gør det samme for et bibliotek.

Begrebet er ældre end alt dette. I Martin Fowlers katalog over mønstre til forretningsapplikationer efterlader en Lazy Load en markør der, hvor de relaterede data skulle have stået, og henter først dataene, hvis noget bruger dem. Så trækker det at læse ét objekt fra en database ikke et stort net af andre objekter med sig, som ingen kigger på. En ORM arbejder på den måde, når den henter en relateret post første gang, koden rører ved den. Django gør det som standard: et felt, der ikke var med i den oprindelige forespørgsel, læses fra databasen, første gang det tilgås, og dokumentationen beskriver det uden at bruge navnet.

Lazy loading er ikke det samme som caching, der gemmer en kopi af noget, som allerede er hentet, så det kan bruges igen. Det er det modsatte af preload: et `<link rel="preload">` beder browseren hente en ressource i forvejen, fordi siden med stor sandsynlighed får brug for den.

Brug begrebet i en beslutning

Brug »Hvad er lazy loading?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Brugerflader, der indlæses hurtigt, virker på en telefon og opfører sig ens i alle de browsere, jeres kunder nu bruger.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har styret full-stack-udviklingen af GIS-kort — full stack-kortløsning, fra database til browser

Har styret full-stack-udviklingen af GIS-kort Kontakt os →

Hvorfor det betyder noget

En side betaler for hver byte, den beder om, uanset om nogen ser den. En lang artikel med fyrre fotografier, en sidefod med et indlejret kort og en produktside med en video helt nederst bruger alle båndbredde, batteri og forbindelsestid på indhold, som nogle læsere aldrig scroller ned til. Med lazy loading bliver de forespørgsler kun sendt for de læsere, der når så langt, og det, den første skærm skal bruge, skal ikke dele forbindelsen med dem.

Hvor tidligt der skal hentes, er overladt til browseren. Standarden foreslår, at margenen tager højde for, hvor hurtigt læseren scroller, og hvor godt netværket er, med en bredere margen på langsomme forbindelser og forbindelser med høj latens, så billederne er på plads, når de kommer til syne. Den kræver også, at margenen ikke afslører oplysninger, der kan bruges til fingerprinting af læseren.

Der er også en regel om privatliv bygget ind. Browsere bruger kun lazy loading, når scripting er slået til. Ellers, forklarer standarden, kunne et website sprede billeder ud over sin markup og regne nogenlunde ud, hvor langt hver læser scrollede, ud fra hvilke billeder serveren bliver bedt om, og hvornår. Med scripting slået fra bliver `loading="lazy"` i al stilhed til en almindelig indlæsning.

For kode beskriver Next.js gevinsten som en lettere første indlæsning: mindre JavaScript, der skal til, før en route kan vises. Et diagrambibliotek, der kun bruges på én fane, eller en editor til formateret tekst bag en Rediger-knap behøver ikke gøre alle de andre sider langsommere.

I en ORM ligger gevinsten i det, der aldrig bliver læst. Fowlers pointe er, at det er bekvemt at indlæse et objekt sammen med alt, hvad der hører til det, indtil det betyder, at et enormt antal objekter bliver indlæst, når der kun skulle bruges nogle få, og Lazy Load bryder den kæde.

Sådan ser det ud i praksis

For billeder og rammer klarer attributten arbejdet. Billedet og kortet herunder ligger langt nede på en side, så begge kan vente:

```
<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="Fiskerbåde i havnen ved daggry"
  loading="lazy"
/>

<iframe
  src="/embeds/route-map.html"
  title="Kort over gåturens rute"
  width="640"
```

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

`width` og `height` giver billedet dets proportioner, før noget af det er nået frem: HTML's regler for rendering omsætter de to tal til et størrelsesforhold, så browseren kan holde en boks i den rigtige form klar. `sizes="auto"` er kun tilladt på et billede med `lazy`. Værdien beder browseren vælge fra `srcset` efter den bredde, billedet faktisk får i layoutet, og den bredde kender browseren, når et lazy-billede skal hentes. `100vw` bagefter er reserven, hvis bredden ikke kan bestemmes.

Noget lazy loading kræver et script. Her hentes et kortbibliotek først, når dets container er ved at komme ind på skærmen:

```
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);
```

`rootMargin` udvider viewporten med 200 pixels foroven og forneden ved denne kontrol, så modulet begynder at blive hentet, lidt før boksen er synlig. Callbacken kører asynkront og kan melde om flere elementer på én gang, og den tjekker `isIntersecting`, fordi observeren også melder om elementer, der ligger uden for området. Fordi `disconnect()` kaldes først, gør callbacken sit arbejde én gang, uanset hvor tit læseren scroller forbi. Indtil da er `route-map.js` slet ikke en del af siden.

