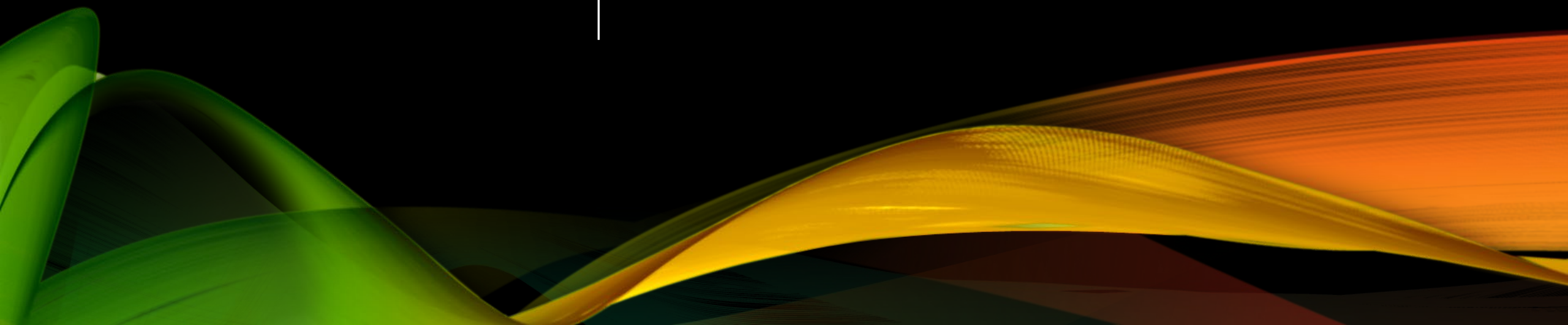
ANDROID STUDIO PURE ANDROID



REQUIREMENTS

LAPTOP & CHARGER

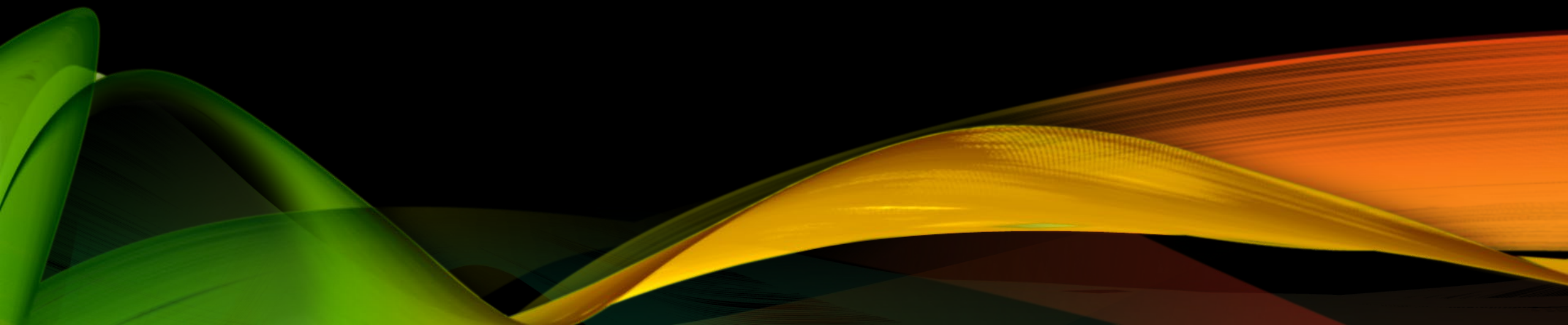
ANDROID STUDIO INSTALLED

KNOWLEDGE OF JAVA (PREFERRED, NOT REQUIRED)

ANDROID PHONE OR EMULATOR SETUP

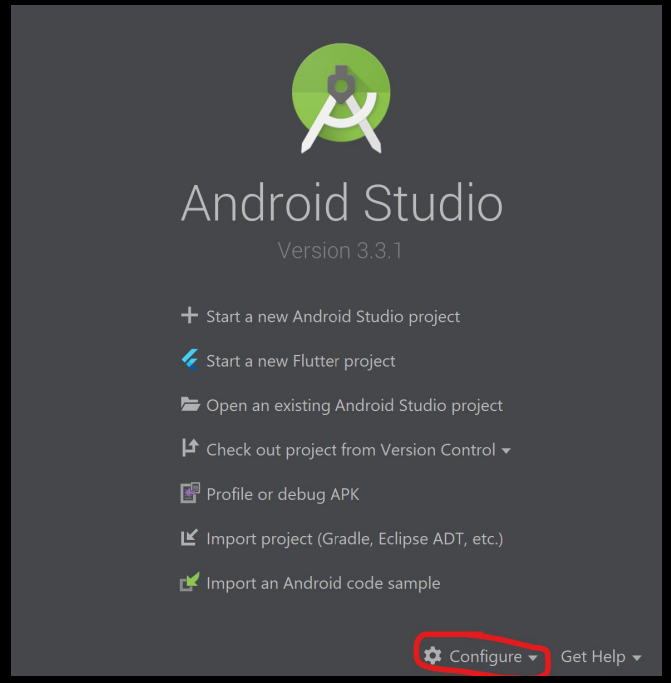
DEVICES

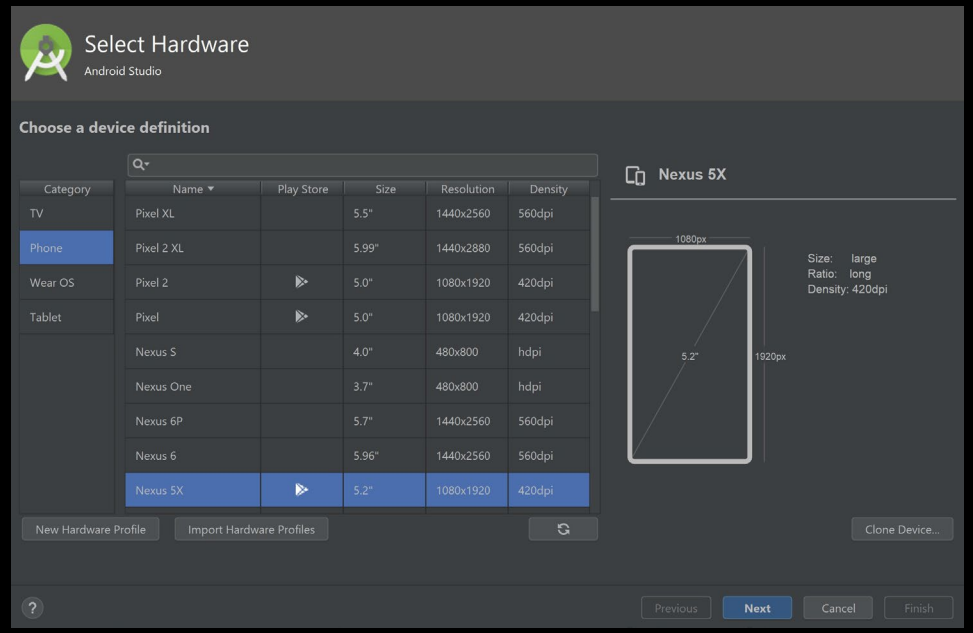
Emulator or Android Phone Setup



EMULATOR

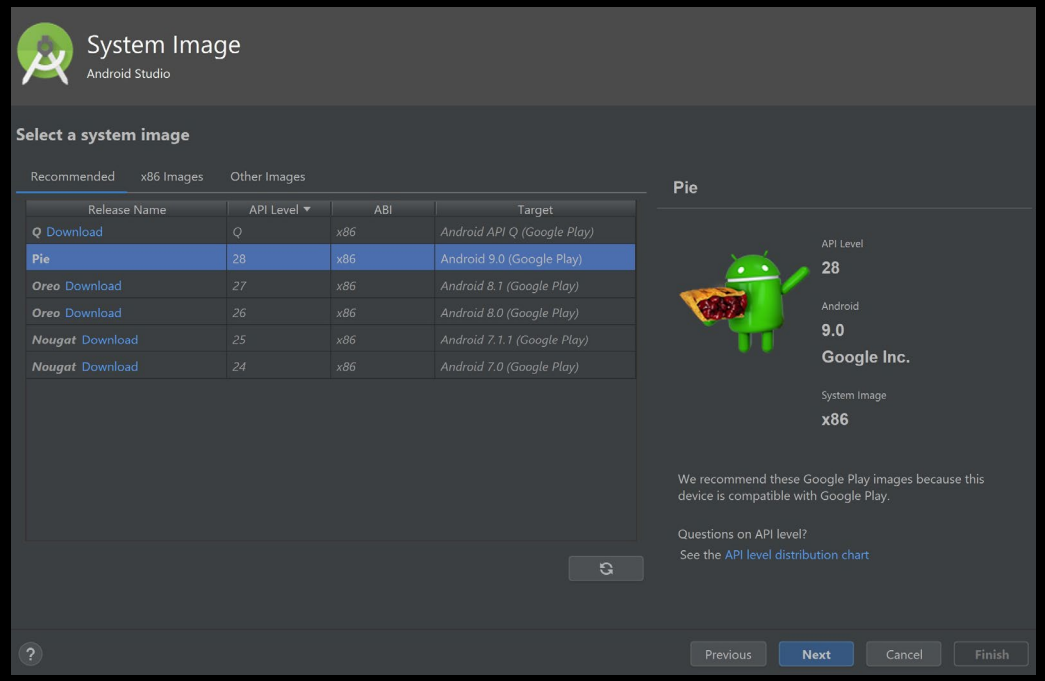
1. Start Android Studio, click Configure (Tools if Project is open)
2. Open AVD Manager and click Create Virtual Device
3. Create based on any device (Default is fine) and click Next

