Engineer

サービスワーカー

ネットワークが使えないときもリクエストに応答できるスクリプト

どのようなものか

サービスワーカーとは、サイトがブラウザーに登録するスクリプトです。ブラウザーはそれをどのページからも独立して実行し、そのワーカーが制御するページのネットワークリクエストを、ワーカー経由で処理します。サービスワーカーはウェブワーカーの一種で、それぞれ専用のスレッド上で動作し、ドキュメントもウィンドウも持ちません。ブラウザーは処理すべきものがあるとワーカーを起動し、なくなれば直ちに停止することもあるため、変数に保持した値は次のイベントまでに失われる場合があります。

ページは一度の呼び出しでサービスワーカーをインストールでき、仕様はそのための条件を三つ定めています。ページはセキュアコンテキストでなければなりません。実際にはHTTPSを意味しますが、開発用には `localhost` とループバックアドレスも認められています。スクリプトはページ自身のオリジンから取得する必要があり、CDNからコードをインポートすることはできても、スクリプト自体をCDNには置けません。また、登録にはスコープがあります。既定ではスクリプトがあるフォルダーであり、サーバーがスクリプトとともに `Service-Worker-Allowed` ヘッダーを送らない限り、それより広くはできません。

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

このワーカーは、コピーを持っていればそれを使って各リクエストに応答し、なければネットワークから応答を得ます。コピーを用意するのはライフサイクルの役目です。スクリプトを取得すると、ブラウザーは `install` を発火します。ワーカーが必要とするものをダウンロードして保存する段階です。ここで `event.waitUntil()` に渡したPromiseが拒否されると、インストールは失敗します。次に `activate` が発火し、以前の版が残したものを片付けます。その後で初めて、ワーカーは `fetch` イベントを受け取ります。ページは読み込まれたときにワーカーと対応付けられるので、最初のワーカーを登録したページは、ワーカーが `clients.claim()` を呼ばない限り、再読み込みするまでワーカーなしで動き続けます。

コピーは、同じ仕様が定義するCache Storageに置かれます。リクエストとレスポンスの組を保存する、名前付きのキャッシュで、ワーカー自身が書き込み、読み取り、削除を行います。ブラウザーのHTTPキャッシュとは別のものです。保存した項目は期限切れにならず、自動的には更新されず、ワーカーのスクリプトを更新しても残ります。ワーカーが削除するか、ブラウザーがサイトのストレージを消去するまで保持されます。

概念を判断に生かす

「サービスワーカーとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 読み込みが速く、スマートフォンでも使え、お客様がどのブラウザーを使っていても同じように動く画面を提供します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 フルスタックのGIS地図開発を指揮 — DBからブラウザーまでのフルスタック地図開発

フルスタックのGIS地図開発を指揮 お問い合わせ →

なぜ重要なのか

サービスワーカーを使うと、ネットワークがまったく使えない状況でもサイトが動作し続ける方法を、サイト側で明示的に制御できます。サービスワーカーは、サイトが用意したコードに従って、リクエストごとに、ネットワークから応答するか、自身が保存したコピーから応答するか、その両方を使うかを決めます。そのため、インストールしたウェブアプリを船上や地下室でも使えるようにできます。サイトのページが一つも開かれていないときでも、プッシュメッセージはサービスワーカーに届きます。

大半のワーカーで中心となるのは、リクエストの種類ごとに戦略を選ぶことです。ワーカーを記述するためのライブラリであるWorkboxは、五つの戦略に名前を付けています。Cache firstはコピーから応答し、コピーがない場合だけネット

ワークへアクセスします。同じ名前のまま内容が変わらないファイルに適しています。Network firstはまずネットワークを試し、失敗したらコピーを使うので、ページに適しています。Stale-while-revalidateはコピーから応答し、バックグラウンドでそのコピーを更新します。両極端に当たるのが、Network onlyとCache onlyです。

サービスワーカーは、利用者の端末上で、そのスコープ内のすべてのページとネットワークの間に立つコードでもあり、訪問が終わった後も残ります。誤りがあれば、修正したスクリプトで置き換わるまで残り続けます。仕様は、そのリスクを踏まえて設計されています。ナビゲーションのたびにブラウザーは新しいスクリプトを確認し、それ以外のイベントでも、前回の確認から1日が過ぎていれば確認します。既定では、メインスクリプトはHTTPキャッシュから取り出すのではなく、サーバーに問い合わせて再検証するため、ワーカーファイルの `max-age` が長くても、壊れた版をそのまま居座らせることはできません。インポートしたスクリプトにも、1日が過ぎれば同じ扱いを適用し、`updateViaCache` が `none` なら常に適用します。ブラウザーは、取得したスクリプトとインストール済みのものをバイト単位で比較し、違いがある場合だけ更新を開始します。Shiftキーを押しながら再読み込みすると、その1回のナビゲーションではワーカーを迂回します。そのため、開発者の端末だけでは不具合が直ったように見えることがあります。

