전자정부 모바일 표준프레임워크

실행환경(Device API)

eGovFrame

Contents

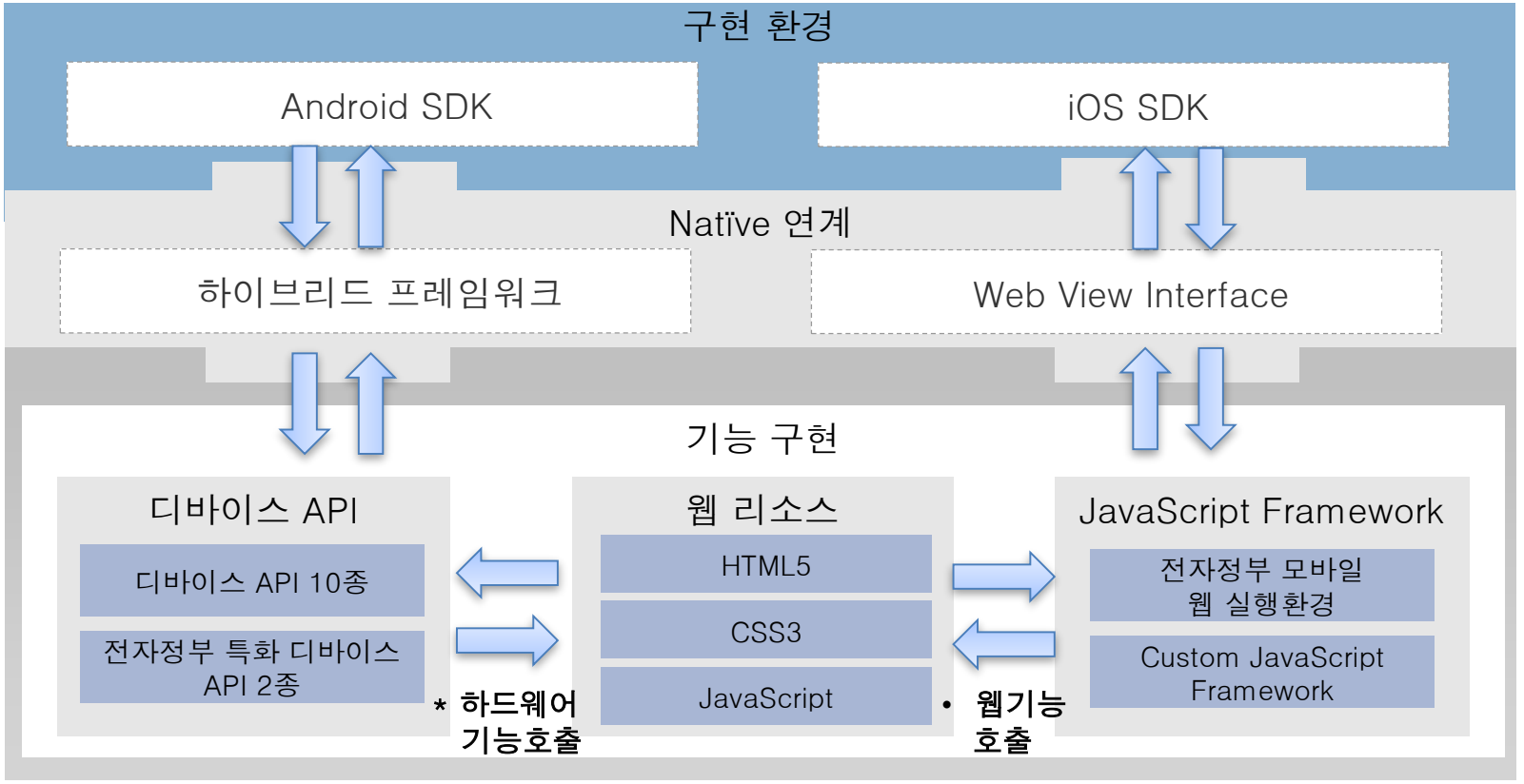
❑ **전자정부 디바이스 API 실행환경**은 기존 모바일 어플리케이션 개발 환경에서 Plug-in 형태로 구현되어 있는 디바이스 API를 추가하여 모바일 하이브리드 어플리케이션을 구현하는 방식이다.

- **모바일 하이브리드 어플리케이션 구현을 위한 하이브리드 프레임워크 기반 제공**
  - 디바이스 고유기능 호출을 위한 Native Code 및 JavaScript Code 제공
- **HTML 페이지에서 디바이스 고유기능의 호출을 위한 디바이스 API 제공**
  - 10종의 오픈 소스 디바이스 API 제공
  - Accelorator, Compass, Camera, Media, File Read/Write, Device, Vibrator, Network, GPS, Contact 등
- **플러그인 형태의 추가 API 제공**
  - NPKI 네이티브 코드와 연계를 위한 JavaScript 형태의 연계 템플릿 API 제공
  - 기존의 전자정부 표준프레임워크로 구현 된 웹 서버 어플리케이션과의 연계를 위한 연계 API 제공
- **하이브리드 어플리케이션 개발을 위한 오픈 소스 소프트웨어 선정**
  - 전자정부 모바일 하이브리드 어플리케이션은 PhoneGap 프레임워크를 기반으로 하며, HTML5 기반의 화면 구성을 위하여 전자정부 모바일웹 실행환경을 적용

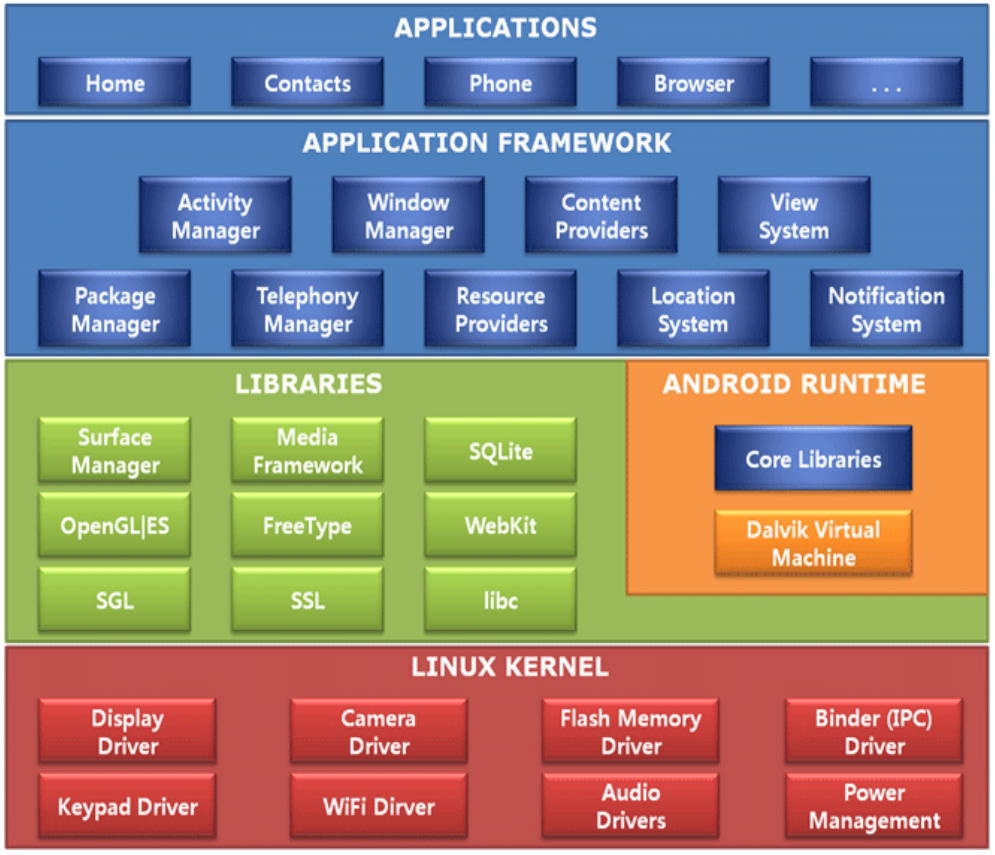
❑ 전자정부 디바이스 API 실행환경은 하이브리드 어플리케이션의 구성 기반이 되며, 하이브리드 어플리케이션 구현 및 실행 시 필요한 기본기능을 제공한다.

환경	설명
실행 환경	• 전자정부 모바일 하이브리드 어플리케이션 개발 프레임워크 라이브러리 제공 - Open Source(PhoneGap) - 표준 코드 및 하이브리드 앱 샘플 템플릿 제공 • Device API 지원 - 12종의 Open Source Device API - 추가 Device API 10종 제공
모바일 디바이스 API 가이드 프로그램	• 모바일 하이브리드 어플리케이션 개발 시 가이드 및 재사용을 위한 디바이스 API 가이드 프로그램 제공 - 총 48종의 디바이스 API 가이드 하이브리드 앱 개발 (안드로이드 24종, iOS 24종) - 디바이스 API 가이드 하이브리드 앱과 통신을 위한 전자정부 표준프레임워크 기반 웹 서버 어플리케이션 개발
개발 환경	• 플랫폼 별 하이브리드 앱 개발 템플릿 프로젝트 생성 도구 제공

❑ 디바이스 API 실행환경은 웹 리소스를 통한 디바이스 하이브리드 어플리케이션 구현을 지원하며 플랫폼 별 SDK를 활용하여 구현된 웹 리소스 내의 JavaScript 형태의 Device API와 각 플랫폼 별 Native Code가 하이브리드 프레임워크 및 웹 뷰 인터페이스를 통해 연동되어 실제 디바이스의 고유 기능을 호출할 수 있도록 지원 한다.

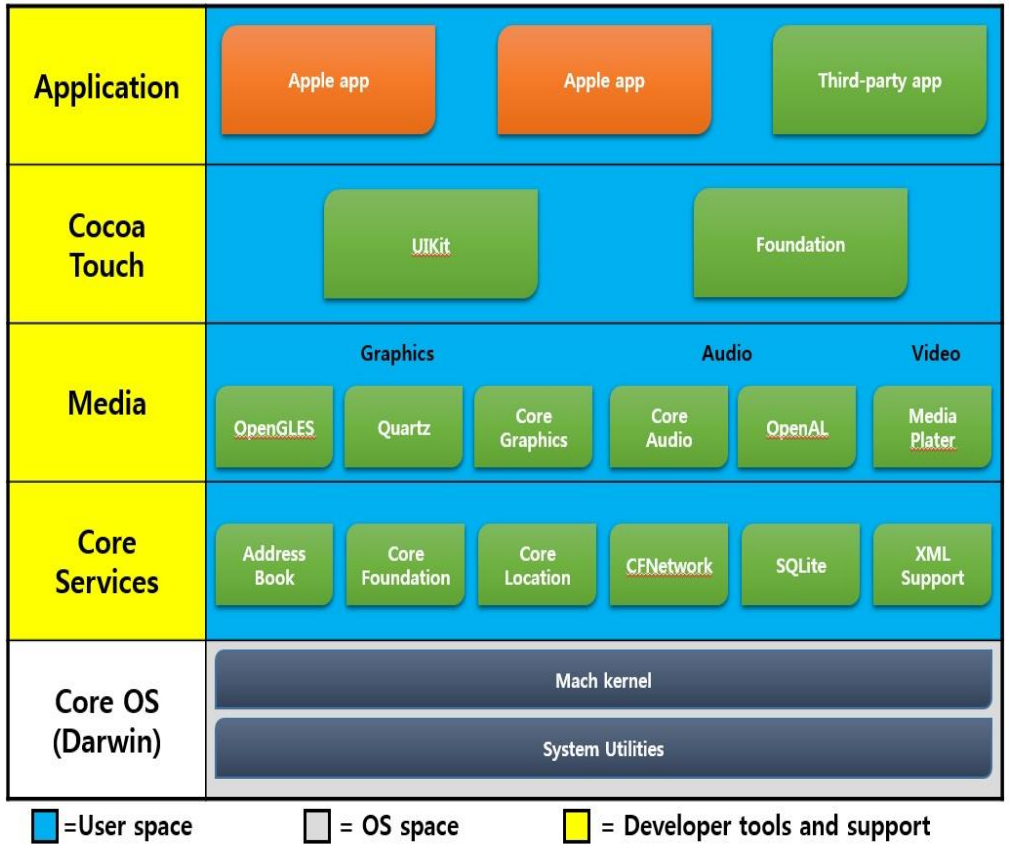


❑ Android 운영체제의 구성은 다음과 같다.



- Linux kernel
  - 하드웨어 드라이버, 프로세스와 메모리 관리, 보안, 네트워크, 전력 관리 등의 핵심 서비스 담당.
- Libraries
  - Android libc와 SSL 같은 다양한 C/C++ 코어 라이브러리
  - 핸드폰에 사용되는 하드웨어를 지원하기 위해 컴파일 되어 핸드폰 공급업체에 의해 핸드폰에 미리 설치 됨
- Android Runtime
  - Core Libraries, Dalvik Virtual Machine으로 구성.
  - 커널위에 존재하며 Dalvik, VM, 코어 라이브러리 등이 포함된다.
- Application Framework
  - Android Application을 만드는데 필요한 기능을 지원, App들을 관리하는 역할을 한다.
  - 직접만든 컴포넌트를 통해 확장 시킬 수 있다.
- Application
  - 안드로이드 아키텍처 다이어그램의 최상위 계층
  - 안드로이드의 특징 중 하나로 모든 어플리케이션이 동일한 수준으로 실행됨

❑ iOS 운영체제의 구성은 다음과 같다.

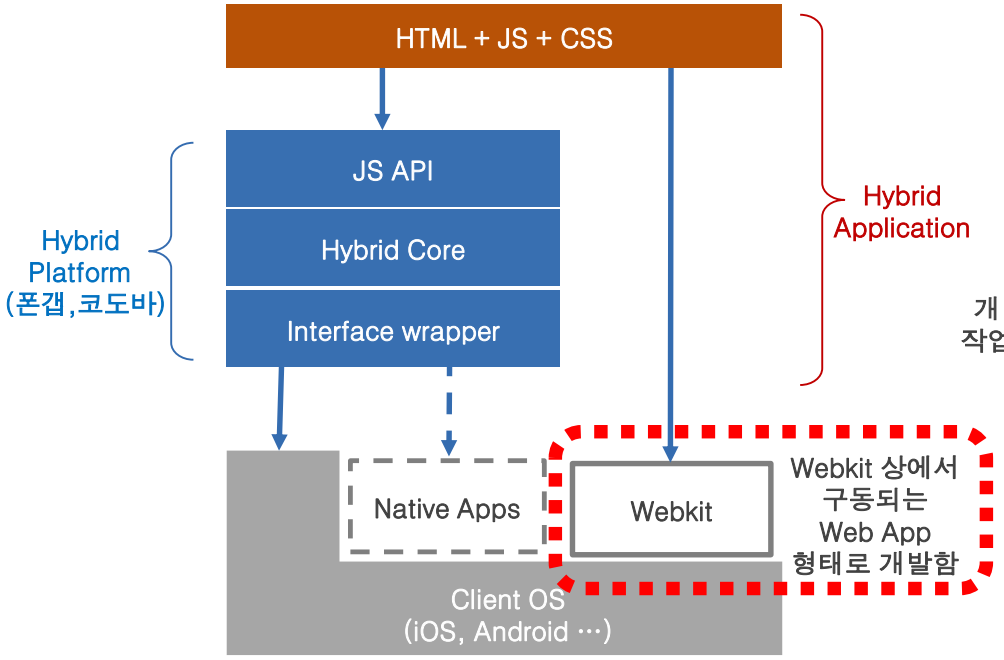


- iOS의 커널
  - Mac OS X와 같은 Mach에 기초한다.
- Core OS와 Core Services 계층
  - 매우 기본 적인 iOS의 인터페이스를 가지고 있다. 데이터 타입들, 봉주르 서비스, 네트워크 소켓 등이 있다.
- Media 계층
  - 2D/3D 그리고, 오디오, 비디오 등의 기반 기술을 가지고 있다. OpenGL ES, Quartz Core Audio와 Core Animation이 있다.
- Cocoa Touch 계층
  - 모든 기술이 Objective-C를 기본으로 하며, 각종 프레임워크로 응용프로그램을 만들 때 가장 기본적인 인프라를 제공한다.

