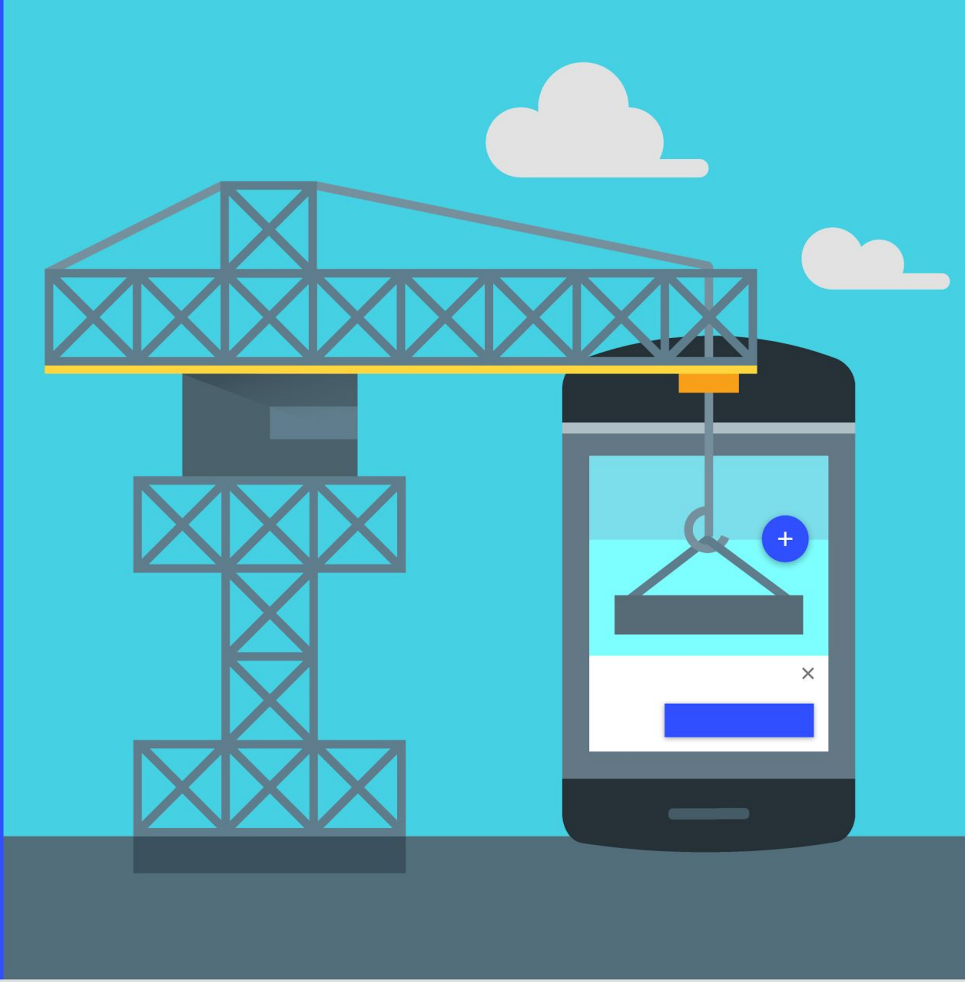
Intro to Web Push and Notifications



What we will cover

- What are Push Notifications?
- The Notification API
    - Example: Request permission
    - Example: Displaying a notification
    - Handling notification interactions in the service worker

What we will cover

- Push API
  - How it works
  - Example: Chrome and Firefox cURL’s
  - Example: Pushing from the server using web-push library
  - Example: VAPID authentication

What are Push Notifications?

What are Push Notifications?

- A **notification** is a message that pops up on the user’s device, outside of the app’s UI (i.e. the browser).
- A **push notification** is a notification created in response to a Push Message from a server
- Push notifications are assembled using two APIs: the Notification API and the Push API

The Notification API

The Notification API

1. Allows developers to display notifications to the user
2. API split into two areas: the Invocation API and Interaction API

Request permission

```
Notification.requestPermission(function(status) {
    console.log('Notification permission
status:', status);
});
```

Invocation API

We can:

- Display a notification
  - Add notification options
  - Add notification actions

Display a notification

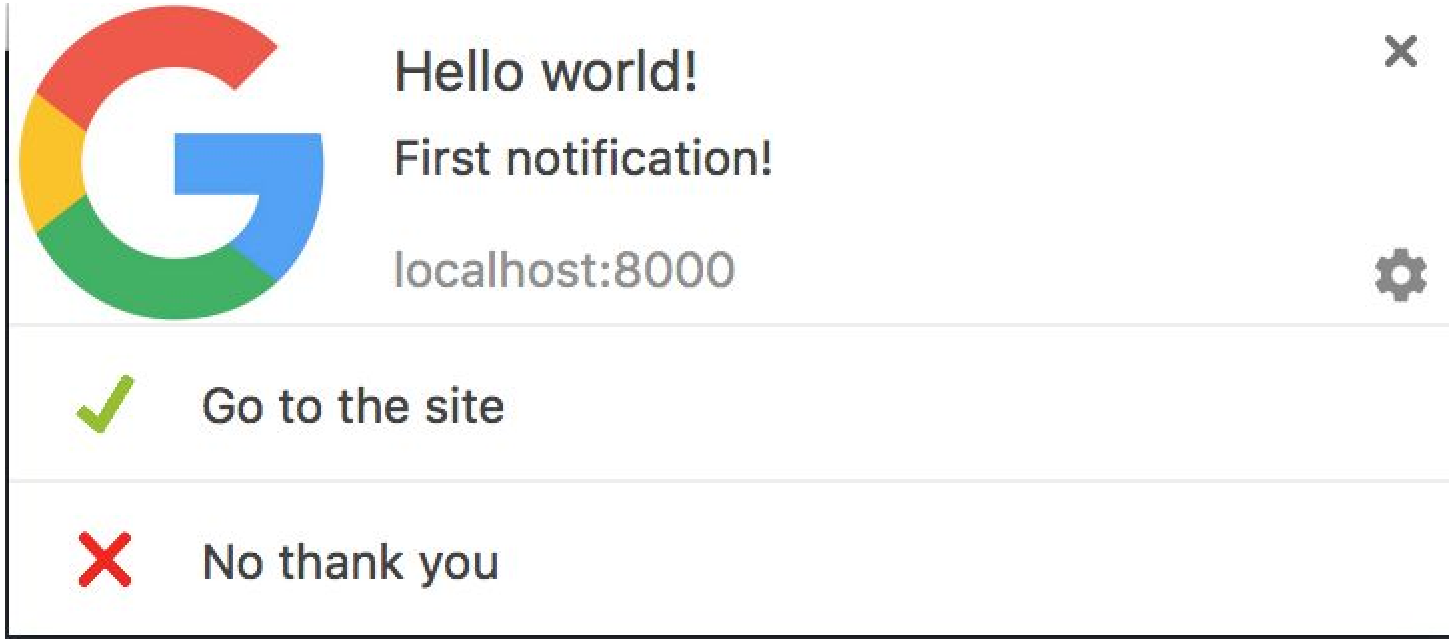
```
function displayNotification() {
  if (Notification.permission == 'granted') {
navigator.serviceWorker.getRegistration()
    .then(function(reg) {
      reg.showNotification('Hello world!');
    });
  }
}
```

Add notification options

```
var options = {
  body: 'Here is a notification body!',
  icon: 'images/example.png',
  vibrate: [100, 50, 100],
  data: {
    dateOfArrival: Date.now(),
    primaryKey: 1
  }
};
reg.showNotification('Hello world!', options);
```

Add notification actions

```
var options = {
  body: 'Here is a notification body!',
  actions: [
    {action: 'explore', title: 'Open the site!',
      icon: 'images/checkmark.png'},
    {action: 'close', title: 'Go away!',
      icon: 'images/xmark.png'},
  ]
};
reg.showNotification('Hello world!', options);
```


Hello world!

