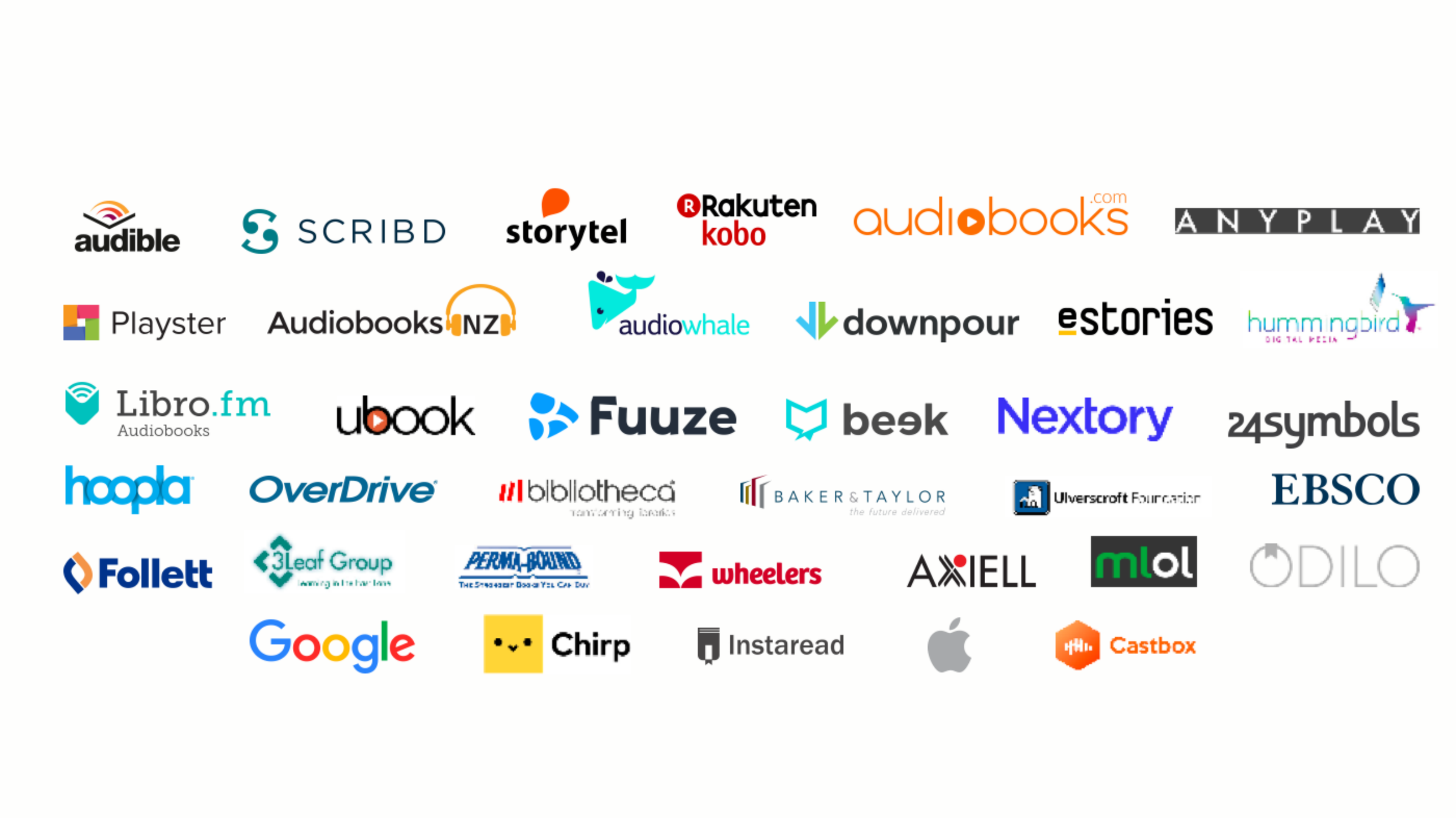
TAKING OUR JAVASCRIPT SDK OFFLINE

CONTENT WARNING

FINDAWAY


audible
SCRIBD
storytel
Rakuten kobo
audiobooks.com
ANYPLAY
Playster
Audiobooks NZ
audiowhale
downpour
estories
hummingbird
Libro.fm
Audiobooks
ubook
Fuuze
beek
Nextory
24symbols
hoopla
OverDrive
bibliotheca
BAKER & TAYLOR
the future delivered
Ulverscroft Foundation
EBSCO
Follett
3Leaf Group
PERMA-BOUND
wheelers
AXIELL
mlol
ODILO
Google
Chirp
Instaread
Castbox


AUDIOENGINE


JS

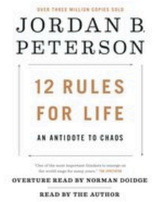
Verizon 3:33 PM

Sort By BOOKLAB

Downloaded | All Books

#AskGaryVee

by Gary Vaynerchuk

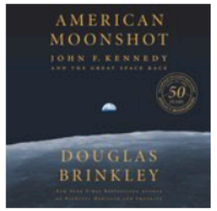


12 Rules For Life

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

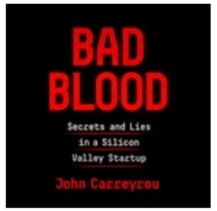
Abaddon's Gate

by James S.A. Corey



American Moonshot

by Douglas Brinkley



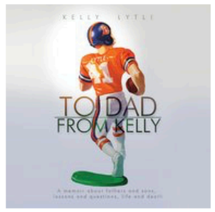
Bad Blood

by John Carreyrou

Verizon 3:34 PM

Sort By Authors Direct

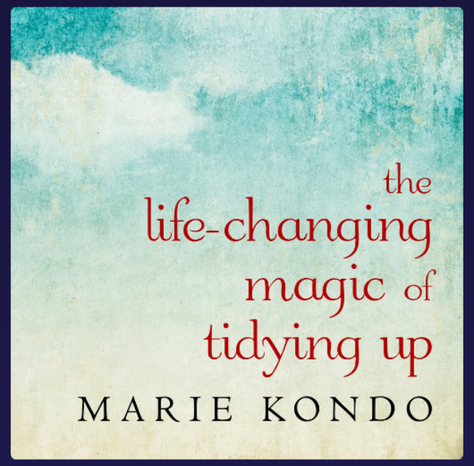
Downloaded | All Books



To Dad, From Kelly

by Kelly Lytle


JS



The Life-Changing Magic of Tidying Up

Written by Marie Kondo | Narrated by Emily Woo Zeller

00:00:00 Introduction 00:09:49

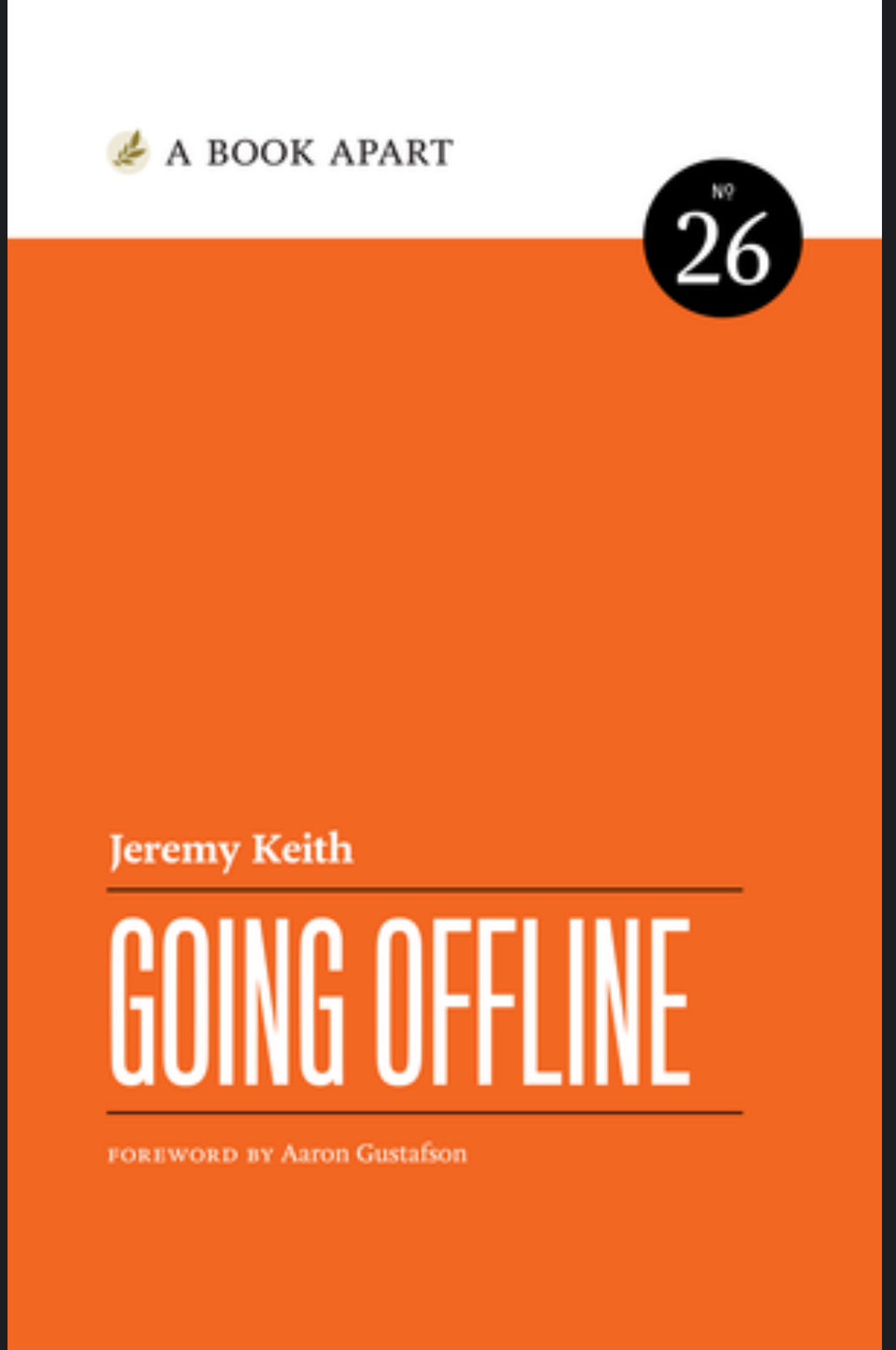
15 15

1.0x


@MRKTRS


LEONE
IGARS
challenge accepted


MDN web docs
moz://a


A BOOK APART
№ 26
Jeremy Keith
GOING OFFLINE
FOREWORD BY Aaron Gustafson

- Local Storage

- Local Storage
- Cookies

- Local Storage
- Cookies
- Session Storage

- Local Storage
- Cookies
- Session Storage
- IndexDB

- Local Storage
- Cookies
- Session Storage
- IndexDB
- ServiceWorkers

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

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


GOOD JOB

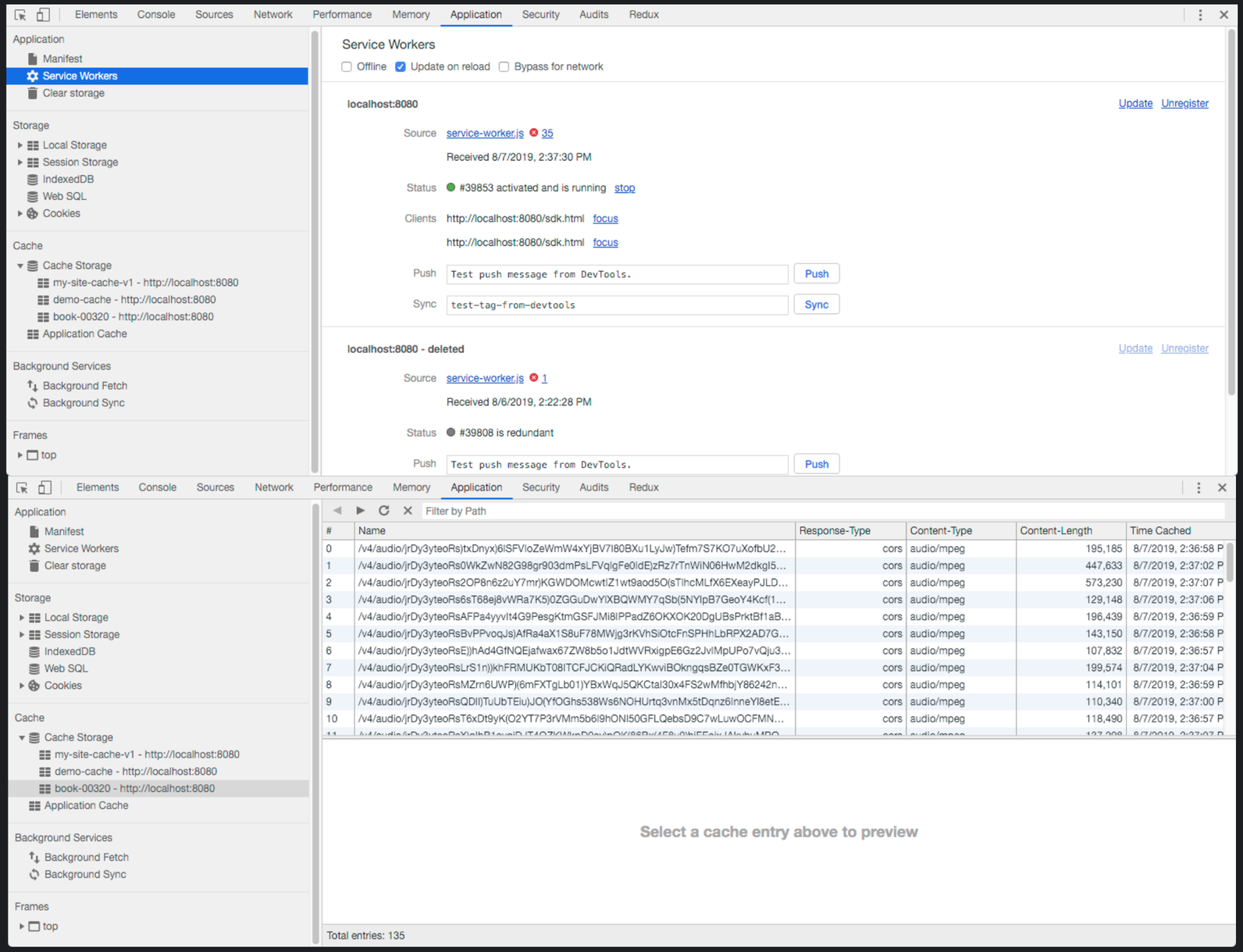
- Service Worker

- Service Worker
- Client

- Service Worker
- Client
- SDK

- Service Worker
- Client
- SDK
- caches

- Service Worker
- Client
- SDK
- caches
- cache


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
Redux
Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Cache
Cache Storage
my-site-cache-v1 - http://localhost:8080
demo-cache - http://localhost:8080
book-00320 - http://localhost:8080
Application Cache
Background Services
Background Fetch
Background Sync
Frames
top
Service Workers
Offline
Update on reload
Bypass for network
localhost:8080
Update
Unregister
Source
service-worker.js
35
Received 8/7/2019, 2:37:30 PM
Status
#39853 activated and is running
stop
Clients
http://localhost:8080/sdk.html
focus
http://localhost:8080/sdk.html
focus
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
localhost:8080 - deleted
Update
Unregister
Source
service-worker.js
1
Received 8/6/2019, 2:22:28 PM
Status
#39808 is redundant
Push
Test push message from DevTools.
Push
Filter by Path
Name Response-Type Content-Type Content-Length Time Cached
0 /v4/audio/jrDy3yteoRs)txDnyx)6iSFVloZeWmW4xYjBV7l80BXu1LyJw)Tefm7S7KO7uXofbU2… cors audio/mpeg 195,185 8/7/2019, 2:36:58 P
1 /v4/audio/jrDy3yteoRs0WkZwN82G98gr903dmPsLFVqlgFe0ldE)zRz7rTnWiN06HwM2dkgI5… cors audio/mpeg 447,633 8/7/2019, 2:37:02 P
2 /v4/audio/jrDy3yteoRs2OP8n6z2uY7mr)KGWDOMcwtlZ1wt9aod5O(sTlhcMLfX6EXeayPJLD… cors audio/mpeg 573,230 8/7/2019, 2:37:07 P
3 /v4/audio/jrDy3yteoRs6sT68ej8vWRa7K5)0ZGGuDwYlXBQWMY7qSb(5NYlpB7GeoY4Kcf(1… cors audio/mpeg 129,148 8/7/2019, 2:37:06 P
4 /v4/audio/jrDy3yteoRsAFPa4yyvlt4G9PesgKtmGSFJMi8lPPadZ6OKXOK20DgUBsPrktBf1aB… cors audio/mpeg 196,439 8/7/2019, 2:36:59 P
5 /v4/audio/jrDy3yteoRsBvPPvoqJs)AfRa4aX1S8uF78MWjg3rKVhSiOtcFnSPHhLbRPX2AD7G… cors audio/mpeg 143,150 8/7/2019, 2:36:58 P
6 /v4/audio/jrDy3yteoRsE))hAd4GfNQEjafwax67ZW8b5o1JdtWVRxigpE6Gz2JvlMpUPo7vQju3… cors audio/mpeg 107,832 8/7/2019, 2:36:57 P
7 /v4/audio/jrDy3yteoRsLrS1n))khFRMUKbT08lTCFJCKiQRadLYKwviBOkngqsBZe0TGWKxF3… cors audio/mpeg 199,574 8/7/2019, 2:37:04 P
8 /v4/audio/jrDy3yteoRsMZrn6UWP)(6mFXTgLb01)YBxWqJ5QKCtal30x4FS2wMfhbjY86242n… cors audio/mpeg 114,101 8/7/2019, 2:36:59 P
9 /v4/audio/jrDy3yteoRsQDll)TuUbTEiuJO(YfOGhs538Ws6NOHUrtq3vnMx5tDqnz6lnneYl8etE… cors audio/mpeg 110,340 8/7/2019, 2:37:00 P
10 /v4/audio/jrDy3yteoRsT6xDt9yK(O2YT7P3rVMm5b6l9hONl50GFLQebsD9C7wLuwOCFMN… cors audio/mpeg 118,490 8/7/2019, 2:36:57 P
Select a cache entry above to preview
Total entries: 135

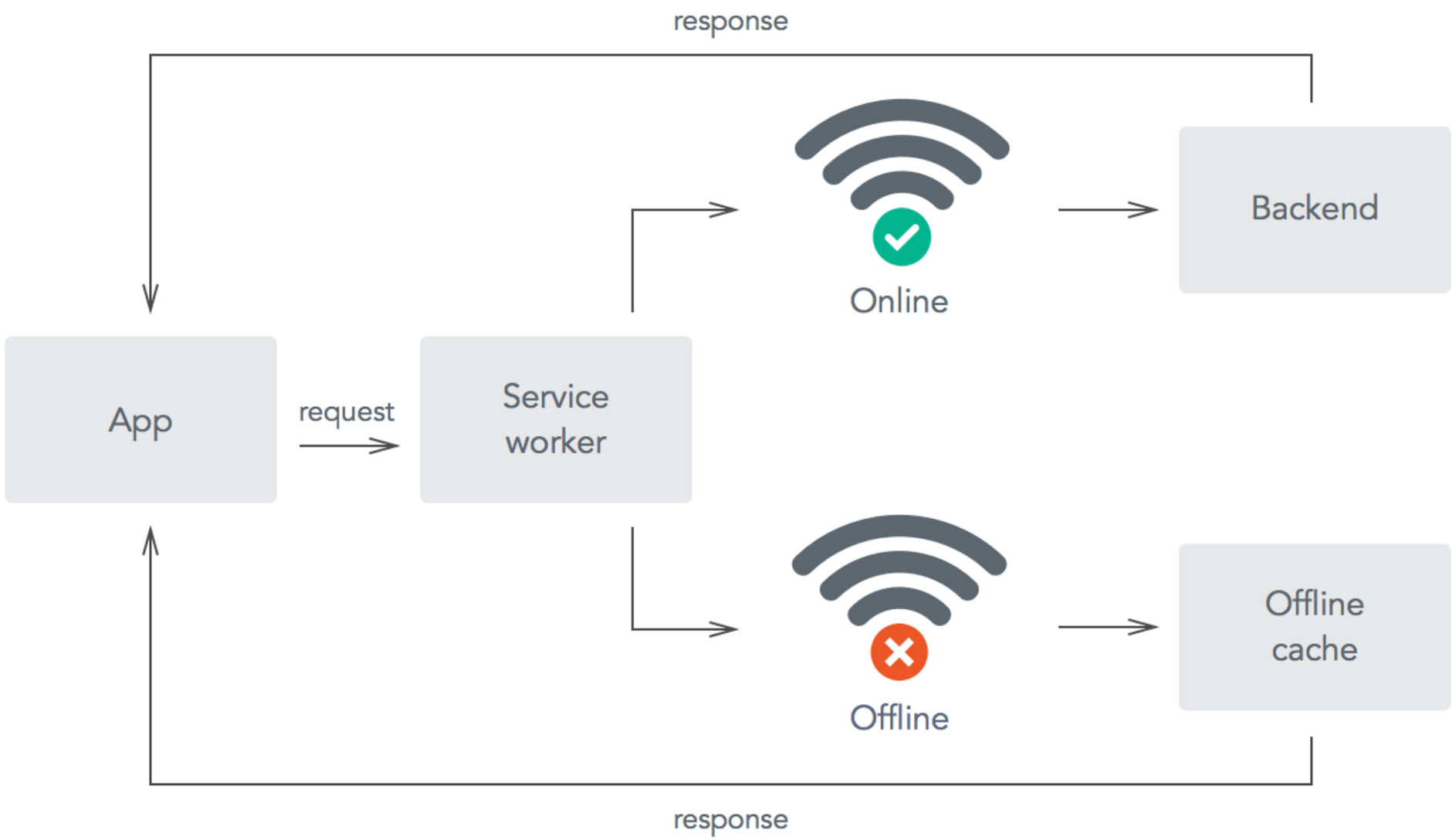


Image By Auth0

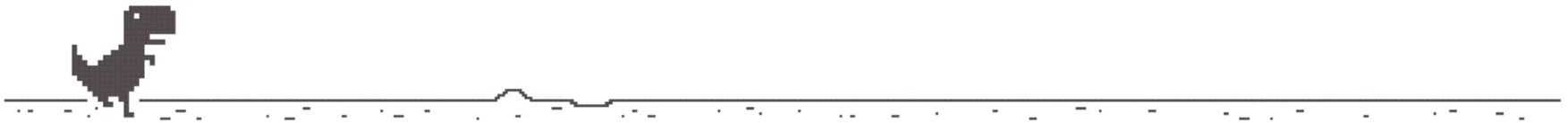
- Install

- Install
- Activate

- Install
- Activate
- Message

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

HI 00070 00007



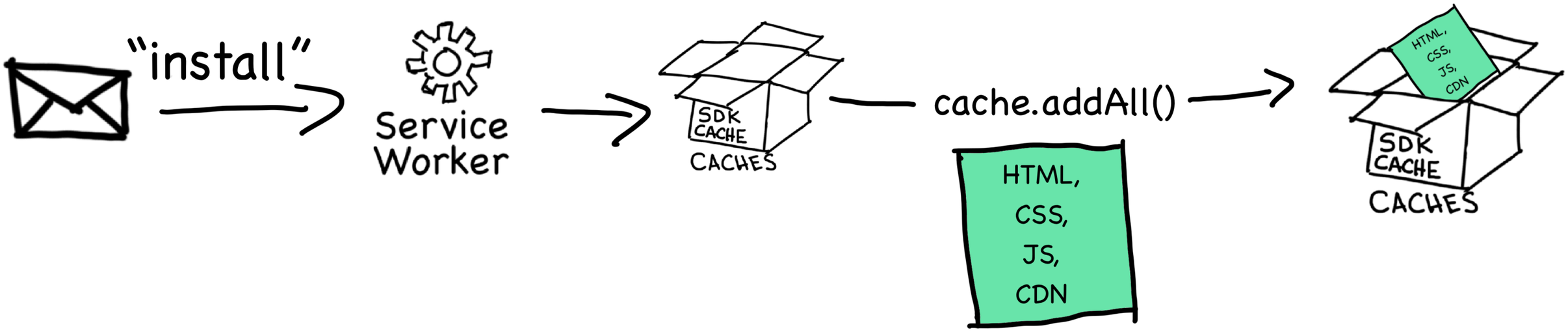
No internet

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))
    );
});
```

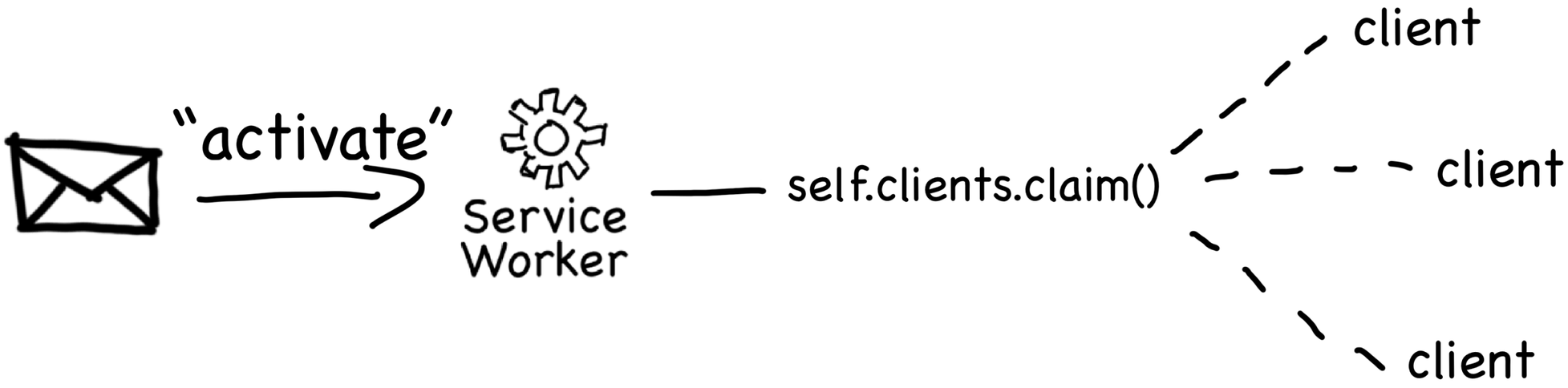
```
var urlsToCache = [
    '/scripts.js',
    '/service-worker.js',
    '/sdk.html',
    '/marx.css',
    '/findaway-sdk.2.3.0-alpha20190319.modern.js'
];

