Version 2024a

BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

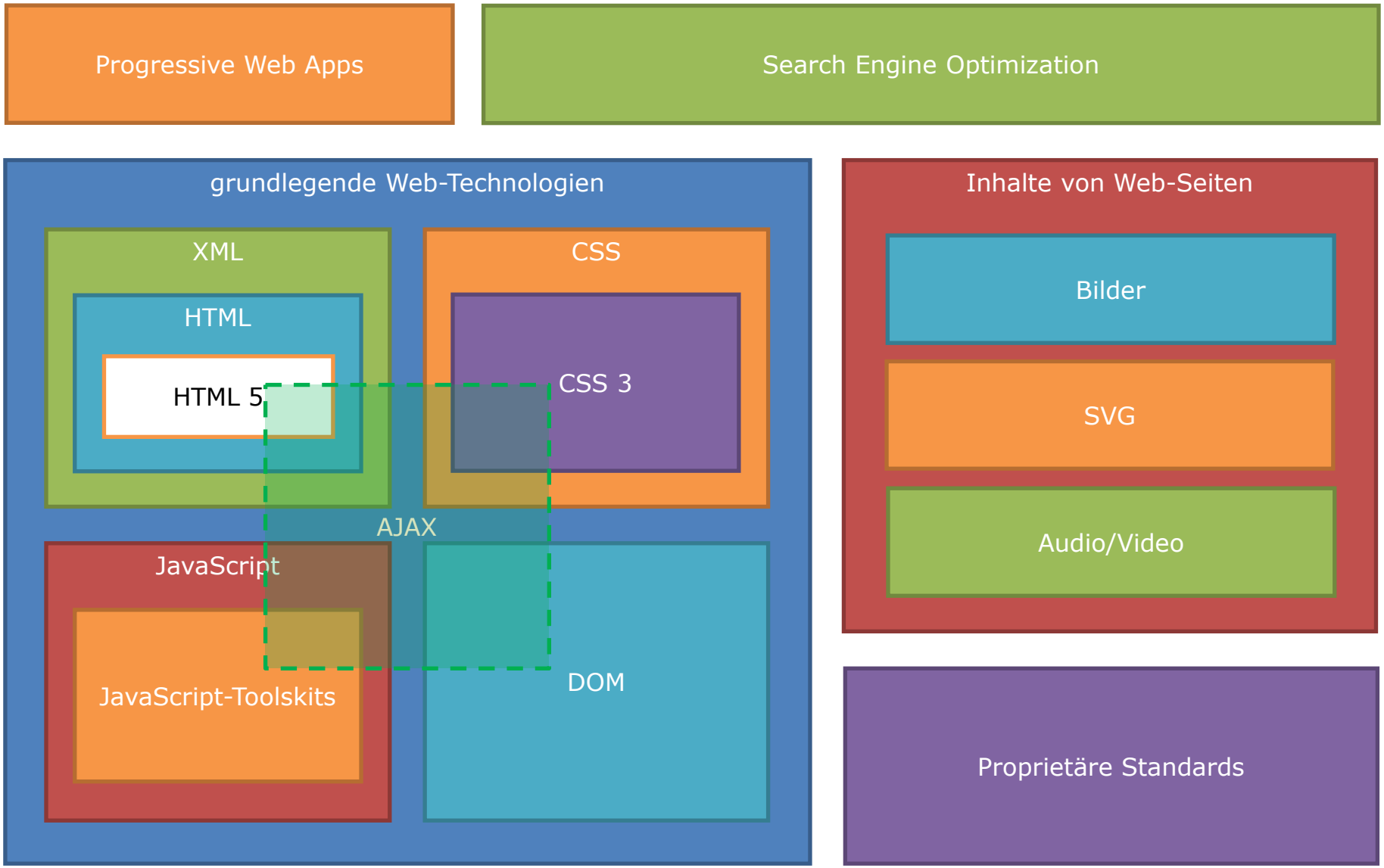
10

Grundlagen der Web-Programmierung

Progressive Web Apps

Prof. Dr.-Ing. Tenshi Hara
tenshi.hara@ba-sachsen.de

Aufbau der Lehrveranstaltung

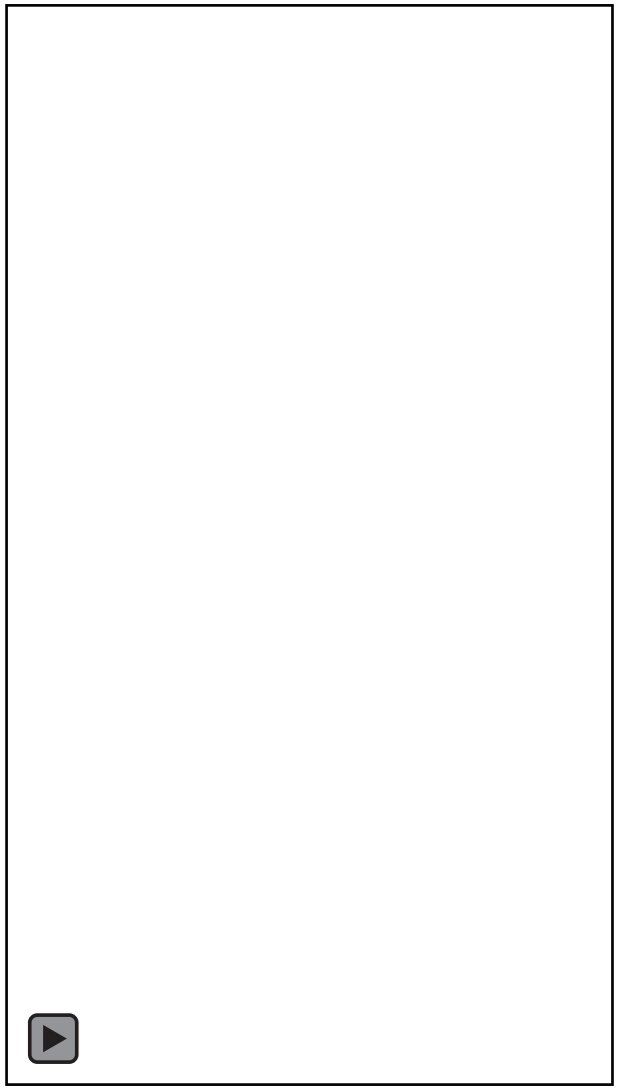


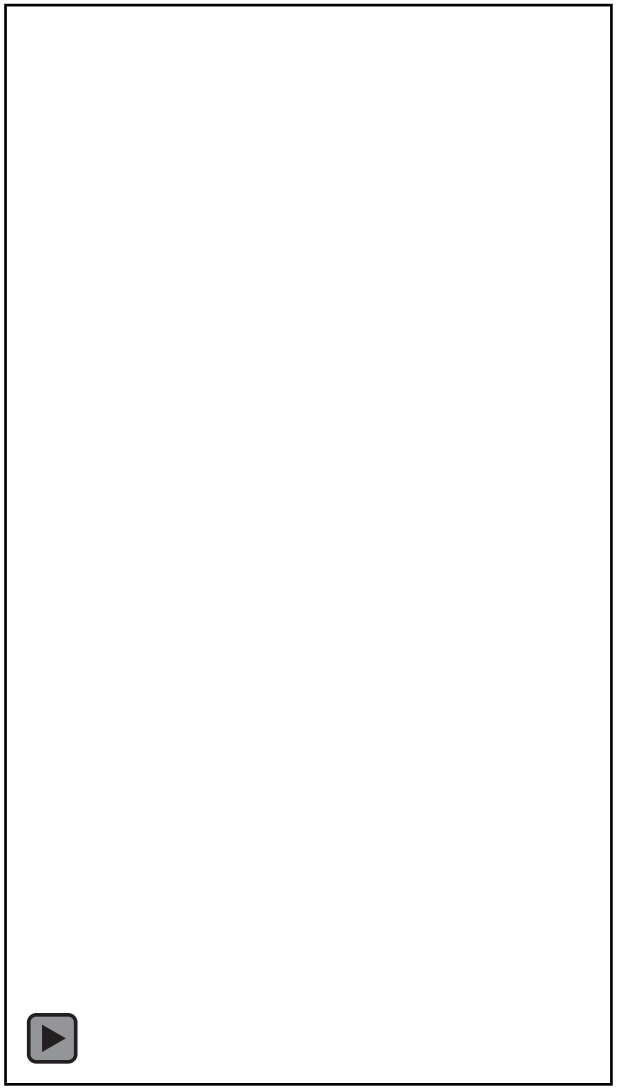
Motivation

- ein Großteil der zum Beziehen von Informationen aus dem Web verwendeten Geräte sind Smartphones und Tablets
- Entwicklung für multiple Plattformen erfordert Zeit und Ressourcen
  → plattformübergr. Ansätze versuchen diese einzusparen
- Vertrieb über App-Stores ist i.d.R. an Bedingungen geknüpft und mittelbar (manchmal auch unmittelbar) Geld
- Web-Seiten leiden oft unter Performance-Problemen
  - Nutzung bei schlechter Verbindung (fast) unmöglich
    → Online-Zwang
  - Nutzer werden zu App-Stores weitergeleitet um den eigentlichen Dienst nutzen zu können

⇒ Progressive Web Apps versuchen Vorteile aller Ansätze zu kombinieren

Motivation





Eigenschaften

Progressive Web Apps

Eigenschaften

- (lokal) installierbar
- progressiv
- verbindungsunabhängig
- App-ähnlich
- wiederbeschäftigend
- sicher
- aktuell
- adressierbar
- responsiv
- entdeckbar

Mittel

- HTML 5
- CSS 3
- Javascript
- Betriebssystembibliothek

[Russel, A.: Progressive Web Apps: Escaping Tabs Without Losing OurSoul]

Progressive Web Apps - Eigenschaften

- installierbar (installable)
  - können auf Wunsch des Nutzers (de)installiert werden
  - bekommen ein eigenes Icon auf dem Homescreen/App-Drawer
- progressiv (progressive)
  - funktionieren für jeden Benutzer auf jedem Gerät
  - nicht vorhandene Funktionalitäten müssen abgefangen werden
- verbindungsunabhängig (connectivity-independent)
