



Code
d'Armor

Les Service Workers



ENSSAT
LANNION

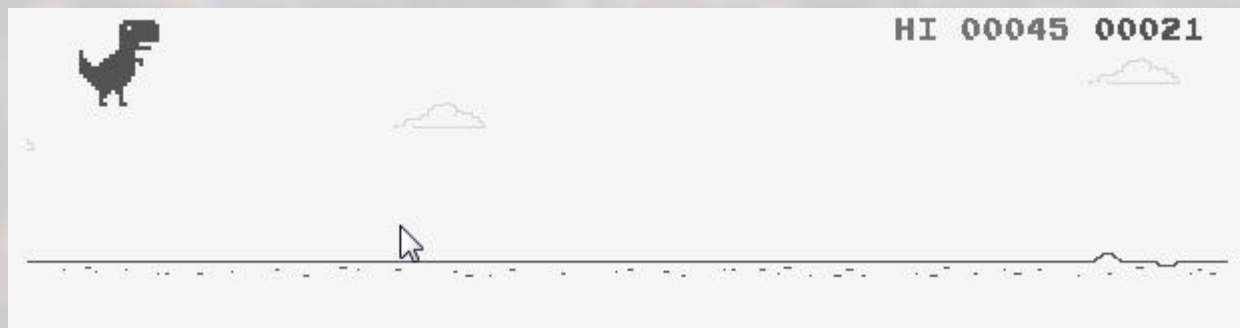
Mathieu Le Bihan
@slek22

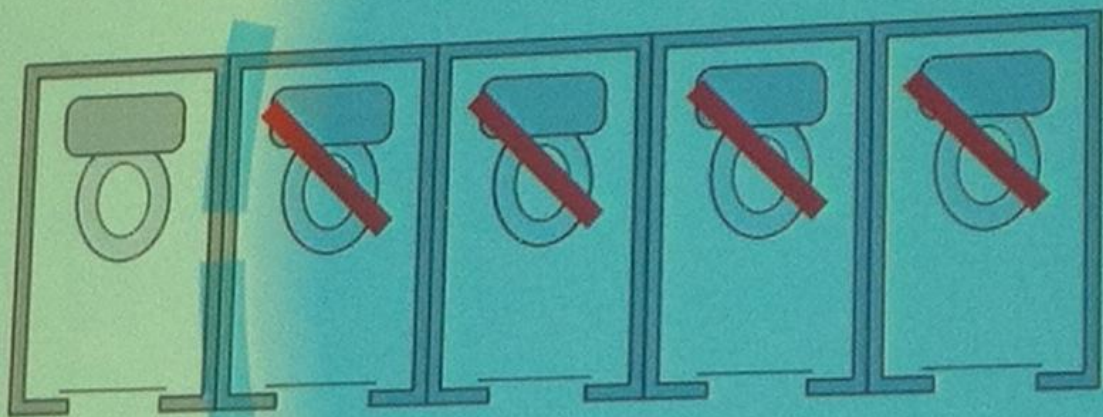
I  **the**
Web

4G







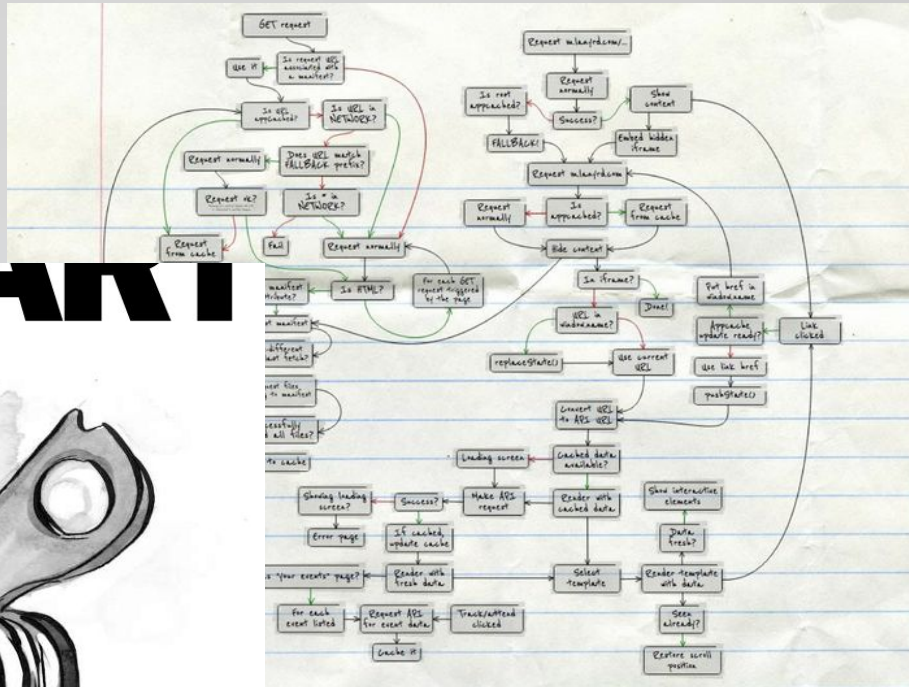


Jake Archibald - @jaffathecake - **Lanyrd.com**

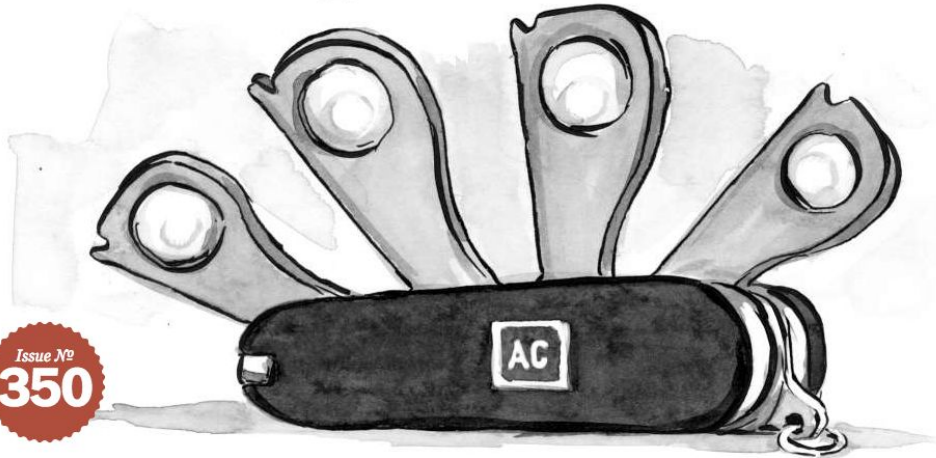
Application Cache

```
cache.manifest
1  CACHE MANIFEST
2  # v1 2016-04-16
3
4  index.html
5  cache.html
6  style.css
7  image1.png
8
9  FALLBACK:
10 / fallback.html
11
12 NETWORK:
13 network.html
```





A LIST APART



Application Cache is a Douchebag

by JAKE ARCHIBALD · May 08, 2012

Published in Application Development, HTML, JavaScript · 47 Comments



Et quand la webapp est fermée ?



Des coquilles native ?