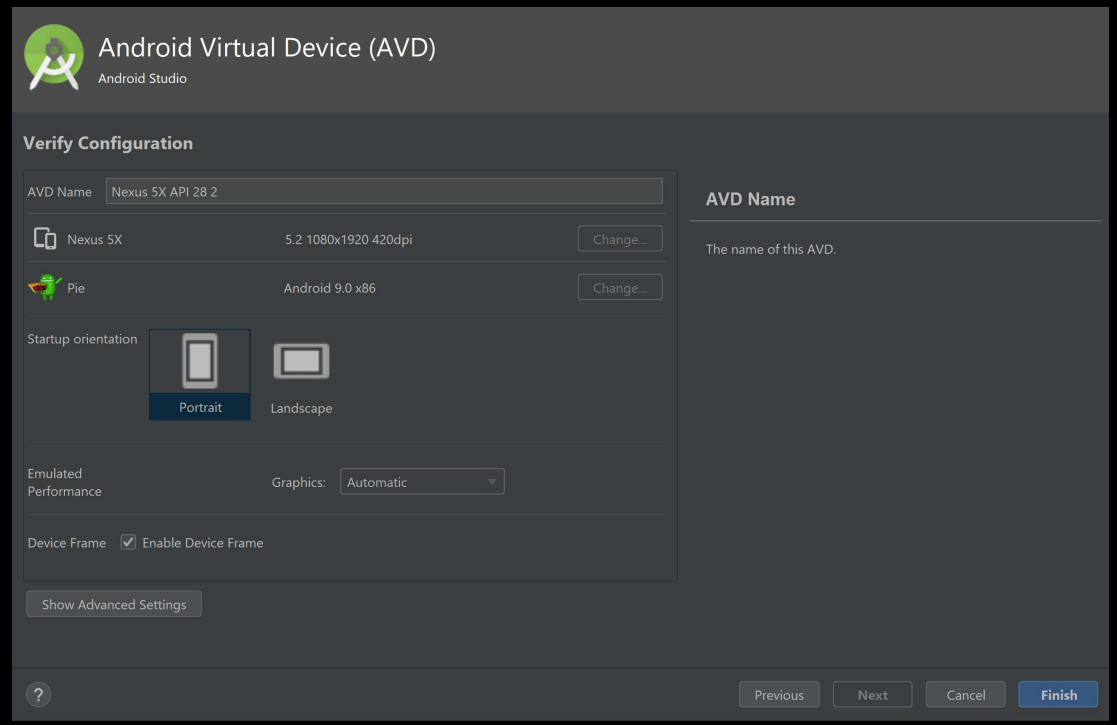




EMULATOR CONTINUED

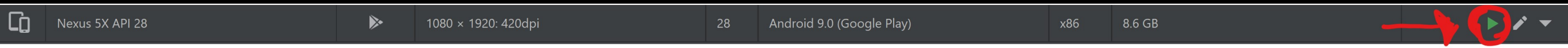
4. Select any version of Android and click Next
   - If you have an older version, it should be fine
5. Verify Settings and Finish



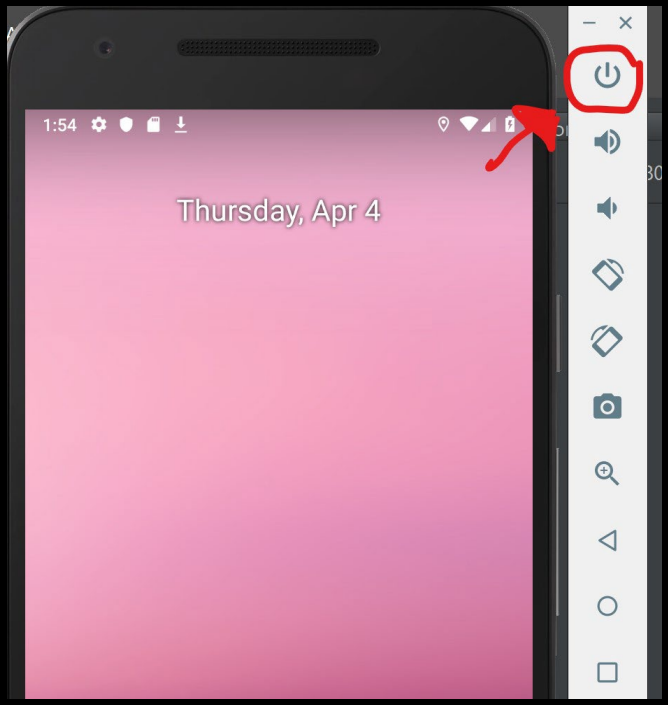


EMULATOR CONTINUED

6. Start your Emulator and Turn it on using Power Button

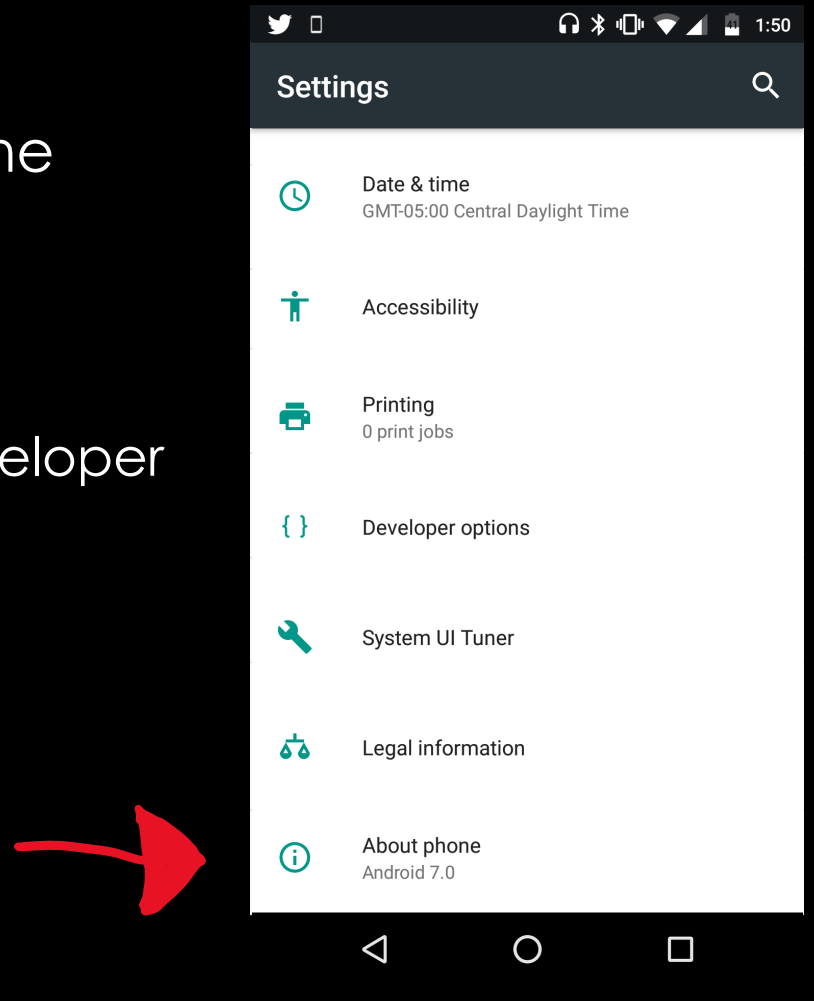


You're ready to start Developing!



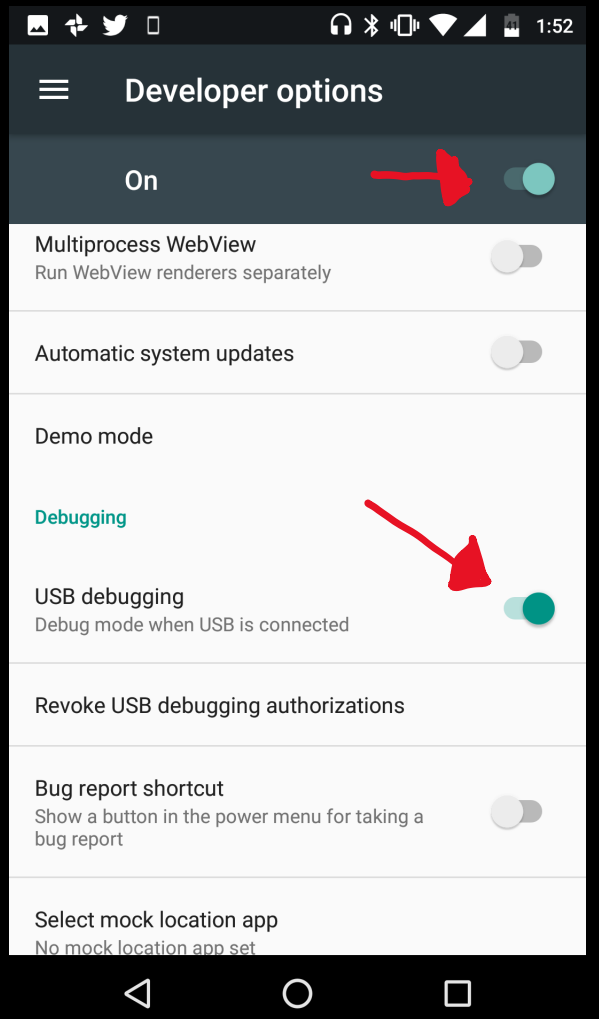
PHYSICAL DEVICE DEVELOPER OPTIONS

1. Go to Settings
2. Scroll to down to bottom and select About Phone
3. Under about Phone, find Build Number
   - Placement can very, you can try searching for it
   - Tap that ~7 times.
   - You should get a message about becoming a developer



DEVELOPER OPTIONS CONTINUED

4. Go back to the Main Settings Page
5. Select Developer Options, Toggle top switch to On
6. Scroll slowly to the section on Debugging
   - Toggle USB Debugging to On
7. Connect Device to Computer
8. If necessary, allow USB Debugging

