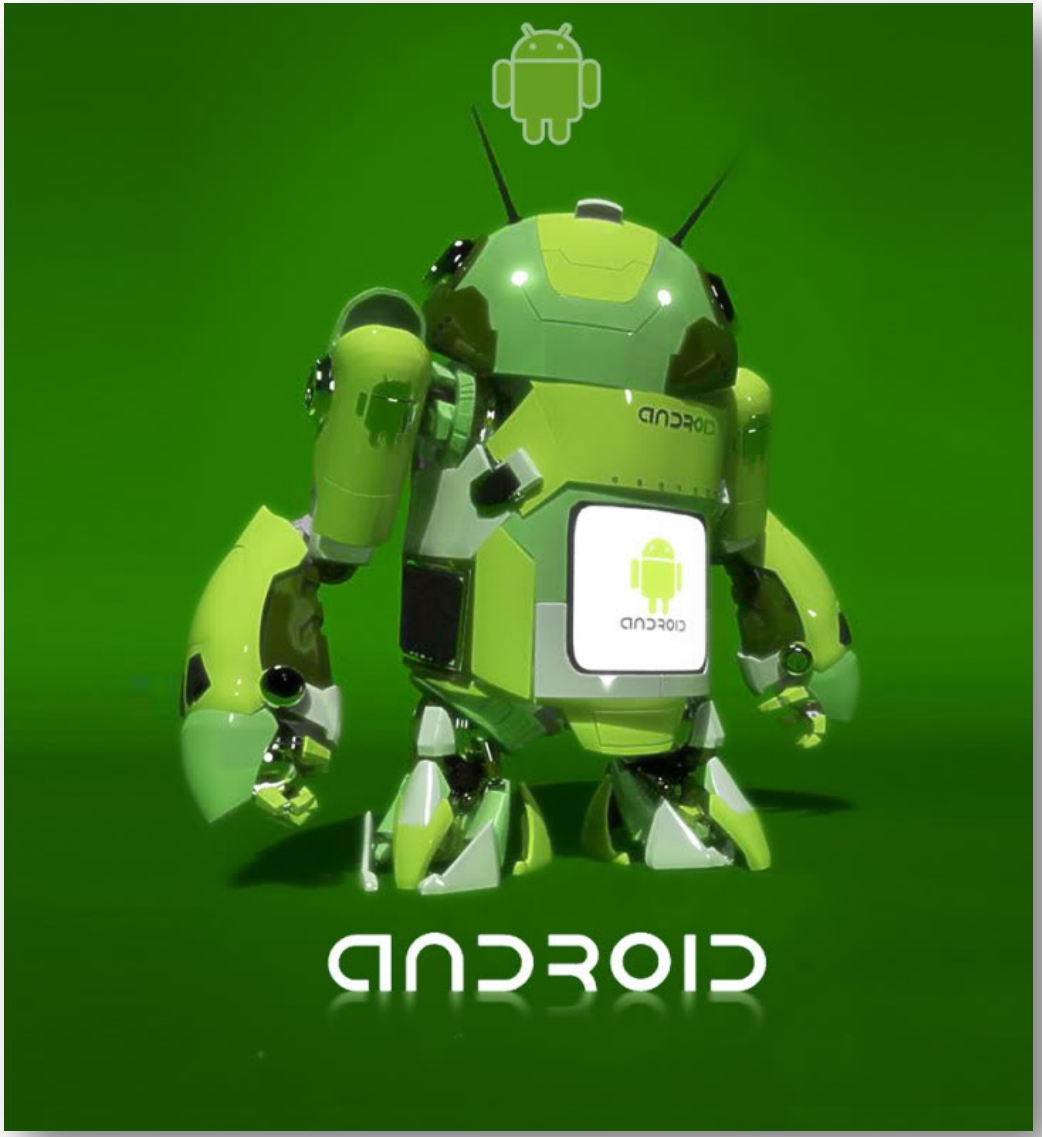
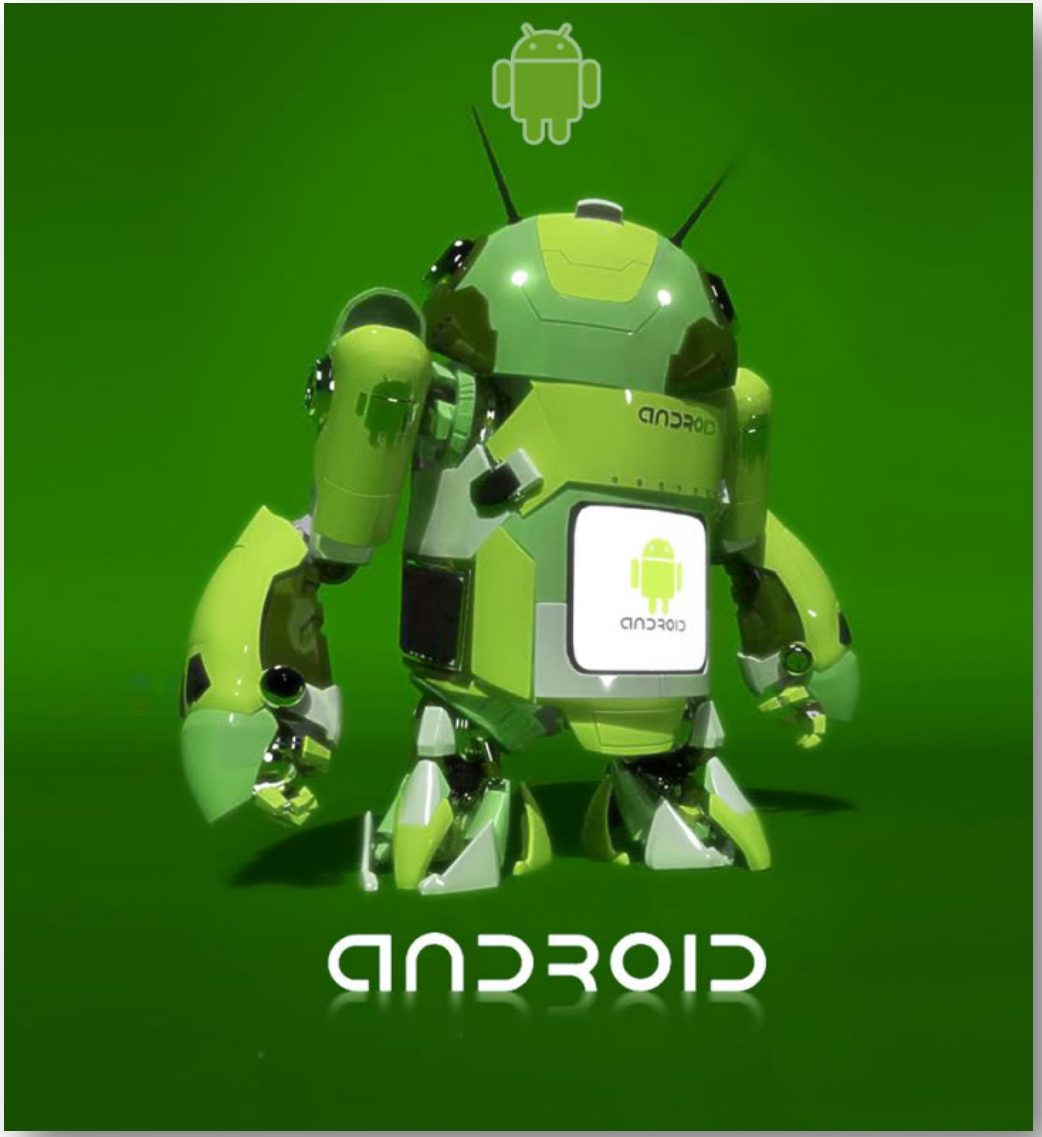
Slide Set 1