기본 구조

❑ 모바일 디바이스 API 실행환경 기반 프레임워크는 하이브리드 어플리케이션을 구현하기 위한 기반 환경으로서 웹 리소스로 구현 된 기능을 네이티브 앱 형태로 래핑하는 역할을 한다.

Hybrid Application의 구조

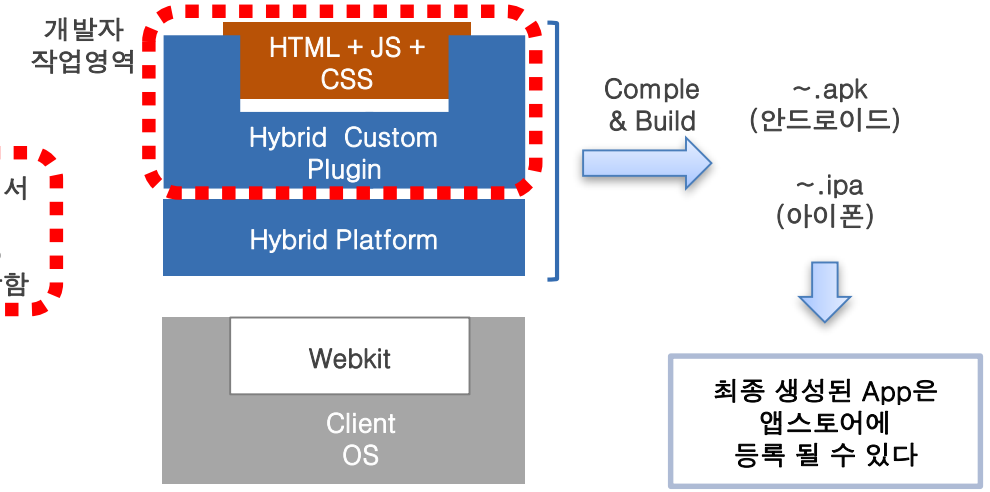


Application 은 Hybrid Platform에서 제공되는 JS를 통해 Native Feature를 이용한다.

Client OS 상의 Hybrid Application의 Package 형태

Application 개발자는 Web 개발 방식으로 HTML, JS, CSS 를 사용하여 개발한다.

Hybrid Platform으로 최종 Build 된 Hybrid Application은 Native Application과 같은 모습으로 Package 되고, 관리된다.

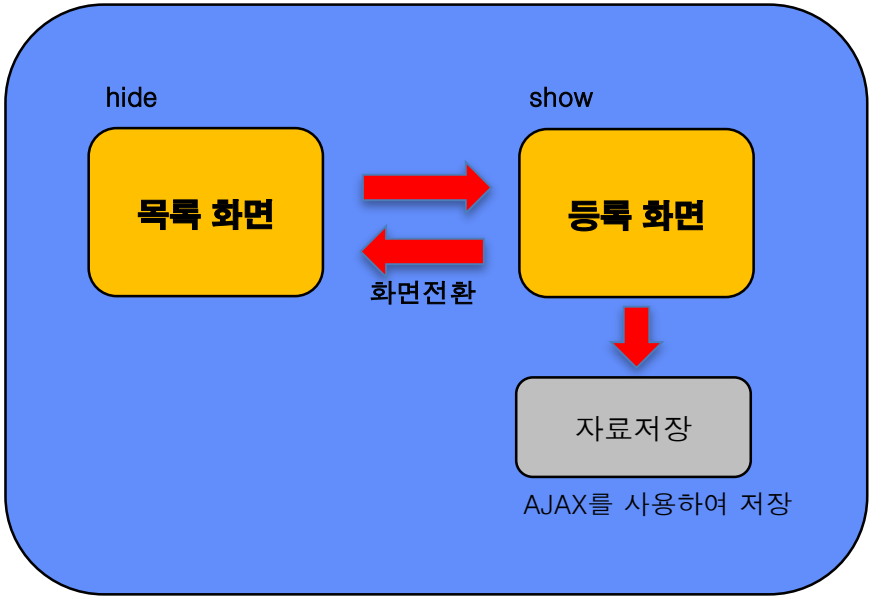


웹어플리케이션 종류

- Single Page Application은 업무단위로 여러 개의 화면을 포함 하고 한페이지내에서 화면단위로 이동하여 처리된다. (화면이동및 분업이 용이)
- Round Trip Application은 웹개발에서 고전적인 방식이며 화면 전환시 파라미터를 전달하고 페이지 이동을 한다.

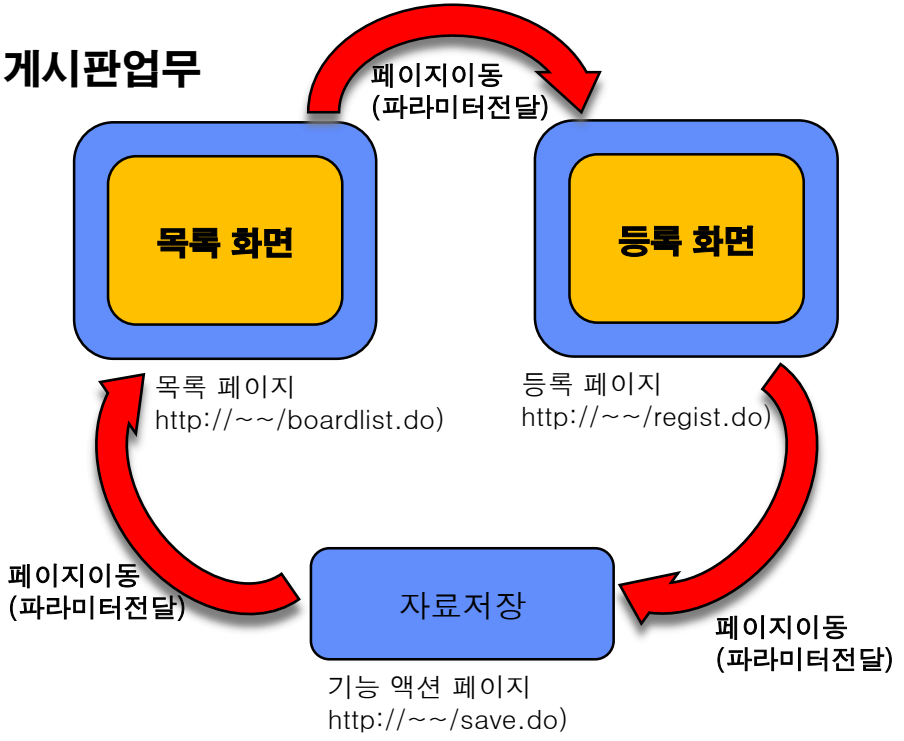
Single Page Application

게시판업무 (http://~~/board.html)

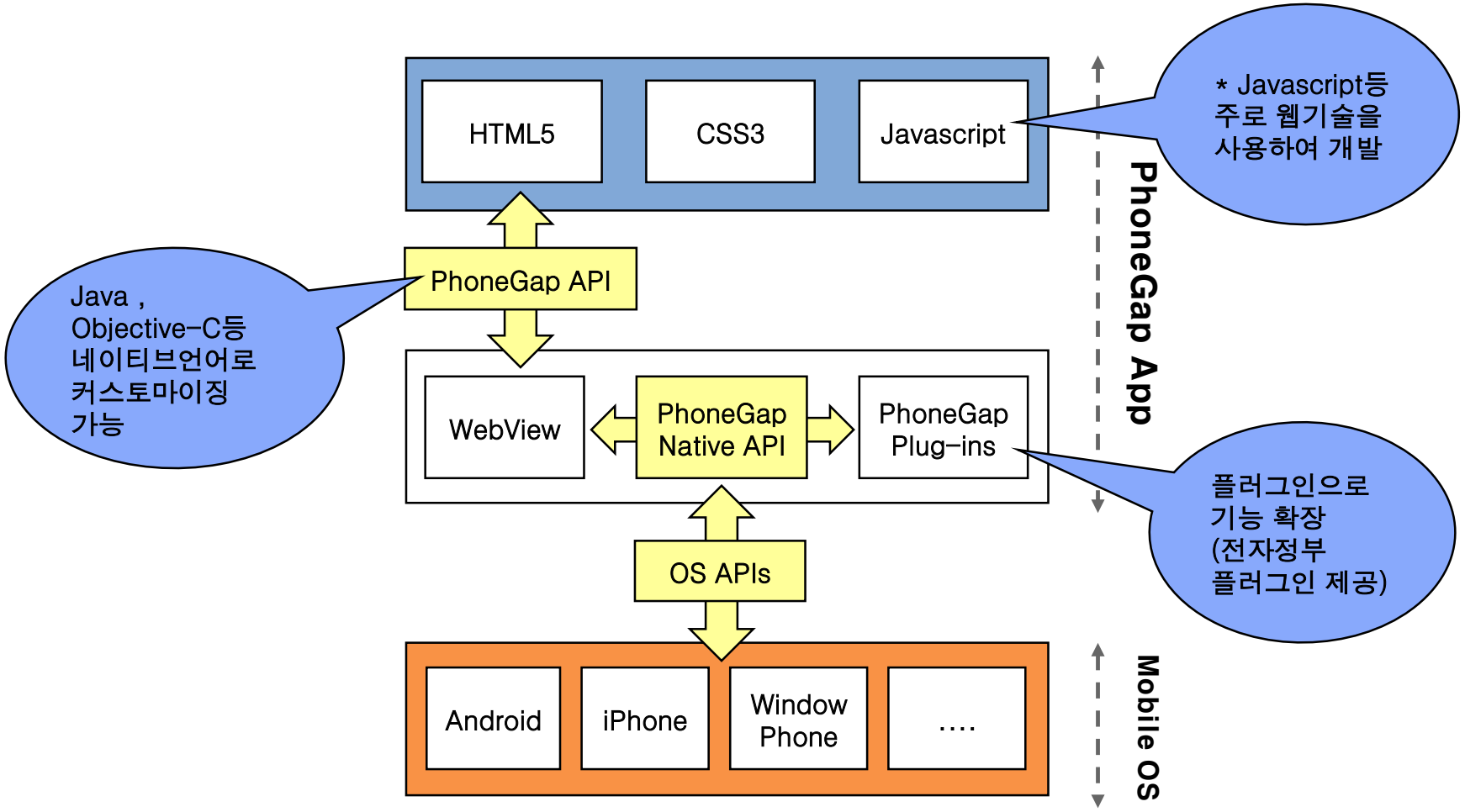


Round Trip Application

게시판업무



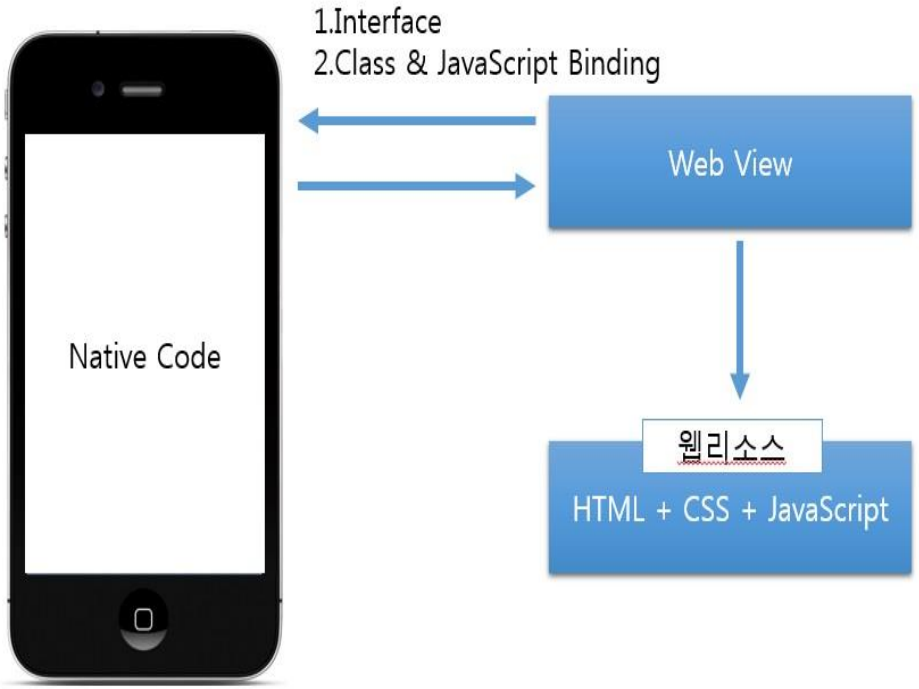
❑ 디바이스 API 실행환경의 주요 구성요소 인 하이브리드 어플리케이션 프레임워크의 구성은 다음과 같다.



❑ 폰갭은 Device API 제어를 위해 미리 구현 한 JavaScript 및 SDK 연계를 위한 Native Code 외에 추가기능을 위해 구현 된 Plug-In을 구성하고 있는 폰갭 Custom Plug-In 과 폰갭 Custom Native Plug-In이 추가된 구조이다.

구분	설명
Mobile Web Application	• HTML, CSS, 사용자 정의 JavaScript
PhoneGap JavaScript Engine	• Device API 제어를 위해JavaScript로 구현되어 제공 • PhoneGap.js
PhoneGap Native Engine	• 플랫폼 별 WebView를 상속받아 Mobile Web Application과 연계를 위한 Native Code • PhoneGap.jar • phoneGap.framework
PhoneGap Custom Plugin	• 기능 확장을 위해 추가 된 Plug-In을 위한 JavaScript 코드 • 사용자 정의 JavaScript Plug-In 을 포함 • 플랫폼 별로 플러그인을 위한 Native 코드 구성이 상이함에 따라 JavaScript 구성 또한 다를 수 있음
PhoneGap Custom Native Plugin	• 기능 확장을 위해 추가 된 Plug-In을 위한 플랫폼 Native 코드

- 폰갭은 플랫폼 별 SDK내에 내장 되어있는 Web View를 활용하여 하이브리드 앱 실행 시 사용자가 구현 한 HTML 파일을 출력하고 JavaScript 소스를 실행한다.
- 하이브리드 어플리케이션의 디바이스 API 호출을 위한 브릿지 역할을 해준다.



