BALANCING PRIVACY & PERFORMANCE

Front-end Forward, 31 August 2017

@declanrek

STARTMAIL

MEET STARTMAIL

Secure webmail

Privacy is a basic human right

Privacy by design

STARTMAIL = SECURE AND PRIVATE WEBMAIL

Inbox (204)
Drafts
Sent
Trash
Spam
Manage folders
Manage filters
Support

From: Subject: Received:

This folder is empty

No message selected

STARTMAIL

Full PGP encryption

Mailbox encryption(user vault)

IP address not stored and minimal logging

No user profile

Deleted = deleted

STARTMAIL

No cloud services

No webpage test

No pagespeed insights

No ngrok

Latency because no CDN

No native apps

Progressive

Responsive

Network Independent

App-Like

Fresh

A PWA is

Safe

Discoverable

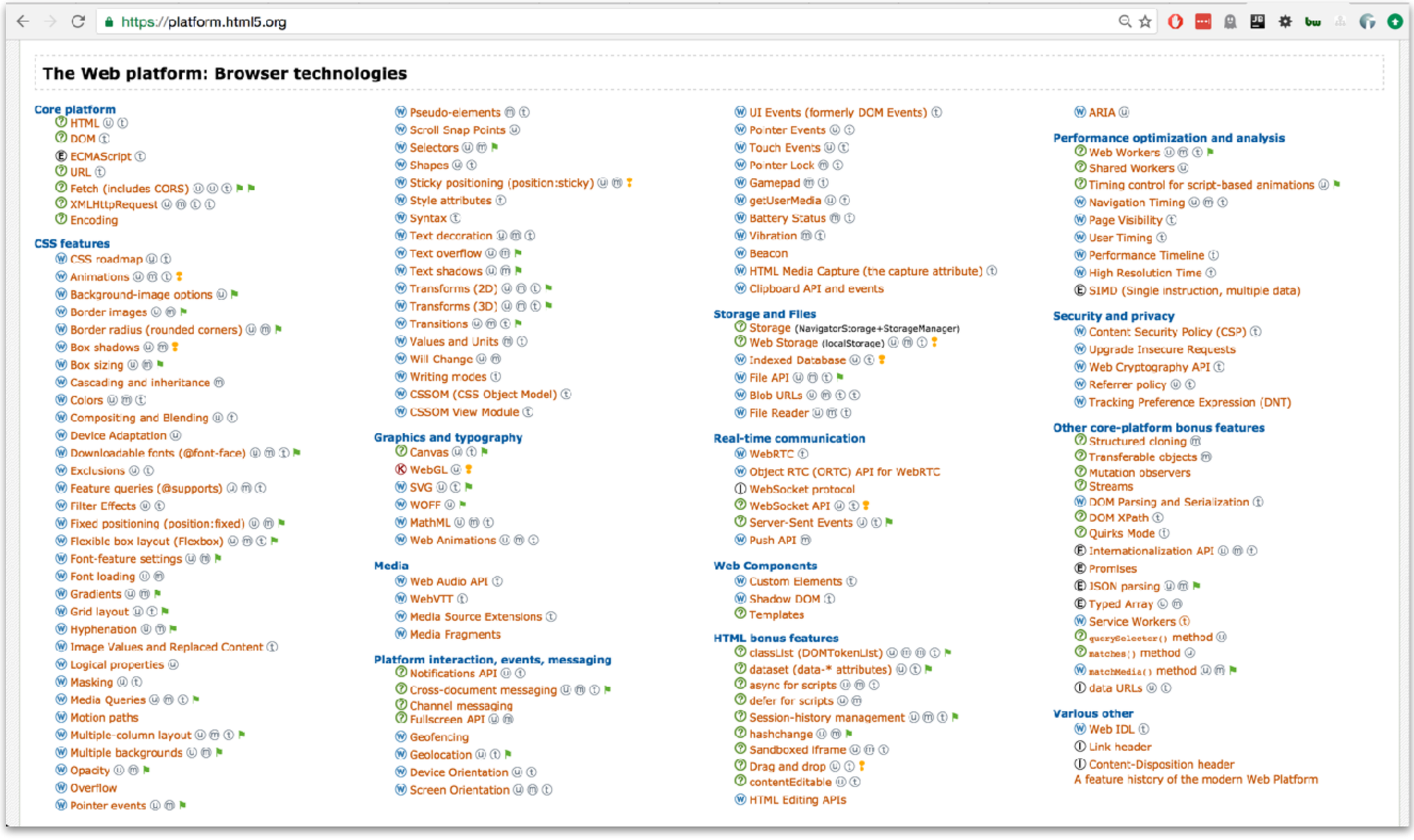
Re-engageable

Installable

Linkable

channel9.msdn.com/... (at 10m40s)

WEB TECHNOLOGY



platform.html5.org

As **new web APIs** become available, front-enders get **more responsibilities**

Security was never my responsibility until...

LATENCY

IP address or hostname www.startmail.com Ping

https://tools.keycdn.com/ping

RESPONSE TIMES

100ms = Instantaneous

1 second = Uninterrupted flow

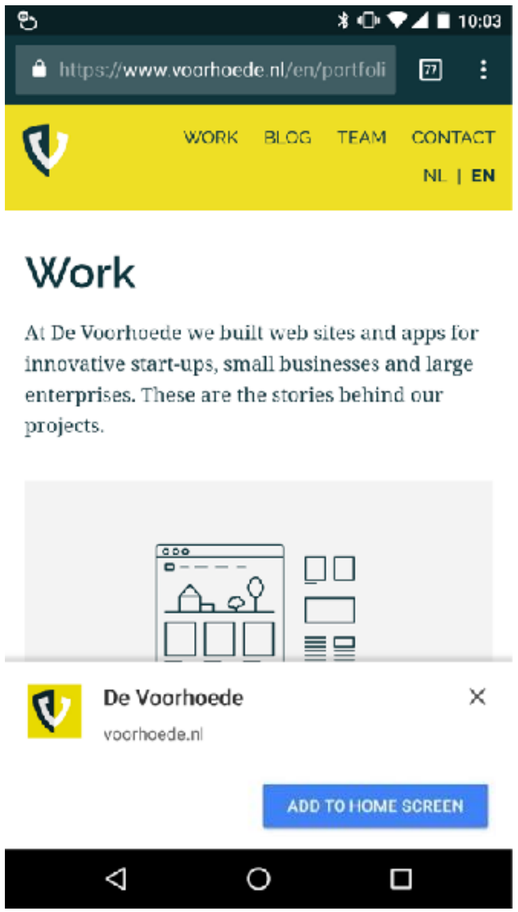
Jacob Nielsen - Response Times: The 3 Important Limits

