

**TAKING OUR  
JAVASCRIPT SDK  
OFFLINE**

**CONTENT WARNING**

FINDAWAY.



A N Y P L A Y



24symbols



EBSCO

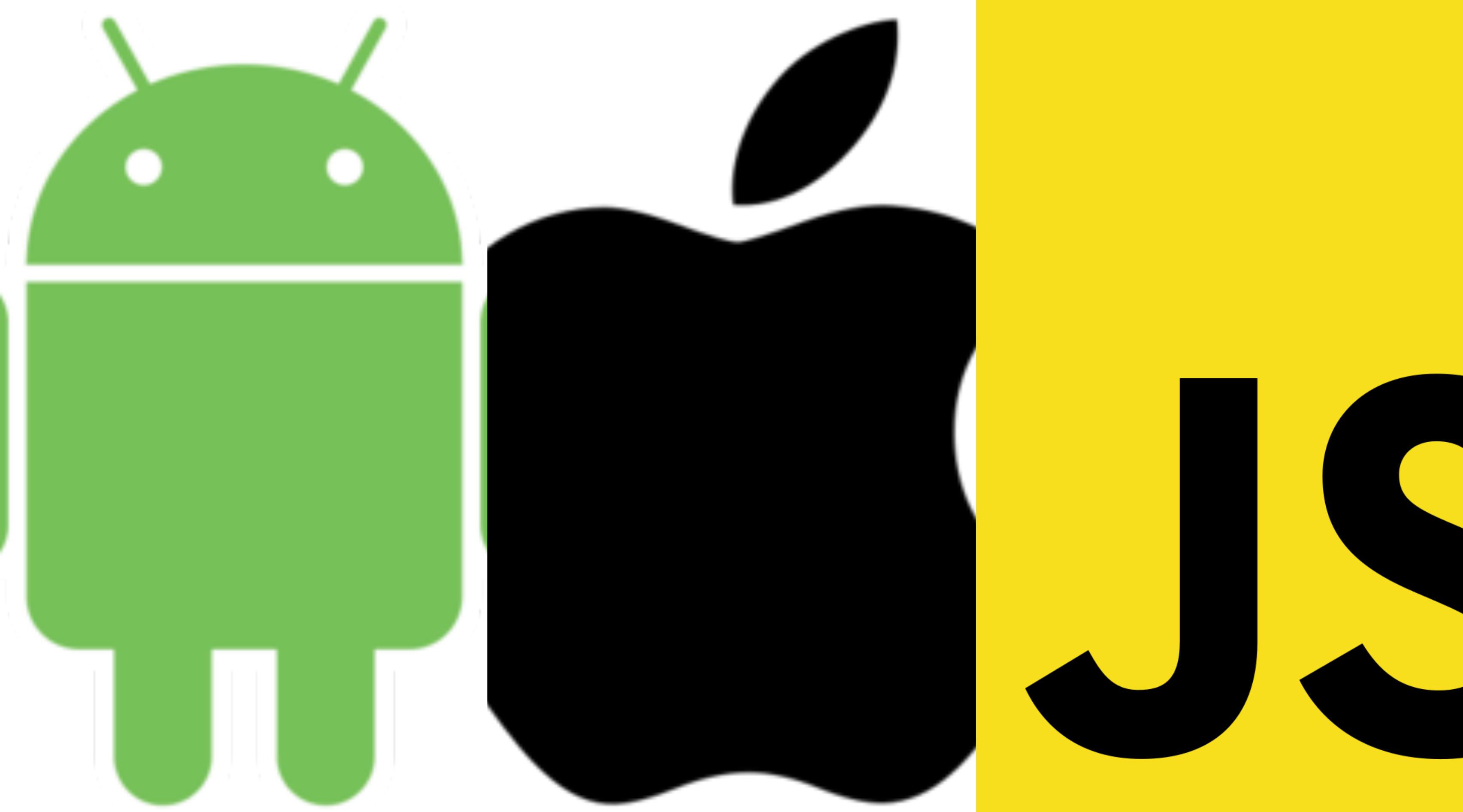


ODILO





AUDIO**ENGINE**



▼ Sort By

BOOKLAB



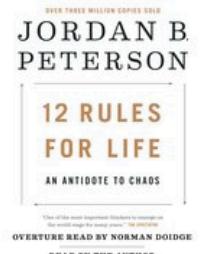
Downloaded

All Books



#AskGaryVee

by Gary Vaynerchuk



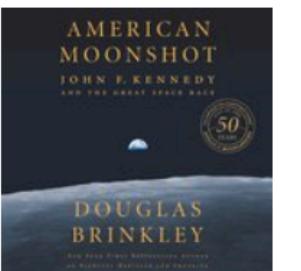
12 Rules For Life

by Jordan B. Peterson, Norman Doidge, MD, Ethan Van Sciver



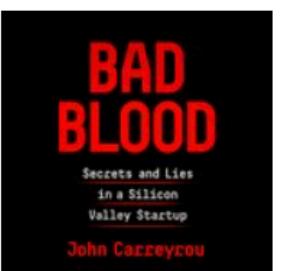
Abaddon's Gate

by James S.A. Corey



American Moonshot

by Douglas Brinkley



Bad Blood

by John Carreyrou



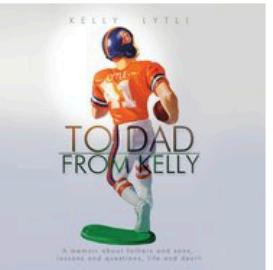
▼ Sort By

Authors Direct



Downloaded

All Books

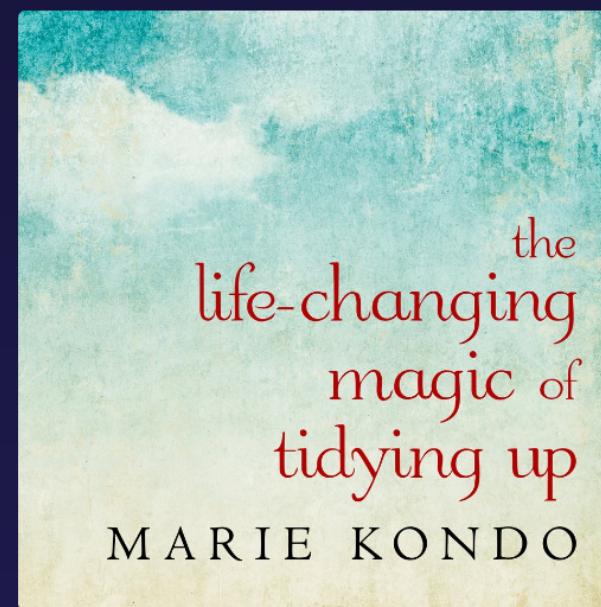


To Dad, From Kelly

by Kelly Lytle







# *The Life-Changing Magic of Tidying Up*

Written by Marie Kondo | Narrated by Emily Woo Zeller



00:00:00

Introduction

00:09:49



- 1.0x +



AMERICAN



@MRKTRS



**challenge accepted**





**MDN web docs**  
moz://a

Jeremy Keith

