

Sending push notifications from web app

- Web push notifications (JavaScript / third-party providers)
- Native iOS push notifications
- Brief history
- Support
- How does it work
- Service workers
- Lab and demo

Web push notifications

introduced by Chrome in early 2015 (42 version)

The version included “two new APIs that together allow sites to push native notifications to their users even after the page is closed—provided the user has granted explicit permission.”

The update allowed websites to send notifications just like apps could, so far.

allegedly the most critical mobile feature that was missing from the web and thus upon release witnessed a tremendous adoption by web publishers

Web push notifications brief history

2003- Blackberry OS launched push email. This saved executives and business class from constantly checking their email. Almost everyone believes that this is what that fueled Blackberry's success.

June 2009- Apple launched APN (Apple push notification). The first push notification service ever

May 2010- Google launched Android Cloud-to-device messaging (C2DM). The first notification service by Google, introduced in Android 2.2

June 2012- C2DM was replaced by GCM to overcome the limitations that C2DM had. This is important because the current delivery of push notifications is facilitated by GCM. It has been added with additional features in its new version as FCM.

April 2015- Chrome launched Chrome 42 with support for native web push notification.

Jan 2015- Firefox extended support to web push with the launch of their version 44. This was not applicable on mobile site.

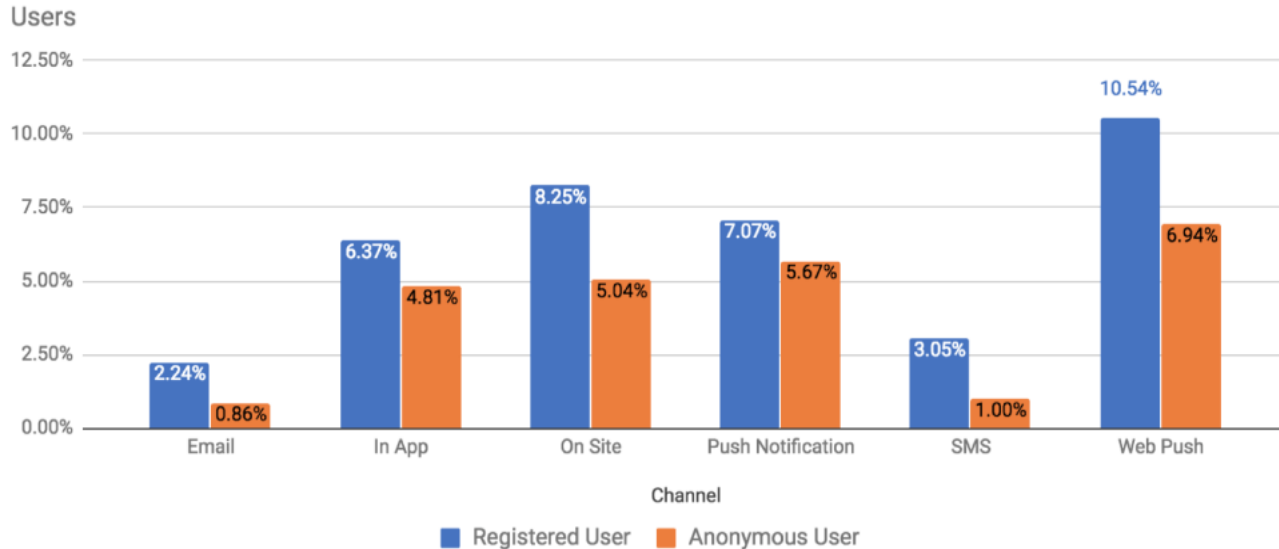
Web push notifications brief history

Aug 2016- Firefox extended support to web push even on the mobile devices.