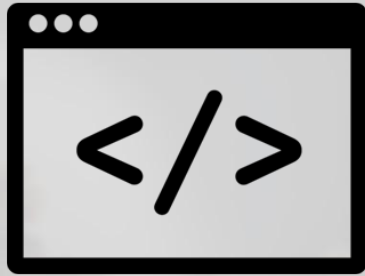
Cobalt



Cordova

Service Worker

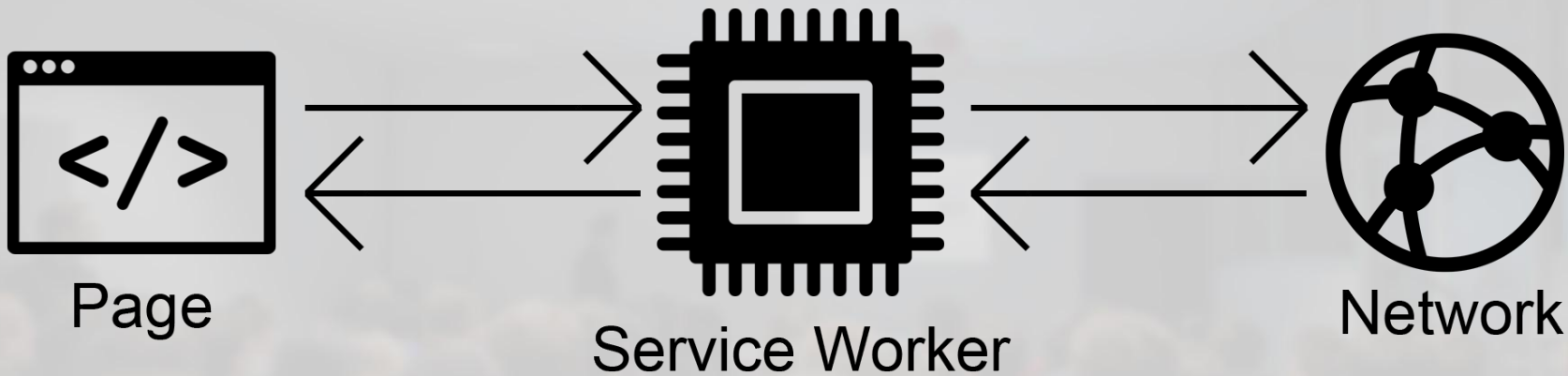


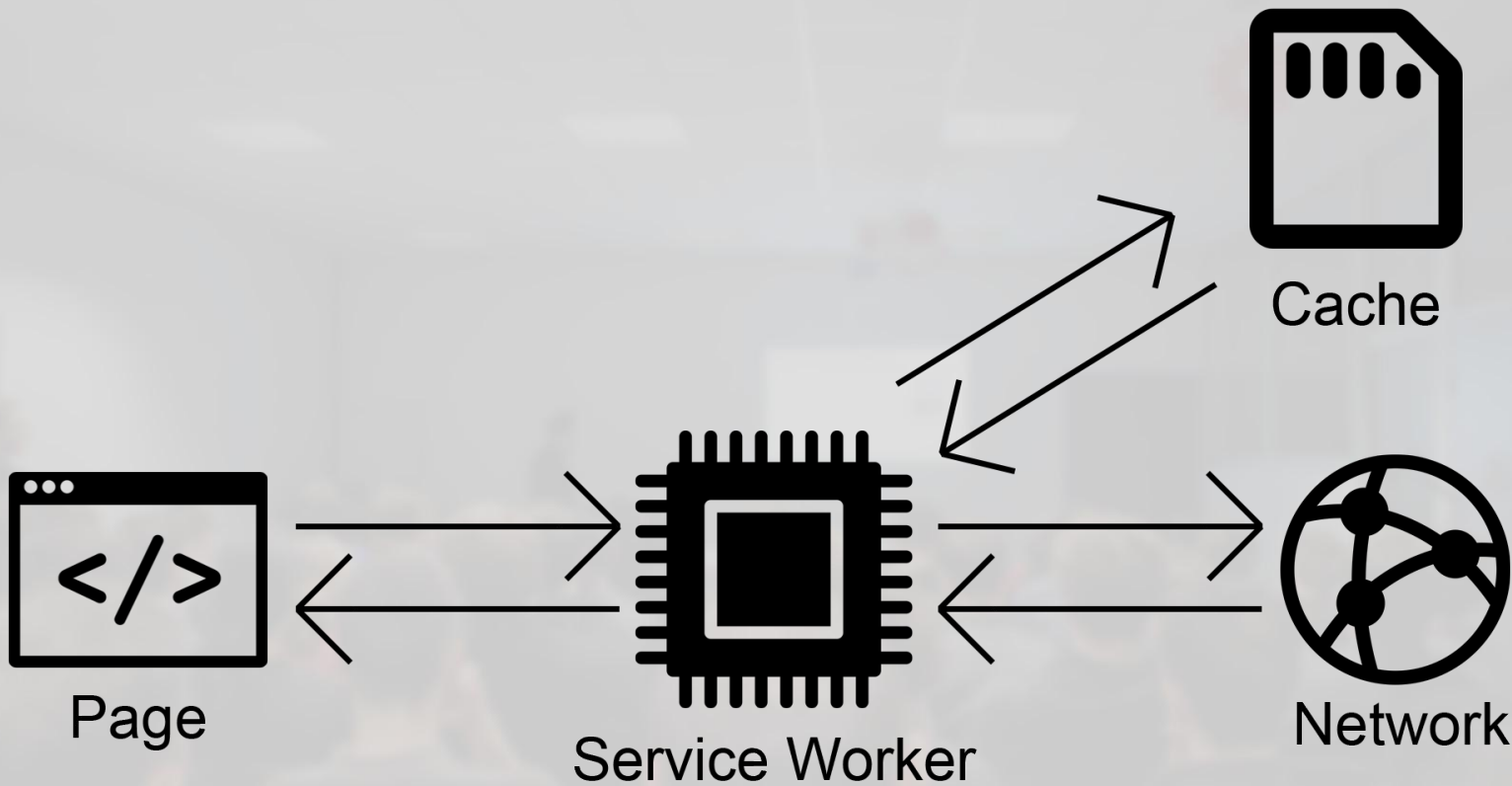


Page



Network







https://

Standardisation



Service Workers

W3C Working Draft 25 June 2015

This version

<http://www.w3.org/TR/2015/WD-service-workers-20150625/>

Latest published version

<http://www.w3.org/TR/service-workers/>

Latest editor's draft

https://slightlyoff.github.io/ServiceWorker/spec/service_worker/

Previous version