FIT2081 – Mobile Application Development
Stephen Huxford
MONASH University
FIT, Clayton

- Unit Administration
- Why Native Android Mobile Apps?
- Java Revision



- Unit Administration
  - Staff
  - Unit Guide
  - Proposed Schedule
  - Assessment
  - Delivery Mode
  - Resources
  - Some IMPORTANT NOTES!!!



Teaching Team

- Lecturer (Clayton)
  - Dr Nawfal Ali
    - nawfal.ali@monash.edu
      - Begin email header with “FIT2081”
  - Consultation (Clayton)
    - Any time with an appointment
    - Crisis meetings as required ☺ or maybe ☹
- Tutors, Lab Tutors (Clayton)
  - TBA by week 2
- Tutor Consultation (Clayton)
  - TBA by week 2
- **Role Account Email**
  - **Send all your concerns, issues, and bugs to the following email:**
    - FIT2081.Clayton-x@monash.edu

Unit Guide Synopsis

- "This unit introduces an industrial strength programming language (with supporting software technologies and standards) and object-oriented application development in the context of mobile application development for smartphones and tablets. The approach is strictly application driven. Students will learn the syntax and semantics of the chosen language and its supporting technologies and standards and object oriented design and coding techniques by analysing a sequence of carefully graded, finished applications. Students will also design and build their own applications."

Proposed Schedule

Week	Date
1	Admin, mobile applications and mobile app development, Java recap
2	Android, Android Studio, project structure, AVD and SDK manager, API levels, platform fragmentation, device fragmentation backward and forward compatibility (the support libraries)
3	Lifecycles, persistence, multiple activities, intents APIs and how to work with them
4	Simple UI: layouts, constraint layout, layout editor, XML, screen size and resolution, Views, ViewGroups (good example of class hierarchy and inheritance)
5	Advanced UI 1: FAB, SnackBar, ListView, App Bar, Options (overflow) menu, NavBar, Toast
6	Advanced UI 2: RecyclerView, CardView, AppBar, Fragments, , Resource Selection (e.g. screen size), Why are API so complicated?
7	DB app local: SQLite
8	DB device shared: Content provider, Rooms
9	Accessing h/w sensors 1
10	Accessing h/w sensors 2
11	Interaction: tap, double tap, gestures, multiple touch, file system (gallery)
12	Recap and Google Play Store

FIT2081 Assessment

ASSESSMENT	% OF FINAL MARK	NOTES
Exam	60%	• Closed book (2 hours) see next slide
Non-Exam		
Pre-reading Quizzes	10%	• 2 attempts, infinite time, score but no answers • 1 per week • Best 10
Workshop Quizzes	10%	• 1 per week • Best 10
Tutes	0%	
Labs	20%	• 1 per week • Best 10

You cannot be marked if you are not in your allocated (as in Allocate+) Workshop or Lab.
You cannot be marked as present if you are not in your allocated (as in Allocate+) Tutorial.

Assessment - Exam

- Exam (60%)
  - 2 hours
    - Just to give you enough time
  - Short answer questions
    - On concepts, uncontextualized code + …
    - A few words, no more than a few sentences
  - Code/Markup Questions
    - You will be asked to comment on/correct presented Code/Markup fragments
    - You will NOT be asked to write Code/Markup completely from scratch (you do this in the labs)
      - Fixing errors and small insertions may be required
  - Sample exam questions will be posted towards the end of the semester

Assessment – Non-Exam

- Non-Exam (10%)
  - Pre-reading Quizzes (best 10)
    - Start from Weeks 2
    - These quizzes test if you have done the pre-reading and have at least a surface understanding of the week's content
    - Details
      - Released before we deal with their content
      - Moodle MCQ
      - Number of questions varies but no more than 10
      - Two attempts allowed, infinite time to quiz close
      - Correct score reported immediately
      - Correct answers only published after quiz closes

Assessment – Non-Exam

- Non-Exam (10%)
  - Workshop Quizzes (best 10)
    - Start from Week 2
    - These quizzes test if you have a deep understanding of the week's content
    - Details
      - Discussion with peers (direct or via social media) is encouraged
      - Use of smart devices to get to the Web is encouraged
      - Usually about 5 questions requiring a few sentences to answer or possible a little code to written
      - Tutors will mark ASAP and publish results on Moodle

Tutorials

- The solution to the upcoming Lab will be discussed in detail
- A chance to ask your tutor questions about the week's work in a less intimidating setting than the workshop
- A chance to discuss approaches to the lab and to discuss the week's material with other students
  - I encourage students who are coping to share their knowledge with other students