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

```
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))
    );
});
```

```
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))
    );
});
```

```
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))
    );
});
```

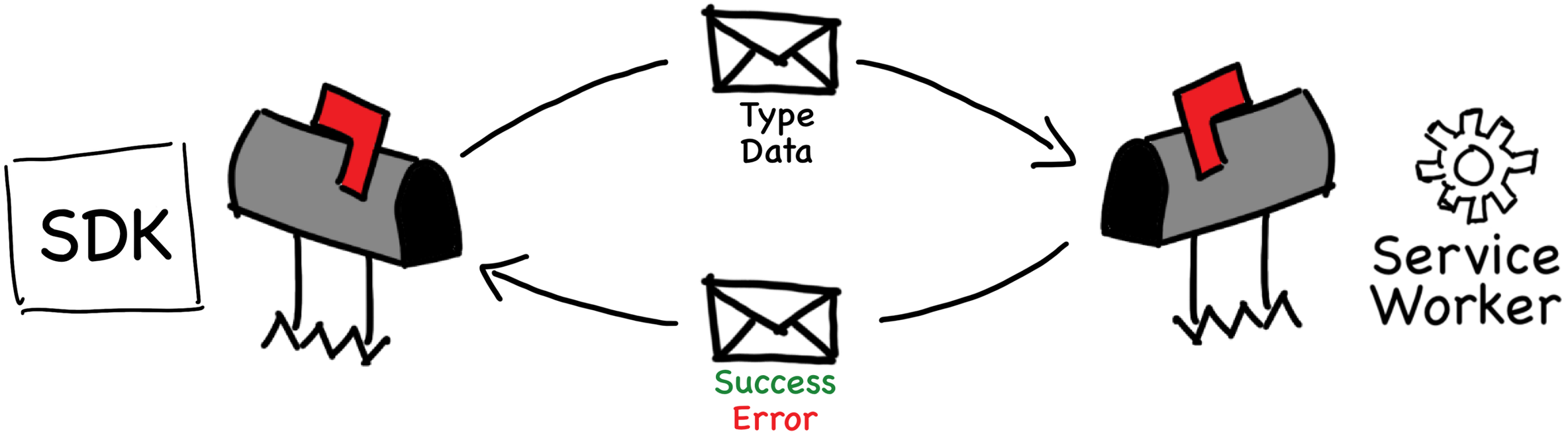
Claim ServiceWorker Clients



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

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


MOVIECLIPS.com
MAKE GIFS AT GIFSOUP.COM


SDK
Type
Data
Success
Error
Service
Worker

```
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]
        );
    });
);
```

```
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]
        );
    });
);
```

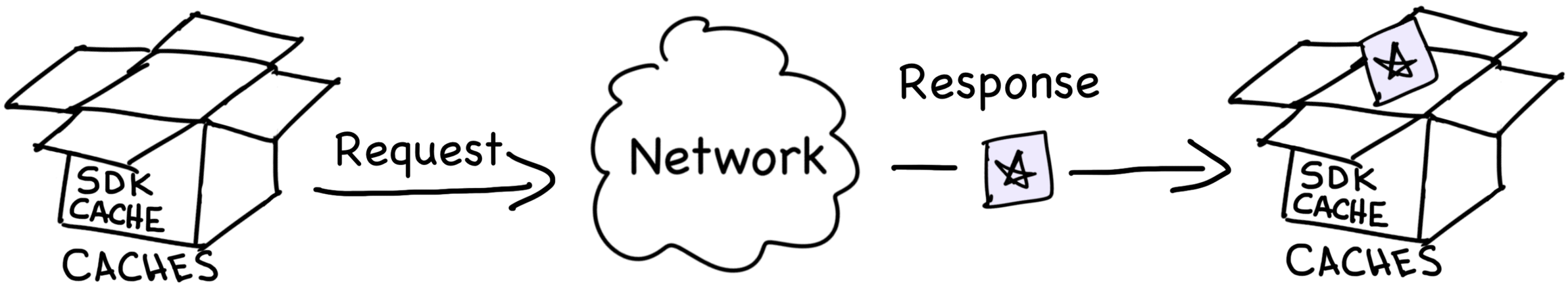
```
SDK.sendMessage = (type, data) => (
    new Promise((resolve, reject) => {
        let messageChannel = new window.MessageChannel();
        messageChannel.port1.onmessage = (event) => {
            // Do something with event infomration from our ServiceWorker
        };

        navigator.serviceWorker.controller.postMessage(
            {type, data},
            [messageChannel.port2]
        );
    });
);
```

```
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



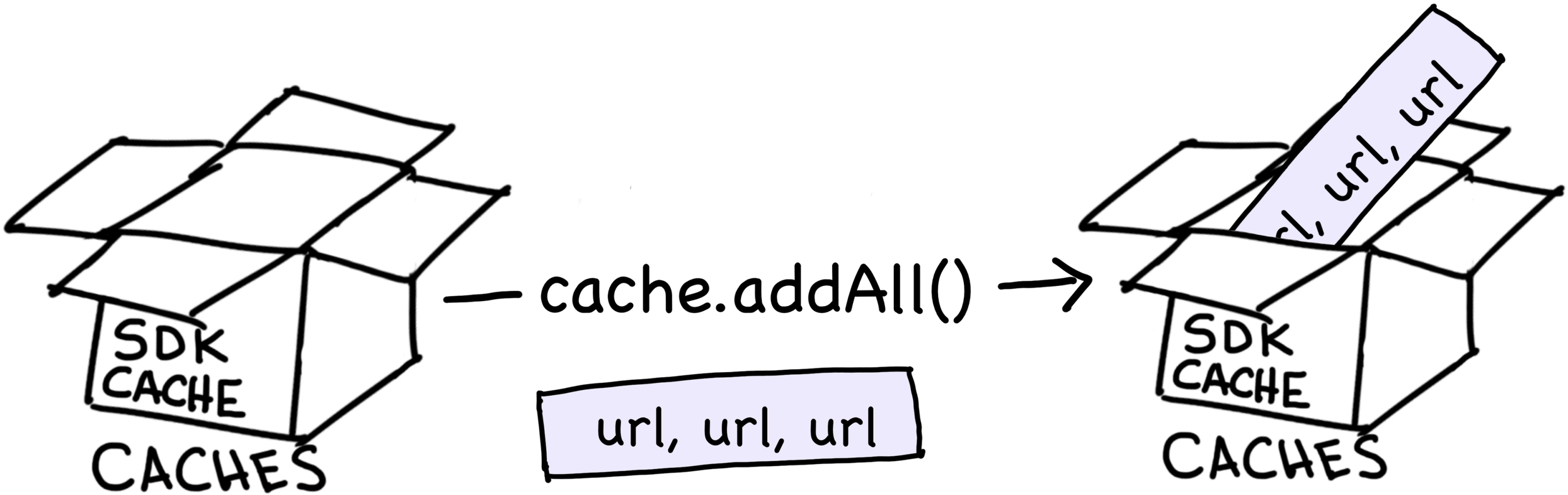
```
const cacheURL = ({url, headers}) => {
    return caches.open(`SDK-cache`)
        .then(cache => {
            fetch(url, {headers: new Headers(headers)})
                .then(resp => cache.put(url, resp))
        });
};
```

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

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

