

Service Worker Deep Dive ...

保呂 毅 (horo@chromium.org)

goo.gl/YxL2L7

自己紹介

保呂 毅 (Tsuyoshi Horo) @horo

Google ソフトウェアエンジニア 6 年目

Chrome チーム 3 年目

- Fetch API / Fetch Event の実装
- DevTools の Service Worker 対応
- Service Worker のパフォーマンス改善



今日の内容

Service Worker 周辺の最近(ここ1年くらい)の動向

- Push Notifications
- Stream
- Unified Media Pipeline
- Background Sync
- Foreign Fetch
- Header-based Installation
- Origin Trials
- AppCache
- DevTools

Service Worker とは

- バックグラウンドで動作する JavaScript 実行環境として登録

```
navigator.serviceWorker.register('./sw.js', {scope: './'});
```

- ページからのネットワークリクエストを横取り

```
self.addEventListener('fetch', function(event) {  
  event.respondWith(new Response('Hello World!'));  
});
```

- ページを開いていなくても Push 通知を受け取り Notification 表示

Is Service Worker Ready?







← → ↻ <https://jakearchibald.github.io/isserviceworkerready/index.html> ☆ ⋮

is SERVICeworker ready?

[Status](#) * [Spec](#) * [Intro](#) * [Resources](#) * [GitHub](#)

ServiceWorker enthusiasm

The first thing any implementation needs.



Chrome: Shipped.

Firefox: Shipped.

Samsung Internet: Shipped. Based on Chromium 44.2403 with some [additions and changes](#). (See "Service Worker" section.)

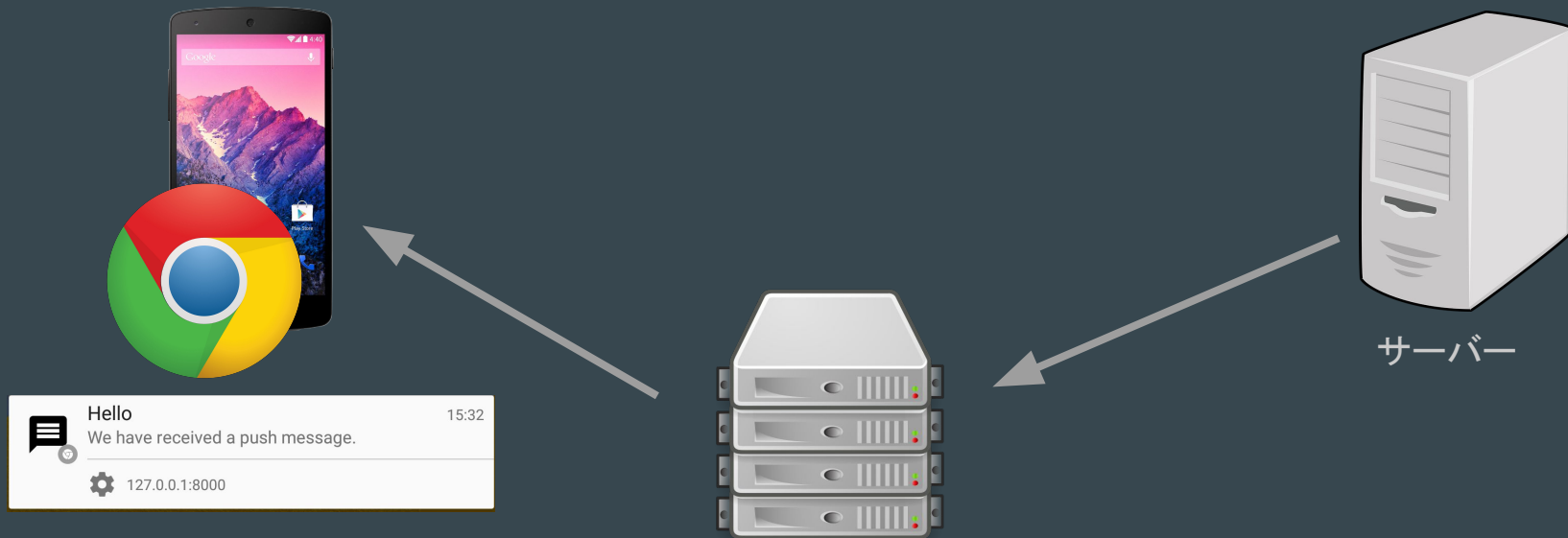
Safari: [Under consideration](#), Brief positive signals in [five year plan](#).

Edge: In development.

