PWA란?

Web의 장점과 App의 장점을 적절하게 합쳐놓은 **웹앱의 구현 방법**.

웹 앱을 PWA로 식별되기 위해 확인해야 하는 핵심 원칙

1. 발견 가능, 따라서 컨텐츠를 검색 엔진을 통해 찾을 수 있습니다.
2. 설치 가능, 따라서 기기의 홈 화면에서 사용할 수 있습니다.
3. 연결 가능, 따라서 간단하게 URL을 전송해 공유할 수 있습니다.
4. 네트워크 독립적, 따라서 오프라인이나 불안한 네트워크 연결에서 동작합니다.
5. 점진적, 따라서 최신 브라우저의 모든 기능은 사용할 수 없지만 이전 브라우저의 기본 기능은 여전히 사용할 수 있습니다.
6. 재 참여 가능(Re-engageable), 따라서 새 컨텐츠가 사용 가능할 때마다 알림을 보낼 수 있습니다.
   반응형, 따라서 모든 기기의 화면이나 브라우저에서 사용할 수 있습니다 — 모바일 폰, 태블릿, 노트북, TV, 냉장고, 등.
7. 안전, 따라서 여러분과 앱 사이의 연결이 여러분의 민감한 데이터에 접근하려는 모든 제 3자로부터 안전합니다.

참고 링크 : https://developer.mozilla.org/ko/docs/Web/Progressive_web_apps/소개

핵심 원칙들을 다시 정리해보면 크게 3가지 요건을 갖추면 된다.

1. HTTPS
2. Manifest
3. Service Worker

3가지만(!)이라고 하기에는 각 항목들에 대해서도 많은 양의 내용이 있다.
가깝고도 먼 PWA로의 길....

HTTPS

PWA는 운영체제의 여러 특별한 권한을 부여받기 때문에, **웹서버와의 보안 연결**은 필수다.
HTTPS를 적용하기 위해 무료 인증서 발급을 받으려 하면 Let's Encryp(https://letsencrypt.org/ko/)로 가게 되더라. -_-a

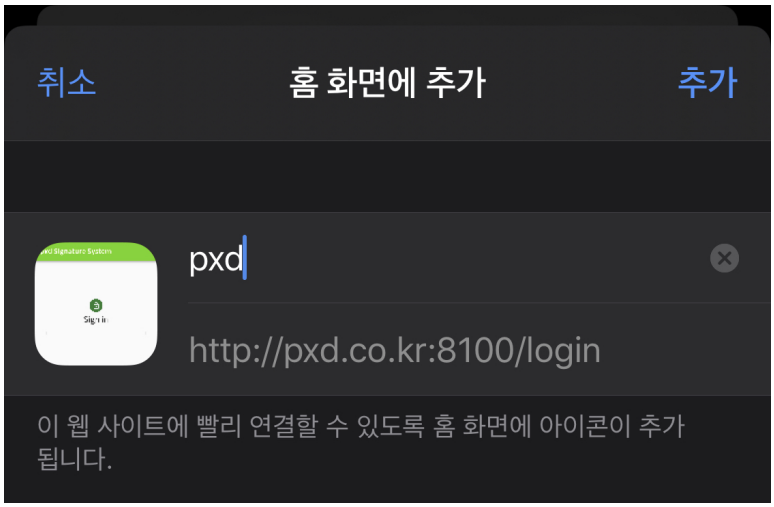
Manifest

사이트와 관련된 정보를 담고 있는 json 파일이다.
브라우저나 검색엔진이 찾을 수 있도록 <head></head> 안에 <link>를 사용하여 참조하면 된다.

```
// pxd 온라인 서명 사이트에 적용한 manifest.json
// create-react-app 으로 리액트 프로젝트를 생성하면 public 폴더에 기본생성 된다.
{
  "short_name": "pxd",
  "name": "pxd Online Signature",
  "icons": [
    {
      "src": "16_gray_text.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "16_gray_text.png",
      "type": "image/png",
      "sizes": "16x16"
    },
    {
      "src": "32_gray_text.png",
      "type": "image/png",
      "sizes": "32x32"
    },
    {
      "src": "152_gray_text.png",
      "type": "image/png",
      "sizes": "152x152"
    },
    {
      "src": "192_gray_text.png",
      "type": "image/png",
      "sizes": "192x192"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

```
<link rel="manifest" href="/manifest.json">
```

- `name` : 사용자 홈 화면에서 아이콘 이름으로 사용.
- `short_name` : 사용자 홈 화면에서 아이콘 이름으로 사용. name이 너무 길 경우 short_name으로 대체.
- `icons` : 아이콘들의 정보(URL, 크기, 타입). 사용자의 기기에 적합한 것을 선택할 수 있도록 여러개를 추가할 수 있다.
- `start_url` : 웹앱 실행시 시작되는 URL
- `display` : 앱이 실행될 때의 속성 (fullscreen, minimul-ui, standalone, browser)
- `theme_color` : 상단 툴바의 색상
- `background_color` : 스플래시 화면 배경 색상
- 최소 요구사항은 `name`과 적어도 하나의 `icons`이다.
  `description`, `short_name`, `start_url`은 권장사항이다.
  이 외에 더 많은 항목들이 있다. (참고 URL : https://developer.mozilla.org/ko/docs/Web/Manifest)



(PWA는 아니지만 manifest.json이 적용된 사이트의 모습)

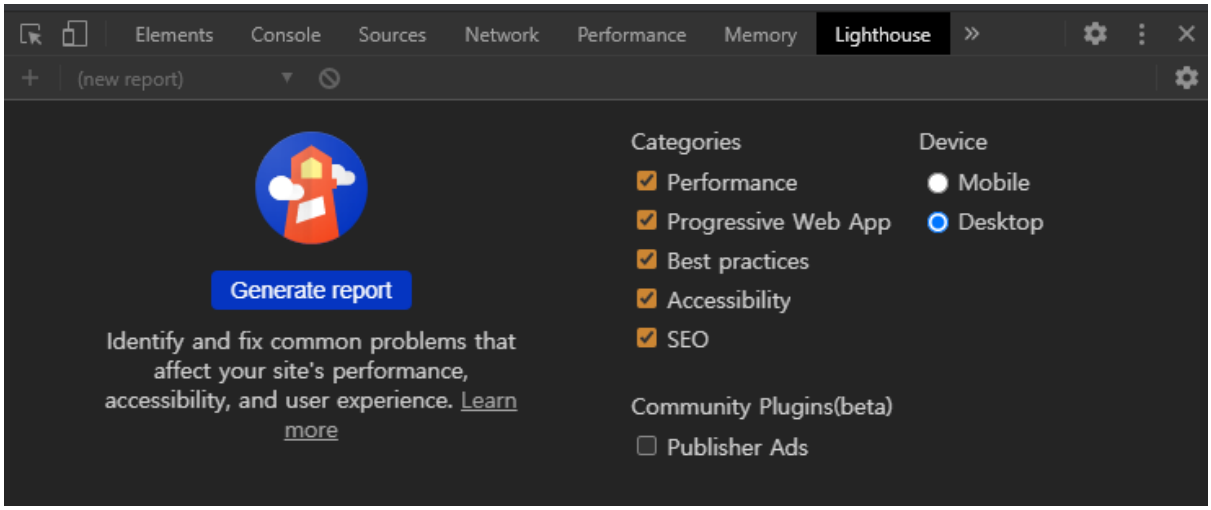
Service Worker

브라우저의 백그라운드에서 실행되는 자바스크립트 워커이다.
오프라인 경험, **주기적 백그라운드 동기화**, **푸시 알림**이 웹에서 지원되고 있다. 서비스 워커는 이러한 모든 기능의 기술적 기반을 제공한다.
향후 지오펜싱(https://developers.google.com/location-context/geofencing/)과 같은 다른 기능을 지원할 수 있다.

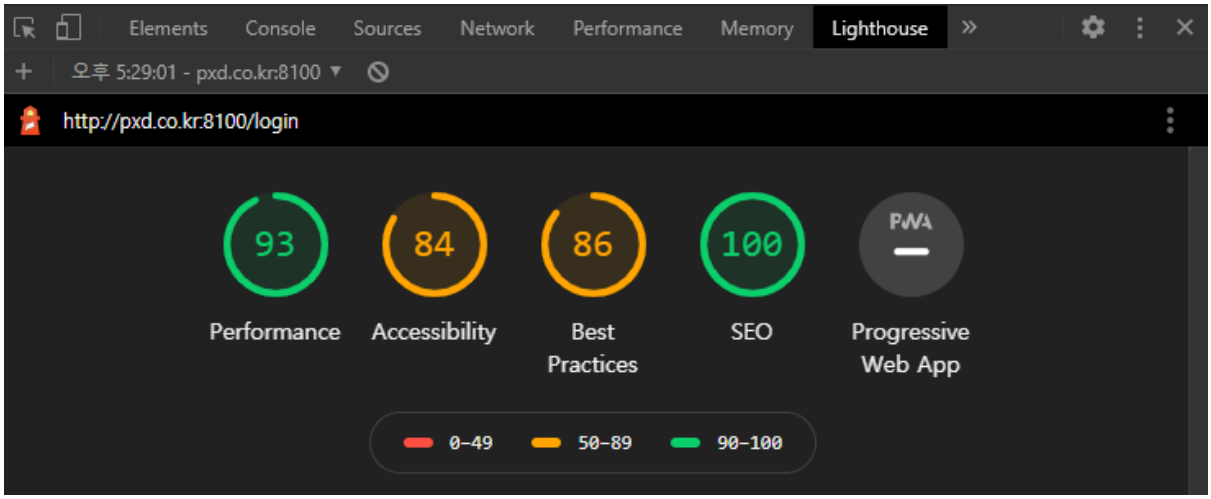
워낙 양이 방대해서 요약은 여기까지만..
설명은 링크로 대체한다.
모질라에서 제공하는 가이드 (https://serviceworke.rs/)
구글에서 제공하는 가이드
(https://developers.google.com/web/fundamentals/primers/service-workers/)

PWA로서 인정(?)을 받으려면 어떻게 확인하는가?

Lighthouse라는 도구를 통해 테스트를 할 수 있다.
(https://developers.google.com/web/tools/lighthouse/)
크롬 브라우저의 개발자 도구를 열어보면 이미 설치되어 있을 것이다. ~~(크롬 만세!)~~



Generate report 버튼을 눌러보자.



PWA만 테스트할 수 있는 것이 아니라 여러가지 항목들에 대해 확인 가능하다.
위의 샘플로 가져온 manifest.json을 적용한 **pxd 온라인 서명 사이트**의 점수이다.
Progressive Web App에는 점수가 나오지 않았는데 PWA로서 인정(?)받지 못한 것이다.
이유는 최소 3가지의 요건을 갖추지 못했기 때문이다. (**https**가 아님..)

그래서 이거 하면 뭐가 좋아?

- 검색이 쉽다.
  앱스토어에 찾아가서 다운로드를 하는 것보다 구글에서 검색 결과로 링크타고 넘어오거나, URL로 직접 찾아오거나 공유할 수 있다.
- 설치가 쉽다.
  브라우저에서 "홈 화면에 추가"만 하면 끝.
- 앱 업데이트가 있을 때 변경된 컨텐츠만 업데이트 할 수 있다. (Native App의 경우에는 앱스토어에서 앱 전체를 다운로드 받아야 한다.)
- HTTPS의 도입으로 안전하다.
  지오로케이션이나 서비스 워커 등 최신 기술들을 적용하기 위한 전제 조건이기도 하고, SSL 적용 여부가 검색 순위에도 영향을 주고 있다.
- 서비스 워커를 사용한 캐싱 덕분에 앱을 설치한 후에 로딩 시간이 줄어 소중한 데이터와 시간을 절약할 수 있다.
  - 네트워크 연결을 먼저 확인하고 연결이 느리거나 안되어있으면 이전에 캐싱한 자원을 사용한다던가..
  - 커스텀 오프라인 페이지를 정의할 수도 있다.