PROGRESSIVE WEB APPLICATIONS

INHALT

back to readme

GETTING-STARTED

IDEA OF #NATIVE-APPS

WAS IST EINE NATIVE APP

- deutsch: systemeigene App
- zum Beispiel eine Mobile App, Desktop App, Windows App
- Gegenteil von einer plattformunabhängigen App (Web- / Hybrid- / Cross-App)

IDEA OF #PROGRESSIVE-ENHANCEMENT

WAS IST PROGRESSIVE VERBESSERUNG

- Ein Prinzip, nach welchem die Websites entwickelt werden

WORIN BESTEHT DIESES PRINZIP

- mit jeder Internetverbindung, aber je besser die Verbindung, desto besser die Funktionalität
- mit jedem Webbrowser, aber je fortgeschrittener der Browser, desto besser die Funktionalität
- mit jedem Gerät, aber je stärker das Gerät, desto besser die Funktionalität
- mit und ohne JS wird ein Inhalt angezeigt

IDEA OF #PWA

WAS IST PWA

- Progressive Web App
- PWA = Progressive Enhancement + Web App
- eine Website mit Merkmalen einer nativen App

WOZU IST PWA

- um eine native Anwendung leicht aufrufbar zu machen, wie eine Webanwendung (über URL)
- um eine native Anwendung betriebsunabhängig zu machen, wie eine Webanwendung
- um eine Webanwendung auch offline nutzbar zu machen, wie eine native Anwendung
- um eine Webanwendung installierbar zu machen, wie eine native Anwendung

---

PWA EXAMPLES

- [https://app.starbucks.com/]
- [https://twitter.com]
- [https://www.pinterest.de/]
- [https://www.trivago.de/]
- [https://angular.io]
- [(spotify desktop)]
- [(visual studio code)]

Listen von PWAs

- [https://findpwa.com/about]
- [https://appsco.pe/]
- [https://pwa.bar/]

PRIMARY TOOLS

Programmierwerkzeuge

- HTML, CSS, JS sind Grund-Technologien

Entwicklungsumgebung nach Wahl

- hier im Kurs: VSCode

Ein Web-Server (Local Server)

- z.B. 'Web Server for Chrome' (PWA & Browser-Erweiterung) dieses Tool in Google Webstore
  - PWA-Teil vom Tool unter chrome://apps
  - Erweiterungsteil vom Tool unter chrome://extensions
- oder 'Live Server' (VSCode Erweiterung & Browser-Erweiterung) dieses Tool auf GitHub

Entwickler-Werkzeuge vom Browser PWA Tools in Chrome

- DevTools > Lighthouse > PWA Abschnitt
- DevTools > Network

DEV WORKFLOW

1. Schreiben vom Manifest
2. Entwicklung von App-Shell (Anwendungshülle)
3. Entwicklung vom ServiceWorker

optional:

1. Umsetzen von Push-Notifications

an HTTPS entweder beim Erstellen oder beim Veröffentlichen der App denken (HTTP ist nur mit localhost möglich)

---

back to readme

FETCH API

#FETCH > IDEA

WOZU IST FETCH API

- um Netzwerk-Anfragen zu erstellen
- um die veraltete XMLHttpRequest-Technologie in den meinsten Fällen zu ersetzen

```
fetch('https://jsonplaceholder.typicode.com/todos/1')
  .then(response => response.json())
  .then(json => console.log(json))
```

FETCH > LINKS

DOKUMENTATION

- bei MDN
- bei MDN, Anleitung
- bei GDevelopers

TOOLS

- Postman (standalone App)
- RESTClient (VSCode Erweiterung von Huachao Mao) - gleich wie Postman, aber direkt in VSCode

SPEZIFIKATION

- von WHATWG, Living Standard

FETCH > JS

Konzept von Fetch ist bei JS durch die Funktion `fetch()` realisiert, die im globalen Scope aufgerufen werden kann.

Folgende Interfaces / Objekte gehören dazu:

- Request MDN docs
- Response MDN docs, MDN docs
- Headers MDN docs
- Body MDN docs

FETCH > POLYFILLS

Bei den älteren Browsern, die die Fetch-API nicht unterstützen, können Fetch-Polyfills genutzt werden

- https://www.npmjs.com/package/cross-fetch
- https://redux.js.org/advanced/async-actions#note-on-fetch
- We use fetch API in the examples. It is a new API for making network requests that replaces XMLHttpRequest for most common needs. Because most browsers don't yet support it natively, we suggest that you use cross-fetch library
- Internally, it uses whatwg-fetch polyfill on the client, and node-fetch on the server, so you won't need to change API calls if you change your app to be universal.
- Be aware that any fetch polyfill assumes a Promise polyfill is already present. The easiest way to ensure you have a Promise polyfill is to enable Babel's ES6 polyfill in your entry point before any other code runs:

