



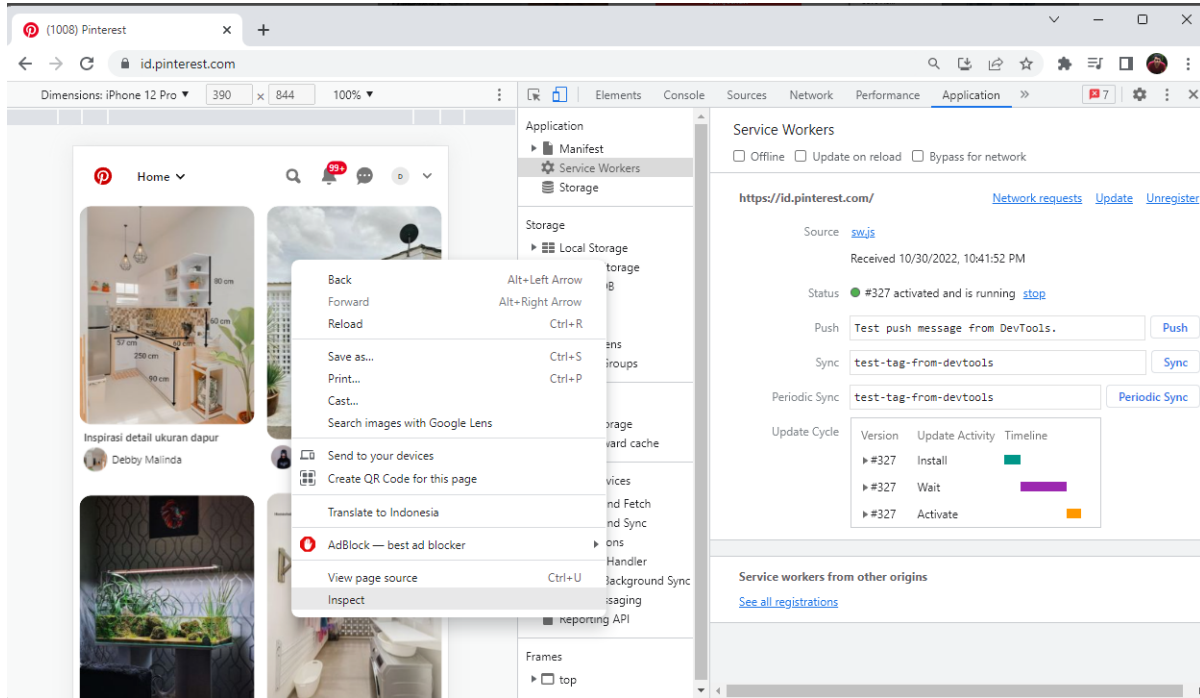
MODUL 7

PENGEMBANGAN APLIKASI PLATFORM KHUSUS
DAVID SETIADI

MODUL 7


SERVICE WORKER

Service workers adalah sebuah web worker berupa file JavaScript yang berjalan secara terpisah dari main thread browser. Service worker akan men-intercept network request, caching atau retrieve dari cache, dan mengirimkan push message.