Feb 2017- Chrome introduced rich web push notifications with Chrome 56

Web push notifications brief history

Reachability of a Web Push is much better than that of any other channel

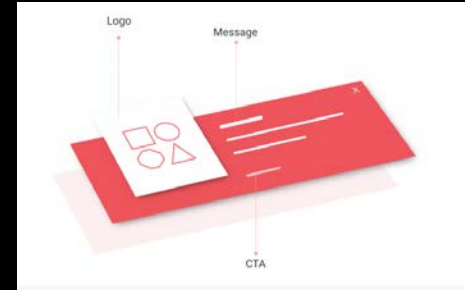


From. <https://monk.webengage.com/resources/webinar-getting-started-with-web-push-notifications/>

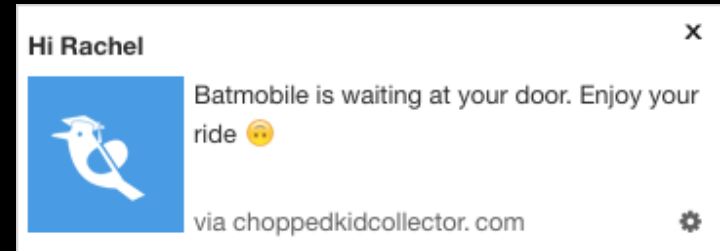
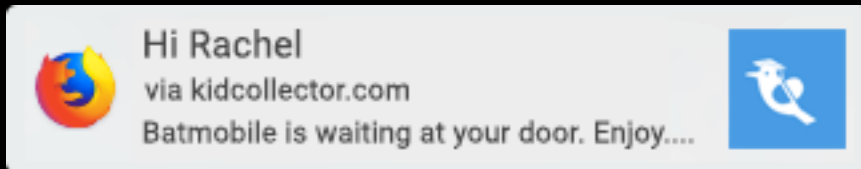
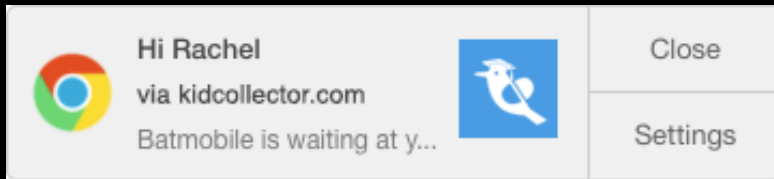
Web push notifications appearance

Appearance of a browser push notification varies

- according to the browser
- according to the OS



In the same Chrome browser, web push looks differently in Windows and Mac and likewise for Firefox.



Web push notifications support

still in a nascent stage not all browsers and OS support it yet

currently supported on desktop:

- Chrome (version 42 and above)
- Firefox (version 44 and above)
- opera (version 27 and above)

currently supported on mobile platforms:

- Android (version 4+) default browser and on Android Chrome
- for Edge - under development
- for Safari - under development

Sending push notifications from web app

Service Workers 📄 - WD

Usage

% of

Global

74.42% + 0.18% = 74.59%

Method that enables applications to take advantage of persistent background processing, including hooks to enable bootstrapping of web applications while offline.

Current aligned

Usage relative

Date relative

Show all

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			63		10.3				
		58	64	11	11.2				4
11	2 16	59	65	11.1	11.3	all	64	11.8	6.2
	17	60	66	TP					
	18	61	67						
			68						

Notes

Known issues (0)

Resources (8)

Feedback

Sending push notifications from web app

Push API 📄 - WD

Usage

% of

Global

73.27% + 1.06% = 74.33%

API to allow messages to be pushed from a server to a browser, even when the site isn't focused or even open in the browser.

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			^{1 2} 49						
			² 63		10.3				
		² 58	² 64	³ 11	11.2				4
11	16	² 59	² 65	³ 11.1	11.3	all	64	11.8	6.2
	17	² 60	² 66	³ TP					
	18	² 61	² 67						
			² 68						

Sending push notifications from web app

Using third party provider

basic subscription costing around 25-30\$/month

[Moengage](#)

[Pushcrew](#)

[Pushengage](#)

[izooto](#)

How does it work

1. Adding the client side logic to subscribe a user to push (i.e. the JavaScript and UI in your web app that registers a user to push messages):
 - generate a set of “application server keys” unique to your server
 - “subscribe” a user to push messaging
 - done in JavaScript with the Push API
 - send the PushSubscription details to backend server
 - save this subscription to a database and use it to send a push message to that user