Support does not include iOS versions of third-party browsers on that platform (see Safari support).

Push Notifications

Push Notifications の仕組み



FCM (Firebase Cloud Messaging) サーバー
(旧 GCM (Google Cloud Messaging))

Push Event



Hello

15:32

We have received a push message.



127.0.0.1:8000

- サーバから FCM へ POST リクエストすることで、Service Worker の Push イベントハンドラが呼び出される

```
self.addEventListener('push', function(event) {  
  event.waitUntil(  
    self.registration.showNotification(  
      'Hello',  
      {  
        body: 'We have received a push message.',  
        icon: 'message.png',  
        tag: 'tag'  
      })  
    ));  
});
```

Notificationclick Event

- Notification がクリックされたらページを開く

```
self.addEventListener('notificationclick', function(event) {  
  event.notification.close();  
  clients.openWindow('/messages');  
});
```

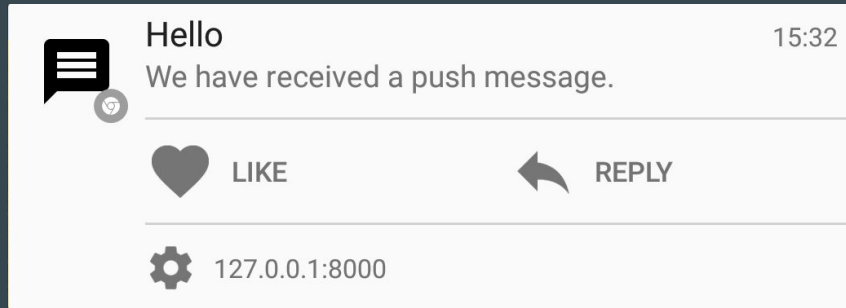
Payload Data

- Chrome 49 まで
 - Push メッセージにデータを含めることができない
 - Fetch API でサーバーに問い合わせる必要あり
- Chrome 50 から
 - Push メッセージに暗号化したデータを含めることができる
 - 詳しくは <https://developers.google.com/web/updates/2016/03/web-push-encryption>
 - Web Push library for Node.js <https://github.com/web-push-libs/web-push>

```
self.addEventListener('push', function(event) {  
  if (event.data) {  
    console.log(event.data.json());  
  }  
});
```

Notification Actions

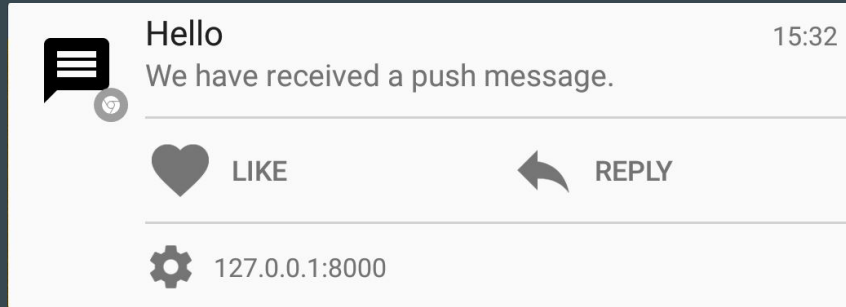
- Chrome 48 からボタンを追加可能



```
self.registration.showNotification(  
  'Hello',  
  {  
    body: 'We have received a push message.',  
    icon: 'message.png',  
    tag: 'tag',  
    data: 1234,  
    actions: [  
      {action: 'like', title: 'Like', icon: 'like.png'},  
      {action: 'reply', title: 'Reply', icon: 'reply.png'}]  
  });
```

Notificationclick Event

- イベントハンドラで動作を指定



```
self.addEventListener('notificationclick', function(event) {  
  var messageId = event.notification.data;  
  event.notification.close();  
  if (event.action == 'like') {  
    silentlyLikeItem();  
  } else if (event.action == 'reply') {  
    clients.openWindow('/messages?reply=' + messageId);  
  } else {  
    clients.openWindow('/messages?reply=' + messageId);  
  }  
});
```

