Deel 4: Push Notifications in JavaScript? Yes, you can in 12 steps!

Waar push notificaties tot voor kort nog een privilege waren voor native apps, is dat nu veranderd. Push notificaties kunnen nu direct naar een PWA verstuurd worden: net als bij een native app hoeft de browser hiervoor niet open te zijn, ook niet in de achtergrond.

Deze tutorial gaat in op het implementeren van de Push API in 12 stappen. Deze API voegen we toe aan de bestaande PWA voor het maken van 'Selfies'.

Project Setup

Als uitgangspunt voor de tutorial, ***clone*** je de volgende Github repository:

git clone https://github.com/petereijgermans11/progressive-web-app

Ga vervolgens in je terminal naar de directory:

cd progressive-web-app
cd pwa-article/pwa-app-native-features-push-api-init

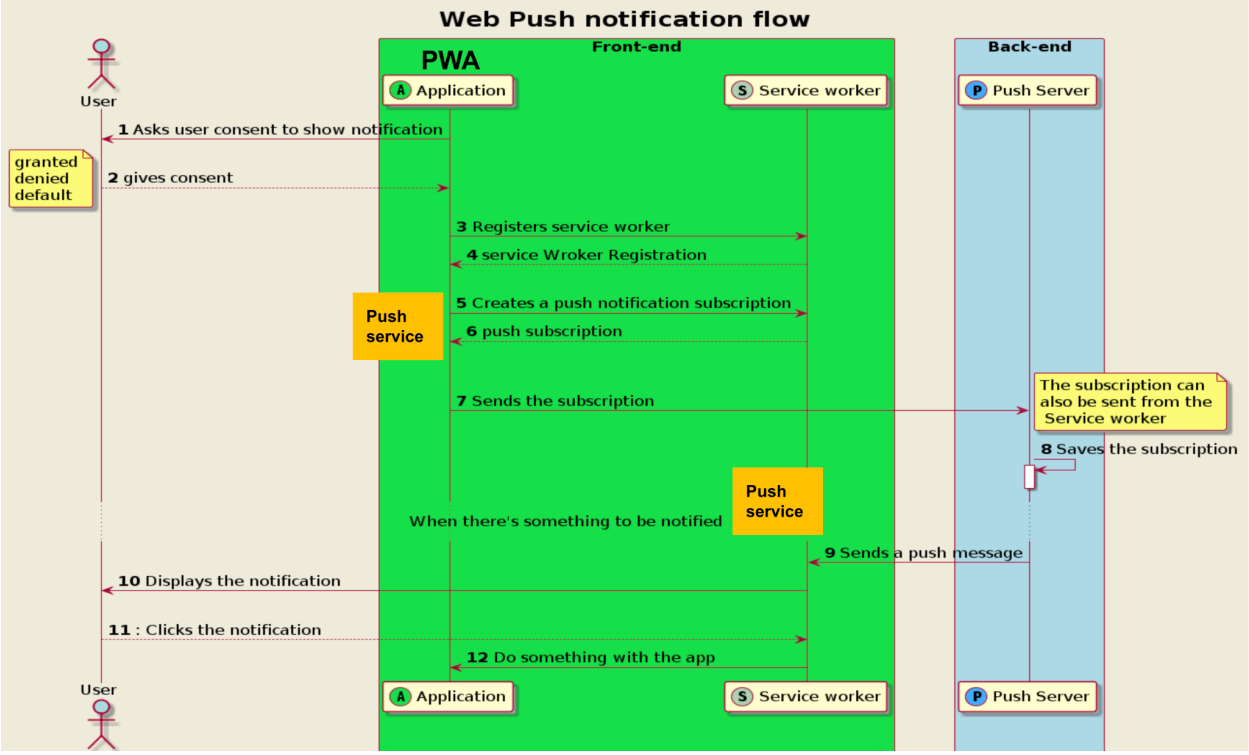
Installeer de dependencies middels: ***npm i && npm start*** en open de webapp op: **http://localhost:8080/index.html**

Push API middels JavaScript

De Web Push Notifications flow

De Web Push Notificatie is een protocol dat 4 actoren impliceert (afbeelding 1):

- de **Gebruiker**: die de notificaties wil ontvangen;
- de **Applicatie of PWA**: die draait in de browser. Browsers die de Push API ondersteunen, bevatten een *push service* die verantwoordelijk is voor het routeren van push notificaties afkomstig van de server richting de gebruiker.
- de **Service Worker**: fungeert als een proxyserver tussen de applicatie, de browser en het netwerk;
- de **Push Server of server**: die push notificaties naar de Service Worker stuurt, via de *push service*.



Afbeelding 1

1. De flow begint wanneer de applicatie de gebruiker toestemming vraagt om de push notificaties te ontvangen.

2. Zodra de gebruiker toestemming heeft gegeven, registreert de applicatie een Service Worker.

3 - 6. Wanneer de PWA de registratie van de Service Worker ontvangt, kan de PWA deze gebruiken om een **push subscription** aan te maken. Voor het aanmaken van een **push subscription** is ook een *push service* nodig. De **push subscription** bevat een *endpoint* van de push service om de push notificatie naar toe te sturen.

7. Op dit punt verzendt de applicatie de aangemaakte **push subscription** naar de **server**. De server heeft het endpoint van de push subscription nodig om push notificaties te verzenden.

8. Wanneer de server de push subscription ontvangt, wordt het opgeslagen in een database met de naam: **subscribtionsDB**.

9. De *server* stuurt een *push notificatie* naar het endpoint uit de *push subscription*. Vervolgens wordt de push notificatie via de *push service* gerouteerd naar de de Service Worker, die luistert naar het ***'push**-event'*.

10. De Service Worker stuurt de push notificatie vervolgens door naar de gebruiker.

11. Wanneer de gebruiker op de push notificatie klikt, ontvangt de Service Worker deze middels een *notificationclick-event*.

12. De Service Worker kan nu vrijwel alles doen wat hij wil met de push notificatie.

Nu je de flow kent, is het tijd om verder te gaan met de echte implementatie.

User Permissions (stap 1 en 2)

Voordat je push notificaties naar een gebruiker kunt sturen, dien je de betreffende gebruiker toestemming te vragen door een prompt weer te geven. Om ervoor te zorgen dat deze prompt wordt gestart, dien je onderstaande code toe te voegen aan de bestaande file **src/js/pushapi.js** (listing 1).

Voeg het volgende toe in src/js/pushapi.js

```
const enableNotificationsButtons = document.querySelectorAll('.enable-notifications');

const askForNotificationPermission = () => {

    Notification.requestPermission(result => {

    if (result === 'granted') {
        displayConfirmNotification();
        // configurePushSubscription();
    }
  });
};

if ('Notification' in window) {
  for (let i = 0; i < enableNotificationsButtons.length; i++) {
    enableNotificationsButtons[i].style.display = 'inline-block';
    enableNotificationsButtons[i].addEventListener('click', askForNotificationPermission);
  }
}
```

Listing 1

Plaats onderstaande code onderaan de **src/index.html,** zodat we gebruik kunnen maken van de **pushapi.js** (listing 2):

```
<script src="src/js/pushapi.js"></script>
```

Listing 2

We vragen de gebruiker om toestemming bij het klikken op de button ‘Enable Notifications’ in onze bestaande PWA (zie afbeelding 2). De knop mag alleen zichtbaar zijn als de browser push notificaties ondersteunt! In de code kun je deze controle herkennen aan de conditie: **'Notification' in *window.***

Afbeelding 2

Voeg het volgende toe in src/css/app.css (listing 3)

```
.enable-notifications {
  display: none;
}
```

Listing 3

Tonen van een Notificatie

Als eerste stap laten we een Notificatie aan de gebruiker zien (met een okay/cancel button) wanneer we toestemming krijgen (listing 4).

