Vorlesungsreihe
Entwicklung webbasierter Anwendungen

Progressive Web Apps (PWA)

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

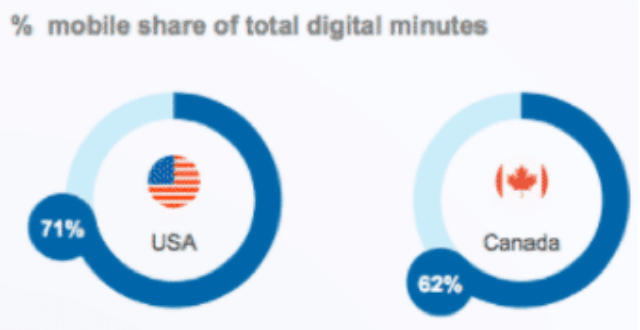
Gliederung

- Einführung **Progressive Web Apps**
  - Aktuelle Situation bei der Anwendungsentwicklung für mobile Geräte
  - Motivation von Google zu PWA
  - Kurze Historie
- PWA-Eigenschaften und Umsetzungsbeispiele
  - Definition laut Google
  - Service-Worker –Beispiel
  - Installation von PW-Apps und Test

Aktuelle Situation auf dem App-Markt

Dominanz der mobilen Nutzung

- In den letzten Jahren hat die mobile Nutzung des Internets bis zu 75% der Online-Zeit erreicht
- im Bereich eCommerce ist der Anteil noch höher
- die Tendenz wird anhalten



https://jmango360.com/wiki/mobile-app-vs-mobile-website-statistics/

Treibende Faktoren

- Viele Anwender verwenden tw. nur noch mobile Hardware
- Desktop-PC-Verkäufe rückläufig
- Verknüpfung von Suchmaschinen und Services (Maps) mit Online – Online-Angeboten führt zu höheren Verkaufszahlen
- bessere Kundenbindung durch ständige Verfügbarkeit der Angebote

Aktuelle Optionen zur Umsetzung mobiler Applikationen

Systemspezifische (Native-) Apps

- **jeweils eigenständige Entwicklung für die BS (Android / iOS (/Win))**
- **Vorteile: beste Leistung und Ausnutzung der Hardware, beste Usability**
- **Nachteile: mehrfache Entwicklung (teuer / zeitaufwändig/ ggf. leicht unterschiedliche Bedienkonzepte oder Spezialoptionen)**
- **tw. restriktive Bedingungen der Appstores (und beachtlich Margen der Anbieter)**

Cross-Plattform Native-Apps

- **Verwendung von Frameworks mit einer einmaligen Entwicklung und dann Portierung auf die verschiedenen Zielsysteme**
- **Bsp.: Xamarin: .Net-basierte Entwicklung (kam von Mono und 2016 wieder von Microsoft aufgekauft) – Framework stellt Wrapper für .NET-Funktionen für die anderen OS bereit**

Hybrid Apps

- **Nach aussen hin wie eine App im jeweiligen System**
- **Intern als Webapp mit HTML/JS/CSS realisiert**
- **Vertrieb über Appstores möglich (bei Apple tw. kritisch)**
- **Bsp.: Phonegap (von Adobe) (**Apache Cordova als Opensource-Version) - https://www.phonegap.com/ **https://de.wikipedia.org/wiki/Adobe_PhoneGap**

Neuer Ansatz mit Progressive Web Apps (PWA)

Grundidee:

- **direkte Entwicklung von Webapps für mobile Geräte mit einem möglichst App-ähnlichen Verhalten (app-like)**
- **=> Übernahme aller Vorteile von native Apps und Vermeidung der Nachteile (mehrfacher Entwicklungsaufwand) – siehe nächste Seite**

Historie

- **Erste Erwähnung des Begriffes durch Google-Entwickler in 2015**
  ➔ https://www.youtube.com/watch?v=MyQ8mtR9WxI
- **seit 2017 stark zunehmende Verwendung in der Community**
- **bereits sehr gute Lauffähigkeit auf Android-Geräten und PCs**