First notification!

localhost:8000

Interaction API

There are two notification interactions we can listen for in the service worker:

- notificationclose
- notificationclick
    - Handle action

notificationclose

```
self.addEventListener('notificationclose',
function(event) {
  var notification = event.notification;
  var primaryKey = notification.data.primaryKey;
  console.log('Closed notification: ' + primaryKey);
});
```

notificationclick

```
self.addEventListener('notificationclick',
function(event) {
  var notification = event.notification;
  var action = event.action;
  if (action === 'close') {
    notification.close();
  } else {
    clients.openWindow('http://www.google.com');
    notification.close();
  }
});
```

The Push API

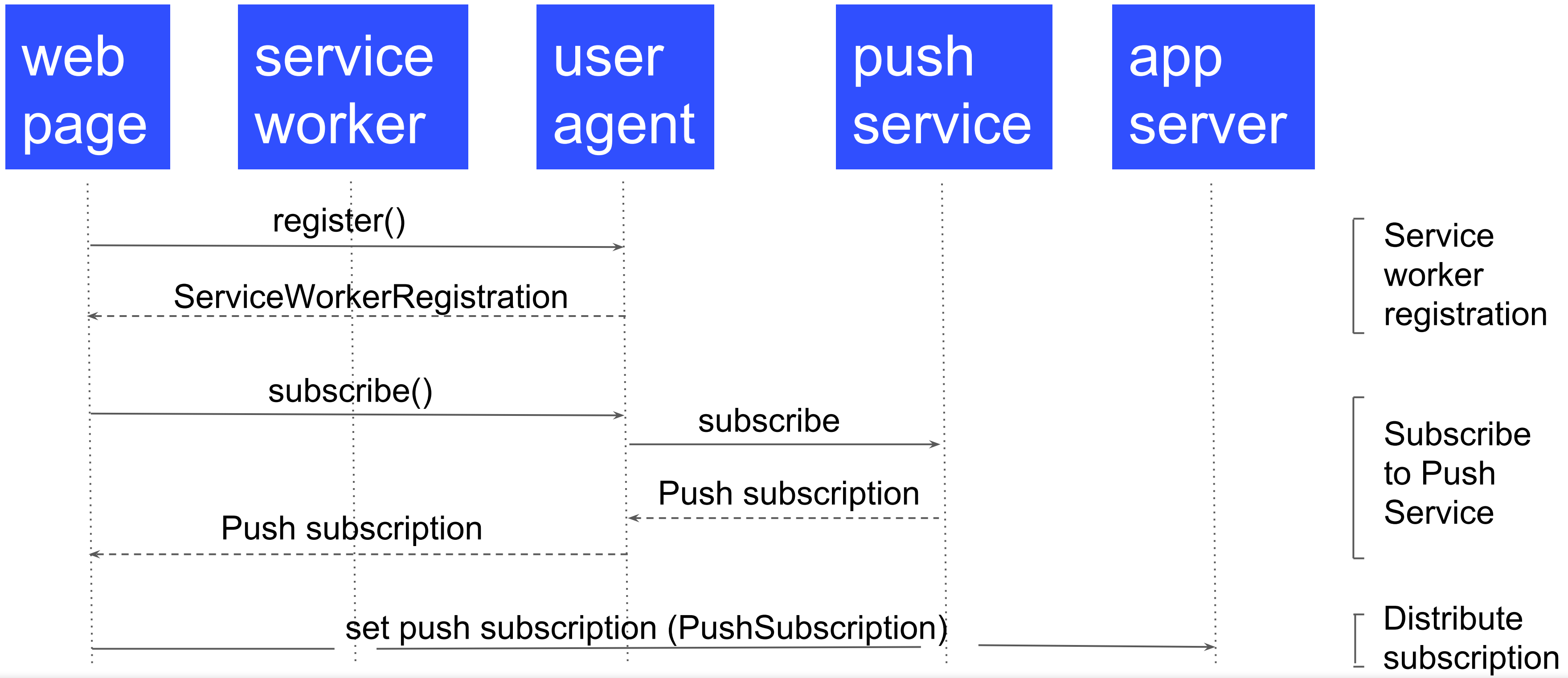
