Android Basics Facilitator Guide

Overview

Welcome to the Android Basics Facilitator Guide!

This guide is meant to help facilitators lead a study group for people with no prior coding experience on how to build Android apps. The curriculum presented here is intended to supplement the online Android Basics courses that were developed by Google and Udacity (as part of the Android Basics Nanodegree).

The suggested format of the study group is to have students take the free online courses on their own. Then students can periodically meet in-person to review the content that was taught and have a chance to get their questions answered. However, facilitators may adapt the course and activities as necessary to fit their own group size, frequency and duration.

The guide contains the following parts:

- Final Project
- Android for Beginners: Building Layouts
- Android for Beginners: Interactivity
- Android for Beginners: Object-Oriented Programming
- Android Basics: Multi-Screen Apps
- Android Basics: Networking (Coming soon!)
- Android Basics: Data Storage (Coming soon!)

Thank you for taking on such an important role for your students. For many of them, this is their first introduction into the exciting world of Android, which includes a vast range of apps and devices, a vibrant community of developers, and the fastest growing user base of any mobile platform today. We hope that you find the experience rewarding as you witness their technical skills advance and their confidence grow. These students will feel empowered when they realize that this skill set unlocks a world of possibilities of what they can create for themselves and their communities.

Good luck on the journey!

Before You Start

Pre-requisites

For facilitators who want to lead a class or study group, we recommend prior experience with developing Android apps, in addition to having taken the Udacity Android Basics courses. This would lead to the most effective classroom experience for students, so a facilitator can help clarify confusing topics for students and unblock them when they get stuck.

For teaching assistants who will be circulating the class and helping to answer students questions, we recommend that they have taken the Udacity Android Basics courses at a minimum. Experience with Android development (outside of the course) is preferred, but not required.

For students to take this course, no prior programming experience needed. However, they should have basic digital literacy skills. They should be familiar with basic computer skills:

- Browsing the web
- Using a smartphone with apps
- Installing software on their computer
- Using a document editor like Google Docs or Microsoft Word

Materials & Equipment

The staff (facilitators and TAs) should have computers so that they can look up resources and help troubleshoot student problems.

Each student will need to bring the following items to class. If access to an Android device is not possible, the emulator that comes with Android Studio can be used instead. Note that the Google Play store is not available on the emulator.

- A computer that satisfies these system requirements
- A computer charger
- Android device
- USB cable to connect the device to their computer
- Headphones (if videos will be watched individually in the classroom)

Classroom should ideally have:

- Internet connection
- Projector (for the facilitator to show any slides or demos)
- Microphone (so the whole class can hear a speaker)
- Device projector (so that a physical Android device can be projected to the class)

Community

The most valuable aspect of gathering together in an in-person study group is to allow students to ask the questions to the staff and each other. We recommended creating a mailing list or social media community to make it easier for everyone to communicate. Students can also share screenshots of their work, interesting articles they find, and helpful tips.

Ongoing Classroom Activities

The following activities are optional and can be done in the classroom at any time in the course. For example, you can make them regular activities at every session or do them periodically throughout the class. If you ever need additional exercises, check out this list.

Android Ecosystem Updates

In the pilot run of this course, each session started with an ecosystem update from 1 - 2 students. This is a leadership opportunity for students to teach others, and it exposes the class to new apps and current events.

1. Have a student share their favorite app and what they like about it.

2. Have another student present an article about the latest news in Android development.

Question Box Time

1. Have each student write down a question about the material that they are wondering about. Collect all the questions in a box.

2. Have each student answer at least one of these questions. Have them research more about the topic and come up with a creative way to recap it to the class so they understand it better.

Small Group Reflection

At the end of a session, you may provide students with a chance to reflect in small groups on what they've learned so far. Here are some guiding questions:

- Can you share what you have done so far?
- How are you feeling?
- What has been challenging? rewarding?

- What lessons have you learned so far?

Final Project

Overview

The most fun part of the course is when students get to create their own apps!

The final project is an **educational app that teaches 5 things** to a user about any topic. Review the project prompt to become familiar with the details.

While you could wait until students have completed all coursework before starting the final project, **we recommend having students start on the final project at the very first session**. This will boost their motivation because they get to work on their app idea from the start. For example, they can build the layout for their app early on, and later make it interactive after they've finished learning relevant concepts. You can pace your class according to the stages of the final project.

Scoping the Project

Students will be quite ambitious in their app ideas, which is great. However, as a facilitator, it would help to set accurate expectations on what they will be able to build at the end of this course (so they don't feel disappointed or discouraged).

In the first session of your class, review the feature scope of the project with students. See the project rubric for details.

The app:

- Should teach a user 5 things about a topic
- Can contain text, images, and other Views
- Should be limited to a single screen that can scroll
- Can link to other apps like maps, phone dialer, web browser, etc...
- Should be interactive with buttons

Also point out what is **out of scope** for this course:

- An app with multiple screens
- Storing user's preferences/data on the device
- Getting data from a server
- Syncing data across devices

Some students may struggle with app ideas, especially with the limitations of what is out of scope for the class. That's why reiterating that the final project should be an educational app is quite important. This can frame their mindset in coming up with an idea that fulfills the app requirements. Encourage students to pick a topic that they personally care about (and in turn, are very knowledgeable about). This will result in the most unique and useful apps for their communities. They can also utilize the brainstorming template in the final project prompt.

Feedback on Projects

Another valuable aspect of the in-person experience is being able to provide students 1:1 feedback on their final projects. Help them with picking an app idea of appropriate scope or provide hints if they get stuck on building a feature. It would be nice to conduct 1:1 check-ins with students at several points throughout the process of building their final projects.

If time allows, you may want to provide students with in-class time to work on their final projects. Then the staff can walk around and answer questions.

Final Project Demos

If your budget allows, organize a fun demo and reception event for students to present their final projects to close family and friends.

To prep for the demos, ask each student to provide the app name and a short 2-3 sentence description of the app. Collect all of these together in a handout to pass out to event attendees.

Invite a panel of judges for the event who can provide constructive feedback after each demo presentation. While you may choose to award prizes for the best app in different categories, students may feel discouraged if aren't selected as a winner. You may prefer that judges ask clarifying questions on how students implemented certain features of their app, acknowledge what the student did well, and also provide suggestions on how to extend the app further.

Share their work

If students feel ready, they can publish their app on Google Play. This is a great way to gain experience by having real users.

We'd love to see what apps your students create! They can post their apps on this student discussion forum.

Course 1: Building Layouts (Part A)

Notes to the instructor are shown in italics and in a gray font color.

Demos

1. Show student examples of cards that have been made (but don't show the videos or go into detail of how they built those cards, that will come later). This is what they will be building at the end of Practice Set 1 Birthday card.

2. There will a lot of new vocabulary in this course. The goal isn't for students to memorize everything, but to gain experience using new words in practice. Open up Vocab Words Glossary for this course. Look up words that were introduced in the initial videos: View, TextView, ImageView, and Button.

Exercises

1. **Growth Mindset**

   In small groups, share a time when they each struggled to learn something new. If they don't want to share, that's okay too.

   Learning programming can be intimidating -- but it shouldn't be! We need your help in breaking down student preconceptions that "you have to be a math whiz", or that "you need to have years of coding experience."

   This exercise promotes a growth mindset to help students feel confident that they can tackle the upcoming challenges of Android development, because they already overcame obstacles with learning something new. See the Udacity discussion forum thread for responses from other students.

   The best way to get students to open up about their experiences, would be to share one of your own. When setting up this exercise, tell students that you will begin by sharing an episode from your life when you struggled to learn something new.

2. **Recognizing Views**

   Choose an app on your Android device. Determine what views are being used in a single screen. Share your insights with other students.

3. **XML Syntax**

   Say Sound Effects Together. Based on the XML Syntax video at 0:35, say all the sound effects together as a class. Ding (open angle bracket), Woosh (TextView), Pew Pew Pew (Attributes), Click (forward slash), Boom (closing angle bracket).

   Even if you feel silly doing this, this is important because the sound effects are used as a mnemonic device to help them remember proper XML syntax. Otherwise they will get confused on why their XML is incorrect.