---

back to readme

BROWSER-STORAGE

---

die ersten 2 sind für PWA wichtig

BROWSER STORAGE OVERVIEW

#CACHE-STORAGE > IDEA

CACHE > LINKS

CACHE > JAVASCRIPT

IDEA OF #WEB-STORAGE

WEB STORAGE > LINKS

IDEA OF #LOCAL-STORAGE

LOCAL STORAGE > LINKS

IDEA OF #SESSION-STORAGE

SESSION STORAGE > LINKS

IDEA OF #INDEXED-DB

INDEXED DB > LINKS

IDEA OF #COOKIES

COOKIES > LINKS

BROWSER STORAGE OVERVIEW

Unter dem Begriff 'Browser Storage' werden Speichermöglichkeiten durch einen Browser gemeint:

- Cache Storage
- Local Storage
- Session Storage
- IndexedDB
- Cookies

Für PWA spielt Cache eine besondere Rolle

#CACHE-STORAGE > IDEA

WAS IST CACHE STORAGE

- Interface (Objekt) in JS

WOZU IST CACHE STORAGE

- um Master-Pfad für alle benannte Cache-Objekte zu gewähren, die erreichbar sind für
    - ein ServiceWorker
    - ein anderer Worker
    - window scope (Cache Storage kann auch ohne (Service)Worker genutzt werden)
- um Mapping von String-Namen zu den entsprechenden Cache-Objekten zu verwalten

---

CACHE STORAGE > LINKS

DOKUMENTATION

- Cache Storage auf MDN

TOOLS

- Chrome DevTools > Application > Cache
    - Anleitung [https://developers.google.com/web/tools/chrome-devtools/storage/cache]
- Firefox Developer Tools > Storage (dt.: Web-Speicher) > Cache
    - Anleitung [https://developer.mozilla.org/en-US/docs/Tools/Storage_Inspector]

SPEZIFIKATION

- Das Cache-Konzept ist Teil der Service-Worker-Spezifikation (das bedeutet nicht, dass es nur zusammen mit SW genutzt werden kann)
    - [https://w3c.github.io/ServiceWorker/#cache-objects]

CACHE > JAVASCRIPT

In JavaScript ist das Cache-Konzept realisiert durch:

- die Eigenschaft 'caches' vom Objekt 'window' (einfach caches z.B. in der Console eingeben)
- Objekt 'CacheStorage'
- Objekt 'Cache'
- (Objekt 'ApplicationCache' ist veraltet)
- die Eigenschaft 'caches' liefert beim Aufruf das Objekt 'CacheStorage'
- Members:
    - delete()
    - has()
    - keys()
    - match()
    - open()
- CacheStorage ist vom Objekt 'Object' abgeleitet
- Dokumentation [https://developer.mozilla.org/en-US/docs/Web/API/CacheStorage]

- Methode 'open()' von 'CacheStorage' liefert ein Promise, welches sich zum Objekt 'Cache' auflöst
- Members von 'Cache'
    - add()
    - addAll()
    - delete()
    - keys()
    - match()
    - matchAll()
    - put()
- Cache ist vom Objekt 'Object' abgeleitet
- Dokumentation [https://developer.mozilla.org/en-US/docs/Web/API/Cache#Methods]

IDEA OF #WEB-STORAGE

WAS IST WEB STORAGE

WOZU IST WEB STORAGE

WEB STORAGE > LINKS

Local Storage und Session Storage sind Teile von Web Storage API

DOKUMENTATION

- Storage Interface
    - [https://developer.mozilla.org/en-US/docs/Web/API/Storage]
    - [https://developer.mozilla.org/de/docs/Web/API/Storage]
- Web Storage API
    - [https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API]

SPEZIFIKATION

- [https://html.spec.whatwg.org/multipage/webstorage.html]

IDEA OF #LOCAL-STORAGE

WAS IST LOCAL STORAGE

WOZU IST LOCAL STORAGE

LOCAL STORAGE > LINKS

DOKUMENTATION

- [https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage]

TOOLS

- Chrome DevTools > Application > Local Storage
  - Anleitung [https://developers.google.com/web/tools/chrome-devtools/storage/localstorage]
- Firefox Developer Tools > Storage (dt.: Web-Speicher) > Local Storage
  - Anleitung [https://developer.mozilla.org/en-US/docs/Tools/Storage_Inspector]

SPEZIFIKATION

- Local Storage API ist zusammen mit IndexedDB API und showNotifications() API in der einzelnen Spezifikation standardisiert
  - [https://storage.spec.whatwg.org/]
- 'localStorage getter' in der Web-Storage-Spezifikation
  - [https://html.spec.whatwg.org/multipage/webstorage.html]

IDEA OF #SESSION-STORAGE

WAS IST SESSION STORAGE

WOZU IST SESSION STORAGE

SESSION STORAGE > LINKS

DOKUMENTATION

- [https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage]

TOOLS

- Chrome DevTools > Application > Session Storage
  - Anleitung [https://developers.google.com/web/tools/chrome-devtools/storage/sessionstorage]
- Firefox Developer Tools > Storage (dt.: Web-Speicher) > Indexed DB
  - Anleitung [https://developer.mozilla.org/en-US/docs/Tools/Storage_Inspector]

SPEZIFIKATION

- 'sessionStorage getter' in der Web-Storage-Spezifikation
  - [https://html.spec.whatwg.org/multipage/webstorage.html]

IDEA OF #INDEXED-DB

WAS IST INDEXED DB

WOZU IST INDEXED DB

INDEXED DB > LINKS

DOKUMENTATION

- [https://developer.mozilla.org/de/docs/Web/API/WindowOrWorkerGlobalScope/indexedDB]

TOOLS

- Chrome DevTools > Application > Indexed DB
    - Anleitung [https://developers.google.com/web/tools/chrome-devtools/storage/indexeddb]
- Firefox Developer Tools > Storage (dt.: Web-Speicher) > Indexed DB
    - Anleitung [https://developer.mozilla.org/en-US/docs/Tools/Storage_Inspector]

SPEZIFIKATION

- Local Storage API ist zusammen mit IndexedDB API und showNotifications() API in der einzelnen Spezifikation standardisiert
    - [https://storage.spec.whatwg.org/]

IDEA OF #COOKIES

WAS SIND COOKIES

WOZU SIND COOKIES

COOKIES > LINKS

DOKUMENTATION

- [https://developer.mozilla.org/en-US/docs/Web/API/Document/cookie]
- [https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies]

TOOLS

- Chrome DevTools > Application > Cookies
    - Anleitung [https://developers.google.com/web/tools/chrome-devtools/storage/cookies]
- Firefox Developer Tools > Storage (dt.: Web-Speicher) > Cookies
    - Anleitung [https://developer.mozilla.org/en-US/docs/Tools/Storage_Inspector]

SPEZIFIKATION

- [https://tools.ietf.org/html/draft-west-cookie-prefixes-05]
- obsolete [https://www.w3.org/TR/DOM-Level-2-HTML/html.html#ID-8747038]

---

back to readme

CACHING STRATEGIES

OFFLINE APP

Progressive Web Apps verfolgen konsequent einen Offline-first-Ansatz

OFFLINE

- keine 404-Status-Seite (z.B. Dinosaurier-Spiel in Chrome), sondern
- eine einfache offline-Seite oder
- Interaktion mit der App ohne Änderung der gecachten Daten oder
- Interaktion mit der App mit Änderung der Daten und Synchronisierung beim nächsten Internet-Zugang

BEISPIEL

- https://www.trivago.com/offline

IDEA OF #APP-SHELL

WAS IST APP SHELL

- eine von Möglichkeiten (Architekturmodel) für PWA
- das minimale HTML, CSS und JS für das korrekte Laufen des UI
- wird den Daten / dem Inhalt gegenüber gestellt

WOZU IST APP SHELL

- um die Ladezeit der Anwendung zu reduzieren (Shell ist immer im Cache)
- um das UI von den Daten zu trennen

OFFLINE PATTERN

- cache-first network
- server-first network
- pre-cache
- offline page
- offline copy of pages
- offline copy with backup offline page
- background sync
- serving cached media

- Kombination von den erwähnten Szenarios

Hier kann man nachschauen, welche Szenarios es noch gibt und wie diese realisiert sind

- [https://developers.google.com/web/tools/workbox/guides/common-recipes]
- [https://developers.google.com/web/tools/workbox/guides/advanced-recipes]
- [https://serviceworke.rs/caching-strategies.html]
- [https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching]

BAD OFFLINE PATTERN

- caching everything
- no update mechanism for cached things

---

back to readme

SERVICE-WORKER

#SERVICE-WORKER > IDEA

WAS IST EIN SERVICE WORKER

- STRENG GESEHEN:
  - ein JS-Objekt (Interface)
  - abgeleitet von Worker
  - läuft nicht im Gültigkeitsbereich von Window, sondern im windoworworkerglobalscope
- IM ÜBERTRAGENEN SINNE:
  - eine JS-Datei, die als Service Worker beim Objekt 'navigator.serviceWorker' registriert wird
- ein Proxy zwischen der Anwendung und dem Server
- nur mit HTTPS möglich, nicht mit HTTP*
- liegt im Ordner 'public'

WOZU IST EIN SERVICE WORKER

- um die Abhängigkeit der App vom Netzwerk zu reduzieren (SW steuert die HTTPS-Requests) offline-Verfügbarkeit zu gewähren (Kontrolle über die Anwendung und start_url)
- um die Web-Anwendung zum Startbildschirm auf mobilen Geräten hinzufügen zu können
- um die Web-Anwendung auf dem Desktop-Rechner installieren zu können
- um die Push-Benachrichtigungen zu ermöglichen

SERVICE WORKER & PWA

- Es stellt für Progressive Web Apps essentielle Funktionen wie das Caching für die Offline-Verwendbarkeit bereit.
- Ein Service Worker ist ein JavaScript, das ein Web-Browser im Hintergrund ausführt.
- Service Worker werden eigens programmiert, im JavaScript der Seite registriert und installiert.
- Auch von nativen Apps bekannte Push-Benachrichtigungen sind mit Service Workern möglich.
- Service Worker bedingen HTTPS, weshalb jede Progressive Web App mit HTTPS läuft.

