Felix Gessert · fg@baqend.com

Service Workers:

The Technology Behind Progressive Web Apps

Workshop at DAHO.AM 2019
Munich, March 28

BaQend

What we are going to cover.

PWAs

- Core Features
- Building Blocks
- Implementation

Service Workers

- Lifecycle
- Network Interception
- Caching Strategies

Use Case

Learnings:
Service Workers
in Production

Why do(n‘t) we love native apps?

Great.

On **Homescreen**
In **App Stores**
Loading **Fast**
Work **Offline**
Use **Phone APIs**
Receive **Push Notifications**

Weak.

Need **Installation**
Not **Cross Platform**
Tedious **Release and Update Processes**
Maintaining **Multiple Versions**

Why do(n‘t) we love native apps?

Progressive Web Apps

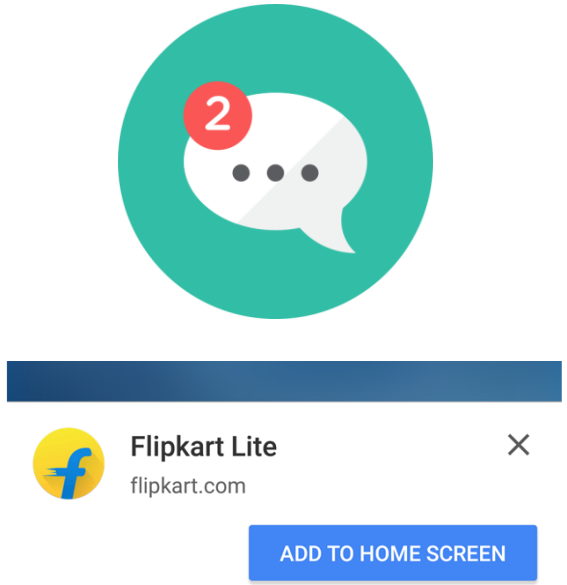
Combine the best from **native** and **web apps**.

What are Progressive Web Apps?

Progressive Web Apps (PWAs)

+

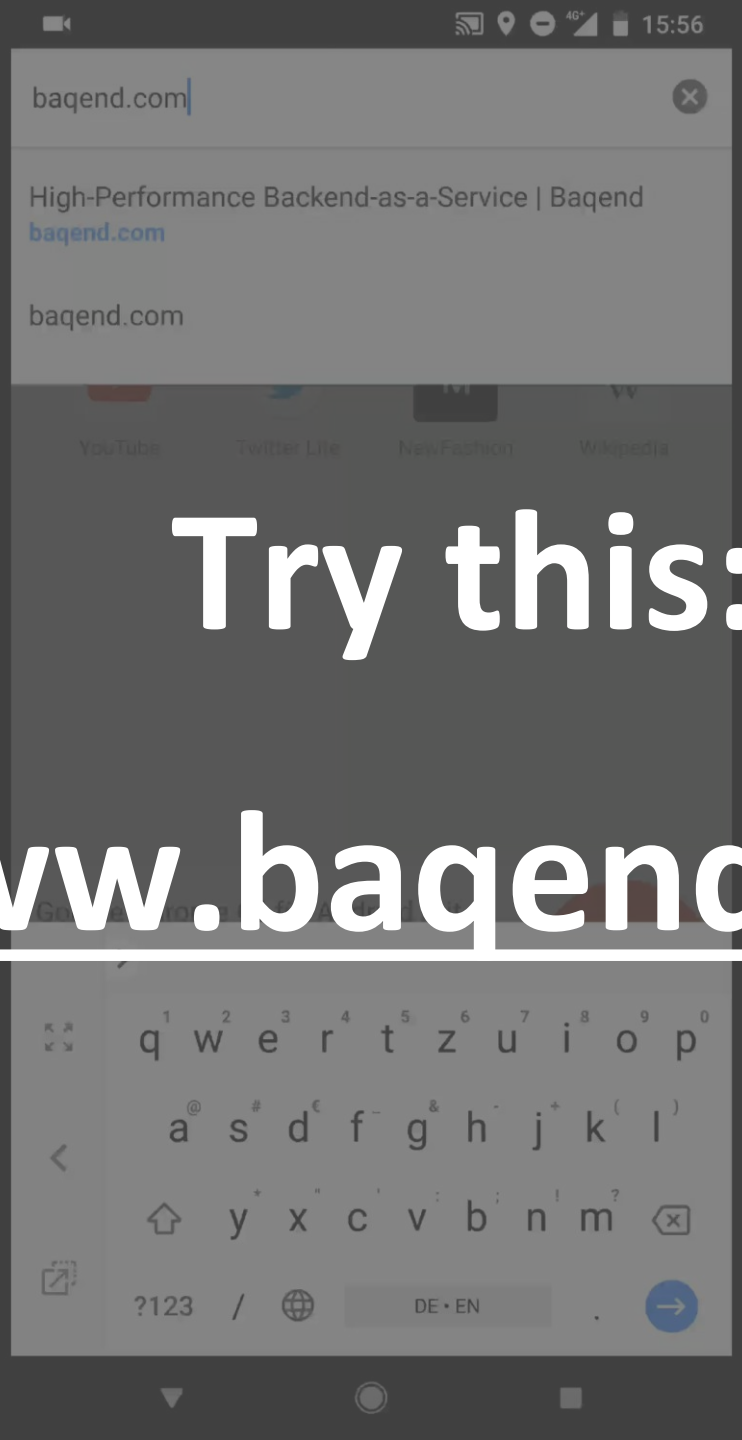
+



Fast **Loads** through Caching

Offline Mode (Synchronization)

Add-to-**Homescreen** and **Push Notifations**



Try this:

www.baqend.com

Advantages of PWAs

Discoverable

E.g. in search engines

Installable

Easy access from home screen

Linkable

Link into apps through URLs

Network independant

Offline mode

Progressive

Enhance on capable browsers

Re-engageable

Engage through Web Push

Responsive

Fit any form factor

Safe

HTTPS & recognizable URLs

https://developer.mozilla.org/en-US/docs/Web/Apps/Progressive/Advantages

> These apps aren’t packaged and deployed through stores, they’re just websites that took all the right vitamins.

Alex Russell, Google

Building Blocks of PWAs

PWAs are **best practices** and **open web standards**

Progessively enhance when supported

1. Manifest

2. Service Worker

Implementing PWAs

PWAs are **best practices** and **open web standards**

Progessively enhance when supported

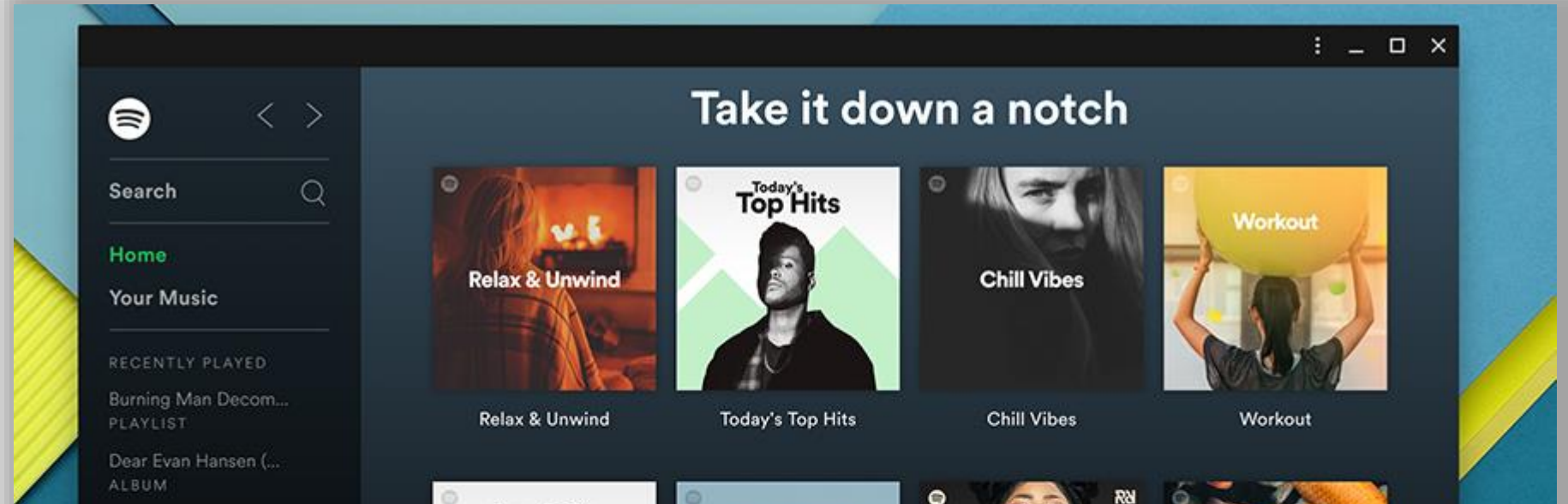
1. Manifest declares Add-to-Homescreen:

```
<link rel="manifest" href="/manifest.json">
{
  "short_name": „DAHO.AM PWA",
  "icons": [
    {"src": "icon-1x.png", "type": "image/png", "sizes": "48x48"}],
  "start_url": "index.html?launcher=true"
}
```

Just Released: Desktop PWAs

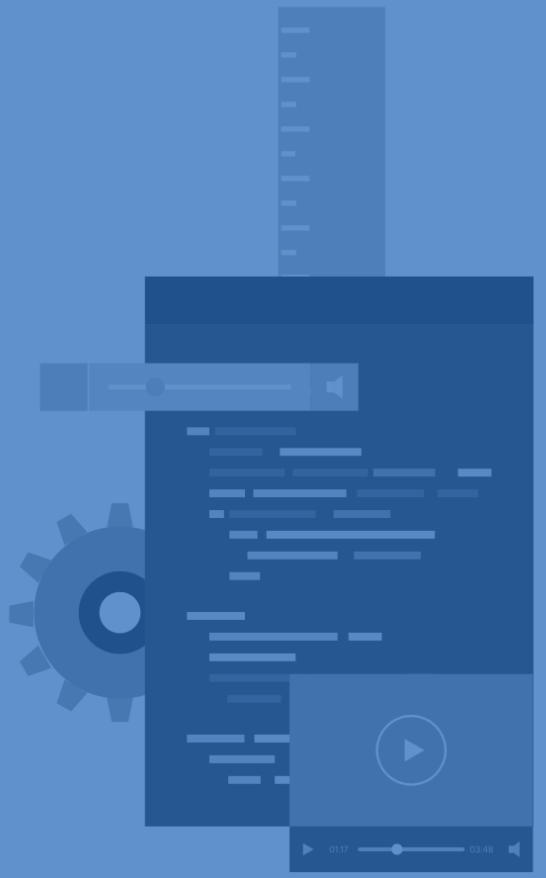
Chrome >73 now supports **Desktop PWAs** on every platform

Customizable installation process/UI (with Event `beforeinstallprompt`)



