

Progressive Web Applications



<https://iuk.one/1066-1047>

Clemens H. Cap

ORCID: 0000-0003-3958-6136

Department of Computer Science
University of Rostock
Rostock, Germany
clemens.cap@uni-rostock.de

Version 2



1. Einführung

Was ist das...

1. Einführung

2. Web App Manifest

3. Neue APIs

4. Service Worker

5. App Shell Architektur

Anforderungen an Progressive Web Applications (PWA)

Nahtlose Integration

- Auf Home Screen positionierbar
- Zugang zu nativen Funktionalitäten (z.B., Push Notifications, Filesystem)
- Zugang zu lokaler Hardware (z.B., Sensorik, Aktorik)

Online: Schnell und Offline: Möglich

- Caching und Updating
- Daten-Synchronisation

Responsive (besser: adaptive)

- Passen auf alle (auch: mobilen) Formfaktoren
- Passen sich an Screen Veränderungen dynamisch an (Landscape, Portrait)

Zentrale Elemente:

- Web App Manifest
- Neue APIs
- Service Worker
- Application Shell Architektur

Single Page Application: SPA

Grundidee:

- Anwendung lädt nur eine Seite
- Anschließend dynamischer Seitenumbau

Vorteile:

- Besser Benutzererfahrung: Keine Seiten-Zentrierung, mehr responsiv
- Geringere Serverlast: Nur Daten versendet
- Schnellere Ladezeiten
- Reichhaltigere, interaktiv-responsive Funktionalitäten

Nachteile:

- Initiale Ladezeiten
- SEO ist schwieriger
- Komplexität der Anwendung

2. Web App Manifest

Metadaten und Bündelung

1. Einführung
2. Web App Manifest
3. Neue APIs
4. Service Worker
5. App Shell Architektur

Web App Manifest

Meta-Daten und Ressourcen einer Anwendung

Zweck

- Ermöglicht die Installation als PWA
- Gewährleistet Vollständigkeit der Assets
- Gestattet Parametrisierung und Aussehen
- Ermöglicht dem Cache verschiedene Preload-Strategien

Form

- JSON Datei
- Key-Value Paare
- `name`: Vollständiger Name der Anwendung
- `short_name`: Name der Anwendung auf dem Home Screen
- `start_url`: URL die beim Start angezeigt wird
- `display`: `fullscreen`, `standalone`, `minimal-ui`, `browser`

2. Web App Manifest

Manifest Beispiel

```
1  {
2    "name": "My Web App",
3    "short_name": "WebApp",
4    "description": "An example of a manifest file for a web application.",
5    "start_url": "/index.html",
6    "display": "standalone",
7    "background_color": "#ffffff",
8    "theme_color": "#000000",
9    "orientation": "portrait",
10   "icons": [
11     {"src": "/images/icons/icon-72x72.png",
12      "sizes": "72x72",
13      "type": "image/png" }
14  ],
15   "lang": "en-US",
16   "dir": "ltr",
17   "scope": "/"
18 }
```


3. Neue APIs

Einige APIs für PWAs

1. Einführung
2. Web App Manifest
- 3. Neue APIs**
4. Service Worker
5. App Shell Architektur

Web Payment API

- Payment ist (eigentlich) ein standardisierter Vorgang
- Aber: Bisher meist Anbieter-spezifisch implementiert

Probleme:

- UI sieht immer wieder anders aus
- Keine Unterscheidung: Authentisch / phishing
- Payment nicht separat in „eigener“ Komponente absicherbar
- User muss Daten immer wieder neu eingeben

Betrifft auch:

- Auslieferungsadressen usw
- Es gibt Auto-Form-Fillin, aber es ist unsicher

Neu:

- Standardisierung aller dieser Vorgänge

Web Storage

Situation:

- Ziel: Persistente, Client-seitige Speicherung
- Bisher: Cookies mit Transport im HTTP Header, eingeschränkt
- Jetzt: Erst in HTML5, nun in separatem Standard

Features:

- Rund 2 – 10 MB je nach Implementierung
- API: Nur über Client Skripting
- Als key-value store ausgelegt
- Nicht direkt durch Server (wie bei Cookies)

Eigenschaften:

