webOS

Web App 개발

1

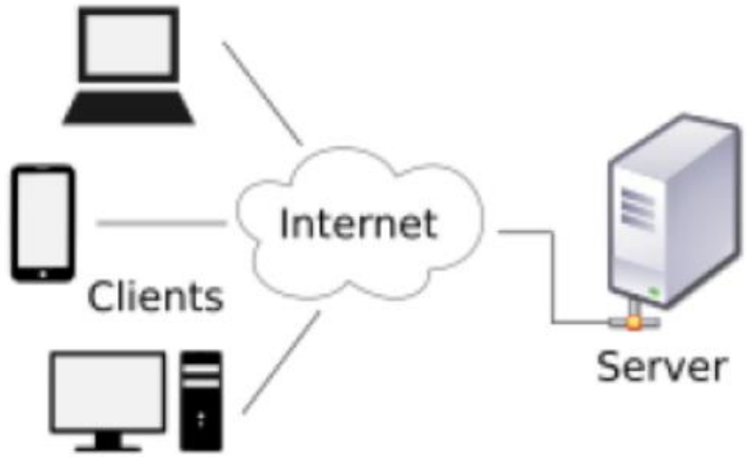
webOS OSE Web App 개발하기

webOS 웹 앱 개발하기

webOS

Web App

클라이언트 서버 모델



- **클라이언트**란 서비스를 사용하는 사용자 혹은 사용자의 단말기를 나타내는 말
- **서버**란 서비스를 제공하는 컴퓨터이며 다수의 클라이언트를 위해 존재하기 때문에 일반적으로 매우 큰 용량과 성능을 가지고 있음
- **클라이언트-서버 모델**은 서비스 요청자인 클라이언트와 서비스 자원의 제공자인 서버 간에 작업을 분리해주는 분산 앱 구조이자 네트워크 아키텍처를 나타냄

webOS

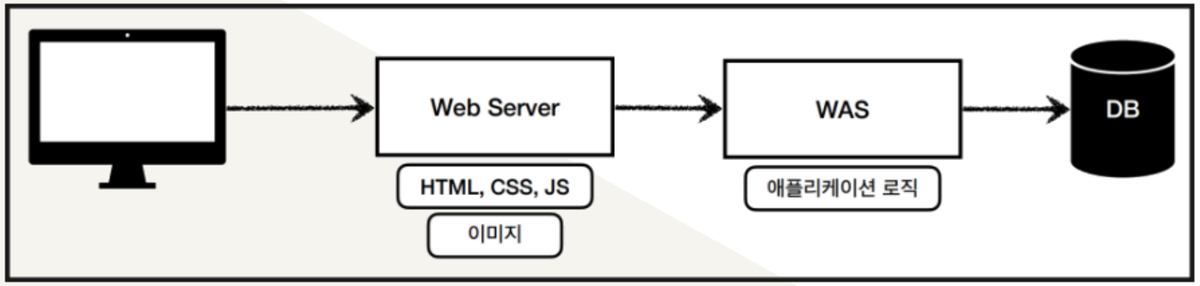
Web App

정의

- 인터넷이나 인트라 넷을 통해 웹 브라우저 위에서 이용할 수 있는 응용 소프트웨어
- 웹 표준에 따라 개발 (호환성 높음)

구성요소

- 웹 브라우저: 사용자가 요청한 웹서버의 결과를 보여주기 위한 애플리케이션
  - 크롬, 파이어폭스, IE 등
- 웹 서버: 웹 브라우저의 요청에 대한 결과를 응답해주는 기능, 정적인 페이지처리
  - 아파치, Nginx, IIS 등
- 애플리케이션 서버: 동적인 페이지 처리
  - 톰캣, 웹로직, 웹스파이어등
- 데이터베이스: 데이터를 저장하는 저장소



webOS

Web App

Web App vs. Native App

차이점 정리

	웹 앱	네이티브 앱
특징	웹사이트를 앱으로 감싸는 형태로, 안드로이드와 iOS 를 한 번에 개발	안드로이드와 iOS, 웹에 사용되는 앱을 각각 만들어야 함
장점	• 웹페이지를 수정하는 것만으로도 업데이트가 즉각 반영됨 • 저렴한 가격과 짧은 소요시간으로도 개발 가능 • 웹 메일, 온라인 전자상거래 및 경매, 인터넷 게시판, 블로그, 게임 등 다양한 기능을 구현할 수 있음	• 빠른 속도와 높은 사양의 그래픽과 성능 지원 • 앱이 설치된 기기의 카메라, GPS 등의 기능을 활용하여 개발 가능 • 네이티브 API를 호출하여 사용함으로 플랫폼과 밀착되어 있음
단점	• 인터넷, 설치된 기기의 상태에 따라 속도에 영향이 있음 • 온라인 상태를 유지해야 하고 서비스 제공에 한계가 있음 • 속도가 느림	• 기능 구현을 위해 많은 양의 코딩 작업이 필요 • 개발자의 높은 기술력을 요구 • 업데이트마다 플랫폼별로 작업해야 함

2

앱 메타데이터

앱의 각종 정보

webOS

앱
메타데이터

appinfo.json

- webOS 애플리케이션의 메타데이터 파일, 앱 root 디렉토리에 위치
- 애플리케이션이 패키지되기 위해서 반드시 필요
- JSON(JavaScript Object Notation) 형태로 저장
  - 주석 (/* 또는 //)이 포함될 수 없음
  - 속성을 나타낼 때는 ‘ ‘ (Single Quotes)가 아니라 “ “ (Double Quotes)를 사용
  - JSON 객체에 대한 표준은 JSON 공식 웹사이트를 참고

webOS

앱 메타데이터

appinfo.json

- 스키마(Schema)

```
{
    "id":string,
    "title":string,
    "main":string,
    "icon":string,
    "type":string,
    "vendor":string,
    "version":string,
    "appDescription":string,
    "resolution":string,
    "iconColor":string,
    "splashBackground":string,
    "transparent":boolean,
    "requiredMemory":number,
    "requiredPermissions":string array
}
```

webOS

앱 메타데이터

appinfo.json

- 입력 예)