  - funktionieren auch bei schlechter oder keiner Netzwerkverbindung
  - Nutzung HTML-5-basierter lokaler Server-Emulation
- App-ähnlich (app-like)
  - fühlen sich wie eine native App an
  - Performance, Navigation und Interaktion ähneln GUI-Philosophie der Geräteplattform

Progressive Web Apps - Eigenschaften

- wiederbeschäftigend (re-engageble)
  - sind kurzweilig und laden zu wiederholter Nutzung ein
  - ermöglichen Push-Notifications und verlassen sich auf diese
- sicher (safe)
  - können nur über HTTPS verwendet werden (Vermeidung von Man-in-the-Middle-Angriffen)
  - lokale Datenhaltung nur im sicheren HTML-5-Storage
- aktuell (fresh)
  - kein klassischer Update Prozess wie bei nativen Apps nötig
  - aktualisieren sich bei bestehender Netzanbindung selbst
- adressierbar (linkable)
  - Links von jedem View (d.h. Seite) können einfach geteilt werden
  - PWA integriert sich als Anbieter für Links („Wollen Sie mit X öffnen“)

Progressive Web Apps - Eigenschaften

- responsiv (responsive)
  - folgen den Vorgaben mobiler Web Applikationen
  - erlauben darüber hinaus gerätespezifische Anpassungen
    → Wegfall der Navigationsleiste (immer Vollbild)
    → Overlay- und PiP-Formate möglich
- entdeckbar (discoverable)
  - werden beim Besuch einer Web-Seite automatisch vorgeschlagen
  - können wie native App im App-Drawer gesucht werden
- abhängig vom Host-Browser
  - werden vom Host-Browser verwaltet
  - werden mit Host-Browser deinstalliert
  - sind vom Datenhaushalt des Host-Browsers abhängig
    → Leeren des Browser-Caches oder Löschen der Browser-Daten betrifft auf alle über diesen Browser installierten PWAs

PWA-Struktur

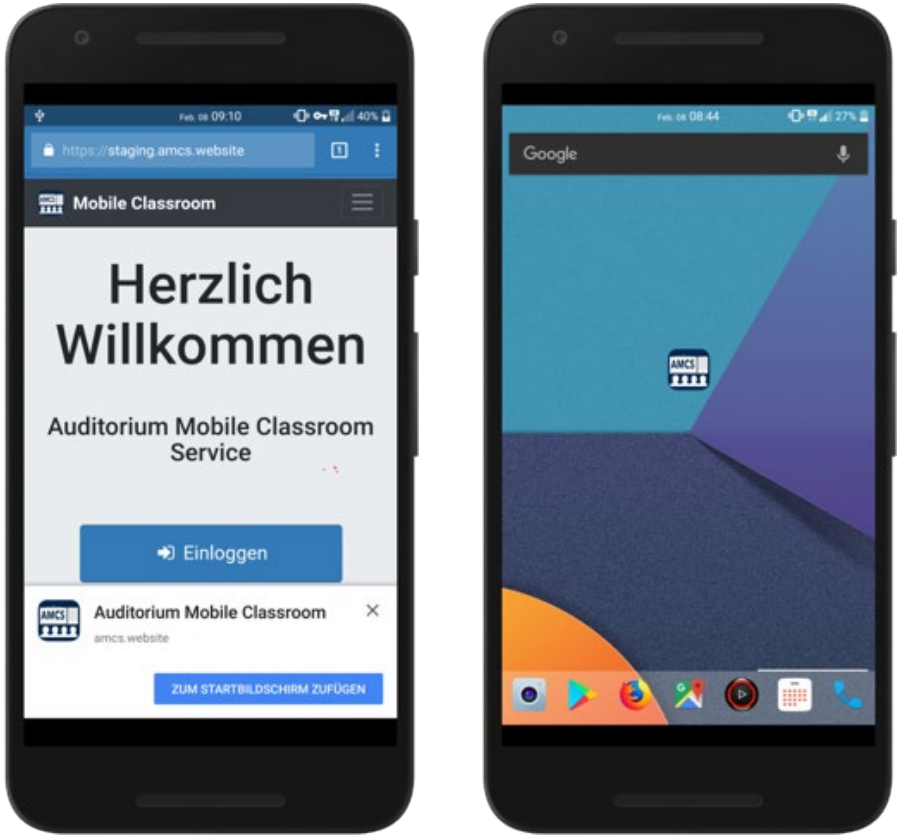
Add to Home Screen (A2HS)

A2HS-Meldung erscheint wenn PWA

- ein Web-App-Manifest hat
- sich als ServiceWorker registriert
- über HTTPs angeboten wird

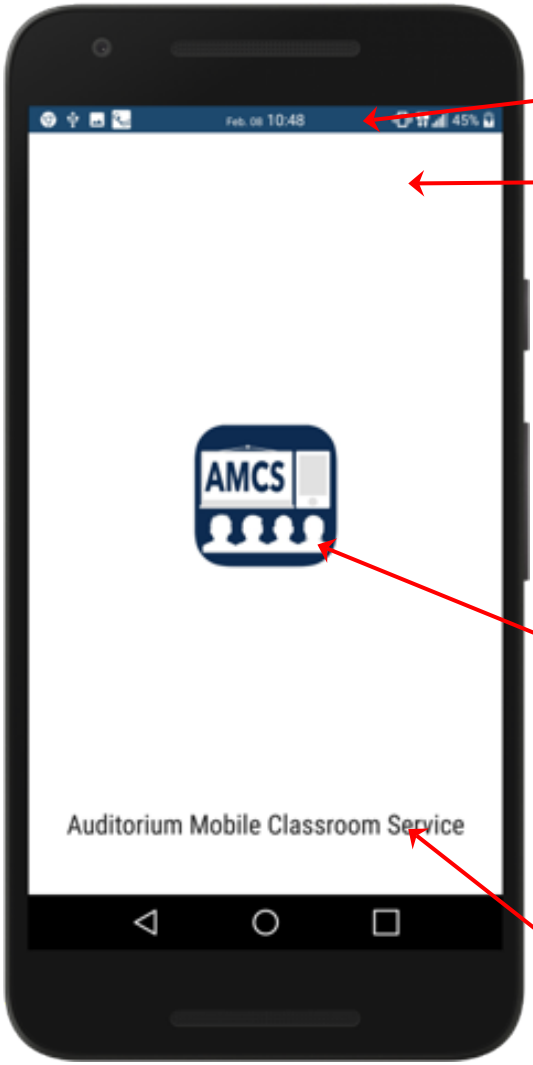
Bei Bestätigung der A2HS-Meldung

- generiert der Browser eine automatisch installierte APK mit einem Android Intent Filter
- Intent Filter fängt alle Links mit URL-Bezug zur PWA ab und öffnet diese in der PWA
- App-Icon wird auf dem Homescreen hinterlegt




```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="https" android:host="amcs.website" android.pathPrefix="/" />
</intent-filter>
```

Web-App-Manifest




```
{
  "short_name": "AMCS",
  "theme_color": "#337ab7",
  "start_url": "./index.html",
  "background_color": "#ffffff",
  "display": "standalone",
  "orientation": "portrait",
  "gcm_sender_id": "103953800507",
  "icons": [
    {"src": "assets/android-chrome-36x36.png" …},
    {"src": "assets/android-chrome-48x48.png" …},
    {"src": "assets/android-chrome-72x72.png" …},
    {"src": "assets/android-chrome-96x96.png" …},
    {"src": "assets/android-chrome-144x144.png" …},
    {
      "src": "assets/android-chrome-192x192.png",
      "sizes": "192x192",
      "type": "image\/png"
    },
    {"src": "assets/android-chrome-512x512.png" …}
  ],
  "name": "Auditorium Mobile Classroom Service"
}
```

Architektur der App-Hülle

Ziel: zuverlässiges und instantanes Laden der GUI

- minimale Ressourcen zur Anzeige der grundlegenden GUI (HTML, CSS, JS, Icons, …)
- Ressourcen werden durch ServiceWorker bei seiner Installation zwischengespeichert
- bei konsekutiven App-Starts können die Ressourcen direkt vom Cache geladen werden
  → keine Netzwerkverbindung nötig

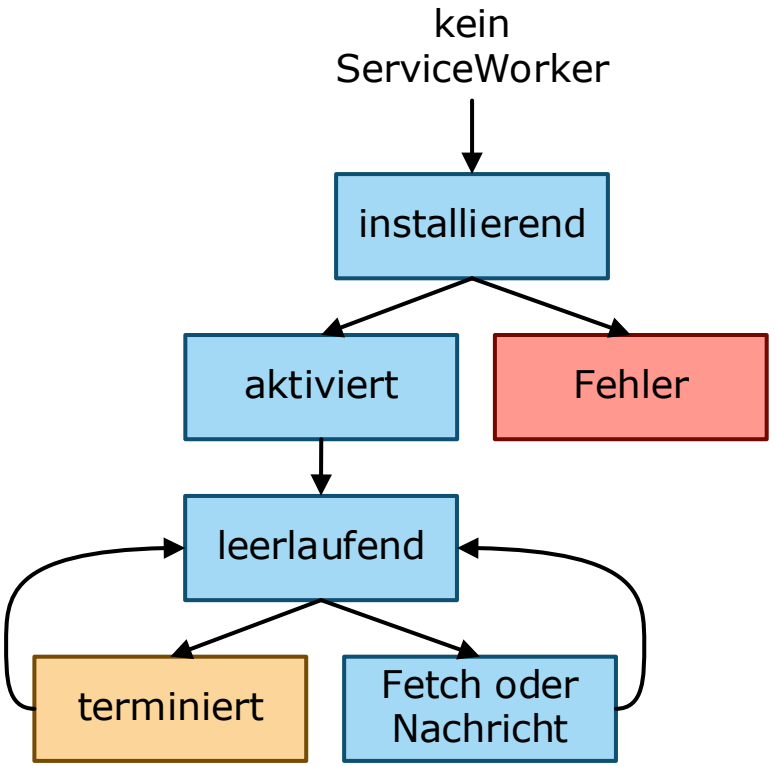
```
self.addEventListener(
  'install',
  event => {
    event.waitUntil(
      caches
        .open('static-v1')
        .then(
          cache => cache.addAll([
            '/index.html',
            '/styles/styles.css',
            '/scripts/app.js',
            '/images/image.png'
          ])
        )
    );
});
```

