PWA

Outline

- What is PWA
- What PWA Can Do
- Install criteria
- Service worker
- Caching Strategy
- Background Sync
- Push notification

What is PWA?

- PWA is a WebApp that looks and behaves as if it is a native app.

- It take advantage of native mobile device features, without requiring the end user to install native app

What PWA Can Do

- Native File System
- Speech Recognition
- Speech synthesis
- Authentication
- Audio Recording
- Vibration
- Wake Lock
- Orientation
- Motion
- NFC
- Multitouch
- Contact Picker
- Bluetooth
- GeoLocation…

More info https://whatpwacando.today/

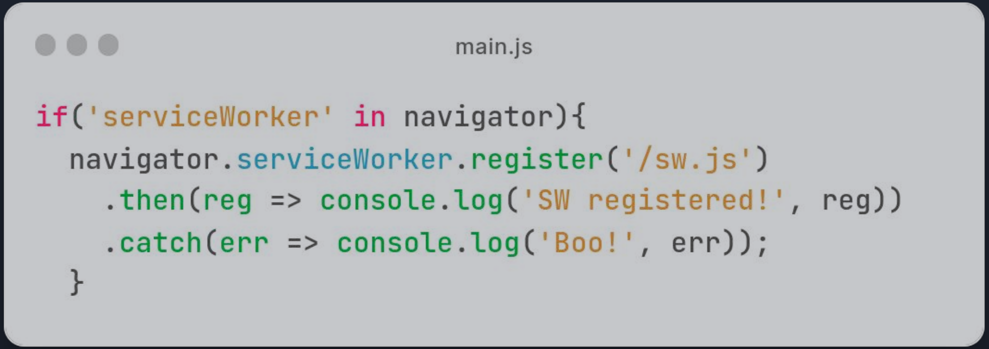
Install Criteria

- Be Served over HTTPS
- Registers a service worker with a **fetch** handler
- Includes a web app manifest
- Meets the user engagement heuristics:
  - The user needs to have clicked or tapped on the page at least once
  - The user needs to have spent at least 30 seconds viewing the page (at any time)

Service Worker

Service Worker

- Acts as programmable proxy between the webpage and the server
- Non blocking
- Can’t interact with the DOM directly


main.js

```
if('serviceWorker' in navigator){
  navigator.serviceWorker.register('/sw.js')
    .then(reg => console.log('SW registered!', reg))
    .catch(err => console.log('Boo!', err));
  }
```

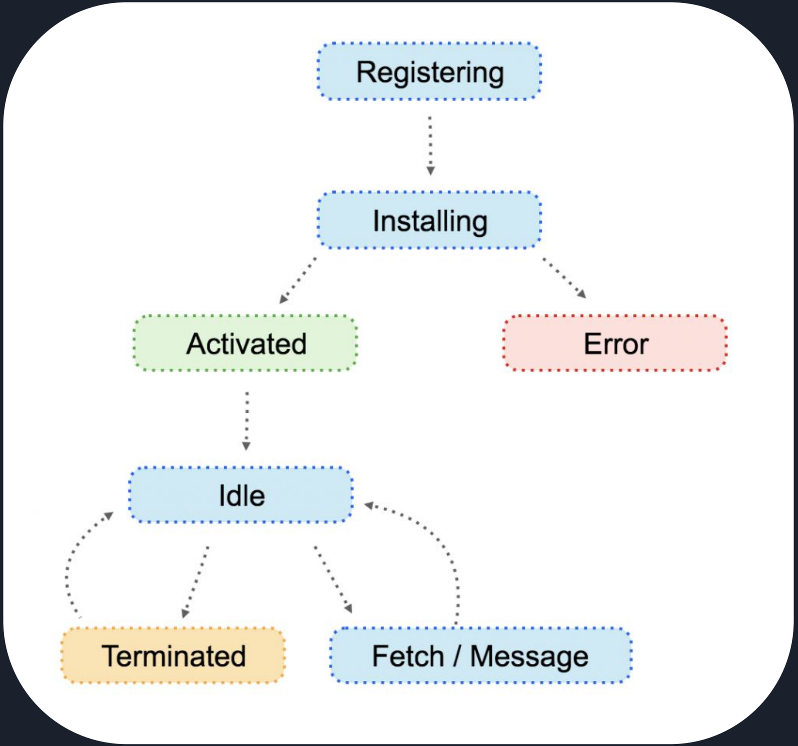
Life Cycle

Install : triggered as soon as the worker executes.

- To cache core file of web page you need, i.e custom Fallback page, main files

Activate : when SW is ready to control clients and handle functional events like push and sync

