Felix Gessert

Service Workers:

The Technology Behind Progressive Web Apps

Frontend

BaQend | code.talks

What we are going to cover.

PWAs

- Core Features
- Building Blocks
- Implementation

Service Workers

- Lifecycle
- Network Interception
- Caching Strategies

Use Case

Service Workers in Production at Baqend

Why do(n‘t) we love native apps?

Progressive Web Apps

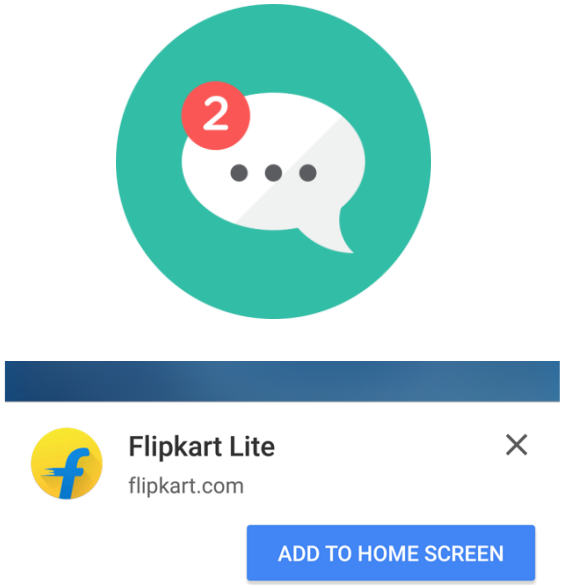
Combine the best from **native** and **web apps**.

What are Progressive Web Apps?

Progressive Web Apps (PWAs)

+

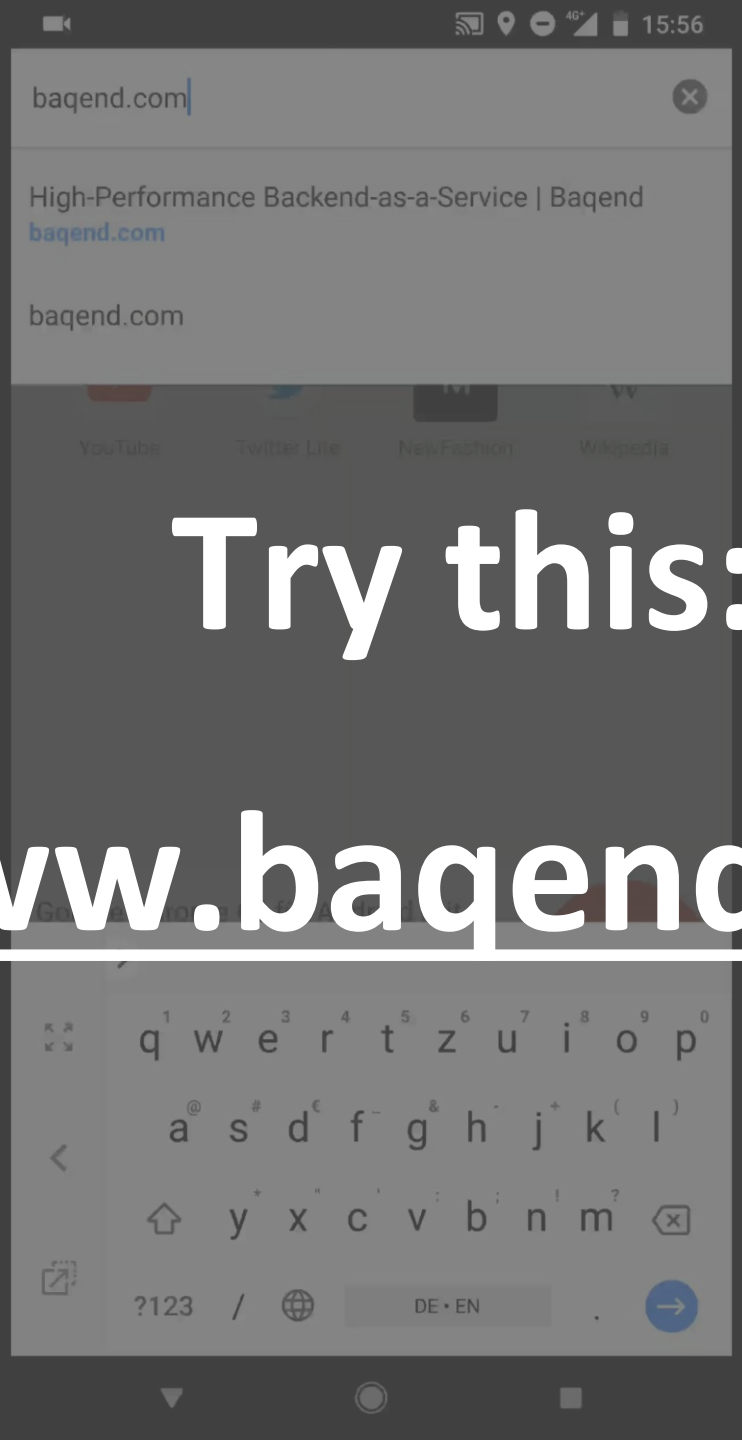
+



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:

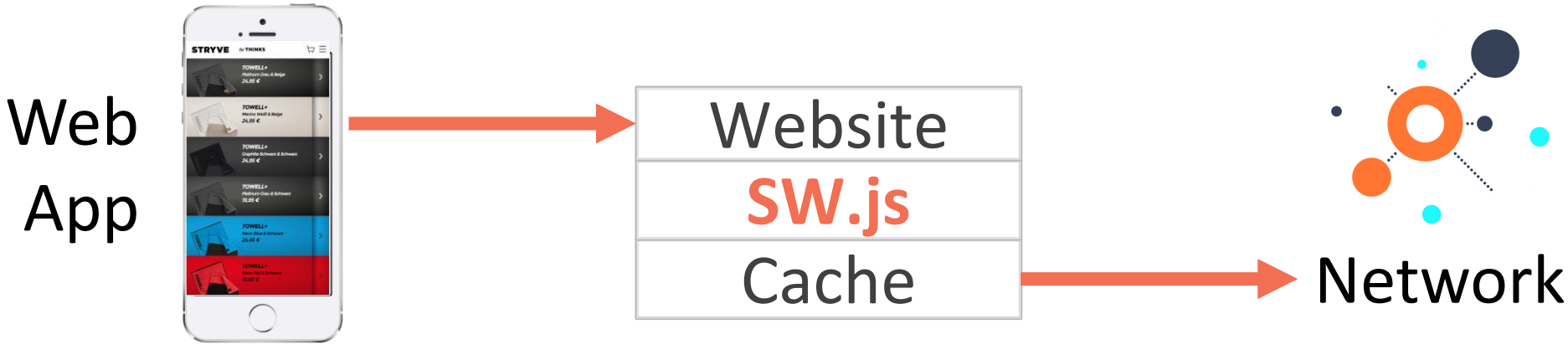
```
<link rel="manifest" href="/manifest.json">
{
  "short_name": "Codetalks PWA",
  "icons": [
    {"src": "icon-1x.png", "type": "image/png", "sizes": "48x48"}],
  "start_url": "index.html?launcher=true"
}
```

Implementing PWAs

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

Gracefully degrade when not supported

2. Service Workers for caching & offline mode:

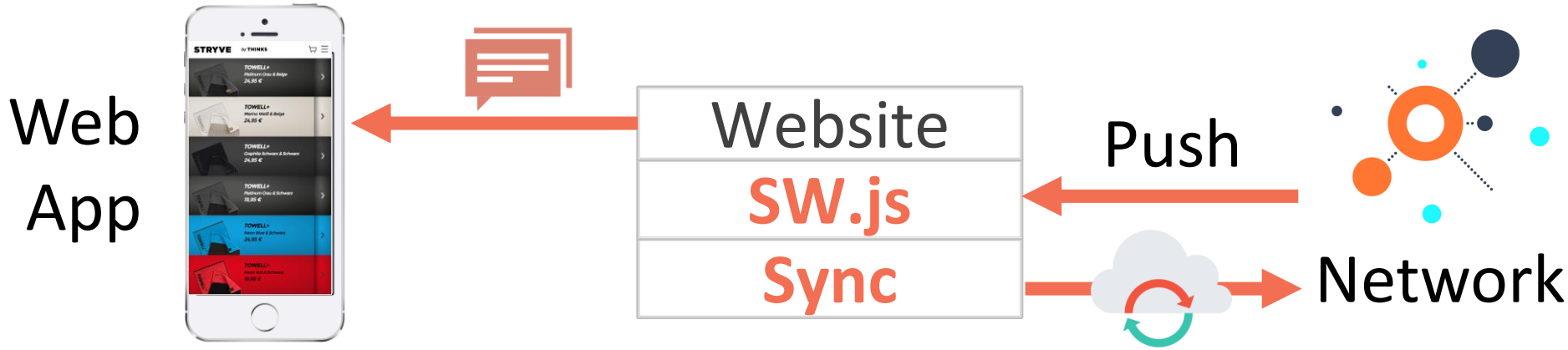


Implementing PWAs

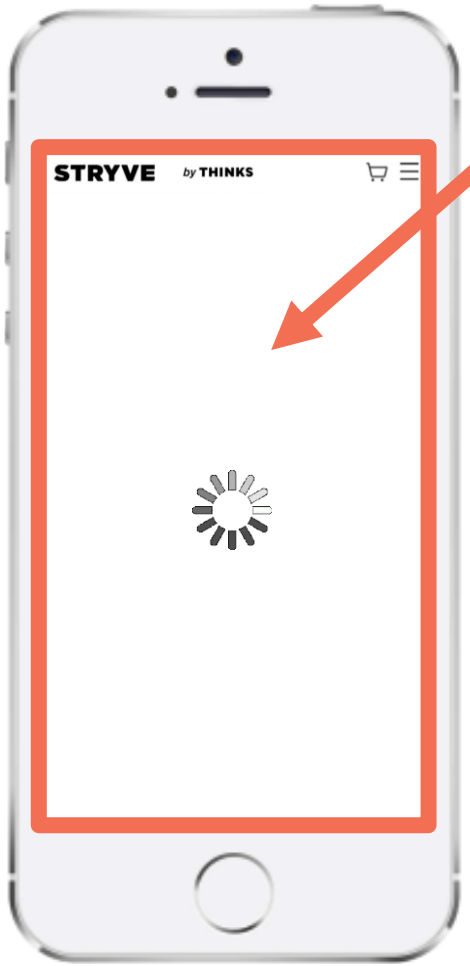
PWAs are **best practices** and **open web standards**

Progressively enhance the user experience

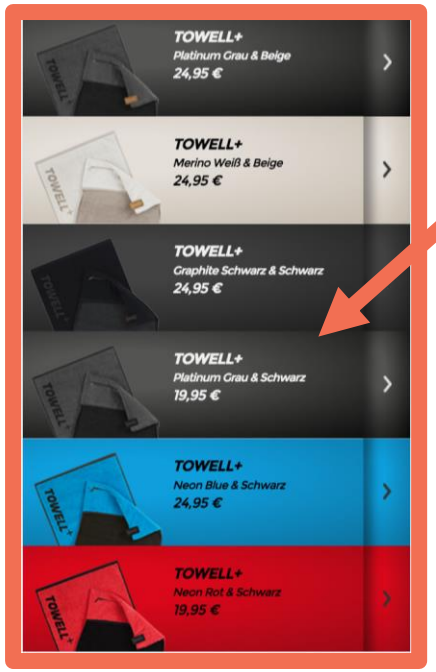
3. Add Web Push and Background Sync:



Typical Architecture: App Shell Model



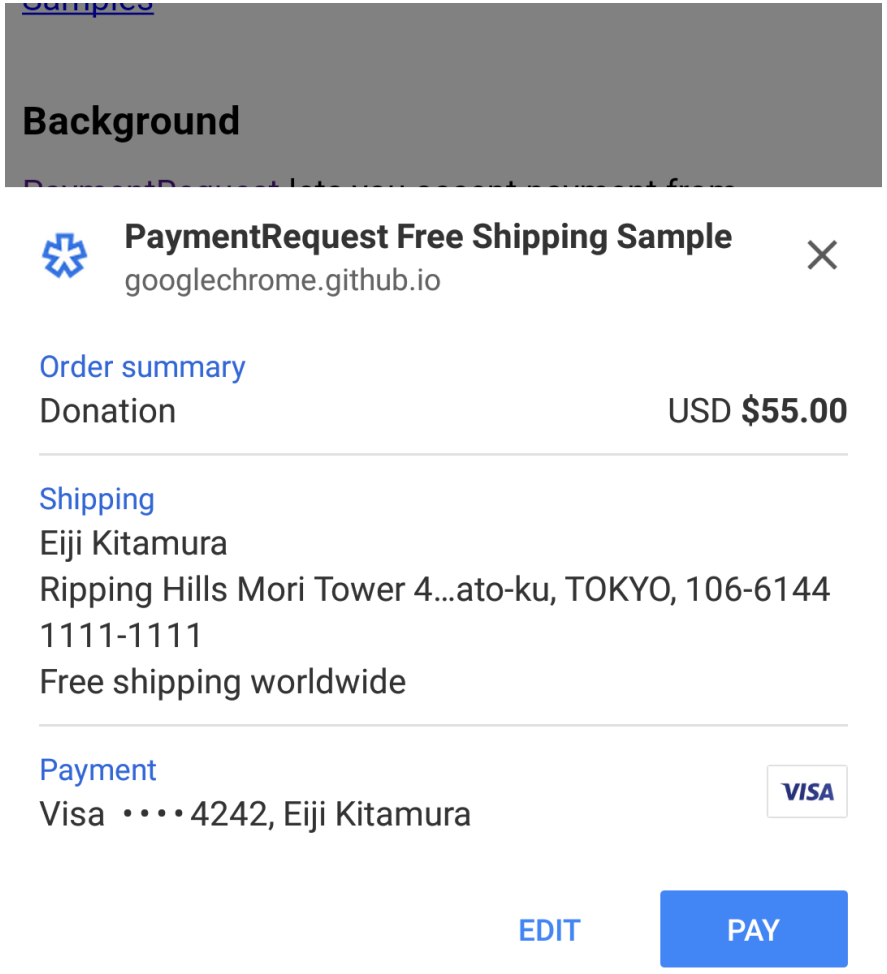
App Shell: **HTML**, **JS**, **CSS, images** with app logic & layout



Content: Fetched on demand & may change more often

What is the future and vision of Progessive Web Apps?

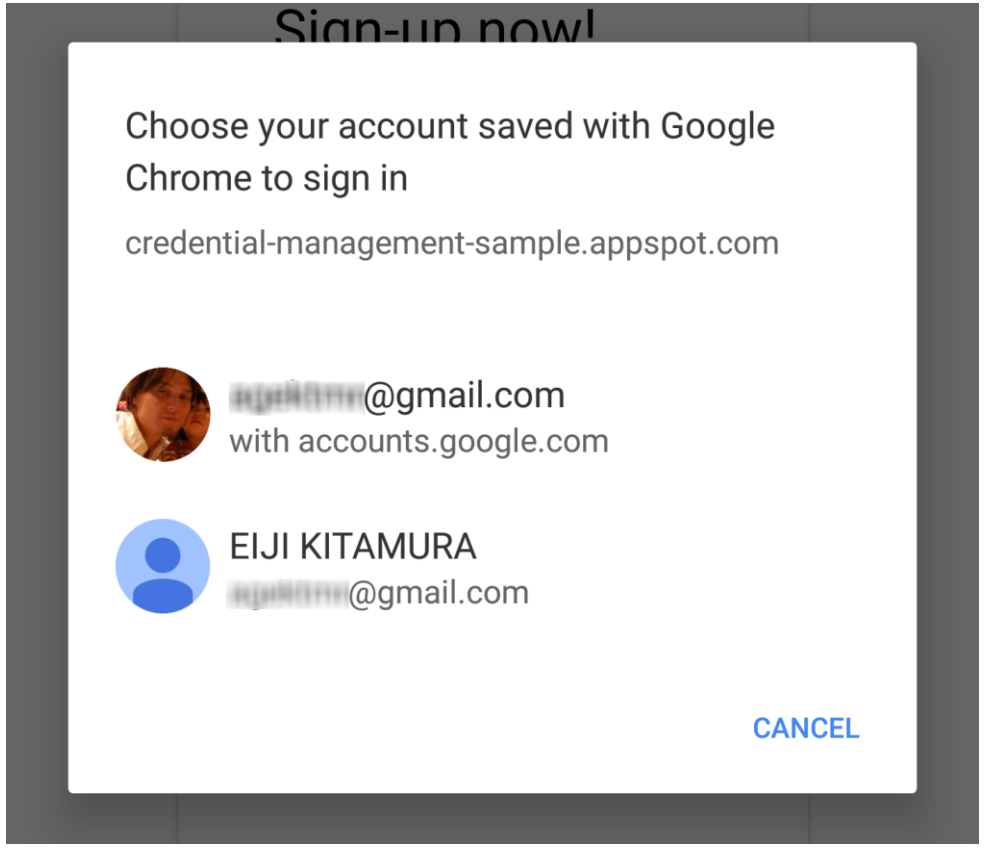
Integrate payment.



Web Payment APIs

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

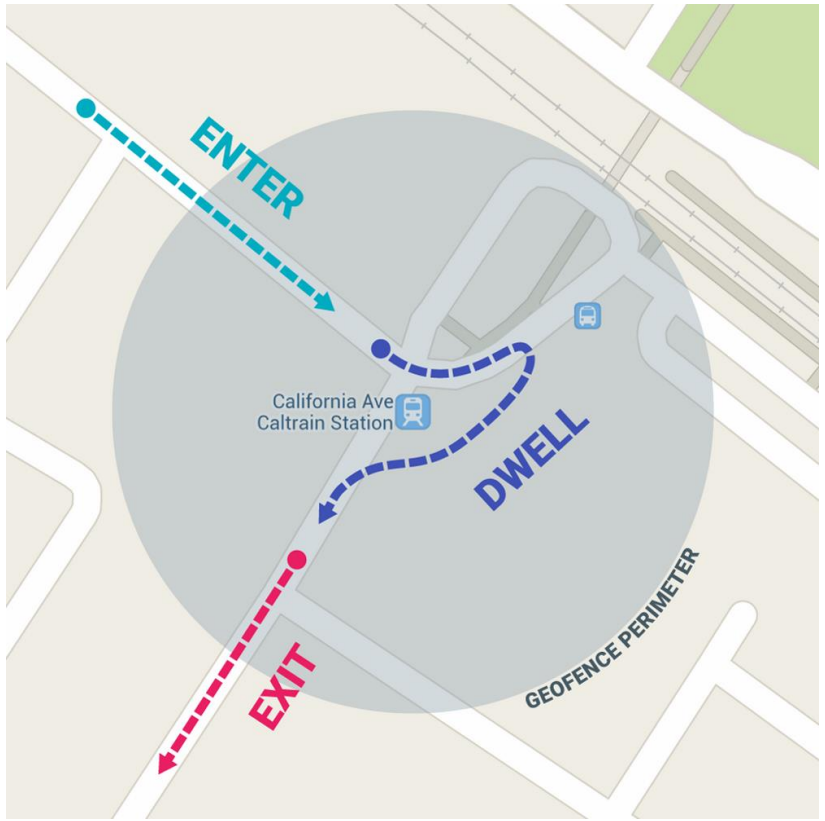
Manage users and logins.



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 of enters a defined area
- Requires **permission**

Build conversational interfaces.

Web Speech API

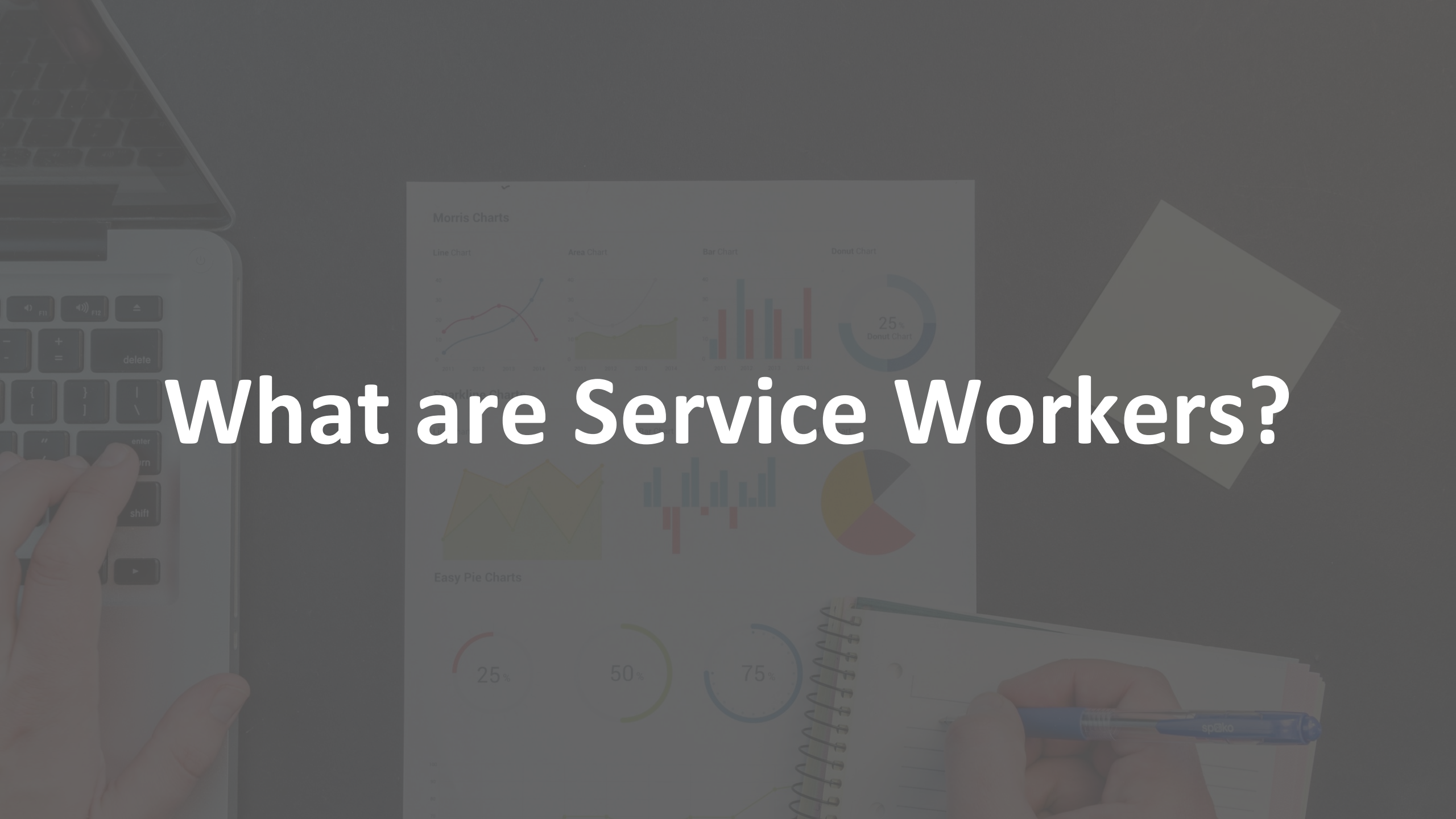
Native Speech Recognition in the Browser:

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

Seemless sharing between apps.

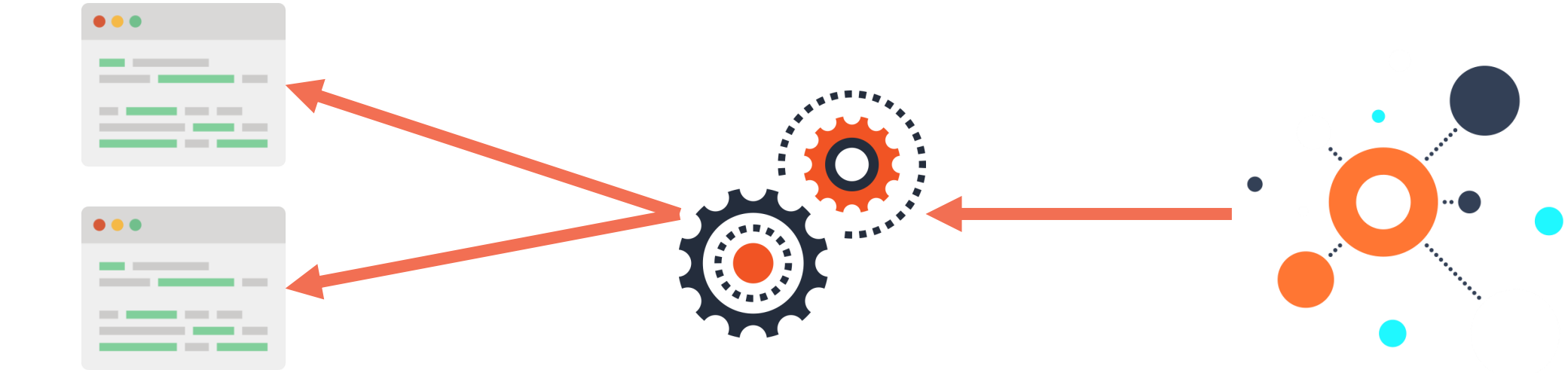
Web Share API

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



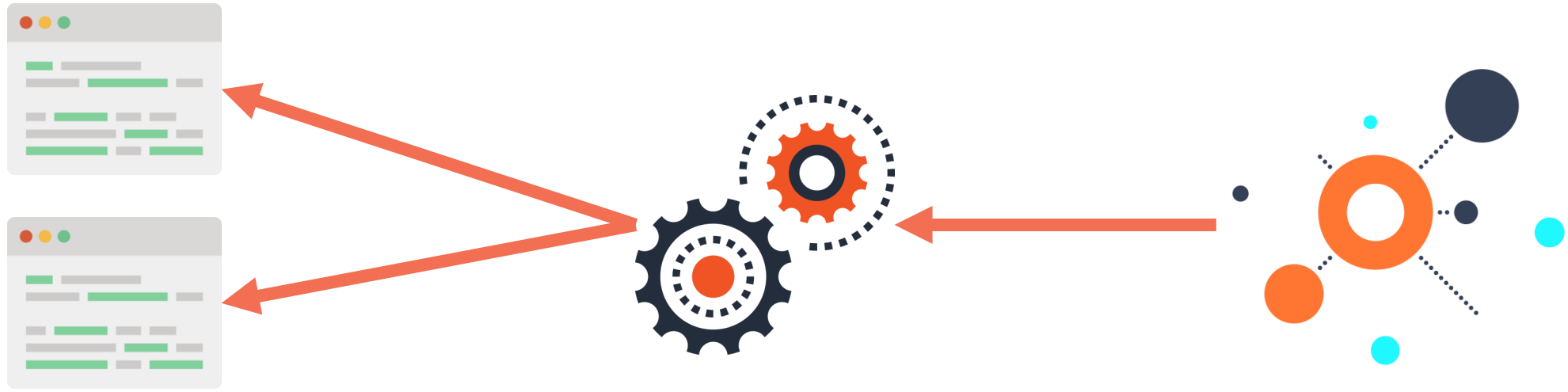
What are Service Workers?

What are Service Workers?



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

What do Service Workers do?



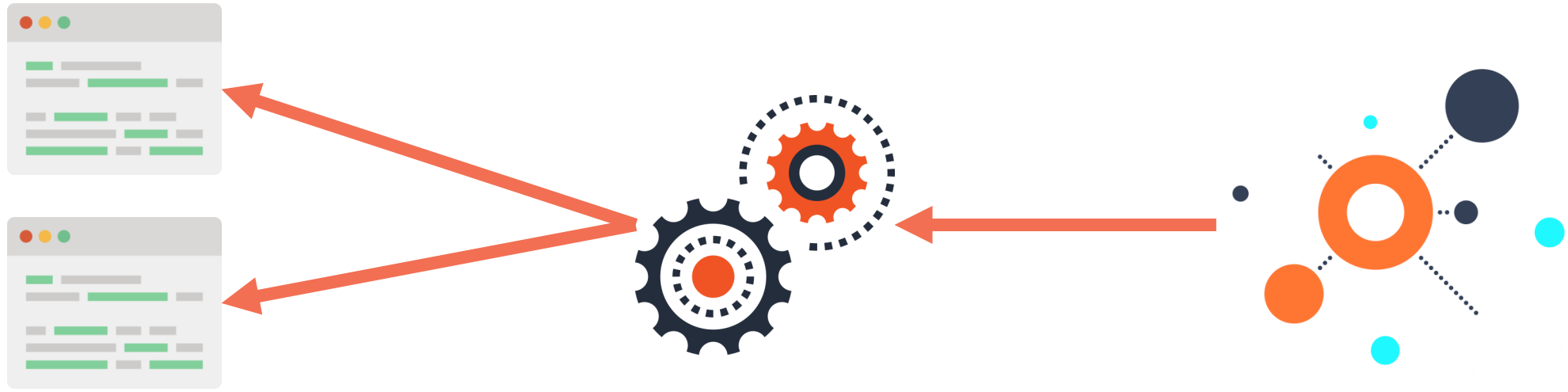
Browser Tabs

Service Worker

Network

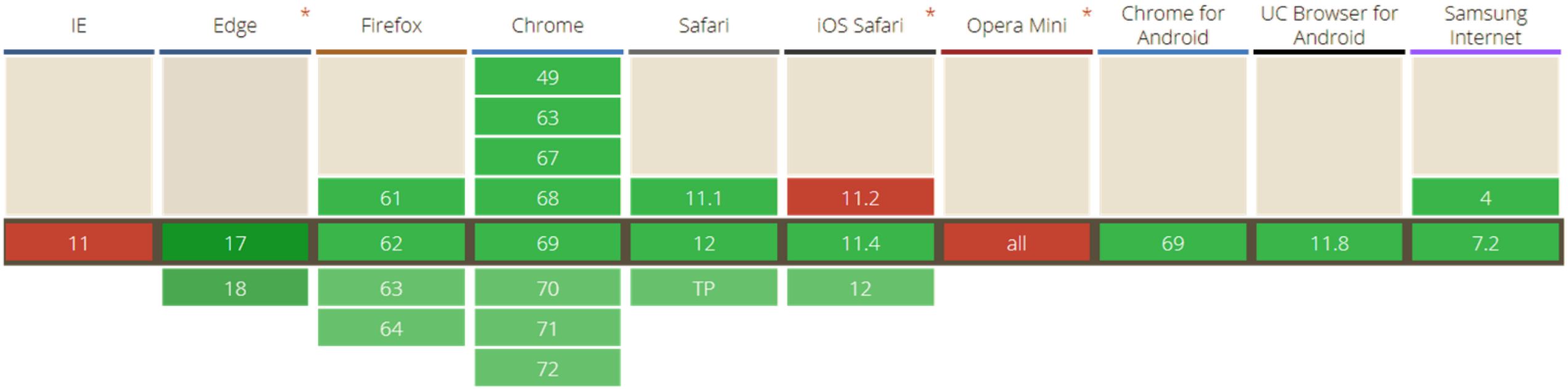
- Cache Data (CacheStorage)
- Store Data (IndexedDB)
- Receive Push
- Respond when Offline

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 **>85%** of browsers.

Requires **TLS Encryption**.

Late, but all in: Microsoft

Publish PWAs to
Microsoft Store

PWA BUILDER

or

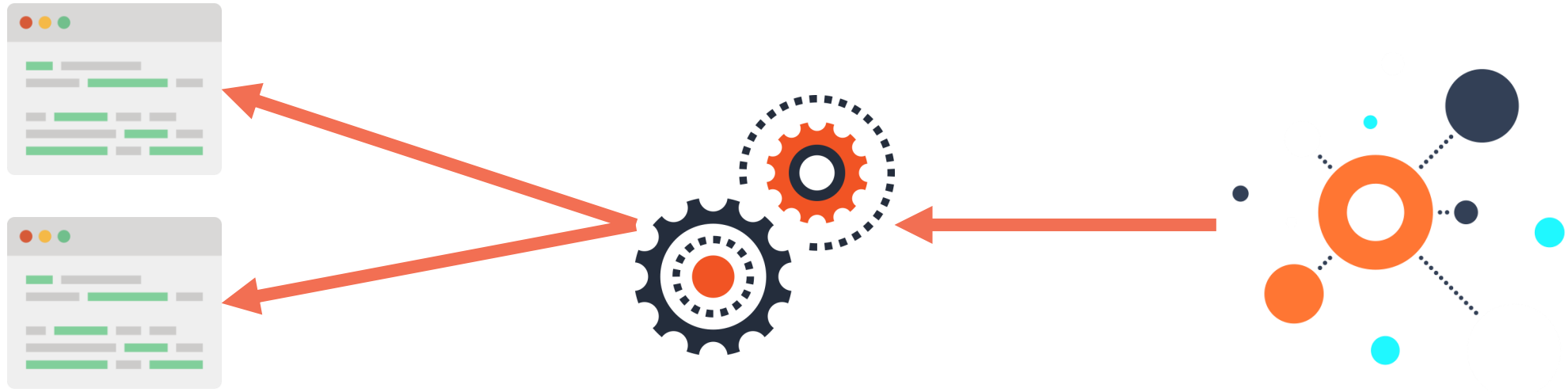
Bing Crawls PWAs

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?



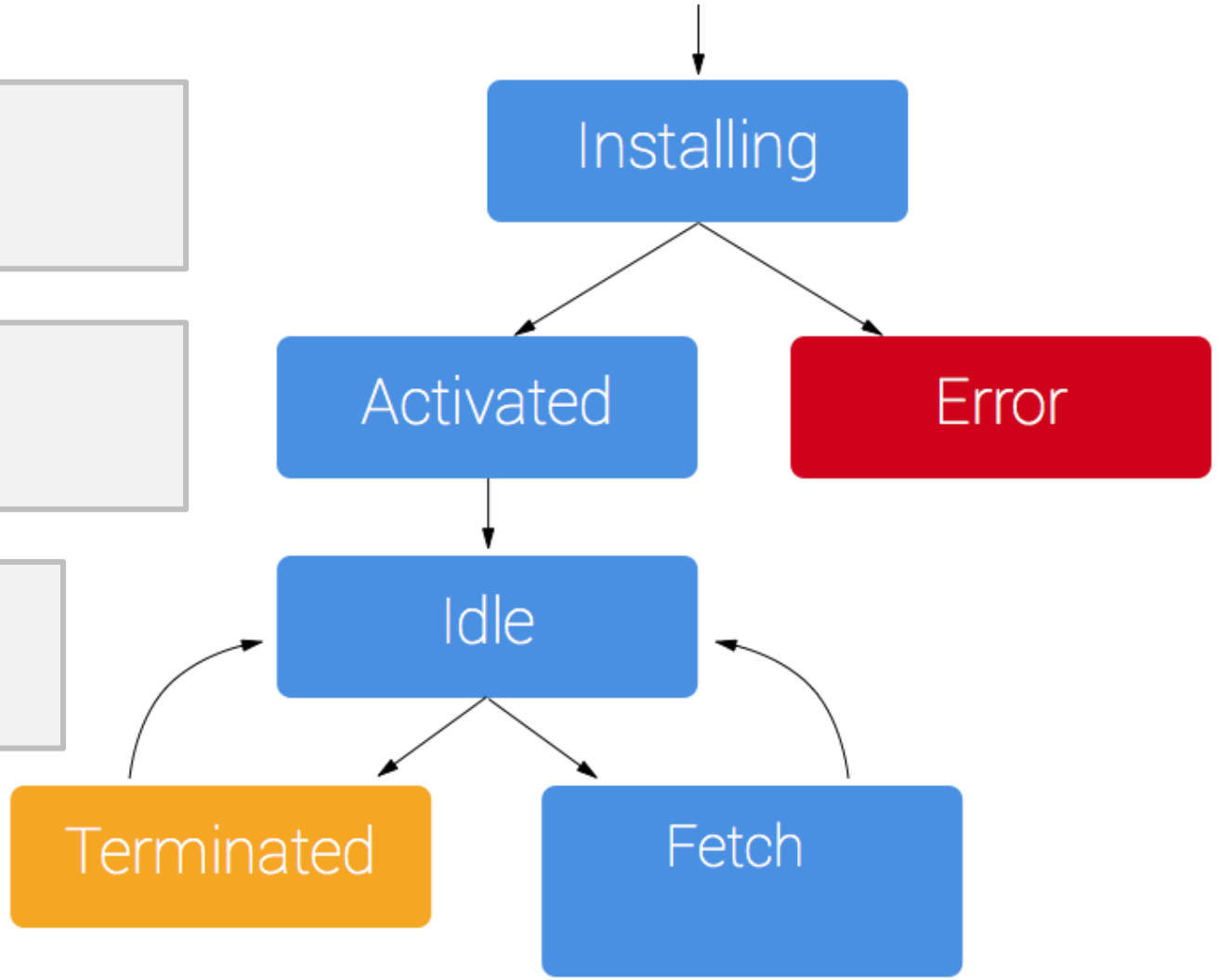

```
<script>
    navigator.serviceWorker.register('/sw.js');
</script>
```

What does the lifecycle look like?

