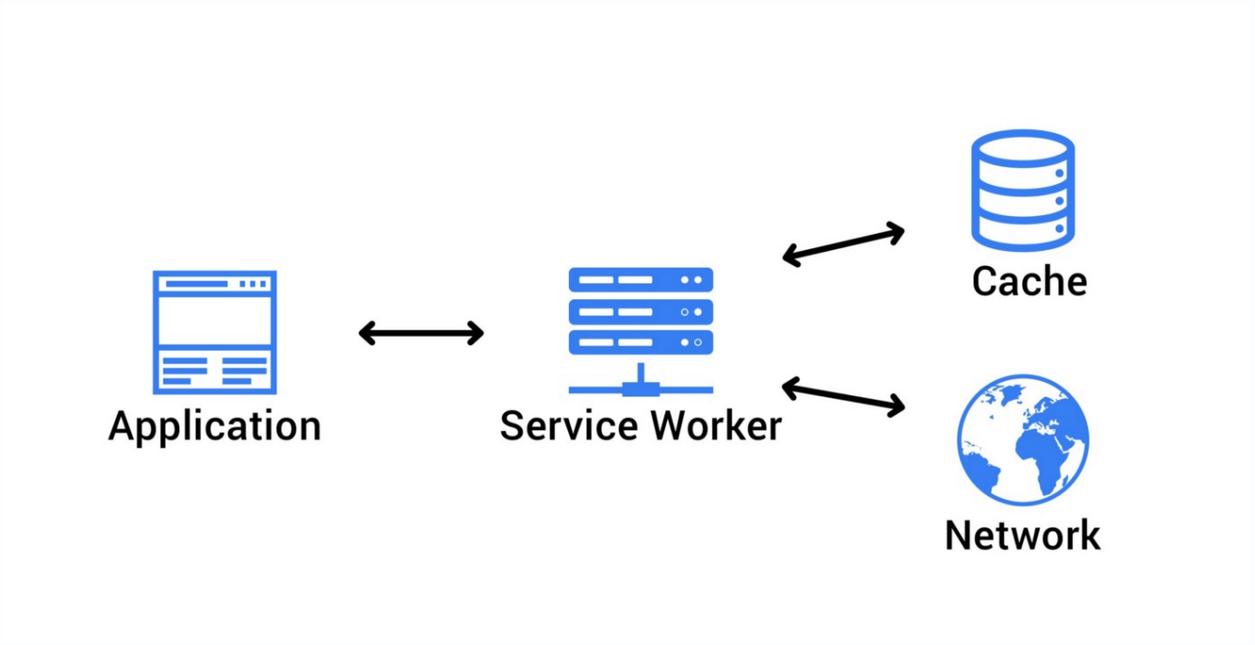
formation.tech

# Service Workers

# Service Workers - Introduction

- Script that run separately from the web page (JavaScript worker)
- Runs in background
- Offers new features :
  - advanced offline control
  - background syncs
  - push notifications
- Can be seen as a programmable network proxy



# Service Workers - Requirements

- Only works in HTTPS (HTTP is possible only on the localhost domain for dev purpose
- Don't have access to the DOM

| IE | Edge * | Firefox | Chrome | Safari | Opera | Safari on iOS * | Opera Mini * | Android Browser * | Opera Mobile * | Chrome for Android | Firefox for Android | UC Browser for Android | Samsung Internet | QQ Browser |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2-32 | | | | | | | | | | | | |
| | | 1 33-43 | | | | | | | | | | | | |
| | | 44 | | | | | | | | | | | | |
| | | 3 45 | | | | | | | | | | | | |
| | | 46-51 | | | | | | | | | | | | |
| | | 3 52 | | | | | | | | | | | | |
| | | 53-59 | | | | | | | | | | | | |
| | | 3 60 | | | | | | | | | | | | |
| | 12-14 | 61-67 | 4-39 | | 10-26 | | | | | | | | | |
| | 2 15-16 | 3 68 | 40-44 | 3.1-11 | 27-31 | 3.2-11.2 | | | | | | | | |
| 6-10 | 17-96 | 69-95 | 45-97 | 11.1-15.1 | 32-82 | 11.3-15.1 | | 2.1-4.4.4 | 12-12.1 | | | | 4-15.0 | |
| 11 | 97 | 96 | 98 | 15.3 | 83 | 15.3 | all | 97 | 64 | 97 | 96 | 12.12 | 16.0 | 10.4 |
| | | 97-98 | 99-101 | 15.4-TP | | 15.4 | | | | | | | | |

