The Technology Behind Progressive Web Apps

Service Workers in Detail

Felix Gessert
fg@baqend.com
May 2018 – New Technology Meetup

@baqendcom
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 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
Building Blocks of PWAs

PWAs are best practices and open web standards

1. Manifest

2. Service Worker

Progressively enhance when supported
Implementing **PWAs**

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": "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:

- Web App
- Website
- SW.js
- Cache
- 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
What is the future of Progressive 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 or enters a defined area
- Requires **permission**
The Future of PWAs is bright.

Web Speech API
Native Speech Recognition in the Browser:

```javascript
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

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?

```html
<script>
  navigator.serviceWorker.register('/sw.js');
</script>
```
How 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
});
```
Intercepting **Network Requests**

There is so much you can do:
- **Rewrite** Request
- **Change** Response
- **Concat** Responses
- **Cache** Responses
- **Serve** Cached Data
- ...
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');
// 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/' });

Scope can be restricted but not widened
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
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.

Backed by 30 Man-Years of Research


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

How Speed Kit solves Cache Coherence

Has **Time-to-Live** (expiration)

False-Positive Rate: \( f \approx \left(1 - \frac{\text{hash}}{n}\right)^k \)

Hash-Functions: \( k = \left\lfloor \ln(2) \cdot \frac{n}{m} \right\rfloor \)

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

**Consistency**: \( \Delta \)-Atomicity, Read-Your-Writes, Monotonic Reads, Monotonic Writes, Causal Consistency

\[ f \approx 1 - e^{-k/n} \]

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

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

- Website with Snippet
- Speed Kit Service Worker
- Baqend Service

Requests → Fast Requests → Realtime, Scheduled

- 3rd Party Services
- Existing Backend
Optimized & Cached Images

Device

Width: 640px

Speed Kit CDN

WebP 640x320px 100 KB

JPG 1280x640px 500 KB

Images transcoded to WebP

Rescaled to match Screen Size

JPG and PNG Recompression
Now, we have a Progressive Web App. How do we measure web performance?
Your Website
1131ms

3.1x Faster

With Speed Kit
366ms

With Speed Kit
0.4s

Excellent

Your Website
1.1s

Poor

Show Details
User-perceived performance

Speed Index: avg. time to visibility

First Meaningful Paint: greatest visible change

\[ \int_0^\infty 1 - VC(t) dt \]

Visual Completeness

Time (s): 0 0.1s 0.2s 0.3s 0.4s 0.5s
Does it work for Your Site?

www.example.com

Go

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/

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.
Web Performance Literature

Good Resources

- https://developers.google.com/web/fundamentals/performance/?hl=en
- https://hpbn.co/
- https://medium.baqend.com/

Performance Tools

- https://developers.google.com/speed/pagespeed/
- https://test.speed-kit.com
- https://www.baqend.com/
- http://www.webpagetest.org/
We are hiring.

- Frontend Developers
- Mobile Developers
- Java Developers
- Web Performance Engineers

Contact us.

Felix Gessert · fg@baqend.com · www.baqend.com/jobs.html