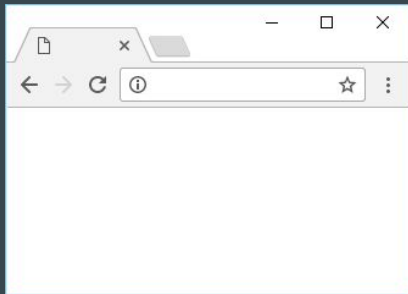
Standard Web Push Protocol

- Chrome 51 まで
 - 事前に FCM (旧 GCM) の登録、ID (gcm_sender_id) の取得が必要
 - manifest.json に gcm_sender_id を記述
 - FCM サーバーへの POST リクエストの Authorization ヘッダに API キー
- Chrome 52 から
 - Voluntary Application Server Identification (VAPID) でサーバー認証することで、FCM への事前登録は必要なし
- 詳しくは
<https://developers.google.com/web/updates/2016/07/web-push-interop-wins>

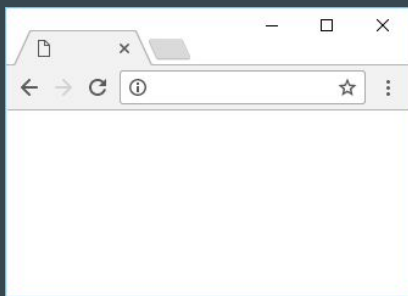
Stream

サーバーからのレスポンスに Service Worker が手を加えてページに返す

- Chrome 51 まで



- Stream を使うと (Chrome 52 から)



ページ

Service Worker

サーバー

Service Worker で処理をしてページに返す

- Fetch イベントハンドラでキャッシュからヘッダとフッタを取り出して、サーバーから取得したコンテンツと文字列連結する。

```
self.addEventListener('fetch', event => {
  var url = event.request.url;
  if (!url.endsWith('.html')) return;

  event.respondWith(
    Promise.all(
      [caches.match('./header.txt'), fetch(url + '.txt'),
       caches.match('./footer.txt')]
    ).then(responseList => Promise.all(responseList.map(res => res.text())))
    .then(textList =>
      new Response(textList.join(''),
        {headers:[['content-type', 'text/html']]})
    ));
});
```

問題点

サーバーからのコンテンツ全体を取得完了するまでページに返せない

解決方法

Stream を使えば、サーバーから取得したコンテンツを徐々にページに返していくことができる

Stream を作ってページに返す

```
self.addEventListener('fetch', event => {  
  var url = event.request.url;  
  if (!url.endsWith('.html')) return;  
  
  var stream = new ReadableStream({  
    【次ページ参照】  
  });  
  
  event.respondWith(new Response(  
    stream,  
    {headers:[['content-type', 'text/html']]});  
});
```

Stream を作ってページに返す

```
var stream = new ReadableStream({start(controller) {
  var promises = [caches.match('./header.txt'), fetch(url + '.txt'),
                  caches.match('./footer.txt')];
  function pushStream(body) {
    var reader = body.getReader();
    return reader.read().then(function proc(result) {
      if (result.done) return;
      controller.enqueue(result.value);
      return reader.read().then(proc);
    });
  }
  promises[0].then(response => pushStream(response.body))
    .then(() => promises[1]).then(response => pushStream(response.body))
    .then(() => promises[2]).then(response => pushStream(response.body))
    .then(() => controller.close());
}});
```

Unified Media Pipeline

Unified Media Pipeline (Android)

Chrome 51 までの <audio> <video> : Android のメディアスタックを利用

Chrome 52 から : デスクトップ版 Chrome と共通化

- Service Worker を使ってキャッシュ
- Blob URL からの再生
- playbackRate の指定
- MediaRecorder からの MediaStreams を Web Audio に送信
- クロスデバイスの開発が容易に

詳しくは : <https://developers.google.com/web/updates/2016/06/ump>

Background Sync

Background Sync を使うと何ができる？

- バックグラウンドでデータを送受信
- オフライン時にメッセージを書いて、オンラインになったときに自動で送信
- Android だと Chrome を閉じていても動作する