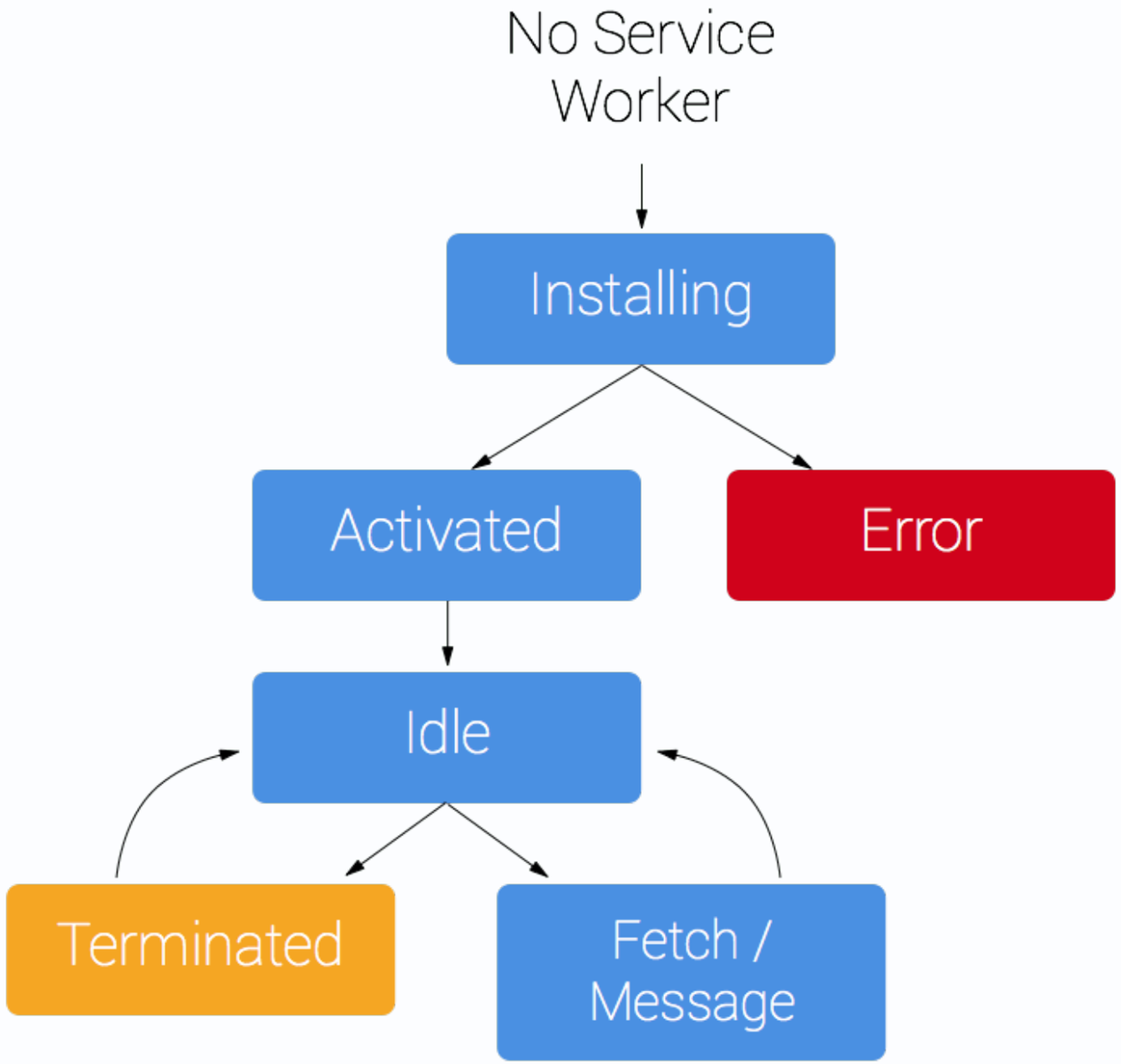
# Service Workers - Registration

- Service Worker need to be registered with the navigator.serviceWorker.register method
- If the content of the Service Worker changes, it will be executed again
- If not the register method is called but the Service Worker did not change, nothing will happen (so we don't have to check if the SW was previously registered)

```
// sw.js
console.log("Will appear on the first registration");
```

```
window.addEventListener('load', async () => {
  try {
    const registration = await navigator.serviceWorker.register('sw.js');
    // Registration was successful
    console.log('ServiceWorker registration successful with scope: ', registration.scope);
  } catch (err) {
    // Registration failed
    console.log('ServiceWorker registration failed: ', err);
  }
});
```

# Service Workers - Lifecycle

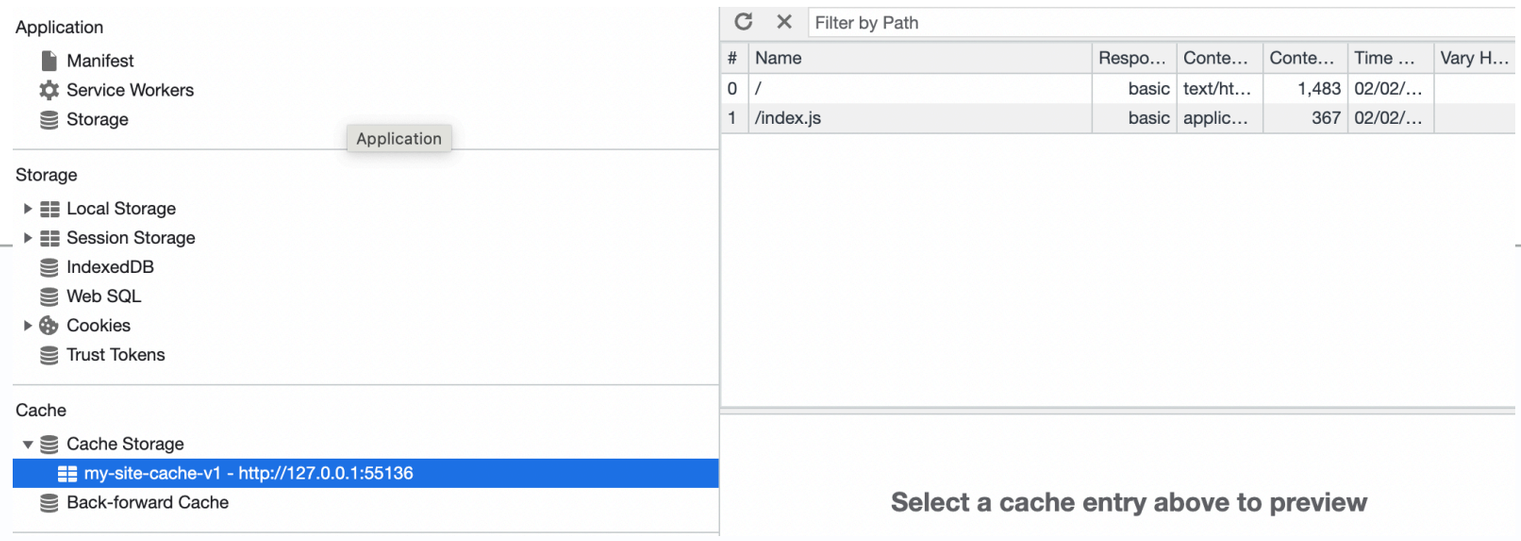


# Service Workers - Install event

- With the install event we can cache static resources (HTML documents, CSS, Scripts, Images...)
- waitUntil() will hold the service worker in the installing phase until tasks complete
- once all assets are fetched and cached, the registration is complete
- when a new version of the app is available, update the cache key

```
// sw.js
const ASSETS_CACHE_KEY = 'my-site-assets-v1';
const urlsToCache = ['/', '/index.js'];

self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(ASSETS_CACHE_KEY).then((cache) => {
      return cache.addAll(urlsToCache);
    }),
  );
});
```



# Service Workers - Install event

- If you want a faster registration of the service worker you can separate important resource from those that can appear on secondary pages

```
// sw.js
const CACHE_KEY = 'my-app-v1';
const blockingUrlsToCache = ['/', '/index.js'];
const nonBlockingUrlsToCache = ['javascript-logo.svg'];

self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_KEY).then((cache) => {
      cache.addAll(nonBlockingUrlsToCache);

      return cache.addAll(blockingUrlsToCache);
    })
  );
});
```

