



# PWA

## PWA

What is PWA?

Web Application? Progressive Web Application?

PWA의 조건?

PWA의 기본 설정 작업

내가 접속한 웹사이트가 PWA인지 확인은 어떻게 알 수 있을까?

## 서비스 워커 파헤치기

서비스 워커를 사용하려면?

서비스워커 시작하기

서비스 워커 등록

React에서 서비스워커 등록해보기

서비스워커로 오프라인 모드 만들기

sw.js에 cache 등록하기

오프라인에서도 API 데이터를 유지하는 방법

푸쉬 알림 팝업

푸쉬 메시지 보내기

# PWA

## PWA

## What is PWA?

- Progressive Web Application
  - 새롭고 강력한 소프트웨어 앱을 만드는 방식
  - HTML, CSS, JavaScript를 이용해 만든 웹앱을 웹 브라우저 APIs와 결합해 크로스 플랫폼에서 동작하는 어플리케이션을 만드는 방법!
  - Google I/O 2016에서 처음 소개

## Web Application? Progressive Web Application?

- Web? WebApplication?
  - 웹페이지와 웹어플리케이션의 기준을 나누는 정의된 개념은 없다.
    - 모두 HTML, CSS, 자바스크립트로 작성하고 동작하는 것도 비슷..
    - 차이라면 웹앱은 사용자와의 상호작용성을 고려한 웹 사이트라고 보면 될듯
      - 웹 애플리케이션이 뭐야 라고 한다면..?
        - 반응형으로 사용성 고려하고...
        - 사용자와 상호작용하며 필요한 정보 제공하고.. (구글 지도 등)
      - 단순히 정적으로 정보만 볼 수 있는 사이트는 웹페이지라고 생각하면 될듯.
- PWA의 장점?
  - PWA는 웹 앱에 네이티브 앱의 장점을 추가한 웹 앱 이다.
    - 웹 앱의 장점이 뛰어난 접근성이라고 한다면 (앱 설치 필요 X, 브라우저+ 인터넷만 있다면 URL로 접속만 하면 끝).
    - 네이티브 앱의 장점은 운영체제의 거의 모든 자원을 쓸 수 있으므로 더 부드러운 사용자 경험을 제공 + 설치하기 때문에 인터넷 없이도 동작하는 프로그램을 만들 수 있다는 장점을 제공한다.
    - PWA는 안드로이드, ios 앱 개발에 필요한 기술이 단 하나도 요구되지 않는다!
    - PWA는 네이티브 앱처럼 푸시알림이 가능하면서도 SEO 최적화가 가능!
    - PWA는 Service Worker 덕분에 앱 동작에 필요한 asset 들과 일부 API call들에 대한 캐싱이 가능, 이 덕분에 PWA는 사용자의 장치가 오프라인이거나 불안정한 저속도 환경에 있더라도 캐싱 된 리소스로 안정적인 앱 사용을 지원
  - PWA는 완전히 새롭게 등장한 기술이 아니라 웹의 기존 기능을 제거하지 않은 상태로 더욱 발전된 형태의 웹 애플리케이션을 개발하려는 움직임이라 볼 수 있다.
- PWA 단점?

- 게임 같은 고사양 앱은 개발하기 힘들다..
  - 운영체제 자원을 직접 사용하는 것은 아니기 때문
    - 하드웨어 자원을 활용은 가능하지만 네이티브 앱 수준은 아님.

## PWA의 조건?

### 1. 사용자의 기기에 설치가 가능해야 한다.

- 사용자는 홈 스크린 아이콘을 통해 앱에 접근하며, 브라우저 탭이 아닌 독립 실행형 창으로 앱이 열려야 합니다.

### 2. 오프라인, 저속도 환경에서도 동작해야 한다.

- 리소스를 캐싱해두어 인터넷 연결이 없는 상태에서도 잘 동작할 수 있어야 합니다.

### 3. 백그라운드 동기화가 가능해야 합니다.

- 사용자가 앱을 사용하고 있지 않을 때에도 동작할 수 있어야 합니다.

### 4. 사용자 재참여를 유도할 수 있도록 푸시 알림이 가능해야 한다.

- 푸시 알림은 PWA를 네이티브 앱으로 견줄 수 있도록 해주는 강력한 기능입니다.

### 5. 다양한 Web API를 사용하여 네이티브 앱과 같은 사용성을 갖출 수 있어야 한다.

- Web API : 사용자 위치, 카메라, 진동, AR, VR 등

### 6. HTTPS 프로토콜을 통해 제공되어야 한다.

- PWA는 신뢰할 수 있는 연결 상태에서만 동작합니다. 따라서 HTTPS를 통한 보안 연결을 통해 서비스가 제공되어야 합니다.

### 7. 빠르게 로드되고, 빠르게 유지되어야 한다.

- Starts fast, stays fast

### 8. 모든 브라우저에서 동작해야 하며, 모든 화면 크기에 대응해야 한다.

- Works in any browser & Responsive to any screen size

## PWA의 기본 설정 작업

- PWA는 기존 기술을 조합하고 확장한 표준 기술
  - 기존 웹페이지도 몇가지 요구사항을 충족시킨다면 쉽게 전환 가능.

### 1. 서비스워커

- 백그라운드에서 실행되는 스크립트 파일

- PWA의 핵심기능인 푸시알림, 백그라운드 동기화, 오프라인 환경 지원, 리소스 캐싱을 해주는 역할을 한다.
- 서비스 워커는 HTTPS환경에서만 동작한다.

## 2. Manifest JSON

- 브라우저에게 PWA에 대한 정보를 제공, 현재의 웹앱이 데스크톱, 모바일 등에 어떻게 설치 되어야 하는지 알려주는 JSON 파일
  - 앱이름, 아이콘, 테마 색상 등 포함

## 내가 접속한 웹사이트가 PWA인지 확인은 어떻게 알 수 있을까?

### 1. Lighthouse를 통해 확인.

- 검사하려는 사이트에서 개발자 도구를 열고 Lighthouse 탭으로 이동
- 프로그래시브 웹 앱이 체크 돼있는지 확인하고 페이지 로드 분석 클릭후 대기
- 검사가 끝나면 PWA인지 확인이 가능하다

스타벅스 PWA : <https://app.starbucks.com/>

일반 웹사이트 :

<http://i10a805.p.ssafy.io/>

## 서비스 워커 파헤치기

- 웹 사용자가 수년간 겪어온 가장 큰 문제 중 하나는 연결의 끊김
- 이러한 문제를 해결하기 위한 다양한 시도가 있었으나 해결되지 않은 문제들이 남음
  - asset 캐싱
  - 사용자 지정 네트워크 요청에 대한 제어 메커니즘
- 서비스 워커는 이러한 문제를 해결
  - 캐시된 asset을 사용하도록 설정 가능
    - 오프라인에서도 기본 환경 제공 가능
  - 서비스 워커는 프록시 서버처럼 작동
    - 요청과 응답을 수정하고 자체 캐시의 항목으로 대체 가능

## 서비스 워커를 사용하려면?

- 서비스 워커를 사용하려면 HTTPS를 통해 코드가 제공돼야 한다
  - 보안상의 이유
- HTTPS를 지원하는 서버가 필요하다.
- 실험을 호스팅하기 위해서 GitHub, Netlify, Vercel 등 서비스 사용 가능
  - localhost도 안전한 출처로 허용함

## 서비스워커 시작하기

### 서비스 워커 등록

- index.html에 서비스 워커를 등록한다.

#### ▼ index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
    <title>PWA Example</title>
    <script>
      if ("serviceWorker" in navigator) {
        navigator.serviceWorker
          .register("./sw.js")
          .then((response) => {
            console.warn("response", response);
          })
          .catch((err) => {
            console.error("err", err);
          });
      }
    </script>
  </head>
  <body></body>
</html>
```



```

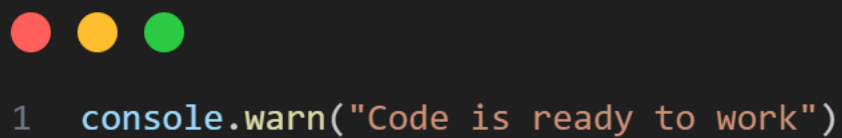
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>PWA Example</title>
7      <script>
8        if ("serviceWorker" in navigator) {
9          navigator.serviceWorker
10             .register("./sw.js")
11             .then((response) => {
12               console.warn("response", response);
13             })
14             .catch((err) => {
15               console.error("err", err);
16             });
17        }
18      </script>
19    </head>
20    <body></body>
21  </html>

```

- sw.js

- ▼ sw.js

console.warn("Code is ready to work")

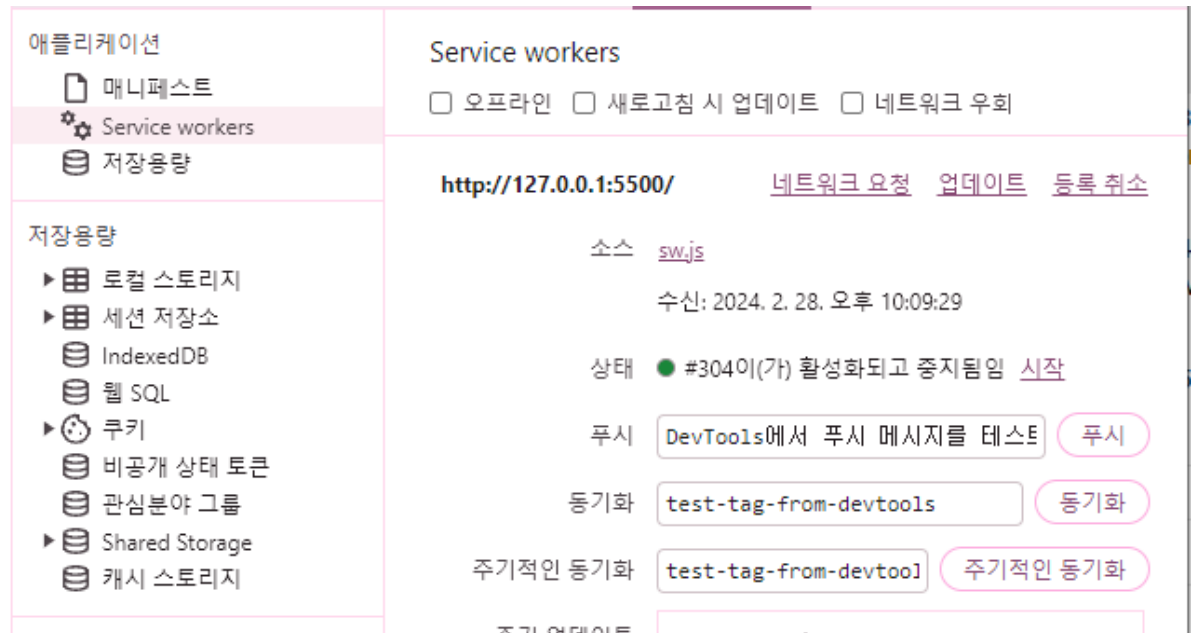
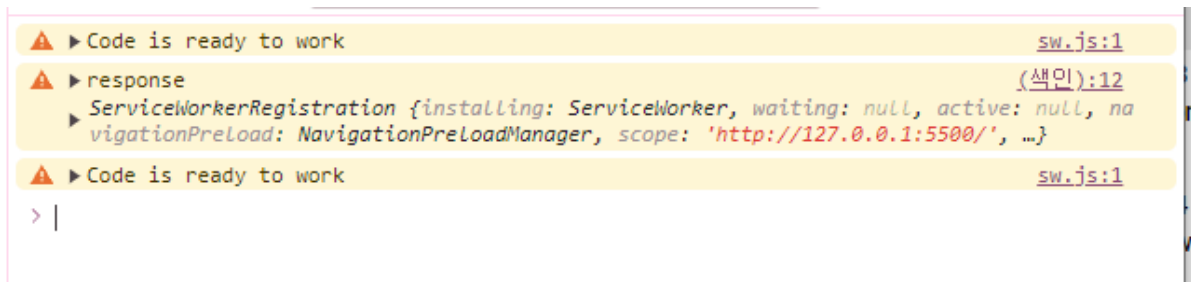


```

1  console.warn("Code is ready to work")

```

- 실행 결과

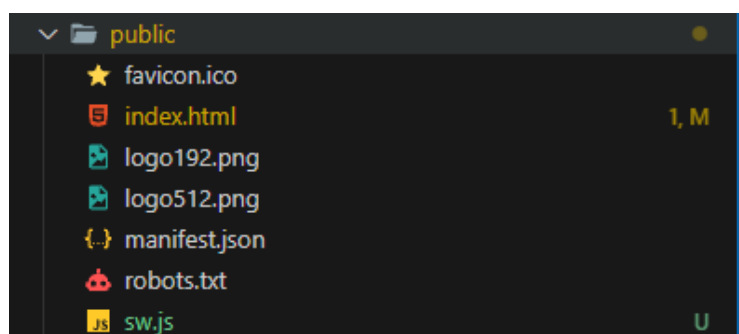


## React에서 서비스워커 등록해보기

React 프로젝트 설치

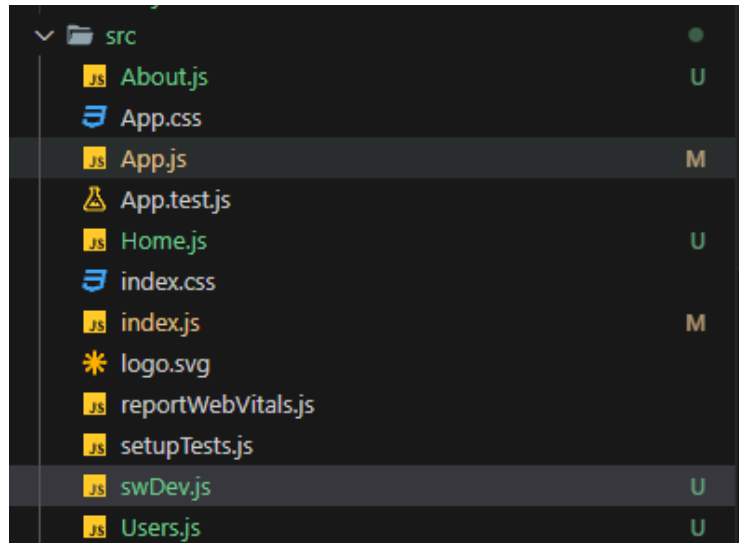
```
npx create-react-app pwa-app
```

- public 폴더에 sw.js 만들기



```
console.warn('ws file is in public folder')
```

- src 폴더에 swDev.js 만들기



```
export default function swDev() {  
  let swUrl = `${process.env.PUBLIC_URL}/sw.js`;  
  navigator.serviceWorker.register(swUrl).then((response) => {  
    console.warn("response", response);  
  });  
}
```

- index.js에 import 및 실행



```

1  import React from "react";
2  import ReactDOM from "react-dom/client";
3  import "./index.css";
4  import App from "./App";
5  import "bootstrap/dist/css/bootstrap.min.css";
6  import swDev from "./swDev";
7
8  const root = ReactDOM.createRoot(document.getElementById("root"));
9  root.render(
10    <React.StrictMode>
11      <App />
12    </React.StrictMode>
13  );
14
15  swDev();

```

```

⚠ ▶ ws file is in public folder sw.js:1
⚠ ▶ response ▶ ServiceWorkerRegistration swDev.js:4
>

```

## 서비스워커로 오프라인 모드 만들기

- 시작하기전 서비스 접속해서 새로고침 해보면 chunk같은 여러 파일이 받아지는것 확인 해보라

## sw.js에 cache 등록하기

```

let cacheData = "V1";
this.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(cacheData).then((cache) => {
      // bundle 파일과 등록할 페이지들을 여기서 넣어주기
      cache.addAll([

```

```

        "/static/js/bundle.js",
        "/index.html",
        "/",
        "/app.js",
        "/style.css",
    ]);
  })
);
});

// fetch 이벤트 발생시... 기존에 저장된 데이터를 보내준다.
this.addEventListener("fetch", (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => {
      if (response) {
        return response;
      }
    })
  );
});
});

```

- 오프라인 모드여도 캐시된 데이터로 보여주기 때문에 페이지가 보인다.

The screenshot shows a web browser with a blue header containing 'Navbar', 'Home', 'About', and 'Users' links. The main content area displays 'Home' and 'Welcome to the home page'. The Chrome DevTools interface is open, showing the 'Service workers' panel. The panel indicates that the service worker is 'Offline' (오프라인) and 'Ready' (준비됨). The URL is 'http://localhost:3000/'. The panel also shows the 'Cache' (캐시) section, which lists various resources like 'app.js', 'bundle.js', 'index.html', 'style.css', etc. The 'Cache' section is currently empty, indicating that the service worker is not yet installed or that the cache is being cleared.

## 오프라인에서도 API 데이터를 유지하는 방법

- 데이터를 가져온후 localStorage나 cache에 데이터를 저장

```
const [data, setData] = useState([]);
const [mode, setMode] = useState("online");
useEffect(() => {
  let url = "https://jsonplaceholder.typicode.com/users";
  fetch(url)
    .then((response) => {
      response.json().then((result) => {
        console.log("result", result);
        setData(result);
        localStorage.setItem("users", JSON.stringify(result));
      });
    })
    .catch((error) => {
      // 에러발생시(인터넷 끊김 등) offline 모드로 상태를 변경하고
      // localStorage에 있는 데이터를 대신 반환.
      setMode("offline");
      let collection = localStorage.getItem("users");
      setData(JSON.parse(collection));
    });
}, []);
```

- sw.js에선

```
this.addEventListener("fetch", (event) => {
  // 데이터를 가져오는데 만약 온라인 상태가 아니라면
  if (!navigator.onLine) {
    event.respondWith(
      // 캐시에서 해당 요청에 대한 응답을 찾는다.
      //만약 캐시에 해당 요청에 대한 응답이 존재한다면, 그 응답을 반환
      caches.match(event.request).then((response) => {
        if (response) {
          return response;
        }
        //해당 요청에 대한 응답이 없다면, event.request.clone()을 통해 원본
        // fetch(requestUrl)을 호출
        // 오프라인 모드 이므로 호출 하는 부분에서 에러가 날 것임
        // 해당 함수의 에러 처리 부분에서 캐시에 저장된 데이터를 반환하면 됨
      })
    );
  }
});
```

```
        let requestUrl = event.request.clone();
        fetch(requestUrl);
    })
};
});
```

## 푸쉬 알림 팝업

- In progress....

## 푸쉬 메시지 보내기

- In progress....

참고자료 :

[https://developer.mozilla.org/ko/docs/Web/Progressive\\_web\\_apps](https://developer.mozilla.org/ko/docs/Web/Progressive_web_apps)