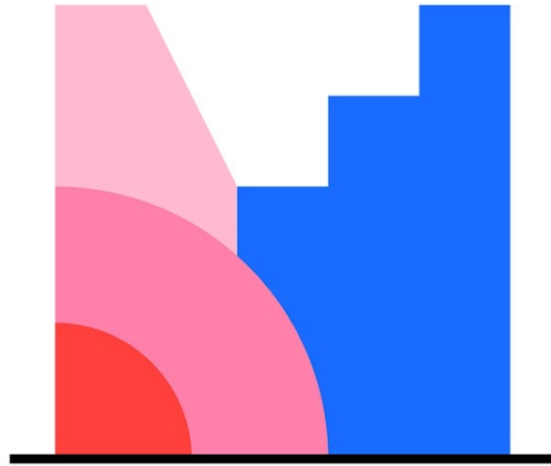


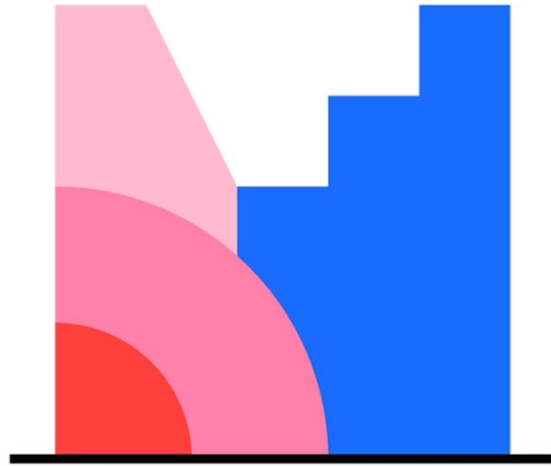
# Blockwoche: Web Programming Lab 🚀

# Recap Clientseitiges JavaScript I














**Mentimeter**

# Recap Serverseitiges JavaScript



**Mentimeter**

# Onboarding – Programm Blockwoche

| Montag<br> | Dienstag<br>  | Mittwoch<br>  | Donnerstag<br>   | Freitag<br>   |
|---|---|--|---|--|
| Architekturansätze<br>von Web<br>Anwendungen<br><br>JavaScript<br>Sprachkonzepte I          | Client-Side-<br>JavaScript I  | Angular  | Angular   | <b>Progressive Web<br/>Apps</b>  |
| JavaScript<br>Sprachkonzepte II   | Client-Side-<br>JavaScript II<br>Frameworks &<br>Typescript   | Angular  | Server-Side-<br>JavaScript  | Authentication @<br>Web Apps   |

# Agenda

- **Progressive Web Apps – Überblick**
- **Offlinefähigkeit von Web Apps**
  - Motivation und High-Level Architektur
  - Assets via ServiceWorker & Cache API bereitstellen
  - Daten via IndexedDB persistieren
- **Offlinefähigkeit von Angular Web Apps** (evtl. Selbststudium)

# Repository

👉 <https://github.com/web-programming-lab/pwa-offline>

# Progressive Web Apps

Überblick

# Was sind Progressive Web Apps (PWAs)?





# Progressive Web Apps (PWAs)?

*The web...but better.*

«Eine Progressive Web App ist eine **Web App**, welche die neusten Web Standards und APIs verwendet um eine «app-like» User Experience ggü. seinen Benutzer zu bieten. Diese Apps sind auf Web Servern deployt, zugreifbar via URLs und indexierbar von Suchmaschinen.» – [Ionic](#)

# Progressive Web Apps (PWAs)?

«Im Vergleich zu traditionellen mobile web apps, sind PWAs viel mehr an **native Apps** angeglichen [...]. Beispielsweise verwenden PWAs **Offline-Speicher** und haben **Zugriff auf native Funktionalitäten** wie z.B. Push Notifcations, Geolocation und die Kamera – alles zugänglich über den Web Browser.» - [Ionic PWA Whitepaper](#)

# Eigenschaften von PWAs

<https://addyosmani.com/blog/getting-started-with-progressive-web-apps/>

# PROGRESSIVE

- ✓ Funktioniert für jeden Benutzer, unabhängig von seiner Browserwahl.
  - ✓ App verbessert sich entlang der Browsermöglichkeiten

# RESPONSIVE

- ✓ Passt in jede Form – Desktop, Mobile, Tablet, was auch immer.

# CONNECTIVITY INDEPENDENT

- ✓ Offline zugreifbar
- ✓ Kann in schlechten Verbindungsszenarien bedient werden.



NO DOWNASOUR!

## No Internet

Try:

- Checking the network cables, modem and router
- Reconnecting to Wi-Fi

ERR\_INTERNET\_DISCONNECTED

# **APP-LIKE**

- ✓ Bietet eine mobile Navigation und Interaktionsmöglichkeiten



# FRESH

- ✓ Immer die aktuellste «Version»

# SAFE

- ✓ Via HTTPS ausgeliefert um mögliche Attacken zu verhindern

# DISCOVERABLE

- ✓ Werden durch Suchmaschinen gefunden.

# RE-ENGAGEABLE

- ✓ Re-engagement einfach durch Features wie z.B. Push-Notifications möglich.



The Washington Post



washingtonpost.com

ADD TO HOME SCREEN



✓ U

erbar

# LINKABLE

- ✓ Kann via URL geteilt und verlinkt werden

# SMALL

- ✓ Die Grösse einer PWA ist ein Bruchteil der Grösse einer nativen App.



doubleclick  
by Google™

# User Study

<https://www.doubleclickbygoogle.com/articles/mobile-speed-matters/>



# Doubleclick User Study

«**53% of users will abandon** a site if it does not load within 3 seconds on a mobile device.»

# Doubleclick User Study

«Most sites take about **19 seconds** to load on a 3G network.»

# Takeaways

## Speed matters.

«Those that can provide a **fast, user-friendly** experience [...], they actually see **increased revenues**. This is where PWAs **separate your company** from the crowd.» - Ionic PWA Whitepaper

# Beispiele



<https://app.starbucks.com/>

Um die Bestellmöglichkeit zugänglicher zu machen, baute Starbucks eine PWA, welche die gleiche User Experience bietet wie ihre Native App.

- Die PWA ist perfekt für Kunden, welche (temporär) eine schlechte Internetverbindung haben.
- Die PWA ist **99.84%** kleiner als die existierende Native-App.
- Die PWA verdoppelte die Anzahl Benutzer, welche aus dem Web eine Bestellung platzierten.

# Wie komme ich zur Progressive Web App?

Für die Erstellung der eigenen PWA hat Google eine [Checkliste](#) veröffentlicht.

Diese umfasst die folgenden Prinzipien:

- Starts **fast**, stays **fast**
- Works in **any browser**
- **Responsive** to any screen
- Provides a **custom offline page**
- Is **installable**



**'Core' Checklist**

# Wie komme ich zur Progressive Web App?

Für die Erstellung der eigenen PWA hat Google eine [Checkliste](#) veröffentlicht.

Diese umfasst die folgenden Prinzipien:

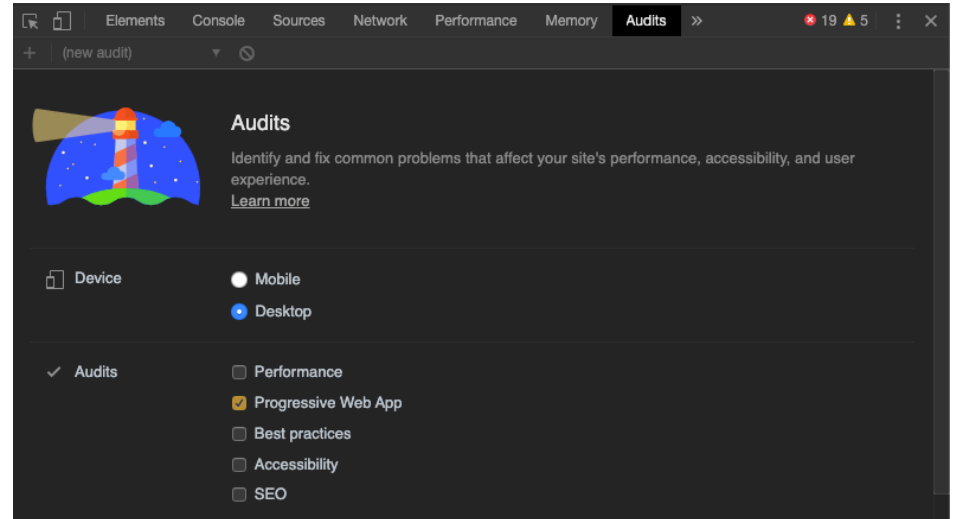
- Provides an **offline experience**
- Is fully **accessible** ([WCAG 2.0](#))
- Can be **discovered** through search
- Works with **any input type**
- Provides **context** for **permission** requests
- Follow **best practices** for healthy code



**‘Optimal’ Checkliste**

# Wie kann ich meine PWA überprüfen?

- Für die Überprüfung von einzelnen Teile der v.a. der 'Core' Checkliste hat Google im Chrome die Audit Funktion. Die Funktion gibt's auch Standalone (Lighthouse).
- Mittels der Lighthouse / Audit Funktion können Aspekte aus der Checkliste überprüft werden.



# Zusammenfassung

- PWAs sind Web Apps, die eine “app-like“ User Experience ggü. dem User bieten, in dem sie die modernen Möglichkeiten des Browser ausschöpfen.
- PWAs sind...  
Progressive, Responsive, Unabhängig von der Verbindung, App-Like, auf dem neusten Stand, sicher, auffindbar via Suchmaschine, installierbar, verlinkbar und klein.
- Um eine Progressive Web App zu bauen, kann die Google Checkliste verwendet werden.
- Die Grösse einer Web App kann wirtschaftliche Einflüsse haben.



# Fragen

> Was sind die Vor- und Nachteile von PWAs ggü. native Apps?



# Offlinefähigkeit von Web Apps

Motivation und High-Level Architekturüberblick

# Demo



**«Never load the same resource twice.» -  
developers.google.com**

# PWA Checkliste von Google - Offlinefähigkeit

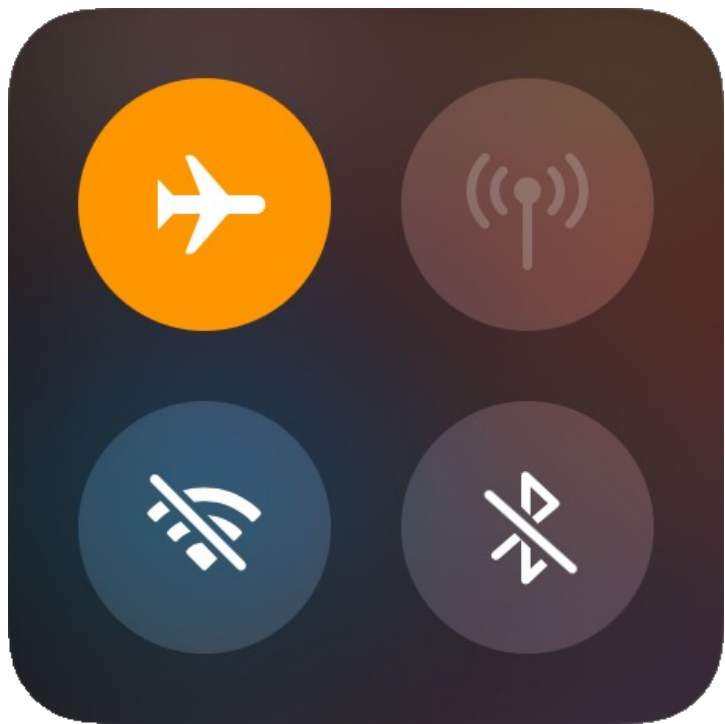
Die Offlinefähigkeit ist ein relevantes Teilproblem des Designs von PWAs.

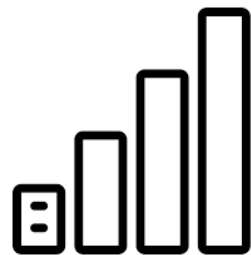
Folgende Punkte aus der PWA-Checkliste betreffen die Offlinefähigkeit:

- «Provides a custom offline page» → Core
- «Provides an offline experience» → Optimal
  - «An offline PWA provides a true app-like experience for users.»

Web App sollte offlinefähig sein → Motivation?

# Motivation für offlinefähige Web Apps?



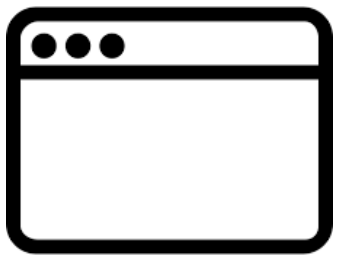






„Akamai study shows that every 100ms delay in website load time can hurt conversion rates by 7% - that is a significant drop in sales [...]“ – [gigaspace.com](https://gigaspace.com)

