Engineer

Server-sent events

"Æ" en envejsstrøm af hændelser fra en server over HTTP

Hvad det er

Server-sent events er en måde, hvorpå en webserver kan skubbe beskeder ud til en side gennem ét almindeligt HTTP-svar, som den holder åbent og skriver i, hver gang der er nyt. De er defineret i HTML-standarden, som WHATWG vedligeholder, og den beskriver både strømmens tekstformat og `EventSource`, den grænseflade i browseren, der læser strømmen. Navnet forkortes ofte til SSE, og formatet bliver ofte kaldt efter sin medietype, `text/event-stream`.

En side åbner en strøm med én linje, `new EventSource(url)`. Browseren sender en `GET`-forespørgsel, som kan bære `Accept: text/event-stream`, og forventer et svar med status 200 og den indholdstype. Serveren sender headerfelterne og gør så ikke svaret færdigt. Hver gang der sker noget, skriver den et par linjer tekst, og browseren gør hver blok af linjer til en hændelse, som siden kan lytte efter.

Formatet er bevidst enkelt. En strøm er altid UTF-8, og hver linje er ét felt. En `data`-linje bærer indholdet, og flere `data`-linjer i træk bliver sat sammen med linjeskift. En `event`-linje giver hændelsen en type, og uden den er typen `message`. En `id`-linje sætter ID'et på den seneste hændelse, og en `retry`-linje angiver, hvor mange millisekunder browseren venter, før den genopretter forbindelsen. En linje, der begynder med et kolon, er en kommentar, som browseren ignorerer. En tom linje afslutter blokken og sender hændelsen af sted, så en blok, der bliver afbrudt før sin tomme linje, bliver smidt væk.

Genforbindelse er en del af standarden. Når forbindelsen ryger, venter browseren i genforbindelsestiden, som starter på en værdi, browseren selv vælger, ifølge standarden nok et par sekunder, og sender så forespørgslen igen. Har serveren givet sine hændelser ID'er, bærer den nye forespørgsel det seneste af dem i et `Last-Event-ID`-headerfelt, og serveren kan fortsætte derfra. Nogle svar afslutter i stedet strømmen for altid: en anden status end 200 eller en anden

indholdstype får browseren til at give op og holde op med at prøve, og standarden udpeger 204 No Content som den måde, en server siger, at strømmen er slut, på.

Retningen går kun én vej. Efter den første forespørgsel går der intet fra siden til serveren over den forbindelse; når siden har noget at sige, sender den en separat HTTP-forespørgsel.

Brug begrebet i en beslutning

Brug »Hvad er server-sent events?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Den del, brugerne aldrig ser: hvor data ligger, hvilke regler der gælder, og hvordan andre systemer må tale med den.

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 designet og drevet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tre databasemotorer

Har designet og drevet 6 ETL/ELT-pipelines Kontakt os →

Hvorfor det betyder noget

Mange funktioner, der ser levende ud, har kun brug for én retning: fremskridtet i et langt job, en notifikation, et tal på et dashboard, et svar, der kommer i dele. Uden en strøm må siden polle. Den spørger med få sekunders mellemrum, de fleste svar siger, at intet er ændret, og nyheder venter op til et helt interval, før nogen ser dem. Med en strøm skriver serveren i samme øjeblik, noget sker.

Det sædvanlige alternativ til at skubbe data ud er WebSocket, som skifter forbindelsen væk fra HTTP til en protokol af sin egen. Server-sent events bliver i HTTP fra start til slut, så de passerer de samme proxyer og load balancere som enhver anden forespørgsel, de bruger de samme cookies og de samme regler for forespørgsler på tværs af origins, og de kan dele én HTTP/2-forbindelse med resten af siden. Genforbindelse og genoptagelse fra den seneste hændelse følger med browseren i stedet for at være kode, som hvert team skriver selv. WebSocket er stadig det rigtige valg, når klienten har lige så meget at sige som serveren, som i en chat eller en delt editor.

Formatet er heller ikke bundet til browseren. Enhver HTTP-klient kan læse det, og specifikationer uden for nettet bruger det. I Model Context Protocol, som AI-agenter kalder værktøjer gennem, lader transporten Streamable HTTP en server besvare en forespørgsel enten med ét JSON-objekt eller med et `text/event-stream`-svar. På den strøm må serveren sende notifikationer og egne

forespørgsler før det endelige svar, og den kan sætte hændelses-ID'er på, så en klient, der mister forbindelsen, kan genoptage med `Last-Event-ID`. Streamable HTTP afløste HTTP+SSE, transporten i den tidligere protokolversion 2024-11-05.

Begrænsningerne er værd at kende, før man vælger det. Strømmen bærer kun tekst, så binære data skal først kodes som tekst. Browserens `EventSource`-konstruktør tager en URL og én indstilling, `withCredentials`, og intet andet, så en side kan ikke føje et `Authorization`-headerfelt til forespørgslen.

