Progressive Web Apps

Getting Started

Vorbereitung

- Klone das Github Repository
- Führe “docker-compose up” in Chatter/ aus
  - Rufe http://localhost in dem Browser auf, es sollte nun eine Website mit dem Titel Chatter zu sehen sein.
- Unter Chatter/nginx/html/ findet sich die gehostete Website, hier werden wir auch arbeiten
  - Chatter/nginx/html/ bezeichnen wir von jetzt an als root [“/”]

Service Worker aufsetzen

- Erstelle das File service-worker.js in /
- Registriere den Service Worker in /js/main.js
  - if ("serviceWorker" in navigator) {
    navigator.serviceWorker.register("/service-worker.js")
    .then(registration => {
    console.log("Registration successful: Scope: " + registration.scope);
    })
    .catch(reason => {
    console.log("Registration failed: Error: " + reason)
    });
    }

Cache aufsetzen - service_worker.js

- Cache Name und Version definieren
- Files welche gecached werden sollen definieren
  - const cacheAssets = [
    "/index.html",
    "/users.html",
    "/chat.html",
    "/css/style.css",
    "/js/chat.js",
    "/js/users.js",
    "/js/index.js",
    "/js/main.js"
    ];

Während dem Installieren

- Event Listener für das Install Event registrieren:
  - self.addEventListener('install', event => { })
- Cache während dem Install Event aufsetzen:
  - caches.open(cacheName)
    .then(cache => {
      return cache.addAll(cacheAssets)
    })
- Das Install Event darf nicht beendet werden solange die Caches noch gespeichert werden.
  - event.waitUntil(Promise)

Install Event

```
self.addEventListener('install', event => {
    console.log("Service worker installing...");
    event.waitUntil(
        caches.open(cacheName)
            .then(cache => {
                return cache.addAll(cacheAssets);
            })
    );
});
```

Alte Caches aufräumen

- Dies machen wir im Activate Event (Da im Install-Event ein alter Service Worker noch ausgeführt sein könnte)
  - self.addEventListener("activate", event => {
    event.waitUntil(
    caches.keys().then(keys => {
    return Promise.all(keys.map(key => {
    if (key !== cacheName) {
    return caches.delete(key);
    }
    }))
    }));

Fetch Event

- Event Listener für *fetch* hinzufügen
  - self.addEventListener("fetch", event => { });
- Wir verwenden die Cache-First Methode für statische Elemente

Cache First

```
function cacheFirst(request) {
    return caches.match(request).then(result => {
        if (result === undefined) {
            return fetch(request);
        }
        return result;
    });
}
```

Fetch Event abfangen

```
self.addEventListener(“fetch”, event => {
        event.respondWith(cacheFirst(event.request));
});
```

Caching für API calls implementieren

- In *fetch* die Anfrage überprüfen, ob sie ein API call ist
  - if (event.request.url.includes("/api") && [...])
- Wir möchten nur get-requests speichern
  - if ([...] && event.request.method === ‘get’)
- Da sich die Chats und Benutzer verändern können, greifen wir hier nur auf den Cache zurück, falls keine Verbindung aufgebaut werden kann
  => networkFirst

Network First - Netzwerkabfrage durchführen

```
function networkFirst(request) {
    return fetch(request)
}
```

Network First - Resultat speichern

```
function networkFirst(event) {
    return fetch(event.request).then(result => {
        const clone = result.clone();
        caches.open(messagesCacheName).then(cache => {
            cache.put(event.request, clone);
        });
        return result;
    })
```

Network First - Fallback zu Cache

```
function networkFirst(event) {
    return fetch(event.request).then(result => {
        [...]
    }).catch(() => {
        return caches.match(event.request);
    })
}
```

Caching für API calls implementieren

```
if (event.request.url.includes("/api") && event.request.method === ‘get’) {
        event.respondWith(networkFirst(event.request));
}
else if (!event.request.includes(“/api”) {
        event.respondWith(cacheFirst(event.request));
}
```

Fetch Event

```
self.addEventListener("fetch", event => {
    if (event.request.url.includes("/api") && event.request.method === ‘get’) {
            event.respondWith(networkFirst(event.request));
        }
        else if (!event.request.includes(“/api”) {
            event.respondWith(cacheFirst(event.request));
        }
});
```

“Add to Homescreen”-Funktionalität hinzufügen

- Manifest.json in root hinzufügen
- Link zu Manifest in html-files hinzufügen
- Install Event aufrufen

Manifest.json - Content


```
{"name": "Chatter",
  "short_name": "Chatter",
  "start_url": "/",
  "display": "standalone",
  "theme_color": "#f4f4f4",
  "icons": [ {
      "src": "img/logo.png",
      "sizes": "512x512",
      "type": "image/png"}]}
```

Manifest.json

- Manifest in den html-files hinzufügen:
  - <head>
    <link rel="manifest" href="manifest.json">
    </head>

In index.js

- zu beforeinstallprompt Event subscriben
  - window.addEventListener("beforeinstallprompt", event => { });
- Die PWA kann nur während einer Benutzereingabe installiert werden => deswegen müssen wir uns das Event speichern und den Install-Button sichtbar machen

index.js

```
let installPrompt;

window.addEventListener("beforeinstallprompt", event => {
    console.log("Install event called");
    event.preventDefault();
    installPrompt = event;
    document.getElementById("install_button").style.visibility = 'visible';
});
```

Auf den Install-Button reagieren

```
function install() {
    installPrompt.prompt();

    installPrompt.userChoice.then(choiceResult => {
        installPrompt = null;
    });
}

document.getElementById("install_button").onclick = install;
```

Notification

Wir zeigen eine Benachrichtigung wenn auf eine Chatnachricht gedrückt wird

Notification

- Um Berechtigung fragen - main.js
  - Notification.requestPermission().then(status => {
    console.log(`Notification permission status: ${status}`)
    })

Notification

- In der Methode chat.js/addMessage

- Überprüfen ob Berechtigung für Notifications vorhanden sind
  - if (Notification.permission == 'granted') {
- Click handler zu Nachrichten-Element hinzufügen
  - element.onclick = () => { [...] }

Notification

- Service-Worker abrufen
  - navigator.serviceWorker.getRegistration().then(registration => { [...] })
- Mit dem Service Worker die Benachrichtigung anzeigen
  - registration.showNotification(`Message from: ${message.from.userName}`, {
    body: message.text,
    icon: 'img/logo.png'
    })

Notification - chat.js/addMessage

```
if (Notification.permission == 'granted') {
        element.onclick = () => {
            navigator.serviceWorker.getRegistration().then(registration => {
                registration.showNotification(title, {
                    body: message.text,
                    icon: 'img/logo.png'
                })
            })
        }
    }
```

Progressive Web App-Demo is finished

- Weitere Tutorials und Quellen:
  - https://developers.google.com/web/ilt/pwa/
  - https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook