SvelteKit PWA

- https://dev.to/100lvlmaster/create-a-pwa-with-sveltekit-svelte-a36

/static/manifest.json

```
{
    "short_name": "svelte-pwa",
    "name": "svelte-test-pwa",
    "start_url": "/",
    "icons": [
      {
        "src": "logo_512.png",
        "type": "image/png",
        "sizes": "512x512"
      }
    ],
    "background_color": "#3367D6",
    "display": "standalone",
    "scope": "/",
    "theme_color": "#3367D6",
    "shortcuts": [
      {
        "name": "How's weather today?",
        "short_name": "Today",
        "description": "View weather information for today",
        "url": "/today?source=pwa",
        "icons": [{ "src": "/icons/logo_192.png", "sizes": "192x192" }]
      },
      {
        "name": "How's weather tomorrow?",
        "short_name": "Tomorrow",
        "description": "View weather information for tomorrow",
        "url": "/tomorrow?source=pwa",
        "icons": [{ "src": "/icons/logo_192.png", "sizes": "192x192" }]
      }
    ],
    "description": "Lofi to lofi beats, while using the scratchpad to list
down tasks and thoughts"
}
```

/src/service-worker.ts

```
/// <reference lib="webworker" />

import { build, files, timestamp } from '$service-worker';
```

```
const worker = (self as unknown) as ServiceWorkerGlobalScope;
const FILES = `cache${timestamp}`;

// `build` is an array of all the files generated by the bundler,
// `files` is an array of everything in the `static` directory
const to_cache = build.concat(files);
const staticAssets = new Set(to_cache);

worker.addEventListener('install', (event) => {
    event.waitUntil(
        caches
            .open(FILES)
            .then((cache) => cache.addAll(to_cache))
            .then(() => {
                worker.skipWaiting();
            })
    );
});

worker.addEventListener('activate', (event) => {
    event.waitUntil(
        caches.keys().then(async (keys) => {
            // delete old caches
            for (const key of keys) {
                if (key !== FILES) await caches.delete(key);
            }

            worker.clients.claim();
        })
    );
});

/**
 * Fetch the asset from the network and store it in the cache.
 * Fall back to the cache if the user is offline.
 */
async function fetchAndCache(request: Request) {
    const cache = await caches.open(`offline${timestamp}`);

    try {
        const response = await fetch(request);
        cache.put(request, response.clone());
        return response;
    } catch (err) {
        const response = await cache.match(request);
        if (response) return response;

        throw err;
    }
}
```

```
worker.addEventListener('fetch', (event) => {
    if (event.request.method !== 'GET' ||
event.request.headers.has('range')) return;

    const url = new URL(event.request.url);

    // don't try to handle e.g. data: URIs
    const isHttp = url.protocol.startsWith('http');
    const isDevServerRequest =
        url.hostname === self.location.hostname && url.port !==
self.location.port;
    const isStaticAsset = url.host === self.location.host &&
staticAssets.has(url.pathname);
    const skipBecauseUncached = event.request.cache === 'only-if-cached' &&
!isStaticAsset;

    if (isHttp && !isDevServerRequest && !skipBecauseUncached) {
        event.respondWith(
            (async () => {
                // always serve static files and bundler-generated assets
from cache.
                // if your application has other URLs with data that will
never change,
                // set this variable to true for them and they will only be
fetched once.
                const cachedAsset = isStaticAsset && (await
caches.match(event.request));

                return cachedAsset || fetchAndCache(event.request);
            })()
        );
    }
});
```

관련 문서

Plugin Backlinks: 아무 것도 없습니다.