# GOING OFFLINE

FOREWORD BY Aaron Gustafson

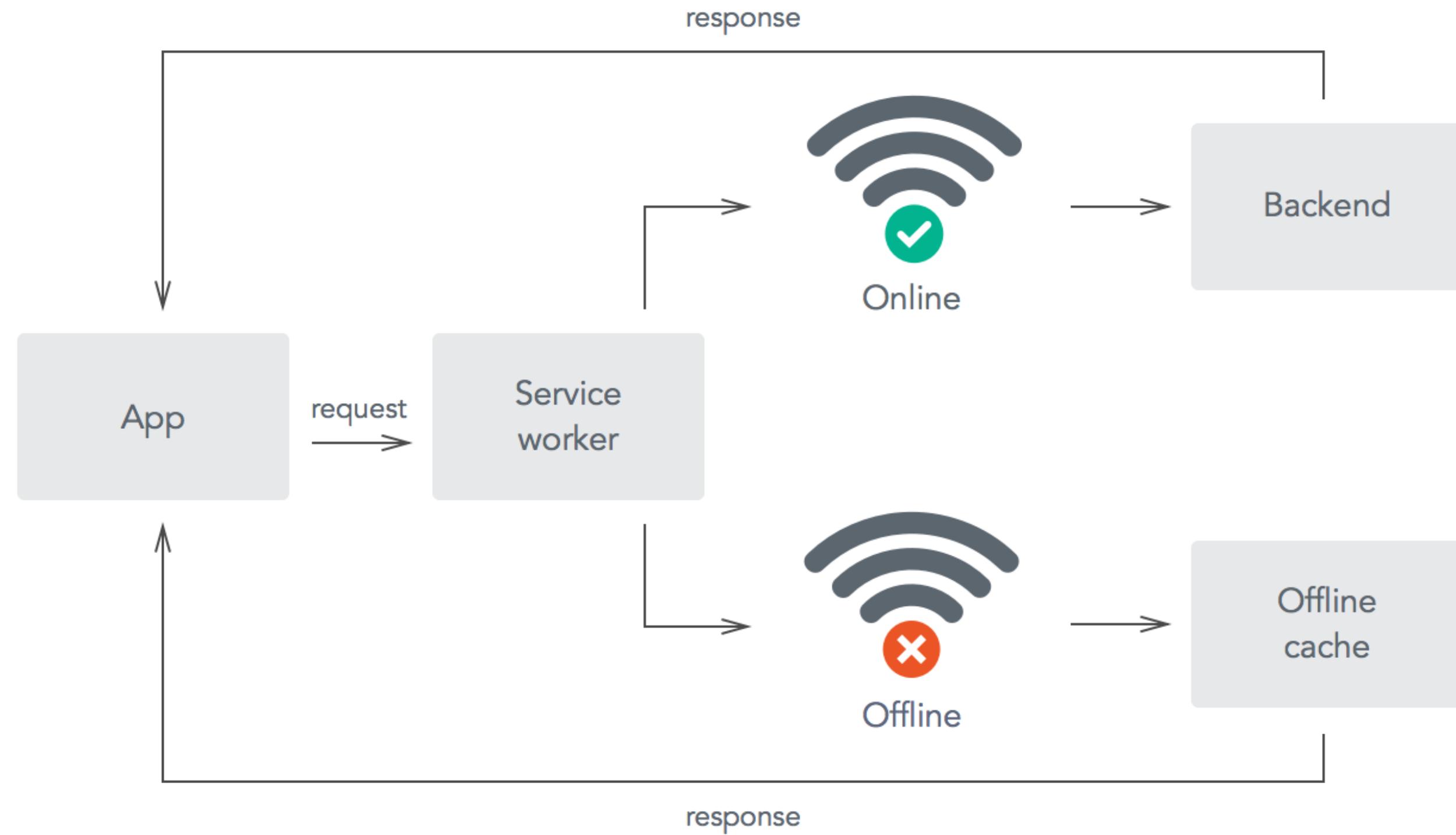
- Local Storage
- Cookies
- Session Storage
- IndexDB
- ServiceWorkers
- Cassette tape and Walkman

- Local Storage
- Cookies
- Session Storage
- IndexDB
- ServiceWorkers
- Cassette tape and Walkman



- Service Worker
- Client
- SDK
- caches
- cache

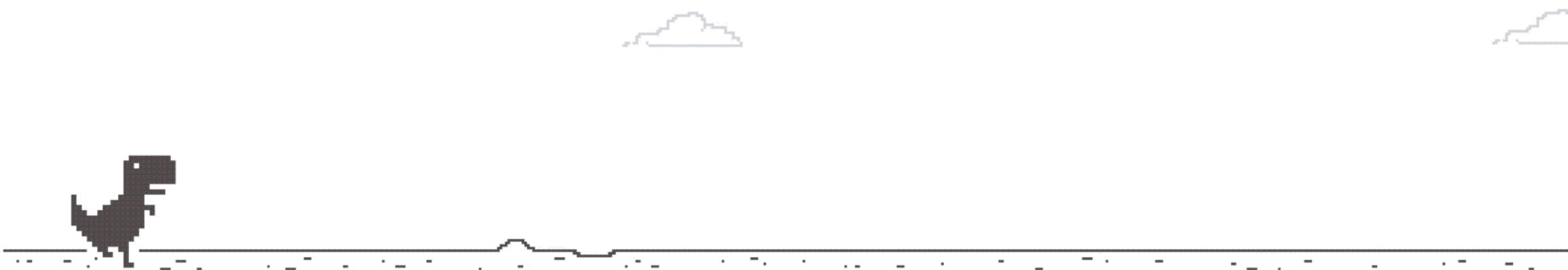




- Install
- Activate
- Message

- `caches.open('name')`
- `cache.put(url, response)`
- `cache.addAll([url,url,url])`

HI 00070 00007



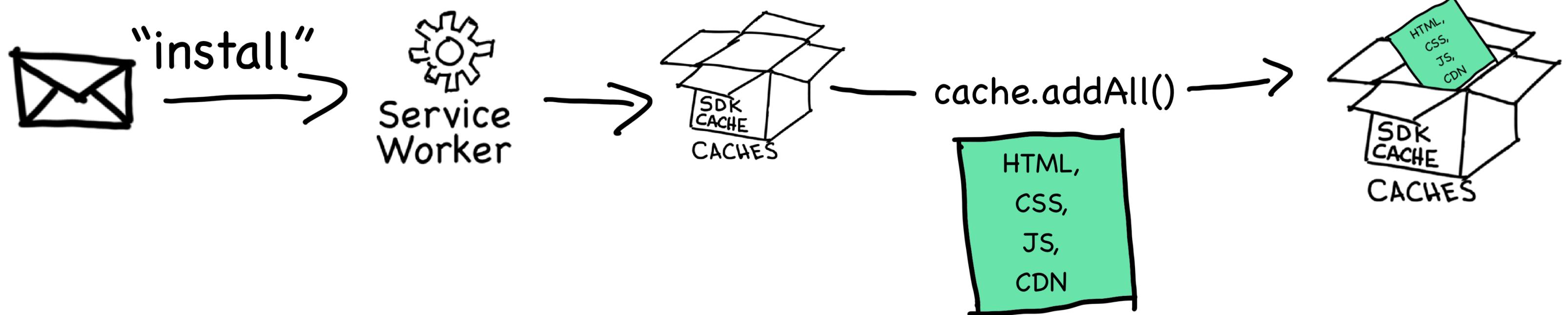
## No internet

Try:

- Checking the network cables, modem, and router
- Reconnecting to Wi-Fi

ERR\_INTERNET\_DISCONNECTED

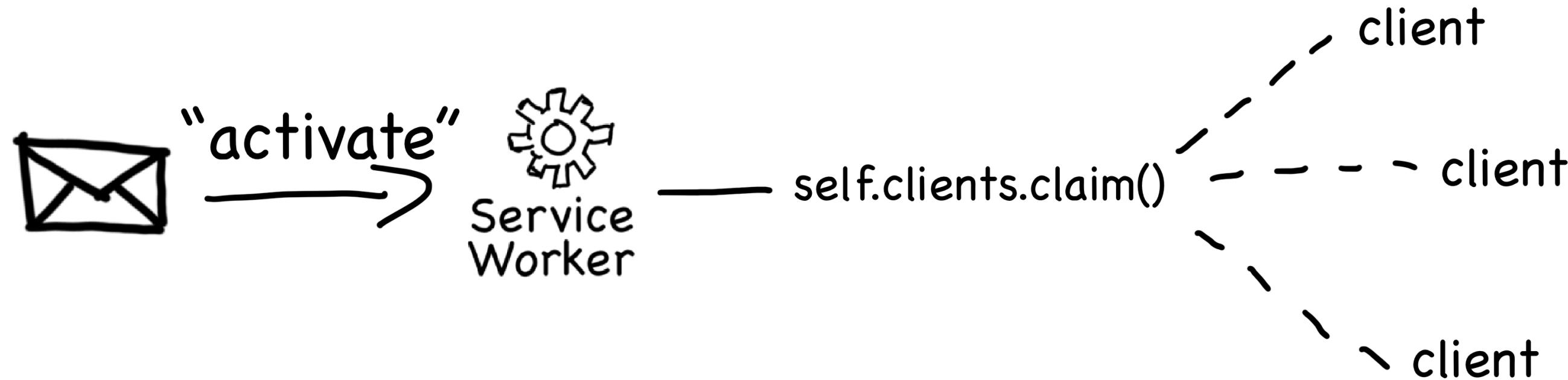
# Caching Application Assets



```
var urlsToCache = [
  '/scripts.js',
  '/service-worker.js',
  '/sdk.html',
  '/marx.css',
  '/findaway-sdk.2.3.0-alpha20190319.modern.js'
];

self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('my-site-cache-v1')
      .then(cache) => cache.addAll(urlsToCache))
  );
});
```

# Claim ServiceWorker Clients



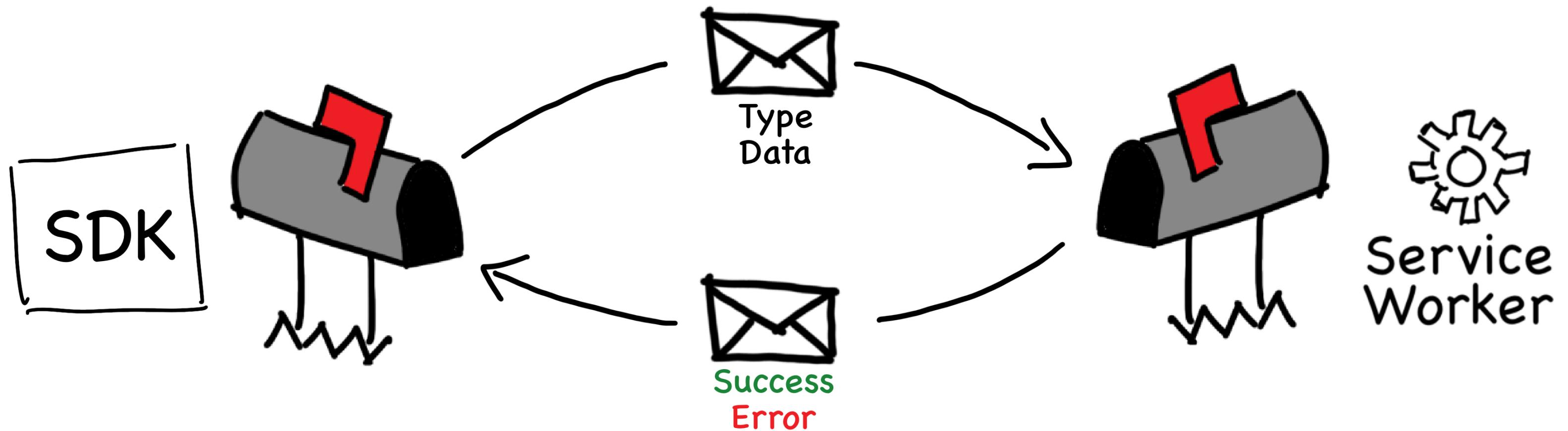
```
self.addEventListener('activate', event => {  
  self.clients.claim();  
});
```

```
if ('serviceWorker' in navigator) {  
  navigator.serviceWorker.register('service-worker.js');  
}
```



MOVIECLIPS.COM

MAKE GIFS AT [GIFSOUP.COM](http://GIFSOUP.COM)



```
self.addEventListener('message', async (event) => {
  let client = event.ports[0];

  try {
    // do something with data in message
    client.postMessage(/*Send something back to the client*/);
  } catch (e) {
    client.postMessage(/*Send an Error back to the client*/);
  }
});
```

```
SDK.sendMessage = (type, data) => (
  new Promise((resolve, reject) => {
    let messageChannel = new window.MessageChannel();
    messageChannel.port1.onmessage = (event) => {
      // Do something with event infomration from our ServiceWorker
    };
    navigator.serviceWorker.controller.postMessage(
      {type, data},
      [messageChannel.port2]
    );
  });
);
```

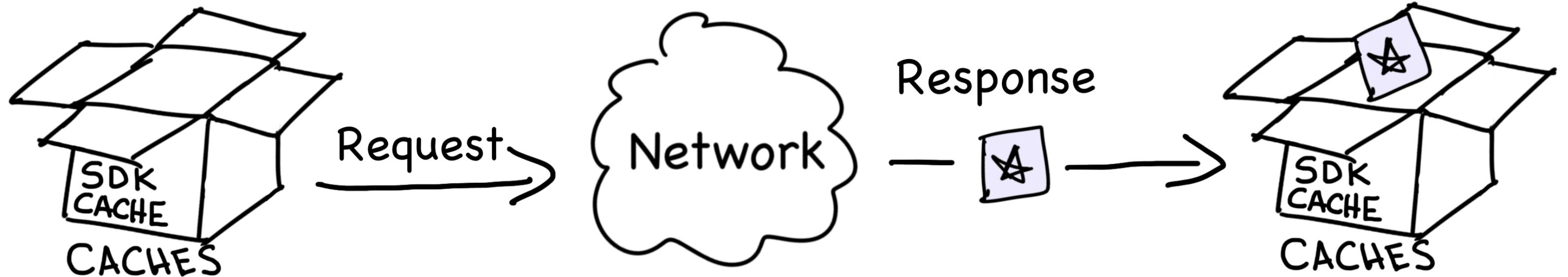
# DEMO



**YEAH, I CAN TAKE A MESSAGE.**

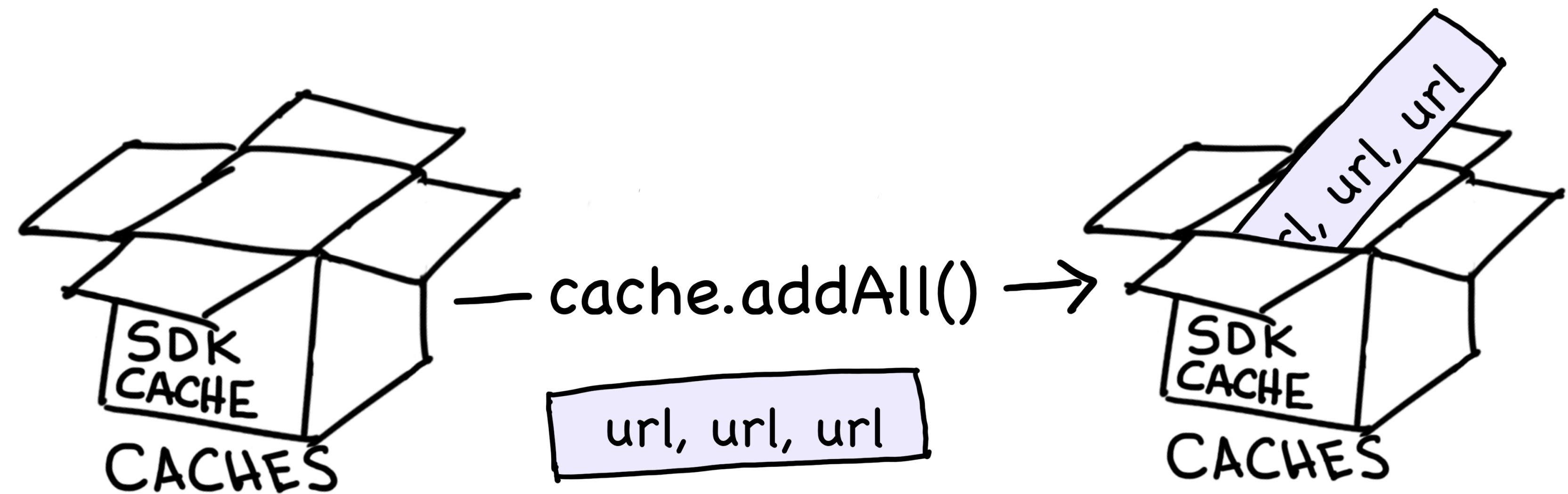


# Cache Requests



```
const cacheURL = ({url, headers}) => {
  return caches.open(`SDK-cache`)
    .then(cache => {
      fetch(url, {headers: new Headers(headers)})
        .then(resp => cache.put(url, resp))
    ));
};
```

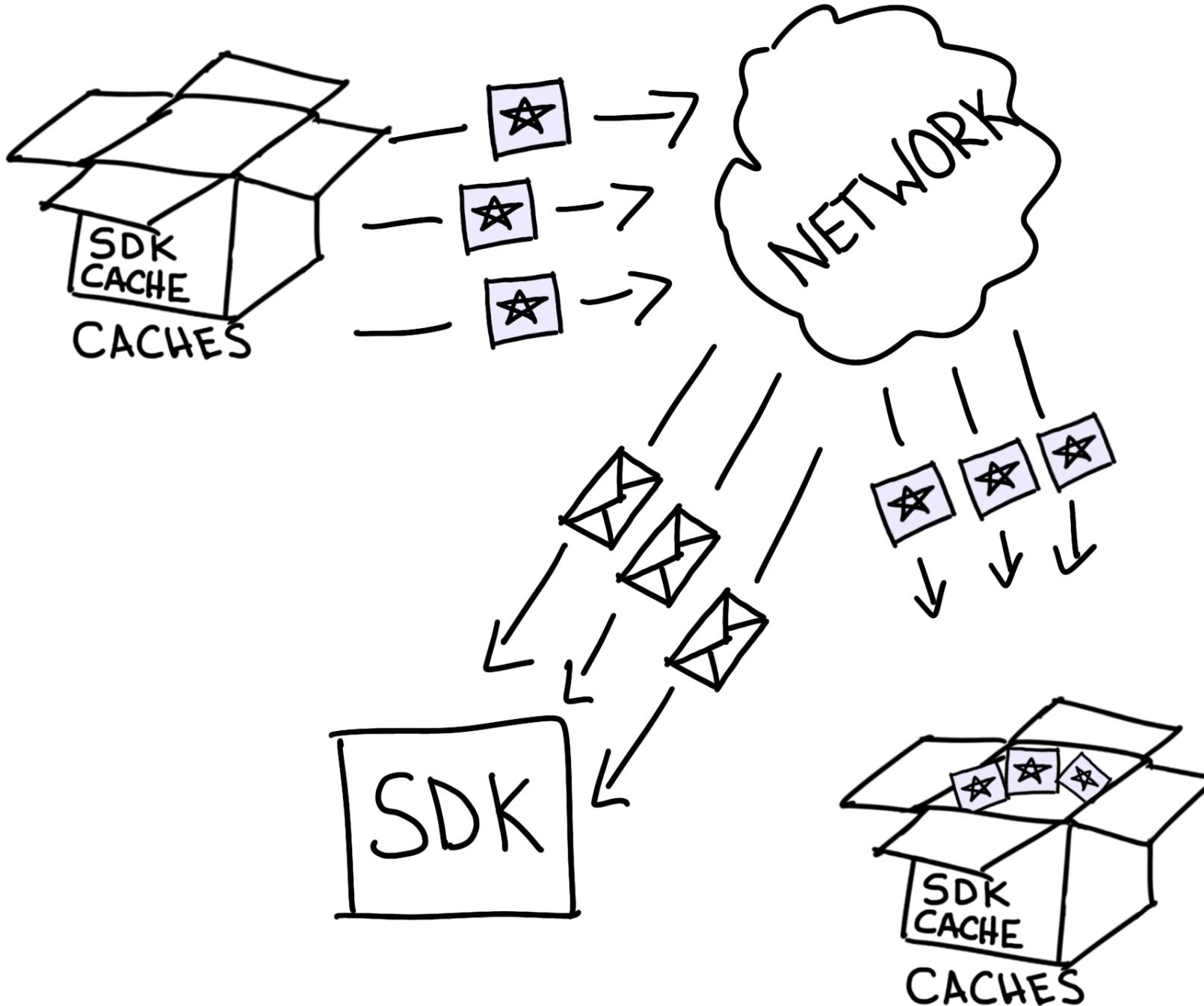
# Cache Audio



```
const cacheAudio = ({id, urls}) => {
  return caches.open(`book-${id}`)
    .then((cache) => cache.addAll(urls))
};
```

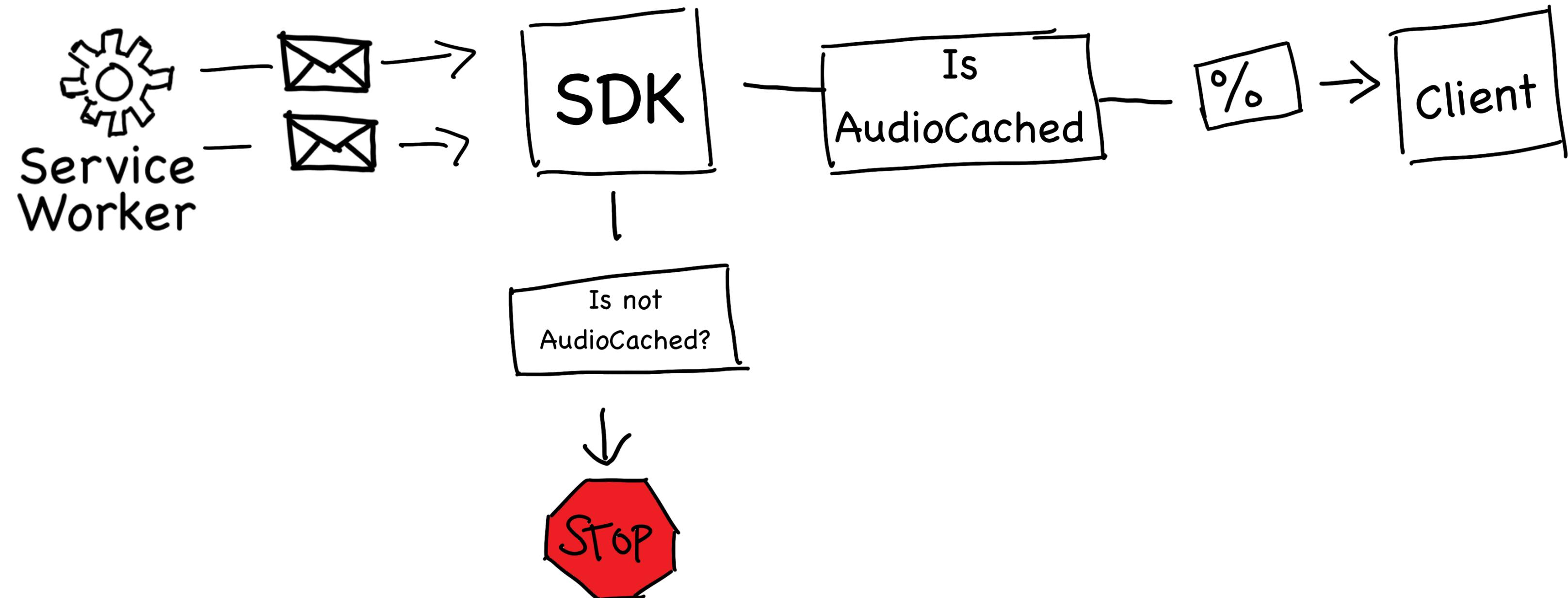


# Cache audio with status



```
urls.forEach(url => {
  fetch(url)
    .then(resp => {
      messageClient({
        type: 'AudioCached' ,
        id: id,
        url
      });
      return cache.put(url, resp);
    })
    .then(() => resolve(url))
    .catch(err => reject(err));
});
```

# Audio file status events



```
navigator.serviceWorker.addEventListener('message', (event) => {
  if (!event.data || event.data.type !== 'AudioCached') return;
  completeUrls.push(event.data.url);
  let percent = completeUrls.length / this.playlist.length;
  sdk.trigger(`cache:${this.id}`, percent);
});
```

# DEMO



```
[  
 {  
   "url": "https://api.url.to/audio/0_0.mp3",  
   "part_number": 0,  
   "chapter_number": 0  
 }  
]
```



**SONGLAND**  
PREMIERES  
NEXT TUESDAY 10/9c



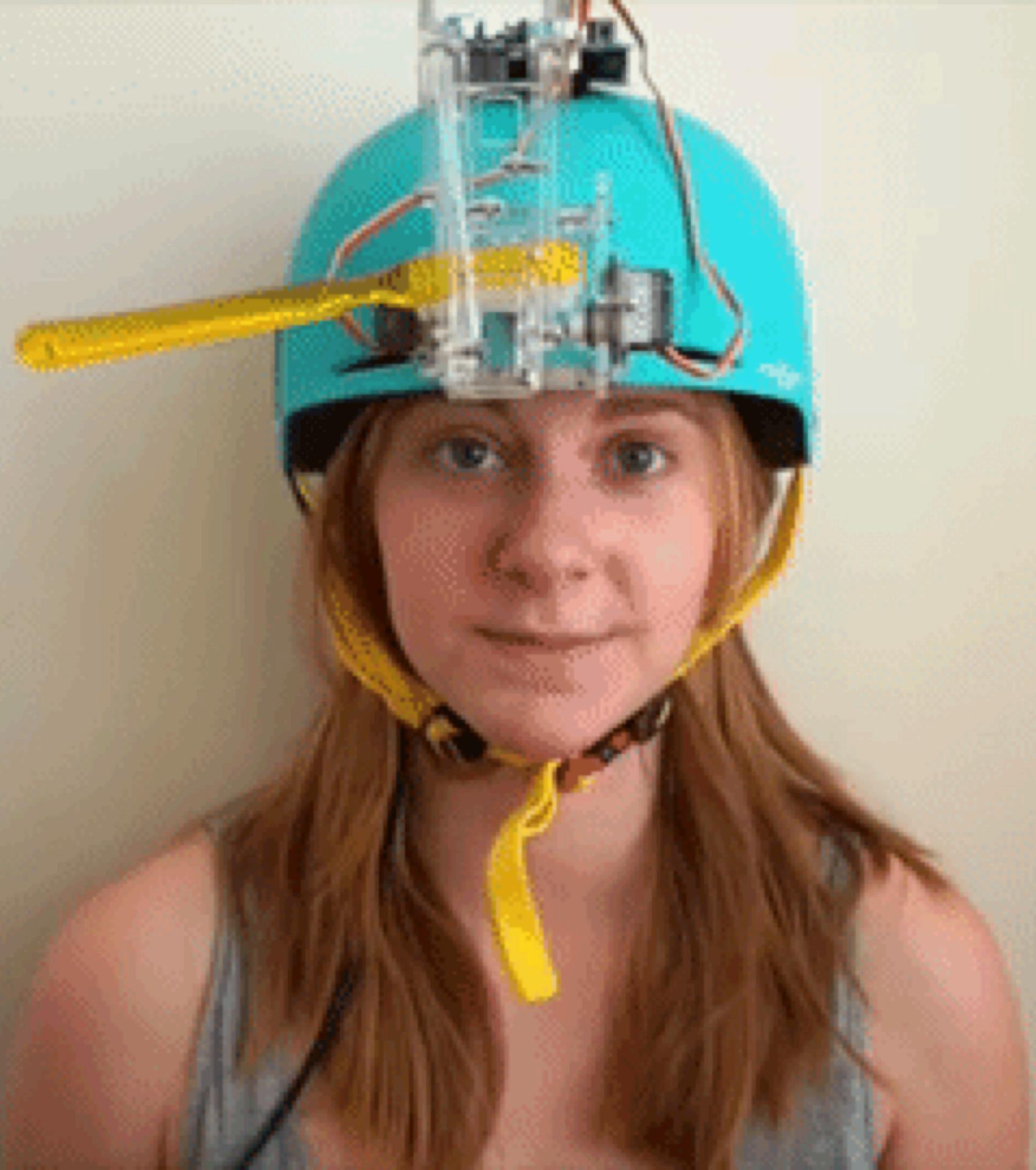




# Cache POST Requests



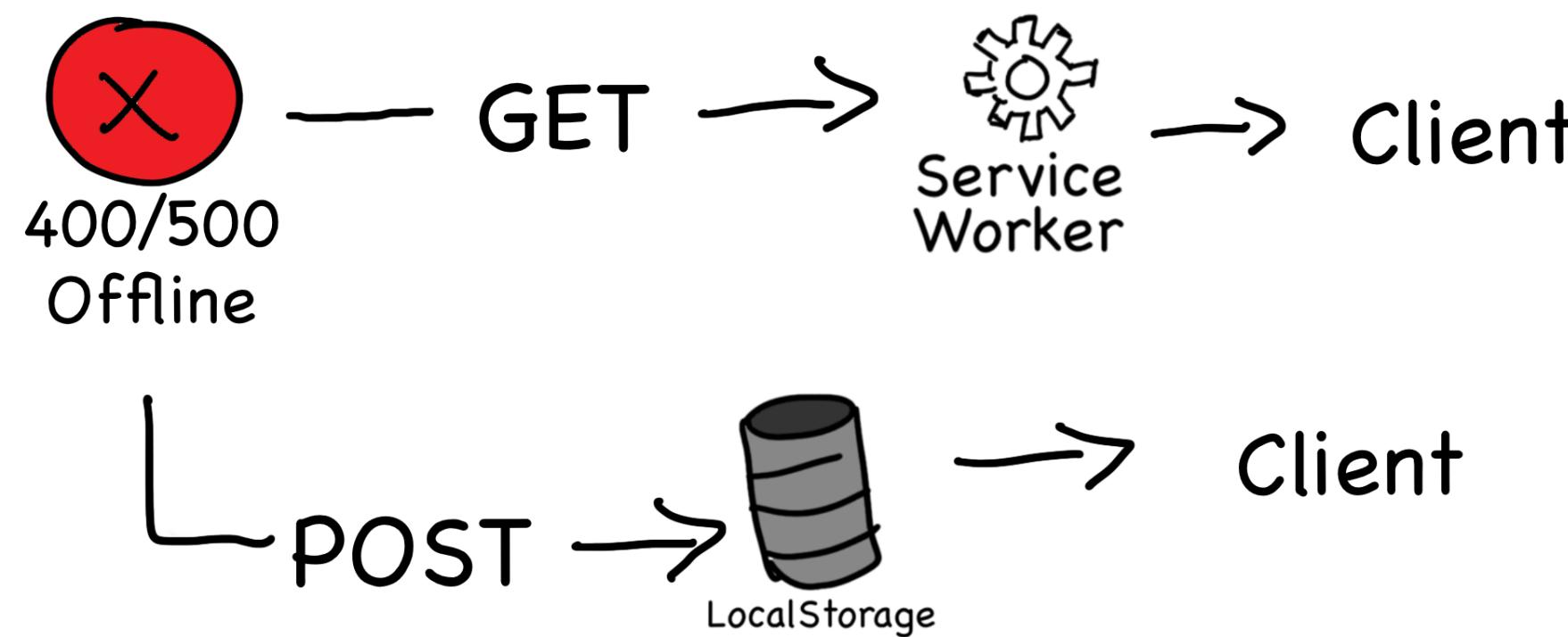
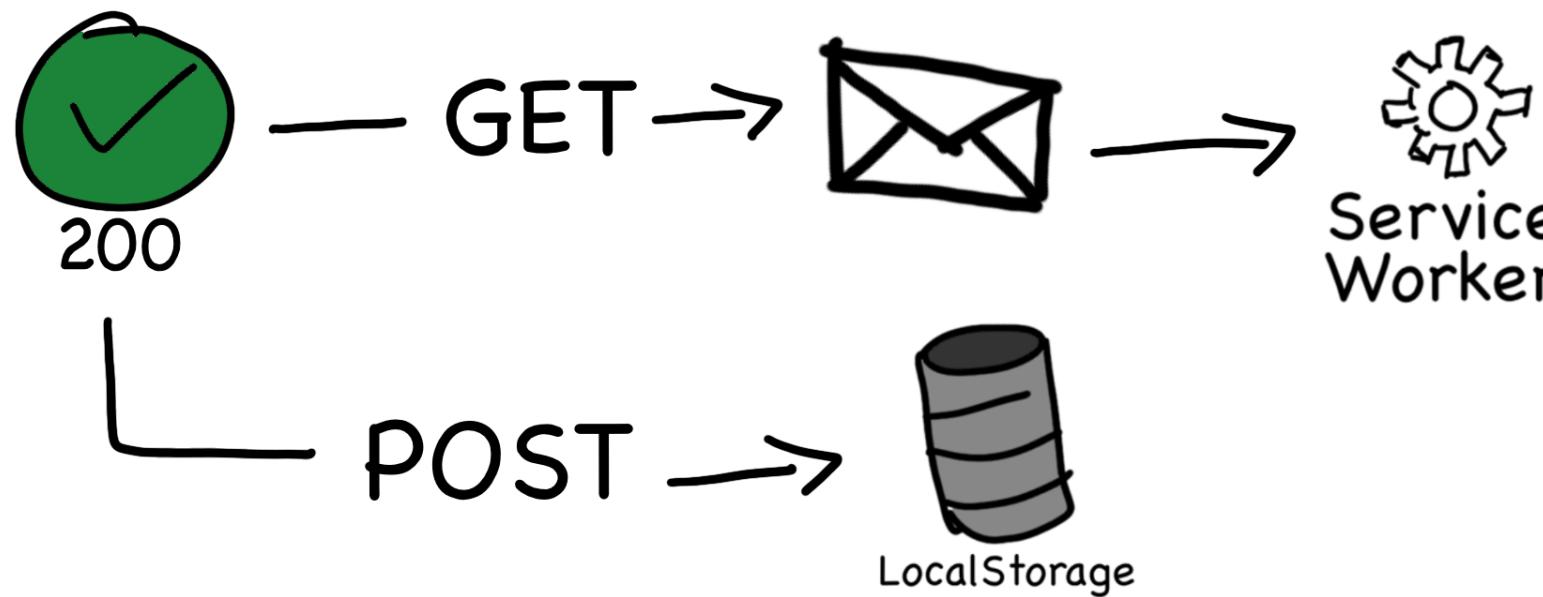
```
if (method === 'POST') {
    // Cache with LocalStorage
    window.localStorage.setItem(url + '-' + method, JSON.stringify(response));
}
```





```
this.enable_cache = options.enable_cache && !!navigator.serviceWorker;
```

# Automatically cache requests



```
const success = ((xhr.status >= 200 && xhr.status < 300) || xhr.status === 304);
```

```
if (success && enable_cache && method !== 'POST') {  
  sendMessage('cacheAPI', {  
    url: url,  
    headers: cacheHeaders  
  });  
}  
}
```

```
if (success && enable_cache && method === 'POST') {
  window.localStorage.setItem(url + '-' + method, JSON.stringify(response));
}
```