<http://www.w3.org/TR/2015/WD-service-workers-20150205/>

Revision history

<https://github.com/slightlyoff/ServiceWorker/commits/master>

Participate

Discuss on public-webapps@w3.org (Web Applications Working Group)

[File bugs](#)

Editors

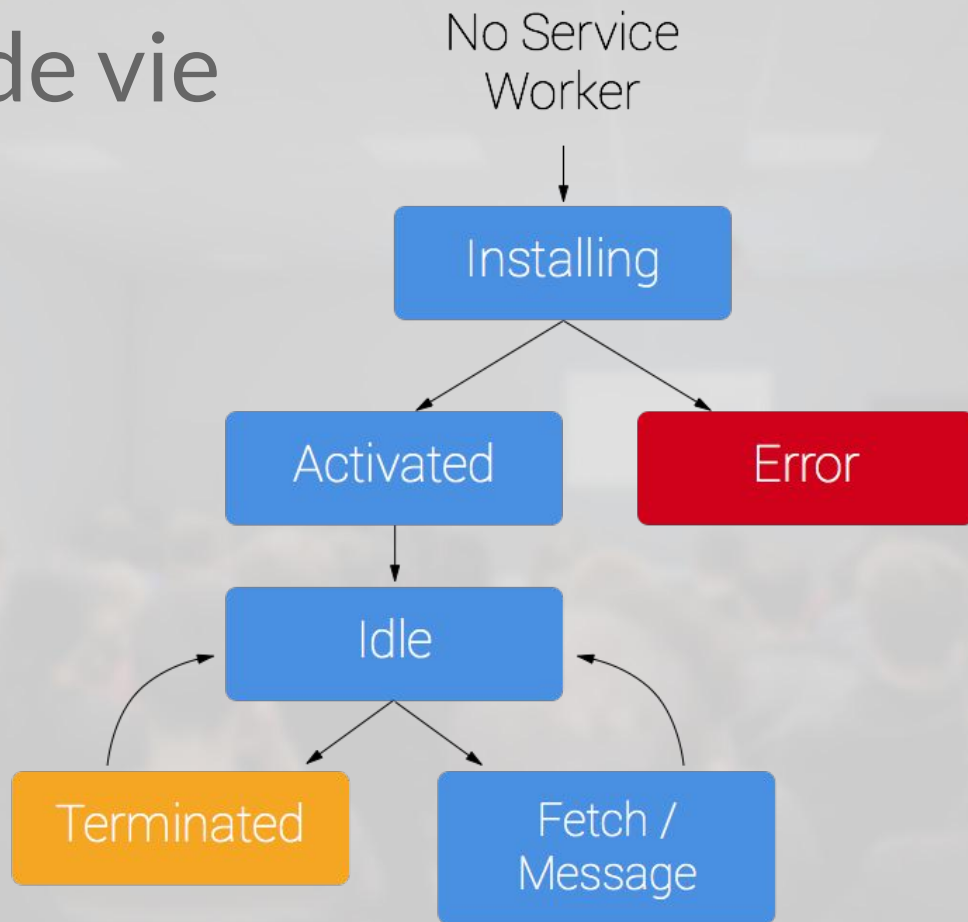
Alex Russell, Google, <slightlyoff@chromium.org>

Jungkee Song, Samsung Electronics, <jungkee.song@samsung.com>

Jake Archibald, Google, <jakearchibald@chromium.org>

Copyright © 2015 W3C® (MIT, ERCIM, Keio, Beihang). W3C liability, trademark and document use rules apply.