**Wie setzen wir die  
Offlinefähigkeit technisch um?**

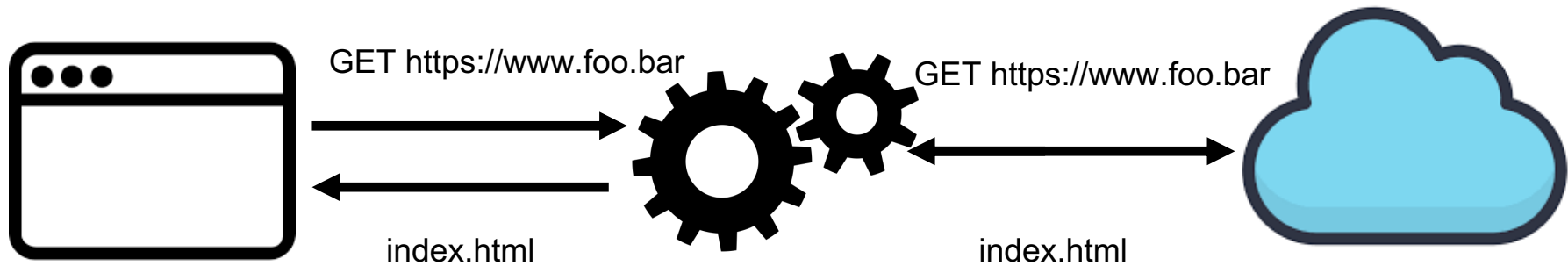


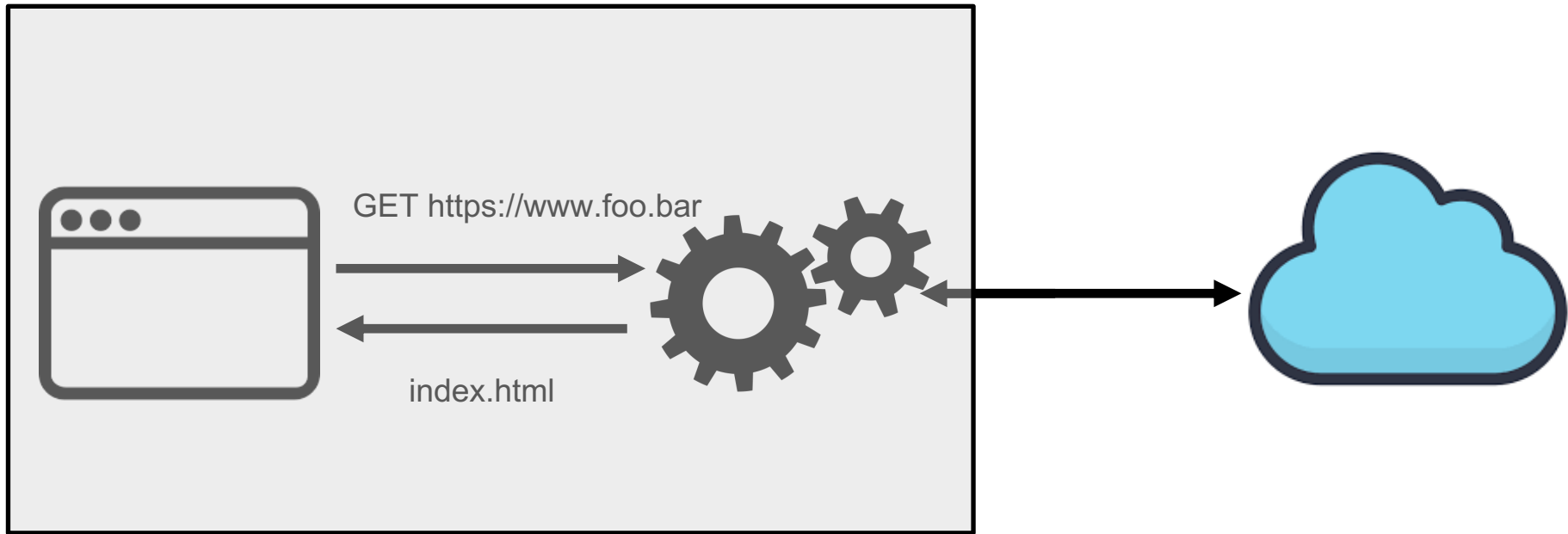
GET https://www.foo.bar

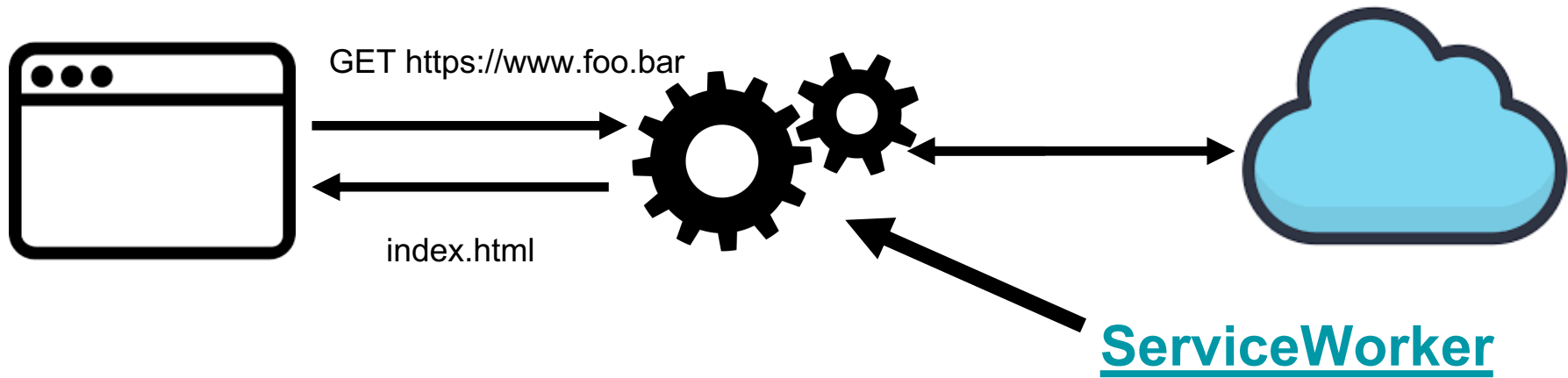


index.html









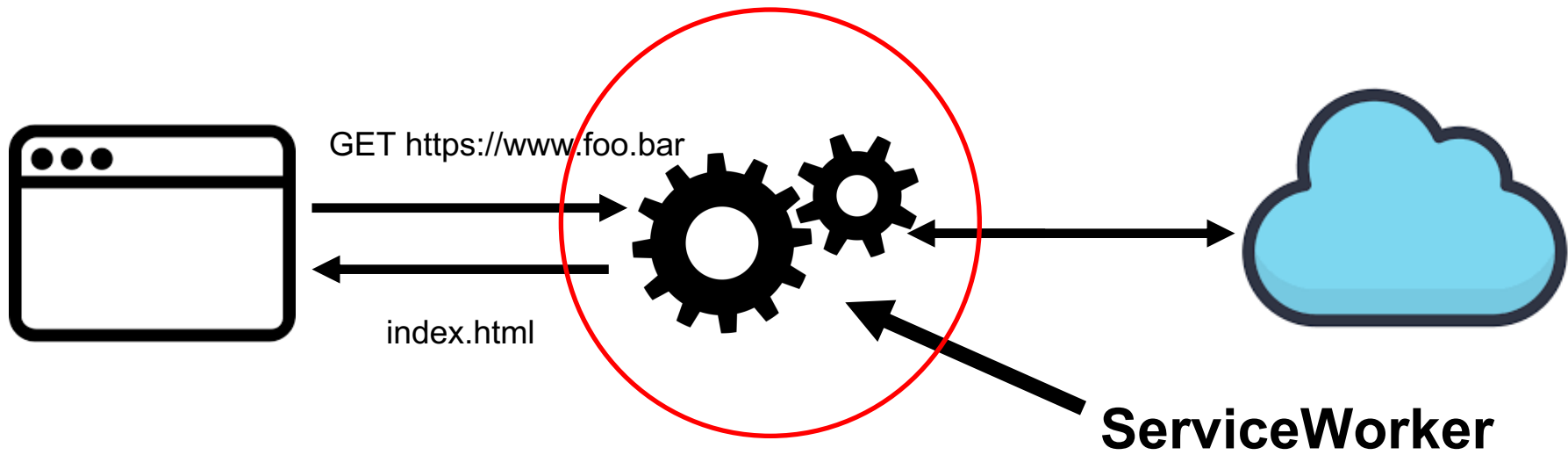
# ServiceWorker

«A service worker is a script that your browser **runs in the background**, separate from a web page, opening the door to features that **don't need** a web page or **user interaction**» - [developers.google.com](https://developers.google.com/service-workers/)

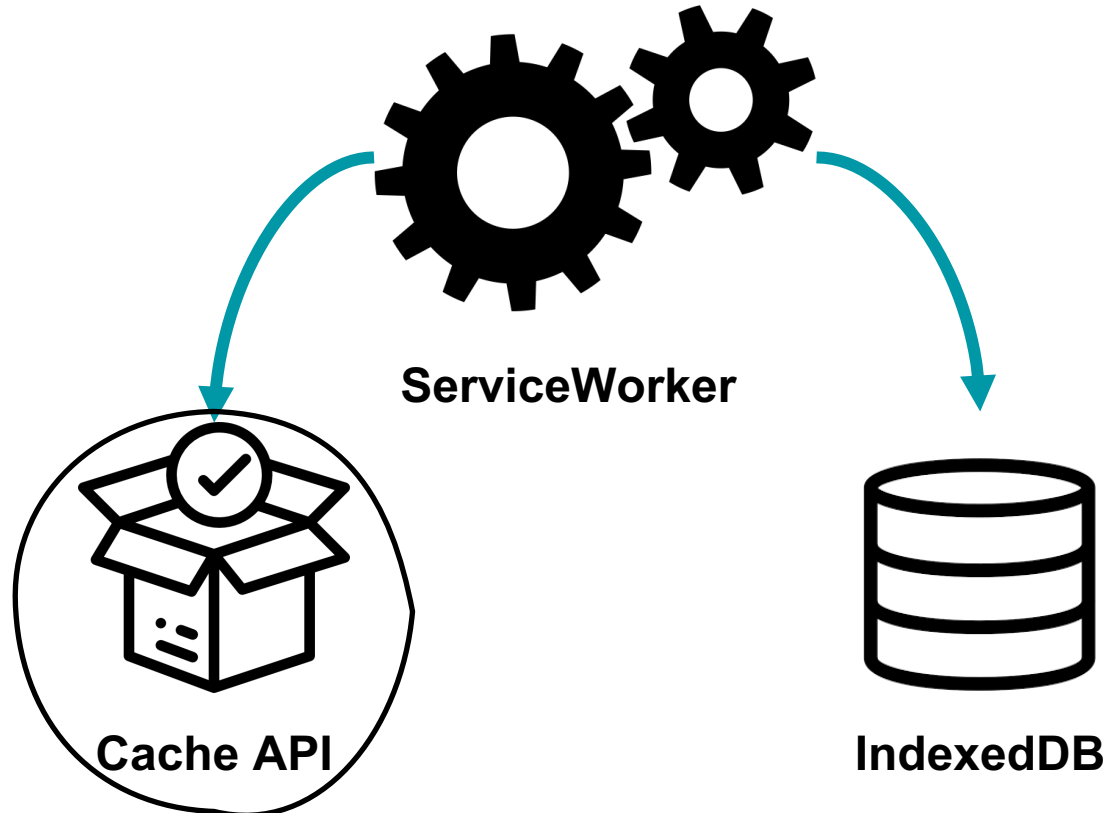
# ServiceWorker - Eigenschaften

- Es ist ein [JavaScript Worker](#), d.h. er kann nicht direkt auf den DOM zugreifen. Der JavaScript Worker arbeitet in einem Worker Thread.
- Ein [Service Worker](#) ist ein «Netzwerk Proxy», welcher die Netzwerk Requests von den einzelnen Seiten verwaltet.
- Wird terminiert, falls nicht verwendet oder neugestartet wenn verwendet. D.h. falls man Informationen über die Neustarts von ServiceWorkern braucht, müssen diese persistiert werden.
- Service Workers verwenden extensiv Promises. Diese sind die Basis für die Verwendung von Service Workern.





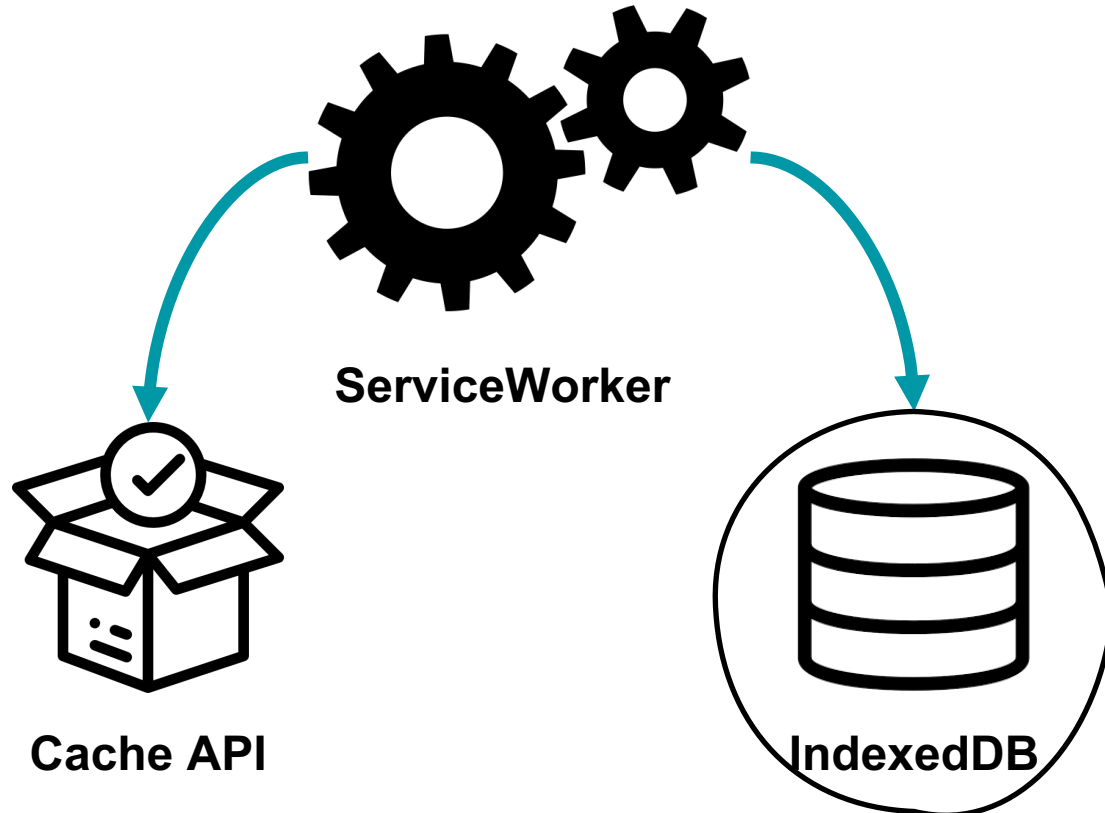
# Browser Unterstützung für Offlineverfügbarkeit



# Offlineverfügbarkeit - Cache API

- Die **Cache API** wird von ServiceWorker verwendet um Application Assets zu cachen.
- Die Cache API ist unabhängig vom Browser HTTP Cache.
- Die Cache API ist **offline verfügbar**. Mittels der Cache API kann eine Web App Offline via Service Worker verfügbar gemacht werden.
- Die Cache API ist ideal um **Ressourcen** mit einer **spezifischen URL** zu speichern.

# Browser Unterstützung für Offlineverfügbarkeit



# Offlineverfügbarkeit – Indexed Database

- Die **IndexedDB** ist eine asynchrone und transaktional nutzbare Browserdatenbank. Sie wird verwendet um Daten lokal zu speichern.
- Die IndexedDB kann einfache oder hierarchische Objekte persistieren und führt Indexe über die perstierten Records.
- Mittels **IndexedDB** können beispielsweise User Aktionen aufgenommen werden während das Gerät offline ist (oder bei schlechte Verbindung) und können ggü. dem Backend ausgeliefert werden, wenn die Verbindung wieder steht.
- Spezifikation: <https://w3c.github.io/IndexedDB/>

# Offlineverfügbarkeit – Browser Storage

| Browser Storage API                    | Data Model  | Persistence | Transactions | Async / Sync |
|--|-------------|-------------|--------------|--------------|
| <a href="#"><u>File System</u></a>     | Byte Stream | Device      | No           | Async        |
| <a href="#"><u>Local Storage</u></a>   | Key/Value   | Device      | No           | Sync         |
| <a href="#"><u>Session Storage</u></a> | Key/Value   | Session     | No           | Sync         |
| <a href="#"><u>Cookie</u></a>          | Structured  | Device      | No           | Sync         |
| <a href="#"><u>WebSQL</u></a>          | Structured  | Device      | Yes          | Async        |
| <a href="#"><u>Cache</u></a>           | Key/Value   | Device      | No           | Async        |
| <a href="#"><u>IndexedDB</u></a>       | Hybrid      | Device      | Yes          | Async        |

<https://developers.google.com/web/fundamentals/instant-and-offline/web-storage>

# Was ist mit den anderen Storage Mechanismen?

- WebSQL & AppCache → Deprecated
- LocalStorage und SessionStorage sind synchron und haben keinen Web Worker Support und sind typenlimitiert (Strings).
- Cookies haben ihre eigene Anwendung (Client / Server)
- Die File System API braucht dedizierte Berechtigungen.

# Empfehlung Offline Storage für PWAs

- Für alle Netzwerk Ressourcen, die benötigt werden um die App zu laden, während das Gerät offline ist, verwendet die **Cache API** (als Teil des Service Worker)
- Für alle anderen Daten **IndexedDB**

## Argumentarium

Beide APIs sind asynchron (Promise-based). Beide APIs können im Window- und ServiceWorker-Kontext verwendet werden.

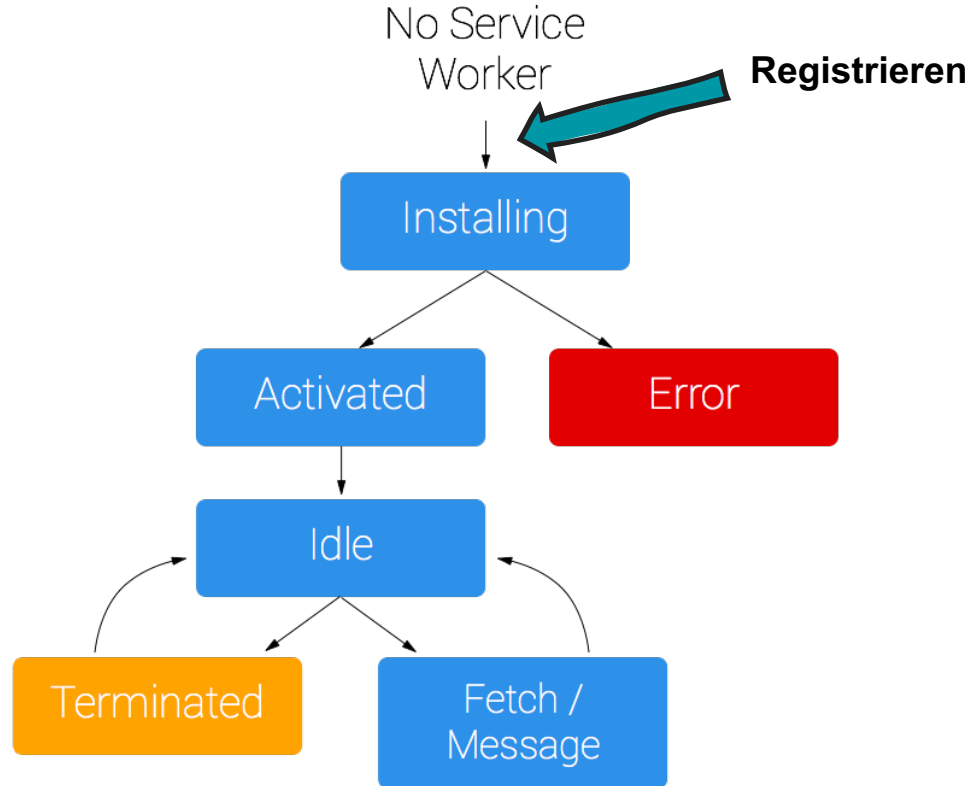


# Offlinefähigkeit von Web Apps

Funktionsweise ServiceWorker und Cache API - Content offline nutzbar machen

**Demo**

# ServiceWorker - Lebenszyklus



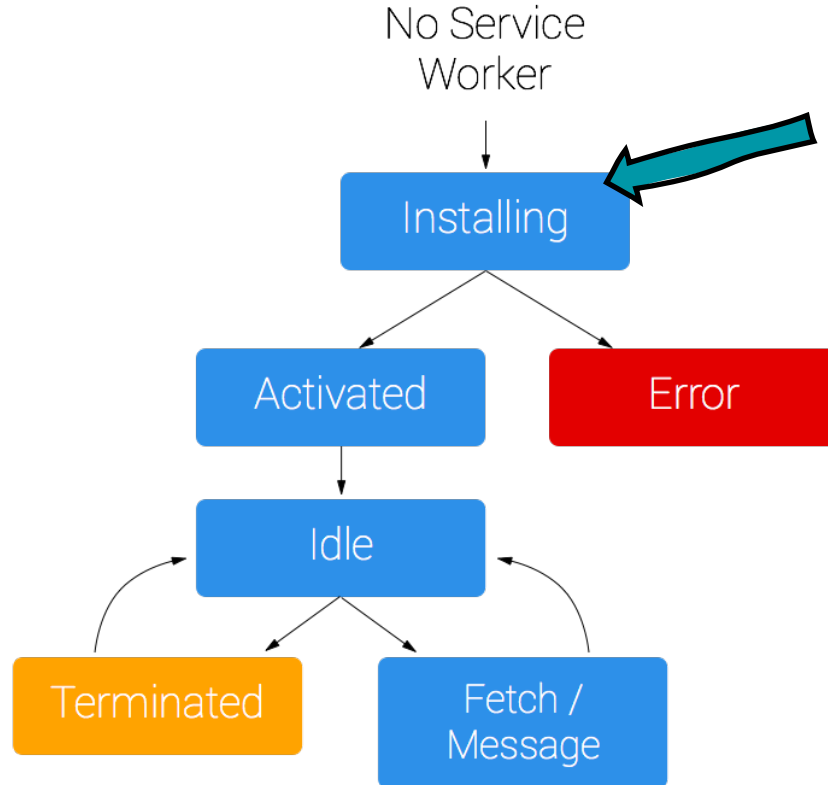
# ServiceWorker – Registrieren in der Web App

Mit dem folgenden Code wird dem Browser mitgeteilt wo die ServiceWorker Datei lebt und so die Installation des ServiceWorkers startet.

```
if ('serviceWorker' in navigator) {  
  window.addEventListener('load', function() {  
    navigator.serviceWorker.register('/sw.js').then(function(registration) {  
      // Registration was successful  
      console.log('ServiceWorker registration successful with scope: ', registration.scope);  
    }, function(err) {  
      // registration failed :(  
      console.log('ServiceWorker registration failed: ', err);  
    });  
  });  
}
```

Analyse von ServiceWorker: via Debug Console, via <chrome://inspect/#service-workers> oder detailliert <chrome://serviceworker-internals>

# ServiceWorker - Lebenszyklus



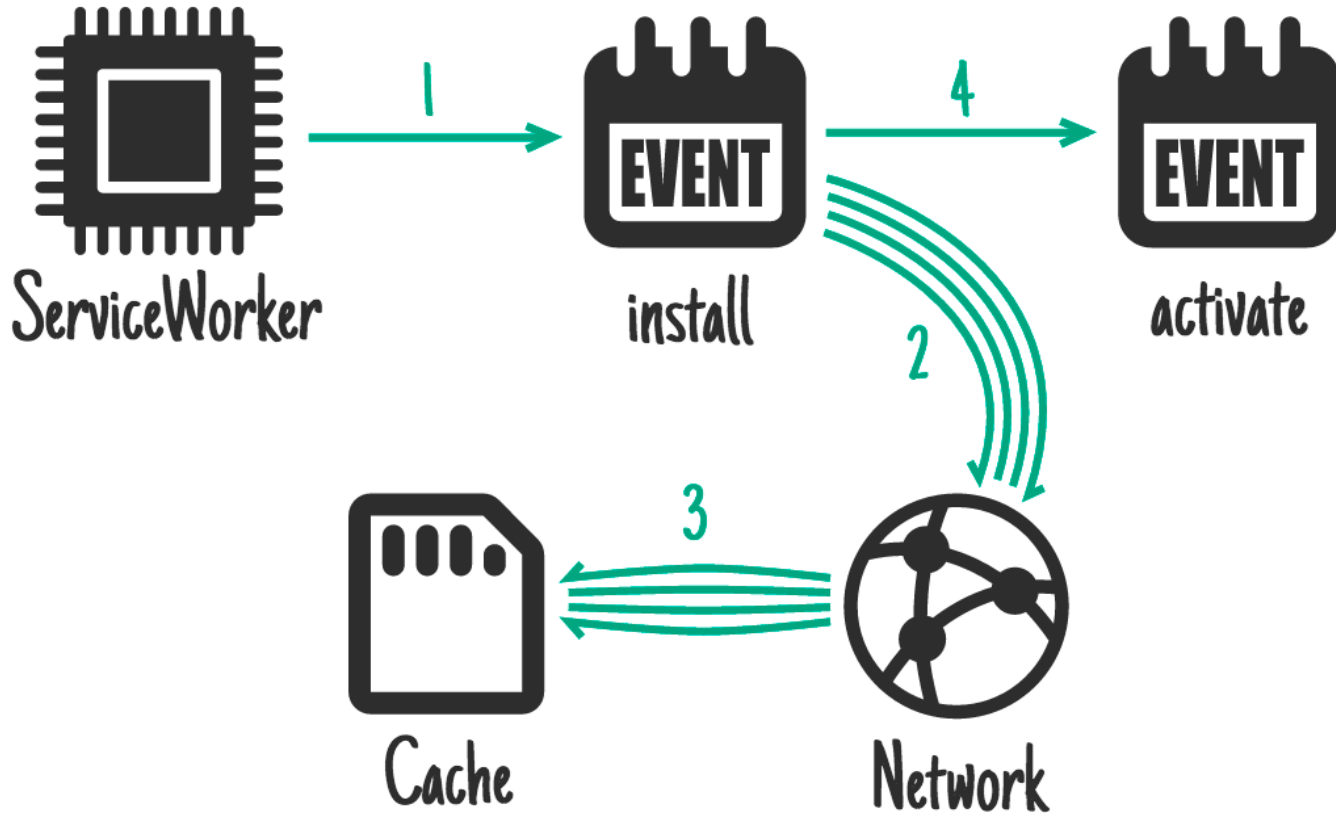
# ServiceWorker – Installation I

Die Registrierung startet den Installationsprozess im ServiceWorker Script, welches via `install` Event getriggert wird.

```
self.addEventListener('install', function(event) {  
  // Perform install steps  
});
```

