Workshop: Service Worker

Building Offline-First Web Apps with Service Workers for your Android mobile phone

Merle Ehm

March 17th , 2017

Introduction

Aim of the Service Worker API

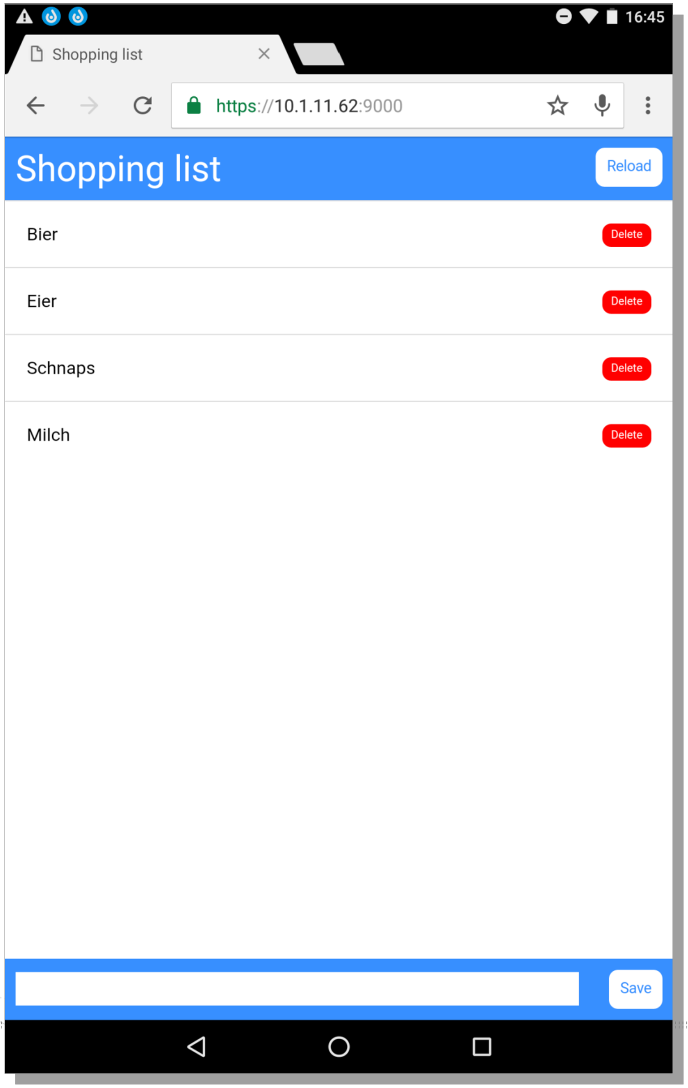
- Building effective offline experiences
- Handling server-push notifications
- Doing sync or other tasks in background (periodically or event-driven)

Technical keypoints

- Runs on a different thread to the main JavaScript (non-blocking) – has no DOM access and others
- Listens to events and gets activated – usually the thread sleeps
- Only available over https for security reasons (apart from localhost)

Project: Shopping List

Making the web app offline available!