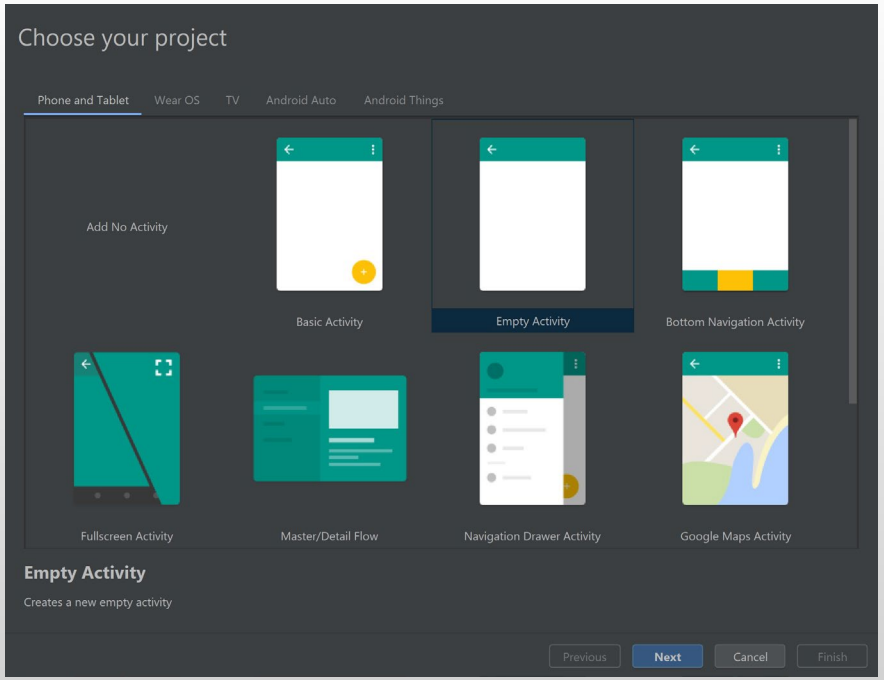


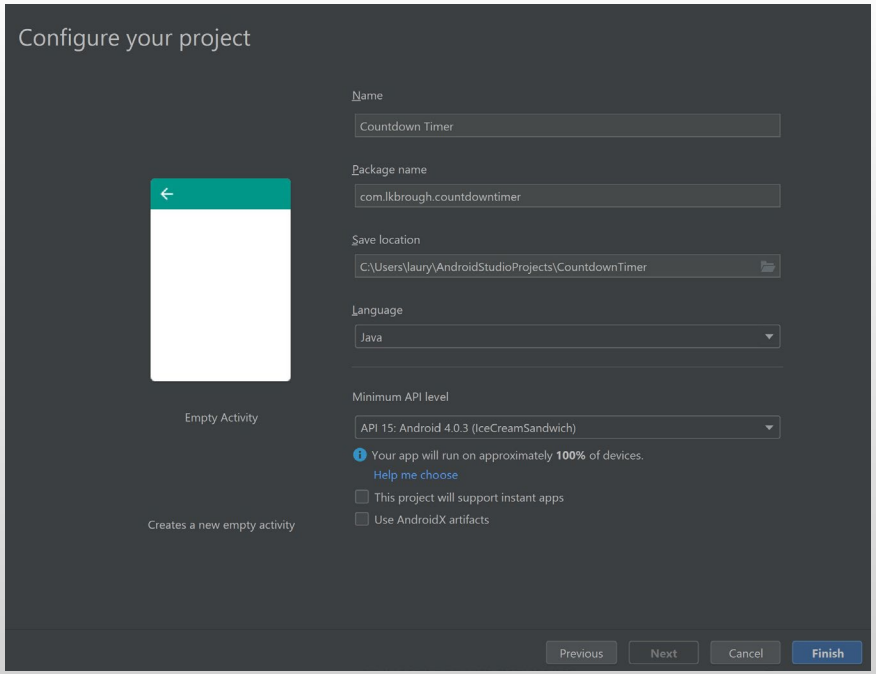
You're ready to start Developing!

