The Study of the Study of Android[1]
by Givon Zirkind

Introduction

The computer industry moves fast. Right? Yes. But, the leader of the computer industry does not change quickly or often. IBM dominated for decades. IBM was replaced by Microsoft. [HIX01] [NOW01] Microsoft's leadership lasted for decades. Recently, Microsoft lost it's leadership in the computer industry to Android. [KOV01] [MAH01] Seeing this trend, I decided to learn Android. Both as a computer scientist and as a computer professional, I would be remiss not to. Indeed, there is a serious shortage of app / mobile programmers, Android and iOS. When I started out studying Android, it was difficult to find good study materials. There is a flood of materials on the Internet. Books galore in the bookstores. But, which are the good ones? I went to a Google Developer's Group in the hope of getting an answer to that question. Unfortunately, I was just told what I already knew. “There is a lot of junk out there.” But, no definitive answer for a good book or tutorial. Having made the journey, I am sharing from my personal experiences, hoping it will help others make the same journey.

Like Champillion, who deciphered the Rosetta stone, one has to first study the necessary prerequisites. Something, all the books mention. It is necessary to know Java, XML and then the “Android Java APIs”. A development tool like Eclipse, IDE is necessity in my opinion. Something all the books mention as well. How much do you need to study of each subject? I will discuss that. A background in art and design; computer human interaction, is very helpful.

There is a tendency to bash the industry leader (find cite ###) and the big guy. Google has done many nice and good things. The world has changed with lots of “free” stuff available on the Internet. OpenSource projects abound. There is no free lunch however. Many of these “freebies” are paid for, by the participants becoming targets of advertising or; being solicited for donations or; advertising for advertisers. My goal is to make observations of the good; what is lacking and; suggestions for improvement. For any technical product to succeed, there must be technicians. For there to be technicians, there has to be good training. The goal of this paper is to show where that good training is.

Nomenclature

Textbooks usually start off by stating terms about the subject of study. This gives us ideas and words to discuss the subject. This brings clarity of thought to the matter. Android is an operating system developed by Google. Android is open source. It has a design where each program, “app”, runs as a unique user. (This is different from programs running in a user's space, under a user's permission.) This gives data protection preventing one app from interfering with or overriding another app's data. This secures threads and processes. A good design, especially for the app-mobile environment. Java is an object oriented computer language. XML is a data layout format. What one uses XML for is the users choice: Database and/or record layout; Transmission Protocol, etc. What one uses for tags in XML, is of your own definition. Android Java APIs are programming functions (methods, routines) that do special functions, often used functions, when programming a mobile device. It appears to me, that the majority of these functions are for screen layout and UI (user interface). Eclipse is an IDE (interactive development environment) for Android apps.

The phrases “Android programming” and “programming apps for Android” need definition. I feel it is a subtle misnomer that causes some confusion. Unless you are manufacturer, trying to get Android to work on your

---

1 The title of this article is patterned after [IVE01]. From the time of the ancient Greeks, until the Renaissance, Hieroglyphics fascinated the European cultures. The study of Hieroglyphics was mired in mistaken beliefs. Since most of the Hieroglyphics were found in the Temples, there was a belief that hieroglyphics were religious and spiritual in nature. A printing press would have proven quite different. From examination of papyri, it seems that the Egyptians loved stories and preserved them in papyri. [GOR01] Not many papyri have survived. This belief that hieroglyphics had cosmological references stymied the study, understanding and decoding of Hieroglyphics. Only after a change in philosophy about Hieroglyphics, was its decipherment possible. This change in philosophy about hieroglyphics was made by Champillion, who dedicated his life to deciphering hieroglyphics. Champillion started his studies by first studying various languages, such as Coptic, in the belief that the languages were similar and that this knowledge would aid him in deciphering hieroglyphics. [FRI02] The analogy is that to program a phone running Google's Android, one first has to study several other computer languages and tools.

device, most “Android programmers” are not Android programmers at all! They are Java programmers or; programmers who know some Java; who also know Google's Java APIs and XML tags for programming apps that work in an Android environment.