Shopping list

https://10.1.11.62:9000

Shopping list

Reload

Bier	Delete
Eier	Delete
Schnaps	Delete
Milch	Delete

Save

ergovia

3 Steps to get effective offline experience

1. Offline first

- Verify the web app can be reached on lost connectivity

2. Updating the cache

- Verify that file changes will be detected and loaded from server

3. Background Sync

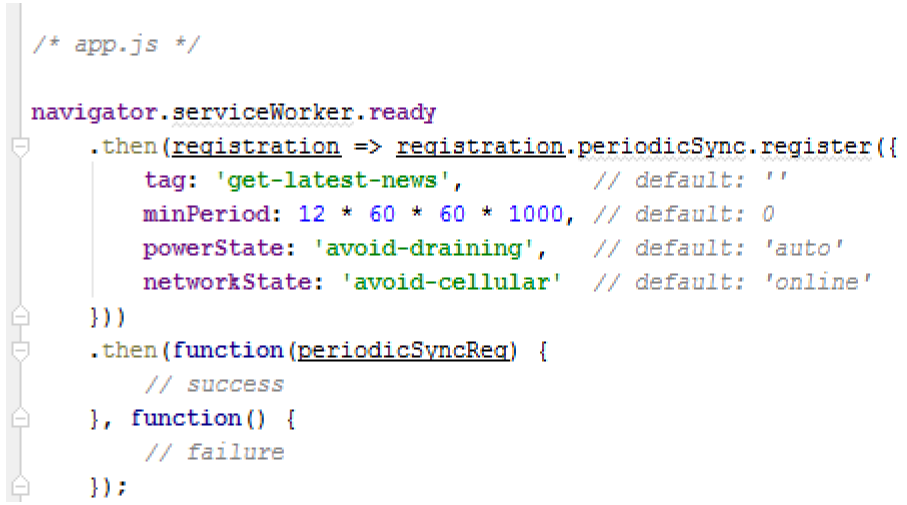
- Verify that save or delete actions are processed on reconnect to the internet

Special: Updating the cached shopping list by server push

- Verify that the shopping list is updated on the client as soon as the server has done a save or delete action