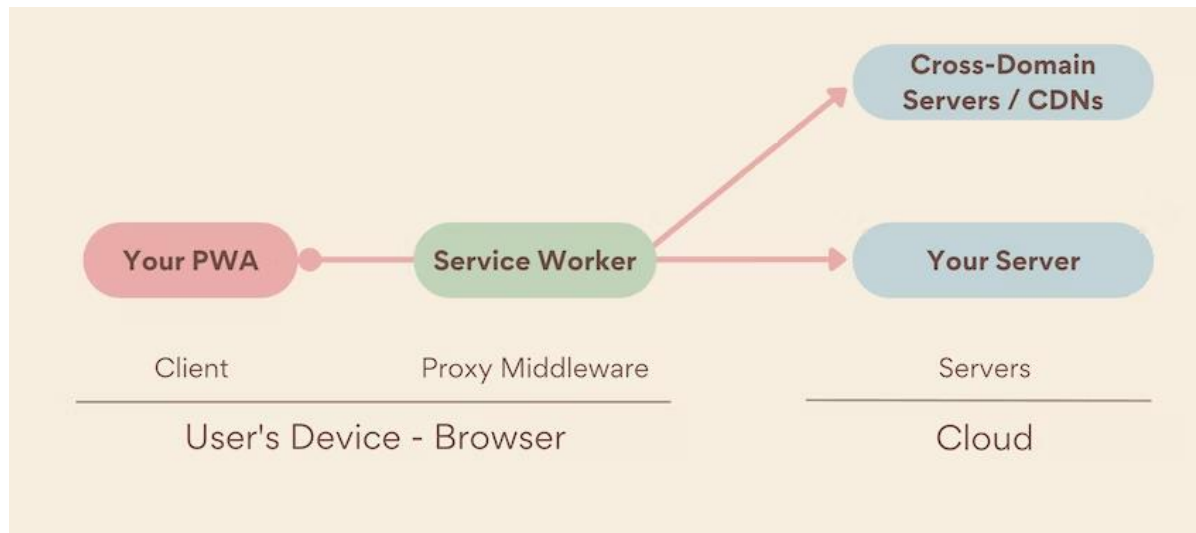


Pengguna berharap aplikasi dapat tetap berjalan dalam koneksi internet yang lambat atau tidak ada samasekali atau offline. Ketika user tidak memungkinkan melakukan request maka aplikasi harus bisa handle dengan baik tanpa crash. Dan user menginginkan aplikasi yang dapat melakukannya dengan cepat seperti yang bisa dilihat dari study **Milliseconds make millions**, yang intinya peningkatan kecepatan load halaman dalam 0.1 detik bisa meningkatkan konversi hingga 10%. Dan singkatnya semua keinginan user diatas dapat dilakukan oleh PWA yaitu dengan adanya service worker.

A 0.1 second improvement of mobile site speed
increases conversion rates by:


8.4%
for retail sites


10.1%
for travel sites



Saat aplikasi meminta sumber daya yang tercakup dalam cakupan pekerja layanan, termasuk saat pengguna offline, pekerja layanan memotong permintaan tersebut, bertindak sebagai proxy jaringan. Kemudian dapat memutuskan apakah harus melayani sumber daya dari cache melalui Cache Storage API, dari jaringan seperti biasanya tanpa service worker, atau membuatnya dari algoritme lokal. Ini memungkinkan Anda memberikan pengalaman serupa dengan yang disediakan oleh aplikasi platform. Ia bahkan dapat bekerja sepenuhnya secara off line.



No internet











































Try:

- Checking the network cables, modem, and router
- Reconnecting to Wi-Fi
- Running Windows Network Diagnostics

ERR_INTERNET_DISCONNECTED

Show saved copy

Hanya saja yang menjadi permasalahan adalah tidak semua browser mendukung service worker ini, untuk mengeceknya kita bisa mengunjungi link ini <https://jakearchibald.github.io/isserviceworkerready/>

Service worker 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: Shipped. Edge: Shipped. Support does not include iOS versions of third-party browsers on that platform (see Safari support).
Promises Not service worker-specific, but required by service worker. <i>Spec.</i>	     
Debugging State of debugging tools.	     
	Chrome: You can debug service worker scripts as any other. "Application" panel in devtools has service worker & cache sections. Firefox: Debuggable from the "Workers" page in about:debugging. Firefox: Web Console can display console messages from service workers. Firefox: about:serviceworkers has some under-the-hood stuff. Opera: Debuggable from the resources panel in Opera developer if you enable super-experimental devtools. Opera: Console messages from the service worker appear in the pages' console in Opera stable. Safari: See 'service workers' in the 'develop' menu to open an inspector for a particular service worker. Edge: See the service worker section in the sources panel. Chrome & Opera: Debuggable from the resources panel in Chrome Canary and Opera developer if you enable super-experimental devtools. Chrome & Opera: Console messages from the service worker appear in the pages' console. Chrome & Opera & Samsung Internet: chrome://serviceworker-internals resp. browser://serviceworker-internals (in Opera developer) has some under-the-hood stuff.
navigator.serviceWorker Namespace for page-side service worker API. <i>Spec. Test.</i>	     
Register / unregister Register for a SW and get a registration instance back, unregister undoes. <i>Spec. Test.</i>	     
postMessage to & from worker <i>Spec. Test.</i>	     
Fetch event Fires for pages and all sub-resources. <i>Spec. Test.</i>	     

Beberapa hal yang perlu diperhatikan dalam service worker adalah

1. Service worker merupakan Javascript Worker sehingga tidak dapat mengakses DOM HTML secara langsung. Service worker akan berkomunikasi dengan halaman web melalui interface postMessage. Informasi lebih lanjut mengenai Javascript Worker dapat dibaca di <https://www.html5rocks.com/en/tutorials/workers/basics/>.