```
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

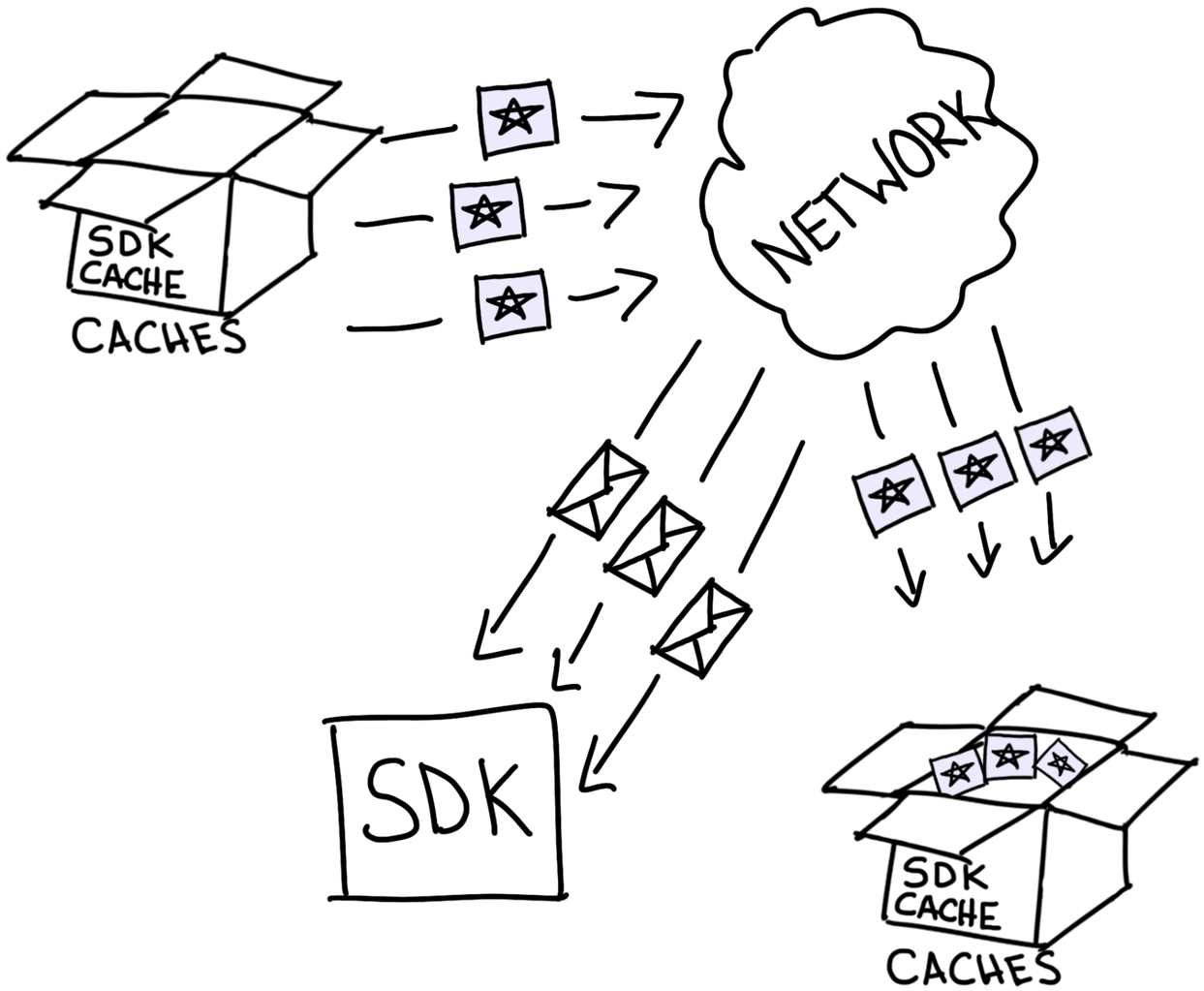


```
const cacheAudio = ({id, urls}) => {
    return caches.open(`book-${id}`)
        .then((cache) => cache.addAll(urls))
};
```

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

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

Cache audio with status



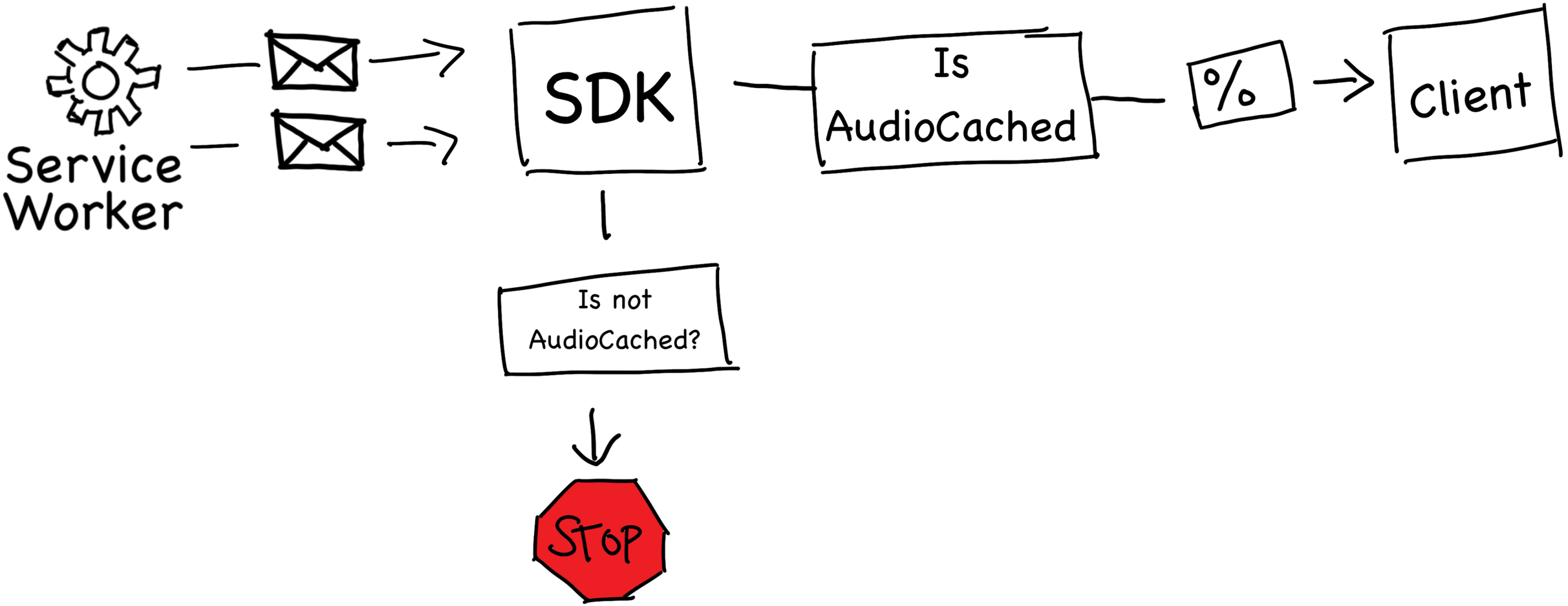
```
urls.forEach(url => {
    fetch(url)
        .then(resp => {
            messageClient({
                type: 'AudioCached',
                id: id,
                url
            });
            return cache.put(url, resp);
        })
        .then(() => resolve(url))
        .catch(err => reject(err));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