ANDROID STUDIO

GETTING STARTED

1. Create a new Empty Activity Project, click Next
2. Call the app Countdown Timer and click Finish





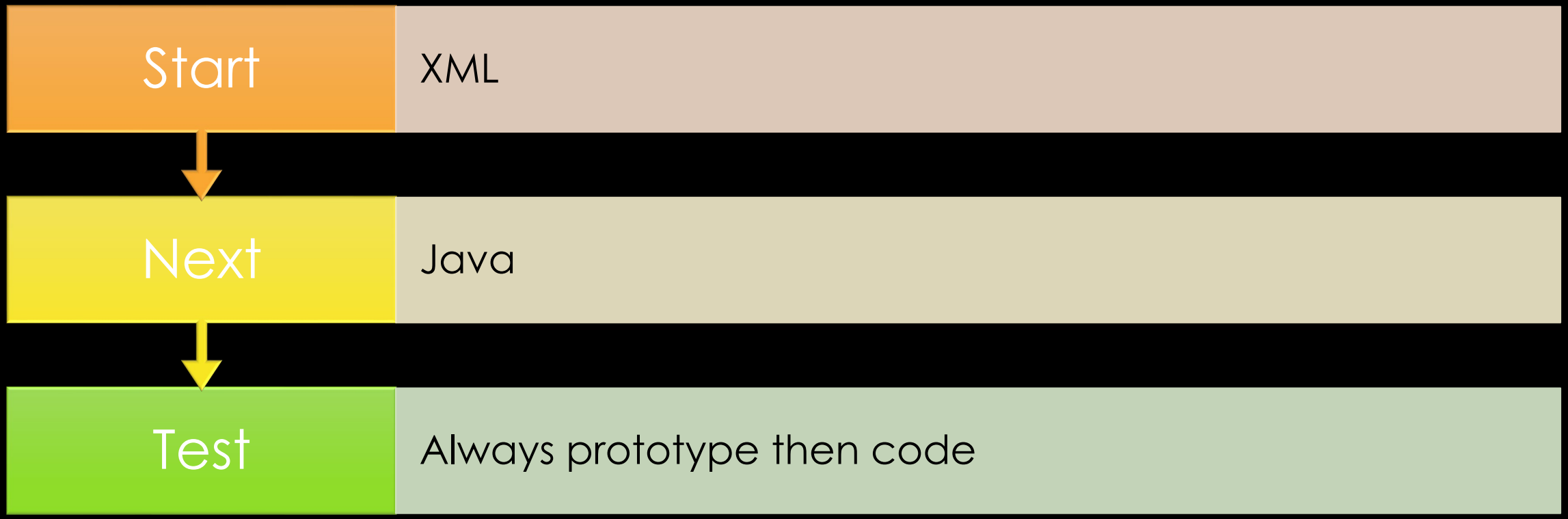
ANDROID BASICS - BACKGROUND

- Android Code is comprised of two main files
  - XML File (Layout/Display)
  - Java File (Code)
- XML can be done with either the GUI or by Text/Code
  - I prefer Text, but for this we will be using GUI since it's quicker
  - If you want to use the Text option, you can

BACKGROUND CONTINUED

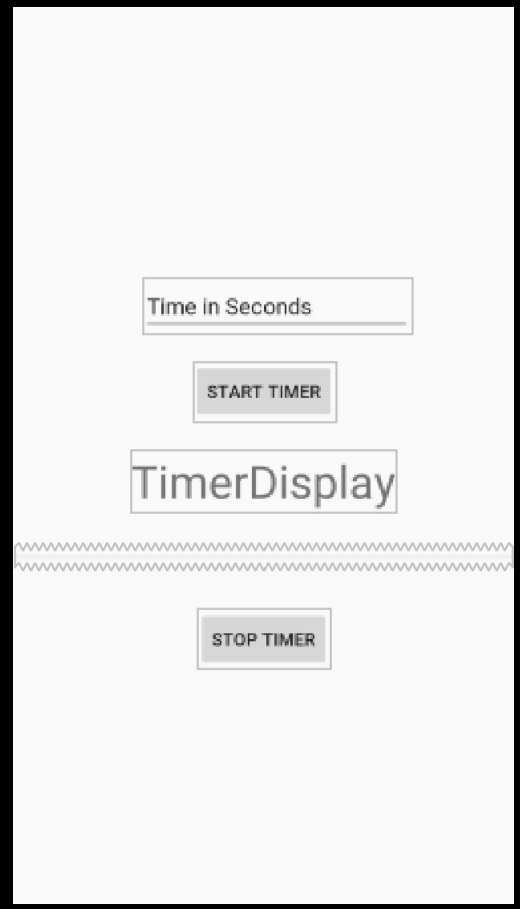
- Java is the main part of your program
  - Connects to XML
  - Can create parts of the XML
- In Java files, refer to the app outside of a java file as R
  - i.e. R.layout.textViewName
    - Refers to an object in the XML file with the id of textViewName
  - Useful for updating layouts with new information

