Engineer

Service worker

et script der svarer på forespørgsler uden netværk

Hvad det er

En service worker er et script, som et websted registrerer hos browseren, og som browseren derefter kører for sig, adskilt fra alle sider; netværksforespørgslerne fra de sider, scriptet styrer, går igennem det. Service workers er en slags web worker: hver af dem kører i sin egen tråd, uden dokument og uden vindue. Browseren starter en, når der er noget, den skal tage sig af, og må stoppe den igen, så snart der ikke er mere at gøre, så alt, hvad den holder i en variabel, kan være væk ved næste hændelse.

En side installerer en med ét kald, og specifikationen stiller tre betingelser for det. Siden skal være en sikker kontekst, hvilket i praksis betyder HTTPS, med `localhost` og loopback-adresserne tilladt under udvikling. Scriptet skal have samme origin som siden, så det kan ikke ligge på et CDN, men det må gerne importere kode derfra. Og registreringen dækker et område, et såkaldt scope: som standard den mappe, scriptet ligger i, og intet bredere, medmindre serveren sender en `Service-Worker-Allowed` -header med scriptet.

```
// på siden
navigator.serviceWorker.register("/sw.js");

// i /sw.js
async function answer(request) {
    const copy = await caches.match(request);
    return copy ?? fetch(request);
}

self.addEventListener("fetch", (event) => {
    event.respondWith(answer(event.request));
});
```

Denne worker svarer på hver forespørgsel med sin egen kopi, når den har en, og ellers fra netværket. Det er livscyklussens opgave at lægge kopierne der. Når scriptet er hentet, udløser browseren `install`, tidspunktet til at hente og gemme det, workeren får brug for; bliver et promise, der er givet til `event.waitUntil()`, afvist, mislykkes hele installationen. Derefter kommer `activate`, tidspunktet til at rydde op efter en ældre version, og først da modtager workeren `fetch`-hændelser. En side kobles til sin worker, når den indlæses, så den side, der registrerede den allerførste worker, kører videre uden, indtil den genindlæses, medmindre workeren kalder `clients.claim()`.

Kopierne ligger i Cache Storage, som samme specifikation definerer: navngivne cacher med par af forespørgsler og svar, som workeren selv fylder, læser og sletter. De er adskilt fra browserens HTTP-cache. En post udløber aldrig, opdateres aldrig af sig selv og overlever en opdatering af workerens script, så den bliver liggende, indtil workeren sletter den, eller browseren rydder webstedets lager.

Brug begrebet i en beslutning

Brug »Hvad er en service worker?« 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

Service workers giver et websted eksplicit kontrol over, hvordan det fungerer helt uden netværk. En service worker afgør forespørgsel for forespørgsel, i kode, webstedet selv har skrevet, om svaret skal komme fra netværket, fra dens egne kopier eller fra begge, og det er det, der gør en installeret webapp brugbar på et skib eller i en kælder. Push-beskeder når også frem til den, når ingen af webstedets sider er åbne.

De fleste workers koger ned til at vælge en strategi for hver slags forespørgsel, og Workbox, et bibliotek til at skrive dem, navngiver fem. Cache først svarer fra kopien og går kun til netværket, når kopien mangler, og egner sig til filer, der

aldrig ændrer sig under samme navn. Netværk først spørger netværket og falder tilbage på kopien og passer til sider. Stale-while-revalidate svarer fra kopien og opdaterer den i baggrunden. Kun netværk og kun cache er de to yderpunkter.

Det er også kode, der står mellem hver side i sit scope og netværket, på læserens enhed, og den overlever besøget: en fejl i den bliver, indtil et rettet script afløser den. Specifikationen er bygget op om den risiko. Hver navigation får browseren til at se efter et nyt script, og andre hændelser gør det, når der er gået et døgn siden sidste tjek. Som standard bliver hovedscriptet genvalideret hos serveren i stedet for at blive taget fra HTTP-cachen, så en lang `max-age` på worker-filen kan ikke holde en defekt version på plads; importerede scripts får samme behandling, når der er gået et døgn, eller altid, når `updateViaCache` er `none`. Browseren sammenligner det hentede script med det installerede byte for byte, og kun en forskel sætter en opdatering i gang. Et shift-reload springer workeren over for den ene navigation, og derfor kan en fejl se rettet ud på en udviklers maskine.

Reglen om at vente er den anden halvdel af det design. En ny version installeres ved siden af den gamle og venter derefter, ubrugt, indtil ingen åben side længere afhænger af den gamle, så to versioner aldrig deles om at betjene én side. Kalder workeren `skipWaiting()`, tager den over med det samme, og prisen er, at ny kode kører under sider, som den gamle har indlæst.

Lager er den sidste omkostning. Cache Storage tæller med i den plads, webstedet optager på læserens enhed, og efter Storage Standard er den plads som standard best-effort: browseren må rydde den, når enheden mangler plads, medmindre webstedet har bedt om vedvarende lager og fået det.

Sådan ser det ud i praksis

Behovet er tydeligst, hvor netværket er svagest. Den progressive webapp til søfarende og feltmedarbejdere betjener folk, der ofte er offline eller på en dårlig forbindelse. Dens service worker er sat op via next-pwa-pluginet, og Workbox' runtime-caching bruger cache først til det, der er det samme fra forespørgsel til forespørgsel, altså skrifttyper, billeder, lyd, video og CDN-filer, side om side med HTTP-headere til cachestyring. Fordi service workers kun for alvor fungerer over HTTPS, blev offline- og installationsadfærden testet over HTTPS på rigtige enheder, og gentagne indlæsninger kommer fra cachen i stedet for fra netværket.

Dette websted vælger det modsatte og siger det. Cookiesiden oplyser, at webstedet hverken har en service worker, IndexedDB eller Cache Storage: intet fra det kører i browserens baggrund, før læseren åbner det, eller efter det lukkes. Siderne leverer ingen JavaScript til afvikling i browseren. Søgekoden er bevaret, men slået fra, og registrerer ingen worker.

Installationssiden drager konsekvensen for alle, der lægger webstedet på en hjemmeskærm: installationen giver det sit eget vindue uden at oprette en administreret offlinekopi. Dette websted leverer hverken en service worker eller scriptstyret lagring til offlinebrug, så den installerede app kræver forbindelse ved normal brug.

En worker registreres fra et script, der skal køres, og det forbyder sitets nuværende Content-Security-Policy med `script-src 'none'` på hver side. JSON-LD er data og bliver ikke kørt. En tidligere kommentar i head-skabelonen beskrev et byg, hvor 550 af 793 dokumenter angav `'none'` og 243 angav `'self'`, svarende til siderne med listefilteret dengang. Tallene beskriver det tidligere byg med script, ikke den nuværende politik.

Hvor det går galt

Den gamle version bliver ved med at svare efter en udrulning. Den nye worker installeres og venter derefter, så længe en åben fane stadig bruger den gamle, så en læser, der aldrig lukker den fane, bliver ved med at køre den gamle kode. Kald `skipWaiting()`, når den nye version er sikker under sider, som den gamle har indlæst, eller vis en besked om, at en opdatering er klar, og lad siden bede den ventende worker om at kalde den.

Sider bliver hængende i en gammel version i ugevis. En cache-først-strategi brugt på HTML besvarer hver navigation fra Cache Storage, hvis poster aldrig udløber og ser bort fra serverens cache-headere. Brug kun cache først til filer, hvis navn skifter, når indholdet gør, og netværk først til sider.

Installationen mislykkes, og den gamle worker bliver. Én adresse på installationstrinnets liste svarer med en fejl, så `cache.addAll()` afviser, det promise, der er givet til `waitUntil()`, afvises med det, og den nye version kasseres. Byg listen ud fra filer, der findes, og hent valgfrie filer senere i stedet for under installationen.

Workeren overser det meste af webstedet. Et script, der serveres fra `/js/sw.js`, har `/js/` som både sit standard-scope og sit bredeste scope, så sider uden for den mappe indlæses uden det. Servér scriptet fra webstedets rod, eller send `Service-Worker-Allowed: /` med det.

Lageret på enheden vokser med hver udgivelse. Cacher overlever en opdatering af workerens script, så hver version, der åbner en cache under et nyt navn, efterlader den forrige. Sæt et versionsnummer i hvert cachenavn, og slet i `activate`-handleren hver cache, som den aktuelle version ikke bruger.

Læs mere

- Service Workers — registrering, scope, opdateringstjekket, livscyklussen og Cache Storage.
- Secure Contexts — hvilke origins der regnes for potentielt troværdige, `localhost` og loopback-adresserne medregnet.
- Fetch Standard — de `Request`- og `Response`-objekter, en `fetch`-handler modtager og returnerer, og hvordan en forespørgsel gives videre til en service worker.
- Storage Standard — de storage buckets, Cache Storage ligger i, kvoteoverslag og hvordan et websted beder om vedvarende lager.
- Workbox: workbox-strategies — de fem caching-strategier og den slags forespørgsel, hver af dem passer til.

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 en service worker?« 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/service-worker/