SERVICE WORKER > LINKS

DOKUMENTATION (MDN)

- [https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API]
- [https://developer.mozilla.org/en-US/docs/Web/API/ServiceWorker]

TOOLS

- Zum Entwickeln und Üben [https://serviceworke.rs/]
- Statistik von Websites mit Service Worker [https://www.chromestatus.com/metrics/feature/timeline/popularity/990]
- Chrome DevTools [https://developers.google.com/web/tools/chrome-devtools/progressive-web-apps#service-workers]

ANLEITUNG

- MDN (von Mozilla Foundation) [https://developer.mozilla.org/de/docs/Web/API/Service_Worker_API/Using_Service_Workers]
- GDevelopers [https://developers.google.com/web/fundamentals/primers/service-workers/]
- W3C [https://w3c.github.io/ServiceWorker/#algorithms]

SPEC

- [https://w3c.github.io/ServiceWorker]

SERVICE WORKER > JAVASCRIPT

In JavaScript ist das Service-Worker-Konzept durch folgende JS-Objekte realisiert

- ServiceWorkerContainer
- ServiceWorker
- ServiceWorkerRegistration
- InstallEvent
- ActivateEvent
- FetchEvent
- ExtendableEvent
- ExtendableMessageEvent
- ServiceWorkerGlobalScope
- Client

- Clients

SERVICE WORKER CONTAINER

'navigator.serviceWorker' liefert folgendes Objekt:

- ServiceWorkerContainer
  - controller
  - getRegistration()
  - getRegistrations()
  - oncontrollerchange
  - onmessage
  - onmessageerror
  - ready
  - register(): Promise<ServiceWorkerRegistration>
  - startMessages()

SERVICE WORKER REGISTRATION

Promise<ServiceWorkerRegistration>

Ableitung: Object > EventTarget > ServiceWorkerRegistration

Members:

- active
- backgroundFetch
- getNotifications()
- installing
- navigationPreload
- onupdatefound
- paymentManager
- periodicSync
- pushManager
- scope
- showNotification()
- sync
- unregister()
- update()
- updateViaCache
- waiting

SERVICE WORKER > SCOPE

- Gültigkeitsbereich vom Service Worker - welche Dateien werden über ihn kontrolliert
- Root Ordner oder Unterordner im Projekt der Anwendung
- Default-Gültigkeitsbereich ist der Ort, wo die SW-Datei liegt

- Alle Unterordner sind auch in diesem Gültigkeitsbereich
- Wenn die Datei 'sw.js' im Root-Ordner liegt, dann kontrolliert der Service Worker die Anfragen (Requests) von allen Webseiten dieser Domäne
- Scope vom SW soll auch Scope vom Manifest und Start URL beinhalten

---

SERVICE WORKER > LIFECYCLE

- Unter Lebenszyklus werden die Ereignisse gemeint, auf die ein Service Worker zuhört:
    - install
    - activate
    - fetch
- Durch ServiceWorkerRegistration-Objekt kann man auch die Zustände vom ServiceWorker abfragen
- Anleitung von GDevelopers
    - [https://developers.google.com/web/fundamentals/primers/service-workers/lifecycle]

SERVICE WORKER > FETCH EVENT

```
self.addEventListener('fetch', evt => {
    // Fetch-Event abfangen
    console.log('SW: Fetch, event.request: ', evt.request);
```

SERVICE WORKER > FETCH REQUEST

fetch event > request

```
bodyUsed: false
cache: "default"
credentials: "omit"
destination: "empty"
headers: Headers {}
integrity: ""
isHistoryNavigation: false
keepalive: false
method: "GET"
mode: "cors"
redirect: "follow"
referrer: "http://127.0.0.1:8887/"
referrerPolicy: "no-referrer-when-downgrade"
signal: AbortSignal {aborted: false, onabort: null}
url: "http://127.0.0.1:8887/manifest.json"
```

```
__proto__: Request
arrayBuffer: ƒ arrayBuffer()
blob: ƒ blob()
bodyUsed: (...)
cache: (...)
clone: ƒ clone()
credentials: (...)
destination: (...)
formData: ƒ formData()
headers: (...)
integrity: (...)
isHistoryNavigation: (...)
json: ƒ json()
keepalive: (...)
method: (...)
mode: (...)
redirect: (...)
referrer: (...)
referrerPolicy: (...)
signal: (...)
text: ƒ text()
url: (...)
```

Abgeleitet von Object

SERVICE WORKER > FIRST INITIALIZATION

- in der separaten Service-Worker-Datei (z.B. sw.js): -- self.skipWaiting() -- self.clients.claim()

-- Verwendet man in dem Handler für Install-Event die Methode self.skipWaiting() und im Handler für Activate-Event self.clients.claim(), übernimmt der Service Worker sofort die Kontrolle -- Wenn man einen Service Worker schon früher aktiviert hatte, so kann man auf diese Weise auch die Aktivierung von einem neueren Service Worker durchführen

- Durch 'self' wird das ServiceWorkerGlobalScope-Objekt angesprochen

---

SERVICE WORKER GLOBAL SCOPE

Durch 'self' in einem Event-Handler in der Datei 'sw.js' wird das ServiceWorkerGlobalScope-Objekt angesprochen