Voeg het volgende toe in src/js/pushapi.js

```
const displayConfirmNotification = () => {
  if ('serviceWorker' in navigator) {
    const options = {
      body: 'You successfully subscribed to our Notification service!',
      icon: 'src/images/icons/app-icon-96x96.png',
      image: 'src/images/main-image-sm.jpg',
      dir: 'ltr',
      lang: 'en-US',
      badge: 'src/images/icons/app-icon-96x96.png',
      tag: 'confirm-notification',
      actions: [
        {
          action: 'confirm',
          title: 'Okay',
          icon: 'src/images/icons/app-icon-96x96.png'
        },
        {
          action: 'cancel',
          title: 'Cancel',
          icon: 'src/images/icons/app-icon-96x96.png'
        }
      ]
```

```
    };
    navigator.serviceWorker.ready
      .then(sw => sw.showNotification('Successfully subscribed!', options));
  }
};
```

Listing 4

Middels de *navigator.serviceWorker.ready…* wordt gecheckt of de Service Worker geactiveerd is. En via de functie *sw.showNotification()* wordt er een Notificatie getoond.

Registreren Service Worker (stap 3 en 4)

Het registreren en activeren van de Service Worker is reeds beschreven in mijn eerste tutorial over PWA.

Abonneren/Subscribing op Push Notifications (stap 5)

Om een gebruiker te abonneren/subscriben op push notificaties, zijn er twee stappen nodig:

- Ten eerste het verkrijgen van toestemming van de gebruiker (stap 1 en 2)
- En het verkrijgen van een **push subscription** via de ***push service*** (stap 5).

Wat is een Push Service?

Elke browser beheert *push notificaties* via zijn eigen systeem, een zogenaamde "**push service**". Wanneer de gebruiker toestemming geeft voor push notificaties, kun je de app laten subscriben/abonneren op de *push service* van de browser. Hierdoor wordt een speciale **push subscription** gemaakt dat de "endpoint-URL" van de *push service* bevat, die voor elke browser anders is (zie listing 5). In stap 9 worden je push notificaties naar deze URLs verzonden, versleuteld met een public key. De *push service* zorgt ervoor dat de push notificatie naar de juiste cliënt verstuurd wordt.

```
{
   'endpoint': 'https://SOME.PUSHSERVICE.COM/SOMETHING-UNIQUE',
   'keys': {
      'p256dh': 'BGhFV5qx5cdOaD_XF293OqMdYSUIrMrzj2-RuzGwOTIhdW8v',
      'auth': 'HA1JEiRAp2HLuVH639Oumw'
   }
};
```

Listing 5

De **endpoint** is de *push service URL*. De ***server*** maakt gebruik van deze endpoint om een push notificatie te verzenden (stap 9).

Het **keys** object bevat de waarden die worden gebruikt om de notificatie te versleutelen.

Hoe weet de push service naar welke client de push notificatie moet worden verzonden?

De endpoint-URL bevat een unieke identificatie. Deze identificatie wordt door de push service gebruikt om de ontvangen push notificatie te routeren naar de juiste device. En wanneer de notificatie door de browser wordt verwerkt, wordt er bepaald welke *Service Worker* (stap 10) de request moet afhandelen middels een **push event**.

Application Server Keys

Voordat je een gebruiker abonneert/subscribed op een push notificatie, moet je een set "applicationServer Keys" genereren.
De applicationServer Keys, ook wel VAPID-sleutels genoemd, zijn uniek voor de server. Ze stellen een push service in staat om te weten op welke server een gebruiker heeft gesubscibed. En de keys zorgen ervoor dat het dezelfde server is die de push notificaties naar die gebruiker verstuurd.

Configureren Push Subscription

In listing 6 zie je hoe je een *push subscription* configureert en hoe je deze *PushSubscription verzendt naar de **server**.*

Let op!
Om de functie hieronder (listing 6) te activeren dien je:

1. In de functie *askForNotificationPermission()* (zie listing 1) de functieaanroep *configurePushSubscription()* te activeren
2. En tevens de functieaanroep *displayConfirmNotification()* **te verwijderen**.