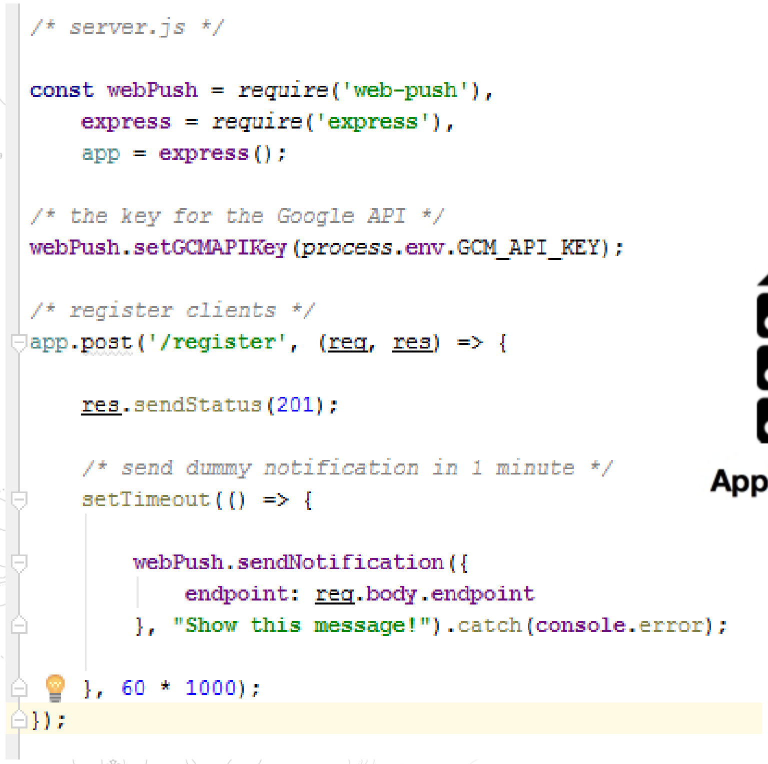
ergovia

Periodic Background Sync

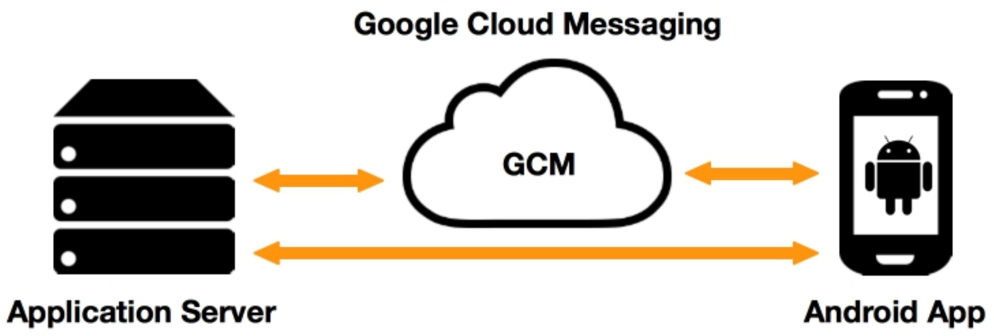


```
/* app.js */

navigator.serviceWorker.ready
    .then(registration => registration.periodicSync.register({
        tag: 'get-latest-news',          // default: ''
        minPeriod: 12 * 60 * 60 * 1000, // default: 0
        powerState: 'avoid-draining',    // default: 'auto'
        networkState: 'avoid-cellular'   // default: 'online'
    }))
    .then(function(periodicSyncReg) {
        // success
    }, function() {
        // failure
    });
```

Server-Push via Google Cloud Messaging

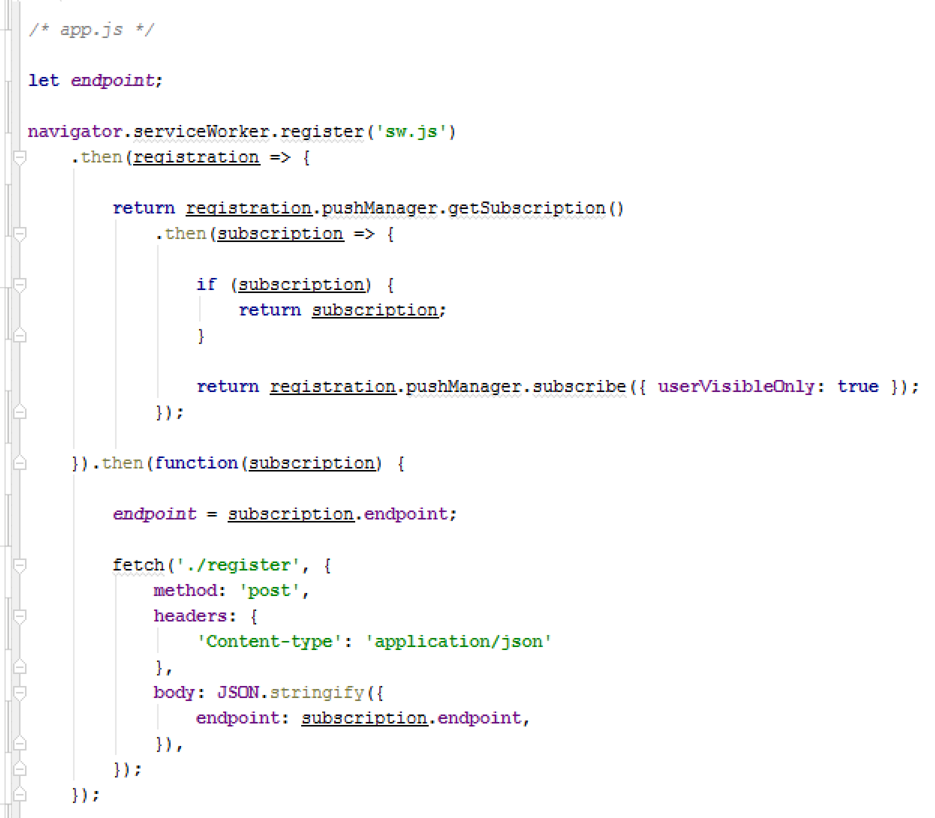


```
/* server.js */

const webPush = require('web-push'),
    express = require('express'),
    app = express();

/* the key for the Google API */
webPush.setGCMAPIKey(process.env.GCM_API_KEY);

/* register clients */
app.post('/register', (req, res) => {

    res.sendStatus(201);

    /* send dummy notification in 1 minute */
    setTimeout(() => {

        webPush.sendNotification({
            endpoint: req.body.endpoint
        }, "Show this message!").catch(console.error);

    }, 60 * 1000);
});
```