DEMO I

The Web App Manifest in Action

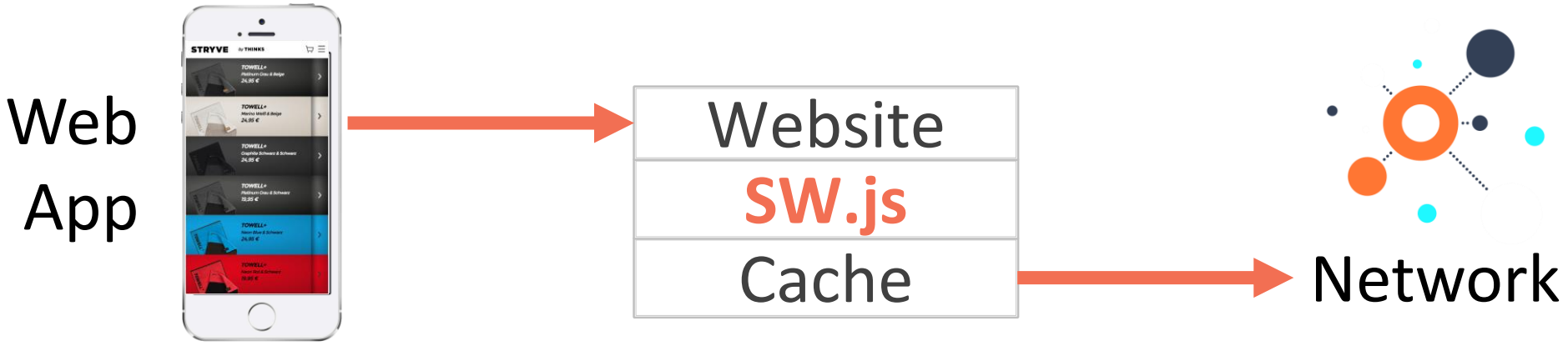


Implementing PWAs

PWAs are **best practices** and **open web standards**

Gracefully degrade when not supported

2. Service Workers for caching & offline mode:

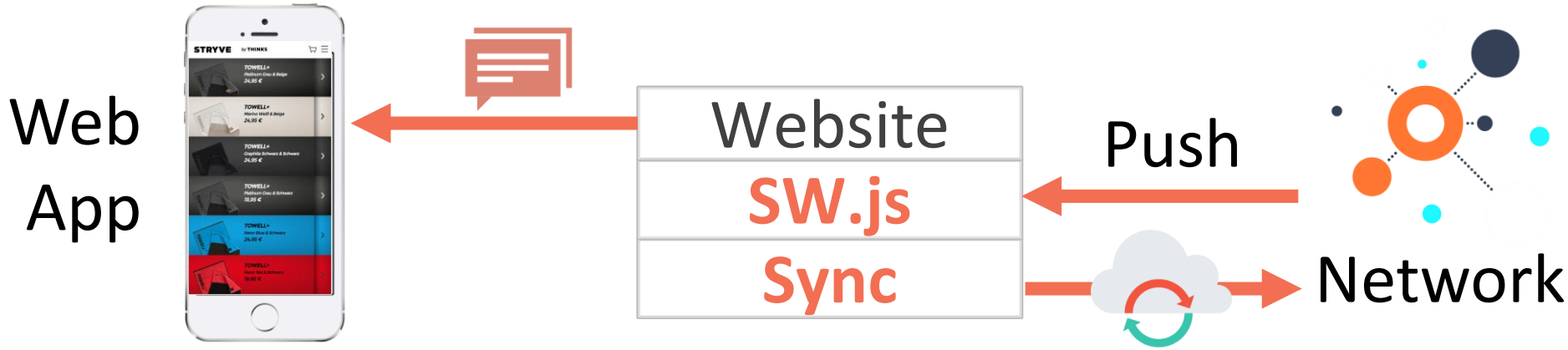


Implementing PWAs

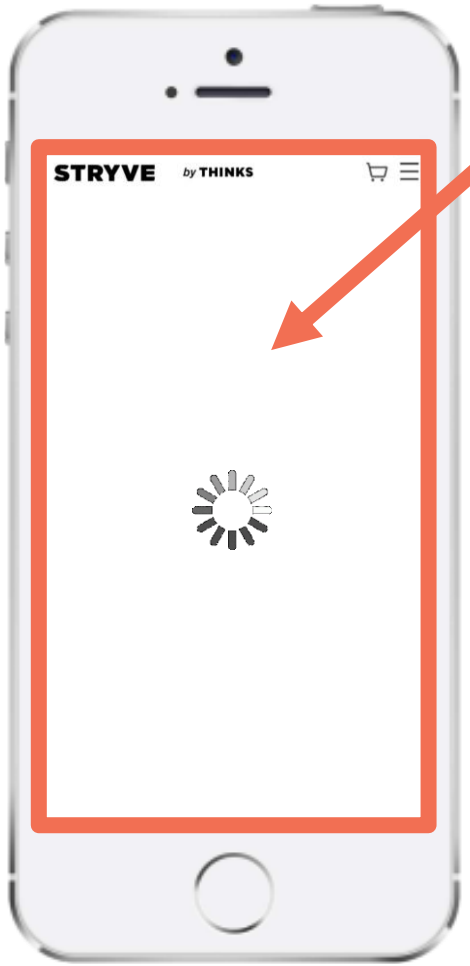
PWAs are **best practices** and **open web standards**

Progressively enhance the user experience

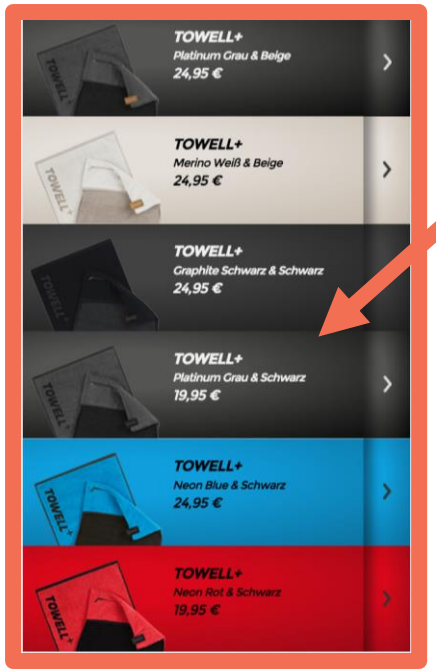
3. Add Web Push and Background Sync:



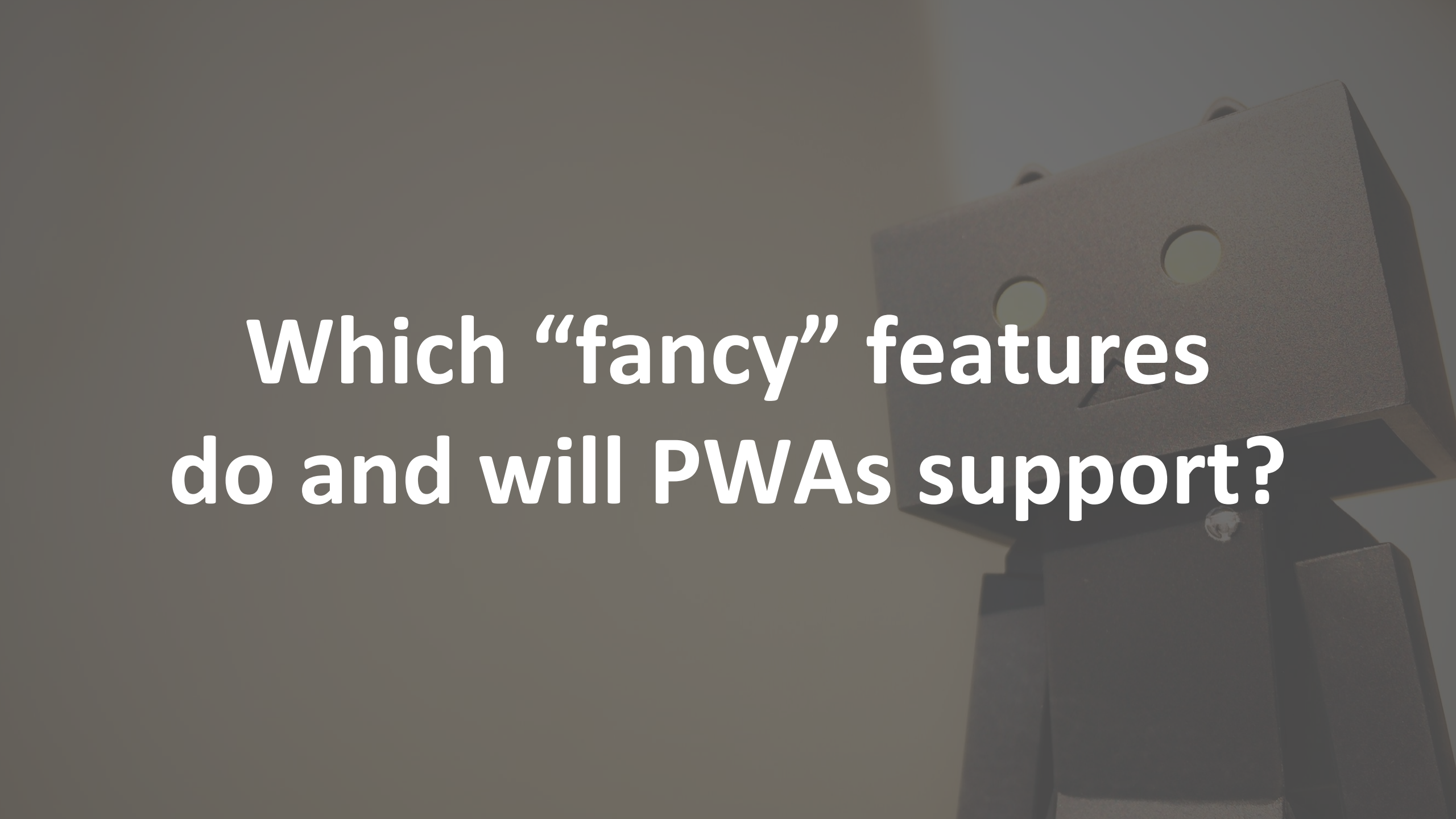
Typical Architecture: App Shell Model



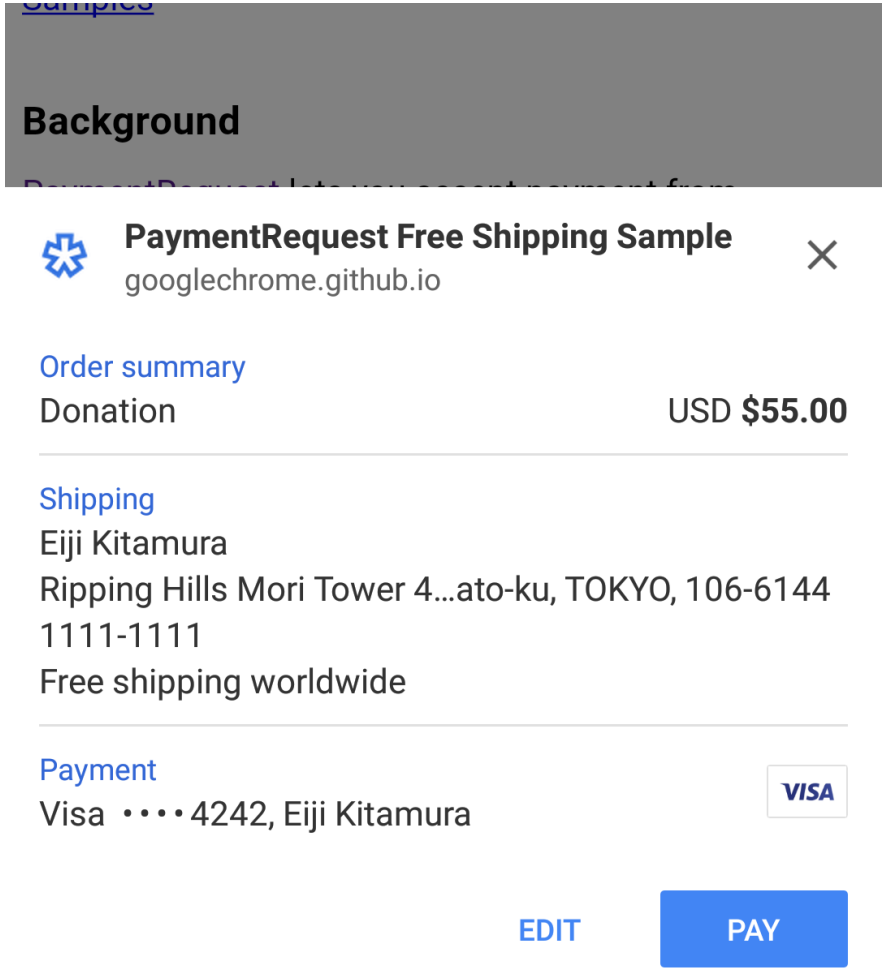
App Shell: **HTML**, **JS**, **CSS, images** with app logic & layout



Content: Fetched on demand & may change more often


Which “fancy” features
do and will PWAs support?

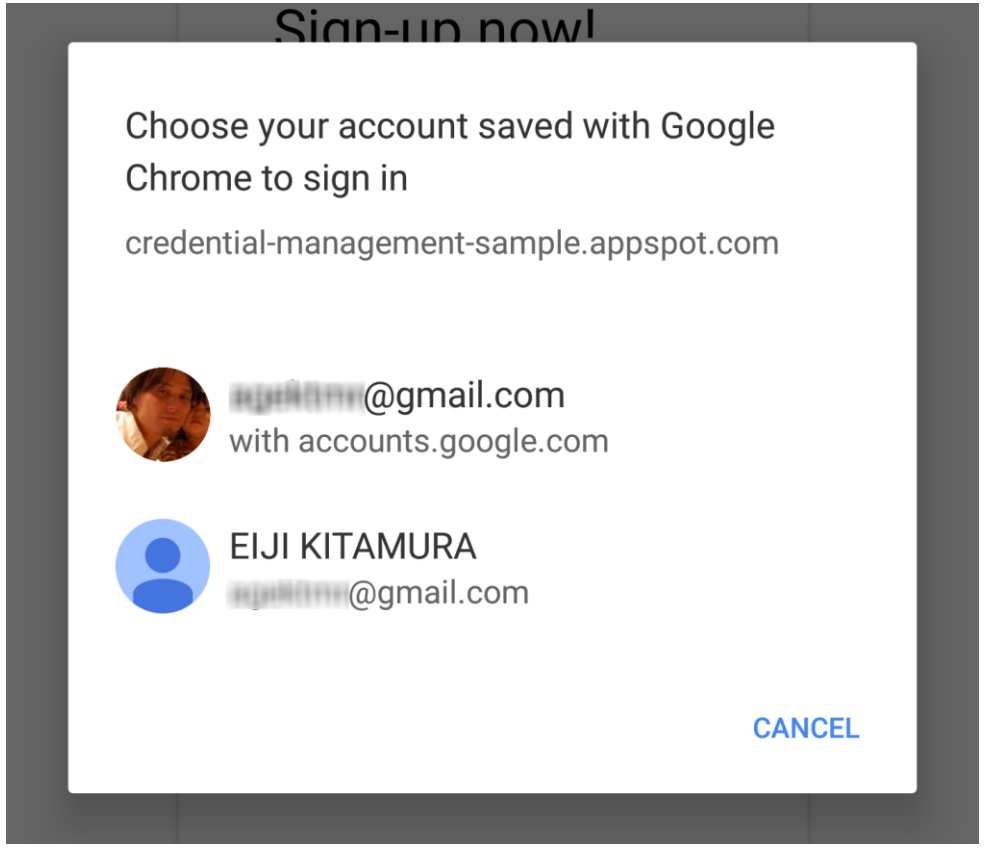
Integrate payment.



Web Payment APIs

- Goal: replace traditional **checkout** forms
- Just ~10 LOC to implement **payment**
- Vendor- & Browser-**Agnostic**

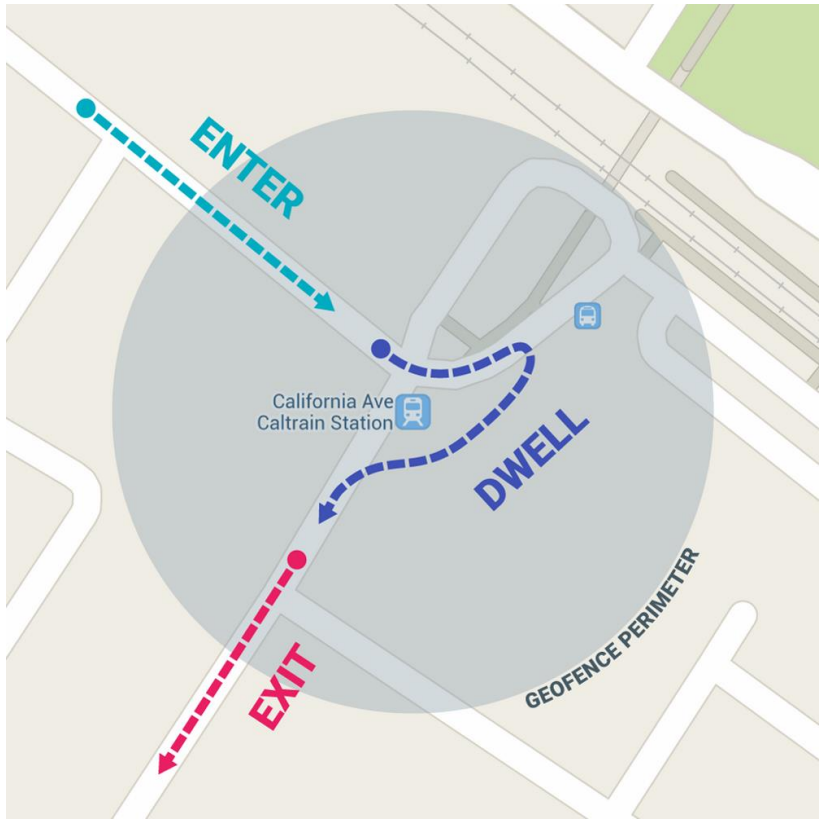
Manage users and logins.



Credentials Management API

1. Click **Sign-in** → Native Account Chooser
2. Credentials API **stores** information for future use
3. **Automatic** Sign-in afterwards

Leverage geolocation.



Geofencing

- **Notify** web app when user leaves or enters a defined area
- Requires **permission**

Build conversational interfaces.

Web Speech API

Native Speech Recognition in the Browser:

```
annyang.addCommands({
  'Hello DAHO.AM': () => {
    console.log('Hello you.');
  }
});
```

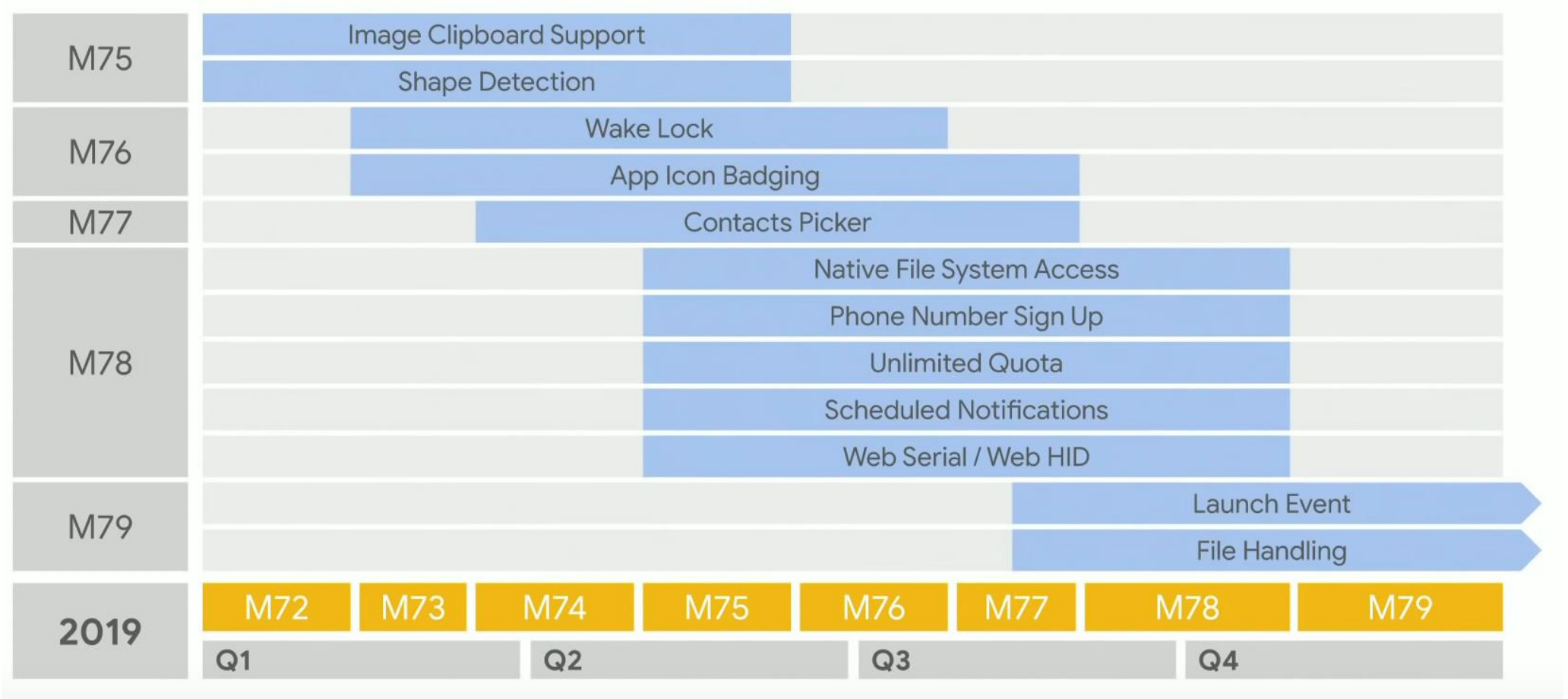
Seemless sharing between apps.

Web Share API

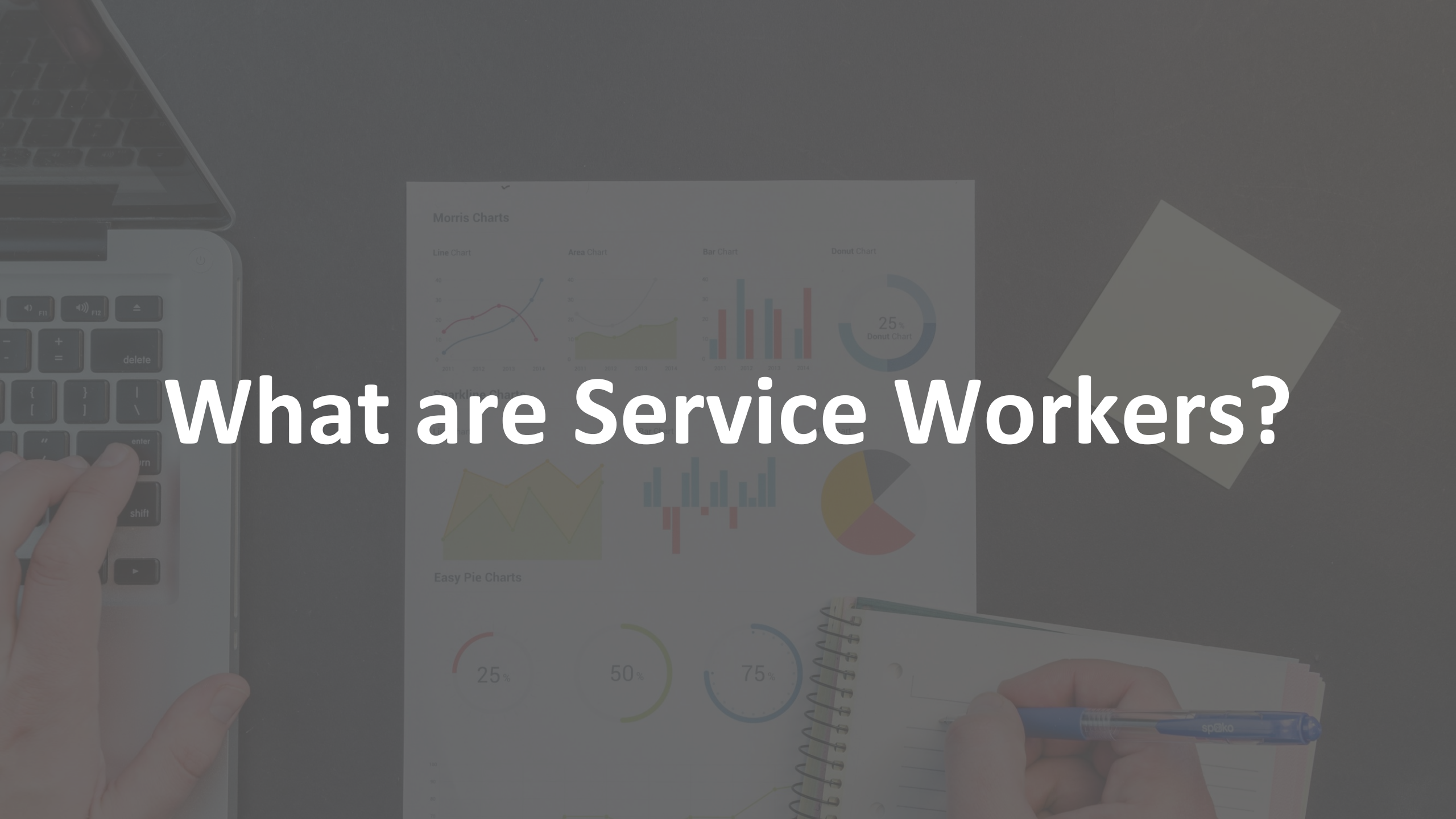
- **Share** site through native share sheet UI
- Service Worker can register as a **Share Target**

And many more in the pipeline.

Planned Shipping Dates of Chrome PWA Features.

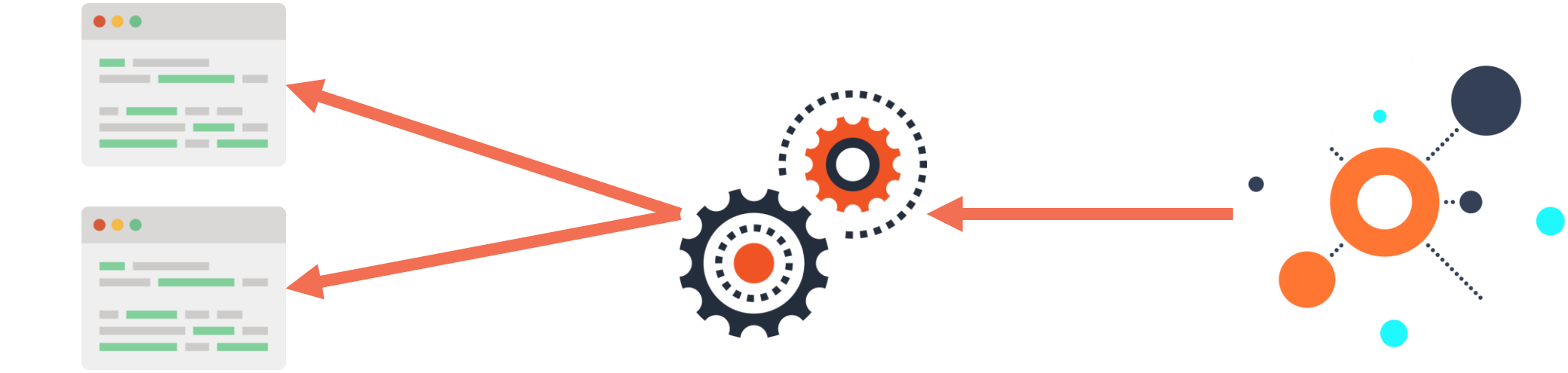


https://www.youtube.com/watch?v=2KhRmFHLuhE (Google I/O 2019)



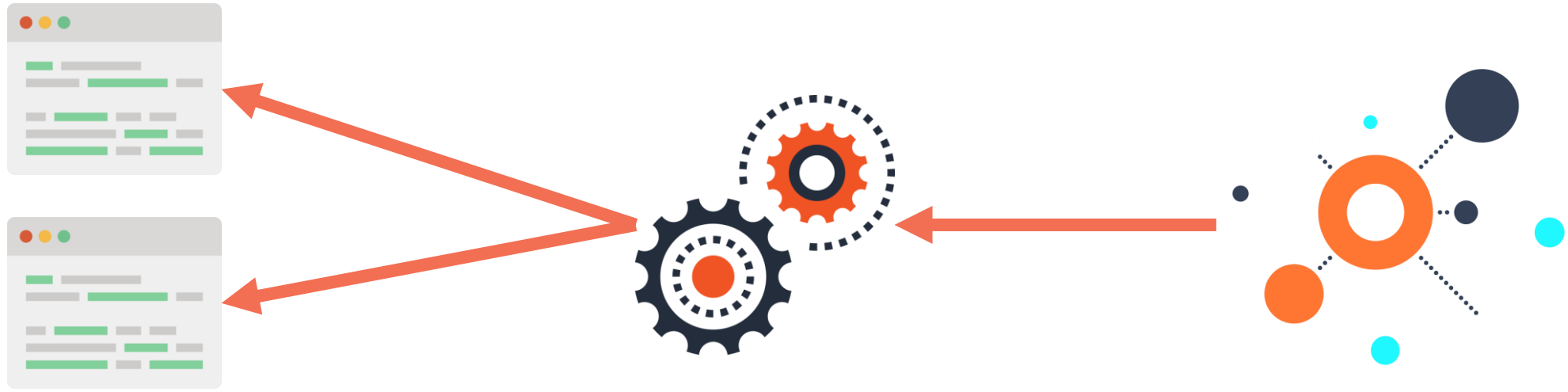
What are Service Workers?

What are Service Workers?



Programmable **Network Proxy**, running as a separate **Background Process**, without any **DOM Access**.

What do Service Workers do?



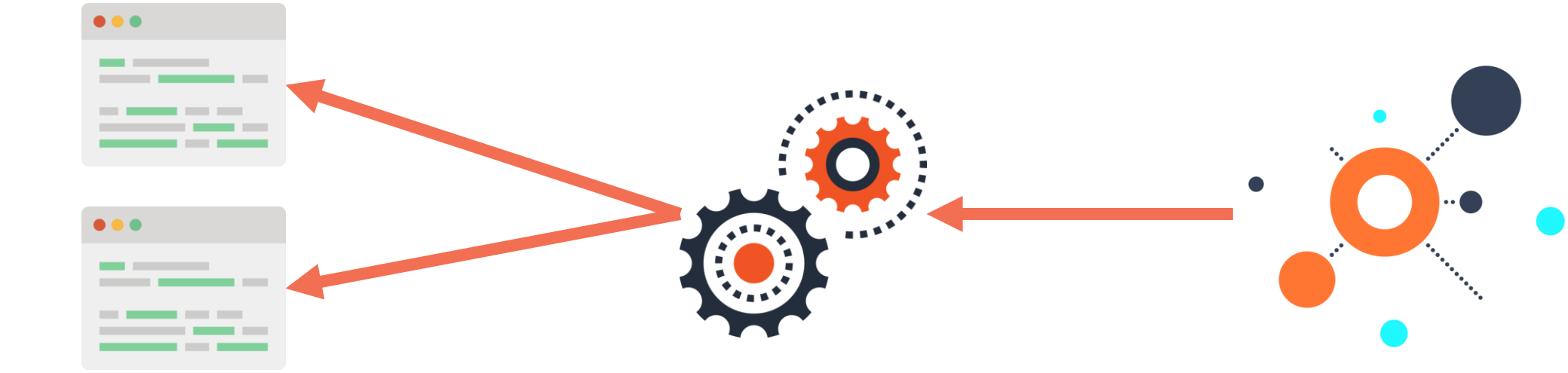
Browser Tabs

Service Worker

Network

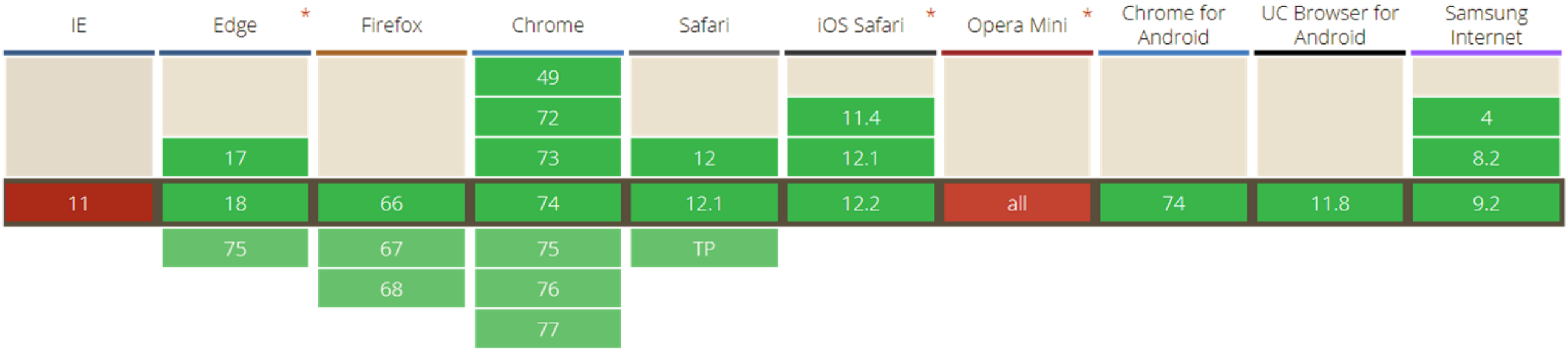
- Cache Data (CacheStorage)
- Store Data (IndexedDB)
- Receive Push
- Respond when Offline

What do Service Workers do?



- **Intercept** HTTP Requests
- **Sync** Data in Background
- Hide **Flaky Connectivity** from the User

Browser Support for Service Workers



Supported by **>90%** of browsers.

Requires **TLS Encryption**.

Late, but all in: Microsoft

Publish PWAs to
Microsoft Store

PWA BUILDER

or

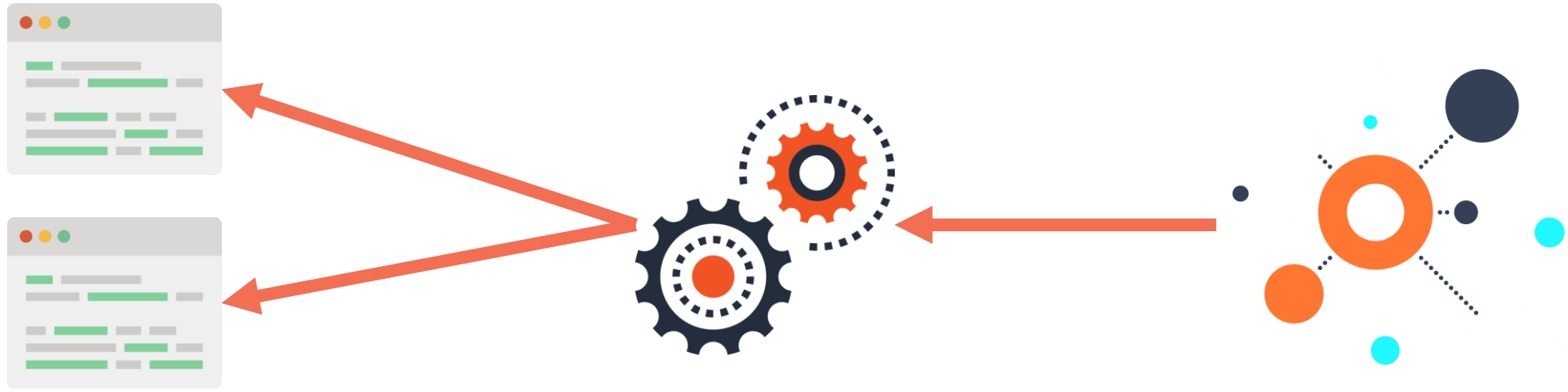
Bing Crawls PWAs

Convert to **AppX**

Microsoft **Store**

https://blogs.windows.com/msedgedev/2018/02/06/welcoming-progressive-web-apps-edge-windows-10/#tqIAYGJrOUcxvCWg.97

How are Service Workers registered?



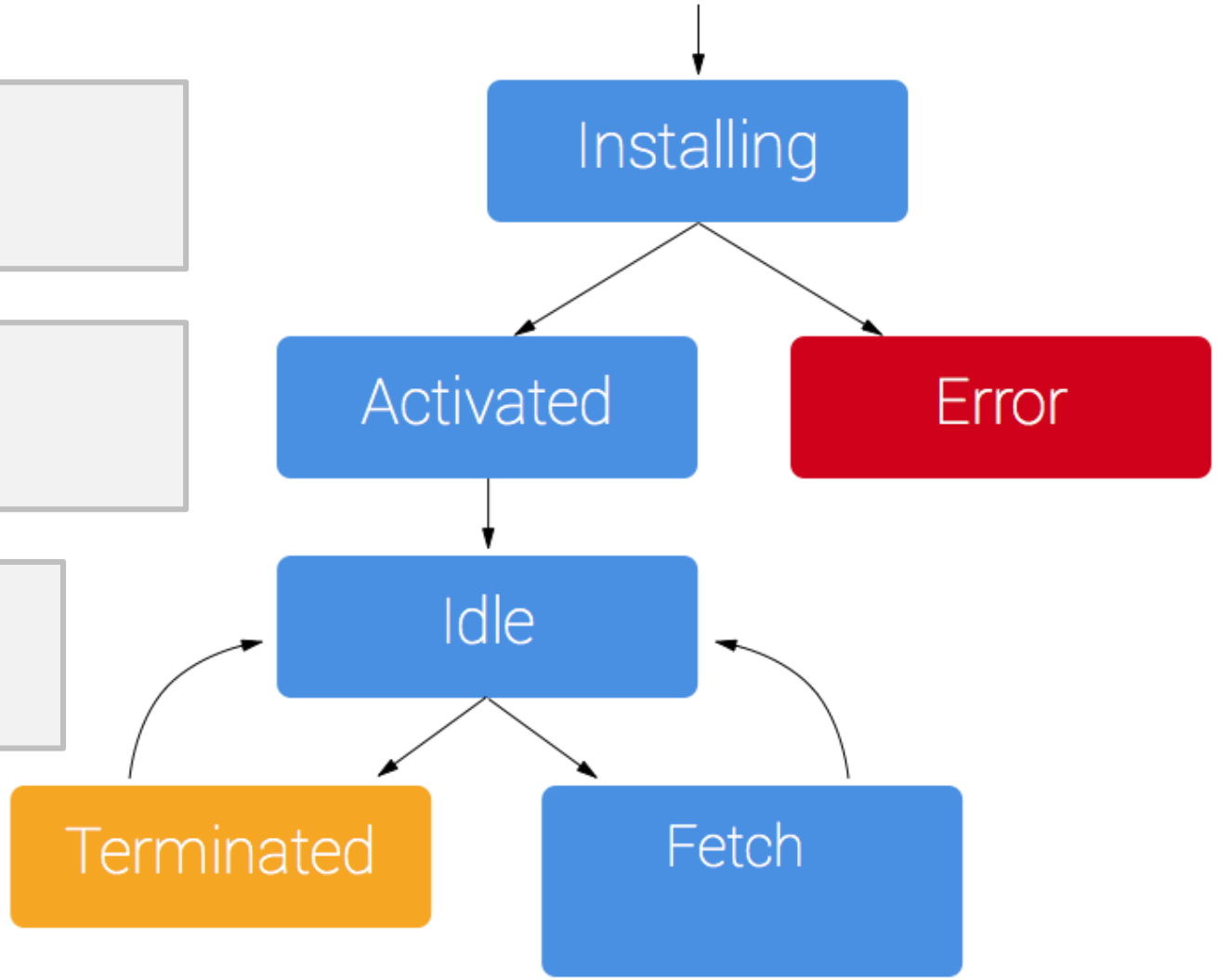

```
<script>
    navigator.serviceWorker.register('/sw.js');
</script>
```

What does the lifecycle look like?

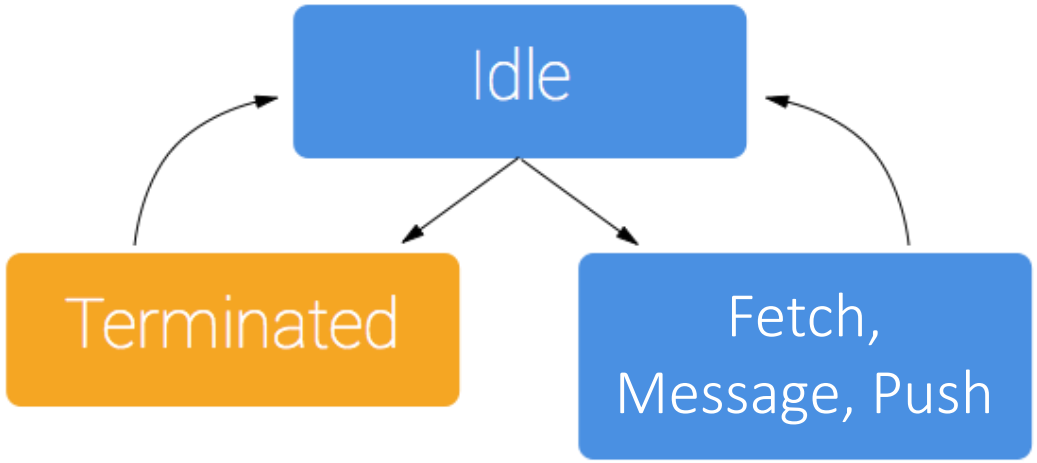
```
self.addEventListener('install', (event) => {
  // Perform install steps
});
```

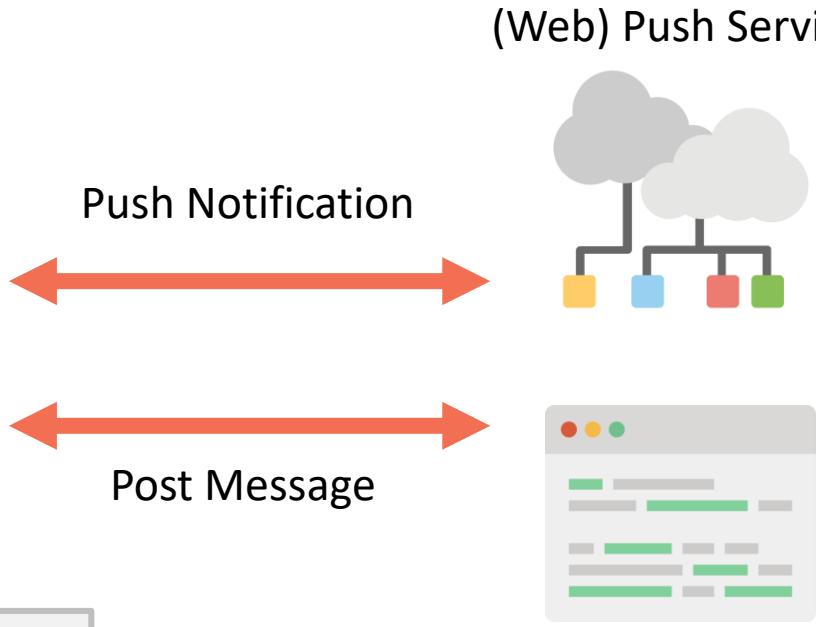
```
self.addEventListener('activate', (event) => {
  // Perform activate steps
});
```

```
self.addEventListener('fetch', (event) => {
  // React to fetch event
});
```