Noch offene Fragen / aktuelle Probleme

- **PWA´s sind auf Apple-Geräte sind leider nur bedingt einsetzbar**
- **Apple hat aber Anfang 2018 wichtige PWA-Komponenten (speziell Service-Worker) neu bereitgestellt, es gibt aber noch Detailprobleme https://t3n.de/news/progressive-web-apps-ios-safari-942769/**
- **Vermutung: PWA läuft dem App-Store-Geschäft entgegen …**

Allgemeiner Nachteil gegenüber native App: kein Copyright-Schutz infolge offenen Sourcecodes …

Progressive Web Apps (PWA) - Definition

Gekürzt nach Google-Def.:
https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/

- **Progressive** - Works for every user, regardless of browser choice because it's built with progressive enhancement as a core tenet.
- **Responsive** - Fits any form factor: desktop, mobile, tablet, or whatever is next.
- **Connectivity independent** - Enhanced with service workers to work offline or on low-quality networks.
- **App-like** - Feels like an app, because the app shell model separates the application functionality from application content .
- **Fresh** - Always up-to-date thanks to the service worker update process.
- **Safe** - Served via HTTPS to prevent snooping and to ensure content hasn't been tampered with.
- **Discoverable** - Is identifiable as an "application" thanks to W3C manifest and service worker registration scope, allowing search engines to find it.
- **Re-engageable** - Makes re-engagement easy through features like push notifications.
- **Installable** - Allows users to add apps they find most useful to their home screen without the hassle of an app store.
- **Linkable** - share the application via URL, does not require complex installation.

Entwicklung von Progressive Web Apps (PWA)

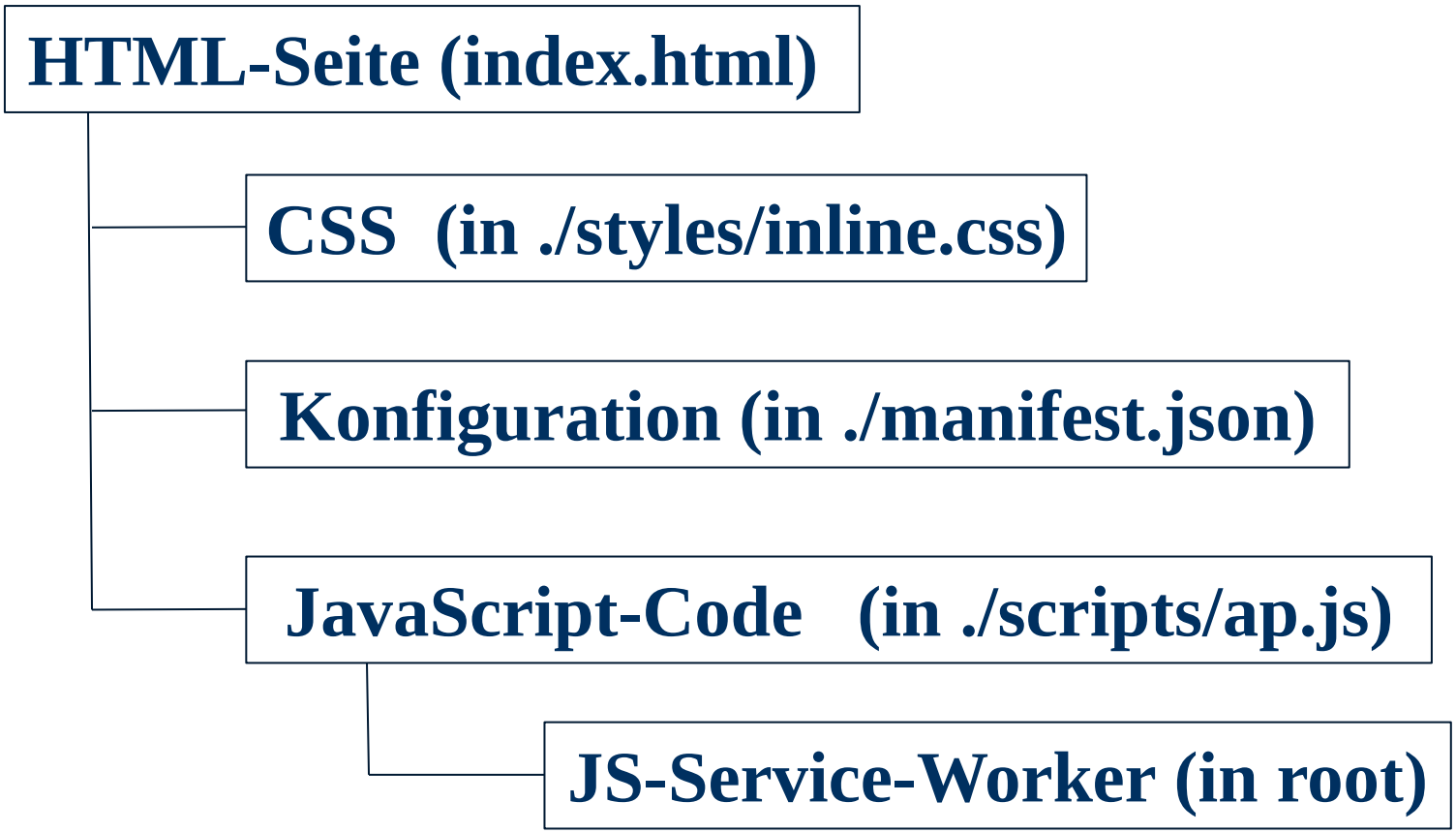
Grundansatz:

- **Normale Entwicklung mit HTML5 / JavaScript und CSS unter Beachtung der Anforderungen von Responsive und Progressive Design**

- **Zusätzliche (neue) Entwicklungsaufgaben**
  - Möglichst **vollständiges Caching** der Applikation und der Daten im Localstorage des Browsers (vgl. HTML5) durch im Hintergrund laufende **Service-Worker**
  - ständige **Prüfung auf neue Daten** (ggf. auch mit Pushkonzepten) und Bereitstellung der Daten beim Starten der Anwendung aus dem Cache (schnellere Sichtbarkeit / oder Offline-Fall) und danach Aktualisierung über Netz

- **Details in der Demo-PWA-App von Google**
  https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/

Basisarchitektur



PWA – Installation von Service-Workern

Registrierung eines Service-Workers aus externem service-worker.js

```
//  Install service  worker in main – Javascript- routine
if ('serviceWorker' in navigator)
 {    navigator.serviceWorker
                    .register('./service-worker.js')
                    .then(function()
                         { console.log('Service Worker Registered'); });
 }
```

PWA – Caching mit Service-Workern

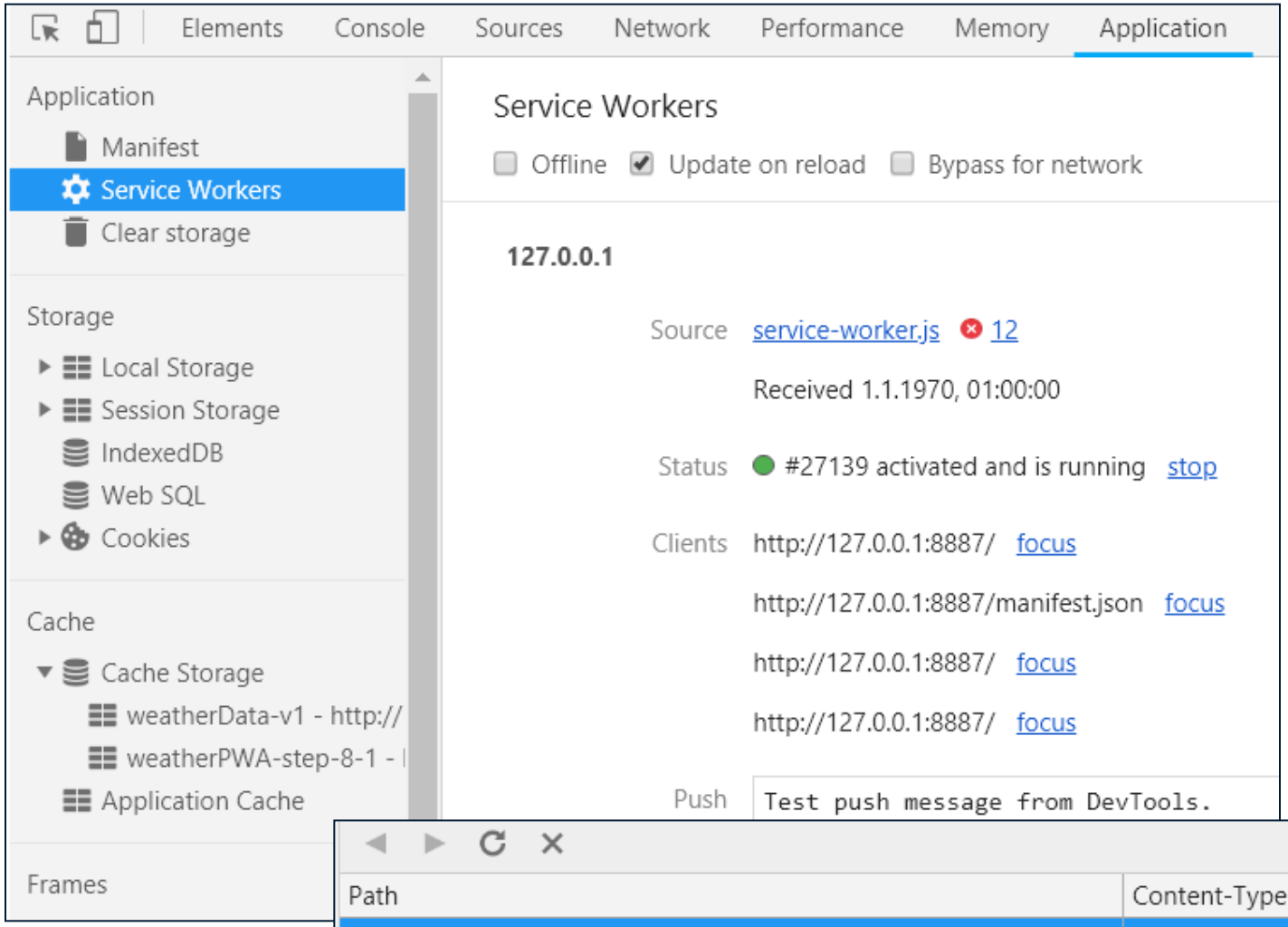
```
var dataCacheName = 'weatherData-v1';       // Eindeutige Kennung für
var cacheName = 'weatherPWA-step-6-5';      // Cachedaten
// Liste von zu chachenden Dateien der App
var filesToCache = [   '/',   '/index.html',   '/scripts/app.js',
  '/styles/inline.css',   '/images/clear.png', '/images/thunderstorm.png',
  '/images/wind.png'
];
//  beim Installieren werden App-Files aus Liste gecacht …

self.addEventListener('install', function(e) {
  console.log('[ServiceWorker] Install');
  e.waitUntil(
    caches.open(cacheName).then(function(cache) {
      console.log('[ServiceWorker] Caching app shell');
      return cache.addAll(filesToCache);
    })    );
});
```

