Engineer

Service Worker

Ein Skript, das Anfragen auch ohne Netz beantwortet

Was es ist

Ein Service Worker ist ein Skript, das eine Website beim Browser registriert. Der Browser führt es getrennt von jeder Seite aus und leitet die Netzwerkanfragen der vom Service Worker kontrollierten Seiten durch dieses Skript. Service Worker sind eine Art Web Worker: Jeder läuft in einem eigenen Thread ohne Dokument und ohne Fenster. Der Browser startet ihn, wenn etwas zu bearbeiten ist, und darf ihn beenden, sobald nichts mehr ansteht. Was in einer Variablen steht, kann deshalb beim nächsten Ereignis schon verschwunden sein.

Eine Seite installiert ihn mit einem einzigen Aufruf. Die Spezifikation stellt drei Bedingungen. Die Seite muss ein sicherer Kontext sein, praktisch also HTTPS, wobei `localhost` und Loopback-Adressen für die Entwicklung zulässig sind. Das Skript muss vom selben Origin wie die Seite stammen. Es kann daher nicht auf einem CDN liegen, darf aber Code von dort importieren. Außerdem hat die Registrierung einen Geltungsbereich: standardmäßig den Ordner des Skripts und keinen größeren, sofern der Server nicht zusammen mit dem Skript einen Header `Service-Worker-Allowed` sendet.

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

Dieser Worker beantwortet jede Anfrage mit seiner eigenen Kopie, falls er eine hat, andernfalls aus dem Netz. Die Kopien bereitzulegen ist Aufgabe des Lebenszyklus. Nach dem Abruf des Skripts löst der Browser `install` aus. Jetzt wird heruntergeladen und gespeichert, was der Worker brauchen wird. Wird ein dort an `event.waitUntil()` übergebenes Promise abgelehnt, scheitert die Installation. Danach folgt `activate`, der Zeitpunkt zum Aufräumen der Reste einer älteren Version. Erst anschließend erhält der Worker `fetch`-Ereignisse. Eine Seite wird beim Laden ihrem Worker zugeordnet. Die Seite, die den ersten Worker registriert hat, arbeitet deshalb bis zum Neuladen ohne ihn weiter, sofern der Worker nicht `clients.claim()` aufruft.

Die Kopien liegen in Cache Storage. Dieselbe Spezifikation definiert diesen Speicher als benannte Caches mit Anfrage-Antwort-Paaren, die der Worker selbst füllt, liest und löscht. Sie sind vom HTTP-Cache des Browsers getrennt. Ein Eintrag läuft nie ab, aktualisiert sich nie selbst und überlebt ein Update des Worker-Skripts. Er bleibt also liegen, bis der Worker ihn löscht oder der Browser den Website-Speicher leert.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Service Worker?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Oberflächen, die schnell laden, auf dem Smartphone funktionieren und sich in allen Browsern Ihrer Kunden gleich verhalten.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Full-Stack-GIS-Kartenentwicklung geleitet — Full-Stack-Kartenentwicklung von der Datenbank bis zum Browser

Full-Stack-GIS-Kartenentwicklung geleitet Kontakt aufnehmen →

Warum es wichtig ist

Service Worker geben einer Website explizite Kontrolle darüber, wie sie ohne jede Netzverbindung weiterarbeitet. Ein Service Worker entscheidet für jede Anfrage mit vom Betreiber geschriebenem Code, ob er sie aus dem Netz, aus eigenen Kopien oder aus einer Kombination beider Quellen beantwortet. Dadurch lässt sich eine installierte Web-App auf einem Schiff oder in einem Keller nutzen. Auch Push-Nachrichten erreichen ihn, wenn keine Seite der Website geöffnet ist.

Bei den meisten Workern läuft es auf die Wahl einer Strategie pro Art von Anfrage hinaus. Workbox, eine Bibliothek zum Schreiben solcher Worker, benennt fünf. Cache first antwortet aus der Kopie und greift nur bei fehlender Kopie aufs Netz zu. Das passt zu Dateien, die sich unter demselben Namen nie ändern. Network first fragt das Netz und fällt bei einem Fehler auf die Kopie zurück. Das passt zu Seiten. Stale-while-revalidate antwortet aus der Kopie und aktualisiert sie im Hintergrund. Network only und cache only sind die beiden Extreme.

Zugleich steht dieser Code auf dem Gerät des Lesers zwischen jeder Seite seines Geltungsbereichs und dem Netz und überlebt den Besuch. Ein Fehler bleibt also, bis ein korrigiertes Skript ihn ersetzt. Die Spezifikation ist um dieses Risiko herum entworfen. Bei jeder Navigation prüft der Browser auf ein neues Skript. Andere Ereignisse tun das, wenn seit der letzten Prüfung ein Tag vergangen ist. Standardmäßig wird das Hauptskript beim Server revalidiert statt aus dem HTTP-Cache bezogen. Ein langes `max-age` auf der Worker-Datei kann dadurch keine fehlerhafte Version festhalten. Importierte Skripte werden ebenso behandelt, wenn ein Tag vergangen ist, oder immer, wenn `updateViaCache` auf `none` steht. Der Browser vergleicht abgerufenes und installiertes Skript Byte für Byte. Nur ein Unterschied startet die Aktualisierung. Ein Neuladen mit Umschalttaste umgeht den Worker für diese Navigation. Deshalb kann ein Fehler auf dem Entwicklungsrechner behoben wirken.

Die Warteregel ist die andere Hälfte des Entwurfs. Eine neue Version wird neben der alten installiert und wartet ungenutzt, bis keine offene Seite mehr von der alten abhängt. So bedienen nicht zwei Versionen gemeinsam dieselbe Seite. Ein Aufruf von `skipWaiting()` erlaubt die sofortige Übernahme, allerdings läuft dann neuer Code unter Seiten, die noch die vorherige Version geladen hat.