4. **Fixing Invalid XML**

   Open up Android Visualizer. Cover your eyes and invite a student to come up to the projector. Have them make one change to your code that breaks your XML (they can delete a character, or add one somewhere). Then try to find it. Explain your reasoning process to the students.

5. **Fixing Invalid XML**

   Create invalid XML to stump your classmate. Write XML for a TextView that intentionally contains several errors. Then give it to a fellow student and have them fix all the errors until it is valid XML again. The trickier, the better! :)

   Students can hand their computer to someone else, or send email code snippets to each other. Examples from past students: 1, 2.

6. **Troubleshooting errors** (Optional)

   Do you remember suggestions in the video for what to do when you get stuck? Any other advice on what to do when you get stuck?
   - Read error messages
   - Compare to working code
   - Undo
   - Ask for help

7. **Syntax errors** (Optional)

   Have a class discussion about why are syntax errors so common? Why does XML have to be so exact?

   Well, computers have different strengths / weakness than people. For example, computers are great at doing simple calculations, very fast. But they're terrible at

understanding what you mean. A programming language is a language BOTH humans and computers can understand.

8. **XML Quiz** (Optional)

   As an individual exercise, have students check out this XML tutorial and you'll see that this markup language is used in other use cases besides Android layouts. Afterwards, if you're ready for a challenge, try this XML quiz!

9. **Density-independent pixels**

   Discuss in small groups. In your own words, describe what a dip, or density independent pixel is.

10. **Density-independent pixels for devices**

    For your Android device, can you find out the dimensions of its screen size *in dips*? Search online or you can find some devices in this Device Metrics list (see "Width x Height dp" column).

11. **View width/height** (Optional)

    Discuss in small groups. Describe a situation where you would want to use wrap_content instead of specifying the width in dips.

12. **Font size in scale-independent pixels**

    Take out your Android device. Change system font setting on your device. Go to Settings app > Display > Font size. Change the default font size on the device. Then open any app of your choice. See how the user interface changes with the smaller or larger font size. How does it look? Does the app handle the different font size gracefully or does information get cut off?

13. **Accessibility** (Optional)

    Discuss in small groups. Will all your users have perfect eyesight? How does this relate to using "sp" (scale-independent pixel) values to specify your font size?

    You can refer to this page in the Material Design spec on accessibility.

14. **Android Development protips on social media**

Find an Android development protip on your favorite social media site and share what you learned with other students. Protips are short tips and you can find a bunch of them on G+ or Twitter under the hashtags #AndroidDev #Protip.

This exercise teaches students to learn from sources outside this class - namely, from other developers in the Android community. Social media is a great way to stay up to date on the latest news in Android development. You can also encourage students to post their own Android development protips on new things that they've learned. This reinforces their own learning and other people will start to see your students as useful sources of Android information.

15. **Learning from documentation** (Optional)

From looking at the TextView or ImageView Android documentation pages, ask students to find a new attribute. Look at the documentation page together and explain out loud how you go about figuring out what that attribute means.

Doing this exercise in front of the classroom requires more Android development expertise. It may be hard for students to grasp what this new attribute does without seeing it. The Android XML visualizer likely does not support this additional attribute, since it was built by Udacity specifically for the topics covered in this course.

Course 1: Building Layouts (Part B)

Exercises

1. **View width/height**

   Have everyone stand up. Stretch out your arms to represent match_parent. Hug yourself to represent wrap_content. Put your hands shoulder width apart to represent fixed dp.

2. **Horizontal vs. Vertical LinearLayout**

   Split up into small groups. Assign different ViewGroups to each small group: vertical LinearLayout or horizontal LinearLayout. Ask each group to come up with their best representation of the given ViewGroup. For example, you can suggest that a student represents a single View - TextView, ImageView, or Button.

   One solution is to demonstrate horizontal LinearLayout by having students stand shoulder-to-shoulder beside each other to simulate a "row of views." To demonstrate a vertical LinearLayout, students can stand in front of each other in a single line to simulate a "column of views."

