The Technology Behind Progressive Web Apps

Service Workers in Detail

Erik Witt
ew@baqend.com

September 12, techcamp 2018, Hamburg

@baqendcom

BaQend

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

What we are going to cover.

PWA

Core Features
Building Blocks
Implementation

Service Workers

Lifecycle
Network Interception
Caching Strategies

Speed Kit

Cache Coherence
Performance-
Measures

Why do(n‘t) we love native apps?

Progressive Web Apps

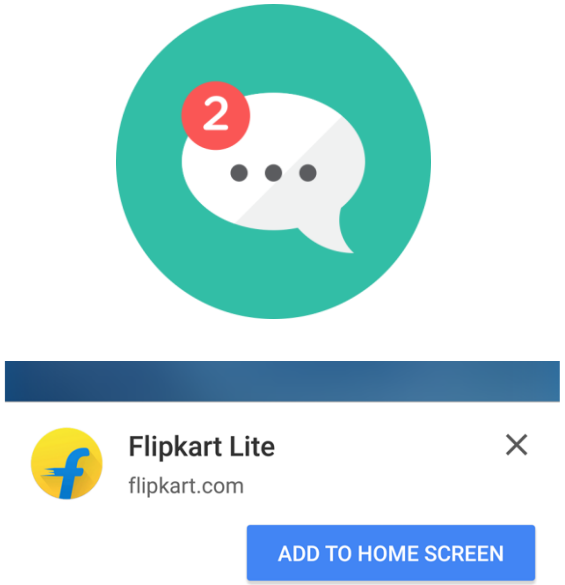
seek to combine the great from **native** and **web apps**

What are Progessive Web Apps?

Progressive Web Apps (PWAs)

+

+



Fast **Loads** through Caching

Offline Mode (Synchronization)

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

Try this:

www.baqend.com

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:

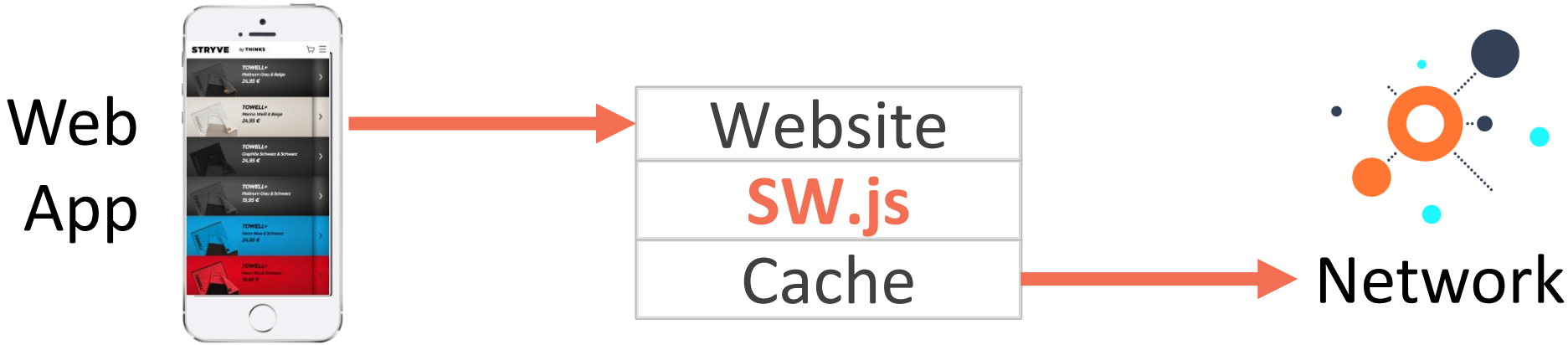
```
<link rel="manifest" href="/manifest.json">
{
  "short_name": "Codetalks PWA",
  "icons": [
    {"src": "icon-1x.png", "type": "image/png", "sizes": "48x48"}],
  "start_url": "index.html?launcher=true"
}
```

Implementing PWAs

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

Gracefully degrade when not supported

2. Service Workers for caching & offline mode:

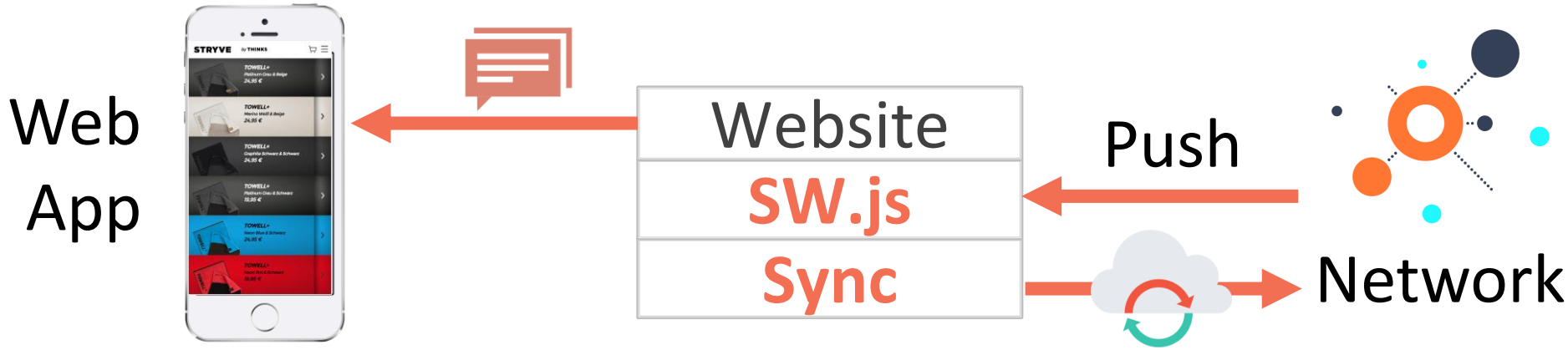


Implementing PWAs

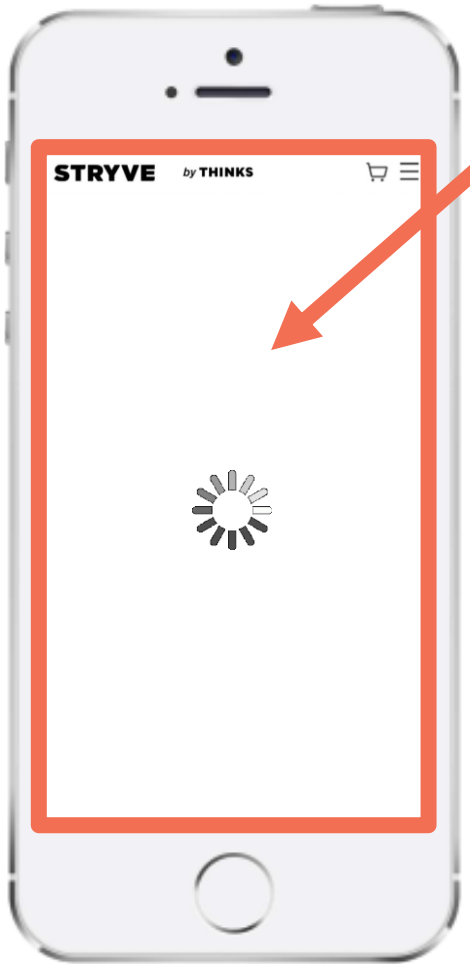
PWAs are **best practices** and **open web standards**

Progressively enhance the user experience

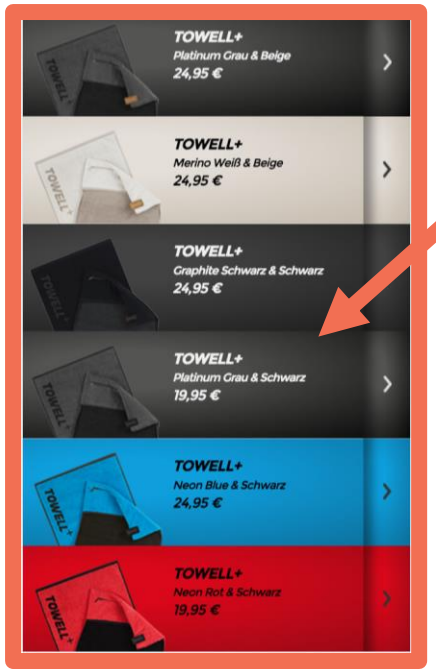
3. Add Web Push and Background Sync:



Typical Architecture: App Shell Model



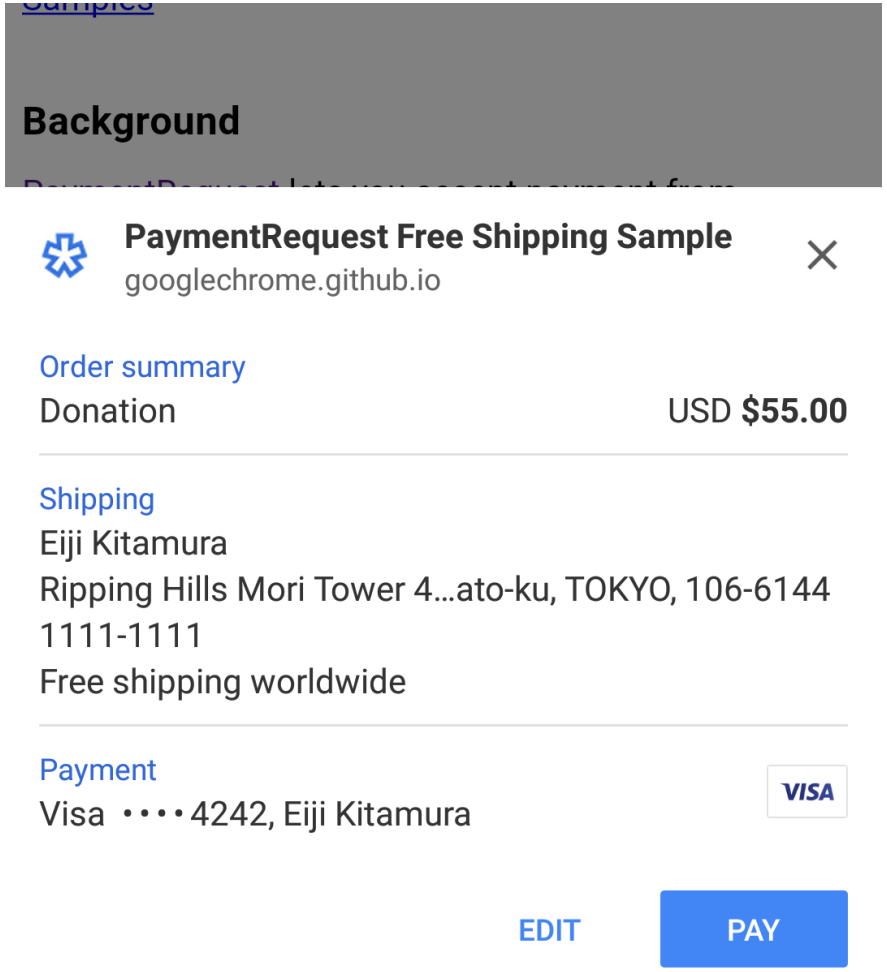
App Shell: **HTML**, **JS**, **CSS, images** with app logic & layout



Content: Fetched on demand & may change more often


What is the future of
Progessive Web Apps?

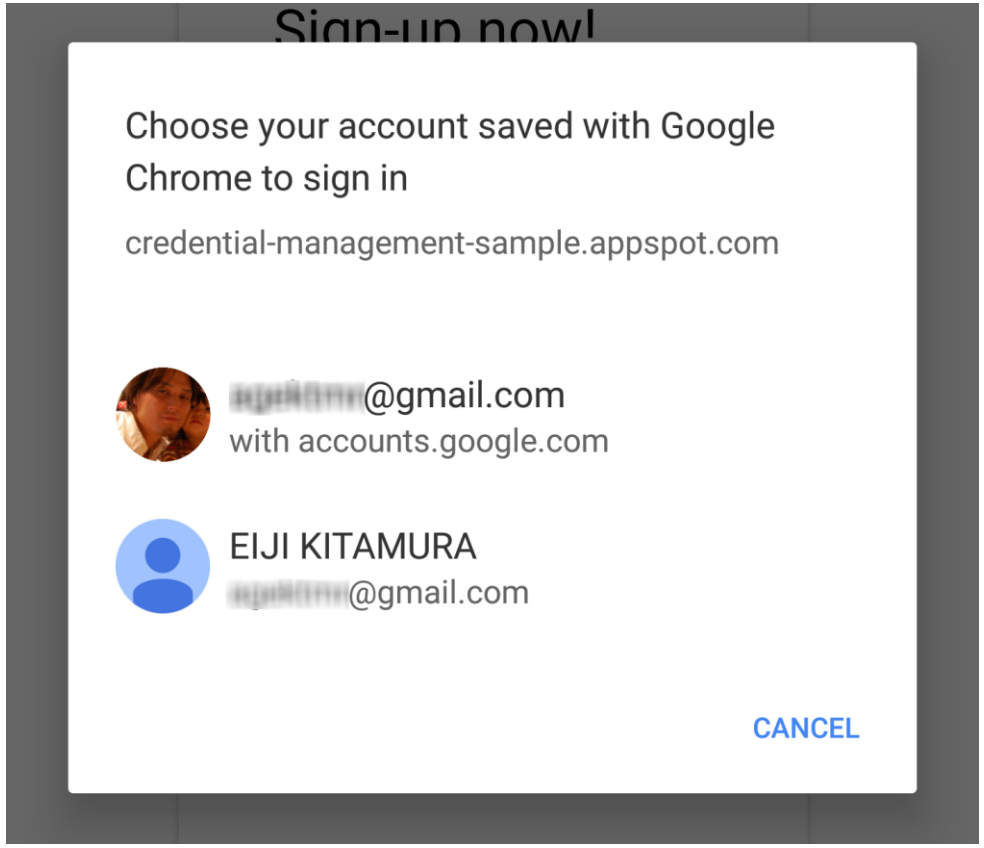
The Future of PWAs is bright.