- Local storage: Ist pro remote domain
- Session storage: Ist pro Browser-Fenster
- Anders als Cookies, die pro-Browser implementiert sind
- Erlaubt mehrere parallele Sessions mit 1 Browser

WebStorage (2)

Einfacher Key-Value Store:

- `localStorage.setItem(key, value);`
- `localStorage.getItem(key);`
- `localStorage.length;`
- `localStorage.removeItem(key);`
- `localStorage.key(index);`
- `localStorage.clear();`

WebStorage (2)

Datentyp: String, Objekte also mittels Serialisierung (`JSON.stringify`, `JSON.parse`)

Synchron: Blockiert den Main Thread

Sicherheit:

- Same origin Restriktion
- Kann aber von jedem Script auf der Seite gelesen werden

Persistenz und Privatheit:

- `localStorage`: Dauerhaft bis zum Löschen durch User
- `sessionStorage`: Nur für aktuelle Sitzung

Situation:

- Ziel: API die ein eingeschränktes clientseitiges SQL(ite) erlaubt
- Teilweise von Browsern unterstützt
- Normierung zunächst noch nicht abgeschlossen. . .
- und dann abgekündigt

Situation:

- Key-Value Store
- Unterstützt Transaktionskonzept
- Interface weitestgehend asynchroner Natur
- Für größere Datenmengen als Webstorage

Eigenschaften:

- Beispiel: Firefox.: Bis 50% der Harddisc des Profile Directories
- Unterliegt same origin Restriktionen
- Direkte Nutzung etwas aufwendig
- Empfohlen wird Nutzung von Wrapper Libraries
- Beispiel: PouchDB

Geolocation API

Als ein Beispiel von vielen weiteren...

- Gestattet Zugriff auf den Ort des Anwenders
- Hilfreich für mobile Anwendungen
- Zugriff erfordert Erlaubnis durch den Benutzer
- Noch nicht ganz einheitlich in der Implementierung
- Je nach Browser und Endgerät unterschiedlich

Grundproblematik neuer APIs

- Viele neue Standards bieten erhöhte convenience
- Preis ist aber
 - Verlust von Privatheit
 - Erhöhte Bindung an einen Browser / Anbieter
 - Wunsch nach cross provider Synchronisation
 - Zusätzliche Sicherheitslücken
 - Notwendigkeit neuer Tools / Regelungen
- Warum nicht gleich den User stärken?

4. Service Worker

Zentrales Element einer (on)(off)line PWA

1. Einführung
2. Web App Manifest
3. Neue APIs
4. Service Worker
5. App Shell Architektur

Service Worker Einführung

Grundidee:

- Script, das im Hintergrund läuft
- Losgelöst von der Webseite selber
- Kann alle Netzwerk-Zugriffe interceptieren und beantworten
- API cache so, dass Mißbrauch von Features erschwert wird
- Beispiel: X-Frame-Options Entfernung geht nicht über Service Worker

Einsatz:

- Caching für offline-Zeiten
- Background Synchronisation
- Push Notifications an den User
- Preloading, prefetching
- Unterschiedliche Versionen einer Anwendung
- Proxy Server für Dienste
- Beispiel: Google APIs als Dev. nutzen ohne die rate limitation zu beanspruchen

Service Worker Lifecycle

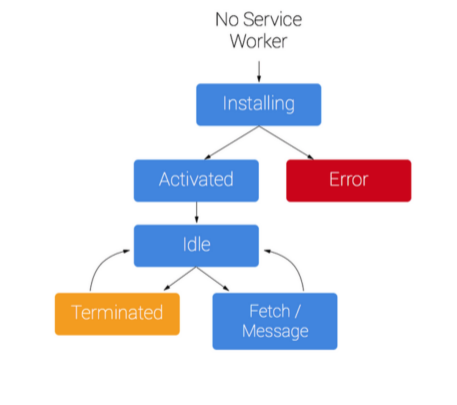


Abb. 1: Lebenszyklus eines Service Workers

Service Worker (1)

Spezifische Sicherheitsaspekte

- Kann gesamte Site total verändern!
- Zugriff nur unter https (oder: localhost)
- Nur Seiten unterhalb eines scope Punktes dürfen nutzen

Diverse Kinderkrankheiten

- Was, wenn der Serviceworker selber gecached wird und der Cache Zeitraum zu lange eingestellt wurde?
- Lösung: Alle 24h neuer Download Versuch

