Ashraf Samir Ali

Push API Report

Introduction

The Push API is a web API that enables web applications to receive messages pushed from a server, even when they are not active or loaded on a user agent. This allows developers to deliver asynchronous notifications and updates to users that opt in, resulting in better engagement with timely new content.

The Push API works together with the Service Worker API and the Notification API. A service worker is a script that runs in the background and can handle events such as push messages. A notification is a message that is displayed to the user on their device.

The Push API consists of two main components: the PushManager interface and the PushSubscription interface. The PushManager interface provides methods for subscribing and unsubscribing users to receive push messages. The PushSubscription interface represents a subscription to a push service and provides properties such as the endpoint URL and methods for managing the subscription.

Prerequisites

To use the Push API, there are some requirements that need to be met:

- Browser support: The browser must support the Push API, Service Worker API, and Notification API. You can check the browser compatibility table on MDN[^1^][1] for more information.
- Service worker registration: The web application must register a service worker that can handle push events. You can learn more about service workers on MDN.
- HTTPS connection: The web application must be served over HTTPS, as push subscriptions are only available on secure contexts. You can use tools such as Let's Encrypt to obtain free SSL certificates for your website.
- User permission: The user must grant permission for the web application to show notifications and receive push messages. You can use the Notification.requestPermission() method to request permission from the user.

Subscription

To subscribe a user to receive push messages, you need to follow these steps:

- Check if the user has granted permission for notifications using Notification.permission. If not, request permission using Notification.requestPermission().
- Get access to the PushManager interface using ServiceWorkerRegistration.pushManager. This requires registering a service worker first.

- Call PushManager.subscribe() with an options object containing at least one property: userVisibleOnly set to true. This indicates that every push message will result in a notification being shown to the user.
- Handle any errors or rejections that may occur during subscription, such as network failures or permission denials.
- Store or send the subscription object returned by PushManager.subscribe() somewhere safe, such as your server or local storage. You will need this object later to send push messages.

Here is an example of how you can subscribe a user using JavaScript:

```javascript
// Register service worker
navigator.serviceWorker.register('service-worker.js')
.then(function(registration) {
  // Check notification permission
  if (Notification.permission === 'granted') {
    // Subscribe user
    return registration.pushManager.subscribe({
      userVisibleOnly: true
    });
  } else {
    // Request permission
    return Notification.requestPermission()
    .then(function(permission) {
      if (permission === 'granted') {
        // Subscribe user
        return registration.pushManager.subscribe({
          userVisibleOnly: true
        });
      } else {
        // Handle permission denial
        throw new Error('Permission denied');
      }
    });
  }
})
```

```
.then(function(subscription) {
  // Store or send subscription object
  console.log(subscription);
})
.catch(function(error) {
  // Handle errors
  console.error(error);
});
```

Sending messages

To send push messages from your server to your subscribed users, you need to do these steps:

- Retrieve or generate your application server keys using tools such as WebPush Codelab. These keys are used for encrypting and authenticating your push messages.
- Retrieve your subscribed users' subscription objects from where you stored them previously. Each subscription object contains an endpoint URL that identifies where to send your push message.
- Create your payload data containing any information you want to send along with your push message. This data must be encrypted using your application server keys according to WebPush protocol specifications.
- Make an HTTP POST request with headers containing authorization information based