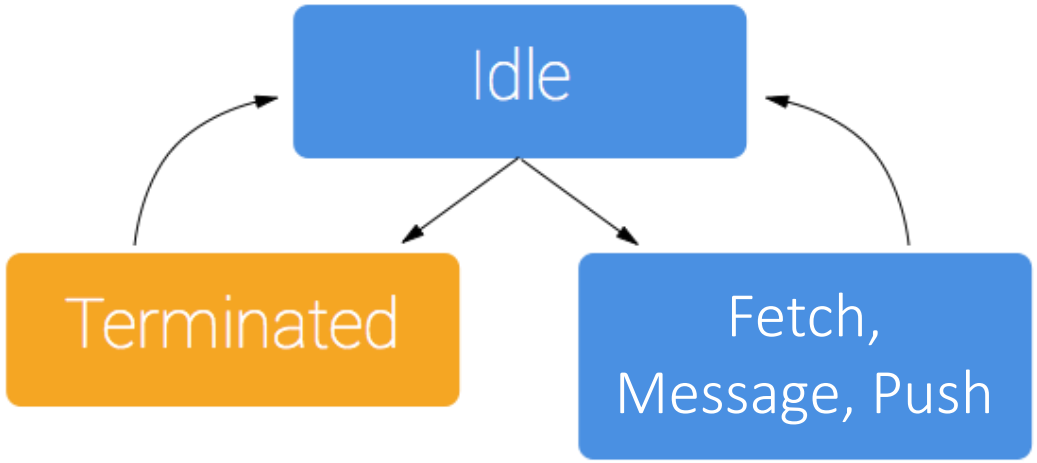
```
self.addEventListener('install', (event) => {
  // Perform install steps
});
```

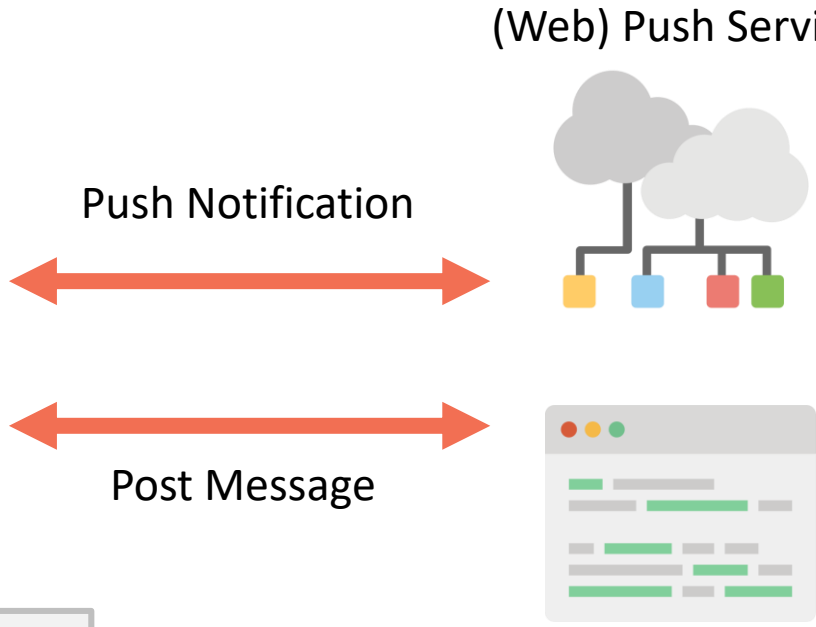
```
self.addEventListener('activate', (event) => {
  // Perform activate steps
});
```

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



How to communicate with Service Workers?



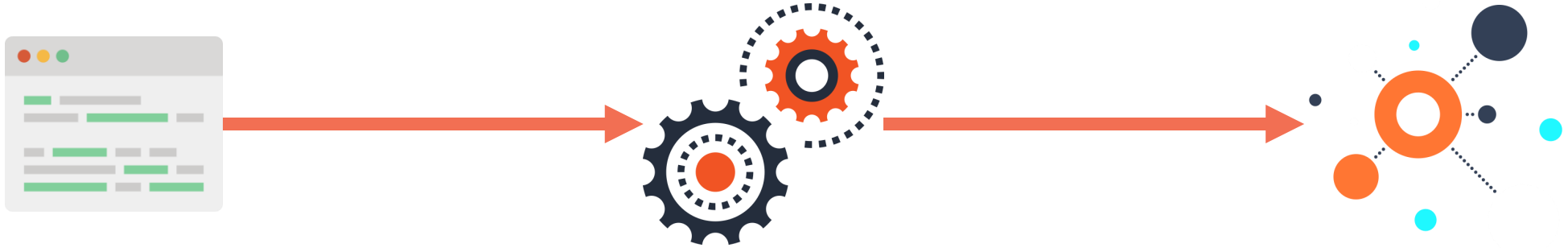



```
self.addEventListener('message', (event) => {
  // Receive message
});
```

```
// 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

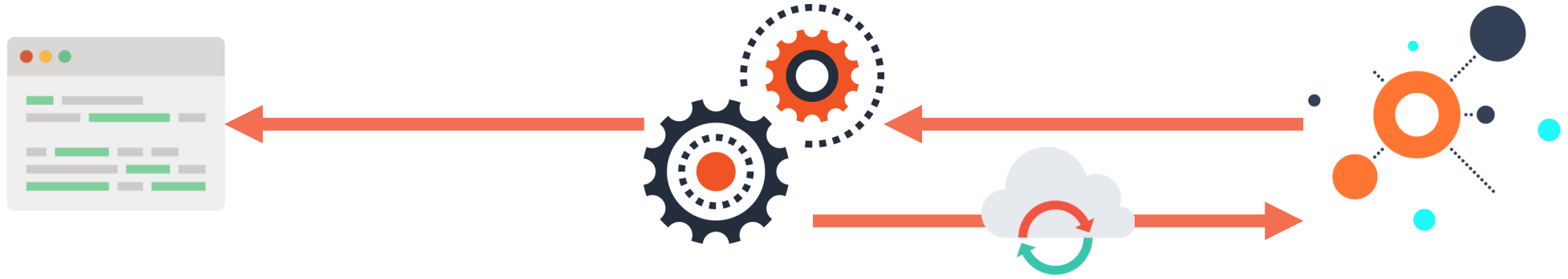
IndexedDB

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

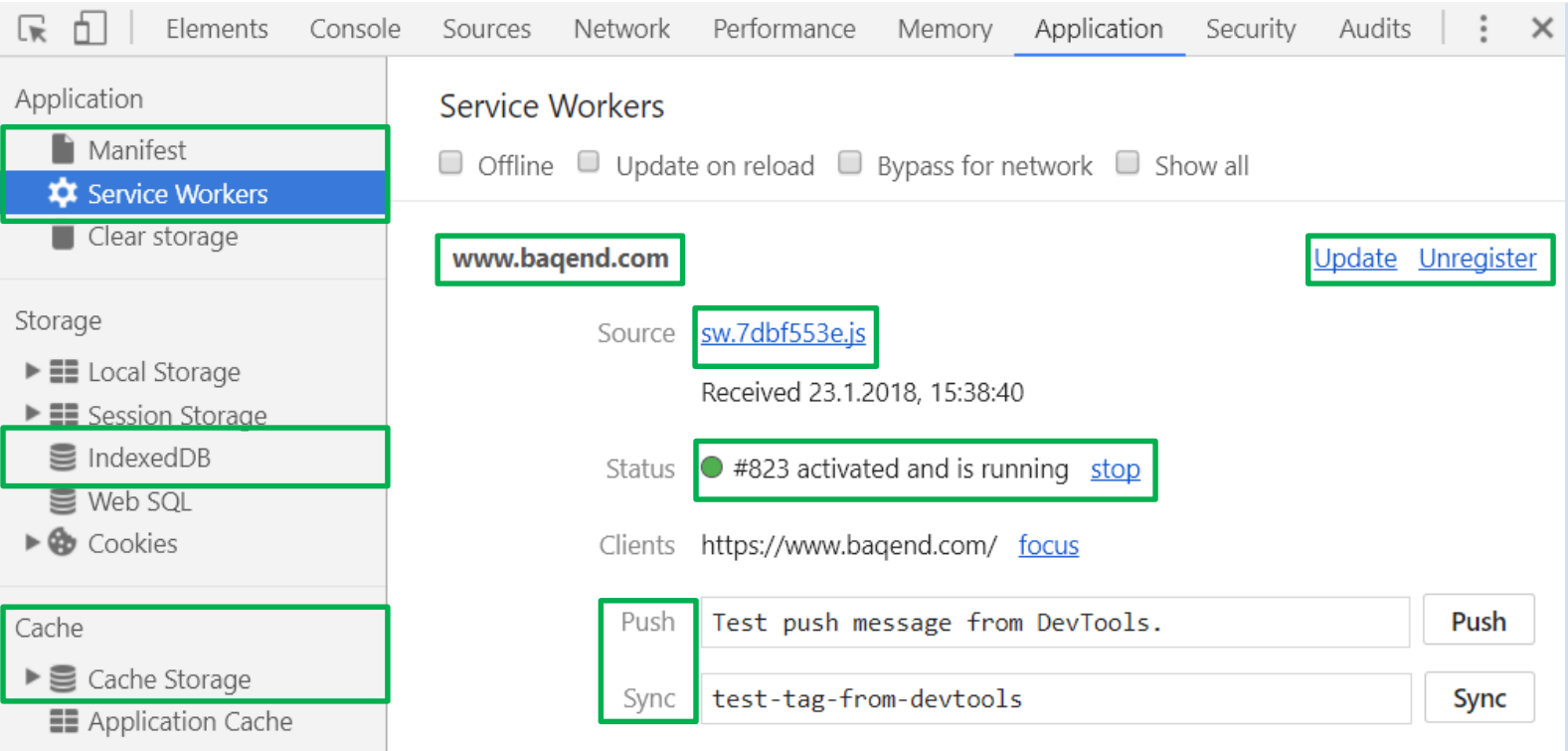
Periodic Sync

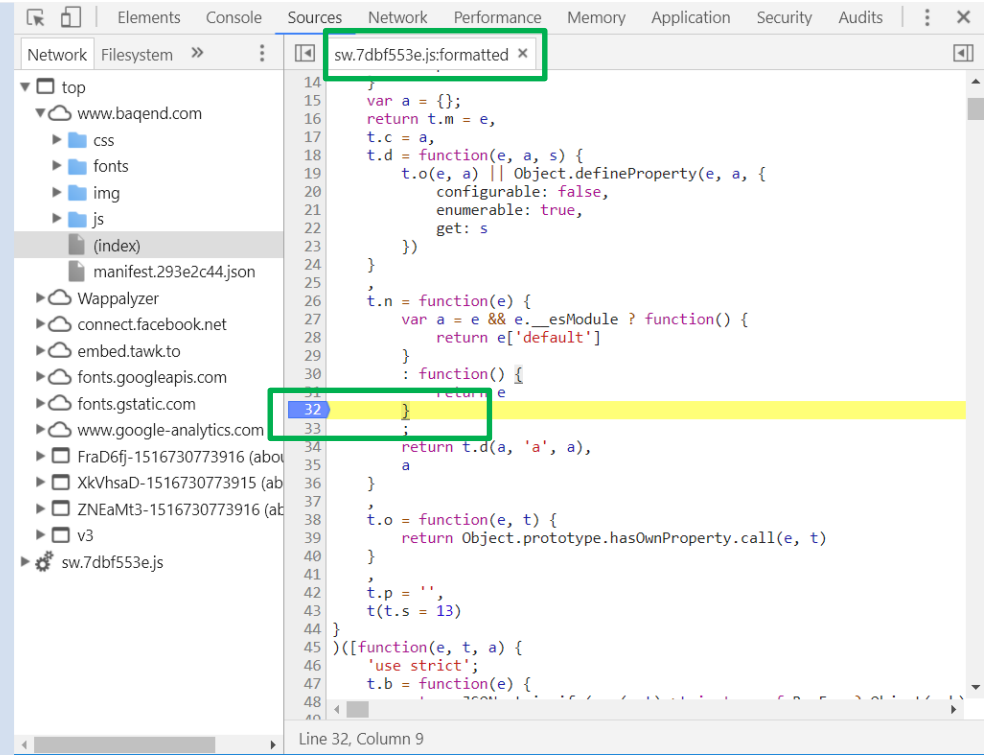
- executed when online, according to **period options**

Use Cases

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

Service Worker Debugging





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)

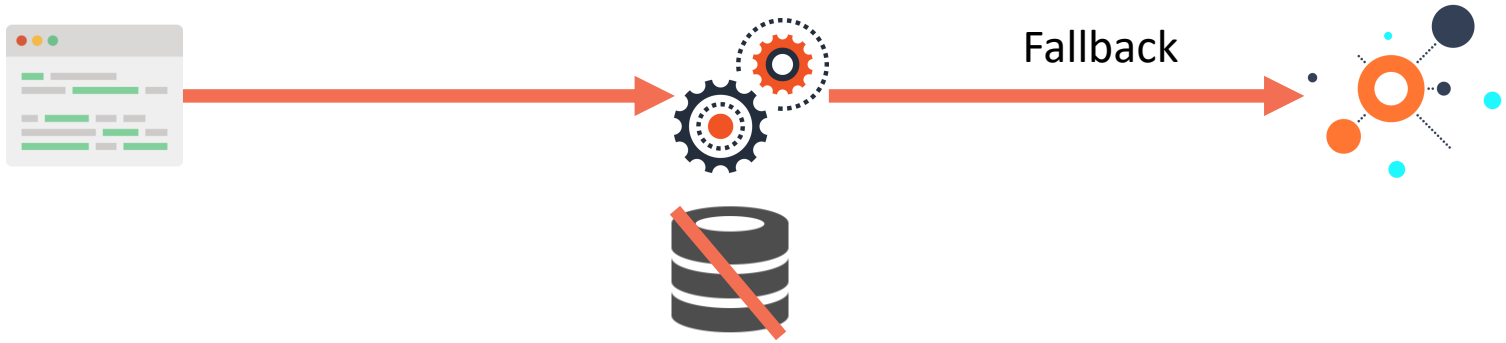
Caching Strategies – Cache Only

Gets all requests from cache or fails.

Caching Strategies – Cache, Network Fallback

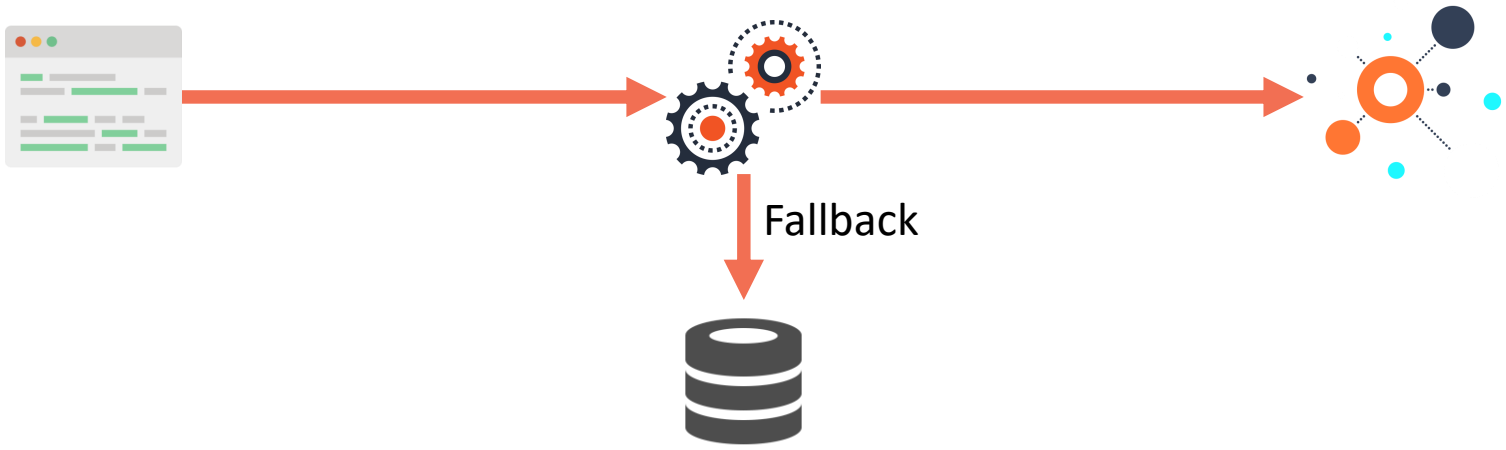
Gets requests from cache & uses network as fallback.

Caching Strategies – Network Only