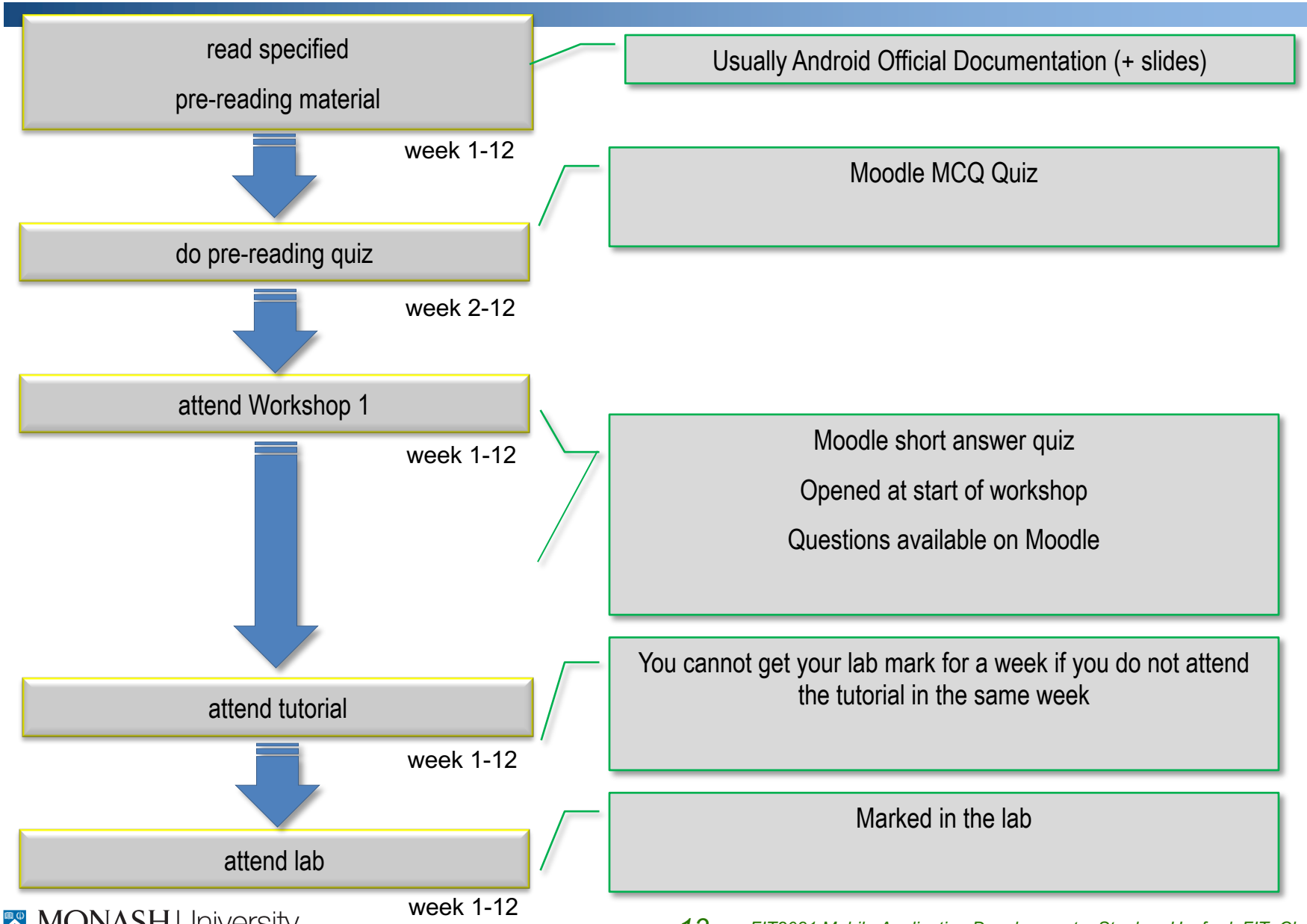
Assessment – Non-Exam

- Non-Exam (20%)
  - Labs (best 10)
    - Assessments start from Week 2
      - Your Lab tasks will be coding exercises
    - You will either code your own app or modify an existing to a given specification
      - In the latter case a good understanding of the existing app's code and functionality are essential
    - Unless otherwise indicated Lab work will be assessed at the end of the Lab session
      - Your tutor will ask you to implement an extra task
      - AND/OR your tutor will ask you a few questions about the app being coded or modified
      - Presenting code you CANNOT confidently explain attracts a mark of 0 immediately
    - See important notes on Labs later in this slide set

Workshops

- Workshops will include
  - Live coding
  - Q and A on the week's content
  - Workshop quiz
  - Some presentation of material

The Week in a Flipped Classroom

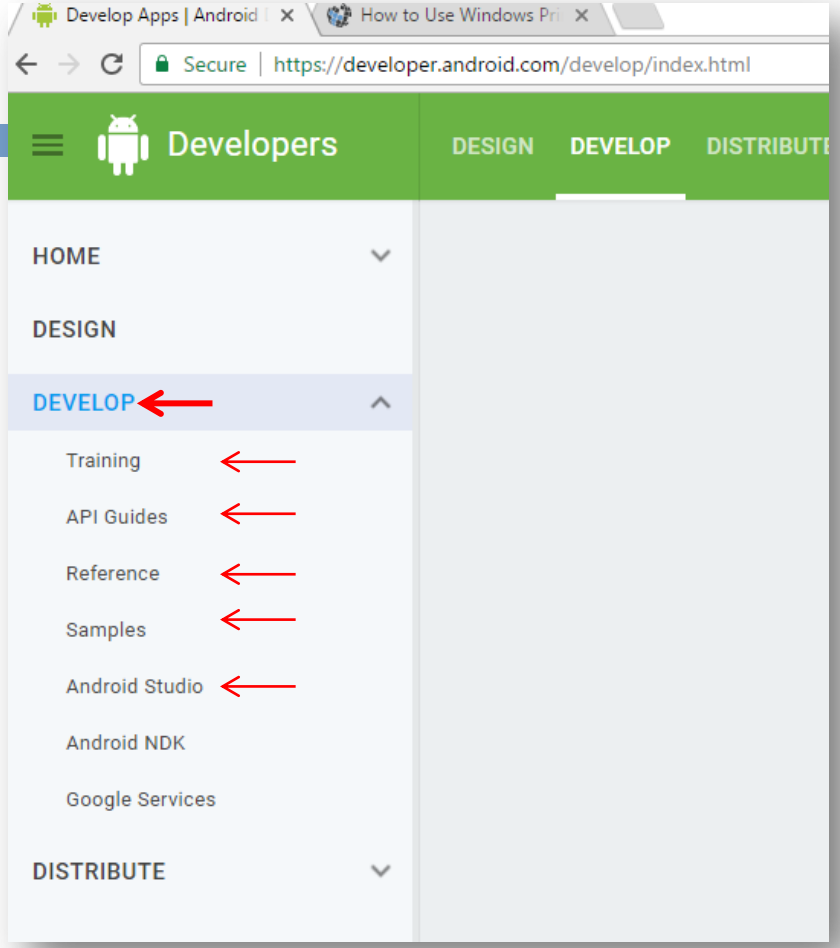


Resources

- Slide Sets and Code Notes
  - Weeks 1 – 4
    - Introduction to Android and Android App Development
    - Detailed slide sets / webpages
  - Weeks 5 – 12
    - Coding specific apps to demonstrate the fundamentals of Android app development
      - Android is huge so we can only cover the fundamentals
    - Detailed slide sets / webpages
    - Code notes to accompany each application
    - The code itself (and any comments it contains)
- Required pre-reading
  - In a traditional classroom this would be summarized on a set of lecture slides which would then be presented to a theatre of half awake students
- Lab - given and modified code
  - You need to understand, in detail, every line of code
  - Any of it can be examined in quizzes and the exam
- The quizzes (pre-reading and workshop)
  - They identify what is most important and what level of understanding is required

Resources

- developer.android.com
  - Especially
  - This is a huge documentation set
  - Sometimes containing the exact answer you seek
  - Often maddingly difficult to understand
  - You must be brave and/or ask Stack Overflow
- Stack Overflow
- Your tutor, your peers, Me



Documentation

- Many students will struggle with the official Android documentation and other prescribed pre-reading
- Learning to deal with such documentation is a fact of life you will face in the workplace/higher degrees
  - Whether it's learning some obscure coding language or environment or researching banking governance policy across multiple national domains or creating a matrix of ERP functionalities the ability to not panic and get what you can from difficult documentation is the same
- Shouldn't I as the leader of the unit be translating this stuff into something you can understand?
  - No, I should be preparing you for the workplace/higher degrees
  - I'm teaching you to fish not giving you fish

