안드로이드 둘러보기

시작하면서

- **목차**
  - 스마트폰과 무선 인터넷
  - 모바일 플랫폼
  - 모바일 시장은 전쟁터
  - 안드로이드 개요
  - 안드로이드 아키텍쳐
  - 안드로이드 애플리케이션
  - 안드로이드 미래

개요

스마트폰

스마트폰의 의미

- 손 안의 PC 혹은 스마트폰은 모바일 인터넷 단말기
- 과거 일부 비즈니스 계층과 얼리 어댑터의 전유물에서 최근 일반 대중의 생활로 빠르게 확산
- 세계 최초의 스마트폰은 1992년 COMDEX에서 IBM이 발표한 것으로 모바일폰, 페이저, PDA와 팩스를 결합한 Simon이란 단말기
- 2008년도 애플이 혁신적인 사용자 인터페이스를 제공하는 아이폰을 출시하면서 급속히 확산
- 국내에서도 WIPI의 휴대폰 의무 탑재를 폐지한 2008년도부터 애플 아이폰, 모토롤라의 모토로이, 삼성전자의 갤럭시 등 출시
- 초기와는 달리 네트워크, 단말기, 컨텐츠 측면에서 스마트폰을 위한 환경이 조성 → 손쉽게 인터넷 사용
- 특히 사용자의 기호를 반영한 양질의 컨텐츠가 다수 개발

스마트폰

◉ 세계 스마트폰 시장 전망

구분		'07년	'08년	'09년	'10년	'11년	'12년	'13년
휴대폰 판매수		1,151	1,209	1,114	1,202	1,306	1,432	1,568
스마트폰	판매수	121	143	178	254	351	469	604
	성장률	49	18	24	43	38	34	29
	비중	10.5	11.8	15.9	21.1	26.9	32.8	38.5

출처: 삼성경제연구소 CEO Information 741호 **(단위: 백만대, %)**

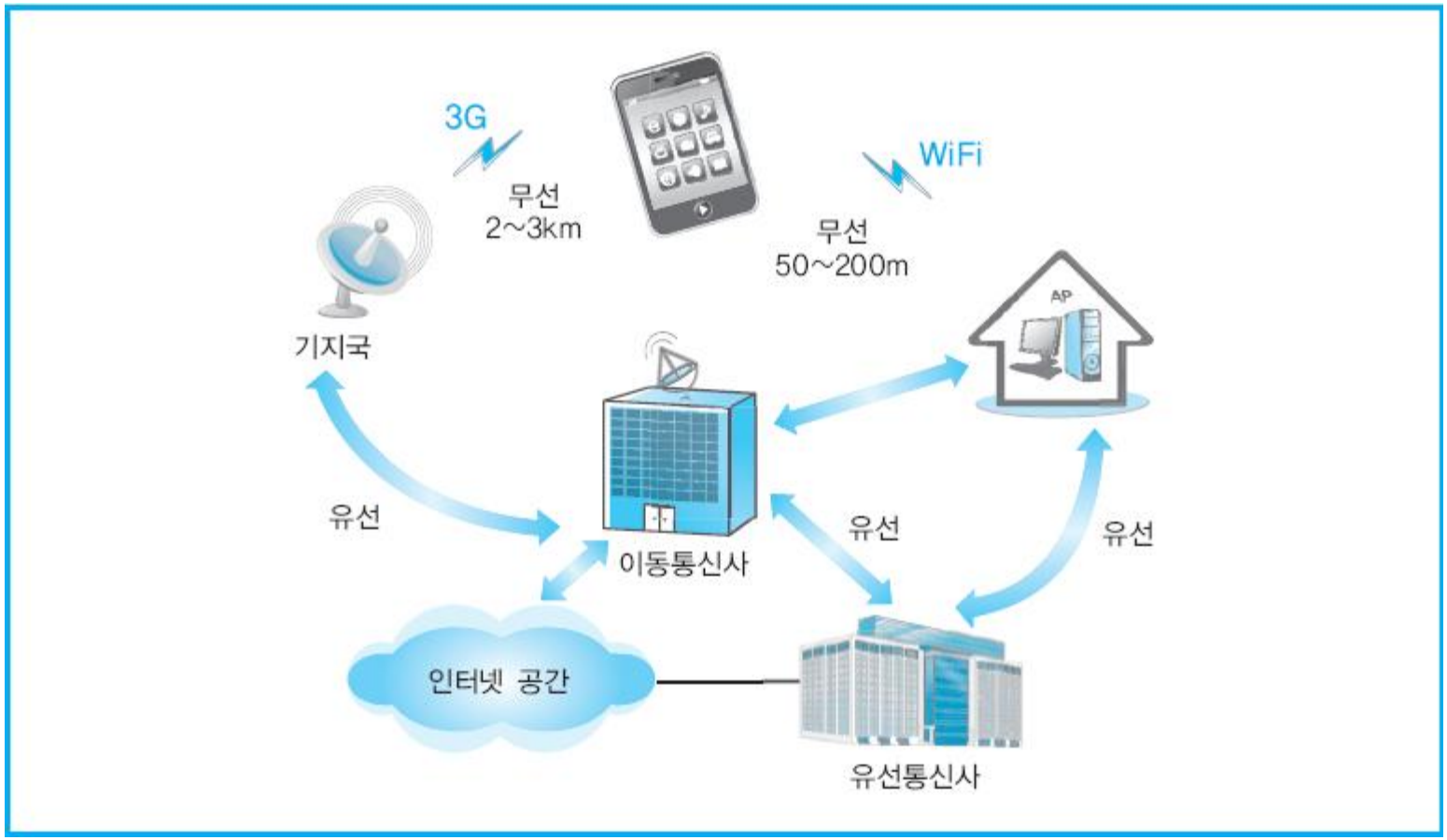
무선 인터넷

- **무선 인터넷의 의의**
  - 유선 케이블 대신에 전파를 사용하며 이동 중에 무선 통신을 기반으로 수행하는 인터넷을 통칭
  - 전파는 물리적 특성상 지역적·국가적 경계선이 없고 제한된 자원
  - 전파는 유선 케이블 없이 통신을 가능하게 하는 편의성을 제공하기 때문에 정보통신 분야에서 주목 대상

- **스마트폰 대중화의 핵심 모바일 인프라**
  - 전파를 이용한 무선통신과 정보기술의 발전
  - 특히 3G 이동통신과 와이파이 → 모바일 빅뱅

무선 인터넷

- 와이파이와 3G 이동통신



모바일 플랫폼

개요

- **모바일 플랫폼이란?**
  - 무선 인터넷 서비스를 제공하기 위하여 PC의 운영체제와 담당하는 미들웨어에 해당하는 기본 소프트웨어
  - 모바일 플랫폼을 도입함으로써 텍스트 위주의 서비스에서 멀티미디어 서비스 제공
- **모바일 플랫폼 방식**

초기 모바일 플랫폼

- **기능**
  - 단문 메시지와 텍스트 기반의 컨텐츠, 저속 및 고비용, 불안전한 구동 기술
- **종류 및 특징**

독점 모바일 플랫폼

춘추 전국 시대

회사	플랫폼	개발 언어	개발 업체	수행 방식
LGT	KVM	Java	SUN	VM
	키티호크	Java	아로마소프트	VM
SKT	SK-VM	Java	XCE	VM
	GVM	C/C++	신지소프트	VM
	WITOP	Java, C/C++	SKT	VM
KTF	MAP	C/C++	Mobiletop	Native
	BREW	C/C++	Qualcomm	Native

- 독점 플랫폼으로 인한 문제점

WIPI

개요

- Wireless Internet Platform for Interoperability
- 정통부에서 Java와 C/C++ 언어를 모두 포함하는 한국형 무선 인터넷 표준 플랫폼
- 비동기 IMT-2000으로 국제 표준으로 상정
- 2005년 4월 1일부터 국내 휴대폰에 탑재 의무화
- 2009년 4월 1일부터 국내 휴대폰에 탑재 자율화 → 모바일 컨텐츠 산업의 활성화 필요

모바일 시장은 전쟁터

모바일 생태계

- **전통적 가치사슬**
  - 구성원
    - 컨텐츠, 플랫폼, 네트워크, 단말기 관련 회사
    - 이통사들이 네트워크와 플랫폼을 장악
  - 컨텐츠 유통 경로
    - CP의 컨텐츠 개발 → 이통사의 검수 → CP가 이통사 서버에 컨텐츠 업로드 → 소비자에게 판매
    - CP가 개발한 컨텐츠가 이통사의 검수과정을 통과하지 못하면?
  - 네트워크와 주파수를 보유한 사업자가 가치사슬의 주도권 확보
    - 네트워크는 막대한 초기 투자 비용 요구, 그러나 가입자당 한계비용은 거의 없음
    - 주파수는 제한된 자원으로 대부분 정부 차원에서 관리
  - 이통사가 통신 서비스뿐만 아니라 단말기 제조와 유통, 단말기 OS와 미들웨어, 컨텐츠와 애플리케이션에 막대한 영향

모바일 생태계

- **생태계 변화 배경**
  - 다양한 멀티미디어 서비스 채택 강화
  - 멀티미디어 컨텐츠 사용 증가
  - 단말기 개발 기간 단축 및 고성능화: 일반폰(Feature Phone) → 스마트폰
    - 고기능성 운영체제 확산
    - Full browsing 등
  - 다양한 애플리케이션 및 솔루션에 대한 사용 요구 증가
  - 데이터 처리 속도 및 메모리 용량의 급격한 증가
  - 무선 네트워크 속도 증가
  - 무선 네트워크 사용료 하락 및 Wi-Fi 등과 같은 우회망 확보
  - 관련 소프트웨어 재사용 용이
  - 휴대폰과 모바일 서비스 연계 비즈니스 모델 등장