1. Get Permission to Send Push Messages



2. Get PushSubscription



3. Send PushSubscription to Your Server

How does it work

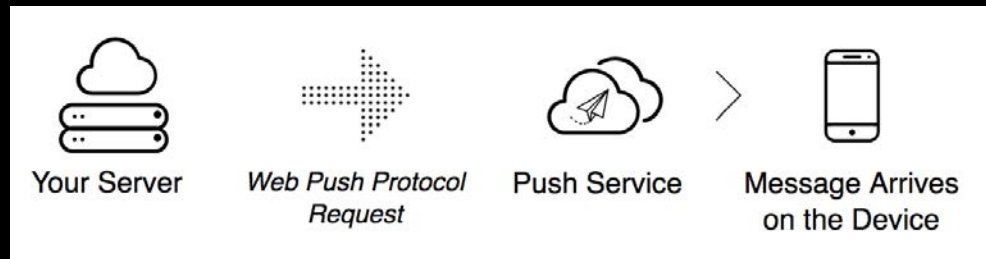
2. The API call from your back-end / application that triggers a push message to a user's device.

-make an API call to a push service:

- what data to send

- who to send the message to

- how the push service should send the message



How does it work

3. Push event on the user's device

- When the push service does deliver a message, the browser will receive the message, decrypt any data and dispatch a push event in your service worker.
- A service worker is a “special” JavaScript file. The browser can execute this JavaScript without your page being open. It can even execute this JavaScript when the browser is closed
- The service worker will receive a “push event” when the push arrives on the device to show a notification.



1. Message Arrives
on Device

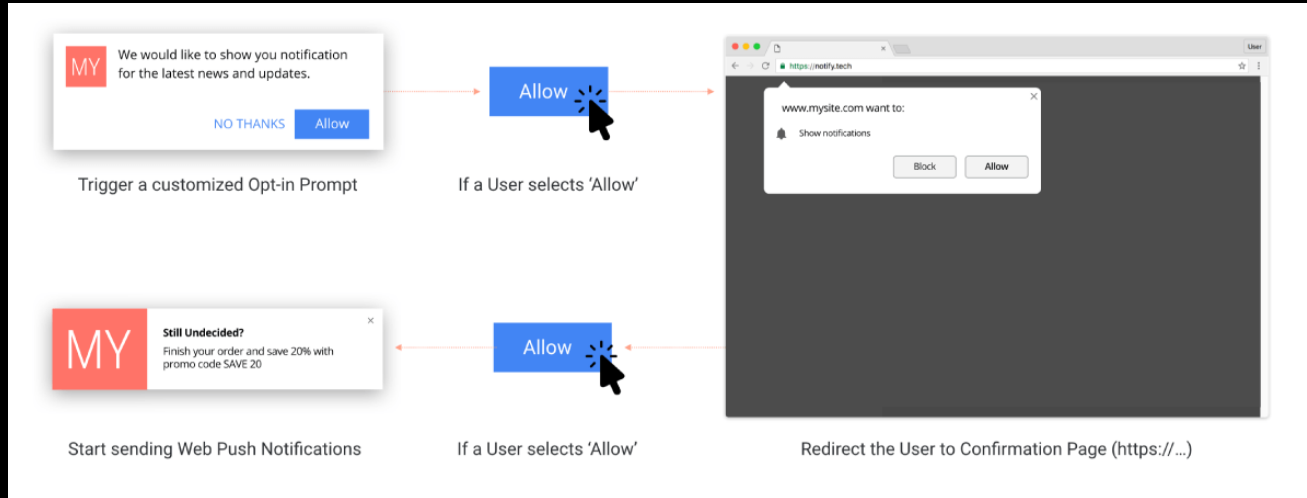


2. Browser Wakes
Up Service Worker



3. Push Event
is Dispatched

How does it work



From. <https://monk.webengage.com/resources/webinar-getting-started-with-web-push-notifications/>

Service worker

- script that runs silently in the background of a browser
- registered against an origin and path
- powerful > can be used to get into and reconstruct network connections and responses
- runs only over HTTPS due to security reasons (therefore web push too only works on https domains)
- localhost is used for development and testing (considered as a secure origin)
- A registered service worker doesn't need the webpage to be opened in order to work.
- runs in a separate thread which is non-blocking in manner
- is designed to be asynchronous
- web push relies on 'service worker' to operate

Sending push notifications to users on iPhone

As of today a web app cannot receive push notifications on any iOS browser (Safari, Chrome for iOS, ...)

iOS does not yet include an implementation of Service Workers, which are needed for push notifications to work. It is currently in development, and it is already working on Safari for OS X

Developers currently have to settle for a native implementation of the app on iOS.