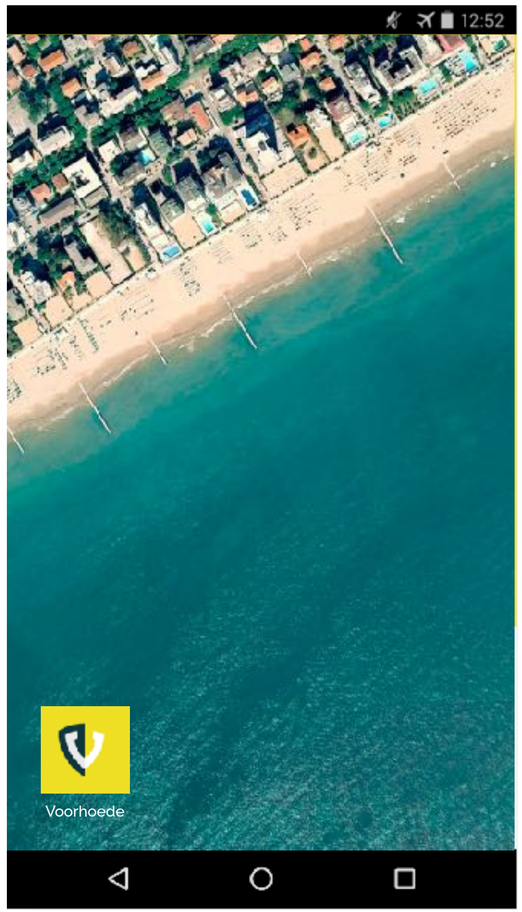
FAKE IT

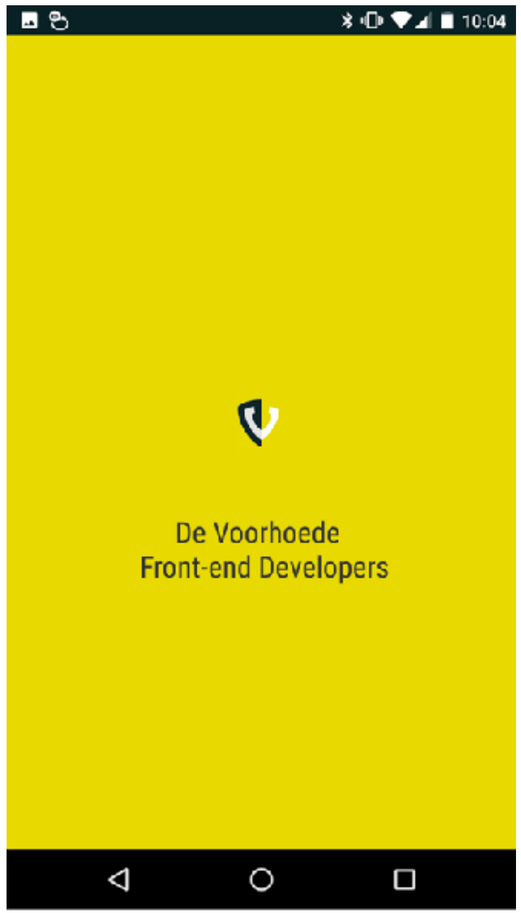
A network proxy in your browser?

SERVICE WORKER

VOORHOEDE.NL

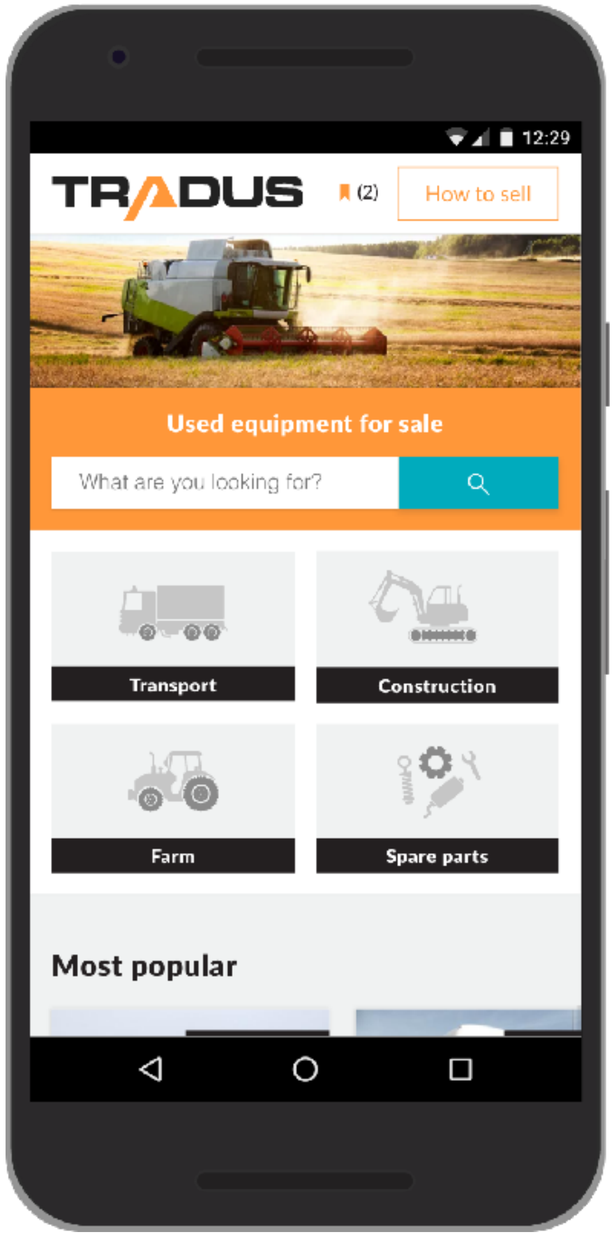


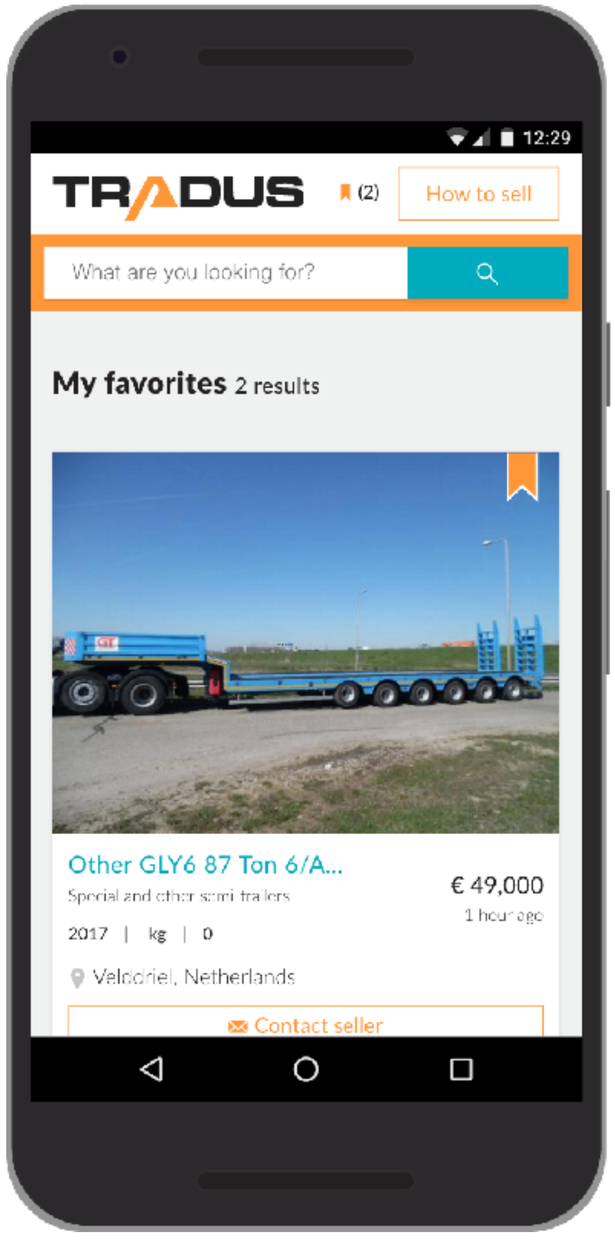


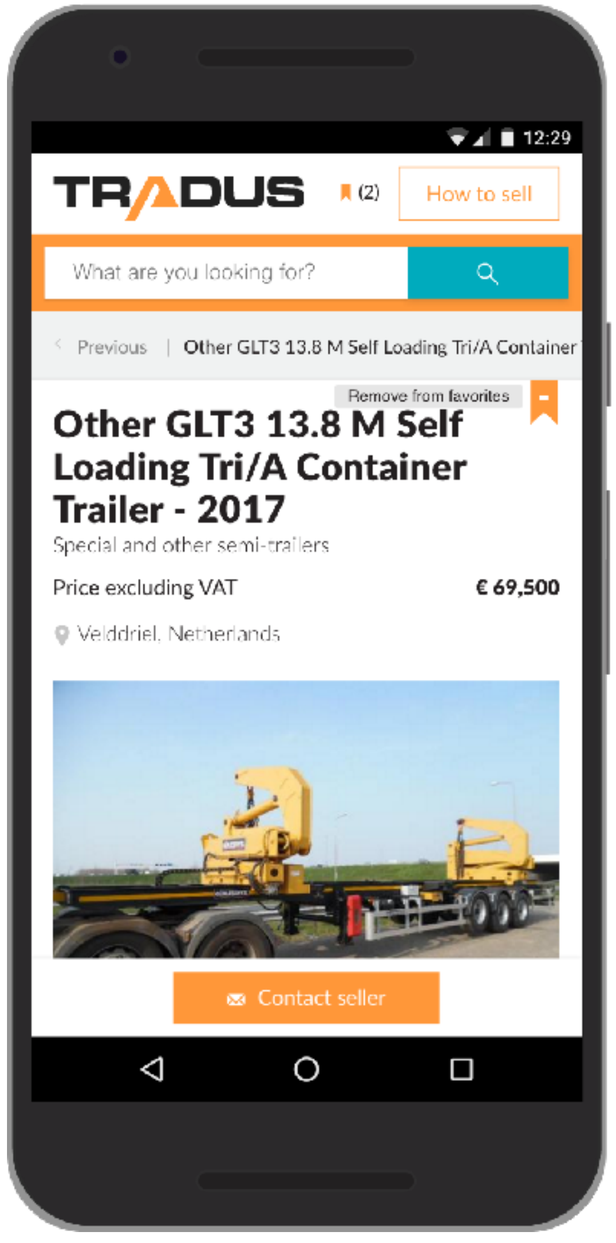


voorhoede.nl

EXAMPLE: TRADUS.COM







tradus.com

TROUW.NL

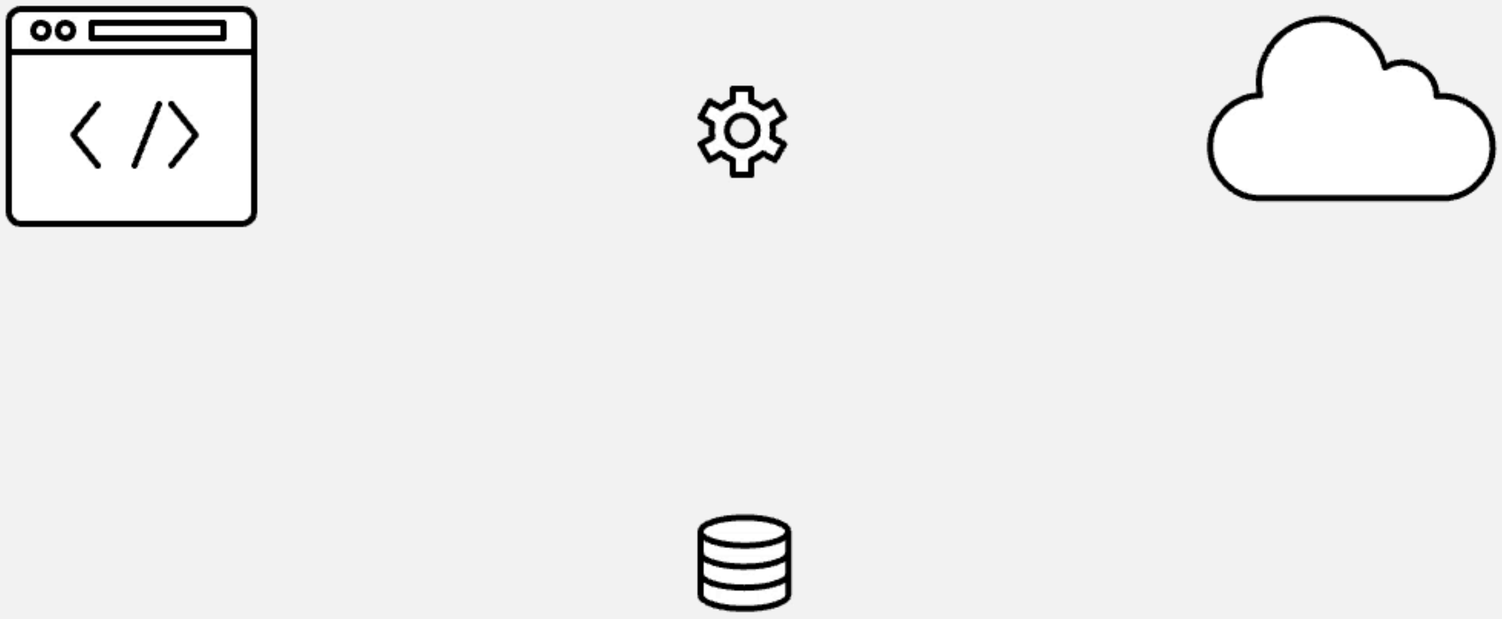
left-to-right: web, Android app, iOS app

read voorhoede.nl/nl/portfolio/trouw-volkskrant-parool/

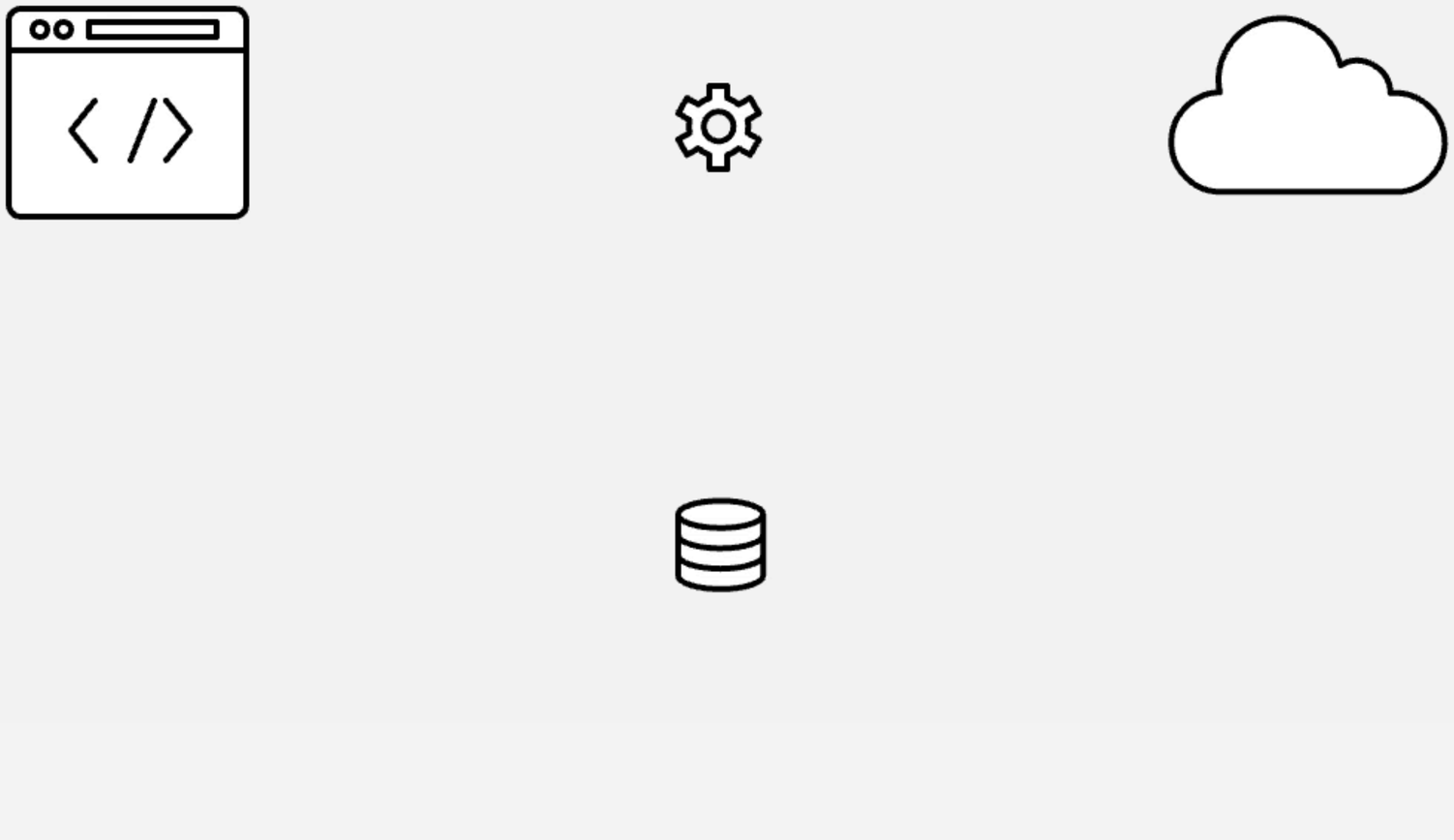
SERVICE WORKER AS PROXY

HIJACK FETCH

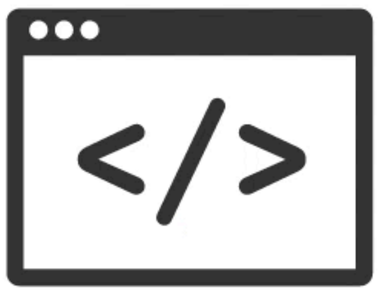
CACHE ON FETCH



SERVE FROM CACHE

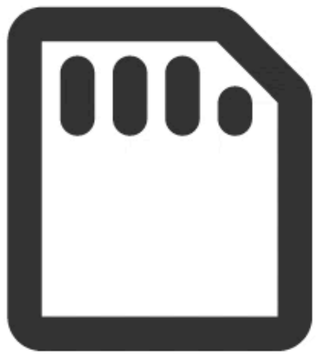