appinfo.json (web app, with optional properties)

```
{
    "id": "com.myco.app.appname",
    "title": "AppName",
    "main": "index.html",
    "icon": "AppName_80x80.png",
    "type": "web",
    "largeIcon": "AppName_130x130.png",
    "vendor": "My Company",
    "version": "1.0.0",
    "appDescription": "This is an app tagline",
    "resolution": "1920x1080",
    "iconColor": "red",
    "splashBackground": "AppName_Splash.png",
    "transparent": false,
    "requiredMemory": 20,
    "requiredPermissions": ["time", "media"]
}
```

webOS

앱
메타데이터

appinfo.json

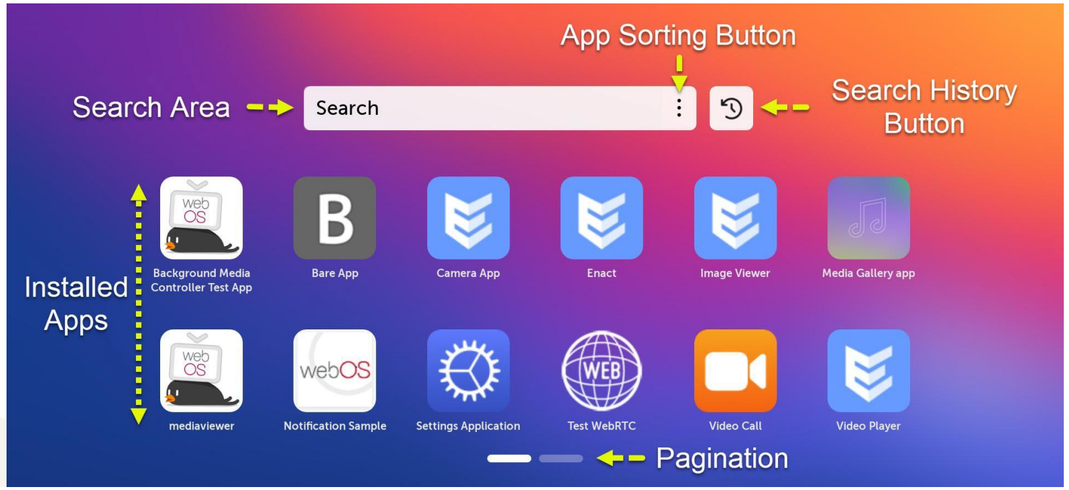
- Id 속성 (필수)
  - 앱마다 고유한 값을 가져야 함
  - 도메인 이름의 역순으로 생성하는 것을 권장 (도메인은 네트워크 상에서 고유함)
  - 앱의 ID는 한번 설정하면 앱을 게시한 이후에는 변경 할 수 없음
  - 주의사항
    - com.palm, com.webos, com.lge, com.palmdts 는 플랫폼에서 예약되어 있기 때문에 사용할 수 없음
    - ID는 소문자(a-z), 숫자(0-9), `-`, `.` 으로만 구성되어야 하고, 길이는 2자 이상이어야 하며, 영숫자로 시작해야 함
- Title 속성 (필수)
  - 홈 런처 및 앱 창에 표시되는 앱의 이름

webOS

앱 메타데이터

appinfo.json

- Icon 속성 (필수)
  - 앱 바와 런치패드에 표시되는 아이콘 이미지
  - PNG 형식으로 파일 경로는 프로젝트 루트 디렉터리로부터 시작되는 상대적 경로



webOS

앱 메타데이터

appinfo.json

- Main 속성 (필수)
  - 앱 실행 시 시작 지점
  - 파일 경로는 프로젝트 루트 디렉터리로부터 시작되는 상대적 경로
  - 입력 예)
    - 웹 앱: “index.html”
    - QML 앱: “main.qml“
    - 네이티브 앱: “main” (실행 바이너리 파일 이름)

webOS

앱
메타데이터

appinfo.json

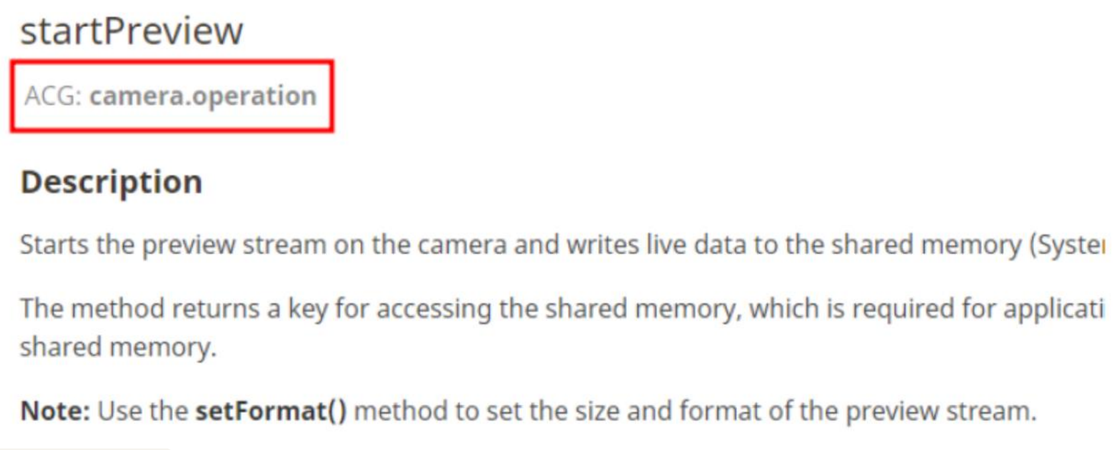
- Type 속성 (필수)
  - 앱의 종류
  - 입력 예)
    - 웹 앱: “web”
    - QML 앱: “qml“
    - 네이티브 앱: “native”

webOS

앱 메타데이터

appinfo.json

- RequiredPermissions 속성 (선택)
  - LS2 API를 사용한다면 반드시 명시해야 하는 속성
  - API의 메서드와 연결된 필수 ACG (Access Control Group) 이름을 지정
  - API Reference의 메서드마다 명시되어 있음



webOS

앱
메타데이터

appinfo.json

- RequiredPermissions 속성 (선택)
  - `ls-monitor` 명령에 `-i` 옵션을 추가하여 확인할 수 있음

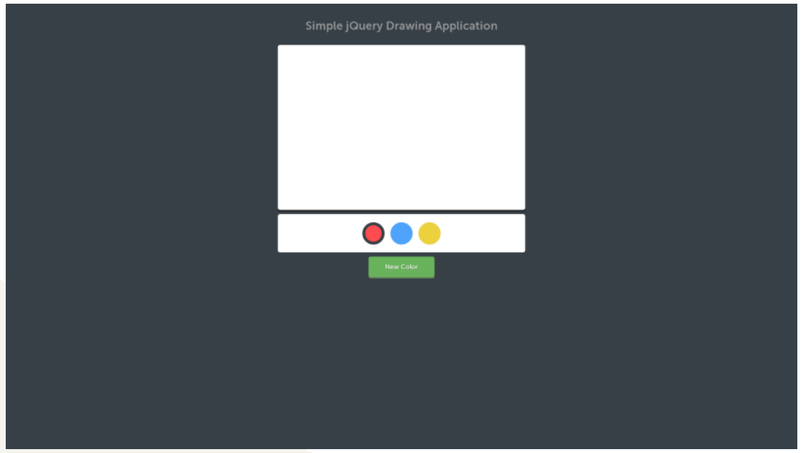
```
root@raspberrypi4-64:/var/rootdirs/home/root#ls-monitor-
icom.webos.service.camera2
METHODSANDSIGNALSREGISTEREDBYSERVICE'com.webos.service.camera2'WITHUNIQU
ENAME'TjsHWZrD'ATHUB
"/":
"getFd":{"provides":["all","camera.operation"]}
"startPreview":{"provides":["all","camera.operation"]}
"setFormat":{"provides":["all","camera.operation"]}
"close":{"provides":["all","camera.operation"]}
"stopPreview":{"provides":["all","camera.operation"]}
"getEventNotification":{"provides":["all","camera.operation"]}
"getInfo":{"provides":["all","camera.query"]}
"setProperties":{"provides":["all","camera.operation"]}
"getProperties":{"provides":["all","camera.operation"]}
"getCameraList":{"provides":["all","camera.query"]}
"stopCapture":{"provides":["all","camera.operation"]}
"open":{"provides":["all","camera.operation"]}
"startCapture":{"provides":["all","camera.operation"]}
```

webOS

jQuery Web App (실습)

실습 – Jquery 기반 웹 앱 개발 (10분)

- 시나리오
  - jQuery sample을 Github에서 다운로드
    https://github.com/mcnitt/simple-jquery-drawing-app
  - appinfo.json 와 icon 파일을 CLI로 생성
  - 패키지 – 인스톨 – 실행하여 결과 확인(실행 신뢰성 확인) > 다른 앱으로 바꿔서 실습하는 방식
- 결과

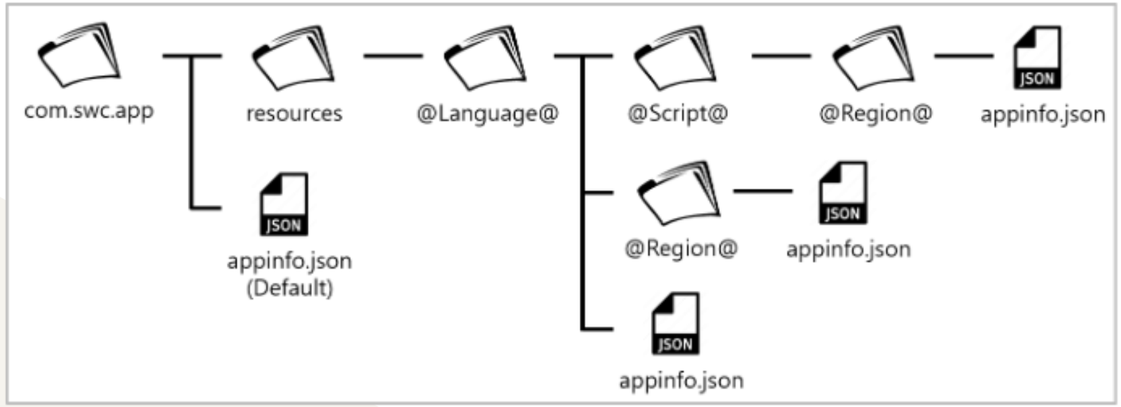


webOS

앱
메타데이터

메타데이터 현지화(Meta Localization)

- 하나의 앱이 둘 이상의 언어로 현지화 된 경우 각 언어에 대한 고유한 메타데이터를 가질 수 있음
- 현지화된 앱의 ID 및 버전 번호는 동일해야 함
- 지역(locale) 설정 값에 따라 홈 런처에서 일치하는 정보를 보여 줌
- 프로젝트 루트 디렉터리에 `resources` 디렉터리를 생성하고, 그 하위로 지역별 디렉터리를 생성한 후, appinfo.json 파일을 지역에 맞도록 생성해주어야 함



webOS

앱 메타데이터

메타데이터 현지화(Meta Localization)

- 각 지역별 appinfo.json 파일에는 `title` 속성과 `appDescription` 속성을 추가할 수 있음

```
{
    "title":"<Translatedapptitle>",
    "appDescription":"<Translatedappdescription>",
}
```

`locale`은 언어, 문자 그리고 국가/지역 정보도 구성됩니다. (예: ko-KR, mn-Cy-MN)
- 언어코드 : https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes
- 문자코드 : https://en.wikipedia.org/wiki/ISO_15924
- 지역코드 : https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes

webOS

메타데이터 현지화 (실습)

실습 – 메타데이터 현지화

- 시나리오
  - CLI의 Generate 명령을 사용해서 webapp 템플릿으로 프로젝트 생성
    - 앱 타이틀 : “English App Title” 로 입력
  - 프로젝트 루트에 `resources > ko > kr ` 리렉터리 생성
    - Kr 디렉터리 하위에 `appinfo.json` 파일 생성

```
{
    "title":"한글 앱 제목",
}
```