Chrome 49 から利用可能

Google



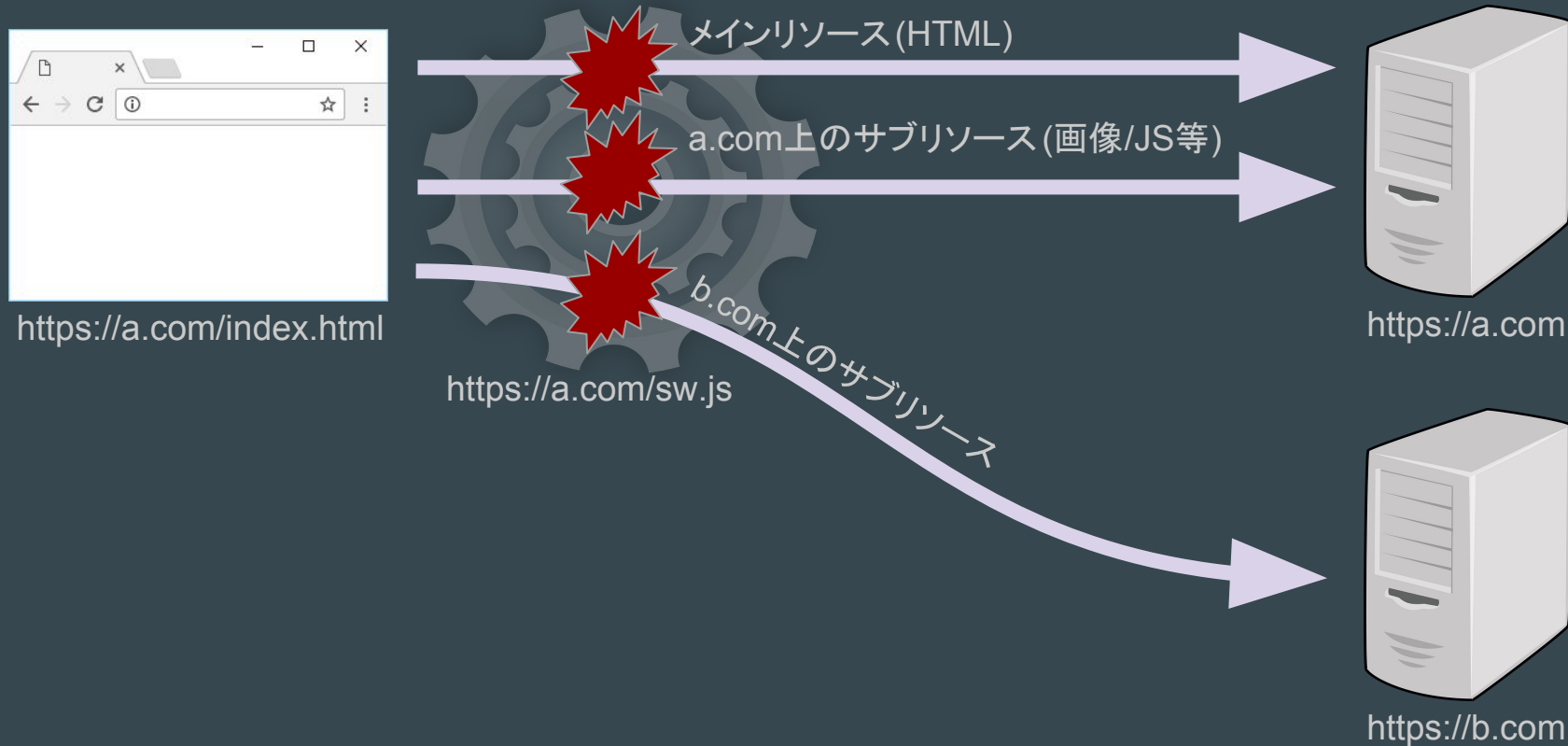
OfflineWiki

Wikipedia Demo

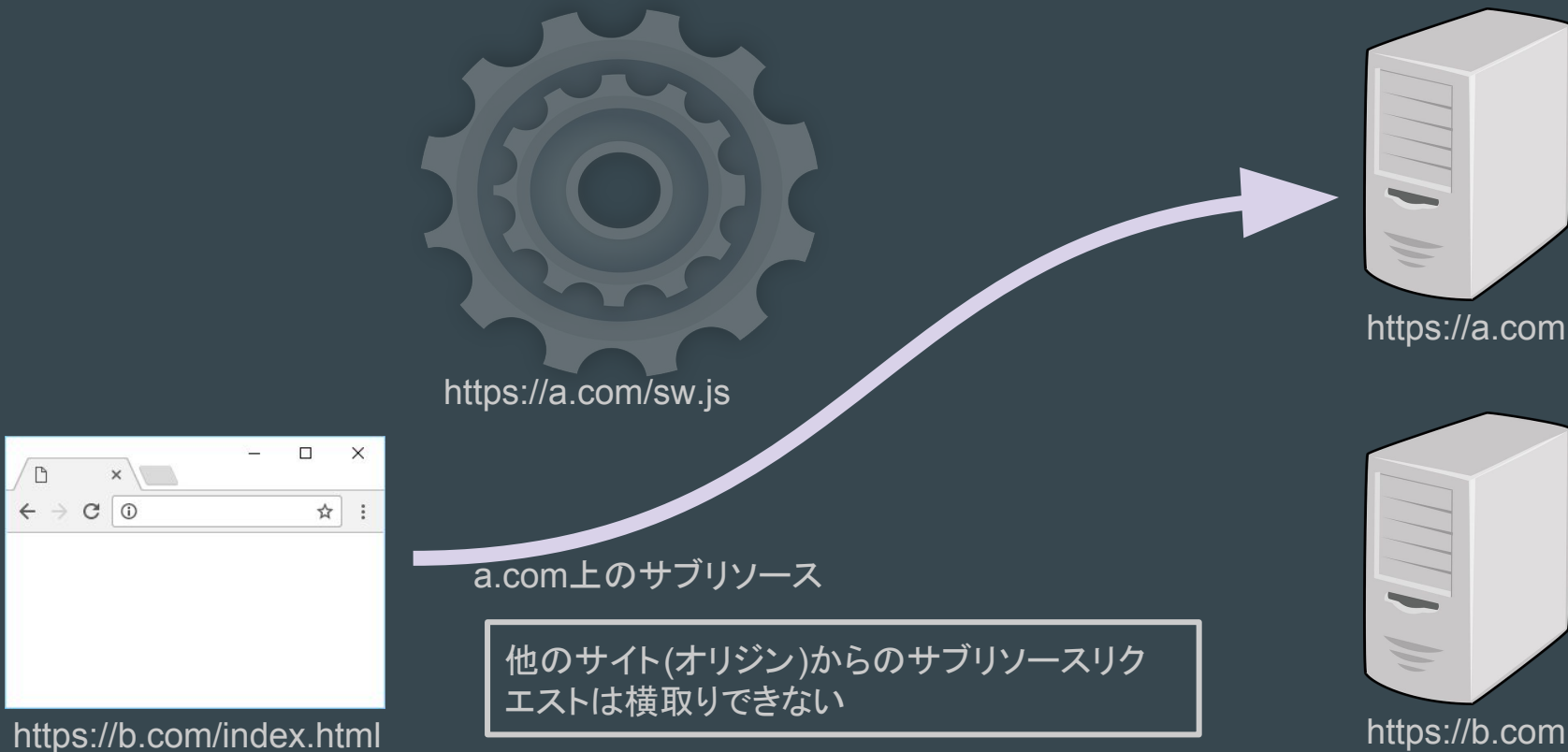


Foreign Fetch

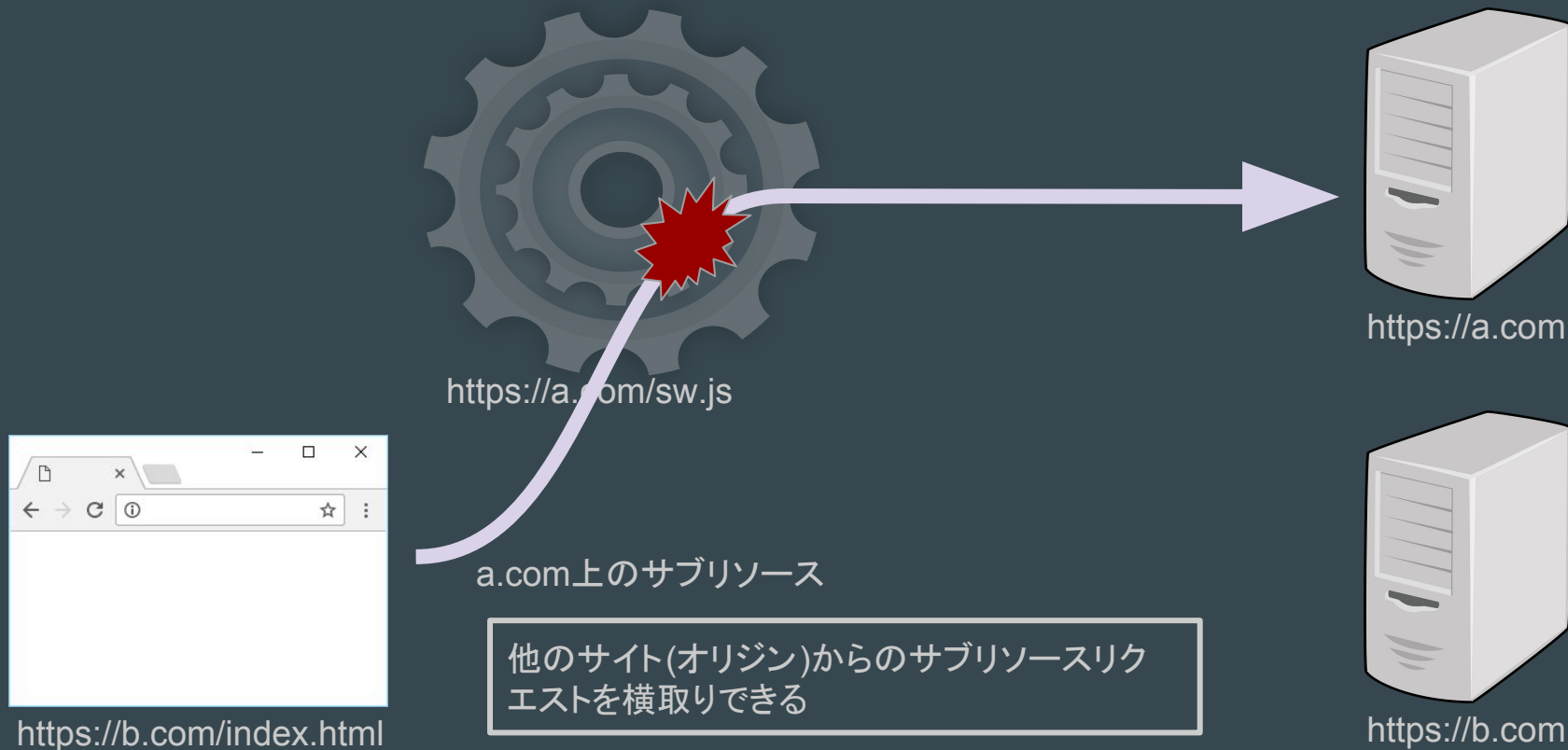
普通の Fetch イベント



普通の Fetch イベント



Foreign Fetch を使うと



普通の Fetch イベント vs Foreign Fetch イベント

共通点：ネットワークリクエストを横取りする

相違点：

普通の Fetch イベント

- 自分のサイトのページからのネットワークリクエストを横取り
- 自分のサイトの HTML ページと、そこからの画像などのサブリソース

Foreign Fetch イベント

- 他のサイトから自分のサイトへのサブリソースのリクエストを横取り
- Web Font サーバーや CDN で活用できる

InstallEvent.registerForeignFetch

Foreign Fetch の登録方法

https://b.com/ からの https://a.com/myscope/ 以下へのリクエストを横取り

```
self.addEventListener('install', function(event) {  
  event.registerForeignFetch({  
    scopes: ['/myscope/'],  
    origins: ['https://b.com/']  
  });  
});
```

- origins に['*']を指定すると、すべてのオリジンからのリクエストを横取りする