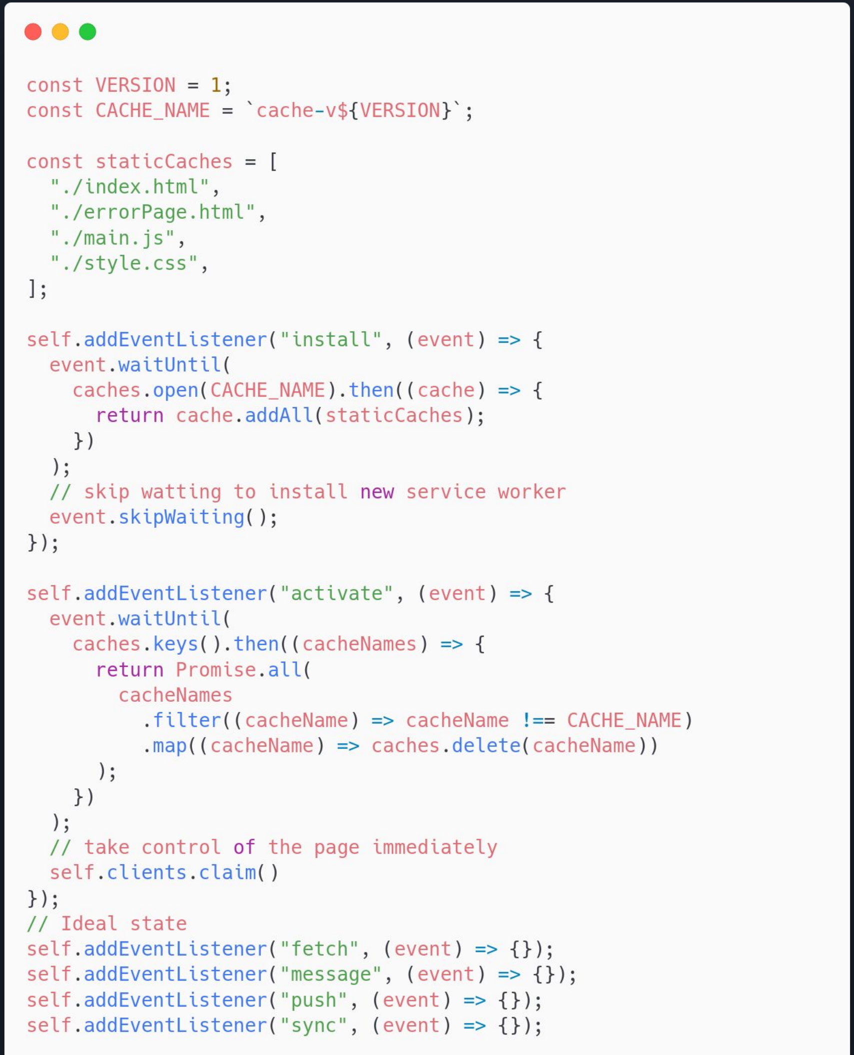
- Ideal for Removing old SW cache files



event.waitUntil(asyncFunction()) : to tell the SW to awaits until the async Function finish

event.skipWatting() : skip waiting to install the new service worker

self.clients.claim()) : to take control the page immediately by new SW



```
const VERSION = 1;
const CACHE_NAME = `cache-v${VERSION}`;

const staticCaches = [
  "./index.html",
  "./errorPage.html",
  "./main.js",
  "./style.css",
];

self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      return cache.addAll(staticCaches);
    })
  );
  // skip watting to install new service worker
  event.skipWaiting();
});

self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames
          .filter((cacheName) => cacheName !== CACHE_NAME)
          .map((cacheName) => caches.delete(cacheName))
      );
    })
  );
  // take control of the page immediately
  self.clients.claim()
});
// Ideal state
self.addEventListener("fetch", (event) => {});
self.addEventListener("message", (event) => {});
self.addEventListener("push", (event) => {});
self.addEventListener("sync", (event) => {});
```

Caching Strategy

Before Diving into

Request Object

# request.destination : It describes the type of content being requested.

audio, audioworklet, document, embed, font, frame, iframe, image, manifest, object, paintworklet, report, script, sharedworker, style, track, video, worker or xslt strings, or the empty string, which is the default value.

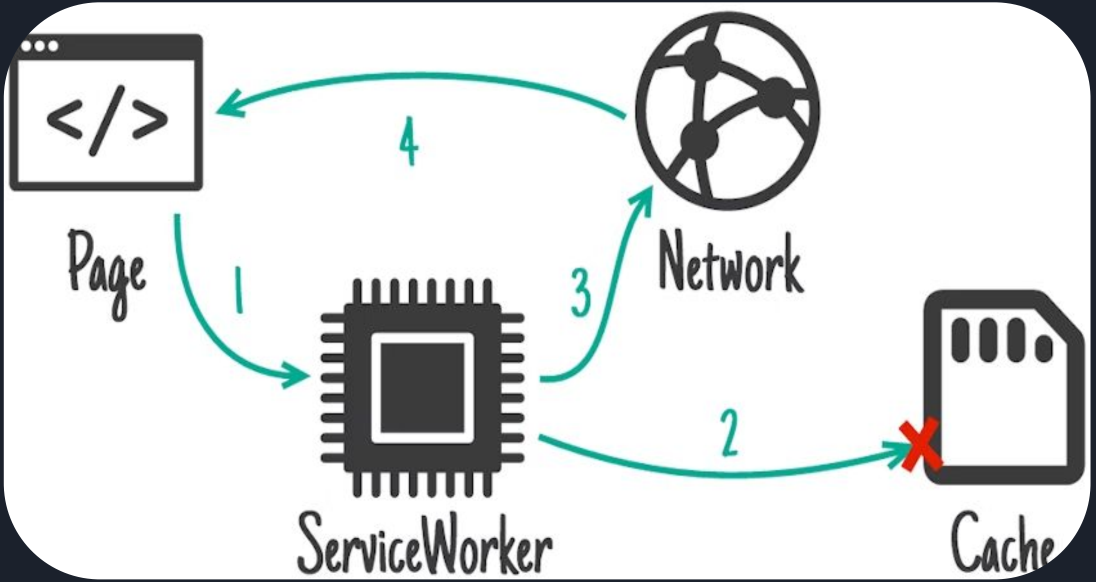
#request.mode : the mode of the request (e.g., cors, no-cors, same-origin, navigate or websocket.)

#request.url : requested url

#request.method : GET, POST, PUT, PATCH, DELETE, HEAD

Cache First

1. The request hits the cache, serve it from cache.
2. If the request is not in the cache, go to the network.
3. Once the network request finishes, add it to the cache, then return the response from the network.



