**E COMMERCE AND E BUSINESS**

Module 1: Intro

**1.1** **History of Android**

The history and versions of android are interesting to know. The code names of android ranges from A to J currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwitch, Jelly Bean, KitKat and Lollipop. Let's understand the android history in a sequence.

1) Initially, Andy Rubin founded Android Incorporation in Palo Alto, California, United States in October, 2003.

2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are Andy Rubin, Rich Miner, Chris White and Nick Sears.

4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

5) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

6) In 2007, Google announces the development of android OS.

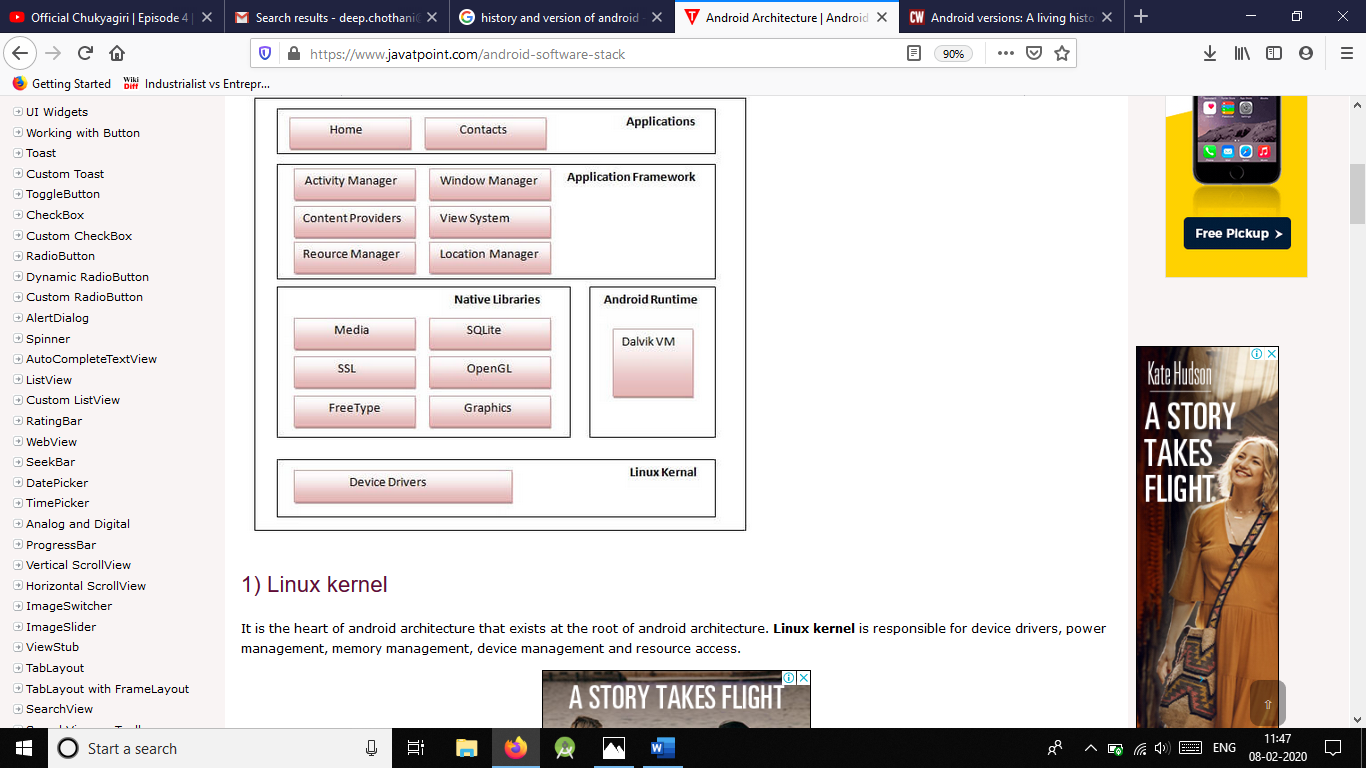
7) In 2008, HTC launched the first android mobile.

**1.2 Android Architecture**

**android architecture** or **Android software stack** is categorized into five parts:

1. linux kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications

Let's see the android architecture first.



1) Linux kernel

It is the heart of android architecture that exists at the root of android architecture. Linux kernel is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries

On the top of linux kernel, their are Native libraries such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

3) Android Runtime

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4) Android Framework

On the top of Native libraries and android runtime, there is android framework. Android framework includes Android API's such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5) Applications

On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernel.

**1.3 Install Android**

Android supports java, C++, C# etc. language to develop android applications. Java is the officially supported language for android. All the android examples of this site are developed using Java language and Eclipse IDE.

Here, we are going to tell you, the required software’s to develop android applications using Eclipse IDE.

There are two ways to install android.

1. By ADT Bundle
2. By Setup Eclipse Manually

1) By Android Studio

It is the simplest technique to install required software for android application. It includes:

* Eclipse IDE
* Android SDK
* Eclipse Plugin

If you download the Android Studio from android site, you don't need to have eclipse IDE, android SDK and eclipse Plugin because it is already included in Android Studio.

If you have downloaded the Android Studio, unjar it, go to eclipse IDE and start the eclipse by clicking on the eclipse icon. You don't need to do any extra steps here.