Foreign Fetch Event

- Response を `respondWith()` に渡す際は、辞書に包む必要がある

```
self.addEventListener('foreignfetch', function(event) {  
  event.respondWith(  
    fetch(event.request).then(res => ({response: res}));  
  });
```

注意:

- ページ側で Response を見ると Opaque になる
- `` で画像の表示はできるが、中身は読めない
- 読めるようにするには CORS の設定が必要

CORS (Cross-Origin Resource Sharing)

- origin を指定すると、ページで中身を読めるようになる
- headers を指定すると、指定したHTTP ヘッダも読めるようになる。

```
self.addEventListener('foreignfetch', function(event) {
  event.respondWith(
    fetch(event.request)
      .then(response =>({response: response,
                        origin: event.origin,
                        headers: ['...']})));
});
```

enable-experimental-web-platform-features

Foreign Fetch はまだ実験中の機能

試す場合は、Chrome 54 (Dev / Canary) 以降で

`chrome://flags/#enable-experimental-web-platform-features` を有効に

Header-based Installation

Service Worker の登録方法

今まで : JavaScript で登録

```
navigator.serviceWorker.register('/sw.js', {scope: '/'});
```

新しい方法 1 : HTTP の Link Header で登録

```
Link: </sw.js>; rel=serviceworker; scope=
```

新しい方法 2 : HTML の Link Element で登録

```
<link rel="serviceworker" scope="/" href="/sw.js">
```

Header-based installation & Foreign Fetch

- サブリソースのリクエストに対する HTTP レスポンスの Link Header で Service Worker をインストールできる
- Iframe 等で HTML を読み込ませる必要なく、Foreign Fetch の Service Worker をインストールできる

enable-experimental-web-platform-features

Header-based installationも実験中の機能

試す場合は、Chrome 54 (Dev / Canary) 以降 で

`chrome://flags/#enable-experimental-web-platform-features` を有効に

Origin Trials

Origin Trials

Chrome の実験的な新しい API を、申請のあった特定のオリジン(ドメイン)で一定期間だけ使ってもらってフィードバックをもらう仕組み。

例:

- Persistent Storage (Mid-October 2016まで)
- Web Bluetooth (Late January 2017まで)
- WebUSB (Match 2017まで)
- Foreign Fetch (Match 2017まで)
- Header-based Installation (予定)

詳しくは: <https://github.com/jpchase/OriginTrials/blob/gh-pages/developer-guide.md>

AppCache

AppCacheは廃止の方向に

Firefox: “Application Cache support will be removed”

<https://www.fxsitecompat.com/en-CA/docs/2016/application-cache-support-will-be-removed/>

Chrome: “Remove AppCache from Insecure Contexts”

<https://www.chromestatus.com/feature/5714236168732672>

オフライン対応は AppCache ではなく、Service Worker を使おう

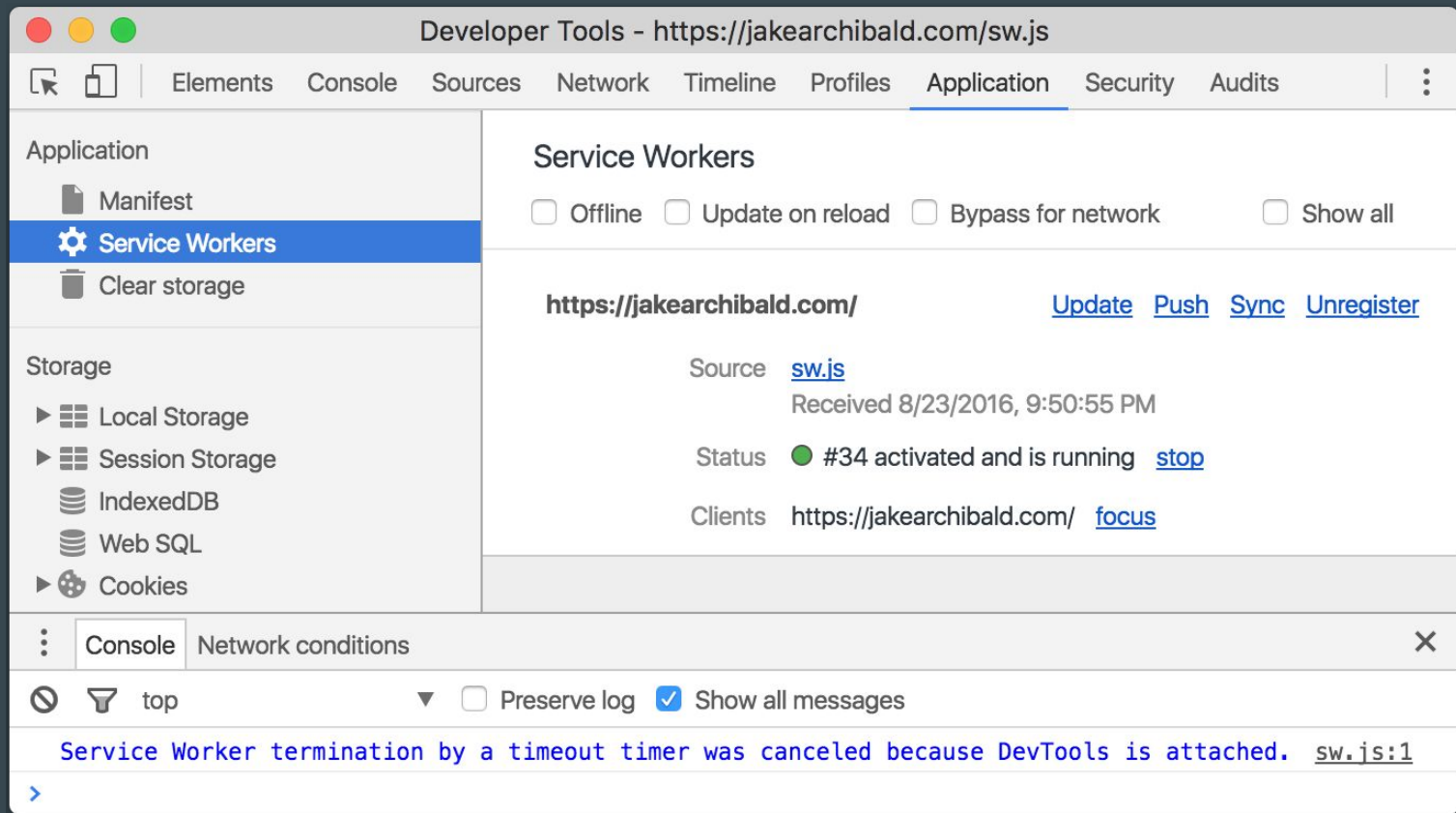
sw-appcache-behavior

AppCache の動作を Service Worker を使ってシミュレーションするライブラリ

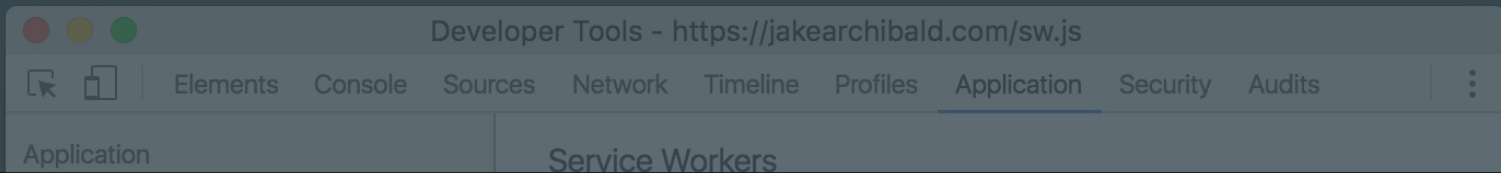
<https://github.com/GoogleChrome/sw-helpers/tree/master/projects/sw-appcache-behavior>

DevTools

DevTools



DevTools



- Service Worker は一定期間イベントが実行されないと停止し、次回イベントで再起動。
- ただし、DevTools を開いている間は停止しない。
- Global に変数を置いて保存していると、再起動時に **消える**。

Service Worker termination by a timeout timer was canceled because DevTools is attached. sw.js:1

今日の内容 まとめ

Service Worker 周辺の最近(ここ1年くらい)の動向

- Push Notifications
- Stream
- Unified Media Pipeline
- Background Sync
- Foreign Fetch
- Header-based Installation
- Origin Trials
- AppCache
- DevTools

関連リンク集

Introduction to Service Worker

- <https://developers.google.com/web/fundamentals/primers/service-worker/>

Push

- <https://developers.google.com/web/updates/2015/03/push-notifications-on-the-open-web>
- <https://developers.google.com/web/updates/2016/01/notification-actions>
- <https://developers.google.com/web/updates/2016/03/web-push-encryption>
