APPIUM: MOBILE AUTOMATION

JOHN BRYCE
Leading in IT Education
a matrix company

1. **Prerequisites & Intro**
2. **Appium Overview**
   a. Types of Mobile Apps
   b. Introduction to Appium
   c. Appium Design
   d. Client/Server Architecture
3. **iOS (iPhone) Apps Automation**
4. **Installations & Env. Perp.**
   a. JDK
   b. Android Studio + Android SDK
   c. Environment Variables
   d. Appium Desktop
5. **Android SDK Tools**
   a. Android Debug Bridge (ADB)
   b. UI Automator viewer
   c. Android Emulator
6. **Prepare Your Device**
   a. Enable Developer Options and USB Debugging
   b. Create Android Virtual Device (AVD)
7. **Inspect Mobile App Elements**
   a. With uiautomatorviewer.bat
   b. With Appium Desktop
8. **Appium Coding**
   a. Maven Dependency
   b. Desired Capabilities
   c. UiAutomator & UiAutomator2
   d. Appium API
   e. Code!
   f. Page Object Design Pattern
9. **Good To Know**
10. **Exercise – Google Maps**
11. **External Resources**

1. Prerequisites & Intro

This course is intended for students who already have a thorough understanding of the following topics:

- Java & object-oriented programming
- Selenium WebDriver
- XPath
- Page Object design pattern
- Apache Maven – dependencies management

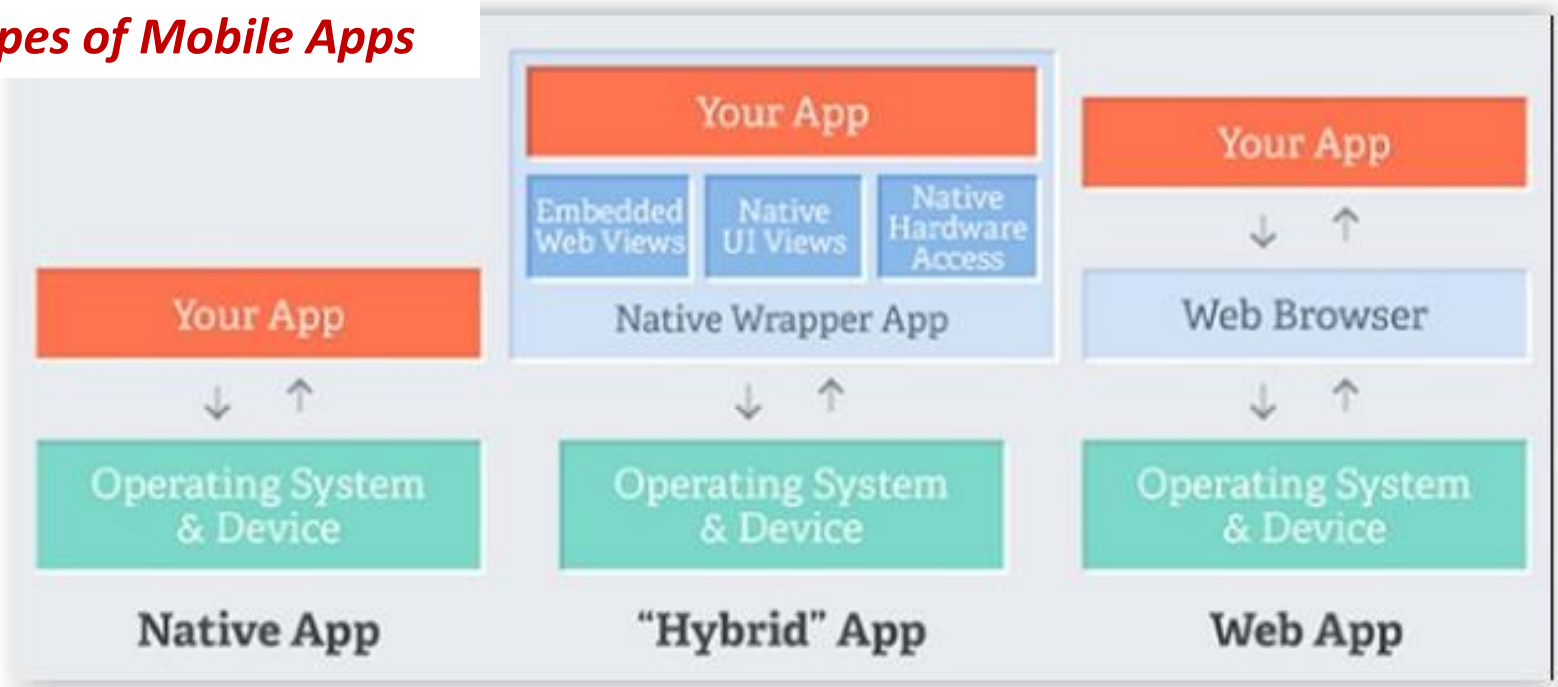
If you're already familiar with Selenium WebDriver, it's fairly easy to begin developing automation code for mobile applications with Appium, which uses the same API as Selenium.