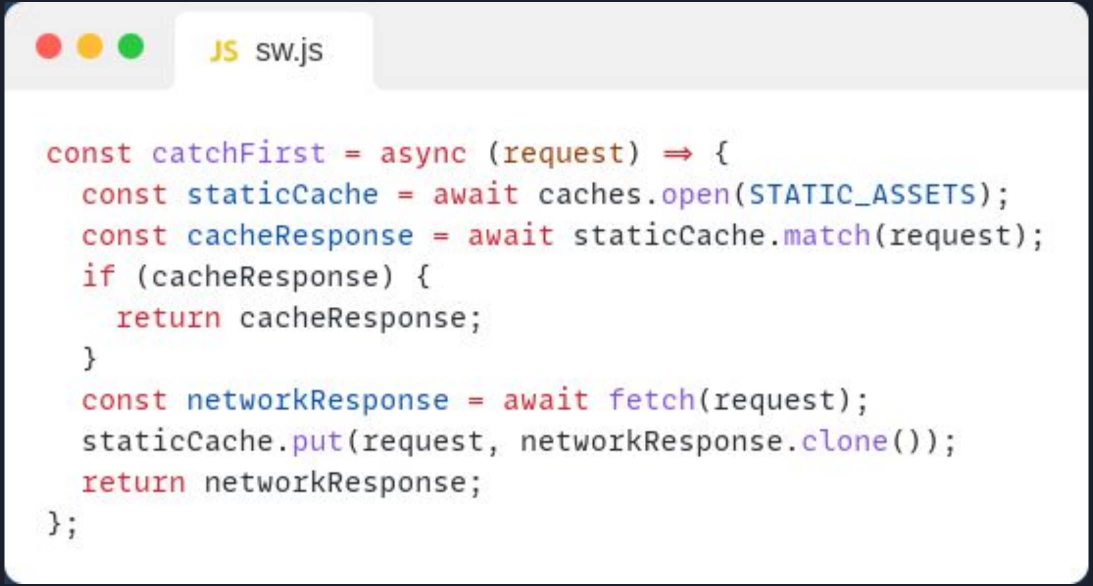
Fast on all network type 🤔

😟 Outdated data

...Continued

Use Cases

- For Static files that couldn't change frequently
- Eg. js, css, fonts, static images...


JS sw.js

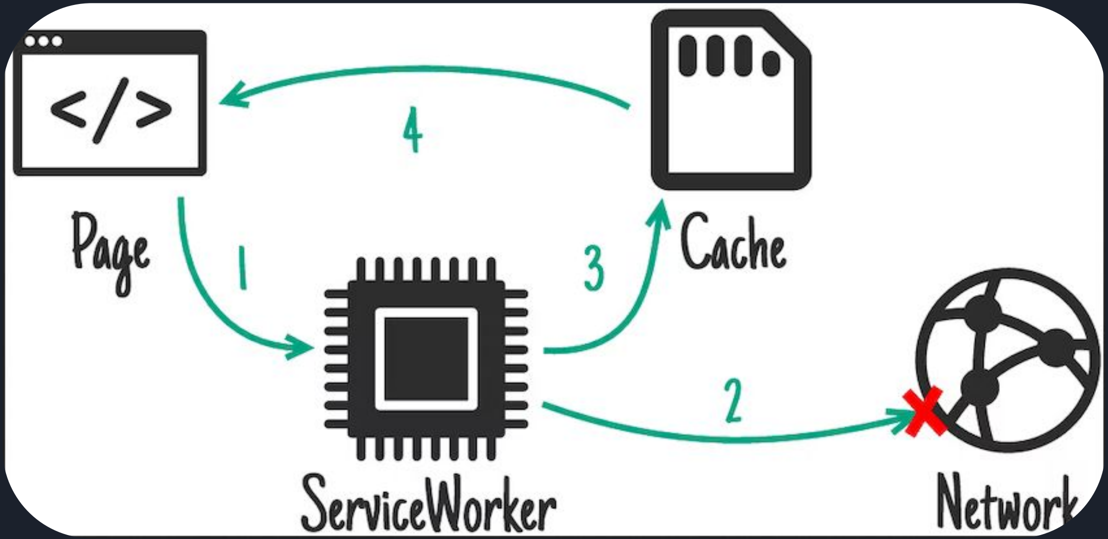
```
const catchFirst = async (request) => {
  const staticCache = await caches.open(STATIC_ASSETS);
  const cacheResponse = await staticCache.match(request);
  if (cacheResponse) {
    return cacheResponse;
  }
  const networkResponse = await fetch(request);
  staticCache.put(request, networkResponse.clone());
  return networkResponse;
};
```

Network First

1. Go to network first for a request, and place the response in the cache.
2. If offline at a later point, fall back to the latest version of that response in the cache.

Fresh uptodate data 🤪

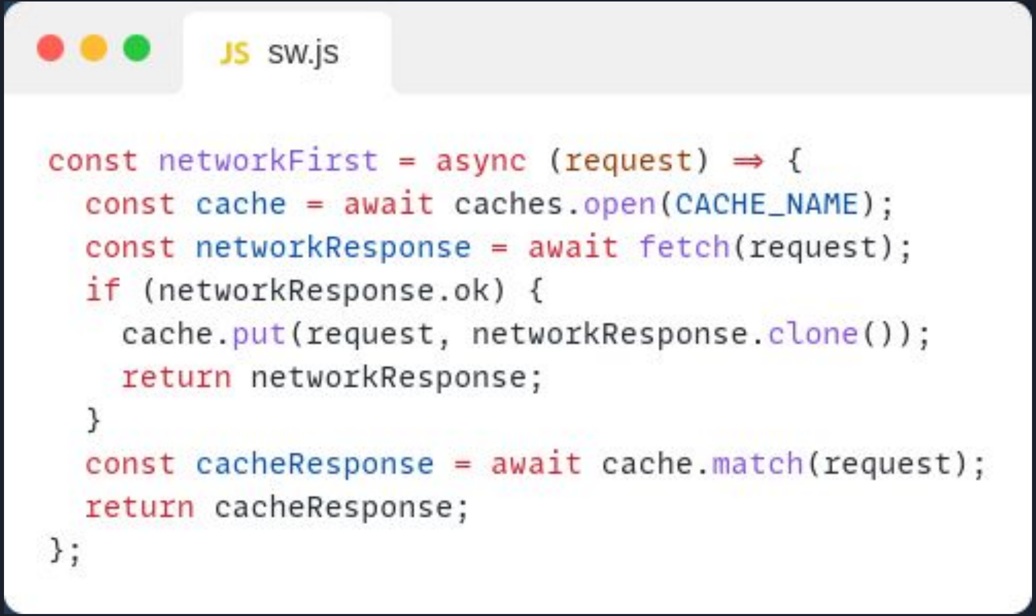
😟 Slow on bad connection



...Continued

Use Cases

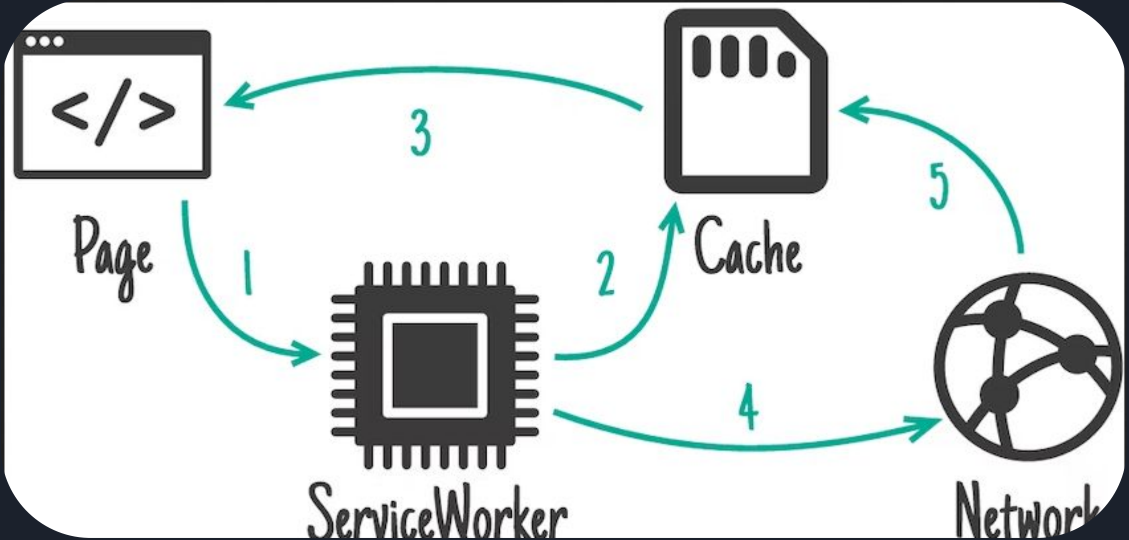
- For Resource that changes frequently
- Great for caching API requests for offline use



```
const networkFirst = async (request) => {
  const cache = await caches.open(CACHE_NAME);
  const networkResponse = await fetch(request);
  if (networkResponse.ok) {
    cache.put(request, networkResponse.clone());
    return networkResponse;
  }
  const cacheResponse = await cache.match(request);
  return cacheResponse;
};
```

Stale-while-revalidate

1. First request request from network and cache it,
2. On subsequent requests, serve from the cache first, then "in the background," refetch from network and update the cache entry.