Cycle de vie



Squelette d'un Service Worker

```
self.addEventListener('install', function(event) {  
  // Ajouter des fichiers dans le cache  
});  
  
self.addEventListener('activate', function(event) {  
  // Mettre à jour le cache  
});  
  
self.addEventListener('fetch', function(event) {  
  // Intercepter les requêtes  
});
```

Enregistrement

```
// main.js

navigator.serviceWorker.register('/worker.js', {
  scope: '/'
}).then(function(registration) {
  console.log("Registration succeeded!");
}, function(error) {
  console.log("Registration failed with "+error);
});
```

Installation

```
// sw.js

self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open('v1').then(function(cache) {
      return cache.addAll([
        '/js/app.js',
        '/css/style.css',
        '/img/foo.png'
        // etc
      ]);
    })
  );
});
```

Installation non bloquante

```
// sw.js

self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open('v1').then(function(cache) {
      cache.addAll(
        // niveaux 11 à 99
      );
      return cache.addAll(
        // niveaux 1 à 10
      );
    })
  );
});
```

Fetch

```
fetch(url).then(function(response) {  
    return response.json();  
}).then(function(data) {  
    console.log(data);  
}).catch(function() {  
    console.log("Booo");  
});
```


Hello World

```
// sw.js

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

Répondre avec le réseau

```
// sw.js

self.addEventListener('fetch', function(event) {
  event.respondWith(
    fetch(event.request)
  );
});
```

Répondre avec le cache

```
// sw.js

self.addEventListener('fetch', function(event) {
  event.respondWith(
    caches.match(event.request).then(function(response) {
      return response || fetch(event.request);
    })
  );
});
```

Fallback

```
// sw.js

self.addEventListener('fetch', function(event) {
  event.respondWith(
    caches.match(event.request).then(function(response) {
      return response || fetch(event.request);
    }).catch(function() {
      return caches.match('/offline.html');
    })
  );
});
```

Aller plus loin

Jake Archibald wrote...

The offline cookbook

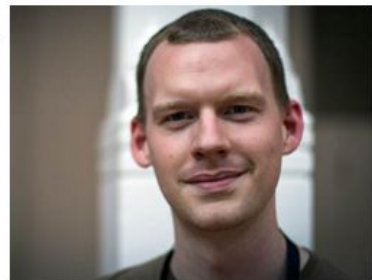
Posted 09 December 2014

Update: Together with Udacity I created a [free offline-first interactive course](#). It involves taking an online-only site to full offline-first glory. Many of the patterns in this article are used.

When AppCache arrived on the scene it gave us a couple of patterns to make content work offline. If those were the patterns you needed, congratulations, you won the AppCache lottery (the jackpot remains unclaimed), but the rest of us were left huddled in a corner [rocking back & forth](#).

With ServiceWorker ([intro](#)) we gave up trying to solve offline, and gave developers the moving parts to go solve it themselves. It gives you control over caching and how requests are handled. That means you get to create your own patterns. Let's take a look at a few possible patterns in isolation, but in practice you'll likely use many of them in tandem depending on URL & context.

All code examples work today in Chrome & Firefox, unless otherwise noted. For full details on




Hello, I'm Jake and that is my face. I'm a developer advocate for Google Chrome.

Elsewhere

 [Twitter](#)

 [Lanyrd](#)

 [Github](#)

 [Google+](#)

 [Flickr](#)

Contact

Feel free to [throw me an email](#), unless

Mise à jour

```
// sw.js

self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open('v2').then(function(cache) {
      return cache.addAll([
        '/js/app.js',
        '/css/style.css',
        '/img/foo.png'
      ]);
    })
  );
});
```

Mise à jour

```
// sw.js

self.addEventListener('activate', function(event) {

  var cacheWhitelist = ['v2'];

  event.waitUntil(
    caches.keys().then(function(keyList) {
      return Promise.all(keyList.map(function(key) {
        if (cacheWhitelist.indexOf(key) === -1) {
          return caches.delete(keyList[i]);
        }
      }));
    }));
  });
});
```

Message worker -> clients

```
// sw.js
```

```
self.clients.matchAll().then(function(clients) {  
  clients.map(function(client) {  
    return client.postMessage('Un message du worker =)');  
  })  
});
```

```
// main.js
```

```
navigator.serviceWorker.addEventListener('message', function(event) {  
  console.log(event.data);  
});
```


Message client -> worker