Sådan ser det ud i praksis

Udvekslingen nedenfor følger et langt eksportjob. Browseren har mistet forbindelsen efter hændelse 41 og genopretter den, så forespørgslen bærer det ID, og serveren fortsætter fra hændelse 42:

```
GET /jobs/4471/events HTTP/1.1
Host: example.com
Accept: text/event-stream
Last-Event-ID: 41

HTTP/1.1 200 OK
Content-Type: text/event-stream
Cache-Control: no-store

retry: 5000

: still working

id: 42
event: progress
data: {"done": 120, "total": 400}

id: 43
event: progress
data: {"done": 400, "total": 400}

id: 44
event: finished
data: {"report": "/jobs/4471/report.pdf"}
```

`retry`-linjen beder browseren vente fem sekunder før enhver senere genforbindelse, og kommentarlinjen bærer slet ingen hændelse; den er der for at holde forbindelsen beskæftiget, mens jobbet er stille. Siden lytter efter de to hændelsestyper ved navn:

```
const source = new EventSource("/jobs/4471/events");
source.addEventListener("progress", (event) => {
  const { done, total } = JSON.parse(event.data);
  showProgress(done / total);
});
source.addEventListener("finished", (event) => {
  showReport(JSON.parse(event.data).report);
  source.close();
});
source.addEventListener("error", () => {
  if (source.readyState === EventSource.CLOSED) showRetryButton();
});
```

Kaldet til `close()` betyder noget. Når serveren afslutter et svar normalt, behandler browseren det som en tabt forbindelse og genopretter den, så en strøm for et færdigt job ellers ville åbne igen efter fem sekunder. `error`-handleren skelner mellem de to slags fejl: så længe `readyState` er `CONNECTING`, prøver browseren stadig, og når den er `CLOSED`, har den givet op.

Arbejde, der kører videre, efter at agenten har fået sit svar, er et af emnerne på siden om din software som værktøjer til AI-agenter her på sitet. Den beskriver, hvordan en virksomheds operationer stilles til rådighed som værktøjer, med en MCP-server, der stiller de samme operationer til rådighed for de agenter, der taler MCP, og hvordan tunge opgaver flyttes over i en kø, så agenten får et svar, mens arbejdet kører videre. I MCP-specifikationen kan en server bruge en hændelsesstrøm som den ovenfor til at sende sine egne forespørgsler eller notifikationer før det endelige svar på klientens forespørgsel.

Hvor det går galt

Hændelser, der kommer i klumper eller alle til sidst. Noget mellem serveren og browseren holder svaret tilbage. nginx, for eksempel, bufferer som standard svarene fra serveren bag sig og sender kun et svar videre, efterhånden som det kommer, når buffering er slået fra. HTML-standarden advarer også om, at HTTP-chunking udført af et lag, der intet ved om strømmens timing, kan gøre den upålidelig. Slå buffering fra for strømmens location, i nginx med `proxy_buffering off` eller med headerfeltet `X-Accel-Buffering: no` i svaret, flush outputtet efter hver hændelse i serverkoden, og test gennem hele vejen, proxy medregnet, frem for mod serveren alene.

Stille strømme, der bliver afbrudt af en timeout. En proxy, der ikke ser nogen bytes i et stykke tid, går ud fra, at forbindelsen er død. I nginx er standardværdien for `proxy_read_timeout` 60 sekunder mellem to læsninger, hvorefter forbindelsen bliver lukket, og HTML-standarden advarer om, at nogle ældre proxyer afbryder forbindelser efter en kort timeout. Den foreslår en kommentarlinje cirka hvert 15. sekund, som koster nogle få bytes, og som browseren ignorerer.

Hændelser, der går tabt ved en genforbindelse. Browseren sender kun `Last-Event-ID`, hvis serveren har givet sine hændelser ID'er, og headerfeltet hjælper kun, hvis serveren stadig kan finde det, der kom efter det ID. Giv hver hændelse et ID, der fungerer som en markør i strømmen, behold et vindue af nylige hændelser til genafspilning, og send hele den aktuelle tilstand, når et ID er for gammelt at genoptage fra. MCP-specifikationen tilføjer en regel, der er værd at kopiere: ID'er hører til én strøm, og en server må ikke genafspille beskeder på én strøm, som var tiltænkt en anden.

Genforbindelse, der aldrig stopper, eller stopper for altid. En strøm, der slutter normalt, kommer tilbage efter genforbindelsestiden, igen og igen, så et færdigt job bliver ved med at genåbne sin strøm, medmindre siden kalder `close()`, eller serveren besvarer genforbindelsen med 204. Det modsatte sker under en udrulning. En proxy, der svarer 502 eller 503, mens serverne genstarter, giver hver åben `EventSource` en anden status end 200, og hver eneste af dem stopper for altid. Lyt efter `error`, og når `readyState` er `CLOSED`, så opret en ny `EventSource` efter en tilfældig forsinkelse, der vokser med hver fejl.

For mange strømme til én browser. Browsere begrænser, hvor mange forbindelser de holder åbne til én server, og en strøm over HTTP/1.1 optager en af dem, så længe den varer. HTML-standarden advarer om, at et site, hvor hver åben side har sin egen `EventSource`, kan løbe ind i den grænse. Servér strømmen over HTTP/2, hvor strømme deler én forbindelse, og RFC 9113 anbefaler at tillade mindst 100 ad gangen, eller del én `EventSource` mellem fanerne gennem en shared worker.

Tokenet i adressen. Fordi `EventSource` ikke kan sende et `Authorization`-headerfelt, er den hurtige løsning et token i URL'en, hvor reverse proxyer og adgangslogge skriver det ned. Autentificér strømmen med den sessionscookie, siden allerede har, og tilføj `withCredentials`, når strømmen kommer fra en anden origin, eller med en kortlivet billet til engangsbrug, som siden henter over en autentificeret forespørgsel, lige før den åbner strømmen.

Læs mere

- HTML Standard: Server-sent events — strømmens format, `EventSource`-grænsefladen, genforbindelse og standardens noter til forfattere om proxyer og forbindelsesgrænser.
- Transports - Model Context Protocol — Streamable HTTP, hvor en server må svare med en hændelsesstrøm, og hvordan en afbrudt strøm bliver genoptaget.
- RFC 9113: HTTP/2 — strømme, der deler én forbindelse, og den indstilling, der begrænser, hvor mange der må være åbne ad gangen.
- nginx: Module ngx_http_proxy_module — `proxy_buffering`, headerfeltet `X-Accel-Buffering` og `proxy_read_timeout`.

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 server-sent events?« 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/server-sent-events/