```
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));
});
```

```
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));
});
```

```
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);
});
```

```
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);
});
```

```
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);
});
```

```
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);
});
```

```
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

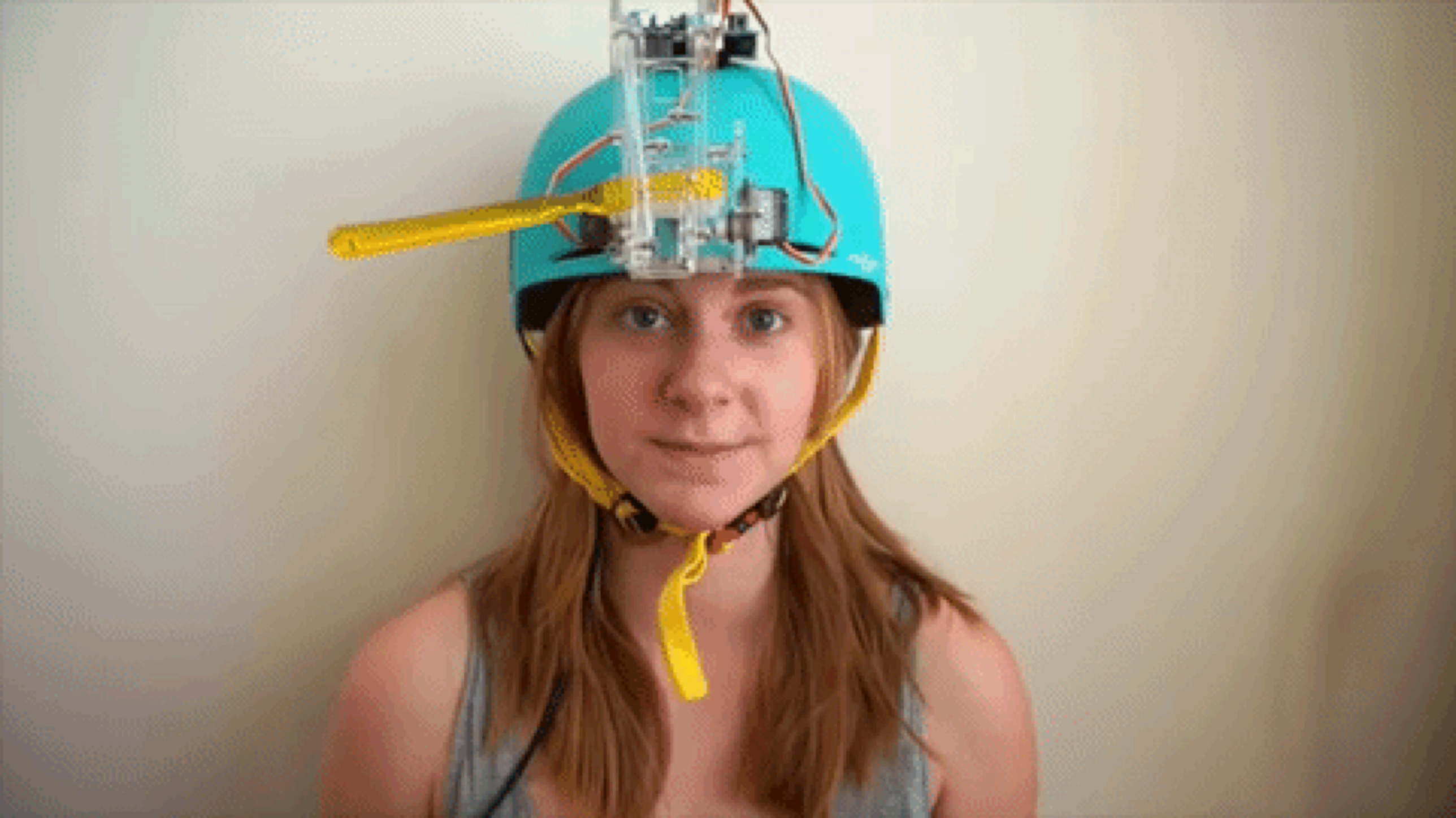

WINAMP
00:08
1. House Of Pain - Jump around
169 kbps 44 kHz
mono stereo
EQ PL
SHUFFLE
WINAMP PLAYLIST
1. House Of Pain - Jump around
2. Chaka Demus & Pliers - Murder She Wrote (A...
3. Sublime - Santeria (100Hitz - 90's Alternative ...
4. Porno For Pyros - Pets (GotRadio - 90's Alter...
5. Third Eye Blind - Semi-Charmed Life (1997) (...
6. The Presidents of the United States of Americ...
7. Cracker - Get Off This (181.FM - 90's Alternat...
8. Simple Minds - She's A River (AceRadio.Net - ...
9. Alice In Chains - Rooster (Big R Radio - 90's ...
10. NIRVANA - Lithium (M2 90 : ONLY 90's HITS -...
11. Hootie and the Blowfish - I Only Wanna Be ...
ADD REM SEL MISC
0:00/?
00:08
LIST OPTS

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


NBC
SONGLAND
PREMIERES
NEXT TUESDAY 10/9c

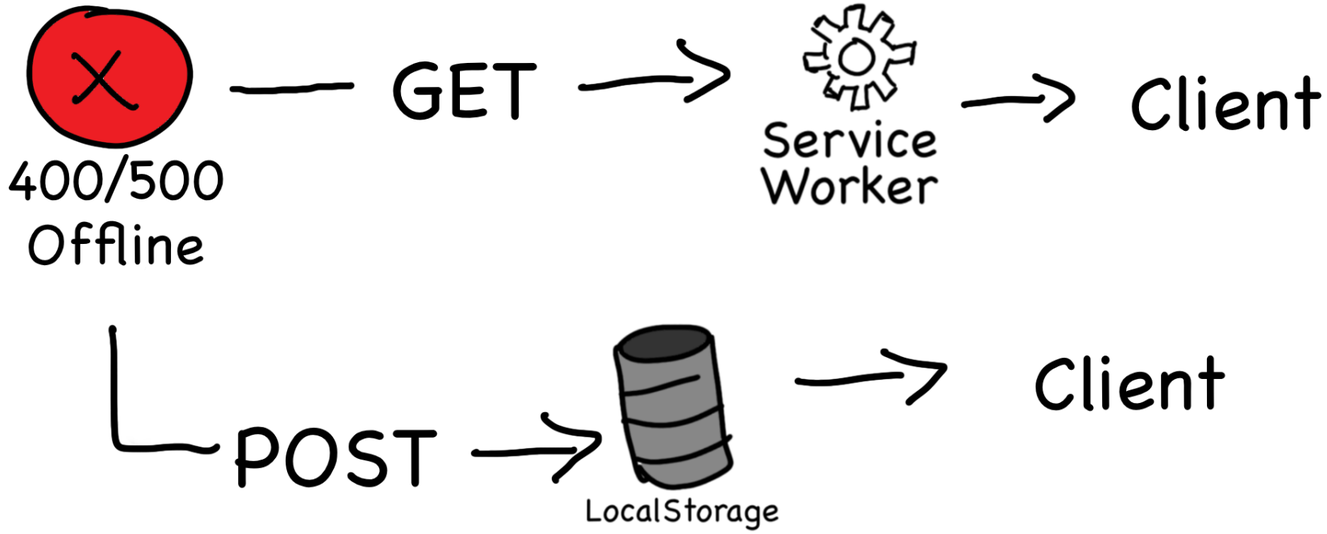
Cache POST Requests

```
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	service-worker.js:65 Script

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

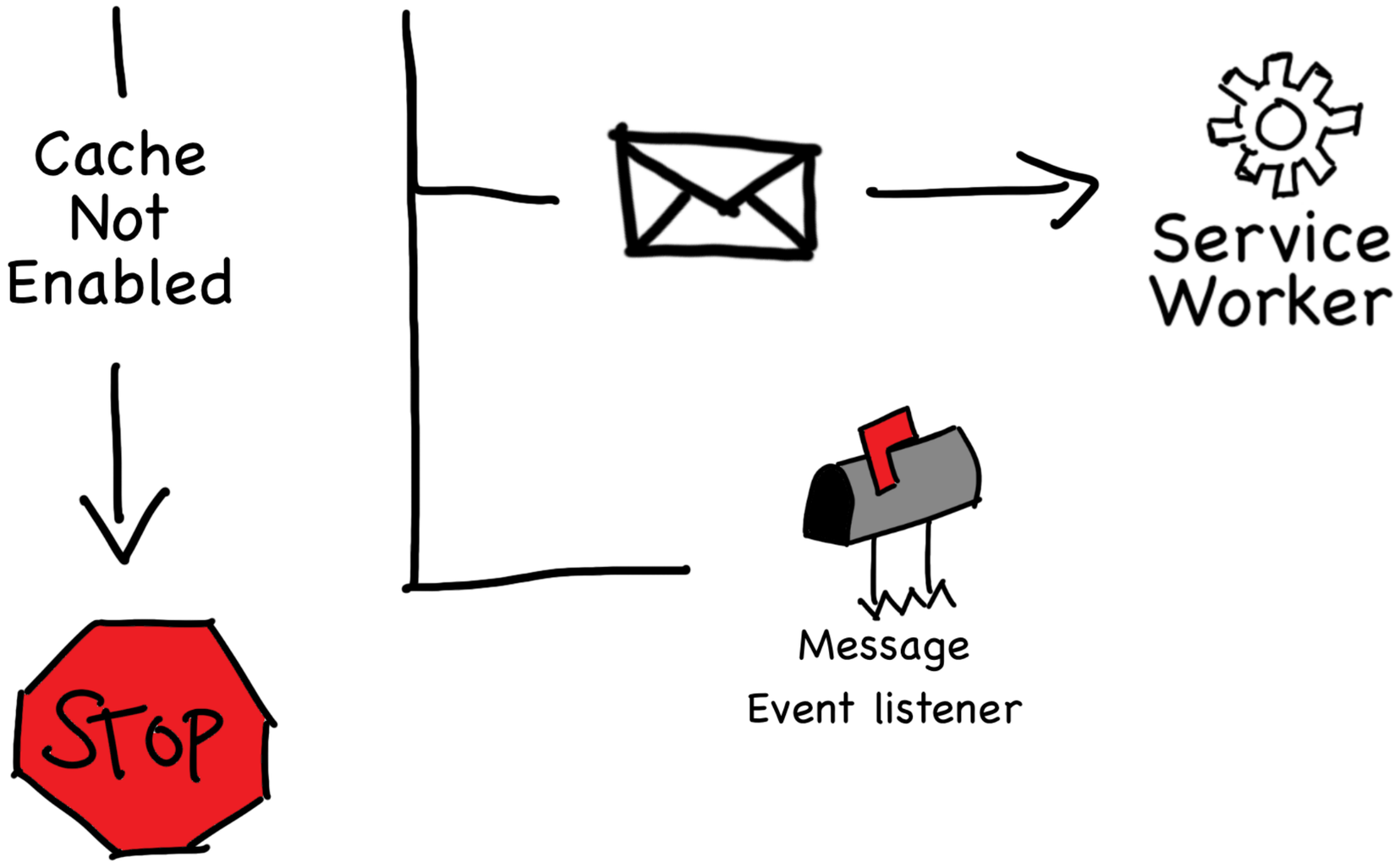

clawshots

- Cache Audio on Demand

- Cache Audio on Demand
- Check for cached book

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

SDK.cache()


Cache
Not
Enabled
STOP
Service
Worker
Message
Event listener

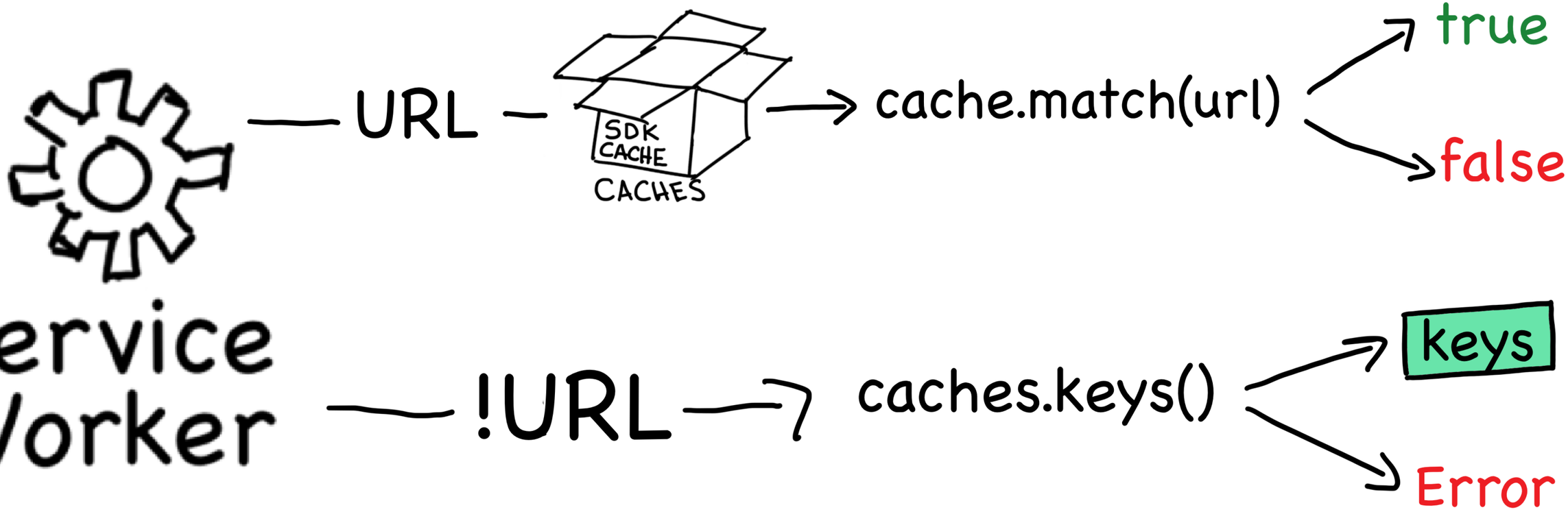
```
cache () {
    return new Promise((resolve, reject) => {
        if (!enable_cache) { return reject(new Error('Caching Not Enabled.')); }

        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));
    });
}
```

```
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));
    });
}
```

```
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));
    });
}
```

```
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));
    });
};
```

```
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));
    });
};
```

```
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));
    });
};
```

```
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));
    });
};
```

```
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));
    });
};
```

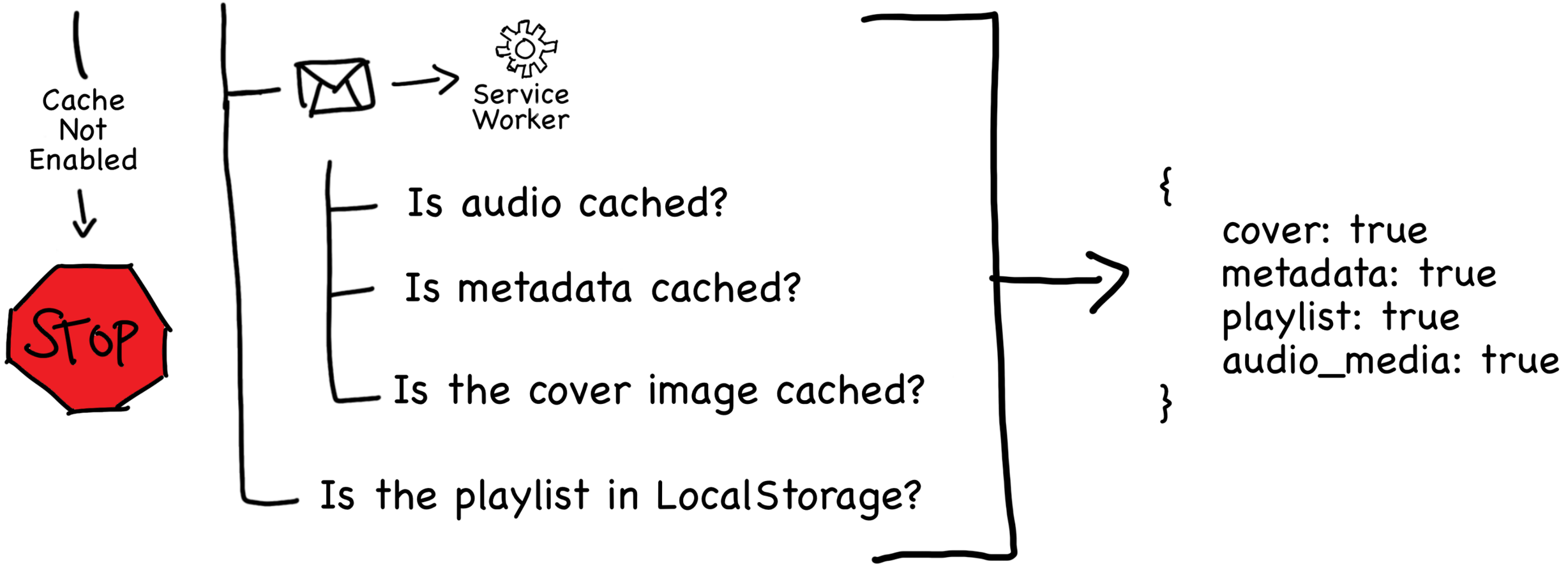
```
sendMessage('checkCaches')
    .then(keys => {
        isAudioCached = keys.includes(`book-${this.id}`);
    });
```

```
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));
    });
}
```

```
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));
    });
}
```

```
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));
    });
}
```

```
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));
    });
}
```

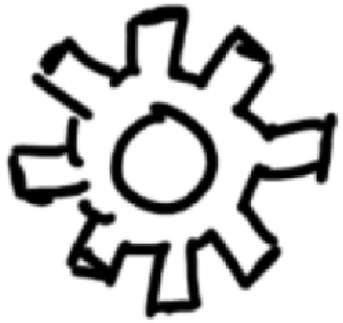
```
checkCache () {
    return new Promise((resolve, reject) => {
        if (!enable_cache) { return reject(new Error('Caching Not Enabled.')); }
        let isAudioCached = false;
        let isMetadataCached = false;

        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));
    });
}
```

```
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));
    });
}
```

```
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));
    });
}
```

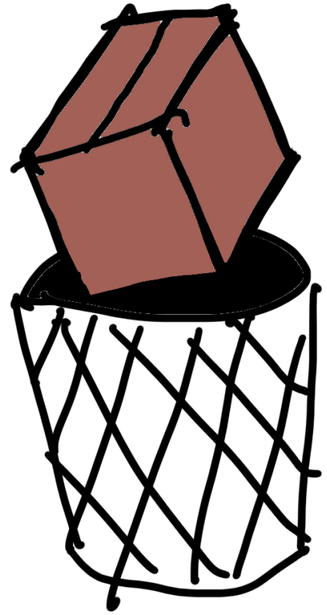
```
checkCache () {
    return new Promise((resolve, reject) => {
        if (!enable_cache) { return reject(new Error('Caching Not Enabled.')); }
        let isAudioCached = false;
        let isMetadataCached = false;

        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

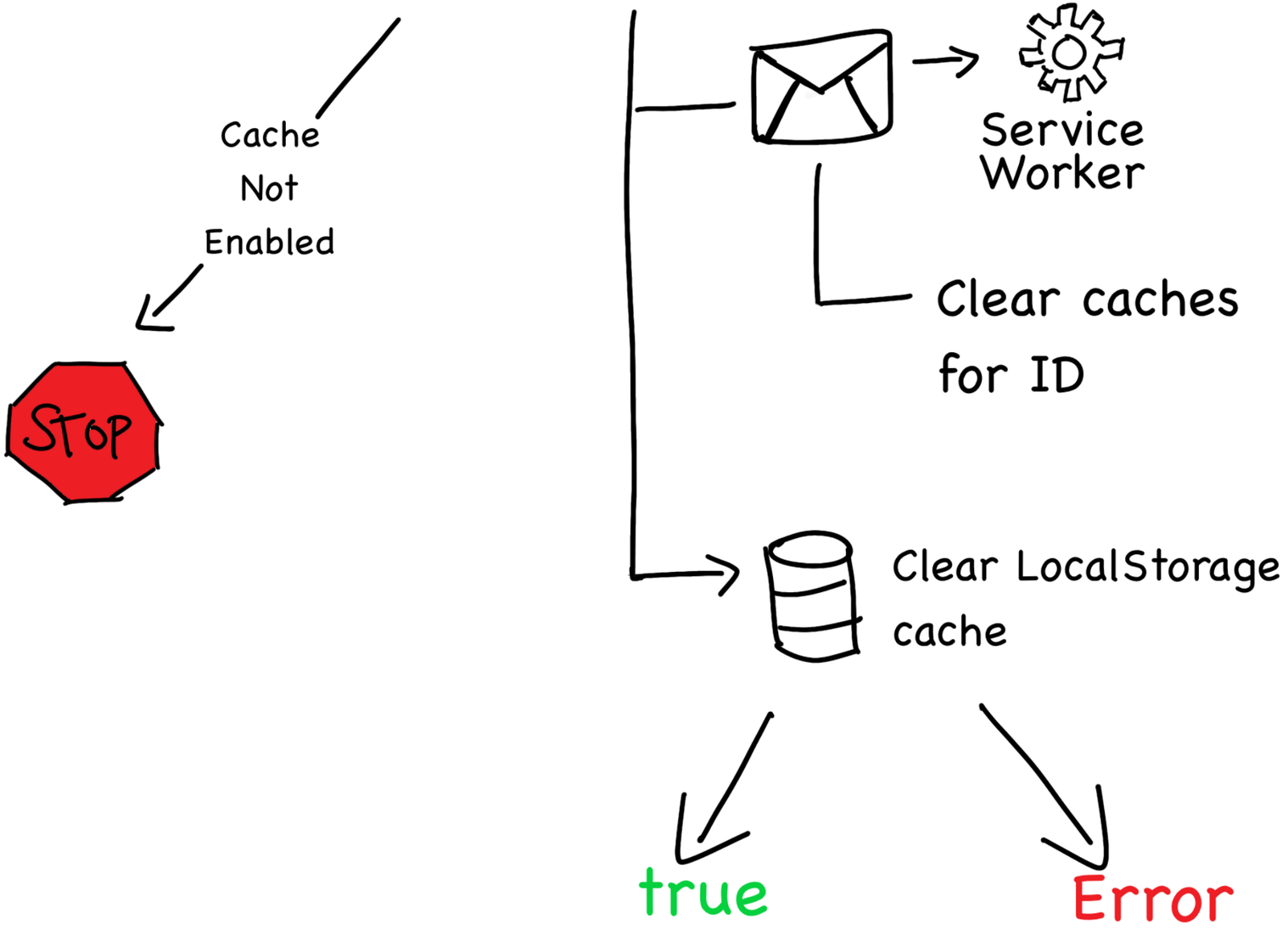
caches.delete()



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

```
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));
    });
}
```

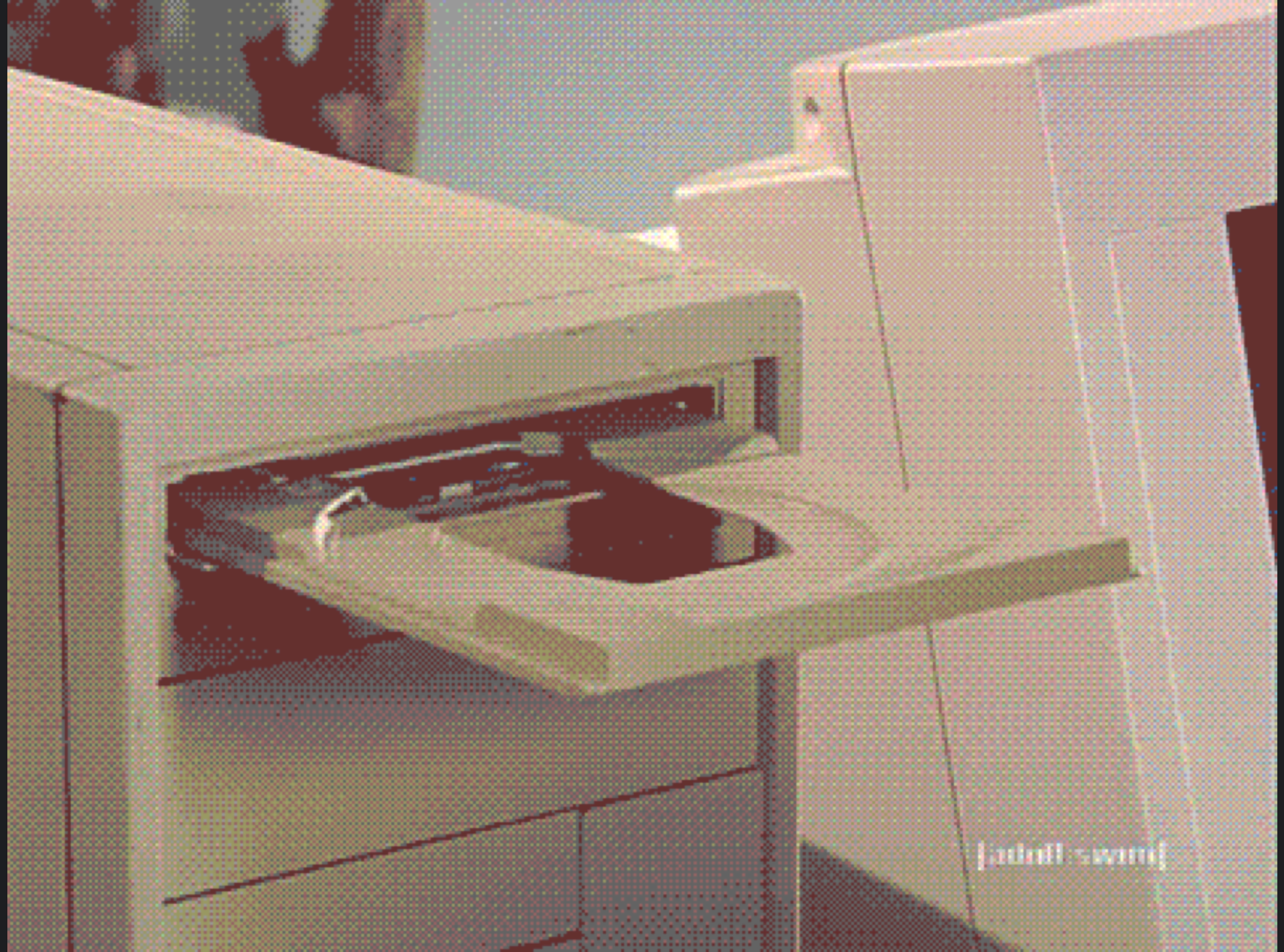
```
clearCache () {
    return new Promise((resolve, reject) => {
        if (!enable_cache) { return reject(new Error('Caching Not Enabled.')); }

        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));
    });
}
```

```
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));
    });
}
```

```
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));
    });
}
```

```
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 - WD

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

Usage: Global — % of all users: 91.54% + 0.39% = 91.93%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			74						
	17	67	75		12.1				4
11	18	68	76	12.1	12.3	all	75	12.12	9.2
	76	69	77	13	13				
		70	78	TP					
			79						

Notes | Known issues (0) | Resources (8) | Feedback

Details on partial support can be found on is ServiceWorker Ready?

THANKS

@camwardzala
camwardzala.com