Architektur der App-Hülle – ServiceWorker

- im Browser im Hintergrund laufendes Skript
- arbeitet als Netzwerk-Proxy (Server-Emulation)
- registriert sich für `install`, `activate`, `message`, `fetch`, `push` und `sync` Ereignisse
- ermöglichte Eigenschaften
  - geschmeidiger Update-Prozess
  - Caching
  - Push-Benachrichtigungen
  - Hintergrundsynchronisierung
- verfügbar in
  - Chrome 42+
  - Firefox 48+
  - Safari 11.1+ (iOS 11.3+ und macOS 10.13.4+)

Update-Prozess

- Browser aktualisiert ServiceWorker-Datei (i.d.R. `service-worker.js`) mindestens einmal täglich im Hintergrund
- bei Änderung wird neue Datei installiert
- neue App-Hülle wird durch `install`-Ereignis geholt
  → bei Erfolg wird ServiceWorker wieder im Hintergrund aktiviert
- Namenszuweisung im HTML-5-Cache wird bei jedem Capistrano-Deploy aktualisiert
  → garantiert Update der App-Hülle



[Gaunt, M.: Service Workers: an Introduction, https://developers.google.com/web/fundamentals/getting-started/primers/service-workers]

Caching Recipes [Archibald, J.: The offline cookbook]

Wo soll gecachet werden?

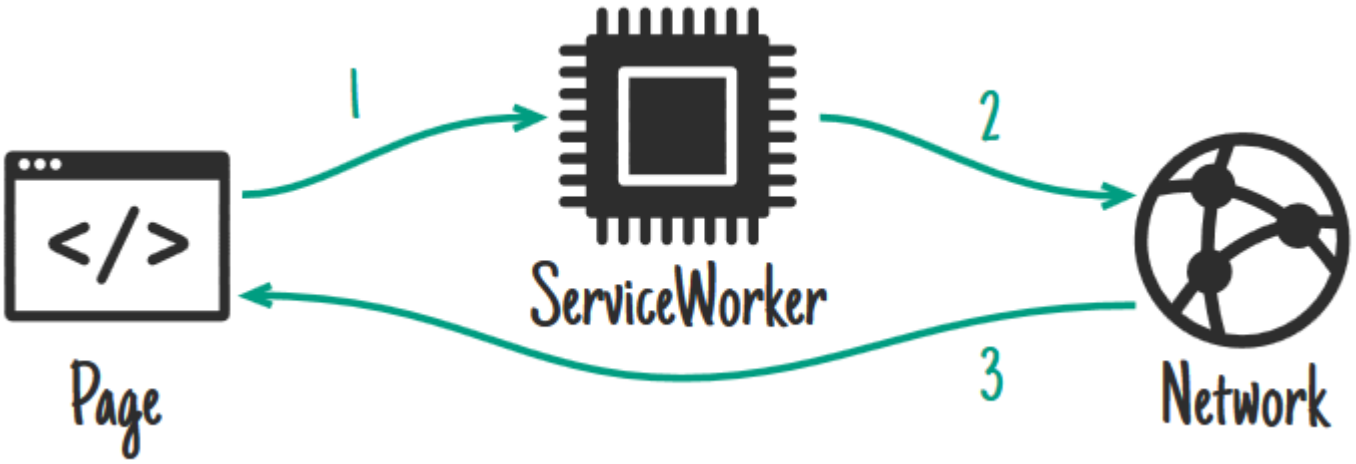
- Abgreifen des Listeners im ServiceWorker
- Cache-API zum Zwischenspeichern von Anfragen
  (kann nur GET-Anfragen speichern)

```
self.addEventListener(
  'fetch', function(event){
    if (event.request.method != 'GET')
      // Netzwerkzugriff
      getData_networkOnly(event);
    else {
      // Cache-Zugriff
      caches
        .open(CACHE_NAME)
        .then(function(cache){
          return cache
            .match(event.request)
            .then(funtion(response){
              // falls Cache nicht leer
              return response ||
                fetch(event.request).then(
                  function(networkResponse){
                    cache.put(
                      event.request,
                      networkResponse.clone()
                    );
                  });
            });
        })
    }
  }
)
```

Caching Recipes und Synchronisierung

Caching Recipes [Archibald, J.: The offline cookbook]

nur Netzwerk

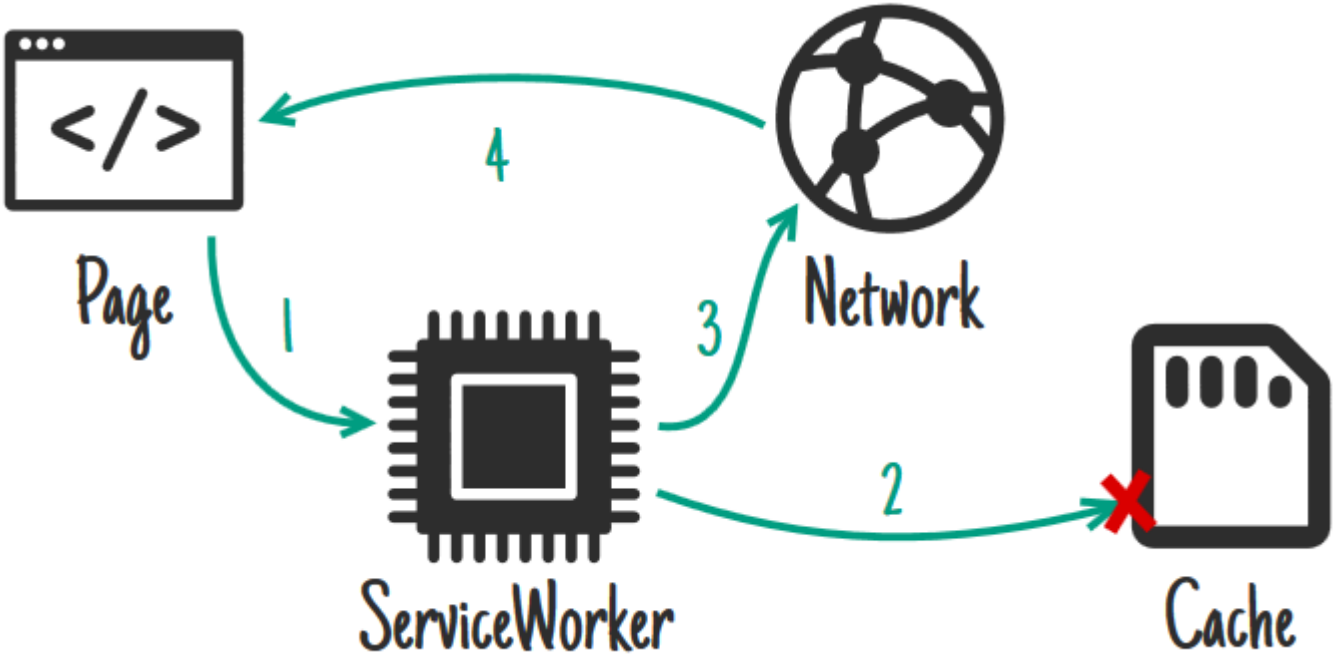


[Archibald, J.: The offline cookbook]