2) By set up eclipse manually

* Install the Java Development Kit (JDK)
* Download and install the Eclipse IDE
* Download and install the android SDK
* Download the ADT plugin for eclipse
* Configuring the ADT plugin
* Create an Android Virtual Device (AVD)
* Create and run the simple android example

**1.4 Android Core Building Blocks**

An android **component** is simply a piece of code that has a well-defined life cycle e.g. Activity, Receiver, Service etc.

The core building blocks or fundamental components of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

Activity: An activity is a class that represents a single screen. It is like a Frame in AWT.

View: A view is the UI element such as button, label, text field etc. Anything that you see is a view.

Intent: Intent is used to invoke components. It is mainly used to:

* Start the service
* Launch an activity
* Display a web page
* Display a list of contacts
* Broadcast a message
* Dial a phone call etc.

For example, you may write the following code to view the webpage.

Intent intent=new Intent(Intent.ACTION\_VIEW);

intent.setData(Uri.parse("http://www.javatpoint.com"));

startActivity(intent);

Service: Service is a background process that can run for a long time. There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

Content Provider: Content Providers are used to share data between the applications

Fragment: Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

AndroidManifest.xml: It contains information about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

Android Virtual Device (AVD): It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

**Android Emulator**

The **Android emulator** is an **Android Virtual Device (AVD),** which represents a specific Android device. We can use the Android emulator as a target device to execute and test our Android application on our PC. The Android emulator provides almost all the functionality of a real device. We can get the incoming phone calls and text messages. It also gives the location of the device and simulates different network speeds. Android emulator simulates rotation and other hardware sensors. It accesses the Google Play store, and much more

Testing Android applications on emulator are sometimes faster and easier than doing on a real device. For example, we can transfer data faster to the emulator than to a real device connected through USB.

The Android emulator comes with predefined configurations for several Android phones, Wear OS, tablet, Android TV devices.

**Dalvik Virtual Machine | DVM**

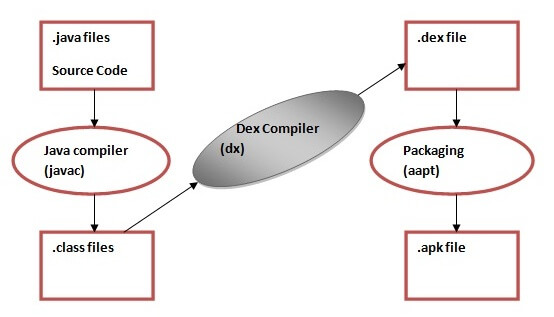
As we know the modern JVM is high performance and provides excellent memory management. But it needs to be optimized for low-powered handheld devices as well.

The **Dalvik Virtual Machine (DVM)** is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for memory, battery life and performance.

Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein.

The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.

Let's see the compiling and packaging process from the source file:



The **javac tool** compiles the java source file into the class file.

The **dx tool** takes all the class files of your application and generates a single .dex file. It is a platform-specific tool.

The Android Assets Packaging Tool (aapt) handles the packaging process.

**AndroidManifest.xml file in android**

The **AndroidManifest.xml file** contains information of your package, including components of theapplication such as activities, services, broadcast receivers, content providers etc.

It performs some other tasks also:

* It is responsible to protect the application to access any protected parts by providing the permissions.
* It also declares the android api that the application is going to use.
* It lists the instrumentation classes. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.

This is the required xml file for all the android application and located inside the root directory.

**Elements of the AndroidManifest.xml file**

The elements used in the above xml file are described below.

<manifest>

manifest is the root element of the AndroidManifest.xml file. It has package attribute that describes the package name of the activity class.

<application>

application is the subelement of the manifest. It includes the namespace declaration. This element contains several subelements that declares the application component such as activity etc.

The commonly used attributes are of this element are icon, label, theme etc.

android:icon represents the icon for all the android application components.

android:label works as the default label for all the application components.

android:theme represents a common theme for all the android activities.

<activity>

activity is the subelement of application and represents an activity that must be defined in the AndroidManifest.xml file. It has many attributes such as label, name, theme, launchMode etc.

android:label represents a label i.e. displayed on the screen.

android:name represents a name for the activity class. It is required attribute.

<intent-filter>

intent-filter is the sub-element of activity that describes the type of intent to which activity, service or broadcast receiver can respond to.

<action>

It adds an action for the intent-filter. The intent-filter must have at least one action element.

<category>

It adds a category name to an intent-filter.

**OS vs Software Stack**

The OS is the core kernel of the system, and the software stack is the software sitting on top of the kernel which enhances and expands the functionality of the system.

Windows tends to blur the line a bit, but Linux makes it a bit more visible.

In Linux, the kernel is the OS. The software stack sitting on top of the kernel can include things like:

* The X window system
* The Gnome window manager
* Applications for managing the system
* And so on...

So things like a control panel application to edit system settings aren't part of the operating system. They're part of the software stack on top of the operating system.

You could have [a very tiny operating system](http://en.wikipedia.org/wiki/MINIX) which doesn't have much of a software stack running on top of it. By itself, it wouldn't provide much usability, but rather a foundation for adding usability.

Android has its Software stack built upon Linux Kernel, which takes care of the power management acts as an interface between other hardware parts of the device.