  - 패키지 – 인스톨 – 실행하여 결과 확인
- 결과
  - 시스템 설정을 한국어로 변경하여 앱 타이틀 확인

2

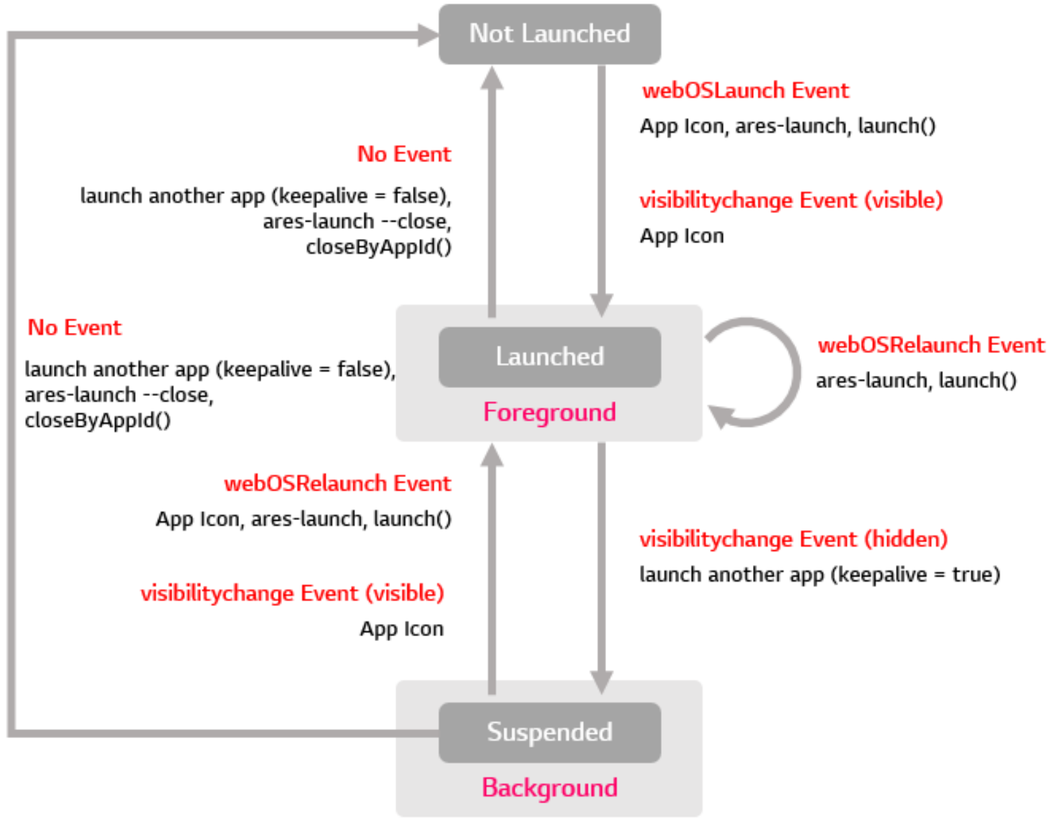
앱 라이프 사이클

앱의 생명주기

webOS

앱
라이프사이클

webOS 앱의 라이프 사이클



webOS

앱
라이프사이클

webOS 앱의 라이프 사이클

- Not launched
  - 웹 앱이 아직 시작되지 않았거나 이미 종료된 상태
  - 시작
    - 앱 아이콘 클릭
    - Ares-launch 명령어를 사용해 시작
    - 앱 관리자 서비스의 launch 메소드를 호출하여 시작
  - 종료
    - `keepAlive` 속성이 `false`로 설정되어 있을때 다른 앱을 시작
    - Ares-launch 명령어의 --close 옵션으로 종료
    - 앱 관리자 서비스의 closeByAppId 메서드를 호출하여 종료

webOS

앱
라이프사이클

webOS 앱의 라이프 사이클

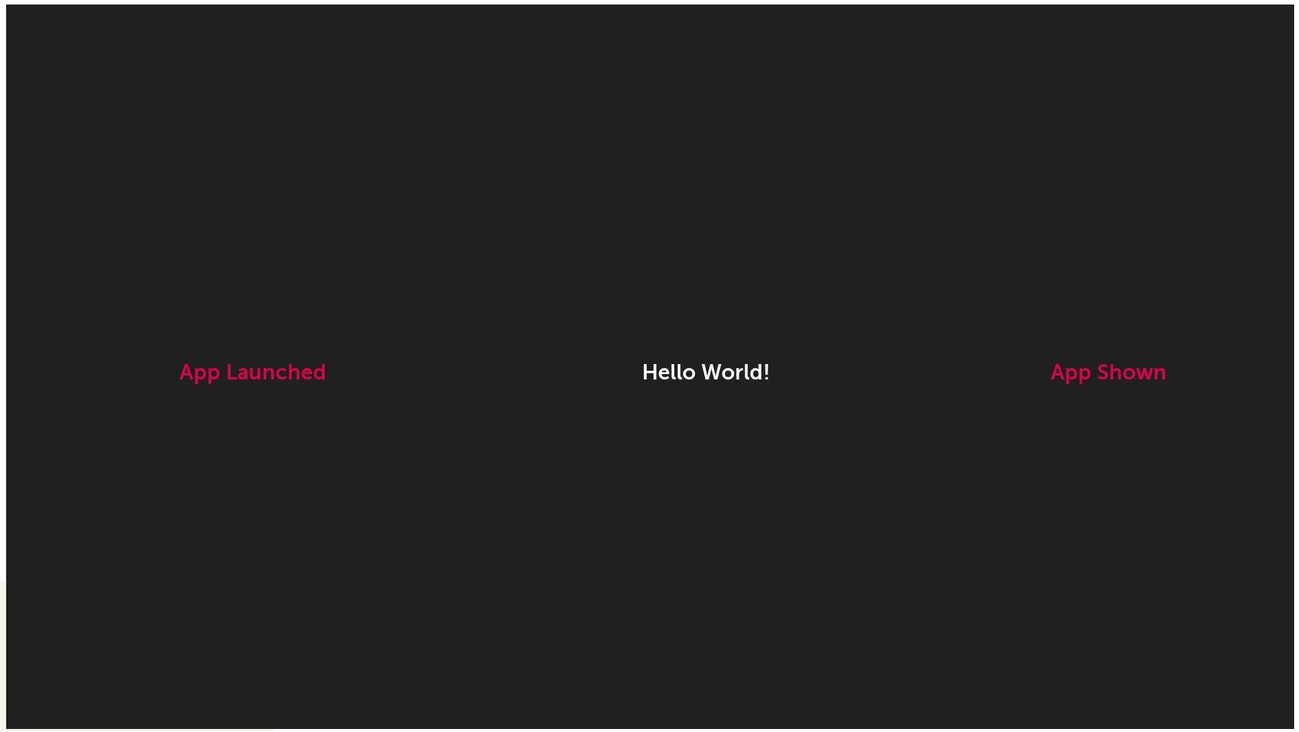
- Launched
  - 웹 앱이 Foreground에서 실행중이고 이벤트(webOSRelaunch, visibilitychange)를 수신 대기 중인 상태
- Suspended
  - 웹 앱이 background에서 일시 중지된 상태이고 이벤트(webOSRelaunch, visibilitychange)를 수신 대기 중인 상태
  - 재시작
    - 일시 중지 중인 앱 아이콘 클릭
    - 일시 중지 중인 앱을 Ares-launch 명령어를 사용해 시작
    - 일시 중지 중인 앱을 앱 관리자 서비스의 launch 메소드를 호출하여 시작
  - 일시 중지
    - `keepAlive` 속성이 `true`로 설정되어 있을때 다른 앱을 시작

