Push App with Service Worker + Manifest + Firebase

개요

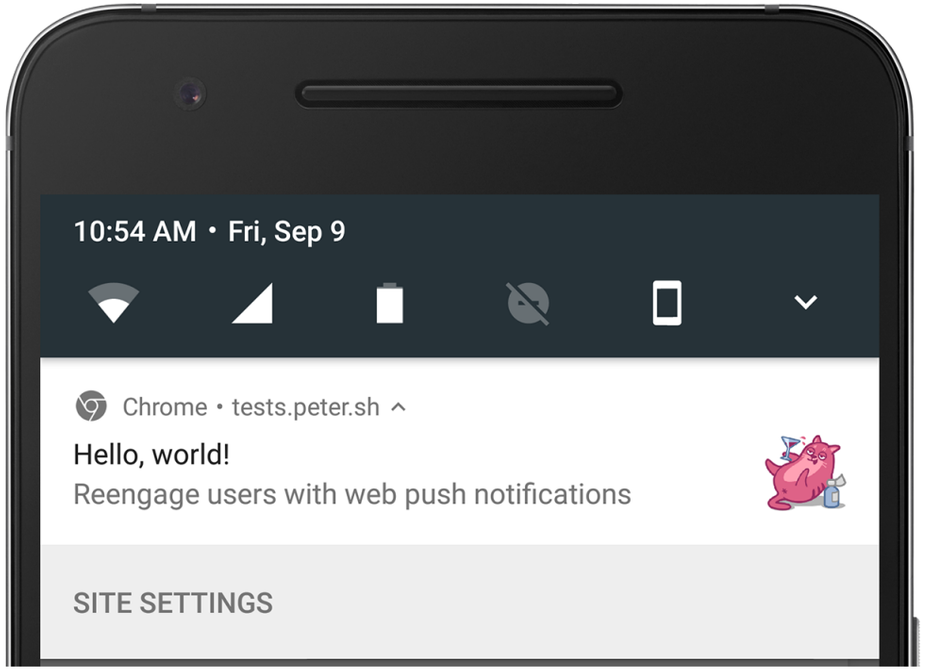
- 앞에서 배운 Service Worker 를 활용하여 Push Messaging 구현방법 학습
- Firebase Cloud Messaging 서비스를 활용하기 위한 기본절차 학습
- Push 메시지 수신이 가능한 간단한 앱을 구현 및 호스팅 실습

목차

- PWA 의 Push API & Notification API 소개
- Push 알람 사례 및 활용 방안
- 일반적인 모바일 Push 알람 구조
- PWA 의 Push 알람 구조
- Firebase Cloud Messaging 서비스 소개
- 구현할 샘플 서비스의 구조
- 구현절차 - 실습

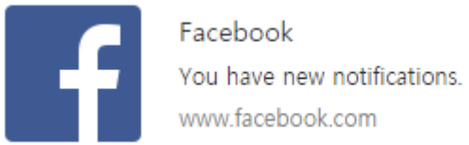
PWA 의 Push API & Notification API 소개

- 모바일에서만 가능했던 Push 알람 기능을 API 로 간단하게 구현 가능
- 브라우저 기반 Push 알람. Windows & Mac & Android 에서 동작
- 구현하기 위해서는 Service Worker 를 필수로 구현해야 한다.