Service Worker (2)

Beispiel für Use Pattern: Vom Netz, wenn nicht verfügbar vom Cache

- Sinnvoll für offline Anwendung
- Wenn Netz plötzlich wieder da ist:
 - User Notification
 - User kann entscheiden, aktuelle Version laden
 - Achtung vor Versions-Mix in Dateien

Beispiel für Use Pattern: Vom Cache, wenn nicht verfügbar vom Netz

- Sinnvoll für Dateien, die sich selten ändern

4. Service Worker

Service Worker Cookbook

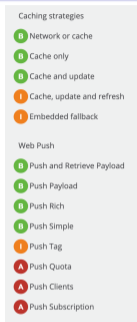


Abb. 2: Service Worker Cookbook <https://serviceworkers.rs/>

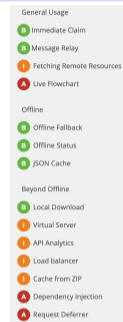


Abb. 3: Service Worker Cookbook <https://serviceworkers.rs/>

Wichtige Support-Frameworks:

- Service Worker Cookbooks
- Chrome Workbox

5. App Shell Architektur

Grundkonzept der Architektur-Entwicklung

1. Einführung
2. Web App Manifest
3. Neue APIs
4. Service Worker
5. App Shell Architektur

5. App Shell Architektur

Grundidee

Was ist die App Shell Idee:

- App Shell ist die minimale Struktur der Anwendung
- Ist in einem Service Worker gecached
- Lädt sofort, begrüßt Benutzer
- Weitere Elemente der Anwendung dann
 - dynamisch geladen
 - entsprechend der User-Anforderungen gecached

Performance: Geringer

Installation:

- Direkt im Browser
 - Keine Notwendigkeit eines App Stores
 - Keine Überprüfung durch App Store Provider
- Geringere Sicherheit und UI Conformance

Entwicklungskosten:

- Nur eine Codebase (Web) statt Swift, Java, ...
- Leichter zu warten
- Risiko: Abhängig von Unterstützung durch Plattform

Anhang

Übersicht

Verzeichnis aller Abbildungen

Abb

Rechtliche Hinweise

§

Zitierweise dieses Dokuments

→

Verzeichnis aller Folien



1	Lebenszyklus eines Service Workers	20
2	Service Worker Cookbook https://serviceworke.rs/	23
3	Service Worker Cookbook https://serviceworke.rs/	23

Rechtliche Hinweise (1)

Die hier angebotenen Inhalte unterliegen deutschem Urheberrecht. Inhalte Dritter werden unter Nennung der Rechtsgrundlage ihrer Nutzung und der geltenden Lizenzbestimmungen hier angeführt. Auf das Literaturverzeichnis wird verwiesen. Das **Zitatrecht** in dem für wissenschaftliche Werke üblichen Ausmaß wird beansprucht. Wenn Sie eine Urheberrechtsverletzung erkennen, so bitten wir um Hinweis an den auf der Titelseite genannten Autor und werden entsprechende Inhalte sofort entfernen oder fehlende Rechtsnennungen nachholen. Bei Produkt- und Firmennamen können Markenrechte Dritter bestehen. Verweise und Verlinkungen wurden zum Zeitpunkt des Setzens der Verweise überprüft; sie dienen der Information des Lesers. Der Autor macht sich die Inhalte, auch in der Form, wie sie zum Zeitpunkt des Setzens des Verweises vorlagen, nicht zu eigen und kann diese nicht laufend auf Veränderungen überprüfen.

Alle sonstigen, hier nicht angeführten Inhalte unterliegen dem Copyright des Autors, Prof. Dr. Clemens Cap, ©2020. Wenn Sie diese Inhalte nützlich finden, können Sie darauf verlinken oder sie zitieren. Jede weitere Verbreitung, Speicherung, Vervielfältigung oder sonstige Verwertung außerhalb der Grenzen des Urheberrechts bedarf der schriftlichen Zustimmung des Rechteinhabers. Dieses dient der Sicherung der Aktualität der Inhalte und soll dem Autor auch die Einhaltung urheberrechtlicher Einschränkungen wie beispielsweise **Par 60a UrhG** ermöglichen.