# Service Workers - Fetch event

- The fetch event acts as a network proxy
- This is where we create the offline ability
- Cache first :

```
self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => {
      if (response) {
        console.log('Cache hit');
        return response;
      }
      return fetch(event.request);
    }),
  );
});
```

- Network first :

```
self.addEventListener('fetch', (event) => {
  event.respondWith(
    fetch(event.request)
      .catch(() => caches.match(event.request))
  );
});
```

# Service Workers - Fetch event

- Cache on network

```
self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => {
      if (response) {
        console.log('Cache hit');
        return response;
      }
      console.log('Network call');
      return fetch(event.request).then((response) => {
        if (!response || response.status !== 200) {
          return response;
        }
        const responseToCache = response.clone();

        caches.open(CACHE_KEY).then((cache) => {
          return cache.put(event.request, responseToCache);
        });

        return response;
      });
    }),
  );
});
```

- Important : clone the response so it will readable again later (res.json()...)

# Service Workers - Activate event

- When we will deploy a new version of our app, we will have to update the cache key so the code in the service worker will change
- It could be time to delete the cache associated to the old cache key :

```
self.addEventListener('activate', (event) => {
  event.waitUntil(
    caches.keys().then((cacheKeys) =>
      Promise.all(
        cacheKeys.map((cacheKey) => {
          if (cacheKey !== CACHE_KEY) {
            return caches.delete(cacheKey);
          }
        }),
      ),
    ),
  );
});
```

# Service Workers - Update

- When the browser detects a new version of our service worker it won't activate it automatically
- The user will activate it after a navigation (back in history, links) or by closing the tab
- The developer can activate it within the DevTools by clicking on "skipWaiting"
- Refreshing the page isn't enough except if you check "Update on reload" in the DevTools
- You can do it automatically by calling self.skipWaiting() in the install event

```
self.addEventListener('install', (event) => {
  self.skipWaiting();
  event.waitUntil(
    caches.open(CACHE_KEY).then((cache) => {
      return cache.addAll(urlsToCache);
    }),
  );
});
```