•SDK 로 부터 상속받은 웹 뷰를 생성해서 웹 뷰 내에 소스를 로드 한 후 네이티브API를 사용할 수 있도록 연계

• Interface Module (BrowserControl <-> Device API)

- iPhone

· shouldStartLoadWithRequest

· stringByEvaluationJavaScriptFromString

→ [gap://] protocol catch

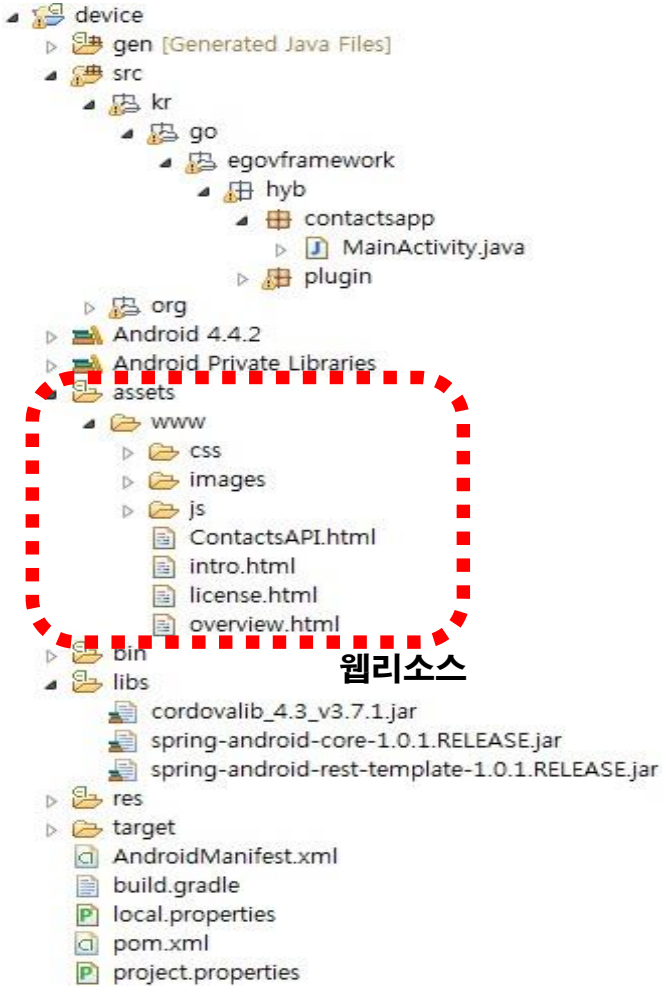
- Android

· addJavascriptInterface

→Class & JavaScript Binding

❑ 디바이스 API 실행환경을 적용한 하이브리드 앱 프로젝트를 생성하면 다음과 같이 **네이티브 코드, 웹 리소스, 네이티브 라이브러리, 환경 설정 파일**로 구성된다.

Android App eclipse Project



iOS App XCode Project

- Android Platform 기반 하이브리드 앱의 메인 클래스 (Activity)는 Phonegap 의 CordovaActivity 클래스를 상속받아 웹 리소스를 loadUrl 형태로 읽어오는 방식이다.

MainActivity.java

```
package kr.go.egovframework.hyb.cameraapp;

import android.os.Bundle;
import android.webkit.WebSettings;

import org.apache.cordova.*;

public class MainActivity extends CordovaActivity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        super.init();

        super.appView.getSettings().setAppCacheEnabled(false);
        super.appView.getSettings().setCacheMode(WebSettings.LOAD_NO_CACHE);
        super.appView.clearCache(true);
        super.appView.getSettings().setJavaScriptEnabled(true);
        super.appView.getSettings().setDomStorageEnabled(true);

        // Set by <content src="index.html" /> in config.xml
        loadUrl(launchUrl);
    }

}
```

메인 클래스(iOS Platform)

- iOS Platform 기반 하이브리드 앱의 메인 클래스 (MainViewController)는 Phonegap의 CDVViewController를 상속받아 config.xml의 url을 호출하며 주요 코드는 다음과 같다.

MainViewController.h

```
#import <Cordova/CDVViewController.h>
#import <Cordova/CDVCommandDelegateImpl.h>
#import <Cordova/CDVCommandQueue.h>

@interface MainViewController : CDVViewController

@end

@interface MainCommandDelegate : CDVCommandDelegateImpl
@end

@interface MainCommandQueue : CDVCommandQueue

@end;
```

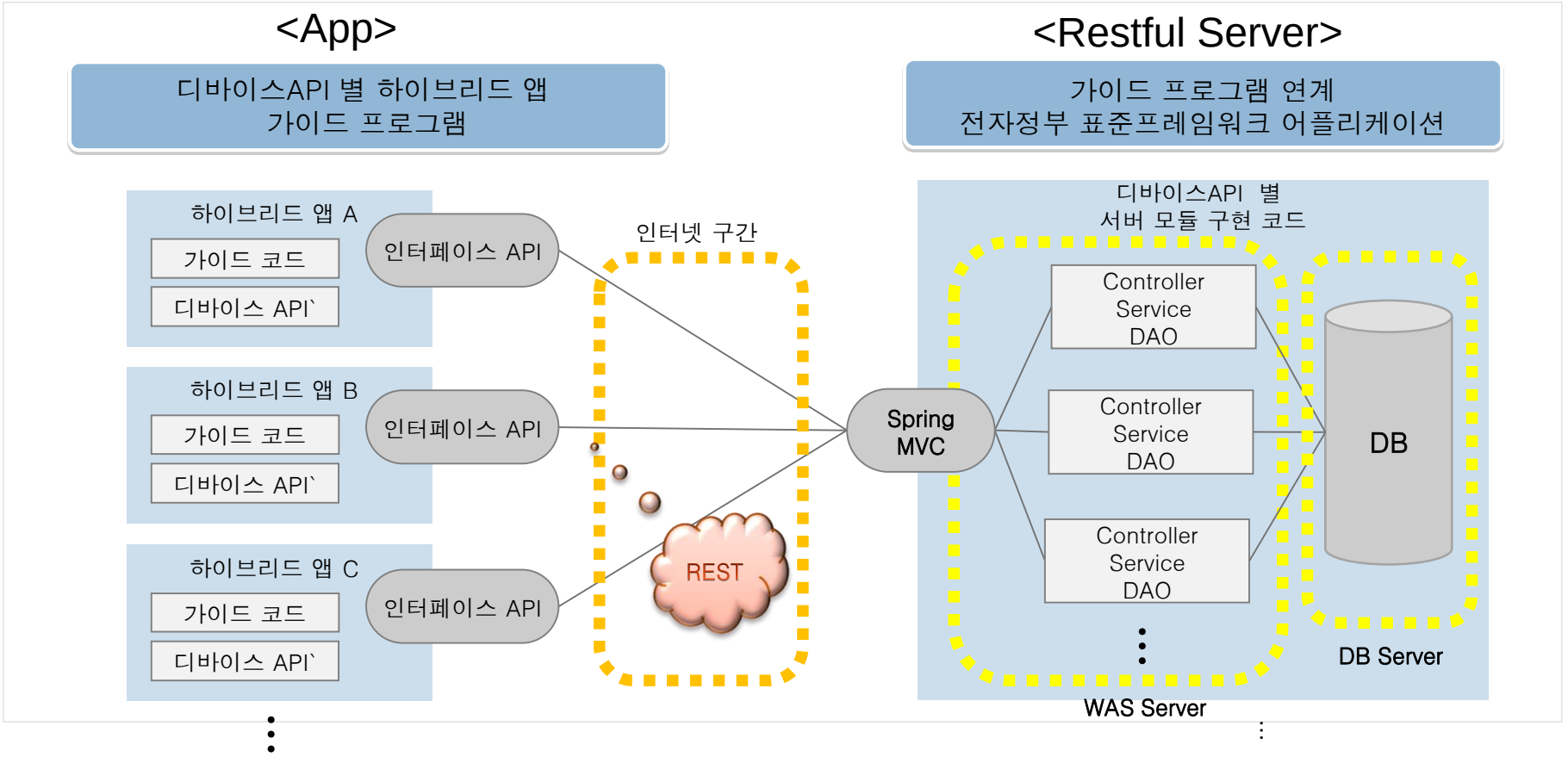
MainViewController.m

```
#import "MainViewController.h"

@implementation MainViewController

- (id)initWithNibName:(NSString*)nibNameOrNil bundle:(NSBundle*)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {

    }
    return self;
}

- (id)init
{
    self = [super init];
    if (self) {

    }
    return self;
}
```

❑ 어플리케이션의 주요 로직을 담고 있는 웹 리소스(HTML5)는 HTML, JavaScript, CSS로 구성되며 **플랫폼에 비종속적으로 iOS와 Android 플랫폼 간 재사용이 가능**하다.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0,
user-scalable=yes" />
  <title>Network API Guide</title>

  <!-- eGovFrame Common import -->
  <link rel="stylesheet" href="css/egovframework/mbl/cmm/jquery.mobile-1.4.5.css" />
  <link rel="stylesheet" href="css/egovframework/mbl/cmm/EgovMobile-1.4.5.css" />
  <script type="text/javascript" src="js/egovframework/mbl/cmm/jquery-1.11.2.min.js"></script>

  <script type="text/javascript" src="js/egovframework/mbl/cmm/jquery.history.js"></script>

  <script type="text/javascript" src="js/egovframework/mbl/cmm/jquery.mobile-1.4.5.min.js"></script>
  <script type="text/javascript" src="js/egovframework/mbl/cmm/EgovMobile-1.4.5.js"></script>
  <script type="text/javascript" src="js/egovframework/mbl/cmm/EgovHybrid.js"></script>

      <link rel="stylesheet" href="css/egovframework/mbl/hyb/guide.css" />

  <!-- iScroll.js import -->
  <script type="text/javascript" src="js/iscroll/iscroll.js"></script>

  <!-- Phonegap.js import -->
  <script type="text/javascript" charset="utf-8" src="js/egovframework/mbl/cmm/cordova.js"></script>


  <script type="text/javascript">
</script>

</head>
```

❑ 각 디바이스API별 특성에 따라 서버 모듈 연계 기능을 쉽게 이해하여 활용할 수 있도록 디바이스 API 가이드 프로그램을 개발환경을 통해 제공한다.



❑ 디바이스 API 실행환경은 추가 플러그인 및 UX 컴포넌트 활용을 위해 폰갭 프레임워크 이외에도 다음과 같은 오픈 소스 라이브러리가 활용 되었다.

Spring For Android	• 전자정부 서버 어플리케이션과 Rest 통신을 위한 안드로이드 용 네이티브 Restful 서비스 • 오픈 소스 라이브러리로 Interface Device API(Android) 개발에 사용 되었다.
ASIHTTPRequest	• 전자정부 서버 어플리케이션과 Rest 통신을 위한 iOS 용 네이티브 Restful 서비스 •오픈 소스 라이브러리로 Interface Device API(iOS) 개발에 사용 되었다.
jQuery Mobile	• 전자정부 모바일 웹 표준프레임워크의 코어 프레임워크로서 하이브리드 앱의 UX 개발 시에 사용된다. 네이티브 어플리케이션의 화면 전환 효과 및 각종 버튼 리스트 등을 활용한 개발을 위해 사용된다.
jQuery	• jQuery Mobile 프레임워크의 코어 프레임워크로 jQuery Mobile을 사용하기 위해서 필수로 사용된다. Dom Control, Ajax, Restful 서비스 연계 등 다양한 기능을 제공하며 자바스크립트를 이용한 개발을 간단하게 해준다.
iScroll	• 어플리케이션의 헤더와 푸터를 고정시킨 채 컨텐츠 내용만 스크롤 해주는 기능을 제공한다. jQuery Mobile에서 제공하지 못하는 UX 효과를 보조하기 위해 사용할 수 있다.

❑ 디바이스 API는 웹 리소스를 통해 디바이스 내의 Native 기능을 호출하기 위하여 디바이스 API 실행환경 내에서 JavaScript 형태로 제공되는 API의 모음이며 각 디바이스 API 별 특징은 다음과 같다.

순번	디바이스 API	설명
1	Accelerator	단말기의 가속도계 정보를 제공하는 API(단말기의 움직임 정보를 x, y, z 축의 값으로 제공)
2	GPS	단말기의 현재 위치에 대한 정보를 제공하는 API
3	Vibrator	단말기의 진동 및 알림음 기능을 호출 할 수 있는 API
4	Camera	단말기의 카메라 촬영 기능을 호출 할 수 있는 API
5	Contact	단말기의 주소록(연락처) 정보를 조회 및 수정 할 수 있는 API
6	Compass	단말기의 방향정보를 조회 할 수 있는 API
7	File Reader/File Writer	단말기의 내장 저장 장치의 파일을 읽기/쓰기 할 수 있는 API
8	Network	단말기의 네트워크 연결 정보를 조회 할 수 있는 API
9	Device	단말기의 기본 정보(UUID, 버전 등)을 조회 할 수 있는 API
10	Media	단말기의 오디오 파일을 컨트롤 할 수 있는 API
11	Interface	전자정부 표준프레임워크 기반 웹 서버 어플리케이션과 연계를 지원하는 API
12	NPKI	단말기에 설치 된 npki 모듈을 호출 할 수 있는 API

❑ 디바이스 API는 웹 리소스를 통해 디바이스 내의 Native 기능을 호출하기 위하여 디바이스 API 실행환경 내에서 JavaScript 형태로 제공되는 API의 모음이며 각 디바이스 API 별 특징은 다음과 같다.

순번	디바이스 API	설명
13	Push Notifications	모바일 앱 사용자에게 다양한 푸시 메세지를 전달할 수 있는 기능을 제공하는 API
14	File Opener	단말기의 사용 가능한 문서 앱의 연동을 제공하는 API
15	Streaming Media	멀티미디어(동영상)을 실시간으로 볼수 있도록 내장 미디어 플레이어로 연동하는 기능을 호출 할 수 있는 API
16	Barcode scanner	바코드, QR코드등 정보를 확인 할 수 있는기능을 호출 할 수 있는 API
17	WebResource Update	웹 리소스의 최신버전 조회 및 버전 업데이트를 진행할 수 있는 기능을 호출 할 수 있는 API
18	Device FileMgmt	디바이스 저장소 내의 폴더(디렉토리) 및 파일 관리(이동,삭제,복사) 기능을 호출 할 수 있는 API
19	JailbreakDetection	디바이스의 루팅 및 탈옥 정보 조회 기능을 호출 할 수 있는 API
20	SocketIO	웹서버의 websocket에 접속하여 양방향 데이터 처리 기능을 사용 할 수 있는 API
21	SQLite	디바이스 내 독립적인 DB를 사용 할 수 있는 기능을 지원하는 API
22	Unzip	단말기의 파일의 압축과 해제 기능을 지원하는 API

❑ 디바이스의 움직임에 대하여 X, Y, Z축의 방향을 감지한다.

유형	구분	설명
Methods	accelerometer.getCurrentAcceleration	• 현재 가속 센서가 감지한 **좌표 값** (x, y, z) 을 구한다. • navigator.accelerometer.getCurrentAcceleration(accelerometerSuccess, accelerometerError);
	accelerometer.watchAcceleration	• **특정 시간 간격**으로 가속 센서가 감지한 **좌표 값** (x, y, z) 을 받아온다. • var watchID = navigator.accelerometer.watchAcceleration(accelerometerSuccess, accelerometerError, [accelerometerOptions]);
	accelerometer.clearWatch	• watchAccelaeration()을 통해 작동 중인 가속 센서(watchID) 감지를 종료한다. • navigator.accelerometer.clearWatch(watchID);
Argument	accelerometerSuccess	• 가속 센서의 method 가 **성공**했을 때 실행하는 함수. (가속 센서의 정보를 가지는 **Acceleration 객체**를 매개변수로 가진다)
	accelerometerError	• 가속 센서의 method 가 **실패**했을 때 실행하는 함수.
	accelerometerOptions	• 감지할 가속 센서의 **옵션**을 **설정**한다. • Options - frequency: 특정 시간 간격으로 가속 센서를 감지한다. (Number) (Default: 10000)
Object	Acceleration	• 가속 센서가 감지한 가속도 정보를 가지는 객체 (**phoneGap**이 **생성**) • x: x 축 가속 값을 기록. (값 : 0 ~ 1) (Number) • y: x 축 가속 값을 기록. (값 : 0 ~ 1) (Number) • z: z 축 가속 값을 기록. (값 : 0 ~ 1) (Number) • timestamp: 측정한 milliseconds 단위의 시간 (DOMTimeStamp)

❑ 제약사항

유형	구분	지원플랫폼	설명
Methods	accelerometer.getCurrentAcceleration	Android	-
		iOS	- iPhone은 어떤 주어진 점에서 가속 센서가 감지한 좌표 값을 주지 않는다. - 특정 시간 간격으로 가속 센서를 감지해야 한다. - getCurrentAcceleration() 은 watchAcceletometer() 을 통해 마지막으로 감지된 좌표 값을 나타낸다.
	accelerometer.watchAcceleration	Android	-
		iOS	- iPhone에서는 가속 센서 감지 간격이 40 milliseconds에서 1000 milliseconds에서만 유효하게 적용된다. - 만약 3초 간격으로 watchAcceleration()을 요청하게 되면, phoneGap을 통해 1초 간격으로 가속 센서의 좌표 값을 받아오지만, phoneGap은 3초 단위로 callback함수를 요청하도록 자체 보정한다.

Device API - Accelerometer

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Acceleration Example</title>

    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    document.addEventListener("deviceready", onDeviceReady, false);

    function onDeviceReady() {
        navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);
    }

    function onSuccess(acceleration) {
        alert('Acceleration X: ' + acceleration.x + '₩n' +
              'Acceleration Y: ' + acceleration.y + '₩n' +
              'Acceleration Z: ' + acceleration.z + '₩n' +
              'Timestamp: '      + acceleration.timestamp + '₩n');
    }

   function onError() {
        alert('onError!');
    }

    </script>
  </head>
  <body>
    <h1>Example</h1>
</body>
</html>
```

❑ 디바이스의 기본 카메라 어플리케이션에 접속하는 기능을 제공한다.