STALE WHILE REVALIDATE



Page

ServiceWorker

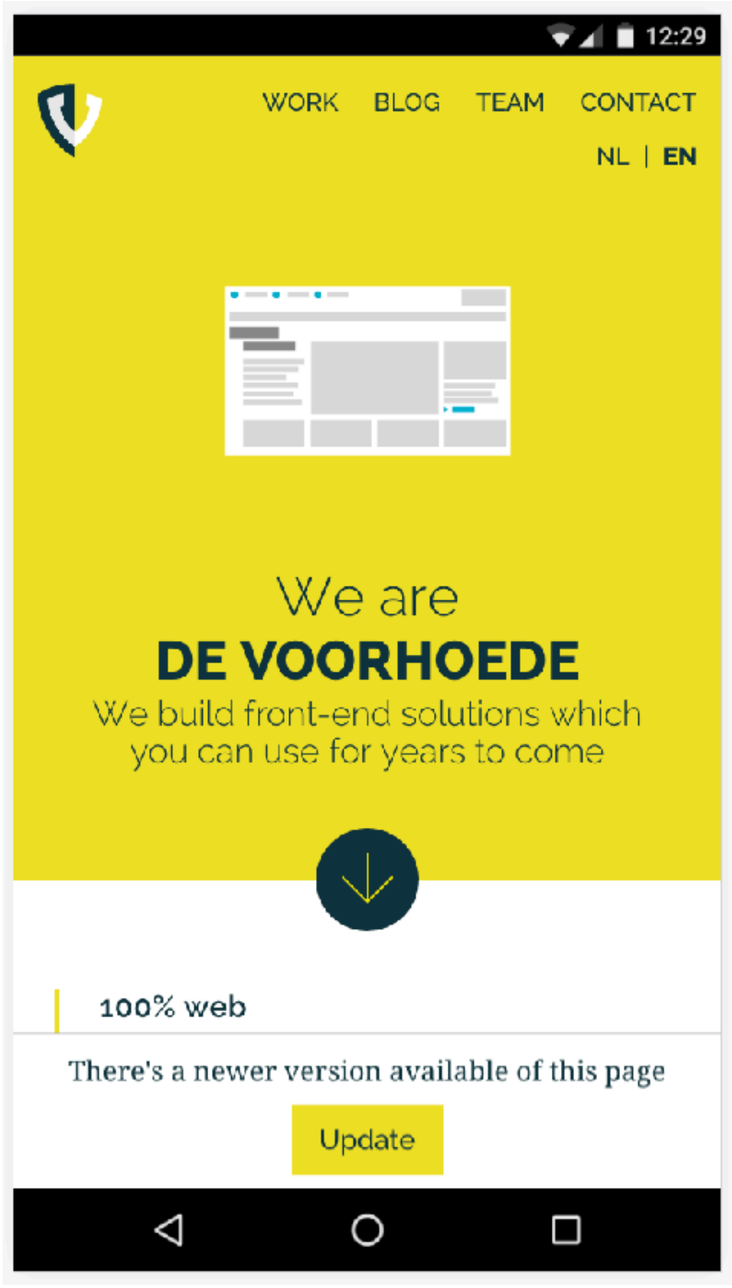


Cache

Network

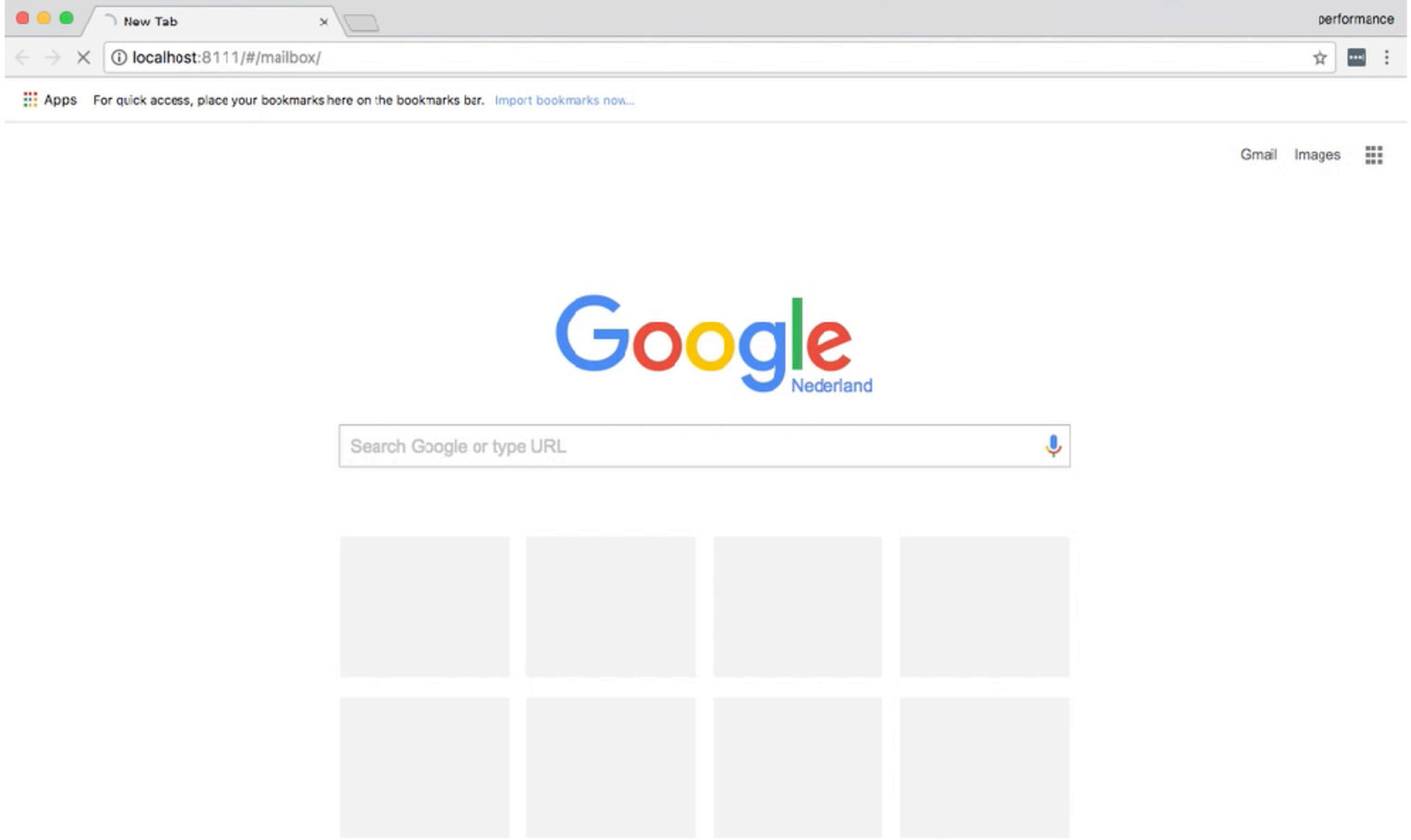
Offline cookbook by Jake Archibald

STALE WHILE REVALIDATE



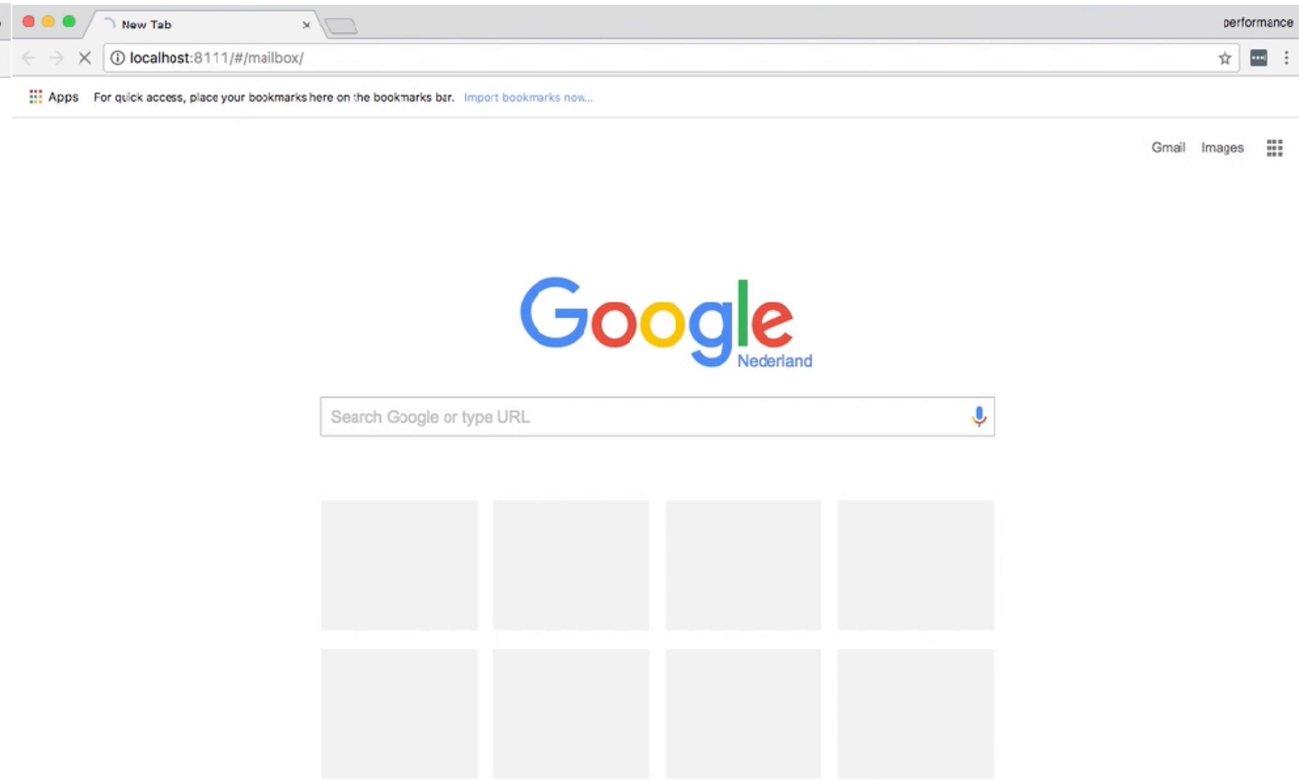
LET'S DO THIS!

