Engineer

service worker

Script que responde a solicitudes cuando la red no puede

Qué es

Un service worker es un script que un sitio registra en el navegador. Este lo ejecuta separado de cualquier página y hace pasar por él las solicitudes de red de las páginas que controla el service worker. Los service workers son una clase de web worker: cada uno se ejecuta en su propio hilo, sin documento ni ventana. El navegador lo inicia cuando hay algo que atender y puede detenerlo en cuanto no queda trabajo. Por eso lo que conserve en una variable puede haber desaparecido para el siguiente evento.

Una página lo instala con una sola llamada, y la especificación impone tres condiciones. La página debe ser un contexto seguro, lo que en la práctica significa HTTPS, con `localhost` y las direcciones de bucle local permitidos para desarrollo. El script debe proceder del mismo origen que la página, por lo que no puede alojarse en una CDN, aunque sí puede importar código desde ella. El registro también tiene un ámbito: por defecto, la carpeta donde está el script, sin ampliarlo salvo que el servidor envíe una cabecera `Service-Worker-Allowed` con el 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));
});
```

Ese worker responde a cada solicitud con su propia copia si dispone de ella, y acude a la red en caso contrario. Colocar allí las copias es tarea del ciclo de vida. Tras obtener el script, el navegador dispara `install`, el momento de descargar y guardar lo que necesitará el worker. Si se rechaza una promesa entregada allí a `event.waitUntil()`, la instalación falla. Después se dispara `activate`, el momento de eliminar lo que dejó una versión anterior. Solo entonces recibe el worker eventos `fetch`. Una página se vincula a su worker al cargarse. Por eso la página que registró el primer worker sigue sin él hasta que se recarga, salvo que el worker llame a `clients.claim()`.

Las copias viven en Cache Storage, definido por la misma especificación: cachés con nombre que contienen pares de solicitudes y respuestas, y que el propio worker rellena, lee y elimina. Son independientes de la caché HTTP del navegador. Una entrada nunca caduca, nunca se actualiza sola y sobrevive a una actualización del script del worker. Permanece hasta que el worker la elimina o el navegador borra el almacenamiento del sitio.

Utilice el concepto en una decisión

Aplique «¿Qué es un service worker?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Interfaces que cargan rápido, funcionan en el teléfono y se comportan igual en los navegadores que utilizan sus clientes.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Dirección del desarrollo integral de mapas GIS — cartografía integral, de la base de datos al navegador

Dirección del desarrollo integral de mapas GIS Póngase en contacto →

Por qué importa

Los service workers dan a un sitio control explícito sobre cómo sigue funcionando sin conexión a la red. Un service worker decide, solicitud por solicitud y con código escrito por el sitio, si responde desde la red, desde sus propias copias o combinando ambas opciones. Esto permite utilizar una aplicación web instalada en un barco o en un sótano. También recibe mensajes push cuando no hay ninguna página del sitio abierta.

En la mayoría de los workers, lo esencial es elegir una estrategia para cada clase de solicitud. Workbox, una biblioteca para escribirlos, da nombre a cinco. Cache first responde con la copia y solo acude a la red si no hay ninguna: encaja con archivos que no cambian mientras conservan el mismo nombre. Network first

pregunta a la red y recurre a la copia si falla: encaja con las páginas. Stale-while-revalidate responde con la copia y la actualiza en segundo plano. Network only y cache only son los dos extremos.

También es código situado entre todas las páginas de su ámbito y la red, en el dispositivo del lector, y sobrevive a la visita: un error permanece hasta que lo sustituye un script corregido. La especificación se diseñó en torno a ese riesgo. Cada navegación hace que el navegador compruebe si hay un script nuevo, y otros eventos lo hacen cuando ha pasado un día desde la última comprobación. Por defecto, el script principal se revalida con el servidor en lugar de obtenerse de la caché HTTP. Así, un `max-age` largo en el archivo del worker no puede mantener una versión defectuosa. Los scripts importados reciben el mismo tratamiento cuando ha pasado un día, o siempre si `updateViaCache` es `none`. El navegador compara el script obtenido con el instalado byte a byte, y solo una diferencia inicia la actualización. Una recarga con Mayús omite el worker para esa navegación. Por eso un fallo puede parecer resuelto en la máquina de quien desarrolla.

La regla de espera es la otra mitad del diseño. Una versión nueva se instala junto a la anterior y espera sin utilizarse hasta que ninguna página abierta dependa de la antigua. Así, dos versiones no sirven una misma página entre ambas. Llamar a `skipWaiting()` permite tomar el control inmediatamente, a costa de ejecutar código nuevo bajo páginas que cargó la versión anterior.