유형	구분	설명
Methods	camera.getPicture	• 카메라로 사진을 찍고 찍은 사진을 가져오거나, 단말기의 앨범에서 사진을 선택해서 가져온다. • base64로 인코딩된 이미지를 가져오거나 사진 파일의 경로를 통해 사진을 가져온다. • navigator.camera.getPicture(cameraSuccess, cameraError, [cameraOptions]);
Arguments	cameraSuccess	• 사진을 찍거나 선택하는데 성공했을 때 실행하는 callback 함수.
	cameraError	• 사진을 찍거나 선택하는데 실패했을 때 메시지를 매개변수로 하는 callback 함수. • Parameters - message: 단말기에서 제공하는 오류 메시지를 문자열로 받아오는 매개변수.
	cameraOptions	• Optional parameters to customize the camera settings. • Options - quality: 이미지의 품질(해상도)을 백분율로 정의한다. (0~100) (Number) - destinationType: 결과 값의 포맷을 정의한다.. navigator.camera.DestinationType (Number) - sourceType: 포토 라이브러리, 카메라에서 찍은 사진 등 선택할 소스를 설정한다. nagivator.camera.PictureSourceType (Number) - allowEdit: 이미지 선택 전, 이미지 편집 여부. (Boolean) - encodingType: 이미지 파일의 인코딩 종류를 정의한다. navigator.camera.EncodingType (Number) - targetWidth: 이미지의 너비(pixel)를 설정한다. 비율은 고정. (Number) - targetHeight: 이미지의 높이(pixel)를 설정한다. 비율은 고정. (Number) - mediaType: pictureSourceType이 PHOTOLIBRARY 또는 SAVEDPHOTOALBUM 일 때, 미디어의 타입을 설정한다. nagivator.camera.MediaType (Number) - correctOrientation: 이미지를 담는 동안 이미지 회전 여부. (boolean) - saveToPhotoAlbum: 이미지를 담은 후, 단말기 저장 여부. (boolean)

❑ 제약사항

유형	구분	지원플랫폼	설명
Arguments	cameraOptions	Android	- allowEdit옵션을 지원하지 않는다. - sourceType은 PHOTOLIBRARY, SAVEDPHOTOALBUM 구분 없이 모두 포토 앨범을 나타낸다. - Camera.EncodingType을 지원하지 않는다. - correctOrientation을 지원하지 않는다. - saveToPhotoAlbum을 지원하지 않는다.
		iOS	- Quality를 50 이하로 설정해야 특정 단말기에서 나타나는 메모리 오류를 피할 수 있다. - destinationType.FILE_URI을 사용할 경우, 사진이 어플리케이션의 임시폴더에 저장된다. - 어플리케이션이 종료될 때, 어플리케이션 임시폴더 내의 사진들은 삭제된다. 사진들의 용량이 대체로 크기 때문에 navigator.fileMgr를 통해 단말기의 용량을 조절해야 한다.

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Capture Photo</title>
    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    var pictureSource;   // picture source
    var destinationType; // sets the format of returned value

    document.addEventListener("deviceready",onDeviceReady,false);

    function onDeviceReady() {
        pictureSource=navigator.camera.PictureSourceType;
        destinationType=navigator.camera.DestinationType;
    }

    function onPhotoDataSuccess(imageData) {

      var smallImage = document.getElementById('smallImage');

      smallImage.style.display = 'block';
      smallImage.src = "" + imageData;
    }

    function onPhotoURISuccess(imageURI) {

      var largeImage = document.getElementById('largeImage');

      largeImage.style.display = 'block';
      largeImage.src = imageURI;
    }
```

Sample Code

```
function capturePhoto() {
      navigator.camera.getPicture(onPhotoDataSuccess, onFail, { quality: 50,
        destinationType: destinationType.DATA_URL });
    }

    function capturePhotoEdit() {
      // Take picture using device camera, allow edit, and retrieve image as base64-encoded string
      navigator.camera.getPicture(onPhotoDataSuccess, onFail, { quality: 20, allowEdit: true,
        destinationType: destinationType.DATA_URL });
    }

    function getPhoto(source) {

      navigator.camera.getPicture(onPhotoURISuccess, onFail, { quality: 50,
        destinationType: destinationType.FILE_URI,
        sourceType: source });
    }

    function onFail(message) {
      alert('Failed because: ' + message);
    }

    </script>
  </head>
  <body>
    <button onclick="capturePhoto();">Capture Photo</button> <br>
    <button onclick="capturePhotoEdit();">Capture Editable Photo</button> <br>
    <button onclick="getPhoto(pictureSource.PHOTOLIBRARY);">From Photo Library</button><br>
    <button onclick="getPhoto(pictureSource.SAVEDPHOTOALBUM);">From Photo Album</button><br>
    <img style="display:none;width:60px;height:60px;" id="smallImage" src="" />
    <img style="display:none;" id="largeImage" src="" />
  </body>
</html>
```

❑ 디바이스의 오디오, 이미지, 비디오 캡쳐를 위한 기능을 제공한다.

유형	구분	설명
Methods	capture.captureAudio	• 단말기의 녹음 프로그램을 호출하여 녹음을 하고, 녹음을 완료하면 녹음한 데이터(clip)를 가져온다. • navigator.device.capture.captureAudio(CaptureCB captureSuccess, CaptureErrorCB captureError, [CaptureAudioOptions options]);
	capture.captureImage	• 단말기의 카메라 프로그램을 호출하여 사진을 찍고, 찍은 이미지를 가져온다. • navigator.device.capture.captureImage(CaptureCB captureSuccess, CaptureErrorCB captureError, [CaptureImageOptions options]);
	capture.captureVideo	• 단말기의 카메라 프로그램을 호출하여 동영상을 촬영하고 촬영한 비디오를 가져온다. • navigator.device.capture.captureVideo(CaptureCB captureSuccess, CaptureErrorCB captureError, [CaptureVideoOptions options]);
	MediaFile.getFormatData	• 미디어 파일의 포맷 정보를 가져온다. • mediaFile.getFormatData(MediaFileDataSuccessCB successCallback, [MediaFileDataErrorCB errorCallback]);
Objects	CaptureAudioOptions	• 녹음을 지원하는 captureAudio() 메소드에 대한 옵션 객체. • Properties - limit: 한번에 녹음할 수 있는 횟수를 정의한다. (기본 값 : 1, 값 : 1 이상) - duration: 녹음할 경우, 최대 녹음시간을 정의한다. - mode: capture.supportedAudioModes에서 지원하는 오디오 모드 중 하나를 지정한다.

❑ 디바이스의 오디오, 이미지, 비디오 캡쳐를 위한 기능을 제공한다.

유형	구분	설명
Objects	CaptureImageOptions	• 이미지 촬영을 지원하는 captureImage() 메소드에 대한 옵션 객체. • Properties - limit: 한번에 사진 촬영할 수 있는 횟수를 정의한다. (기본 값 : 1, 값 : 1 이상) - mode: capture.supportedImageModes 에서 지원하는 오디오 모드중 하나를 지정한다.
	CaptureVideoOptions	• 동영상 촬영을 지원하는 captureVideo() 메소드에 대한 옵션 객체. • Properties - limit: 한번에 촬영할 수 있는 횟수를 정의한다. (기본 값 : 1, 값 : 1 이상) - duration: 촬영할 경우, 최대 촬영시간을 정의한다. - mode: capture.supportedVideoModes 에서 지원하는 비디오 모드중 하나를 지정한다.
	CaptureError	• 미디어 촬영 중에 발생할 수 있는 오류코드를 기록하는 객체. • Properties - code: 아래 정의된 상수 중 하나를 값으로 사용한다.. • Constants - CaptureError.CAPTURE_INTERNAL_ERR: 카메라가 사진을 찍거나 마이크로폰이 녹음을 하는 데 실패했을 경우의 오류. - CaptureError.CAPTURE_APPLICATION_BUSY: 카메라 또는 녹음 프로그램을 다른 서비스로 인해 호출할 수 없는 경우의 오류. - CaptureError.CAPTURE_INVALID_ARGUMENT: 옵션의 설정이 올바르지 않을 경우 phoneGap API에서 발생하는 오류. - CaptureError.CAPTURE_NO_MEDIA_FILES: 사용자가 촬영을 중단하고 임의로 빠져 나왔을 때 발생하는 오류. - CaptureError.CAPTURE_NOT_SUPPORTED: 요청한 미디어 촬영이 지원하지 않을 때 발생하는 오류.

❑ 디바이스의 오디오, 이미지, 비디오 캡쳐를 위한 기능을 제공한다.

유형	구분	설명
Objects	CaptureCB	• 미디어 촬영에 성공했을 때 실행하는 함수. • function captureSuccess(MediaFile[ ] mediaFiles) { ... };
	CaptureErrorCB	• 미디어 촬영에 실패했을 때 실행하는 함수. • function captureError(CaptureError error) { ... };
	ConfigurationData	• 단말기에서 지원하는 캡처 속성들을 기록할 수 있는 객체. • Properties - type: 미디어 유형을 ASCII 코드로 인코딩한 소문자 문자열. (DOMString) - height: 이미지나 비디오의 높이(pixel)를 정의하며, 오디오의 경우 0으로 처리한다. (Number) - width: 이미지나 비디오의 너비(pixel)를 정의하며, 오디오의 경우 0으로 처리한다. (Number)
	MediaFile	• 캡처한 파일의 속성들을 기록할 수 있는 객체. • Properties - name: 경로를 제외한 파일명. (DOMString) - fullPath: 파일명을 제외한 전체경로. (DOMString) - type: 미디어 파일의 mime type. (DOMString) - lastModifiedDate: 파일의 최종수정날짜. (Date) - size: byte 단위를 가지는 파일의 크기. (Number) • Methods - MediaFile.getFormatData: 미디어 파일의 포맷을 리턴한다.

❑ 디바이스의 오디오, 이미지, 비디오 캡쳐를 위한 기능을 제공한다.

유형	구분	설명
Objects	MediaFileData	• 미디어 파일에 대한 포맷 정보를 나타내는 객체. • Properties - codecs: 오디오 및 비디오의 포맷. (DOMString) - bitrate: 오디오나 비디오의 전송 속도를 나타낸다. 이미지의 경우 0. (Number) - height: 이미지나 비디오의 높이를 나타낸다. 오디오의 경우 0. (Number) - width: 이미지나 비디오의 너비를 나타낸다. 오디오의 경우 0. (Number) - duration: 오디오와 비디오의 재생시간을 나타낸다. 이미지의 경우 0. (Number)
properties	supportedAudioModes	• 단말기에서 지원하는 오디오 녹음 포맷. (ConfigurationData[])
	supportedImageModes	• 단말기에서 지원하는 이미지 크기와 포맷. (ConfigurationData[])
	supportedVideoModes	• 단말기에서 지원하는 비디오 포맷. (ConfigurationData[])
Scope		• capture는 navigator.device 객체에 속하며, 전역에서 쓸 수 있다. • var capture = navigator.device.capture;

❑ 제약사항

유형	구분	지원플랫폼	설명
Methods	capture.captureAudio	Android	-
		iOS	- iOS는 단순한 유저 인터페이스를 가지기 때문에 기본적으로 오디오 녹음을 지원하지 않는다.
	MediaFile.getFormatData	Android	- 미디어 파일의 포맷 정보를 수집하는 API에 대해 제한이 있어, 일부 MediaFileData 속성을 지원하지 않는다. - MediaFileData : codecs, bitrate를 지원하지 않는다.
		iOS	- 미디어 파일의 포맷 정보를 수집하는 API에 대해 제한이 있어, 일부 MediaFileData 속성을 지원하지 않는다. - MediaFileData : codecs 을 지원하지 않는다. - MediaFileData : bitrate 는 iOS4 에서만 지원한다.
Objects	CaptureAudioOptions	Android	- duration 파라미터를 지원하지 않는다. - 프로그램적으로 녹음 시간을 제한할 수 없다. - mode 파라미터를 지원하지 않는다. - 오디오 녹음 포맷을 프로그램적으로 정의할 수 없다. - 오디오 녹음의 포맷은 audio/amr 만을 사용할 수 있다.
		iOS	- limit 파라미터를 지원하지 않는다. - 한 번에 하나의 녹음만 실행할 수 있다. - 오디오 녹음 포맷을 프로그램적으로 정의할 수 없다. - 오디오 녹음의 포맷은 audio/wav 만을 사용할 수 있다.

❑ 제약사항

유형	구분	지원플랫폼	설명
Objects	CaptureImageOptions	Android	- mode 파라미터를 지원하지 않는다. - 이미지의 크기와 포맷을 프로그램적으로 정의할 수 없다. - 하지만 단말기 사용자가 이미지의 크기는 지정할 수 있다. - 이미지는 JPEG (image/jpeg) 로 저장된다.
		iOS	- limit 파라미터를 지원하지 않는다. - 한 번에 하나의 이미지만을 캡처할 수 있다 - mode 파라미터를 지원하지 않는다. - 이미지의 크기와 포맷을 프로그램적으로 정의할 수 없다. - 이미지는 JPEG (image/jpeg) 로 저장된다.
	CaptureVideoOptions	Android	- duration 파라미터를 지원하지 않는다. - 비디오 촬영 크기를 프로그램적으로 제한할 수 없다. - mode 파라미터를 지원하지 않는다. - 비디오의 크기와 파라미터를 프로그램적으로 정의할 수 없다. - 비디오는 3GPP (videp/3gpp) 로 기본적으로 촬영된다.
		iOS	- limit 파라미터는 지원하지 않는다. - 한 번에 하나의 비디오 촬영만 할 수 있다. - duration 파라미터를 지원하지 않는다. - 비디오 촬영 크기를 프로그램적으로 제한할 수 없다. - mode 파라미터를 지원하지 않는다. - 비디오의 크기와 파라미터를 프로그램적으로 정의할 수 없다. - 비디오는 MOV (video/quicktime) 로 기본적으로 촬영된다.

❑ 제약사항

유형	구분	지원플랫폼	설명
Objects	MediaFileData	Android	• MediaFileData는 다음과 같이 지원한다. - codecs: 지원하지 않음. 항상 null 이다. - bitrate: 지원하지 않음. 항상 0 이다. - height: 지원. (단, image와 video 파일인 경우). - width: 지원. (단, image와 video 파일인 경우). - duration: 지원. (단, audio와 video 파일인 경우).
		iOS	• MediaFileData는 다음과 같이 지원한다. - codecs: 지원하지 않음. 항상 null 이다. - bitrate: iOS4 단말기에서 오디오인 경우에만 지원. (이미지와 비디오인 경우, 항상 0.) - height: 지원. (단, image와 video 파일인 경우). - width: 지원. (단, image와 video 파일인 경우). - duration: 지원. (단, audio와 video 파일인 경우).

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Capture Audio</title>
    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8" src="json2.js"></script>
    <script type="text/javascript" charset="utf-8">
    function captureSuccess(mediaFiles) {
        var i, len;
        for (i = 0, len = mediaFiles.length; i < len; i += 1) {
            uploadFile(mediaFiles[i]);
        }
    }
    function captureError(error) {
        var msg = 'An error occurred during capture: ' + error.code;
        navigator.notification.alert(msg, null, 'Uh oh!');
    }
    function captureAudio() {
        navigator.device.capture.captureAudio(captureSuccess, captureError, {limit: 2});
    }
    function uploadFile(mediaFile) {
        var ft = new FileTransfer(),
        path = mediaFile.fullPath,
        name = mediaFile.name;
        ft.upload(path, "http://my.domain.com/upload.php", function(result) {
                console.log('Upload success: ' + result.responseCode);
                console.log(result.bytesSent + ' bytes sent');
            }, function(error) {
                console.log('Error uploading file ' + path + ': ' + error.code);
            }, { fileName: name });
    }
    </script> </head>
    <body> <button onclick="captureAudio();">Capture Audio</button> <br> </body>
</html>
```

Device API - Compass

❑ 디바이스의 방향(방위각)을 구하는 기능을 제공한다.

유형	구분	설명
Methods	compass.getCurrentHeading	• 단말기의 방위 센서가 감지한 현재 방향을 가져온다. (0 ~ 359.9) • navigator.compass.getCurrentHeading(compassSuccess, compassError, compassOptions);
	compass.watchHeading	• 특정 시간 간격으로 방위 센서가 감지한 빙향 정보를 가져온다. • var watchID = navigator.compass.watchHeading(compassSuccess, compassError, [compassOptions]);
	compass.clearWatch	• watchHeading()을 통해 작동 중인 방위 센서(watchID)를 중지한다. • navigator.compass.clearWatch(watchID); • watchID: The ID returned by compass.watchHeading.
	compassSuccess	• 방위 센서의 메소드가 성공했을 때 실행하는 함수. • Parameters - heading: 방위 센서가 감지한 방위 정보. (compassHeading)
	compassError	• 방위 센서의 메소드가 실패했을 때 실행하는 함수.

❑ 디바이스의 방향(방위각)을 구하는 기능을 제공한다.

유형	구분	설명
Arguments	compassError	• 방위 센서의 메소드가 실패했을 때의 파라미터.
	compassOptions	• 방위 센서의 옵션을 정의한다. • Options - frequency: 특정 시간 간격으로 방위 센서를 감지한다. (Number) (기본: 100) - filter: watchHeadingFilter() 메소드에 사용하는 속성으로, 특정 filter 값 이상일 때만 감지한다. (Number)
	compassHeading	• 방위 센서에서 전달받은 방향 정보를 나타낸다. • Properties - magneticHeading: 자기 방위 값을 나타낸다. (0 ~ 359.99) (Number) - trueHeading: 진북을 기준으로 진방위 값을 나타낸다. (0 ~ 359.99) 음수는 진방위 값을 나타낼 수 없음을 의미한다. (Number) - headingAccuracy: 방위 각도에 대한 편차 값을 나타낸다. (Number) - timestamp: 방위를 감지한 시각을 milliseconds 단위로 나타낸다.

❑ 제약사항

유형	구분	지원플랫폼	설명
Argumetns	compassOptions	Android	- filter 를 지원하지 않는다.
		iOS	-
	compassHeading	Android	- trueHeading 을 지원하지 않는다. magneticHeading과 동일. - magneticHeading과 magneticHeading 이 동일하기 때문에 headingAccuracy 는 항상 0 이다.
		iOS	- trueHeading은 navigator.geolocation.watchLocation()이 동작하고 있을 때만 감지된다. - iOS4 이상의 단말기에서는 단말기가 회전했을 때 trueHeading 값을 감지한다.

Device API - Compass

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Compass Example</title>

    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    document.addEventListener("deviceready", onDeviceReady, false);

    function onDeviceReady() {
        navigator.compass.getCurrentHeading(onSuccess, onError);
    }

    function onSuccess(heading) {
        alert('Heading: ' + heading.magneticHeading);
    }

    function onError(compassError) {
        alert('Compass Error: ' + compassError.code);
    }

    </script>
  </head>
  <body>
    <h1>Example</h1>
    <p>getCurrentHeading</p>
  </body>
</html>
```

Device API - Connection

❑ 디바이스의 방향(방위각)을 구하는 기능을 제공한다.

유형	구분	설명
Properties	connection.type	• 단말기에서 지원하는 통신방법에 대한 정보를 나타낸다.
Constants		• Connection.UNKNOWN : 알 수 없는 통신 방식. • Connection.ETHERNET : 이더넷 통신 방식. • Connection.WIFI : Wi-Fi 통신 방식. • Connection.CELL_2G : 2G 통신 방식. • Connection.CELL_3G : 3G 통신 방식. • Connection.CELL_4G : 4G 통신 방식. • Connection.NONE : 통신을 지원하지 않음.

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>navigator.network.connection.type Example</title>
    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    document.addEventListener("deviceready", onDeviceReady, false);
    function onDeviceReady() {
        checkConnection();
    }

    function checkConnection() {
        var networkState = navigator.network.connection.type;

        var states = {};
        states[Connection.UNKNOWN]  = 'Unknown connection';
        states[Connection.ETHERNET] = 'Ethernet connection';
        states[Connection.WIFI]     = 'WiFi connection';
        states[Connection.CELL_2G]  = 'Cell 2G connection';
        states[Connection.CELL_3G]  = 'Cell 3G connection';
        states[Connection.CELL_4G]  = 'Cell 4G connection';
        states[Connection.NONE]     = 'No network connection';

        alert('Connection type: ' + states[networkState]);
    }

    </script>
  </head>
  <body>
    <p>A dialog box will report the network state.</p>
  </body>
</html>
```

❑ 디바이스의 연락처 데이터베이스에 접속하는 기능을 제공한다.

유형	구분	설명
Methods	contacts.create	• 새로운 Contact 객체를 생성한다. • var contact = navigator.contacts.create(properties);
	contacts.find	• 단말기가 데이터베이스를 검색하여, 각각의 연락처 정보를 갖는 하나 이상의Contact 객체를 받는다. • navigator.contacts.find(contactFields, contactSuccess, contactError, contactFindOptions); • Parameters - contactFields: 검색하고자 하는 연락처 정보를 정의한다. (값 : Contact 객체의 값) (DOMString[]) [필수] - contactSuccess: 연락처 데이터베이스 검색이 성공하였을 때 실행하는 함수. [필수] - contactError: 검색에 실패하였을 때 실행하는 함수. [선택] - contactFindOptions: 검색 조건을 정의한다. [선택]
Arguments	contactFields	• contacts.find 함수 파라미터로 사용된다. Contact 객체의 필드로 검색 결과로 나타나는 내용들을 정의하는데 사용된다. • ["name", "phoneNumbers", "emails"]
	contactSuccess	• contacts.find 함수 실행 성공 후에 수행되며 Contact 배열을 결과 값으로 가진다.
	contactError	• contact 관련 함수 실행에 실패하였을 때 수행된다.

	contactFindOptions	- contacts.find 함수의 선택적 옵션이다.
		- 연락처 데이터베이스 검색 후에 나타나는 결과들의 내용을 정의한다..
		- {
		filter: "",
		multiple: true,
		};

❑ 디바이스의 연락처 데이터베이스에 접속하는 기능을 제공한다.

유형	구분	설명
Objects	Contact	• 사용자의 개인 혹은 사무 용도의 연락처 정보를 가지는 객체이다.. • Properties - id: 연락처에 대한 고유한 Id이다. (DOMString) - displayName: 사용자의 화면에 출력하는 연락처 이름이다. (DOMString) - name: 연락처에 저장된 이름 정보를 모두 포함하고 있다. (ContactName) - nickname: 연락처에 저장된 별명이다. (DOMString) - phoneNumbers: 전화번호를 배열 형태로 저장하고 있다. (ContactField[]) - emails: 관련 연락처에 대한 이메일 정보의 배열이다. (ContactField[]) - addresses: 관련 연락처의 주소에 대한 배열이다. (ContactAddresses[]) - ims: 관련 연락처에 대한 메신저 주소 정보의 배열이다. (ContactField[]) - organizations: 소속 단체 정보에 대한 배열이다. (ContactOrganization[]) - birthday: 관련 연락처에 대한 생일이다. (Date) - note: 관련 연락처에 대한 메모이다. (DOMString) - photos: 관련 연락처에 대한 사진이다. (ContactField[]) - categories: 연락처에 대한 분류 카테고리의 배열이다. (ContactField[]) - urls: 연락처에 대한 홈페이지 주소의 배열이다. (ContactField[]) • Methods - clone: 호출한 Contact 객체를 복제하여 새로운 Contact 객체를 생성한다. 새로운 Contact 객체는 null이기 때문에 새로운 Contact 객체로 저장할 수 있도록 지원한다. - remove: 연락처 데이터베이스에서 호출한 Contact 객체를 삭제하고 성공 했을 경우에는 Contact 객체를, 실패했을 경우에는 ContactError 객체를 전달받는다. - save: 신규 Contact 객체 경우에는 단말기 연락처 데이터베이스에 등록하고, 존재하는 경우 연락처를 수정한다.

❑ 디바이스의 연락처 데이터베이스에 접속하는 기능을 제공한다.

유형	구분	설명
Objects	ContactName	• Contact 객체의 name 속성에 해당하는 관련 속성들의 객체. • Properties - formatted: 해당 연락처의 전체이름. (DOMString) - familyName: 전체 이름에서 '성'을 의미한다. (DOMString) - givenName: 전체 이름에서 '이름'을 의미한다. (DOMString) - middleName: 전체 이름에서 성과 이름 사이의 '중간 이름'. (DOMString) - honorificPrefix: 'Mr.' 나 'Dr.' 같은 접두 호칭을 의미한다. (DOMString) - honorificSuffix: 'Esq.'와 같은 접미 호칭을 의미한다. (DOMString)
	ContactField	• Contact 객체의 phone numbers, email address, urls 속성에서 사용하는 필드 객체. • Properties - type: 필드 유형을 정의한다. (DOMString) (예: 'home' 은 집전화, 'mobile' 은 휴대전화.) - value: phone number 등과 같은 필드에 대한 값을 정의한다. (DOMString) - pref: 'true' 일 경우 필드 내, 다수의 값들의 대표 값을 지정한다. (boolean)
	ContactAddress	• Contact 객체의 주소 속성에 대한 객체. • Properties - pref: ContactAddress 객체의 대표 값 여부. (boolean) - type: 필드 유형을 정의한다. (DOMString) (예: 'home'은 집.) - formatted: 출력을 위한 전체 주소. (DOMString) - streetAddress: 동 / 읍 / 면 / 가 와 같은 전체 주소. (DOMString) - locality: '시' / '도' 표기. (DOMString) - region: '주' / '지방' 표기. (DOMString) - postalCode: 우편번호 표기. (DOMString) - country: 도시이름 표기. (DOMString)

❑ 디바이스의 연락처 데이터베이스에 접속하는 기능을 제공한다.

유형	구분	설명
Objects	ContactOrganization	• Contact 객체의 organization 속성에 관련된 객체. • Properties - pref: ContactOrganization 객체의 대표 값 여부. (boolean) - type: 필드 유형을 정의한다. (DOMString) (예: 'work' 은 직장.) - name: 소속의 이름을 정의한다. (DOMString) - department: 소속 부서를 정의한다. (DOMString) - title: 소속 내의 직함을 정의한다. (DOMString)
	ContactFindOptions	• contacts.find 함수에서 사용되는 검색 조건을 정의하는 객체. • Properties - filter: 검색어 또는 검색조건을 정의한다. (DOMString) (Default: "") - multiple: 검색 결과으로 다수의 연락처를 가져올 것인지의 여부. (Boolean) (기본: false)
	ContactError	• 에러가 발생했을 경우, contactError 함수 호출 시, 전달받는 객체. • Properties - code: 에러 코드를 나타낸다.

❑ 제약사항

유형	구분	지원플랫폼	설명
Objects	Contact	Android	• Android 2.x - categories: 지원하지 않음. 항상 null 값을 리턴한다. • Android 1.x - name: 지원하지 않음. 항상 null 값을 리턴한다. - nickname: 지원하지 않음. 항상 null 값을 리턴한다. - birthday: 지원하지 않음. 항상 null 값을 리턴한다. - photos: 지원하지 않음. 항상 null 값을 리턴한다. - categories: 지원하지 않음. 항상 null 값을 리턴한다. - urls: 지원하지 않음. 항상 null 값을 리턴한다.
		iOS	- displayName: 지원하지 않음. 항상 null 값을 리턴한다. ContractName 객체가 있을 경우에는 name 이나 nickname, “” 을 갖는다. - birthday: 자바스크립트의 Date 객체를 사용해야 한다.. - photos: 호출한 사진은 단말기의 임시폴더에 저장되며, File URL을 리턴한다. 임시 폴더는 앱이 종료될 때 삭제된다. - categories: 지원하지 않음. 항상 null 값을 리턴한다.

❑ 제약사항

유형	구분	지원플랫폼	설명
Objects	ContactAddress	Android	• Android 2.x - pref: 지원하지 않음. 항상 false 값을 리턴한다. • Android 1.x - pref: 지원하지 않음. 항상 false 값을 리턴한다. - type: 지원하지 않음. 항상 null 값을 리턴한다. - streetAddress: 지원하지 않음. 항상 null 값을 리턴한다. - locality: 지원하지 않음. 항상 null 값을 리턴한다. - region: 지원하지 않음. 항상 null 값을 리턴한다. - postalCode: 지원하지 않음. 항상 null 값을 리턴한다. - country: 지원하지 않음. 항상 null 값을 리턴한다.
		iOS	- pref: 지원하지 않음. 항상 false 값을 리턴한다. - formatted: 현재 지원하지 않음.
	ContactField	Android	- pref: 지원하지 않음. 항상 false 값을 리턴한다.
		iOS	- pref: 지원하지 않음. 항상 false 값을 리턴한다.

