

Progressive Web App

김순곤

soongon@hucloud.co.kr

전체목차

- 1st Day :
 - Progressive Web App 개요
 - 기본 튜토리얼
- 2nd Day :
 - manifest.json
 - Service Worker
- 3rd Day :
 - Service Worker 고급 tooling
 - 성능 최적화

1. 모바일 앱 개발 방식

모바일 앱 만드는 방법

1. 웹앱

- HTML5, CSS3, Javascript - Web 표준을 사용해서 만드는 모바일용 웹
- CSS 프레임워크 (Bootstrap), JavaScript 라이브러리 사용

2. 하이브리드 앱

- 웹앱을 네이티브 디바이스(안드로이드, 아이폰) 용으로 포팅
- Apache Cordova(코르도바), 아이오닉 등.

3. 네이티브 앱

- 네이티브 디바이스에 맞게 해당 기술을 이용해 프로그래밍
- 안드로이드(java), 아이폰(swift), 윈도우폰(c#)

모바일 앱 개발 방식 - WebApp

- 웹앱 (Web Application) - 브라우저 주소를 통해 접속하는 방식
 - 모바일용 웹사이트
 - Mobile-Centric or Mobile-First
 - 웹 사이트는 웹 브라우저용 뿐만 아니라 모바일 용도 같이 개발 필요
 - 모바일을 먼저 기획하고 브라우저 웹으로 변환(Mobile First)
 - 솔루션으로 반응형 웹 (Responsive Web, RWD)
 - HTML5 + CSS3 + JavaScript --> 표준 웹 기술 사용
 - ▶ UI 프레임워크
 - ▶ Bootstrap, Framework7, KENDO UI 등 사용

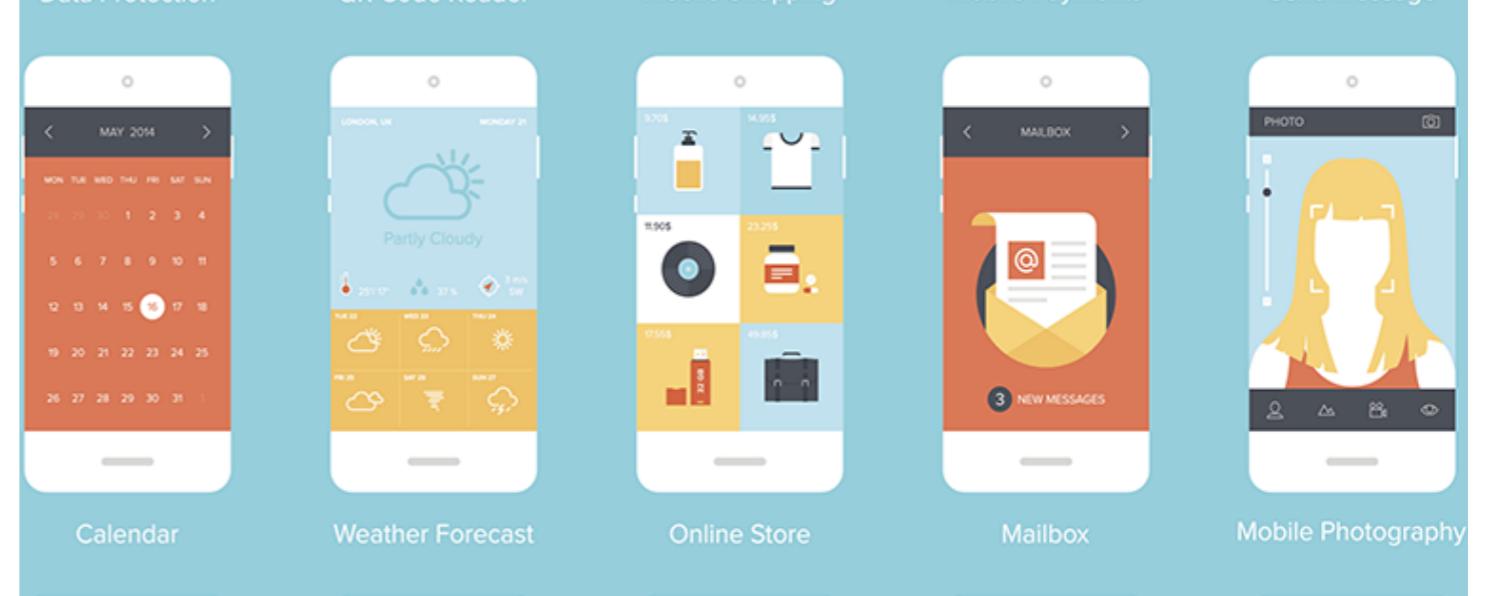
모바일 앱 개발방식 - 네이티브 앱

- 안드로이드 앱
 - Android Studio 개발툴 사용 - Java와 Kotlin 언어 사용
- 아이폰 앱
 - XCode 개발툴 사용 - 맥에서 개발, Swift 언어 사용
- 원도우폰 앱
 - Visual Studio 개발툴 사용 - C# 사용

모바일 앱 개발 방식 - 하이브리드 앱

- 웹 표준 기술을 통해 웹앱으로 개발 후 오픈소스 크로스 프레임워크로 이용하여 네이티브 앱으로 변환시켜 배포되는 앱 형식
- 네이비브 앱과 동일한 기능
 - 네이티브 장비에 접근 가능 (카메라, 내부 주소, 가속도 센서 등)
 - 서버 푸시 수신 가능
 - 앱스토어에 등록 및 다운 가능
- 웹 기술로 개발되어 짐
 - HTML, CSS, JavaScript
 - 네이티브 모바일 플랫폼의 WebView에서 실행

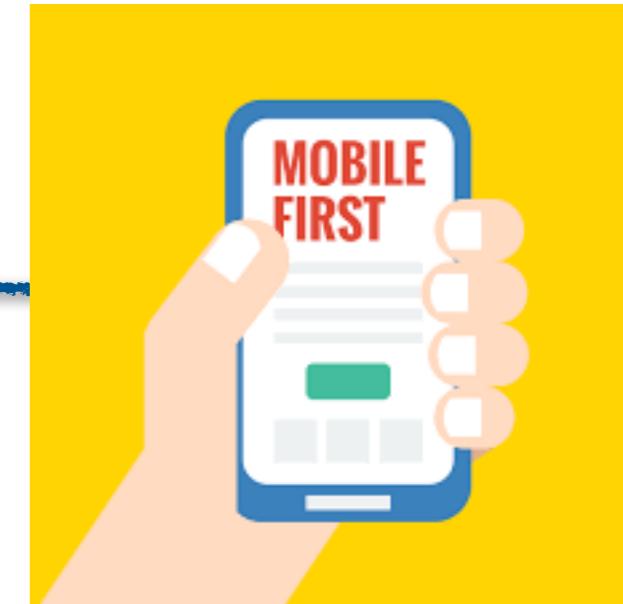




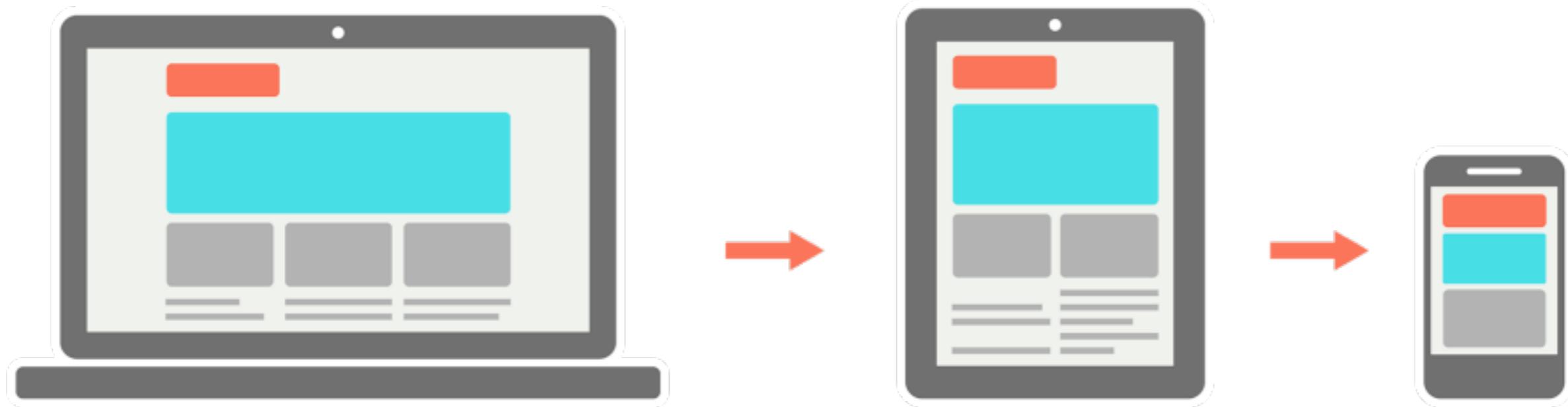
2. 모바일 웹 개발



Mobile First ?

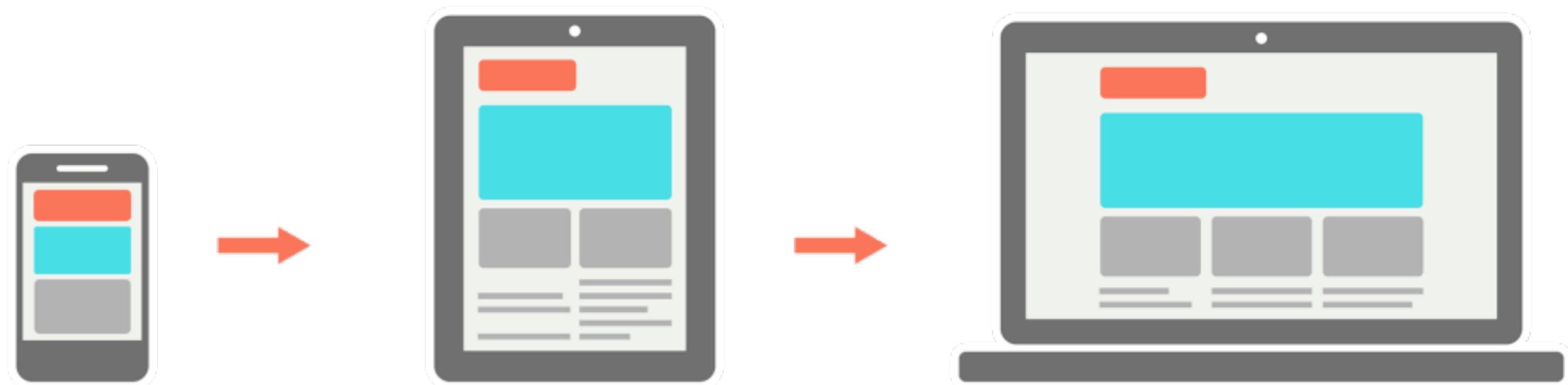


- why?
 - 비즈니스 웹사이트 기획 시 모든 디바이스와 브라우저에 접근 가능하게 기획되고 개발 되어야 함
- Responsive Web Design (RWD)
 - 웹 사이트를 다양한 화면 사이즈에서 보여지게 하는 전략
 - 핸드폰, 태블릿, 데스크탑 브라우저



Responsive Web Design

Mobile First Web Design



개발 시 필수 기술

- HTML5
 - 프론트엔드 웹의 기본, 개발자 디자이너 모두 필수 소양 기술
- CSS3
 - 디자인 관점으로 접근
 - BootStrap 사용으로 더욱 쉽고 편리하게..
- JavaScript
 - 순수 자바스크립트 만으로는 개발이 쉽지 않아요.. (vanilla javascript)
 - jQuery - 자바스크립트를 쉽게 사용할 수 있게 만들어 짐
 - 최근 프론트엔드 웹 전용 프레임워크(라이브러리) : ReactJS, Angular, Vue.js





3. PWA



Progressive Web App (PWA) ?

