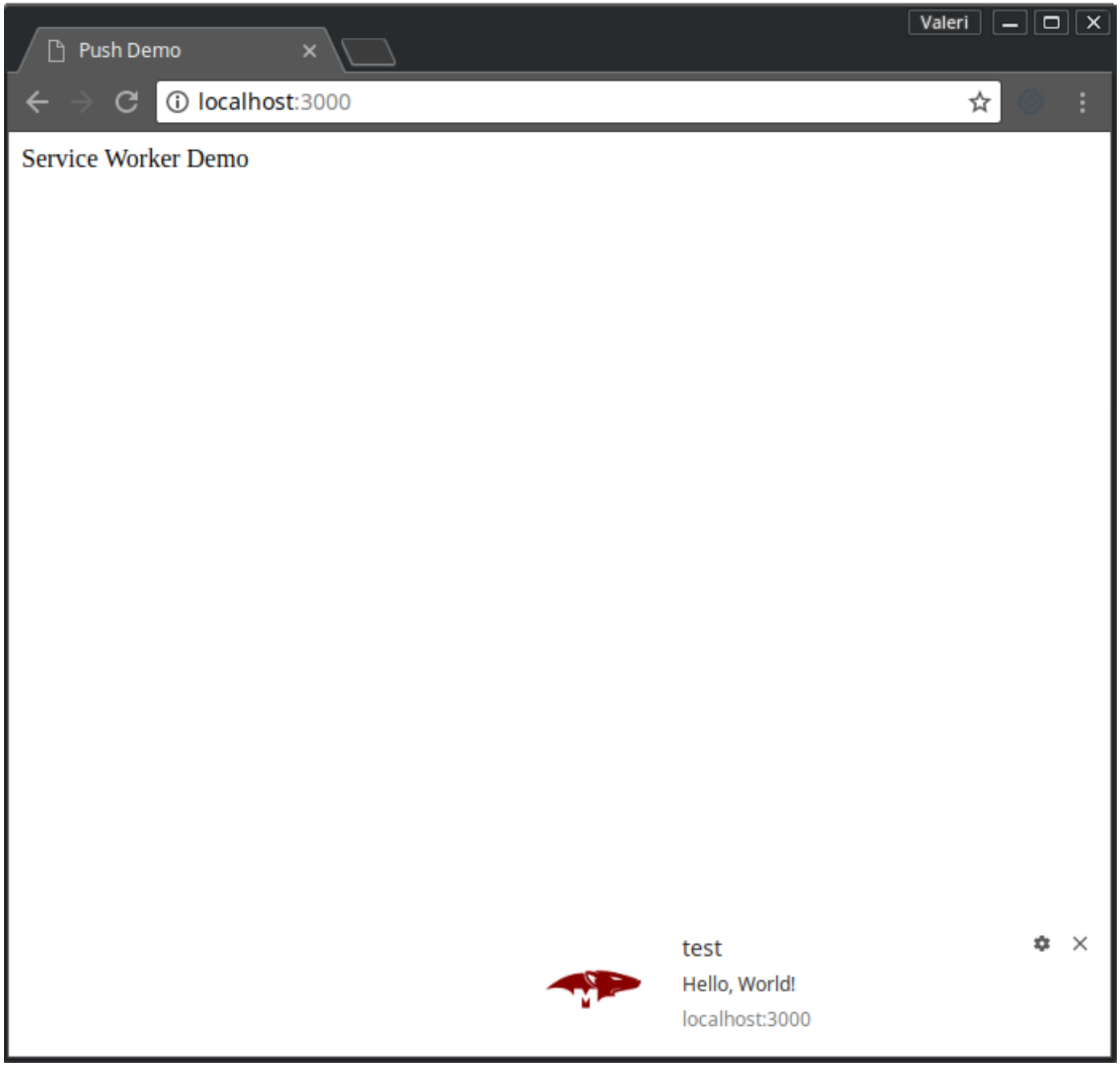
Sending Web Push Notifications from Node.js

by Valeri Karpov @code_barbarian May 08, 2018

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Using service workers, you can send push notifications to Chrome straight from your Node.js app. The excellent `web-push` npm module lets you send push notifications without going through an intermediary service like PubNub. This article will walk you through setting up a "Hello, World" example of web push notifications using a vanilla JavaScript frontend and Express on the backend. The final result will look like what you see below. The full source for this project is available on GitHub.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Credentials and Server Setup