 00320	api.findawayworld.com/v4/accounts/application_test_account/audiobooks	GET	200 OK	fetch	<a href="#">service-worker.js:65</a>	Script
---	---	-----	-----------	-------	--------------------------------------	--------

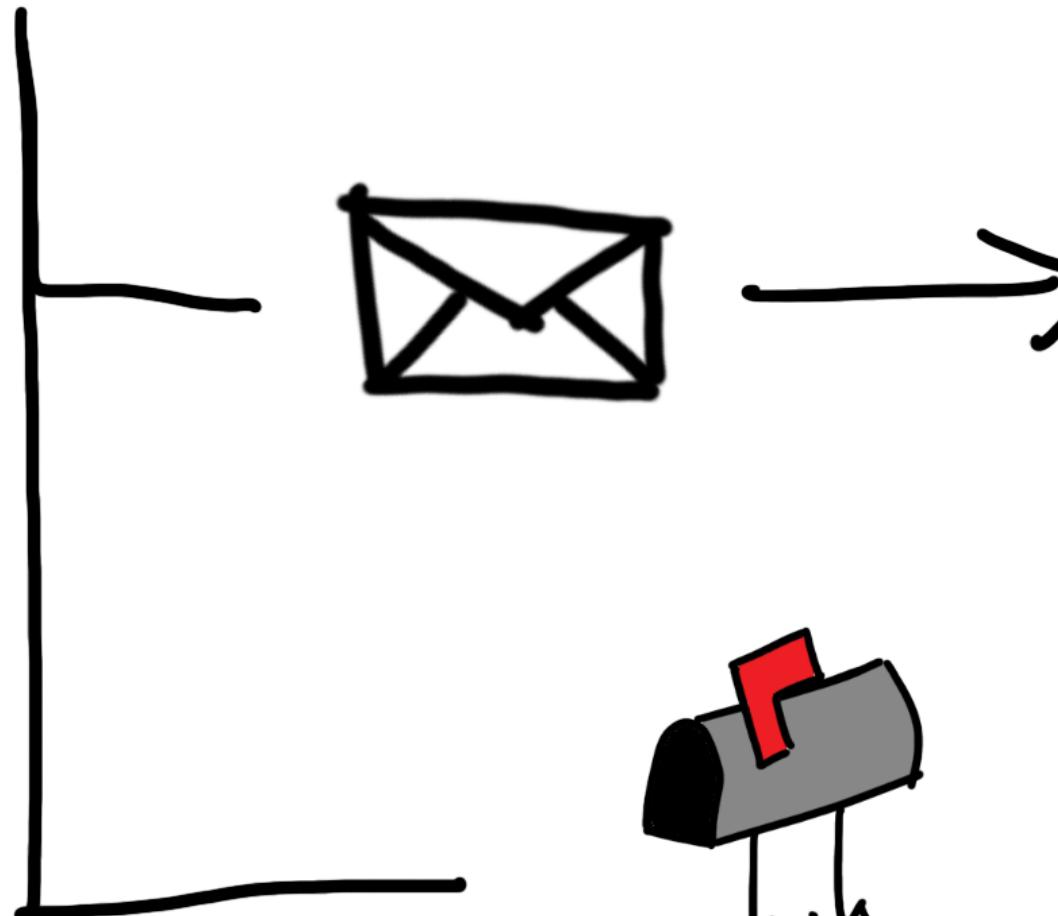
```
var lsCache = window.localStorage.getItem(url + '-' + method);
if (!success && method === 'POST' && enable_cache && lsCache) {
    resolve(JSON.parse(lsCache));
    return false;
}
```



- Cache Audio on Demand
- Check for cached book
- Clear cached book

# SDK.cache()

Cache  
Not  
Enabled



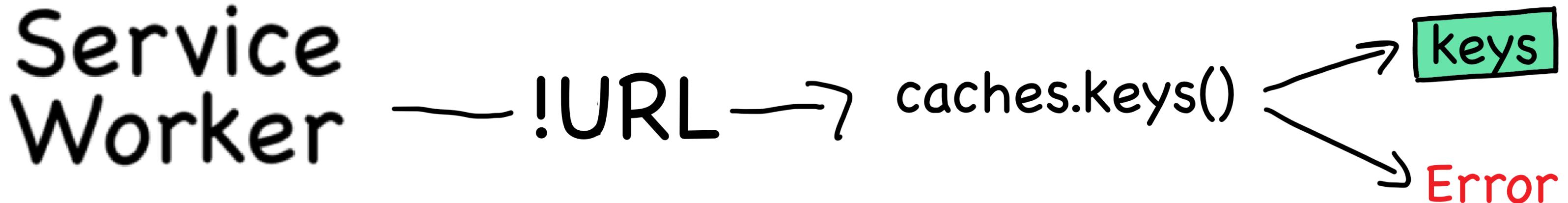
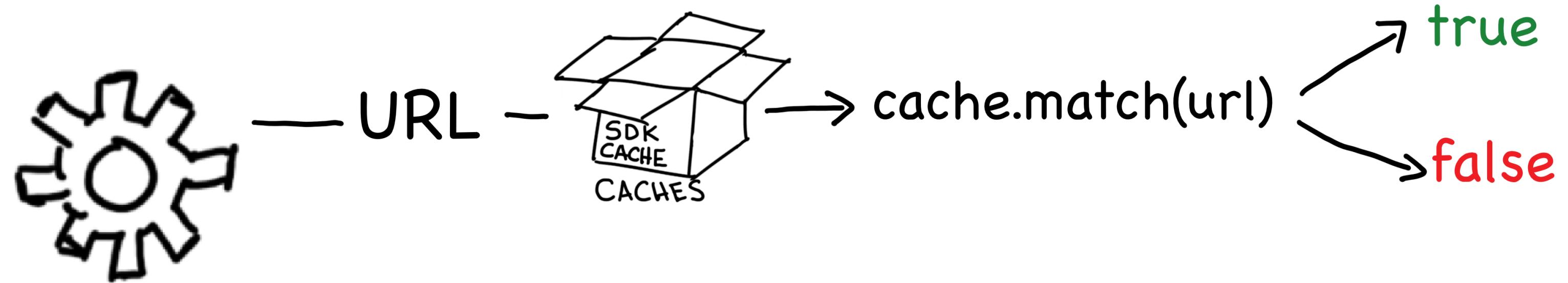
Service  
Worker

```
cache () {
  return new Promise((resolve, reject) => {
    if (!enable_cache) { return reject(new Error('Caching Not Enabled.')); }

    let cachedAudio = [];
    navigator.serviceWorker.addEventListener('message', (event) => {
      if (!event.data || event.data.type !== 'AudioCached') return;
      cachedAudio.push(event.data.url);
      let percent = cachedAudio.length / this.playlist.length;
      sdk.trigger(`cache:${this.id}`, percent);
    });
  });

  return sendMessage('cacheAudio', {
    id: this.id,
    urls: this.playlist.map(p => p.url)
  })
  .then(resp => resolve(resp))
  .catch(err => reject(err));
});
```

# Check Caches



```
const checkCaches = (data) => {
  return new Promise((resolve, reject) => {
    if (data.url) {
      return caches.open('demo-cache')
        .then(cache => cache.match(data.url))
        .then(() => resolve(true))
        .catch(err => reject(err));
    }
    return caches.keys()
      .then(keys => resolve(keys))
      .catch(err => reject(err));
  });
};
```

```
sendMessage('checkCaches')
  .then(keys => {
    isAudioCached = keys.includes(`book-${this.id}`);
  });
}
```

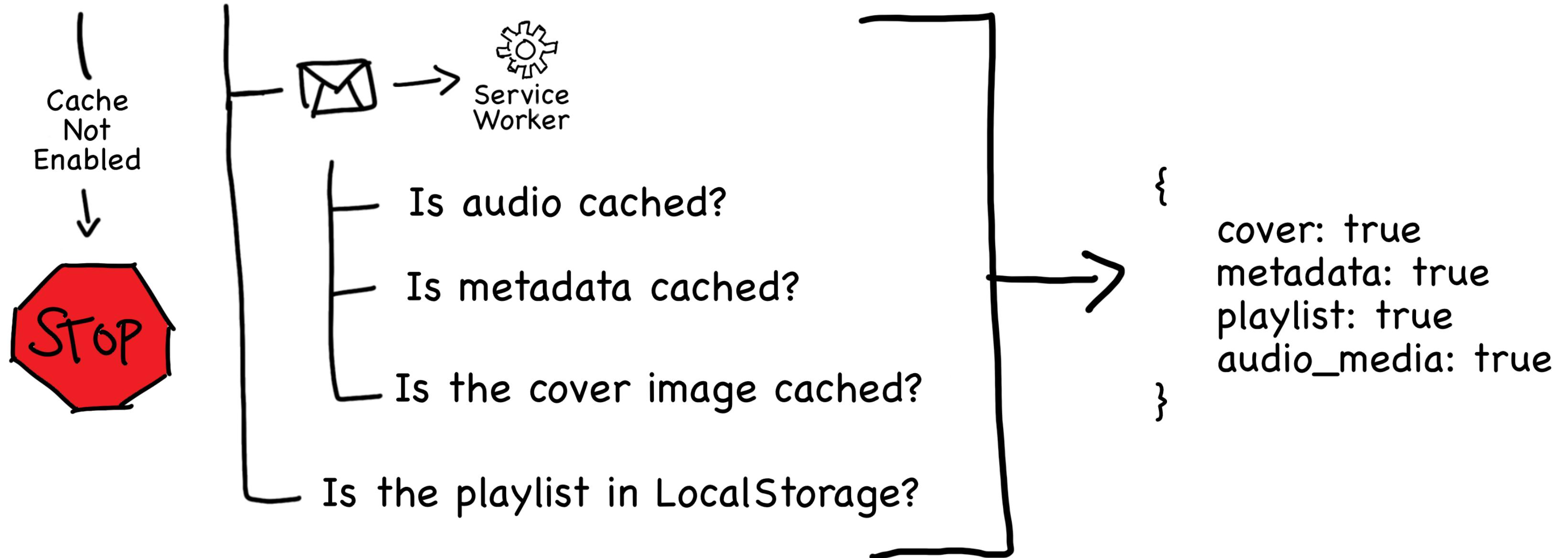
```
let url = urlFor('audiobooks', {id: this.id, account_id: this.account_id});  
sendMessage('checkCaches', {url})  
  .then(cached => {  
    isMetadataCached = cached;  
  });
```

```
sendMessage('checkCaches', {url: this.cover_url})
  .then(cached => {
    isCoverCached = cached;
  });

```

```
let playlistUrl = urlFor('playlist', {id: this.id, account_id: this.account_id});  
let isPlaylistCached = !!(window.localStorage.getItem(` ${playlistUrl}-POST`));
```

# SDK.checkCache()



```
checkCache () {
    return new Promise((resolve, reject) => {
        if (!enable_cache) { return reject(new Error('Caching Not Enabled.')); }
        let isAudioCached = false;
        let isMetadataCached = false;

        sendMessage('checkCaches')
            .then(keys => {
                if (keys.error) throw new Error(keys.error);
                isAudioCached = keys.includes(`book-${this.id}`);

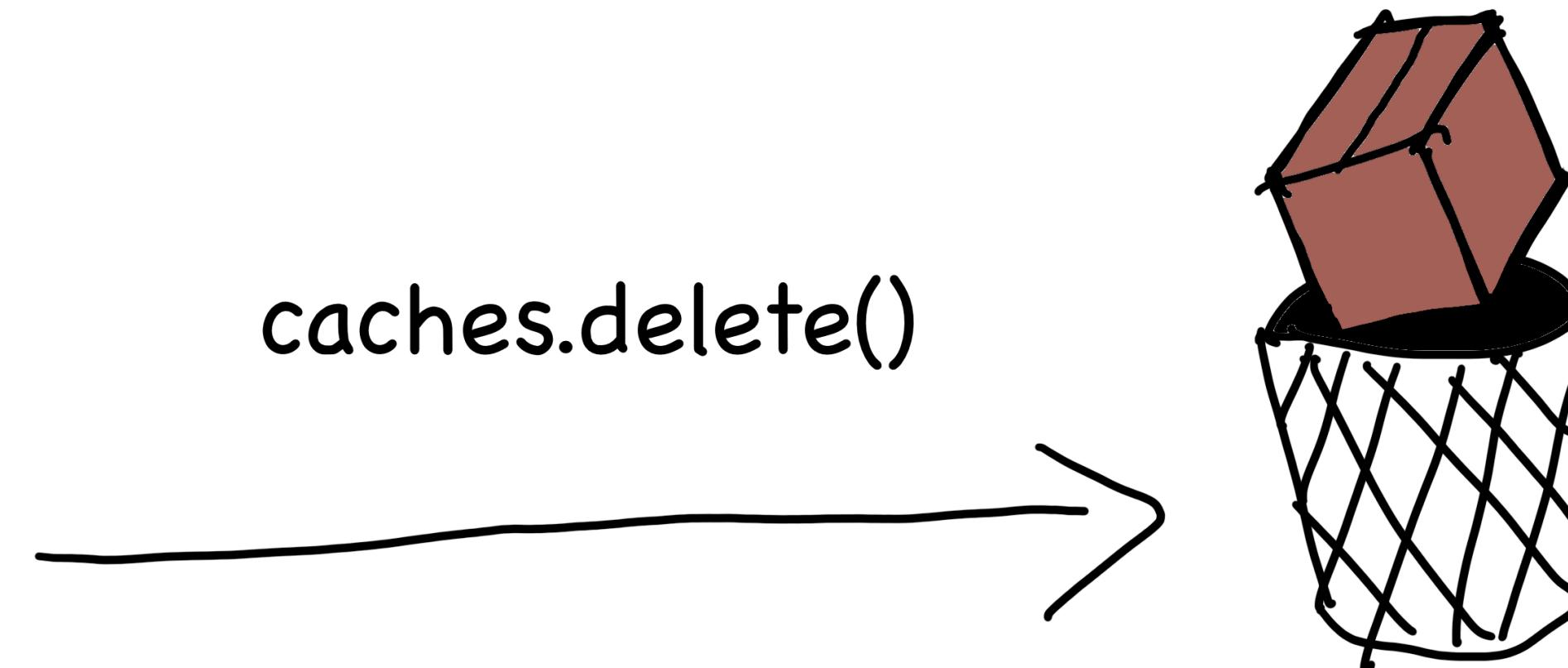
                let url = urlFor('audiobooks', {id: this.id, account_id: this.account_id});
                return sendMessage('checkCaches', {url});
            })
            .then(cached) => {
                isMetadataCached = cached;
                return sendMessage('checkCaches', {url: this.cover_url});
            })
            .then(isCoverCached) => {
                let playlistUrl = urlFor('playlist', {id: this.id, account_id: this.account_id});
                let isPlaylistCached = !(window.localStorage.getItem(`${playlistUrl}-POST`));

                resolve({
                    cover: isCoverCached,
                    metadata: isMetadataCached,
                    playlist: isPlaylistCached,
                    audio_media: isAudioCached
                });
            })
            .catch(err => reject(err));
    });
}
```

# Clear Caches

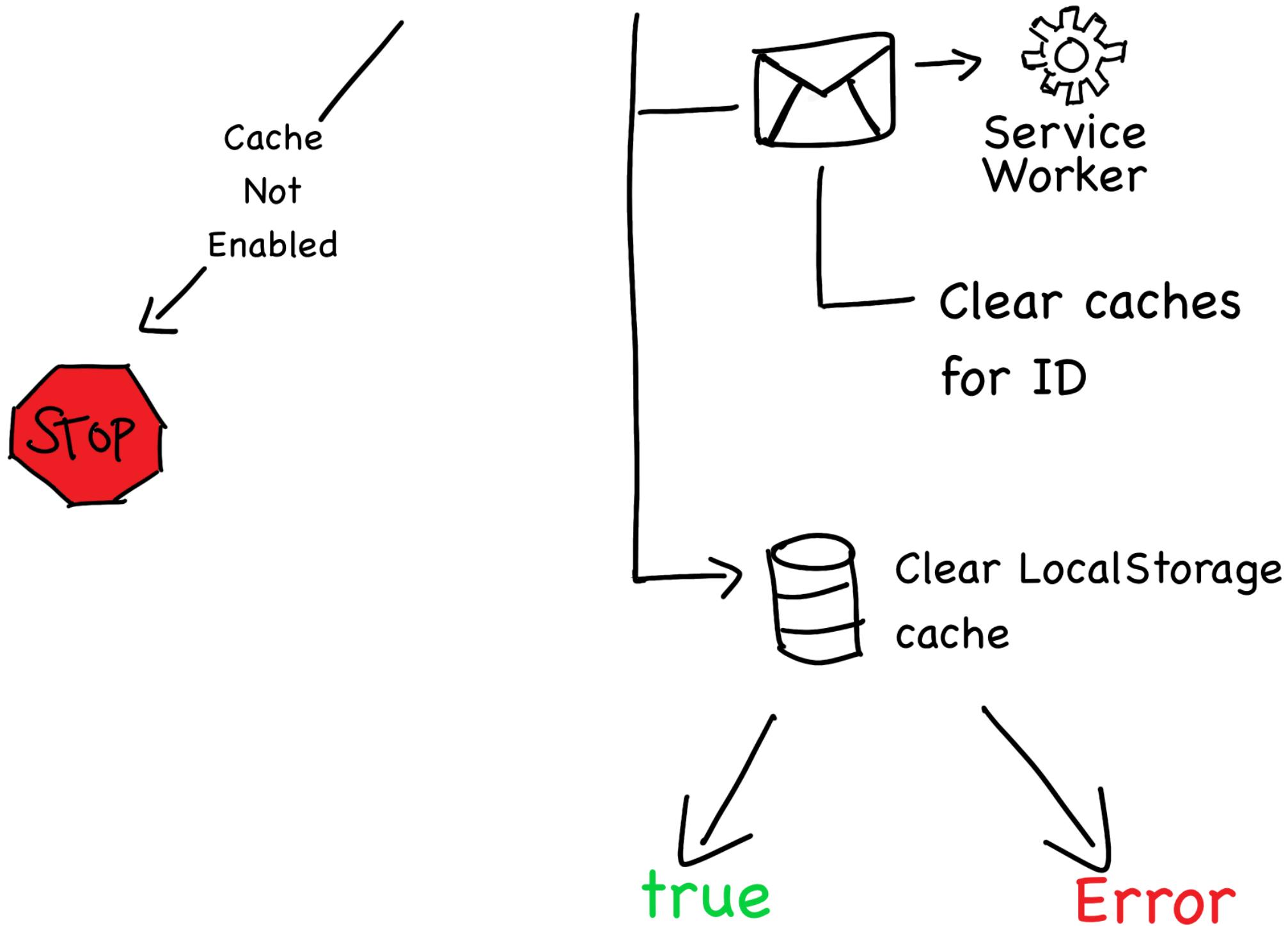
Service  
Worker

`caches.delete()`



```
const clearCaches = ({id}) => {
  return new Promise((resolve, reject) => {
    return caches.delete(`book-${id}`)
      .then(() => resolve(true))
      .catch(err => reject(err));
  });
};
```

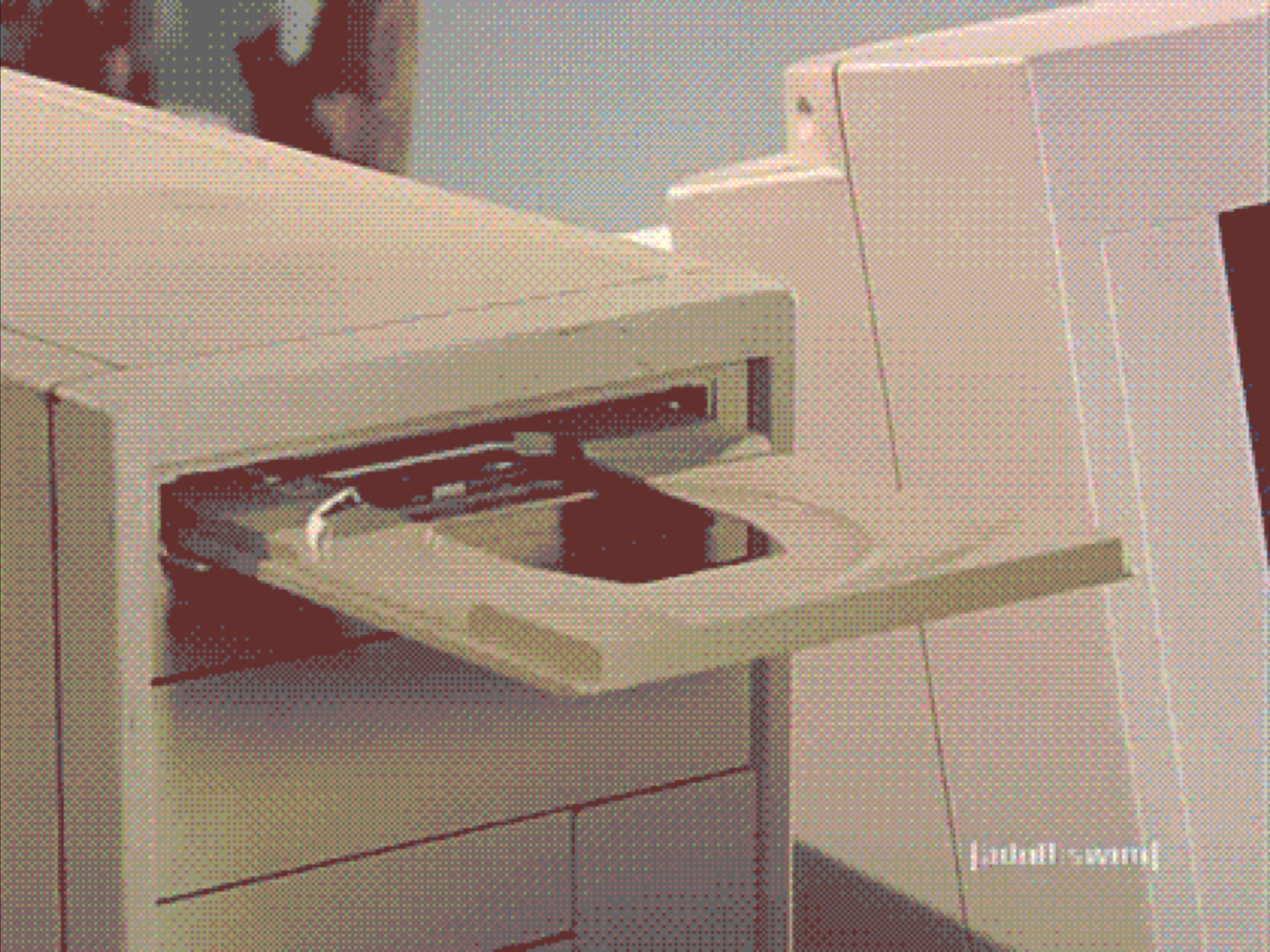
# SDK.clearCache()



```
clearCache () {
  return new Promise((resolve, reject) => {
    if (!enable_cache) { return reject(new Error('Caching Not Enabled.')); }

    return sendMessage('clearCaches', {id: this.id})
      .then(() => {
        let playlistUrl = urlFor('playlist', {id: this.id, account_id: this.account_id});
        window.localStorage.removeItem(` ${playlistUrl}-POST`);
        resolve(true);
      })
      .catch(err => reject(err));
  });
}
```

# DEMO



```
importScripts('ae-service-worker.js');
```

HAPPY NEW YEAR!







# Service Workers

Usage

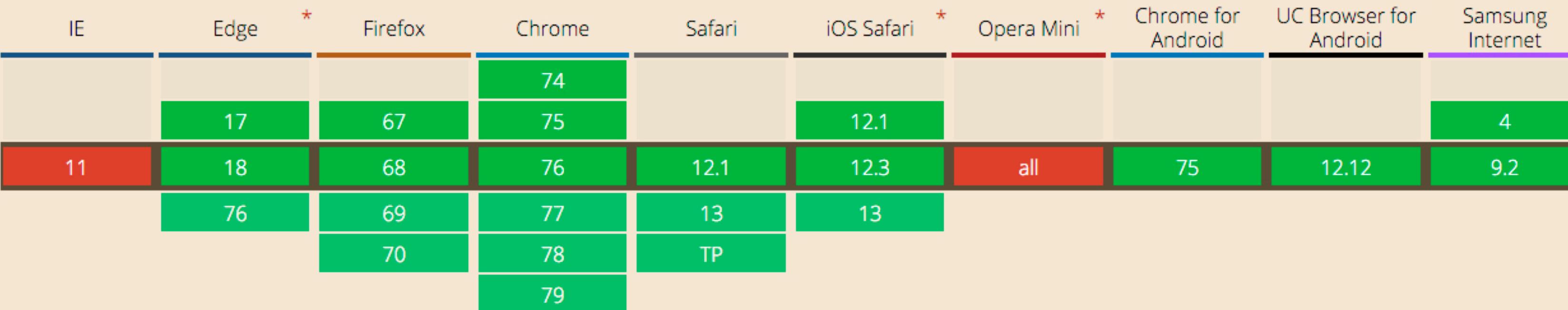
% of all users



91.54% + 0.39% = 91.93%

Method that enables applications to take advantage of persistent background processing, including hooks to enable bootstrapping of web applications while offline.

Current aligned Usage relative Date relative [Apply filters](#) Show all ?



Notes

Known issues (0)

Resources (8)

Feedback

Details on partial support can be found on [is ServiceWorker Ready?](#)

# THANKS

@camwardzala  
camwardzala.com