Der Speicher ist der letzte Kostenpunkt. Cache Storage verbraucht Platz im Website-Speicher auf dem Gerät des Lesers. Nach dem Storage Standard wird dieser Platz standardmäßig ohne dauerhafte Zusicherung gewährt. Bei Platzmangel auf dem Gerät darf der Browser ihn löschen, sofern die Website nicht persistenten Speicher beantragt und bewilligt bekommen hat.

Wie es in der Praxis aussieht

Der Nutzen zeigt sich am deutlichsten dort, wo das Netz am schwächsten ist. Die progressive Web-App für Seeleute und Außendienstkräfte wird von Menschen genutzt, die oft ohne oder nur mit schlechter Verbindung unterwegs sind. Ihr

Service Worker wird über das Plugin next-pwa konfiguriert. Workbox verwendet im Laufzeitcache cache first für Dinge, die sich zwischen Anfragen nicht ändern: Schriftarten, Bilder, Audio, Video und CDN-Ressourcen, zusammen mit HTTP-Headern zur Cache-Steuerung. Weil Service Worker erst über HTTPS richtig funktionieren, wurden Offline- und Installationsverhalten über HTTPS auf echten Geräten getestet. Wiederholte Ladevorgänge kommen aus dem Cache statt aus dem Netz.

Diese Website entscheidet sich ausdrücklich für das Gegenteil. Die Cookie-Seite hält fest, dass es weder Service Worker noch IndexedDB noch Cache Storage gibt: Nichts von der Website läuft im Hintergrund, bevor jemand sie öffnet oder nachdem sie geschlossen wurde. Ihre Seiten liefern kein ausführbares JavaScript aus. Der Suchcode bleibt deaktiviert erhalten und registriert keinen Worker.

Die Installationsseite erklärt die Folge für alle, die die Website zum Startbildschirm hinzufügen: Die Installation gibt ihr ein eigenes Fenster, legt aber keine verwaltete Offline-Kopie an. Diese Website liefert weder einen Service Worker noch skriptgesteuerten Offline-Speicher aus. Die installierte App braucht daher für die normale Nutzung eine Verbindung.

Ein Worker wird durch ein ausführbares Skript registriert. Die Content-Security-Policy dieser Website verbietet das inzwischen auf allen Seiten mit `script-src 'none'` . JSON-LD bleibt auf inerte Daten beschränkt. Ein älterer Kommentar in der Head-Vorlage dokumentierte einen Build, in dem 550 von 793 Dokumenten `'none'` und 243 `'self'` deklarierten. Letztere entsprachen den Seiten mit dem damals enthaltenen Listenfilter. Diese Zahlen beschreiben den früheren Build mit aktivierten Skripten und nicht die aktuelle Richtlinie.

Wo es schiefgeht

Nach einem Deployment antwortet weiterhin die alte Version. Der neue Worker installiert sich und wartet dann, solange noch ein offener Tab den vorherigen verwendet. Wer diesen Tab nie schließt, führt weiter den alten Code aus. Rufen Sie `skipWaiting()` auf, wenn die neue Version sicher unter Seiten laufen kann, die die vorherige geladen hat. Oder zeigen Sie eine Aktualisierungsmeldung an und lassen Sie die Seite dem wartenden Worker eine Nachricht schicken, damit er den Aufruf ausführt.

Seiten bleiben wochenlang auf dem alten Stand. Eine Cache-first-Strategie für HTML beantwortet jede Navigation aus Cache Storage. Dessen Einträge laufen nie ab und beachten die Cache-Header des Servers nicht. Reservieren Sie cache first für Dateien, deren Name sich mit ihrem Inhalt ändert, und verwenden Sie network first für Seiten.

Die Installation scheitert, und der alte Worker bleibt. Eine Adresse aus der Liste des Installationsschritts antwortet mit einem Fehler. Dadurch wird `cache.addAll()` abgelehnt, ebenso das an `waitUntil()` übergebene Promise. Die neue Version wird verworfen. Bauen Sie die Liste aus vorhandenen Dateien auf und laden Sie optionale Dateien später statt während der Installation.

Der Worker ignoriert den größten Teil der Website. Ein unter `/js/sw.js` ausgeliefertes Skript hat `/js/` sowohl als standardmäßigen als auch als maximalen Geltungsbereich. Seiten außerhalb dieses Ordners laden ohne ihn. Liefern Sie das Skript an der Website-Wurzel aus oder senden Sie damit `Service-Worker-Allowed: /` .

Die Speicherbelegung auf dem Gerät wächst mit jeder Veröffentlichung. Caches überleben ein Update des Worker-Skripts. Jede Version, die einen Cache unter neuem Namen öffnet, lässt daher den vorherigen zurück. Nehmen Sie eine Version in jeden Cache-Namen auf und löschen Sie im Handler für `activate` alle Caches, die die aktuelle Version nicht verwendet.

Weiterführende Lektüre

- Service Workers — Registrierung, Geltungsbereich, Aktualisierungsprüfungen, Lebenszyklus und Cache Storage.
- Sichere Kontexte — Welche Origins als potenziell vertrauenswürdig gelten, einschließlich `localhost` und Loopback-Adressen.
- Fetch Standard — Die Objekte `Request` und `Response` , die ein `fetch` -Handler erhält und zurückgibt, und die Übergabe einer Anfrage an einen Service Worker.
- Storage Standard — Die Speicher-Buckets für Cache Storage, Quotenschätzungen und der Antrag einer Website auf persistenten Speicher.
- Workbox: workbox-strategies — Die fünf Cache-Strategien und die jeweils passende Art von Anfrage.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ein Service Worker?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/service-worker/