Erik Witt

Service Workers in Action

Building an Offline-Capable Webshop in under 90 minutes

Mobile

Univiersität Hamburg
code.talks
Baqend
What I Have Todo with Service Workers?

Research:
- NoSQL Databases
- Caching Algorithms for Dynamic Data

Practice:
- Speed Kit Lead
- Combining Caching Algorithms with Service Workers
Theory

Practice

Agenda

Case Study

PWA Overview

Basic PWA

Service Worker Overview

Offline + Caching Strategies

Cache Coherence

Implement Speed Kit

Implement Web Push
Demo


Repository:
https://github.com/ErikWitt/code-talks19
Demo

What are Progressive Web Apps?
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

Progressive Web Apps
Combine the best from native and web apps.
Progressive Web Apps (PWA)

- Fast Loads through Caching
- Offline Mode (Synchronization)
- Add-to-Homescreen and Push Notifications
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

https://developer.mozilla.org/en-US/docs/Web/Apps/Progressive/Advantages
Just Released: **Desktop PWAs**

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

**Customizable** installation process/UI (with Event beforeinstallprompt)
PWAs are best practices and open web standards

Progressively enhance when supported

1. Manifest
2. Service Worker
PWA Implementation

PWAs are best practices and open web standards. Progressively enhance 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"
}
```
PWAs are best practices and open web standards.

Progressively enhance when supported.

2. **Service Workers** for caching & offline mode:

```
Web App  →  Website
          ↓  SW.js  ↓  Cache
          →  Network
```
PWAs are best practices and open web standards. Progressively enhance when supported.

3. Add **Web Push** and **Background Sync**:
Which “Fancy” Features Do and Will PWAs Support?
Web Payment APIs

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

Credentials Management API

1. Click **Sign-in** → Native Account Chooser

2. Credentials API stores information for future use

3. **Automatic** Sign-in afterwards
Geofencing

- **Notify** web app when user leaves or enters a defined area
- Requires **permission**
Web Speech API
Native Speech Recognition in the Browser:

```javascript
annyang.addCommands({
  'Hello Code Talks': () => {
    console.log('Hello you.');
  }
});
```
Web Share API

- **Share** site through native share sheet UI
- Service Worker can register as a **Share Target**
Many More in the **Pipeline**

Planned Shipping Dates of Chrome PWA Features.

<table>
<thead>
<tr>
<th>Year</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></td>
<td></td>
</tr>
<tr>
<td>Q1</td>
<td>M72</td>
<td>M73</td>
<td>M74</td>
<td>M75</td>
<td>M76</td>
</tr>
<tr>
<td>Q2</td>
<td>M77</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td>M78</td>
<td></td>
<td></td>
<td></td>
<td>M79</td>
</tr>
<tr>
<td>Q4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Image Clipboard Support
- Shape Detection
- Wake Lock
- App Icon Badging
- Contacts Picker
- Native File System Access
- Phone Number Sign Up
- Unlimited Quota
- Scheduled Notifications
- Web Serial / Web HID
- Launch Event
- File Handling

[https://www.youtube.com/watch?v=2KhrmFHLuhE (Google I/O 2019)]
Demo
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

- **Cache** Data (CacheStorage)
- **Store** Data (IndexedDB)

Service Worker

Network

- Receive **Push**
- Respond when **Offline**
What do Service Workers Do?

- **Intercept** HTTP Requests
- **Sync** Data in Background
- Hide **Flaky Connectivity** from the User
Service Workers Browser Support

<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>Ver</td>
<td>17</td>
<td>68</td>
<td>69</td>
<td>77</td>
<td>13</td>
<td>13.1</td>
<td>all</td>
<td>76</td>
<td>12.1</td>
<td>9.2</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>69</td>
<td>70</td>
<td>78</td>
<td>TP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10.1</td>
</tr>
<tr>
<td></td>
<td>76</td>
<td>70</td>
<td>71</td>
<td>79</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supported by >90% of browsers. Requires TLS Encryption.
How are Service Workers Registered

Browser Tabs ➔ Service Worker ➔ Network