모바일 시장 변화

◉ 모바일 시장의 주도권

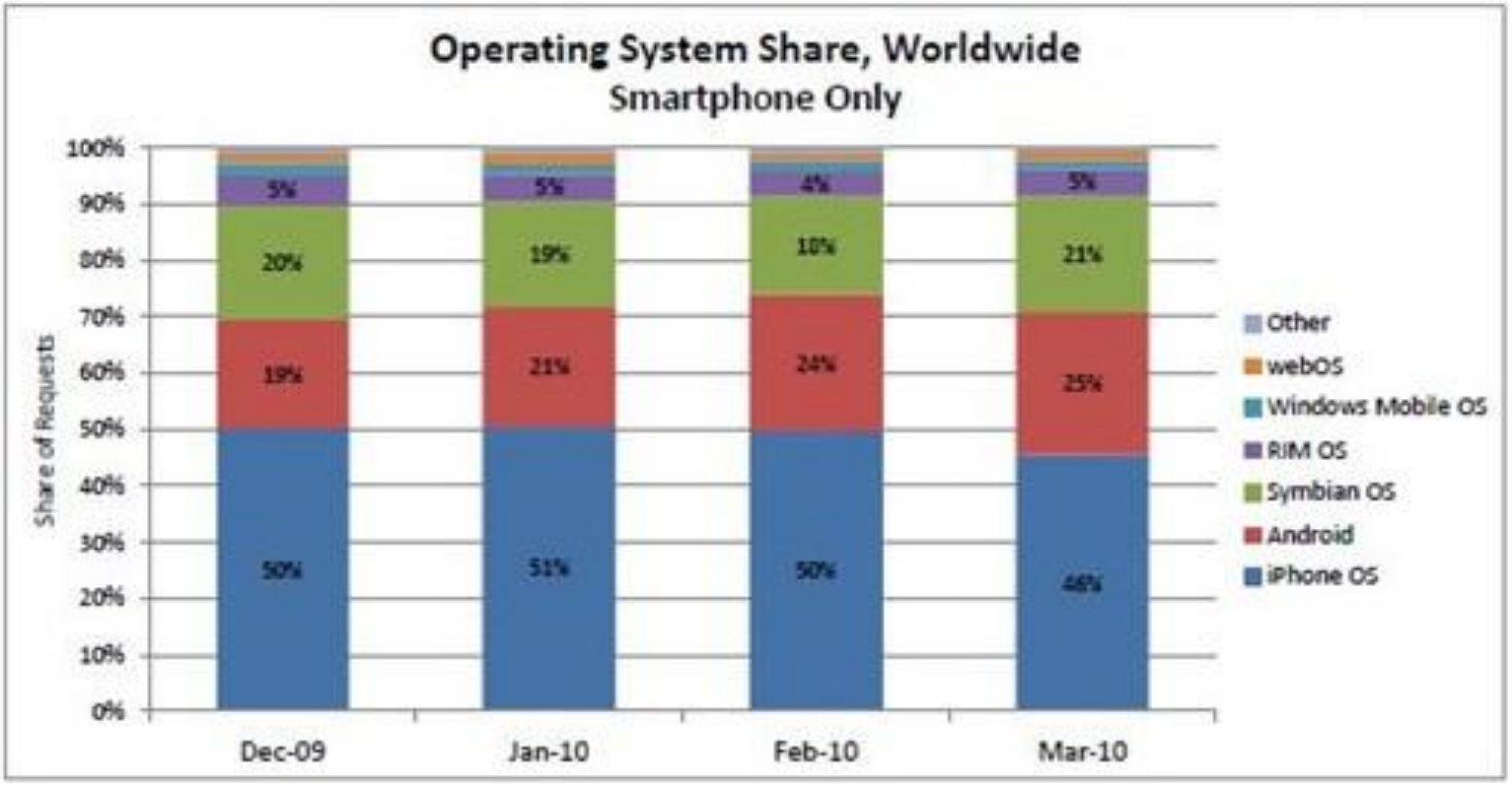
- 모바일 시장의 질서를 주도하는 힘의 중심은 네트워크에서 모바일 서비스 플랫폼으로 이동

◉ 모바일 시장 산업 구조

분 류	과 거	현 재
휴대폰	노키아, 삼성 전자, 모토 로라, LG 전자, 소니 에릭슨	기존의 5대 휴대폰 업체 + 스마트폰 3사 (Apple, RIM, HTC)
플랫폼	심비안, 윈도우 모바일, RIM OS, Linux	기존 OS (심비안 오픈화) + 신규 OS (구글의 안드로이드 등)
컨텐츠	● 이동통신사 중심의 포털 서비스	이동 통신사 + 하드 웨어 업체 + 인터넷 업체