# Service Workers - Update

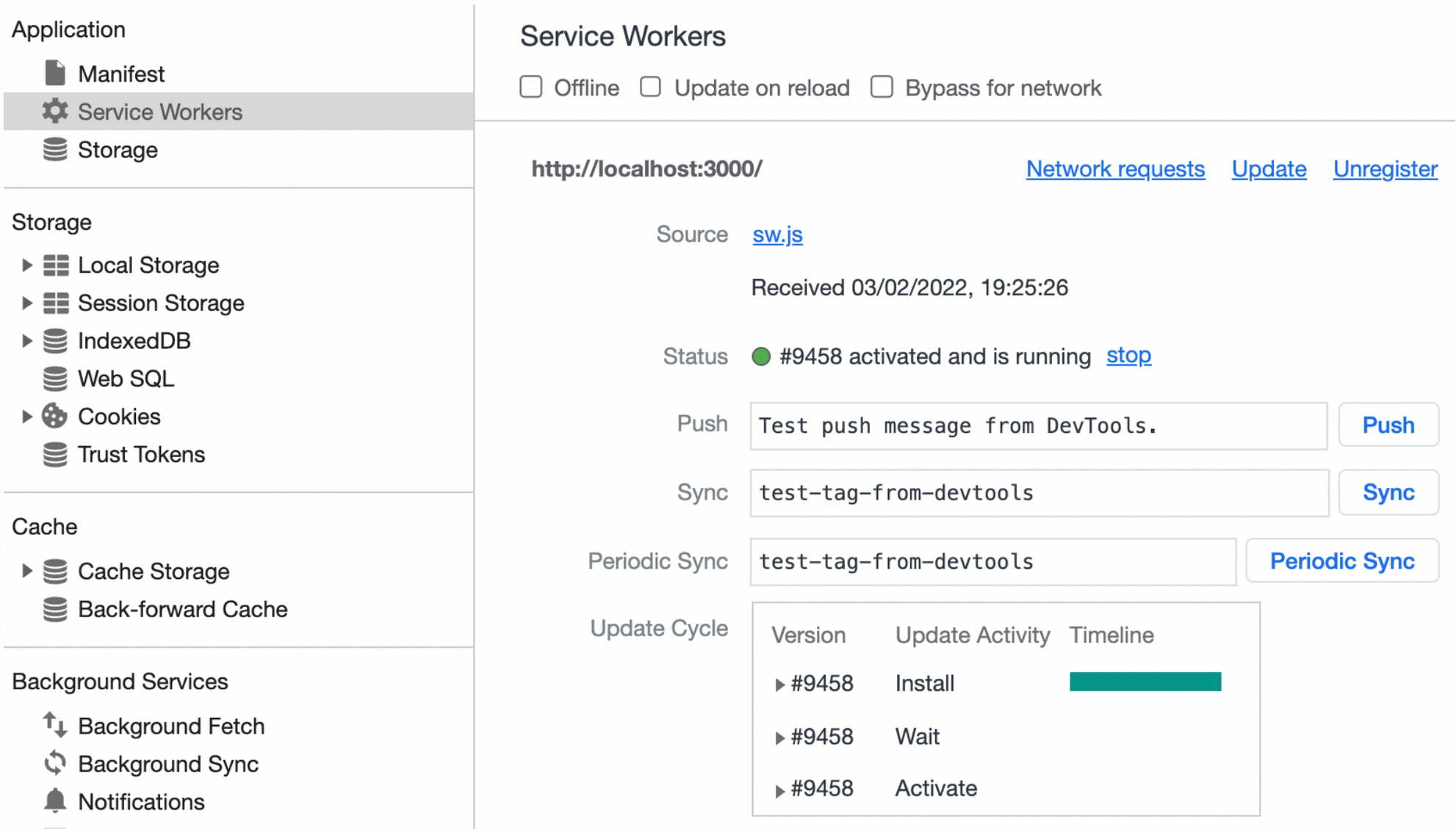
- When the user visits the app for the first time, the service worker is register but network traffic won't be proxied until a page refresh
- By calling clients.claim() in the activate event, the service worker will become immediately available

```
self.addEventListener('activate', (event) => {
  event.waitUntil(clients.claim());
});
```

# Service Workers - Update

# Service Workers - Tools

- chrome://serviceworker-internals/
- Chrome DevTools



# Service Workers - Resources

- https://web.dev/offline-cookbook/
- https://developers.google.com/web/ilt/pwa/caching-files-with-service-worker
- https://deanhume.com/displaying-a-new-version-available-progressive-web-app/
- https://developers.google.com/web/fundamentals/primers/service-workers
- https://github.com/mdn/serviceworker-cookbook

# Service Workers - Workbox

- Workbox is a set of tools created by Google who intend to simplify the creation of the service worker
- It is recommended to use it with a bundler such as webpack or Rollup
- It can created the service worker automatically from the built assets or be configure with helper functions

# Service Workers - Workbox

- To use it with webpack, simply install workbox-webpack-plugin
  `npm i workbox-webpack-plugin -D`
- The package exports 2 plugins :
  - GenerateSW : that will generate the Service Worker from the built files and a config object
  - InjectManifest : that will built the Service Worker from a configuration

# Service Workers - Workbox

- The GenerateSW class takes options such as :
  - clientsClaim: true/false
  - skipWaiting: true/false

```
const HtmlWebpackPlugin = require('html-webpack-plugin');
const { GenerateSW } = require('workbox-webpack-plugin');

/** @type {import('webpack').Configuration} */
const config = {
  entry: './src/index.js',
  plugins: [
    new HtmlWebpackPlugin({
      template: './src/index.html',
    }),
    new GenerateSW(),
  ],
};

module.exports = config;
```

```
webpack --mode production

LOG from GenerateSW
<i> The service worker at service-worker.js will precache
<i>         2 URLs, totaling 636 B.
```

# Service Workers - Workbox

- The InjectManifest plugin take the path to the service worker with the swSrc option

```
const HtmlWebpackPlugin = require('html-webpack-plugin');
const { InjectManifest } = require('workbox-webpack-plugin');

/** @type {import('webpack').Configuration} */
const config = {
  entry: './src/index.js',
  plugins: [
    new HtmlWebpackPlugin({
      template: './src/index.html',
    }),
    new InjectManifest({
      swSrc: './src/service-worker.js'
    }),
  ],
};

module.exports = config;
```