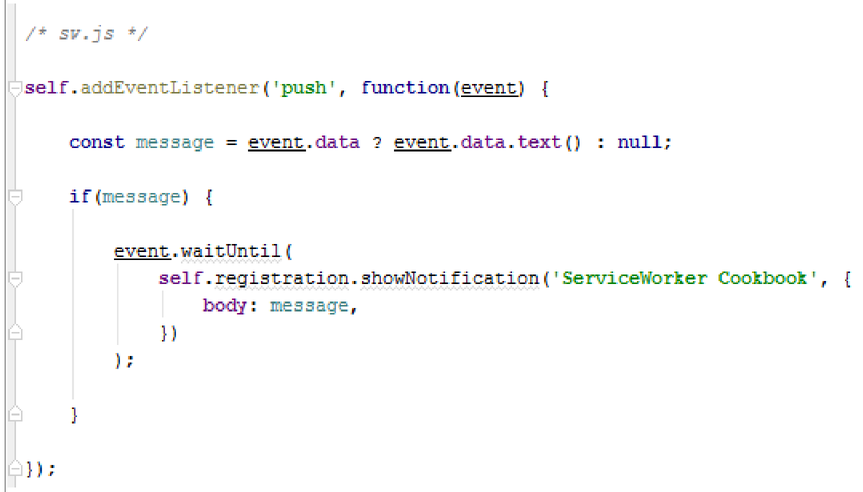


ergovia

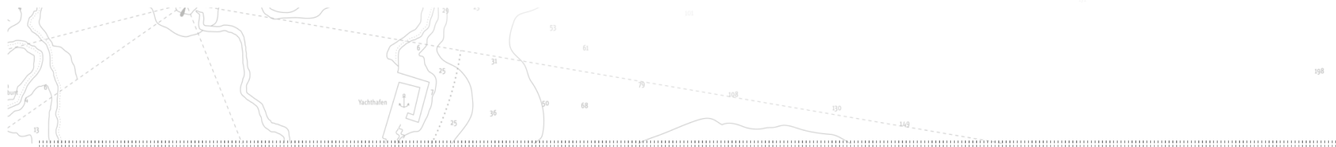
Server-push on the client side



```
/* app.js */

let endpoint;

navigator.serviceWorker.register('sw.js')
    .then(registration => {

        return registration.pushManager.getSubscription()
            .then(subscription => {

                if (subscription) {
                    return subscription;
                }

                return registration.pushManager.subscribe({ userVisibleOnly: true });
            });

    }).then(function(subscription) {

        endpoint = subscription.endpoint;

        fetch('./register', {
            method: 'post',
            headers: {
                'Content-type': 'application/json'
            },
            body: JSON.stringify({
                endpoint: subscription.endpoint,
            }),
        });
    });
```



```
/* sw.js */

self.addEventListener('push', function(event) {

    const message = event.data ? event.data.text() : null;

    if(message) {

        event.waitUntil(
            self.registration.showNotification('ServiceWorker Cookbook', {
                body: message,
            })
        );

    }

});
```



ergovia

Conclusion

What is good?

- Offers improvements for offline cache handling in comparison to the old Application Cache API – more control, more possibilities
- Background syncing is a helpful feature for apps based on local storages
- Server push notification can be the end of polling for new or changed data

What is bad?

- No support on Safari, no support on iOS
- Complicated and complex API

Conclusion

What is good?

- Offers improvements for offline cache handling in comparison to the old Application Cache API – more control, more possibilities
- Background syncing is a helpful feature for apps based on local databases
- Server push notification can be the end of polling for new or changed data

What is bad?

- No support on Safari, no support on iOS
- Complicated API