Notes on Labs – This is CRITICAL

- Labs
  - You will not be able to complete your lab work in a lab session (not even close as the semester progresses)
  - It is assumed in Labs that you have already completed the required lab work (after a thorough preparation in your tutorial) and just need marking OR
  - You have hit some roadblocks (despite your thorough tutorial preparation, it can happen) and need your lab tutor to help you around them
  - Can I remind you a unit should take 12 hours of your time per week (this is the Monash recommendation)
    - FIT2081 has 5 hours contact per week leaving 7 hours for pre-reading, doing the pre-reading quiz, tutorial and lab preparation, attending a tutor consultation session

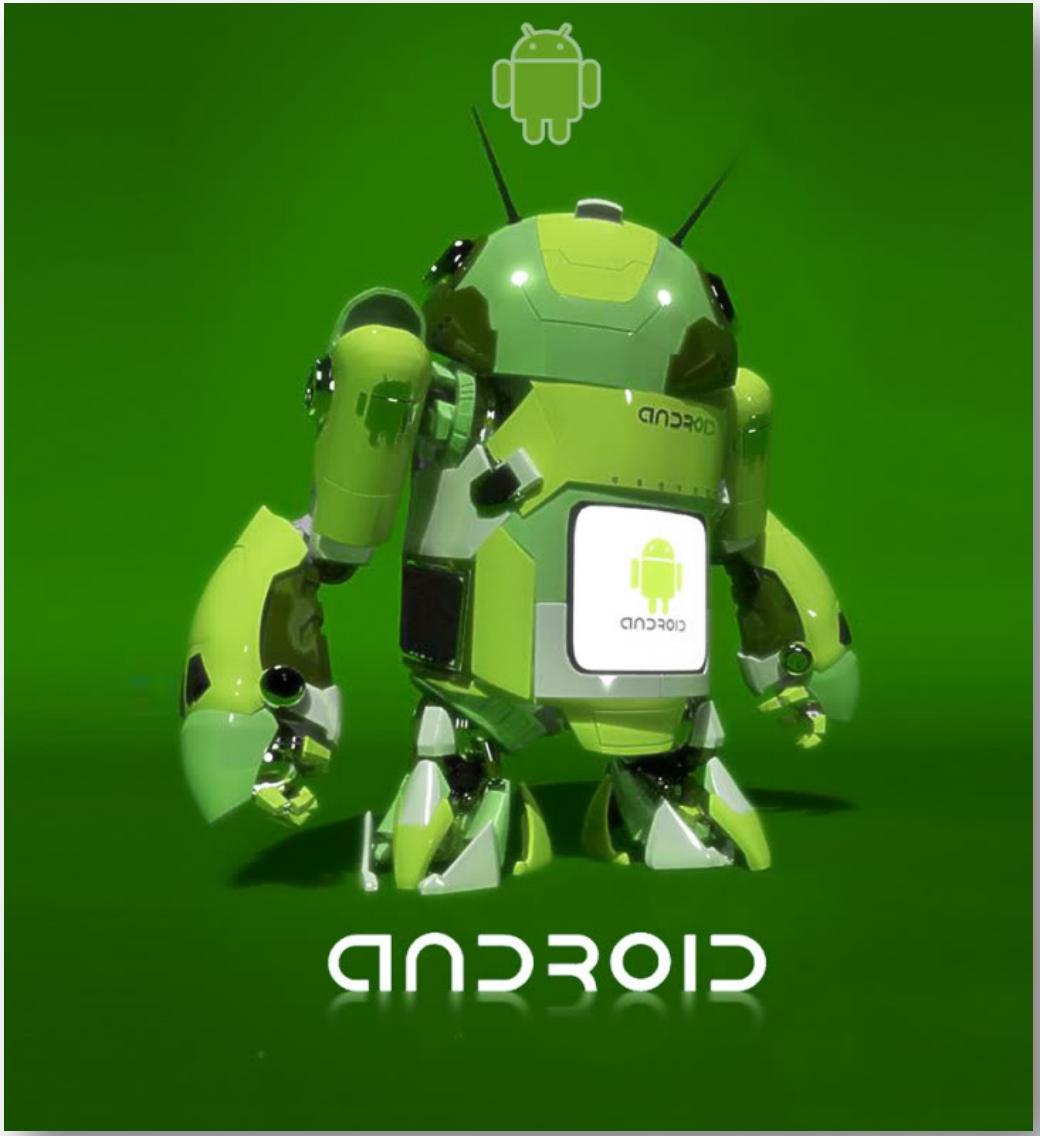
Notes on Labs – This is CRITICAL

- YES!
  - Lab tutors will prioritise students who have done some work and hit a roadblock AND can describe what they have done and what the problem is
    - This does not include “My roadblock is I have done no preparation or work and therefore cannot get started.”
  - If you have completed the lab work you can be interviewed at the lab tutors earliest convenience and leave
- NO!
  - You cannot get marks for presenting code you do not completely understand
    - i.e. you cannot fully and confidently answer any question on the code that you tutor asks you
  - You cannot argue with lab tutors about the marks they give
    - Come see me if you have an issue.
  - Tutors cannot mark students who are not in their Allocate+ scheduled lab

Final Admin Notes

- I very much want to teach you this exploding application development area
- I have spent hundreds of hours selecting pre-reading material and preparing app to code or modify not to mention testing the lab set up
  - By the end of this unit you will be able to program non-trivial Android Apps
- BUT
  - If you are an average student you will need the 12 hours each week (and you will have to attend all workshops, tutorials and labs)
  - If this unit is core for you and you are not a strong programmer
    - You need a support plan e.g. a weekly meeting with a tutor in tutor consultation, a study group with peers, etc.
  - If you are a strong programmer
    - Android is a non-trivial application development environment, that why Android developers are relatively are and command very high salaries
    - Be challenged and have fun

- Why Native Android Mobile Apps?
  - Mobile App Types
    - And our choice of type
  - Mobile App Platforms
    - And our choice of platform



Sources

- The following are just some of the sources I used to construct the following discussion on Mobile Application Types

Source: http://www.nngroup.com/articles/mobile-native-apps/

Source: http://blogs.telerik.com/appbuilder/posts/12-06-14/what-is-a-hybrid-mobile-app-

Source: https://developer.mozilla.org/en-US/docs/WebAPI

Source: http://wiki.developerforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

Source: https://developer.mozilla.org/en-US/docs/WebAPI

https://www.safaribooksonline.com/library/view/building-hybrid-android/9781449361907/ch01.html

https://developer.mozilla.org/en-US/docs/Archive/Firefox_OS/Quickstart/For_mobile_developers

https://cordova.apache.org/docs/en/latest/guide/overview/

https://webrtchacks.com/webrtc-hybrid-applications

Source: www.androidauthority.com/html-5-vs-native-android-app-607214/

Source: http://www.developer.com/ws/proto/the-three-types-of-mobile-experiences.html