- nicht für GET-Anfragen
- zum Senden von Antworten und Hintergrundsynchronisierung

Caching Recipes [Archibald, J.: The offline cookbook]

Cache mit Netzwerk-Rückfall

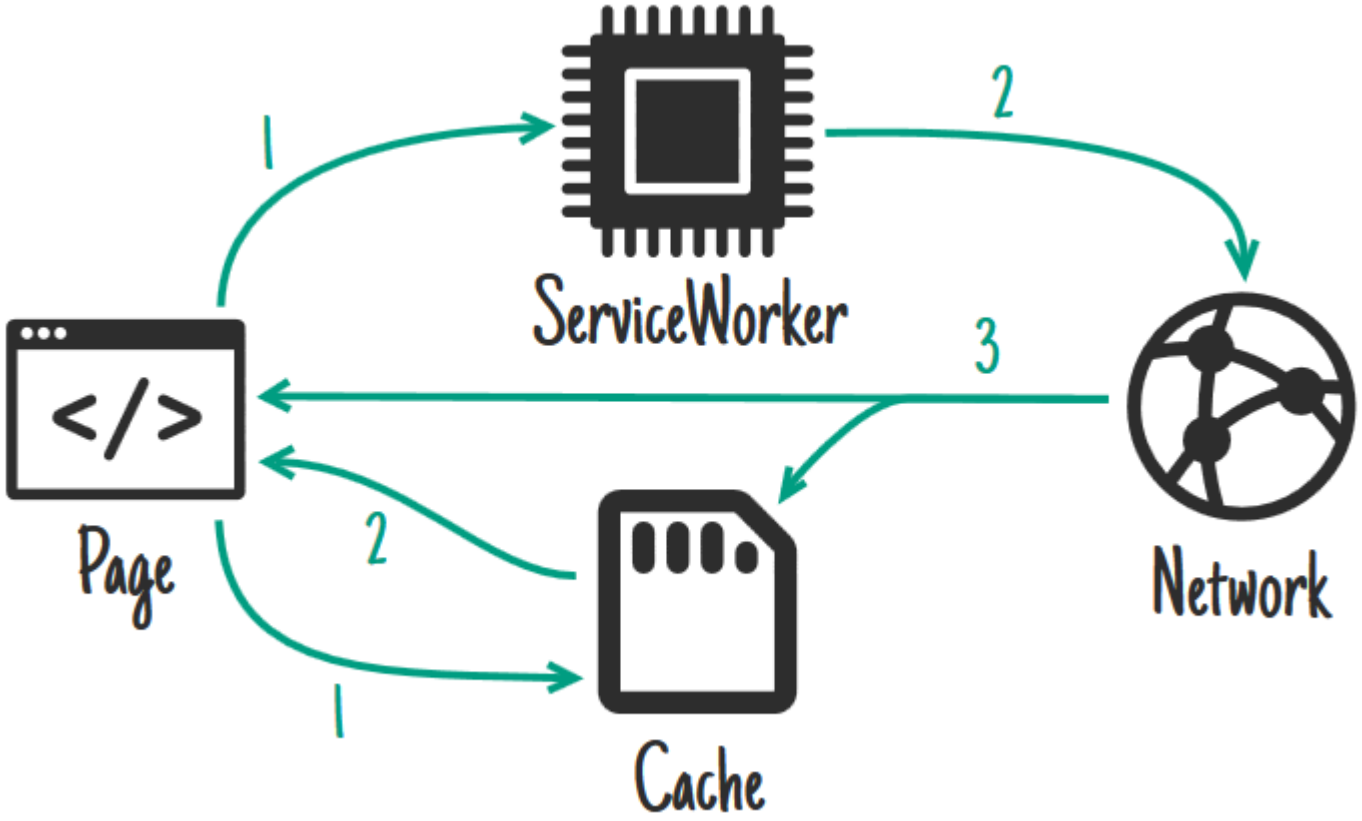


[Archibald, J.: The offline cookbook]

- für HTML/CSS/JS/PNG/Übersetzungs-Dateien
- Zeitstempel als Query-Parameter
  → bleibt bei Anfrage erhalten, wird aber beim Cachen entfernt

Caching Recipes [Archibald, J.: The offline cookbook]

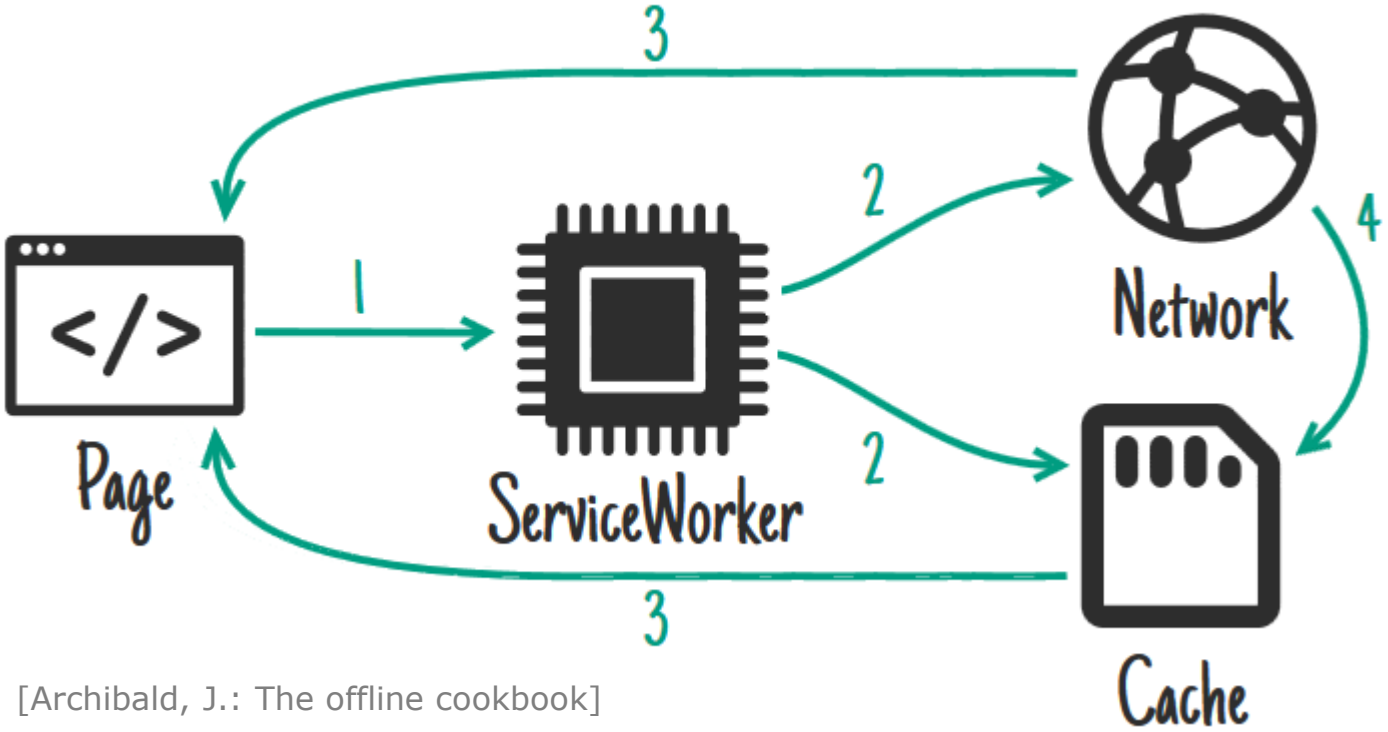
Cache, Update und Auffrischen



[Archibald, J.: The offline cookbook]

Caching Recipes [Archibald, J.: The offline cookbook]

Cache-Netzwerk-Rennen mit Update und Aktualisierung
(Cache & Network Race with Update and Refresh)



[Archibald, J.: The offline cookbook]

- Cache kann in bestimmten Situationen langsamer sein, als Netzwerk
- bietet dem Nutzer das jeweils schneller verfügbare Ergebnis

Push-Benachrichtigungen – Firebase

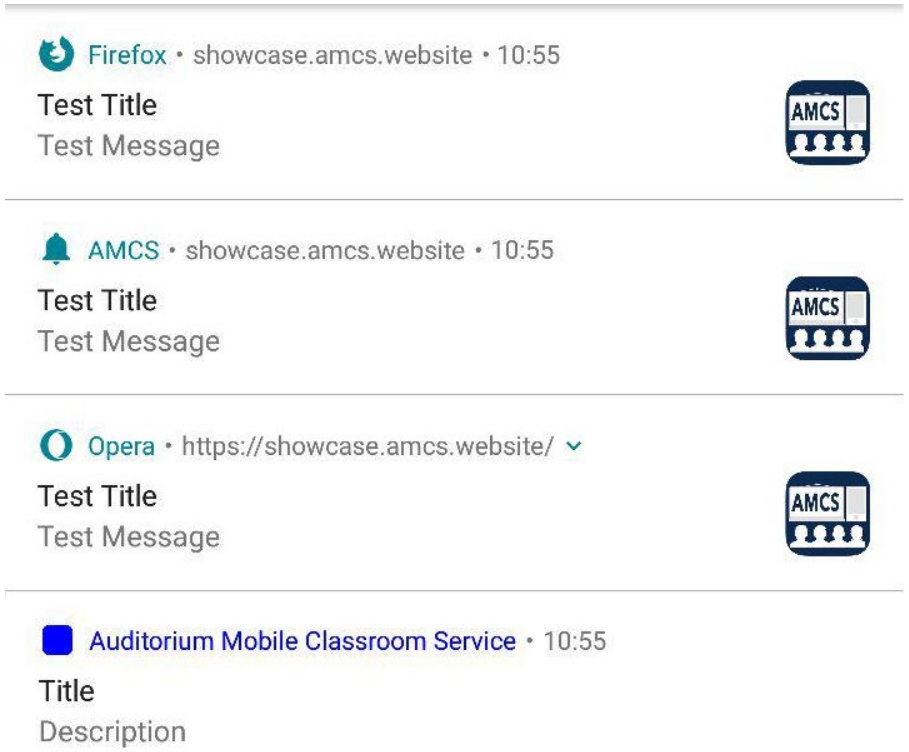
Firebase ist ein mobiler Push-Benachrichtigungsdienst von Google.

Vorgehen

- registrieren der PWA bei Firebase unter API-Zugriffskennung → Token
- PWA sendet Firebase-Token an Backend-Server der App
- registrieren des Tokens für Push-Ereignisse im ServiceWorker
- Nachrichten werden vom Backend via Firebase auf das Gerät gepusht
- ServiceWorker fängt Ereignis ab und zeigt Push-Benachrichtigung an
  → unabhängig von Aktivität der PWA (`active`, `terminated`, `background`)
- funktioniert in
  - Firefox Mobile 48+
  - Chrome for Android 42+
  - Opera mobile 37+

Push-Benachrichtigungen – Firebase

- Firefox und Opera verhalten sich anders als Chrome
- Mini-Icon wird bei PWA nicht verändert → Browser-Icon
- unterschiedliche Eigenschaften abhängig davon ob PWA im Vordergrund oder Hintergrund
- Funktionalität in Abhängigkeit vom App-Status nativ-ähnlich oder Browser-gekapselt
- in Desktop-Browsern sind Benachrichtigungen nur möglich, wenn der entsprechende Tab geöffnet ist



Hintergrundsynchronisierung

- Anfordern einer Hintergrundsynchronisierung zum asynchronen Senden
- Zwischenspeichern von Anfragen im LocalStorage mit asynchronem Zugriff (idbKeyVal)
- Abgreifen von Anfragen im Synchronisierungsereignis
  - bei Erfolg: Benachrichtigen des Views
  - bei Fehler: Synchronisierung wird neu angesetzt

```
if ('serviceWorker' in navigator && 'SyncManager' in window) {
  navigator.serviceWorker.ready.then(function(reg) {
    return reg.sync.register('tag-name');
  })
  catch(function() {
    // Registrieren für Synchronisierung fehlgeschlagen
    // (bspw. wegen betriebssystemseitiger Begrenzungen)
    postDataFromThePage();
  });
}
else {
  // ServiceWorker oder Synchronisierung werden nicht unterstützt
  postDataFromThePage();
}
```

Hintergrundsynchronisierung

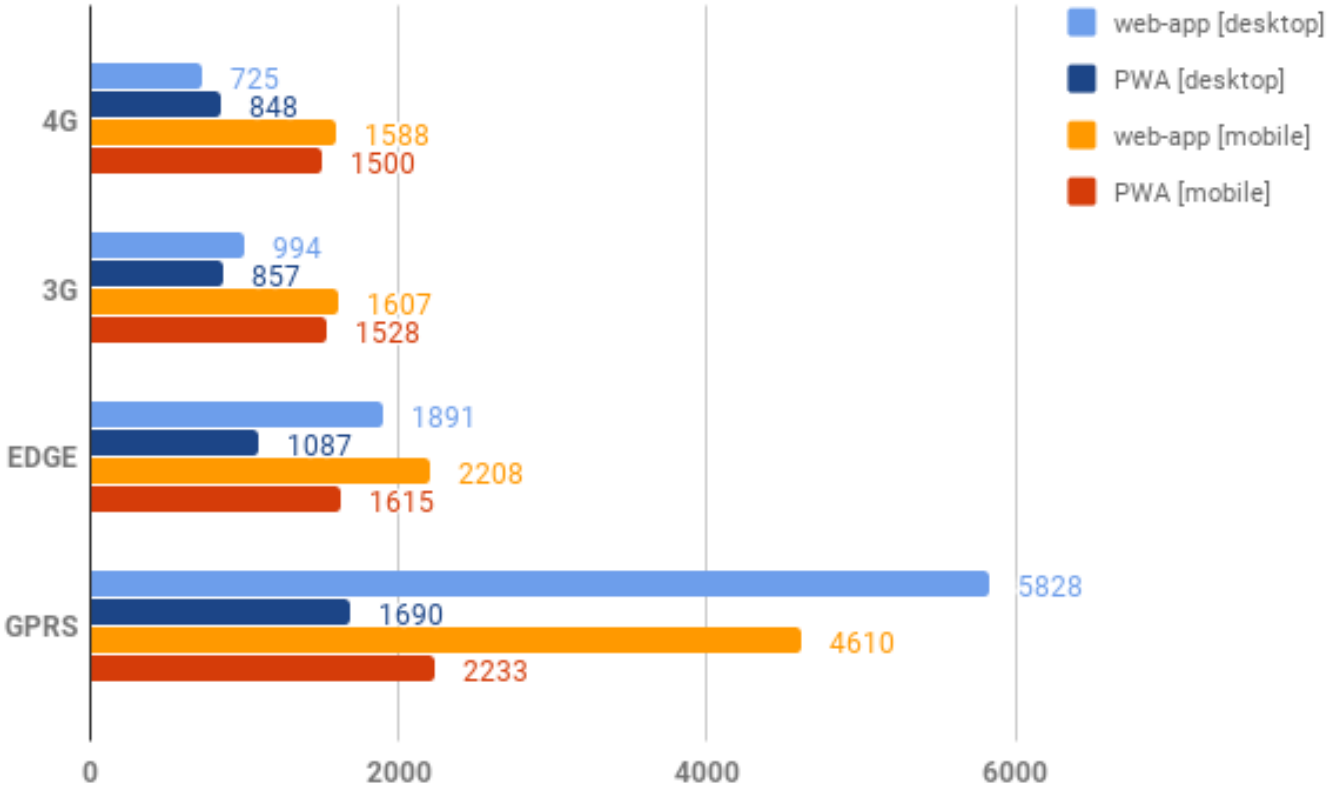
- derzeit nur von Chrome/Chromium offiziell unterstützt
  (Opera & Firefox: RC-Feature)
- Synchronisierung wird neu angesetzt wenn
  - Netzwerkverbindung neu aufgebaut wird
  - der Browser sie einplant

  → Kodezuschlag zur Garantie einer progressiven Verbesserung
- Entkopplung von Anfrage und View

  → Kodezuschlag wegen zusätzlichem asynchronen Speicher und Nachrichten-Listener

Performanz

Zeit bis letztes Element geladen und angezeigt wurde [AMCS]

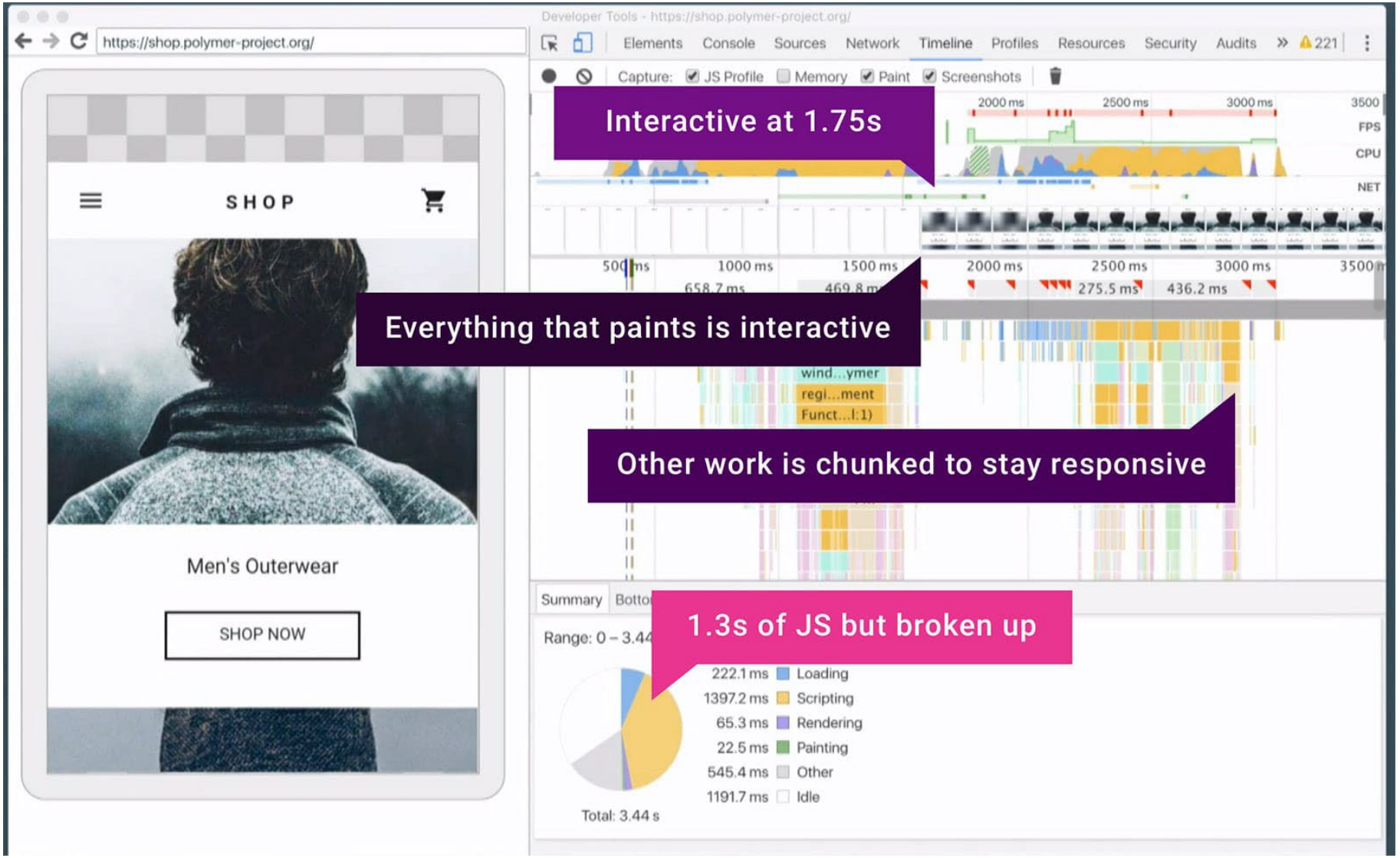


PRPL Pattern

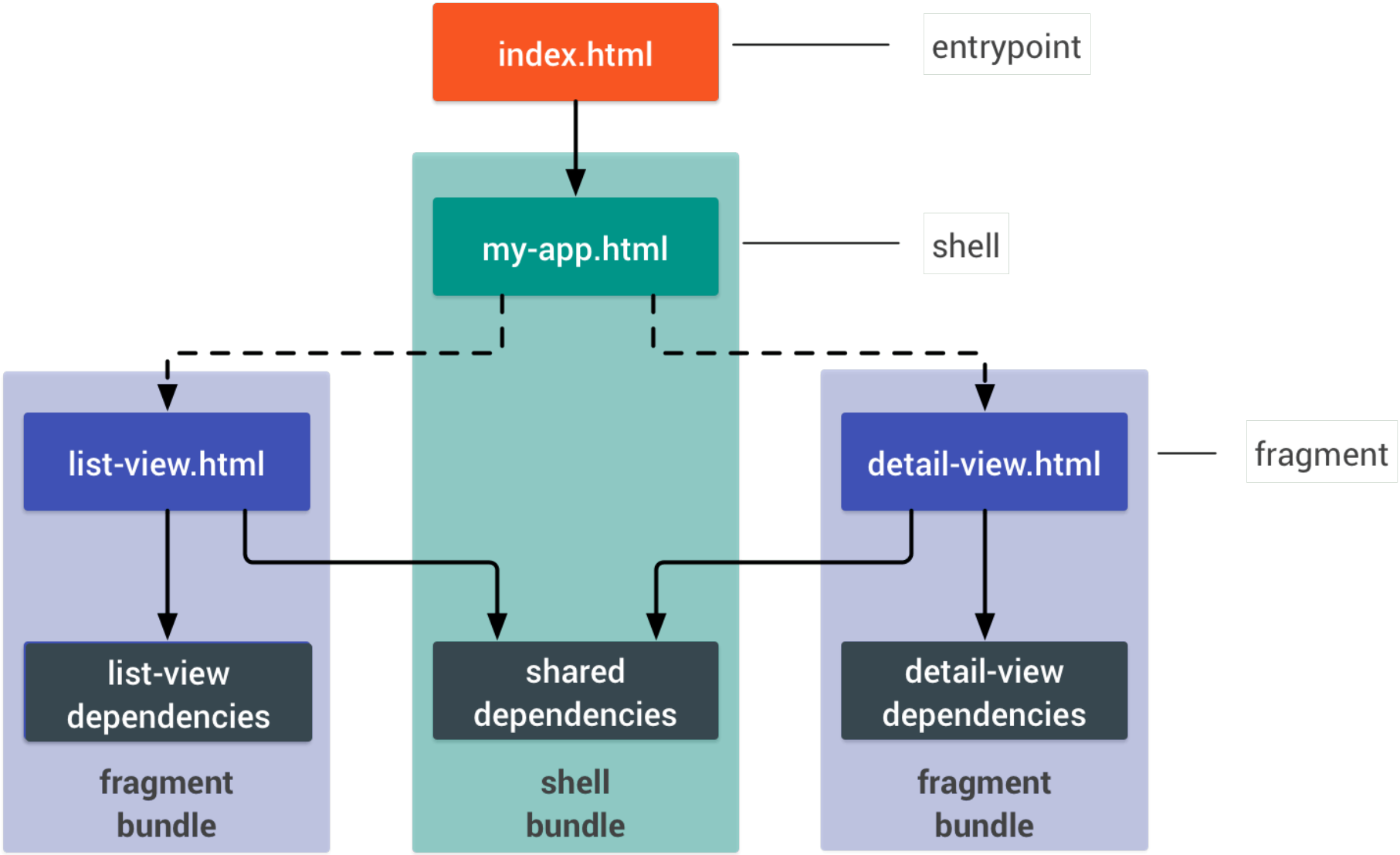
- Push – kritische Ressourcen der initialen URL Route
- Render – initiale Route
- Pre-Cache – alle weiteren verfügbaren Routen
- Lazy-Load – Erstelle übrige Routen bei Bedarf

- Pattern zur Strukturierung und Bereitstellung von PWAs
- Schwerpunkt auf Performanz der Bereitstellung und des App-Start
  - minimale Zeit bis zur Interaktivität (Minimum Time-to-Interactive)
    - insbesondere beim Erststart (unabhängig vom Einstiegspunkt)
    - insbesondere auf echten Mobilgeräten
  - maximale Puffereffizienz
  - Einfachheit der Entwicklung und Bereitstellung

PRPL-Beispiel



PRPL – App-Struktur



Zusammenfassung

Zusammenfassung

- PWA als weitere Möglichkeit, plattformübergreifend Dienste anzubieten
- Offizieller Stand: derzeit sehr experimentell
- 97% der Android-Geräte unterstützen ServiceWorker
  [Stand: 03/2024; https://caniuse.com/#feat=serviceworkers]
- PWA-Unterstützung hängt von ServiceWorker-Unterstützung und installiertem Browser ab
- Nutzern bleibt Speicherverbrauch, etc. der PWA verborgen
- Netzwerknutzung wird durch PWA mit Cache-Strategien (Recipes) signifikant reduziert
- nur unter Android und macOS (ab Sonoma) werden Push-Benachrichtigungen offiziell unterstützt
  → wirken ähnlich zu denen nativer Apps

Referenzen

Russel, A.: Progressive Web Apps: Escaping Tabs Without Losing OurSoul; https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/

Jake Archibald: The offline cookbook; https://jakearchibald.com/2014/offline-cookbook/

https://thenounproject.com/term/sd-card/6185/

https://thenounproject.com/term/cpu/72043/

https://thenounproject.com/term/code/17547/

https://thenounproject.com/term/network/12676/

Referenzen

MDN web docs: SyncManager; https://developer.mozilla.org/en-US/docs/Web/API/SyncManager

Gaunt, M.: Service Workers: an Introduction | Web | Google Developers; https://developers.google.com/web/fundamentals/getting-started/primers/service-workers

Can I use... Support tables for HTML5, CSS3, etc; https://caniuse.com/