Members:

- clients
- registration
- serviceWorker
- skipWaiting()
- oninstall
- onactivate

- onfetch
- onmessage
- onmessageerror

SERVICE WORKER > TOOLS

Zahlreiche Frameworks, wie z. B. Angular mit dem Mobile Toolkit, stellen Service Worker bereit, so dass man diese nicht selbst entwickeln muss

Developing: s. PWA.pptx > FRAMEWORKS > WORKBOX [https://www.pwabuilder.com/serviceworker] [https://serviceworke.rs/]

Managing:

- Chrome DevTools > Application > Service Workers*

---

back to readme

WEB-APP-MANIFEST

#WEB-APP-MANIFEST > IDEA

WAS IST WEB APP MANIFEST

- eine JSON-Datei
- normalerweise im Ordner 'public' (Dateien, die an Client gleich geschickt werden)
- verlinkt in allen html-Dateien

WOZU IST WEB APP MANIFEST

- ohne PWA, um Metadaten der Website, wie Name Beschreibung etc. zu sammeln
- bei PWA bietet Manifest zusammen mit dem Service Worker die Offline-Funktionalität an
- um zu kontrollieren, wie die Anwendung für den Benutzer erscheint
- damit die Anwendung den Status 200 auch offline schickt
- um thematisiertes Splash-Screen (Begrüßungsschirm) einzubauen
- um die Adresszeile vom Browser zu verstecken (Theme-Color Prop)

MANIFEST > LINKS

DOKUMENTATION

- MDN [https://developer.mozilla.org/en-US/docs/Web/Manifest#Members]
- GDevelopers (WebDev) [https://web.dev/add-manifest/]

TOOLS

- Generator [https://app-manifest.firebaseapp.com/]
- Visualisierung mit weiterführenden Links
    - Chrome DevTools > Application > Manifest [https://developers.google.com/web/tools/chrome-devtools/progressive-web-apps#manifest]

SPEZIFIKATION

- W3C Editor's Draft [https://w3c.github.io/manifest/]
- W3C Working Draft [https://www.w3.org/TR/appmanifest/]

MANIFEST > MEMBERS

Auswahl

- display
  - mit dem Wert 'standalone' wird die App in einem separaten Browser-Window geöffnet
- orientation
  - soll die App im Portrait- oder Landscape-Modus geöffnet werden
- start_url
  - welche Seite soll geöffnet werden, wenn die App startet zum ersten Mal
- icons zusammen mit short_name
  - wie soll der verlinkte Icon im Dock (Desktop) oder im App Launcher (Smartphone) aussehen und welcher Name soll unter dem Icon stehen
- name, icons, colors
  - definieren Splash Screen

viel mehr [https://developer.mozilla.org/en-US/docs/Web/Manifest#Members]

MANIFEST > ICONS

- Damit die Anwendung installierbar ist, braucht Chrome die Icons mindestens in den Größen 192x192px und 512x512px. Man kann auch andere Größen hinzufügen
- Lädt man Bilder in einer anderen Größe hoch als bei "sizes" steht, sieht man in DevTools>Application>Manifest eine Warn-Meldung

```
"icons": [
  {
    "src": "/assets/icon-192x192.png",
    "sizes": "192x192",
    "type": "image/png"
  },
  {
    "src": "/assets/icon-512x512.png",
    "sizes": "512x512",
    "type": "image/png"
  }
]
```

#SPLASH-SCREEN > IDEA

WAS IST SPLASH SCREEN

- Deutsch: Begrüßungsschirm

- Das, was der Benutzer kurz sieht, bevor die Anwendung gestartet ist

WOZU IST SPLASH SCREEN

- um die Qualität der Benutzererfahrung zu erhöhen
- um die Ladezeiten (Fetch) interessant zu befüllen

Ein minimales Splash Screen wird durch die Manifest-Eigenschaften 'name', 'icons' und 'colors' definiert.

mehr

ADDRESS BAR

Mit dem Meta-Tag 'Theme-Color' kann die Adresszeile an die Stile der Anwendung angepasst werden

mehr

```
<meta name="theme-color">
```

MANIFEST & SAFARI

Auch mit dem Manifest ist die Anwendung für Safari nicht installierbar.

Meldung: 'Does not provide a valid apple-touch-icon'

- Safari auf iOS unterstützt (noch) kein Manifest prüf nach
- Die Icons, Farben und App-Name werden über traditionelle meta- und link-Tags übermittelt prüf nach
- Benötigt wird mindestens ein nicht transparentes 192px (oder 180px) quadratisches PNG
- Anleitung [https://web.dev/apple-touch-icon/?utm_source=lighthouse&utm_medium=devtools]

```
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black">
<meta name="apple-mobile-web-app-title" content="Weather PWA">
<link rel="apple-touch-icon" href="/images/icons/icon-152x152.png">
```

MANIFEST > #MASKABLE-ICON

- Die Manifest-Datei sollte auch ein Maskable-Icon enthalten
- Warnmeldung in Lighthouse: 'Manifest doesn't have a maskable icon.'
- WAS IST MASKABLE ICON
    - Ein Format von Icons
    - Maskable Icon ist PWA-Begriff für Adaptive Icon

- WOZU IST MASKABLE ICON
    - Damit alle Icons, auch mit transparentem Hintergrund, auf allen Android-Geräten wie gewünscht aussehen
    - damit das Bild den kompletten verfügbaren Raum einnimmt (kein letterboxing entsteht)
- ANLEITUNG
    - [https://web.dev/maskable-icon/]
    - [https://web.dev/maskable-icon-audit/?utm_source=lighthouse&utm_medium=devtools]
- TOOL
    - [https://maskable.app/editor]

---

back to readme

INSTALLABILITY

INSTALLABILITY > CRITERIA

- Durch den Web-App-Manifest ist eine Web-Anwendung installierbar
- Auf dem Handy kann eine Progressive Web App über "Add-To-Homescreen"-Prompt installiert werden
- Dieses Dialogfenster erscheint nur, wenn ein Event 'Beforeinstallprompt' geschossen wird
- Dieses Event wird vom Browser ausgelöst, wenn er merkt, dass die Anwendung installierbar ist
- Auf dem Rechner findet man die installierte App auf dem Desktop (Start-Bildschirm) und im Browser durch die Adresse chrome://apps/
- Bei 'Windows-Einstellungen > Apps' sind PWAs nicht aufgelistet

INSTALLABILITY > BEFOREINSTALLPROMPTEVENT

Ableitung: Object > Event > BeforeInstallPrompt

Members:

- bubbles
- cancelBubble
- cancelable
- composed
- currentTarget
- defaultPrevented
- eventPhase
- path
- **platforms**
- **prompt()**
- returnValue
- srcElement
- target
- timeStamp
- type
- **userChoise : Promise**

Fett markiert sind die Members, die speziell für dieses Objekt sind

INSTALLABILITY > APPINSTALLED EVENT

Wurde das Dialogfenster vom Benutzer am Anfang ignoriert, kann er die Anwendung später auch über das Drei-Punkte-Menü installieren.

Da wird das Event 'appinstalled' ausgelöst.

Members:

- bubbles
- cancelBubble
- cancelable
- composed
- currentTarget
- defaultPrevented
- eventPhase
- **isTrusted**
- path
- returnValue
- srcElement
- target
- timeStamp
- type: "appinstalled"

INSTALLABILITY > DISPLAY MODE

Sind für die installierten Variante der PWA besondere CSS-Stile vorgesehen, kann man das über spezielle Media-Queries abfragen

```
@media all and (display-mode: standalone) {
  body {
    background-color: yellow;
  }
}
```

INSTALLABILITY > UNINSTALL

Auf dem Rechner

- die Anwendung starten
- Drei-Punkte-Menü rechts oben
- Befehl '[App-name] deinstallieren '

---

back to readme

PUSH-NOTIFICATIONS

#PUSH-NOTIFICATIONS > IDEA

- Gewöhnliche Websites (keine PWAs) und auch native Apps setzen auch diese Technologien ein
- Es kann z.B. so aussehen [https://web-push-book.gauntface.com/demos/notification-examples/]
- Bei PWAs spielt die Kommunikation zwischen dem Server und dem Client eine besondere Rolle
- Push-Benachrichtigungen sind dabei neben dem Service Worker und Web-App-Manifest ein Hauptwerkzeug
- Hinter den Begriffen 'push notifications' oder 'web notifications' stehen zwei separate APIs: Push API und Notifications API

PUSH NOTIFICATIONS > PUSH API

DOKUMENTATION

- [https://developer.mozilla.org/de/docs/Web/API/Push_API]

TOOLS

- einen Web-Server immitieren z.B. mit 'Web Server for Chrome' (Browser-Erweiterung)
  - google webstore
- oder 'Live Server' (VSCode Erweiterung)
  - [https://ritwickdey.github.io/vscode-live-server/]
- einen Push-Server immitieren z.B. mit npm-Packet 'web-push'
  - [https://www.npmjs.com/package/web-push]

    - [https://serviceworke.rs/web-push.html]
- oder anderen Bibliotheken
    - [https://github.com/web-push-libs/]

SPEZIFIKATIONEN

- [https://www.w3.org/TR/push-api/]
- [https://datatracker.ietf.org/doc/rfc8292/?include_text=1]
- [https://tools.ietf.org/html/draft-ietf-webpush-encryption-09]
- Versionierung der Spezifikation [https://datatracker.ietf.org/wg/webpush]

PUSH NOTIFICATIONS > NOTIFICATIONS API

DOKUMENTATION [https://developer.mozilla.org/de/docs/Web/API/Notifications_API]

SPEZIFIKATION [https://notifications.spec.whatwg.org/]

PUSH NOTIFICATIONS & JS

In JavaScript ist das Konzept von Push-Benachrichtigungen durch folgende Objekte realisiert

- PushManager
- PushSubscription
- PushSubscriptionOptions

PUSH NOTIFICATIONS > PUSH MANAGER

PushManager (Ableitung: Object)