NEW STARTMAIL ON HIGHSPEED INTERNET



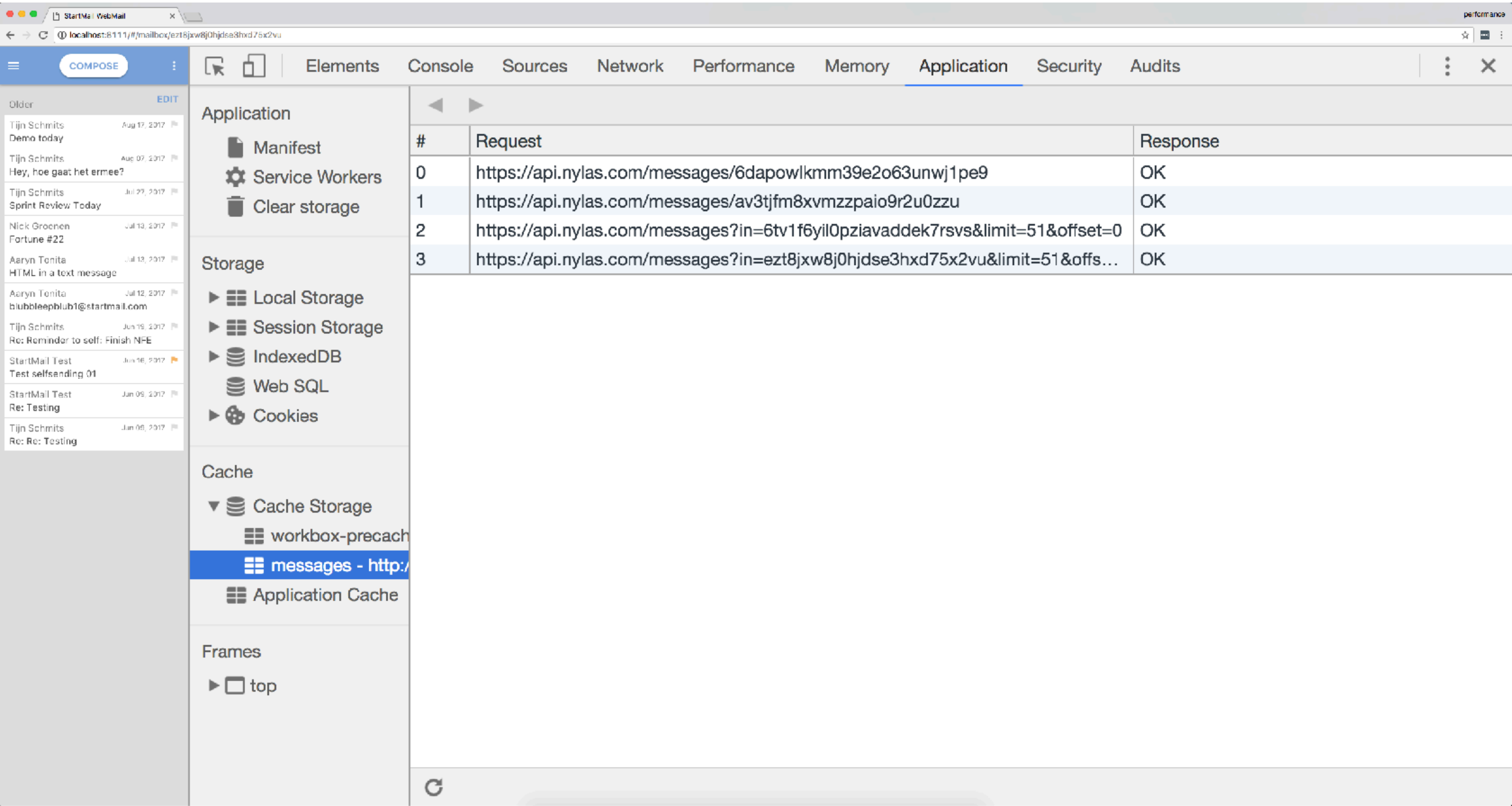
NEW STARTMAIL WITH SERVICE WORKER


localhost:8801/#/mailbox/
performance
localhost:8801/#/mailbox/

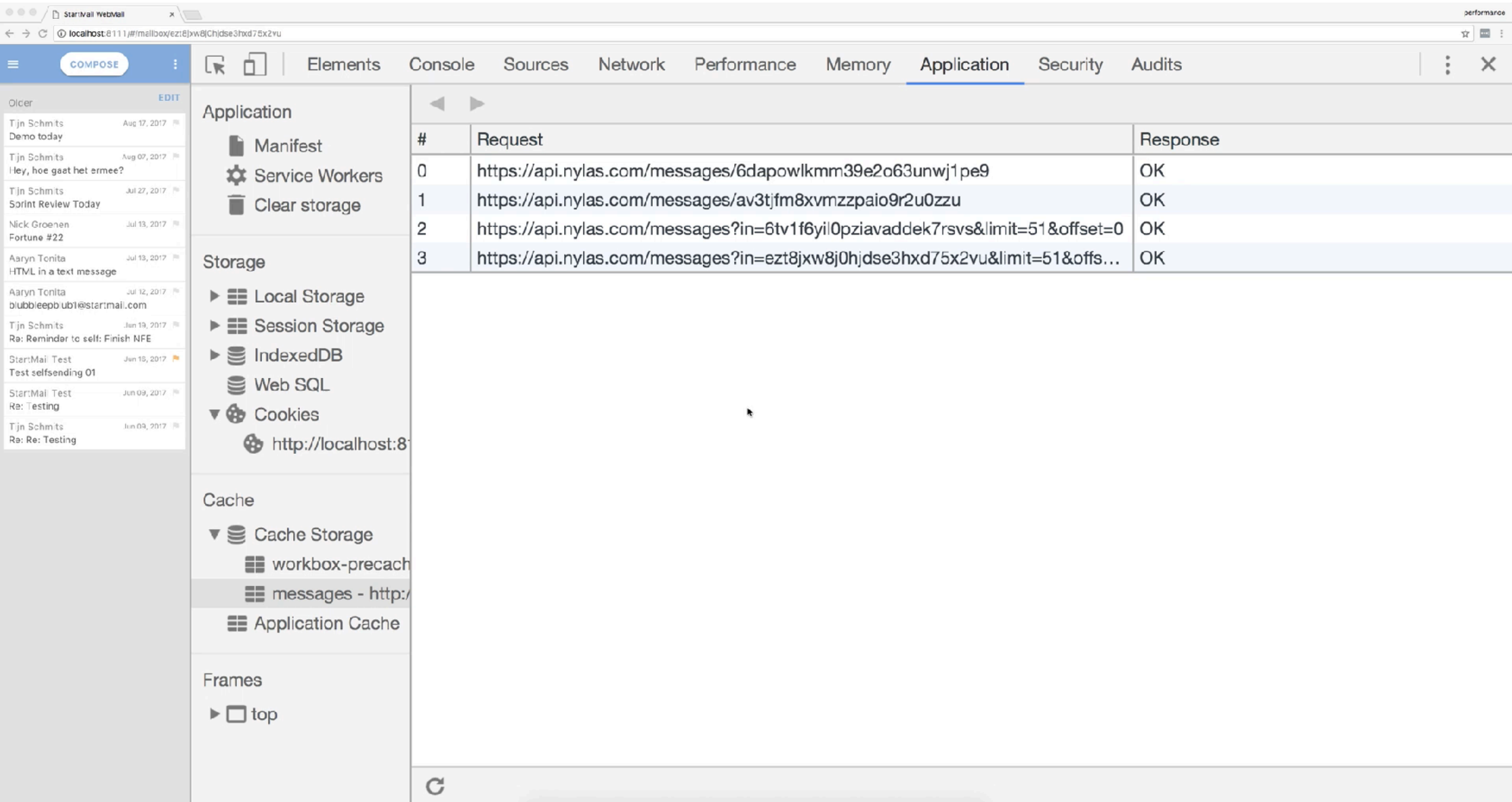

New Tab
performance
localhost:8111/#/mailbox/
Apps
For quick access, place your bookmarks here on the bookmarks bar.
Import bookmarks now...
Gmail
Images
Google
Nederland
Search Google or type URL

24s vs 16s

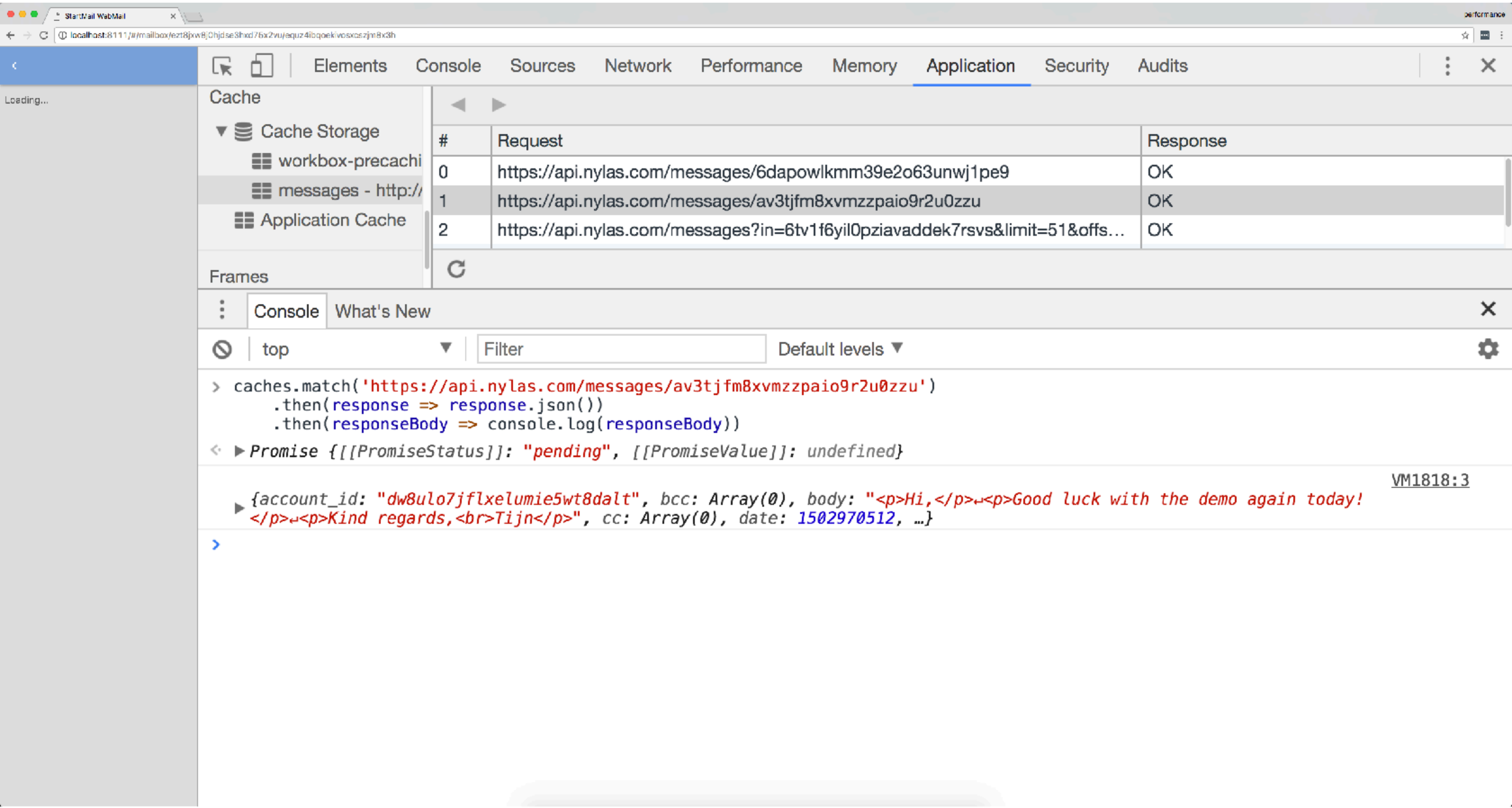
CACHED RESPONSES!!



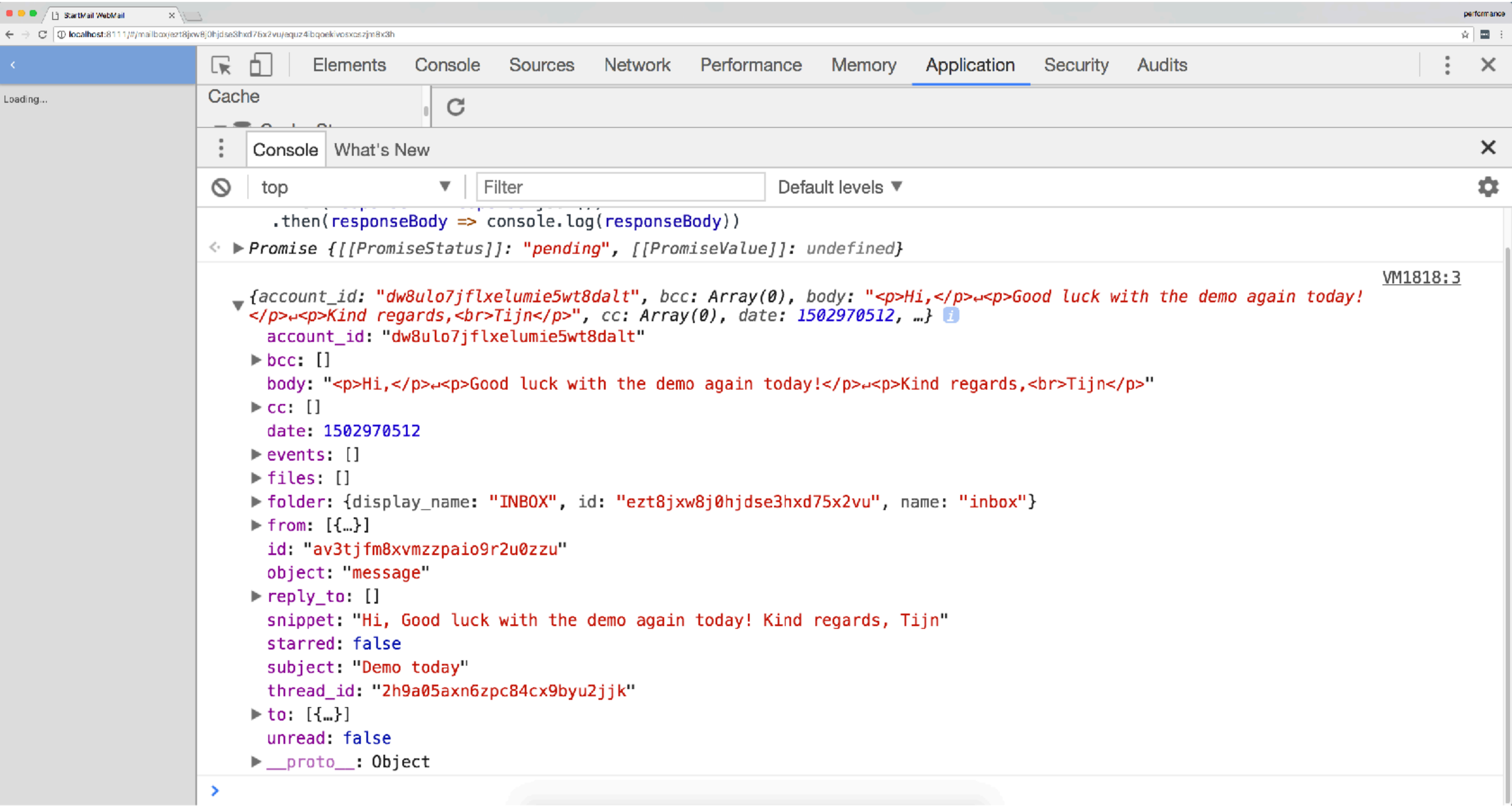
OFFLINE === UNAUTHENTICATED



CACHED RESPONSES?!



CACHED RESPONSES?!!??!



I'm caching **sensitive data**!

ALL EXAMPLES ARE PUBLIC

pwa.rocks

Delete onbeforeunload?

Authentication on cache storage?

Let's try encryption?

ENCRYPTION

ON FETCH

```
self.addEventListener('fetch', event => {
  const request = event.request;
  if (isApiGetRequest(request)) {
    event.respondWith(tryCacheThenNetwork(request, ‘messages-cache'));
  } else {
    event.respondWith(fetch(event.request));
  }
});
```

TRY CACHE THEN NETWORK

```
function tryCacheThenNetwork(request, cacheName) {
  return caches.open(cacheName)
    .then(cache => cache.match(request.url))
    .then(response => {
      if (response) {
        return decryptResponse(response);
      } else {
        return fetchAndCache(request, cacheName);
      }
    });
}
```

FETCH AND CACHE

```
async function fetchAndCache(request, cacheName) {
  const response = await fetch(request.url, {
    headers: new Headers(),
    credentials: 'include'
  });

  const encryptedResponse = await encryptResponse(response.clone());

  return caches.open(cacheName)
    .then(cache => cache.put(request, encryptedResponse))
    .then(() => response)
}
```

CHANGE THE RESPONSE

```
function encryptResponse(response) {
  return response.text()
    .then(text => encrypt(text, ‘supersecretpassword‘))
    .then(encryptedBody => {
      const options = {
        status: response.status,
        statusText: response.statusText,
        headers: {'content-type': 'application/json'}
      };
      return new Response(JSON.stringify(encryptedBody), options);
    });
}
```

ENCRYPTING A STRING WITH FORGE

```
function encrypt(message, password) {
  const salt = forge.random.getBytesSync(128);
  const key = forge.pkcs5.pbkdf2(password, salt, 4, 16);
  const iv = forge.random.getBytesSync(16);

  const cipher = forge.cipher.createCipher('AES-CBC', key);
  cipher.start({iv: iv});
  cipher.update(forge.util.createBuffer(message));
  cipher.finish();

  const cipherText = forge.util.encode64(cipher.output.getBytes());

  return {
    cipher_text: cipherText,
    salt: forge.util.encode64(salt),
    iv: forge.util.encode64(iv)
  };
}
```

Forge

In cryptography, a **salt** is random data that is used as an additional input to a one-way function that "hashes" a password or passphrase.

Wikipedia - Salt

ENCRYPTING A STRING WITH FORGE

```
function encrypt(message, password) {
  const salt = forge.random.getBytesSync(128);
  const key = forge.pkcs5.pbkdf2(password, salt, 4, 16);
  const iv = forge.random.getBytesSync(16);

  const cipher = forge.cipher.createCipher('AES-CBC', key);
  cipher.start({iv: iv});
  cipher.update(forge.util.createBuffer(message));
  cipher.finish();

  const cipherText = forge.util.encode64(cipher.output.getBytes());

  return {
    cipher_text: cipherText,
    salt: forge.util.encode64(salt),
    iv: forge.util.encode64(iv)
  };
}
```

Forge

An **Initialization Vector** is an unpredictable random number used to “initialize” an encryption function, resulting in different ciphertexts everytime you encrypt the same message

ENCRYPTING A STRING WITH FORGE

```
function encrypt(message, password) {
  const salt = forge.random.getBytesSync(128);
  const key = forge.pkcs5.pbkdf2(password, salt, 4, 16);
  const iv = forge.random.getBytesSync(16);

  const cipher = forge.cipher.createCipher('AES-CBC', key);
  cipher.start({iv: iv});
  cipher.update(forge.util.createBuffer(message));
  cipher.finish();

  const cipherText = forge.util.encode64(cipher.output.getBytes());

  return {
    cipher_text: cipherText,
    salt: forge.util.encode64(salt),
    iv: forge.util.encode64(iv)
  };
}
```

Forge

We can encrypt a string

THE MESSAGE AS OBJECT


```
{
    account_id: "dw8ulo7jflxelumie5wt8dalt",
    bcc: [],
    body: "<p>Hi,</p>\n<p>Good luck with the demo again today!</p>\n<p>Kind regards,<br>Tijn</p>",
    cc: [],
    date: 1502970512,
    events: [],
    files: [],
    folder: {
        display_name: "INBOX",
        id: "ezt8jxw8j0hjdse3hxd75x2vu",
        name: "inbox"
    },
    from: [
        {
            email: "tijn@startmail.com",
            name: "Tijn Schmits"
        }
    ],
    id: "av3tjfm8xvmzzpaio9r2u0zzu",
    object: "message",
    reply_to: [],
    snippet: "Hi, Good luck with the demo again today! Kind regards, Tijn",
    starred: false,
    subject: "Demo today",
    thread_id: "2kbj6ob1pke0g8f04j62z7e9g",
    to: [
        {
            email: "blubbleepblub1@startmail.com",
            name: ""
        }
    ],
    unread: false
}
```

THE MESSAGE AS STRING


```
'{"account_id":"dw8ulo7jflxelumie5wt8dalt","bcc":[],"body":"<p>Hi,</
p>\n<p>Good luck with the demo again today!</p>\n<p>Kind
regards,<br>Tijn</p>","cc":[],"date":1502970512,"events":[],"files":
[],"folder":
{"display_name":"INBOX","id":"ezt8jxw8j0hjdse3hxd75x2vu","name":"inbox"
},"from":[{"email":"tijn@startmail.com","name":"Tijn
Schmits"}],"id":"av3tjfm8xvmzzpaio9r2u0zzu","object":"message","reply_t
o":[],"snippet":"Hi, Good luck with the demo again today! Kind regards,
Tijn","starred":false,"subject":"Demo
today","thread_id":"2kbj6ob1pke0g8f04j62z7e9g","to":
[{"email":"blubbleepblub1@startmail.com","name":""}],"unread":false}'
```

ENCRYPTED MESSAGE

```
{
    cipher_text: ‘2W9UmQxx2FUHG..’,
    iv: ‘vomfUHS2AZrBr6XyTmM2mw..’,
    salt: ‘YDX/0pAtvbXpe3WAVALX..’
}
```

DECRYPTION

SERVE FROM CACHE

```
function tryCacheThenNetwork(request, cacheName) {
  return caches.open(cacheName)
    .then(cache => cache.match(request.url))
    .then(response => {
      if (response) {
        return decryptResponse(response);
      } else {
        return fetchAndCache(request, cacheName);
      }
    });
}
```