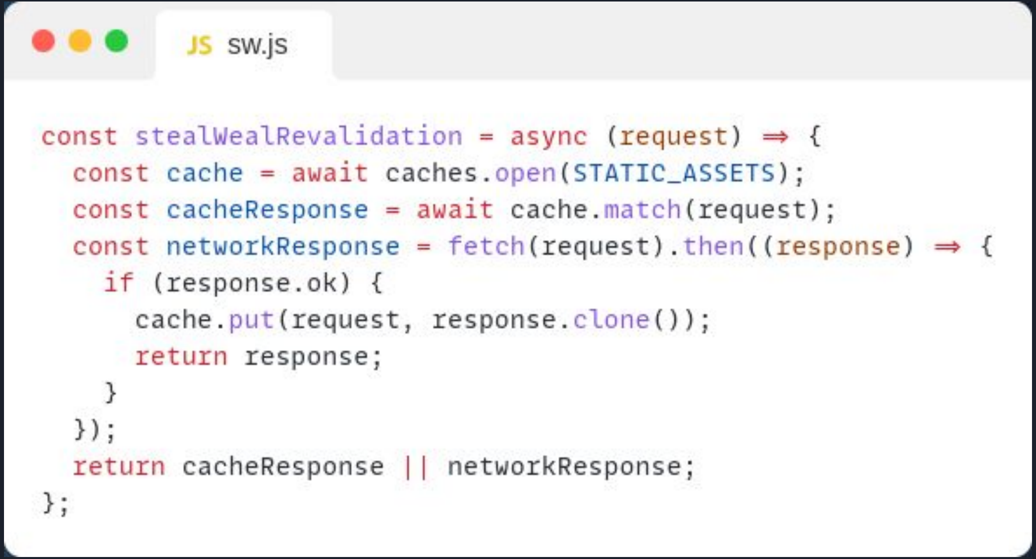
Fast on all network type

Uptodate but not latest data

...Continued

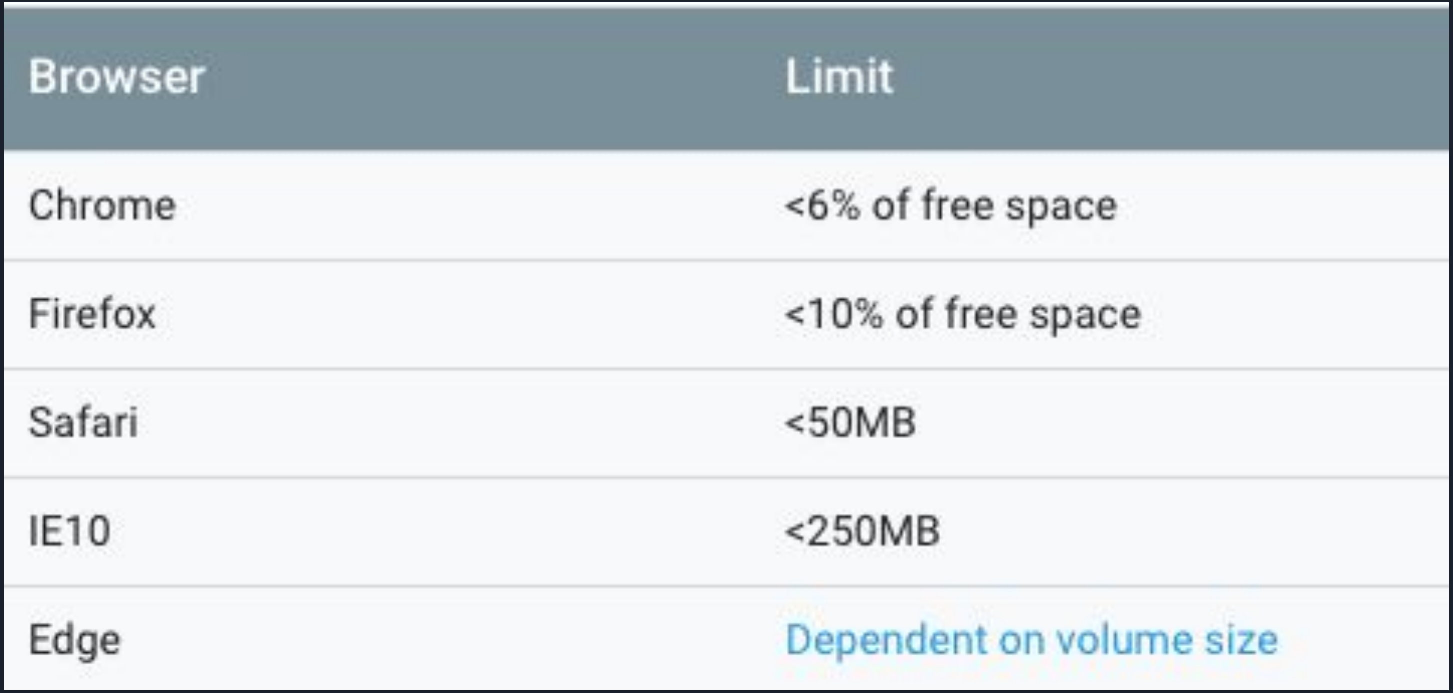
Use Cases

- frequently updating resources where having the very latest version is non-essential.


JS sw.js

```
const stealWealRevalidation = async (request) => {
  const cache = await caches.open(STATIC_ASSETS);
  const cacheResponse = await cache.match(request);
  const networkResponse = fetch(request).then((response) => {
    if (response.ok) {
      cache.put(request, response.clone());
      return response;
    }
  });
  return cacheResponse || networkResponse;
};
```

How much can I store?



Browser	Limit
Chrome	<6% of free space
Firefox	<10% of free space
Safari	<50MB
IE10	<250MB
Edge	Dependent on volume size

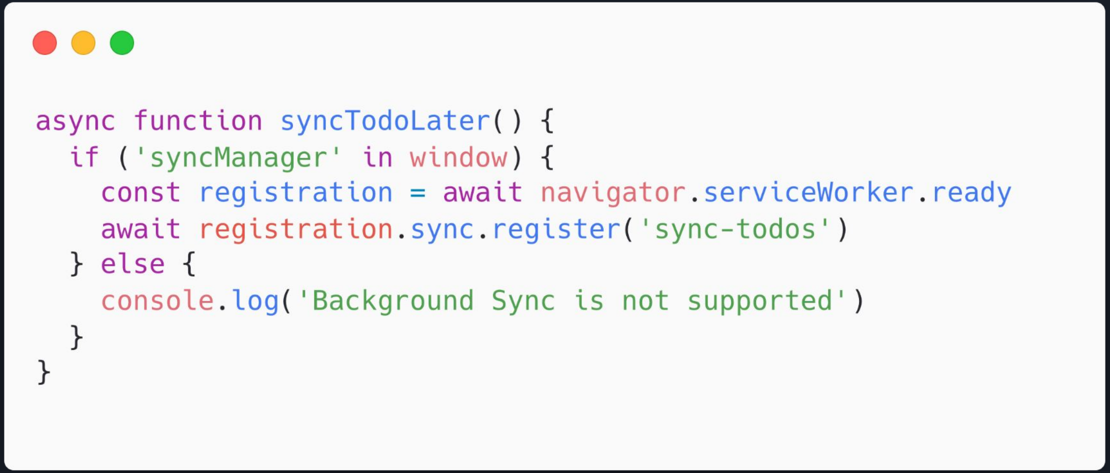
Background Sync

What is Background Sync

Allows web applications to defer tasks to be run in a service worker until the user has a stable network connection

SyncManager

Registers tasks to be run in a service worker at a later time with network connectivity. These tasks are referred to as background sync requests.