3. **LinearLayout Weight**

   Continuing from exercise #2, ask the students to demonstrate LinearLayout weight. Again, they can have a student represent a single View.

   A possible solution is that there are 3 students are arranged in a row (simulating 3 TextViews in a row) but one student has their arms spread out very wide, indicating that they have a large weight value assigned to them. Hence, they take up the remaining space available in the parent ViewGroup.

4. **RelativeLayout**

   Continuing from exercise #3, ask the students to demonstrate RelativeLayout. Again, they can have a student represent a single View.

   A possible solution is that one student stands in a spot that is assumed to be the middle of the parent ViewGroup. Have another student stand beside them. Have another student positioned relative to that. Explain that the initial student is positioned in the

center of the RelativeLayout, and that the remaining students are positioned relative to (to the left of, to the right of, etc..) that initial student.

5. **ViewGroups** (Optional)

   Point students to StackOverflow and have them search for a question on ViewGroups. Is there a question that they understand and could have answered themselves?

6. **Padding vs. Margin**

   In small groups, discuss the difference between padding and margin. Can they find examples in apps or screenshots online that would illustrate the point?

   Setting padding or margin on a Button view may make the concept clearer (as opposed to setting it on a TextView).

7. **Metrics and Keylines in Material Design** (Optional)

   Have students take out their Android device. Install the Keyline Pushing app. Turn on the baseline grid and open up your favorite app. What do you observe? Read about metrics and keylines from the Android Design Guide.

8. **Create rap song** (Optional)

   Come up with a rap song to summarize what you learned in this lesson. For example, a rap song about ViewGroups: "If you want 'em in a row, LinearLayout you should know. If you want 'em scattered around, RelativeLayout is how it's going down."

9. **Material Design Guidelines** (Optional)

   Split up the sections among the class so each person can read a part of the guidelines. Everyone should read the intro / principles section. Then everyone should give the rest of the class an overview of their section.

   The goal is to get everyone familiar with the major parts of the guidelines. Students don't need to remember all the details, but they should know that parts of the guidelines exist, so that they can refer back to them later as needed.

Course 1: Birthday Card (Practice Set)

You can either have them install Android Studio and build their birthday card app in the classroom or at home.

Demo

- Show how to build a “Hello World” app in Android Studio using the “Empty Activity” template. Run the app on a device and on the emulator.

Exercises

1. **Android Studio Install Party**

   Have Teaching Assistants available to help students with problems with the JDK or Android Studio install process.

   For any issues that you run into, try to search online for other users with the same problem. Once you’ve exhausted those options and are still stuck, you can provide feedback to the Tools team here.

2. **Protips on Android Studio**

   Read and share tips on using Android Studio. Write up your own post and share on social media. For example, see Google Developer Expert Philippe Breault’s Android Studio pro-tips on G+ by searching for #androidstudioprotip or checking out his blog.

3. **Import sample** (Optional)

   See instructions to “Import sample from GitHub” directly into Android Studio. Run it on your device. The samples are great resources if you want to see an example of how to use a specific Android API.

4. **Android Studio Videos** (Optional)

   Search YouTube for more videos on how to use Android Studio. Become familiar with the most commonly used features in Android Studio. For example, you can search for “Android Studio tour”. Or you can watch DevBytes on Android Studio on the Google Developers YouTube channel.

5. **Android Developer Summit Videos** (Optional)

   Check out these talks by the Android Studio tools team from the 2015 Android Developer Summit: What's New in Android Studio and Android Studio for Experts.

6. **Deep-dive into Android Studio** (Optional)

   Have students research answers to these challenging questions. What is an AVD? What is inside an apk file? How does Android Studio create an apk file? What is gradle?

7. **Birthday Card Demos**

   Ask the class for volunteers to share their birthday card with the class. Have them describe how they built the app with different Views and ViewGroups.

8. **Birthday Card Code Review**

   Pair up with a classmate and provide feedback on each other's code for their birthday card.

9. **Give Feedback on Birthday Cards**

   Look at what other students have created on the course discussion forum. We can use the following prompts to structure student response:
   - Pick one student submission on the forum
   - Identify things they they did well
   - How can they improve

Course 2: Making an App Interactive (Part A)

Exercises

1. **Research** (Optional)

   What is an activity?
   Read more about the file structure of an app.
   Why do statements in Java end with a semicolon.
   What else can you do in DDMS?

2. **Create an error and fix it**

   Open Android Studio to the MainActivity.java of Just Java project it to the class. Have 2 students stand up. One student looks away momentarily. One student intentionally creates an error in the code. Have the other student try to fix the problem in front of the class.

3. **Variables**

   Search for and read a java tutorial on variables.

4. **Integers**

   Find an app on your phone that likely uses an integer variable to keep track of some piece of data. For example, an integer can be used to keep track of the number of new email messages you have in your inbox. Describe what value the variable is initialized with and what it gets updated to.

5. **Button Click Behavior**

   Modify the button behavior so that something else happens when you click on the “+” or “-” buttons in the Just Java app. Feel free to search online for tutorials to figure this out.

Course 2: Making an App Interactive (Part B)

Exercises

1. **Nested View Groups**

   Pick an app. Use the Hierarchy Viewer tool in Android Studio to see all the views in the layout for the current screen.

   Student-Directed Learning. Pick an app. Try to draw the view hierarchy diagram for it. Then try to create a new project in Android Studio and build the layout for part of a screen in the app on your own (i.e. build a list item).

   Discuss in Small Groups. Learn about some other ViewGroups like GridLayout and FrameLayout, and figure out what they can be used for. Share what you learned with peers around you.

2. **Escape sequences**

   Try displaying strings in your app that contain the escape sequences in Java such as the new line character or single quote or double quote.

3. **Data Types**

   Pick an app from Google Play. Determine what data type is being used to store different pieces of information on the screen. For example, in the Phone app, the number of missed calls is stored in an integer variable. Meanwhile, the name of a contact is stored as a string variable. You can use the Java Data types cheat sheet.

4. **Concept map**

   Draw a concept map of the various ideas you have learned in this lesson. Afterwards, see if anyone would like to share their concept map with the group.

5. **Variables**

   Each student creates a quiz question that consists of a code snippet to initialize / update a String variable called message. The solution is the ending value of message after all lines of code have been executed. The trickier the problem, the better.

   Then get two students to go to the front of the classroom to compete against each other on these quiz questions.

Project the quiz question on the projector. Have each competitor write their answer down, and then reveal their answer to the class. Then show the quiz solution on the projector, and tally up points for whoever got it correct. If needed, explain the answer for people in the class who may have gotten it wrong.

6. **Vocab words**

   Create a bingo board with the vocab words or with Java keywords. If a word is chosen, someone must explain what that word is again to remind the class.

7. **Extend the app**

   Try to modify the app in a new way that we haven't taught you in the course so far. You can change the app functionality, the appearance of the user interface, or apply to a different use case from basketball such as lap counting for running. Search online for how to accomplish this task and then try to implement it in your app. Don't worry if it takes some time. Usually it will require looking at multiple sites and lots of trial and error to get it to work.

8. **App Quality Guidelines**

   Assign each student a guideline from the Core App Quality Guidelines. Have each student read it, research it, and find an example of an app that does it well. Have them present that to the class.

9. **Material Design Showcase Apps**

   Have each pair of students look at one app from the Material Design showcase apps in Google Play. After downloading the app on the device and playing around with it, determine a list of things that the app did well based off of the Core App Quality Guidelines. If you take it a step further, can you name what Material Design principles they followed?

10. **App Onboarding Flow**

    Discuss onboarding flow. Try out a new app. Write up a paragraph about what your first experience was like - positive or negative, and whether or not you will continue to use the app. Share your post on a personal blog or on social media, or with the developer of that app.

11. **Material Design on Other Form Factors**

Try out Material Design on the web using sample app. Check out the Inbox app on mobile and web. Make observations about what it means to adapt to different form factors.

12. Finding and Reporting a Bug

Try a new app on your device. Try to make it crash or try to find a bug. Report it to the developer (see their contact info on Google Play store listing).