Push 알람 사례 및 활용 방안

- Facebook 새 게시글 알람

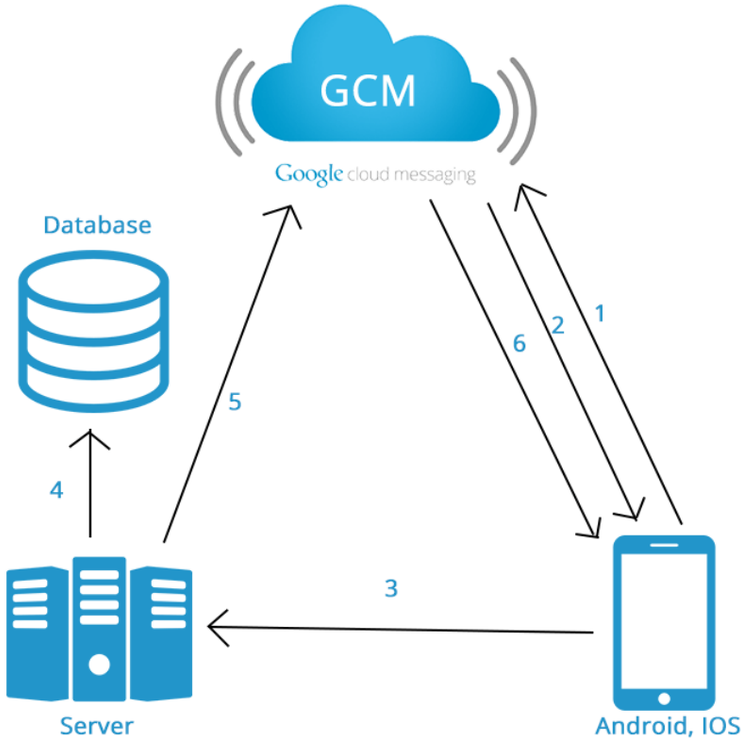


- Twitter Lite 소개 영상. Case Study
- Alibaba Case Studies
- Wego (PWA + AMP + Web Payment + Credential API)

일반적인 Mobile Push 알람 구조

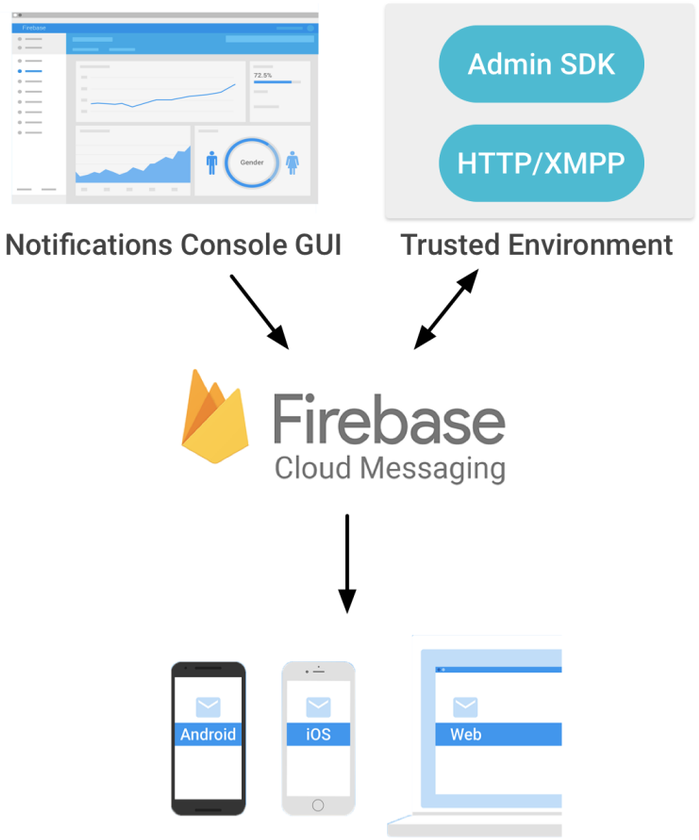
- Android : GCM (Google Cloud Messaging) & FCM
- iOS : APNS (Apple Push Notification Service) & FCM

Android Google Cloud Messaging Architecture



PWA 의 Push 알람 구조

- FCM : GCM 의 최신버전으로 Mobile + Web 모두 지원



Firebase Cloud Messaging 서비스 소개

- Mobile & Web Push 알람을 위한 콘솔창 제공
- Push 등록된 기기에 대한 정보 및 알림 전송 기능
- 보낸 Push 메시지에 대한 분석 기능 제공
- 가능한 프로토콜 HTTP & XMPP (미들웨어 메시징 프로토콜)

오늘 구현할 실습 예제 설명

- 서비스워커를 지원하는 단말기의 키를 획득하여 DB에 저장하고
- DB 에 저장된 키로 해당 단말기에 Push 메시지를 보내 수신하는 예제

완성된 예제 시연

실습 - 구성파일

- Push 알림 허용 버튼을 가진 html 파일 1개
- 해당 페이지 스타일링을 위한 css 파일 1개
- 서비스워커 등록 및 앱 로직을 위한 js 파일 1개
- 푸시 알람을 처리할 js 파일 1개

실습 - 동작방식

- 웹 사이트 접속 후 서비스워커 등록
- Push 허용 버튼 활성화
- 획득한 브라우저 key 와 FCM 프로젝트 key 조합하여 Push 요청
- Push 메시지 수신 및 알람 처리

실습 - 준비사항

- Firebase 프로젝트 생성
- Server Key in Firebase Cloud Messaging Tab
- Sender ID in Firebase Cloud Messaging Tab

1. Push **전송을 위한** Firebase **프로젝트** Server Key **획득**

- Firebase console 에서 해당 프로젝트의 설정으로 이동
- 클라우드 메시징의 Server Key 활용

2. 기기별 Push Key 획득을 위한 Sender ID 등록

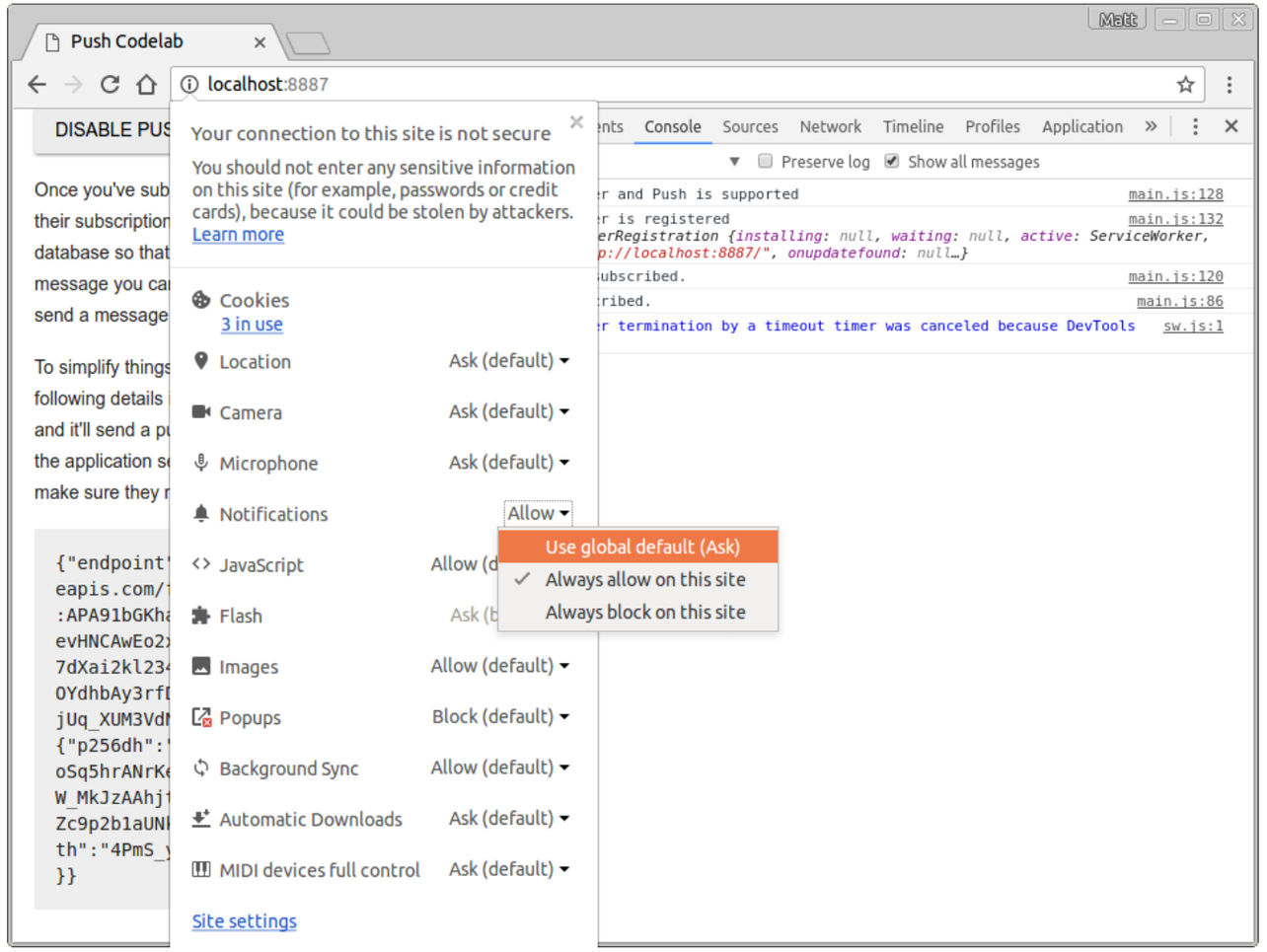
- Firebase 초기 config 에서 messagingSenderId 값을 복사

```
var config = {
  // ...
  messagingSenderId: "800635767370"
};
```

- App Manifest Json 파일 생성 후 설정 및 gcm_sender_id 속성 추가

```
"gcm_sender_id": "800635767370"
```

참고 #1 : 웹 사이트의 Push 알림 설정 확인



참고 #2 : Subscribe 시 기존 예제의 public key 제외

```
swRegistration.pushManager.subscribe({
    userVisibleOnly: true,
    // 아래는 public 키 생성 예제 사이트를 참고할 때 필요
    // applicationServerKey: applicationServerKey
})
```

실습 - 구현절차

프로젝트 다운로드 및 Firebase 프로젝트 구성

1. 샘플 코드 다운로드
2. 압축 해제 후 해당 프로젝트 폴더 위치로 이동 `cd 폴더이름/app`
3. Firebase 프로젝트 초기화 `firebase init` . DB + Hosting 선택

서비스워커 구현 - 초기화

4. 서비스워커 지원여부 확인 및 등록
5. `sw.js` 에 로그 추가하여 개발자 도구 `Console` 에서 로그 확인
6. 서비스워커의 구독 여부에 따라 페이지 UI 초기화. `initialiseUI()`
7. 구독 여부에 따른 버튼 활성화 표시. `updateBtn()`
8. 푸시 활성화 버튼에 클릭 이벤트 추가

서비스워커 구현 - Subscription

9. 구독 상태로 전환하는 `subscribeUser()` 구현 및 `subscription` 객체 확인
10. 등록한 subscription 정보를 화면에 표시하는 `updateSubscriptionOnServer()` 구현
11. 애플리케이션 실행하여 알람 허용여부 경고창 뜨는지 확인

Firebase 구현 - 서버에 Key 저장

12. `sendDeviceKeytoFirebase()` 를 `updateSubscriptionOnServer()` 에 추가
13. `firebase-db.js` 에 `sendDeviceKeytoFirebase()` 를 추가하고 `db.ref.set()` 기능 구현
14. `db.ref.set()` 에 필요한 Utility Functions 구현

Notification 구현 - Push 알람 수신창 UX

15. `sw.js` 에 `self.addEventListener('push', fn)` 구현
16. 아래에 `self.addEventListener('notificationclick', fn)` 구현
17. 개발자도구 `Application` 패널에서 Notification 표시 여부 확인
18. `push_curl_command.txt` 의 명령어를 이용하여 해당 브라우저에 푸시 메시지 전송

Firebase **구현 - 해당** Key **제거**

19. `unSubscribeUser()` 로 구독 비활성화 기능 구현
20. 비활성화된 이전 subscription 객체 콘솔 로그로 확인

추가 과제

`firebase-db.js` 의 `removeDeviceKeyInFirebase()` 를 구현하여 구독하지 않는 브라우저 키 값을 Firebase DB 에서 삭제해보세요.

API 참고

참고

- Push Web App Code Labs
- JS Client Cloud Messaging Config
- Push Notifications on the Open Web
- FCM, Firebase Cloud Messaging
- GCM Image reference

끝