❑ 제약사항

유형	구분	지원플랫폼	설명
Objects	ContactName	Android	- formatted: 부분적으로 지원함. honorificPrefix, givenName, middleName, familyName and honorificSuffix 순으로 리턴 하지만, 저장되지 않는다.
		iOS	- formatted: 부분적으로 지원함. Composite Name을 리턴 하지만, 저장되지 않는다.
	ContactOrganization	Android	• Android 2.x - pref: 지원하지 않음. 항상 false 값을 리턴한다. • Android 1.x - pref: 지원하지 않음. 항상 false 값을 리턴한다. - type: 지원하지 않음. 항상 null 값을 리턴한다. - title: 지원하지 않음. 항상 null 값을 리턴한다.
		iOS	- pref: 지원하지 않음. 항상 false 값을 리턴한다. - type: 지원하지 않음. 항상 null 값을 리턴한다. - name: 부분적으로 지원함. Organization에 대입한 배열중, 첫 번째 ContactOrganization만 유효하다. - department: 부분적으로 지원함. Organization에 대입한 배열중, 첫 번째 ContactOrganization만 유효하다. - title: 부분적으로 지원함. Organization에 대입한 배열중, 첫 번째 ContactOrganization만 유효하다.

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contact Example</title>
    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    document.addEventListener("deviceready", onDeviceReady, false);

    function onDeviceReady() {

        var contact = navigator.contacts.create();
        contact.displayName = "Plumber";
        contact.nickname = "Plumber";       //specify both to support all devices
        var name = new ContactName();
        name.givenName = "Jane";
        name.familyName = "Doe";
        contact.name = name;

        contact.save(onSaveSuccess,onSaveError);

        var clone = contact.clone();
        clone.name.givenName = "John";
        console.log("Original contact name = " + contact.name.givenName);
        console.log("Cloned contact name = " + clone.name.givenName);

        contact.remove(onRemoveSuccess,onRemoveError);
    }
```

Sample Code

```
function onSaveSuccess(contact) {
        alert("Save Success");
    }
    function onSaveError(contactError) {
        alert("Error = " + contactError.code);
    }

    function onRemoveSuccess(contacts) {
        alert("Removal Success");
    }
    function onRemoveError(contactError) {
        alert("Error = " + contactError.code);
    }

    </script>
  </head>
  <body>
    <h1>Example</h1>
    <p>Find Contacts</p>
  </body>
</html>
```

❑ 디바이스의 하드웨어 또는 소프트웨어에 대한 정보를 제공한다.

유형	구분	설명
Properties	device.name	• 단말기의 제품명이나 모델명. • var string = device.name;
	device.phonegap	• 웹앱에 탑재한 폰갭 API의 버전 정보. • var string = device.phonegap;
	device.platform	• 단말기의 운영체제(OS)의 이름. • var string = device.platform;
	device.uuid	• 단말기 출고 시, 제작자가 부여한 범용 고유 식별자(UUID). • var string = device.uuid;
	device.version	• 단말기의 운영체제(OS)의 버전 정보. • var string = device.version;

Variable Scope		• 범용적으로 window 객체로부터 단말기 정보를 구할 수 있다.
		• // These reference the same `device`
		//
		var phoneName = window.device.name;
		var phoneName = device.name;

❑ 제약사항

유형	구분	지원플랫폼	설명
Properties	device.name	Android	• 단말기별로 차이가 있지만, 모델명 대신에 제품명을 출력한다. -제품명은 간혹 코드를 나타낸다. -Nexus One 은 "Passion", Motorola Droid 은 "voles"
		iOS	• 모델명 대신 단말기에 등록한 사용자의 이름을 출력한다. -iTunes 내에 설정한 이름. -e.g. "Joe's iPhone"
	device.platform	Android	-
		iOS	- 모든 단말기는 'iPhone' 으로 출력한다. - 공급자에 따라 다르게 출력될 수 있다.
	device.version	Android	- Android 2.1 미만 버전에서는 지원하지 않는다.
		iOS	-

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Device Properties Example</title>

    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    document.addEventListener("deviceready", onDeviceReady, false);

    function onDeviceReady() {
        var element = document.getElementById('deviceProperties');

        element.innerHTML = 'Device Name: '     + device.name     + '<br />' +
                            'Device Cordova: '  + device.cordova  + '<br />' +
                            'Device Platform: ' + device.platform + '<br />' +
                            'Device UUID: '     + device.uuid     + '<br />' +
                            'Device Version: '  + device.version  + '<br />';
    }

    </script>
  </head>
  <body>
    <p id="deviceProperties">Loading device properties...</p>
  </body>
</html>
```

❑ W3C의 File API를 기반으로 파일 읽기와 쓰기 기능의 API를 제공한다.

유형	구분	설명
Objects	File	• 단일 파일의 정보를 갖는 객체. • properties: - name: 파일의 이름. (DOMString) - fullPath: 파일의 이름을 포함하는 전체 경로. (DOMString) - type: 파일의 mime 타입. (DOMString) - lastModifiedDate: 파일이 마지막으로 수정된 날짜. (Date) - size: bytes 단위의 파일 크기. (long)
	FileReader	• 파일을 읽을 수 있도록 하는 객체. • properties: - readyState: 객체의 상태 값을 제공한다. (EMPTY, LOADING, DONE) - result: 읽어온 파일의 내용을 가지고 있는 속성. (DOMString) - error: 오류 정보를 가지고 있는 속성. (FileError) - onloadstart: 파일을 읽기 시작했을 때 호출하는 함수. (Function) - onload: 파일을 성공적으로 읽었을 때 호출하는 함수. (Function) - onabort: abort() 함수에 의해서 중단되거나 기타 다른 사유로 인해 중단 됐을 때 실행할 함수. (Function) - onerror: 파일을 읽는 도중, 에러가 발생했을 때 호출하는 함수. (Function) - onloadend: 성공, 실패와 무관하게 파일 읽기 완료했을 때 호출하는 함수. (Function) • Methods : - abort: 파일 읽기를 강제로 중단 시키는 함수. - readAsDataURL: Base64 인코딩으로 읽고 오는 함수. - readAsText: 파일을 텍스트로 읽어오는 함수.

❑ W3C의 File API를 기반으로 파일 읽기와 쓰기 기능의 API를 제공한다.

유형	구분	설명
Objects	FileWriter	• 파일 쓰기 정보를 갖고있는 객체. • properties: - readyState: 객체의 상태 값. (INIT, WRITING, DONE) - fileName: 쓰기할 파일의 이름. (DOMString) - length: 쓰기할 파일의 길이. (long) - position: 쓰기할 파일 포인터의 위치. (long) - error: 오류 정보를 갖고 있는 객체. (FileError) - onwritestart: 쓰기를 시작할 때 호출하는 함수. (Function) - onwrite: 쓰기를 완료했을 때 호출하는 함수. (Function) - onabort: abort() 함수에 의해서 중단되거나 기타 다른 사유로 인해 중단 됐을 때 실행할 함수. (Function) - onerror: 파일을 쓰는 도중, 에러가 발생했을 때 호출하는 함수. (Function) - onloadend: 성공, 실패와 무관하게 파일 쓰기 완료했을 때 호출하는 함수. (Function) • Methods : - abort: 파일 쓰기를 강제로 중단하는 함수. - seek: 지정한 위치로 파일 포인터를 이동하는 함수. - truncate: 지정한 위치까지만 보존하고, 그 이후 데이터는 삭제하는 함수. - write: 현재 파일 포인터로부터 문자열을 파일에 쓰는 함수.
	FileSystem	• 파일 시스템 정보를 갖는 객체. • properties: - name: 파일 시스템의 이름. (DOMString) - root: 파일 시스템의 루트(root) 디렉토리 객체. (DirectoryEntry)

❑ W3C의 File API를 기반으로 파일 읽기와 쓰기 기능의 API를 제공한다.

유형	구분	설명
Objects	FileEntry	• 파일 시스템에 있는 디렉토리를 지정하는 객체. • properties: - isFile: 항상 true이다. (boolean) - isDirectory: 항상 false이다. (boolean) - name: 경로를 제외한 최종 파일 이름. (DOMString) - fullPath: 루트(root)를 포함하는 전체 경로. (DOMString) • NOTE: 다음은 W3C에서 정의하고 있지만, PhoneGap에서는 따르지 않는다: - filesystem: FileEntry 에 설정되어 있는 FileSystem 객체. (FileSystem) • Methods : - getMetadata: 파일에 대한 메타데이터. - moveTo: 파일 시스템의 다른 위치로 파일을 이동한다. - copyTo: 파일 시스템의 다른 위치로 파일을 복사한다. - toURI: 파일의 위치를 URI 형식으로 변환하여 리턴한다. - remove: 파일을 삭제한다.. - getParent: 현재 디렉토리의 상위 경로를 구한다. - createWriter: FileWriter 객체를 생성한다. - file: 파일 정보를 가지는 File 객체를 생성한다.

❑ W3C의 File API를 기반으로 파일 읽기와 쓰기 기능의 API를 제공한다.

유형	구분	설명
Objects	DirectoryEntry	• W3C Directories and Systems을 따르는 파일 시스템 디렉토리 관련 객체. • properties: - isFile: 항상 false 이다. (boolean) - isDirectory: 항상 true 이다. (boolean) - name: 경로를 제외한 DirectoryEntry의 이름. (DOMString) - fullPath: 루트(root)로 부터의 절대경로. (DOMString) • NOTE: 다음은 W3C에서 정의하고 있지만, PhoneGap에서는 따르지 않는다: - filesystem: DirectoryEntry 에 설정되어 있는 FileSystem 객체. (FileSystem) • Methods : - getMetadata: 파일에 대한 메타데이터. - moveTo: 파일 시스템의 다른 위치로 디렉토리를 이동한다. - copyTo: 파일 시스템의 다른 위치로 디렉토리를 복사한다. - toURI: 디렉토리의 위치를 URI 형식으로 변환하여 리턴한다. - remove: 디렉토리를 삭제한다.. - getParent: 현재 디렉토리의 상위 경로를 구한다. - createWriter: FileWriter 객체를 생성한다. - getDirectory: 폴더를 가져오거나, 생성한다. - getFile: 파일을 생성하거나, 가져온다. - removeRecursively: 하위 폴더의 모든 컨텐츠를 삭제한다.

❑ W3C의 File API를 기반으로 파일 읽기와 쓰기 기능의 API를 제공한다.

유형	구분	설명
Objects	DirectoryReader	• 디렉토리 내의 파일들과 디렉토리들을 읽는 객체. • Directories and Systems 규약을 따른다. • Methods : - readEntries: 디렉토리 내의 객체들을 읽어온다.
	FileTransfer	• 서버에 파일을 업로드할 수 있도록 하는 파일 전송 객체. • Methods : - upload: 서버로 파일을 전송한다.
	FileUploadOptions	• 파일 업로드시, 전달할 매개변수를 정의하는 업로드 속성객체. • Properties : - fileKey: 업로드할 파일의 변수명. (DOMString) 설정하지 않으면 “file” 로 정의된다. - fileName: 서버에 저장될 파일의 변수명. (DOMString) 설정하지 않으면 "image.jpg“ 로 정의된다. - mimeType: 데이터를 업로드시, mimeType을 설정한다. (DOMString) 설정하지 않으면 "image/jpeg“ 로 정의된다. - params: 서버에 추가로 전달할 매개변수. (Object) - chunkedMode: 대형 스트림 모드 여부. (Boolean) 설정하지 않으면 "true“ 로 정의된다.
	FileUploadResult	• FileTransfer 객체를 통해 업로드에 성공했을 때 서버에서 받아오는 객체. • Properties : - bytesSent: 서버에 업로드한 용량 정보. (long) - responseCode: 서버에서 받아온 HTTP 응답코드. (long) - response: 서버에서 응답받은 데이터. (DOMString)

❑ W3C의 File API를 기반으로 파일 읽기와 쓰기 기능의 API를 제공한다.

유형	구분	설명
Objects	Flags	• 파일이나 디렉토리를 호출할 때, 존재하지 않으면 생성할 것인지를 설정하는 객체. • Properties : - create: 파일이나 디렉토리가 없을 때, 생성할 것인지의 여부. (boolean) - exclusive: create 속성과 같이 사용해야 하며, 이미 파일이나 디렉토리가 있을 경우 생성 실패로 처리할 것인지의 여부. (boolean)
	LocalFileSystem	• 단말기의 루트(root) 파일 시스템에 접근할 수 있도록 지원하는 파일 시스템. • Methods : - requestFileSystem: filesystem 객체를 요청하는 함수. (Function) - resolveLocalFileSystemURI: URI로 DirectoryEntry 나 FileEntry 객체를 요청하는 함수. (Function) • Constants : - LocalFileSystem.PERSISTENT: 보관용 저장 공간이며, 사용자가 임의로 삭제할 수 없는 공간이다. - LocalFileSystem.TEMPORARY: 임시용 저장 공간이며, 안정성을 보장하지 못하는 공간이다.
	Metadata	• 파일이나 디렉토리의 상태 정보를 제공하는 객체. • Properties : - modificationTime: 파일이나 디렉토리의 최종 수정일 정보. (Date)

❑ W3C의 File API를 기반으로 파일 읽기와 쓰기 기능의 API를 제공한다.

유형	구분	설명
Objects	FileError	• File API 함수 수행 시, 에러가 발생했을 때 에러 정보를 갖는 객체. • Properties : - code: 다음과 같은 에러 코드를 갖는다. • Constants : - FileError.NOT_FOUND_ERR - FileError.SECURITY_ERR - FileError.ABORT_ERR - FileError.NOT_READABLE_ERR - FileError.ENCODING_ERR - FileError.NO_MODIFICATION_ALLOWED_ERR - FileError.INVALID_STATE_ERR - FileError.SYNTAX_ERR - FileError.INVALID_MODIFICATION_ERR - FileError.QUOTA_EXCEEDED_ERR - FileError.TYPE_MISMATCH_ERR - FileError.PATH_EXISTS_ERR
	FileTransferError	• 파일 전송 시, 에러가 발생했을 때 에러 정보를 갖는 객체. • Properties : - code: 다음과 같은 에러코드를 갖는다. (int) • Constants : - FileTransferError.FILE_NOT_FOUND_ERR - FileTransferError.INVALID_URL_ERR - FileTransferError.CONNECTION_ERR

❑ 제약사항

유형	구분	지원플랫폼	설명
Objects	FileReader	Android	-
		iOS	- Encoding 파라미터는 지원하지 않음 - UTF8 인코딩이 항상 사용된다.
	FileUploadResult	Android	-
		iOS	- 업로드 성공 시, FileUploadResult 객체에 responseCode와 bytesSent을 갖지 않는다.
	DirectoryEntry	Android	• W3C Directories and Systems 규약 기반이다.. • 다음은 W3C에서 정의하고 있지만, PhoneGap에서는 따르지 않는다: - filesystem: DirectoryEntry 에 설정되어 있는 FileSystem 객체. (FileSystem)
		iOS	
	FileEntry	Android	• W3C Directories and Systems 규약 기반이다.. • 다음은 W3C에서 정의하고 있지만, PhoneGap에서는 따르지 않는다: - filesystem: FileEntry 에 설정되어 있는 FileSystem 객체. (FileSystem)
		iOS	

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>FileReader Example</title>
    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    function onLoad() {
        document.addEventListener("deviceready", onDeviceReady, false);
    }

    function onDeviceReady() {
        window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, gotFS, fail);
    }

    function gotFS(fileSystem) {
        fileSystem.root.getFile("readme.txt", null, gotFileEntry, fail);
    }

    function gotFileEntry(fileEntry) {
        fileEntry.file(gotFile, fail);
    }

    function gotFile(file){
        readDataUrl(file);
        readAsText(file);
    }
```

Sample Code

