Push Notifications

SESSION #15

Web Push

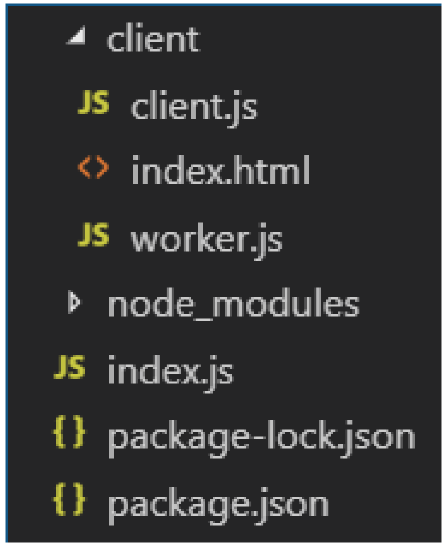
- Web push requires that push messages triggered from a backend be done via the Web Push Protocol
- If you want to send data with your push message, you must also encrypt that data according to the Message Encryption for Web Push spec

Two Technologies

- Push and notification use different, but complementary, APIs:
  - Push is invoked when a server supplies information to a service worker
    - Supported by Push API
  - A notification is the action of a service worker or web page script showing information to a user
    - Supported by Notification API

Example

- Start by preparing the server side (index.js)
- Write the client side code:
  - (client.js): Registering work service and send subscription
  - (index.html): The html file to display in the client side
  - (worker.js): The service worker needed to display the notification message



```
◢ client
   JS client.js
   <> index.html
   JS worker.js
 ▷ node_modules
 JS index.js
 {} package-lock.json
 {} package.json
```

Requirements

- const express = require('express');
- const webpush = require('web-push');
- const bodyParser = require('body-parser');
- const path = require('path');

- const app = express();

VAPID Keys

- Voluntary Application Server Identification – VAPID defines a handshake between your app server and the push service and allows the push service to confirm which site is sending messages
- The process is pretty simple:
  - Your application server creates a public/private key pair
    - The public key is given to your web app
  - When the user elects to receive pushes, add the public key to the subscribe() call's options object
  - When your app server sends a push message, include a signed JSON Web Token along with the public key

Generating VAPID Keys

- .\node_modules\.bin\web-push   generate-vapid-keys

```
Public Key:
BBx40eGK_MfJGex_Lha0SLA0-mSphh2MKJFp9Sjh9KF9iqu91S8Ho74Rqr06LDHB_23_6vdq-yMIgZYF2RP6Pyg
Private Key:
UzmzZTryWlbWGpGuBh9cmGXfXz3156pwYDXUeQ0i9g0
```

- Or, you can use the web-push node library to generate them:

```
function generateVAPIDKeys() {
    var curve = crypto.createECDH('prime256v1');
    curve.generateKeys();

    return {
        publicKey: curve.getPublicKey(),
        privateKey: curve.getPrivateKey(),
    };
}
```

Setting VAPID Keys Information

- Get the VAPID keys and pass them to setVapidDetails function

```
const publicVapidKey = 'BBx40eGK_MfJGex_Lha0SLA0-mSphh2MKJFp9Sjh9KF9iqu91S8Ho74Rqr06LDHB_23_6vdq-yMIgZYF2RP6Pyg';
const privateVapidKey = 'UzmzZTryWlbWGpGuBh9cmGXfXz3156pwYDXUeQ0i9g0';

webpush.setVapidDetails('mailto:mahmoud_elias@yahoo.com',
                        publicVapidKey,
                        privateVapidKey);
```

Serving Static Files

- Use the following code to serve static files such as images, CSS files, and JavaScript files in a directory named *client*:
  - app.use(express.static(path.join(__dirname, "client")));
  - As a result all files found in the corresponding folder (and all sub-folders) are easily accessible by:
    - localhoast:3000/client/…

Managing Subscription Route in the Server

```
app.post('/subscribe', (req, res) => {
   // Get pushSubscription object
   const subscription = req.body;

   // Create payload   : OPTIONAL
   const payload = JSON.stringify({ title: 'Push Test'} );

   // Pass object into sendNotification
   webpush.sendNotification(subscription, payload).catch(err => console.error(err));
});
```

Example – client.js

```
const publicVapidKey = 'BBx40eGK_MfJGex_Lha0SLA0-mSphh2MKJFp9Sjh9KF9iqu91S8Ho74Rqr06LDHB_23_6vdq-yMIgZYF2RP6Pyg';

// Check for service worker
if ('serviceWorker' in navigator) {
   send().catch(err => console.log(err));
}

// Register SW, Register Push, Send Push
async function send() { … }

// When using your VAPID key in your web app, you'll need to convert the URL safe base64 string
// to a Uint8Array to pass into the subscribe call
function urlBase64ToUint8Array(base64String) { … }
```

Example – client.js (continue)

```
async function send() {
   // Register Service Worker
   console.log('Registering service worker ...');
   const register = await navigator.serviceWorker.register('/worker.js', {
      scope: '/'
   });
   console.log('Service Worker Registered.');

   // Register Push
   console.log('Registering Push ...');
   const subscription = await register.pushManager.subscribe({
      userVisibleOnly: true,
      applicationServerKey: urlBase64ToUint8Array(publicVapidKey)
   });
   console.log('Push Registered.');
```

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

urlBase64ToUint8Array

- Fetch from: https://www.npmjs.com/package/web-push

```
function urlBase64ToUint8Array(base64String) {
    const padding = '='.repeat((4 - base64String.length % 4) % 4);
    const base64 = (base64String + padding)
        .replace(/-/g, '+')
        .replace(/_/g, '/');

    const rawData = window.atob(base64);
    const outputArray = new Uint8Array(rawData.length);

    for (let i = 0; i < rawData.length; ++i) {
        outputArray[i] = rawData.charCodeAt(i);
    }
    return outputArray;
}
```

Example – worker.js

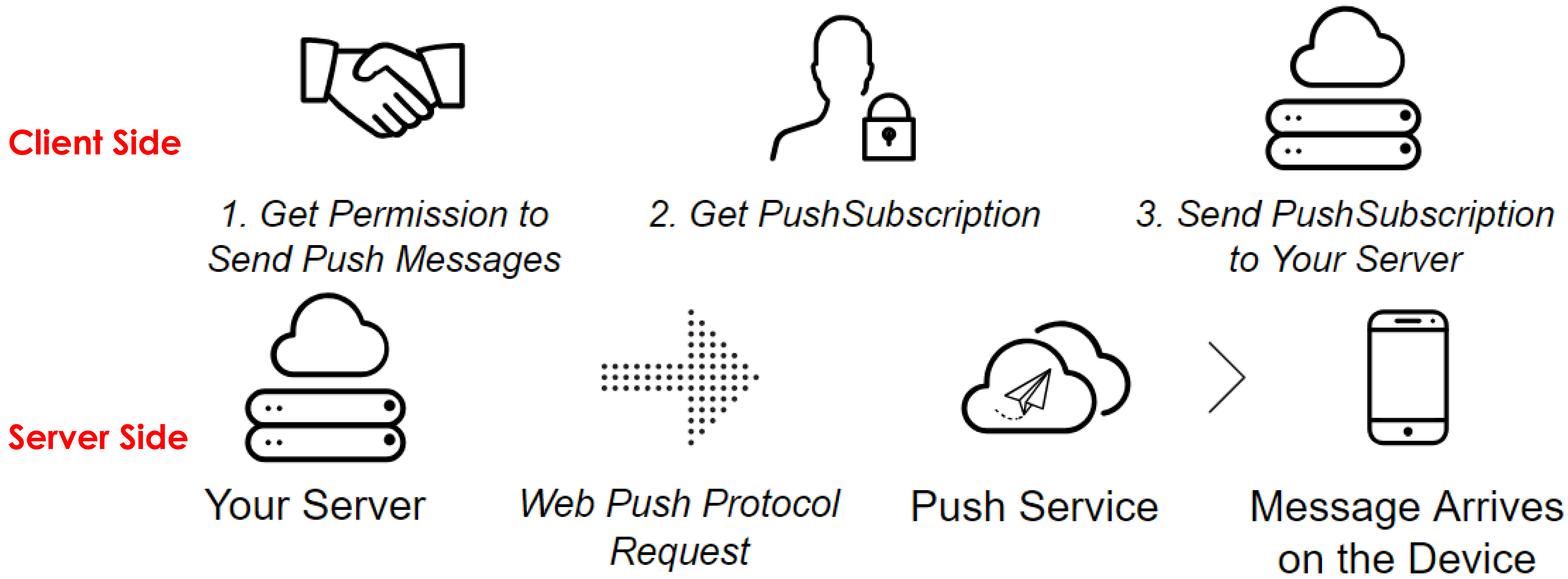
```
self.addEventListener('push', e => {
    try {
        const data = e.data.json();
        console.log('Push Received.');
        self.registration.showNotification(data.title, {
            body: 'Notified by Syriatel Company!',
            icon: 'http://www.syriatel.sy/sites/all/themes/syriatel/images/my_syriatel.png'
        });
    } catch (error) {
        console.log('Error while receiving Push data: ', error);
    }
});
```

index.html

```html
<!DOCTYPE html>
<html>
<head>
   <meta charset='utf-8'>
   <meta http-equiv='X-UA-Compatible' content='IE=edge'>
   <title>Push Notification Using Node</title>
   <meta name='viewport' content='width=device-width, initial-scale=1'>
</head>
<body>
   <h1>Push Notification Using Node</h1>

   <script src="client.js"></script>
</body>
</html>
```

Push Notifications


Client Side
1. Get Permission to Send Push Messages
2. Get PushSubscription
3. Send PushSubscription to Your Server
Server Side
Your Server
Web Push Protocol Request
Push Service
Message Arrives on the Device

Subscription Request

- A typical subscription object looks like this

```
{"endpoint": "https://fcm.googleapis.com/fcm/send/dopVR08jXfE:APA9…TFW1daL9y51gOdpP877K",
  "keys":
   {"p256dh": "BNDUTEbTuu…1tm333CFuRApMJ7S9zrMAktfF6s",
    "auth": "8mJ92tDammp-IrVXDszg_w"
   }
}
```

```
{ "endpoint" : "https://updates.push.services.mozilla.com/wpush/v2/gAAAABcsR7W…vAbN0plbsY1JQ",
  "keys" :
  {"auth": "lKAGN8Bj2QKtRNUShGF3DQ",
   "p256dh": "BHzlqYxzsy38sAIL4lkel2RF0cQEBG…_Bn4qHt_MePTiVwHDSAGH2Yxg"
  }
}
```

Subscription Request (continue)

- **endpoint**: It's a URL that contains a unique identifier
  - This identifier is used to route the message that you send to the correct device, and when processed by the browser, identifies which service worker should handle the request
- **keys**: the keys **p256dh** and **auth** are used in encryption process

Mini-Project

SESSION #16

Book Store

- Books are identified by:
  - ISBN: numeric
  - Title: alphanumeric
  - Author: alphanumeric
  - Keywords: comma separated words
  - Publisher: alphanumeric

Functions to Verify

- Server side:
  - Manage books and users: CRUD
  - Notification for new books related to a given author/keywords
- Client side:
  - Manage books (use reactive forms and animation)
  - Search for books
  - Work offline
  - Display notifications