sw.js

# ServiceWorker – Installation II



# ServiceWorker – Installation III

```
var CACHE_NAME = 'my-site-cache-v1';
var urlsToCache = [
  '/',
  '/styles/main.css',
  '/script/main.js'
];

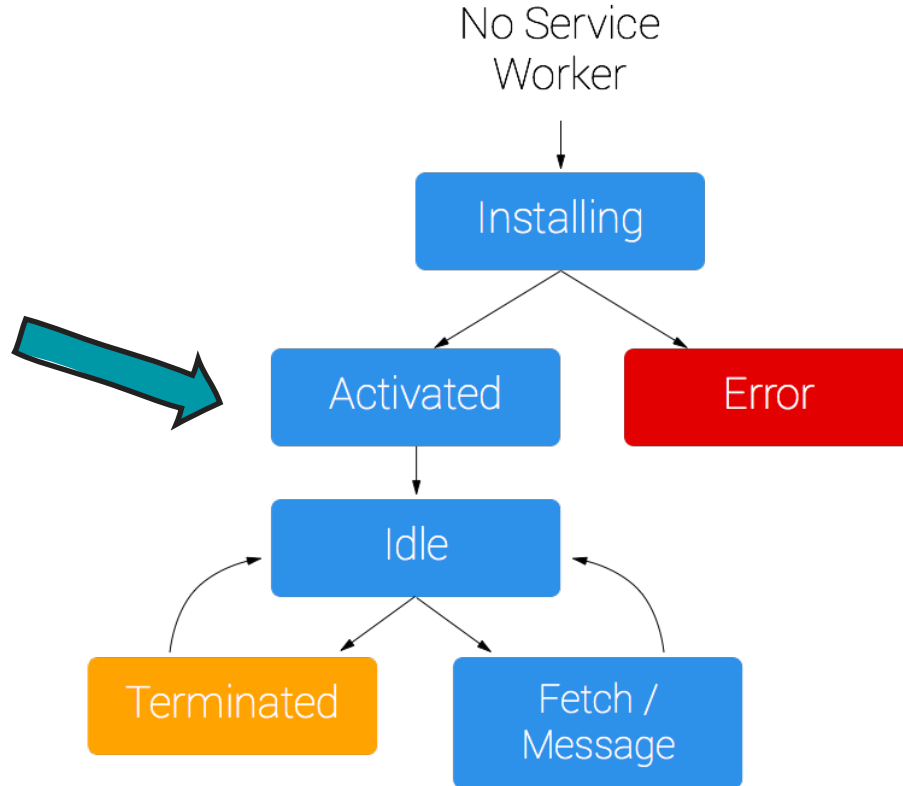
self.addEventListener('install', function(event) {
  // Perform install steps
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(function(cache) {
        console.log('Opened cache');
        return cache.addAll(urlsToCache);
      })
  );
});
```

sw.js

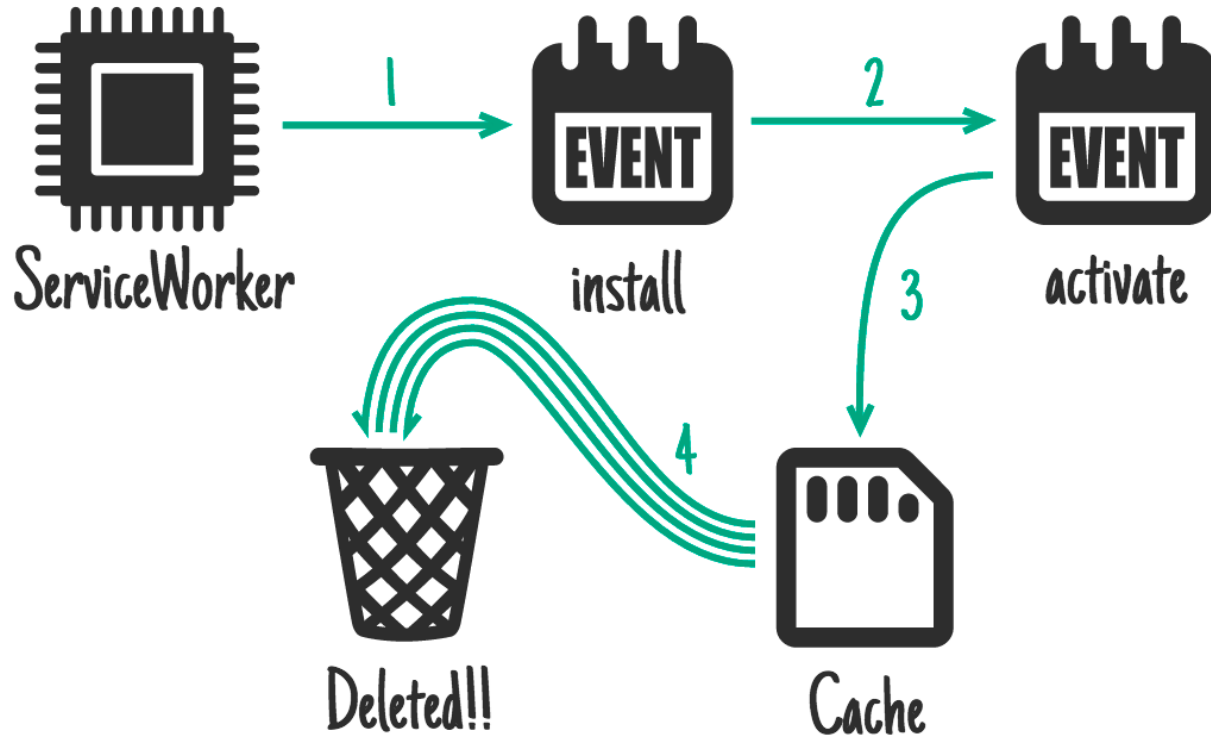


# ServiceWorker - Lebenszyklus

**Clean-Up & Migration**



# ServiceWorker – Clean-up & Migration



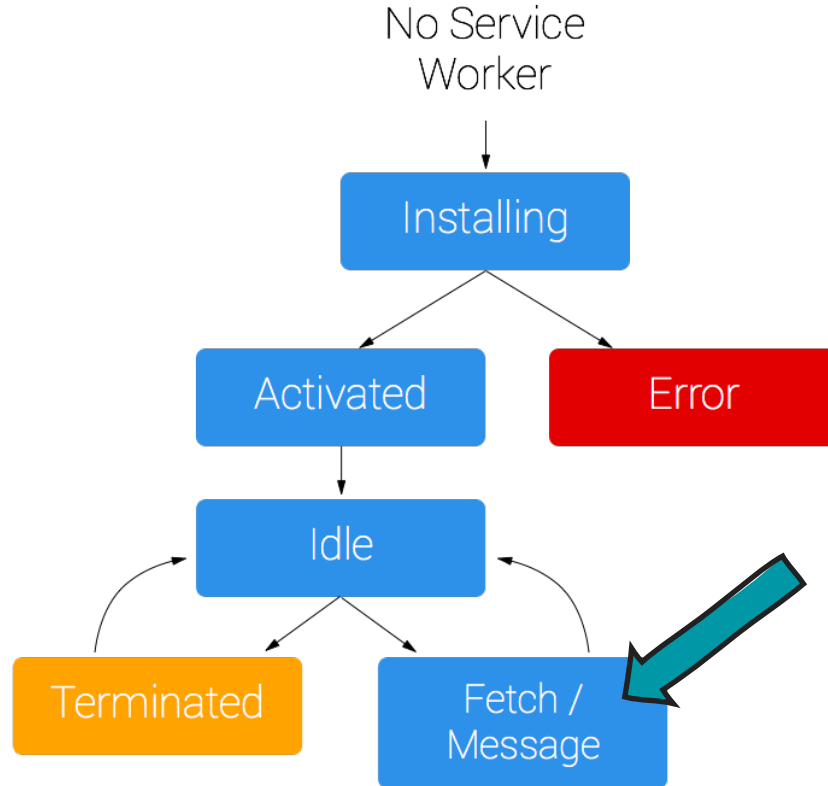
# ServiceWorker – Aktualisieren

I.d.R. wird im `activate` Callback der Cache aktualisiert. So kann die alte Instanz keine veralteten Cache-Einträge mehr ausliefern.

Szenario: my-site-cache-v1 aufteilen in pages-cache-v1 und blog-posts-cache-v1

```
self.addEventListener('activate', function(event) {  
  
    var cacheWhitelist = ['pages-cache-v1', 'blog-posts-cache-v1'];  
  
    event.waitUntil(  
        caches.keys().then(function(cacheNames) {  
            return Promise.all(  
                cacheNames.map(function(cacheName) {  
                    if (cacheWhitelist.indexOf(cacheName) === -1) {  
                        return caches.delete(cacheName);  
                    }  
                })  
            );  
        })  
    );  
});
```

# ServiceWorker - Lebenszyklus



**Intercepting  
XHR- oder  
Fetch-Requests**

# ServiceWorker – Fetch

Der ServiceWorker erhält sämtliche fetch-Events in seiner Domäne.

```
self.addEventListener('fetch', function(event) {  
  event.respondWith(  
    caches.match(event.request)  
      .then(function(response) {  
        // Cache hit - return response  
        if (response) {  
          return response;  
        }  
        return fetch(event.request);  
      })  
  )  
});
```

