# This website would like to show you notifications…

Web Push notifications in Grails

# Who am I?

molecular biologist
turned software
developer

# Who are you?

Why did you chose to come to this workshop?

# Why use web notifications?

# Mobile Apps use push notifications

# Boosted conversions (Jumia, 9x)

# Great mechanism for internal apps as well (long processing)

# Whenever you want your user to come back to your app after a while

# What are we going to do today?

1. Setup example project
2. Intro to Notification and related APIs
3. Local (non-push) Notifications
4. Notification API in detail
5. Service workers and their relation to Notifications
6. Implementing push web notification in a Grails App

# Example project

- clone from https://bit.ly/2LQXf0w
- import to IDE
- do a test run

# Promises

# (aka CompletableFutures)

# Notification API

# Notification API

1. `window.Notification`
2. The difference between local and push notifications
3. Browser Support
4. Requirements

# Local notifications

# Detecting support in the browser (exercise)

# Asking user for permission (exercise)

# Permission UX

# Notification API review

# Title and body (exercise)

# Visual options

- icon
- image

# Actions

- action
- title
- icon
- Notification.maxActions

# Grouping notifications

- tag
- renotify

# Additional behaviour

- silent (no vibration, sound or screen wakeup – relevant on mobile)
- requireInteraction (doesn’t hide unless user interacts – relevant on desktop)

# Additional behaviour

- badge (mobile)
- vibrate (mobile)
- sound (unimplemented?)
- timestamp

# Service worker

# Service Worker

- Cornerstone of PWAs
- Available even on iOS (but limited)
- Required for push notifications, since it’s event target

# Web Push Protocol

Your Server

*Web Push Protocol Request*

Push Service

Message Arrives on the Device

1 *Sign Authorization Header with Private Key*

Hi + =

Your Server

2 Send Message to http://endpoint.io/ID1

4 201 OK Response

3 *Decrypt Authorization Header*

+ = Hi

http://endpoint.io/ID1

Push Service

5 Message Sent to Device

Public Key

Private Key

# Two keypairs

# Two key pairs

- Server (generated by you):
  - Private VAPID key
  - Public VAPID key
- Browser:
  - auth and p256dh (available via `subscription` object)
  - private (kept by browser and generally unavailable to developer)

# web-push library

# web-push library

- add dependency:
  `compile "nl.martijndwars:web-push:3.1.1"`

- make sure BouncyCastle is installed as Security Provider
  `Security.addProvider(new BouncyCastleProvider())`

- create a `PushService` with VAPID keys

- send notifications

- …

- profit!

# Feature detection (exercise)

# Configuring Grails (need to have sw.js in root)

# Configuring grails for SW

- In `UrlMappings.groovy`:

```
static excludes =  [
    '/sw.js'
]
```

- In `application.yml`:

```
grails:
    resources:
        pattern: ,/**'
```

# Registering SW (exercise)

# VAPID generator in Groovy console

https://gitlab.com/snippets/1735762

# Get subscription object from PushManager

# Send subscription back to the backend

# Send push notifications to clients!

# Receive notification in ServiceWorker

# Common patterns

# Common patterns

- Send analytics when notification is dismissed:

```
self.addEventListener('notificationclose', function(event) {
   const dismissedNotification = event.notification;

   const promiseChain = notificationCloseAnalytics();
   event.waitUntil(promiseChain);
});
```

# Common patterns

- Send analytics when notification is dismissed
- Open a new window
- Focus an existing window
- Merge notifications with `tag` option
- Message existing page from push event