All the training materials that I have read, make it a point to try and explain this. It took a while to register. There is the Android release level vs. the (Java) API level you are working with. There is some correlation between the two levels. For example Android release “x” will relate to API level “z”. The correlation is usually only a happenstance of timing—as far as the app programmer is concerned. One can often mix the different release levels without a problem.

Like most manufacturers, Google has its own language. Not over the top, but sometimes confusing with the rest of the computer world. For example, the use of the word “cursor” as the index or position in the file, is not quite what I am accustomed to. The cursor usually being the spot on the screen where you type. A “Spinner” does not spin. It is a drop down. There are graphics that do spin, round and round, while you wait for something to load. But, that is a progress bar. A very bent bar, but a bar. I personally find the names misnomers. Again, these are not major issues and stereotypical of the computer industry.

<u>Prerequisites</u>

As for Android, the operating system, an app programmer needs to know only a few concepts. These concepts are stated in most documentation. Books, Google's documentation, etc.

In addition, it is wise to have basic computer concepts, variables, basic programming constructs, input/output, database, etc.

Basic programming constructs and Java. Although, it appears to me, one can be quite functional in producing apps with minimal knowledge of Java. In my experience, one must be well versed in arrays. Lists are an integral part of the graphical design of apps. The processing of these list commonly will involve arrays.

Object Oriented Programming. Again, minimal knowledge appears to me to suffice. The concepts of classes, methods, inheritance, implementation, will the uniqueness of objects will do. One has to get very comfortable with the “black box” concept. APIs and their objects' methods are used extensively. One must use and be comfortable with using qualified object attributes in the form of methods. This can become tricky with supplying parameters and using returned values.

Intent and Activity. In an area, between or an intersection of Android and Java app programming for Android, is the concept of Intent and Activity. These concepts are critical to understanding app programming in Android. All the books and documentation I have read, covers these concepts in their beginning. The Google development docs gives an very good explanation of these concepts.

There is an option to program in C and “raw” low level format. C programming is implemented as a call from a Java program to a C program. I have insufficient knowledge to comment on either implementing a C program or a low level program.

The building blocks. Google provides pre-structured and pre-designed graphical objects, artifacts, many interactive. Ex. Buttons, drop down lists (Spinners), progress bars, boxes for text or images, check boxes, screen layouts—linear, table, grid, lists, etc. The first step is to become familiar with these available “features”. They are the building blocks of apps. In combination and in many permutations, together, this is how apps are made.

Google gives many design suggestions. So far, all I have read are good suggestions. One suggestion particularly stuck out in my mind. Regarding messages, titles and descriptions of how buttons work, Google says to keep it brief; do not repeat yourself and; “We don't tell our people what to do.” Tell them what it does. Tell them how it works. But, “We don't tell our people what to do.” Personally, I find this a bit comical! Also, very indicative of the philosophy of the OpenSource, freeware nature of Google. A far cry from a highly structured IBM or Microsoft with lengthy many termed contracts.

Which building blocks are a “must know”? I have never seen a list. Google does tell you in the docs that it is important to learn how to do a ListView. Many applications display lists. It is an important feature. Buttons and text seem critical to me too.

Google provides excellent documentation—in the conceptual sense—about the building blocks, in the training section of Google's developer site. Google explains quite well, with sufficient accompanying pictures; what the building blocks are; what they do and; how to use them—in a conceptual sense. Programming these artifacts is a different matter.

It is the programmer's creativity in assembling and combining these building blocks that makes a great and spectacular app.

<u>The IDE (Interactive Development Environment)</u>

Eclipse is provided by Google as development environment for Android app programing. Eclipse is somewhat intuitive. As one learns more of the functions of the graphical building blocks, it becomes easier to recognize what app features are available and how to use the Eclipse GUI to quickly build an app. Some features are better programmed by going into the code. An option which is also available.

Like many large programs, Eclipse has so many features, it is impossible to know them all. Hitting a button unknowingly can cause much grief and ruin a perfectly good working program. [FRI01] [LAJ01] Nevertheless, Eclipse is a good, essential and necessary tool in developing Android apps.

Eclipse is a memory hog. This should be seriously rethought by Google. Often, Eclipse will just start using huge chunks of memory. The powers to be ought to try to optimize Eclipse's use of memory.