- <https://developers.google.com/web/updates/2016/03/notifications>
- <https://developers.google.com/web/updates/2016/07/web-push-interop-wins>
- <https://github.com/web-push-libs/web-push/>
- <https://developers.google.com/cloud-messaging/>

Stream

- <https://developers.google.com/web/updates/2016/06/sw-readablestreams>
- <https://www.youtube.com/watch?v=Pii-LaWOyuo>
- <https://jakearchibald.com/2016/streams-ftw/#creating-your-own-readable-stream>

Unified Media Pipeline

- <https://developers.google.com/web/updates/2016/06/ump>

Background Sync

- <https://developers.google.com/web/updates/2015/12/background-sync>
- <https://github.com/WICG/BackgroundSync/blob/master/explainer.md>
- <https://ponyfoo.com/articles/backgroundsync>

Foreign Fetch

- https://github.com/slightlyoff/ServiceWorker/blob/master/foreign_fetch_explainer.md
- https://slightlyoff.github.io/ServiceWorker/spec/service_worker/#on-foreign-fetch-request-algorithm
- <https://crbug.com/540509>
- <https://github.com/slightlyoff/ServiceWorker/pull/751>

Header-based installation

- <https://crbug.com/582310>
- <https://github.com/slightlyoff/ServiceWorker/issues/685#issuecomment-176473764>

Origin Trials

- <https://github.com/jpchase/OriginTrials/blob/gh-pages/developer-guide.md>
- https://docs.google.com/a/google.com/forms/d/e/1FAIpQLSfO0_ptFl8r8G0UFhT0_xhV17eabG-erUWBDiKSRDTqEZ_9ULQ/viewform

Web Bluetooth

- <https://developers.google.com/web/updates/2015/07/interact-with-ble-devices-on-the-web?hl=en>

WebUSB

- <https://developers.google.com/web/updates/2016/03/access-usb-devices-on-the-web>