webOS

앱
라이프사이클

webOS 앱의 라이프 사이클

- 샘플 앱 코드 설명 및 실행



2

Luna Service API

webOS 플랫폼이 제공하는 API

webOS

Luna Service

LS (Luna Service) 2 API

- webOS 플랫폼이 제공하는 JSON 기반의 API

LS2 API 구성

- Service URI
  - 서비스의 URI (예: luna://com.webos.service.devmode 형태)
- 메서드(Method)
  - 각 서비스는 기능별로 분류된 하나 이상의 메서드를 제공
  - 서비스에 공통적인 메서드는 루트 범주에 그룹화되고 특정 기능과 관련된 메서드는 관련 범주에 그룹화
  - 예)
    - getPreferences / setPreferences
    - time/getSystemTime / time/setSystemTime

webOS

Luna Service

LS2 API 구성

- 매개변수 (Parameter)
  - 매개변수는 서비스 요청 옵션들로써 서비스 제공자에게 JSON 객체로 전달
  - `subscribe`와 같은 예약 속성을 가질 수 있음
  - 예)
    - time/getSystemTime
      - 별도의 입력 매개변수를 요구하지 않음
      - `subscribe` 속성을 추가해서 보낼 수 있음
    - time/setSystemTime
      - 입력 매개변수로 `utc` 값을 요구
      - `timestamp` 값을 추가해서 보낼 수 있음

webOS

Luna Service

LS2 API 구성

- 호출 응답 (Call Response)
  - 호출 응답은 서비스 클라이언트의 요청에 대한 서비스 제공자의 응답 데이터를 포함하는 JSON 객체
  - 호출 응답은 호출의 성공 또는 실패를 알려주기 위해 예약된 `returnValue` 속성을 포함함
  - returnValue 값
    - True
      - 호출은 성공했고, 요청에 대한 추가적인 속성들을 가짐
    - False
      - 호출에 실패했고, `errorCode`와 `errorText`와 같은 에러정보에 대한 추가적인 속성을 가짐

webOS

Luna Service

LS2 API 구성

- 구독 응답 (Subscription Response)
  - 구독 응답은 구독에 대한 서비스의 응답 데이터를 포함하는 JSON 객체
  - `returnValue`와 `subscribed`속성
    - `subscribed` 속성은 구독 상태를 나타냄
    - `returnValue` 속성은 작업의 상태를 나타냄

webOS

Luna Service

메서드 호출

- 대상 기기(Target Device:SSH)에서 호출하는 방법
- 네이티브 앱 또는 서비스에서 호출하는 방법
- 자바스크립트 서비스에서 호출하는 방법
- 웹 앱에서 호출하는 방법

네이티브 앱 또는 서비스에서 호출하는 방법은 강의에 미포함

자바스크립트 서비스에서 호출하는 방법은 JS 서비스 수업에서 설명

webOS

Luna Service

대상 기기(Target Device)에서 LS2 API 호출

- `luna-send` 라는 명령어를 통해서 LS2 API를 호출
- 서비스의 동작 여부 또는 테스트를 위해서 사용되며, Python 등에서 Shell Command로 명령 후 결과 값을 얻어 올 수 있음
- 예) systemservice로부터 Time 정보를 얻어오는 호출

```
#luna-send -n 1 -f
luna://com.webos.service.systemservice/time/getSystemTime'{}'
```

- **대상 기기 연결(Emulator)**

cmd

```
ssh -p 6622 -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null
root@localhost
```

webOS

Luna Service

| 대상 기기(Target Device)에서 LS2 API 호출

- 결과

```sh
{
        "returnValue":true,
        "timezone":"America/Los_Angeles",
        "localtime":{
                "month":8,
                "day":2,
                "hour":22,
                "year":2022,
                "minute":54,
                "second":20
        },
        "TZ":"PDT",
        "offset":-420,
        "systemTimeSource":"ntp",
        "utc":1659506060,
        "NITZValid":false,
        "isDST":true,
        "timeZoneFile":"/var/luna/preferences/localtime",
        "timestamp":{
                "source":"monotonic",
                "sec":107,
                "nsec":268520565
        }
}
```

webOS

Luna Service

Web App에서 LS2 API 호출

- webOS OSE 2.0 부터는 WebOSServiceBridge Object를 사용하도록 변경
  - onservicecallback 속성 : 서비스 제공자로부터 전달받은 응답을 처리하기 위한 콜백 함수를 지정
  - call 메서드 : 서비스 제공자에게 서비스 요청을 전달
  - cancel 메서드 : 서비스 요청을 취소

JS

```
function quickServiceTest(){
    var bridge = new WebOSServiceBridge();
    var url = 'luna://com.webos.service.applicationmanager/running';

    bridge.onservicecallback = callback;

    function callback(msg){
        var response = JSON.parse(msg);
        console.log(response.returnValue);
    }

    var params = '{}';
    bridge.call(url, params);
}
```

webOS

Luna Service

Web App에서 LS2 API 호출

- WebOSServiceBridge 객체 생성
  - WebOSServiceBridge 생성자로 WebOSServiceBridge 객체(bridge)를 생성

```JS
var bridge = new WebOSServiceBridge();
```