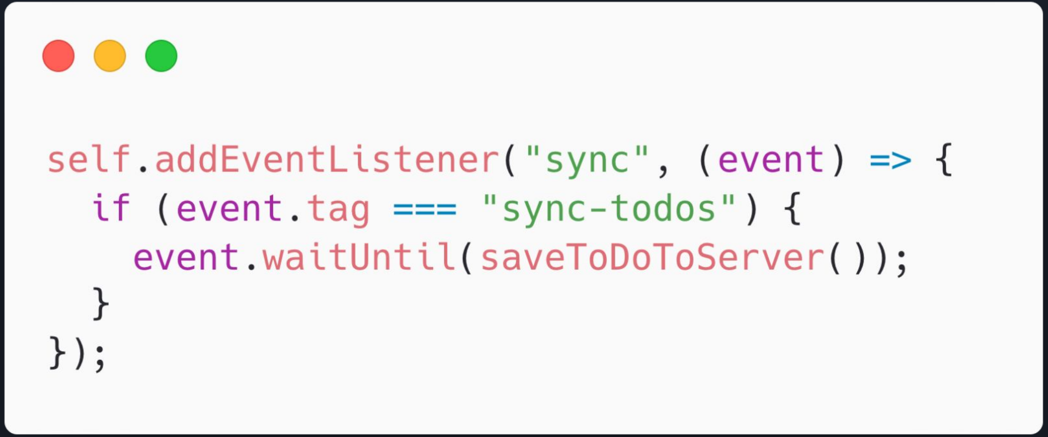


```
async function syncTodoLater() {
  if ('syncManager' in window) {
    const registration = await navigator.serviceWorker.ready
    await registration.sync.register('sync-todos')
  } else {
    console.log('Background Sync is not supported')
  }
}
```

SyncEvent

Represents a synchronization event, sent to the global scope of a ServiceWorker.

It provides a way to run tasks in the service worker with network connectivity.



```
self.addEventListener("sync", (event) => {
  if (event.tag === "sync-todos") {
    event.waitUntil(saveToDoToServer());
  }
});
```

Push Notifications

What is Push notification

Enable to bring information to users even when they're not using the website.

Benefit

# For users

- A way to receive timely, relevant, and precise information.

# For website owner

- A way to increase user engagement.

Implementation

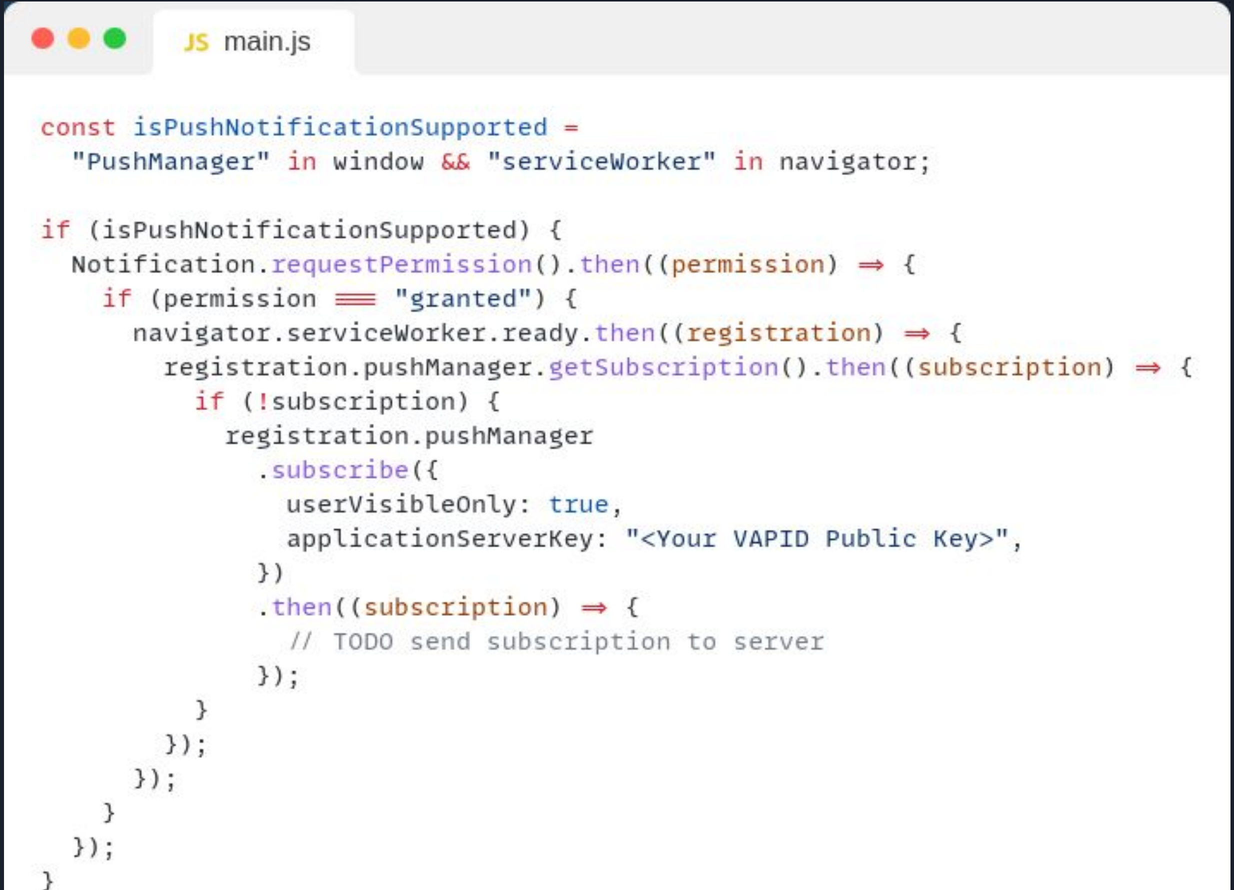
Client Side

1. Request permission to show notification
2. Subscribe a user to push and send subscription data to server
3. Show notification when receive “push” event in service worker

