第3章 SERVICE WORKERのインストールと有効化

ここではTodoアプリにService Workerの組み込みと、その有効化を行います。この章のコードはすでに `www/js/app.js` に記述済みです。ファイルを見て、コードとその内容を確認してください。

Service Workerをインストールする

まずService Workerのインストール法についてです。インストールといってもService Workerは単なるJavaScriptファイルであり、それを別なJavaScriptファイルから読み込むだけです。

現在、 `www` 以下に sw.js というファイルがあります。このファイルがService Workerです。次に `www/js/app.js` からService WorkerのJavaScriptファイルを読み込みます。

Service Workerに対応しているかチェック

まずレガシーなWebブラウザ対策としてService Workerに対応しているかチェックします。対応していない場合は何もしません。

```
// app.jsに記述済み
if ('serviceWorker' in navigator) {
  // Service Worker対応
}
```

`navigator.serviceWorker.register` を使ってService Workerをインストールします。Promiseで返ってきますので、 `then` で処理を繋ぎます。
`registration.onupdatefound` はService Workerのファイルがアップデートされていると実行されるイベントです。ファイルの更新はWebブラウザが自動的に行ってくれますが、オンラインでないといけませんので注意してください。

```
// app.jsに記述済み
navigator.serviceWorker
  .register('/sw.js')
  .then(function(registration) {
    // 登録成功
    registration.onupdatefound = function() {
```

```
      console.log('アップデートがあります！');
    }
  })
  .catch(function(err) {
    // 登録失敗 :(
    console.log('ServiceWorker registration failed: ', err);
});
```

Service Workerがインストールされたらキャッシュ処理を実行

次に **sw.js** を見てみます。Service Workerでの役割の一つがキャッシュです。キャッシュというとWebブラウザ標準で用意されていたキャッシュと混同してしまいますが、CACHE APIは開発者が自由にコントロールできるキャッシュで、本来のURLにアクセスする前に呼ばれるものです。そのためCACHEの内容を改変したり、オンラインとオフラインの状態で表示内容を変えるといったことも自由にできます。

キャッシュする際に必要なのはキャッシュ名と、キャッシュしたいURLのリストです。それらは **sw.js** の一番上に指定してあります。

```
// すでに記述済み
// キャッシュ名（変更可）
const CACHE_NAME = 'YOUR_CACHE_NAME';
// キャッシュするURL
const urlsToCache = [
  '/',
  // : 省略
];
self.skipWaiting();
```

次にService Workerがインストールされると install イベントが実行されます。
event.waitUntil はその中の処理（今回はキャッシュのためのリクエストする処理）が完了するのを保証してくれます。全体として非同期処理で行われるので各処理が確実に終わるのを待つ必要があります。

```
// すでに記述済み
// Service Workerがインストールされた時に呼ばれる処理
self.addEventListener('install', event => {
  // Service Workerがバージョンアップしている時に、新しいものを有効にする処
```

```
  event.waitUntil(self.skipWaiting());
  // キャッシュ登録処理を完了するのを保証する
  event.waitUntil(
    // キャッシュを開く
    caches.open(CACHE_NAME)
      // 指定したURLをキャッシュに登録する
      .then(cache => {
        urlsToCache.map(url => {
          // アクセスして結果を受け取る
          fetch(new Request(url))
            .then(response => {
              // 結果をキャッシュに登録する
              return cache.put(url, response);
            }, err =>  console.log(err));
        });
      })
  );
});
```

注意点

ここで分かるかと思いますが、Service Workerの中でfetchを使ってリクエスト処理を行っています。つまり通常のWebブラウザの中で行うHTTPリクエスト（画像やJavaScriptファイル、HTMLなど）を行う処理とService Workerとのリクエストは「別物」ということです。

そのため、Webブラウザからアクセスする時には特別な情報（ヘッダーなど）を付与しているとService Workerでは別なコンテンツがキャッシュされてしまう可能性があります。

ここまででService Workerのインストールが完了しました。第4章 Service Workerを使った表示高速化、オフライン対応について解説します。