모바일 시장 변화

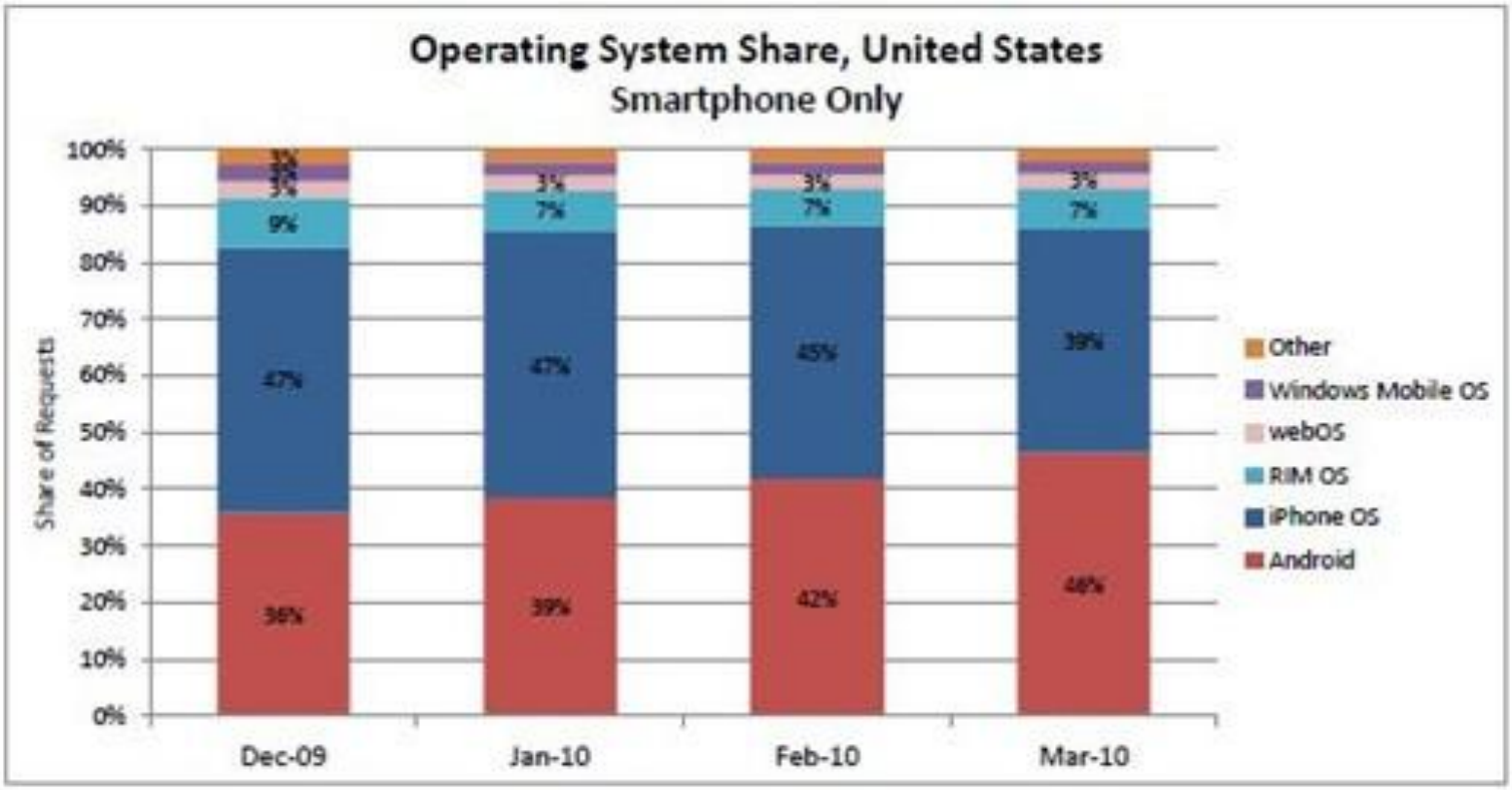
- 세계 모바일 플랫폼 점유율



(출처: 애드몹 보고서, 2010. 3)

모바일 시장 변화

- 북미 모바일 플랫폼 점유율



(출처: 애드몹 보고서, 2010. 3)

컨텐츠 장터

업체별 스마트폰 플랫폼과 컨텐츠 장터

업체	플랫폼	개방/폐쇄	컨텐츠 장터
애플	아이폰 OS	폐쇄형	애플 앱스토어
구글	안드로이드	개방형	안드로이드마켓
RIM	블랙베리	폐쇄형	앱월드
노키아	심비안	개방형	오비 스토어
MS	윈도모바일	개방형	윈도 마켓 플레이스
삼성	바다	개방형	삼성 애플리케이션 스토어

컨텐츠 장터

- **AppStore와 Android Market**

	AppStore	Android Market
서비스 시기	2008. 7. 11	2008. 8.28
컨텐츠 등록	Apple의 통제	자율
개발도구 사용	등록자만 가능	누구나 가능
수익 분배	CP와 Apple이 7:3	CP와 이통사(혹은 솔루션업체)가 7:3
VoIP	이통사와의 관계로 불허	허용
단말기 출시	2007. 6. 29	2008. 10. 22
단말기 제조	Apple	모든 단말기 제조 업체

컨텐츠 장터

- **WAC**
  - Wholesale App. Community
  - 2011년 초 구축 예정
  - 세계 24개 통신 및 단말 제조 업체가 공동 참여하는 개방형 컨텐츠 거래 장터, 즉 수퍼 앱스토어단말기 제조사: 삼성전자, LG전자, 소니에릭슨, 노키아(추후 참여)
  - 참여사들의 운영 플랫폼이 다른 상황에서 표준화 유도의 어려움

PC와 스마트폰 시장

- **PC 시장**
  - 80년대 초 PC 시장의 최강자는 애플
  - 애플은 개인용 컴퓨터 시대를 만든 주역이자 가능성을 확인
  - 애플은 폐쇄형 시스템IBM은 오픈 아키텍쳐
  - 애플의 몰락
    - 혁신적인 UI 운영체제 및 우수한 디자인
    - 고가 및 많은 적군
    - 외부 업체의 참여와 상생에 소홀

PC와 스마트폰 시장

- **스마트폰 시장**
  - 애플이 스마트폰의 선두주자는 아님
  - 애플
    - 폐쇄적 모형
    - 이통사도 애플이 OK 해야 아이폰을 공급 → 다수의 잠재적 적군 (MWC 2010의 WAC이 증거)
    - 외부 SW업체 및 개발자들은 아이폰에 열광
  - 구글이나 MS
    - 다양한 단말기 제조업체, 이통사가 자사 플랫폼을 탑재한 스마트폰 제조
    - 컨텐츠 거래 장터도 애플보다는 유연하며 이통사와의 공존을 강조
  - 역사의 반복? 그러면 애플은 왜?
    - 풍부한 컨텐츠
    - iPod와 iTunes 성공 경험

안드로이드 개요

개요

- **안드로이드 의미**
  - 인간의 모습을 한 로봇
  - 안드로이드는 검색 광고 업계의 거인 구글이 모바일 시장에 진출하기 위하여 인수한 조그마한 신생 업체의 이름에 불과
  - 그러나 이제 안드로이드가 아이폰과 함께 현재 스마트폰을 이끌어 나갈 주요 단어