Server Side

1. Generate vapid key
2. Save client subscription to database
3. Send push notification to client

Subscribe Push – client

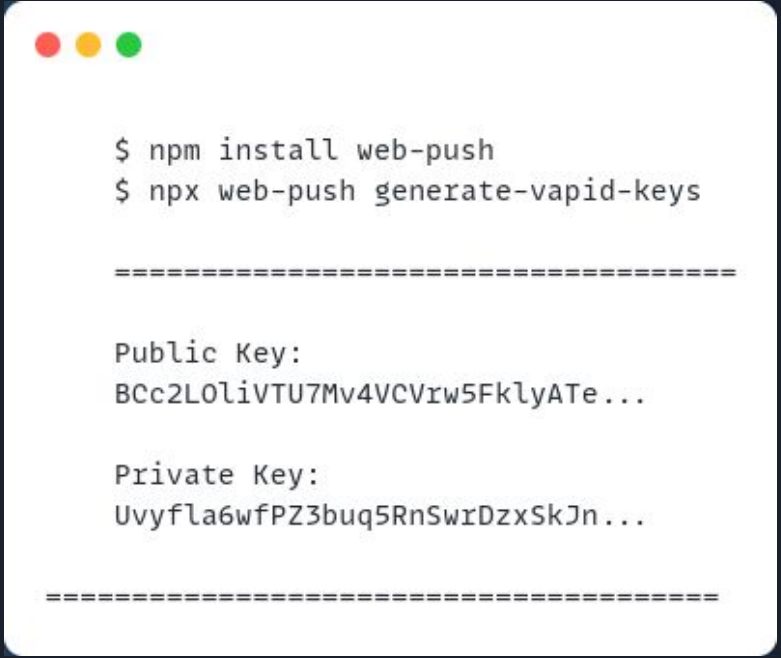


```javascript
const isPushNotificationSupported =
  "PushManager" in window && "serviceWorker" in navigator;

if (isPushNotificationSupported) {
  Notification.requestPermission().then((permission) => {
    if (permission === "granted") {
      navigator.serviceWorker.ready.then((registration) => {
        registration.pushManager.getSubscription().then((subscription) => {
          if (!subscription) {
            registration.pushManager
              .subscribe({
                userVisibleOnly: true,
                applicationServerKey: "<Your VAPID Public Key>",
              })
              .then((subscription) => {
                // TODO send subscription to server
              });
          }
        });
      });
    }
  });
}
```

Generate Vapid Keys – server

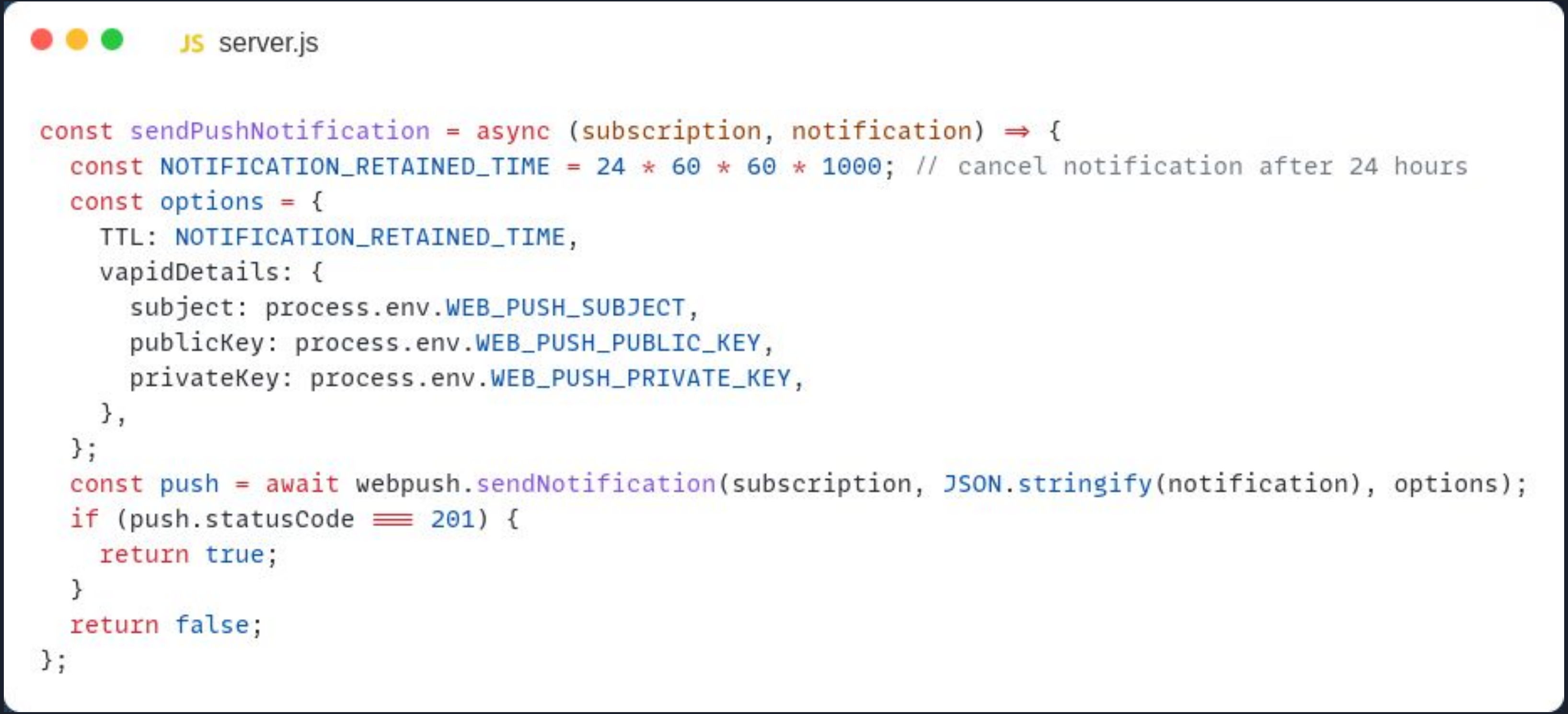
VAPID means “Voluntary Application Server Identification” for Web Push and is defined by RFC 8292.