How to communicate with Service Workers?






```
self.addEventListener('message', (event) => {
  // Receive message
});
```

```
// Send message to browser tab
const client = await clients.get('id');
client.postMessage(someJsonData);
```

```
self.addEventListener('push', (event) => {
  // Receive push notification
});
```

Intercepting Network Requests

```
self.addEventListener('fetch', (event) => {
  // React to fetch event
  const { url } = event.request;
  event.respondWith((async () => {
    const request = new Request(url.replace('.com', '.de'))
    const response = await fetch(request);
    const text = await response.text();
    const newText = text.replace('Goethe', 'Schiller');
    return new Response(newText, { status: 200 });
  })());
});
```

There is so much you can do:

- **Rewrite** Requests
- **Change** Responses
- **Concat** Responses
- **Cache** Responses
- **Serve** Cached Data
- …

Service Worker Scope

Scope determines which requests go to the Service Worker

```
// Default (and maximum) scope is location of Service Worker
// Gets all requests starting with '/path/'
navigator.serviceWorker.register('/path/sw.js');
```

Service Worker Scope

Scope can be **restricted** but **not widened**

```
// Scope option can further limit which requests got to Service Worker
// Gets all requests starting with '/path/subpath/'
navigator.serviceWorker.register('/path/sw.js', { scope: '/path/subpath/' });
```

Service Worker Persistence

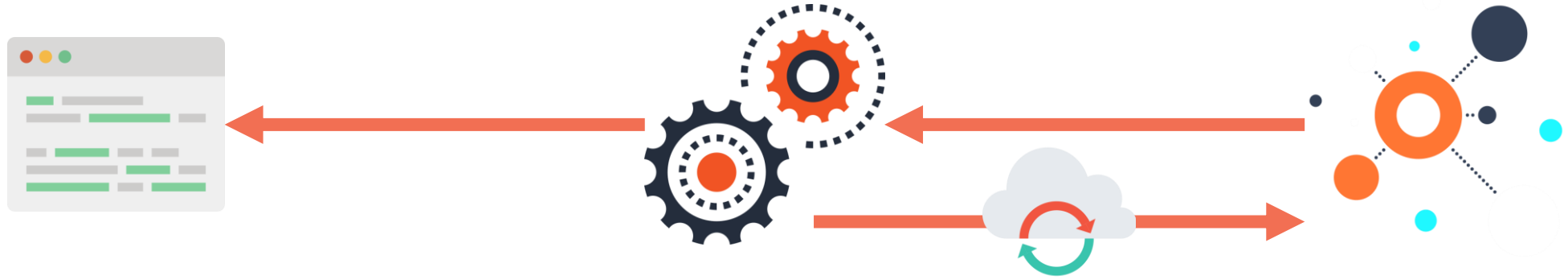
IndexedDB

an actual database in the browser

- Stores Data **Persistently**
- Stores **Structured** Data
- Supports **Range Queries**
- **Browser Support** 94%

Experimental

Service Worker Background Sync



One-off Sync

- executed when user is **online**
- **retried** when failed (exponential backoff)

Use Cases

- Save **file** when online again
- Send **email** when online again

Periodic Sync

- executed when online, according to **period options**

Use Cases

- Load updates to **social media time-line** when browser closed

Service Worker Debugging

Elements Console Sources Network Performance Memory Application Security Audits

Application
- Manifest
- Service Workers
- Clear storage

Storage
- Local Storage
- Session Storage
- IndexedDB
- Web SQL
- Cookies

Cache
- Cache Storage
- Application Cache

Service Workers

Offline Update on reload Bypass for network Show all

www.baqend.com Update Unregister

Source sw.7dbf553e.js

Received 23.1.2018, 15:38:40

Status #823 activated and is running stop

Clients https://www.baqend.com/ focus

Push Test push message from DevTools. Push

Sync test-tag-from-devtools Sync

Elements Console Sources Network Performance Memory Application Security Audits

Network Filesystem

sw.7dbf553e.js:formatted

top
- www.baqend.com
  - css
  - fonts
  - img
  - js
    - (index)
    - manifest.293e2c44.json
- Wappalyzer
- connect.facebook.net
- embed.tawk.to
- fonts.googleapis.com
- fonts.gstatic.com
- www.google-analytics.com
- FraD6fj-1516730773916 (abo
- XkVhsaD-1516730773915 (ab
- ZNEaMt3-1516730773916 (ab
- v3

sw.7dbf553e.js

```
14
15  var a = {};
16  return t.m = e,
17  t.c = a,
18  t.d = function(e, a, s) {
19      t.o(e, a) || Object.defineProperty(e, a, {
20          configurable: false,
21          enumerable: true,
22          get: s
23      })
24  }
25  ,
26  t.n = function(e) {
27      var a = e && e.__esModule ? function() {
28          return e['default']
29      }
30      : function() {
31          return e
32      }
33      ;
34      return t.d(a, 'a', a),
35      a
36  }
37  ,
38  t.o = function(e, t) {
39      return Object.prototype.hasOwnProperty.call(e, t)
40  }
41  ,
42  t.p = '',
43  t(t.s = 13)
44  }
45  )([function(e, t, a) {
46      'use strict';
47      t.b = function(e) {
48
```

Line 32, Column 9

DEMO II

Looking into Service Workers In Chrome Dev Tools

Service Worker Caching

Cache Storage

Stores Request/Response pairs

Cache Storage

- **Programmatically** managed
- **Persistent** and non-expiring

- Supports only **HTTP**
- Only caches **GET** requests (no HEAD)

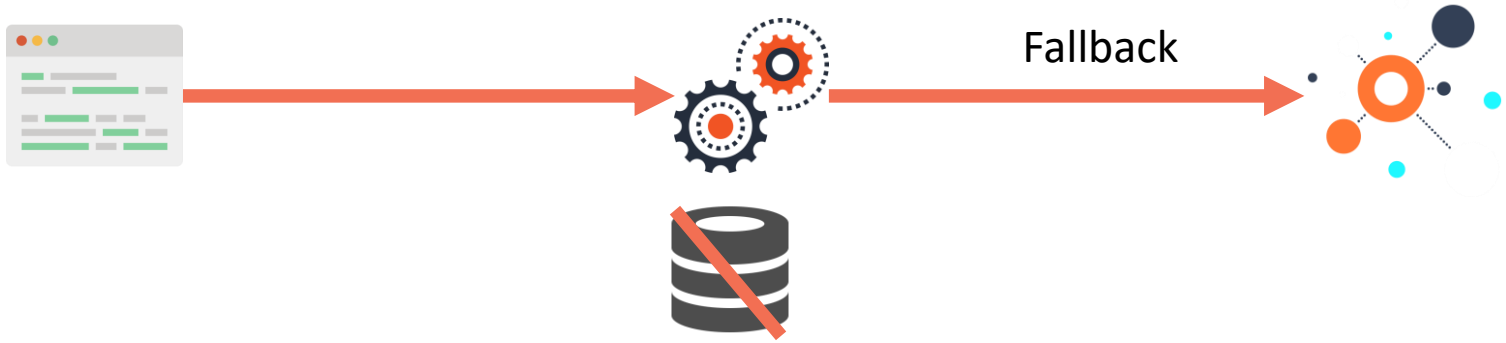
Caching Strategies – Cache Only

Gets all requests from cache or fails.

Caching Strategies – Cache, Network Fallback

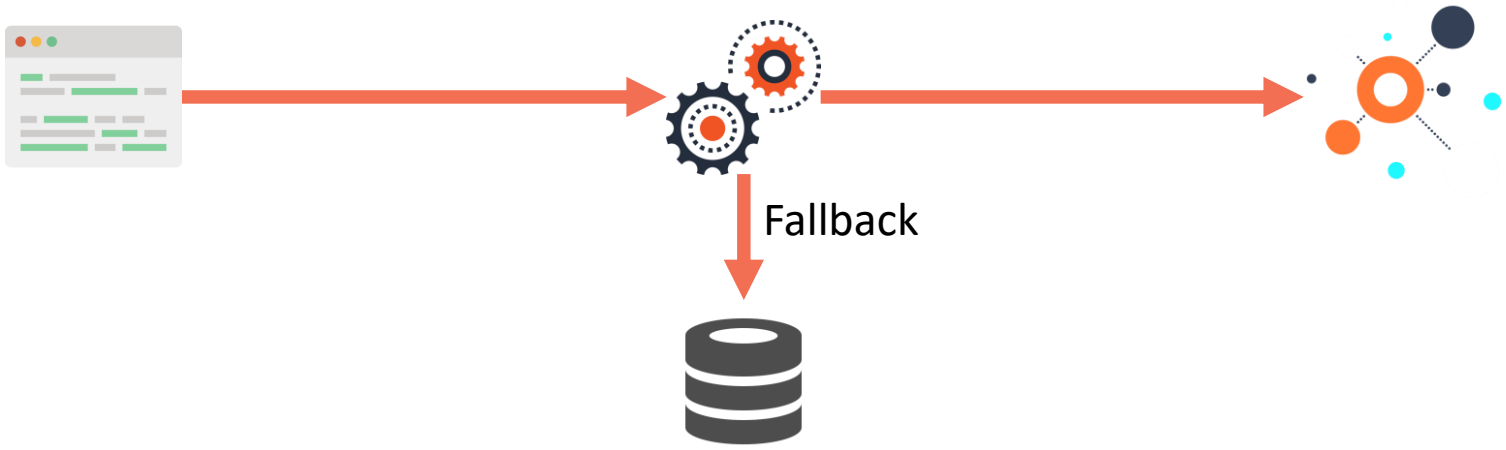
Gets requests from cache & uses network as fallback.

Caching Strategies – Network Only



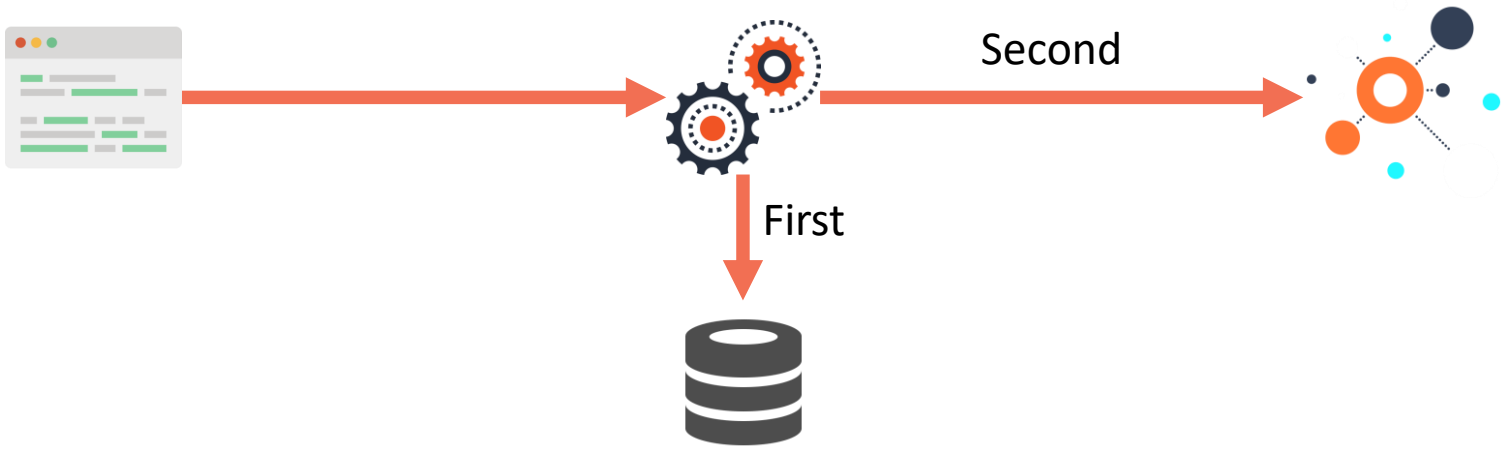
Gets requests from network only.