```
function readDataUrl(file) {
        var reader = new FileReader();
        reader.onloadend = function(evt) {
            console.log("Read as data URL");
            console.log(evt.target.result);
        };
        reader.readAsDataURL(file);
    }

    function readAsText(file) {
        var reader = new FileReader();
        reader.onloadend = function(evt) {
            console.log("Read as text");
            console.log(evt.target.result);
        };
        reader.readAsText(file);
    }

    function fail(evt) {
        console.log(evt.target.error.code);
    }

    </script>
  </head>
  <body>
    <h1>Example</h1>
    <p>Read File</p>
  </body>
</html>
```

❑ 디바이스의 GPS 센서에 의해 제공되는 geolocatino 객체에 접근하는 API이다.

유형	구분	설명
Methods	geolocation.getCurrentPosition	• 단말기의 현재 위치를 Position 객체에 담아 구한다. • navigator.geolocation.getCurrentPosition(geolocationSuccess, [geolocationError], [geolocationOptions]); • Parameters - geolocationSuccess: 현재 위치를 성공적으로 구하였을 때, 수행하는 함수. - geolocationError: (선택) 에러가 발생하였을 때, 수행하는 함수. - geolocationOptions: (선택) geolocation 옵션.
	geolocation.watchPosition	• 단말기의 위치 변동을 감지한다. • var watchId = navigator.geolocation.watchPosition(geolocationSuccess, [geolocationError], [geolocationOptions]); • Parameters - geolocationSuccess: 위치 변동을 성공적으로 구하였을 때, 수행하는 함수. - geolocationError: (선택) 에러가 발생하였을 때, 수행하는 함수. - geolocationOptions: (선택) geolocation 옵션. • Returns - String: 특정 간격으로 감지하고 있는 watch Id를 리턴한다. 또한, 이 값과 geolocation.clearWatch를 통해 감지를 중단할 수 있다.
	geolocation.clearWatch	• watchID 파라미터를 통해 단말기의 위치 변동 감지를 중단시킬 수 있다.. • navigator.geolocation.clearWatch(watchID); • Parameters - watchID: The id of the watchPosition interval to clear. (String)

❑ 디바이스의 GPS 센서에 의해 제공되는 geolocatino 객체에 접근하는 API이다.

유형	구분	설명
Arguments	geolocationSuccess	• 위치 정보를 성공적으로 가져왔을 경우, 수행하는 사용자 정의 함수. • Parameters - position: 단말기의 위치 정보. (Position)
	geolocationError	• geolocation 관련 함수 수행 시, 에러가 발생했을 경우 수행하는 함수. Parameters - error: 단말기의 에러 정보. (PositionError)
	geolocationOptions	• 위치 정보를 구할 경우, 선택적으로 설정할 수 있는 옵션. • { maximumAge: 3000, timeout: 5000, enableHighAccuracy: true }; • Options - frequency: milliseconds 단위로 특정 간격간에 위치 정보를 설정한다. W3C 스펙을 따르지 않고 있기 때문에, 앞으로 phoneGap에서 지원하지 않을 예정이다. (Number) (기본: 10000) - enableHighAccuracy: 어플리케이션이 가장 정확한 정보를 감지하도록 설정하는 옵션이다. (Boolean) - timeout: 위치 감지 요청의 제한시간이다. (geolocation.getCurrentPosition, geolocation.watchPosition 요청 후, geolocationSuccess이 호출될 때까지의 시간.) (Number) - maximumAge: milliseconds 단위로 위치 정보를 저장하고 있는 시간. (Number)

❑ 디바이스의 GPS 센서에 의해 제공되는 geolocatino 객체에 접근하는 API이다.

유형	구분	설명
Objects	Position	• geolocation API들을 통해 구해지는 위치 정보를 갖고 있는 객체. • Properties - coords: 위성 좌표. (Coordinates) - timestamp: milliseconds 단위로 위치 정보를 구하는 특정 시간 간격. (DOMTimeStamp)
	PositionError	• 에러가 발생하였을 때, 에러 정보를 가지는 객체. • Properties - code: 에러 코드는 아래와 같다.. - message: 에러 메시지를 나타낸다. • Constants - PositionError.PERMISSION_DENIED: 권한 부족. - PositionError.POSITION_UNAVAILABLE: 위치를 감지할 수 없음. - PositionError.TIMEOUT: 감지 시간 초과.
	Coordinates	• Position 객체의 coords 속성 객체이며, 위성 좌표의 정보를 가지고 있는 객체. • Properties - latitude: 위도. (Number) - longitude: 경도. (Number) - altitude: 타원체로부터 위치까지의 미터 단위 높이. (Number) - accuracy: 위도, 경도(미터 단위)의 정확도. (Number) - altitudeAccuracy: 고도(미터 단위)의 정확도. (Number) - heading: 진북 방향을 시계 방향의 각도로 나타낸 값. (Number) - speed: 초 단위의 단말기 속도(ground speed). (Number)

❑ 제약사항

유형	구분	지원플랫폼	설명
Objects	Coordinates	Android	- altitudeAccuracy: 단말기에서 지원하지 않는다. 항상 null 이다.
		iOS	-
	Position	Android	-
		iOS	- timestamp: milliseconds 단위 대신에 seconds 단위를 쓴다.
Arguments	geolocationOptions	Android	- frequency: milliseconds 단위로 위치 정보를 감지하는 특정 시간 간격. W3C 스펙을 따르지 않고 있기 때문에 곧 삭제될 예정이다. - iOS 2.x 버전의 가상기기에서는 enableHighAccuracy 을 'true' 로 설정해야만 위치정보를 가져올 수 있다.
		iOS	- frequency: milliseconds 단위로 위치 정보를 감지하는 특정 시간 간격. W3C 스펙을 따르지 않고 있기 때문에 곧 삭제될 예정이다.

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Device Properties Example</title>
    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    document.addEventListener("deviceready", onDeviceReady, false);
    function onDeviceReady() {
        navigator.geolocation.getCurrentPosition(onSuccess, onError);
    }

    function onSuccess(position) {
        var element = document.getElementById('geolocation');
        element.innerHTML = 'Latitude: '           + position.coords.latitude              + '<br />' +
                            'Longitude: '          + position.coords.longitude             + '<br />' +
                            'Altitude: '           + position.coords.altitude              + '<br />' +
                            'Accuracy: '           + position.coords.accuracy              + '<br />' +
                            'Altitude Accuracy: '  + position.coords.altitudeAccuracy      + '<br />' +
                            'Heading: '            + position.coords.heading               + '<br />' +
                            'Speed: '              + position.coords.speed                 + '<br />' +
                            'Timestamp: '          + position.timestamp                    + '<br />';
    }
    function onError(error) {
        alert('code: '    + error.code    + '₩n' +
              'message: ' + error.message + '₩n');
    }
    </script>
  </head>
  <body>
    <p id="geolocation">Finding geolocation...</p>
  </body>
</html>
```

❑ 디바이스의 오디오 파일을 재생 또는 녹음 할 수 있는 API이다.

유형	구분	설명
• var media = new Media(src, mediaSuccess, [mediaError], [mediaStatus]); • Note: 이 객체는 편리함 때문에 구현되었지만, W3C를 기반으로 구현된 것이 아니다. 앞으로 Media 객체는 다른 객체로 대체될 것이며, 새로 만들어질 객체는 W3C 기반으로 구현될 것이다.		
Parameters	src	• audio 컨텐츠를 포함하고 있는 URI. (DOMString)
	mediaSuccess	• (선택) play/record, stop이 성공적으로 수행되었을 경우, Media 객체를 가지고 호출되는 함수. (Function)
	mediaError	• (선택) 에러가 발생하였을 때, 호출되는 함수. (Function)
	mediaStatus	• (선택) 상태의 변화가 있을 경우, 호출되는 함수. (Function)
Methods	media.getCurrentPosition	• 오디오 파일의 현재 재생 위치 값을 구한다. • media.getCurrentPosition(mediaSuccess, [mediaError]); • Parameters - mediaSuccess: 초 단위의 현재 위치를 성공적으로 구하였을 때, 호출하는 함수. - mediaError: (선택) 현재 위치를 구하다가 실패했을 경우, 호출하는 함수.
	media.getDuration	• 오디오 파일의 재생 지연 시간. • media.getDuration();
	media.pause	• 오디오 파일의 재생을 멈춘다. • media.pause();
	media.play	• 오디오 파일 재생을 시작하거나, 재실행한다. • media.play();
	media.release	• 운영체제(OS) 혹은 메모리 내의 Media 객체에 등록된 오디오 파일을 해제한다. • media.release();

❑ 디바이스의 오디오 파일을 재생 또는 녹음 할 수 있는 API이다.

유형	구분	설명
Methods	media.seekTo	• 오디오 파일의 재생 위치를 설정한다. • media.seekTo(milliseconds); • Parameters - milliseconds: milliseconds 단위로 오디오 파일의 재생 위치를 설정한다.
	media.startRecord	• 오디오 파일의 녹음을 시작한다. • media.startRecord();
	media.stop	• 오디오 파일의 재생을 멈춘다. • media.stop();
	media.stopRecord	•오디오 파일의 녹음을 멈춘다. •media.stopRecord();
Additional ReadOnly Parameters	_position	• 현재 오디오 파일의 재생 위치(초 단위). • 하지만 자동적으로 업데이트 되지 않기 때문에, getCurrentPosition 함수를 통해 업데이트한 후 호출해야 현재 재생 위치를 구할 수 있다.
	_duration	• 미디어 객체의 재생 지연 시간(초 단위).
Object	MediaError	• 에러가 발생했을 경우, 호출되는 함수에서 파라미터로 쓰이는 에러 정보 객체. • Properties - code: 에러 코드는 아래와 같은 코드를 갖는다.. - message: 에러 메시지를 나타낸다. • Constants - MediaError.MEDIA_ERR_ABORTED - MediaError.MEDIA_ERR_NETWORK - MediaError.MEDIA_ERR_DECODE - MediaError.MEDIA_ERR_NONE_SUPPORTED

❑ 제약사항

유형	구분	지원플랫폼	설명
Methods	media.startRecord	Android	-
		iOS	- 녹음할 파일이 존재하거나, .wav 파일을 미리 생성해야 한다. File API는 파일을 생성해주지 않는다.

Sample Code

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
   <html>
     <head><title>Media Example</title>
       <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
       <script type="text/javascript" charset="utf-8">
       document.addEventListener("deviceready", onDeviceReady, false);
       function onDeviceReady() {
           playAudio("http://audio.ibeat.org/content/p1rj1s/p1rj1s_-_rockGuitar.mp3");
       }
       var my_media = null;
       var mediaTimer = null;
       function playAudio(src) {

           my_media = new Media(src, onSuccess, onError);
           my_media.play();

           if (mediaTimer == null) {
               mediaTimer = setInterval(function() {
                   my_media.getCurrentPosition(
                       function(position) {
                           if (position > -1) {
                               setAudioPosition((position) + " sec");
                           }
                       },
                       // error callback
                       function(e) {
                           console.log("Error getting pos=" + e);
                           setAudioPosition("Error: " + e);
                       }
                   );
               }, 1000);
           }
       }
```

Sample Code

```
    function pauseAudio() {
        if (my_media) {
            my_media.pause();
        }
    }
    function stopAudio() {
        if (my_media) {
            my_media.stop();
        }
        clearInterval(mediaTimer);
        mediaTimer = null;
    }
    function onSuccess() {
        console.log("playAudio():Audio Success");
    }
    function onError(error) {
        alert('code: '    + error.code    + '₩n' +
              'message: ' + error.message + '₩n');
    }