2. Service worker merupakan network proxy yang dapat diprogram. Dengan cara ini maka kita dapat mengendalikan bagaimana menangani permintaan sumber daya jaringan dari suatu halaman web
3. Service worker langsung mati jika tidak digunakan dan direstart saat diperlukan. Dengan cara ini maka penggunaan memory akan lebih hemat.
4. Service menggunakan Promises secara intensif. Informasi lebih lanjut mengenai Promises silahkan mengunjungi <https://developers.google.com/web/fundamentals/getting-started/primers/promises>.

Berikut beberapa ketentuan dari service workers.

- Service worker didesain full asynchronous, synchronous XHR and localStorage tidak dapat digunakan dalam service worker.
- Service worker dapat menerima push messages dari server ketika app tidak aktif. Memungkinkan user menerima notifikasi, walaupun browser dalam keadaan tertutup.
- Service worker tidak dapat mengakses DOM secara langsung. Untuk berkomunikasi dengan sebuah halaman, service worker menggunakan metoda postMessage() untuk mengirim data dan “message” event listener untuk menerima data.
- Service worker hanya berjalan di https, untuk mencegah serangan tipe “man-in-the-middle”.
- Ketika tidak digunakan, service worker akan idle dan di restart ketika berikutnya dibutuhkan. Jika Anda membutuhkan menyimpan data persisten setelah restart, gunakan IndexedDB.
- Service worker menggunakan promise, Anda diharapkan sudah memahami promise JavaScript.

Fungsi Service Workers

Dengan service workers, kita bisa mengatur network requests, cache requests untuk meningkatkan performance dan menyediakan offline access dengan menggunakan cached content.

Service workers menggunakan 2 APIs agar app bisa bekerja offline:

- **Fetch**, cara standar untuk mengambil konten dari network.
- **Cache**, menyimpan dalam storage, cache bersifat persistent dan independent dari browser cache atau network status.

Service workers juga memungkinkan kita untuk membuat web app menjadi feel native.

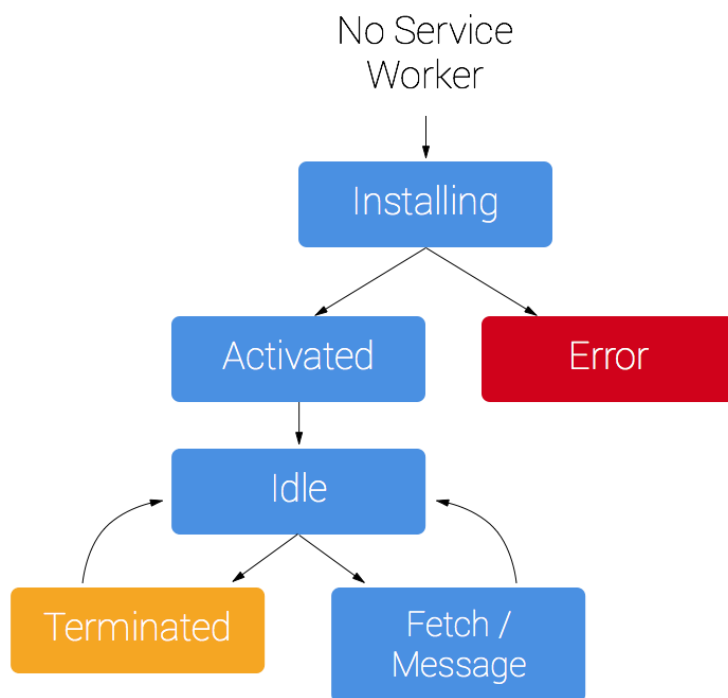
- **Notifications API**: berguna untuk menampilkan dan berinteraksi notifikasi dengan menggunakan sistem notifikasi dari operating system.
- **Push API**: API yang memungkinkan aplikasi Anda berlangganan layanan push dan menerima push message. Message dikirim ke service worker, digunakan untuk memperbarui status lokal atau menampilkan pemberitahuan kepada pengguna. Karena service worker memiliki thread

terpisah, mereka dapat menerima dan menampilkan notifikasi bahkan ketika browser tidak berjalan.