웹앱을 네이티브 앱과 비슷한 기능과 성능을 가질수 있게 만드는 기법들

- 2015년 구글 크롬 엔지니어 알렉스 러셀이 고안
 - Google I/O 2016에서 소개된 미래의 웹 기술
- 새로운 기술이나 프레임워크가 아님
- 네이티브 앱과 같은 사용자 경험을 갖게 하는 것이 최종 목표
- 점진적 (Progressive) 개선을 통해 네이티브 앱에 가까워짐

Progressive Web App (PWA) ?

- PWA 는 다음 기능을 포함한 웹 사이트
 - 오프라인 지원
 - 빠른 로딩타임
 - 보안성이 좋음
 - 푸시 알림을 받을 수 있음
 - URL 표시바 없이 전체화면으로 볼 수 있음
 - 홈 스크린에 앱 추가 가능

PWA 특징

- 가볍다!
 - 네이티브 앱 : 200M 이상, PWA 수 KBs
- 네이티브 플랫폼 코드가 필요 없다.
- 개발, 빌드, 릴리즈 작업이 쉽다.
- 검색엔진에 의해 검색 가능
- HTML5 웹 표준 앱
 - 반응형 웹사이트 - 모바일 또는 데스크탑 용 앱 개발 가능
 - 오프라인 실행 가능
 - 푸쉬 알림 가능

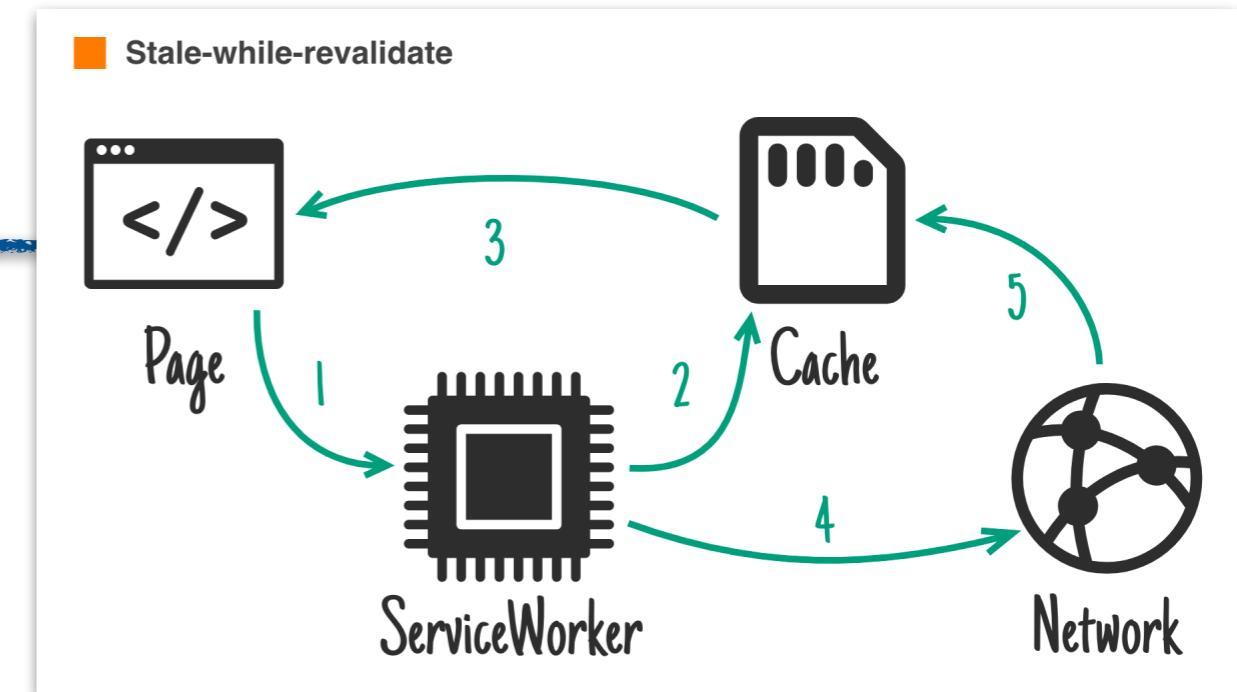
PWA 기본 컨셉

- 반응형
- 네이티브 앱과 유사한 느낌
- 오프라인 지원
- 설치가능
- 재참여 유도
- 검색 가능
- 자체로 업데이트 됨
- 보안성 - HTTPS
- 오래된 브라우저에서도 작동됨 (적은 기능 이라도.. progressive)
- URL 사용하여 쉽게 접근 가능 (앱스토어 설치가 아니라)

PWA 핵심 기술

- Service Worker

- 오프라인 실행을 지원해 주는 기술
 - PWA 핵심이자 필수 기술
 - 크롬, 파이어폭스, 사파리에서 지원 <https://caniuse.com/#feat=serviceworkers>



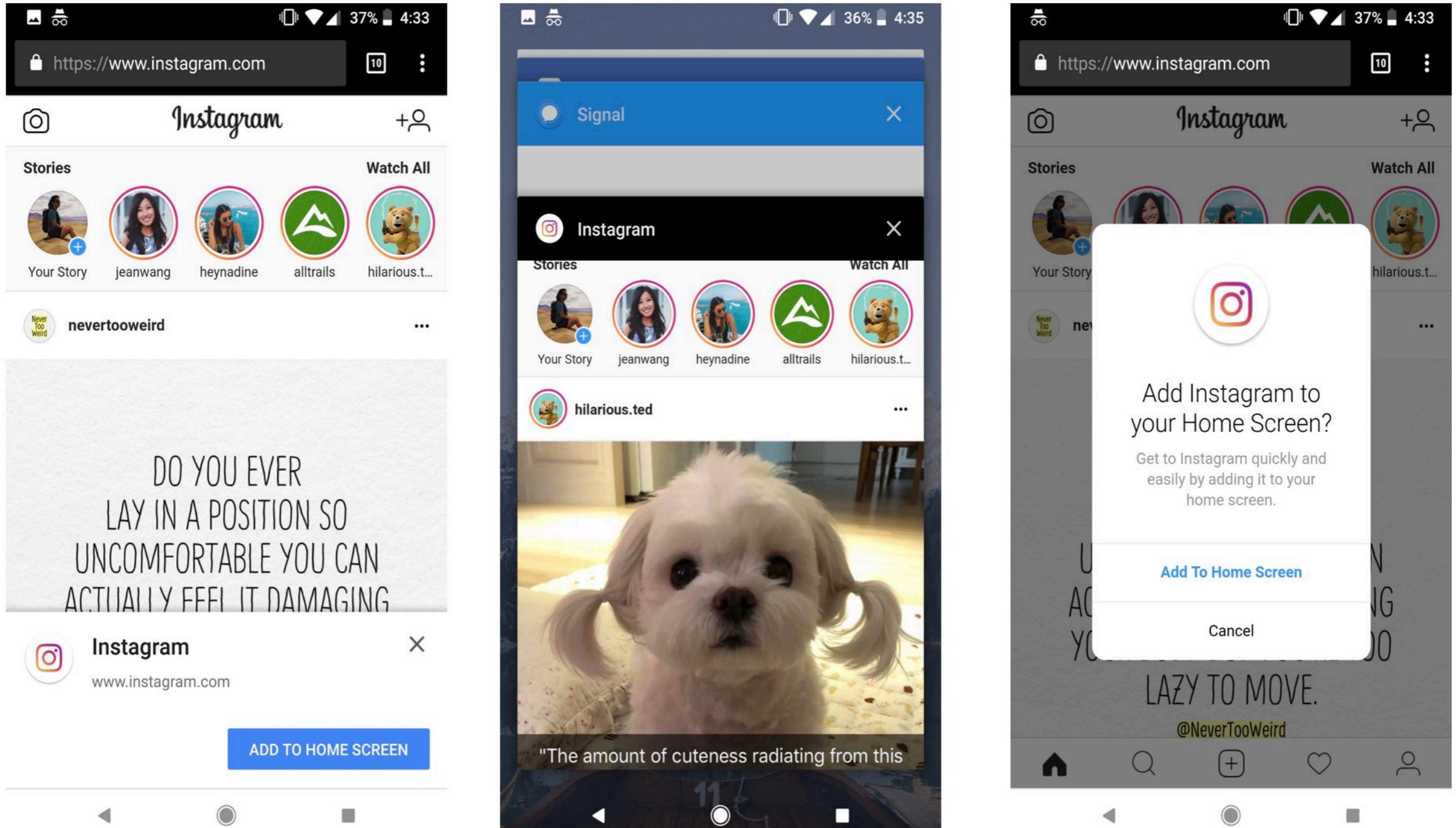
- App Mainfest

- .json 파일로 작성된 PWA 의 설정파일
 - HTML 헤더에 포함시킴 `<link rel="manifest" href="/manifest.webmanifest">`

- App Shell

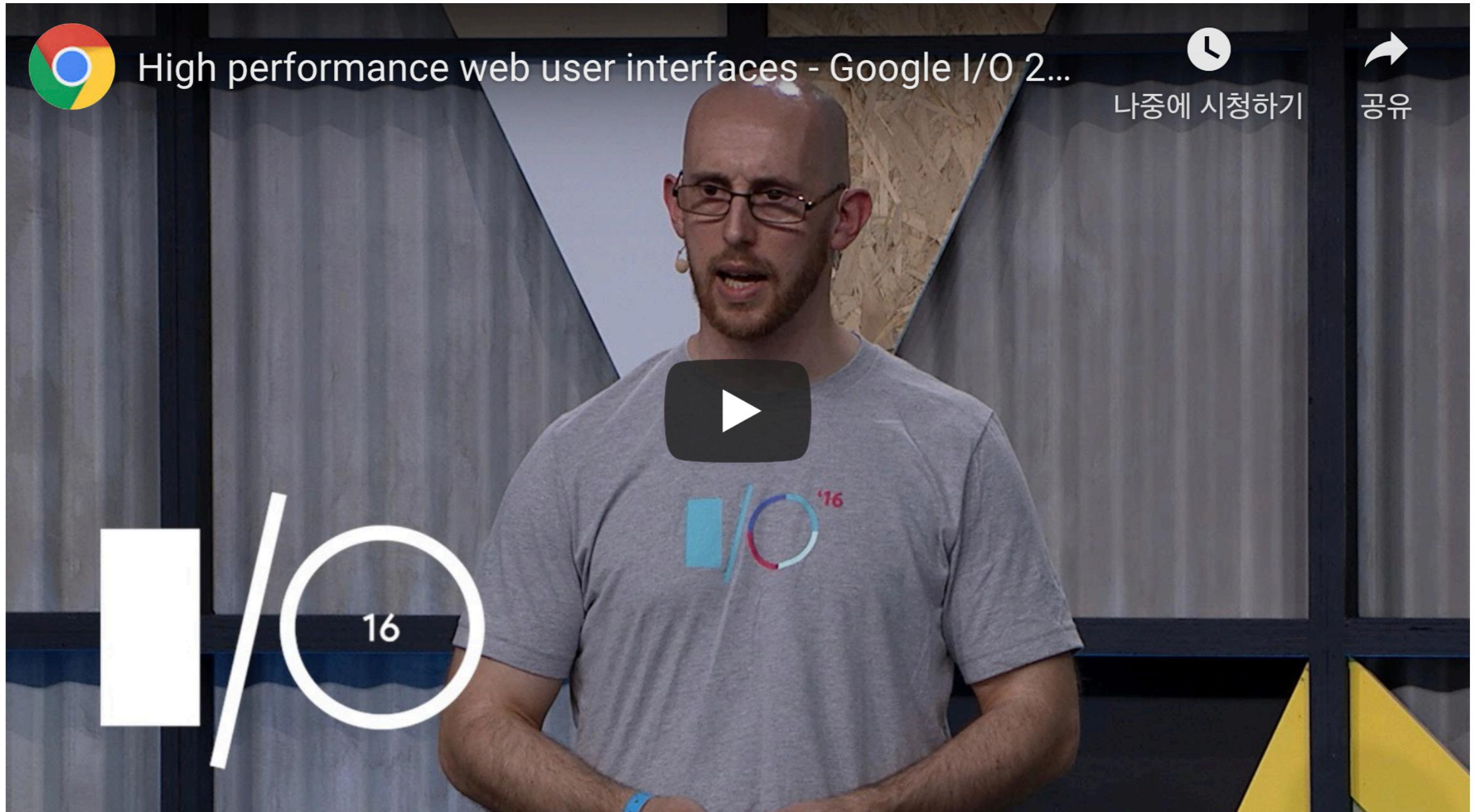
- 기술이 아니라 디자인 컨셉
 - 웹앱의 업데이트를 먼저 로딩하기 위해 고안됨
 - 컨텐츠(데이터)와 분리되어 캐시되어 사용