13. Trivia Game

Look up a piece of trivia about Android. Write up a question and answer. Split the class up into teams to play the trivia game. The team with the most correct answers wins.

Course 3: Object Oriented Programming (Part A)

Exercises

1. **Methods**

   Discuss in Small Groups. Search online for an article about Java methods. Try to read it and understand as much as you can. Then explain the article at a high-level to a fellow classmate.

2. **Methods**

   Create a comic strip out of what you have learned about methods. Share with your peers

3. **Android Resources**

   Peer teaching. Split the class in half. Have one group read the Android Documentation article on Providing Resources. Have the other group read the article on Accessing Resources. The students reading the same article can work together and discuss if necessary. Then re-group the whole class and have someone explain the Providing Resources article, while the someone else explains the Accessing Resources article.

4. **Open source apps**

   Watch this video as an initial example. Find an open source app from this list. Click on the sources link to see the code for the app. For example, look at the code for the Google I/O app (a conference schedule app). Find the resources "res" folder and look at the files in there.

5. **Read and understand code.**

   Android framework classes. Install this Chrome extension so that the Android documentation pages have a link to the source code. With a peer, look at the source code for a different View besides TextView and ImageView such as Button or Spinner or RadioButton. Discuss observations and try to understand some takeaways from the code.

6. **Objects**

Demo for class. Use either a house/floorplan prop or a cookie cutter. Show how object instances can be made off of the floorplan or cookie cutter, and how the instances can vary slightly.

Demo for class. Have smaller boxes (that represent variables) and a larger box (that represents an object like TextView). Ask a student to use the boxes to explain objects to the class (based on what they saw in the Udacity videos). If the student struggles with the explanation, feel free to have other students chime in to help.

7. **Classes in the Android Framework**

   Search online for Java classes that are in the Android framework - for example, TextView or ImageView are Java classes that we've discussed together already. Try to find the names of at least 5 other Java classes. As a challenge, try to have a high-level idea of what functionality that class provides. For example, a TextView displays text while an ImageView displays an image on the screen.

   Afterwards, ask if anyone wants to share the Java classes in the Android framework that they found online. You can make a list together on the board. If they know what that class does, they should also mention that too. Emphasize that these are all classes that they can take advantage of when building their app, so they don't need to waste time building their own version.

8. **findViewById**

   As a first step, have each student come up with a question about what they just saw in the videos (it was a lot of information) and write down the question. { Pause } As a second step, have them brainstorm ways to research the answer (ideas are here). { Pause } As a third step, have them try to figure out the answer to that question.

   Have students share their question and their own answer with a peer. Encourage students to give constructive feedback to each other and vet if the answer sounds reasonable or requires more research. If they need an expert opinion, they can try to ask a teaching assistant, the facilitator, or an online forum.

9. **Guest Interview** (Optional)

   If you know an Android developer, invite them in for a Q&A. Encourage students to ask the Android developer questions.

Course 3: Object Oriented Programming (Part B)

Exercises

1. **Imposter Syndrome**

   Skim through this article on Why Learning to Code is So Damn Hard. Ask if students can identify with any of the feelings they talk about. See if anyone want to share their thoughts from the article. Ask if anyone has any advice on how they've dealt with imposter syndrome in the past.

   Alternatively, you can read through some posts on the discussion forum thread about imposter syndrome.

2. **Data Types**

   Now that we know a new data type (boolean), try to find a real-life use case for boolean in an app from Google Play. You can refer to the Java Data types cheat sheet.

   For example, in the Gmail app, marking an app as "starred" or not is stored as a boolean because there are 2 possible states.

3. **EditText**

   After students have completed the challenging task of hooking up an EditText field in the Just Java app, discuss in small groups what they found challenging and how they were able to get past that. Can they think of better ways to approach these types of tasks in the future?

4. **EditText Attributes**

   In Android Studio, experiment with setting different attributes on the EditText view and see how that changes your app. For example, try changing the android:inputType to "number" or "textPassword". Refer to the EditText documentation page or the TextFields guide.

5. **Control Flow Statements**

   Check out the fun IFTTT website. How does it relate to what we just learned?