- 서비스 URI와 매개변수 정의
  - 서비스 제공자에게 요청할 메서드를 포함하는 서비스 URI와 매개변수를 정의

```JS
var uri ='luna://com.webos.service.db/putKind';
var params = JSON.stringify({
        "id":"test.db-api:1",
        "owner":"APP_ID“
});
```

webOS

Luna Service

Web App에서 LS2 API 호출

- 서비스 콜백함수 정의
  - 서비스 요청으로 서비스 제공자로부터 수신한 응답 데이터를 처리하기 위하여 콜백 함수를 정의

```JS
bridge.onservicecallback = callback;
function callback(msg){
        var response=JSON.parse(msg);
        console.log(response.returnValue);
}
```

- 서비스 호출
  - 서비스 요청을 위해 call 메서드로 서비스 URI와 매개변수를 서비스 제공자에게 전달

```JS
bridge.call(uri,params);
```

webOS

Luna Service

Web App에서 LS2 API 호출

- 서비스 호출 취소
  - 서비스 요청 후, cancel 메서드를 사용하여 서비스 호출을 취소할 수 있음
  - cancel 메서드는 서비스 호출의 취소가 필요한 경우에만 사용
    - 구독에 대한 취소 시에 사용 할 수 있음

JS

```
var bridge = new WebOSServiceBridge();
bridge.onservicecallback = function(msg){
        varresponse=JSON.parse(msg);
        console.log(response.returnValue);
};
bridge.call("luna://com.webos.service.applicationmanager/running",'{"subscri
be":true}');
bridge.cancel();
```

webOS

Access Control Group

Web App에서 LS2 API 호출

- ACG 적용을 위한 명시
  - 사용할 메서드의 ACG 이름을 appinfo.json 파일에 명시해주어야만 정상적으로 접근
  - 예) Application Manager의 launch 메서드와
    DB 서비스의 putKind 메서드 사용

```
{
    ...
    "requiredPermissions":[
        "application.launcher",
        "database.operation"
    ]
}
```

requiredPermissions 속성을 수정한 후에는 반드시 앱을 재설치 해야 앱이 정상적으로 LS2 API를 사용 할 수 있음

대상 기기에 이미 설치된 앱의 appinfo.json 파일에서 requiredPermissions 속성을 수정해도 LS2 API의 사용권한이 부여되지 않기 때문에 재설치가 필요

webOS

Access Control Group

Access Control Group (ACG) 동작 원리

- webOS는 서비스 클라이언트가 서비스 제공자용 인터페이스에 무단으로 접근하는 것을 방지하기 위한 보안 정책
- ACG 사용 단계
    1. 서비스 제공자가 구성 파일에 ACG 권한을 정의
    2. 서비스 클라이언트가 이전 단계에서 정의된 ACG 권한을 설정
    3. 서비스 클라이언트가 메서드를 호출

       서비스 클라이언트의 ACG와 Trust level이 동일하지 않다면 Luna Bus는 서비스 제공자의 메서드에 대한 접근 요청을 거부함

webOS

Access Control Group

Access Control Group (ACG) 동작 원리

- ACG 권한을 정의하기 위한 구성 파일 (Built-In Service의 경우)

구성 파일	배포된 디렉터리
Role File	/usr/share/luna-service2/roles.d/
Service File	/usr/share/luna-service2/services.d/
API Permission File	/usr/share/luna-service2/api-permissions.d/
Client Permission File	/usr/share/luna-service2/client-permissions.d/
Groups File	/usr/share/luna-service2/groups.d/

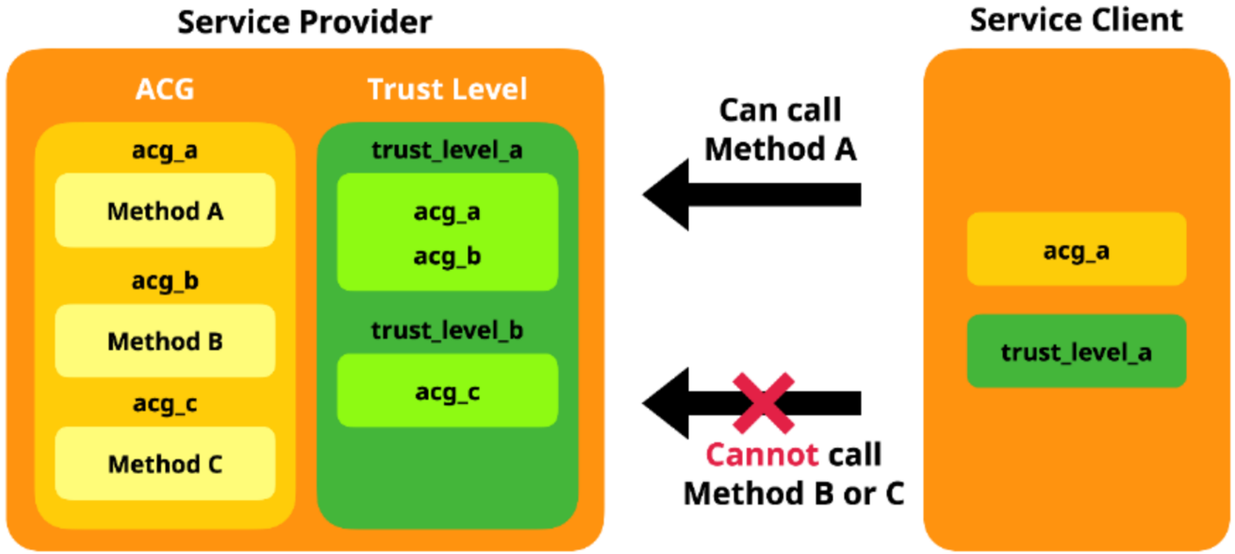
앱이나 외부 서비스는 앱을 설치하는 과정에서 구성 파일들을 appinfo.json 이나 services.json 파일로부터 자동으로 생성되지만, 빌트인 서비스는 개발자가 구성 파일을 수동으로 만들어주어야 함

webOS

Access Control Group

Access Control Group (ACG) 동작 원리

- ACG와 Trust Level



webOS

Access Control Group

Access Control Group (ACG) 동작 원리

- ACG와 Trust Level
  - webOS OSE에서는 trust level로 `oem`, `part`, `dev` 중 하나를 사용할 수 있음

