Engineer

service worker

Script pouvant répondre aux requêtes sans réseau

Ce que c’est

Un service worker est un script qu’un site enregistre auprès du navigateur. Celui-ci l’exécute indépendamment de toute page et lui fait traiter les requêtes réseau des pages contrôlées par ce worker. Les service workers sont une catégorie de web workers : chacun s’exécute sur son propre fil d’exécution, sans document ni fenêtre. Le navigateur en démarre un lorsqu’il a quelque chose à traiter et peut l’arrêter dès qu’il ne reste rien à faire. Une valeur conservée dans une variable peut donc avoir disparu à l’événement suivant.

Une page en installe un par un seul appel, soumis à trois conditions dans la spécification. La page doit être un contexte sécurisé, ce qui signifie en pratique HTTPS, avec `localhost` et les adresses de bouclage autorisés pour le développement. Le script doit provenir de la même origine que la page : il ne peut donc pas résider sur un CDN, même s’il peut en importer du code. Enfin, l’enregistrement porte sur un périmètre : par défaut, le dossier du script, sans pouvoir l’élargir à moins que le serveur n’envoie un en-tête `Service-Worker-Allowed` avec ce script.

```
// on the page
navigator.serviceWorker.register("/sw.js");

// in /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));
});
```

Ce worker répond à chaque requête avec sa propre copie s'il en possède une, et par le réseau sinon. C'est le cycle de vie qui permet de constituer ces copies. Une fois le script récupéré, le navigateur déclenche `install` : c'est le moment de télécharger et de stocker ce dont le worker aura besoin. Si une promesse passée à `event.waitUntil()` y est rejetée, l'installation échoue. Vient ensuite `activate`, le moment de supprimer ce qu'une ancienne version a laissé. Ce n'est qu'après cette étape que le worker reçoit des événements `fetch`. Une page est associée à son worker au chargement : celle qui a enregistré le premier worker continue donc sans lui jusqu'à son rechargement, sauf si le worker appelle `clients.claim()`.

Les copies résident dans Cache Storage, également défini par cette spécification : des caches nommés contenant des paires requête-réponse, que le worker remplit, consulte et supprime lui-même. Ils sont distincts du cache HTTP du navigateur. Une entrée n'expire jamais, ne s'actualise jamais seule et survit à une mise à jour du script du worker. Elle reste donc jusqu'à sa suppression par le worker ou jusqu'à ce que le navigateur efface le stockage du site.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'un service worker ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des interfaces rapides, utilisables sur téléphone et cohérentes dans tous les navigateurs de vos clients.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Développement cartographique GIS full-stack — cartographie full-stack, de la base au navigateur

Développement cartographique GIS full-stack Contactez-nous →

Pourquoi c'est important

Les service workers permettent à un site de contrôler explicitement la manière dont il continue de fonctionner sans aucun réseau. Pour chaque requête, un service worker décide, avec du code écrit par le site, de répondre par le réseau, par ses propres copies ou par les deux. C'est ce qui rend une application web installée utilisable sur un navire ou dans un sous-sol. Les messages push lui parviennent aussi lorsqu'aucune page du site n'est ouverte.

La plupart des workers consistent à choisir une stratégie par type de requête. Workbox, une bibliothèque pour les écrire, en nomme cinq. La priorité au cache (cache first) répond avec la copie et ne sollicite le réseau que s'il n'y en a pas : cela convient aux fichiers qui ne changent jamais sous un même nom. La priorité au

réseau (network first) interroge le réseau et se rabat sur la copie : cela convient aux pages. La réutilisation avec revalidation (stale-while-revalidate) répond avec la copie et l'actualise en arrière-plan. Les deux extrêmes sont « réseau seulement » (network only) et « cache seulement » (cache only).

Ce code se place aussi entre chaque page de son périmètre et le réseau, sur l'appareil du lecteur, et survit à la visite : une erreur reste en place jusqu'à son remplacement par un script corrigé. La spécification est conçue autour de ce risque. Chaque navigation fait vérifier au navigateur la présence d'un nouveau script ; d'autres événements déclenchent cette vérification lorsqu'une journée s'est écoulée depuis la précédente. Par défaut, le script principal est revalidé auprès du serveur plutôt que pris dans le cache HTTP : un long `max-age` sur le fichier du worker ne peut donc pas maintenir une version défectueuse. Les scripts importés reçoivent le même traitement après une journée, ou systématiquement lorsque `updateViaCache` vaut `none`. Le navigateur compare le script récupéré à celui qui est installé, octet par octet, et seule une différence déclenche une mise à jour. Un rechargement avec la touche Maj ignore le worker pour cette seule navigation : une erreur peut ainsi sembler corrigée sur la machine du développeur.

La règle d'attente est l'autre moitié de ce dispositif. Une nouvelle version s'installe à côté de l'ancienne, puis attend sans être utilisée jusqu'à ce qu'aucune page ouverte ne dépende encore de l'ancienne. Deux versions ne prennent ainsi jamais en charge une même page à elles deux. L'appel à `skipWaiting()` permet une prise de relais immédiate, au prix de l'exécution du nouveau code sous des pages chargées par l'ancien.

Le stockage est le dernier coût. Cache Storage est comptabilisé dans l'espace occupé par le site sur l'appareil du lecteur. Selon Storage Standard, sa conservation n'est pas garantie par défaut : le navigateur peut l'effacer si l'appareil manque de place, sauf si le site a demandé et obtenu un stockage persistant.

À quoi cela ressemble en pratique

L'intérêt est le plus évident là où le réseau est le plus faible. L'application web progressive construite pour les marins et le personnel de terrain s'adresse à des personnes souvent hors ligne ou mal connectées. Son service worker est configuré avec le plugin next-pwa. Le cache à l'exécution de Workbox applique cache first à ce qui ne change pas d'une requête à l'autre — polices, images,

audio, vidéo et ressources de CDN — en complément des en-têtes HTTP de contrôle du cache. Les service workers ne se comportant réellement comme prévu qu'en HTTPS, le fonctionnement hors ligne et l'installation ont été testés en HTTPS sur de vrais appareils. Les chargements suivants utilisent le cache au lieu du réseau.

Ce site fait le choix inverse et l'annonce. La page consacrée aux cookies précise qu'il n'utilise ni service worker, ni IndexedDB, ni Cache Storage : rien de ce site ne s'exécute en arrière-plan avant son ouverture ou après sa fermeture par le lecteur. Ses pages ne livrent aucun JavaScript exécutable. Le code de recherche est conservé, mais désactivé, et n'enregistre aucun worker.

La page d'installation en tire la conséquence pour toute personne qui ajoute le site à son écran d'accueil : l'installation lui donne sa propre fenêtre sans créer de copie hors ligne gérée. Ce site ne fournit ni service worker ni stockage hors ligne géré par script ; l'application installée a donc besoin d'une connexion pour une utilisation normale.

L'enregistrement d'un worker se fait depuis un script exécutable, ce que la Content-Security-Policy de ce site interdit désormais avec `script-src 'none'` sur chaque page. JSON-LD reste une donnée inerte. Un ancien commentaire du modèle d'en-tête consignait une version du site où 550 des 793 documents indiquaient `'none'` et 243 indiquaient `'self'`, ce qui correspondait alors aux pages portant le filtre de liste. Ces chiffres décrivent l'ancienne version avec scripts, pas la politique actuelle.

Où les problèmes surviennent

L'ancienne version continue de répondre après un déploiement. Le nouveau worker s'installe, puis attend tant qu'un onglet ouvert utilise encore l'ancien. Un lecteur qui ne ferme jamais cet onglet continue donc d'exécuter l'ancien code. Appelez `skipWaiting()` lorsque la nouvelle version peut fonctionner sans risque sous des pages chargées par l'ancienne, ou affichez un avis de mise à jour et faites envoyer par la page un message au worker en attente pour lui demander cet appel.

Les pages restent à une ancienne version pendant des semaines. Une stratégie qui privilégie le cache pour le HTML répond à chaque navigation depuis Cache Storage, dont les entrées n'expirent jamais et ignorent les en-têtes de cache du

serveur. Réservez la priorité au cache aux fichiers dont le nom change avec le contenu, et la priorité au réseau aux pages.

L'installation échoue et l'ancien worker reste en place. Une adresse de la liste utilisée à l'installation renvoie une erreur : la promesse renvoyée par `cache.addAll()` est alors rejetée, entraînant le rejet de celle passée à `waitUntil()`, et la nouvelle version est abandonnée. Construisez cette liste à partir de fichiers existants et récupérez les fichiers facultatifs plus tard, plutôt qu'à l'installation.

Le worker ignore la majeure partie du site. Un script servi depuis `/js/sw.js` a `/js/` comme périmètre par défaut et comme périmètre maximal : les pages hors de ce dossier se chargent sans lui. Servez le script à la racine du site ou envoyez `Service-Worker-Allowed: /` avec lui.

Le stockage sur l'appareil augmente à chaque version. Les caches survivent à une mise à jour du script du worker. Chaque version qui ouvre un cache sous un nouveau nom laisse donc le précédent en place. Mettez une version dans chaque nom de cache et, dans le gestionnaire `activate`, supprimez tout cache que la version actuelle n'utilise pas.

Pour aller plus loin

- Service Workers — enregistrement, périmètre, vérification des mises à jour, cycle de vie et Cache Storage.
- Contextes sécurisés — les origines considérées comme potentiellement fiables, y compris `localhost` et les adresses de bouclage.
- Fetch Standard — les objets `Request` et `Response` reçus et renvoyés par un gestionnaire `fetch`, et le passage d'une requête à un service worker.
- Storage Standard — les compartiments de stockage contenant Cache Storage, les estimations de quota et la demande de stockage persistant par un site.
- Workbox : workbox-strategies — les cinq stratégies de cache et le type de requête auquel chacune convient.



Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'un service worker ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/service-worker/