Caching Strategies – Network, Cache Fallback

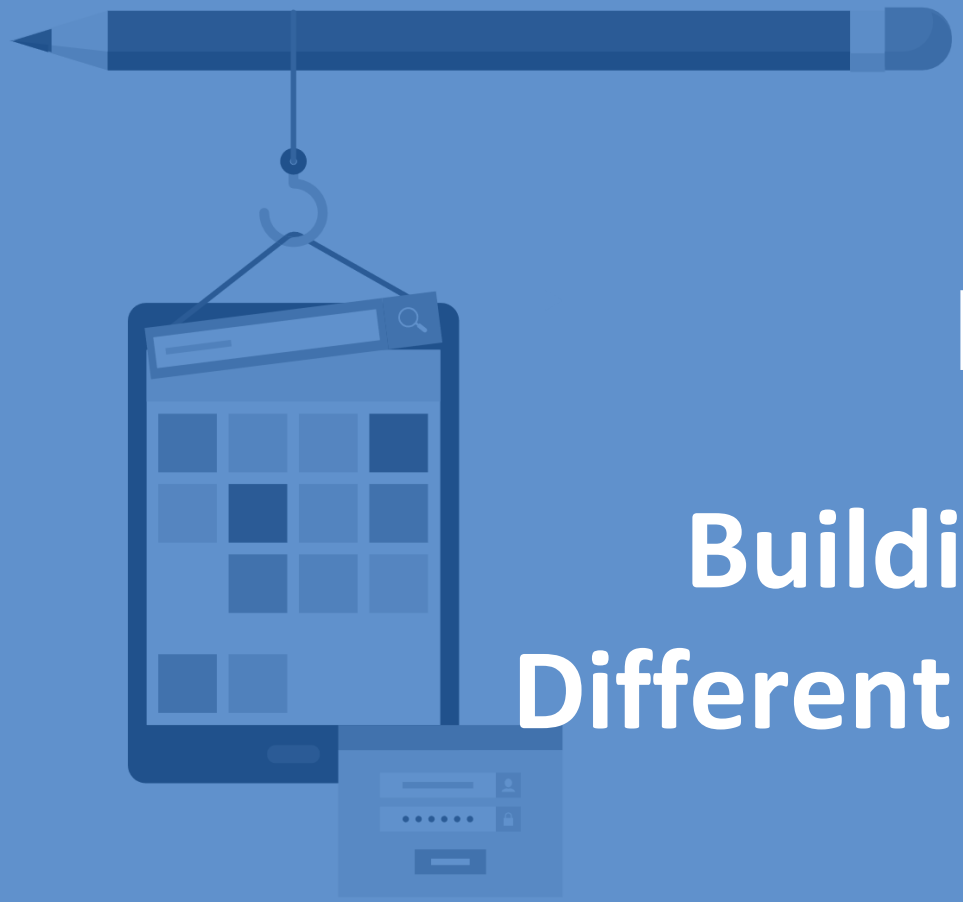


Gets requests from network, the cache acts as fallback (offline mode).

Caching Strategies – Cache, then Network



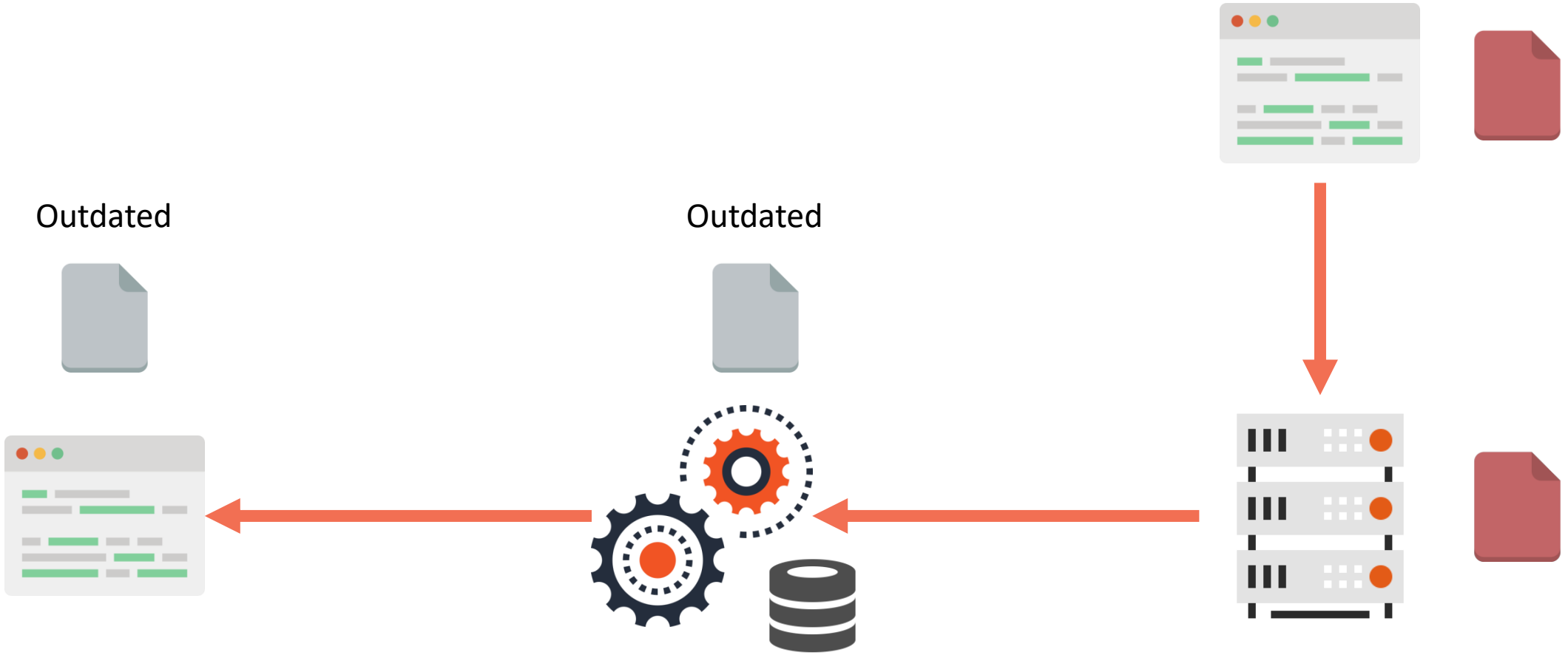
Gets requests from cache first and from network in background.



DEMO III

Building a PWA With Different Caching Strategies

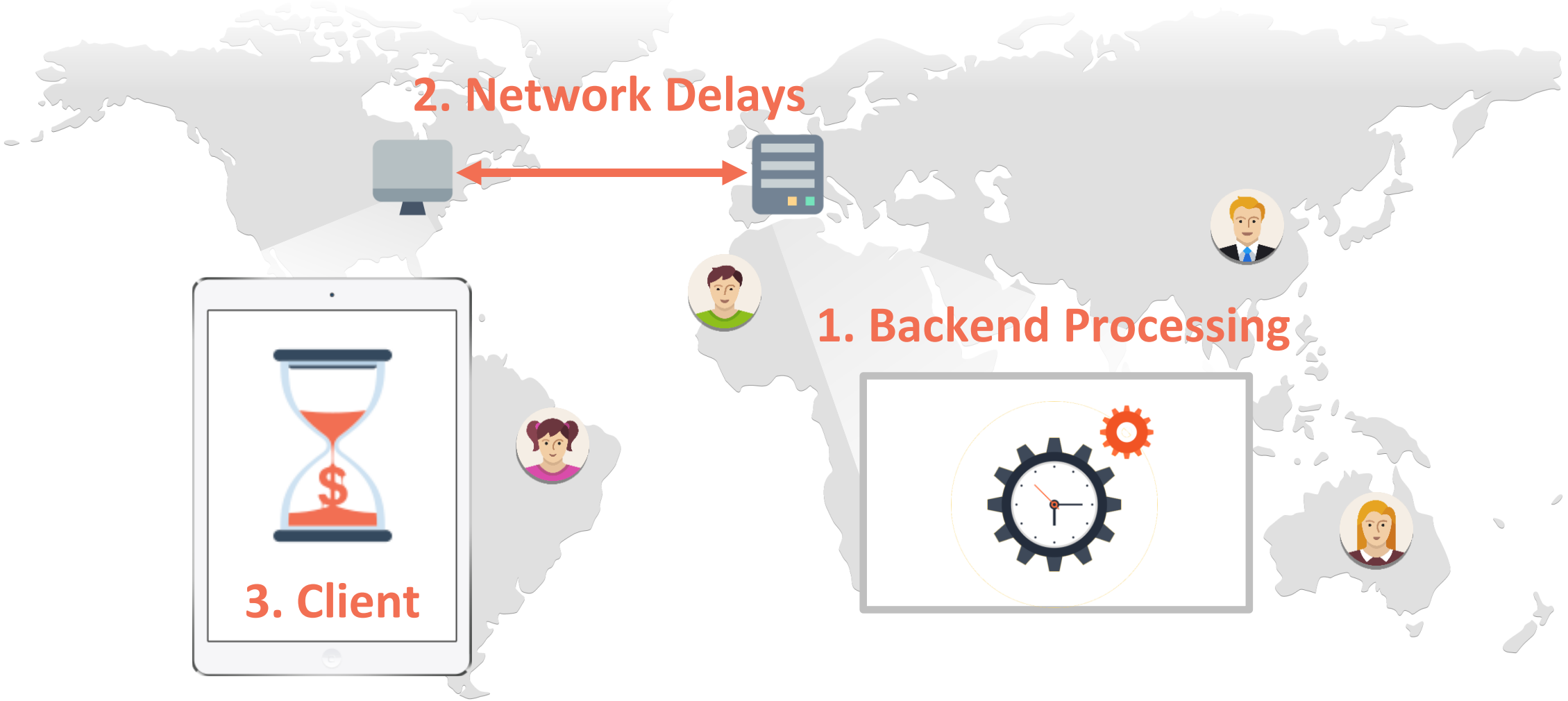
Major Challenge: Cache Coherence



All strategies either serve **outdated data** or **degrade performance**

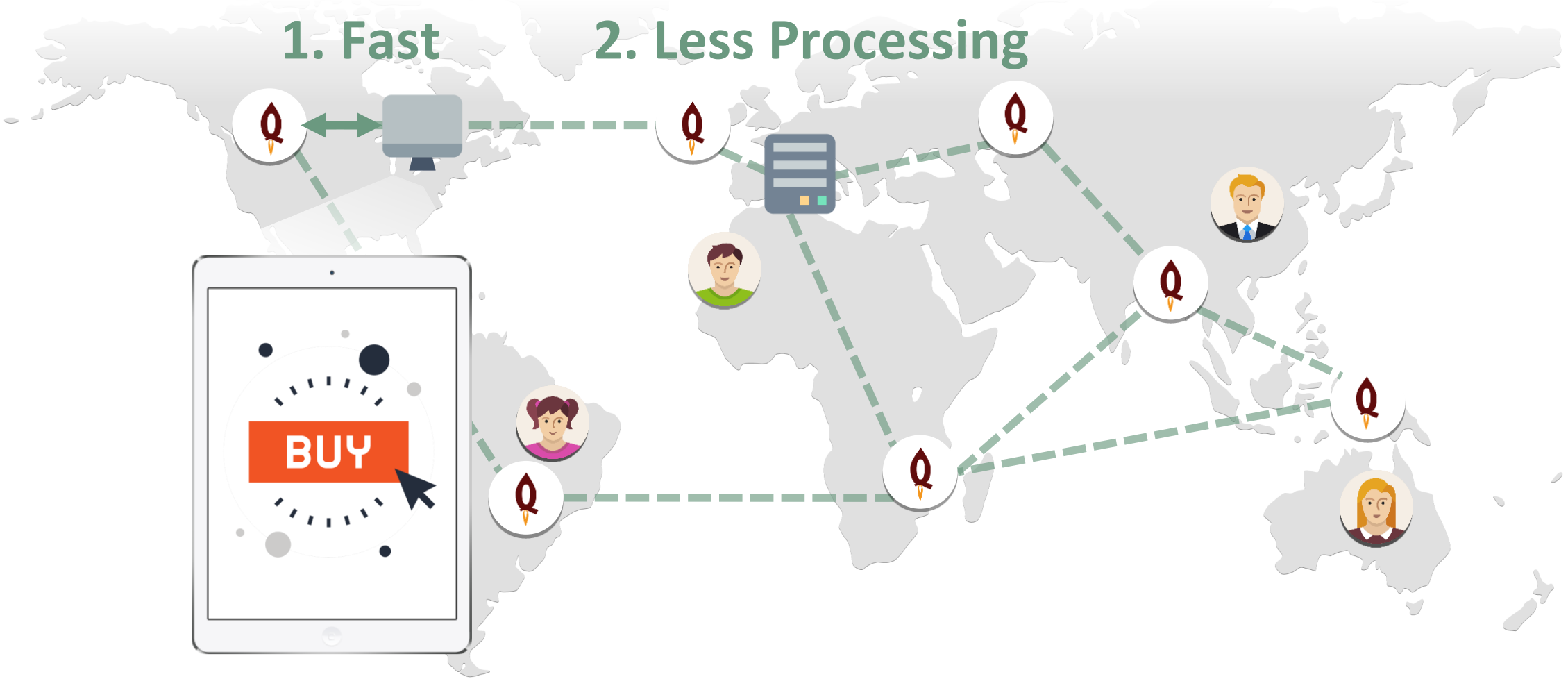
How we use Service Workers at Baqend

Three things slow web apps down.