Trust level of your service	Access to `dev` APIs	Access to `part` APIs	Access to `oem` APIs
dev	Allowed	Not Allowed	Not Allowed
part	Allowed	Allowed	Not Allowed
oem	Allowed	Allowed	Allowed

webOS

Access Control Group

Access Control Group (ACG) 동작 원리

- ACG와 Trust Level 설정

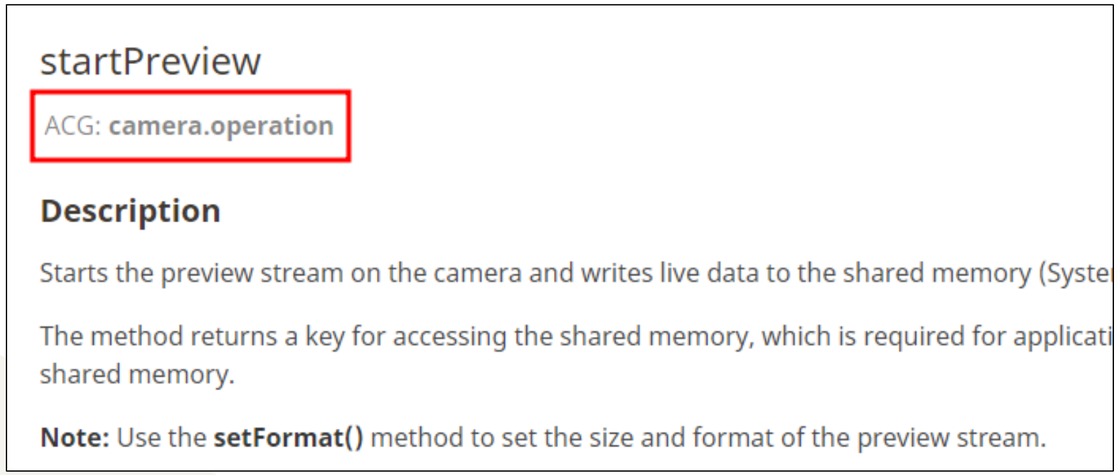
Type	For ACG	For Trust Level
App	Set up the `requiredPermissions` property in appinfo.json.	No need to set up. Trust level is set to `OEM` automatically.
External Service	Set up the `requiredPermissions` property in appinfo.json of the app packaged with the external service.	
Built-in Service	Set up Configuration files.	

webOS

Access Control Group

Access Control Group (ACG) (1/2)

- webOS의 보안 정책으로 서비스 메소드 별로 접근 권한을 부여하여 해당 권한을 appinfo.json 파일에 requiredPermissions 속성에 명시해 주어야 사용이 가능함
- webOS OSE 개발자 사이트의 API References 문서에 각 메소드 하위에 ACG 이름이 명시되어 있음



webOS

Access Control Group

Access Control Group (ACG) (1/2)

- ls-monitor 명령에 –i 옵션으로 확인하고자 하는 서비스의 모든 메서드별 ACG 이름을 확인할 수 있음

```
root@raspberrypi4-64:/var/rootdirs/home/root# ls-monitor -i com.webos.service.camera2

METHODS AND SIGNALS REGISTERED BY SERVICE 'com.webos.service.camera2' WITH UNIQUE NAME 'TjsHWZrD' AT HUB

  "/":
    "getFd": {"provides":["all","camera.operation"]}
    "startPreview": {"provides":["all","camera.operation"]}
    "setFormat": {"provides":["all","camera.operation"]}
    "close": {"provides":["all","camera.operation"]}
    "stopPreview": {"provides":["all","camera.operation"]}
    "getEventNotification": {"provides":["all","camera.operation"]}
    "getInfo": {"provides":["all","camera.query"]}
    "setProperties": {"provides":["all","camera.operation"]}
    "getProperties": {"provides":["all","camera.operation"]}
    "getCameraList": {"provides":["all","camera.query"]}
    "stopCapture": {"provides":["all","camera.operation"]}
    "open": {"provides":["all","camera.operation"]}
    "startCapture": {"provides":["all","camera.operation"]}
```

webOS

Access Control Group

Access Control Group (ACG) (1/2)

- ls-monitor 명령에 –i 옵션으로 확인하고자 하는 서비스의 모든 메서드별 ACG 이름을 확인할 수 있음

```
root@raspberrypi4-64:/var/rootdirs/home/root# ls-monitor -i com.webos.service.camera2

METHODS AND SIGNALS REGISTERED BY SERVICE 'com.webos.service.camera2' WITH UNIQUE NAME 'TjsHWZrD' AT HUB

  "/":
    "getFd": {"provides":["all","camera.operation"]}
    "startPreview": {"provides":["all","camera.operation"]}
    "setFormat": {"provides":["all","camera.operation"]}
    "close": {"provides":["all","camera.operation"]}
    "stopPreview": {"provides":["all","camera.operation"]}
    "getEventNotification": {"provides":["all","camera.operation"]}
    "getInfo": {"provides":["all","camera.query"]}
    "setProperties": {"provides":["all","camera.operation"]}
    "getProperties": {"provides":["all","camera.operation"]}
    "getCameraList": {"provides":["all","camera.query"]}
    "stopCapture": {"provides":["all","camera.operation"]}
    "open": {"provides":["all","camera.operation"]}
    "startCapture": {"provides":["all","camera.operation"]}
```

webOS

System Keboard

Access Control Group (ACG) (1/2)

- ls-monitor 명령에 –i 옵션으로 확인하고자 하는 서비스의 모든 메서드별 ACG 이름을 확인할 수 있음

```
root@raspberrypi4-64:/var/rootdirs/home/root# ls-monitor -i com.webos.service.camera2

METHODS AND SIGNALS REGISTERED BY SERVICE 'com.webos.service.camera2' WITH UNIQUE NAME 'TjsHWZrD' AT HUB

  "/":
     "getFd": {"provides":["all","camera.operation"]}
     "startPreview": {"provides":["all","camera.operation"]}
     "setFormat": {"provides":["all","camera.operation"]}
     "close": {"provides":["all","camera.operation"]}
     "stopPreview": {"provides":["all","camera.operation"]}
     "getEventNotification": {"provides":["all","camera.operation"]}
     "getInfo": {"provides":["all","camera.query"]}
     "setProperties": {"provides":["all","camera.operation"]}
     "getProperties": {"provides":["all","camera.operation"]}
     "getCameraList": {"provides":["all","camera.query"]}
     "stopCapture": {"provides":["all","camera.operation"]}
     "open": {"provides":["all","camera.operation"]}
     "startCapture": {"provides":["all","camera.operation"]}
```

webOS

System Keboard

System Keboard

- webOS OSE는 사용자 입력을 위하여 시스템 키보드를 제공
- 웹 앱에서 시스템 키보드는 HTML의 Input 요소와 연동하여 동작
  - text
  - number
  - email
  - url

```html
<form>
   <input type=”text”>
   <input type=”number”>
   <input type=”email”>
   <input type=”url”>
</form>
```

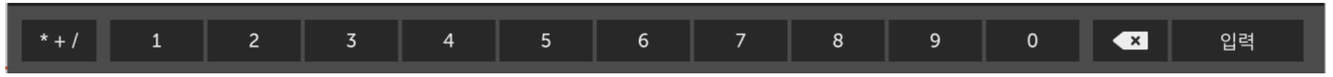
webOS

System Keboard

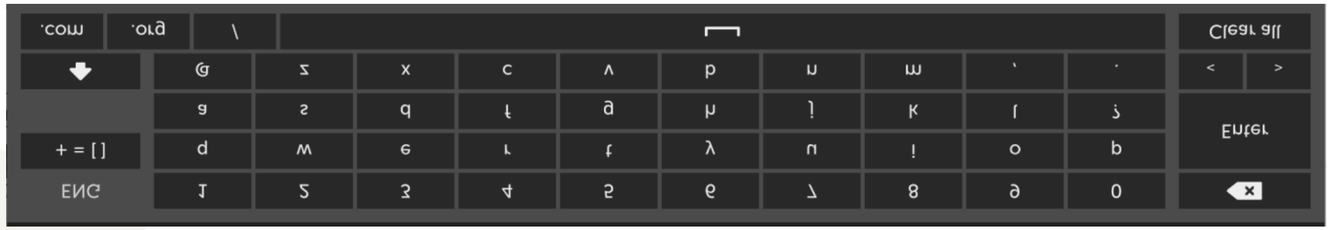
System Keboard: text

- Text

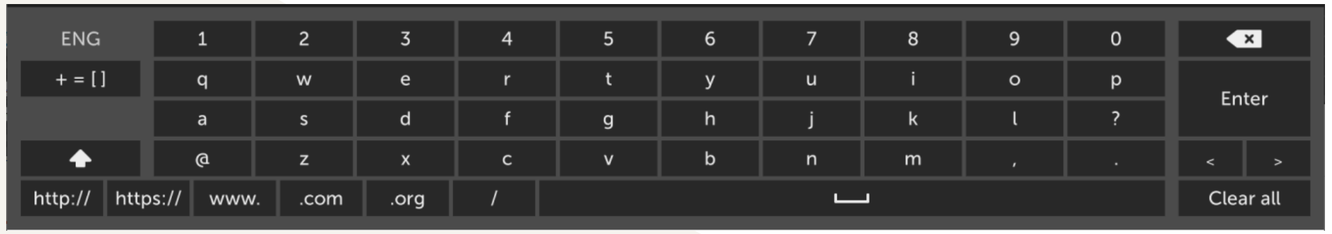
- Number



- Email



- url



webOS

LS2 API 호출 (실습)

실습 – LS2 API Call

- 문제
  - 아래 시나리오에 따라 앱을 개발하여 실행해보고 에러가 발생하는 부분에 대한 수정 (버튼을 눌러 유투브 앱이 실행되면 정상 동작)
- 시나리오
  - webOS Studio의 프로젝트 마법사로 basic web app 생성
    - 프로젝트 이름 : “basic_app” 입력
    - 앱 ID : ksw.emp.boot.lscall.<이니셜>
  - 프로젝트 루트에 index.html 파일 편집
    - API와 Parameter 변수 추가 (window.onload에 추가)

```JS
var LaunchApi = 'luna://com.webos.service.applicationmanager/launch';
var LaunchParams = '{"id":"com.webos.app.test.youtube"}';
```

webOS

LS2 API 호출 (실습)

실습 – LS2 API Call

- 시나리오
  - 프로젝트 루트에 index.html 파일 편집
    - Callback Function 추가 (window.onload에 추가)

JS

```
function launch_callback(msg){
        var arg = JSON.parse(msg);
        if (arg.returnValue) {
            console.log("[APP_NAME: example web app] CALLLAUNCH_SUCCESS
response : " + arg.Response);
        }
        else {
            console.error("[APP_NAME: example web app] CALLLAUNCH_FAILED
errorText : " + arg.errorText);
        }
    }
```

webOS

LS2 API 호출 (실습)

실습 – LS2 API Call

- 시나리오
  - 프로젝트 루트에 index.html 파일 편집
    - onclick 이벤트 핸들러 추가 (window.onload에 추가)

```JS
document.getElementById("runapp").addEventListener("click", function(){
        bridge.onservicecallback = launch_callback;
        bridge.call(LaunchApi, LaunchParams);
    });
```

    - Html 버튼 추가 (body에 추가)

```JS
<button id="runapp">Launch app</button>
```

webOS

LS2 API 호출 (실습)

실습 – LS2 API Call

- 시나리오
  - 프로젝트 루트에 index.html 파일 편집
    - Css 추가 (style 태그에 추가)

JS

```
button {
        background-color: #4CAF50;
        border: none;
        color: white;
        padding: 15px 32px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 4px 2px;
        cursor: pointer;
    }
```

webOS

LS2 API 호출 (실습)

실습 – LS2 API Call

- 시나리오
  - Run (패키징 – 설치 – 실행)
  - 디버깅 결과 확인
    - ares-inspect 사용
  - 에러 수정 후 다시 패키징 – 설치 – 실행

Question