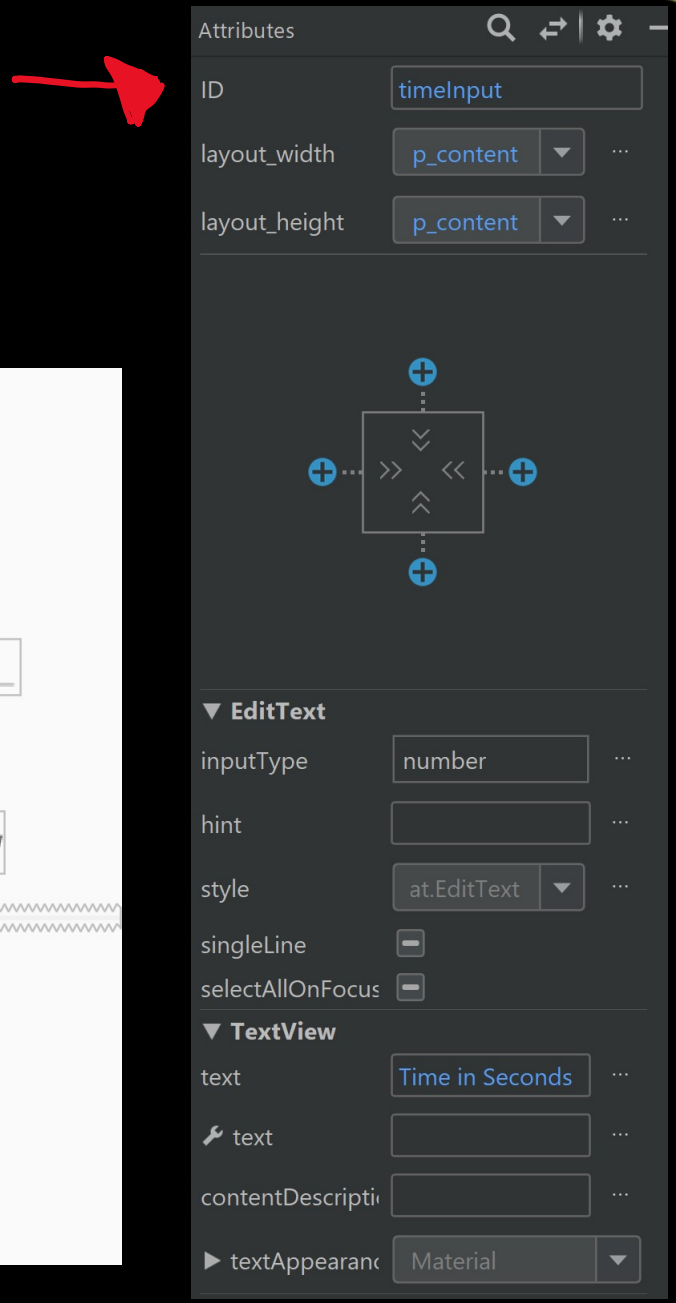
PROJECT: CREATE A COUNTDOWN TIMER



XML

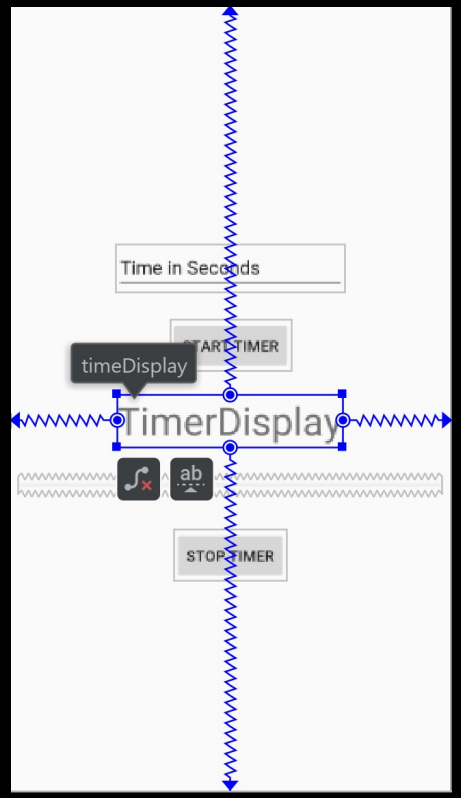
- Things you'll need
  - EditText for Time
  - Button to Start Timer
  - TextView for Timer Display
  - Optional: Button to Stop Timer
  - Optional: Progress Bar
- ALWAYS ID your objects in XML
  - Allows access in Java files

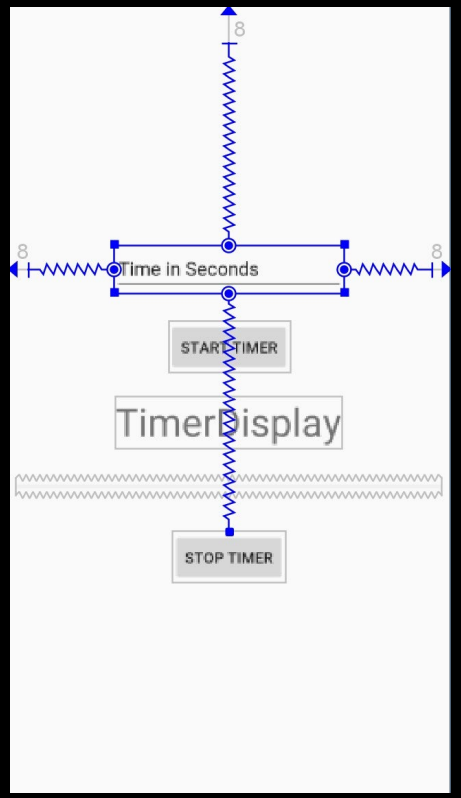


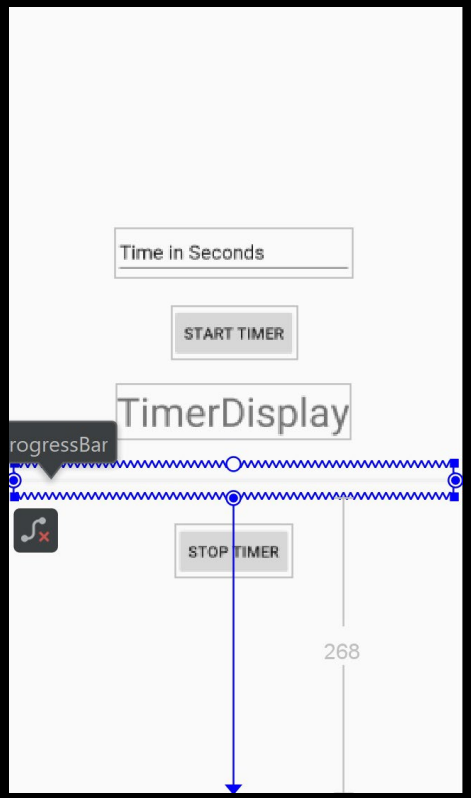


XML - CONSTRAINTS

- Describes position on screen
- Drag arrows from each side of a box to define constraints to different edges
  - Easiest to have some sort of base in the very center of the screen
- Can be defined by spacing against sides or by spacing from another object







JAVA & XML

- Right now, we have a button that does nothing. Let's change that!
- Write a method to retrieve data inputted into our EditText and send it to the timer
- Note: Anytime that a method is being called using a button, the button passes the view that it is (the button itself). Account for that in the parameters
  - startTimer(View view)

ONCREATE

- All android activities start with an onCreate method
  - Sets up display based on the attached XML file
- You can add other startup pieces onto it like we will now.
- In the Class, create a variable for a TextView for your display
- In the onCreate, save our display to it by finding it by id.

```
public class MainActivity extends AppCompatActivity {
    TextView display;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        display = (TextView) findViewById(R.id.timerDisplay);
    }
```

STARTTIMER(VIEW VIEW)

- https://we.tl/t-tC92pmB9fW
- Copy file using File Explorer and Paste into Android Studio (Project/Java/Main)
- We need a timer, text to display/update, and the data from our textview
- Retrieve Data from EditText
  - Save the Edit Text Field
    - EditText userInputField = (EditText) findViewById(R.id.timeInput);
  - Save the text from it
    - String input = userInputField.getText().toString();
  - Parse your string to an int

CREATE A TIMER

- Create a new timer
  - Class has been provided! Feel free to look through this file at one point as there are a lot of cool things going on (updating the view)
  - UpdatingCountDownTimer(int time, int CountDownInterval, TextView display, ProgressBar progress)
- CountDownInterval should simply be 1 for 1 second as we want the timer to update every second.
- If you're not using the progress bar (which none should be yet) pass it as a null
- After the timer is created, start the timer by calling it's start method.

STARTTIMER

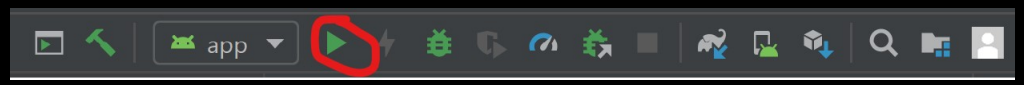
```
public void startTimer(View view){
    EditText yes = (EditText) findViewById(R.id.UserInput);
    String input = yes.getText().toString();
    int time = Integer.parseInt(input);
    UpdatingCountDownTimer timer = new UpdatingCountDownTimer(time, 1, display, null);
    timer.start();
}
```

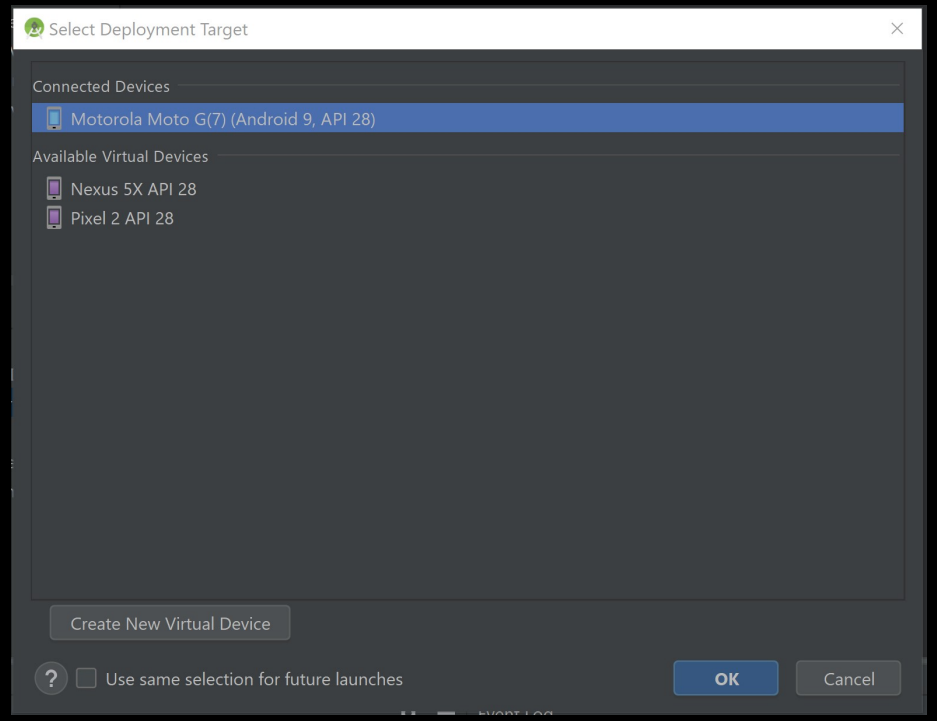
ADD THE METHOD TO THE BUTTON

- Switch back to the XML Document and Click on the Button
- In the Attributes panel, you'll see an onClick after a little bit of scrolling
- As the onClick attribute, type in our startTimer method (without parameters!)

RUN

1. Run the app with the play button in the top right hand corner
2. Select your device
   - Could be emulator or physical device, which are split into two different sections





ERRORS?

- Raise your hand if you have errors
- Talk to the person next to you
- Read through the error

PROJECT FINISHED

- Optional: Make it Pretty
- Optional: Add progress bar
- Optional: Add stop timer button
- Optional: Add another screen

BEAUTIFY

- Use the style, colors, and other xml files in the res/values directory of your android app
  - Defining Style, not applying
- Apply the style in the AndroidManifest

PROGRESS BAR

- UpdatingCountDownTimer was written to support progress bars
- Add a progress bar similar to how you added other elements and save it similar to how we saved the TextView

STOP BUTTON

- Another button, another method (similar to start)
- Save current time, destroy previous timer, modify start timer
  - Next time user presses start timer, start based on your saved time.

PROBLEMS? COMMENTS? IMPROVEMENTS?

Email: lauryn.brough@hotmail.com

Project: https://github.com/lkbrough/CountDownTimer

Hope everyone enjoyed this workshop, learned some very basic android, and enjoys the rest of the Hackathon!