- In the service worker you will have access to ES Modules and have at least to put this code to precache the built files :

```
import { precacheAndRoute } from 'workbox-precaching';

precacheAndRoute(self.__WB_MANIFEST);
```

# Service Workers - Workbox

- Most of the configuration will then be the workbox-strategies module https://developers.google.com/web/tools/workbox/modules/workbox-strategies
- These strategies are described is this guide: https://web.dev/offline-cookbook/

```
import { precacheAndRoute } from 'workbox-precaching';
import { registerRoute } from 'workbox-routing';
import { NetworkFirst, CacheFirst } from 'workbox-strategies';
import { CacheableResponse } from 'workbox-cacheable-response';
import { ExpirationPlugin } from 'workbox-expiration';

precacheAndRoute(self.__WB_MANIFEST);

registerRoute(
  ({ url }) => url.origin === 'https://jsonplaceholder.typicode.com',
  new NetworkFirst({
    cacheName: 'jsonplaceholder',
    plugins: [
      new CacheableResponse({ statuses: [200] }),
    ]
  })
);

registerRoute(
  ({ request }) => request.destination === 'image',
  new CacheFirst({
    cacheName: 'images',
    plugins: [
      new ExpirationPlugin({ maxAgeSeconds: 60 * 60 * 24, maxEntries: 50 }),
    ]
```

formation.tech

# Progressive Web Apps

# Progressive Web Apps - Introduction

- Progressive Web Apps is a marketing term created by Google that illustrate web apps created by a number of new web APIs
- A PWA app is :
  - Discoverable, so the contents can be found through search engines.
  - Installable, so it can be available on the device's home screen or app launcher.
  - Linkable, so you can share it by sending a URL.
  - Network independent, so it works offline or with a poor network connection.
  - Progressively enhanced, so it's still usable on a basic level on older browsers, but fully-functional on the latest ones.
  - Re-engageable, so it's able to send notifications whenever there's new content available.
  - Responsively designed, so it's usable on any device with a screen and a browser—mobile phones, tablets, laptops, TVs, refrigerators, etc.
  - Secure, so the connections between the user, the app, and your server are secured against any third parties trying to get access to sensitive data.

# Progressive Web Apps - Introduction

- Web APIs to create a PWA
  - Service Worker
  - Web App Manifest
  - Push Notifications
  - Add to Home Screen
  - ...

# Progressive Web Apps - Web app manifests

- The web app manifest provides information about a web application
- PWA manifests include its name, author, icon(s), version, description, and list of all the necessary resources
- Generate the icons with https://www.pwabuilder.com/imageGenerator
- Example :

```
{
  "$schema": "https://json.schemastore.org/web-manifest-combined.json",
  "name": "My wonderful app",
  "short_name": "MyApp",
  "start_url": ".",
  "display": "standalone",
  "background_color": "#fff",
  "description": "A demo for our training",
  "icons": [{
    "src": "images/touch/homescreen48.png",
    "sizes": "48x48",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen192.png",
    "sizes": "192x192",
    "type": "image/png"
  }]
}
```