6. **Control Flow Statements**

Pick an app on your device. Can students try to find somewhere where a control flow statement is probably used? For example, if there's no internet connection, then an error dialog may pop up. Have them share their findings with a fellow student.

7. **Control Flow Statements**

   Think about ways you could use what you just learned to enhance your final project app. Share ideas and brainstorm with your small group.

8. **Intents**

   Have a ball which represents the information to be passed from one activity to another. Have 5 volunteers stand up with signs on them that say which activity they represent. Then have one student (the current activity) toss the ball to another activity. Use the ball analogy in the Intents video in the course to guide the demonstration.

9. **Intents**

   Try to find an example of an intent in any app on your device. Share your observations with a fellow student on why you think an intent is used there.

10. **Styles and Themes**

    Download an app from the Best In-Class Material Design Apps list. Then look at the style and theme used in that app to give it a distinct brand and user experience. Share your observations with a fellow student.

11. **More Topics in Android**

    Think about your app and look at the list of So Many Things to Explore in Android. Can you think of 1 topic area you want to explore after this course is over, because it would greatly improve the user experience of your final project app? For example adding location or mapping capabilities to your app. No need to investigate the details now or implement it now. Just take a look at the high-level areas to explore in Android.

Course 4: Multi-Screen Apps

Exercises

Starting with this course, the concepts are best understood with hands-on practice writing code in Android Studio. We recommend utilizing the exercises laid out in the online course to build the Miwok language app. The exercises can be completed individually, in pairs or small groups.

If the coding tasks are being completed with two or more people, one per person should be the driver and another can be the navigator. The driver is the person communicating what to type and the navigator should be typing. Both students should be there to support and assist each other.

If the exercises are being completed with three or more people additional roles can be added. Below are additional ideas for roles that can be added depending on group size. After each exercise, students should switch responsibilities in order to gain experience in each role.

Role	**Responsibility**
Debugger	Reviews code and corrects errors
Code Reviewer	Checks code for errors
Clarity Expert	Reviews code to ensure code is concise
Teacher	Summarizes learnings and insights for the entire group to ensure a complete understanding

Another option is to have students build a language app for a different language that they are familiar with. They can model the new app after the structure of the Miwok app. Students can even record their own audio files for how to pronounce vocabulary words, which can be played within the app.

Break Ideas

If your sessions run for long periods at a time, we recommend breaks every 45 - 60 minutes so that students can come back refreshed and ready to learn. Below is a list of fun break ideas if you want the class to do an activity together.

1. **Androidify**

   Have students Androidify themselves by creating a fun Android character that is fully customizable from physical attributes to outfits and accessories. Print or share the graphic on social media. They can even set the graphic as their profile photo.

2. **Developer events**

   As a class, look for interesting Android / developer / design related meetups in the area. This is a great way to become exposed to the ecosystem and meet people in the same industry.

3. **Cardboard**

   Check out the virtual reality world with a phone and some cardboard! See the site for more details.

4. **Android game**

   Find an Android game (maybe even one that uses sensors) on the Google Play store. Install and play it. For example, Google created this Pie Noon and Zooshi games. You could try a multi-player game as well. Have students share favorites with their classmates.

5. **Open-source app**

   Check out the source code for an open-source Android app. For example, check out one from this list.

6. **Spider web**

   This activity is probably best done later in the course after students have gotten to know each other better. Have everyone sit or stand in a circle. Throw a ball of yarn to someone in the circle and give them a compliment related to something they've done in this Android course. For example, if they were helpful in debugging, or if they created a great app, or if they explained a concept well. Then that person toss the remaining yarn to someone else and give them a compliment. Continue until everyone in the group has caught the yarn and it has created a large spider web.

7. **Take a walk**

Enjoy some fresh air with a walk outside.

8. **YouTube videos**

   Share favorite YouTube videos (on Android or another topic) and watch them together.

9. **Company visit**

   Visit a local company that does Android development to see firsthand what it's like to be a developer. Organize a Q&A or panel with a mix of engineers, designers, and product managers from the company.

License

Written By

Google Developers

Last updated: 06/2016