There are so many components to programming an app, each with its own syntax, that keeping track of it all can be quite a difficult task. An IDE (Interactive Development Environment), such as Eclipse is a must. While the auto complete function can be quite annoying, it can also be quite useful in illuminating methods and attributes that would be useful in a programming solution. Eclipse also renders the screen layouts. A very useful function. The rendering applies to different platforms (devices and languages). Not all commands are rendered. (The underline <u></u> does not render. An oversight, I am sure.

Eclipse also has an emulator. The emulator is a disaster! At first I thought my machine had just frozen. Then, I thought I just didn't have enough memory when the emulator did not run. That was until I read [BUR02] that you can expect it to take 10 minutes for the emulator to load! That is just plain ridiculous! That is something that has to be fixed! I don't know what other developers do. But, I simply load my app onto a phone and test it.

All books and courses ([RAT01]; Porter, Adam; Programming Mobile Applications for Android Handheld Systems, Coursera.Org, University of Maryland) that I have read or seen, describe how to load and use Eclipse as the very first step in teaching how to program for Android. This is a correct approach in my opinion.

<u>Java</u>

Being a programmer and having studied C++, I am familiar with object oriented programming. I did read through the beginning of some Java books. [GOS01] [FEL01] (Gosling is the "Java Bible". Gosling authored Java. It is a bit too technical to be an easy read. But, it does cover every feature of Java, in detail. Felkner is an easier read and more practical approach.) Also, I have used some Java books as references. Using Oracle's tutorials [ORA01] and Java certification as a yardstick, I would rate myself on a level of Java certification 1.5 (There is a Java certification level 1 & 2.) This seems sufficient to program for apps.

Since lists are an often used component of Android apps and; lists are often programmed with arrays and array lists; in addition, preferences are also often programmed with arrays; it is necessary to understand well how Java handles arrays, arraylists, etc.

<u>Learning the Java APIs</u>

First, I turned to Google's developer site and Google's docs. The Google docs provide a lot of information. In my opinion, there is information overload and a lack of sufficient indexing to zero in on the documentation you may need. A major flaw to the Google API docs is the "insufficiency" or "incompleteness" of examples. (Explained in depth below.) In fact, there are no Eclipse examples! Since Eclipse is the major development tool for Google's

Android app programming, this is a serious shortcoming. (More on this, other disadvantages of the Google docs and recommendations for improvement later.)

Being overwhelmed by the sheer volume of the Google docs and unaware of how to sort them as well as make “views” of relevant subsets of the documentation; I felt the Google docs were too technical for a beginner. I turned to self help, teach yourself books instead. If I had known that Google provides an index that provides views into different categories of docs: Training-Development-Reference, etc. I might have read through the Google docs instead of self study books. With hindsight, this might have been a better approach. However, the self help books provided other fundamentals, such as using Eclipse, that are not in the Google Android development docs.

While I may have doubted the advantage or necessity of reading through the Google docs; now, I know it is a necessity and of advantage. Reading through the Google “Development” docs is a must for knowing how to program the Google Java APIs for app programming. Although at what point in the process one should start with the Google Java API docs, I am still unsure.