Fonte kan deles op, så en browser kun henter de dele, en side bruger, helt uden script. En note her på sitet om en understregningsfejl i Safari beskriver, hvordan sitets skrifttype, Ubuntu, dengang blev leveret som seks filer, tre vægte til latinsk tekst og tre til kyrillisk, hver med sit eget tegnområde. Browseren bruger de

latinske varianter til latinsk tekst og de kyrilliske til kyrillisk tekst. En sprogbetegnelse med kyrilliske tegn kan derfor kræve en kyrillisk variant selv på en side, der ellers er på engelsk eller dansk. Noten kalder det ikke lazy loading, men det samme behovsstyrede valg holder en ubrugt fontvariant ude af sidens fontanmodninger, medmindre den udtrykkeligt indlæses med preload.

De filer, som hver side skal bruge med det samme, kræver den modsatte beslutning. En portfolio-post om at skære stylesheet-bundtet ned forklarer, hvorfor en font, der lå indlejret i stylesheetet som base64, blev taget ud. Den var lagt der for at få den første visning hurtigere frem, men en separat fil med preload giver den samme fordel, mens den indlejrede kopi lagde sin vægt oven i hver eneste side, hvad enten den blev brugt der eller ej. På det tidspunkt blev tre vægte af fonten hentet med preload i stedet, og posten fortæller, at det var målinger fra rigtige besøg og ikke en trang til at gøre rækken hel, der satte den tredje på listen. Lazy loading og preload er to sider af samme vurdering: det, den første skærm skal bruge, kommer tidligt, og resten venter på sin tur.

Hvor det går galt

Det første billede kommer sidst. Et billede med `lazy` venter på mere end netværket. Hentningen genoptages først, når browserens observer melder det tæt på viewporten, og observere bliver først opdateret sent i hver renderingsrunde, efter at stilarter og layout er beregnet. For et fotografi langt nede på siden koster ventetiden ingenting. For det store billede øverst forsinker den netop det billede, som alle besøgende ser. Udelad `loading`, eller sæt den til `eager`, for alt, der kan dukke op på den første skærm på en telefon eller en computer, og gem `lazy` til det, der kommer længere nede.

Siden hopper, når billederne lander. Et billede uden `width` og `height` har ingen kendt form, før filen er nået frem, og skubber så alt under sig nedad. Med lazy loading sker det, mens læseren scroller, ofte midt i en sætning. Giv alle billeder deres mål i markuppen eller i CSS. Om `sizes="auto"` er HTML-standarden helt tydelig: uden angivne mål bliver sådan et billede sandsynligvis tegnet i 300 gange 150 pixels, og standarden opfordrer derfor kraftigt til at angive dem.

Billeder, der kræver et script for at findes. En hjemmelavet løsning lægger den rigtige adresse i en `data-src`-attribut og lader et script kopiere den over i `src`, når billedet nærmer sig. Hvis scriptet ikke bliver indlæst, kaster en fejl eller bliver blokeret, kommer billedet aldrig frem, og alt, der læser markuppen uden at køre scripts, ser et billede uden kilde. Den indbyggede attribut svigter den modsatte

vej: med scripting slået fra lader standarden browseren hente billedet med det samme. Brug `loading="lazy"` til billeder og rammer, og gem scripts til det, attributten ikke dækker.

At udskyde det, hvert besøg skal bruge. Lazy loading betaler sig, når meget af det udskudte aldrig bliver brugt. Når alt bliver brugt, deler det bare én hentning op i mange. En løkke, der læser en relateret post for hvert element i en liste, sender en separat forespørgsel for hvert element: Djangos dokumentation tæller én forespørgsel for listen plus én pr. element i standardtilstanden og kalder det N+1-problemet. En side, hvor hver komponent beder om sine egne data, efterhånden som den dukker op, gør det samme over netværket. Hent det, der med sikkerhed skal bruges, sammen med listen, i én forespørgsel til databasen eller ét kald over netværket, og gem lazy loading til det, der sandsynligvis bliver sprunget over.

Læs mere

- HTML Standard: Lazy loading attributes — nøgleordene `lazy` og `eager`, browserens egen intersection observer og reglen, der slår lazy loading fra uden scripting.
- Intersection Observer — W3C's arbejdsudkast bag observeren: roots, margener, tærskler og hvordan callbacks sættes i kø.
- How to lazy load Client Components and libraries — `next/dynamic`, `React.lazy()` med `Suspense`, at springe rendering på serveren over og at hente et bibliotek med `import()`.
- Lazy Load — Martin Fowlers mønster fra Patterns of Enterprise Application Architecture med dets fire varianter.
- Fetch modes — hvordan Django henter felter, der ikke kom med i den oprindelige forespørgsel, og de tilstande, der samler eller blokerer de hentninger.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er lazy loading?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/lazy-loading/