Most of the effort to start working with Appium is to install all the required software, set up the environment, and to get familiarized with all the tools of the ecosystem – that's the focus of this course!

2. Appium Overview

JOHN BRYCE
Leading in IT Education
a matrix company

a. Types of Mobile Apps



Native apps are those written using the iOS, Android, or Windows SDKs.

Hybrid apps have a wrapper around a "webview" -- a native control that enables interaction with web content.

Mobile web apps are web apps accessed using a mobile browser

2. Appium Overview

http://appium.io/

b. Introduction to Appium

- Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS mobile, Android mobile, and Windows desktop platforms.
- Appium is "cross-platform": it allows you to write tests against multiple platforms (iOS, Android, Windows), using the same API. This enables code reuse.

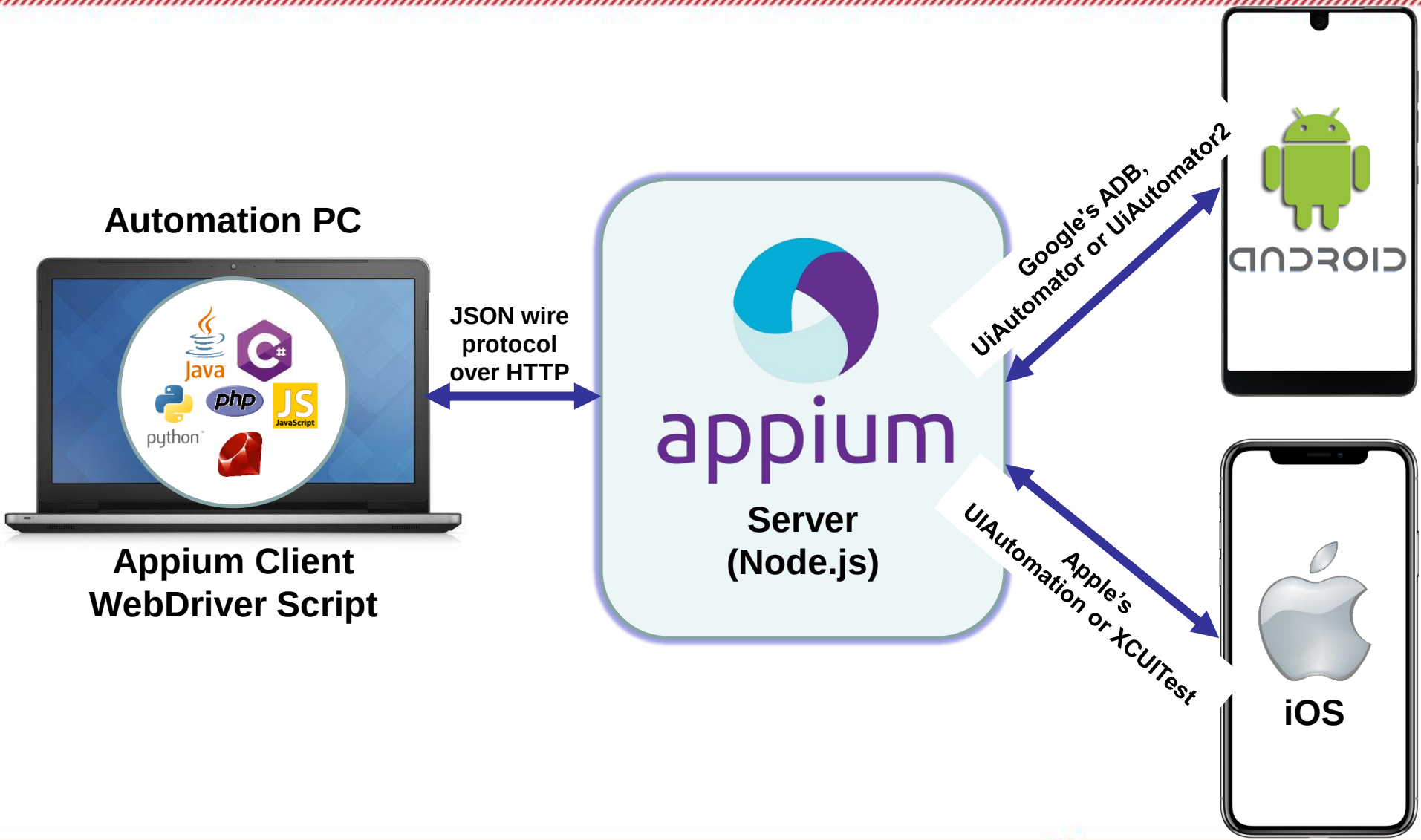
2. Appium Overview

JOHN BRYCE
Leading in IT Education
a matrix company

c. Appium Design

- Appium wraps **vendor-provided** frameworks in one API - the WebDriver API. (see the illustration on the next slide).
- Selenium WebDriver specifies a client-server protocol (known as the JSON Wire Protocol) which has become the de facto standard for automating web browsers. Appium extended the WebDriver protocol with extra API methods useful for mobile automation.
- Given this client-server architecture, a client written in any language can be used to send the appropriate HTTP requests to the server.

2. Appium Overview



2. Appium Overview

JOHN BRYCE
Leading in IT Education
a matrix company

d. Client/Server Architecture

- Appium is a web server (written in Node.js) that exposes a REST API. It receives connections from a client, listens for commands, executes those commands on a mobile device, and responds with an HTTP response representing the result of the command execution.

- There are client libraries (in Java, Ruby, Python, PHP, JavaScript, and C#) which support Appium's extensions to the WebDriver protocol. When using Appium, you want to use these client libraries instead of your regular WebDriver client.

3. iOS (iPhone) Apps Automation

JOHN BRYCE
Leading in IT Education
a matrix company

The XCUITest Driver for iOS

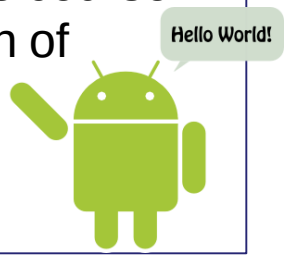
http://appium.io/docs/en/drivers/ios-xcuitest/

Appium's primary support for automating iOS apps is via the **XCUITest** driver. This driver leverages Apple's XCUITest libraries under the hood in order to facilitate automation of your app.

Requirements

- Apple's XCUITest library is only available on iOS simulators and devices that are running iOS 9.3 or higher.
- **A Mac computer with macOS 10.11 or 10.12.**
- Xcode 7 or higher is required.

NOTE: The rest of this course focuses on automation of **Android** applications.