it adds an additional layer of protection to the subscriptions, since they can only be used by someone who has the VAPID private key



```
$ npm install web-push
$ npx web-push generate-vapid-keys

=======================================

Public Key:
BCc2LOliVTU7Mv4VCVrw5FklyATe...

Private Key:
Uvyfla6wfPZ3buq5RnSwrDzxSkJn...

=======================================
```

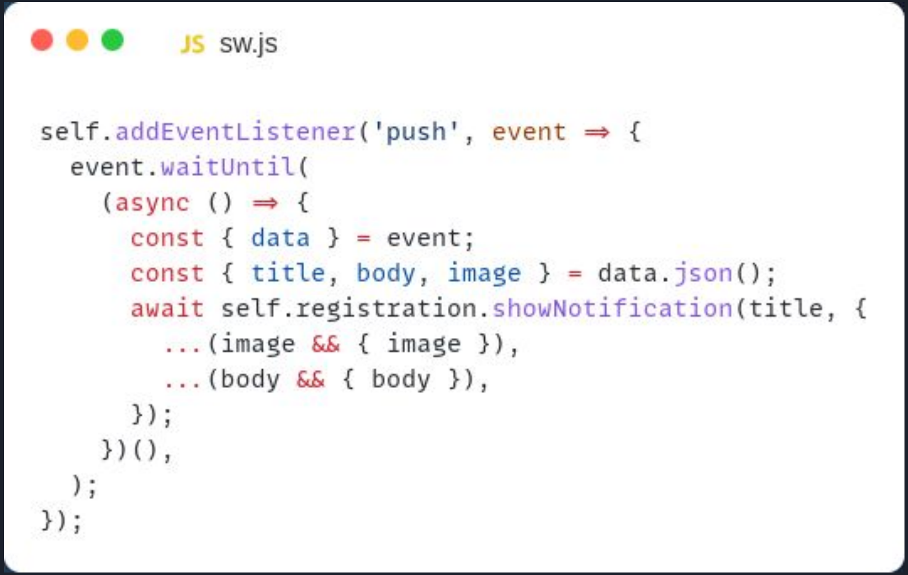
Send Push Notification – server



```javascript
const sendPushNotification = async (subscription, notification) => {
  const NOTIFICATION_RETAINED_TIME = 24 * 60 * 60 * 1000; // cancel notification after 24 hours
  const options = {
    TTL: NOTIFICATION_RETAINED_TIME,
    vapidDetails: {
      subject: process.env.WEB_PUSH_SUBJECT,
      publicKey: process.env.WEB_PUSH_PUBLIC_KEY,
      privateKey: process.env.WEB_PUSH_PRIVATE_KEY,
    },
  };
  const push = await webpush.sendNotification(subscription, JSON.stringify(notification), options);
  if (push.statusCode === 201) {
    return true;
  }
  return false;
};
```

Show Push Notifications – client

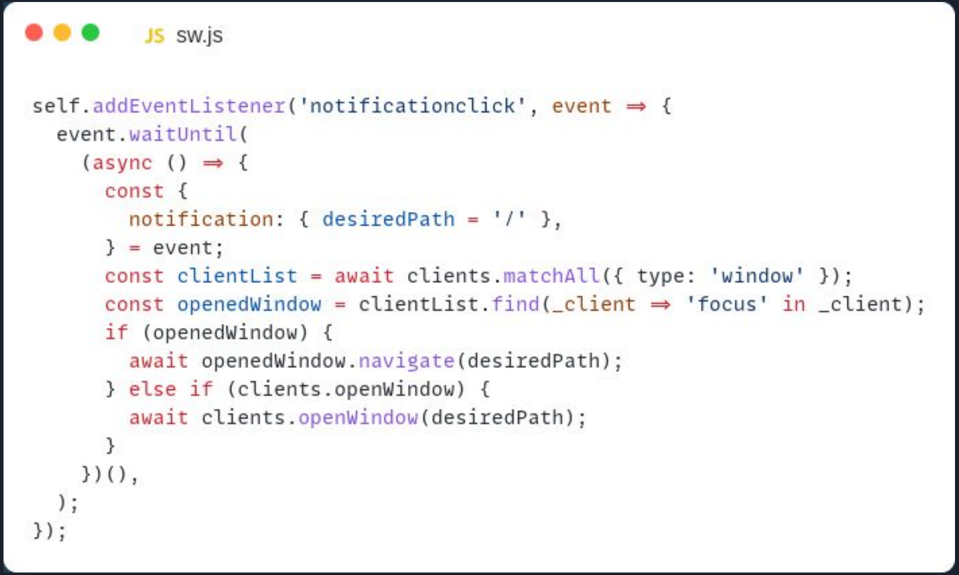
Show Notification



sw.js

```javascript
self.addEventListener('push', event => {
  event.waitUntil(
    (async () => {
      const { data } = event;
      const { title, body, image } = data.json();
      await self.registration.showNotification(title, {
        ...(image && { image }),
        ...(body && { body }),
      });
    })(),
  );
});
```

Handle Notification Click



sw.js

```javascript
self.addEventListener('notificationclick', event => {
  event.waitUntil(
    (async () => {
      const {
        notification: { desiredPath = '/' },
      } = event;
      const clientList = await clients.matchAll({ type: 'window' });
      const openedWindow = clientList.find(_client => 'focus' in _client);
      if (openedWindow) {
        await openedWindow.navigate(desiredPath);
      } else if (clients.openWindow) {
        await clients.openWindow(desiredPath);
      }
    })(),
  );
});
```

Thank You

Sources
https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
https://web.dev/progressive-web-apps/
https://whatpwacando.today/