# Progressive Web Apps - Web app manifests

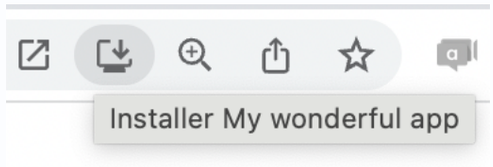
- Include the manifest with a link tag

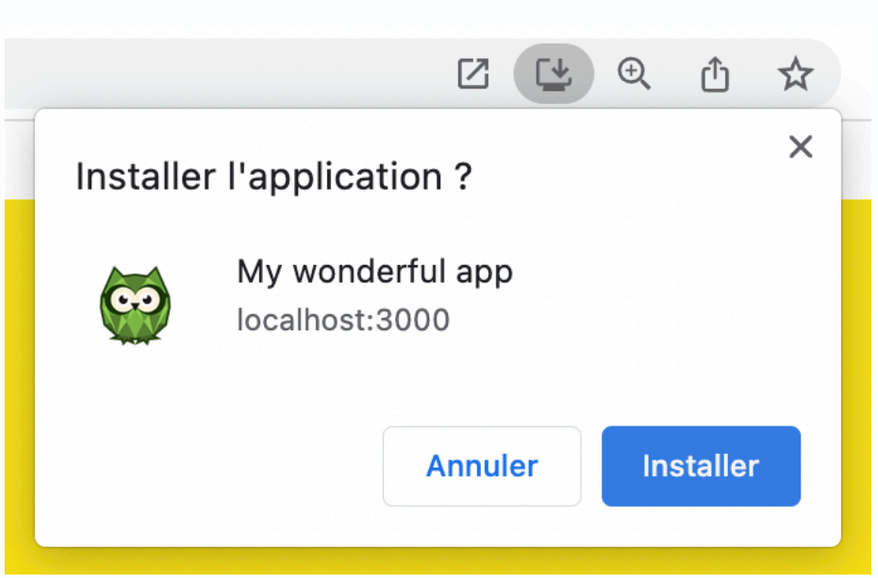
```
<link rel="manifest" href="/assets/app.webmanifest">
```

- In some browsers (Chrome 47 and later, for example), a splash screen is displayed for sites launched from a homescreen. This splash screen is auto-generated from properties in the web app manifest, specifically:
  - name
  - background_color
  - The icon in the icons array that is closest to 128dpi for the device.

# Progressive Web Apps - Add to Home Screen

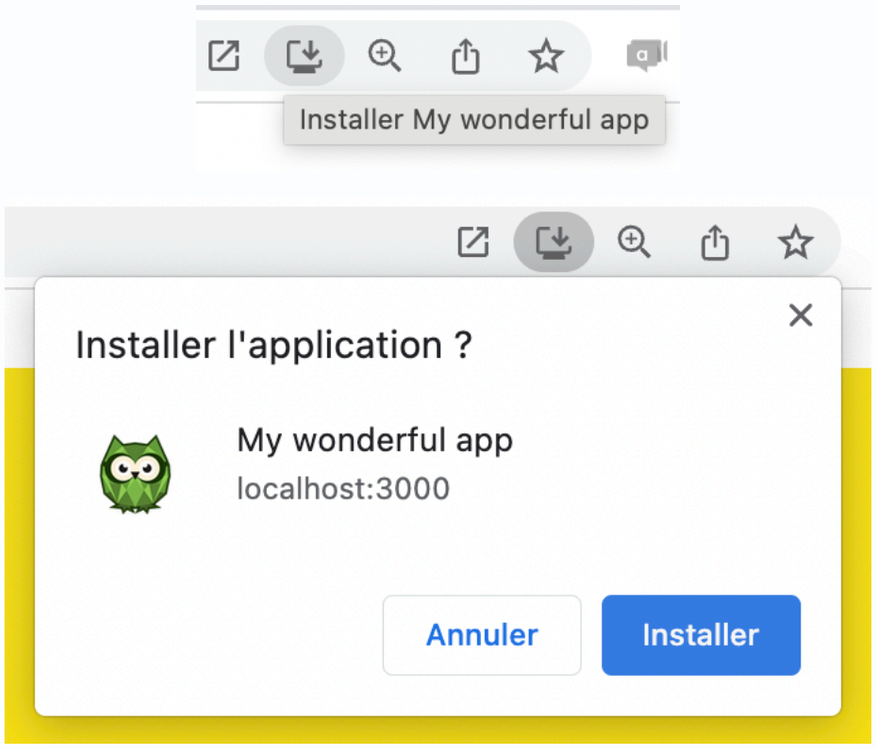
- Add to Home screen (or A2HS for short) is a feature available in modern browsers that allows a user to "install" a web app



# Progressive Web Apps - Add to Home Screen

- Add to Home screen (or A2HS for short) is a feature available in modern browsers that allows a user to "install" a web app




- Your app can show a specific UI to prompt the installation : https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Add_to_home_screen#javascript_for_handling_the_install

# Progressive Web Apps - Push Notification

- The Push API gives web applications the ability to receive messages pushed to them from a server, whether or not the web app is in the foreground, or even currently loaded
- It requires the use of a service worker
- Demo : https://github.com/mdn/serviceworker-cookbook/tree/master/push-simple