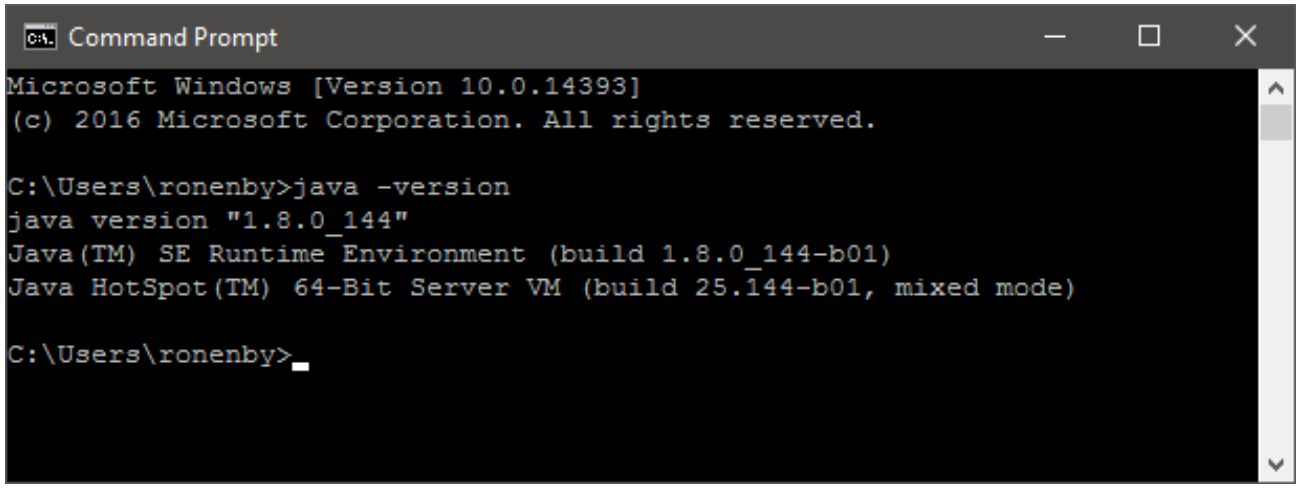
4. Installations & Env. Perp.

JOHN BRYCE
Leading in IT Education
a matrix company

a. JDK (Java Development Kit)

Make sure **JDK 8** (or newer) is installed on your PC.
https://www.oracle.com/technetwork/java/javase/downloads/index.html