DECRYPTING

```
function decryptResponse(response) {
  return response.json()
    .then(responseBody => decrypt(responseBody, ‘supersecretpassword‘))
    .then(decryptedBody => {
      const options = {
        status: response.status,
        statusText: response.statusText,
        headers: {'content-type': 'application/json'}
      };
      return new Response(JSON.stringify(decryptedBody), options);
    });
}
```

DECRYPTING

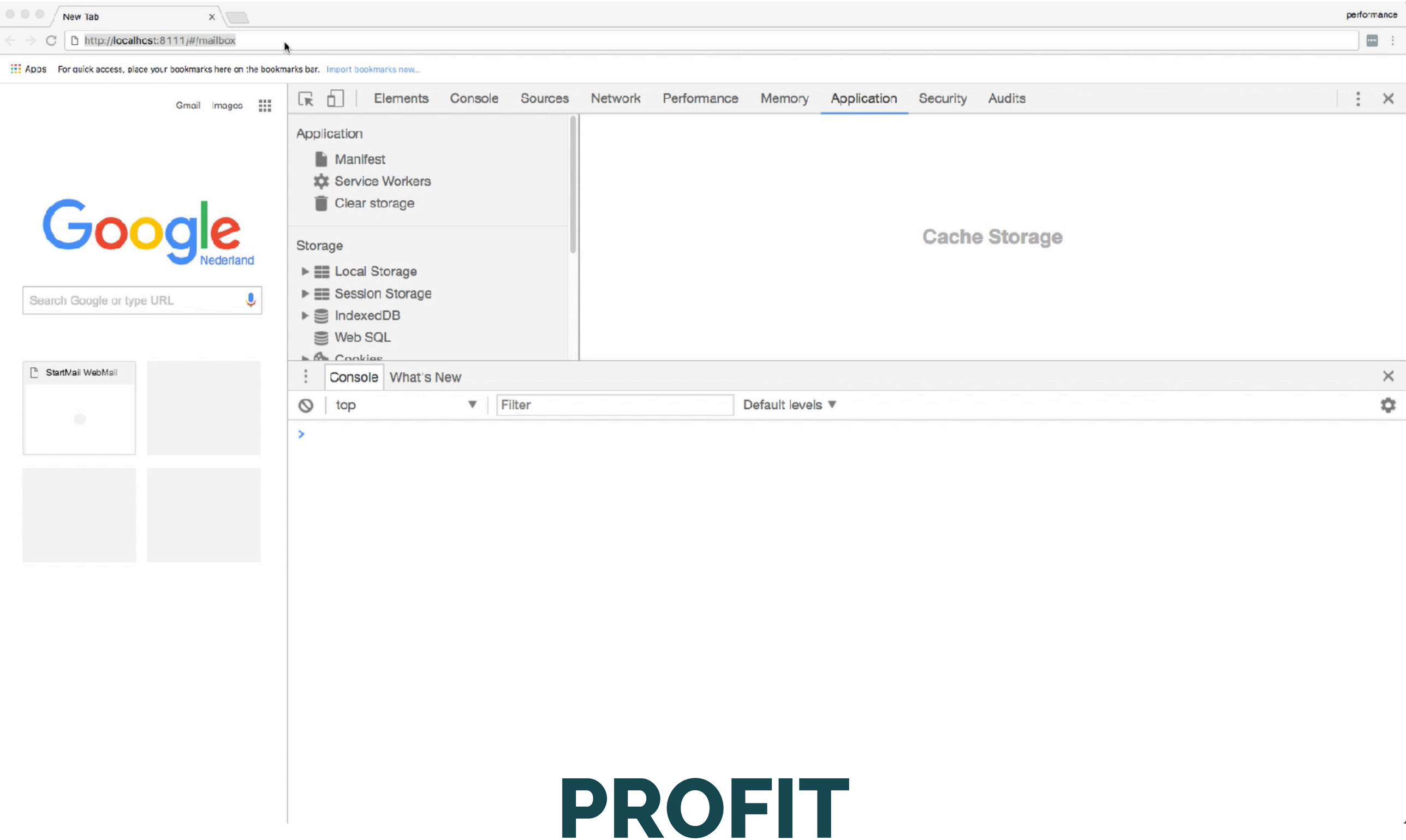
```
function decrypt({ cipherText, salt, iv }, password) {
  const key = forge.pkcs5.pbkdf2(password, forge.util.decode64(salt), 4, 16);
  const decipher = forge.cipher.createDecipher('AES-CBC', key);
  decipher.start({iv: forge.util.decode64(iv)});
  decipher.update(forge.util.createBuffer(forge.util.decode64(cipherText)));
  decipher.finish();

  return decipher.output.toString();
}
```

DECRYPTING

```
function decryptResponse(response) {
  return response.json()
    .then(responseBody => decrypt(responseBody, ‘supersecretpassword‘))
    .then(decryptedBody => {
      const options = {
        status: response.status,
        statusText: response.statusText,
        headers: {'content-type': 'application/json'}
      };
      return new Response(decryptedBody, options);
    });
}
```

PUTTING IT TOGETHER


New Tab
performance
http://localhost:8111/#/mailbox
Apps
For quick access, place your bookmarks here on the bookmarks bar.
Import bookmarks now...
Gmail
Images
Google
Nederland
Search Google or type URL
StartMail WebMail
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cache Storage
Console
What's New
top
Filter
Default levels

PROFIT

One more thing...

Where to store the supersecretpassword?

Making an authenticated request results in a response with the **crypto key as response header**

AS A RESPONSE HEADER WHEN ENCRYPTING

```
async function fetchAndCache(request, cacheName) {
  const response = await fetch(request.url, {
    headers: new Headers(),
    credentials: 'include'
  });

  const encryptionKey = response.headers.get(‘crypto-key');
  const encryptedResponse = await encryptResponse(response.clone(), encryptionKey);

  return caches.open(cacheName)
    .then(cache => cache.put(request, encryptedResponse.clone()))
    .then(() => response)
}
```

FOR DECRYPTING

From the page when logging in:

```
fetch(‘/api/auth', {
    method: 'POST',
    headers: {
      'content-type': ‘application/json'
    },
    body: { username, password }
})
```

Resulting in response with crypto key for decrypting

FOR DECRYPTING

From the page when checking authentication status:

```
fetch(‘/api/auth', {
    method: 'GET',
    credentials: ‘include’
})
```

Resulting in response with crypto key for decrypting

AS A REQUEST HEADER WHEN DECRYPTING

From the page:

```
fetch(‘/api/messages/12345‘, {
    method: 'GET',
    headers: {
      'content-type': ‘application/json’,
      ‘crypto-key’: cryptoKeyFromAuthResponse
    },
    credentials: 'include'
})
```

AS A REQUEST HEADER WHEN DECRYPTING

```
function tryCacheThenNetwork(request, cacheName) {
  return caches.open(cacheName)
    .then(cache => cache.match(request.url))
    .then(response => {
      if (response) {
        return decryptResponse(response, request.headers.get('crypto-key'));
      } else {
        // we could even strip out the crypto-key header,
        so it will never hit the network
        return fetchAndCache(request, cacheName);
      }
    });
}
```

We give up offline, but have control over the network

WEB CRYPTOGRAPHY API

Web Cryptography - CR

JavaScript API for performing basic cryptographic operations in web applications

Global	87.15% + 3.41% = 90.56%
unprefixed:	75.2%
Netherlands	86.99% + 7.22% = 94.21%
unprefixed:	61.75%

Current aligned | Usage relative | Date relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49			3 9.3		4.4	
	14	52	58		46	3 10.2		4.4.4	
1 11	15	54	59	3 10.1	47	3 10.3	all	56	59
	16	55	60	11	48	11			
		56	61	3 TP	49				
		57	62						
		58	63						

Notes | Known issues (0) | Resources (13) | Feedback

Many browsers support the `crypto.getRandomValues()` method, but not actual cryptography functionality under `crypto.subtle`.

Firefox also has support for unofficial features.

In Chrome the API is only usable over secure connections. (corresponding bug)

1 Support in IE11 is based an older version of the specification.

3 Supported in Safari using the `crypto.webkitSubtle` prefix.

Can I use Web Cryptography

Thanks!

Startmail is hiring!

Startmail

What do you think?

@declanrek

DE VOORHOEDE

front-end developers