In order to set up web push, you need to create a set of VAPID keys. VAPID keys identify who is sending the push notification. The `web-push` npm module can generate VAPID keys for you, so let's install `web-push` along with some other dependencies and use `web-push generate-vapid-keys` to create the keys.

```
$ npm install express@4.16.3 web-push@3.3.0 body-parser@1.18.2 express-static@1.2.5

+ express@4.16.3
+ web-push@3.3.0
+ body-parser@1.18.2
+ express-static@1.2.5
added 62 packages in 1.42s
$
$ ./node_modules/.bin/web-push generate-vapid-keys

=======================================

Public Key:
BOynOrGhgkj8Bfk4hsFENAQYbnqqLSigUUkCNaBsAmNuH6U9EWywR1JIdxBVQOPDbIuTaj0tVAQbczNLkC5zftw

Private Key:
<OMITTED>

=======================================

$
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In order to support older browsers you may need to also get a GCM API key, but you don't need this in desktop Chrome 63 or higher.

Next, create a file called `index.js` that will contain your server. You'll need to `require()` and configure the web-push module with your VAPID keys. In the interest of simplicity, put the VAPID keys in the `PUBLIC_VAPID_KEY` and `PRIVATE_VAPID_KEY` environment variables.

```
const webpush = require('web-push');

const publicVapidKey = process.env.PUBLIC_VAPID_KEY;
const privateVapidKey = process.env.PRIVATE_VAPID_KEY;

// Replace with your email
webpush.setVapidDetails('mailto:val@karpov.io', publicVapidKey, privateVapidKey);
```

Next, add a `/subscribe` endpoint to your Express app. Your browser JavaScript will send an HTTP request to this endpoint with a `PushSubscription` object in the request body. You need the `PushSubscription` object in order to send a push notification via `webpush.sendNotification()`.

```
const app = express();

app.use(require('body-parser').json());

app.post('/subscribe', (req, res) => {
  const subscription = req.body;
  res.status(201).json({});
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  const payload = JSON.stringify({ title: 'test' });

  console.log(subscription);

  webpush.sendNotification(subscription, payload).catch(error => {
    console.error(error.stack);
  });
});
```

That's all you need on the server side. You can find the complete source on GitHub. Now, you need to create a client `client.js` and a service worker `worker.js`.

Client and Service Worker

First, in order to serve up your static assets to the client, use the `express-static` npm module to configure your Express app to serve static files from the top-level directory. Just make sure you put this `app.use()` call **after** your `/subscribe` route handler, otherwise Express will look for a `subscribe.html` file instead of using your route handler.

```
app.use(require('express-static')('./'));
```

Next, create an `index.html` file that will serve as an entry point for your application. The only part of this file that really matters is the `<script>` tag that pulls in the client-side JavaScript, the rest is a placeholder.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
<html>
  <head>
    <title>Push Demo</title>
    <script type="application/javascript" src="/client.js"></script>
  </head>

  <body>
    Service Worker Demo
  </body>
</html>
```

Now that you have an entry point, create a JavaScript file called `client.js`. This file will be responsible for telling the browser to initialize your service worker and making the HTTP request to `/subscribe`. The below example uses async/await, because if your browser supports service workers it should support async/await as well.

```
// Hard-coded, replace with your public key
const publicVapidKey = 'BOynOrGhgkj8Bfk4hsFENAQYbnqqLSigUUkCNaBsAmNuH6U9EWywR1JIdxBVQOP

if ('serviceWorker' in navigator) {
  console.log('Registering service worker');

  run().catch(error => console.error(error));
}

async function run() {
  console.log('Registering service worker');
  const registration = await navigator.serviceWorker.
    register('/worker.js', {scope: '/'});
  console.log('Registered service worker');

  console.log('Registering push');
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  const subscription = await registration.pushManager.
    subscribe({
      userVisibleOnly: true,
      // The `urlBase64ToUint8Array()` function is the same as in
      // https://www.npmjs.com/package/web-push#using-vapid-key-for-applicationserverke
      applicationServerKey: urlBase64ToUint8Array(publicVapidKey)
    });
  console.log('Registered push');

  console.log('Sending push');
  await fetch('/subscribe', {
    method: 'POST',
    body: JSON.stringify(subscription),
    headers: {
      'content-type': 'application/json'
    }
  });
  console.log('Sent push');
}
```