```javascript
<script>
    navigator.serviceWorker.register('/sw.js');
</script>
```
Service Worker Lifecycle

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

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

self.addEventListener('fetch', (event) => {
  // React to fetch event
});
```
const client = await clients.get('id');
client.postMessage(someJsonData);

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

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

Scope

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

IndexedDB
an actual database in the browser

- Stores Data Persistently
- Stores Structured Data
- 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
Cache Storage
Stores Request/Response pairs

Cache Storage
- Programmatically managed
- Persistent and non-expiring

- Supports only HTTP
- Only caches GET requests (no HEAD)
Break => Demo
What Are Service Workers

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

Network, Cache Fallback

Gets requests from network, the cache acts as fallback (offline mode).
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: Cache, Then Network

Gets requests from cache first and from network in background.
Major Challenge

Cache Coherence + Personalization
How To Solve Cache Coherence

Automatic Browser Cache Coherence

Validate Freshness

Expiration Cache

Invalidation Cache

invalidate

Add to Server Cache Sketch

Compact Cache Sketch

[GSW+15, GSW+17]
How To Solve Cache Coherence

Example

1500 entries & 0.0019% false positives → 7 KB size

>25% cache hits on HTML in production

[GSW+15, GSW+17]
How to Solve Personalized Content

Browser (blocks marked by selector)

Fast & Anonymous

Caching Infrastructure

Replace Dynamic Blocks

Slow & Personalized

Origin Server
We Are Baqend

We bring performance research to practice.

30+ man-years of web performance research

Novel technology for caching dynamic data

Speed Kit – SaaS for e-commerce speed
How Speed Kit Works

Website → Speed Kit Service Worker → Fast Requests → Speed Kit Cloud

- Real-Time Sync
- Origin Backend
- 3rd Parties
Accelerated 3rd Parties

With Speed Kit

Speed Kit CDN

Warm & Fast HTTP/2 Connection

Browser

Without Speed Kit

Cold & Slow TCP/TLS Handshakes

Tag Com. CDN

Static Trbo

trbo

3rd Party Servers

AB Tasty Assets
Automatic Image Optimization

Device

Optimizer (on the edge)

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

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

Width: 640px
Dynamic Blocks in Detail

How the browser builds a page

Browser → HTML → CSS → JS → Execution → Painted
Dynamic Blocks in Detail

Origin Request

Browser ➔ HTML ➔ HTML

Painted

CSS

delay execution

Complete

JavaScript
How Dynamic Blocks Help Rendering

Speed Kit

Original

100 ms 200 ms 300 ms

time
What do we gain?
Real-User Monitoring

Browser

Cloud Backend

Tracking Beacon

- Time-to-First-Byte
- First (Contentful) Paint
- DOM Timer
- First Input Delay
- Session Length
- Time on Site
- First User Interaction
- Bounce Rate
- Cart Size
- Transactions
- Conversion Rate
- Revenue
- Page Views & Sessions
- Browser Distribution
- JavaScript Errors
- Caching Insights

Performance
User Engagement
Business KPIs
Q&A Metadata

Timing API
Service Worker
Unhandled Errors
First Contentful Paint

Original: 3.4 s
Speed Kit: 0.5 s

*Median performance compared between the two A/B test split groups on the Appelrath website*
*comparing appelrath.com Chrome UX report data from October 2018 with January 2019*
Appelrath: Business Impact

- User-based Conversion Rate: + >10%
- Overall Time for Speed Kit Setup: 2 Days
Wrap Up

PWAs
Super cool alternative to native apps

Service Workers
Powerful programmable network proxy

Use Case
Speed Kit: Smart CDN through 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 Service Workers

Recommended Books

*Building Progressive Web Apps* by You Yore

*Progressive Web Apps* by Dhruv Bali andlogin

*Beginning Progressive Web App Development* by Agapitov

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**

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

Questions?

Erik Witt, Speed Kit Lead   ew@baqend.com