```
// main.js
```

```
function sendMessage(message) {  
  return new Promise(function(resolve, reject) {  
    const messageChannel = new MessageChannel();  
  
    messageChannel.port1.onmessage = function(event) {  
      resolve(event.data);  
    };  
  
    navigator.serviceWorker.controller.postMessage(message, [messageChannel.port2])  
  });  
}
```

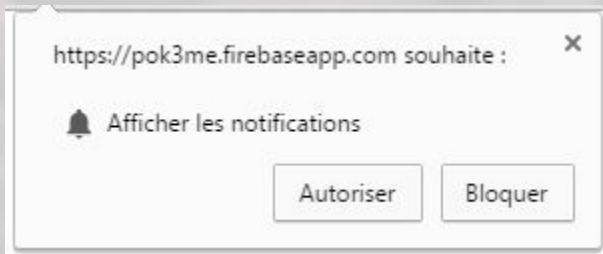
```
// sw.js
```

```
self.addEventListener('message', function(event) {  
  event.ports[0].postMessage('Message reçu : '+event.data);  
});
```

Push

```
// main.js

navigator.serviceWorker.ready.then(function(registration) {
  registration.pushManager.subscribe(
    {userVisibleOnly: true}
  ).then(function(subscription) {
    console.log(subscription.endpoint);
    // save endpoint somewhere
  });
});
```



Push

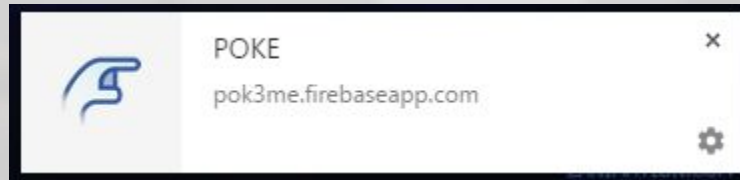
```
// sw.js

self.addEventListener('push', function(event) {
  var title = 'POKE';
  var icon = 'assets/poke.png';
  var tag = 'push';

  event.waitUntil(self.registration.showNotification(title, {
    icon: icon,
    tag: tag
  }));
});
```

Push

Ne reste plus qu'à faire un POST sur l'endpoint préalablement sauvegardé :



Un peu plus spécifique pour GCM

```
manifest.json
1 {
2   "name": "Poke Me - Service Workers Demonstration",
3   "short_name": "Poke Me",
4   "start_url": "/index.html",
5   "display": "standalone",
6   "gcm_sender_id": "<Projet ID>"
7 }
8
9
```

```
fetch(endpoint, {
  method: 'post',
  headers: new Headers({
    'Content-Type': 'application/json',
    'Authorization': 'key='+GOOGLE_API_KEY
  }),
  body: JSON.stringify({
    registration_ids: [ registrationId ]
  })
})
```

Endpoint

RegisrationID

[https://android.googleapis.com/gcm/send/f-00cl-8RE4:APA91bER2 ... 23k-I0OAS0Ben04GZz1](https://android.googleapis.com/gcm/send/f-00cl-8RE4:APA91bER2...23k-I0OAS0Ben04GZz1)



<https://pok3me.firebaseio.com/>

Debugging



`chrome://inspect/#service-workers`



`about:serviceworkers`

Background Synchronisation

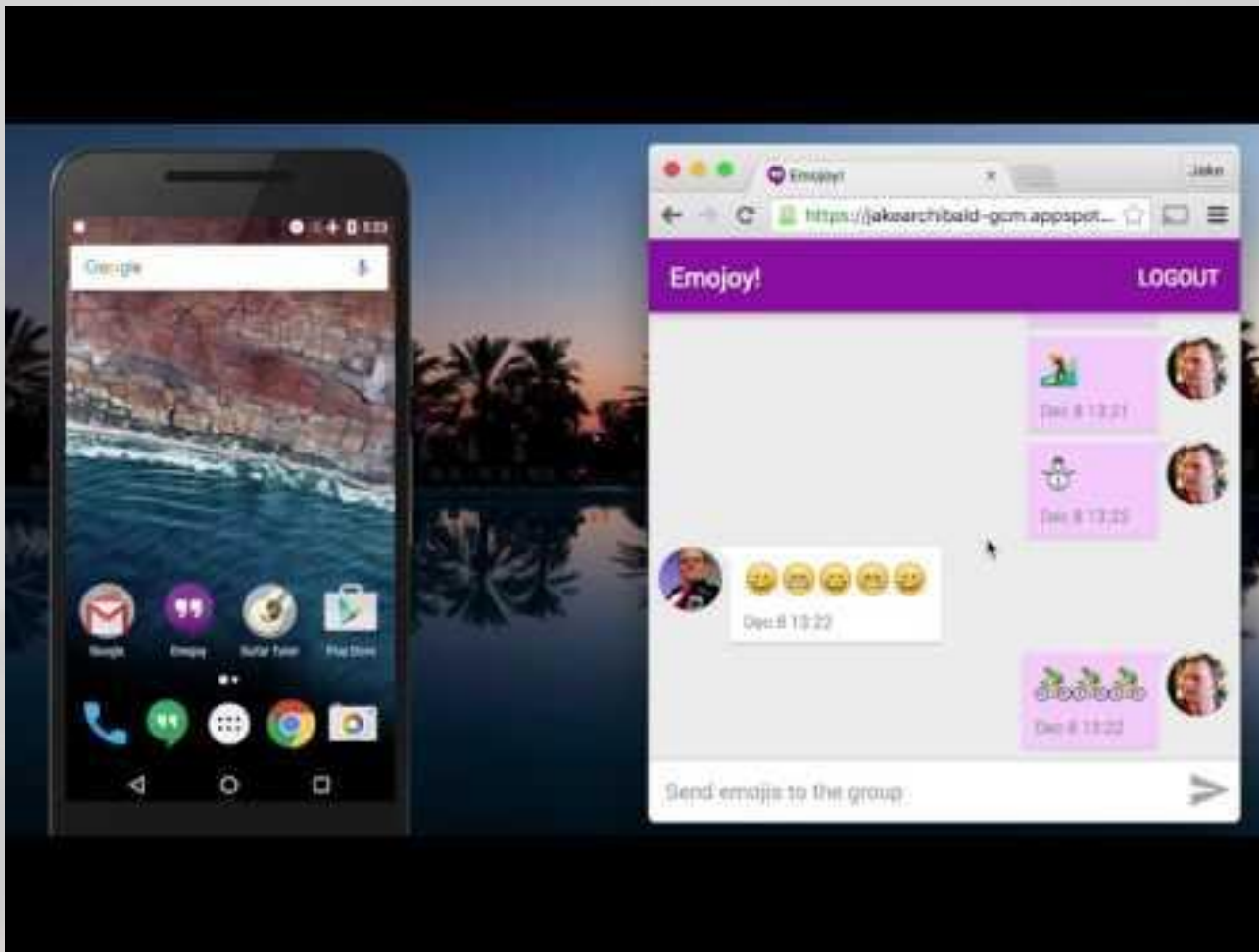
```
// main.js

navigator.serviceWorker.ready.then(function(registration) {
  return registration.sync.register('myFirstSync');
});

// sw.js

self.addEventListener('sync', function(event) {
  if (event.tag == 'myFirstSync') {
    event.waitUntil(doSomeStuff());
  }
});
```

→ <chrome://flags/#enable-experimental-web-platform-features>



Ce n'est qu'un début !



- Periodic Synchronisation
- Geofencing
- ...



Can I use it ?

is SERVICEWORKER ready?

Status * Spec * Intro * Resources * GitHub

<h3>ServiceWorker enthusiasm</h3> <p>The first thing any implementation needs.</p>	
	<p>Chrome: Shipped.</p> <p>Firefox: Shipped.</p> <p>Safari: Under consideration, Brief positive signals in five year plan.</p> <p>Edge: High priority, work has begun.</p> <p>Support does not include iOS versions of third-party browsers on that platform (see Safari support).</p>
<h3>Promises</h3> <p>Not ServiceWorker-specific, but required by ServiceWorker. Spec.</p>	
	<p>Edge: Devium build</p>

Liens utiles

- <https://serviceworkers.rs>
- <https://jakearchibald.com>
- <https://jakearchibald.github.io/isserviceworkerready/>
- <https://miguelmota.com/blog/getting-started-with-service-workers/>
- <http://alistapart.com/article/application-cache-is-a-douchebag>

Crédits

Icônes :

- <https://thenounproject.com/term/sd-card/6185/>
- <https://thenounproject.com/term/network/12676/>
- <https://thenounproject.com/term/cpu/72043/>
- <https://thenounproject.com/term/code/17547/>



Code
d'Armor

Merci