Payment Request API

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

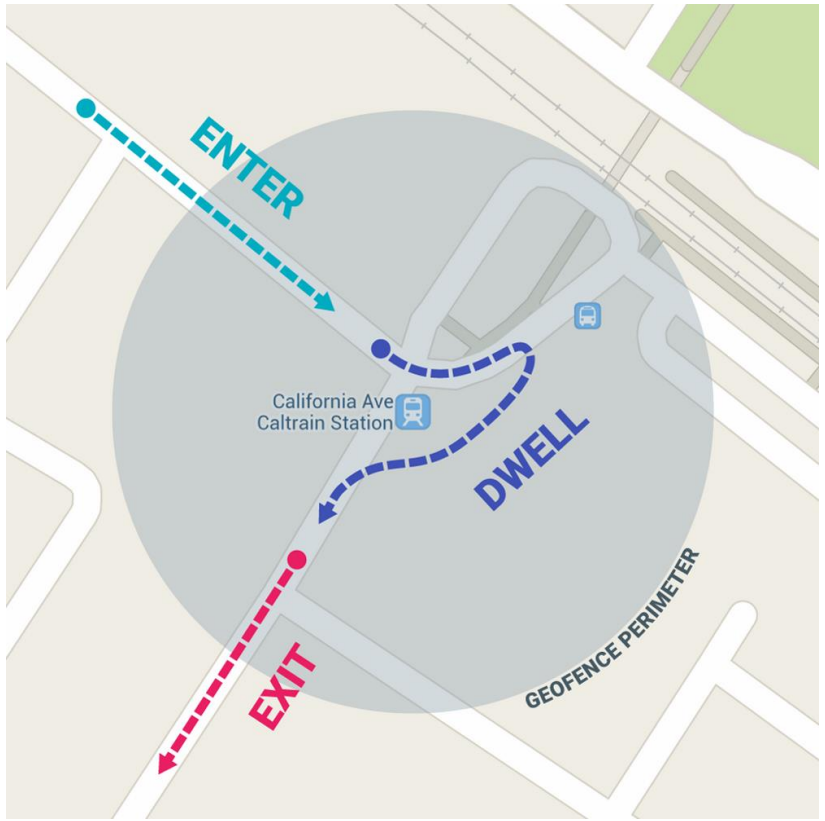
The Future of PWAs is bright.



Credentials Management API

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

The Future of PWAs is bright.



Geofencing

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

The Future of PWAs is bright.

Web Speech API

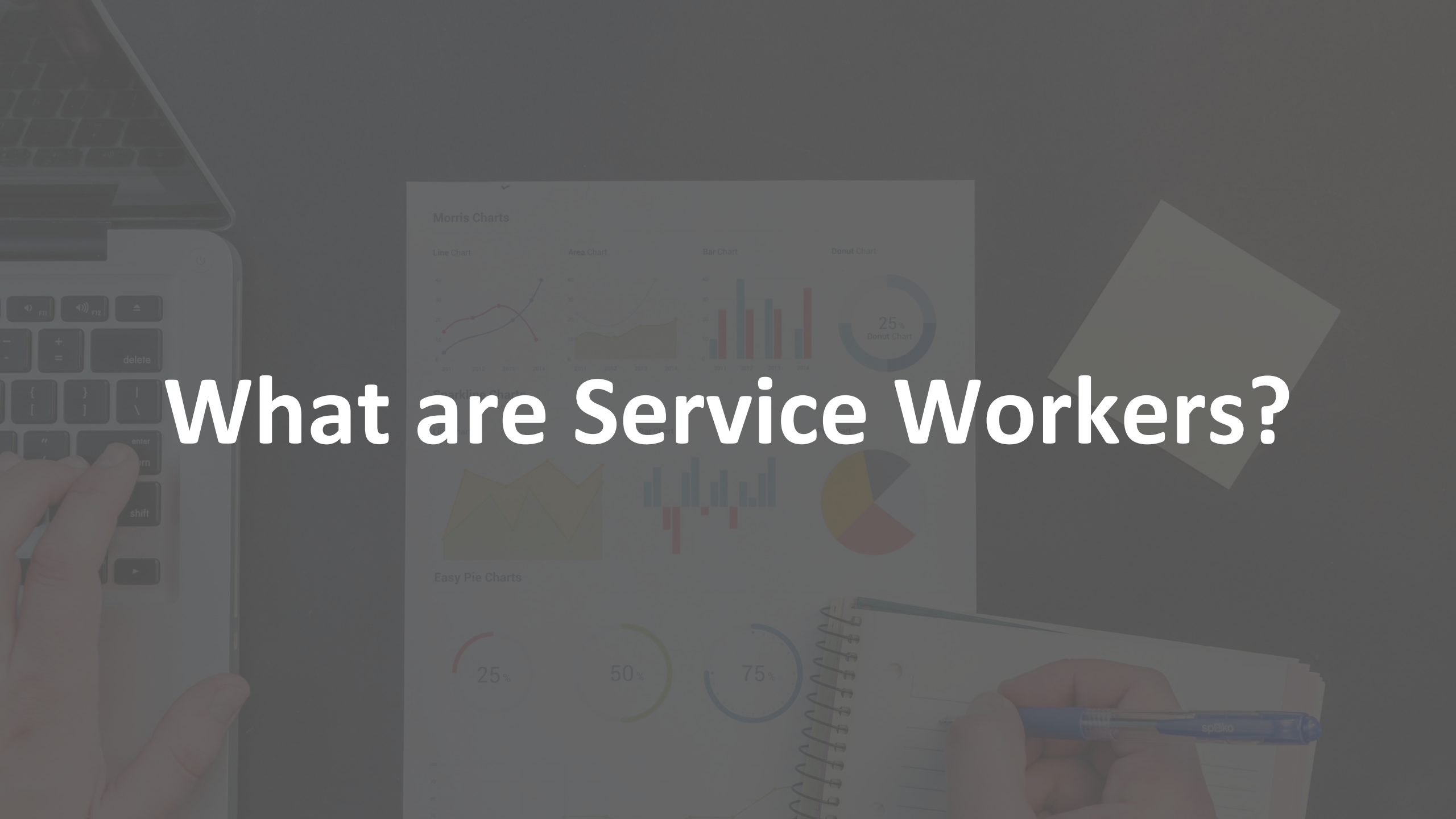
Native Speech Recognition in the Browser:

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

The Future of PWAs is bright.

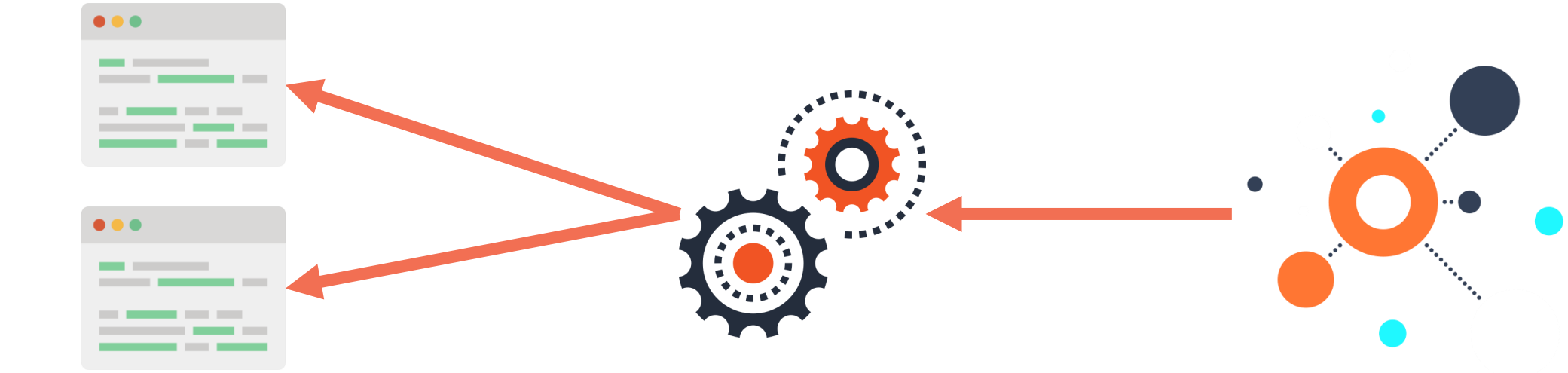
Web Share API

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



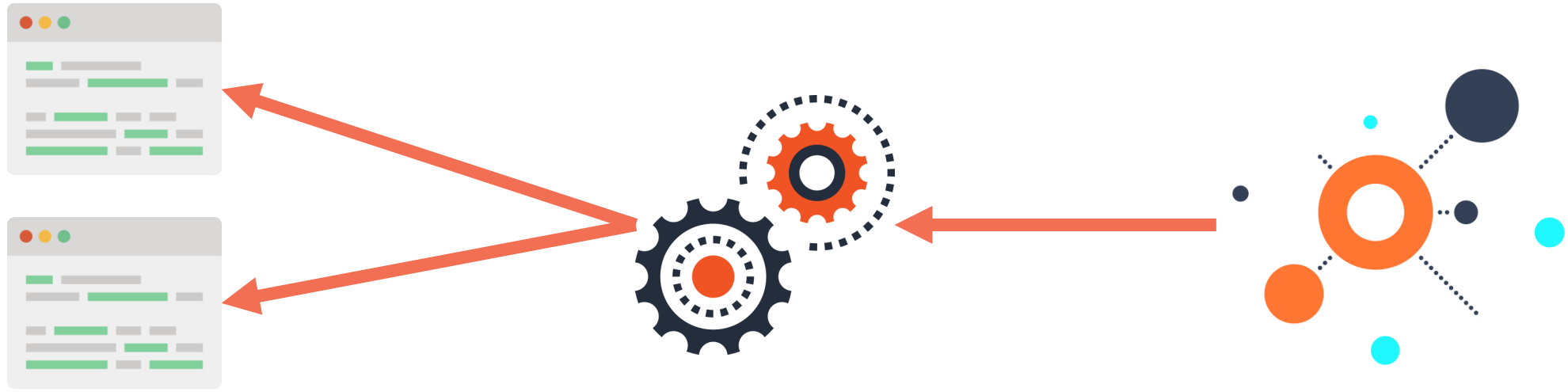
What are Service Workers?

What are Service Workers?



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

What do Service Workers do?



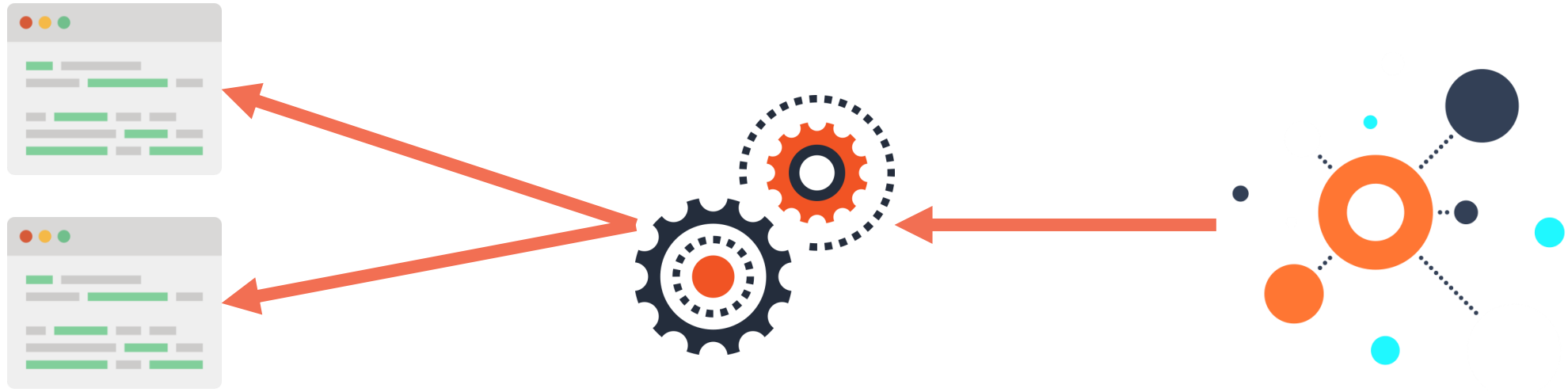
Browser Tabs

Service Worker

Network

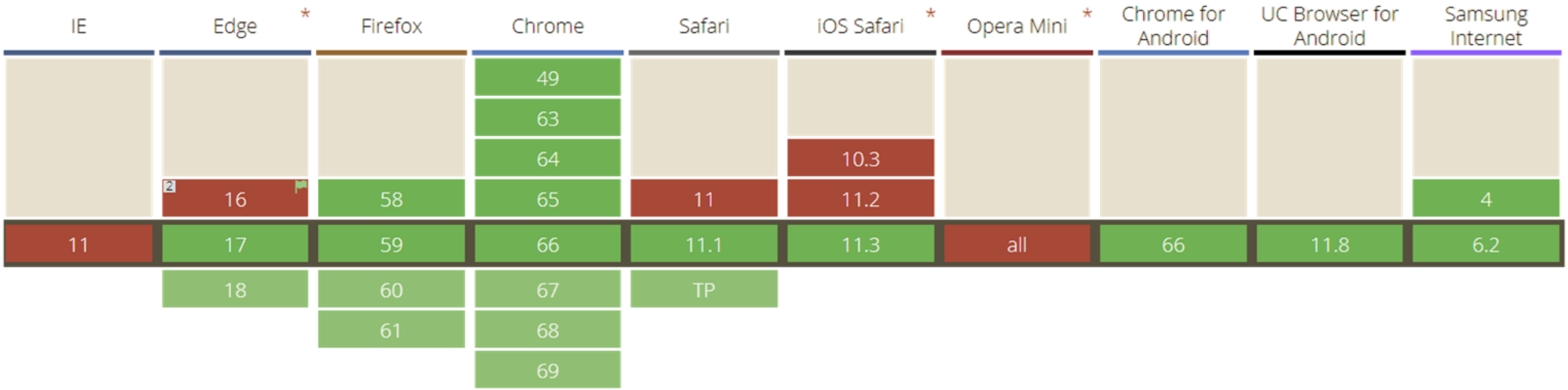
- **Cache** Data (`CacheStorage`)
- **Store** Data (`IndexedDB`)
- Receive **Push**
- Respond when **Offline**

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 **~90%** of browsers.

Requires **TLS Encryption**.

Late, but all in: Microsoft

Publish PWAs to
Microsoft Store

PWA BUILDER

or

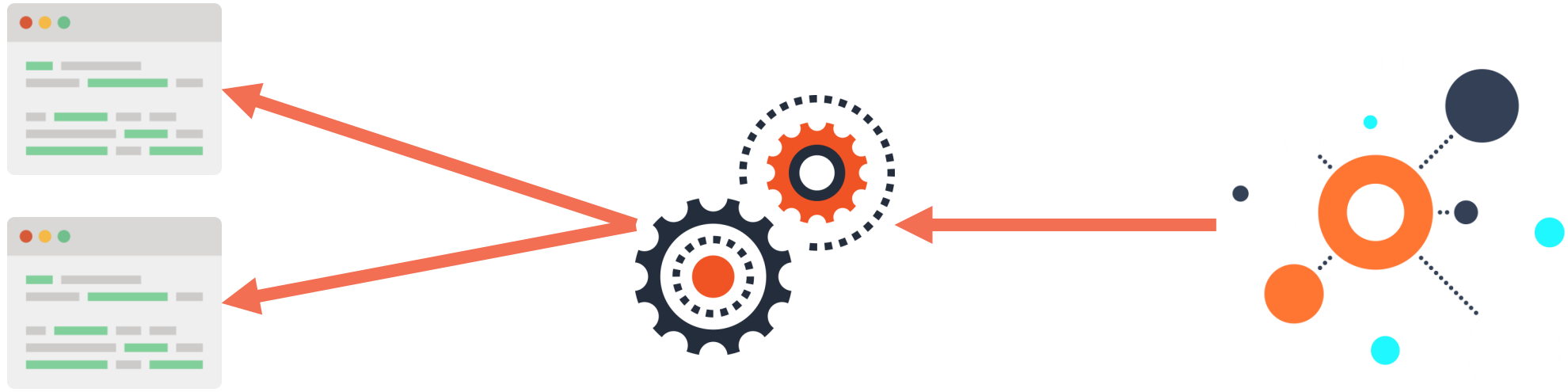
Bing Crawls PWAs

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?



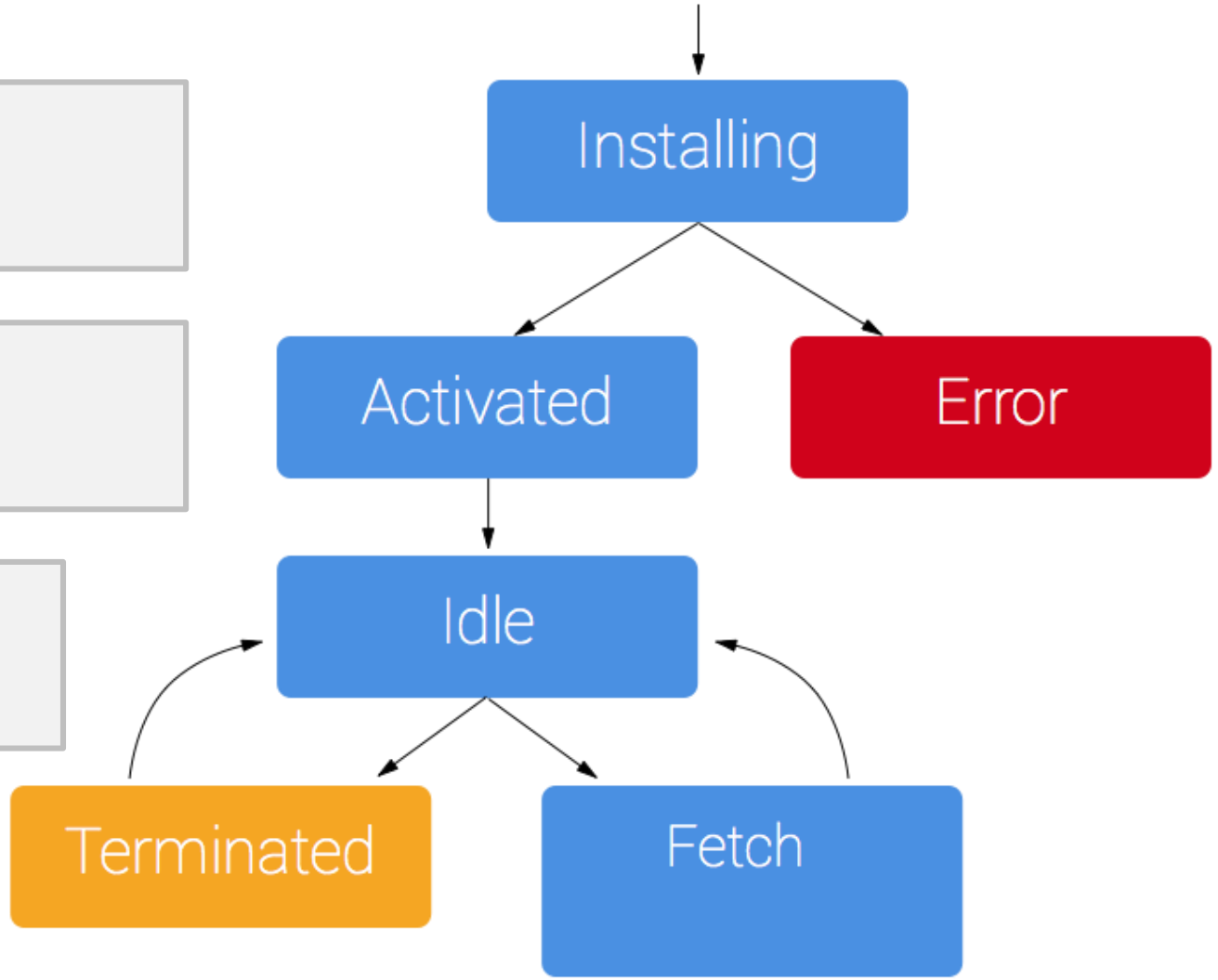

```
<script>
    navigator.serviceWorker.register('/sw.js');
</script>
```

How does the Lifecycle look like?

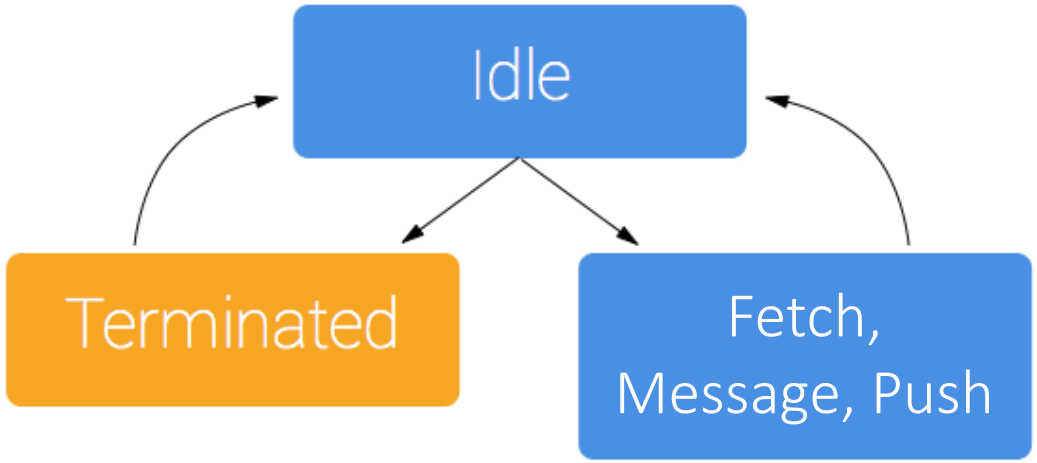
```
self.addEventListener('install', (event) => {
  // Perform install steps
});
```

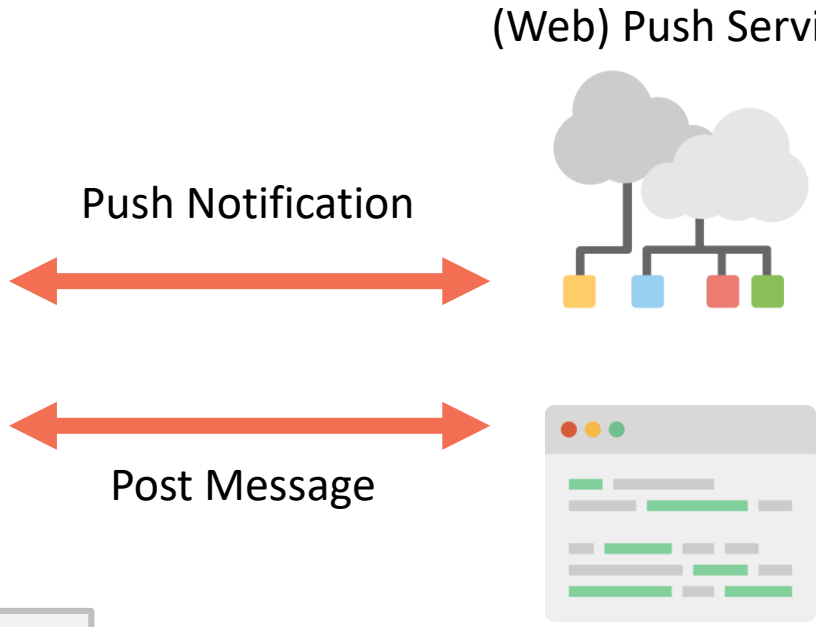
```
self.addEventListener('activate', (event) => {
  // Perform activate steps
});
```

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



How to Communicate with Service Workers?



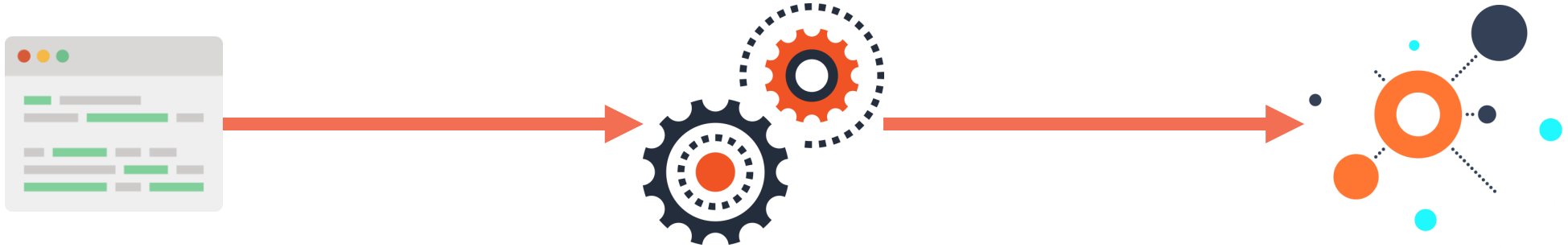



```
self.addEventListener('message', (event) => {
  // Receive message
});
```

```
// 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** Request
- **Change** Response
- **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

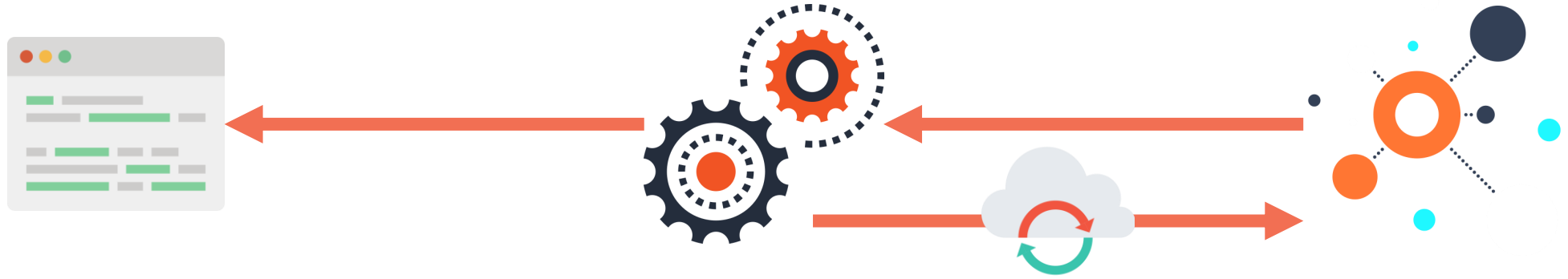
IndexedDB

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

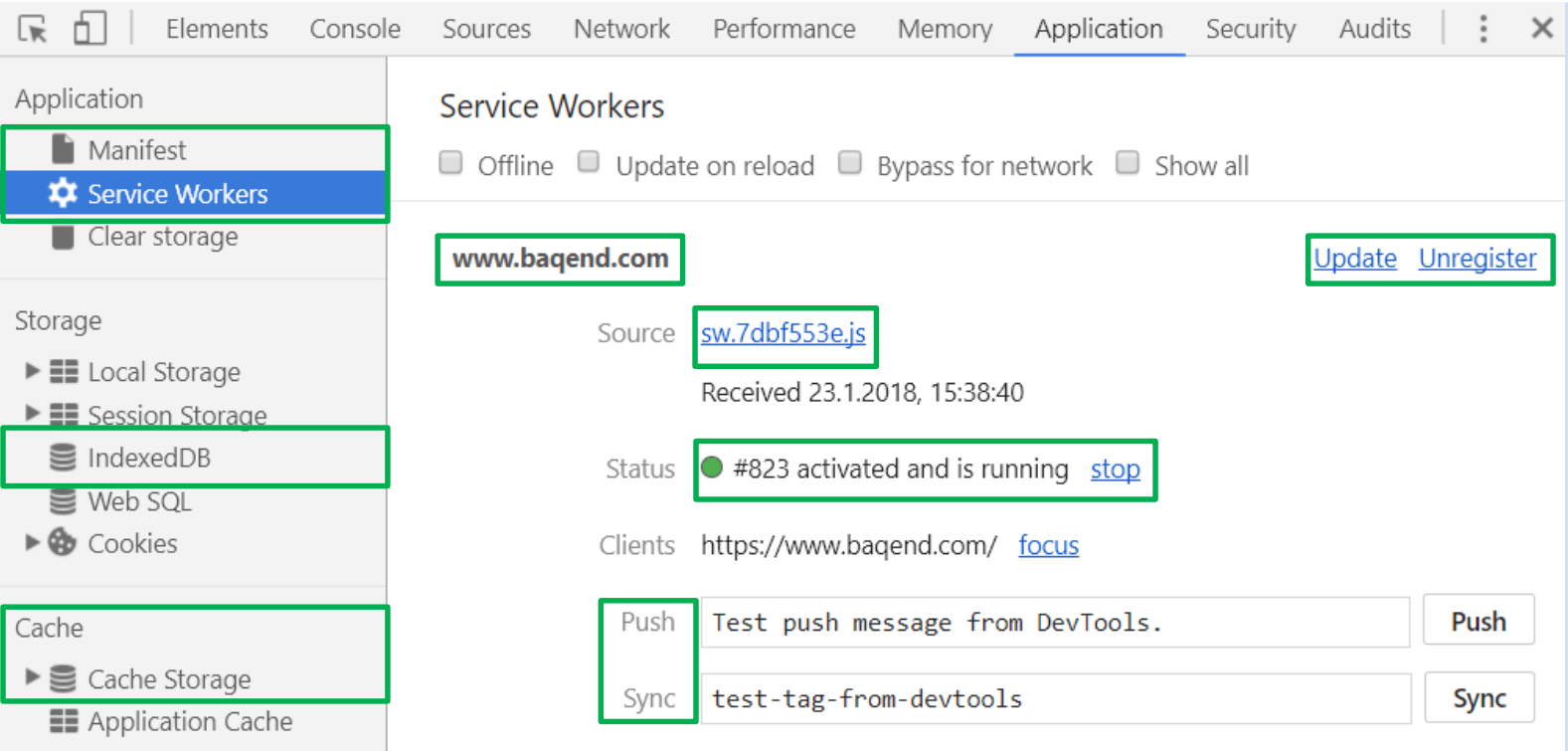
Periodic Sync

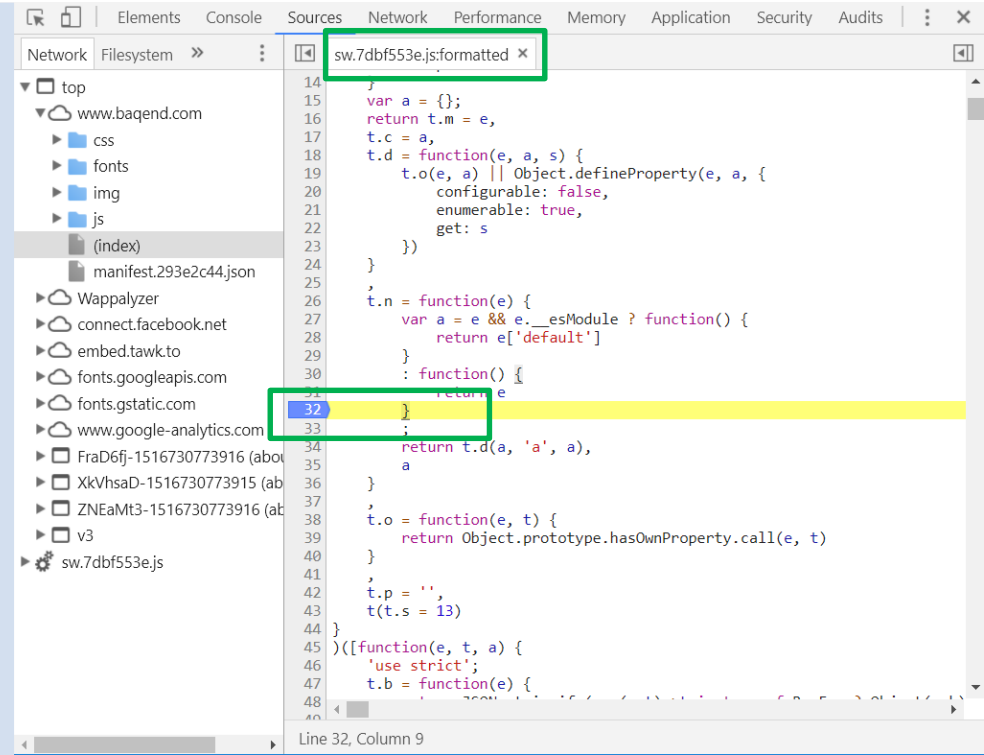
- executed when online, according to **period options**

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)

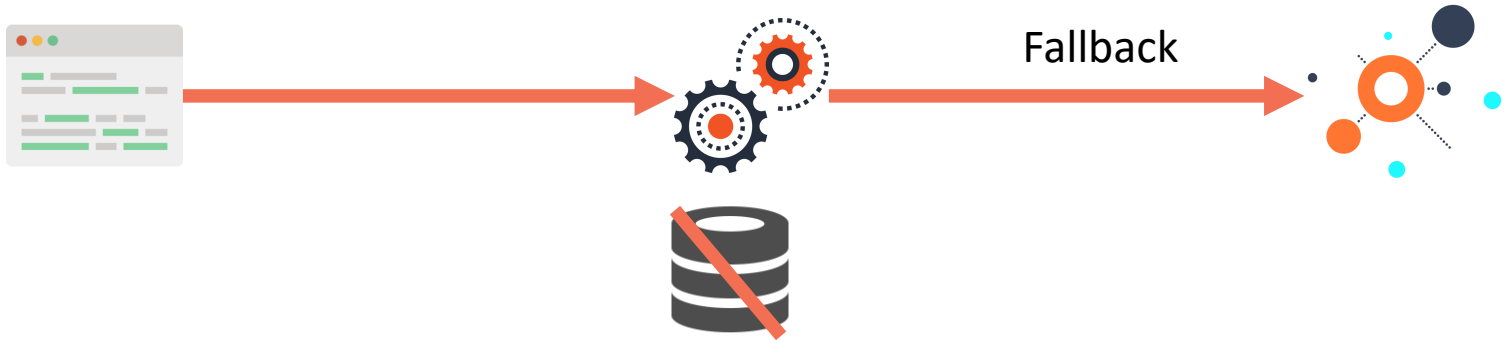
Caching Strategies – Cache Only

Gets all requests from cache or fails.

Caching Strategies – Cache, Network Fallback

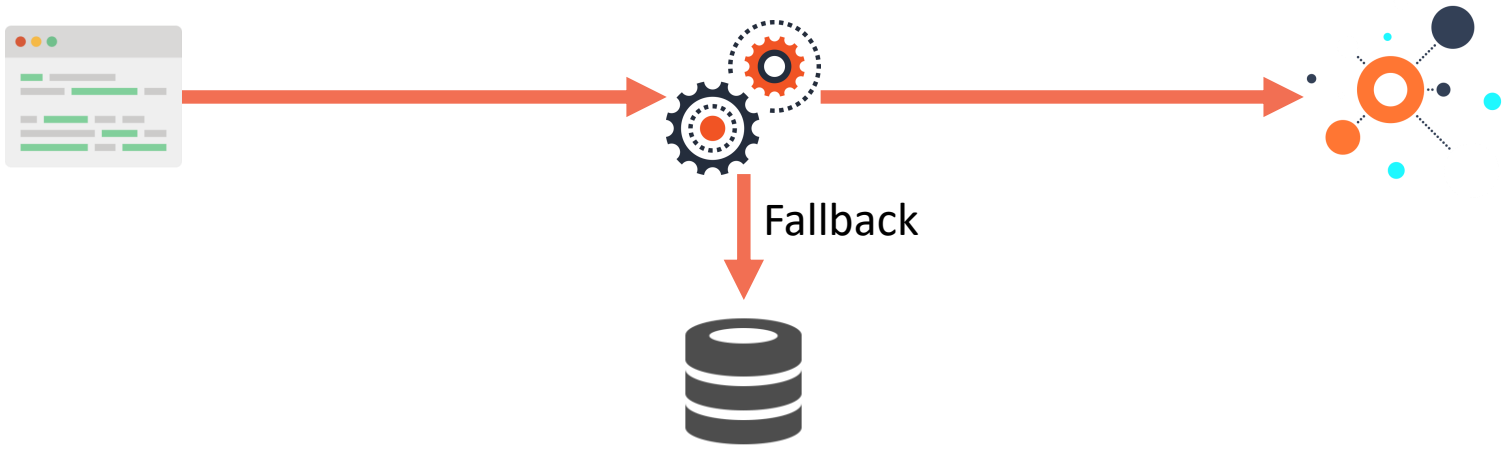
Gets requests from cache & uses network as fallback.

Caching Strategies – Network Only



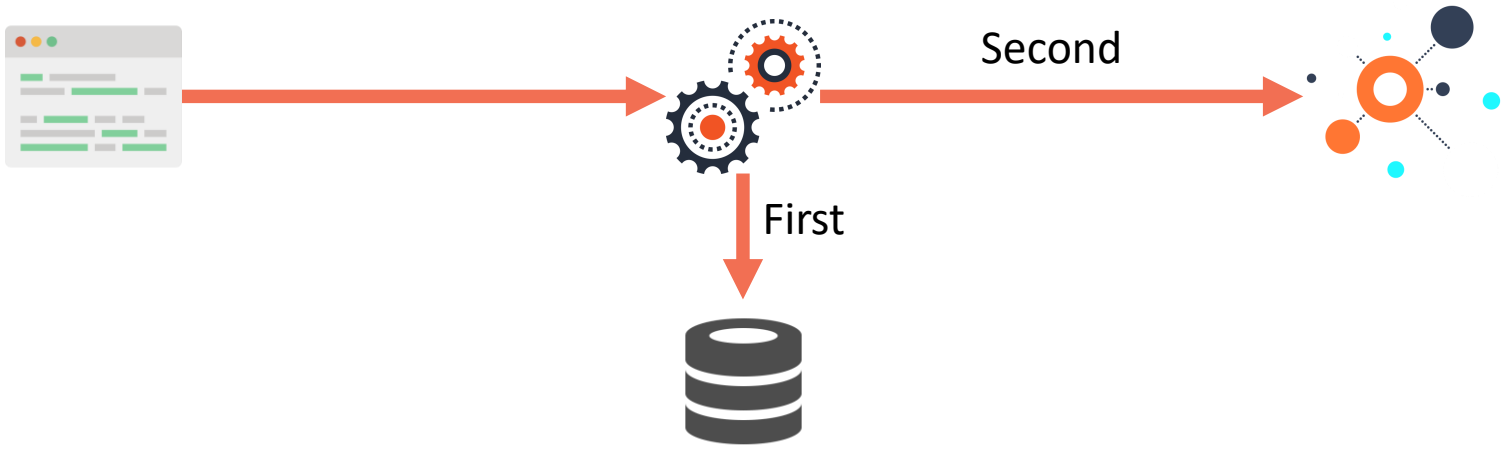
Gets requests from network only.

Caching Strategies – Network, Cache Fallback



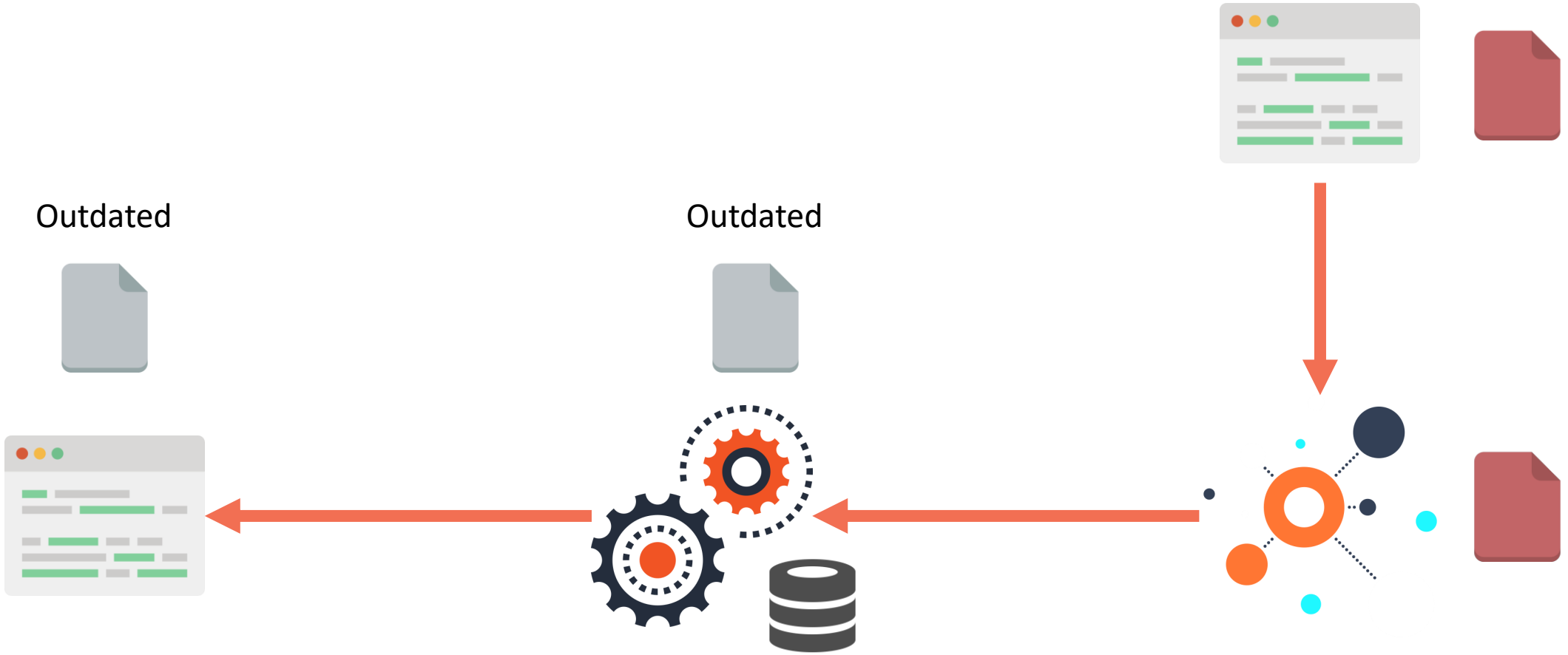
Gets requests from network, the cache acts as fallback (offline mode).

Caching Strategies – Cache, then Network



Gets requests from cache first and from network in background.