Die Bereitstellung der Inhalte erfolgt hier zur persönlichen Information des Lesers. Eine Haftung für mittelbare oder unmittelbare Schäden wird im maximal rechtlich zulässigen Ausmaß ausgeschlossen, mit Ausnahme von Vorsatz und grober Fahrlässigkeit. Eine Garantie für den Fortbestand dieses Informationsangebots wird nicht gegeben.

Die Anfertigung einer persönlichen Sicherungskopie für die private, nicht gewerbliche und nicht öffentliche Nutzung ist zulässig, sofern sie nicht von einer offensichtlich rechtswidrig hergestellten oder zugänglich gemachten Vorlage stammt.

Use of Logos and Trademark Symbols: The logos and trademark symbols used here are the property of their respective owners. The YouTube logo is used according to brand request 2-9753000030769 granted on November 30, 2020. The GitHub logo is property of GitHub Inc. and is used in accordance to the GitHub logo usage conditions <https://github.com/logos> to link to a GitHub account. The Tweedback logo is property of Tweedback GmbH and here is used in accordance to a cooperation contract.

Disclaimer: Die sich immer wieder ändernde Rechtslage für digitale Urheberrechte erzeugt ein nicht unerhebliches Risiko bei der Einbindung von Materialien, deren Status nicht oder nur mit unverhältnismäßig hohem Aufwand abzuklären ist. Ebenso kann den Rechteinhabern nicht auf sinnvolle oder einfache Weise ein Honorar zukommen, obwohl deren Leistungen genutzt werden.

Daher binde ich gelegentlich Inhalte nur als Link und nicht durch Framing ein. Lt EuGH Urteil 13.02.2014, C-466/12 ([Pressemitteilung](#), [Blog-Beitrag](#), [Urteilstext](#)). ist das unbedenklich, da die benutzten Links ohne Umgehung technischer Sperren auf im Internet frei verfügbare Inhalte verweisen.

Wenn Sie diese Rechtslage stört, dann setzen Sie sich für eine Modernisierung des völlig veralteten Vergütungs- und Anreizsystems für urheberrechtliche Leistungen ein. Bis dahin klicken Sie bitte auf die angegebenen Links und denken Sie darüber nach, warum wir keine für das digitale Zeitalter sinnvoll angepaßte Vergütungs- und Anreizsysteme digital erbrachter Leistungen haben.

Zu Risiken und Nebenwirkungen fragen Sie Ihren Rechtsanwalt oder Gesetzgeber.

Weitere Hinweise finden Sie im Netz [hier](#) und [hier](#) oder [hier](#).

Zitierweise dieses Dokuments

Wenn Sie Inhalte aus diesem Werk nutzen oder darauf verweisen wollen, zitieren Sie es bitte wie folgt:

Clemens H. Cap: Progressive Web Applications. Electronic document. <https://iuk.one/1066-1047>
26. 6. 2024.

Bibtex Information: <https://iuk.one/1066-1047.bib>

```
@misc{doc:1066-1047,  
  author      = {Clemens H. Cap},  
  title       = {Progressive Web Applications},  
  year        = {2024},  
  month       = {6},  
  howpublished = {Electronic document},  
  url         = {https://iuk.one/1066-1047}  
}
```

Typographic Information:

Typeset on ?today?

This is pdfTeX, Version 3.14159265-2.6-1.40.21 (TeX Live 2020) kpathsea version 6.3.2

This is pgf in version 3.1.5b

This is preamble-slides.tex myFormat©C.H.Cap

Titelseite	1
1. Einführung	
Anforderungen an Progressive Web Applications (PWA)	3
Kern-Technologien	4
Single Page Application: SPA	5
2. Web App Manifest	
Web App Manifest	7
Manifest Beispiel	8
3. Neue APIs	
Web Payment API	10
Web Storage	11
WebStorage (2)	12
WebStorage (2)	13
Web SQL Database	14
IndexedDB	15
Geolocation API	16
Grundproblematik neuer APIs	17




4. Service Worker

Service Worker Einführung	19
Service Worker Lifecycle	20
Service Worker (1)	21
Service Worker (2)	22
Service Worker Cookbook	23

5. App Shell Architektur

Grundidee	25
Vergleich mit Nativer Anwendung	26

Legende:

-  Fortsetzungsseite
-  Seite ohne Überschrift
-  Bildseite