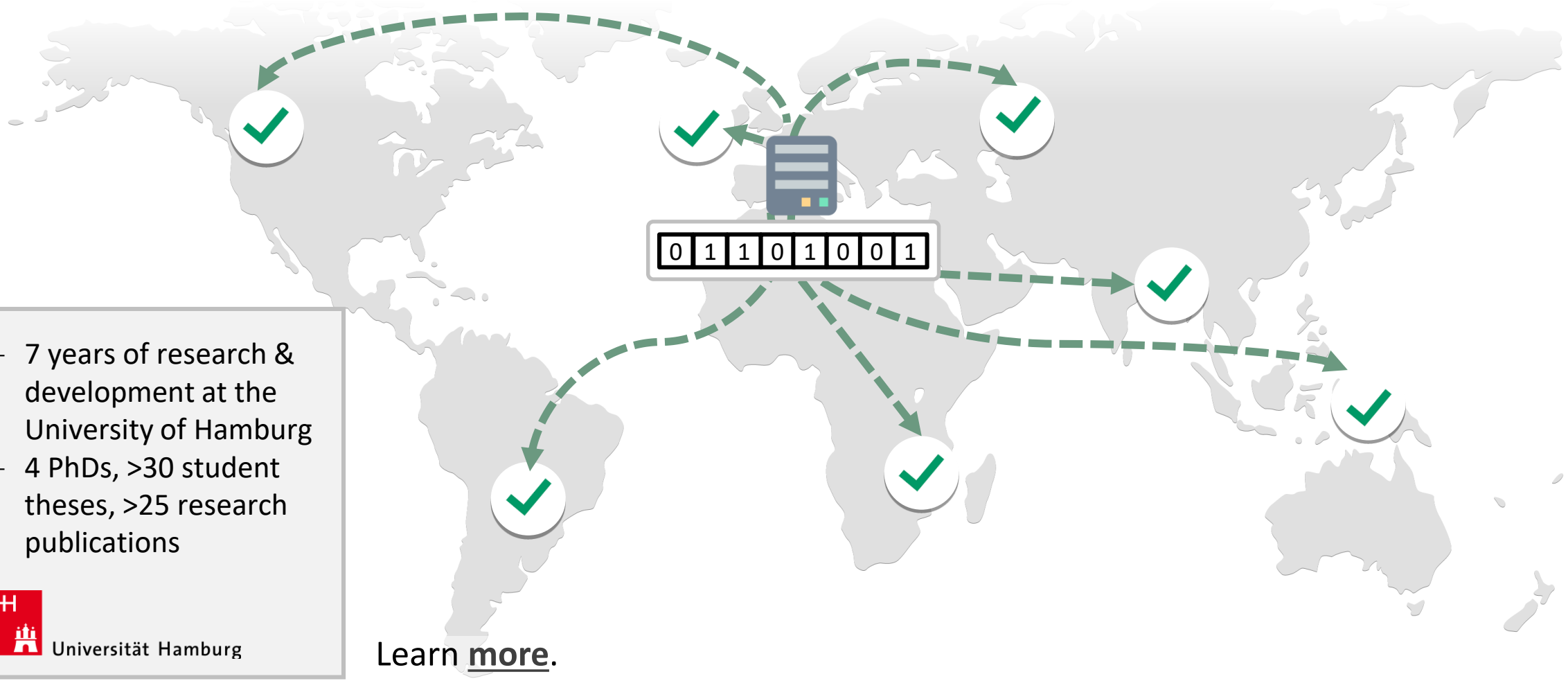
Solution: Speed Kit

Service Worker rewrites & accelerates slow requests.



The magic: dynamic data is kept up-to-date.

Backed by 30 man-years of research.

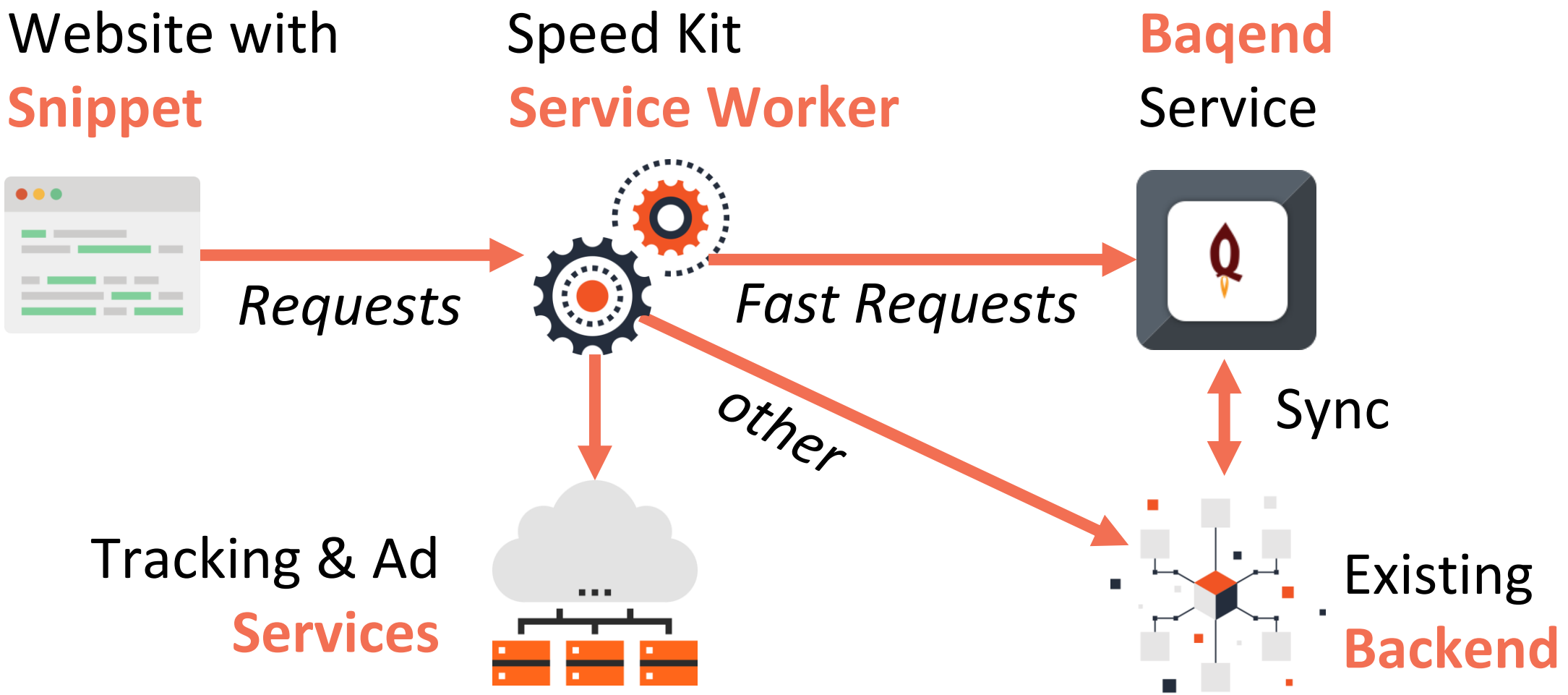


- 7 years of research & development at the University of Hamburg
- 4 PhDs, >30 student theses, >25 research publications

Universität Hamburg

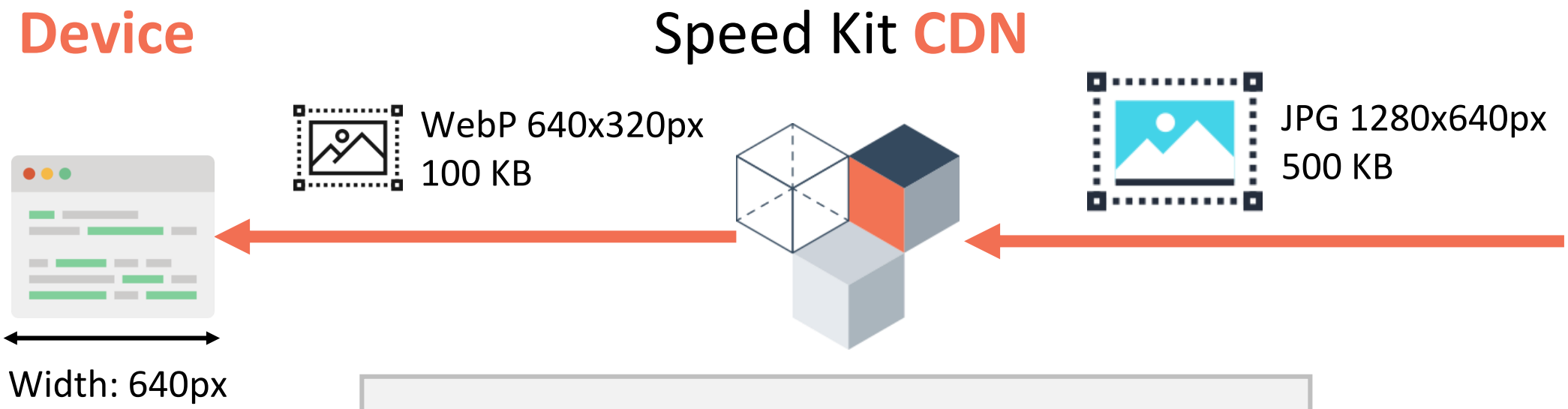
Learn more.

How Speed Kit leverages Service Workers.



Use case I: optimize images.

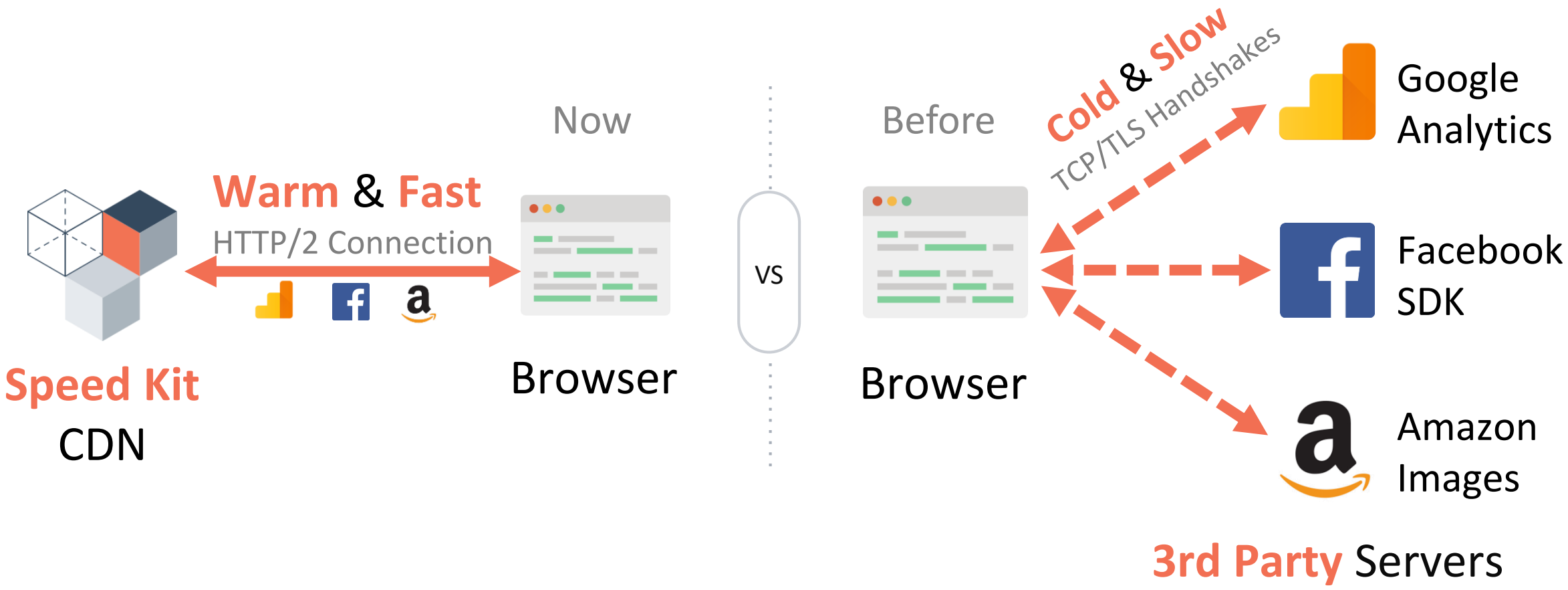
SW sends client resolution → responsive image.



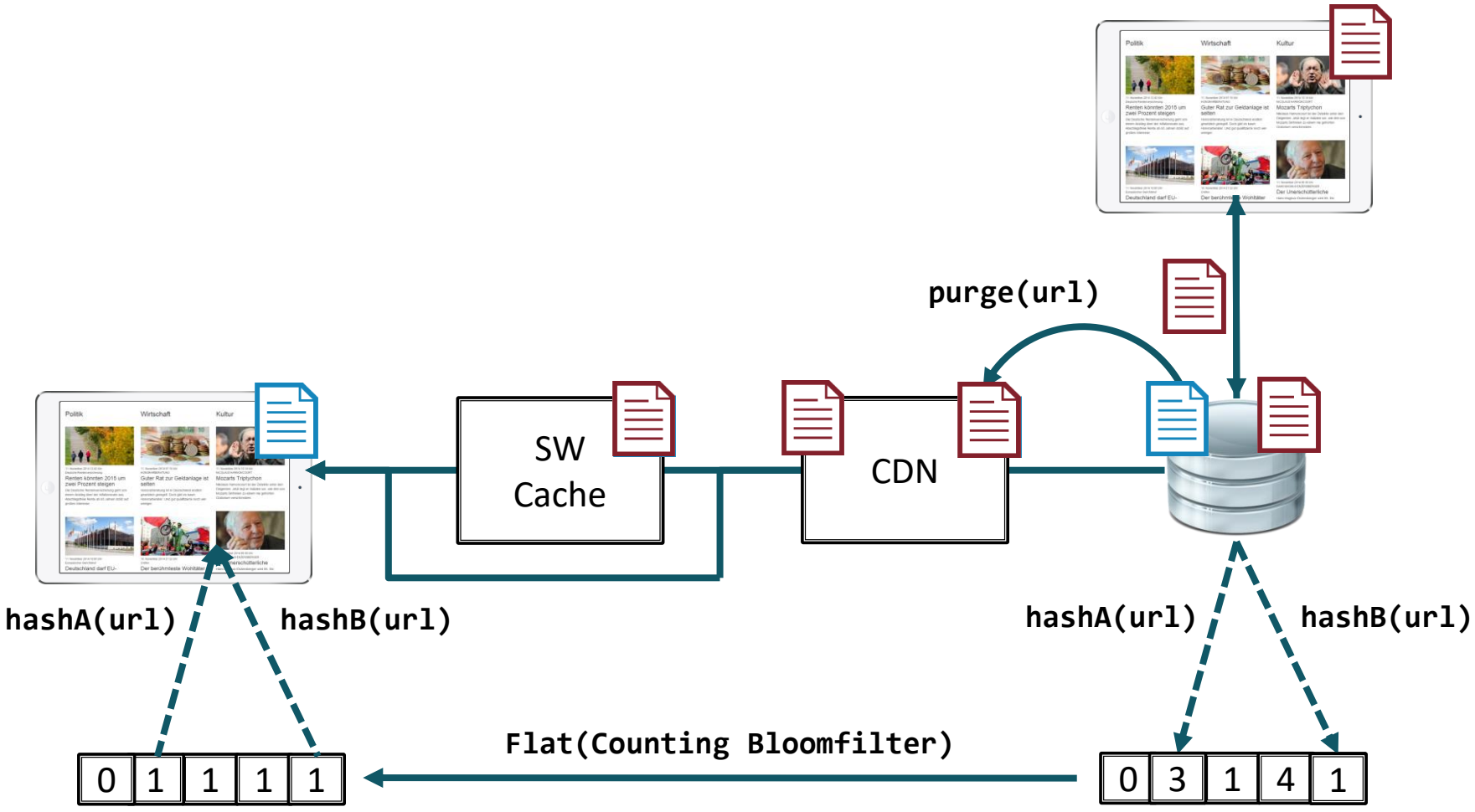
- Images transcoded to **WebP**
- Rescaled to match **Screen Size**
- JPG and PNG **Recompression**

Use case II: re-route 3rd party dependencies.