El almacenamiento es el último coste. Cache Storage consume el espacio que el sitio ocupa en el dispositivo del lector. Según el Estándar de Almacenamiento, ese espacio se ofrece por defecto sin garantía de conservación: el navegador puede borrarlo si falta espacio en el dispositivo, salvo que el sitio haya solicitado almacenamiento persistente y se le haya concedido.

Cómo se aplica en la práctica

La utilidad se ve con mayor claridad donde la red es más débil. La aplicación web progresiva creada para gente de mar y personal de campo atiende a personas que a menudo están sin conexión o tienen una conexión deficiente. Su service worker se configura mediante el plugin next-pwa, y la caché en tiempo de ejecución de Workbox aplica cache first a lo que no cambia entre solicitudes: fuentes, imágenes, audio, vídeo y recursos de CDN, junto con cabeceras HTTP de control de caché. Como los service workers solo funcionan realmente como

deben sobre HTTPS, el comportamiento sin conexión y de instalación se probó sobre HTTPS en dispositivos reales. Las cargas repetidas vuelven desde la caché, en lugar de la red.

Este sitio toma la decisión contraria y la declara. La página de cookies indica que no hay service worker, IndexedDB ni Cache Storage: nada del sitio se ejecuta en segundo plano antes de que el lector lo abra o después de que lo cierre. Sus páginas no incluyen JavaScript ejecutable. El código de búsqueda se conserva desactivado y no registra ningún worker.

La página de instalación explica la consecuencia para quien añada el sitio a su pantalla de inicio: la instalación le da una ventana propia sin crear una copia sin conexión gestionada. Este sitio no incluye ningún service worker ni almacenamiento sin conexión gestionado por scripts, por lo que la aplicación instalada necesita conexión para su uso normal.

Un worker se registra desde un script ejecutable, que la política Content-Security-Policy de este sitio prohíbe ahora con `script-src 'none'` en todas las páginas. JSON-LD se mantiene como datos inertes. Un comentario anterior de la plantilla de cabecera documentó una compilación en la que 550 de 793 documentos declaraban `'none'` y 243 declaraban `'self'`, coincidiendo con las páginas que entonces incluían el filtro de listas. Esas cifras describen aquella compilación anterior con scripts habilitados, no la política actual.

Dónde surgen los problemas

La versión antigua sigue respondiendo después de un despliegue. El worker nuevo se instala y después espera mientras alguna pestaña abierta siga utilizando el anterior. Quien nunca cierre esa pestaña seguirá ejecutando el código antiguo. Llame a `skipWaiting()` cuando la versión nueva pueda ejecutarse con seguridad bajo páginas cargadas por la anterior, o muestre un aviso de actualización y haga que la página envíe un mensaje al worker en espera para que lo llame.

Las páginas permanecen semanas en una versión antigua. Una estrategia cache first aplicada al HTML responde a cada navegación desde Cache Storage. Sus entradas nunca caducan y no atienden a las cabeceras de caché del servidor. Reserve cache first para los archivos cuyo nombre cambia cuando lo hace su contenido y utilice network first para las páginas.

La instalación falla y permanece el worker antiguo. Una dirección de la lista del paso de instalación responde con un error. Por eso se rechaza `cache.addAll()`, también se rechaza la promesa entregada a `waitUntil()` y se descarta la versión nueva. Construya la lista a partir de archivos que existan y obtenga los opcionales más adelante, en lugar de hacerlo durante la instalación.

El worker ignora la mayor parte del sitio. Un script servido desde `/js/sw.js` tiene `/js/` como ámbito predeterminado y también como máximo. Las páginas fuera de esa carpeta se cargan sin él. Sirva el script desde la raíz del sitio o envíe `Service-Worker-Allowed: /` con él.

El espacio ocupado en el dispositivo crece con cada publicación. Las cachés sobreviven a una actualización del script del worker. Por eso cada versión que abre una caché con un nombre nuevo deja la anterior atrás. Incluya una versión en cada nombre de caché y elimine en el manejador `activate` todas las cachés que la versión actual no utilice.

Lecturas adicionales

- Service Workers — Registro, ámbito, comprobación de actualizaciones, ciclo de vida y Cache Storage.
- Contextos seguros — Qué orígenes se consideran potencialmente fiables, incluidos `localhost` y las direcciones de bucle local.
- Estándar Fetch — Los objetos `Request` y `Response` que recibe y devuelve un manejador `fetch`, y cómo se entrega una solicitud a un service worker.
- Estándar de Almacenamiento — Los contenedores de almacenamiento donde reside Cache Storage, las estimaciones de cuota y cómo solicita un sitio almacenamiento persistente.
- Workbox: workbox-strategies — Las cinco estrategias de caché y la clase de solicitud adecuada para cada una.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es un service worker?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

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

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/service-worker/