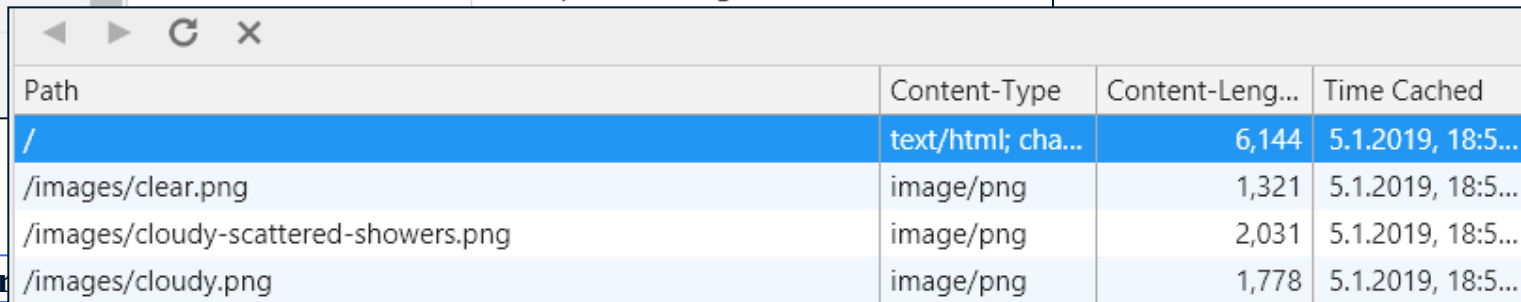
PWA – Abruf der gechachten Daten

```
// Abruf über Direkt-Netzwerk-Abruf oder Cache (falls Offline)
self.addEventListener('fetch', function(e) {
  console.log('[Service Worker] Fetch', e.request.url);
  var dataUrl = 'https://query.yahooapis.com/v1/public/yql';
  if (e.request.url.indexOf(dataUrl) > -1) {
    /* falls Data-URL dann „Network and chache data “  */
    e.respondWith(  caches.open(dataCacheName).then(function(cache) {
        return fetch(e.request).then(function(response){  // Cache data !
          cache.put(e.request.url, response.clone());
          return response;
        });       })     );
  } else {     /*  Get app shell files from cache  */
    e.respondWith(
      caches.match(e.request).then(function(response) {
        return response || fetch(e.request);
      })
    );
```

PWA – Tools zum Check der Service-Worker

// über Dev-Tools -> Application können die Serviceworker-Stati und die Cache-Zustände überprüft werden