- **Google?**

안드로이드 플랫폼

개요

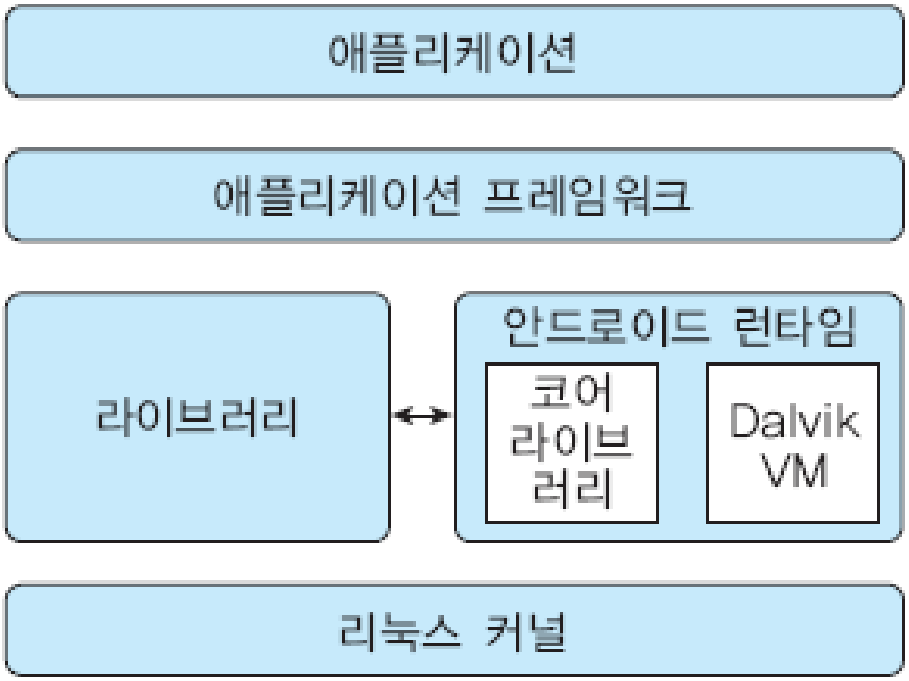
- Open, Complete. Free한 플랫폼
- 운영체제, 미들웨어, 자바로 개발된 핵심 애플리케이션(key application)을 포함하는 모바일 기기의 소프트웨어 집합체
- 단말기에서 하드웨어를 제외한 나머지 모든 소프트웨어 계층
- 개방형 플랫폼으로 소스 코드를 완전 개방 → 제한 없이 안드로이드 기반의 모바일 기기 제작 가능
- 2007년 11월 구글과 OHA(Open Handset Alliance)라는 모임에 의해 발표한 모바일 플랫폼
- 아파치 2.0 라이센스를 가지며 소스 코드로 모든 사람이 빌드 가능
- 기본적으로 JAVA 프로그래밍 언어를 사용하며 어플리케이션의 빌드, 컴파일, Test 및 디버그를 할 수 있는 SDK를 제공

안드로이드 플랫폼

- **특징**
  - 애플리케이션 프레임워크: 컴포넌트의 재사용과 대체를 가능케 함
  - Dalvik 가상머신: 모바일 디바이스에 최적화
  - 통합(Integrated) 브라우저: 오픈 소스 Webkit 엔진 기반
  - 최적화된 그래픽: 구글이 만든 2D 그래픽 라이브러리 사용
  - OpenGL ES 1.0 스펙에 기반한 3D 그래픽 사용(하드웨어 가속은 선택 사항)
  - SQLite: 정형화된 데이터 저장공간을 위함
  - 미디어 지원: 일반적 포맷(MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)을 지원
  - GSM Telephony, Bluetooth, EDGE, 3G, WiFi, 카메라, GPS, 나침반, 가속도계(하드웨어 의존적)
  - 풍부한 개발 환경: 디바이스 에뮬레이터, 디버깅 도구, 메모리 및 성능 프로파일링, Eclipse IDE를 위한 플러그인

안드로이드 아키텍처

- 구조



안드로이드 아키텍처

- **애플리케이션**
  - Gmail과 연동되는 이메일 클라이언트
  - SMS 문자 메시지 프로그램
  - 캘린더
  - 구글 맵 애플리케이션
  - Webkit 기반 웹 브라우저
  - 컨택트(contact)

  - 네이티브 애플리케이션과 서드파티 애플리케이션이 하드웨어 접근 혹은 실행 권한 등 모든 면에서 동등
    - 동일한 API 사용
    - 동일한 런타임에서 실행

안드로이드 아키텍처

- **애플리케이션 프레임워크**
  - 자바 기반의 애플리케이션 프레임워크. 대부분 JNI(Java Native Interface)를 통해 native C/C++ 코드로 작성

안드로이드 아키텍처

- **라이브러리**
  - Binoic Libc는 임베디드 디바이스를 위한 최적화된 libc
  - Function Library
    - 웹 브라우저를 위한 Webkit
    - PacketVideo의 OpenCORE 플랫폼 기반의 미디어 프레임워크
    - 가벼운 데이터 베이스인 SQLite
  - Native Server
    - Surface Flinger
    - Audio Flinger
  - Hardware Abstraction Library
    - 사용자 공간의 C/C++ 라이브러리 계층

안드로이드 아키텍처

- **안드로이드 런타임**
  - 여러개의 VM을 동시에 실행시키도록 디자인되어 있으며 프로세스 독립과 메모리 관리 쓰레드를 OS에 의존
  - 모든 안드로이드 애플리케이션은 Dalvik 가상머신 내에서 자신의 인스턴스를 가지고 자신의 프로세스내에서 동작

안드로이드 아키텍처

- **안드로이드 런타임**
  - 달빅 VM
    - 성능의 최적화를 위하여 자체적으로 달빅(Dalvik)이라는 JVM(Java Virtual Machine)을 개발
    - 안드로이드는 이러한 달빅이라는 런타임 환경을 통해 더 확실하게 HW와 SW를 계층화
    - 달빅은 임베디드 단말에 최적화된 VM으로 메모리 보호(Memory Protection), 가비지 컬렉션(Garbage-collection), 라이프 사이클 관리 등의 특징
    - 달빅이 동작하는 모든 단말에서 안드로이드 애플리케이션이 동일하게 실행 가능
    - Sun의 JVM보다 작은 메모리 환경에서도 잘 실행할 수 있도록 성능 개선의 효과도 얻으면서 자바를 탑재할 때의 라이선스 비용의 문제도 해결

안드로이드 아키텍처

◉ 안드로이드 런타임

- Core Libraries

안드로이드 Core 라이브러리는 Java Standard Edition과 Java Mobile Edition과는 다르지만, 중복되는 부분이 상당히 있다.

J2SE 5.0 Supported	J2SE 5.0 Not supported	3rd Party libraries	Android Specific libraries	
java.awt.font	java.applet	org.apache.http	android	android.preference
java.io	java.awt	org.json	android.app	android.provider
java.lang	java.beans	org.xml.sax	android.content	android.sax
java.math	java.lang.management	org.xmlpull.v1	android.content.pm	android.telephony
java.net	java.rmi		android.content.res	android.telephony.gsm
java.nio	javax.accessibility		android.database	android.test
java.security	javax.activity		android.database.sqlite	android.test.mock
java.sql	javax.imageio		android.graphics Provides	android.test.suitebuilder
java.text	javax.management		android.graphics.drawable	android.text
java.util	javax.naming		android.graphics.drawable.shapes	android.text.method
javax.crypto	javax.print		android.hardware	android.text.style
javax.microedition.khronos	javax.rmi		android.location	android.text.util
javax.net	javax.security.auth.kerb		android.media	android.util
javax.security	eros		android.net	android.view
javax.sql	javax.security.auth.spi		android.net.http	android.view.animation
javax.xml.parsers	javax.security.sasl		android.net.wifi	android.webkit
org.w3c.dom	javax.sound		android.opengl	android.widget
org.xml.sax	javax.swing		android.os	com.google.android.maps
	javax.transaction			dalvik.bytecode
	javax.xml			dalvik.system
	org.ietf.*			
	org.omg.*			
	org.w3c.dom.*			

안드로이드 아키텍처

- **리눅스 커널**
  - X-Window와 같은 내장 윈도우 시스템 포함 않음
  - 표준 리눅스 유틸리티 전체를 포함하지 않음
  - Alarm, Ashmem, Binder, Power Management, Low Memory killer 등 확장 및 추가
  - 메모리 및 프로세스 관리, 인가 기반의 보안 모델, 오픈 소스 기반 등의 장점을 이용하기 위하여 리눅스 커널을 안드로이드에서 이용