The Push API

An interface that gives service workers the ability to receive Push Messages

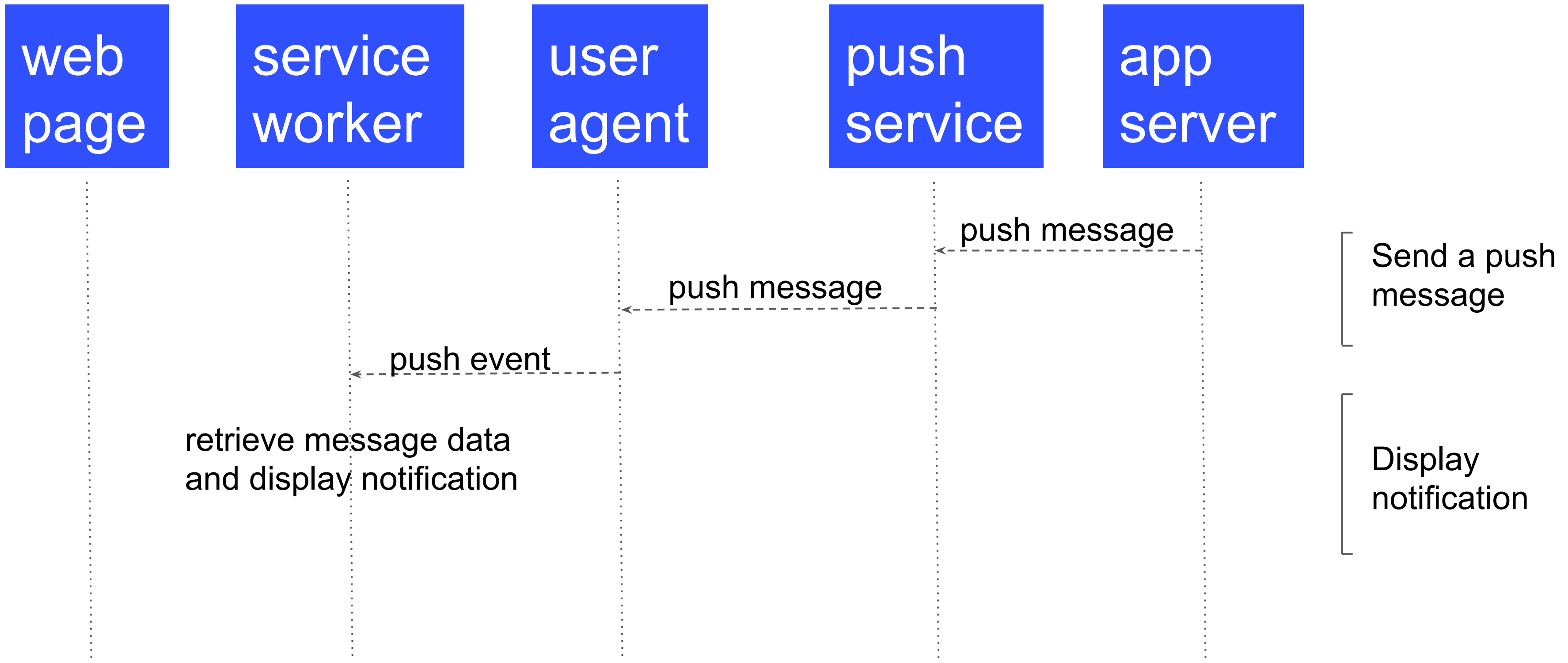
How it works

1. Subscribe to “push service”
2. Get subscription object and save to server
3. Send push messages to “endpoint URL” encrypted with the public key
4. Receive the push event in the service worker
5. Show a notification

Subscribe to the push service



Send a push notification



Create a project on Firebase

1. In the Firebase console, select **Create New Project**.
2. Supply a project name and click **Create Project**.
3. Select the gear icon next to your project name at top left, and select **Project Settings**.
4. Select the **Cloud Messaging** tab. You can find your server key and sender ID in this page. Save these values.

Check if we have subscription object

```
navigator.serviceWorker.ready
.then(function(reg) {
reg.pushManager.getSubscription()
.then(function(sub) {
    if (sub == undefined) {
      // ask user to register for Push
    } else {
      // We have subscription, update database
      console.log('Subscription object: ', sub);
    }
  });
});
```

Subscribe to push service

```
navigator.serviceWorker.register('sw.js')
.then(function(reg) {
  reg.pushManager.subscribe({
    userVisibleOnly: true
  }).then(function(sub) {
    // send sub.toJSON() to server
  });
}).catch(function(err) {
  console.log('Registration failed: ', err);
});
```

The subscription object

```
{"endpoint":"https://android.googleapis.com/gcm/send/f1Lsxk
KphfQ:APA91bFUx7ja4BK4JVrNgVjpg1cs9lGSGI6IMNL4mQ3Xe6mDGxvt_
C_gItKYJI9CAx5i_Ss6cmDxdWZoLyhS2RJhkcv7LeE6hkiOsK6oBzbyifvK
CdUYU7ADIRBiYNxIVpLIYeZ8kq_A",

"keys":{"p256dh":"BLc4xRzKlKORKWlbdgFaBrrPK3ydWAHo4M0gs0i1o
EKgPpWC5cW8OCzVrOQRv-1npXRWk8udnW3oYhIO4475rds=",

"auth":"5I2Bu2oKdyy9CwL8QVF0NQ=="}}
```

Send a message to Chrome using cURL

```
curl "https://android.googleapis.com/gcm/send"

--request POST

--header "Authorization:
key=AIzaSyBVImB3hJJ76mIlpcXfMX8J5D4xnFo2fFI"

--header "Content-Type: application/json"

-d "{\"to\":\"fLSFAvG3aCE:APA9…….Wzse8bE\"}"
```

Send a message to Firefox using cURL

```
curl
"https://updates.push.services.mozilla.com/wpush/v1/gAAAAABXrNWm
1D6n_JJ.......Og328Ouc_53"
--request POST --header "TTL: 60" --header "Content-Length: 0"
--header "Authorization: Bearer eyJ0eXAiOiJKV1QiL.....aAwTKj-mYxw"
--header "Crypto-Key:
p256ecdsa=BDd3_hVL9fZi9Ybo2UUzA......IiBHXRdJI2Qhumhf6_LFTeZaNn
dIo"
```

Send a message from the server

```
var webPush = require('web-push');
webPush.setGCMAPIKey('AIzaSyBVImB3hJJ...8J5D4xnFo2fFI'
);
webPush.sendNotification(subscription.endpoint, {
  userPublicKey: subscription.keys.p256dh,
  userAuth: subscription.keys.auth,
  payload: JSON.stringify({
    'message': 'Hello world!'
  })
});
```

What is VAPID?

- Voluntary Application Server Identification for Web Push (VAPID) protocol is an optional method to identify your service
- VAPID uses JSON Web Tokens (JWT) to carry identifying information
- A JWT contains a three properties called a Claim. The claim has:
    - Audience attribute
    - Subscriber property
    - Expiration time value

Identify your app with VAPID Auth

```
var serviceKeys = webPush.generateVAPIDKeys();
webPush.sendNotification(subscription.endpoint, {
    userPublicKey: subscription.keys.p256dh,
    userAuth: subscription.keys.auth,
    vapid: {
      subject: 'email@example.com',
      publicKey: serviceKeys.publicKey,
      privateKey: serviceKeys.privateKey
    },
    payload: JSON.stringify({
      'message': 'Hello world!'
    })
  });
```

Handle the “push” event

```
self.addEventListener('push', function(e) {
  var data = e.data.json();
  var options = {
    body: 'Here is a notification',
  };
  e.waitUntil(
self.registration.showNotification(data.message, options)
  );
});
```