The Technology Behind Progressive Web Apps

Service Workers in Detail

Erik Witt
ew@baqend.com
24.01.2018
@baqendcom

BaQend

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

What we are going to cover.

PWA

Core Features
Building Blocks
Implementation

Service Worker

Lifecycle
Network Interception
Caching Strategy
etc.

Speed Kit

Cache Coherence
Performance-Measures

Why do(n‘t) we love native apps?

Progressive Web Apps

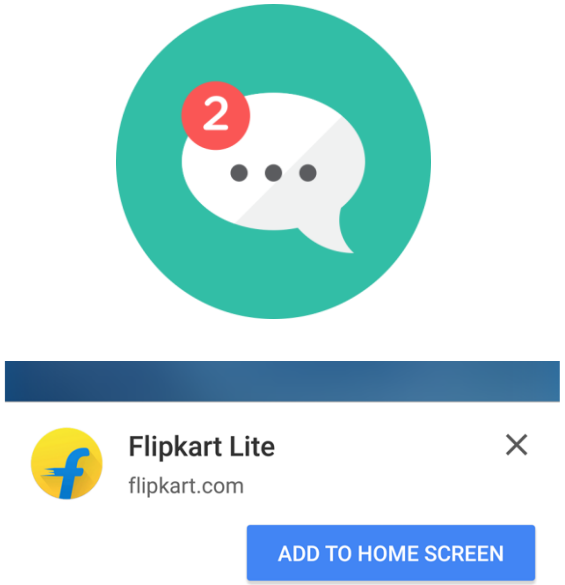
seek to combine the great from **native** and **web apps**

What are Progessive Web Apps?

Progressive Web Apps (PWAs)

+

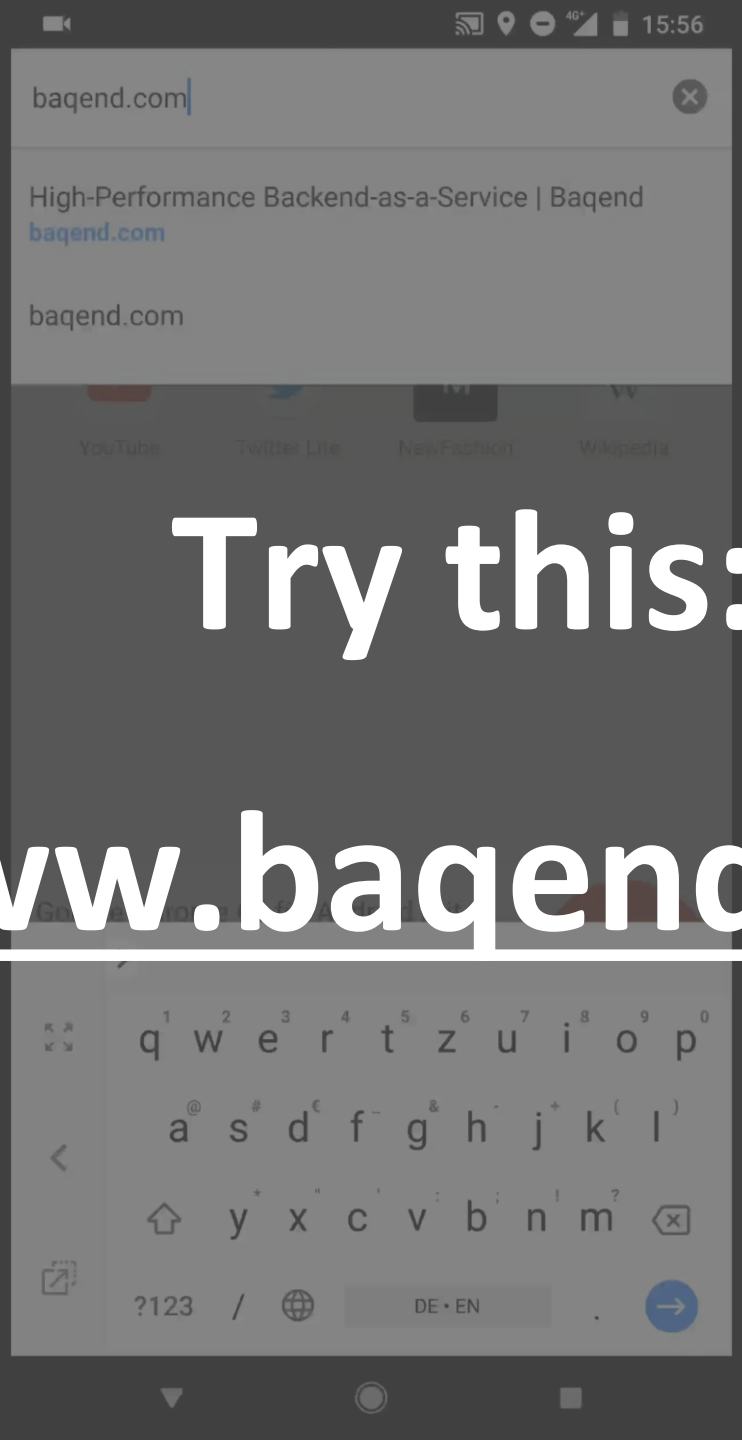
+



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

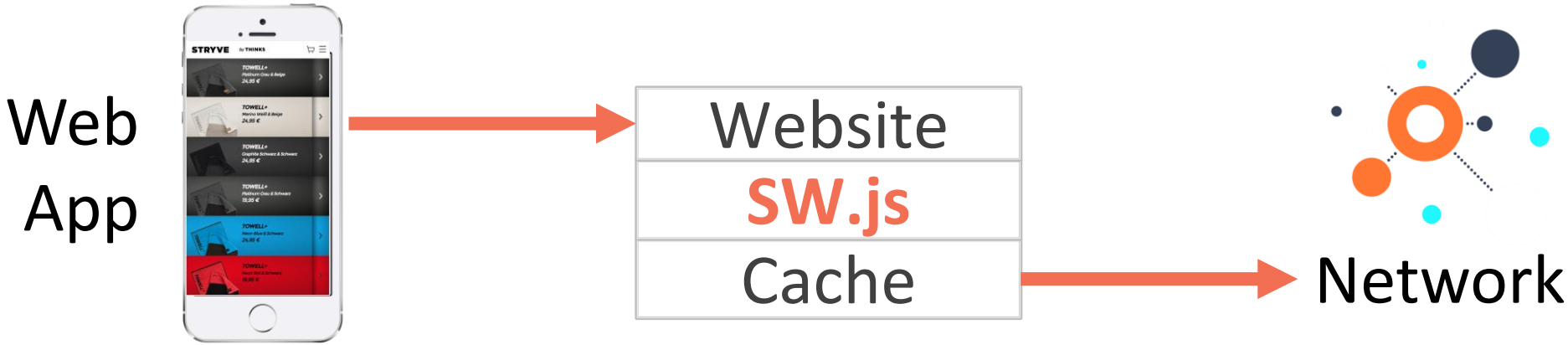
1. **Manifest** declares Add-to-Homescreen:

```
<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

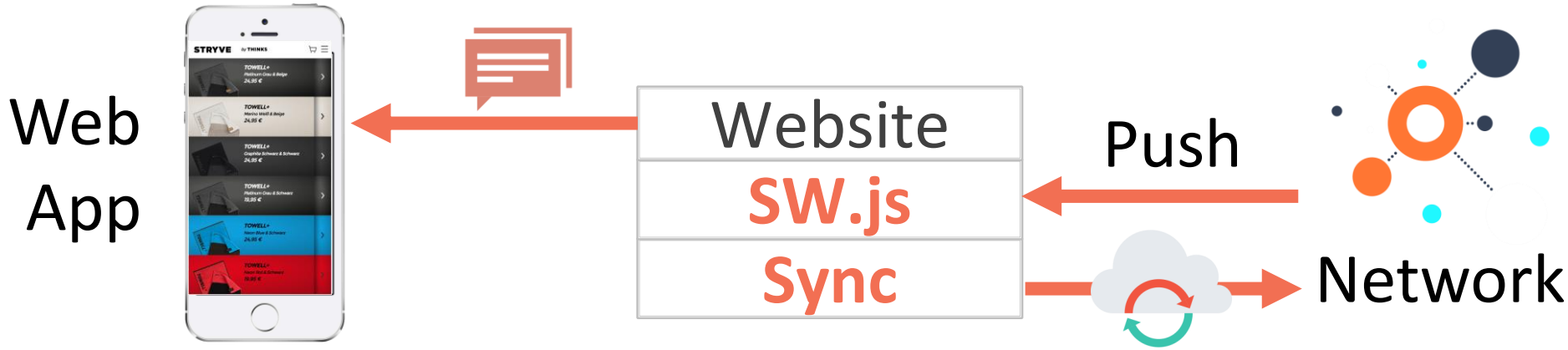
2. Service Workers for caching & offline mode:



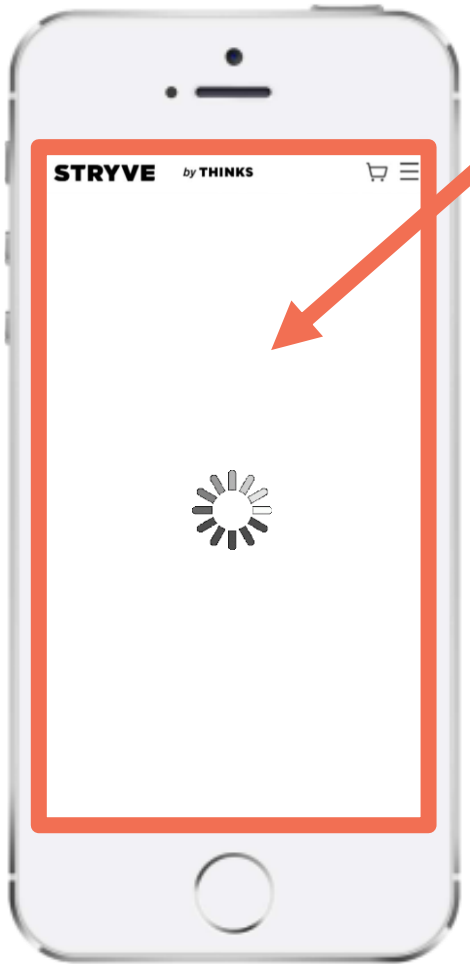
Implementing PWAs

- PWAs are **best practices** and **open web standards**
- **Progressively enhance** the user experience

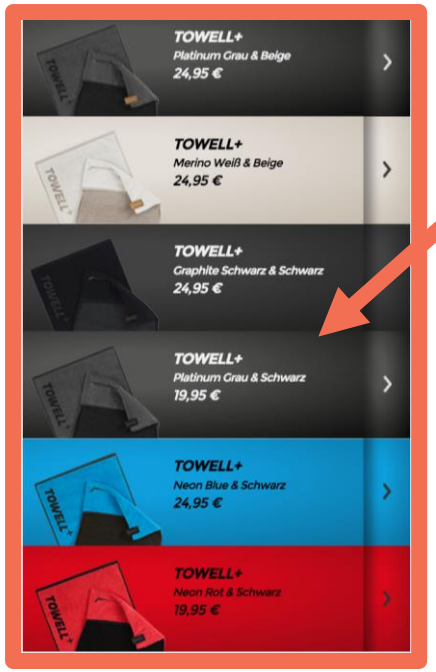
3. Add Web Push and Background Sync:



Typical Architecture: App Shell Model



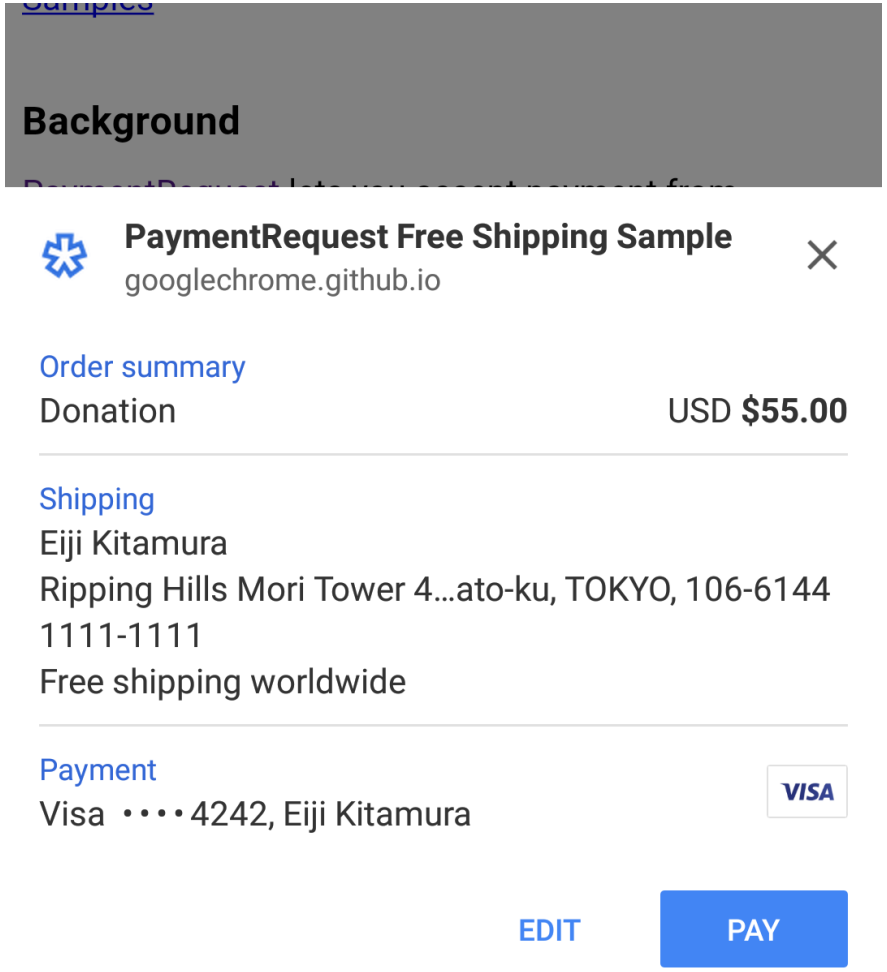
App Shell: **HTML**, **JS**, **CSS, images** with app logic & layout



Content: Fetched on demand & may change more often

What is the future of Progessive Web Apps?

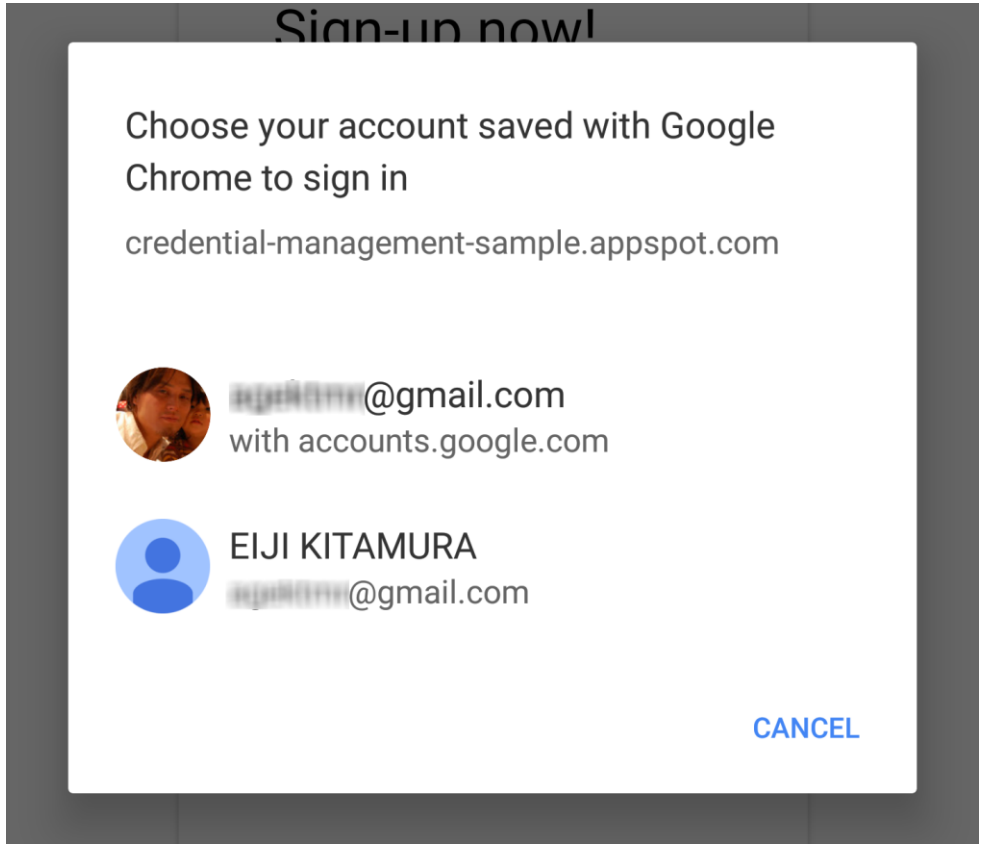
The Future of PWAs is bright.



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.

Web Speech API

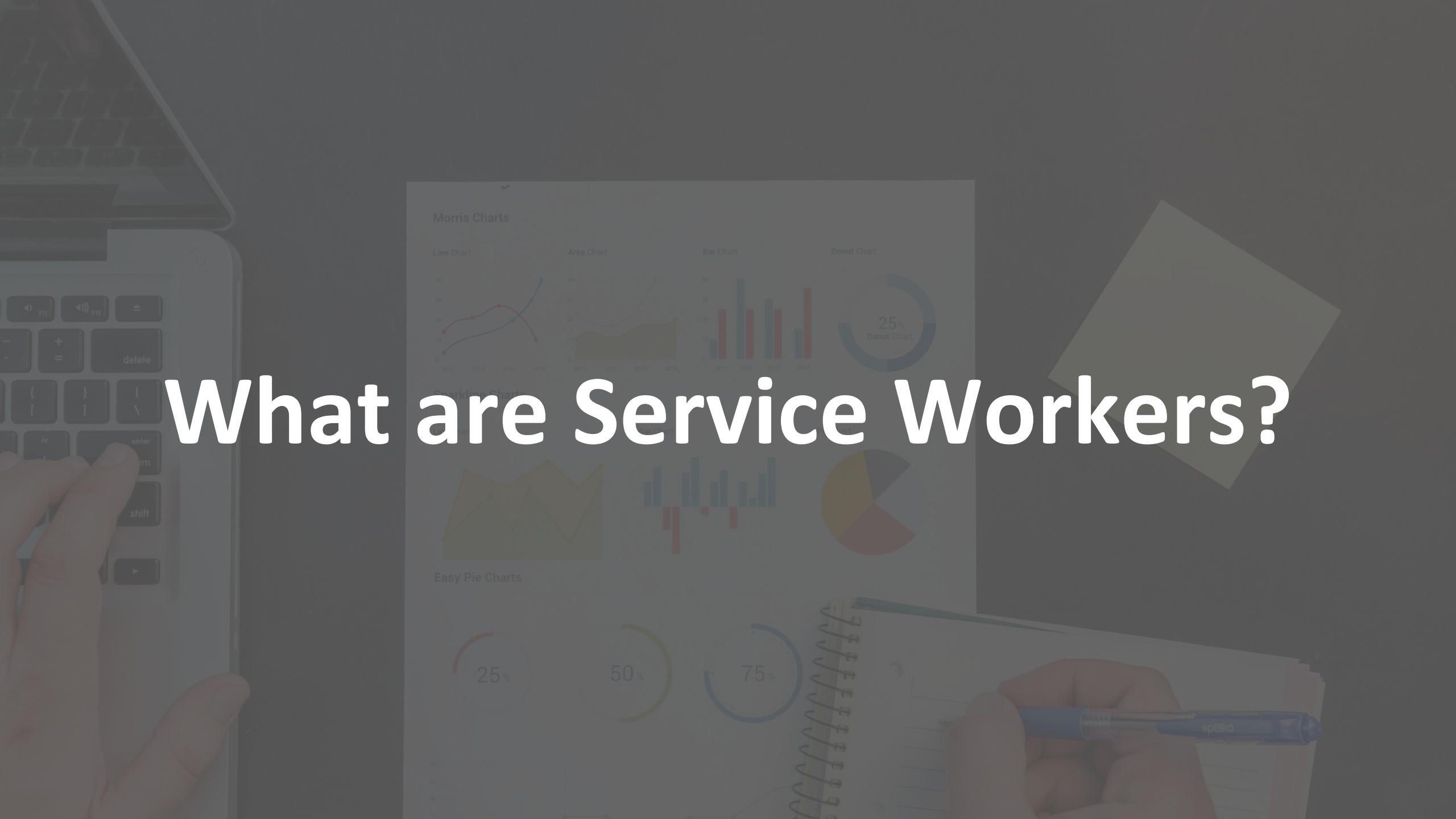
Native Speech Recognition in the Browser:

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

The Future of PWAs is bright.

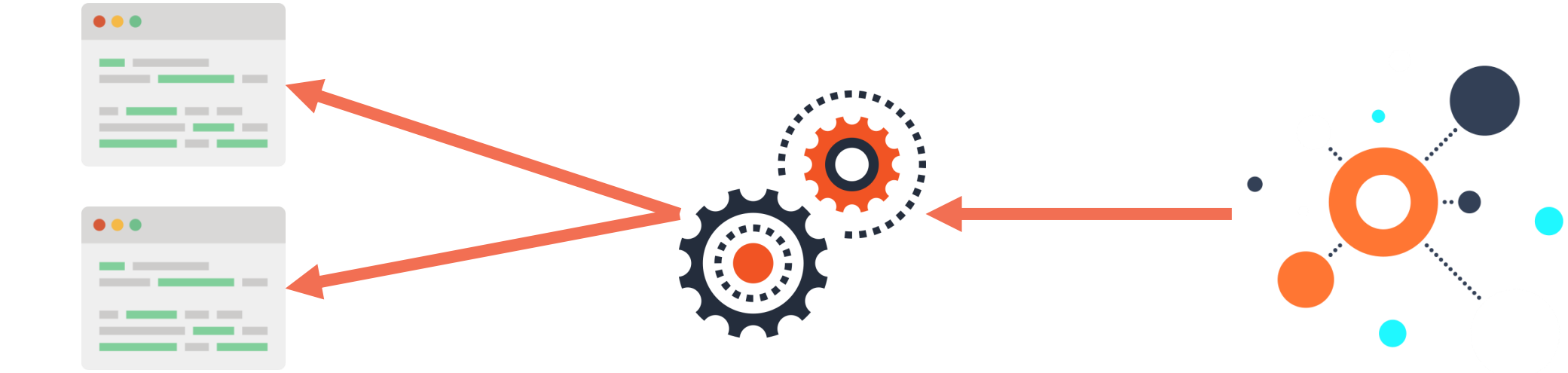
Web Share API

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



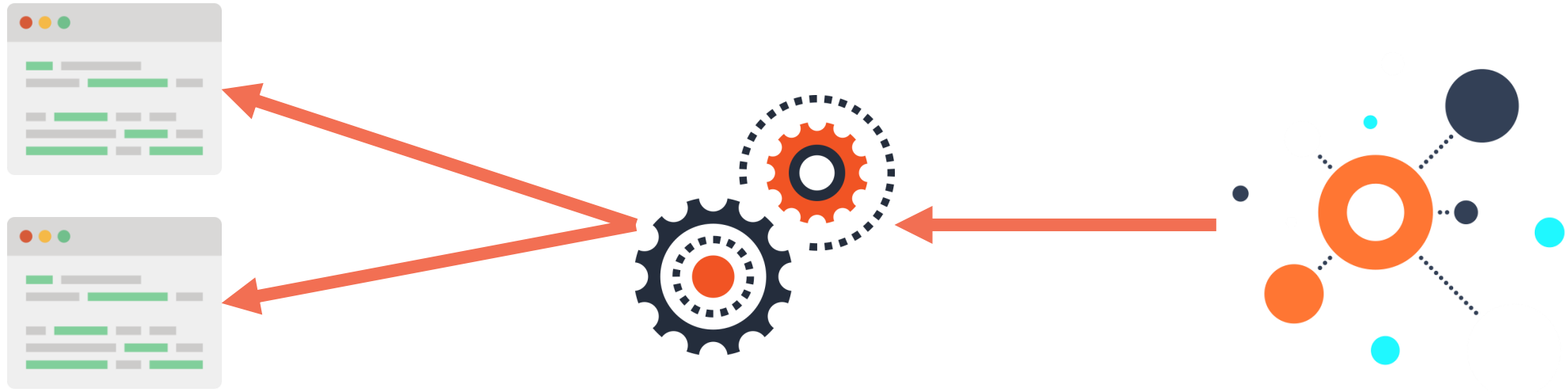
What are Service Workers?

What are Service Workers?



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

What do Service Workers do?



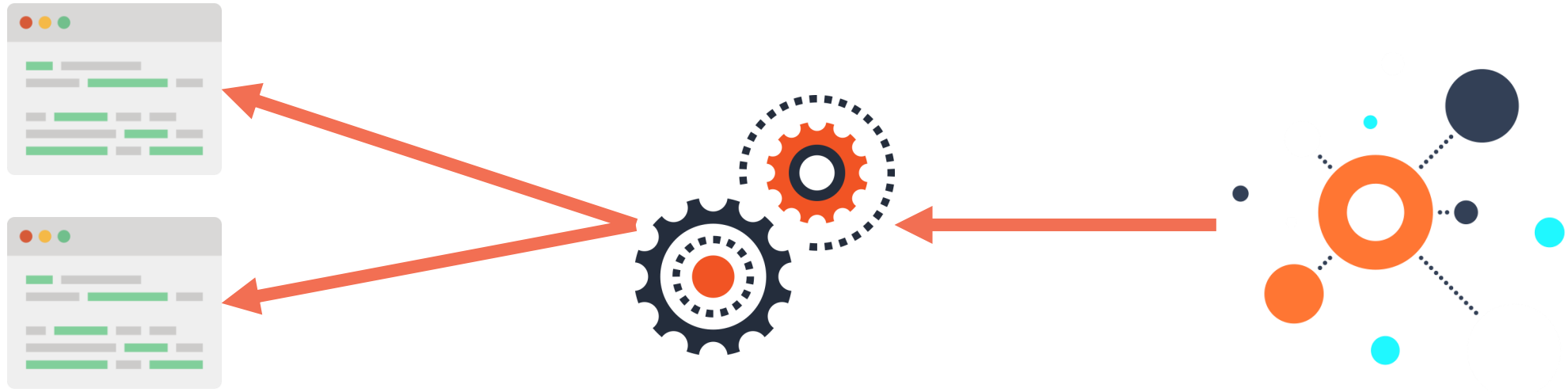
Browser Tabs

Service Worker

Network

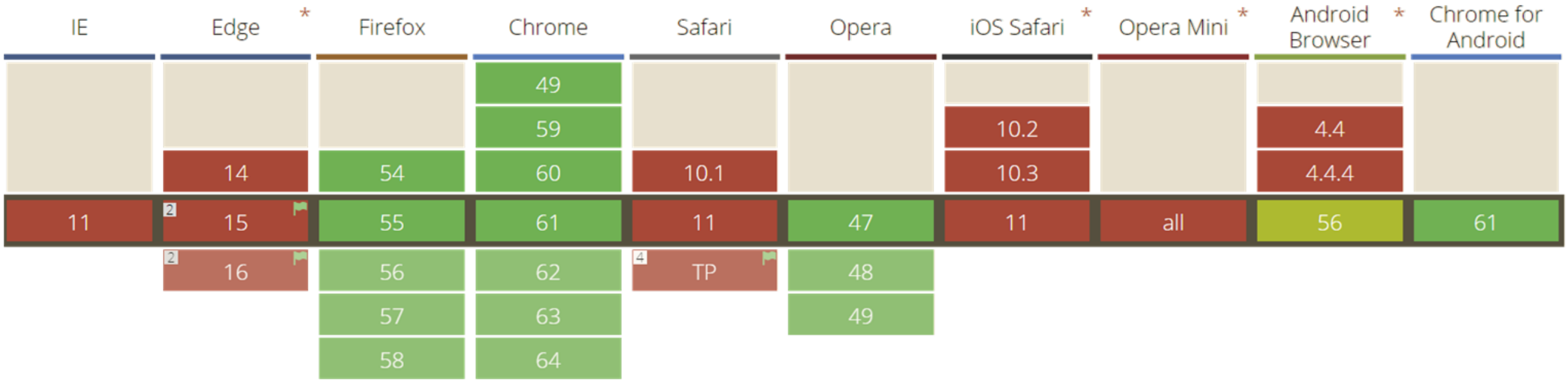
- Cache Data (CacheStorage)
- Store Data (IndexedDB)
- Receive Push
- Respond when Offline

What do Service Workers do?



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

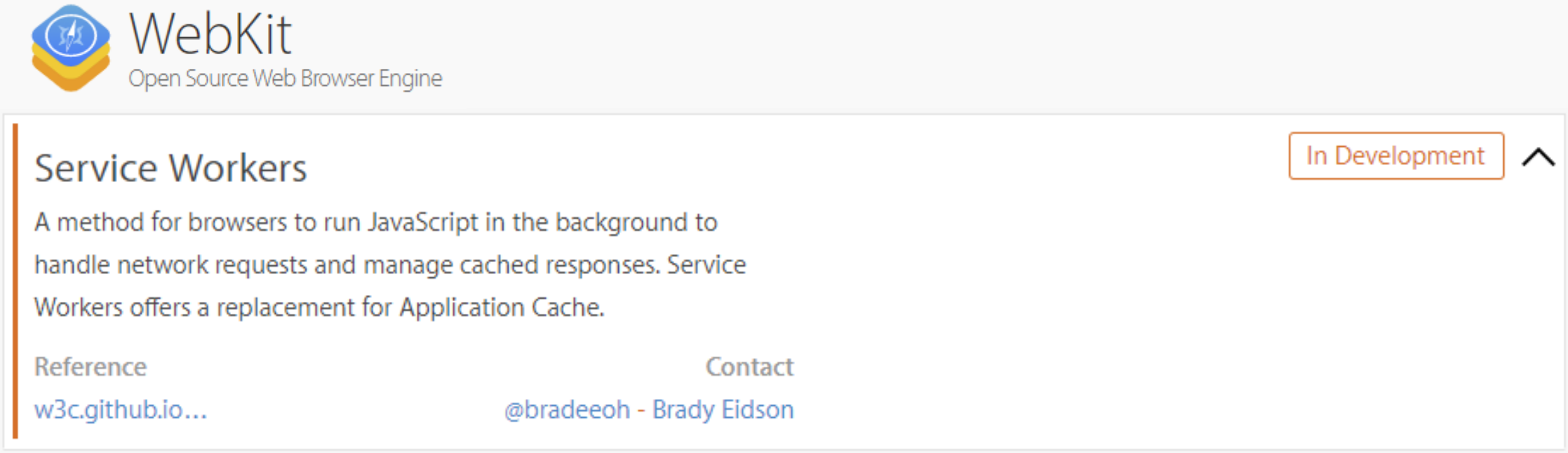
Browser Support for Service Workers



Supported by **75%** of browsers.

Requires **TLS Encryption**.

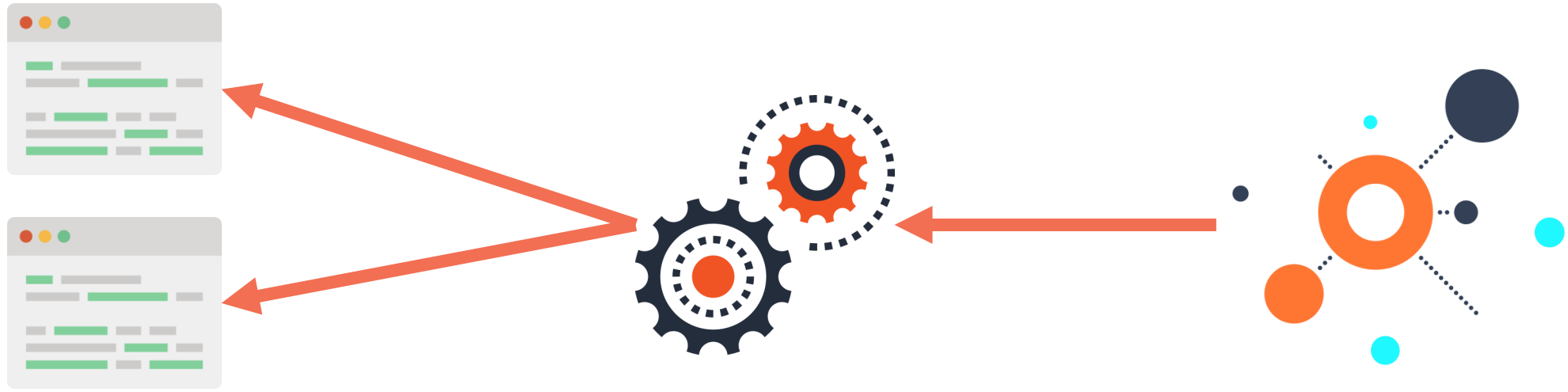
Browser Support for Service Workers



Safari: In Development

Edge: Implemented, but Toggled

How are Service Workers registered?



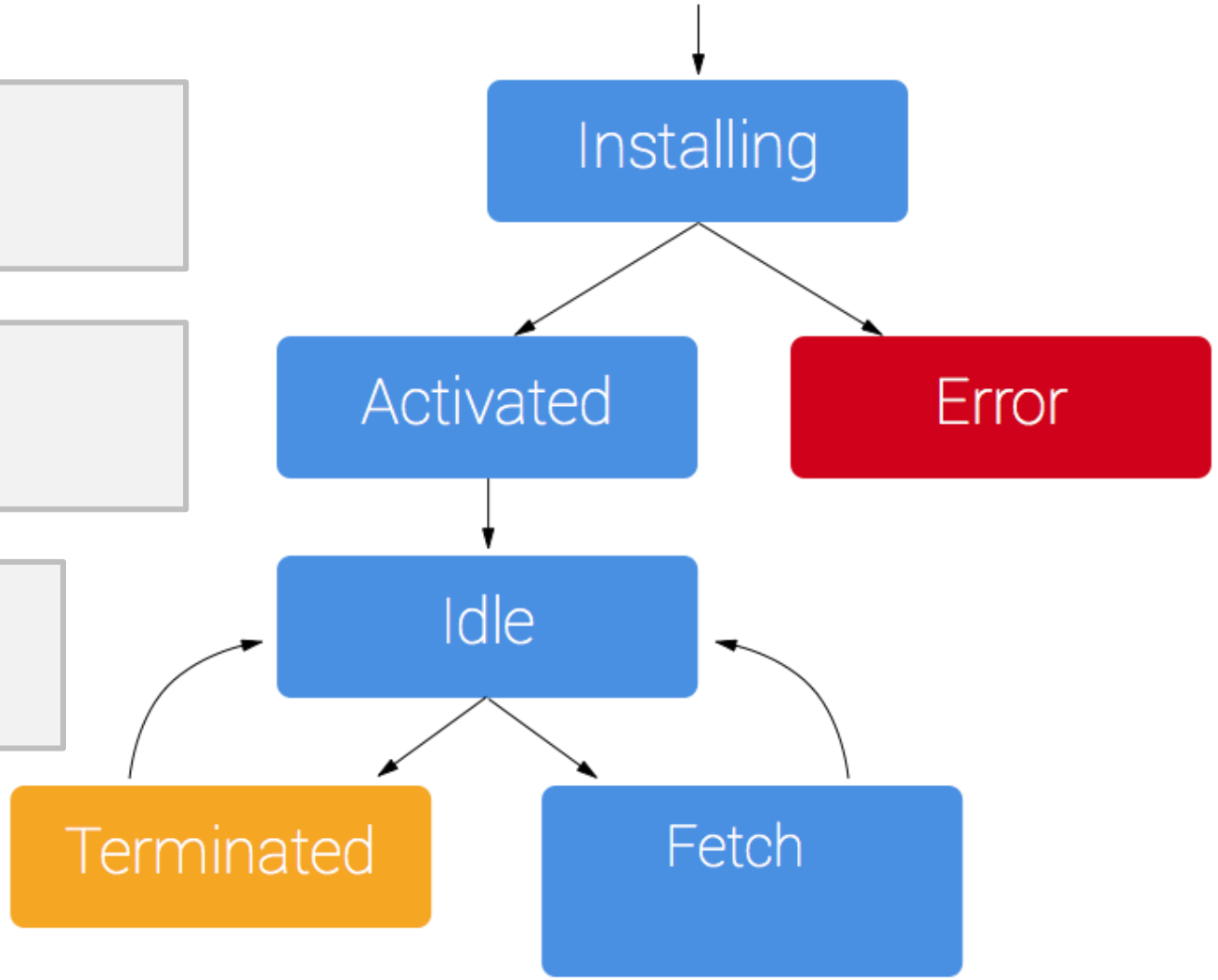

```
<script>
    navigator.serviceWorker.register('/sw.js');
</script>
```

How does the Lifecycle look like?

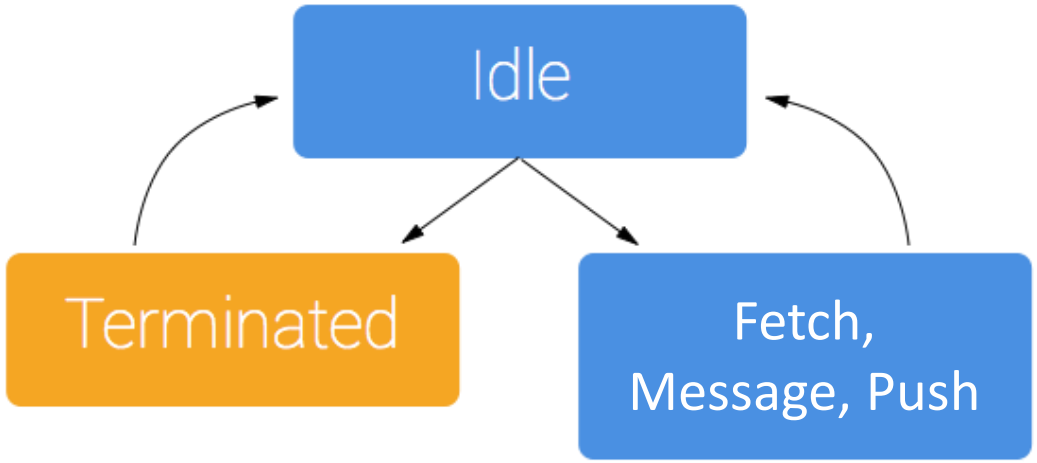
```
self.addEventListener('install', (event) => {
  // Perform install steps
});
```

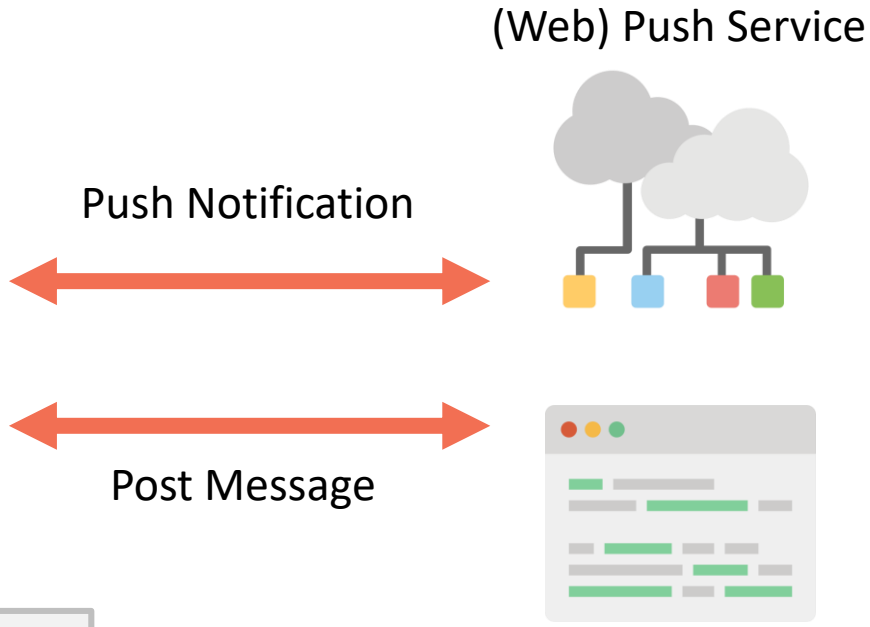
```
self.addEventListener('activate', (event) => {
  // Perform activate steps
});
```

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



How to Communicate with Service Workers?



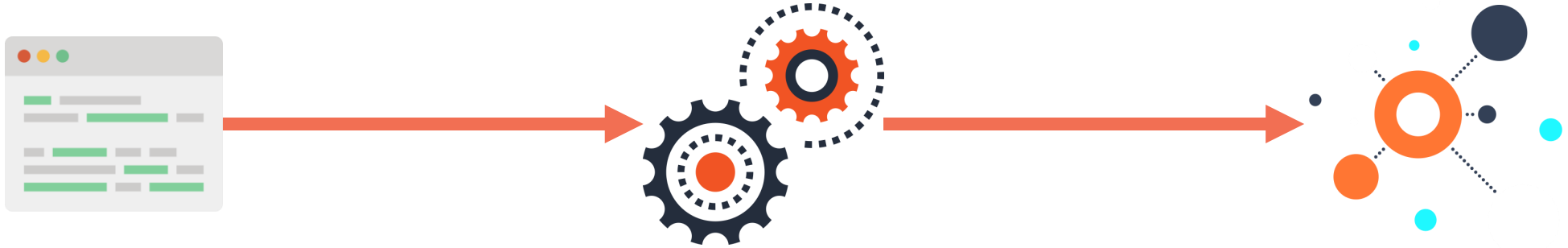



```
self.addEventListener('message', (event) => {
  // Receive message push
});
```

```
// 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 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 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/' });
```

```
// Widening the scope is NOT ALLOWED
// unless explicitly allowed by HTTP header 'Service-Worker-Allowed: /'
// Gets all requests from the domain
navigator.serviceWorker.register('/path/sw.js', { scope: '/' });
```

Service Worker Persistence

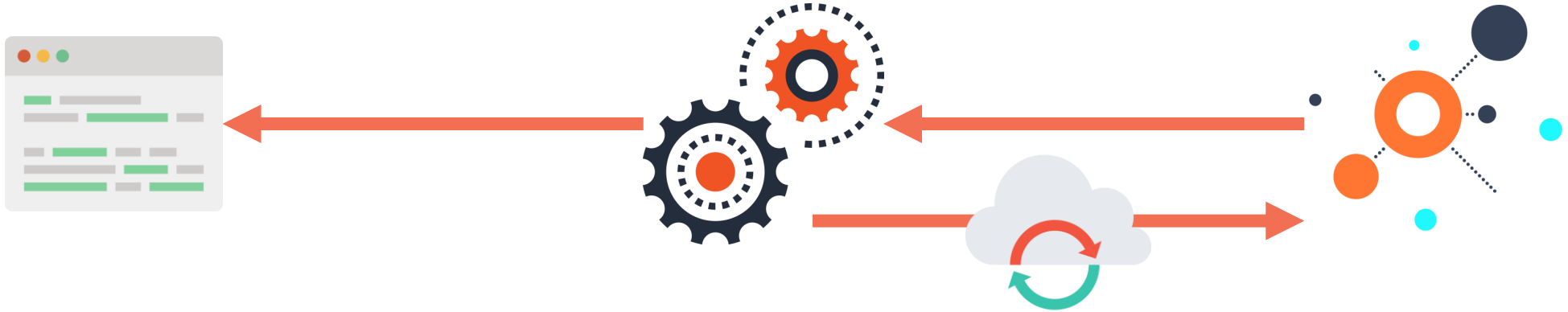
IndexedDB

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

```
// Register a sync at the Service Worker
registration.sync.register('keyword')
```

```
// React to sync event in Service Worker
self.addEventListener('sync', (event) => {
  if (event.tag == 'keyword') }
    event.waitUntil(...);
  }
});
```

One-off Sync is

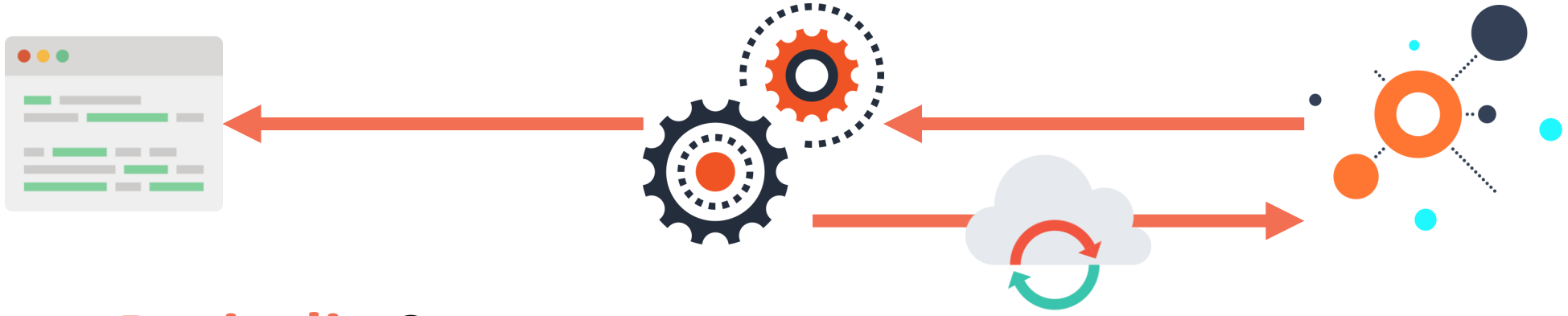
- executed when user is online
- retried when failed (exponential backoff)

Example Use Cases

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

Experimental

Service Worker Background Sync



Periodic Sync

```
// Registers a periodic sync
registration.periodicSync.register(options)
```

```
// Execute periodic sync in Service Worker
self.addEventListener('periodicsync', (event) => {
if (event.registration.tag == 'keyword') {
    event.waitUntil(...);
  }
});
```

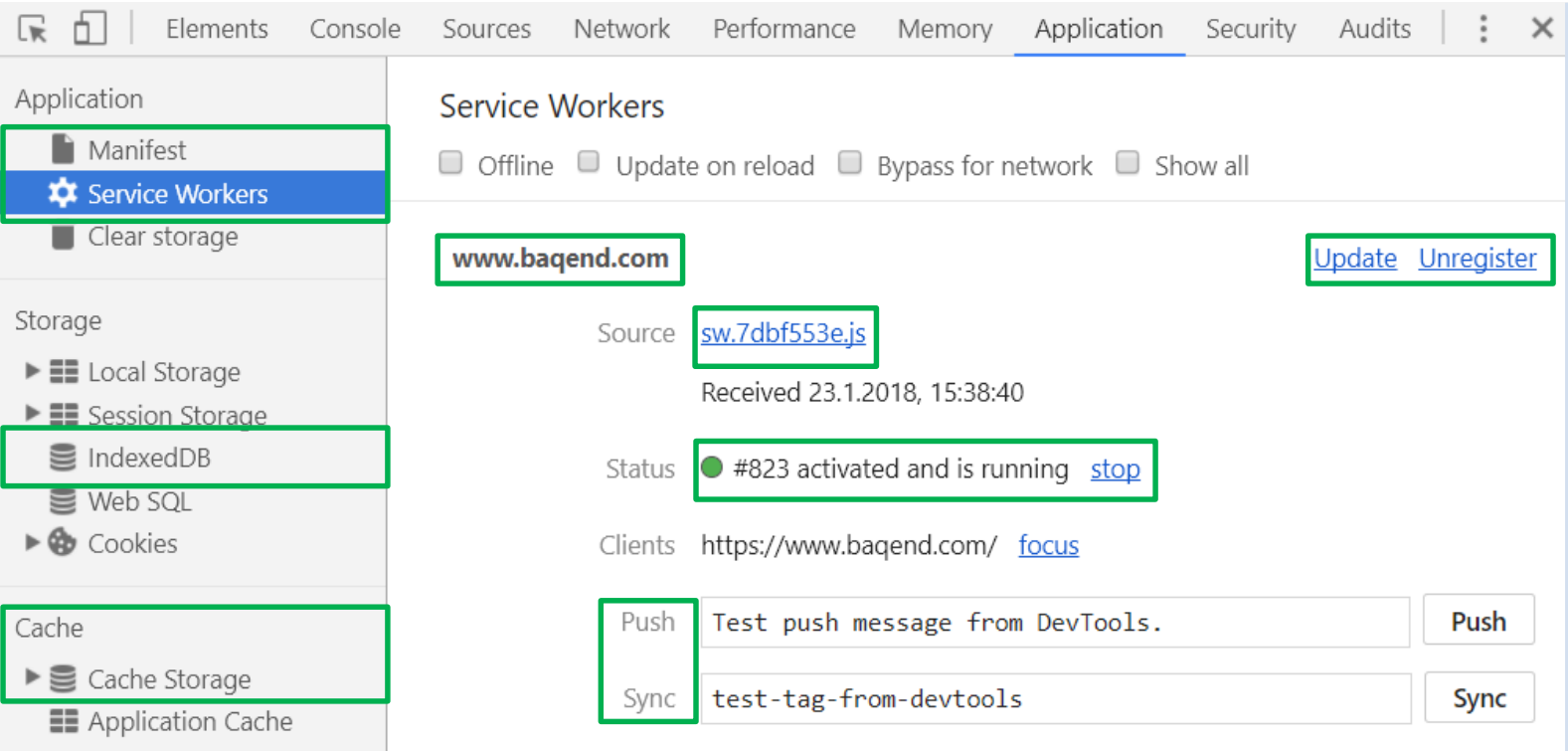
Periodic Sync is

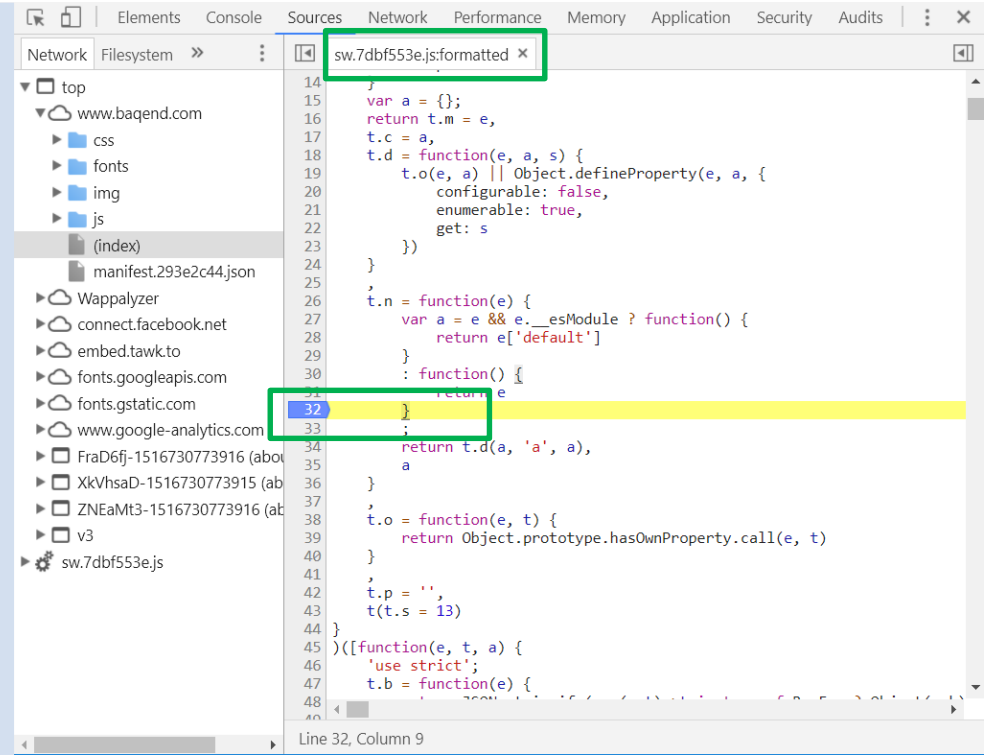
- executed when online, according to period options
- retried when failed

Example Use Case

- Load updates to social media timeline when browser closed

Service Worker Debugging





Service Worker Caching

Cache Storage

Stores Request/Response pairs

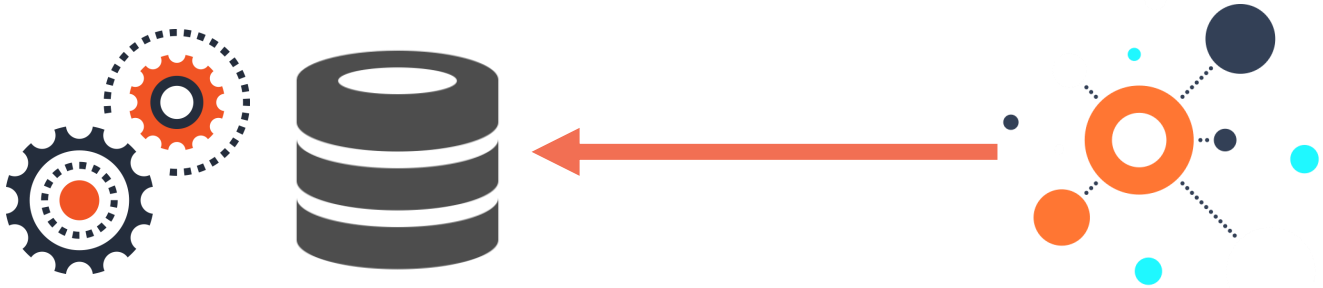
```
// Putting Request/Response pair in cache
const cache = await caches.open('name');
cache.put(request, response);
```

```
// Retrieving Response from cache
const response = await caches.match(request);
return response || fetch(request);
```

Cache Storage

- Programmatically managed
- Persistent and non-expiring
- Supports only HTTP
- Only caches GET requests (no HEAD)

How Resources get into the Cache



```
// Cache resource when fetched
self.addEventListener('fetch', (event) => {
  event.respondWith((async () => {
    const cache = await caches.open('name');
    const response = await fetch(request);
    await cache.put(request, response.clone());
    return response;
  })());
});
```

```
// Cache resources as install dependency
const urlsToCache = ['index.html', 'style.css',
'app.js' ];

self.addEventListener('install', (event) => {
  event.waitUntil((async () => {
    const cache = await caches.open('name');
    await cache.addAll(urlsToCache);
  })());
});
```

Caching Strategies – Cache Only

```
// Serve cached Response
self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.match(event.request)
  );
});
```

Cache only strategy gets all requests from cache or fails.

- Fast responses (or none)
- Only for pre-cached requests
- Only for static resources
- Needs asset hashing or versioning of cache

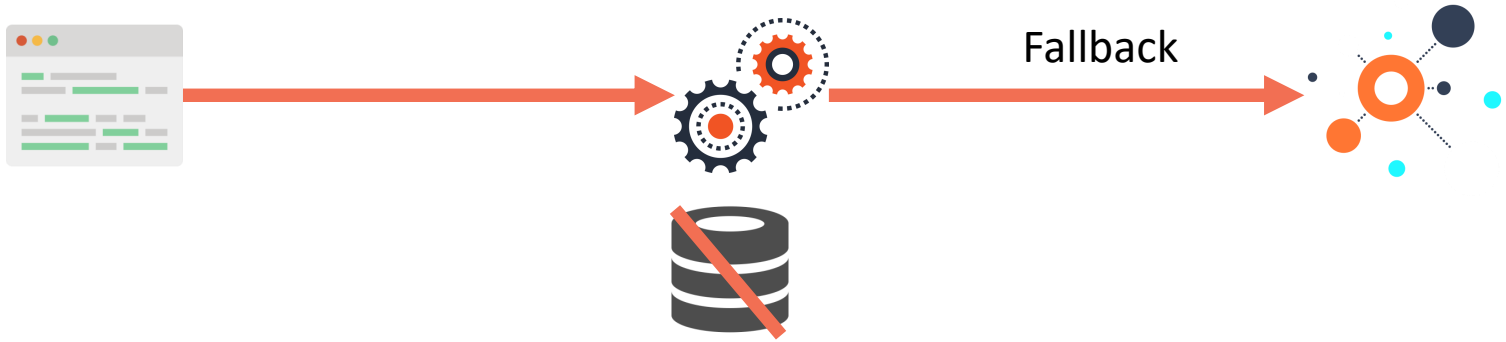
Caching Strategies – Cache, Network Fallback


```
// Serve cached Response or fallback to network
self.addEventListener('fetch', (event) => {
  event.respondWith(async () => {
    const response = await caches.match(request);
    return response || fetch(request);
  });
});
```

This strategy gets request from cache and from network as fallback.

- Fast responses for cached resources, slow for others
- Only for static resources cacheable
- Needs asset hashing or versioning of cache

Caching Strategies – Network Only

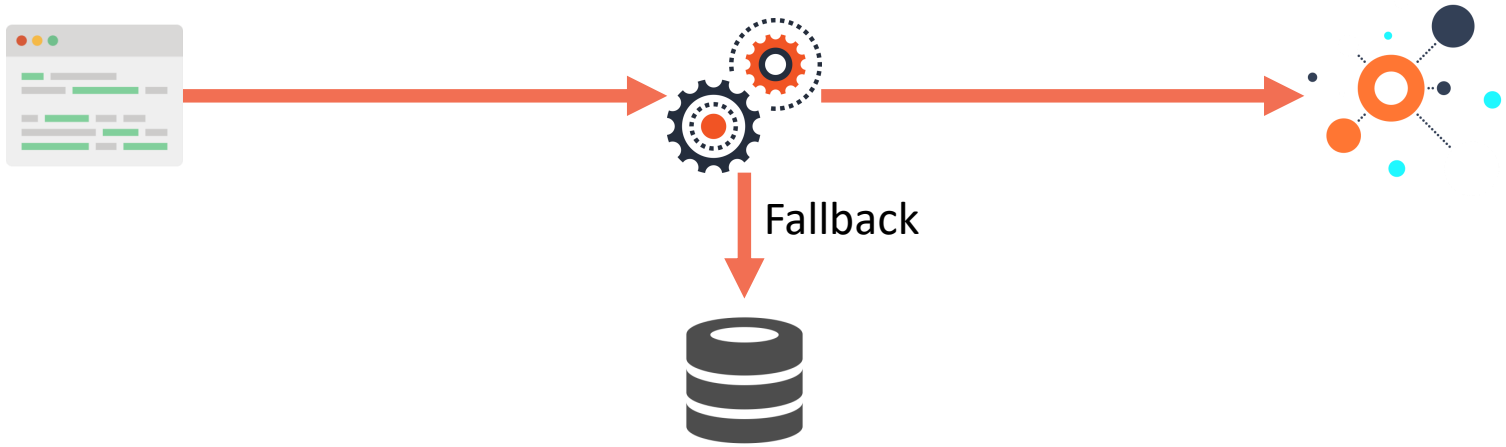



```
// Serve everything from network
self.addEventListener('fetch', (event) => {
  event.respondWith(async () => {
    return fetch(request)
  });
});
```

This strategy gets request from network only.

- Slow responses
- Always up-to-date

Caching Strategies – Network, Cache Fallback

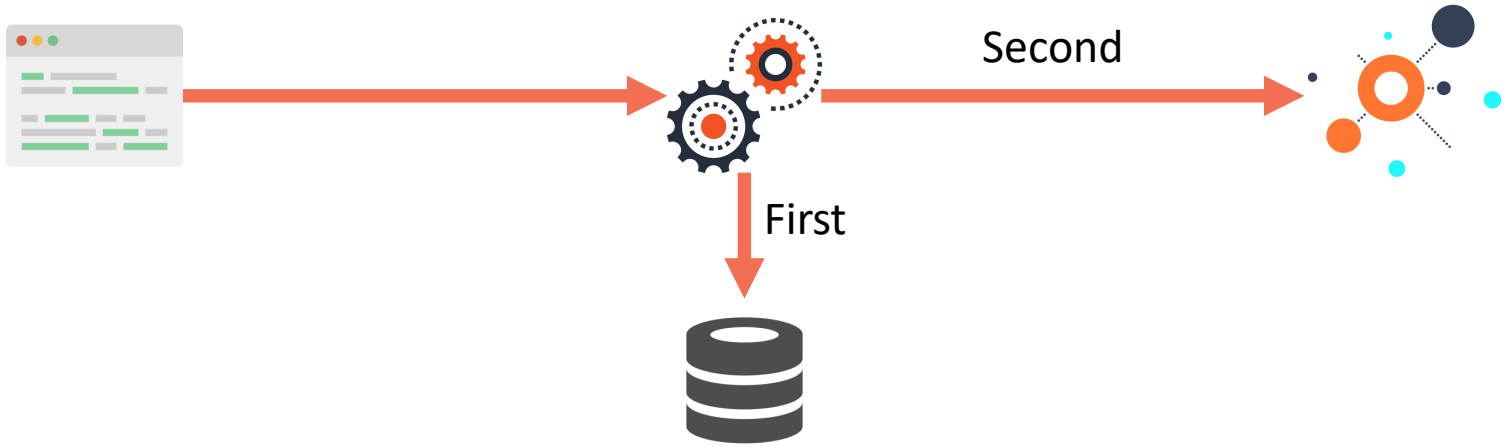



```
// Serve network Response or fallback to cache
self.addEventListener('fetch', (event) => {
  event.respondWith(
    fetch(event.request).catch(function() {
      return caches.match(event.request);
  });
});
```

This strategy gets request from network and from cache as fallback.

- Slow responses from network
- Effectively offline mode (better use `navigator.onLine`)

Caching Strategies – Cache, then Network

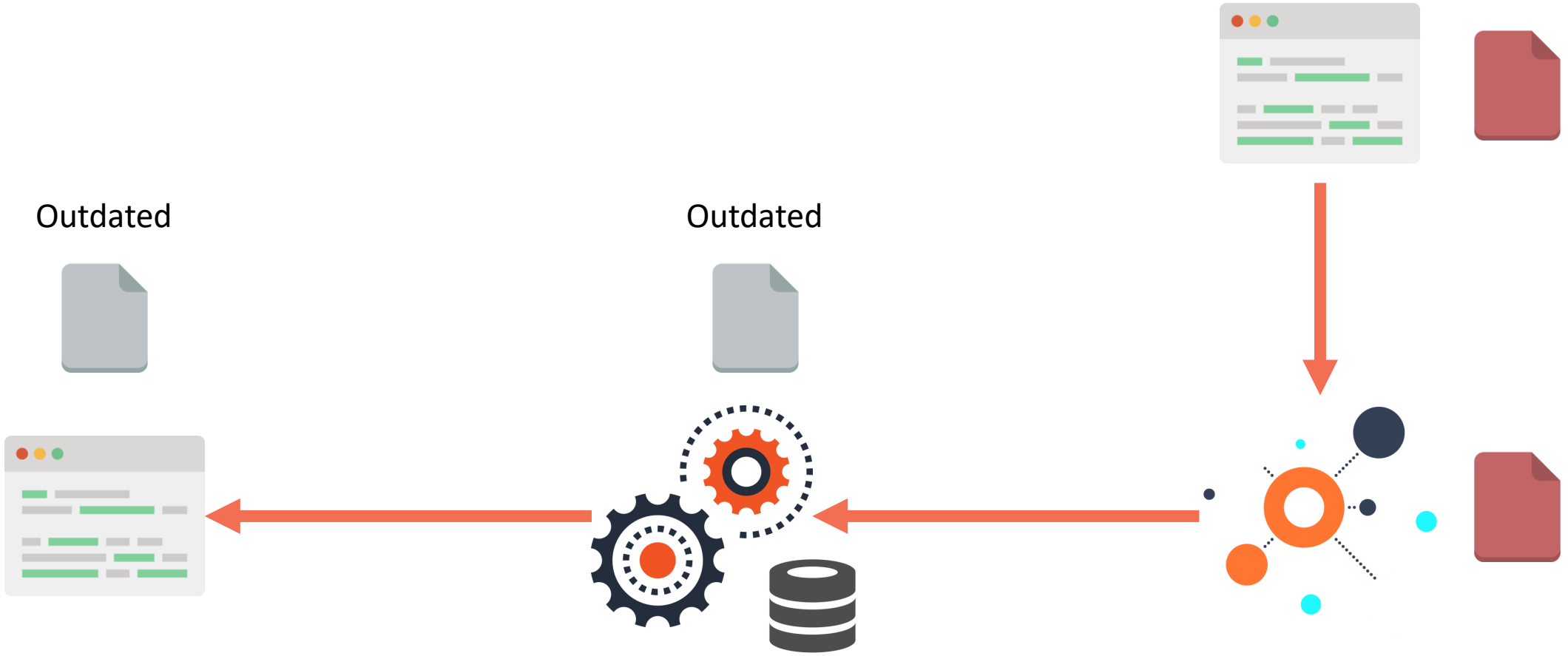



```
// Idea:
// Serve resource from cache
// In the background:
// * fetch resource from network
// * send new response to browser
// * apply changes to DOM or ask user to
reload the page
```

This strategy gets requests from cache first and from network in background.

- Fast initial response
- Uses a lot of bandwidth
- Used for static app shell or message inboxes

Major Challenge: Cache Coherence

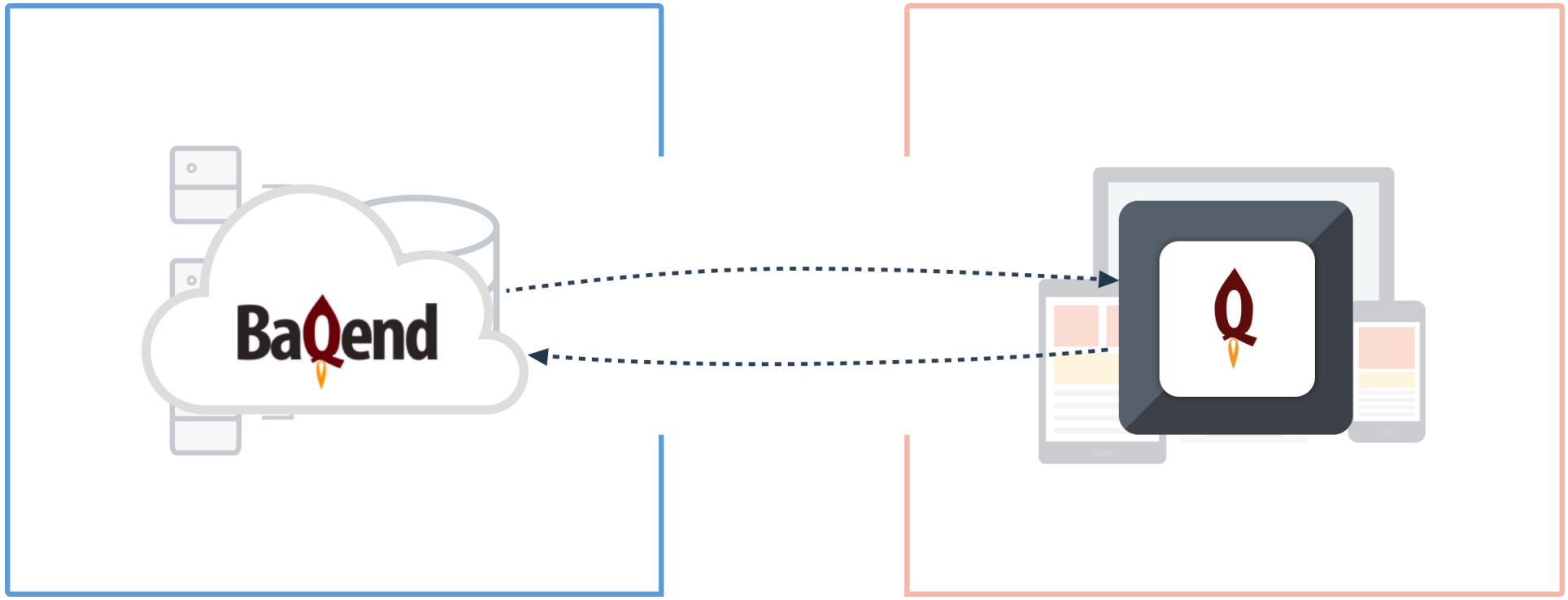


All caching strategies either serve outdated data or degrade performance

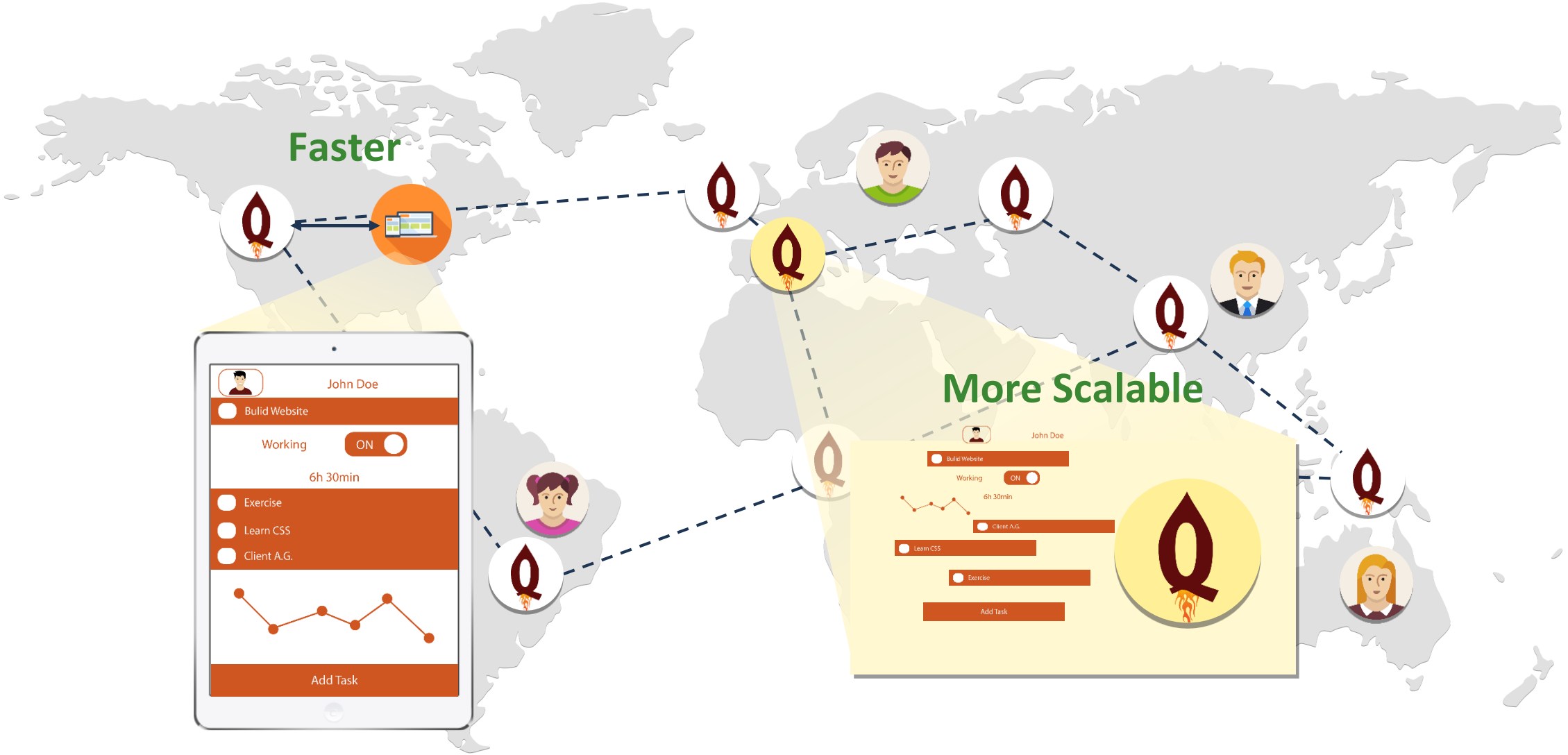
What we do with Service Workers

Speed Kit

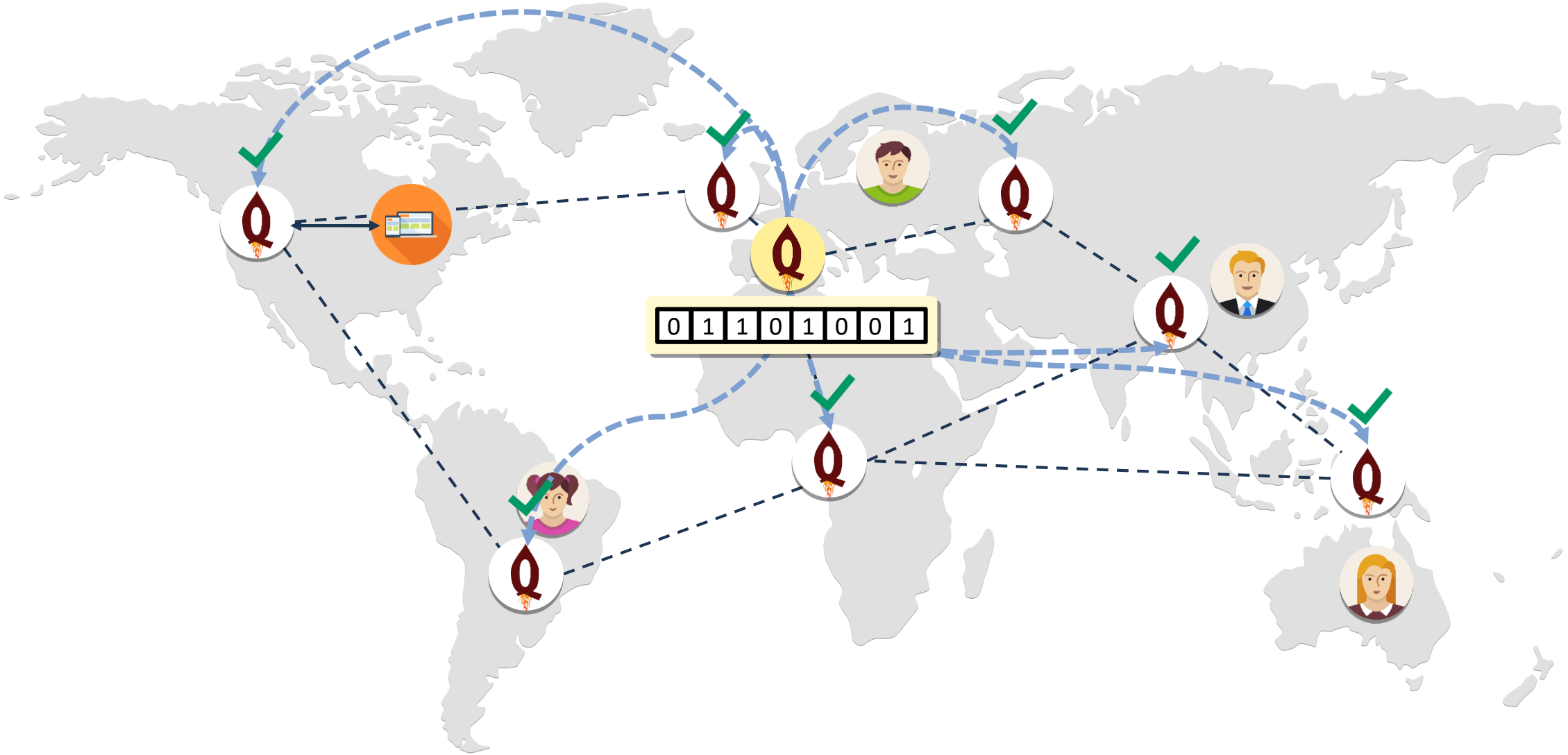
Turning Websites into Instantly-Loading Progressive Web Apps



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

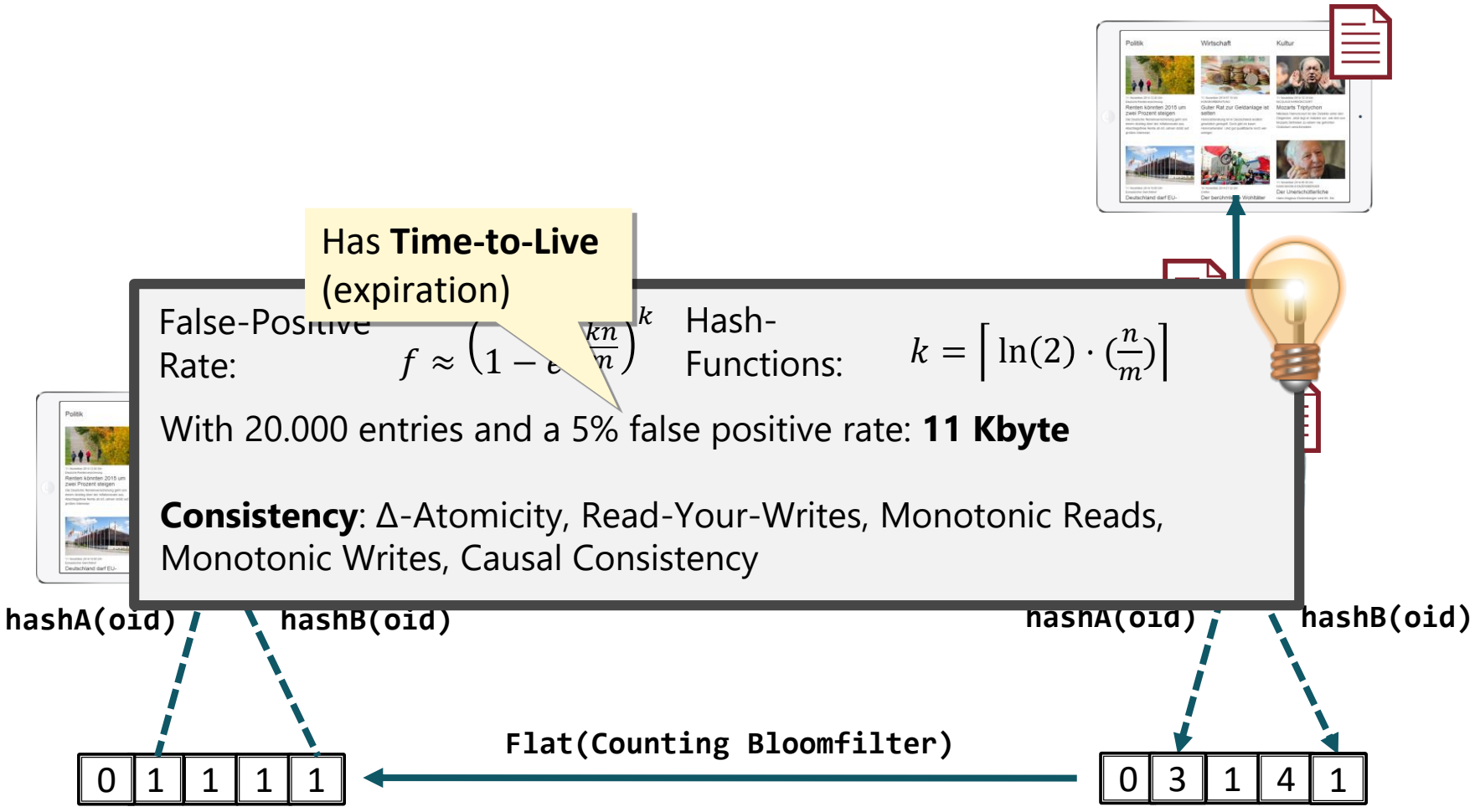
F. Gessert, N. Ritter „Scalable Data Management: NoSQL Data Stores in Research and Practice“, *32nd IEEE International Conference on Data Engineering, ICDE,* 2016

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

Backed by

30 Man-Years of Research

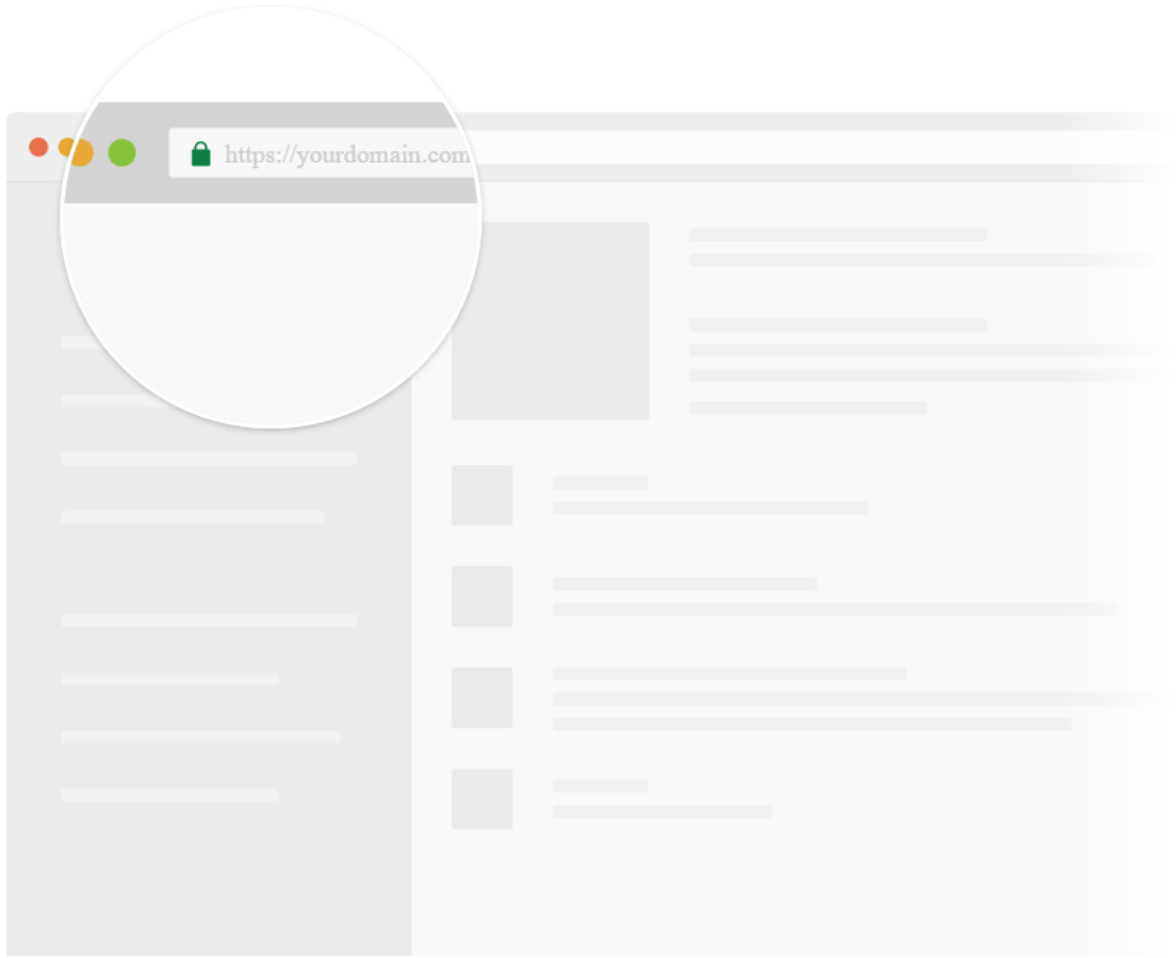
How Speed Kit solves Cache Coherence



Adding Speed Kit to a Site

1. Configure Domain

Set which URLs Baqend should accelerate.

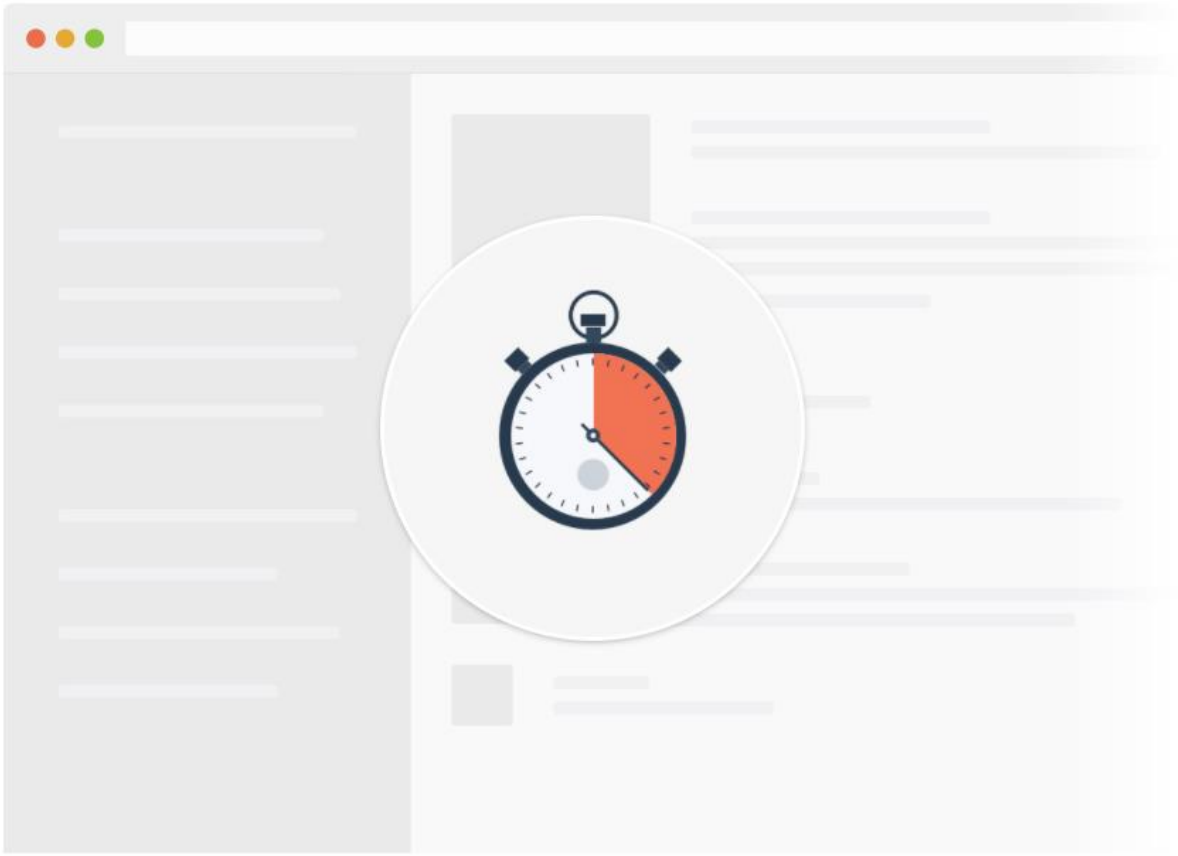


2. Include Code Snippet

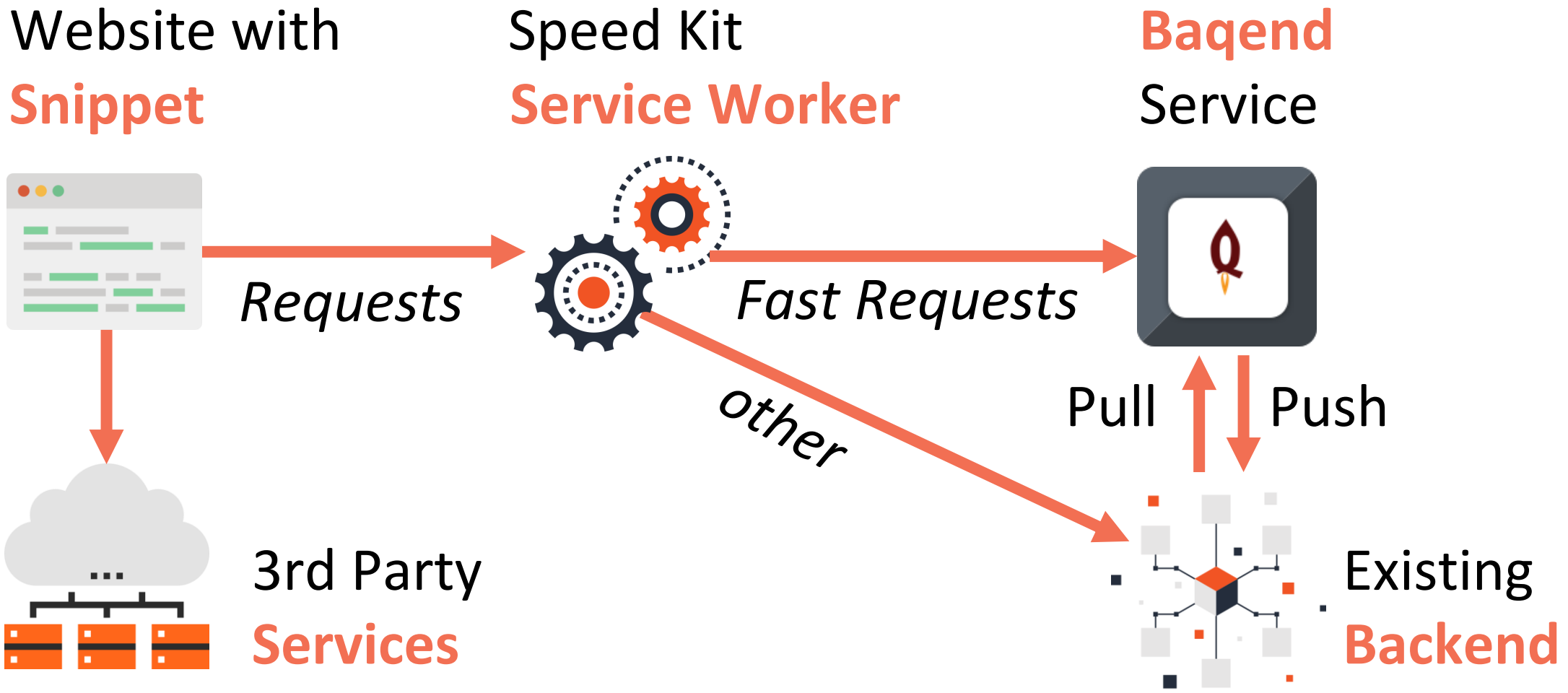
Add the Speed Kit Service Worker to the website.

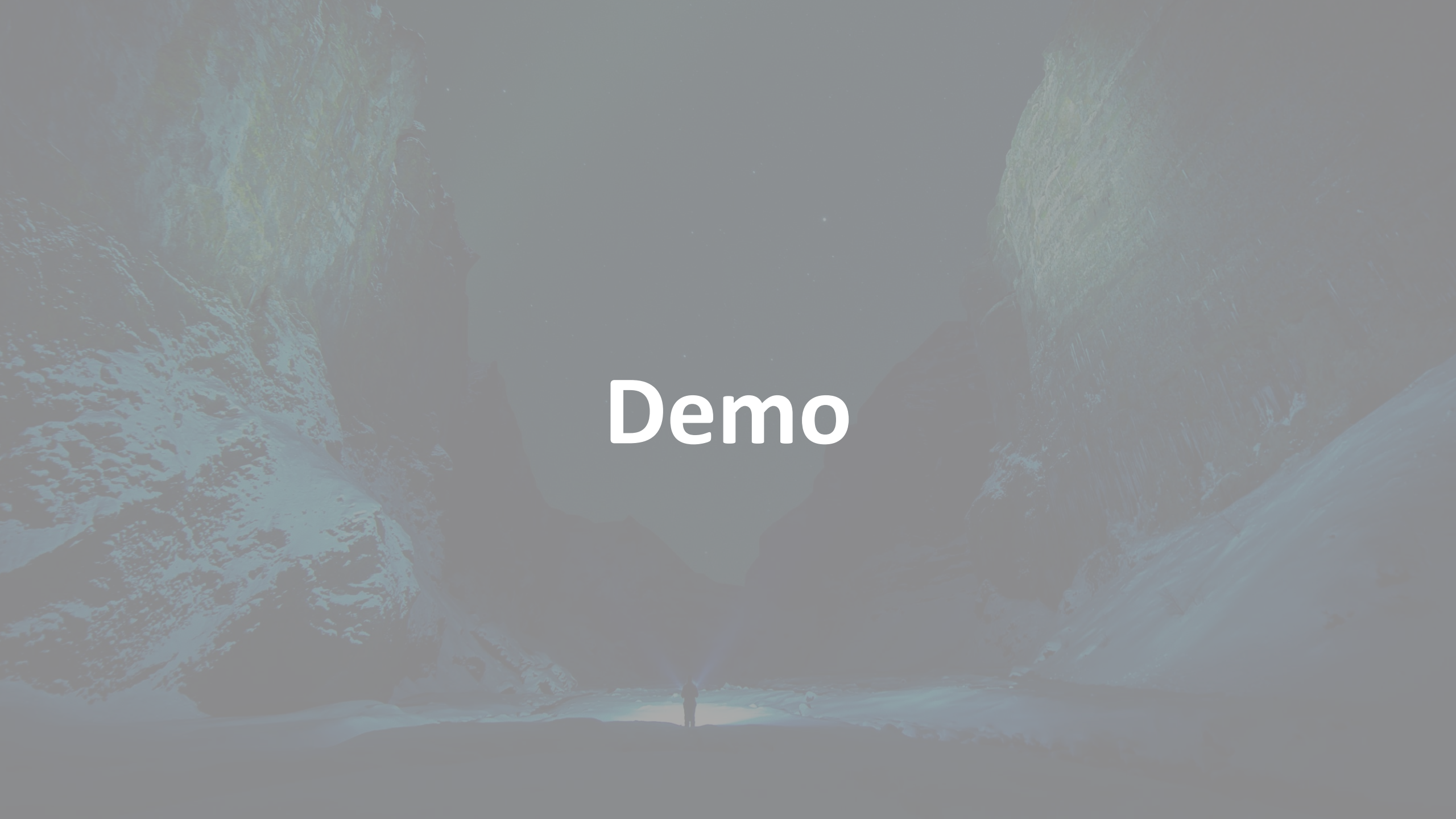
3. Requests Accelerated

Speed Kit routes the requests through Baqend‘s caches.



How it works under the hood




Demo

Now, we have a PWA, HTTP/2, CDNs, etc.

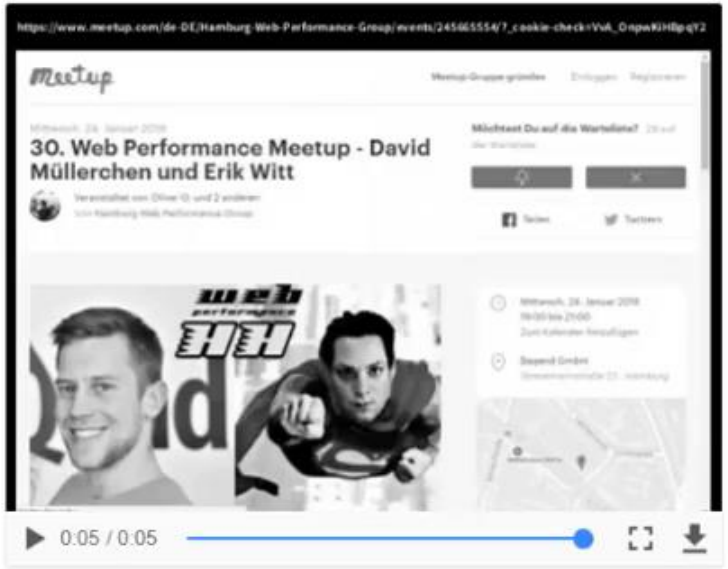
How do we measure web performance?

Page Speed Analyzer

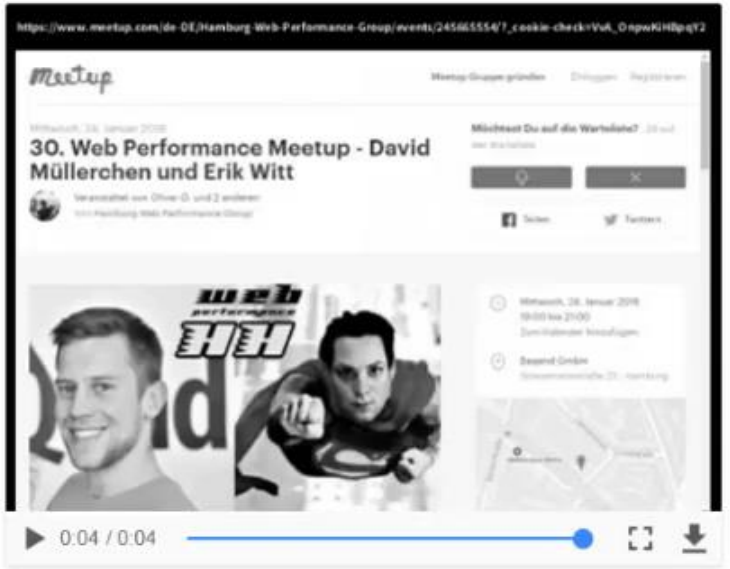
https://www.meetup.com/de-DE/Hamburg-Web-Performance-Group/events/245 Go

Domains	Requests	Response Size
7	30	3.23 MB

Your Website

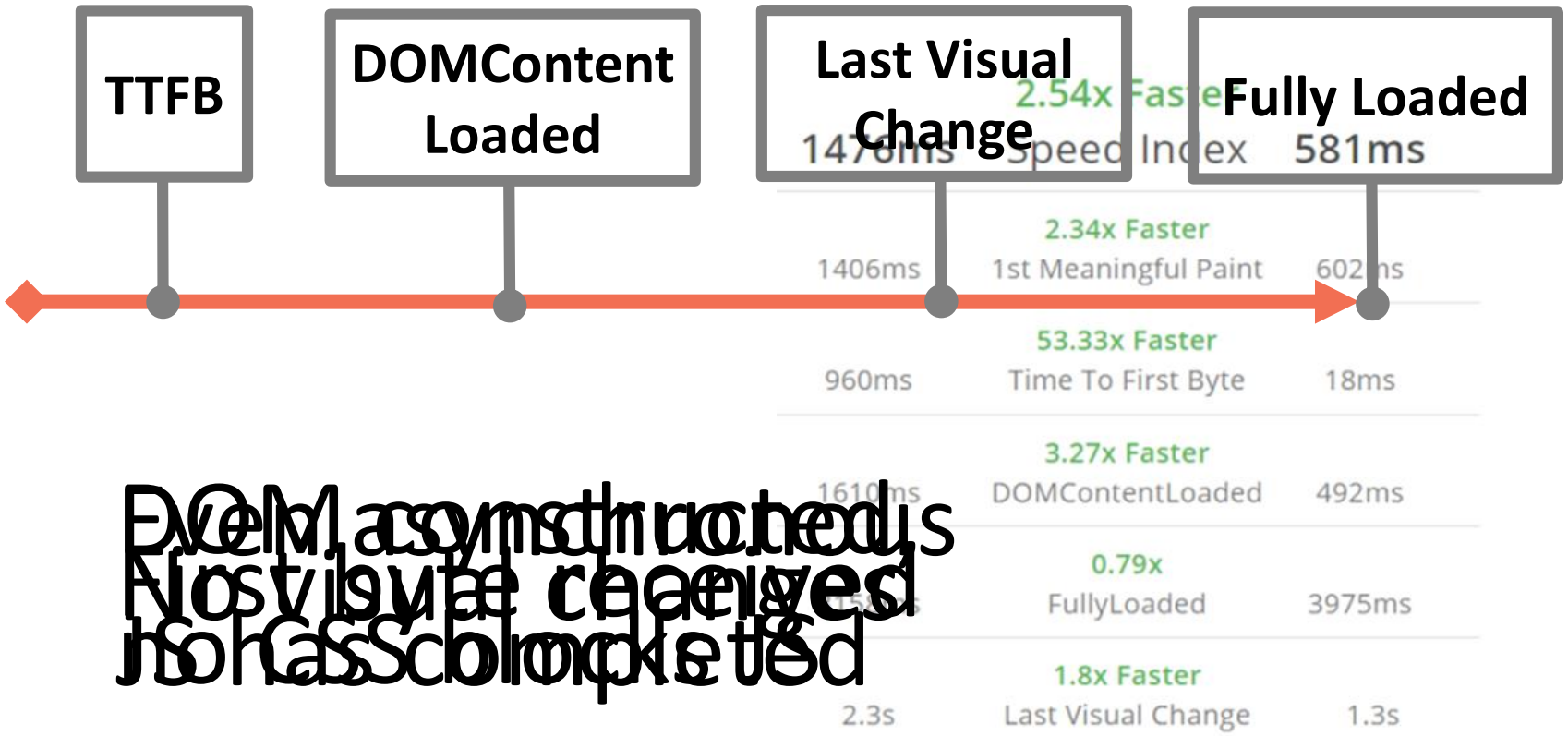


Your Website with Speed Kit



Your Website		Your Website with Speed Kit
	2.54x Faster	
1476ms	Speed Index	581ms
	2.34x Faster	
1406ms	1st Meaningful Paint	602ms
	53.33x Faster	
960ms	Time To First Byte	18ms
	3.27x Faster	
1610ms	DOMContentLoaded	492ms
	0.79x	
3158ms	FullyLoaded	3975ms
	1.8x Faster	
2.3s	Last Visual Change	1.3s

Measuring Web Performance



Measuring Web Performance

How can we measure **user-perceived** performance?

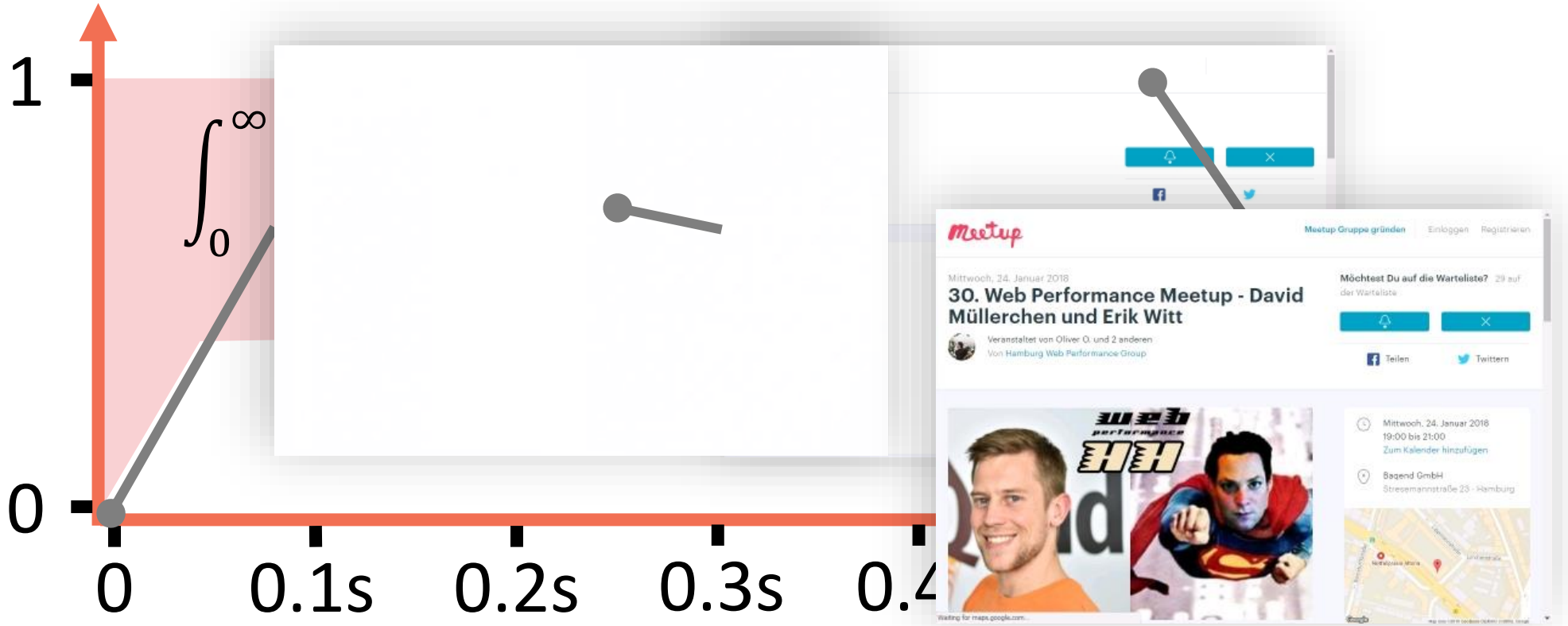
Even asynchronous
JS has completed

46.93x Faster
14737ms Speed Index 314ms
Time To First Byte
8909ms 142ms
45.18x Faster
18344ms DOMContentLoaded 406ms
18.36x Faster
18615ms FullyLoaded 1014ms
38.38x Faster
18.8s Last Visual Change 0.5s

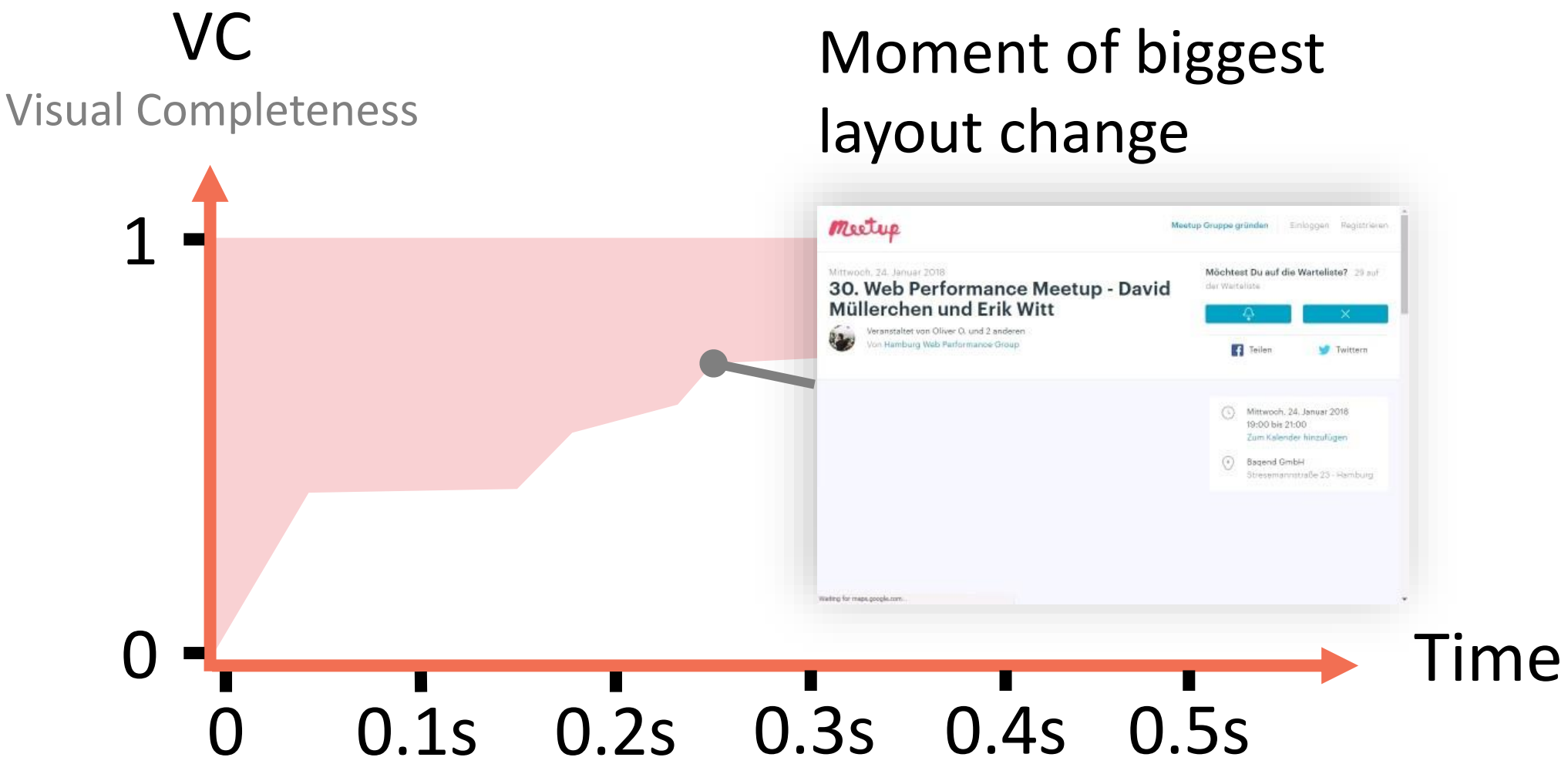
The Speed Index

VC

Visual Completeness



The First Meaningful Paint



Does it work for Your Site?

www.example.com Go

Want to double your free teer?

Send a mail with WPMEETUP to support@baqend.com

test.speed-kit.com

Wrap Up

PWA

Super cool alternative to native apps

Service Worker

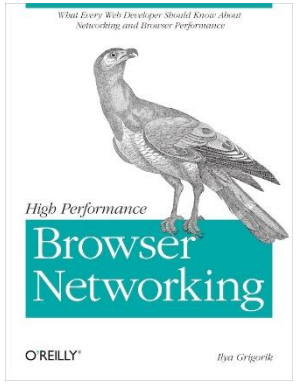
Powerful programmable network proxy

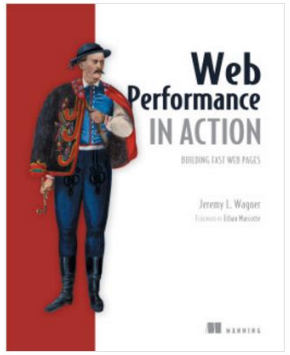
Speed Kit

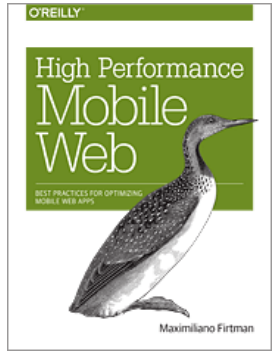
Combines Service Worker and cache coherence

Web Performance Literature

Good Resources







https://hpbn.co/

Google Developers Performance

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

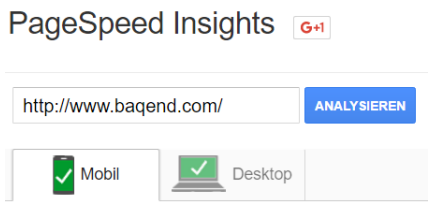
Website Performance Optimization
The Critical Rendering Path

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

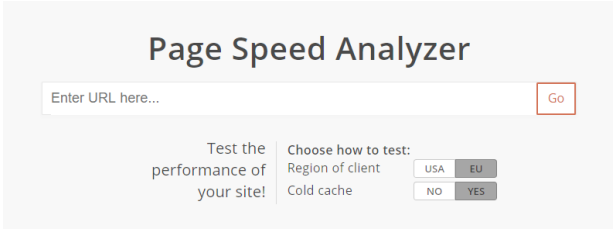
Baqend Blog
On Building a Faster Web

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/

We are hiring.

Frontend Developers
Mobile Developers
Java Developers
Web Performance Engineers

Contact us.

BaQend

Erik Witt · ew@baqend.com · www.baqend.com/jobs.html