Finally, you need to implement the `worker.js` file that `client.js` loads. This is where your service worker logic lives. In a service worker, you get a 'push' event when your subscription receives a push notification.

```
console.log('Loaded service worker!');

self.addEventListener('push', ev => {
  const data = ev.data.json();
  console.log('Got push', data);
  self.registration.showNotification(data.title, {
    body: 'Hello, World!',
    icon: 'http://mongoosejs.com/docs/images/mongoose5_62x30_transparent.png'
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

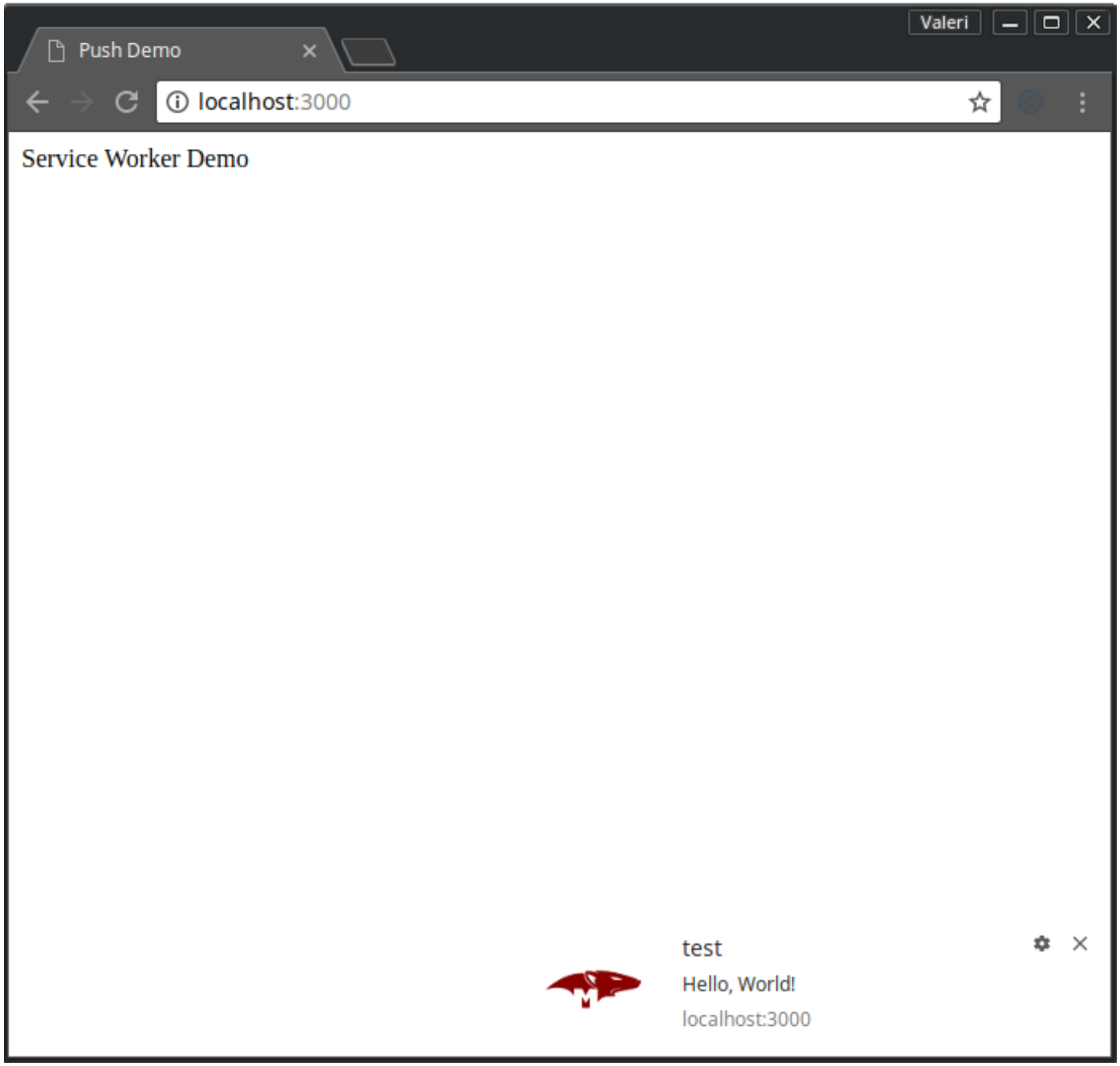
```
  });
});
```

And that's it! Start your server with the correct environment variables:

```
$ env PUBLIC_VAPID_KEY='OMITTED' env PRIVATE_VAPID_KEY='OMITTED' node .
```

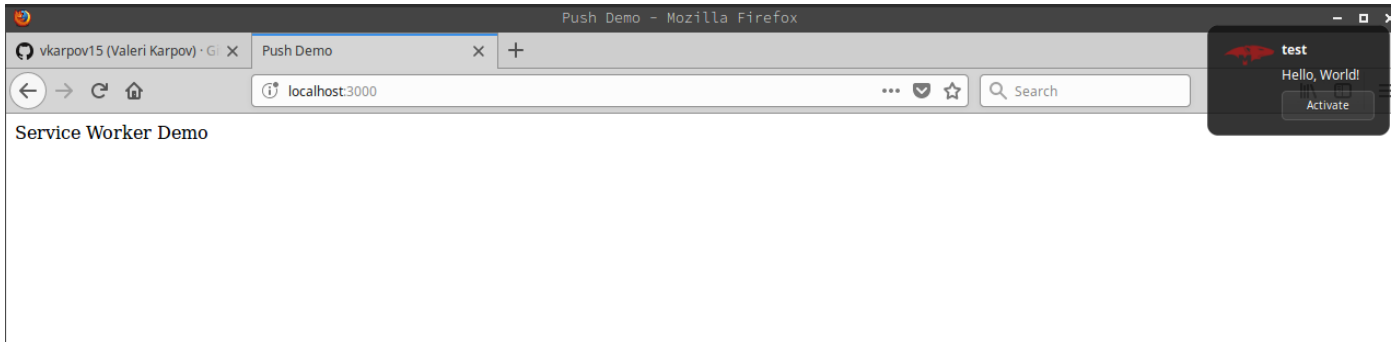
Navigate to `http://localhost:3000` in Chrome, and you should see the below push notification!

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

These notifications aren't just limited to Chrome, this same code works with Firefox as well.



Moving On

Web push is just one of numerous advantages service workers provide. With a single npm module, you can send push notifications to most modern browsers. Give service workers a shot next time you want to add push notifications to your web app!

Found a typo or error? Open up a pull request! This post is available as markdown on Github

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

0 Comments The Code Barbarian 1 Login

Recommend Tweet Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON THE CODE BARBARIAN

For vs forEach() vs for/in vs for/of in JavaScript

20 comments • 9 months ago

Avatar StoneCypher — there is only one way to iterate an array. `.map()`anyone who suggests otherwise should be drawn and quartered

Async Await Error Handling in JavaScript | www.thecodebarbarian.com

2 comments • 4 months ago

Avatar vkarpov15 — Can you please elaborate?

The 80/20 Guide to Maps in JavaScript | www.thecodebarbarian.com

4 comments • 2 months ago

Avatar Ilan Copelyn — `Object.create(null)` is worth mentioning. See https://davidwalsh.name/obj...

A Practical Guide to Symbols in JavaScript

3 comments • 3 months ago

Avatar Tony Brown — Thanks for this article. I'll have to start using Symbol more since it seems to provide a way to create private variables in

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD