

Progressive Web Apps

순수 자바스크립트로
PWA만들기

[학습목표]

01. PWA(프로그레시브 웹 앱)
02. PWA 필수 요소와 주요 기능
03. 매니페스트 작성하기
04. 서비스 워커 작성하기
05. 메인 화면 작성하고 실행하기
06. 서비스워커 동작 확인 및 삭제

| 01. PWA(프로그레시브 웹 앱)

배경

- 2015년 구글 크롬 엔지니어인 알렉스 러셀(Alex Russell)이 고안한 개념
- '웹 VS 앱'이라는 진부한 싸움은 이제 그만! → 웹에서 출발해 앱으로 가는 여정에서 출발지와 도착지는 명확



“미래의 웹 기술!
프로그레시브 웹앱”

특징

- 웹의 장점은 그대로 유지하면서 네이티브 앱의 강점으로 무장
- 네이티브 앱의 강력한 기능성과 웹의 뛰어난 접근성을 모두
- 갖춘 가장 이상적인 형태의 웹앱이라는 평가

| 01. PWA(프로그레시브 웹 앱)

“네이티브 앱은 기기에 최적화된 기능을 구현하다.”

구분	장점	단점
개발	<ul style="list-style-type: none">스마트폰에 최적화된 기능을 구현할 수 있다	<ul style="list-style-type: none">개발 난이도가 높고 시간이 오래 걸린다같은 서비스를 운영체제별로 만들어야 한다
배포	<ul style="list-style-type: none">전 세계를 시장으로 삼을 수 있다	<ul style="list-style-type: none">업데이트가 생기면 다시 설치해야 한다앱 시장이 포화 상태여서 선택 받기가 어렵다
사용	<ul style="list-style-type: none">홈 화면에서 아이콘을 눌러 손쉽게 접속한다알림을 통해 재방문을 유도할 수 있다	<ul style="list-style-type: none">기기의 용량을 많이 차지한다

| 01. PWA(프로그레시브 웹 앱)

“모바일 웹앱은 비용이 저렴하고 업데이트하기 쉽다.”

구분	장점	단점
개발	<ul style="list-style-type: none">• 이미 익숙한 웹 기술을 그대로 이용할 수 있다• 개발 시간을 단축할 수 있다	<ul style="list-style-type: none">• 모든 하드웨어의 기능을 사용할 수 없다• 네이티브 앱과 같은 푸시 알림 기능을 사용할 수 없다
배포	<ul style="list-style-type: none">• 웹 브라우저만 있으면 어디든 배포할 수 있다	<ul style="list-style-type: none">• 앱 스토어, 플레이 스토어를 이용할 수 없다
사용	<ul style="list-style-type: none">• 실시간으로 유지·보수 할 수 있다	<ul style="list-style-type: none">• 네이티브 앱과 같은 빠르고 풍부한 사용자 경험에 제약이 있다• 인터넷 접속이 끊어지면 사용할 수 없다

| 01. PWA(프로그레시브 웹 앱)

“하이브리드 앱은 네이티브 앱과 웹앱의 강점을 합치다.”

구분	장점	단점
개발	<ul style="list-style-type: none">• 기존에 사용하던 웹 개발 기술을 모든 운영체제에서 그대로 사용할 수 있다• 같은 코드를 모바일 운영체제 별로 다르게 패키징할 수 있다	<ul style="list-style-type: none">• 하드웨어 기능을 사용할 수 있으나 연결해주는 플러그인에 의존해야 하므로 제약이 있을 수 있다
배포	<ul style="list-style-type: none">• 네이티브 앱처럼 앱 스토어, 플레이 스토어에 배포할 수 있다	<ul style="list-style-type: none">• 네이티브 앱의 배포와 같으므로 업데이트가 생기면 다시 내려 받아야 한다• 앱 시장이 포화 상태여서 선택 받기가 어렵다
사용	<ul style="list-style-type: none">• 네이티브 앱과 유사한 사용자 경험을 제공한다	<ul style="list-style-type: none">• 네이티브 앱과 같은 성능을 내는 데는 한계가 있다

| 01. PWA(프로그래시브 웹 앱)

"프로그래시브 웹 앱 = 네이티브 앱 + 모바일 웹앱 + 하이브리드앱

구분	장점	단점
개발	<ul style="list-style-type: none">이미 익숙한 웹 기술을 그대로 이용할 수 있다.개발 시간을 단축할 수 있다.푸시 알림, 오프라인 캐시, HTTPS를 사용할 수 있다.	<ul style="list-style-type: none">하드웨어 사용은 웹 API를 통하므로 웹표준을 지원하는 브라우저가 반드시 필요하다.
배포	<ul style="list-style-type: none">웹 브라우저만 있으면 어디든 배포할 수 있다.'홈 화면 설치' 기능을 통해 OS 응용프로그램으로 설치할 수 있다.실시간으로 유지, 보수할 수 있다.	<ul style="list-style-type: none">앱 스토어, 플레이 스토어를 이용할 수 없다. 하지만 코르도바를 사용하면 동일한 코드베이스로 배포할 수 있다.
사용	<ul style="list-style-type: none">빠른 실행속도로 네이티브 앱과 유사한 사용자 경험을 제공한다.	<ul style="list-style-type: none">안드로이드, 윈도우 OS는 PWA의 모든 기능을 사용할 수 있으나 현재 iOS의 경우는 일부만 사용할 수 있다.

| 01. PWA(프로그레시브 웹 앱)

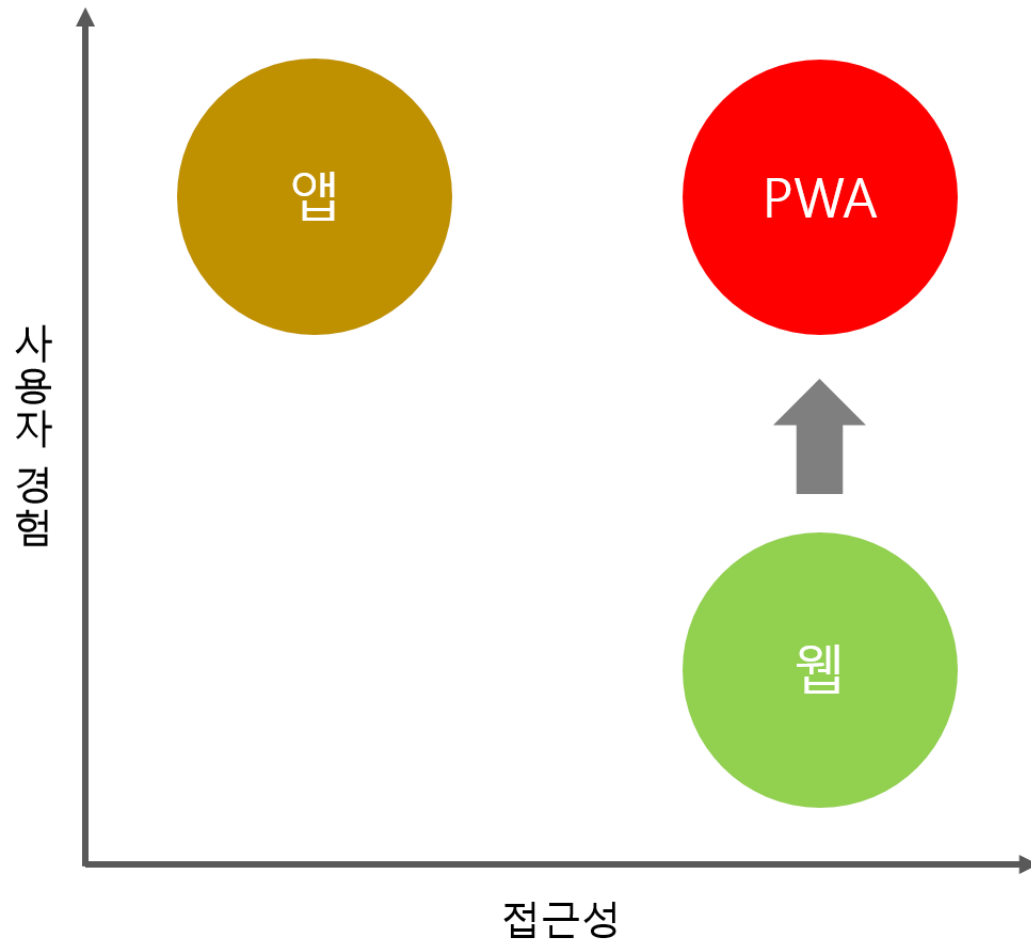


강점 (네이티브 앱 + 모바일 웹앱 + 하이브리드앱)

→ 프로그레시브 웹앱

본질은 웹이지만 앱처럼 쓸 수 있어야 한다 → 네이티브 앱과 똑같은 사용자 경험을 제공하는 것이 궁극적인 목표 → 앱을 향해 조금씩 앞으로 나아가야(progressive) 한다는 철학

| 01. PWA(프로그레시브 웹 앱)



| 02. PWA 필수 요소와 주요 기능

PWA 필수 요소

- 1) 서비스 워커
- 2) 웹앱 매니페스트
- 3) 3) HTTPS

주요 기능(네이티브 앱과 같은 경험)

- 4) 푸시 알림
- 5) 홈 화면에 추가(A2HS, Add To Home Screen)
- 6) 웹 API

필수 요소



서비스 워커



매니페스트



HTTPS

중요 기능



푸시 알림



홈 화면에 추가

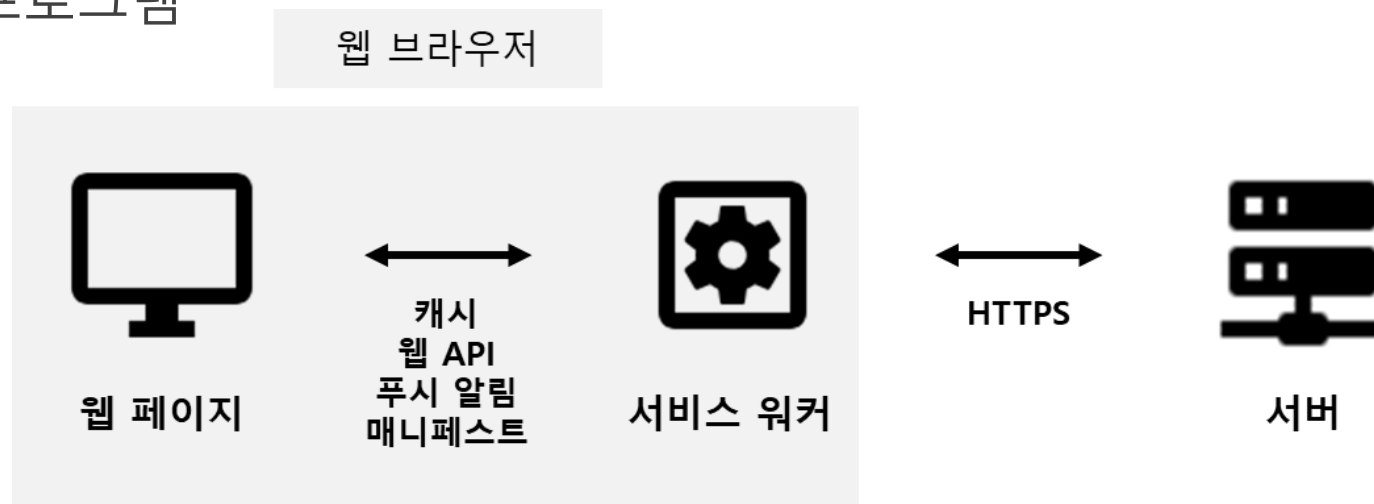


웹 API

| 02-1. PWA 필수 요소

1. 서비스 워커

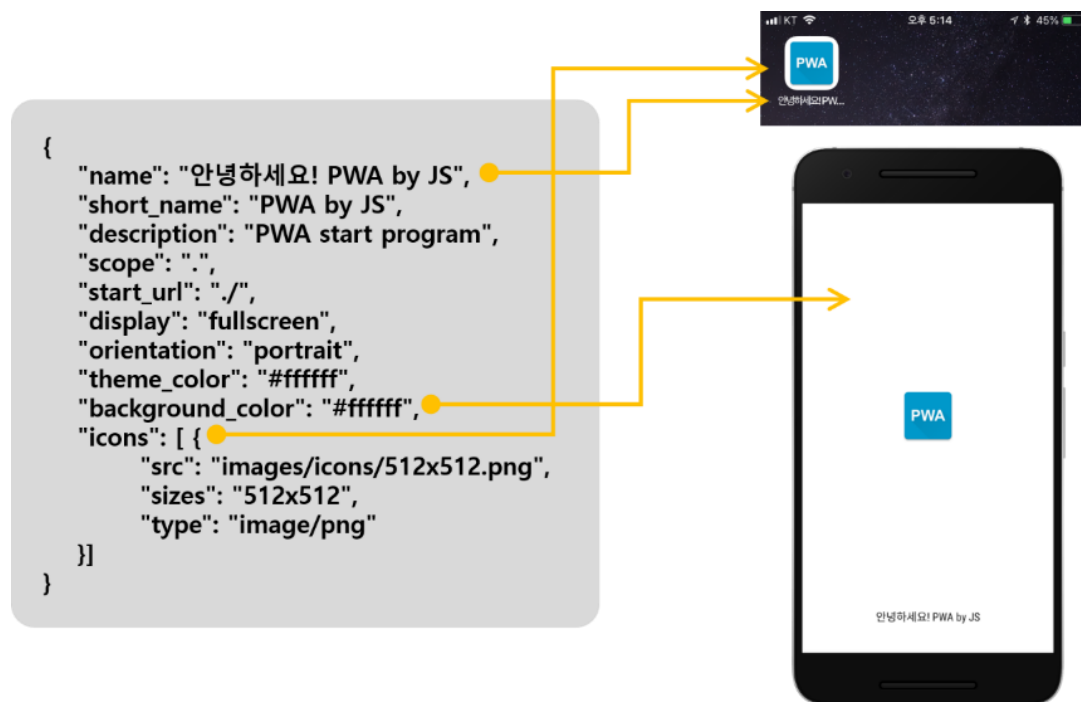
- 웹 브라우저 내에 있음
- 웹 페이지와 분리
- 항상 실행되고 있는 백그라운드 프로그램
- 캐시 관리



| 02-1. PWA 필수 요소

2. 웹앱 매니페스트

- 앱 소개 정보
- 앱 기본 설정
- JSON 파일 형식
- A2HS (Add to Home Screen)



| 02-1. PWA 필수 요소

3. 보안을 강화한 'HTTPS'

- 암호화와 인증을 거쳐 보안을 강화한 웹 통신 규약
- 반응속도가 http보다 빠름
(<http://www.httpvshttps.com>)
- PWA는 로컬에서도 동작 가능

HTTP VS HTTPS



| 03. 실습

순수 자바스크립트로 PWA를 개발

→ PWA 기본 원리 익히기

[실습 1] 프로젝트 폴더 설정

[실습 2] 매니페스트 작성

[실습 3] 서비스 워커

[실습 4] index 문서 작성

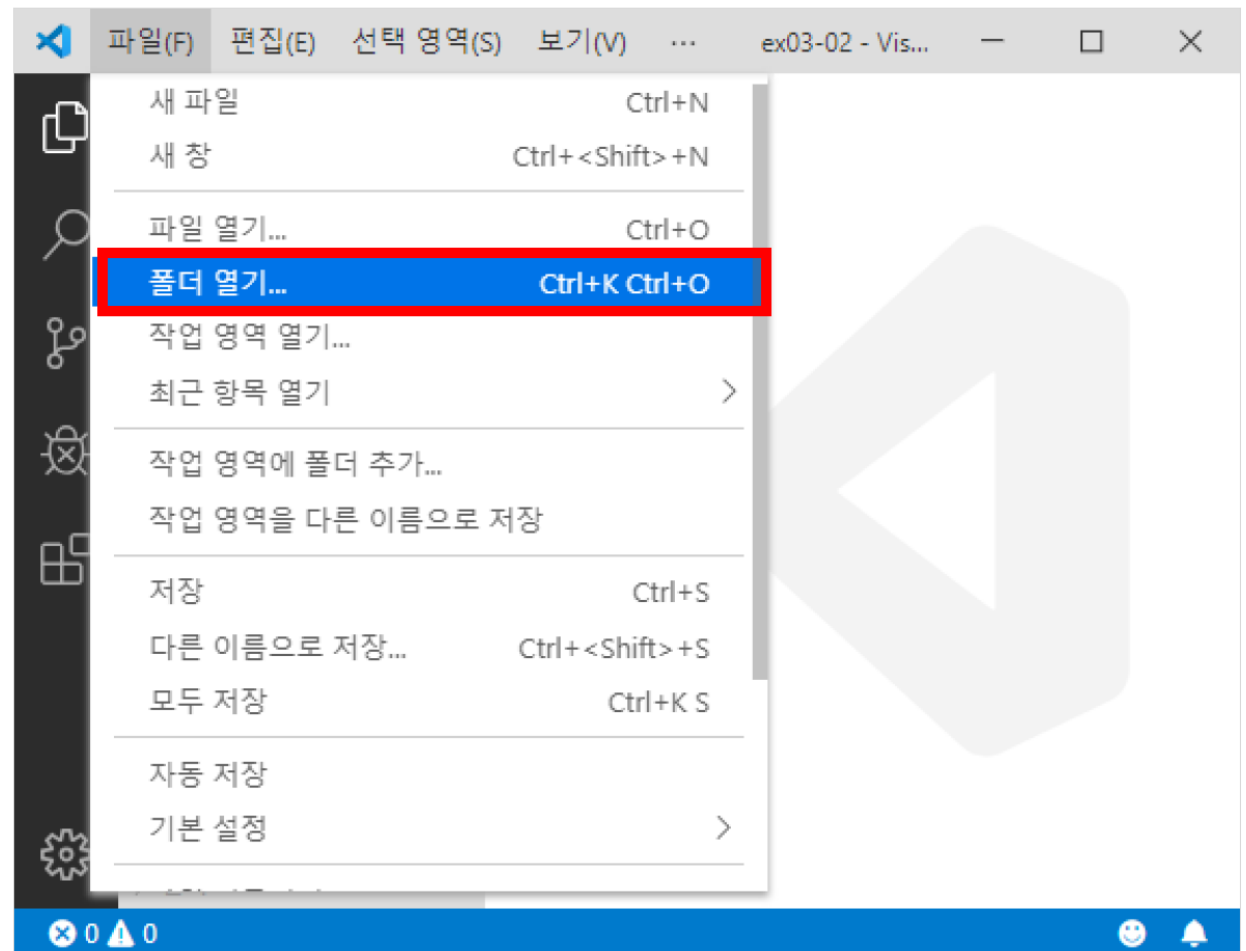
[실습 5] 콘솔 메시지, 캐시 동작 확인

[실습 6] 서비스 워커 삭제하고 프로그램 종료

[실습 1] 프로젝트 폴더 설정

1. 프로젝트 폴더 설정

- 프로젝트에 사용할 폴더 생성
- VSCode에서 폴더 선택



[실습 2] 매니페스트 작성 (manifest.json)

2. 매니페스트 준비

- 파일 > 새 파일 (새 텍스트 파일)
- root 디렉토리에 “manifest.json”으로 저장

```
01: {  
02:   "name": "안녕하세요! PWA by JS",  
03:   "short_name": "PWA by JS",  
04:   "description": "PWA start program",  
05:   "scope": ".",  
06:   "start_url": "./",  
07:   "display": "fullscreen",  
08:   "orientation": "portrait",  
09:   "theme_color": "#ffffff",  
10:   "background_color": "#ffffff",  
11:   "icons": [  
12:     {  
13:       "src": "images/icons/android-chrome-512x512.png",  
14:       "sizes": "512x512",  
15:       "type": "image/png"  
16:     }  
17:   ]  
18: }
```


[실습 2] 매니페스트 작성 (manifest.json)

```
{  
  "name" : "Hello, PWA",  
  "short_name" : "PWA",  
  "description" : "start for PWA",  
  "scope" : ".",  
  "start_url" : "./",  
  "display" : "fullscreen",  
  "orientation" : "portrait",  
  "theme_color" : "#ffffff",  
  "background_color" : "#ffffff",  
  "icons" : [  
  
  ]  
}
```

[실습 2] 매니페스트 작성 (manifest.json)

```
"icons" : [  
  {  
    "src" : "images/icons/android-chrome-512x512.png",  
    "sizes" : "512x512",  
    "type" : "image/png"  
  }  
]
```

| ① 기본 정보 작성

name

- 첫 화면(스플래시 스크린)에 출력될 이름
- 옴니(다운로드)버튼에도 출력
- 홈 화면 아이콘

short_name

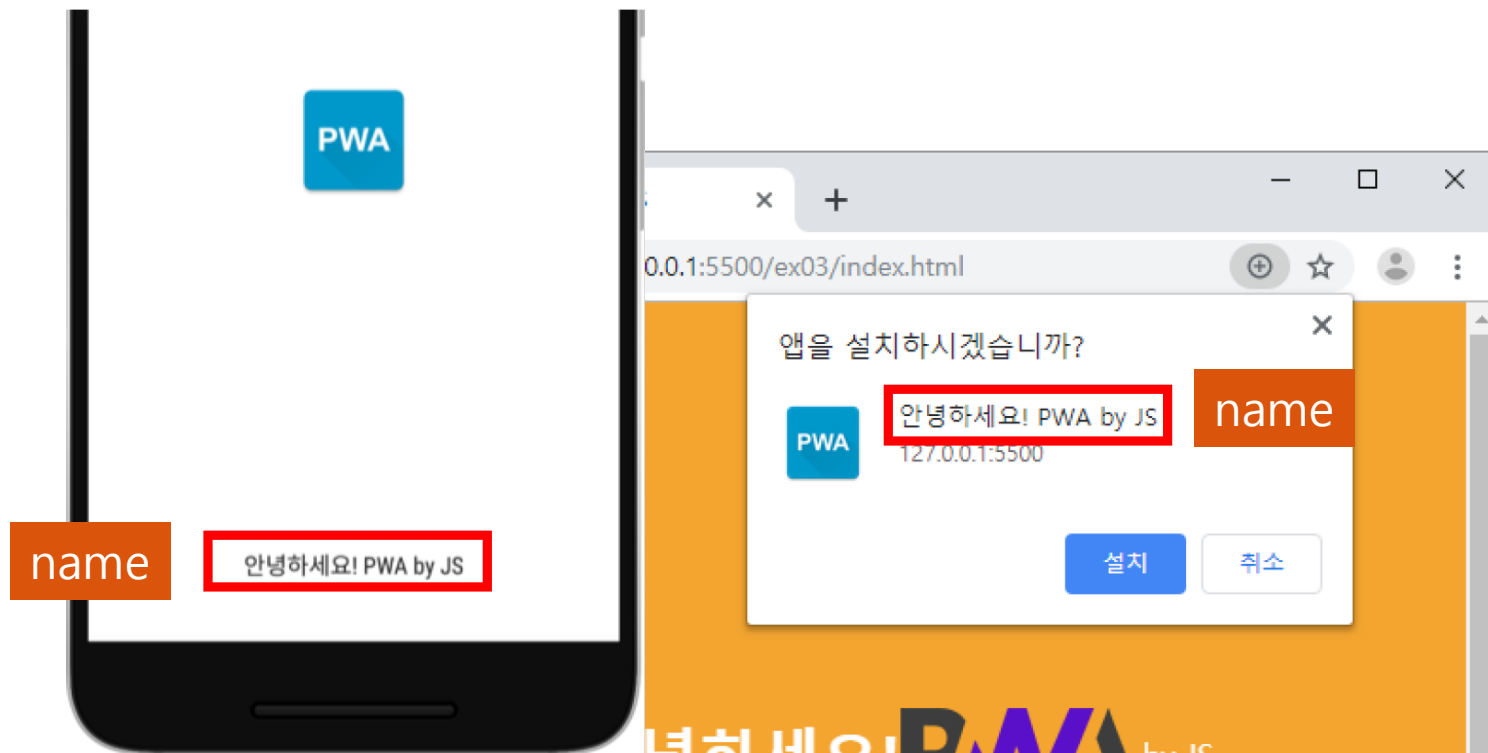
- 주로 모바일에서 사용
- 설치 배너에 출력

description

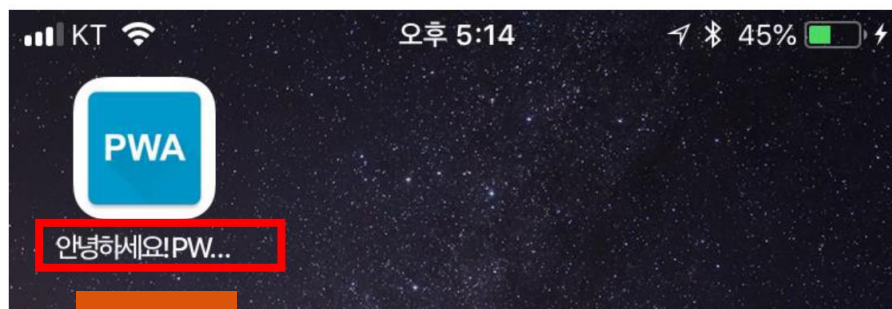
- 웹 크롤러(검색엔진)이 정보 가져갈 때 사용

① 기본 정보 작성

```
"name" : "Hello, PWA",  
"short_name" : "PWA",  
"description" : "start for PWA",
```



| ① 기본 정보 작성



name



short-name

| ② 시작 URL 설정

스코프(scope)

- 매니페스트에 정의된 내용이 적용될 수 있는 파일들의 경로 범위를 지정
- 웹앱이 어느 폴더에 있는지 지정
- "." : 현재 폴더에 있는 모든 파일

start_url

- 프로그램을 실행하면 시작될 URL을 루트 경로(./)로 설정
- index.html 파일이 있는 폴더 지정

```
"scope" : ".",  
"start_url" : "./",
```

| ③ display : 화면 표시 방법 설정

fullscreen

- 기기의 최대 화면
- 운영체제가 fullscreen을 지원하지 않으면 standalone으로 적용

```
"display" : "fullscreen",
```

standalone

- 브라우저의 주소표시줄, 상태표시줄 제거하여 일반 앱처럼 표시
- 가장 보편적으로 사용

| ③ display : 화면 표시 방법 설정

minimal-ui

- 상단에 주소 표시줄 추가
- 운영체제가 minimal-ui를 지원하지 않으면 standalone으로 적용

browser

- 웹 브라우저와 동일한 모습으로 실행

③ display : 화면 표시 방법 설정



Fullscreen



standalone



minimal-ui



browser

| ④ orientation : 화면 실행 방향 설정

portrait

- 세로로 화면 실행

landscape

- 가로로 화면 실행

```
"orientation" : "portrait",
```

| ④ orientation : 화면 실행 방향 설정



portrait



landscape

| ⑤ 테마와 배경 화면 색상 설정

theme_color

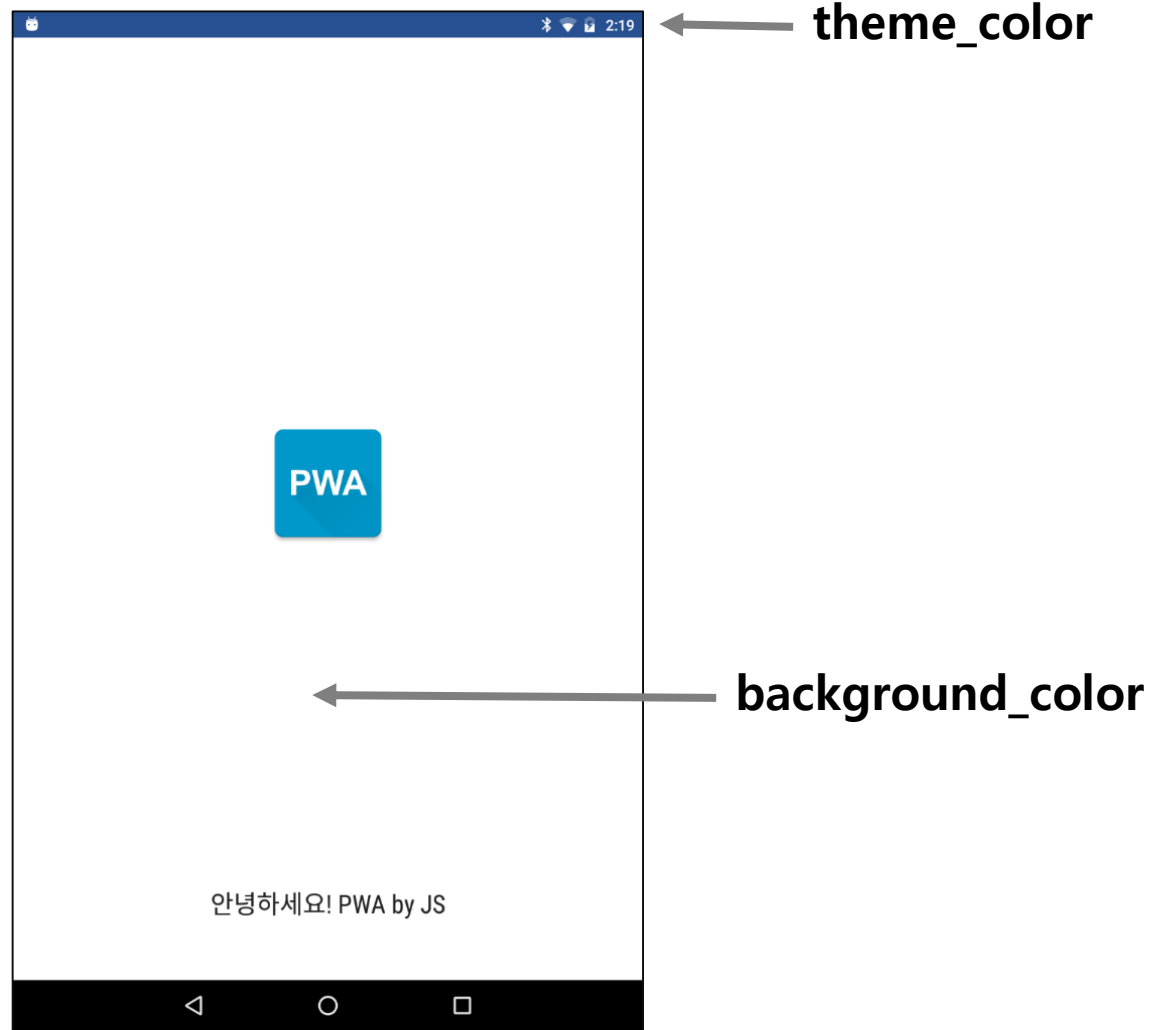
- 앱 테마 색상

background_color

- 앱 화면 배경 색

```
"theme_color" : "#ffffff",  
"background_color" : "#ffffff",
```

| ⑤ 테마와 배경 화면 색상 설정



| ⑥ 스플래시 스크린 설정

src

- 이미지의 절대 주소 또는 상대 주소

sizes

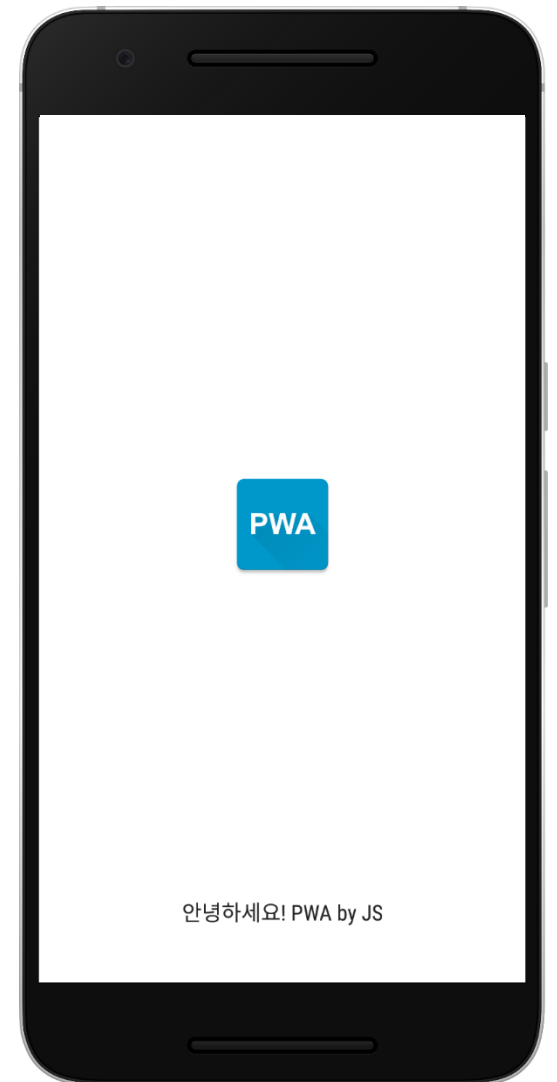
- 이미지의 픽셀 크기

type

- 이미지의 파일 유형

⑥ 스플래시 스크린 설정

```
"icons" : [  
  {  
    "src" : "images/icons/android-chrome-192x192.png",  
    "sizes" : "192x192",  
    "type" : "image/png"  
  },  
  {  
    "src" : "images/icons/android-chrome-512x512.png",  
    "sizes" : "512x512",  
    "type" : "image/png"  
  }  
]
```



⑦ 바로가기 설정

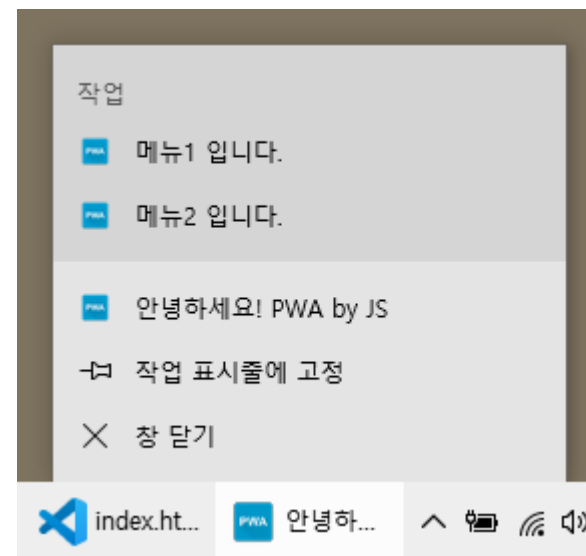
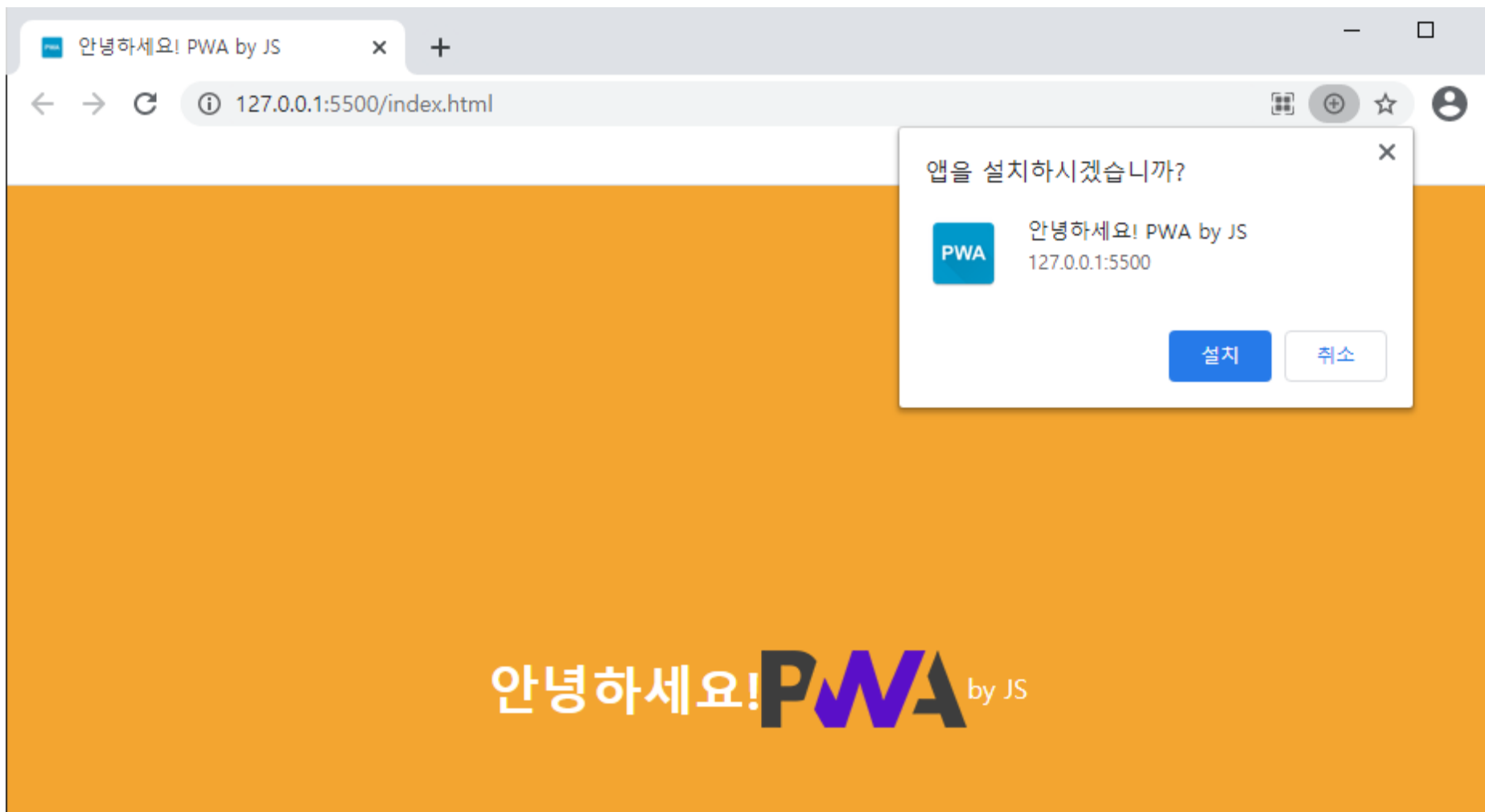
shortcuts

- MS 제품군 OS 아래쪽 아이콘 > 마우스 오른쪽 > 메뉴들 중 상단 메뉴
ex) 새창
- name: 마우스 오른쪽 클릭 시 출력될 이름
- short_name : 모바일에서 앱 설치 후 클릭했을 때 출력
- Index.html 실행 > 옴니버튼 > 다운로드 실행 > 운영체제 하단 설치된 파일에서
마우스 오른쪽 > 이름에 작성된 내용과 아이콘 확인

⑦ 바로가기 설정

```
"shortcuts": [  
  {  
    "name": "첫 번째 메뉴",  
    "short_name": "MENU 1",  
    "description": "첫 번째 메뉴입니다.",  
    "url": "./",  
    "icons": [{ "src": "images/icons/*.png", "sizes": "192x192" }]  
  },  
  {  
    "name": "두 번째 메뉴",  
    "short_name": "MENU 2",  
    "description": "두 번째 메뉴입니다.",  
    "url": "./",  
    "icons": [{ "src": "images/icons/*.png", "sizes": "192x192" }]  
  }  
]  
}
```

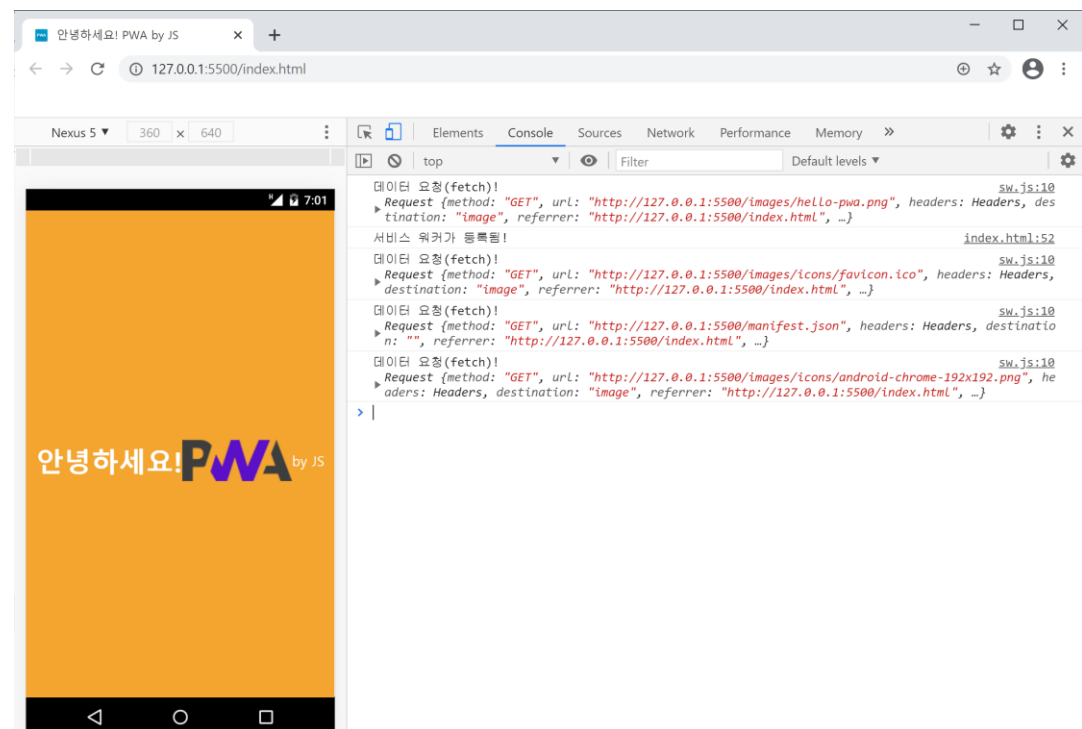
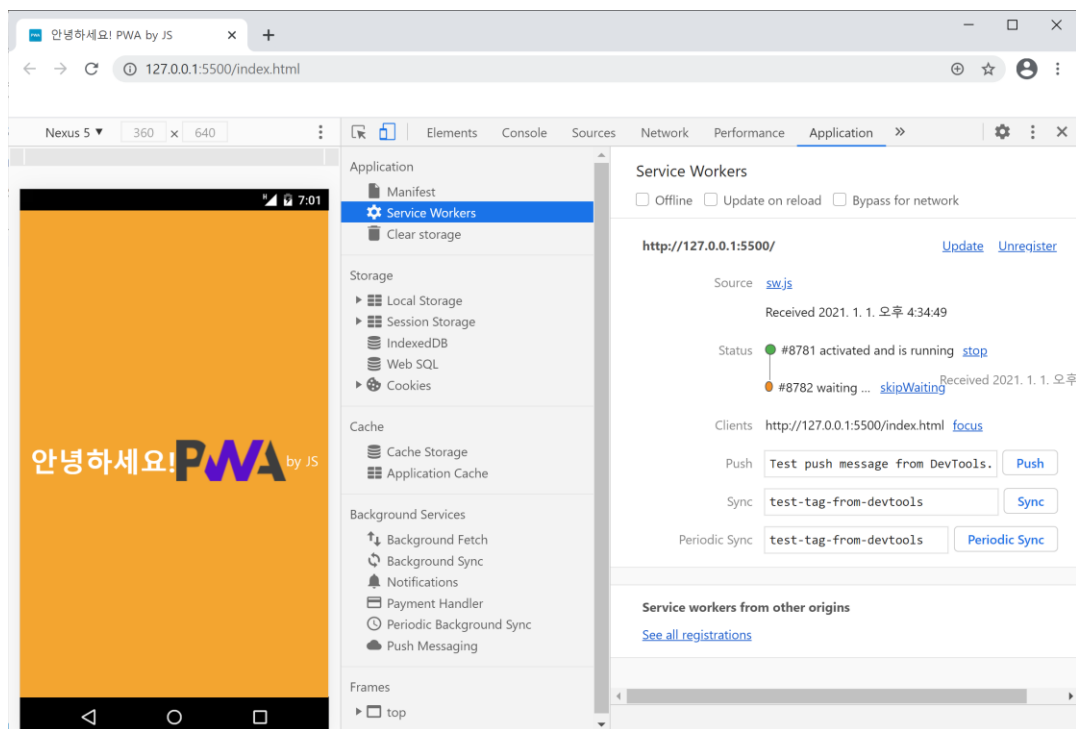
⑦ 바로가기 설정



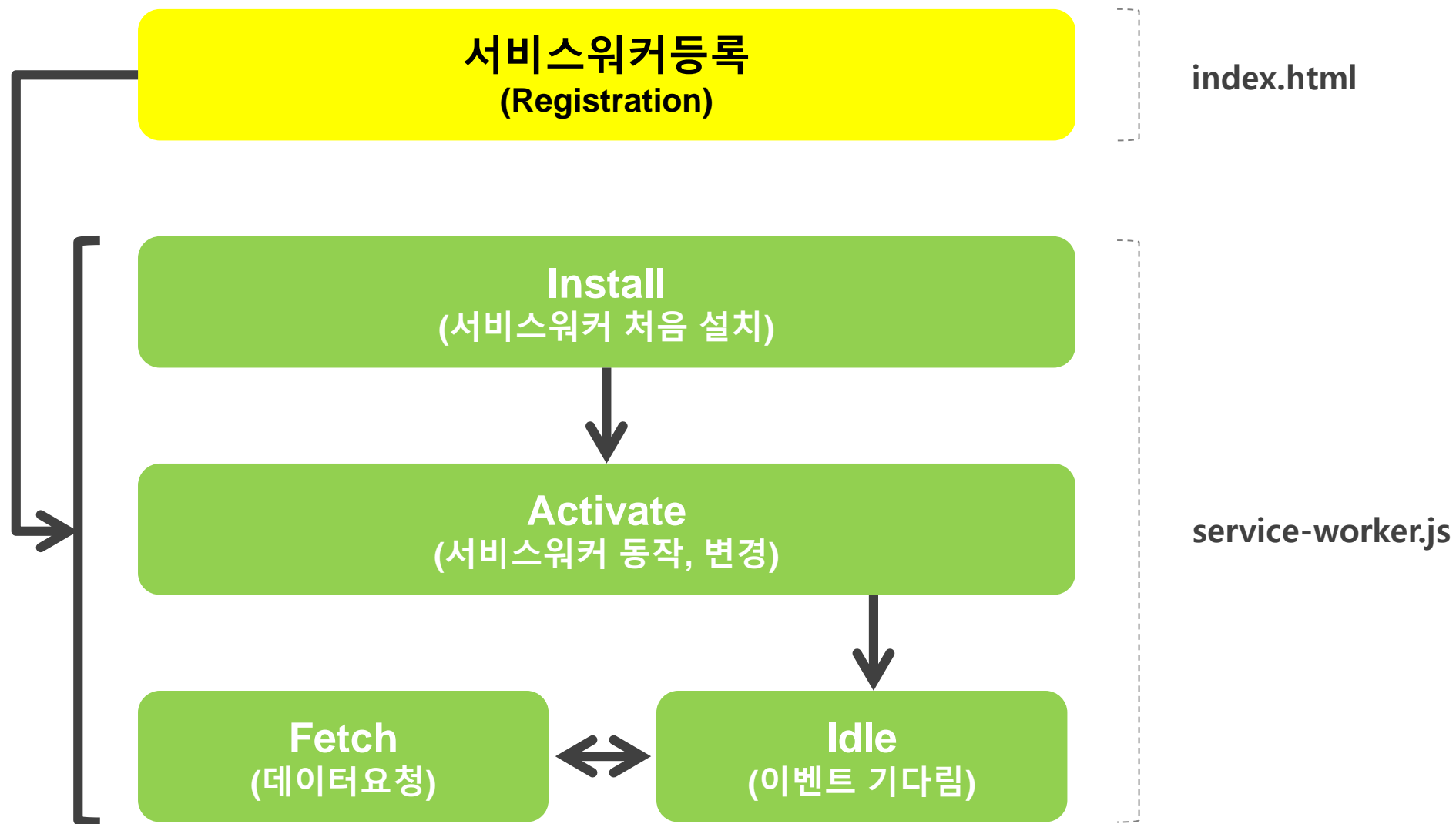
| 03-2. 매니페스트 작성

* 보편적으로 사용되므로 주로 템플릿 사용하여 작성함

03-3. 서비스 워커



| 서비스워커 생애주기(Lifecycle)



| 서비스워커 생애주기(Lifecycle)

```
self.addEventListener("install", pEvent => {  
  console.log("서비스 워커 설치 완료!");  
});
```

서비스 워커가 등록됨!

서비스워커 설치(install)함!



| 서비스워커 생애주기(Lifecycle)

```
self.addEventListener('activate', pEvent => {  
  console.log('서비스워커 동작 시작!');  
});
```

서비스 워커가 등록됨!

서비스워커 설치(install)함!

서비스워커 동작(activate) 시작됨!

> |

| 서비스워커 생애주기(Lifecycle)

Service Workers

☐ Offline ☐ Update on reload ☐ Bypass for network


http://127.0.0.1:5500/

[Update](#) [Unregister](#)

Source [sw.js](#)

Received 2021. 1. 1. 오후 4:08:11

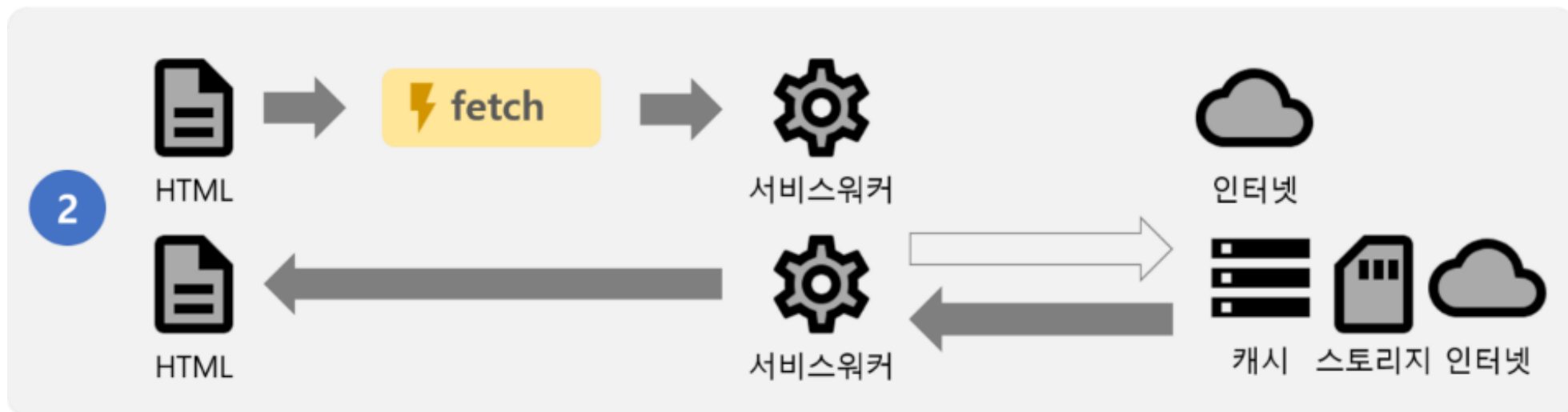
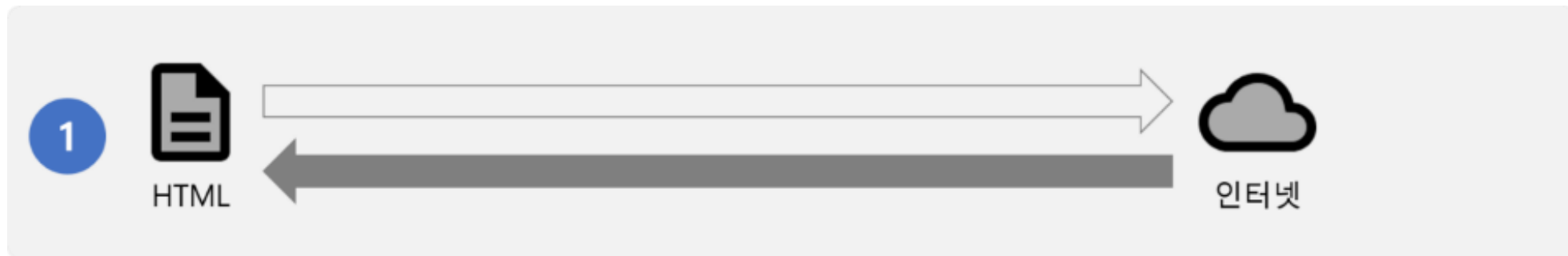
Status  #8774 activated and is running [stop](#)

 #8775 waiting to activate [skipWaiting](#)

Received 2021. 1. 1. 오후 4:08:32

Clients <http://127.0.0.1:5500/index.html> [focus](#)

| 서비스워커 생애주기(Lifecycle)



| 서비스워커 생애주기(Lifecycle)

```
self.addEventListener('fetch', pEvent => {  
  console.log("데이터 요청(fetch)!", pEvent)  
});
```

| 서비스워커 생애주기(Lifecycle)

데이터 요청(fetch)!

sw.js:10

```
FetchEvent {isTrusted: true, request: Request, clientId: "ad42098c-0a65-4971-a702-dd456701cfbe", resultingClientId: ""  
  bubbles: false  
  cancelBubble: false  
  cancelable: true  
  clientId: "ad42098c-0a65-4971-a702-dd456701cfbe"  
  composed: false  
  ▶ currentTarget: ServiceWorkerGlobalScope {clients: Clients, registration: ServiceWorkerRegistration, serviceWorker...  
    defaultPrevented: false  
    eventPhase: 0  
    ▶ handled: Promise {<fulfilled>: undefined}  
      isReload: false  
      isTrusted: true  
    ▶ path: []  
    ▶ preloadResponse: Promise {<fulfilled>: undefined}  
    ▶ request: Request {method: "GET", url: "http://127.0.0.1:5500/images/hello-pwa.png", headers: Headers, destination...  
      resultingClientId: ""  
      returnValue: true  
    ▶ srcElement: ServiceWorkerGlobalScope {clients: Clients, registration: ServiceWorkerRegistration, serviceWorker: S...
```

| 서비스워커 생애주기(Lifecycle)

```
self.addEventListener('fetch', pEvent => {  
  console.log("데이터 요청(fetch)!", pEvent.request)  
});
```

| 서비스워커 생애주기(Lifecycle)

데이터 요청(fetch)!

sw.js:10

▶ `Request {method: "GET", url: "http://127.0.0.1:5500/images/hello-pwa.png", headers: Headers, destination: "image", referrer: "http://127.0.0.1:5500/index.html", ...}`

서비스 워커가 등록됨!

index.html:52

데이터 요청(fetch)!

sw.js:10

▶ `Request {method: "GET", url: "http://127.0.0.1:5500/images/icons/favicon.ico", headers: Headers, destination: "image", referrer: "http://127.0.0.1:5500/index.html", ...}`

데이터 요청(fetch)!

sw.js:10

▶ `Request {method: "GET", url: "http://127.0.0.1:5500/manifest.json", headers: Headers, destination: "", referrer: "http://127.0.0.1:5500/index.html", ...}`

데이터 요청(fetch)!

sw.js:10

▶ `Request {method: "GET", url: "http://127.0.0.1:5500/images/icons/android-chrome-192x192.png", headers: Headers, destination: "image", referrer: "http://127.0.0.1:5500/index.html", ...}`

>

[실습 3] 서비스 워커 작성 (service_worker.js)

1. 서비스워커 작성

- 파일 > 새 파일 (새 텍스트 파일)
- root 디렉토리에 “service_worker.js”로 저장

[실습 3] 서비스 워커 작성 (service_worker.js)

```
const sCacheName = "hello-pwa"; // 캐시 제목
const aFilesToCache = [ // 캐시할 파일 지정
    './',
    './index.html',
    './manifest.json',
    './images/hello-pwa.png'
];
```

[실습 3] 서비스 워커 작성 (service_worker.js)

```
// 서비스워커 실행 & 캐시파일 저장
self.addEventListener("install", pEvent => {
  console.log("서비스 워커 설치 완료!");
  pEvent.waitUntil(
    caches.open(sCacheName)
      .then(pCache => {
        console.log("캐시에 파일 저장 완료!");
        return pCache.addAll(aFilesToCache);
      })
  );
});
```


[실습 3] 서비스 워커 작성 (service_worker.js)

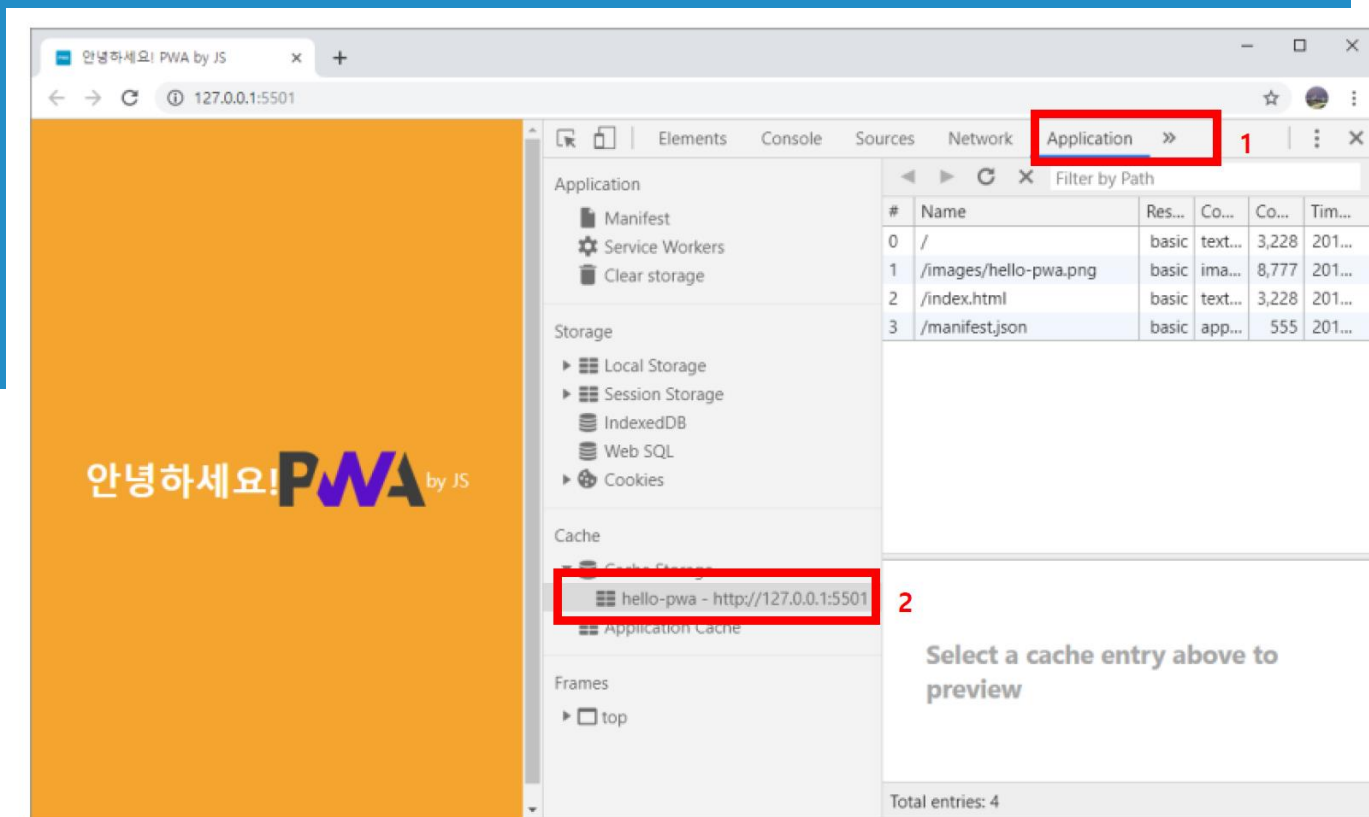
```
// 고유 번호 할당받은 서비스 워커 동작 시작
self.addEventListener('activate', pEvent => {
  console.log('서비스워커 동작 시작됨!');
});
```

[실습 3] 서비스 워커 작성 (service_worker.js)

```
// 고유 번호를 할당 받은 서비스워커 작동
self.addEventListener('fetch', pEvent => {
  pEvent.respondWith(
    caches.match(pEvent.request)
      .then(response => {
        if(!response){
          console.log("네트워크로 데이터 요청!", pEvent.request)
          return fetch(pEvent.request)
        }
        console.log("캐시에서 데이터 요청!", pEvent.request)
        return response;
      }).catch(err => console.log(err))
  );
});
```

① 캐시 제목과 파일 설정

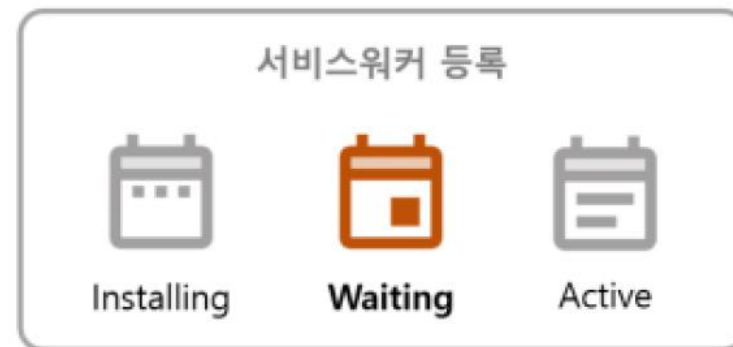
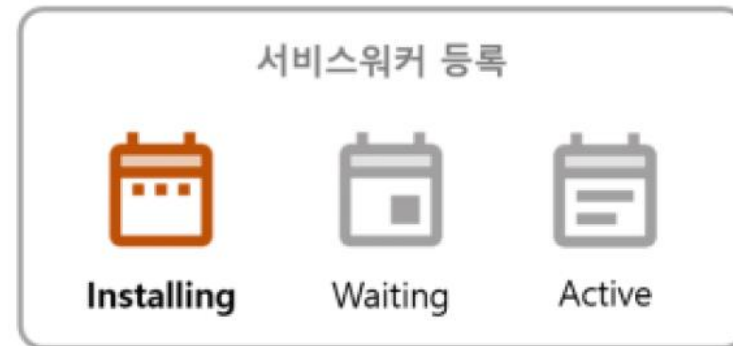
```
const sCacheName = "hello-pwa"; // 캐시 제목
const aFilesToCache = [ // 캐시할 파일 지정
  './',
  './index.html',
  './manifest.json',
  './images/hello-pwa.png'
];
```



② 서비스 워커 설치 및 캐시 파일 저장 - install 이벤트

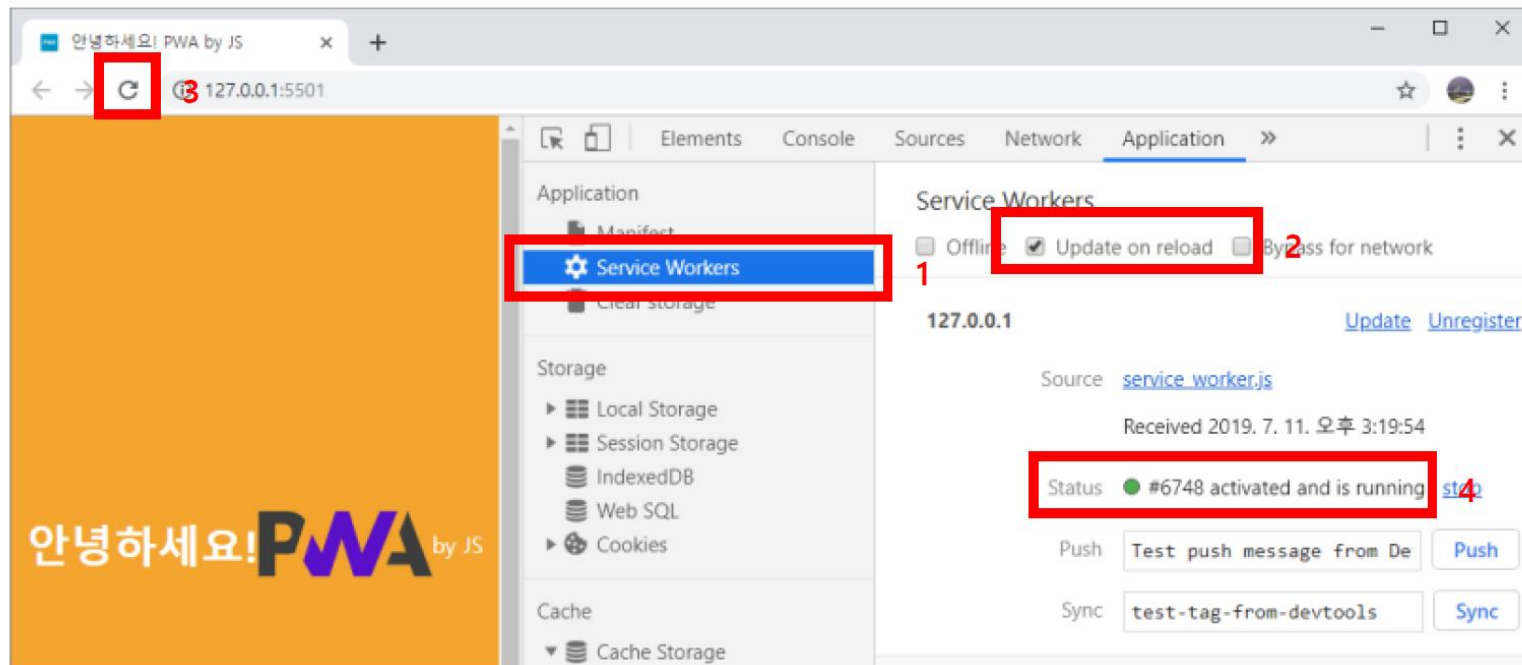
```
// 서비스워커 실행 & 캐시파일 저장
self.addEventListener("install", pEvent => {
  console.log("서비스 워커 설치 완료!");
  pEvent.waitUntil(
    caches.open(sCacheName)
      .then(pCache => {
        console.log("캐시에 파일 저장 완료!");
        return pCache.addAll(aFilesToCache);
      })
  );
});
```

② 서비스 워커 설치 및 캐시 파일 저장 - install 이벤트

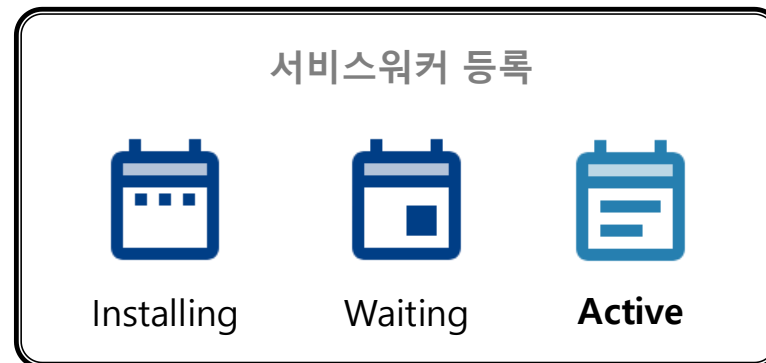
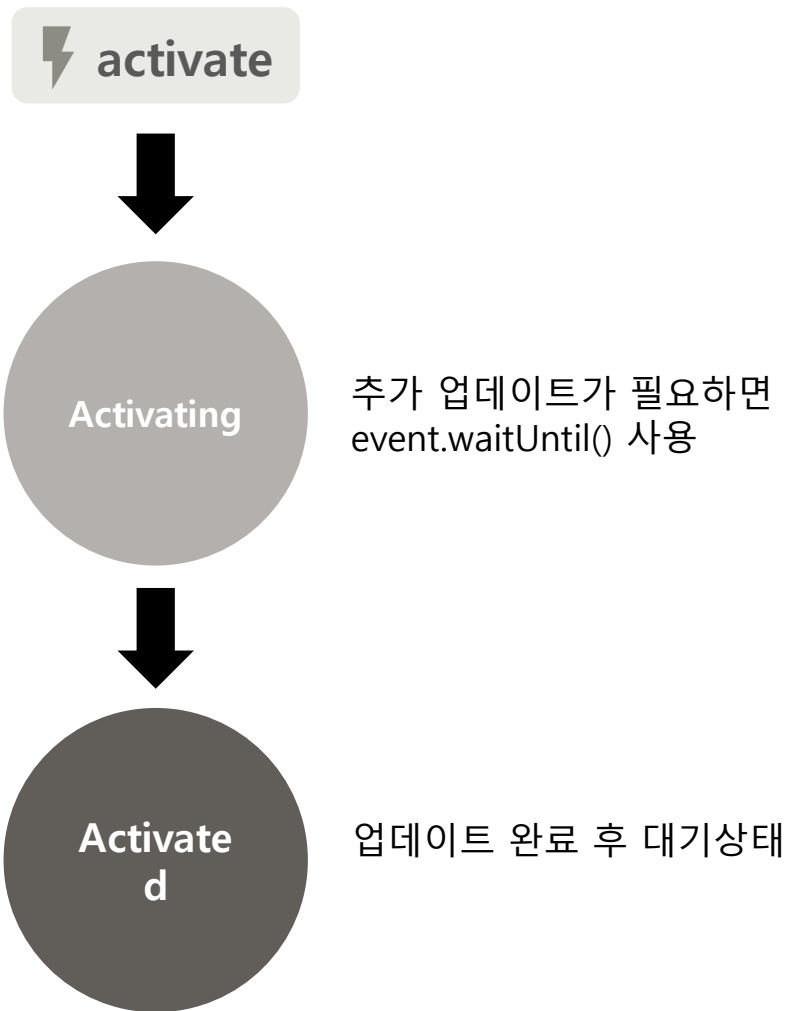


③ 서비스 워커 업데이트 - activate 이벤트

```
// 고유 번호 할당받은 서비스 워커 동작 시작  
self.addEventListener('activate', pEvent => {  
  console.log('서비스워커 동작 시작됨!');  
});
```



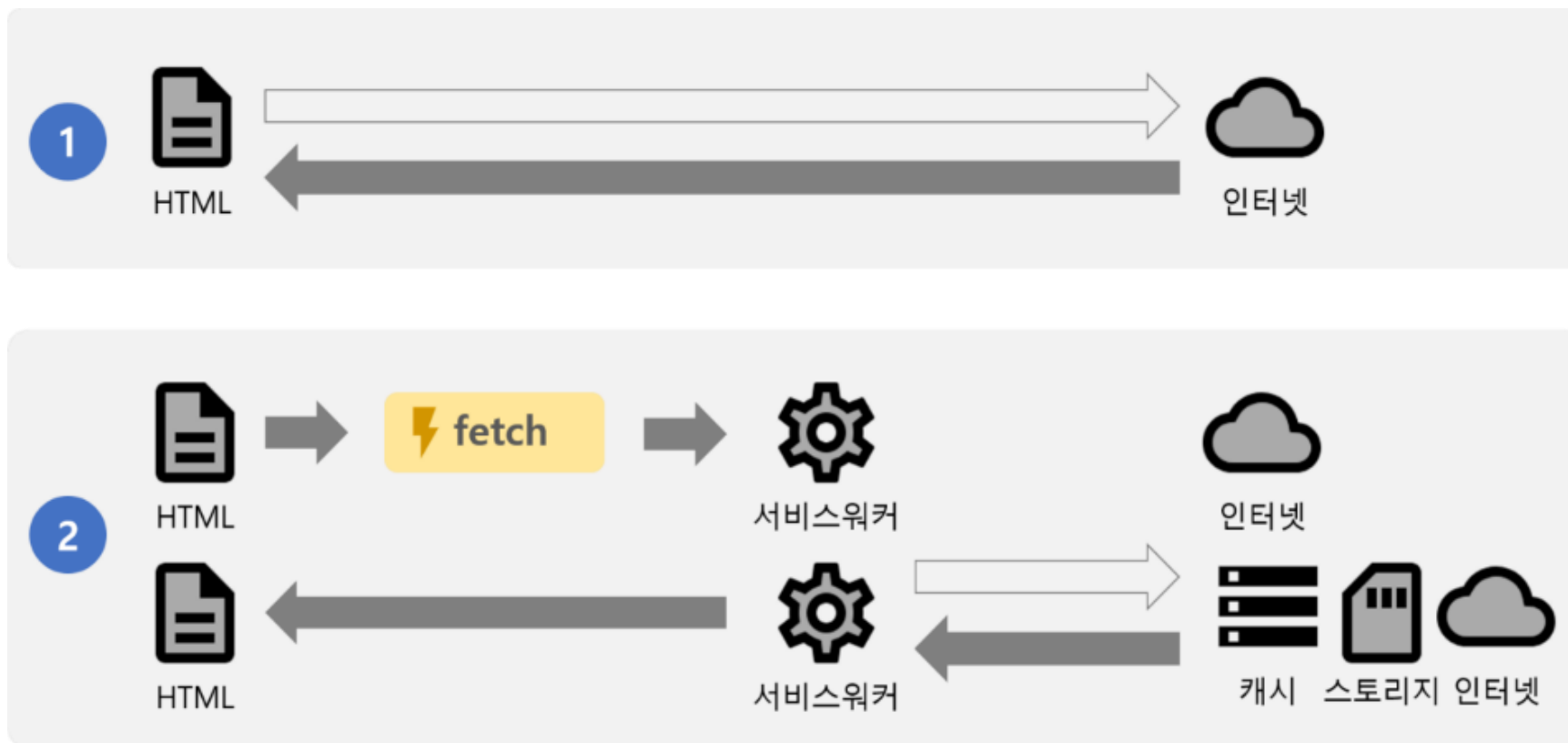
| ③ 서비스 워커 업데이트 - activate 이벤트



④ 오프라인 전환 시 동작 - fetch 이벤트

```
// 고유 번호를 할당 받은 서비스워커 작동
self.addEventListener('fetch', pEvent => {
  pEvent.respondWith(
    caches.match(pEvent.request)
    .then(response => {
      if(!response){
        console.log("네트워크로 데이터 요청!", pEvent.request)
        return fetch(pEvent.request)
      }
      console.log("캐시에서 데이터 요청!", pEvent.request)
      return response;
    }).catch(err => console.log(err))
  );
});
```


| ④ 오프라인 전환 시 동작 - fetch 이벤트



| 서비스 워커의 주요 이벤트 복습하기

install

- 서비스워커가 처음 설치될 때 실행
- 캐시 파일 저장

activate

- 서비스워커 설치 완료 시 실행
- 서비스워커의 업데이트 담당
- 기존 캐시 제거

fetch

- 서비스워커 실행 시 작업할 내용 작성
- 브라우저가 서버에 HTTP 요청 시 오프라인 상태면 캐시 파일 읽기

| 서비스 워커 & 캐시 데이터 삭제

The screenshot shows the Chrome DevTools Application tab. The left sidebar has the 'Storage' section selected. The main panel displays the 'Storage' overview for the URL `http://127.0.0.1:5500/`. It shows a usage of 19.5 kB out of a 307265 MB quota. A donut chart breaks down the usage into 18.4 kB for Cache storage and 1.1 kB for Service Workers. At the bottom, there is a 'Clear site data' button and a checkbox for 'including third-party cookies'.

Application

- Manifest
- Service Workers
- Storage

Storage

- Local Storage
- Session Storage
- IndexedDB
- Web SQL
- Cookies
- Trust Tokens
- Interest Groups

Cache

- Cache Storage
- Back/forward cache

Background Services

- Background Fetch
- Background Sync

Storage

`http://127.0.0.1:5500/`

Usage

19.5 kB used out of 307265 MB storage quota

[Learn more](#)

19.5 kB

18.4 kB Cache storage

1.1 kB Service Workers

19.5 kB Total

☐ Simulate custom storage quota

[Clear site data](#) ☐ including third-party cookies

Application

[실습 4] index 문서 작성 (index.html)

```
<!DOCTYPE html>
<!-- 언어를 한글로 설정 -->
<html lang="ko">

<head>
  <meta charset="utf-8">
  <!-- 매니페스트 연결, 테마색상 변경 -->
  <link rel="manifest" href="manifest.json">
  <meta name="theme-color" content="#ffffff">
```

[실습 4] index 문서 작성 (index.html)

<!-- 뷰포트, 파비콘 설정-->

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="shortcut icon" href="images/icons/favicon.ico">

<link rel="icon" type="image/png" sizes="16x16" href="images/icons/favicon-16x16.png">

<link rel="icon" type="image/png" sizes="32x32" href="images/icons/favicon-32x32.png">

<title>PWA by JS</title>

[실습 4] index 문서 작성 (index.html)

```
<style>
html,
body {
    height: 100%; /* html, body 높이를 100%로 고정 */
    background-color: #F3A530;
    color: #ffffff;
}
```

[실습 4] index 문서 작성 (index.html)

```
.container {  
    height: 100%;           /* 높이를 100%로 고정 */  
    display: flex;          /* 컨테이너를 flexbox 로 변경 */  
    justify-content: center; /* 가운데 정렬 */  
    align-items: center;     /* 중간 정렬 */  
}  
</style>  
</head>
```

[실습 4] index 문서 작성 (index.html)

```
<body>
  <div class="container">
    <h1>Hello, PWA!</h1>
    
    <p>by Javascript</p>
  </div>
```


[실습 4] index 문서 작성 (index.html)

```
<!-- 서비스 워커 등록 -->
<script>
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker
      .register('./service_worker.js')
      .then(function () {
        console.log('서비스 워커가 등록됨!');
      })
  }
</script>
</body>
</html>
```

| ① 언어 설정

언어를 미리 설정하지 않을 경우

- PWA를 실행할 때마다 '다른 언어로 번역하시겠습니까?'라는 질문을 받음
- 사용자 경험 크게 감소

```
<!DOCTYPE html>  
<!-- 언어를 한글로 설정 -->  
<html lang="ko">
```

② 매니페스트 파일 연결

매니페스트 파일 연결

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

상태 표시줄을 흰색으로 설정

```
<meta name="theme-color" content="#ffffff">
```

③ 뷰포트 설정

width=device-width

- 기기의 해상도로 width 자동 설정

initial-scale=1

- 처음 보여지는 크기를 width의 100%로 설정

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

④ 파비콘 설정



```
<link rel="shortcut icon" href="images/icons/favicon.ico">  
  <link rel="icon" type="image/png" sizes="16x16"  
href="images/icons/favicon-16x16.png">  
  <link rel="icon" type="image/png" sizes="32x32"  
href="images/icons/favicon-32x32.png">
```

| ⑤ 스타일 작성

앱에 적용될 디자인 작성

```
<style>  
  html, body {  
  
  }  
  
  .container {  
  
  }  
</style>
```

⑤ 스타일 작성 - 플렉스 박스 설정

플렉스 박스

- 모바일 기기의 크기를 자동으로 고려하여 최적의 레이아웃을 배치 가능
- html, body의 높이를 반드시 100%로 고정

```
html, body {  
  height: 100%; /* html, body 모두 높이를 100%로 고정 */  
  background-color: #F3A530;  
  color: #ffffff;  
}
```

⑤ 스타일 작성 - 화면 요소 배치



```
.container {  
  height: 100%;           /* 높이를 100%로 고정 */  
  display: flex;          /* 컨테이너를 flexbox 로 변경 */  
  justify-content: center; /* 가운데 정렬 */  
  align-items: center;     /* 중간 정렬 */  
}
```


⑥ 내용 요소 작성

화면에 출력될 내용 작성

```
<div class="container">  
  <h1>Hello, PWA!</h1>  
    
  <p>by Javascript</p>  
</div>
```

⑦ 서비스 워커 등록

서비스워커를 등록하기 위한 스크립트 작성

```
<!-- 서비스 워커 등록 -->
<script>
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker
      .register('./service_worker.js')
      .then(function () {
        console.log('서비스 워커가 등록됨!');
      })
  }
</script>
```

| ⑦ 서비스 워커 등록

navigator.serviceWorker

- ServiceWorkerContainer라는 읽기 전용의 객체가 반환
- 이 객체 안에 있는 register() 메서드를 이용해 모바일 브라우저가 서비스 워커를 지원하는지 확인한 후 서비스 워커를 등록

register() 메서드

- 서비스 워커(service_worker.js)를 index.html 파일에 등록

then() 메서드

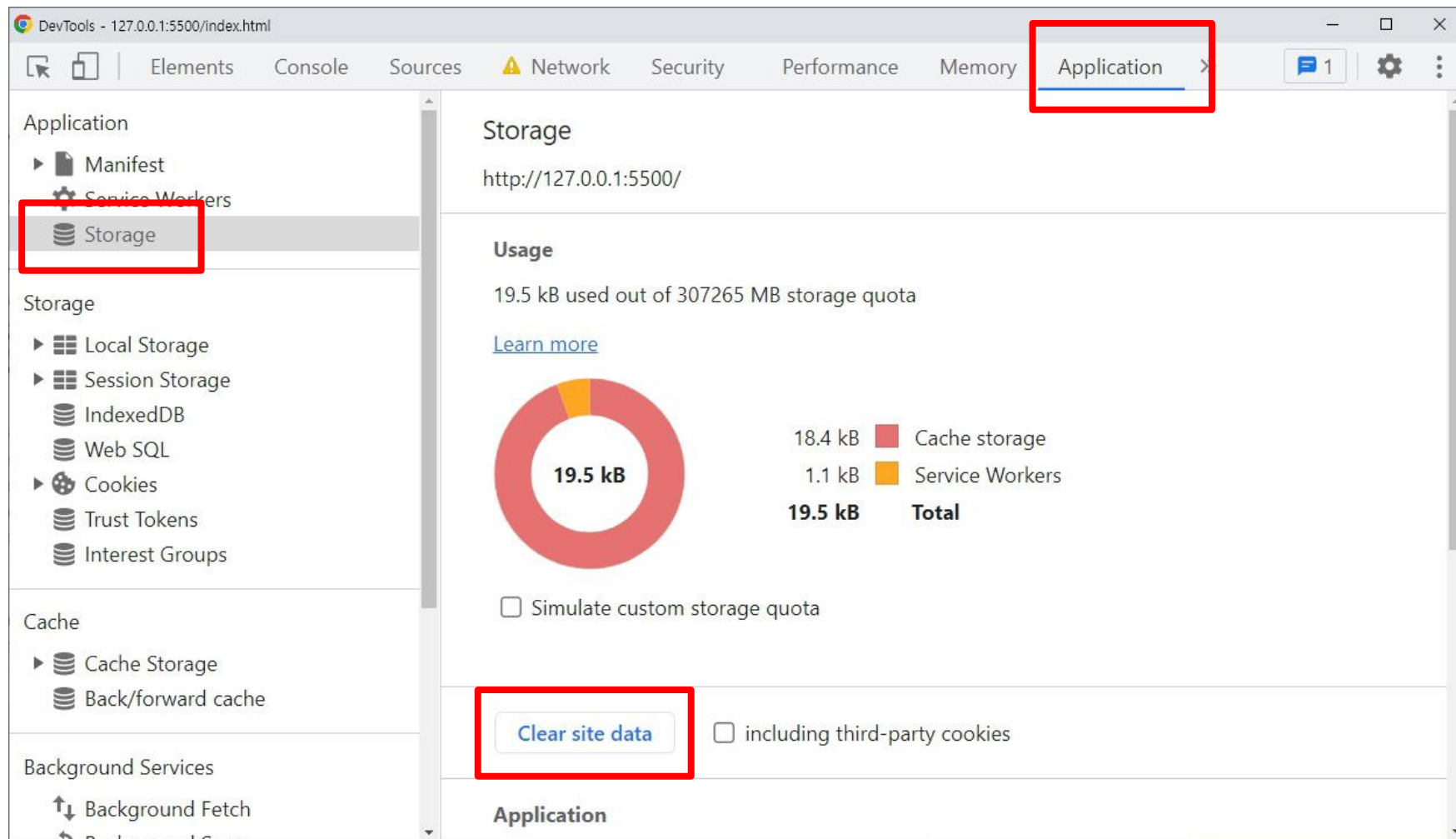
- register() 메서드의 실행이 성공하면 실행
- register() 메서드가 실행되면 콘솔에 성공 메시지를 출력

[실습 5] 콘솔 메시지, 캐시 동작 확인

콘솔 메시지, 캐시 동작 확인

- index.html 을 Go Live로 실행
- 개발자 도구 실행
- Application 탭에서 서비스워커 동작 확인
- Network 탭을 온라인 상태에서 캐시된 파일 확인 & 브라우저 파비콘 확인
- Network 탭을 오프라인으로 변경 후 새로 고침
- Network 탭에서 캐시된 파일 확인 & 브라우저 파비콘 확인

[실습 6] 서비스 워커 삭제하고 프로그램 종료



[실습 6] 캐시 변경하고 서비스 워커 다시 등록하기

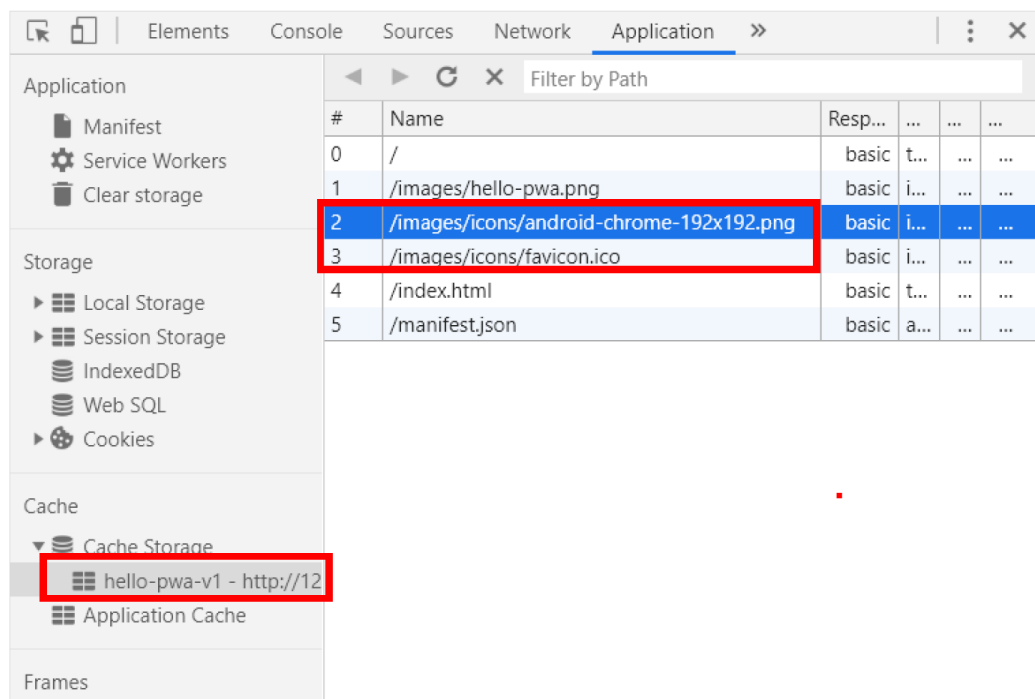
콘솔 메시지, 캐시 동작 확인

- index.html 을 Go Live로 실행
- 개발자 도구 실행
- Application 탭에서 캐시 내용 확인
- 서비스워커 파일 수정(내용 임의)
- 페이지 새로고침 후 개발자 도구 Application 탭에서 변경된 내용 확인
- 캐시 삭제 후 새로고침하여 확인

[실습 6] 캐시 변경하고 서비스 워커 다시 등록하기

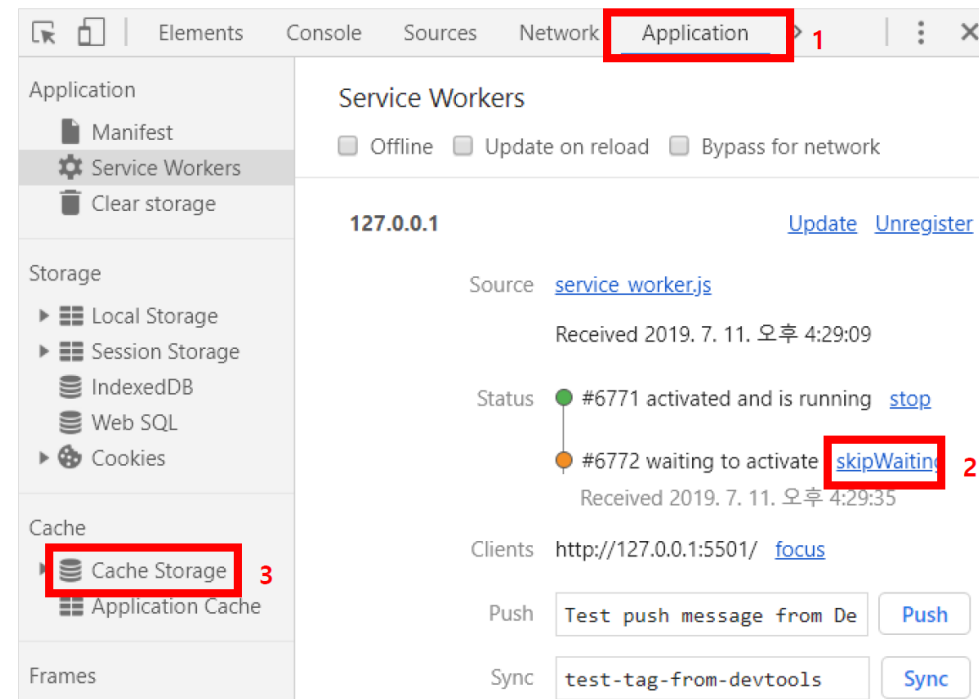
```
const sCacheName = "hello-pwa-v2"; // 캐시 제목
const aFilesToCache = [ // 캐시할 파일 지정
  './',
  './index.html',
  './manifest.json',
  './images/hello-pwa.png',
  './images/hello-pwa.png', // 파비콘 이미지 캐시
  './images/icons/android-chrome-192x192.png', // 아이콘 이미지 캐시
];
```

[실습 6] 캐시 변경하고 서비스 워커 다시 등록하기



The screenshot shows the Chrome DevTools Application panel. The left sidebar has a red box around 'hello-pwa-v1 - http://127.0.0.1:5501/'. The main panel shows a table of cached resources. A red box highlights the first three rows of the table.

#	Name	Resp...
0	/	basic	t...
1	/images/hello-pwa.png	basic	i...
2	/images/icons/android-chrome-192x192.png	basic	i...
3	/images/icons/favicon.ico	basic	i...
4	/index.html	basic	t...
5	/manifest.json	basic	a...



The screenshot shows the Chrome DevTools Application panel with the 'Service Workers' tab selected. A red box highlights the 'Application' tab in the top bar. The 'Service Workers' section shows two workers. The second worker, #6772, is in a 'waiting to activate' state, and a red box highlights the 'skipWaiting()' button next to it. The 'Cache' section on the left has a red box around 'Cache Storage' with a red '3' next to it.

Service Workers

☐ Offline ☐ Update on reload ☐ Bypass for network

127.0.0.1 [Update](#) [Unregister](#)

Source [service_worker.js](#)

Received 2019. 7. 11. 오후 4:29:09

Status ● #6771 activated and is running [stop](#)

● #6772 waiting to activate [skipWaiting\(\)](#) 2

Received 2019. 7. 11. 오후 4:29:35

Clients [http://127.0.0.1:5501/](#) [focus](#)

Push [Push](#)

Sync [Sync](#)

Cache

3 [Cache Storage](#)

[Application Cache](#)

[실습 6] 캐시 변경하고 서비스 워커 다시 등록하기

The screenshot shows the Chrome DevTools Application tab. The left sidebar has the 'Storage' item selected under the 'Application' section. The main panel displays the 'Storage' section for the URL 'http://127.0.0.1:5500/'. It shows a usage of 19.5 kB out of a 307265 MB storage quota. A donut chart breaks down the usage into 18.4 kB for Cache storage and 1.1 kB for Service Workers. At the bottom, there is a 'Clear site data' button and a checkbox for 'including third-party cookies'.

Application

- Manifest
- Service Workers
- Storage

Storage

- Local Storage
- Session Storage
- IndexedDB
- Web SQL
- Cookies
- Trust Tokens
- Interest Groups

Cache

- Cache Storage
- Back/forward cache

Background Services

- Background Fetch

Storage

http://127.0.0.1:5500/

Usage

19.5 kB used out of 307265 MB storage quota

[Learn more](#)

19.5 kB

18.4 kB Cache storage

1.1 kB Service Workers

19.5 kB Total

☐ Simulate custom storage quota

[Clear site data](#) ☐ including third-party cookies

Application