인스타그램 PWA 사례



구글 PWA 키노트

<https://youtu.be/thNyy5eYfbc>



참고 : MS가 구글과 함께 PWA에 올인하는 이유

<http://www.ciokorea.com/column/39944>

- 윈도우10과 크롬70의 조합으로 PWA를 앱으로 바로 실행
 - 크롬메뉴, 시작메뉴, 작업 표시줄 고정 앱으로 사용
- 실제로 PWA 대부분은 안드로이드 앱 대체재로 사용될 것임
- 마이크로소프트와 구글의 원원전략

참고 : PWA 사례 사이트

○ 성공사례

□ <https://www.pwastats.com/>

The screenshot shows a grid of cards on the PWA Stats website, each detailing a success story for a different company's PWA implementation. The cards include:

- Trebo**: Launched a PWA and saw a 4x increase in conversion rate year-over-year. Conversion rates for repeat users saw a 3x increase and their median interactive time on mobile dropped to 1.5s.
- The Best Western River North Hotel**: Sees 300% increase in revenue with new Progressive Web App.
- OpenSooq**: New PWA is only 28.3KB and has improved engagement with a 25% increase in average time on page resulting in 260% more leads.
- Petlove**: PWA resulted in a 2.8x increase in conversion and a 2.8x increase in time spent on site. Using simplified signup and auto sign back in, 2x more users go to checkout already signed in.
- Grand Velas Riviera Maya resort**: Increased its Black Friday conversion rate by 53% due to its progressive web app's speed and notifications.
- Tinder**: Cut load times from 11.91 seconds to 4.69 seconds with their new PWA. The PWA is 90% smaller than Tinder's native Android app. User engagement is up across the board on the PWA.
- Trivago**: Saw an increase of 150% for people who add its PWA to the home screen. Increased engagement led to a 97% increase in clickouts to hotel offers. Users who go offline while browsing can continue to access the site and 67% continue to browse the site when they come back online.
- Uber**: PWA was designed to be fast even on 2G. The core app is only 50k gripped and takes less than 3 seconds to load on 2G networks.
- West Elm**: Progressive web app saw a 15% increase in average time spent on site and a 9% lift in revenue per visit.
- BookMyShow**: PWA takes less than 3 seconds to load and increased conversion rates over 80%. The PWA is 54x smaller than Android and 1.80x smaller than iOS app.
- Ele.me**: Biggest food ordering and delivery company in mainland China, decreased load times by 11.6% on pre-cached pages and by 6.35% on all pages.

Each card includes relevant hashtags such as #Conversions, #Offline, #Performance, #Service Worker, #Travel, #App Shell, #Classified, #Engagement, #Home Screen, #Notifications, #Service Worker, #Leisure, #Home Screen, #E-commerce, #Engagement, #Home Screen, #Social, #IndexedDB, #Leisure, #Notifications, #Service Worker, and #Home Screen.

○ 도입사례

□ <https://developers.google.com/web/showcase/>

Web Mainfest

Web Mainfest

- mainfest.json 파일
- 홈스크린에 추가하기
- 기존 기능을 지원하기 위해 Polyfil 사용하기

Mainfest Generator

- o <https://app-manifest.firebaseio.com/>

The [Web App Manifest](#) is a JSON document that provides application metadata for [Progressive Web Apps](#). Use the form below to generate the JSON file and optionally upload an app icon.

App Name
PWA Market Progressive Web App

Short Name
PWAmarket

Theme Color

Background Color

Display Mode
Standalone

Orientation
Any

Application Scope
.

Start URL
/index.html

manifest.json

```
{  
  "name": "PWA Market Progressive Web App",  
  "short_name": "PWAmarket",  
  "theme_color": "#ffffff",  
  "background_color": "#ffffff",  
  "display": "standalone",  
  "scope": ".",  
  "start_url": "/index.html"  
}
```

COPY

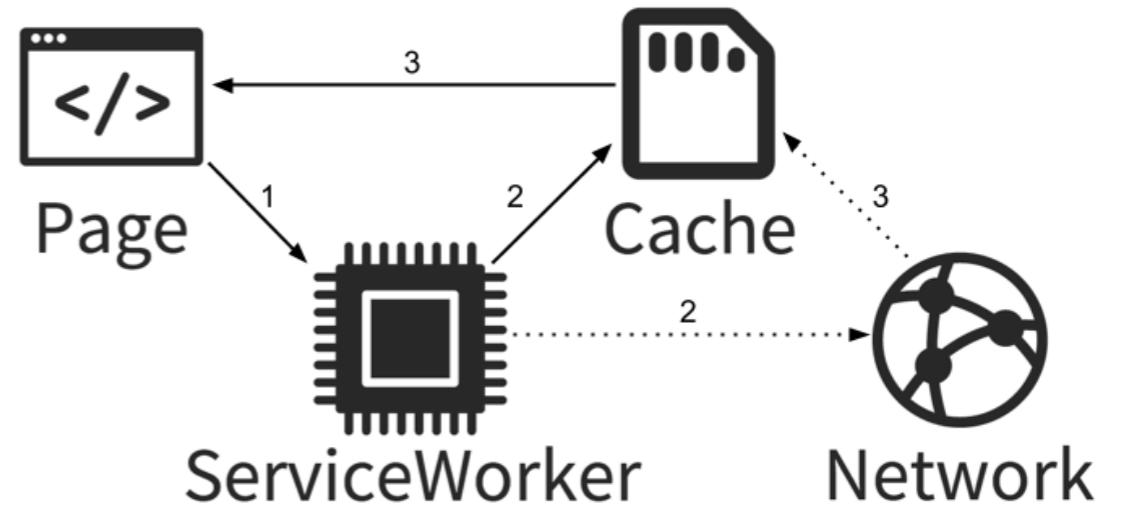
Generate Icons

The Web App Manifest allows for specifying icons of varying sizes. Upload a 512x512 image for the icon and we'll generate the remaining sizes.

ICON

GENERATE .ZIP

Hosted on [Firebase](#).

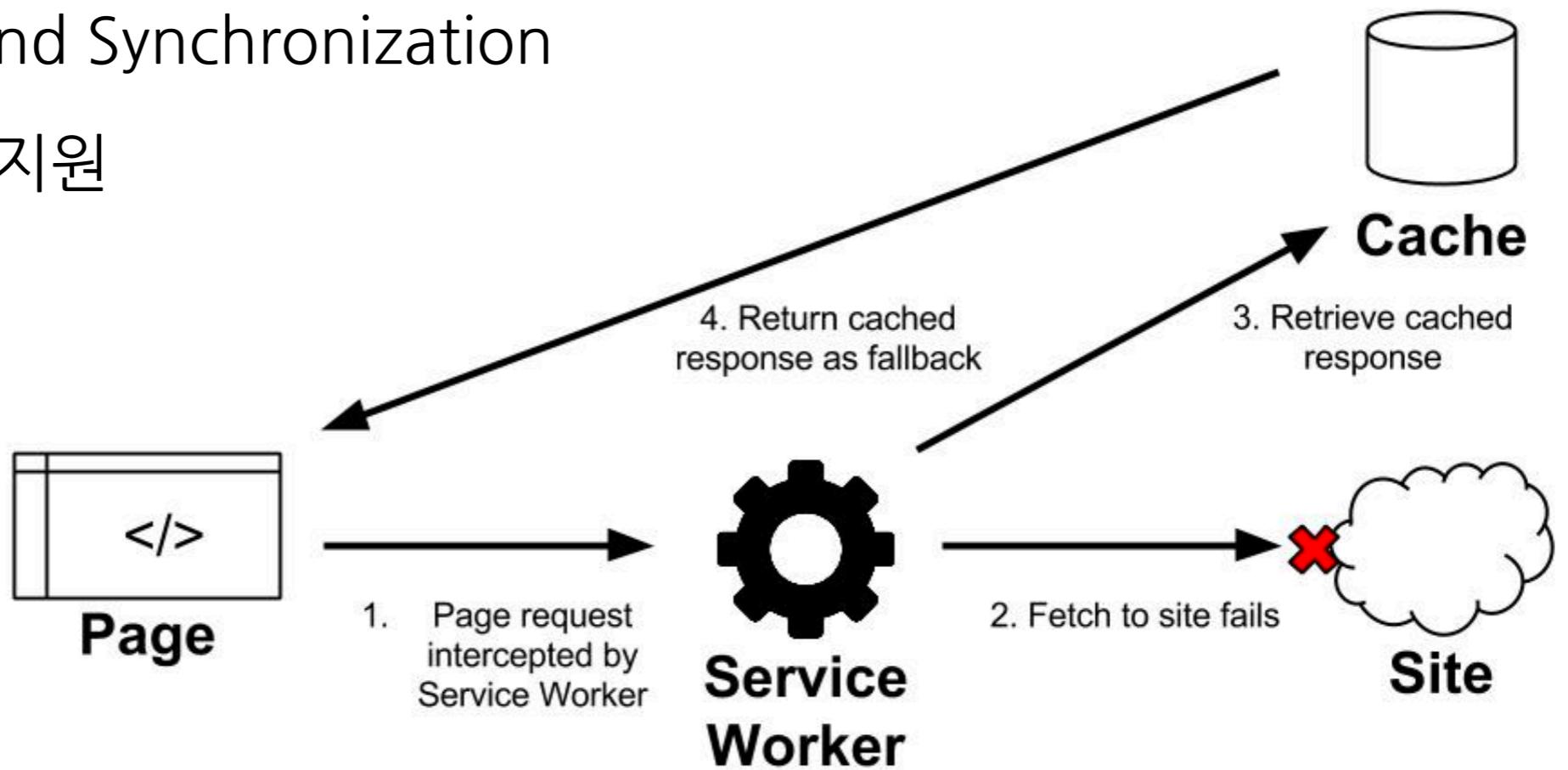


Service Worker



Servie Worker 소개

- PWA 핵심 코어 기술
 - Offline Access 제공
 - Push Notifications
 - Background Synchronization
 - 캐시 기능 지원



Service Worker 특징

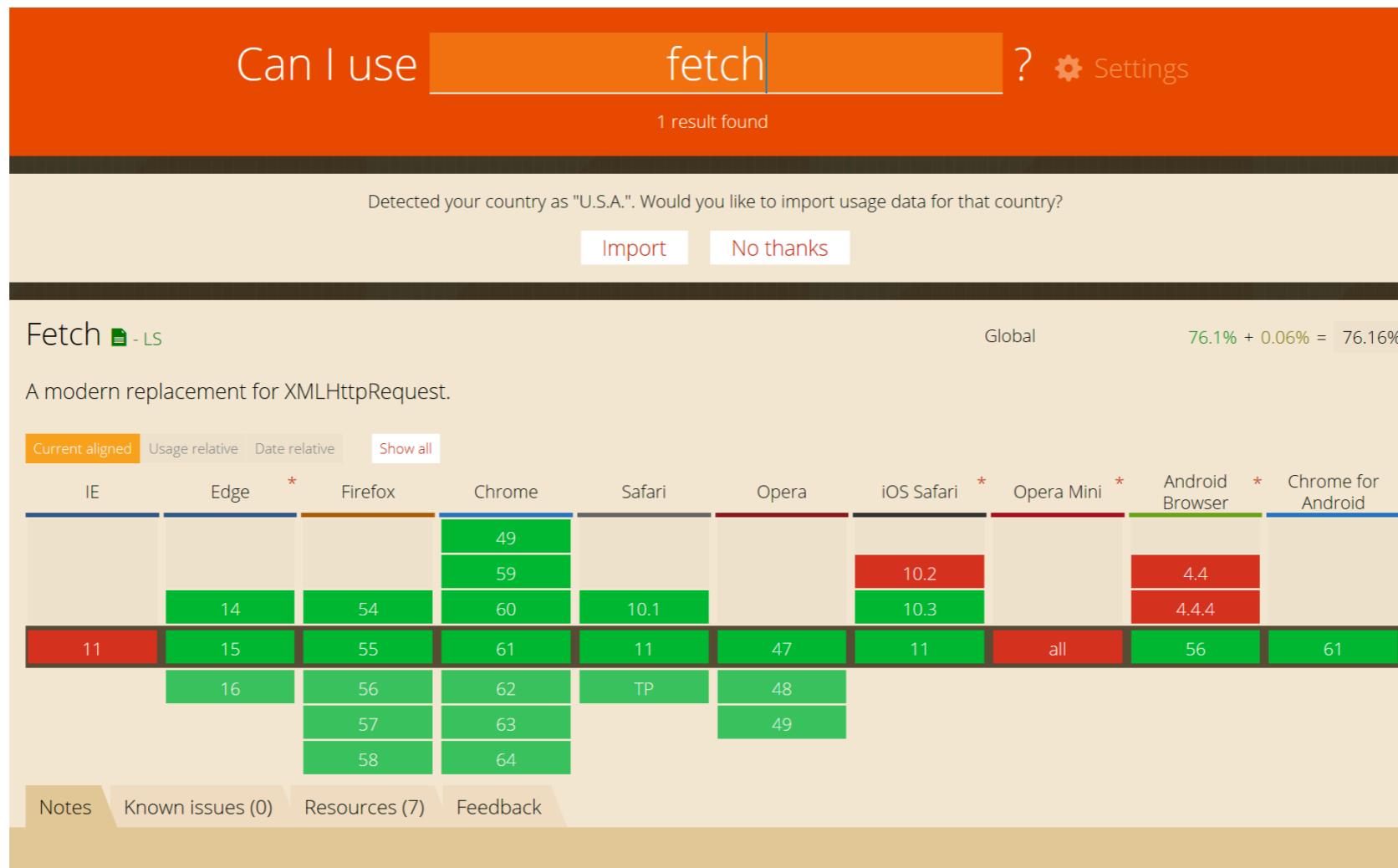
- 독립된 Thread에서 동작
 - 메인 쓰레드(UI 쓰레드)와 분리되어 자신만의 작동 스코프를 가짐
 - DOM 엑세스 불가, Local Storage 와 XHR API 접근 불가
 - 네트워크 요청을 가로채고 처리하는 핵심기능 수행
- 최신 Web API와 작동
 - Fetch API <https://flaviocopes.com/fetch-api/>
 - Cache API <https://flaviocopes.com/cache-api/>
- HTTPS에만 작동됨
 - Localhost에서는 개발모드를 위해 작동됨

Service Worker 쓰레드

- UI쓰레드가 아닌 별도의 쓰레드로 동작
 - DOM 에 접근 불가
 - 이벤트 드리븐 방식 : 브라우저나 OS 가 서비스 워커를 초기화
- UI 쓰레드와 서비스 워커 쓰레드는
 - Messaging API 를 통해 통신
 - 쓰레드 간 텍스트 메세지를 주고 받을 수 있음
 - UI 쓰레드에 서버 메세지가 도달 하기 전 중간에서 수정 가능
- Push Notifications 처리 가능
 - 이전에는 web worker 를 통해 웹기반 푸시 메세지 처리 가능

Fetch API

- AJAX 요청을 위한 API
 - XMLHttpRequest
 - Fetch - Promise 사용 (async~await 지원)

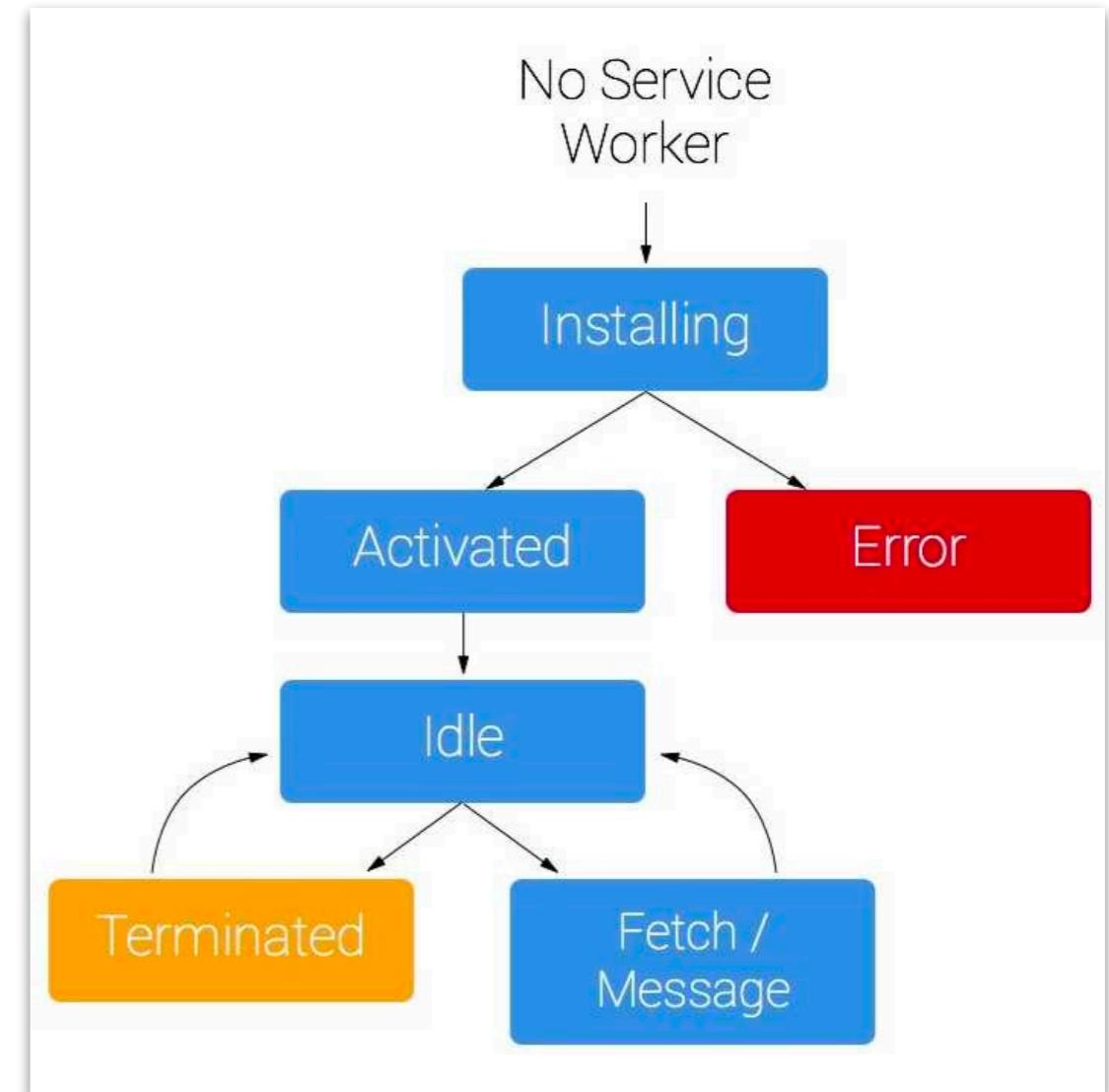


Service worker 업데이트

```
self.addEventListener('install', evt => {
    // install 이벤트 발생 시 작업
});  
  
self.addEventListener('activate', evt => {
    // activate 이벤트 발생 시 작업
});  
  
self.addEventListener('fetch', evt => {
    // fetch 이벤트 발생 시 작업
});
```

Service worker 라이프 사이클

- 웹페이지와 별개로 동작
 - 서비스워커 등록 : Registration
 - 서비스워커 설치 : Installation
 - 서비스워커 활성화 : Activation



Register

Download

Install

Activate

서비스 워커 등록 (Registering)

- 서비스워커는 웹 페이지에 반드시 등록 되어야 함
 - register 메서드 사용

```
if ('serviceWorker' in navigator) {  
  navigator.serviceWorker.register('sw.js')  
    .then(() => console.log('Service Worker Registered..'));  
}
```

- scope 부여 가능

```
register('sw.js', {scope: '/api/'})
```

서비스 워커 설치 (installing)

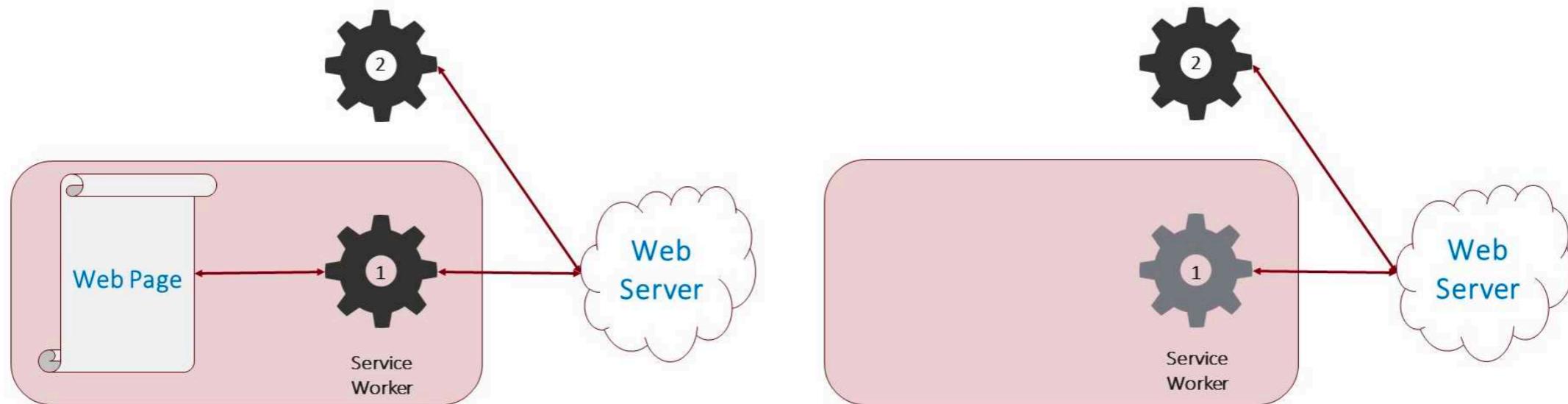
- 설치단계에서 static assets 설치 시도
 - Pre-caching 수행
 - 모든 파일이 성공적으로 캐시되면 서비스 워커가 설치됨
 - 파일 다운로드 및 캐시가 실패하면 설치 단계가 실패하고 활성화 되지 않음
 - 실패 시 다음에 다시 시도하고 설치가 이루어지면 캐시 완료

서비스워크 클라이언트

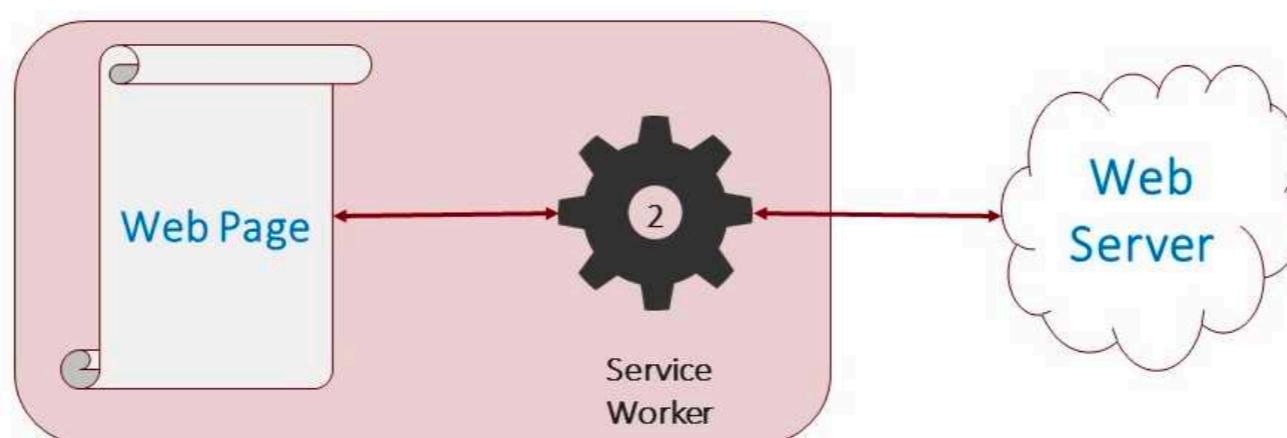
- Service worker clients
 - 서비스워커는 UI쓰레드와 별도의 컨텍스트로 동작
 - 여러 브라우저 탭, 푸시 알림, 백그라운드 싱크 이벤트가 될 수 있음

서비스워커 업데이트

- 새로운 서비스워커가 등록되면 기존 서비스 워커가 종료될 때 까지 기다린다.



- 기존 서비스워커의 모든 클라이언트가 종료되면 새로운 서비스 워커가 활성화 된다.



서비스워커 스코프, Scope

- 서비스워커는 단일 도메인에 제한됨 : 보안기능
 - sw.js 파일은 일반적으로 루트 폴더나 스코프의 루트 폴더에 위치

Service Workers

Offline Update on reload Bypass for network Show all

http://localhost:15001/

Source [sw.js](#)
Received 10/16/2017, 10:48:01 AM
Status ● #1180 activated and is stopped [start](#)

http://localhost:15001/finance/

Source [sw.js](#)
Received 10/16/2017, 10:48:01 AM
Status ● #1181 activated and is stopped [start](#)

http://localhost:15001/marketing/

Source [sw.js](#)
Received 10/16/2017, 10:48:01 AM
Status ● #1182 activated and is stopped [start](#)

http://localhost:15001/hr/

Source [sw.js](#)
Received 10/16/2017, 10:48:01 AM
Status ● #1183 activated and is stopped [start](#)
Clients http://localhost:15001/hr/ [focus](#)

서비스워커 이벤트

- 코어 이벤트, Core Events
 - Install
 - Activate
 - Message
- 기능적 이벤트, Functional Events
 - fetch
 - sync
 - push

Cache API

Caches & Cache

Caches 객체

Caches API

- 네임드 캐시의 컬렉션
 - 데이터 저장 메카니즘
 - 요청 / 응답 객체를 페어로 저장 가능
 - PWA 오프라인 기능을 지원하기 위한 서비스워커 스펙 중 일부
 - 이름이 부여된 캐시객체 (named caches) 를 표현
 - 지원 메서드
 - ▶ open : 이름으로 캐시객체 조회, 없으면 새로 만들어서 반환 (Cache 객체)
 - ▶ match : 요청(request)객체로 조회, 해당 응답이 있으면 프로미스로 반환
 - ▶ has : 네임드 캐시가 있는지 체크, 프로미스로 반환
 - ▶ delete : 캐시 이름으로 삭제, 삭제되면 true 를 포함한 프로미스 반환
 - ▶ keys : 캐시 이름을 배열로 프로미스로 반환

caches.open

- 이름으로 검색 후 Cache 객체를 프로미스로 반환
 - 해당 캐시가 없을 때는 새롭게 생성

```
caches.open(cacheName).then(function (cache) {  
    //do something here  
});
```

caches.match

- Request 를 통해 검색
 - 일치하는 Response 가 있으면 이를 프로미스로 반환
 - 모든 캐시를 검색하여 첫번째 일치하는 것을 반환

```
return caches.match(request).then(function (response) {  
    return response;  
})
```

caches.has()

- 해당 이름의 캐시가 있는지 여부를 프로미스로 반환

```
caches.has(cacheName).then(function(ret) {  
    // true: your cache exists!  
});
```

caches.delete()

- 이름으로 캐시를 삭제
 - 해당 이름의 캐시가 있으면 삭제하고 여부를 프로미스로 반환

```
Cached.delete(cacheName)
  .then((ret) => { console.log(cacheName + " deleted: " + res );}
```

caches.keys()

- 모든 캐시의 이름을 배열로된 프로미스로 반환

```
caches.keys().then(cacheNames => {
  cacheNames.forEach(value => {
    if (value.indexOf(version) < 0) {
      caches.delete(value);
    }
  });
  return;
})
```

Cache 객체

cache.match()

- Cache 인터페이스
 - 저장된 response 에 대한 메서드와 속성을 포함
 - Cache 객체는 직접 만들수 없음 (Caches.open()을 통해 생성됨)
- match 메서드는 request 객체 또는 URL 을 파라미터로 받음
- URL 을 파라미터로 받을 때는
 - 같은 URL 이지만 HEAD나 HTTP 메서드가 다를 수 있으므로
 - options 라는 파라미터를 통해 URL을 차별화 가능

cache.matchAll

- match 메서드와 유사하지만 모든 일치하는 response 를 반환
- 특히 경로를 이용한 검색일 때 유용

```
caches.open("podcasts").then(function(cache) {  
    cache.matchAll('/images/').then(function(response) {  
        response.forEach(function(element, index, array) {  
            //do something with each response/image  
        });  
    });  
});
```

Cache 객체의 add 와 addAll

- 주로 static assets 이나 response 를 즉시 저장하고 싶을 때
 - Request 객체 또는 URL을 파라미터로 사용
 - ▶ 즉, 캐싱 할 데이터의 경로를 전달
 - request - download - store in the cache 과정 진행
 - ▶ put 메서드와 비교

```
const precache_urls = [...]

caches.open(preCacheName).then(function (cache) {
  return cache.addAll(precache_urls);
})
```

cache.put

- put 메서드는 request 와 response 두 개의 파라미터를 가짐
 - key / value 방식임
- response 를 사용하고 싶으면 복사해서 사용
 - response.clone() 을 이용 카피해서 사용
 - response 는 한번 만 사용 가능

```
fetch(request).then(function (response) {  
    var rsp = response.clone();  
    //cache response for the next time around  
    return caches.open(cacheName).then(function (cache) {  
        cache.put(request, rsp);  
        return response;  
    });  
});
```

개발환경 세팅

개발환경 세팅

- 개발환경 세팅
 - 모바일 에뮬레이터 - 모바일 웹 브라우저
 - 웹 서버
 - 에디터 또는 IDE



개발환경 세팅

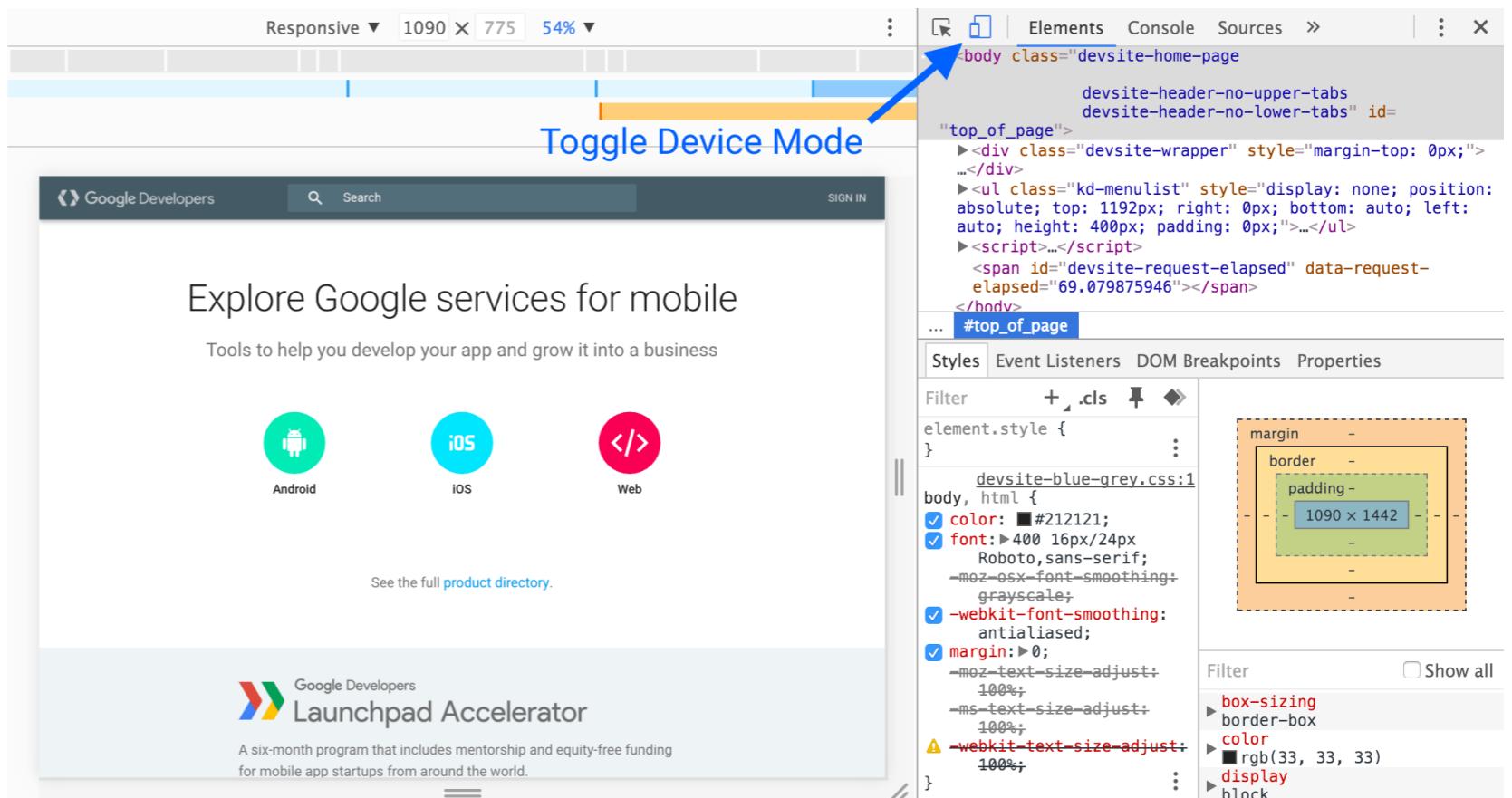
○ 개발환경 세팅

□ 모바일 에뮬레이터 - 모바일 웹 브라우저

▶ 크롬 브라우저 사용 - 크롬 모바일 에뮬레이터

□ 웹 서버

□ 에디터 또는 IDE



개발환경 세팅

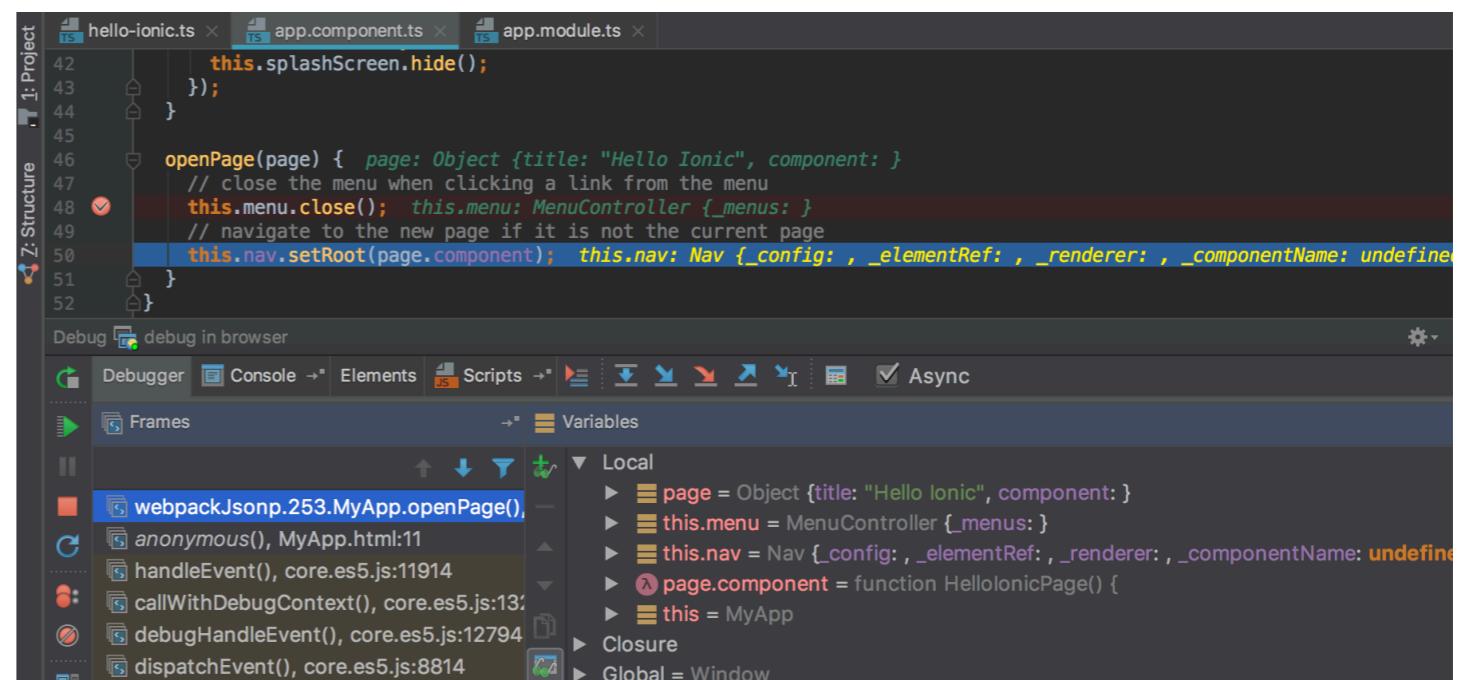
○ 개발환경 세팅

- ## ▣ 모바일 에뮬레이터 - 모바일 웹 브라우저

□ 웹 서버

▶ 웹스톰에 포함됨

▣ 에디터 또는 IDE



개발환경 세팅

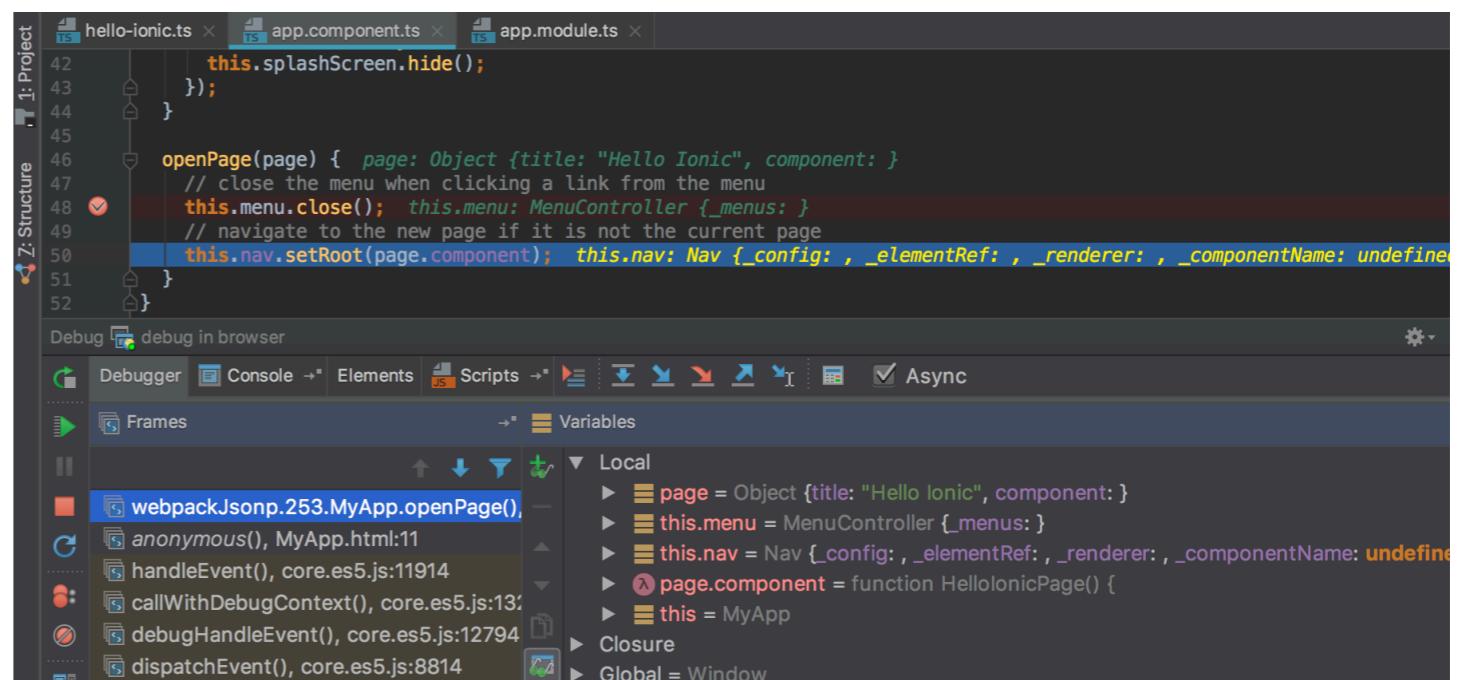
- 개발환경 세팅

- 모바일 에뮬레이터 - 모바일 웹 브라우저

- 웹 서버

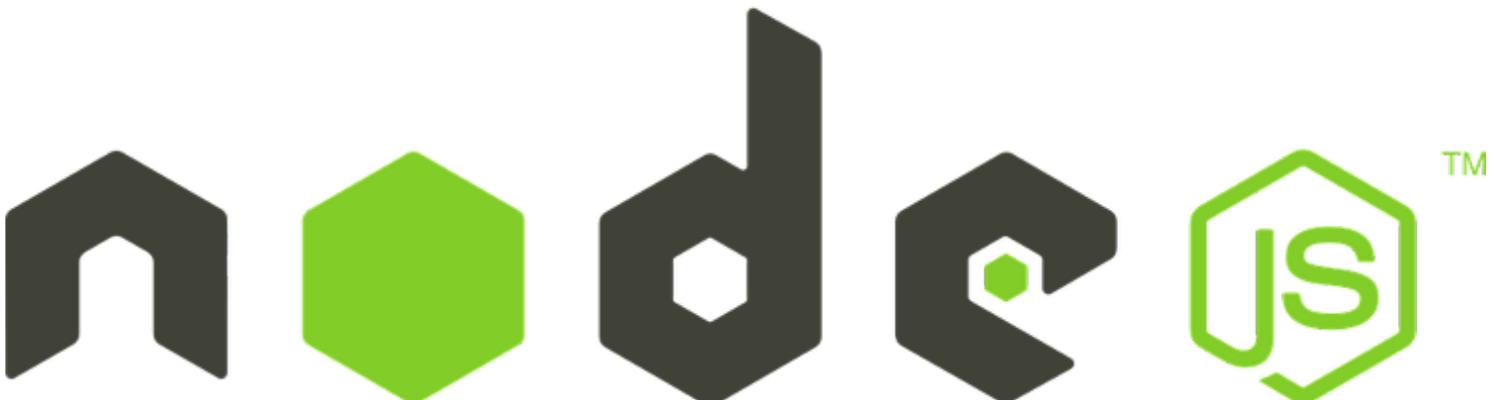
- 에디터 또는 IDE

▶ 웹스톰 사용



개발환경 세팅

- 개발환경 세팅
 - 모바일 에뮬레이터 - 모바일 웹 브라우저
 - 웹 서버
 - 에디터 또는 IDE
 - ▶ 웹스톰 사용
- 추가적으로,
 - node.js : 모바일을 포함한 프론트엔드 개발 생태계의 중심



Tutorial - 1

Hello-world app

- o index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>My hello world page</title>
    <link rel="stylesheet" href="main.css">
  </head>
  <body>
    <h1 class="vertical-container">Hello World</h1>
  </body>
</html>
```

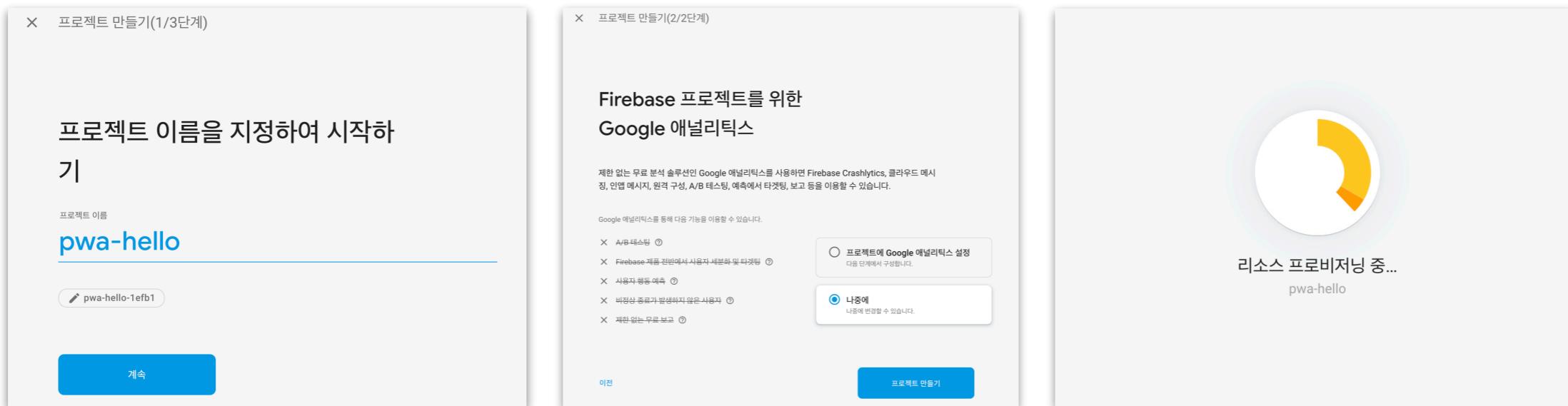
Hello-world app

- main.css - flexbox 사용

```
body {  
    background-color: #FF9800;  
    color: black;  
}  
.vertical-container {  
    height: 300px;  
    display: -webkit-flex;  
    display: flex;  
    -webkit-align-items: center;  
        align-items: center;  
    -webkit-justify-content: center;  
        justify-content: center;  
}  
h1.vertical-container {  
    font-size: 275%;  
}
```

Firebase 에 호스팅 하기

○ Firebase 프로젝트 생성



웹앱에 firebase 추가

The image consists of four screenshots from a Firebase setup wizard, illustrating the process of adding Firebase to a web application.

- Screenshot 1: App Registration**

Step 1: App Registration

App Name: pwa-tutorial-firebase

Also set up Firebase Hosting for this app. [Learn more](#)

Available hosting: pwa-hello-1efb1 (No files yet)

Next
- Screenshot 2: Firebase SDK Integration**

Step 2: Firebase SDK Integration

Script to add to body tag:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="/__/firebase/6.3.4.firebaseio.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#available-services -->

<!-- Initialize Firebase -->
<script src="/__/firebase/init.js"></script>
```

[Web Firebase documentation](#) | [API Reference](#) | [Samples](#)

Next
- Screenshot 3: Firebase CLI Installation**

Step 3: Firebase CLI Installation

Firebase Hosting is required to host your site. You can skip this step if you're not using Firebase Hosting.

Run `npm install -g firebase-tools`

[Firebase CLI documentation](#) | [npm documentation](#)

Next
- Screenshot 4: Deployment**

Step 4: Deployment

Deployment status: Not deployed

Deploy now

Project root directory

Run `$ firebase login`

Run `$ firebase init`

Prepare deployment

Deploy to [pwa-hello-1efb1.web.app](#)

[Deploy now](#)

Firebase 설정

Firebase SDK snippet

자동 ? CDN ? 구성 ?

스크립트를 복사하여 <body> 태그 하단에 붙여넣으세요. Firebase 서비스를 사용하기 전에 진행해야 합니다.

```
<!-- The core Firebase JS SDK is always required and must be listed
<script src="https://www.gstatic.com/firebasejs/6.3.4.firebaseio.js">

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#config-web-app -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyAPGs58zNUIZvkr72Kt5m7w3swz7gKLXBM",
    authDomain: "pwa-hello-1efb1.firebaseio.com",
    databaseURL: "https://pwa-hello-1efb1.firebaseio.com",
    projectId: "pwa-hello-1efb1",
    storageBucket: "",
    messagingSenderId: "922819731967",
    appId: "1:922819731967:web:12dd1dbf2b755b53"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```



Firebase 초기화 하기

- node.js 설치 후 프로젝트 루트 디렉토리에서 실행

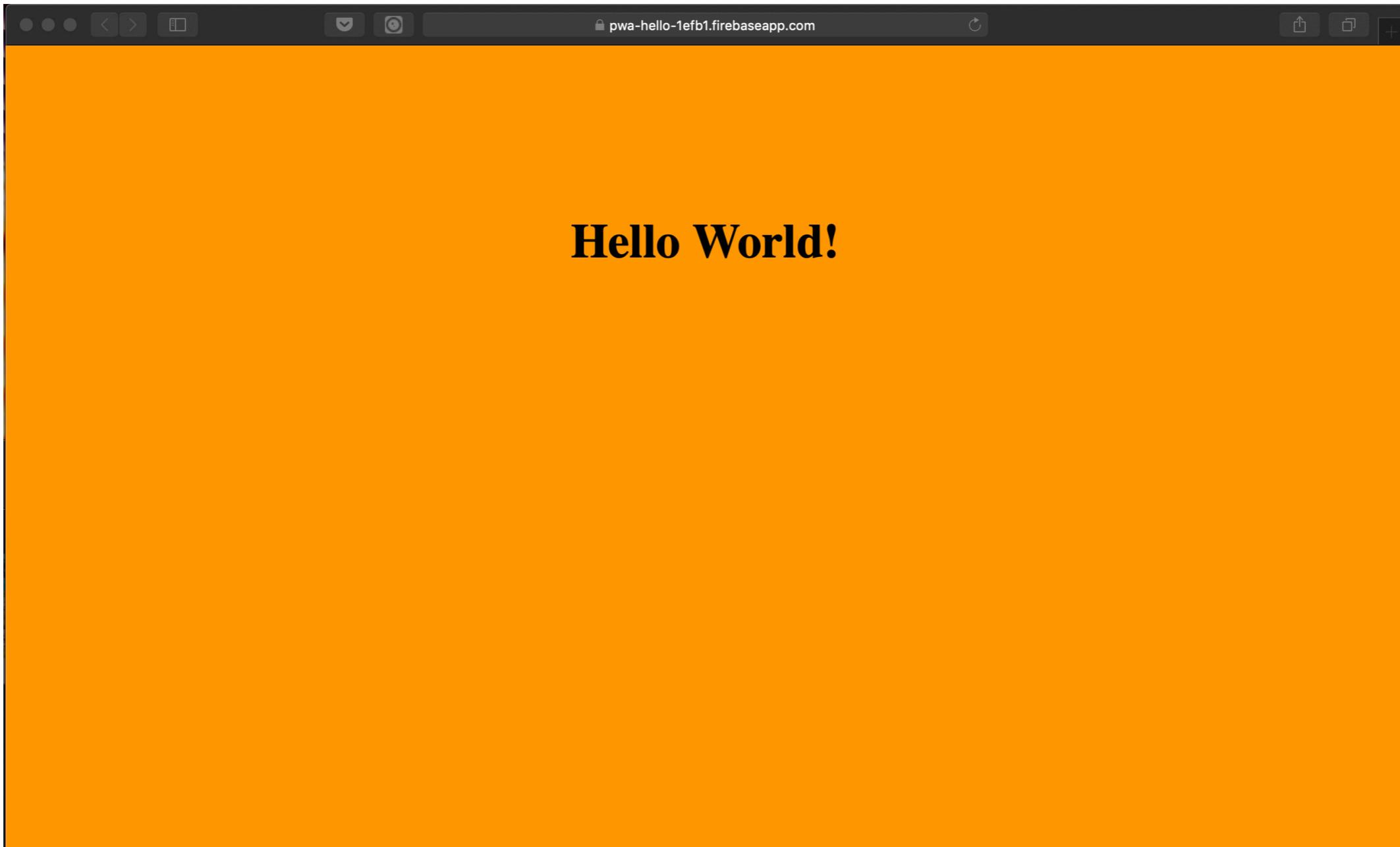
```
$ npm install -g firebase-tools  
$ firebase init # firebase.json 생성 (REQUIRED)
```

Public 디렉토리 설정 필요
Index.html 은 overwrite 하지 않음

```
→ pwa-firebase001 firebase init  
  
#####
##  ##  ##  ##  ##  ##  ##  ##  ##  
##  ##  ##  ##  ##  ##  ##  ##  ##  
##  ##  ##  ##  ##  ##  ##  ##  ##  
##  ##  ##  ##  ##  ##  ##  ##  ##  
##  ##  ##  ##  ##  ##  ##  ##  ##  
  
You're about to initialize a Firebase project in this directory:  
  
/Users/soongon/WebstormProjects/pwa-firebase001  
  
? Which Firebase CLI features do you want to set up for this folder? Press Space  
<i> to invert selection)  
o Database: Deploy Firebase Realtime Database Rules  
o Firestore: Deploy rules and create indexes for Firestore  
o Functions: Configure and deploy Cloud Functions  
o Hosting: Configure and deploy Firebase Hosting sites  
o Storage: Deploy Cloud Storage security rules
```

Firebase 프로젝트 배포

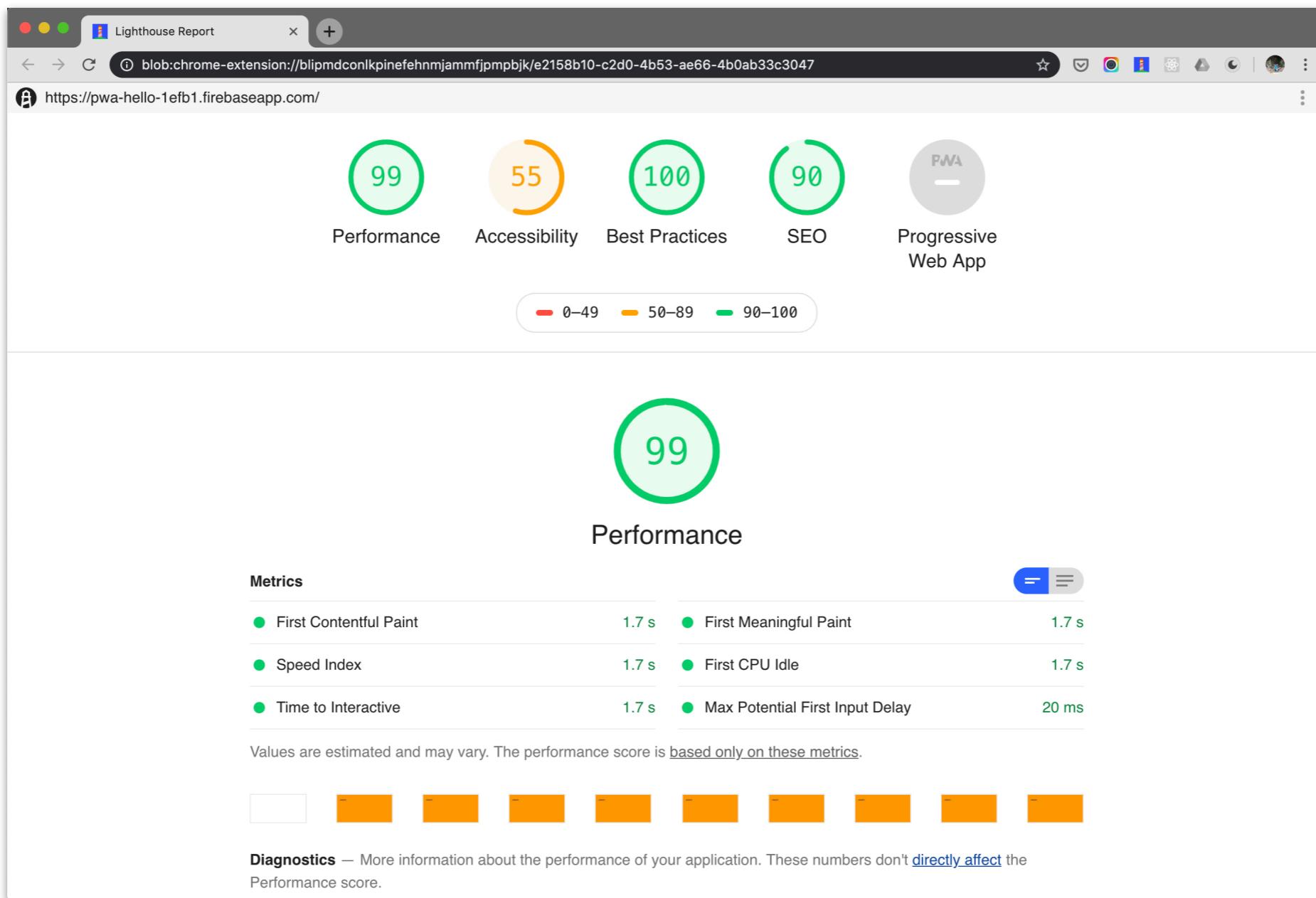
```
$ firebase deploy
```



Lighthouse 실행



- 크롬 Lighthouse 확장 설치 / 실행



대안 : 로컬호스트에 배포

- http-server 사용