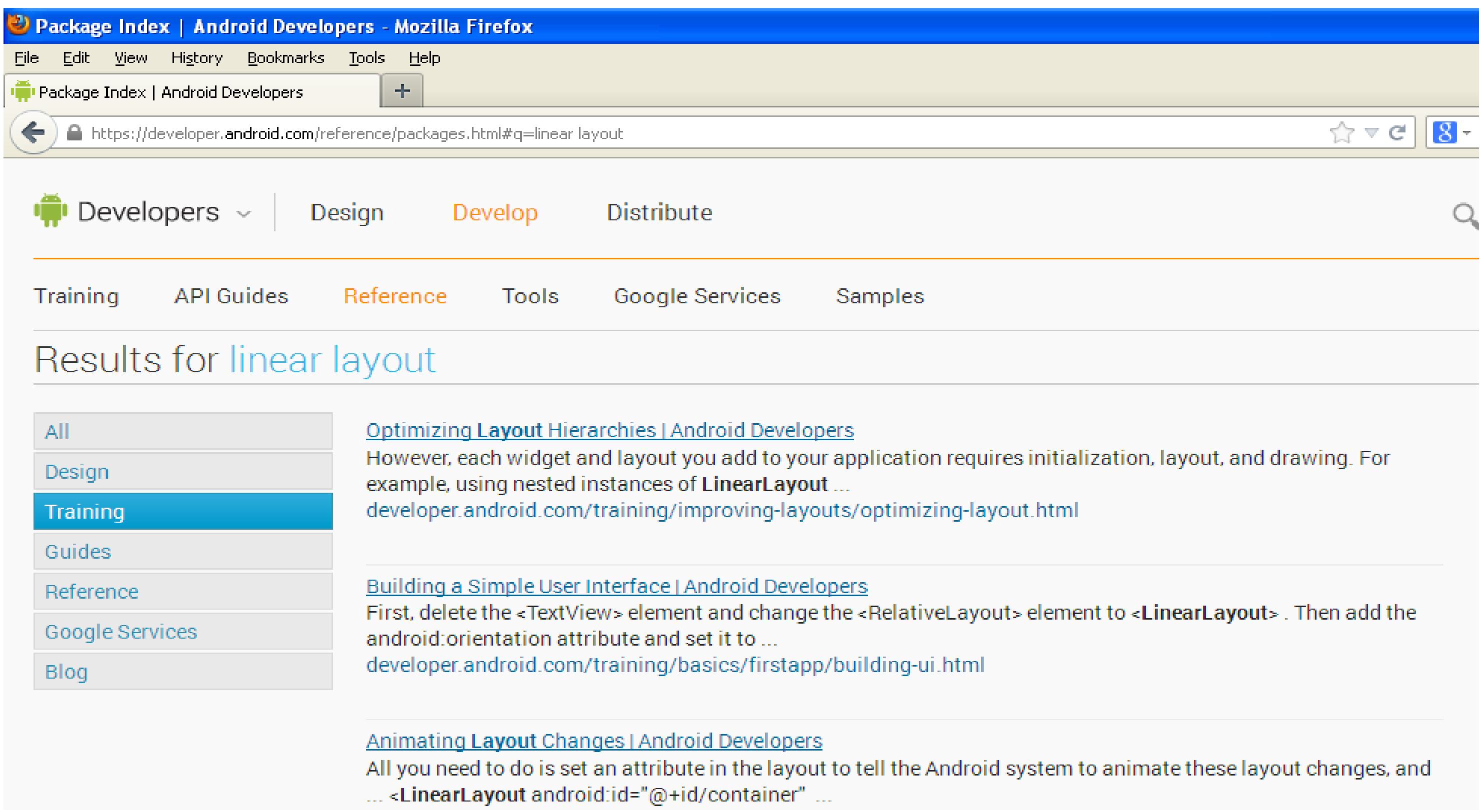

Illustration 1: The Developer's site with a list of documents sorted by type.

There is a secondary index on the side of the listing of documents. I didn't get it at first. Only after many searches and trying to narrow down the searches to the development and not training or reference docs, did I click on one of the buttons in the index “to see what it does”. That clarified the difference between the training and development docs.

Teach Yourself Books. Like all computer manuals and instruction books, “the manual is wrong”. Some books had examples that could never have compiled! I am not talking about an occasional typo! I mean, every example in every chapter had code that needed to be debugged! Without my background and experience in computer programming and debugging, I could have never gotten those programs working! Those books would have been useless! Yet, those books did provide me, with a thorough explanation, of sample programs that demonstrated basic app programs and programming functions. They did provide insight and some basic Java API knowledge.

The “For Dummies” series is good, in my opinion. [BUR03] by Burd is a good beginners book, in my opinion. I do not understand why he started with explaining a “Service”. Even so, it is a good book that covers the basics with *working* examples. He does have one excellent piece of advice for programmers, neophytes or veterans with years of experience. “Just keep staring at it. Eventually you will see the difference between what you wrote and what the working code looks like.”

