Service Workers: The Technology Behind Progressive Web Apps

Workshop at DAHO.AM 2019
Munich, March 28
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 Notifications
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 independent**
  Offline mode

- **Progressive**
  Enhance on capable browsers

- **Re-engageable**
  Engage through Web Push

- **Responsive**
  Fit any form factor

- **Safe**
  HTTPS & recognizable URLs
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**

Progressively enhance when supported

1. **Manifest**
2. **Service Worker**
Implementing **PWAs**

PWAs are **best practices** and **open web standards** when supported.

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

```html
<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:

[Diagram showing the flow of data from Web App to Website, SW.js, Cache, and Network]
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:

```javascript
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.

<table>
<thead>
<tr>
<th>Year</th>
<th>M72</th>
<th>M73</th>
<th>M74</th>
<th>M75</th>
<th>M76</th>
<th>M77</th>
<th>M78</th>
<th>M79</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019</td>
<td></td>
<td></td>
<td></td>
<td>M5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- M5 (Q1): Image Clipboard Support
- M6 (Q2): Shape Detection
- M6 (Q2): Wake Lock
- M7 (Q3): App Icon Badging
- M7 (Q3): Contacts Picker
- M8 (Q4): Native File System Access
- M8 (Q4): Phone Number Sign Up
- M8 (Q4): Unlimited Quota
- M8 (Q4): Scheduled Notifications
- M9 (Q4): Web Serial / Web HID
- M9 (Q4): Launch Event
- M9 (Q4): File Handling

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?

- **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**

<table>
<thead>
<tr>
<th></th>
<th>IE</th>
<th>Edge</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>iOS Safari</th>
<th>Opera Mini</th>
<th>Chrome for Android</th>
<th>UC Browser for Android</th>
<th>Samsung Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11</td>
<td>18</td>
<td>66</td>
<td>74</td>
<td>12.1</td>
<td>12.2</td>
<td>all</td>
<td>74</td>
<td>11.8</td>
<td>9.2</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>67</td>
<td>75</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supported by **>90%** of browsers.
Requires **TLS Encryption**.
Late, but all in: Microsoft

Publish PWAs to Microsoft Store

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?

```javascript
navigator.serviceWorker.register('/sw.js');
```
What does the **lifecycle** look like?

```javascript
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?

```javascript
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
});
```
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

- Stores Data Persistently
- Stores Structured Data

IndexedDB
an actual database in the browser

- Supports Range Queries
- Browser Support 94%
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 timeline** when browser closed
Service Worker Debugging

Service Workers

www.baqend.com

Source: sw//7dbf553e.js

Status: #823 activated and is running stop

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

Push: Test push message from DevTools.

Sync: test-tag-from-devtools
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.

1. Backend Processing
2. Network Delays
3. Client
Solution: Speed Kit
Service Worker rewrites & accelerates slow requests.

1. Fast
2. Less Processing
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

Learn [more](#).
How Speed Kit leverages Service Workers.

Website with Snippet

Requests

Speed Kit Service Worker

Fast Requests

Baqend Service

Sync

Existing Backend

Tracking & Ad Services
Use case I: optimize images.
SW sends client resolution → responsive image.

Device

Speed Kit CDN

- WebP 640x320px
  - 100 KB
- JPG 1280x640px
  - 500 KB

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

Width: 640px
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.

Has Time-to-Live (expiration)

False-Positive Rate:

\[ f \approx (1 - e^{-\frac{kn}{m}})^k \]

Hash-Functions:

\[ k = \left\lfloor \ln(2) \cdot \left(\frac{n}{m}\right) \right\rfloor \]

With 20,000 entries and a 5% false positive rate: **11 Kbyte**

**Consistency:** Δ-Atomicity, Read-Your-Writes, Monotonic Reads, Monotonic Writes, Causal Consistency

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/

Rethinking Web Performance with Service Workers
30 Man-Years of Research in a 30-Minute Read

This article surveys the current state of the art in page speed optimization. It contains the gist of more than 30 man-years of research that went into Speed Kit, an easy-to-use web performance plugin to accelerate any website.
Learn more about Services Workers.

Recommended Books

- Building Progressive Web Apps
- Progressive Web Apps
- Beginning Progressive Web App Development

Guides & Tutorials

- Progressive Web Apps
  - A new way to deliver amazing user experiences on the web.
  - https://developers.google.com/web/progressive-web-apps/

Blogs

- Baqend Blog
  - On Building a Faster Web
  - https://blog.baqend.com/
- Ilya Grigorik
  - Internet plumber
  - https://www.igvita.com/
- Jake Archibald
  - I discovered a browser bug
  - https://jakearchibald.com/

Progressive web apps

- Jump to: PWA advantages, Core PWA guides, Technology guides
Thank You

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