```
npm install -g http-server
```

- index.html 이 있는 디렉토리에서

```
http-server ./
```

```
Starting up http-server, serving ./
Available on:
  http://192.168.219.103:8080
  http://127.0.0.1:8080
  Hit CTRL-C to stop the server
```

- http://localhost:8080 으로 확인

Service Worker 등록

- index.html 에 포함된 js/app.js 에 추가

```
if ('serviceWorker' in navigator) {  
  navigator.serviceWorker.register('js/sw.js')  
    .then(() => console.log('Service Worker Registered ..'));  
}
```

Static 리소스 캐싱

- hello-world-page 라는 이름으로 다음 파일 캐싱 (sw.js)

- /
- /index.html
- /css/main.css

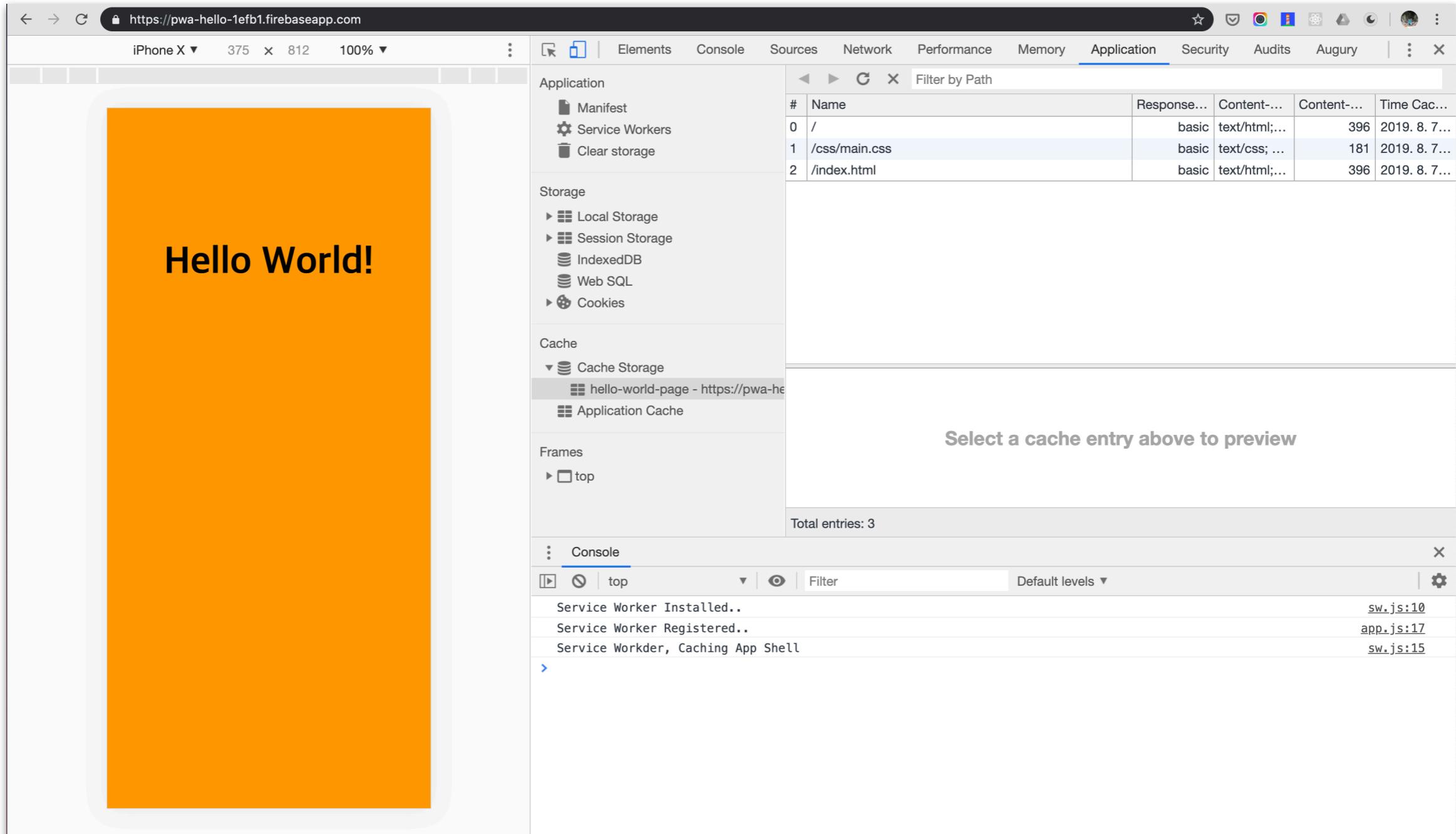
```
self.addEventListener('install', evt => {
    console.log('Service Worker Installed..');

    evt.waitUntil(
        caches.open(cacheName)
            .then(caches => {
                console.log('Service Workder, Caching App Shell');
                return caches.addAll(filesToCache);
            })
    );
});

self.addEventListener('activate', evt => {
    evt.waitUntil(self.clients.claim());
});

self.addEventListener('fetch', event => {
    event.respondWith(
        caches.match(event.request, {ignoreSearch:true}).then(response => {
            return response || fetch(event.request);
        })
    );
});
```

리소스 캐싱 테스트



manifest.json 작성

- Add to homescreen 기능 지원
- index.html에 연결

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

```
{  
  "name": "Hello World PWA",  
  "short_name": "Hi",  
  "icons": [  
    {  
      "src": "icons/icon-128x128.png",  
      "sizes": "128x128",  
      "type": "image/png"  
    }, {  
      "src": "icons/icon-144x144.png",  
      "sizes": "144x144",  
      "type": "image/png"  
    }, {  
      "src": "icons/icon-152x152.png",  
      "sizes": "152x152",  
      "type": "image/png"  
    }, {  
      "src": "icons/icon-192x192.png",  
      "sizes": "192x192",  
      "type": "image/png"  
    }, {  
      "src": "icons/icon-512x512.png",  
      "sizes": "512x512",  
      "type": "image/png"  
    }  
  ],  
  "start_url": "/index.html",  
  "gcm_sender_id": "103953800507",  
  "display": "standalone",  
  "background_color": "#FF9800",  
  "theme_color": "#FF9800"  
}
```

마지막 - Lighthouse 실행

- 완벽하지는 않지만 PWA에 꽤 가까워짐.

Results for: <https://hello-world-pwa-8669c.firebaseio.com/>
Oct 27, 2017, 1:26 PM PDT • ▶ Runtime settings



Category	Score
Progressive Web App	100
Performance	99
Accessibility	100
Best Practices	94

Progressive Web App 
These checks validate the aspects of a Progressive Web App, as specified by the baseline [PWA Checklist](#).

1 Failed Audits

- ▶ Page load is fast enough on 3G 
First Interactive was found at 1,480 ms, however, the network request latencies were not sufficiently realistic, so the performance measurements cannot be trusted.
- ▶ 10 Passed Audits
- ▶ Manual checks to verify

Tutorial - 2

News PWA

News API 서비스

- <https://newsapi.org/>

The screenshot shows the homepage of the News API. At the top, there is a navigation bar with links for "Get started", "Documentation", "News sources", "Pricing", and an email address "soongon@gmail.com". Below the navigation bar, the main heading is "Search worldwide news with code". A subtext below it reads: "Get breaking news headlines, and search for articles from over 30,000 news sources and blogs with our news API". A blue button labeled "Get API key" is located in the center. On the left, there is a card titled "All articles about Bitcoin from the last month, sorted by recent first". It shows a GET request URL: `https://newsapi.org/v2/everything?q=bitcoin&from=2019-07-09&sortBy=publishedAt&apiKey=5dbbf0c9a84f409c84538550b23edb55`. Below the URL, a JSON snippet is displayed, showing the response structure with fields like "status", "totalResults", and "articles". On the right, there is another card titled "Top business h". It shows a GET request URL: `https://newsapi.org/v2/top-headlines?country=US&category=business&apiKey=5dbbf0c9a84f409c84538550b23edb55`. Below the URL, a JSON snippet is displayed, showing the response structure with fields like "status", "totalResults", and "articles".

PWA 적용 전

The screenshot shows a browser window with three tabs: "PWA Tutorial", "News", and "News API - A JSON API for live". The main content area displays news articles from USA Today. The developer tools panel is open, specifically the Application tab. In the Application tab, the Manifest section is selected, showing a message "No manifest detected". Below this, there is a description: "A web manifest allows you to control how your app behaves when launched and displayed to the user." and a link "Read more about the web manifest". The other sections in the Application tab include Storage (Local Storage, Session Storage, IndexedDB, Web SQL, Cookies), Cache (Cache Storage, Application Cache), and Frames (top). The Console tab at the bottom shows a log entry: "status: 'ok', sources: Array(134)" from "app.js:19".

manifest.json 파일 적용

Web App Manifest Generator

The Web App Manifest is a JSON document that provides application metadata for Progressive Web Apps. Use the form below to generate the JSON file and optionally upload an app icon.

App Name Placeholder	Short Name Placeholder
Theme Color <code>#2196f3</code>	Background Color <code>#2196f3</code>
Display Mode Browser	Orientation Any
Application Scope <code>/</code>	Start URL <code>/</code>

manifest.json

```
{  
  "theme_color": "#2196f3",  
  "background_color": "#2196f3",  
  "display": "browser",  
  "scope": "/",  
  "start_url": "/"  
}
```

Generate Icons

The Web App Manifest allows for specifying icons of varying sizes. Upload a 512x512 image for the icon and we'll generate the remaining sizes.

ICON news-icon.png

GENERATE .ZIP

Hosted on [Firebase](#).

감사합니다.

soongon@hucloud.co.kr