- getSubscription(): Promise`<PushSubscription>` || null
- permissionState()
- subscribe(): Promise`<PushSubscribtion>`

Objekt 'PushManager' ist abgeleitet vom Objekt 'Object'

[https://developer.mozilla.org/de/docs/Web/API/PushManager]

PUSH NOTIFICATIONS > PUSH SUBSCRIPTION

PushSubscription

- endpoint
- expirationTime
- getKey()
- options
- toJSON()
- unsubscribe()

Objekt 'PushSubscription' ist abgeleitet vom Objekt 'Object'

PUSH NOTIFICATIONS > PUSH SUBSCRIPTION OPTIONS

PushSubcriptionOptions (Ableitung: Object)

- applicationServerKey: UInt8Array
- userVisibleOnly: boolean

Objekt 'PushSubcriptionOptions' ist abgeleitet vom Objekt 'Object'

'userVisibleOnly' soll auf 'true' gesetzt werden, wenn die Benachrichtigungen jedes Mal gezeigt werden sollen, wenn eine Push-Nachricht kommt

---

#VAPID-KEYS > IDEA

WAS SIND VAPID SCHLÜSSEL

- VAPID: Voluntary Application Server Identification, de: Freiwillige Identifikation des Anwendungsservers
- ein Paar aus einem öffentlichen und einem privaten Schlüssel
- ein Schlüssel ist ein String (ähnlich wie Passwort-String)

WOZU SIND VAPID SCHLÜSSEL

- die Schlüssel werden bei Push-Benachrichtigungen gebraucht
- die Schlüssel sind nötig, damit die Anwendung auf dem Client-Gerät und das entsprechende Back-End auf einem Server sich gegenseitig erkennen

DOKUMENTATION : BLOG

- [https://blog.mozilla.org/services/2016/04/04/using-vapid-with-webpush/]

TOOLS : GENERATOREN

- [https://vapidkeys.com/]
- [https://d3v.one/vapid-key-generator/]
- oder auch als npm Packet [https://github.com/web-push-libs/web-push]
- web-push generate-vapid-keys [--json]

SPEZIFIKATION:

- [https://tools.ietf.org/html/draft-ietf-webpush-vapid-01]
- Versionierung der Spezifikation: [https://datatracker.ietf.org/doc/rfc8292/]

VAPID KEYS > APPLICATION SERVER KEY

- Private Key befindet sich auf der Server-Seite der Anwendung
- Den öffentlichen Schlüssel speichert man auf der Client-Seite der Anwendung
- Der wird gebraucht bei der Methode 'subscribe()' vom PushManager

- der öffentliche Schlüssel ist ursprünglich laut den Regeln von Base64 URL kodiert und soll in JavaScript in ein Uint8Array umgewandelt werden

```
swRegistration.pushManager.subscribe({
    userVisibleOnly: true,
    applicationServerKey: applicationServerKey
})
```

PUSH NOTIFICATIONS > LAB

live Beispiel [https://d3v.one/apps/pwa/push/]

---

back to readme

PWA-TOOLS-&-FRAMEWORKS

SECONDARY TOOLS

Werkzeug zum Hinzufügen von Offline-Funktionalitäten zu den Websites

- siehe Kapitel PWA > FRAMEWORKS
- [https://developers.google.com/web/tools/workbox]

Werkzeug zum Umwandeln einer Website in die PWA oder zum Üben

- siehe Kapitel PWA > FRAMEWORKS
- [https://www.pwabuilder.com/]

Sammlung von Werkzeugen für PWA mit TWA

- siehe Kapitel PWA > FRAMEWORKS
- [https://github.com/GoogleChromeLabs/bubblewrap]

PWA TOOLS > #WORKBOX

Werkzeug zum Hinzufügen von Offline-Funktionalitäten zu den Websites
[https://github.com/GoogleChrome/workbox]

DOKUMENTATION [https://developers.google.com/web/tools/workbox]

ANLEITUNGEN

- [https://codelabs.developers.google.com/codelabs/workbox-lab/index.html#0]
- [https://codelabs.developers.google.com/codelabs/workbox-indexeddb/index.html?index=..%2F..index#0]

PWA TOOLS > #PWA-BUILDER

Werkzeug zum Umwandeln einer Website in die PWA oder zum Üben [https://www.pwabuilder.com/]

[https://www.pwabuilder.com/serviceworker]

PWA > FRAMEWORKS > #ANGULAR & CO

Das Framework Angular setzt auch Service-Worker ein und kann als ein PWA-Framework genutzt werden [https://angular.io/]

im Fall von SPA-Frameworks haben die Service Worker:

- die gleiche Routing-Logik wie der Web-Server
- die gleiche Templates-Logik wie der Web-Server

---

back to readme

CLIENT-DEVICE-ACCESS

WHAT WEB CAN DO

Werkzeug zum Testen vom Browser

- um zu wissen, auf welche Funktionen vom Endgerät ein konkreter Browser zugreifen kann
- [https://whatwebcando.today/]

FILE ACCESS

Zugriffe auf das native Dateisystem oder das Adressbuch sind mit Progressive Web Apps erst in Testversionen wie Google Chromium möglich.

[https://www.heise.de/developer/artikel/Google-Projekt-Fugu-Die-Macht-des-Kugelfisches-4255636.html]

HARDWARE ACCESS

Hardwarezugriff PWAs bieten ebenfalls die Möglichkeit, verschiedene native Funktionen zu implementieren. Es kann auf Sensorikdaten und Kameras zugegriffen werden oder Pushnotifications erstellt werden. [https://goingmeta.io/progressive-web-app/]

Werkzeug zum Testen vom Browser um zu wissen, auf welche Funktionen vom Endgerät ein konkreter Browser zugreifen kann [https://whatwebcando.today/]

---

back to readme

FACTS

#TWA > IDEA & LINKS

WAS IST TWA

- Trusted Web Activities
- eingeführt mit der Chrome-Version 72 Anfang 2019

WOZU SIND TWA

- um Google-Play-Store unter Android als Distributionsplattform für PWAs zu nutzen

ANLEITUNG

- [https://blog.chromium.org/2019/02/introducing-trusted-web-activity-for.html]
- [https://developers.google.com/web/android/trusted-web-activity/]

TOOLS

- Sammlung von Werkzeugen für PWA mit TWA [https://github.com/GoogleChromeLabs/bubblewrap]

GOOD PWA CHECKLIST

- ... Sachen aus den früheren Abschnitten
- Plus: (These checks are required by the baseline PWA Checklist but are not automatically checked by Lighthouse. They do not affect your score but it's important that you verify them manually.)
    - die Site funktioniert in verschiedenen Browsern *
    - **
    - jede Seite (Page) hat eine URL ***
- [https://web.dev/pwa-checklist/]

ADVANTAGES OF PWA

- PWAs benötigen weniger Speicherplatz auf dem Endgerät

- um nicht mehrere Versionen einer Anwendung für verschiedene Plattformen entwickeln zu müssen
- Apps werden erst von den Web Stores validiert und freigegeben. PWAs sind sofort einsatzbereit.
- Aufgerufene PWAs gewährleisten, dass jeder Nutzer die aktuellste Version verwendet. Bei einer App können Nutzer die Aktualisierung verzögern.

DISADVANTAGES OF PWA

- PWAs haben limitierten Zugriff auf die Funktionen eines Endgeräts.
- Sollen PWAs mehrere mobile Browser unterstützen, schnellen die Kosten für die Entwicklung und Wartung in die Höhe.
- Nutzer finden eine Progressive Web App womöglich schlechter, da diese nicht in App Stores gelistet sein muss.

BROWSER SUPPORT

- Ein vollständiger Support von Progressive Web Apps ist derzeit nur mit Vorabversionen gegeben. Chrome und Firefox sind voll kompatibel, von Safari gibt es positive Signale und Microsoft Edge nutzt künftig ebenfalls Googles Chrome-Module. Auf iOS ist die Offline-Funktionalität verfügbar, seit mit Version 11.3 Service Worker und das Web-App-Manifest in Safari 11.1 implementiert wurden.
- Da PWAs auf Progressive Enhancement setzen, können sie auch in Browsern verwendet werden, die die Service-Worker-Technik nicht unterstützen; nur ist dann eine Internetverbindung nötig.
- es funktioniert unter Windows ab Version 7, macOS. Linux, Xbox One, Android, iOS, iPadOS und natürlich Chrome OS, allerdings nicht mit jedem Browser
- Nicht nutzbar für PWA sind die Browser in Geräten wie Smart TV, Apple TV, Android TV, Chromecast, Playstations und anderen Konsolen, sowie digitalen Assistenten oder VR/AR-Hardware.

AUSBLICK

Was kommt als nächstes?

Die Chromium-Fraktion treibt die Möglichkeiten der PWA-Erstellung weiter voran. Unter Chrome, Samsung Internet und Microsofts neuem Edge gesellen sich neue API, etwa WebNFC, zu den bereits vorhandenen wie Webassembly, WebRTC, Webpush, Webauthn, Webshare, WebXR oder Zahlungsschnittstellen hinzu. Ebenso dürfen wir wohl die Anpassung der Specs des Web App Manifest erwarten, die die Nutzung des populären Dark Mode möglich machen wird.

---

back to readme

ANHANG

PWA > HASHTAGS USED IN THESE SLIDES

Diese Hashtags sind wie Glossar gedacht. Geben Sie ein Hashtag aus der Liste in die Suchfunktion ein, um sich die Definition zum gesuchten Begriff anzuschauen

- #app-shell
- #bubblewrap
- #manifest
- #maskable-icon
- #push-notifications
- #pwa
- #pwa-builder
- #service-worker
- #twa
- #vapid-keys
- #web-app-manifest
- #workbox

PWA > SOURCES USED IN THESE SLIDES

QUELLEN, genutzt für den Slides

https://www.hosteurope.de/blog/warum-sie-progressive-web-apps-pwa-entwickeln-sollten/ Verarbeitet am 30.08.2020

https://t3n.de/news/webdesign-steht-um-jahr-2020-1242754/ Verarbeitet am 30.08.2020

---

back to readme