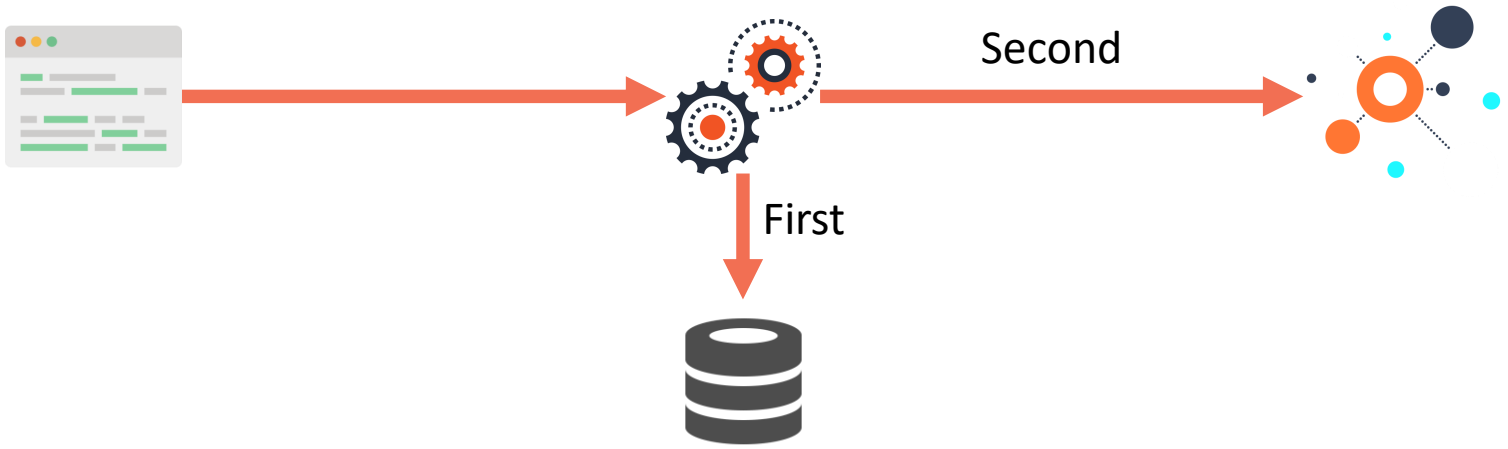
Gets requests from network only.

Caching Strategies – Network, Cache Fallback



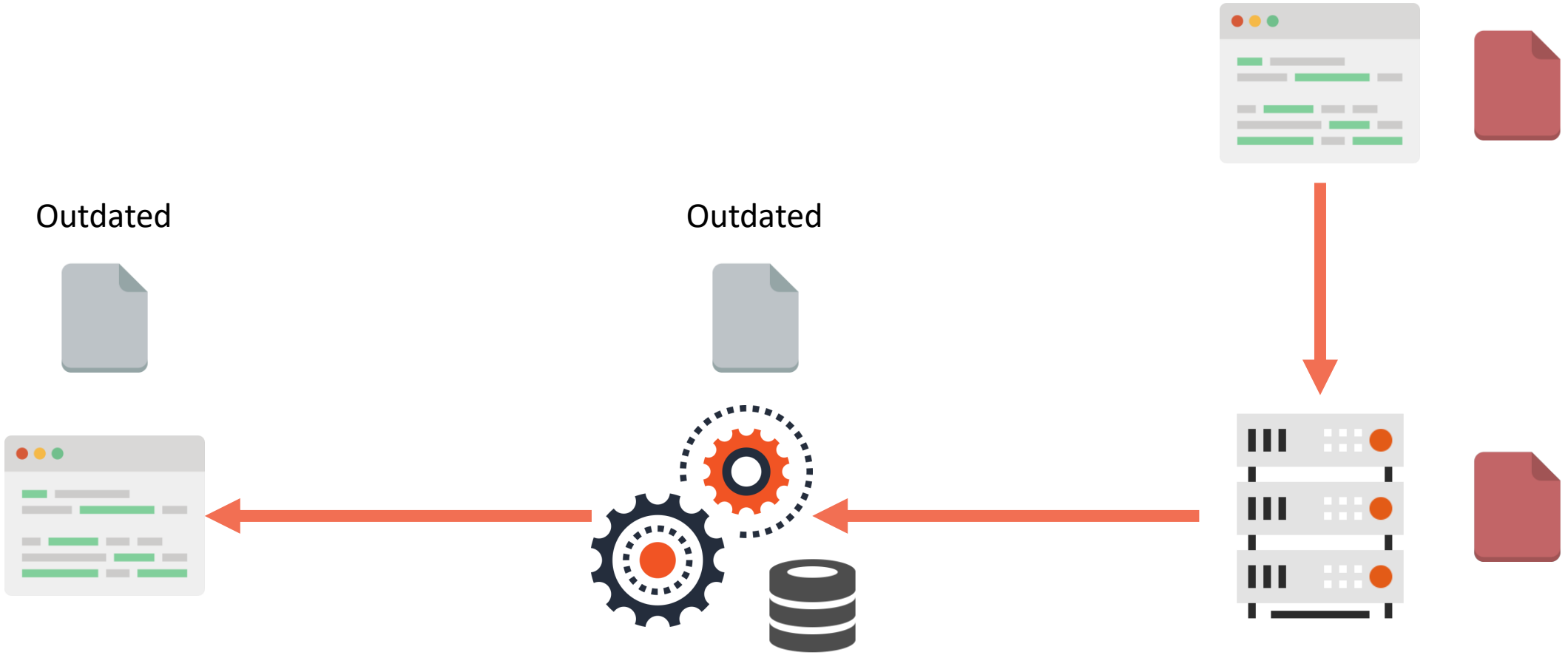
Gets requests from network, the cache acts as fallback (offline mode).

Caching Strategies – Cache, then Network



Gets requests from cache first and from network in background.

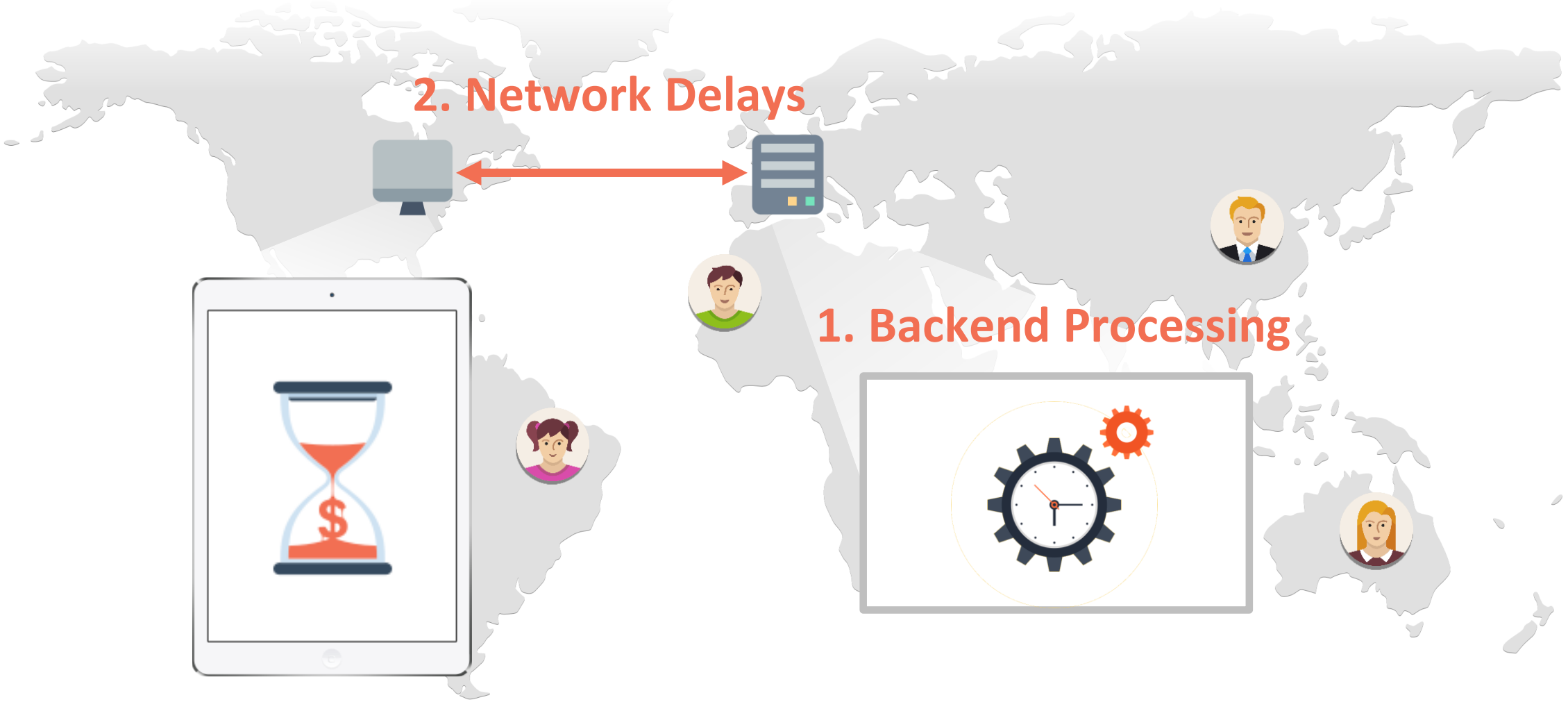
Major Challenge: Cache Coherence



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

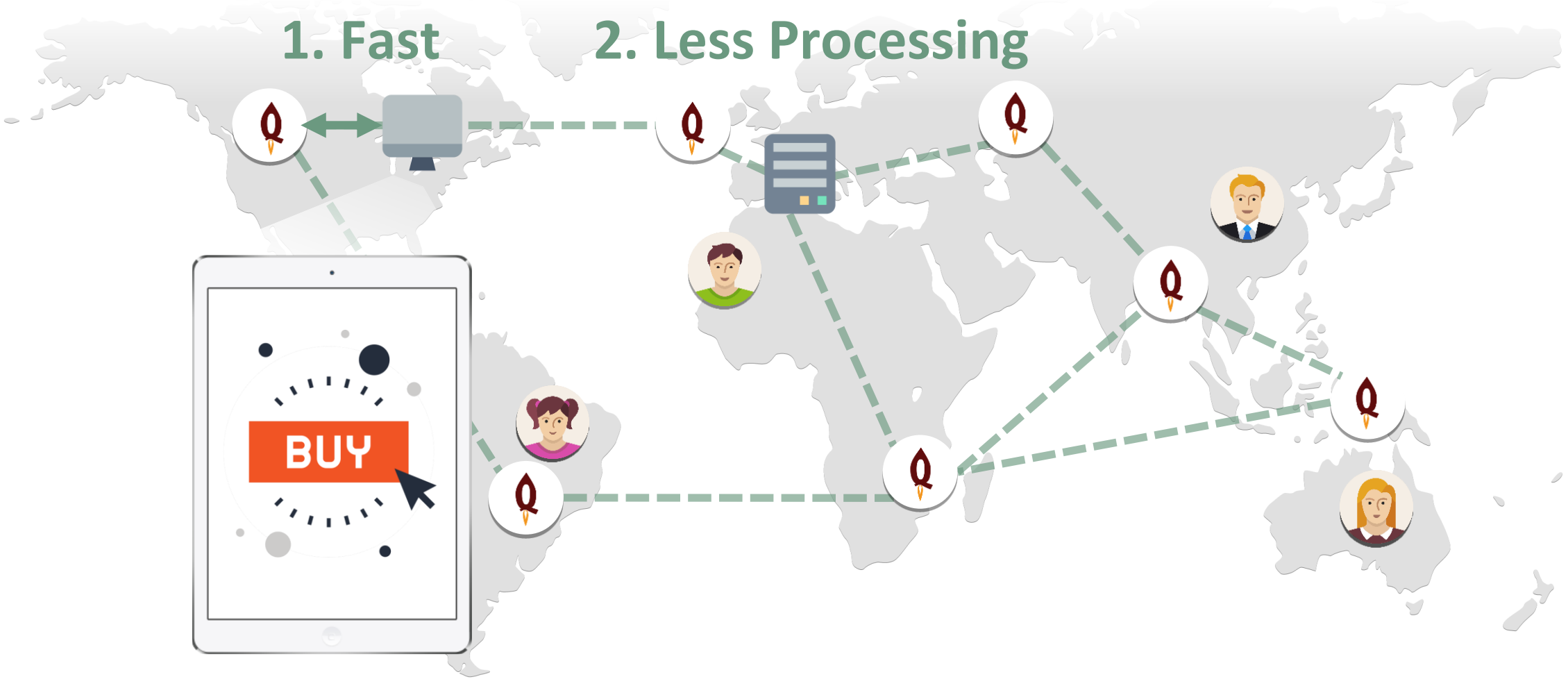
How we use Service Workers at Baqend

Problem: slow backends & networks.



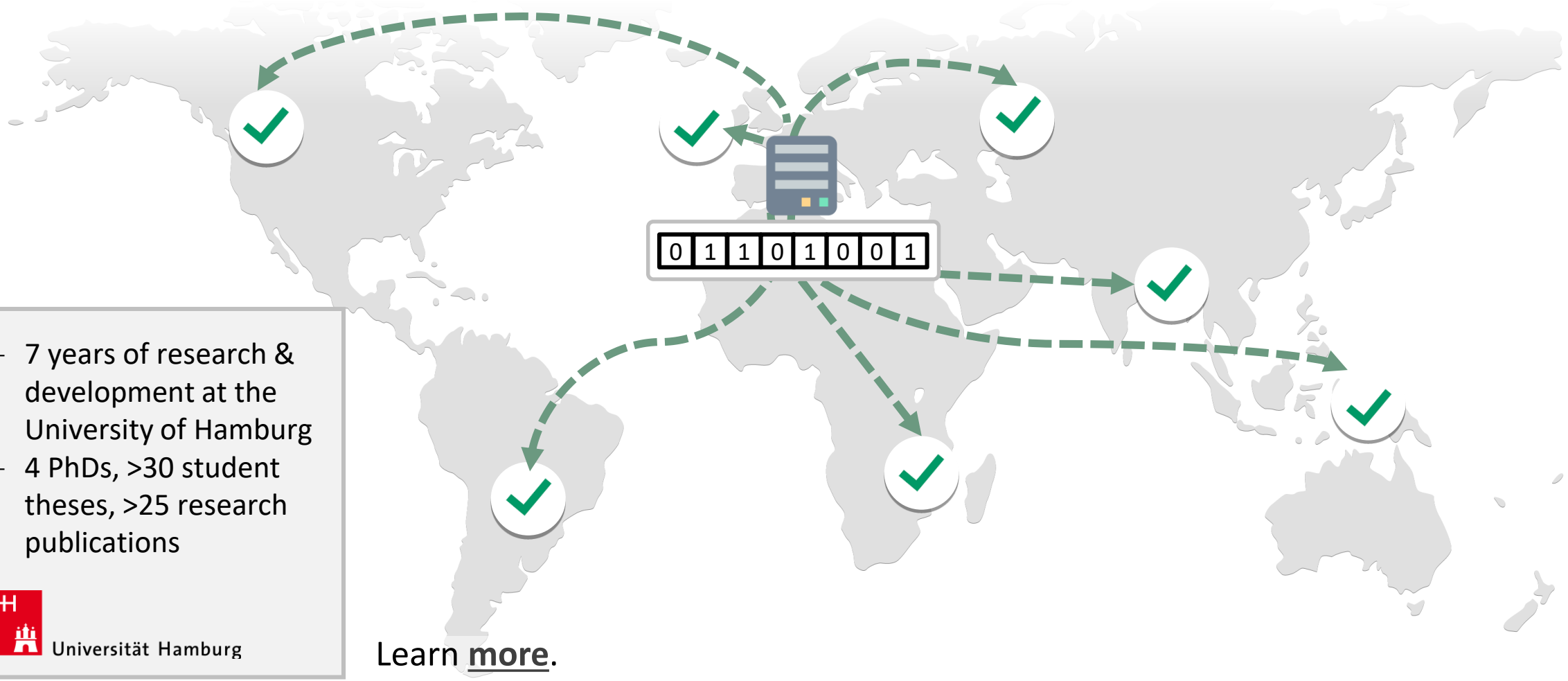
Solution: Speed Kit

Service Worker rewrites & accelerates slow requests.



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

Backed by 30 man-years of research.

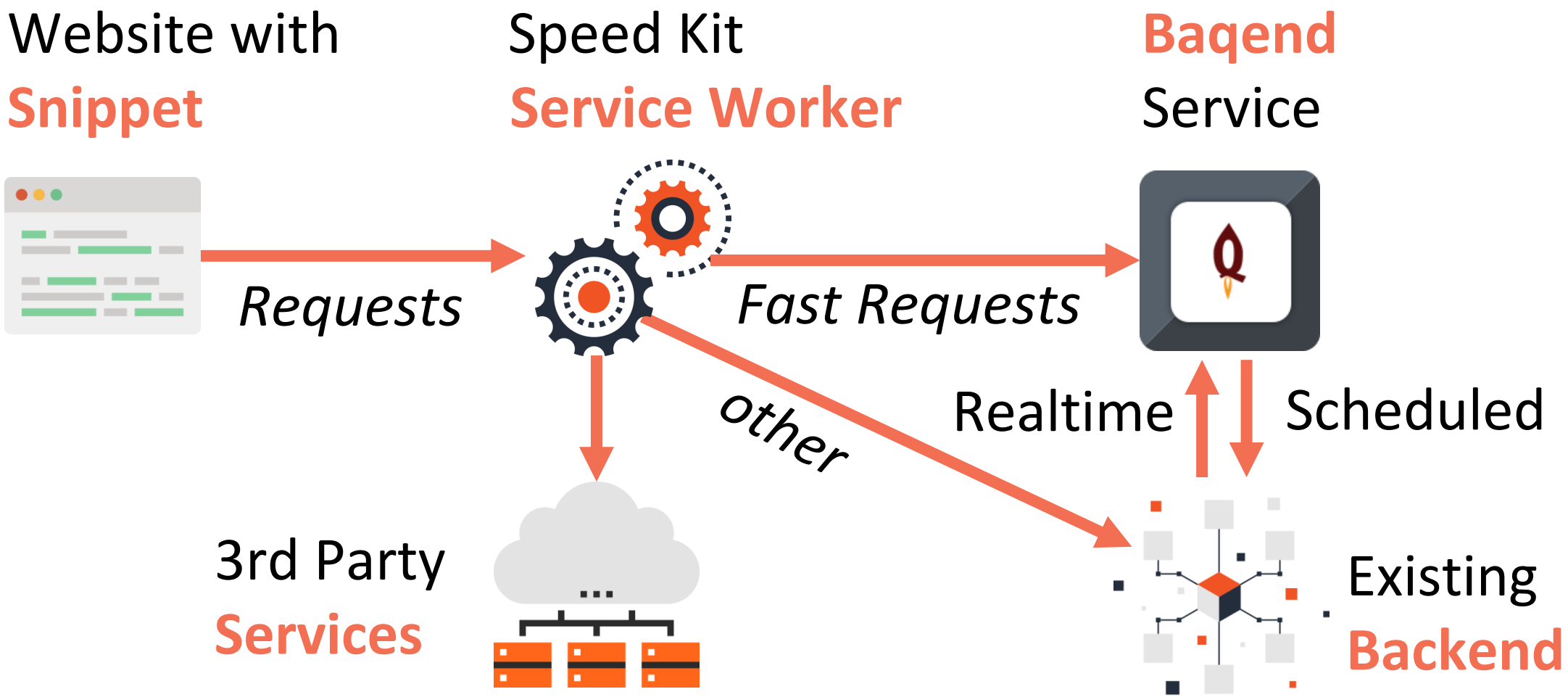


- 7 years of research & development at the University of Hamburg