Major Challenge: Cache Coherence

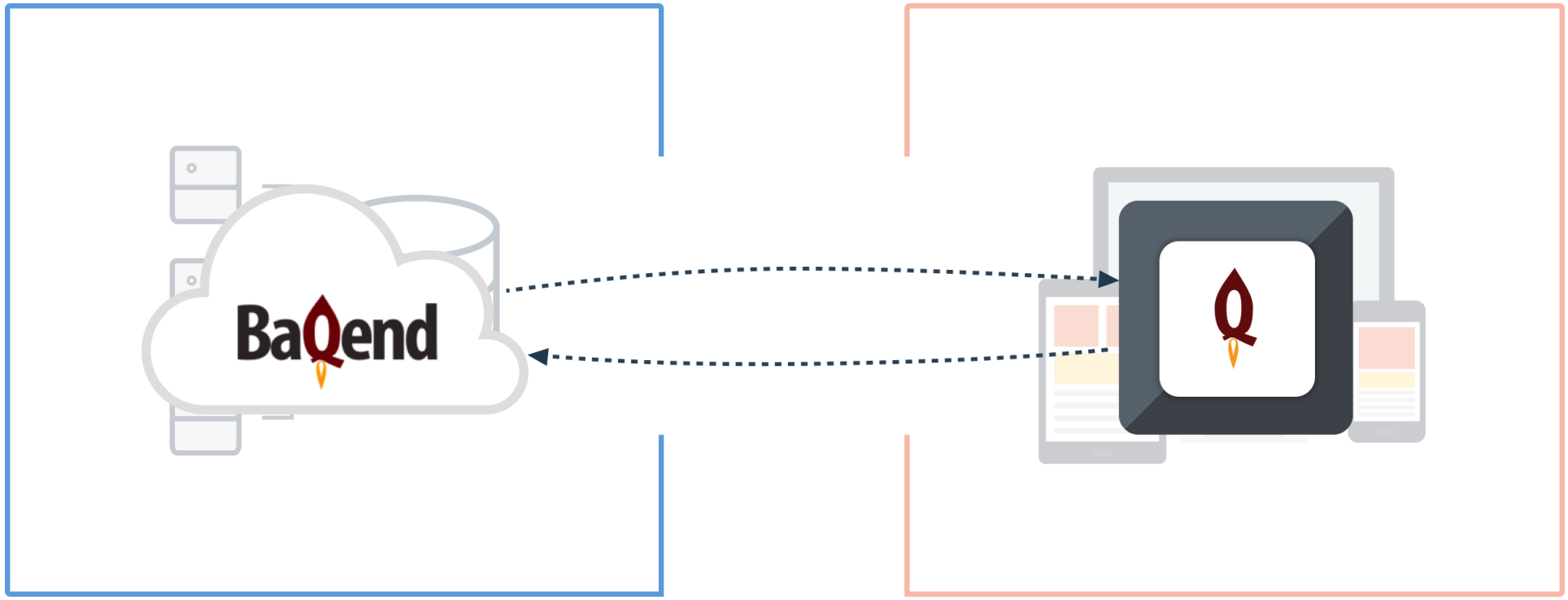


All strategies either serve **outdated data** or **degrade performance**

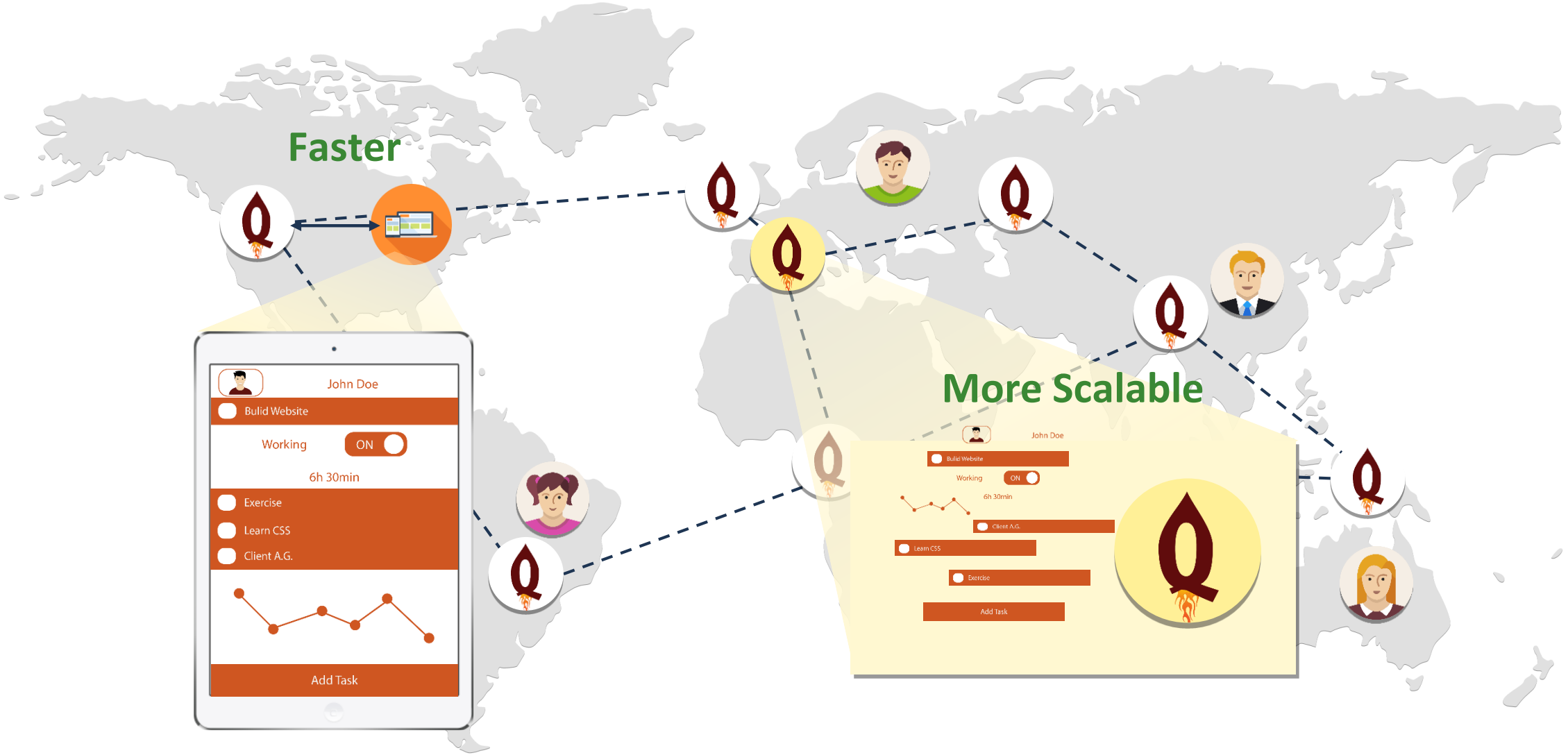
What we do with Service Workers

Speed Kit

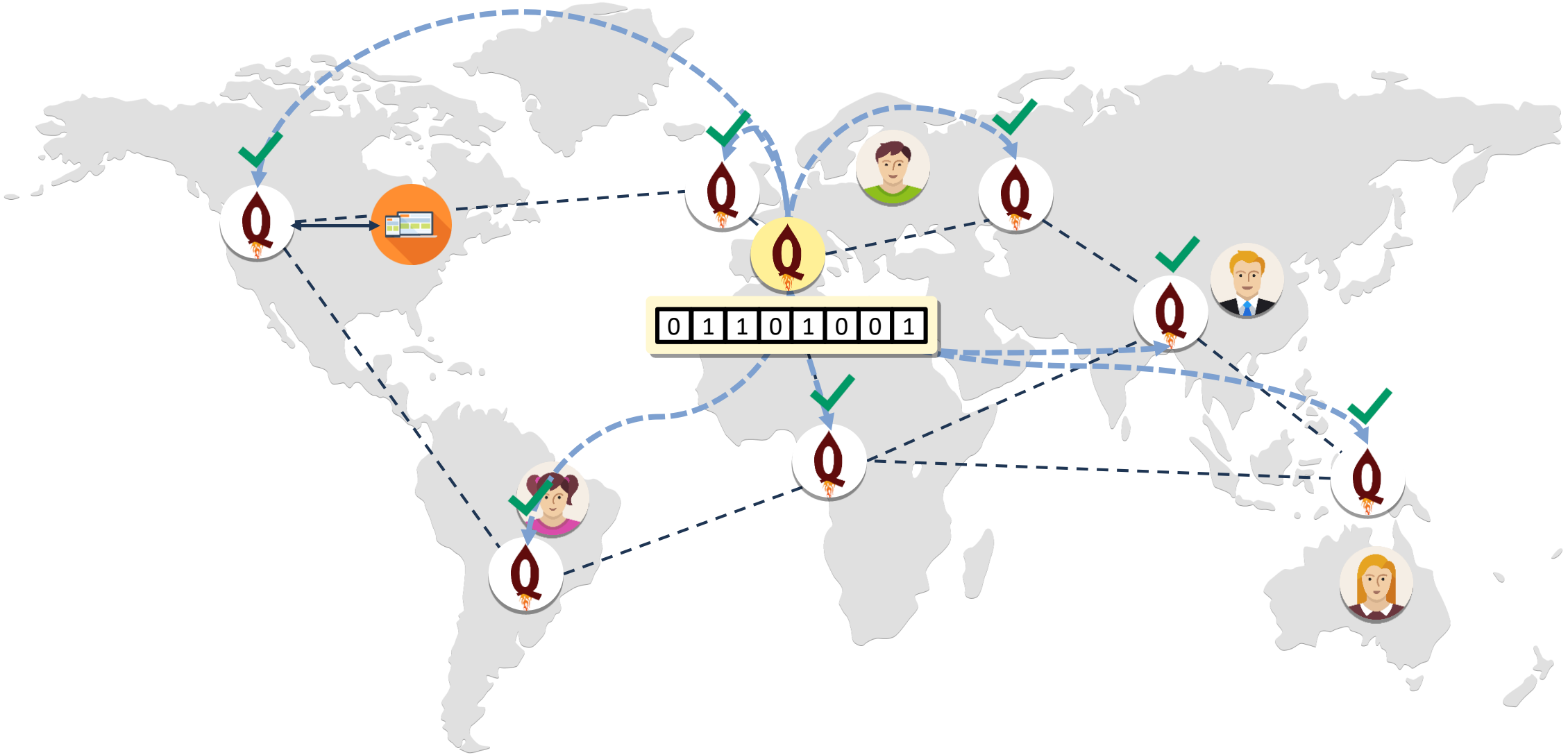
Turning Websites into Instantly-Loading Progressive Web Apps



What Speed Kit does.



What Speed Kit does.



What Speed Kit does.

F. Gessert, F. Bücklers, und N. Ritter, „ORESTES: a Scalable Database-as-a-Service Architecture for Low Latency“, in *CloudDB 2014*, 2014.

F. Gessert und F. Bücklers, „ORESTES: ein System für horizontal skalierenden Zugriff auf Cloud-Datenbanken“, in Informatiktage 2013, 2013.

F. Gessert und F. Bücklers, *Performanz- und Reaktivitätssteigerung von OODBMS vermittels [illegible]*. Bachelorarbeit, 2010.

M. Schaarschmidt, F. Gessert, und N. Ritter, „Towards Automated Polyglot Persistence“, in BTW 2015.

S. Friedrich, W. Wingerath, F. Gessert, und N. Ritter, „NoSQL OLTP Benchmarking: A Survey“, in *44. Jahrestagung der Gesellschaft für Informatik*, 2014, Bd. 232, S. 693–704.

W. Wingerath, F. Gessert, S. Friedrich, N. Ritter „Real-time stream processing for Big Data“, *Big Data Analytics it - Information Technology,* 2016