Sending push notifications to users

check for installed service workers on your browser for different domains;

In Chrome:

<chrome://serviceworker-internals>

In Firefox:

<about:serviceworkers>

Sending push notifications to users

Ex. project structure:

index.html

-home screen with the form to send push notifications

index.js

-where service worker registration is initiated

manifest.json

-controls what the user sees when launching the home screen --

server.js

-server side script

service-worker.js

-adds event listeners to the service worker workspace upon registration

Source by:

<https://www.cronj.com/blog/browser-push-notifications-using-javascript/>

service-worker.js

```
"use strict";  
var url = [];  
var count = 0;  
  
self.addEventListener('install', function(event) {  
    event.waitUntil(self.skipWaiting()); //will install the service worker  
});  
  
self.addEventListener('activate', function(event) {  
    event.waitUntil(self.clients.claim()); //will activate the serviceworker  
});  
  
// Register event listener for the 'notificationclick' event.  
self.addEventListener('notificationclick', function(event) {  
    event.notification.close();
```

service-worker.js

```
event.waitUntil(  
  clients.matchAll({  
    type: "window"  
  })  
  .then(function(clientList) {  
  
    if (clients.openWindow) {  
var c = count;  
count++;  
    return clients.openWindow(url[c]);  
    }  
  })  
);  
});
```

index.js

```
var endpoint;
var key;
var authSecret;

// Register a Service Worker.
navigator.serviceWorker.register('service-worker.js')
  .then(function(registration) {
    // Use the PushManager to get the user's subscription to the push service.

    //service worker.ready will return the promise once the service worker is
    registered. This can help to get rid of

    //errors that occur while fetching subscription information before
    registration of the service worker

    return
```

index.js

```
navigator.serviceWorker.ready.then(function(serviceWorkerRegistration) {
    return serviceWorkerRegistration.pushManager.getSubscription()
        .then(function(subscription) {
            // If a subscription was found, return it.
            if (subscription) {
                return subscription;
            }

            // Otherwise, subscribe the user (userVisibleOnly allows to specify
            that we don't plan to
            // send browser push notifications that don't have a visible effect
            for the user).
            return serviceWorkerRegistration.pushManager.subscribe({
                userVisibleOnly: true
            });
        });
});
});
});
```

index.js

```
}).then(function(subscription) { //chaining the subscription promise object

  // Retrieve the user's public key.
  var rawKey = subscription.getKey ? subscription.getKey('p256dh') : '';
  key = rawKey ?
    btoa(String.fromCharCode.apply(null, new Uint8Array(rawKey))) : "";
  var rawAuthSecret = subscription.getKey ? subscription.getKey('auth') : '';
  authSecret = rawAuthSecret ?
    btoa(String.fromCharCode.apply(null, new Uint8Array(rawAuthSecret))) : "";

  endpoint = subscription.endpoint;
  // Send the subscription details to the server using the Fetch API.
  fetch('/register', {
method: 'post',
```

index.js

```
headers: {
  'Content-type': 'application/json'
},
body: JSON.stringify({
  endpoint: subscription.endpoint,
key: key,
  authSecret: authSecret,
}),
});
});
// Ask the server to send the client a notification (for testing purposes, in real
// applications the notification will be generated by some event on the server).
document.getElementById('doIt').addEventListener('click', function() {
```


index.js

```
fetch('/sendNotification', {
  method: 'post',
  headers: {
    'Content-type': 'application/json'
  },
  body: JSON.stringify({
    endpoint: endpoint,
    key: key,
    authSecret: authSecret,
    title: document.getElementById("notificationTitle").value,
    body: document.getElementById("notificationBody").value,
    icon: document.getElementById("notificationIcon").value,
    link: document.getElementById("notificationLink").value
  }),
}); });
```

manifest.json

```
{  
  "name": "Push Notifications",  
  "short_name": "push Notifications",  
  "start_url": "./index.html",  
  "display": "standalone",  
  "gcm_sender_id": "xxxxxxxxxxxx",  
  "gcm_user_visible_only": true  
}
```

index.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Service Workers -Push Notifications</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="manifest" href="manifest.json">
</head>
<body>
  <span> Title</span>
  <input type="text" id="notificationTitle" />
<br/>
  <span> body</span>
  <input type="text" id="notificationBody" />
<br/>
```

index.html