Developing Mobile Apps

- There are 3 Fundamentally Different Types of Mobile Apps
  - They require different development skills and platforms
  - They have different capabilities, advantages and disadvantages
    - These are constantly changing as development in this extremely dynamic space continues at a pace that can easily overwhelm IT professionals
- The Three Types are
  - Native Apps
  - Mobile Web Apps
  - Hybrid Apps

Terms

- SDK = Software Development Kit
  - A bundle of all the software components necessary to develop and deploy on a given development platform (usually does not include an IDE)
  - We will use:
    - The Java Software Development Kit (JDK)
    - The Android SDK
      - Which uses the JDK to compile Java classes (step 1 in a 2-step compilation)
- Class Library aka Application Programming Interface (API)
  - Code that we do not write but can call (execute) to perform common but complicated tasks
    - As we will see later it's mainly in the form of classes which we can use to instantiate objects then invoke class methods on these objects to accomplish these common but complicated tasks
    - The public methods of these classes form the API of the library
  - These libraries are generally huge and perform all manner of tasks from creating a responsive UI to interacting with hardware like GPS and accelerometers etc.
- Integrated Development Environment (IDE)
  - A software environment that contains and/or orchestrates all the tools developers need to develop applications
  - e.g. very smart, language sensitive code editors, interface designers, debuggers, device telemetry monitors, device emulators, version control, etc.

Native Apps

See slide 27 for a diagram

- Characteristics
  - App's compiled code runs directly on a device's platform
    - e.g. Android, iOS
  - Built using the SDK tools and languages provided and recommended by the platform vendor
    - e.g. Using Java (or more rarely c/c++) with Android Studio sitting on the Android SDK sitting on the JDK
      - This stack has been deployed for Windows, Mac OSX and Linux
    - e.g. Objective C/Cocoa or Swift with the XCode IDE
      - This stack only runs on Mac OSX

There are cross compilers that can take code in C# say (Xamarin does this) and compile it into a Native App for a given platform. Are these Native Apps?

Mobile Web Apps

See slide 27
for a diagram

- Characteristics
  - A Web site (built using any Web site development technology) designed for smart device displays and accessed (like any Web site) by a device's browser

Mobile Web Apps

The point is be wary when comparing Mobile Web Apps with Native and Hybrid Apps unless you are totally across all developments in Web application development space

- Ongoing Developments
  - The Web application development space is ridiculously dynamic
    - Every week it seems some new, game-changing HTML5/CSS3 feature or JS library is released
    - These all change what is possible with a Mobile Web App
  - Some Examples
    - HTML5/CSS3 advances make it possible for a Mobile Web App to look and feel like a Native App
    - Improvement in JS execution speeds make it possible to approach the performance of a Native App
    - With a few meta elements in a Web page's header the page can be rendered full page without any browser décor and an icon placed on the devices home page that will launch the app in a way indistinguishable from a Native app
    - Browser Local storage and application caching make off-line operation possible
    - Browser JS → Native API bridges are improving fast allowing, for instance, a Mobile Web App to access a device's hardware
    - The latest technology is Service Workers which will allow push notification to a Mobile Web App - the holy grail of App engagement once thought only possible in a Native App
    - Chrome remote debugging tools allow debugging of client side JS code on the device once only the domain of native apps

The quality, completeness and cross-browser consistency of such bridges is critical to Mobile Web Apps

Hybrid Apps

See slide 27 for a diagram

Hybrids try to get the plusses of Native and Mobile Web apps without their minuses. That why they exist as a type of Mobile app.

See the subsequent slides (33 → …) on Mobile app issues to understand this.

- Characteristics
  - Written using a language and development environment other than the recommended languages for the platform but deployed as a Native App
  - Two main types:
    - A thin native app shell contains a Web app
      - The shell contains a JS → Native API bridge
      - The shell contains a native component that interacts with the platform's Web rendering engine to maintain a UI
      - e.g. Cordova/PhoneGap, Trigger.io, Ionic, and Sencha
    - A cross compiler is used to convert code into a Native App executable for each required target platform (is this a Native app)
      - e.g. Xamarin, Appcelerator, Embarcadero FireMonkey, or RubyMotion
      - It's possible for developers to be able to call through transparently from their own language to the underlying platform's API giving complete native access
      - What about UI designers and other development tools?
      - Single code base: Yes! Simplicity of a Web App: No!

Hybrid Apps

Building Hybrid Android Apps with Java and JavaScript: Applying Native Device APIs:

by Nizamettin Gok, Nitin Khanna

- More
  - It's actually possible (although not easy) to build a Hybrid Web App without any 3rd party support
  - But usually a 3rd party such as Xamarin and Ionic provide a complete IDE that:
    - Critically uses a single code base to deploy to multiple platforms
    - Creates and constantly updates the JS → Native API bridge in the thin native client it deploys on each platform
      - The quality, completeness and cross browser consistency of such bridges is critical to Hybrid Apps

Some Important Issues

*depends on Browser or thin native client consistency. UI and Native API access can be problematic. So, write once and a "bit".

- Companies Absolutely Require a Mobile Presence
  - This usually means a presence on both major mobile platforms
- Native Apps
  - Requires scarce, relatively technically sophisticated, costly developers, one set per target platform
  - Developers have complete access to the device's features
- Mobile Web Apps
  - Requires abundant, relatively less sophisticated, cheaper Web developers,
  - The same "App" functions on all platforms (write once)*
  - Developer access to device features and the Native API in general depends on the quality, completeness and cross browser consistency of the browser's JS → Native API bridge
- Hybrid Web Apps (the enclosed Web app type)
  - Requires abundant, relatively less sophisticated, cheap Web developers,
  - The same "App" functions on all platforms (write once) that the hybrid app IDE creates thin clients for *
  - Developer access to device features and the Native API in general depends on the quality, completeness and cross browser consistency of a 3rd party's JS → Native API bridge

More Issues

Not any more!!! Google have forked Webkit to create Blink which has already diverged significantly. Render wars anybody!!!

- Cost
  - Write once is cheaper and Web developer skills are cheaper
  - Write once can be over-hyped though
    - Although WebKit (rendering engine) is virtually a standard on Mobile browsers their JS engines wrt JS → Native API are not consistent
    - Hybrid Apps are also at the mercy of the quality completeness and consistency of their IDE's thin native clients for the various target platforms
- UI Look and Feel (L&F)
  - Web L&F is not Android/iOS L&F
  - HTML5 allows development of a sophisticated UI (swiping etc.) but it's often still a different user experience
- Offline
  - In-browser data storage is limited compared to local file storage, does your App require offline functionality?
- Discoverability and Installation
  - App store vs URL and bookmarking (do mobile users bookmark?)
  - Installation versus no installation required

HTML5 + CSS getting closer L&F all the time + with a few html meta tags bookmarking the site places a specified icon on the device's home screen which "launches" the mobile Web app in a native-app like full screen mode

In-browser Local Storage already exists and is improving all the time

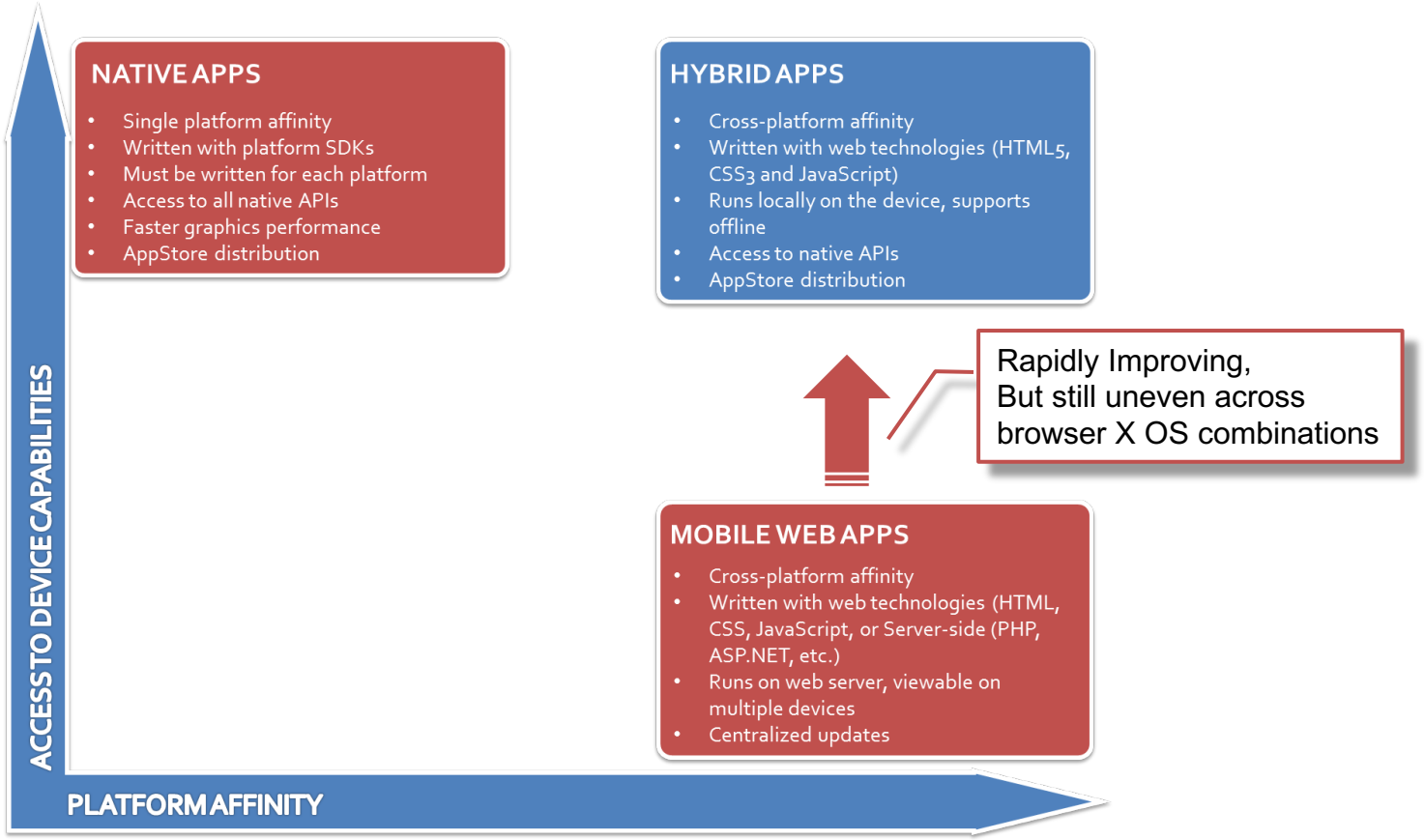
HTML5 now includes application caching

Not required anymore now home screen launch icons are possible

More Issues

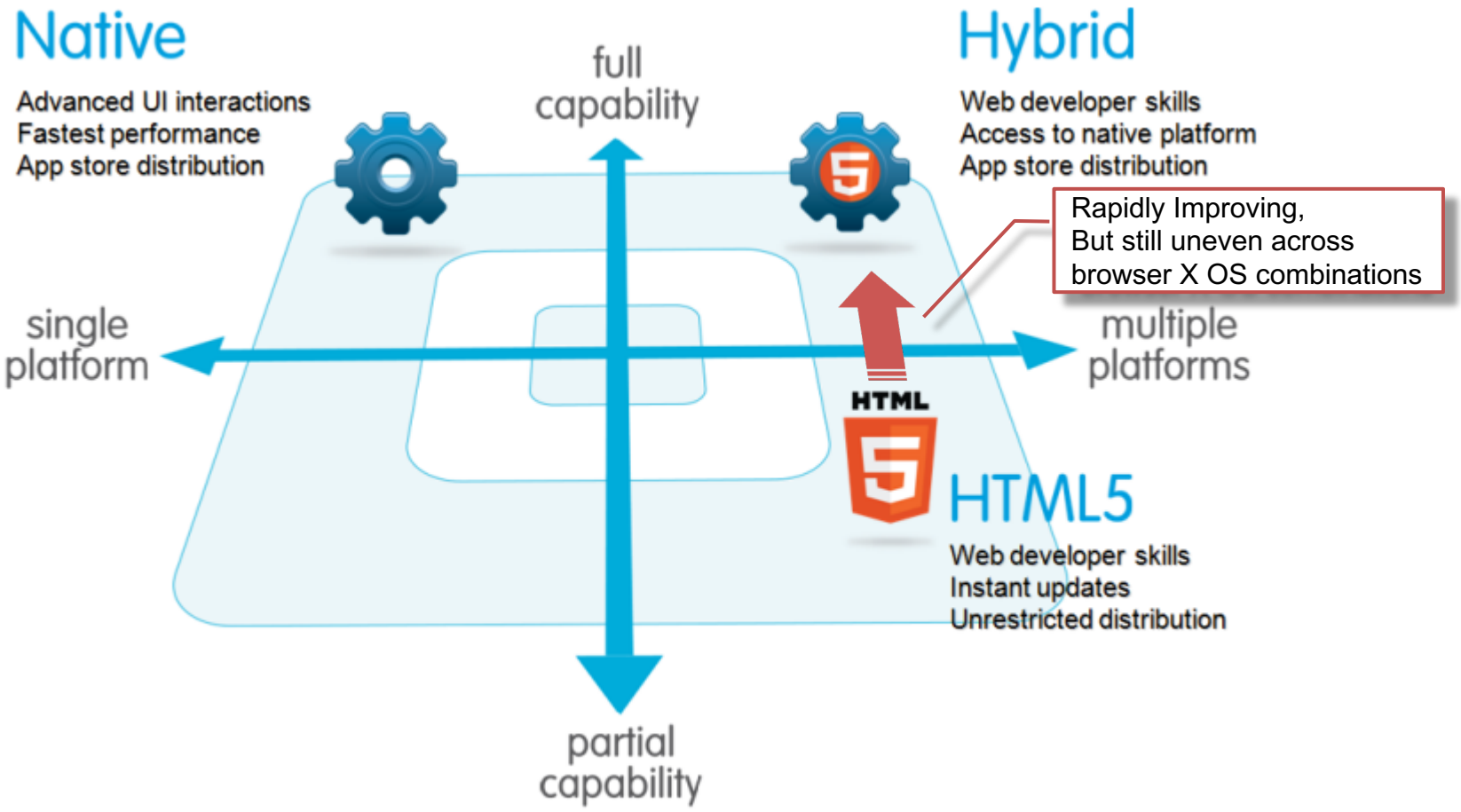
- Speed
  - Native will always shine here and for fluid, complex graphics it's essential
  - Many important types of App do not require this kind of speed
- Security
  - Mobile Web Apps are subject to all the security risks of normal Web Apps
  - Native Apps have none of these risks
- Content restrictions, approval process, fees
  - Anything in an App store (Native and Hybrid) usually shares its purchase price with the store owner and must undergo a lengthy approval process (in the case of Apple's App Store)
  - The Web is free of any of these encumbrances
- Maintenance
  - Multiple platforms to update consistently if not write once, update related delays and re-approval if in an app store (especially Apple's)
  - Web updates are instantaneous to all platforms

Characteristics / Pros / Cons



Source: http://blogs.telerik.com/appbuilder/posts/12-06-14/what-is-a-hybrid-mobile-app-

Characteristics / Pros / Cons



Source: http://wiki.developerforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

Summary

- Surprisingly (to me at least) all 3 types seem to be thriving equally in their own niches
  - btw Native apps can include UI widgets that are essentially embedded browser windows which opens up many possibilities
    - Native and Mobile Web Apps mash-ups
    - e.g. circumventing Apple's approval process
- Something we have not considered is that a poorly designed app is bad app no matter how its implemented
  - And the design requirements for a Mobile application are very different to those for a desktop application
  - This page is a good starting point for this hugely important and large field of study outside the scope of this current unit
    - https://www.thinkwithgoogle.com/articles/principles-of-mobile-app-design-introduction.html

FIT2081 – Which Mobile App Type?

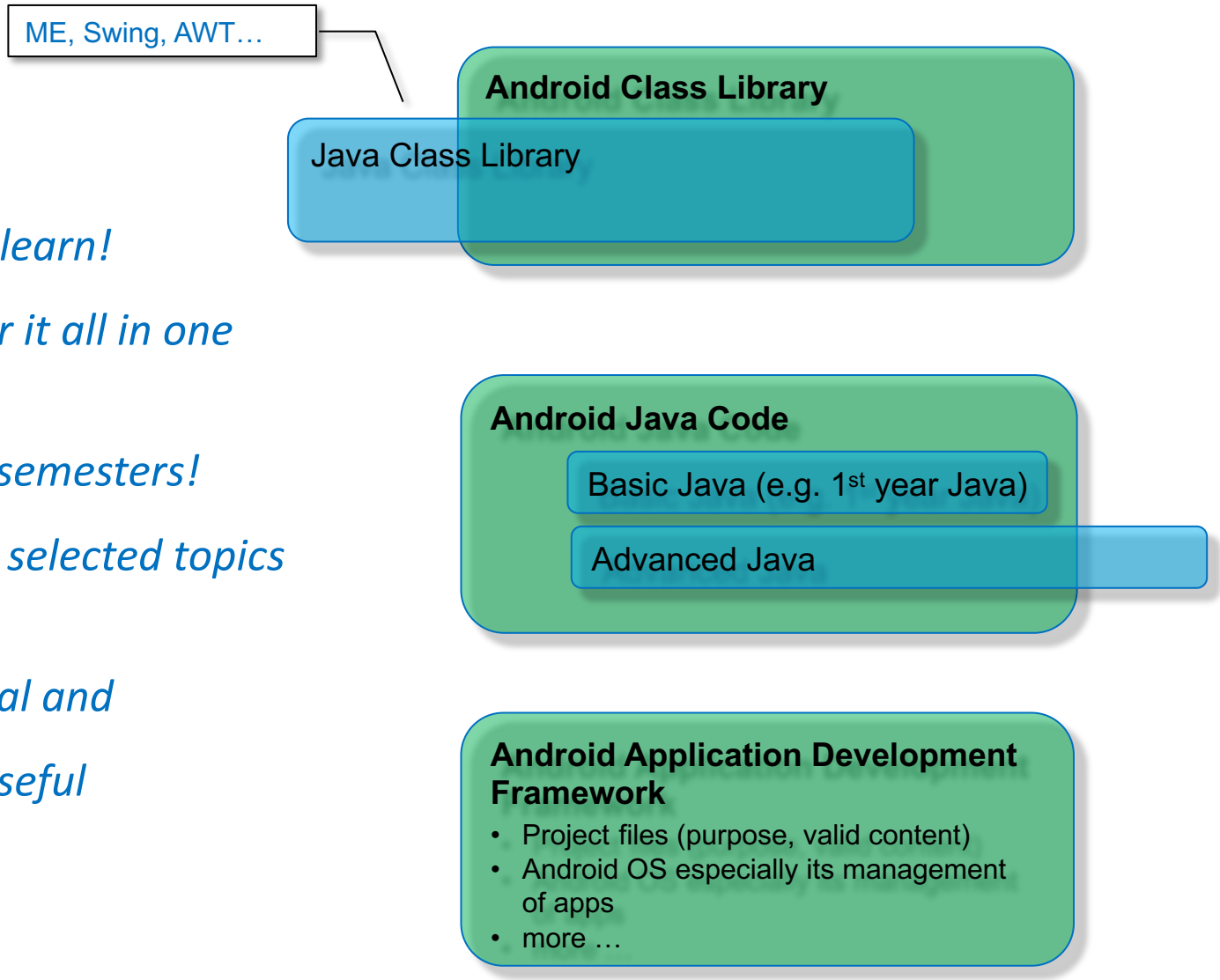
- Native Apps - Why? ✓
  - Free, rock solid IDE's with good device emulations. NOW!
    - For Android this is a very recent development (consider yourselves very lucky)
- Web Apps and Hybrid Apps ✗
  - Are in a state of flux as the technologies they depend upon rapidly evolve
    - For Web apps this is browser implementation of HTML5 (look and feel) and WebAPI (access to device)
    - For Hybrid apps this is the various specialised IDEs especially the quality and consistency of the thin native clients they create for deployment on multiple platforms
    - i.e. they are moving targets so anything I show you could be rapidly outdated/irrelevant which is a waste of my time and yours
  - Mobile Web Apps
    - JavaScript is the standard client side programming language for Web applications
    - It's a slack/quirky, interesting language that requires great programmer discipline if bugs are to be avoided. It also has an unbelievably dynamic/confusing/exciting API space and eco-system
    - It's take on OO is very non-standard and difficult
    - This is not to deny it's a very hot language right now that is even beginning to be used beyond client side Web programming but the JavaScript vs TypeScript battle is also looming
  - Hybrid apps
    - Have to choose a particular IDE + community among many
    - Many skills learned will not transfer to other Hybrid app systems
    - IDEs are not free

FIT2081 – Which Mobile App Platform?