待機のルールは、この設計のもう半分です。新しい版は古い版と並行してインストールされ、古い版に依存する開いたページが一つもなくなるまで、使われずに待機します。これにより、一つのページを二つの版が分担して処理することを防ぎます。`skipWaiting()` を呼ぶと直ちに引き継げますが、その代わり、古い版が読み込んだページで新しいコードが動くことになります。

最後の負担はストレージです。Cache Storageは、サイトが利用者の端末に保持するデータの使用量に含まれます。Storage Standardでは、そのストレージは既定でベストエフォート扱いです。サイトが永続ストレージを要求し、許可を得た場合を除き、端末の空き容量が不足すると、ブラウザーが消去することがあります。

実際の活用例

サービスワーカーを使う理由が最も明確になるのは、通信環境が悪い場所です。船員や現場スタッフ向けに構築したプログレッシブウェブアプリの利用者は、オフラインや接続不良に直面することがよくあります。サービスワーカーはnext-pwaプラグインで設定し、Workboxのランタイムキャッシュは、リクエストごとには変わらないフォント、画像、音声、動画、CDNのアセットにCache firstを適

用します。HTTPのキャッシュ制御ヘッダーも併用します。サービスワーカーの挙動を実際に確かめるにはHTTPSが必要なため、オフライン動作とインストール動作を実機でHTTPS経由で検証しました。繰り返しの読み込みは、ネットワークではなくキャッシュから返されます。

このサイトは逆の選択をし、そのことを明記しています。Cookieのページは、サービスワーカー、IndexedDB、Cache Storageを使っていないと説明しています。利用者がサイトを開く前や閉じた後に、このサイトの何かがバックグラウンドで動くことはありません。ページには実行可能なJavaScriptを含めていません。検索コードは保持していますが無効になっており、ワーカーを登録しません。

インストールのページでは、サイトをホーム画面に追加する利用者にとって何が変わるのかを説明しています。インストールすると専用のウィンドウで開けるようになりますが、管理されたオフライン用のコピーが作られるわけではありません。このサイトはサービスワーカーも、スクリプトで管理するオフラインストレージも配信しないため、インストールしたアプリも通常の利用には接続が必要です。

ワーカーの登録には実行可能なスクリプトを使いますが、このサイトのContent-Security-Policyは現在、すべてのページの `script-src 'none'` でそれを禁止しています。JSON-LDは実行されないデータのままです。以前のheadテンプレートのコメントには、793文書のうち550文書が `'none'` 、243文書が `'self'` を指定したビルドの記録があり、当時リストフィルターを持っていたページと一致します。これらの数字は、スクリプトが有効だった以前のビルドを示しており、現在のポリシーではありません。

問題が起きるところ

デプロイ後も古い版が応答し続ける。 新しいワーカーはインストール後、開いているタブのいずれかが古い版を使う間は待機します。そのため、タブを閉じない利用者は古いコードを使い続けます。古い版が読み込んだページでも新しい版が安全に動く場合は `skipWaiting()` を呼ぶか、更新通知を表示し、ページから待機中のワーカーへメッセージを送り、それを呼び出させてください。

ページが何週間も古い版のままになる。 HTMLにCache firstを適用すると、すべてのナビゲーションにCache Storageから応答します。その項目は期限切れにならず、サーバーのキャッシュヘッダーも参照しません。Cache firstは、内容が変わると名前も変わるファイルに使い、ページにはNetwork firstを使ってください。

インストールが失敗し、古いワーカーが残る。 インストール段階の一覧にある一つのアドレスがエラーを返すと、`cache.addAll()` が拒否され、`waitUntil()` に渡したPromiseも拒否されるため、新しい版は破棄されます。実在するファイルから一覧を作り、必須ではないファイルはインストール時ではなく後から取得してください。

ワーカーがサイトの大半に作用しない。 `/js/sw.js` から配信されるスクリプトは、既定のスコープも最大のスコープも `/js/` になるため、そのフォルダー外のページはワーカーなしで読み込まれます。スクリプトをサイトのルートから配信するか、`Service-Worker-Allowed: /` を付けて送信してください。

リリースのたびに端末のストレージ使用量が増える。 ワーカーのスクリプトを更新してもキャッシュは残るため、各版が新しい名前のキャッシュを開くと、前の版のものが残ります。すべてのキャッシュ名にバージョンを入れ、`activate` ハンドラーで、現在の版が使わないキャッシュを一つずつ削除してください。

参考資料

- Service Workers — 登録、スコープ、更新確認、ライフサイクル、Cache Storageを定めています。
- Secure Contexts — `localhost` とループバックアドレスを含め、潜在的に信頼できると見なすオリジンを定めています。
- Fetch Standard — `Request` と `Response` のオブジェクト、`fetch` ハンドラーがリクエストを受け取ってレスポンスを返す仕組み、サービスワーカーへリクエストを引き渡す方法を定めています。
- Storage Standard — Cache Storageを収めるストレージバケット、使用量の上限の見積もり、サイトが永続ストレージを要求する方法を定めています。
- Workbox：workbox-strategies — 五つのキャッシュ戦略と、それぞれに適したリクエストの種類を説明しています。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「サービスワーカーとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/service-worker/