The Order of Studying the Subjects

1Eclipse
2The Building Blocks
3XML Layout tags
4Java
5Java APIs

This seems to me to be the best path. Most books start off teaching Eclipse. This is a good place to start. Eclipse has a lot of components. It serves several functions and be the basis for learning all the other, following subjects. It will provide an environment for learning Java; the building blocks; the XML layout tags and the Java APIs.

Once one has the basics of Eclipse down, one can then study the building blocks and seem them in mock screen layouts. Point and drag will produce layouts. The underlying xml tags can be studied. This fixes the basic building blocks in one's mind. How they look. How they work. How they can be put together.

It is wise to study the basic building blocks before studying Java and the Java APIs. Java is a necessary skill for the Java APIs. The Java APIs may or may not work with the basic building blocks. But, as apps are inherently graphical programs, knowing the graphic components first is the most reasonable approach.

As some Java is necessary for programming the Java APIs, Java should be learned before studying the Java APIs. Then, the Java APIs should be studied.

Of the Java APIs themselves, there seems to be a consensus of the APIs that need to be studied and in which order. Intents and Activities are a necessary foundation. Fragments are to follow up Activities with. Then, a variety of APIs not necessarily in this order. But, all at the same level: Preferences, Various Inputs, Spinners (drop downs), Buttons, Images, Settings. Then, the list processing. Then, the study can diversify into areas of personal interest as required. Such as, the more difficult and less often used features such as timers, alarms, media, content providers (other apps, system apps, that provide data, like the contact app provides names and phone numbers), sensors, GPS, camera, microphone, interfacing with Google Maps, etc.

The Google Docs Shortcomings

The depth of navigation is 3, often 5+.

Example to load Google Development documentation one goes to the main index page https://developer.android.com/index.html. Then, one clicks on “Develop” to go to https://developer.android.com/develop/index.html. Then, one clicks on the “API Guides” title. Which brings up a sidebar of categories or chapters to choose from. One clicks on the category in the sidebar and a drop down opens up. Now you can choose an individual article.

Having to click that often just to get to the index I need, is too much. Conventional wisdom is that a user should not have to click more than 3 times to get to something often used. [MSD01]

While the Development docs do have sample code, often this code is incomplete. Prior knowledge may be assumed. While there are references for further reading and prerequisites (assumed knowledge), this is not always the case. Sometimes, this is critical and a necessary piece of knowledge is not explained. Judging by the number of questions asked on StackOverFlow.com, it appears this happens often.

A simple example. In the explanation of the implementation of a Spinner, a statement is made about needing a “string-array”. Right. What's a string-array? There is no link to the definition of a string-array. In the linear hierarchy of the docs, I am not sure, string-array was discussed prior to the explanation of the implementation of a Spinner.

Another example, in the explanation of the implementation of Listview, there is mention, with a snippet of code, that an “onClick” routine must be used for checking or removing the check of a row item. OK. Where do I put this onClick routine? A sample of the entire routine is given. But, there is no example of the routine being placed in the code. Was this to be a separate method in the class? Or, inserted into an existing method? The later is the case. Again, StackOverFlow.com comes to the rescue with questions and answers that serve as examples and clues to the undocumented matters.

A more technical example, when a screen rotates, it looses data. It is as if the program was stopped and restarted. There is a way to Save the state before rotation and Restore the state after rotation. Thank you very much for the warning! Great job! So I don't lose my hair trying to fix the “bug”! Exactly how do I use this Restore method? Yes I see the snippet of codes. The methods. Do I just cut and paste them into my program? I was under the impression that these Save and Restore routines were more “black boxes”, that saved and restored the app state for you. No. You have to save and restore the state yourself. These routines just do the saving and restoring at the proper time.

<u>Object Oriented Programming Considerations</u>

These examples bring us to the discussion of an issue in Object Oriented Programming like Java and programming for apps in Android. For one, there is a coherency deficiency. To program something, you have to do things in several “places” in different files and sections of code together.

For example (http://developer.android.com/guide/topics/ui/menus.html), to make a “Setting” there is a special method that needs to be added, “onOptionsSelected”; references must be added to the program's Manifest file; there is programming for the button that must be done in the “main” or “onCreate” method. When you do select the option, then there must be a called method “Activity” that actually does the changes.

Sometimes in order to program a certain function, there may be up to 5 different components.

The docs do not always give you all the pieces. Or, the docs do not give all the pieces clearly. An assumption of prior knowledge. That famous professorial comment to a paper comes to mind: “Obvious? Obvious to whom?!”

This can lead to errors like, something wrong with a graphic artifact in the “R class”, even a misspelled name—which gives the computer assumption the graphic does not exist—can make your previously working program no longer compile—without a clue or hint as to why. In fact, errors with the “R class” gave me a lot of grief when I started programming apps for Android. StackOverFlow.com has lots of questions about this, with the same answer expressing the same sentiment that this error is common and causes a lot of grief.

Another example. You can program everything “right”. Your program compiles. But, the program has a fatal exception because you have not added the new “Activity” to the program's manifest.

In addition, in my opinion, there are too many “black boxes”. This is an inherent problem to Object Oriented Programming. This is inherent to Java. There is not enough documentation to the black boxes, in my humble opinion. The references are a bit lacking too. I feel like a script kiddie. It makes for extra work debugging. If you do not really know how it works, you can not fix it.

For example, “isEmpty()”. A common attribute. What exactly does “isEmpty” mean? Null, space or “”? Perhaps 0? What exactly does it mean that a “ListView.isEmpty()”? Is the return a null or 0? Uninitialized or; has no rows? Questions like this have come up in my debugging my apps. Sometimes, for curiosity's sake, I play with it to figure it out. Other times, I find a quick, workable solution. Sometimes, the debugging process is slowed down and requires some reverse engineering, testing to ascertain just what these attributes really do.

[Once one becomes a proficient app for Android programmer, one can then delve the depths of these details and become an expert in Android app programming.]

Also, while data protection is good. Too much of a good thing is not. Similar to structured programming, where the structure becomes ridiculous overkill, the overuse of data protection makes for unwieldy and awkward programs. The overuse of data protection is an involved philosophical and computer science debate. But, something to consider in a live, quickly changing Object Oriented Programming environment.

[See http://programmers.stackexchange.com/questions/262/will-java-still-be-relevant-in-5-years for an interesting discussion if and why Java will continue to be used, it's advantages and disadvantages. Verbosity is certainly a shortcoming of Java.]

Other Resources

Stackoverflow.com is a must! I found myself referring to Stackoverflow.com often. In fact, with almost every new feature (API) that I study, there is usually a need to refer to Stackoverflow.com either for examples or for information that the Google docs leave out. The questions and answers on Stackoverflow.com will help clarify what the Google docs say.

In addition, it appears from the working of answers; that Google software engineers troll Stackoverflow.com and post answers. These Google software engineers provide some very insightful advice, which only they in know, can provide.

Also, I personally find it comforting to know that other people have had the problem too. It's good to know, "It is not just me" who did not understand the Google docs.

It is a good idea to peruse Stackoverflow.com from time to time, about various APIs you use. This can give you insight, tricks and tips, in using the APIs, XML layouts and building blocks. In fact, many of the tricks and tips become future Android "widgets"--reusable structures, like date or time pickers.

Graphic troubleshooting

Apps are inherently graphical programs. Not all apps are graphical. A "Service" running in the background, may have no user interface to speak of. But, most apps are graphical. "Activities" are GUIs. Apps are mostly made up of "Activities. This introduces a new element to debugging. The graphics have to be debugged.

For one, all screens have to be tested for rotation. What renders nicely in portrait mode, may not render well in landscape mode. What renders well on one device, may not render well on another device. All of this has to be tested and accounted for.

The Java "package" for the app provides ways to handle all these different graphical states. It is a simple task, that requires work. Easily done, if you put your mind to it.

As an anecdote on this subject of graphical debugging, while I was learning how to make an "action bar", I copied the Google provided icon into my app. I checked to make sure the color theme was correct. But, when I tested the app, the icons did not appear. I checked the code. I reread the apps. Then, I checked the icons again in a graphical editor. Computers doth make fools of men. Kind of hard to see a black icon on a black background.

Multi-Lingual

The XML layout has a built in structure to easily make multilingual layouts possible. To me, this is just a WOW! I remember when large banks had special programmers and programming teams to write routines for their software to be multilingual.

There are linguistic considerations to deal with. Translations must be contextual. When we "settings" in the computer sense, we mean parameters. We do not mean settings in an environmental sense or, table settings. Yet, the ability to make an app multilingual is just amazing!

Reuseable code

Good programming should include "reusable code". The Java APIs are reusable. Fragments are true reusable code. Fragments can be incorporated into Activites. Mixed and matched. So, that an app running on different devices can display functionality with the size constraints of different devices. However, if I want a similar layout or release level, so far, I have found that I need to cut & paste from one "project" to another. (A project is the collection of all the files—Java classes, XML layouts, manifest, graphics, etc.--that go into an app.) This creates errors from the "R" class. Cutting and pasting will import things you do not want to import. This causes a lot of grief. However, there are libraries that can contain reuseable generic Java classes. Math routines and Google Licensing are examples.

<u>Programming considerations</u>

The more things change, the more they remain the same. As computers advance, the same old questions arise. Saving disk space. Programs should use as little memory as possible. Optimization. How to make a process run as fast as possible. Handling exceptions. Mobile programming is not different. Apps that are too big tend to be deleted by users. Apps with poor response time are not used. Running apps can be removed from memory to save space. Their states are saved. But, we don't want to waste too much space saving all these states. Have large files? Maybe you should store them on their own removable storage device—an SD card.

<u>ΔGoogle – The Rate of Change of Google</u>
<u>Or; This Feature Has Been Deprecated</u>

The rate of change of Google is very high. In the short time that I have been studying Android app programming, Android went through 2 release levels, the APIs several release levels. The API level KitKat came out. The change is quick. From what I have seen, the change is good. It does make any teach yourself book obsolete before printing!

There is backwards compatibility.

If one chooses to become an Android app programmer, one must be ready to constantly learn.

<u>Suggestions</u>

In my opinion, the Google should be offered in book form for those who want it. Not that Google should print these books and release them to Barnes & Nobles or Amazon. But, there are publishers who can print a book one at a time, for authors who sell publish. I believe Google should offer such a service. I believe there will be buyers. Even though Google Android changes very quickly that books on Android are "deprecated" before publishing; still the older releases still work; are sometimes easier to use and as reference tools, a book is good to have around. There is a special educational value to reading in book form. The option should be available for those who choose it.

Also, the Google docs should be available for purchase on CDs. Other vendors, like Oracle do so. There are buyers. CDs have their time and place. Not everyone has the availability of high speed connections necessary to read the Google docs at their leisure. Especially, while Eclipse, the memory hog it is, is chewing up resources, that make an Internet connection slow to a crawl.

For convenience, there are those who might download the Google Developer's Documentation, if it was available for download.

An index to the docs, like an index to a book would be most helpful. Even though one can search on a word, the results are often overload. While it is easy to find anything, if you know where to look, neophytes especially, or; an experienced app programming wanting to learn a new feature—but not knowing what it is called by Google; can have a difficult time searching for information. An index would be helpful.

Likewise, the reference pages need a book like index; better, more thorough explanations. Also, certain reference pages, especially the list of XML layout tags and their attributes, should be very easy to find! It should not require searches and searches to find that page!

Illustration 2: Examples Explaining Wrap Content and Fill Parent Attributes [YON01]

> A picture is worth a thousand words. The examples for the explanation of these basic XML layout attributes are very intuitive and easy to understand. Reading about this could take some time for the idea to register and crystallize. Why should developers spend hours reading and experimenting when they could easily peruse pictures and very quickly understand how all the attributes affect the graphics?

It would do well to have graphical examples for what each attribute of an XML layout tag does. App programing is innately graphical. It is fitting and appropriate to have graphical examples, especially of how XML layout attributes alter the layout.

As I pointed out above, it would be good if the index sort to the Google docs had a title.

<u>What You Need to Program Apps for Android</u>

So far, I have discussed areas of knowledge. Now, for the necessary software and hardware.

Eclipse, which we have already mentioned.