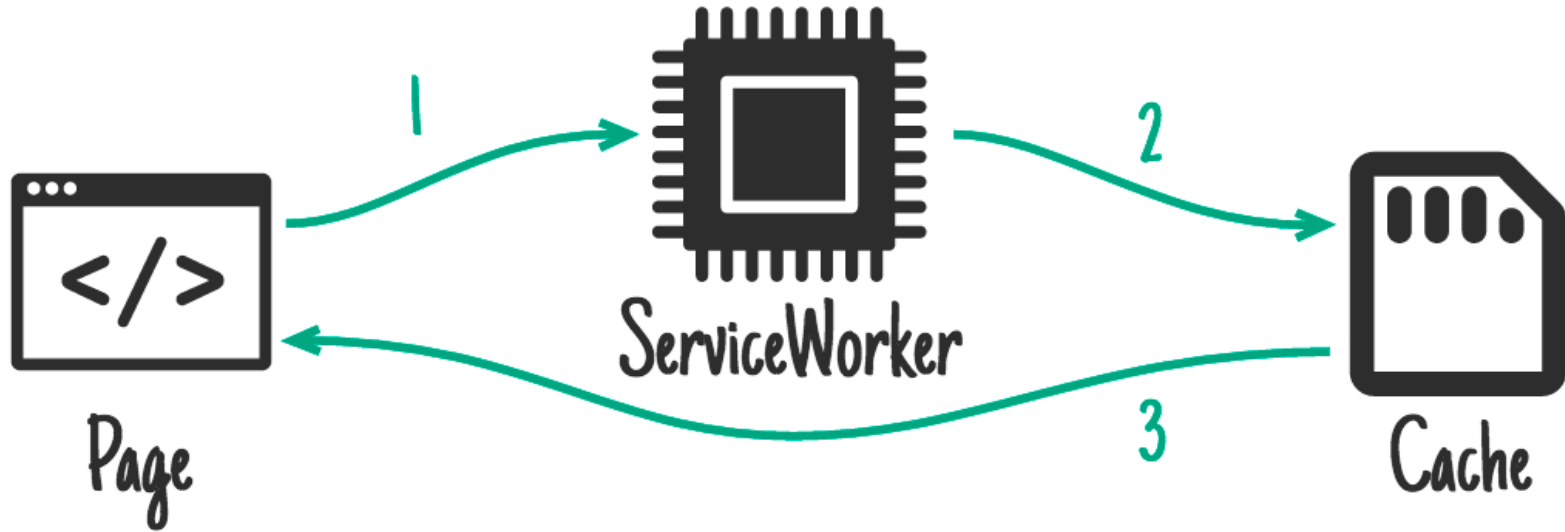
sw.js

# ServiceWorker – Fetch-Strategien

Es gibt verschiedene Fetch-Strategien:

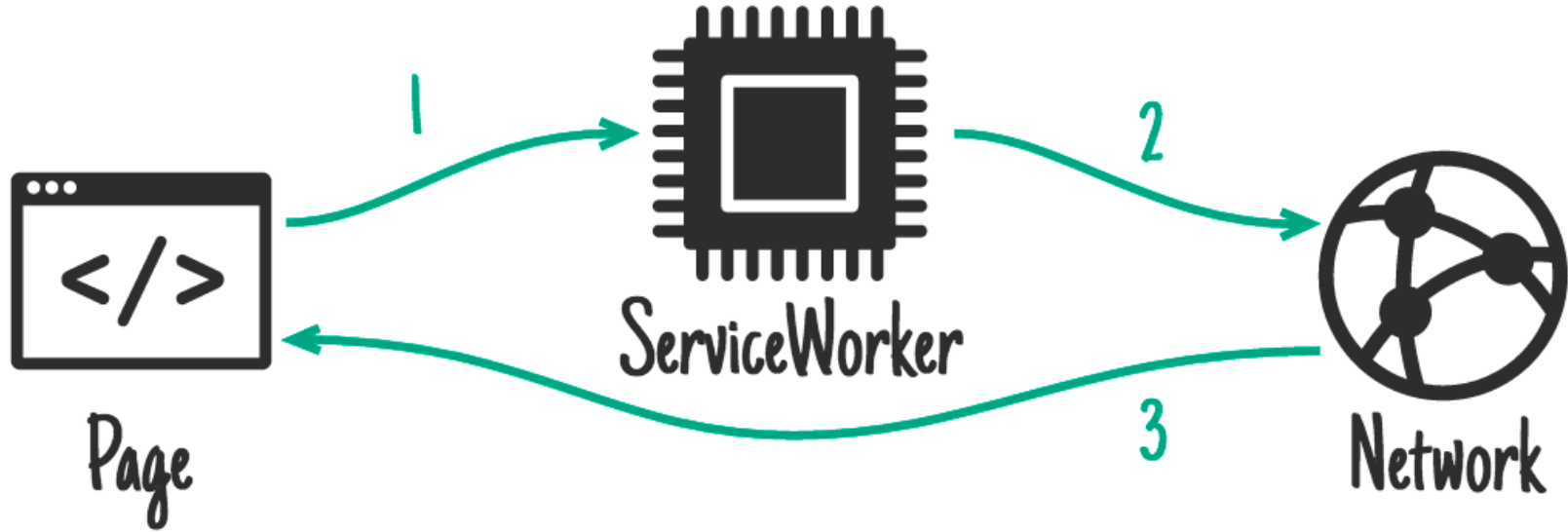
- Cache only
- Network only
- Cache, falling back to network
- Cache & network race
- Network falling back to cache
- Cache then network

# ServiceWorker – Cache-Only



Ideal für statische Assets, gecacht via install Event

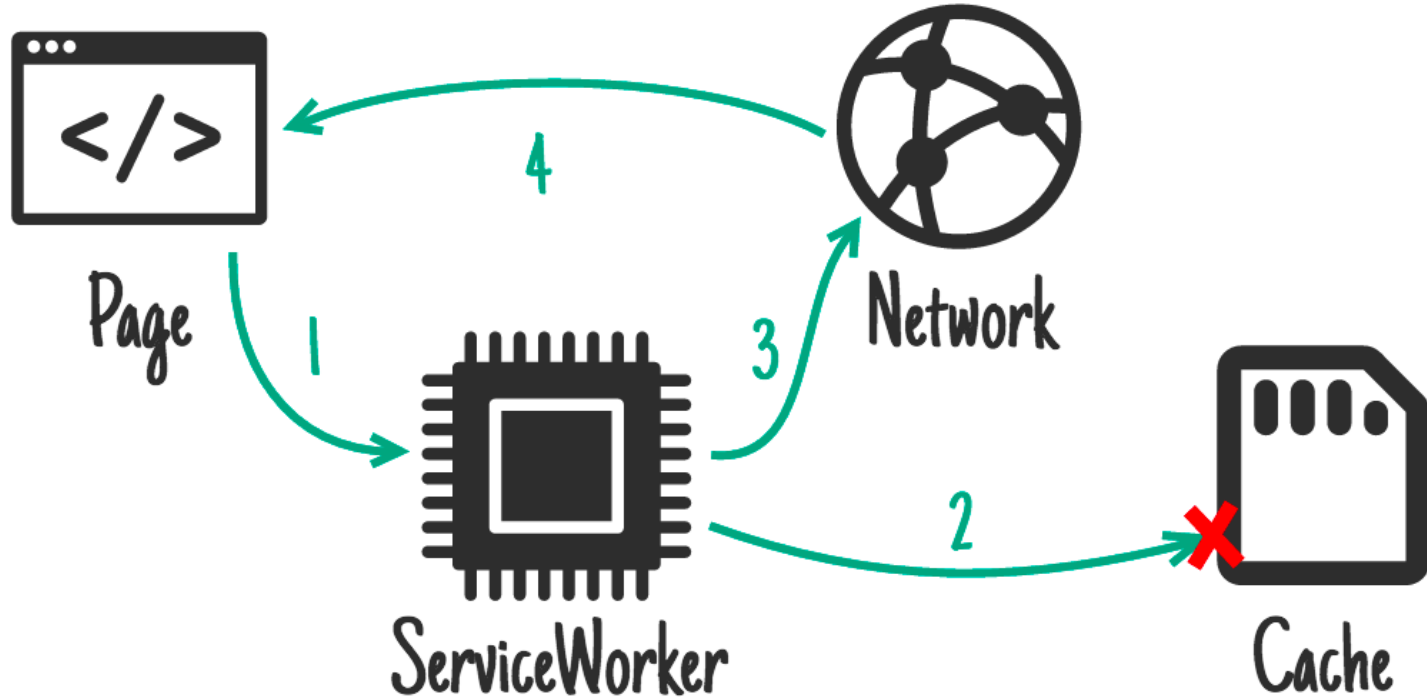
# ServiceWorker – Network only



Ideal für Dinge die nicht offline Verfügbar sein können, wie z.B. Analytics-Pings oder non-GET Requests.