F. Gessert, W. Wingerath, S. Friedrich, N. Ritter “NoSQL Database Systems: A Survey and Decision Guidance”, *Computer Science - Research and Development,* 2016

F. Gessert, S. Friedrich, W. Wingerath, M. Schaarschmidt, und N. Ritter, „Towards a Scalable and Unified REST API for Cloud Data Stores“, in *44. Jahrestagung der GI*, Bd. 232, S. 723–734.

F. Gessert, M. Schaarschmidt, W. Wingerath, S. Friedrich, und N. Ritter, „The Cache Sketch: Revisiting Expiration-based Caching in the Age of Cloud Data Management“, BTW 2015.

F. Gessert und F. Bücklers, *Kohärentes Web-Caching von Datenbankobjekten im Cloud Computing*. Masterarbeit 2012.

W. Wingerath, S. Friedrich, und F. Gessert, „Who Watches the Watchmen? On the Lack of Validation in NoSQL Benchmarking“, in BTW 2015.

F. Gessert, „Skalierbare NoSQL- und Cloud-Datenbanken in Forschung und Praxis“, BTW 2015

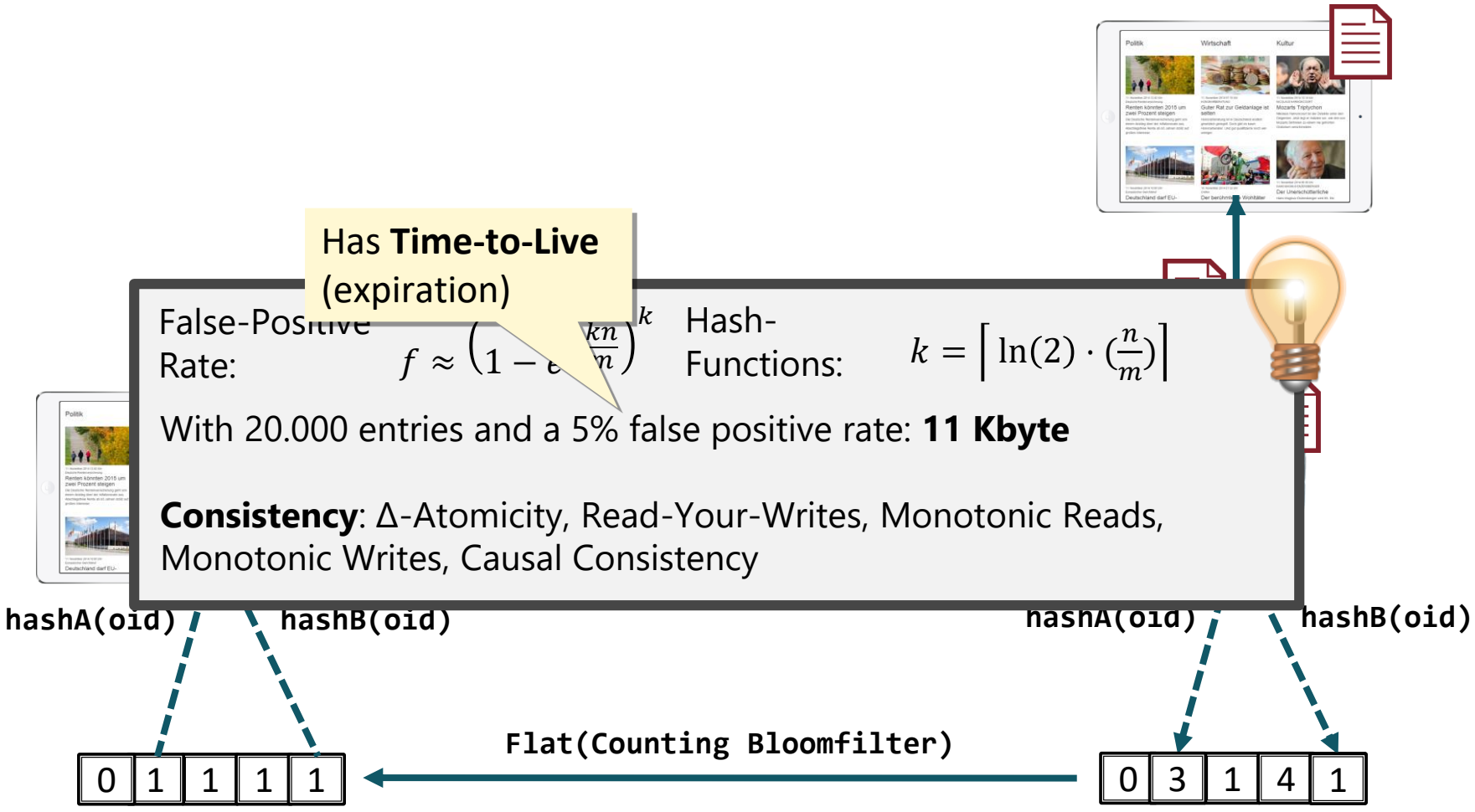
F. Gessert, N. Ritter „Scalable Data Management: NoSQL Data Stores in Research and Practice“, *32nd IEEE International Conference on Data Engineering, ICDE,* 2016

F. Gessert, N. Ritter „Polyglot Persistence“, *Datenbank Spektrum*, 2016.

Backed by

30 Man-Years of Research

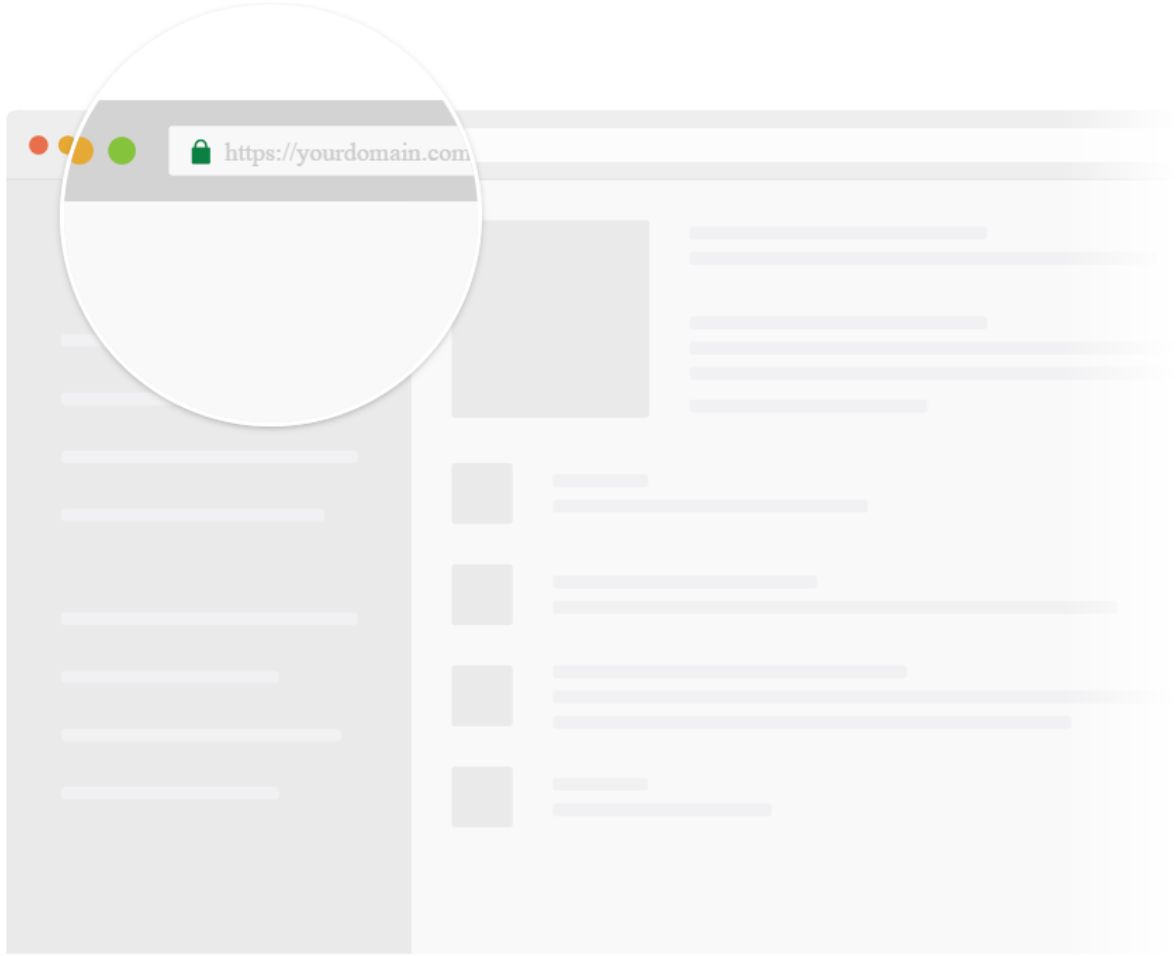
How Speed Kit solves Cache Coherence



Adding Speed Kit to a Site

1. Configure Domain

Set which sites/URLS Baqend should accelerate (white- and blacklist, dynamic blocks).

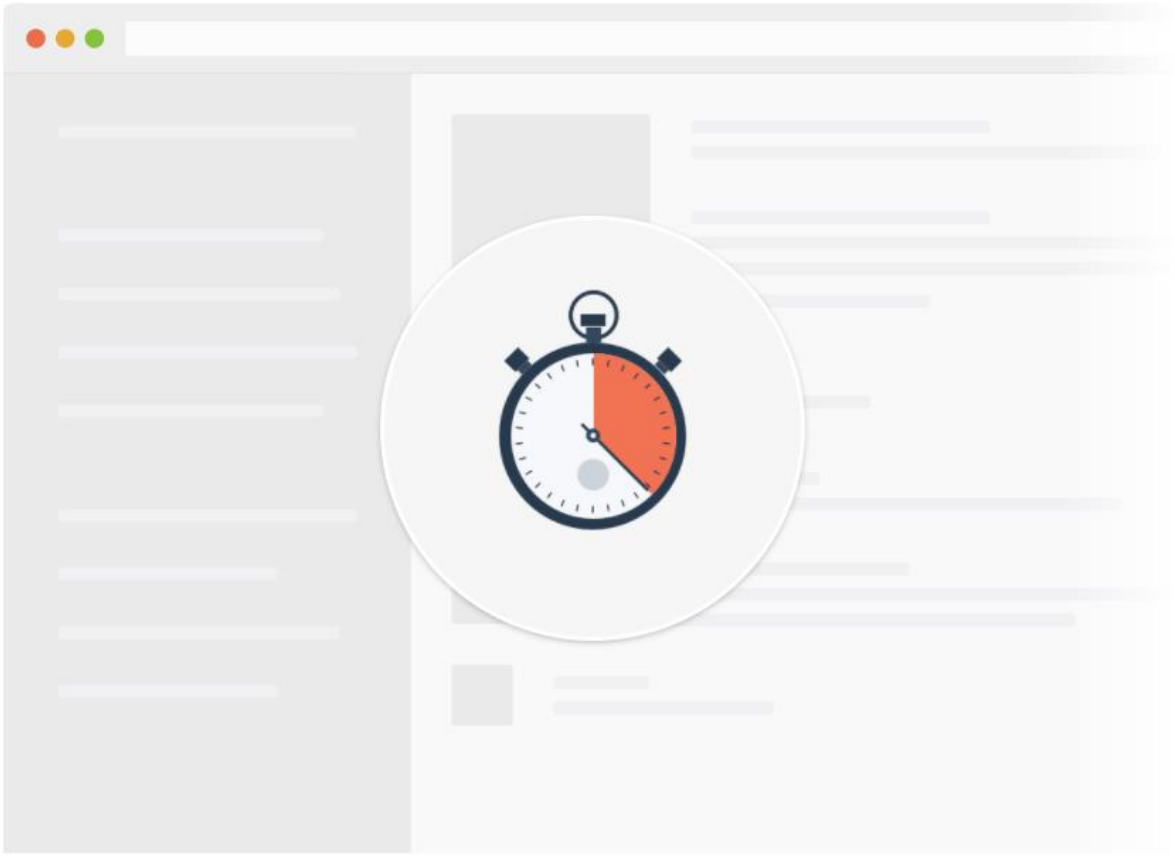


2. Include Code Snippet

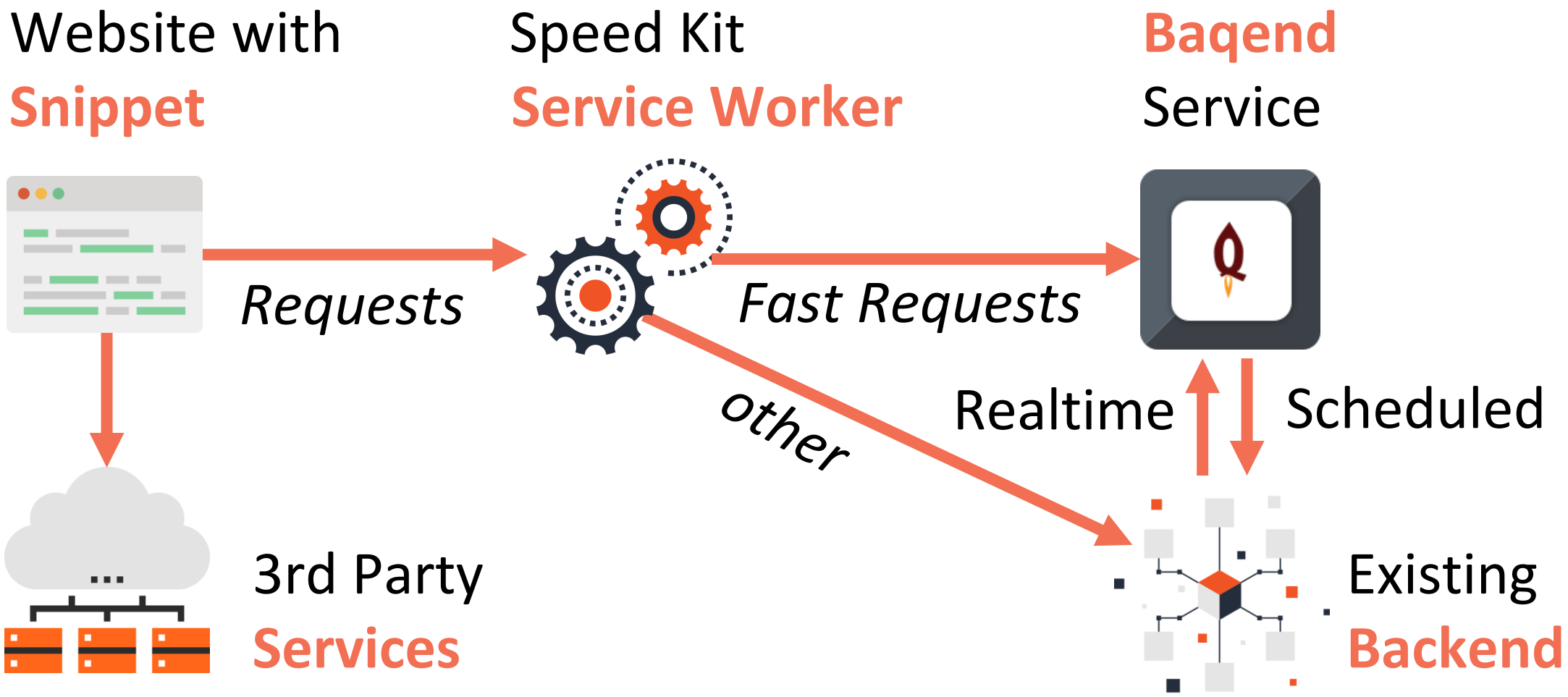
Add Speed Kit to your website's HTML and upload the Service Worker.

3. Speed Kit is Active

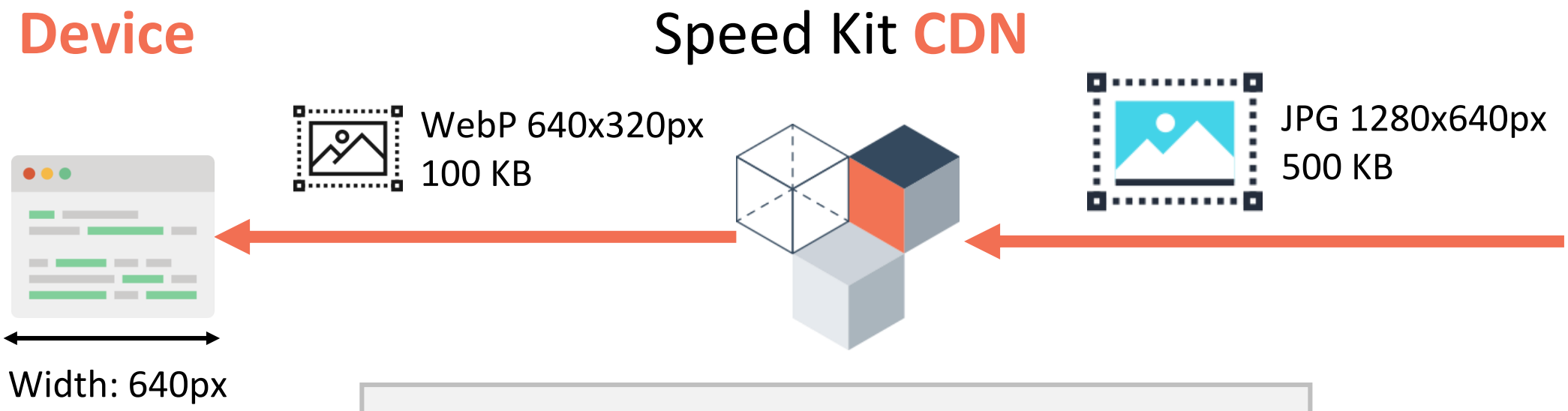
Speed Kit intercepts requests and serves them through Baqend's infrastructure.



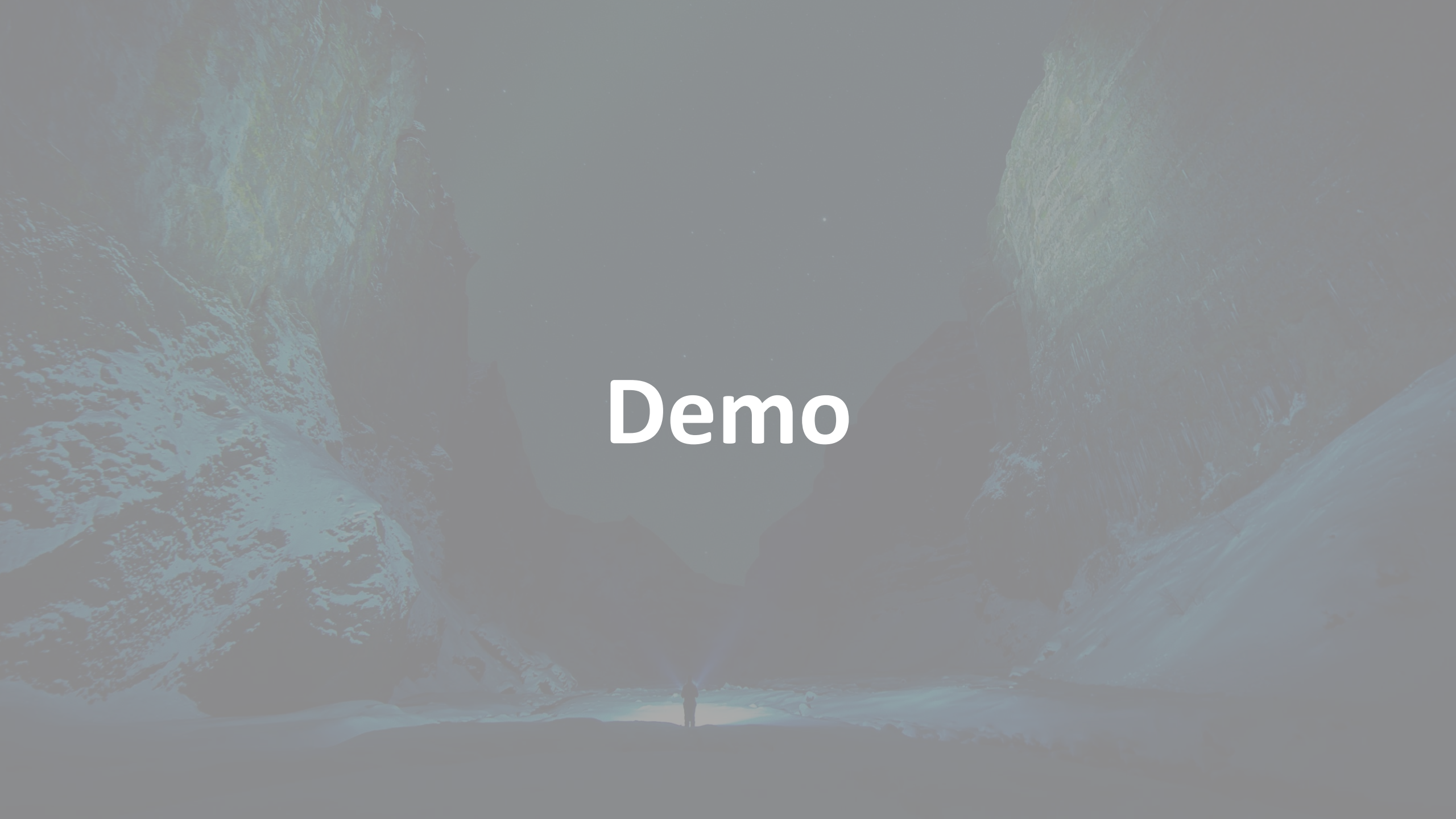
How it works under the hood



Optimized & Cached Images



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


Demo

Now, we have a Progressive Web App.

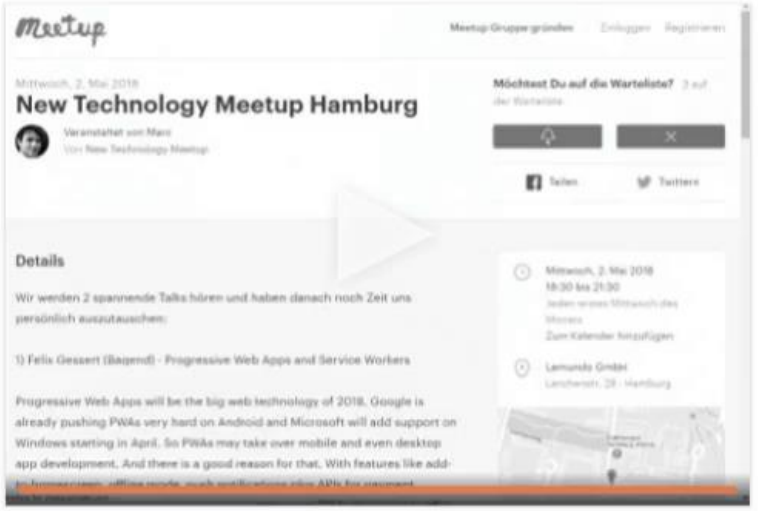
How do we measure web performance?

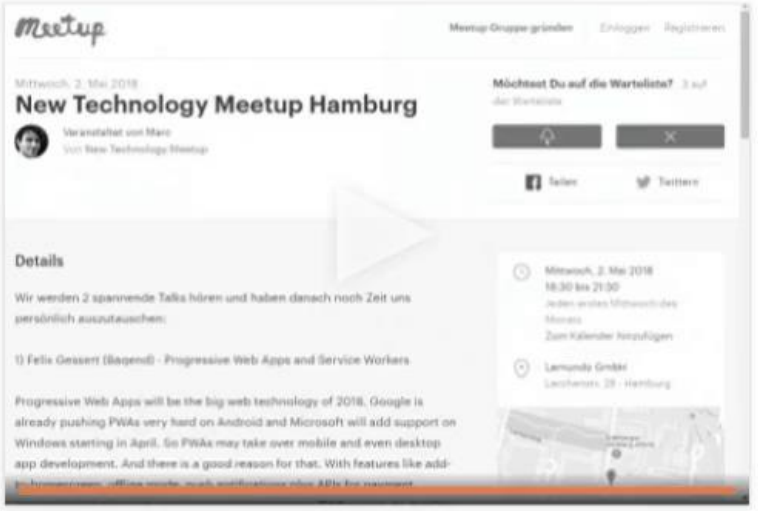
https://www.meetup.com/de-DE/meetup-group-bdCSNBEI/events/249149394/

Your Website
1131ms

3.1x Faster

With Speed Kit
366ms





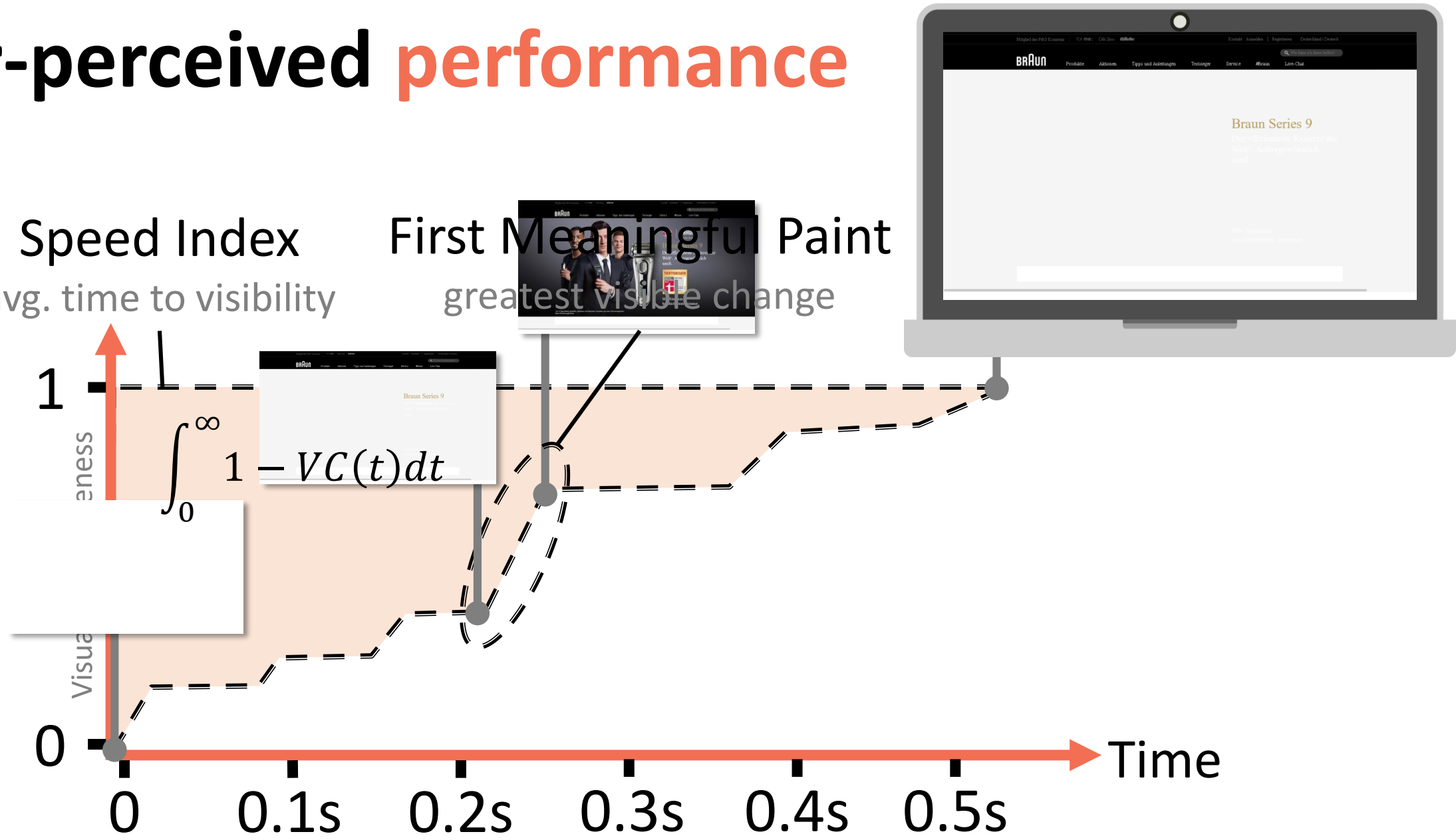
With Speed Kit 0.4s

Your Website 1.1s

Excellent — Poor

Show Details

User-perceived performance



Does it work for Your Site?

test.speed-kit.com

Wrap Up

PWA

Super cool alternative to native apps

Service Workers

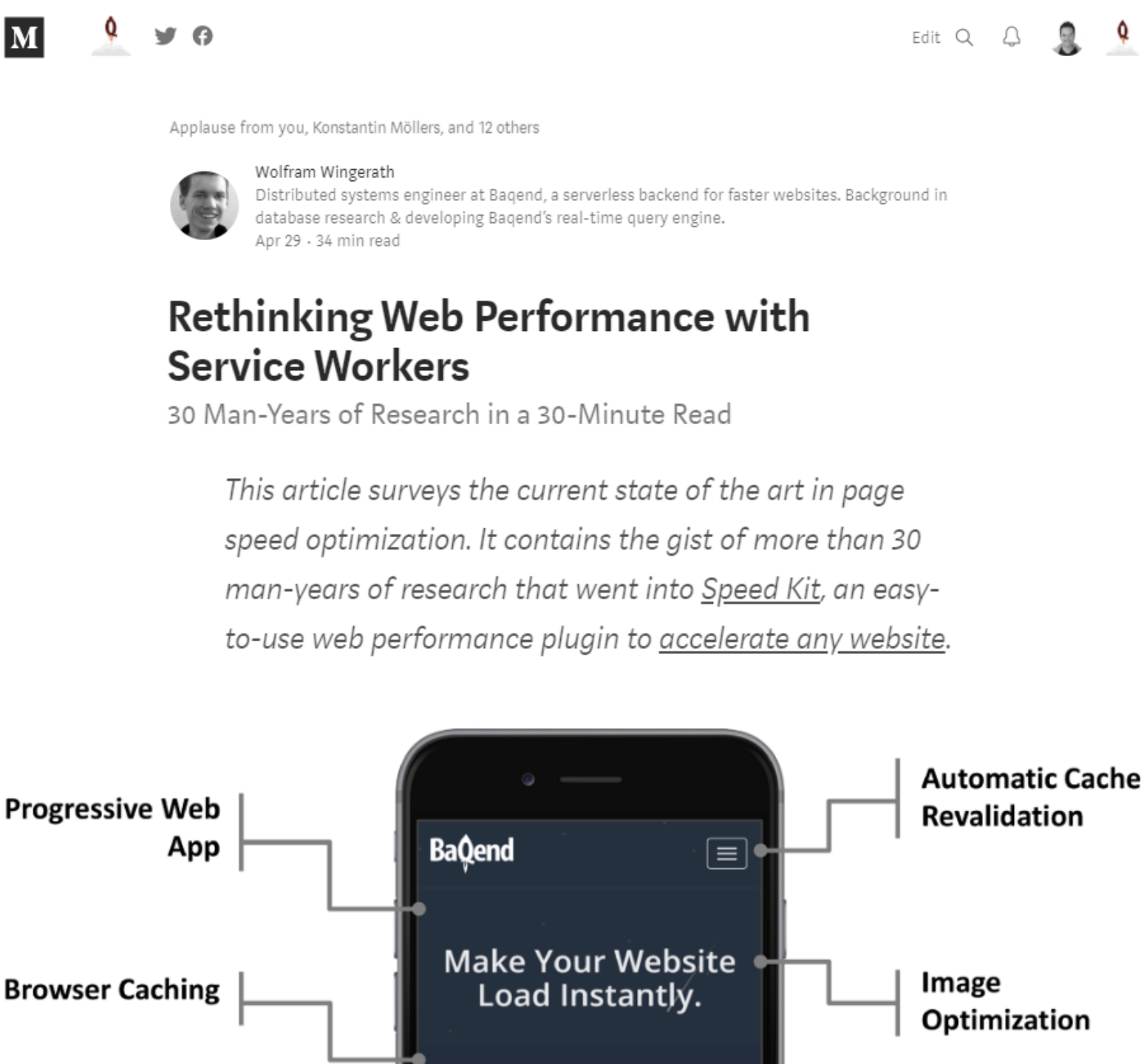
Powerful programmable network proxy

Speed Kit

Combines Service Workers & cache coherence

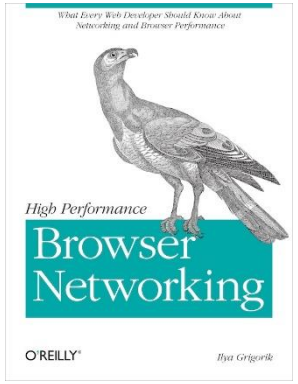
Learn more about this topic:

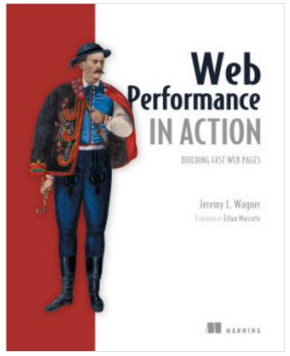
https://blog.baqend.com/

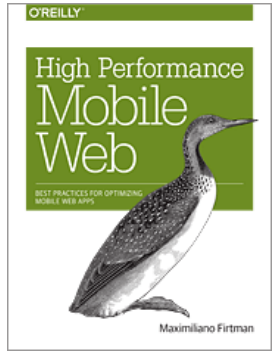


Web Performance Literature

Good Resources







https://hpbn.co/

Google Developers Performance

https://developers.google.com/web/fundamentals/performance/?hl=en

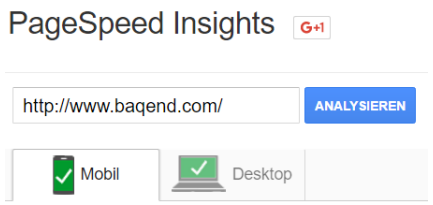
Website Performance Optimization
The Critical Rendering Path

https://www.udacity.com/course/website-performance-optimization--ud884

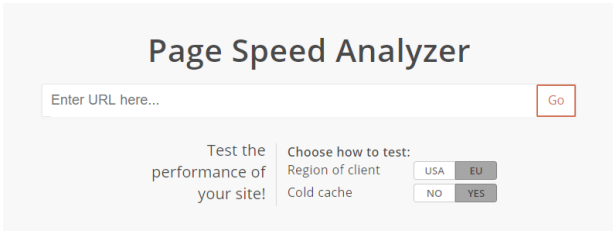
Baqend Blog
On Building a Faster Web

https://medium.baqend.com/

Performance Tools



https://developers.google.com/speed/pagespeed/



https://test.speed-kit.com

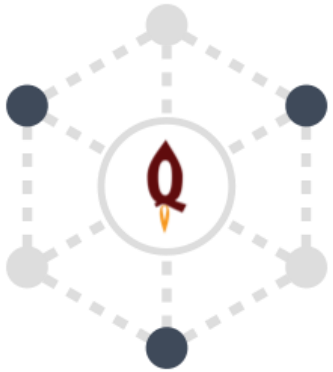
BaQend

https://www.baqend.com/

WEBPAGETEST

http://www.webpagetest.org/

Contact us.



Our Product

Speed Kit:

- Accelerates *Any* Website
- Pluggable
- Easy Setup

test.speed-kit.com

Erik Witt

ew@baqend.com

Web Performance Engineer

Our Services

- Web & Data Management Consulting
- Performance Auditing
- Implementation Services

consulting@baqend.com

BaQend