- **Background sync API:** Memungkinkan Anda menunda tindakan hingga pengguna memiliki konektivitas yang stabil. Berguna untuk memastikan bahwa data terkirim dengan baik. API ini juga memungkinkan server melakukan update secara berkala sehingga saat aplikasi digunakan sudah diperbaharui.
- **Channel Messaging API:** Channel komunikasi antara web workers, service workers dan web app. Contoh: notifikasi konten baru dan updates yang membutuhkan interaksi user.

Service Worker Lifecycle

Lifecycle dari service worker adalah Registration, Installation dan Activation.



Registration dan scope

Registration akan memberitahu browser dimana service worker berada, dan akan diinstal di background.

Contoh:

```
if ('serviceWorker' in navigator) {  
  navigator.serviceWorker.register('/service-worker.js')  
    .then(function(registration) {  
      console.log('Registration successful, scope is:',  
registration.scope);  
    })  
    .catch(function(error) {  
      console.log('Service worker registration failed, error:', error);  
    });  
}
```

Installation

Setelah proses registrasi selesai, tahap instalasi akan dilakukan. Instalasi akan dilakukan jika service workers dianggap baru oleh browser, bisa terjadi karena memang belum ada service worker atau karena ada perbedaan dari service workers yang diinstall sebelumnya.

Install event akan aktif saat proses instalasi. Kita dapat menambahkan install event listener untuk melakukan tugas tertentu. Contoh, selama proses install, service workers dapat melakukan precache bagian dari web app, agar saat loading berikutnya aplikasi dapat dibuka lebih cepat.

```
// Listen for install event, set callback  
self.addEventListener('install', function(event) {  
  // Perform some task  
});
```

Activation

Setelah terinstall, fase berikutnya adalah activation. Jika ada pages terbuka dan dikontrol oleh service workers sebelumnya, service worker baru akan masuk ke waiting state. Service worker baru akan aktif ketika tidak ada lagi halaman yang terbuka yang masih menggunakan service worker sebelumnya. Hal ini untuk memastikan hanya ada 1 versi service worker berjalan.

Ketika service worker diaktivasi, activate event akan terjadi. Pada event listener ini dapat digunakan untuk membersihkan cache yang sudah kedaluarsa.

```
self.addEventListener('activate', function(event) {  
  // Perform some task  
});
```