Java virtual machine to run Java and compile Java programs.

A phone and/or tablet to load your apps onto.

A USB cable to transfer your “apk's” from your computer to your phone or tablet.

<u>Summary</u>

It seems to me that learning Android app programming well, is an average 2 year project. (“As Boom Lures App Creators, Tough Part Is Making a Living“, New York Times, David Streitfeld, November 17, 2012) One online school claims to teach Android & iOS in a year. But, from what I have seen, 2 years is more realistic. The information is available for free. You just have to apply yourself. I hope the information provided here is of assistance to those who choose to study Android app programming and makes their learning easier.

<u>Bibliography</u>

[BUR01] Burd, Barry; Beginning Programing with Java for Dummies; Wiley Publishing; Hoboken, NJ; 2011; 2nd Edition

[BUR02] Burd, Barry; Eclipse for Dummies; Wiley Publishing; Hoboken, NJ; 2005

[BUR03] Burd, Barry; Android Application Development All-in-One for Dummies; Wiley Publishing; Hoboken, NJ; 2012

[FEL01] Felker, Donn; Android Application Development for Dummies; Wiley Publishing; Hoboken, NJ; 2011

[FRI01] Fried, Jason and Hanson, David Heinemeier; Rework; New York, Crown Business March 9, 2010

[FRI02] Friedrich, Johannes; Extinct Languages; New York, Dorset Press, 1957

[FRO01] Froehlich, Christopher; The Complete Idiot's Guide to Android App Development; Alpha Books, 2011

[GOR01] Gordon, Cyrus H.; Before the Bible; London, Collins, 1962

[GOS01] Gosling, James; Arnold, Ken; Holmes, David; The Java Programming Language, 4th Edition; Addison Wesley Professional, 2005

[HIX01] Hixon, Todd; Is Microsoft The Next IBM?; Forbes, August 12, 2013; http://www.forbes.com/sites/toddhixon/2013/08/12/is-microsoft-the-next-ibm/

[IVE01] Iversen, Eirk; The Myth of Egypt and Its Hieroglyphics in European Tradition; Princeton, NJ; Princeton University Press, 1993

[LAJ01] Laja, Peep; 8 Effective Web Design Principles You Should Know, May 17, 2012; http://conversionxl.com/8-universal-web-design-principles-you-should-to-know/#. (Note: The period is part of the link.)

[KOV01] Kovach, Steve; Android Is The Most Popular Smartphone Operating System In The US; Business Insider; http://www.businessinsider.com/kantar-smartphone-market-share-2013-6

[LOW01] Lowe, Doug; Java All-In-One Desk Reference for Dummies; Wiley Publishing; Indianapolis, Indiana, NJ; 2005

[MAH01] Mahapatra, Lisa; Android Vs. iOS: What's The Most Popular Mobile Operating System In Your Country?; International Business Times; November 11, 2013; http://www.ibtimes.com/android-vs-ios-whats-most-popular-mobile-operating-system-your-country-1464892

[MSD01] Moore, Chris; Do not religiously follow the '3 click rule'. But also think about SEO; November 22, 2010; http://blogs.msdn.com/b/searchblog/archive/2010/11/22/get-more-links-do-not-religiously-follow-the-3-click-rule.aspx

[NOW01] Nowak; Microsoft vs. IBM: a tale of two divergent PC companies; Canadian Business, May 29, 2013; http://www.canadianbusiness.com/technology-news/microsoft-vs-ibm-a-tale-of-two-divergent-pc-companies/

[ORA01] Oracle Corporation; The Java Tutorials; http://docs.oracle.com/javase/tutorial/

[RAT01] Ratabouil, Sylvain; Android NDK Beginner's Guide; Packt Publishing, 2012

[YON01] Yong, MK; Android wrap_content and fill_parent example, http://www.mkyong.com/android/android-wrap_content-and-fill_parent-example/

Opinions & Observations

Apple vs. Google
$25 vs. $90
the design police vs. EULA
tall men shops will always be there
•learn to make layouts first
•play with rinky dink java pgms to see results
small medium enterprises will want Apple before Google
joke multi-lingual
graphical debugging & joke graphical debugging