- 4 PhDs, >30 student theses, >25 research publications

Universität Hamburg

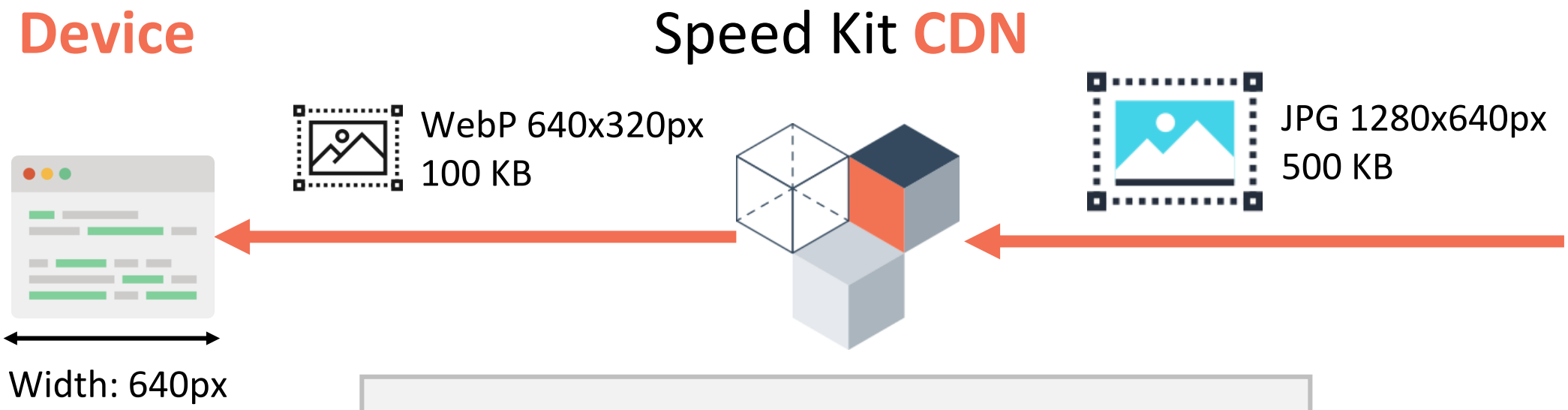
Learn more.

How Speed Kit leverages Service Workers.



Use case I: optimize images.

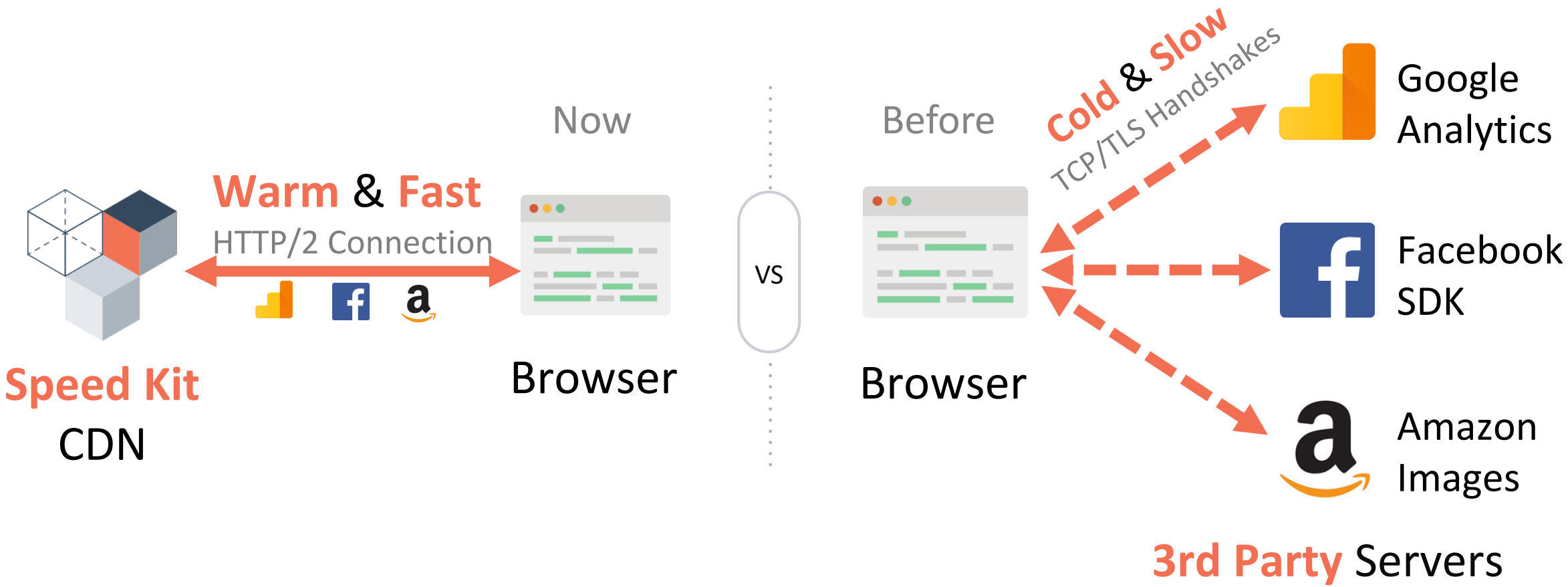
SW sends client resolution → responsive image.



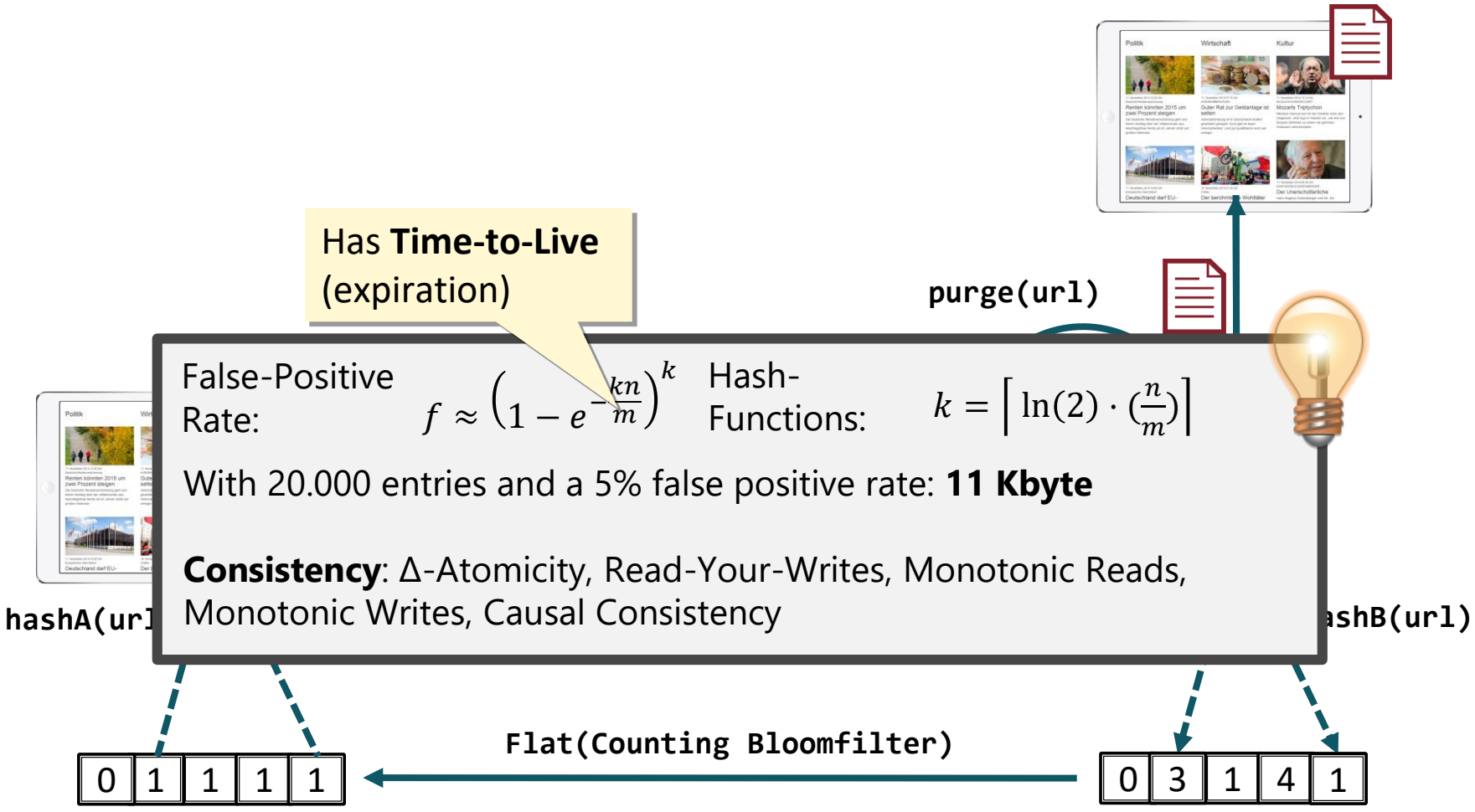
- Images transcoded to **WebP**
- Rescaled to match **Screen Size**
- JPG and PNG **Recompression**

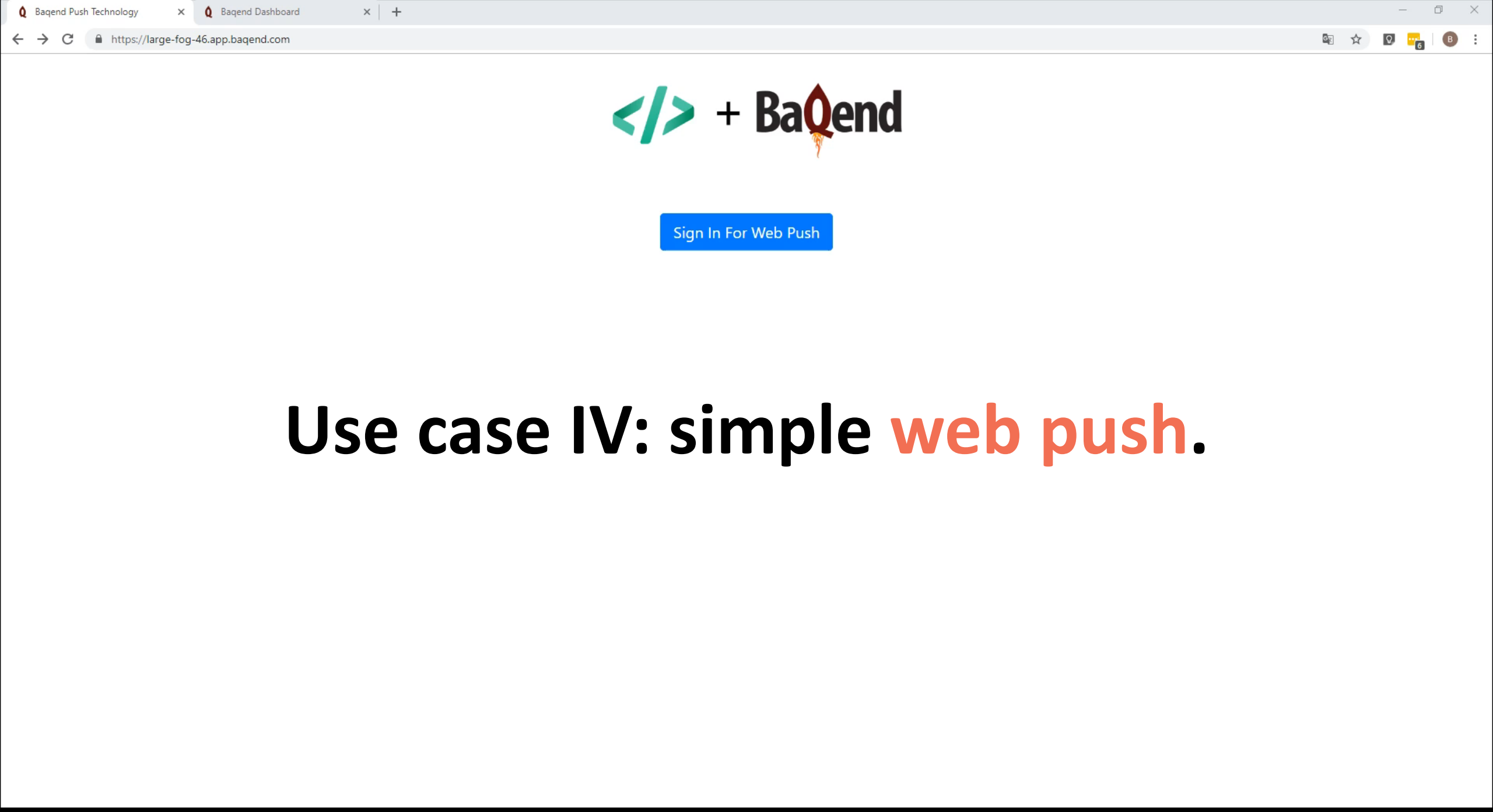
Use case II: re-route 3rd party dependencies.

Service Workers can manipulate other domains.



Use case III: handling cache coherence.





Use case IV: simple web push.

Demo:
Looking into Service Workers

Now, we have a Progressive Web App.

How do we measure its performance?

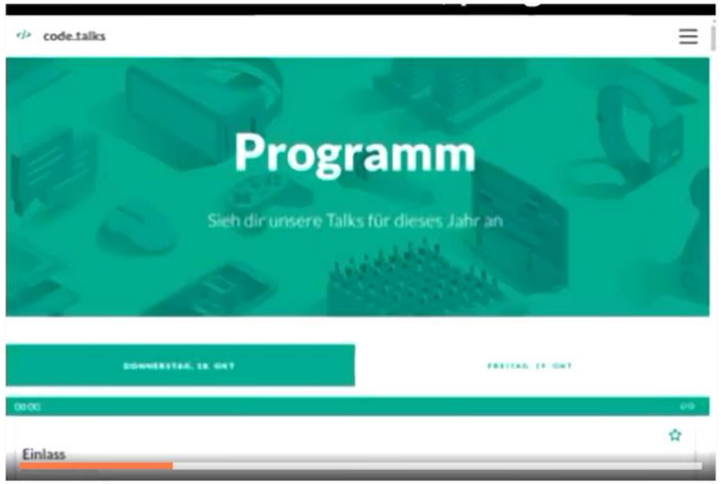
A PWA can make a huge difference.

https://www.codetalks.de/program

Your Website
4300 ms

4.4x
Faster

With Speed Kit
980 ms



Your Website
4.3s

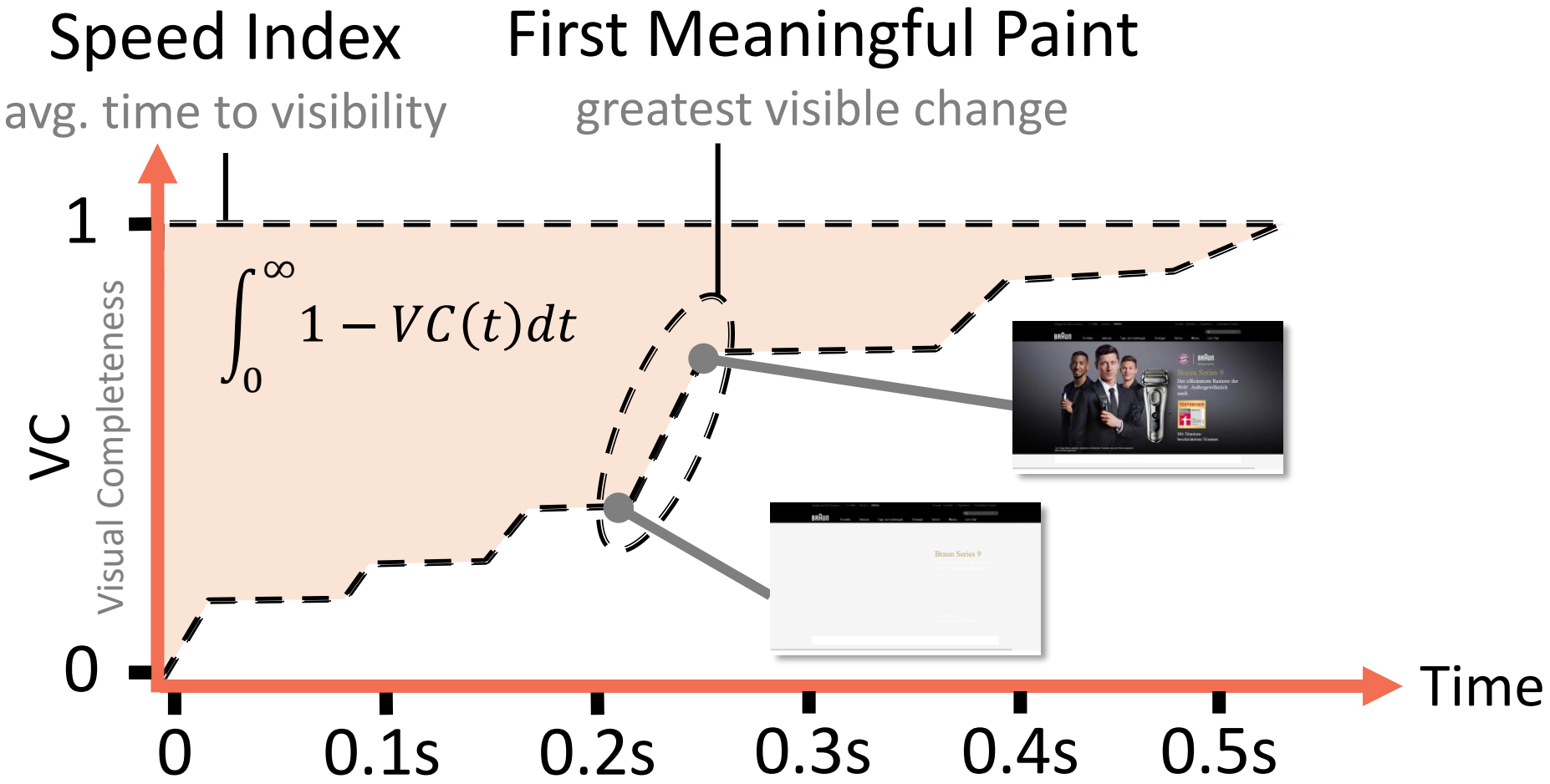
With Speed Kit
1s

Average

Fast

Measuring PWA performance.

User-perceived performance.



Test your site.

test.speed-kit.com

Wrap Up.

PWAs

Super cool
alternative
to native apps

Service Workers

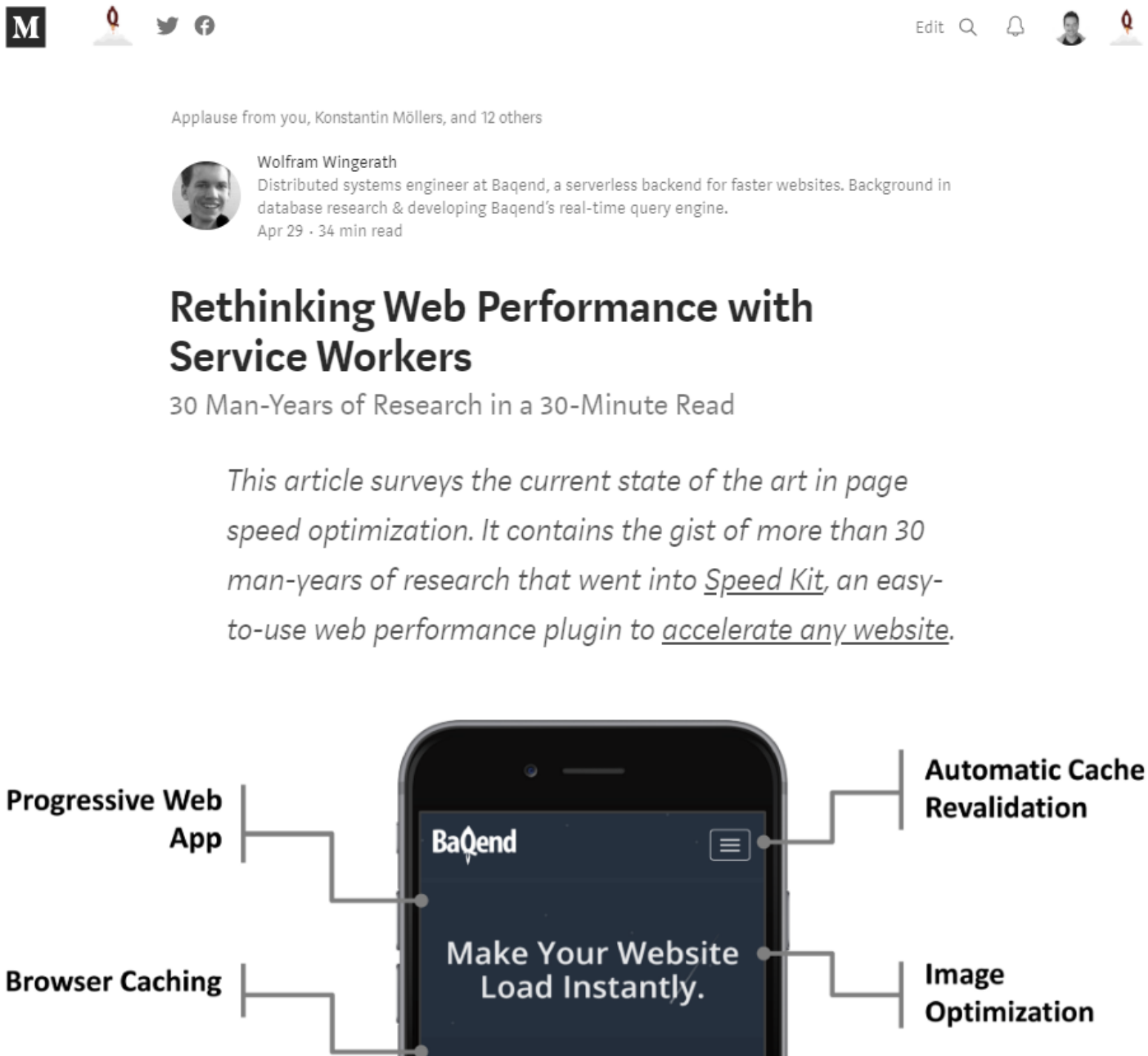
Powerful
programmable
network proxy

Use Case

Speed Kit:
Smart CDN though
Service Workers

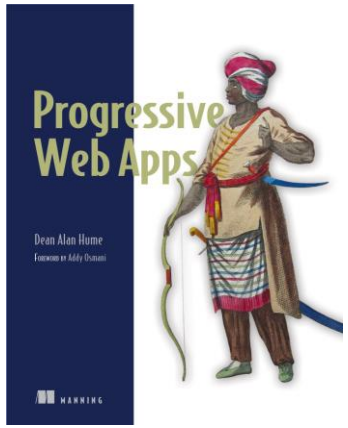
Learn more about this topic:

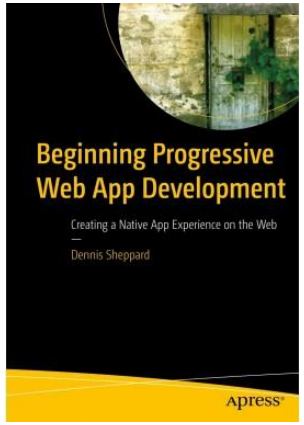
https://blog.baqend.com/



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

Catch our other talks!

14:00 - Kino 7 - Buzzing Technologies

Creating High-Performance Web Apps with WebAssembly

15:00 - Kino 6 - Architecture

Real-Time Processing Explained: A Survey of Storm, Samza, Spark & Flink

BaQend Felix Gessert · fg@baqend.com · www.baqend.com