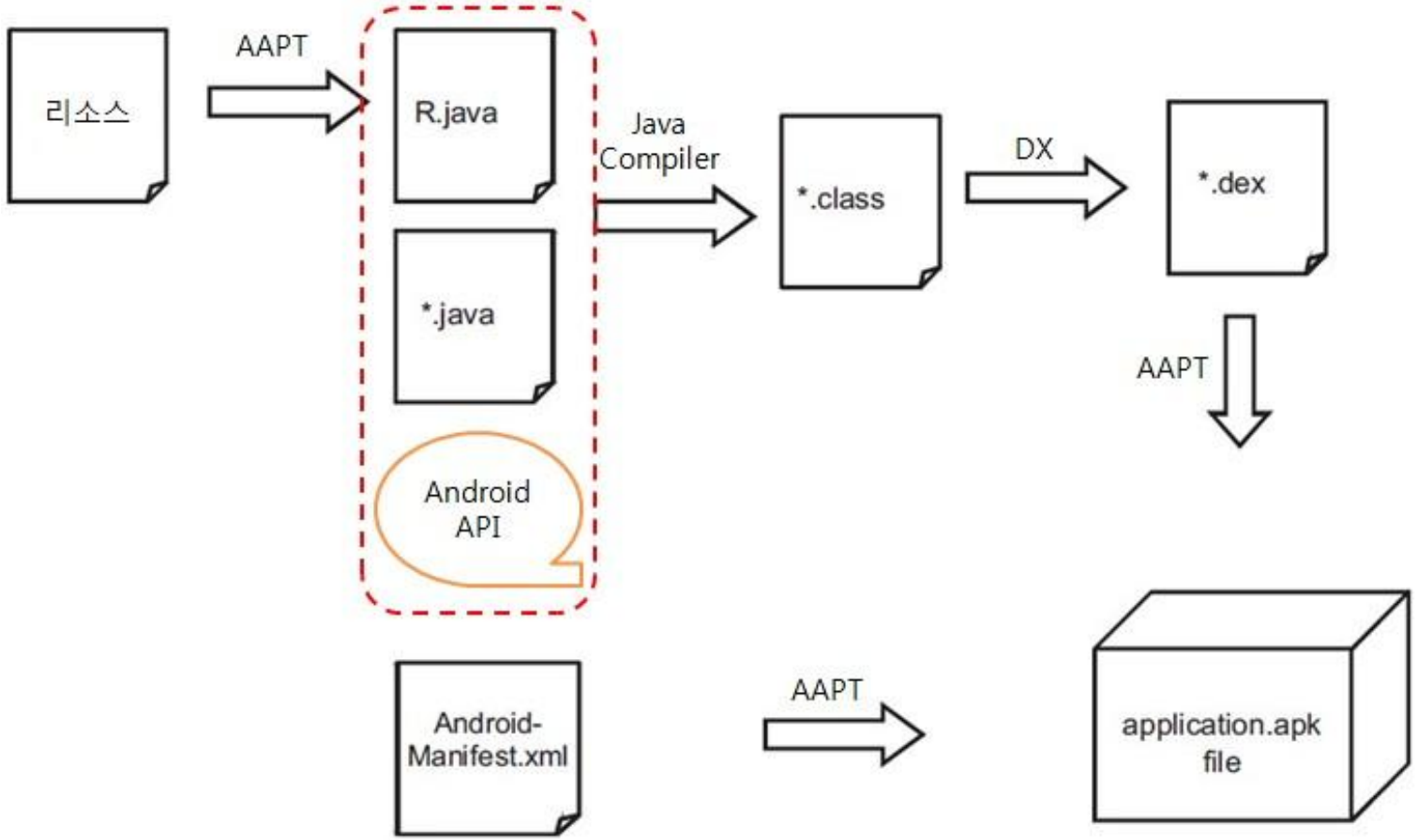
안드로이드 애플리케이션

개요

- 안드로이드 애플리케이션은 일반적으로 자바 프로그래밍 언어로 작성
- 소스 코드는 컴파일된 후 클래스 파일을 거쳐 애플리케이션에 필요한 리소스 등과 함께 AAPT라는 안드로이드 도구에 의하여 하나의 패키지로 통합
- 아카이브 파일로 압축된 안드로이드 패키지는 .apk라는 확장자를 가지며 하나의 안드로이드 애플리케이션으로 간주
- 또한 안드로이드 패키지는 모바일 디바이스에 배포 및 설치하기 위한 수단

개요

- 애플리케이션 빌드 과정



애플리케이션 컴포넌트

- **개요**
  - 안드로이드 애플리케이션은 자바 언어로 작성
  - 컴파일 후 압축된 아카이브 파일(.apk 파일)로 모바일 디바이스에 배포
  - apk 파일 내부에 있는 모든 코드는 하나의 애플리케이션으로 간주
  - 애플리케이션은 4가지 컴포넌트의 조합으로 구성
    - Activity
    - Service
    - Broadcast Receiver
    - Content Provider
  - 애플리케이션의 컴포넌트를 AndroidManifest.xml에 포함
  - Intent는 어떤 작업에 대한 요청, 즉 Activity, Service, Broadcast Receiver 컴포넌트를 활성화하는데 필요한 명령과 데이터를 담은 클래스

애플리케이션 컴포넌트

- **Activity**
  - 뷰와 이벤트 응답으로 이루어진 인터페이스
  - 애플리케이션에 하나 이상의 Activity 포함 가능
  - 하나의 화면에서 다른 화면으로 바뀌면 이전 화면을 위한 Activity는 Activity Stack에 저장
  - 화면과 화면 사이를 이동할 때 Intent라는 클래스를 사용
  - MS 윈도우의 윈도우와 비슷하지만 윈도우와 달리 동시에 여러 개 돌릴 수 없고 Activity Stack에 저장

애플리케이션 컴포넌트

- **Service**
  - 비쥬얼한 사용자 인터페이스 없이 정해지지 않은 시간 동안 내부에서 실행되는 코드
  - 예를 들어 백그라운드 음악 재생, 네트워크 상에서 데이터 전송, 혹은 Activity에게 계산 결과를 제공
- **Broadcast Receiver**
  - 외부 이벤트를 처리하는데 사용되며 사용자에게 발생한 이벤트의 종류를 알려줌
  - 시간대 변경, 배터리 부족, 사진 촬영, 사용자 언어 설정 변경 등
- **Content Provider**
  - 다른 애플리케이션에게 유용한 애플리케이션 데이터의 특정 집합을 생성
  - 자신의 데이터를 SQLite 혹은 파일에 저장 가능
  - 애플리케이션 사이에 데이터를 공유할 때 유용

프로세스와 생명주기

- **개요**
  - 안드로이드에서는 모든 프로그램이 리눅스 기반에서 실행
  - 따라서 모든 프로그램은 실행될 때 하나의 프로세스를 가짐
  - 프로세스는 애플리케이션 컴포넌트가 실행되는 장소
  - 매니페스트 파일에서 <process> 엘리먼트를 사용하여 컴포넌트가 실행될 프로세스를 지정 가능
  - <process> 엘리먼트를 사용하여 각 컴포넌트가 자신만의 프로세스에서 실행되도록 할 수 있으며, 또한 다수의 컴포넌트가 하나의 프로세스를 공유하도록 설정 가능
  - 안드로이드 시스템은 메모리가 부족할 경우 중요하지 않은 순서대로 프로세스를 메모리에서 제거

애플리케이션 종류

- **Managed Application**
  - 가장 기본적이고 주요한 안드로이드 애플리케이션 개발 방식
  - 달빅 VM에서 동작하는 순수 자바로만 개발한 프로그램
  - 안드로이드는 달빅 위에 여러 프레임워크 API를 제공하고 있고, OpenGL이나 Multimedia 등의 라이브러리에서 지원하는 기능 또한 API 형태로 달빅 위에서 이용할 수 있도록 제공
  - 개발자는 SDK에서 제공하는 이클립스 플러그인을 통해서 쉽게 UI를 구성하면서 개발 가능
  - 순수 자바로만 개발을 해도 대부분의 Rich UI가 구성 가능하고 백그라운드 서비스까지 개발이 가능하기 때문에 사실상 대부분의 안드로이드 애플리케이션은 이러한 방식으로 개발
  - 다만 자바라는 언어의 특성상, 약간의 속도 저하와 예측 불가능한 Garbage collection의 수행이라는 약점
  - 하드웨어나 라이브러리와 밀접하게 결합이 되어야 하는 경우는 기존의 프레임워크 API로는 불충분