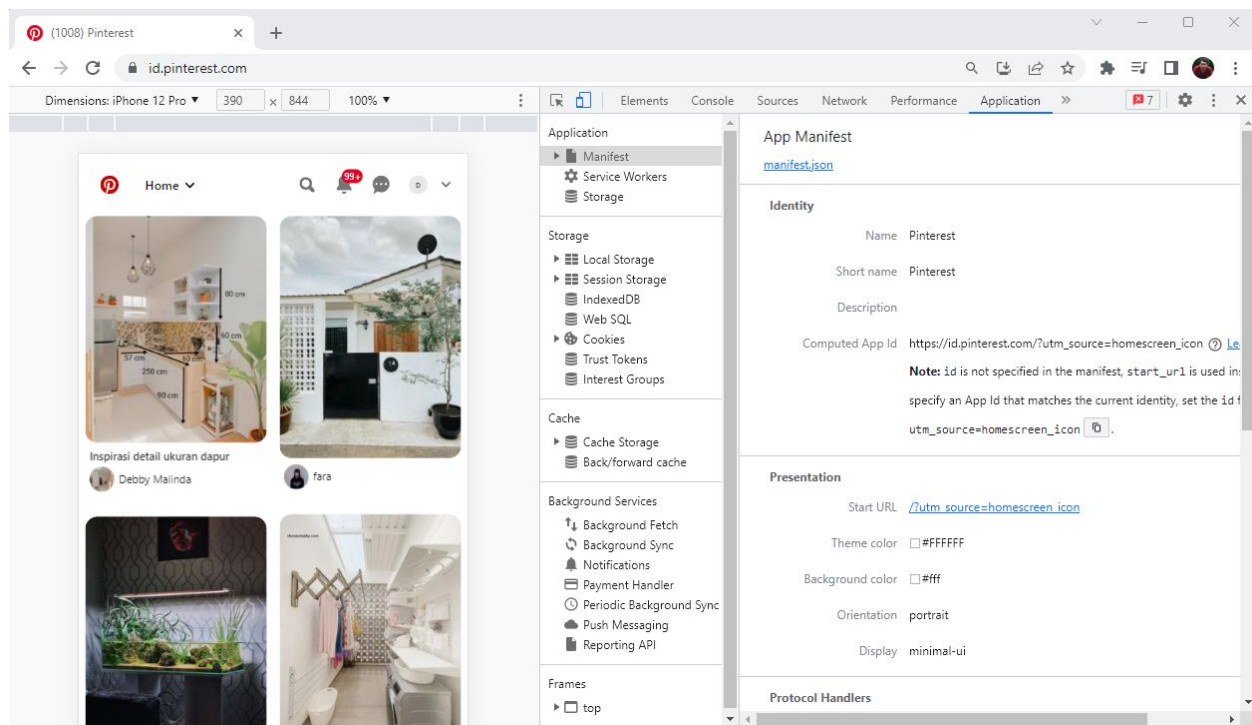
Setelah aktif, service worker akan mengatur seluruh halaman dalam scope, dan mulai mendengarkan events dari halaman tersebut.

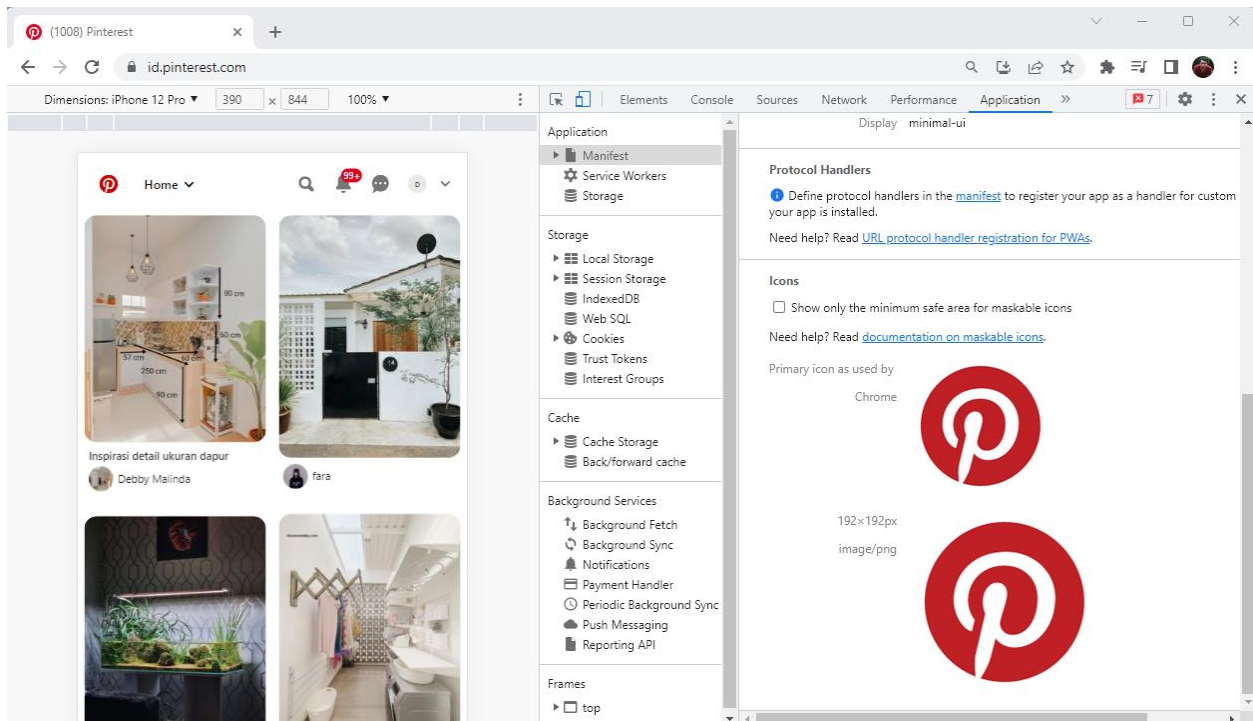
Untuk pages yang diload sebelum service worker aktif tidak akan dikontrol oleh service worker tersebut.

Service worker akan mengambil alih kontrol setelah menutup dan membuka aplikasi kembali. Atau service worker memanggil fungsi `clients.claim`. Hal ini bertujuan menjaga konsistensi dari site.

Web App Manifest

Secara teknis web app manifest adalah JSON yang sederhana yang menjelaskan secara sistematis dan terstruktur mengenai aplikasi kita. Pada web app manifest ini akan memberikan nama aplikasi, icon yang dipakai sebagai pembeda secara visual dengan aplikasi lain dan sebagainya.





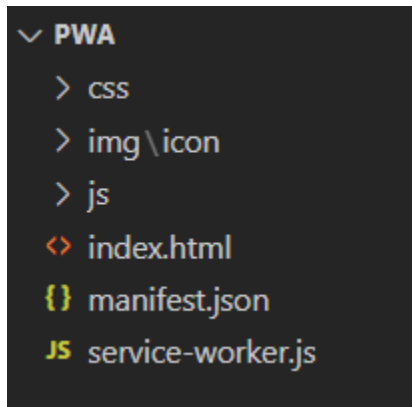
Contoh manifest.json :

```

JS init.js  <> index.html  {} manifest.json  JS service-worker.js
{} manifest.json > ...
1  {
2    "theme_color": "#f69435",
3    "background_color": "#f69435",
4    "display": "standalone",
5    "scope": "/",
6    "start_url": "/index.html",
7    "name": "Belajar PWA",
8    "short_name": "TestPwa",
9    "description": "test",
10   "icons": [
11     {
12       "src": "img/icon/icon-192x192.png",
13       "sizes": "192x192",
14       "type": "image/png"
15     },
16     {
17       "src": "img/icon/icon-256x256.png",
18       "sizes": "256x256",
19       "type": "image/png"
20     },
21     {
22       "src": "img/icon/icon-384x384.png",
23       "sizes": "384x384",
24       "type": "image/png"
25     },
26     {
27       "src": "img/icon/icon-512x512.png",
28       "sizes": "512x512",
29       "type": "image/png"
30     }
31   ]
32 }

```

Dengan susunan file directory :



Jika kita kesulitan untuk membuat manifest.json kita bisa menggunakan bantuan generator online contoh : <https://app-manifest.firebaseapp.com/>

Kita tinggal mengisi beberapa form dan akan otomatis menggenerate manifest.json termasuk icon yang dibutuhkan dengan berbagai ukuran.

A screenshot of the 'Web App Manifest Generator' website. The page has a blue header with the title 'Web App Manifest Generator'. Below the header, there's a description of the Web App Manifest and a form to generate it. The form includes fields for 'App Name', 'Short Name', 'Theme Color' (set to #2196f3), 'Background Color' (set to #2196f3), 'Display Mode' (set to Browser), 'Orientation' (set to Any), and 'Application Scope'. To the right of the form, there's a 'manifest.json' section showing the generated JSON code, a 'COPY' button, and a 'Generate Icons' section with an 'ICON' button and a 'GENERATE .ZIP' button.

Referensi :

- <https://skillplus.web.id/service-workers-teori/>
- <https://medium.com/hammercode/berkenalan-dengan-service-worker-578a45951df7>
- <https://web.dev/learn/pwa>
- <https://sekolahkoding.com/kelas/belajar-progressive-web-app-pwa>
- <https://developers.google.com/codelabs/pwa-training>
- <https://web.dev/add-manifest/>
-