Voeg toe in src/js/pushapi.js

```
const configurePushSubscription = () => {
  if ('serviceWorker' in navigator && "PushManager" in window) {
    let serviceWorkerRegistration;
     // Service worker registratie (stap 4)
    navigator.serviceWorker.ready
      .then(registration => {
        serviceWorkerRegistration = registration;
        return registration.pushManager.getSubscription();
      })
      .then(subscription => {
        if (subscription === null) {
          // Create a new Push Subscription (stap 5 en 6)
          return serviceWorkerRegistration.pushManager.subscribe({
            userVisibleOnly: true,
```

```
                    applicationServerKey: urlBase64ToUint8Array(
                        'BPg36y0YwKrMOgutw18ZeX9Ps3fBy5tNnA_OdPlor'
                    )
                });
            }
        })
        // Verzenden Push Subscription naar de server (stap 7)
        .then(pushSubscription => {
            return fetch(`${SERVER_URL}/subscriptions`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                    'Accept': 'application/json'
                },
                body: JSON.stringify(pushSubscription)
            });
        })
        .then(response => {
            if (response.ok) {
                displayConfirmNotification();
            }
        })
        .catch(error => console.log(error));
    }
};
```

Listing 6

Ten eerste wordt in bovenstaande code gechecked of 'push notificaties' en de Service Worker worden ondersteund in de browser, middels de condities: **if** (**'serviceWorker' in** ***navigator*** **&& "PushManager" in window**) { }

Als de Service Worker geregistreerd en actief is (***navigator*****.serviceWorker.ready...**), controleren we of er een push subscription aanwezig is via de aanroep: **registration.pushManager.getSubscription().**
Als het *null* is, roepen we **registration.pushManager.subscribe()** aan om een nieuwe *push subscription* te verkrijgen via de *push service*. Geef hiervoor dan de applicationServerKey mee.

Hoe Application Server Keys aanmaken?

Je kunt een public en privé set van applicationServerKey's aanmaken vanuit de **hoofdmap** van **server**. Clone hiervoor eerst de **server** via:

*git clone https://github.com/petereijgermans11/progressive-web-app-**server***

Ga via de terminal in de ***root folder*** van deze server staan en voer het volgende commando uit om de keys te genereren:

```
npm i && npm run web-push
```

1. Zorg ervoor dat je na het genereren van de *keys* de public key in je frontend vervangt, in de functie *configurePushSubscription()* in **src/js/pushapi.js**.

2. En ook dat je de public *en* private keys aan de serverkant in het **routes/routes.js**-bestand vervangt (listing 7).

```
const VAPID_PUBLIC_KEY = 'plak hier je public key';
const VAPID_PRIVATE_KEY = 'plak hier je private key';

webpush.setVapidDetails(VAPID_MAIL, VAPID_PUBLIC_KEY, VAPID_PRIVATE_KEY);
```

Listing 7

Versturen van de *Push Subscription* naar de ***server*** (stap 6 en 7)

Zodra je de gebruiker hebt gesubscribed op push notificaties en een *push subscription* hebt terugontvangen van de *push service*, wordt de *push subscription* naar de ***server*** verstuurd (zie listing 6). Op deze server sla je tenslotte deze *push subscription* op in een ***subscriptionsDb*** database.

Ontvangen van Push Subscriptions op de server (stap 8)

In de vorige stappen heb je de server reeds gekloond.
Ga vervolgens in je terminal naar de root-directory van deze server.
En installeer de dependencies en start de *server* middels: ***npm i && npm start***
De server draait op localhost:3000

In de ***progressive-web-app-server***/routes/**routes.js** staat de code al op ons te wachten om de push subscriptions te ontvangen en vast te leggen in een *node-json-db* met de naam **subscriptionsDb**. Dit is een simpele database om informatie op te slaan in een json-file. De push subscriptions komen binnen via de POST url **`/subscriptions`. (listing 8)**

```
app.post('/subscriptions', (req, res) => {
 const subscription = req.fields;
 subscription.id = uuid.v4();

 subscriptionsDb.push(`subscriptions/${subscription.id}`, subscription, false);
 res.status(200).send('subscription saved');
});
```

Listing 8

Versturen van Push Notificaties met webpush naar de gebruiker (Stap 9)

Vanaf nu kunnen we push notificaties naar de gebruiker sturen. In ons voorbeeld wordt er een push notificatie middels de ***webpush API*** naar de betreffende gebruiker verzonden, op het moment dat er een *'Selfie'* wordt ontvangen op de server. Hoe je 'Selfies' verstuurd met je PWA wordt beschreven in mijn voorgaande tutorials.

Reeds bestaande code in route.js:

```
app.post('/selfies', (req, res) => {
  const post = {title: req.fields.title, location: req.fields.location};
  const selfieFileName = path.basename(req.files.selfie.path);
  post.selfieUrl = `${req.protocol}://${req.get('host')}/images/${selfieFileName}`;
  post.id = req.fields.id;

  selfiesDb.push(`selfies/${post.id}`, post, false);

  const subscriptions = subscriptionsDb.getData('/');
  Object.values(subscriptions).forEach(subscription => {
    if (subscription.endpoint && subscription) {
      webpush.sendNotification(subscription, JSON.stringify({
        title: 'New Selfie Added!',
        content: `${post.title} @ ${post.location}`,
        imageUrl: post.selfieUrl,
        openUrl: 'help'
    })).catch(error => console.log(error));
  }
 });
 res.status(200).send({message: 'Selfie stored', id: post.id});
});
```

Listing 9

In bovenstaande code (listing 9) komen de verstuurde 'Selfies' binnen via de POST url **'/selfies'.**
Deze Selfies worden opgeslagen in een node-json-db met de naam ***selfiesDb***.
Vervolgens worden de subscriptions uit de **subscriptionsDb** gehaald, middels **subscriptionsDb.getData('/');**
Tenslotte wordt voor iedere gevonden subscription een push notificatie verzonden met de ***webpush.sendNotification().***

Ontvangen van Push Notificaties via de Service Worker (stap 10)

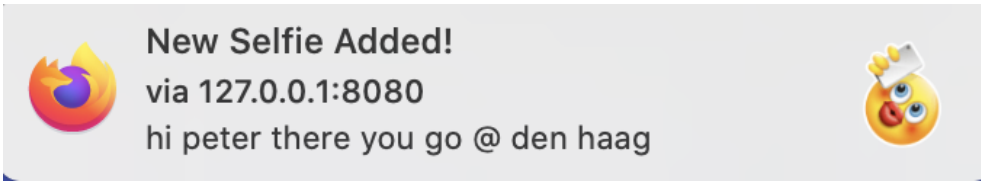
Om push notificaties te ontvangen als gebruiker, dienen we de volgende code toe te voegen in onze Service Worker (listing 10):

Voeg toe aan de bestaande sw.js

```
self.addEventListener('push', event => {
  const data = JSON.parse(event.data.text());
   const options = {
      body: data.content,
      icon: 'src/images/icons/app-icon-96x96.png',
      badge: 'src/images/icons/app-icon-96x96.png',
      data: {
         url: data.openUrl
      }
   };
   event.waitUntil(
      self.registration.showNotification(data.title, options)
   );
});
```

Listing 10

De bovenstaande code luistert naar de ***push-event*** en leest de payload van de gegevens die vanaf de *server* zijn verzonden. Met deze payload-gegevens kun je vervolgens een push notificatie weergeven in de browser (afbeelding 3) met behulp van ***self*.registration.showNotification()**.



Afbeelding 3

Testen

Om de push notificaties te testen dien je via je PWA een ‘Selfie’ te maken en te versturen. Dit kan uiteraard middels **localhost:8080**.

Maar als je wilt testen op je mobiel of tablet dan kan je ***ngrok*** gebruiken voor een publieke url. Installeer ngrok middels: **npm install -g ngrok.**

Waarom krijg je geen Push notificaties?

- Schakel push notifications aan in je browser.
- Je dient voordat je een ‘Selfie’ verstuurt, op de button ‘Enable Notifications’ (afbeelding 2) te klikken om toestemming te verlenen.
- De applicationServer Key’s zijn niet ingesteld.
- Alleen onder localhost: ‘Unregister’ handmatig je Service Worker via Chrome Dev Tools en reload je PWA.

Tenslotte

Nadat je bovenstaande stappen hebt doorlopen, dien je nog stap 11 en 12 te implementeren. Deze stappen zijn reeds geïmplementeerd in de final versie:
pwa-app-native-features-push-api-final

BIO:

Peter Eijgermans is Frontend Developer en Codesmith bij Ordina JTech. Hij deelt graag zijn kennis met anderen middels workshops en presentaties.