    function setAudioPosition(position) {
        document.getElementById('audio_position').innerHTML = position;
    }
    </script>
  </head>
  <body>
    <a href="#" class="btn large" onclick="playAudio('http://audio.ibeat.org/content/p1rj1s/p1rj1s_-_rockGuitar.mp3');">Play Audio</a>
    <a href="#" class="btn large" onclick="pauseAudio();">Pause Playing Audio</a>
    <a href="#" class="btn large" onclick="stopAudio();">Stop Playing Audio</a>
    <p id="audio_position"></p>
  </body>
</html>
```

❑ 진동 및 소리를 통한 알림 관련 API이다.

유형	구분	설명
Methods	notification.alert	• 설정된 경고창(alert) 나 dialog box을 보여준다. • navigator.notification.alert(message, alertCallback, [title], [buttonName]) - message: 메세지의 내용. (String) - alertCallback: 경고창이나 대화상자가 사라질 때, 호출하는 함수.(Function) - title: 경고창이나 대화상자의 제목. (String) (선택, 기본: "Alert") - buttonName: 경고창이나 대화상자에 보여지게 되는 버튼 이름. (String) (선택, 기본: "OK")
	notification.confirm	• 실행 여부를 선택하는 확인형 대화상자. • navigator.notification.confirm(message, confirmCallback, [title], [buttonLabels]) - message: 메세지의 내용. (String) - confirmCallback: – 버튼들 중 하나를 클릭했을 때, 호출되는 함수. 클릭한 번호를 전달받는다. (1, 2 또는 3). (Number) - title: 경고창이나 대화창의 제목. (String) (선택, 기본: "Confirm") - buttonLabels: 경고창이나 대화상자에 보여지게 되는 버튼 이름. 버튼들은‘,’ 로 구분한다. (String) (선택, 기본: "OK,Cancel")
	notification.beep	• 단말기의 비프(beep)음을 재생한다. • navigator.notification.beep(times); - times: 반복되는 비프(beep)음 재생 횟수. (Number)
	notification.vibrate	• 지정한 시간 동안 단말기의 진동을 울리게 한다. • navigator.notification.vibrate(milliseconds) - time: Milliseconds 단위의 진동이 울리는 시간. (Number)

❑ 제약사항

유형	구분	지원플랫폼	설명
Methods	notification.beep	Android	- 환경 설정 > 소리 > 알림음 설정에서 지정한 알림을 재생.
		iOS	• 비프(beep)음 설정을 무시한다. • iPhone에서는 beep API를 지원하지 않는다. -PhoneGap에서는 media API를 통한 오디오 파일 재생으로 비프(beep)를 지원한다. -사용자는 비프(beep)음의 파일을 저장해야 한다.. -파일은 30초를 넘지 않아야 하며, www/root 밑에 'beep.wav' 이름으로 저장해야 한다.
	notification.vibrate	Android	-
		iOS	- time: 미리 설정한 진동 시간을 무시하고 아이폰 설정을 따른다.

❑ 진동 및 소리를 통한 알림 관련 API이다.

유형	구분	설명
Methods	notification.alert	• 설정된 경고창(alert) 나 dialog box을 보여준다. • navigator.notification.alert(message, alertCallback, [title], [buttonName]) - message: 메세지의 내용. (String) - alertCallback: 경고창이나 대화상자가 사라질 때, 호출하는 함수.(Function) - title: 경고창이나 대화상자의 제목. (String) (선택, 기본: "Alert") - buttonName: 경고창이나 대화상자에 보여지게 되는 버튼 이름. (String) (선택, 기본: "OK")
	notification.confirm	• 실행 여부를 선택하는 확인형 대화상자. • navigator.notification.confirm(message, confirmCallback, [title], [buttonLabels]) - message: 메세지의 내용. (String) - confirmCallback: – 버튼들 중 하나를 클릭했을 때, 호출되는 함수. 클릭한 번호를 전달받는다. (1, 2 또는 3). (Number) - title: 경고창이나 대화창의 제목. (String) (선택, 기본: "Confirm") - buttonLabels: 경고창이나 대화상자에 보여지게 되는 버튼 이름. 버튼들은‘,’ 로 구분한다. (String) (선택, 기본: "OK,Cancel")
	notification.beep	• 단말기의 비프(beep)음을 재생한다. • navigator.notification.beep(times); - times: 반복되는 비프(beep)음 재생 횟수. (Number)
	notification.vibrate	• 지정한 시간 동안 단말기의 진동을 울리게 한다. • navigator.notification.vibrate(milliseconds) - time: Milliseconds 단위의 진동이 울리는 시간. (Number)

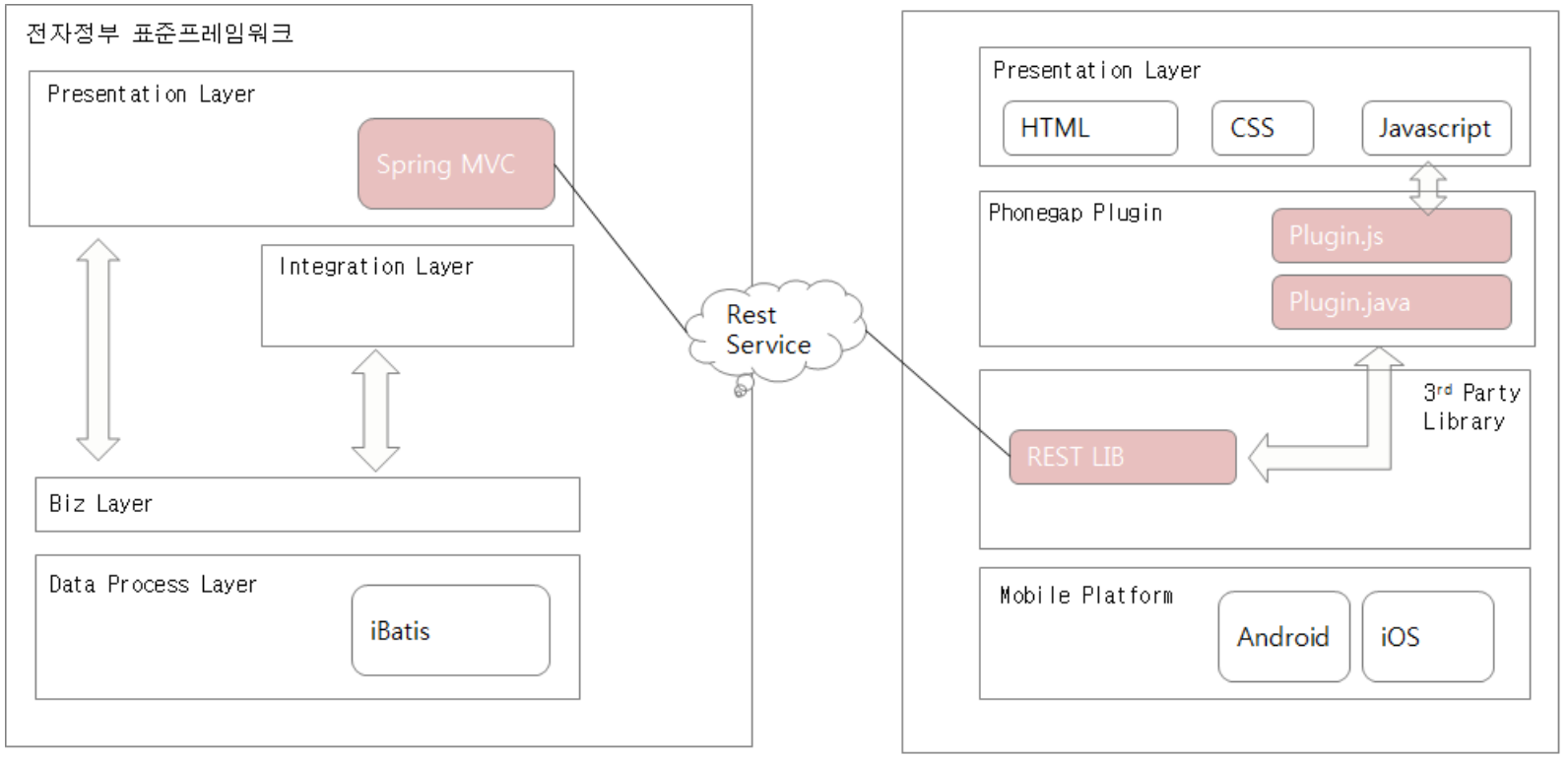
❑ 제약사항

유형	구분	지원플랫폼	설명
Methods	notification.beep	Android	- 환경 설정 > 소리 > 알림음 설정에서 지정한 알림을 재생.
		iOS	• 비프(beep)음 설정을 무시한다. • iPhone에서는 beep API를 지원하지 않는다. -PhoneGap에서는 media API를 통한 오디오 파일 재생으로 비프(beep)를 지원한다. -사용자는 비프(beep)음의 파일을 저장해야 한다.. -파일은 30초를 넘지 않아야 하며, www/root 밑에 'beep.wav' 이름으로 저장해야 한다.
	notification.vibrate	Android	-
		iOS	- time: 미리 설정한 진동 시간을 무시하고 아이폰 설정을 따른다.

Sample Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Notification Example</title>

    <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    document.addEventListener("deviceready", onDeviceReady, false);

    function onDeviceReady() {
        // Empty
    }

    function alertDismissed() {
        // do something
    }

    function showAlert() {
        navigator.notification.alert(
            'You are the winner!',  // message
            alertDismissed,         // callback
            'Game Over',            // title
            'Done'                  // buttonName
        );
    }

    </script>
  </head>
  <body>
    <p><a href="#" onclick="showAlert(); return false;">Show Alert</a></p>
  </body>
</html>
```

❑ 전자정부 표준 프레임워크 서버 어플리케이션과 모바일 하이브리드 앱은 Loosely Coupling 되어 있으며, Restful 방식으로 웹서비스를 통해 통신한다.



① 전자정부 표준 프레임워크 기반의 서버 어플리케이션과 모바일 클라이언트는 웹서비스를 통해 Loosely Coupling 된다.

② 웹서비스는 REST Service 프로토콜을 이용하여 클라이언트와 통신한다.

③ 웹서비스를 이용하는 모바일 클라이언트는 전자정부 표준프레임워크 REST 서비스 뿐만 아니라, 표준 REST 서비스를 구현한 서버와의 통신이 가능하다.

- 특정 서버와 HTTP/HTTPS 통신을 하는 static library이다.
- GET, POST, PUT, DELETE로 특정서버와 통신 파일 upload / download 가 가능하다.

Spring for Android

Spring for Android 는 안드로이드 어플리케이션 개발을 단순화 하기 위한 Spring framework 의 확장기능 이다.

① REST Client 로써 Java 기반의 RestTemplate 사용을 포함
② 트위터, 페이스북 같은 소셜 네트워크에 대한 인증을 지원
③ JSON, XML, RSS, Atom 에 대한 third-party 라이브러리를 사용할 수 있음
④ 최근 1.0 버전으로 Release 되었음
⑤ Maven 지원

Function List

구분	설명
window.plugins.EgovInterface.get	- HTTP GET Method를 수행한다.
window.plugins.EgovInterface.post	- HTTP POST Method를 수행한다.
window.plugins.EgovInterface.geturl	- 환경설정에서 설정한 SERVER_URL을 얻어온다.

```
window.plugins.EgovInterface.get(url,accept_type, null, function(jsondata) {
var data = JSON.parse(jsondata);
                    ...
});

window.plugins.EgovInterface.post(url, accept_type, null, function(jsondata) {
var data = JSON.parse(jsondata);
                    ...
});

window.plugins.EgovInterface.geturl(function(url){
                    ...
});
```

❑ Sample Code

```
window.plugins.EgovInterface.get(url,accept_type, null, function(jsondata) {
		var data = JSON.parse(jsondata);
		var list_html = "";
		var totcnt = data.networkInfoList.length;
		for (var i = 0; i < totcnt; i++) {
			result = data.networkInfoList[i];
			list_html += "<li><h3>UUID : " + result.uuid + "</h3>";
			list_html += "<p><strong>Network Connection Type : " + result.networktype + "</strong></p>";
			list_html += "<p>Availibility : " + result.useYn + "</p></li>";

		}
		var theList = $('#theList');
		theList.html(list_html);

		$.mobile.changePage("#networkInfoList", "slide", false, false);
		theList.listview("refresh");
		setTimeout(loadiScroll, 200);
});
```

```
var uri = "/nwk/deleteNetworkInfo.do";
//Accept_Type setting
var accept_type = "json";
// http post method call
egovHyb.post(url, accept_type, null, function(jsondata) {
		var data = JSON.parse(jsondata);
		if(data.resultState == "OK"){
			$.mobile.changePage("#networkInfo", { transition: "slide", reverse: true });
		}else{
			$("#alert_dialog").click( function() {
				jAlert('데이터 삭제 중 오류가 발생 했습니다.', '삭제 오류', 'c');
			});
		}
});
```

- 특정 서버와 HTTP/HTTPS 통신을 하는 static library이다.
- GET, POST, PUT, DELETE로 특정서버와 통신 파일 upload / download 가 가능하다.

추가 파일

① EgovInterface Framework
- CFNetwork.framework
- MobileCoreServices.framework
- SystemConfiguration.framework
- Security.framework

② Libraries

EgovInterface API를 연결하는 header 파일
- EgovComModule.h
- libEgovComModule.a

③ EgovInterface

해당 header 파일과 구현부 파일이 있어야만 EgovInterface API와 연동
- EgovInterface.h
- EgovInterface.m

Function List

구분	설명
initWithURL:delegate	- 초기화와 기본적인 설정을 한다.
setURL	- 통신할 곳의 URL을 설정한다.
setTimeOutSecondsl	- time out 초를 설정한다. Default로 10초로 설정되어 있다.
addPost:key	- 입력하고자 하는 값을 key값의 request 파라미터에 설정한다.
startAsynchronous	- 설정하고 입력한 값으로 비동기 통신을 시작한다.

구분	설명
onNetworkStarted	- 통신이 시작된 후 일어나는 이벤트
onNetworkFailed	- 통신이 실패한 후 일어나는 이벤트
onNetworkFinished :responseString: responseStatusCode:	- 통신이 성공한 후 일어나는 이벤트

❑ Sample Code

```
//생성방법
EGovComModule *m_module = [[[EGovComModule alloc] initWithURL:[NSString stringWithFormat:@"%@/index.do",kSERVER_URL]
delegate:self] autorelease];

//wi-fi 및 네트워크 체크
    if (m_module.isWifi) {
        NSLog(@"wi-fi");
    } else {
        if (m_module.isNetworkConnected) {
            NSLog(@"3G/4G");
        } else {
            NSLog(@"Network Disconnected");
        }
    }
//통신 시작
    [m_module startAsynchronous];

//key value로 값을 추가하고자 할 경우
    [m_module addPost:@"값" key:@"key"];
//통신 시작 이벤트
- (void)onNetworkStarted
{
}
//통신 실패 이벤트
- (void)onNetworkFailed:(NSError*)error
{
}
//통신 완료 이벤트
- (void)onNetworkFinished:(NSData*)responseData responseString:(NSString*)responseString
responseStatusCode:(NSInteger)responseStatusCode
{
}
```

➘ 네이티브 Function을 호출 하기위해 cordova.exe 자바스크립트를 이용한다.

```
exec(<successFunction>, <failFunction>, <service>, <action>, [<args>]);
```

구분	설명
successFunction	- iOS 네이티브 함수가 정상 호출 되었을 때, 호출되는 자바스크립트 콜백 함수
failFunction	- iOS 네이티브 함수의 호출 결과 에러가 발생했을 때, 호출 되는 자바스크립트 콜백 함수
service	- 플러그인 설정파일인 plugin.xml에서 설정된 플러그인의 Name
action	- 서비스 구분을 위해 iOS 네이티브 함수로 전달되는 파라미터
args	- iOS 네이티브 함수 호출에 필요한 파라미터 배열

➘ Res/xml/plugin.xml

```
<plugin name="<service_name>" value="<full_name_including_namespace>"/>
```

구분	설명
name	- cordova.exe 함수에 전달되는 플러그인 서비스 Name
value	- cordova.exe 함수에 의해 호출되는 iOS 네이티브 클래스

- 플러그인 작성을 위해서는 폰갭 프레임워크에서 제공하는 Plugin 클래스를 상속받아야 한다.
- cordova.exe에 의해 호출되는 메서드인 excute 메서드를 작성해야 하며 PluginResult클래스를 반환한다.

```
public class Echo extends CordovaPlugin {
    @Override
    public boolean execute(String action, JSONArray args, CallbackContext callbackContext) throws JSONException
{
        if (action.equals("echo")) {
            String message = args.getString(0);
            this.echo(message, callbackContext);
            return true;
        }
        return false;
    }

    private void echo(String message, CallbackContext callbackContext) {
        if (message != null && message.length() > 0) {
            callbackContext.success(message);
        } else {
            callbackContext.error("Expected one non-empty string argument.");
        }
    }
}
```

구분	설명
action	- 플러그인 서비스 구분을 위해 사용되는 Action명
args	- 플러그인 호출 자바스크립트에서 넘겨주는 파라미터들
callbackId	- 플러그인 실행 완료 후 호출되는 자바스크립트 콜백 Function

➘ 네이티브 Function을 호출 하기위해 cordova.exe 자바스크립트를 이용한다.

```
exec(<successFunction>, <failFunction>, <service>, <action>, [<args>]);
```

구분	설명
successFunction	- iOS 네이티브 함수가 정상 호출 되었을 때, 호출되는 자바스크립트 콜백 함수
failFunction	- iOS 네이티브 함수의 호출 결과 에러가 발생했을 때, 호출 되는 자바스크립트 콜백 함수
service	- 플러그인 설정파일인 plugin.xml에서 설정된 플러그인의 Name
action	- 서비스 구분을 위해 iOS 네이티브 함수로 전달되는 파라미터
args	- iOS 네이티브 함수 호출에 필요한 파라미터 배열

➘ Cordova.plist

```
<key>service_name</key>
<string>PluginClassName</string>
```

구분	설명
service_name	- cordova.exe 함수에 전달되는 플러그인 서비스 Name
PluginClassName	- cordova.exe 함수에 의해 호출되는 iOS 네이티브 클래스

➘ 플러그인 작성을 위해서는 폰갭 프레임워크에서 제공하는 CDVPlugin 클래스를 상속받아야 한다.

```
#import <Cordova/CDVPlugin.h>

@interface Echo : CDVPlugin

- (void) echo:(NSMutableArray*)arguments withDict:(NSMutableDictionary*)options;

@end

/******** Echo.m Cordova Plugin Implementation *******/

#import "Echo.h"
#import <Cordova/CDVPluginResult.h>

@implementation Echo

- (void) echo:(NSMutableArray*)arguments withDict:(NSMutableDictionary*)options
{
    NSString* callbackId = [arguments objectAtIndex:0];

    CDVPluginResult* pluginResult = nil;
    NSString* javaScript = nil;

    @try {
        NSString* echo = [arguments objectAtIndex:1];

        if (echo != nil && [echo length] > 0) {
            pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_OK messageAsString:echo];
            javaScript = [pluginResult toSuccessCallbackString:callbackId];
        } else {
            pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_ERROR];
            javaScript = [pluginResult toErrorCallbackString:callbackId];
        }
    } @catch (NSException* exception) {
        pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_JSON_EXCEPTION messageAsString:[exception reason]];
        javaScript = [pluginResult toErrorCallbackString:callbackId];
    }

    [self writeJavascript:javaScript];
}

@end
```

구분	설명
arguments	- 플러그인 호출 자바스크립트에서 넘겨주는 파라미터
options	- 플러그인 호출 자바스크립트에서 넘겨주는 파라미터 중 마지막 배열 값