```
<span> Icon Url</span>
  <input type="url" id="notificationIcon" />
<br/>
<span> Link</span>
  <input type="url" id="notificationLink" />
<br/>
<button id="doIt">Send notification</button>
<script src="./index.js"></script>
</body>

</html>
```

server.js

```
var express = require('express');
var app = express();
var webPush = require('web-push');
var bodyParser = require('body-parser')
app.set('port', 5000);
app.use(express.static(__dirname + '/'));
app.use(bodyParser.json())

webPush.setGCMAPIKey(process.env.GCM_API_KEY);
app.post('/register', function(req, res) {
  // A real world application would store the subscription info.
  res.sendStatus(201);
});

app.post('/sendNotification', function(req, res) {
  console.log(req.body)
```

server.js

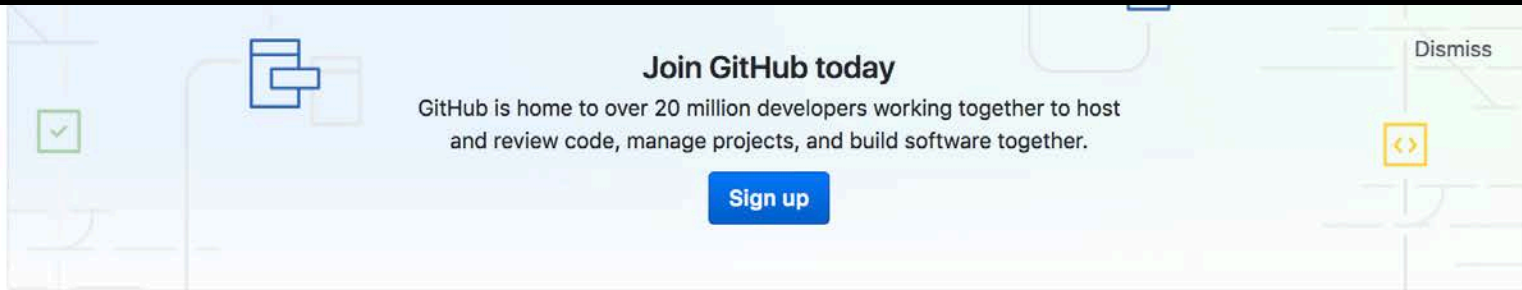
```
webPush.sendNotification(req.body.endpoint, {
  payload: JSON.stringify({
    'title': req.body.title,
    'icon': req.body.icon,
    'body': req.body.body,
    'url': req.body.link
  }),
  userPublicKey: req.body.key,
  userAuth: req.body.authSecret,
})
.then(function() {
  console.log("sent push")
res.sendStatus(201);
}, function(err) {
  console.log('webpusherr', err);
}); });
```

server.js

```
app.listen(app.get('port'), function() {  
  console.log('Node app is running on port', app.get('port'));  
});
```

Github demo

<https://github.com/gauntface/simple-push-demo>



Join GitHub today

GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.






[Sign up](#)

Dismiss

A simple example of use push notifications on the web using Service Workers <https://gauntface.github.io/simple-pu...>

🕒 297 commits 🌿 5 branches 📦 1 release 👤 14 contributors 📄 Apache-2.0

Branch: **master** ▾ [New pull request](#) [Find file](#) [Clone or download](#) ▾

 gauntface committed on Jan 2 Block safari from throwing error (#183)	Latest commit 2107a71 on Jan 2
 gulp-tasks	Tidy up (#142) 2 years ago
 project	fixing publishing script 2 years ago
 src	Block safari from throwing error (#183) 3 months ago
 test	Yoid 2 (#170) 7 months ago

References

based on the following studies and publications:

<https://monk.webengage.com/resources/webinar-getting-started-with-web-push-notifications/>

<https://monk.webengage.com/web-push-notification-guide/>

<https://monk.webengage.com/web-push-notification-guide/>

<https://monk.webengage.com/resources/webinar-getting-started-with-web-push-notifications/>

<https://www.cronj.com/blog/browser-push-notifications-using-javascript/>