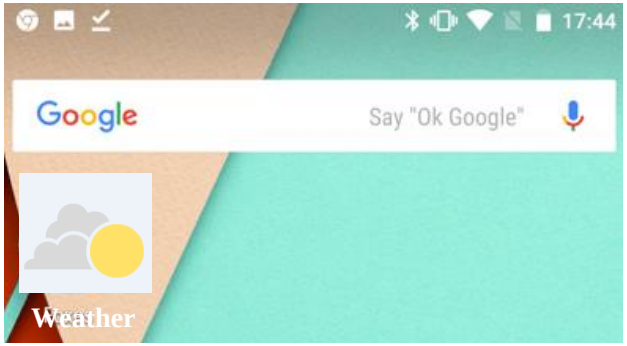




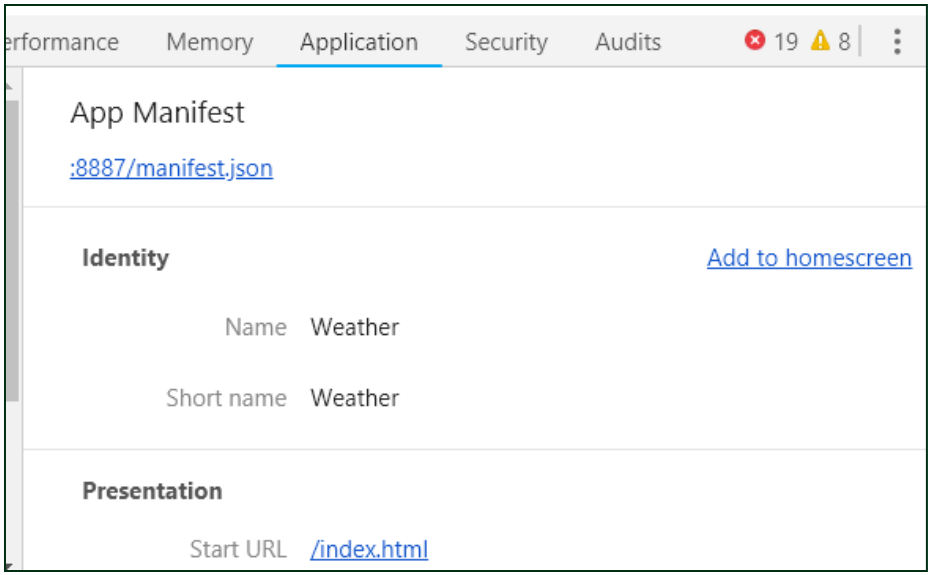
PWA – Installation der Anwendungslinks

Bereitstellung eines Add-to Homescreen - Buttons

-> Einbindung der App-Definition in manifest.json




```
{   "name": "Weather",
  "short_name": "Weather",
  "icons": [{
    "src": "images/icons/icon-128x128.png",
      "sizes": "128x128",
      "type": "image/png"
    }],
  "start_url": "/index.html",
  "display": "standalone",
  "background_color": "#3E4EB8",
  "theme_color": "#2F3BA2"
}
```

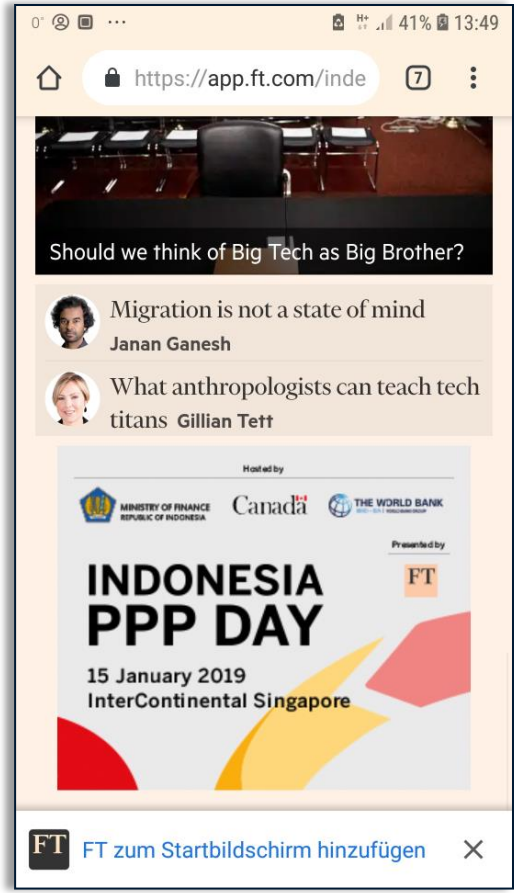


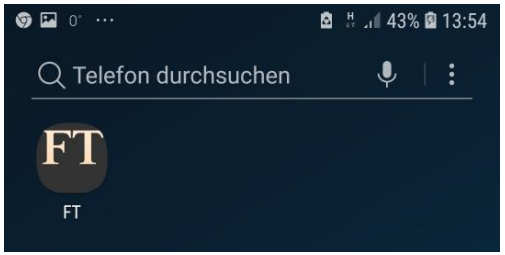
PWA – Referenzen

Verschiedene Webseiten bieten Listen von PWA-Apps (oder Demos)

=> https://pwa.rocks/

- eine der ersten PWA-Apps wurde durch die Financial-Times erstellt - https://app.ft.com





Zusammenfassung zu PWA

Progressive Web Apps

eignen sich laut der Definitionsliste von Google (vgl. Folie 6)

- als **leichtgewichtige Universal-Apps** für alle OS (tw . noch mit Einschränkungen bei iOS)

wenn

- die funktionalen Anforderungen mit HTML/JavaScript beherrschbar sind,
- die UI weborientiert realisierbar ist (also eher weniger für komplexe Spiele oder umfangreiche Frontends)

Vorteile gegenüber Native-Apps sind dann eine

- eine **nur einmalige (meist einfachere) Entwicklung**
- damit auch schnelle Innovationszyklen.

Kritisch sind noch Detailunterschiede der Browser (speziell bei Apple-Geräten) und der fehlende Copyright-Schutz.