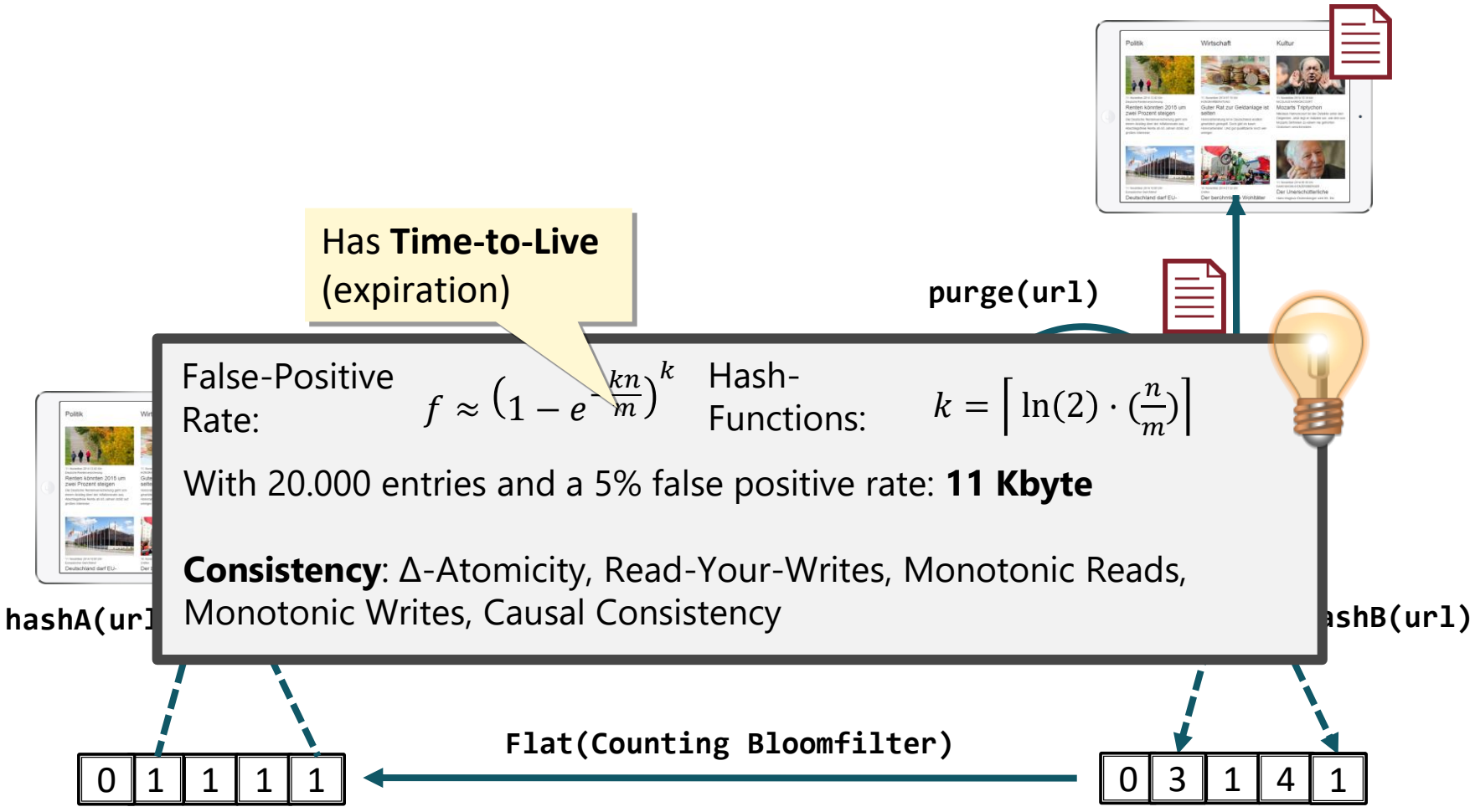
Service Workers can manipulate other domains.

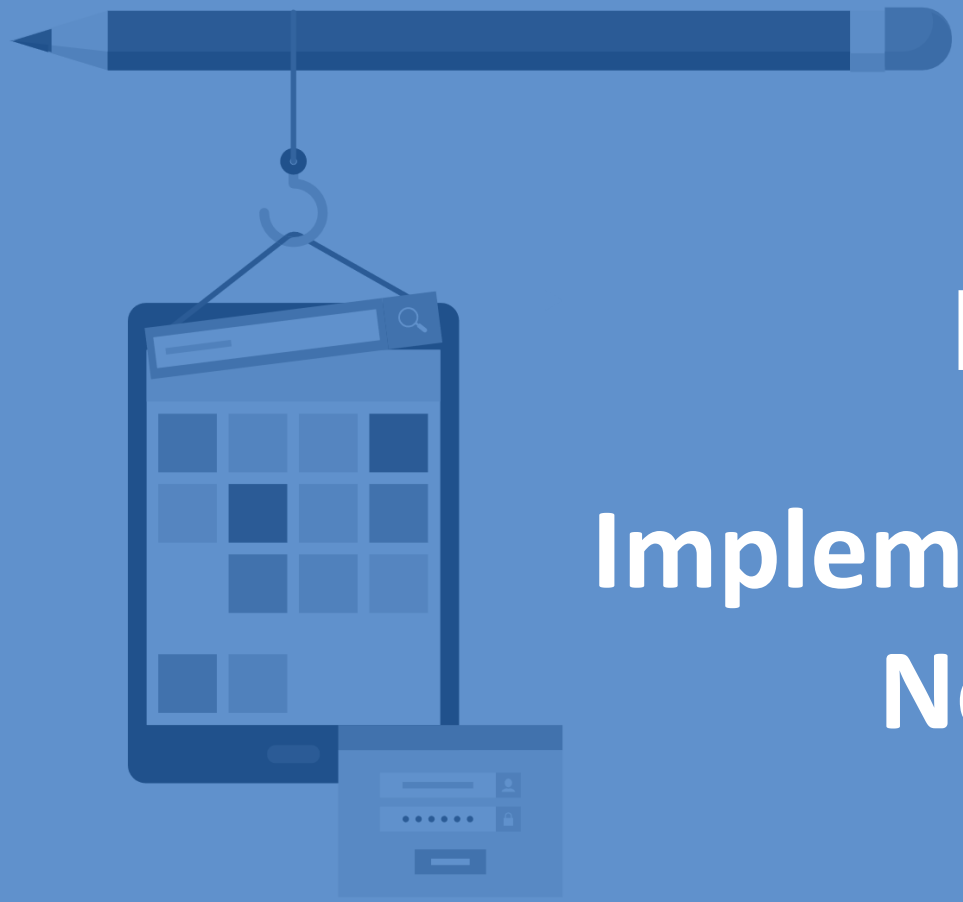


Use case III: handling cache coherence.



Use case III: handling cache coherence.





DEMO IV

Implementing Web Push Notifications

Wrap Up.

PWAs

Super cool
alternative
to native apps

Service Workers

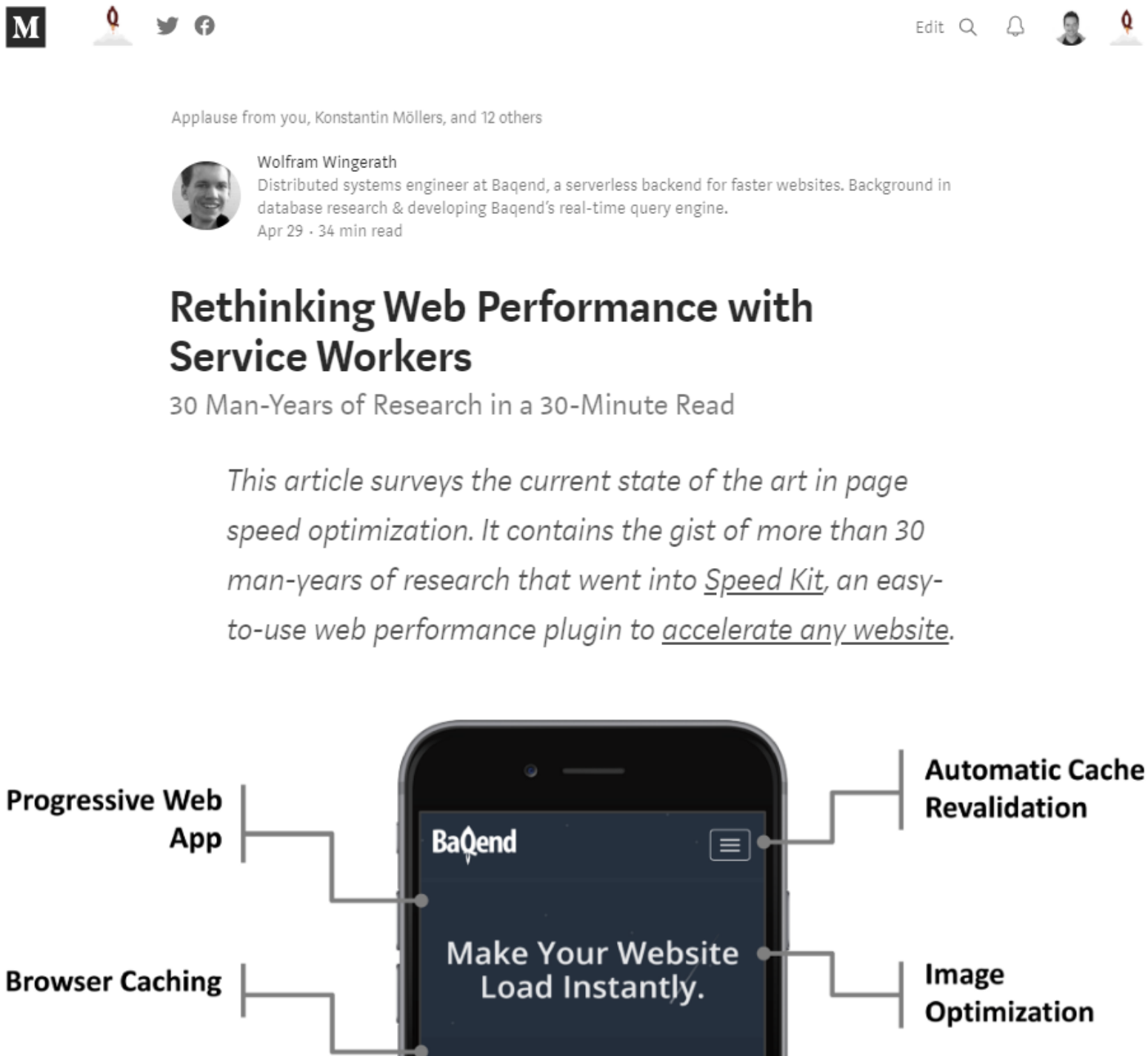
Powerful
programmable
network proxy

Use Case

Speed Kit:
Smart CDN though
Service Workers

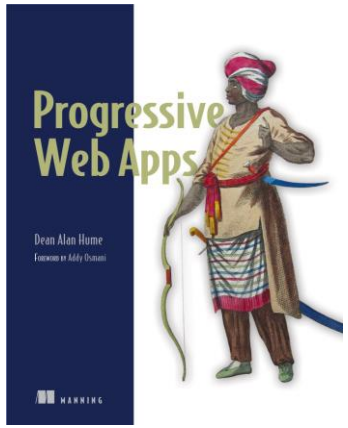
Learn more about this topic:

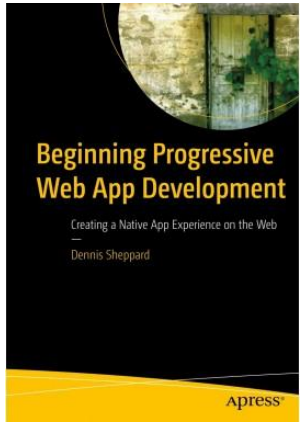
https://blog.baqend.com/



Learn more about Services Workers.

Recommended Books





Blogs

Baqend Blog
On Building a Faster Web

https://blog.baqend.com/

Ilya Grigorik
Internet plumber

https://www.igvita.com/

Jake Archibald wrote...
I discovered a browser bug

https://jakearchibald.com/

Guides & Tutorials

Progressive Web Apps

A new way to deliver amazing user experiences on the web.

https://developers.google.com/web/progressive-web-apps/

Progressive web apps

Jump to: PWA advantages Core PWA guides Technology guides

https://developer.mozilla.org/en-US/docs/Web/Apps/Progressive

Thank You

Get in Touch:

fg@baqend.com
www.baqend.com
https://github.com/DivineTraube/dahoam