애플리케이션 종류

- **Native Application**
  - 달빅 내에서 C/C++으로 구성된 동적 라이브러리를 적재하여 JNI(Java Native Interface) 형태로 함수를 호출하는 방식
  - 구글이 최근 NDK(Native Development Kit)를 따로 배포하여 C/C++의 코드를 손쉽게 .so 형태의 라이브러리로 cross-compile하여 생성 가능
  - CPU의 의존도가 높은 프로그램, 반응 속도가 중요한 입력 창의 검색부분, 대용량의 데이터 처리, 또한 하드웨어를 직접 다루거나 기존 달빅 API로 구현하기 어려운 경우 효율적
  - 달빅 위에서 돌아가는 것이 아니기 때문에 달빅 내의 Sandbox로 보호받는 효과를 누릴 수 없으므로 메모리 제어를 신경 써야 하는 등 프로그램 개발에 보다 세심한 관리가 필요

애플리케이션 종류

- **AJAX Application**
  - JavaScript와 Ajax 등으로 웹페이지를 작성하고 안드로이드의 브라우저를 통해서 수행하는 방식
  - 웹킷(Webkit) 코어와 SquirrelFish 자바스크립트 엔진을 사용하고 있으므로 PC 기반의 동작과 유사한 기능 제공
  - 구글의 대표 웹 서비스(Gmail, Google Docs 등)가 순수하게 Ajax로만 구현
  - 안드로이드에서는 브라우저 내에서 이러한 Ajax나 HTML5의 일부 규격을 제공하여 이질감 없이 웹 애플리케이션을 일반 애플리케이션처럼 개발 가능
  - 항상 브라우저를 통해서 수행이 되어야 하므로 백그라운드 서비스 등이 불가능하고, 시스템이나 프레임워크 내의 접근이 불가능
  - UI를 구성하는 렌더링 속도가 브라우저를 더 거치기 때문에 일반적인 달빅 애플리케이션에 비해 굉장히 느린 편 → UI의 속도가 중요하지 않고, 보다 범용적이고 간단하게 만들 수 있는 애플리케이션 개발에 적합한 방식

안드로이드의 미래

개요

- **안드로이드는 모바일 컴퓨팅의 미래를 만들어갈 강한 잠재력이 있는 모바일 플랫폼**
- **안드로이드는 오픈 소스 환경이며 여러 모바일 업체와 개발자들이 참여하고 있으며 발전 속도가 매우 빠르다.**
- **안드로이드는 구글 및 OHA의 주도로 이루어지지만, 플랫폼에 포함되어 있는 수많은 라이브러리는 계속 향상된 버전을 적용**
  - 예를 들어 리눅스 커널은 버전 2.6.18부터 시작했지만 안드로이드 1.6은 리눅스 커널 2.6.29을 사용하며, 5월에 발표한 안드로이드 2.2는 최신 리눅스 커널인 2.6.32 기반
  - 웹킷의 경우도 오픈 소스의 변경 사항에 따라 끊임없이 버전업

개요

◉ 안드로이드 플랫폼 버전별 변화

버전	1.5	1.6	2.0/2.1
코드명	Cupcake	Donut	Eclair
릴리스	2009. 4. 30	2009. 9. 15	2009. 10. 26/ 2010. 1. 12
리눅스 커널	2.6.27	2.6.29	2.6.29
추가 기능	다국어 지원, 자동완성 입력기 등	제스처gesture, 전체 검색universal search, TTSText to Speech 등	라이브 배경화면 live wallpaper, 디지털 줌digital zoom, 멀티터치multi-touch 등

개요

- **코드명 규칙**
  - 안드로이드 코드명은 음식 이름을 사용
  - 알파벳 순서대로 명명
- **코드명과 버전**

안드로이드 1.1


안드로이드 1.5
(cupcake)


안드로이드 1.6
(donut)


안드로이드 2.0/2.1
(Éclair)


안드로이드 2.2
(FroYo)

안드로이드 전망

- **안드로이드 플랫폼의 개방과 무료**
  - 일반적으로 폐쇄적이고 독점적인 정책은 사용자로부터 외면
  - 운영체제의 경우 IBM OS2, 컴퓨터의 경우 애플 매킨토시, 비디오 레코더의 경우 소니의 베타 방식 등은 성능이 매우 우수하지만 폐쇄적인 환경을 고집함으로 말미암아 실패
  - 안드로이드는 모든 소스를 오픈하며 어떤 모바일 단말기 제조사도 안드로이드 SDK를 사용하고 수정해도 라이선스에 문제가 없다.
  - 따라서 일개 회사에 종속적인 애플 아이폰과는 달리 안드로이드는 모든 모바일 단말기 뿐만 아니라 임베디드 단말기를 위한 플랫폼으로 널리 사용 가능

안드로이드 전망

- **안드로이드 SDK의 편의성과 강력함**
  - 개발자를 제대로 지원하지 못하면 성공하기 어렵다.
  - 아이폰 개발 도구는 사용하기 어렵고 주관적일 수 있는 애플리케이션 등록 과정으로 인하여 개발자를 주저하게 함
  - 애플 Object-C 언어는 C 언어와 유사하지만 아이폰 애플리케이션 이외에는 거의 미사용
  - 안드로이드는 C/C++ 언어로 개발하는 경우도 있지만 자바 프로그래밍 언어를 주로 사용. 더구나 개발 환경도 이클립스라는 좋은 도구의 도움 가능
  - 안드로이드에서는 네이티브 애플리케이션과 서드파티 애플리케이션이 평등 → 미리 설치된 애플리케이션을 확장하거나 아예 대체 가능

안드로이드 전망

- 모바일 서비스의 오픈화라고 하는 판도라 상자
- 개방된 모바일 세계에서 사용의 편리함, 디자인, 견고성, 가격 등에 대하여 다른 제품과 차별화하는 것이 성공의 중요 요소
- 구글의 행보