- Android ✓
  - It easily has the highest device count
  - Java is far and away the most coded language in the world so much so it can be considered an industry standard in many application spaces
    - It's very useful to have on your CV
  - Can be developed in Windows, MacOSX, Linux
- iOS ✗
  - Objective C is quite exotic compared to Java in its syntax and terminology and is not widely used outside of the Apple eco system
  - The arrival of Swift overcomes the exotic nature of the required development language but not its general applicability
  - Can only be developed in MacOSX

Looking Ahead – With/Without Previous Java

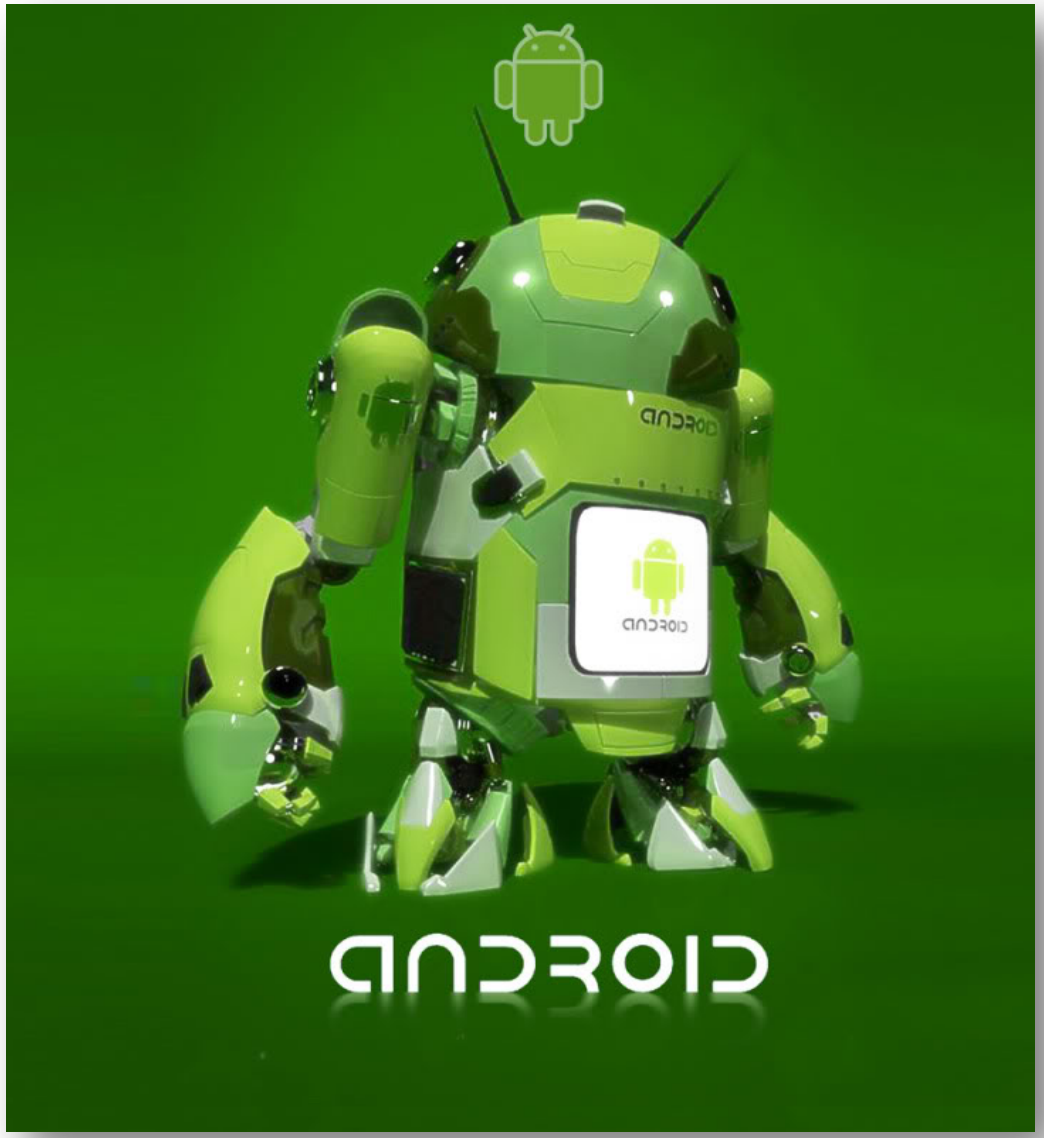
- *There is a lot to learn!*
- *We cannot cover it all in one semester*
  - *Not even 4 semesters!*
- *So we will cover selected topics that are:*
  - *Fundamental and*
  - *Generally useful*



Java Revision

These 3 Android projects are conversions of FIT1051 Java projects and can be downloaded from Moodle

- JavaRevision1
  - Inheritance Revision
- JavaRevision1p5
  - Interface Revision
- JavaRevision2
  - Listener Revision



JavaRevision1 – Inheritance Revision

- Revision Points
  - Super and Sub Classes ("extends")
  - Abstract classes and methods
  - @Override
    - And overriding itself
    - Calling super
    - Calling super.someOverridenMethod(…)
      - Leveraging possibilities
  - Instance variables should be private
    - They are (directly) accessible in subclasses
    - But public methods are so their accessors and mutators can be called in subclasses to manipulate them
  - A class inherits from all its ancestors not just its parent
  - toString(…)
  - Polymorphism using Inheritance (Upcasting)
    - Code
  - Downcasting

Very detailed analysis of the project code in pre-semester Java Revision download

JavaRevision1p5 – Interface Revision

- Revision Points
  - An Interface is a contract or promise (“implements”)
    - A class can only have one super class but can implement multiple interfaces
  - What’s the point?
  - An Interface is a Type
    - Entirely equivalent to a class in this respect
    - Classes and Interfaces can form mixed extends/implement Type hierarchies
      - So, upcasting (and therefore polymorphism) and downcasting to/from Class and Interface types is possible
    - An object of a class can be referenced by:
      - A variable of any of its class’s ancestor types
      - A variable of any the Interfaces its class implements
        - » Or indeed any Interface implemented by any of its class’s ancestor class’s
  - Polymorphism using Interfaces (possible because they are types)
  - Employee implements Payable but does not contain implementing code for getPaymentAmount the only method of the Payable Interface. Why?
  - BasePlusCommissionEmployee’s and getPaymentAmount. Discuss!

Very detailed analysis of the project code in pre-semester Java Revision download

JavaRevision2 – Listener Revision

- The easy way (see the “more” button)
  - Using button widgets onClick property/attribute
- The hard way (see all other clicks)
  - In-place instantiation
  - new someInterface(){…}
    - Makes no sense. Why?
    - How does compile this
  - Anonymous classes
  - Inner Classes (non-static nested classes)
    - Why?
    - Visibility?
  - Activity classes as listeners
    - Pros and cons

Very detailed analysis of the project code in pre-semester Java Revision download