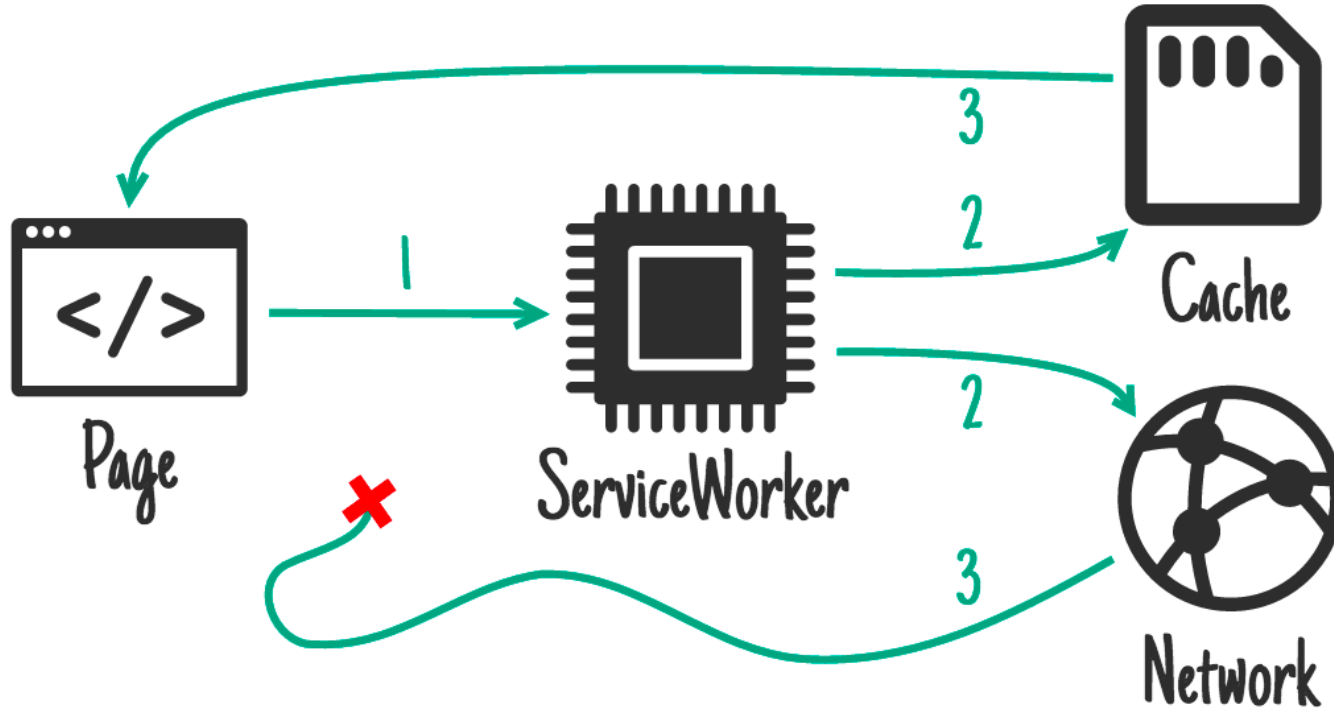


# ServiceWorker – Cache, falling back to network



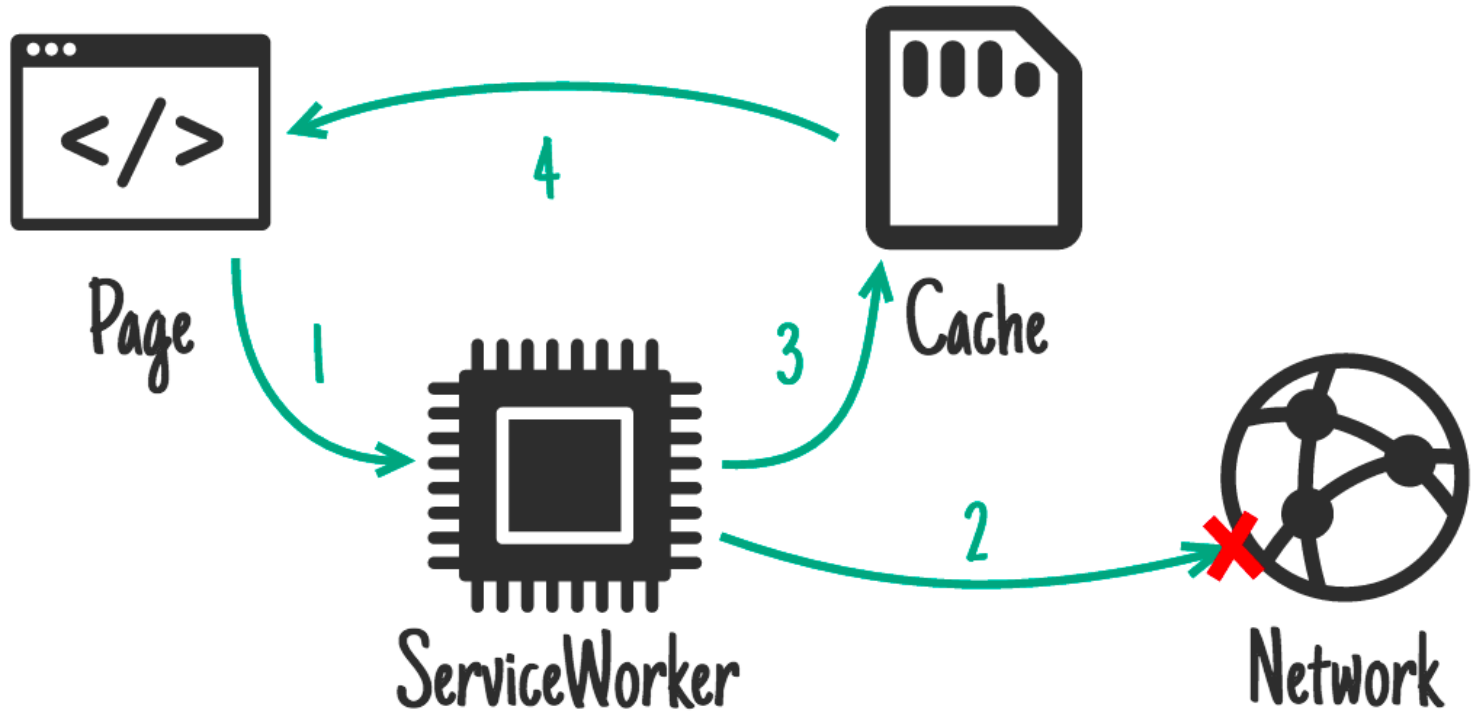
Ideal für Offline-First, die meisten Requests können so verarbeitet werden.

# ServiceWorker – Cache & Network race



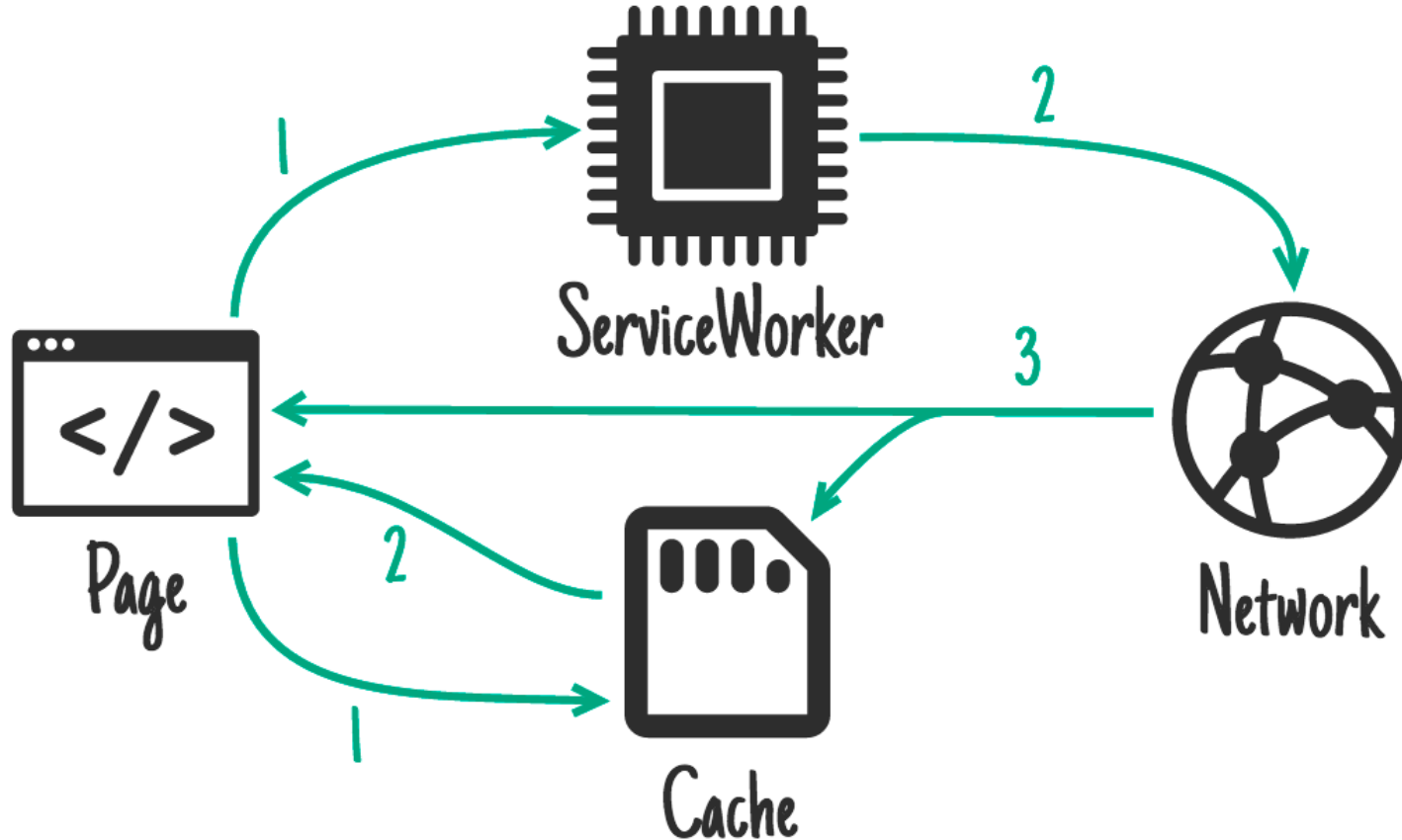
Ideal für kleinere Assets und Geräten mit einem langsam Disk-Zugriff

# ServiceWorker – Network falling back to cache



Ideal für Ressourcen welche oft ändern, z.B. “Social Media Timelines, Game Leader Boards“

# ServiceWorker – Cache then network



Ideal für Ressourcen welche oft ändern, z.B. “Social Media Timelines, Game Leader Boards“

# ServiceWorker – Fetch-Strategien

Es gibt verschiedene Fetch-Strategien:

- Cache only
- Network only
- Cache, falling back to network
- Cache & network race
- Network falling back to cache
- Cache then network

I.d.R. setzt man nicht eine Strategie durch, sondern je nach Art des Requests resp. Response die entsprechende Strategie.

# ServiceWorker – Fetch-Strategien umsetzen

```
self.addEventListener('fetch', function(event) {  
  // Parse the URL:  
  var requestURL = new URL(event.request.url);  
  
  // Handle requests to a particular host specifically  
  if (requestURL.hostname == 'api.example.com') {  
    event.respondWith(/* some combination of patterns */);  
    return;  
  }  
  // Routing for local URLs  
  if (requestURL.origin == location.origin) {  
    // Handle article URLs  
    if (/^\/article\/\./.test(requestURL.pathname)) {  
      event.respondWith(/* some other combination of patterns */);  
      return;  
    }  
    if (/\.webp$/i.test(requestURL.pathname)) {  
      event.respondWith(/* some other combination of patterns */);  
      return;  
    }  
    if (request.method == 'POST') {  
      event.respondWith(/* some other combination of patterns */);  
      return;  
    }  
  }  
}
```

# Übungsaufgabe (30')

1. Ermittle den Zustand online / offline und gebe diesen aus.
2. Mache den Offline-Seed offline fähig, d.h. der Reload ist auch möglich, wenn der Browser offline ist.

Übung: exercises / online-offline

Starten via «`npm run start`» → `http://localhost:8080`

# Offlinefähigkeit von Web Apps

Funktionalitäten einer Web App offlinefähig gestalten mittels IndexedDB



**Demo**

🔍 📄

Elements

Console

Sources

⚠️ Network

Performance

Memory

Application

Security

Audits

🚨 12 ⋮ ✕

Application

Manifest

⚙️ Service Workers

🗑️ Clear storage

Storage

▶ 📁 Local Storage

▶ 📁 Session Storage

▼ 📁 IndexedDB

▶ 📁 \_\_dbnames - http://localhost:

▼ 📁 notes - http://localhost:8000

▼ 📁 notes

message

notebookId

title

date

synched

📁 Web SQL

▶ 🍪 Cookies

Cache

▶ 📁 Cache Storage

📁 Application Cache

🔄 ⏪ ▶ Start from key

🚫 ✕

| # | Key (Key path: "id") | Value  |
|---|----------------------|--|
| 0 | 7                    | <div>▼ {notebookId: "c917d160-31e3-11ea-9d6d-63b408063302", title: "Test-Note", notebookId: "c917d160-31e3-11ea-9d6d-63b408063302", title: "Test-Note Offline", message: "This is offline", date: "2020-01-19T12:51:53.957Z", synch: false, id: 7}</div> |

Total entries: 1 | Key generator value: 8

# Indexed Database API - IndexedDB

- Objektdatenbank, i.d.R. werden JavaScript Objekte gespeichert
- Es können aber auch Bilder oder andere Dateien in der DB gespeichert werden.
- Es können Indizes angelegt werden, um eine effiziente Suche zu ermöglichen
- Die IndexedDB funktioniert event-basiert.
- Promise-Based Zugriff via DexieJS

# Dexie.js Promise Wrapper für IndexedDB



# IndexedDB– Create DB via Dexie.js

```
/*  
|-----|  
| Declare your database |  
|-----|  
*/  
  
const db = new Dexie('MyDatabase');  
  
// Declare tables, IDs and indexes  
db.version(1).stores({  
  friends: '++id, name, age'  
});
```

# IndexedDB – Query via Dexie.js

```
/*
|-----|
| Then run some queries |
|-----|
*/

// Find some old friends
const oldFriends = await db.friends
    .where('age').above(75)
    .toArray();

// or make a new one
await db.friends.add({
    name: 'Camilla',
    age: 25,
    street: 'East 13:th Street',
    picture: await getBlob('camilla.png')
});
```

# Übungsaufgaben (30')

1. Erstelle ein Formular, das Vor- und Nachname entgegen nimmt und in der IndexedDB speichert.
2. Lies die persistierten Personen aus und rendere diese entsprechend in einer Liste.

|       |         |      |
|-------|---------|------|
| Meier | Joachim | Save |
|-------|---------|------|

- Roos Patrick
- Schürmann Andreas

# Offlinefähigkeit von Angular Web Apps

@angular/pwa Package



# PWA mit Angular Web Apps

- Die Angular Plattform bietet ein [eigenes PWA Package an](#), um die Web Assets einfach offline verfügbar zu machen.

angular.json

```
npx ng add @angular/pwa
```

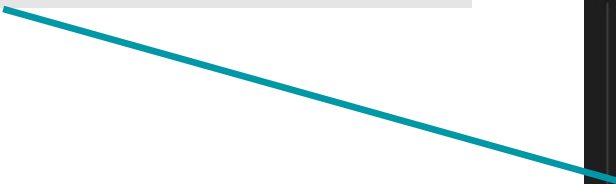
```
"architect": {  
  "build": {  
    "builder": "@angular-devkit/build-angular:browser",  
    "options": {  
      "outputPath": "dist/angular-offline-app",  
      "index": "src/index.html",  
      "main": "src/main.ts",  
      "polyfills": [  
        "zone.js"  
      ],  
      "tsConfig": "tsconfig.app.json",  
      "inlineStyleLanguage": "scss",  
      "assets": [  
        "src/favicon.ico",  
        "src/assets",  
        "src/manifest.webmanifest" ] } } } } Installable  
    "styles": [  
      "src/styles.scss"  
    ],  
    "scripts": [],  
    "serviceWorker": true,  
    "ngswConfigPath": "ngsw-config.json" } } } } Offline & Performance  
  "configurations": {
```

# Service Workers in Angular Web Apps

- Die Angular Plattform bietet ein [eigenes PWA Package an](#), um die Web Assets einfach offline verfügbar zu machen.

app.module.ts

npx ng add @angular/pwa



```
import { ServiceWorkerModule } from '@angular/service-worker';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    ServiceWorkerModule.register('ngsw-worker.js', {
      enabled: !isDevMode(),
      // Register the ServiceWorker as soon as the application is stable
      // or after 30 seconds (whichever comes first).
      registrationStrategy: 'registerWhenStable:30000'
    })
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# Service Worker Configuration in Angular Apps

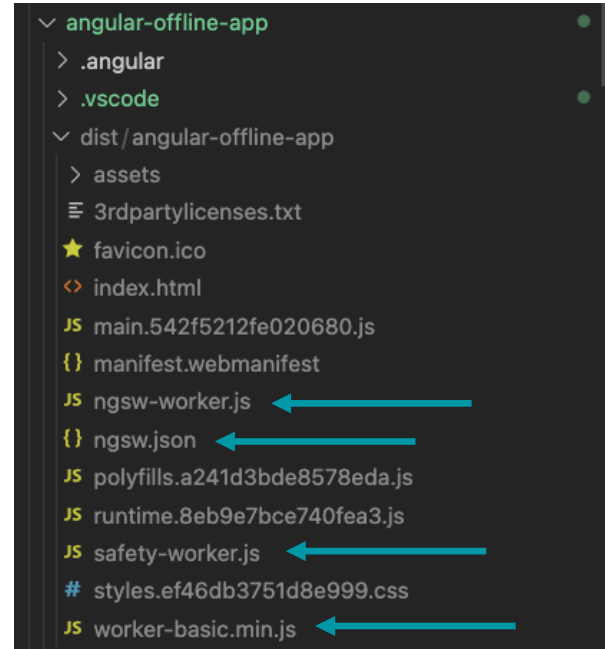
- Angular bietet eine high-level Konfiguration der Service Worker an. Die Konfiguration befindet sich im erstellten [ngsw-config.json](#).
- In dieser Konfiguration können die Assets deklariert werden, welche offline zur Verfügung gestellt werden sollen (via Service Worker).
- Zusätzlich kann die [Caching Strategie](#) für die Installation sowie Update definiert werden (Install Mode / Update Mode)

ngsw-config.json

```
{
  "$schema": "./node_modules/@angular/service-worker/config/schema.json",
  "index": "/index.html",
  "assetGroups": [
    {
      "name": "app",
      "installMode": "prefetch",
      "resources": {
        "files": [
          "/favicon.ico",
          "/index.html",
          "/manifest.webmanifest",
          "/*.css",
          "/*.js"
        ]
      }
    },
    {
      "name": "assets",
      "installMode": "lazy",
      "updateMode": "prefetch",
      "resources": {
        "files": [
          "/assets/**",
          "/*. (svg|cur|jpg|jpeg|png|apng|webp|avif|gif|otf|ttf|woff|woff2)"
        ]
      }
    }
  ]
}
```

# Service Worker Configuration in Angular Apps

- Im Build Prozess (ng build) wird die Service Worker Manifest Datei (ngsw.json), die Service Worker (ngsw-worker.js) und die dazugehörigen Registrierungsscripts (safety-worker.js, worker-basic.min.js) generiert und in die Angular Bundles verpackt (main.xyz.js).



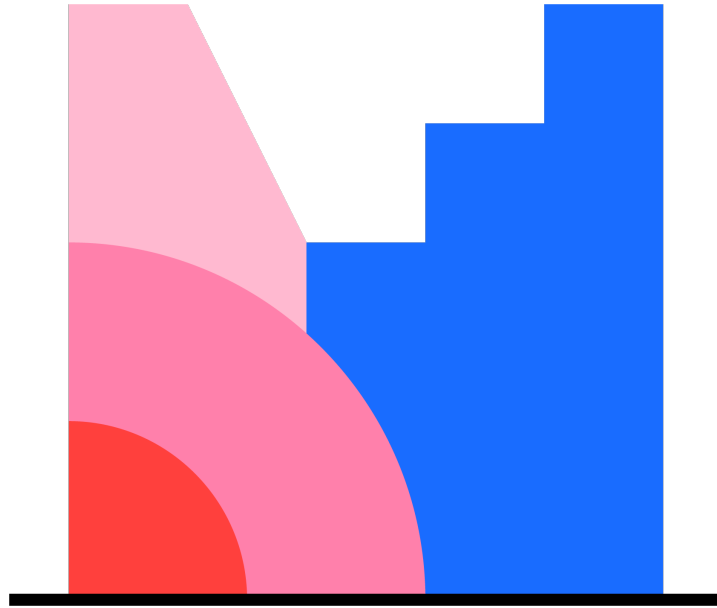
# Übungsaufgabe (15')

Erstellen Sie eine Angular App mit welcher die Web Assets offline verfügbar sind.

1. Erstellen Sie ein neues Angular Projekt
2. Fügen Sie das @angular/pwa package hinzu
3. Bauen Sie das Projekt & prüfen Sie die ob die Assets offline verfügbar sind

Tipp: http-server auf dist Verzeichnis

## Recap PWA



**Mentimeter**