Verify Java is properly installed by executing the command line:
java –version



4. Installations & Env. Perp.

JOHN BRYCE
Leading in IT Education
a matrix company

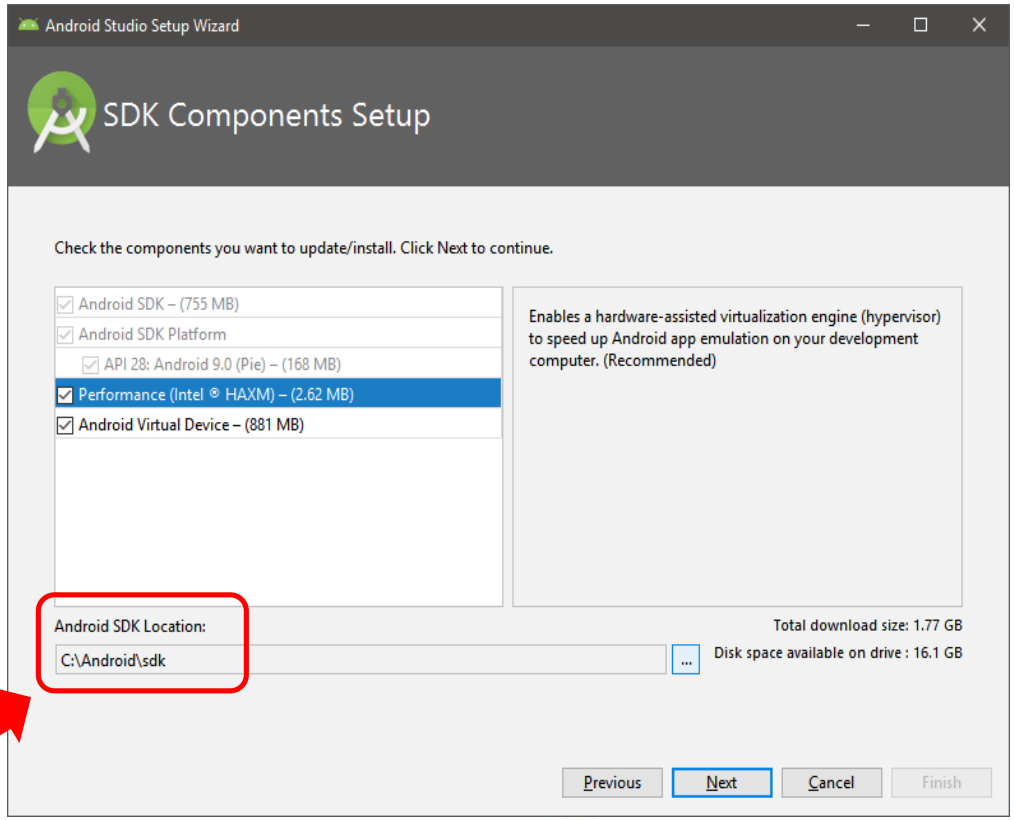
b. Android Studio + Android SDK

Download and install **Android Studio** and **Android SDK** from:
https://developer.android.com/studio/

NOTE: Installation process takes a very long time (~1 hour) – The whole process can't be demonstrated in class.

Select all checkboxes:

1. Android SDK
2. Android SDK Platform
3. Performance
4. Android Virtual Device



Remember the path you provide for: "Android SDK Location"

4. Installations & Env. Perp.

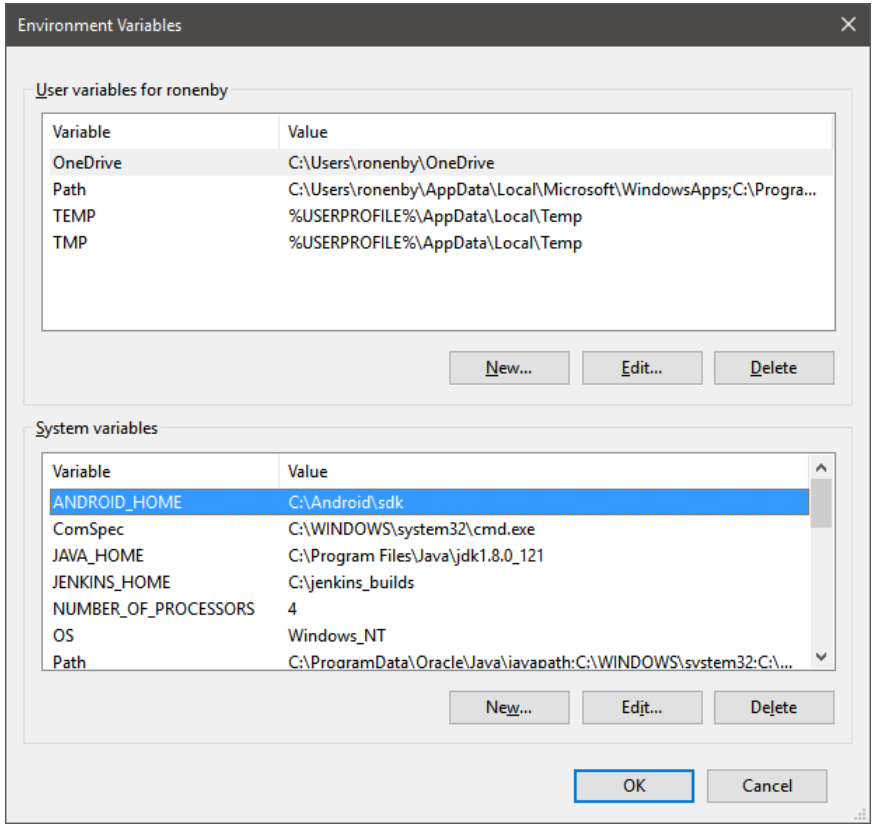
JOHN BRYCE
Leading in IT Education
a matrix company

c. Environment Variables

After a successful installation of the Android SDK, add the following variables to the system environment variables:

1. **ANDROID_HOME** =
   <path to the location of Android SDK>
   Example: **C:\Android\sdk**

2. Add values to the **path** variable:
   1. **%ANDROID_HOME%\tools**
   2. **%ANDROID_HOME%\platform-tools**



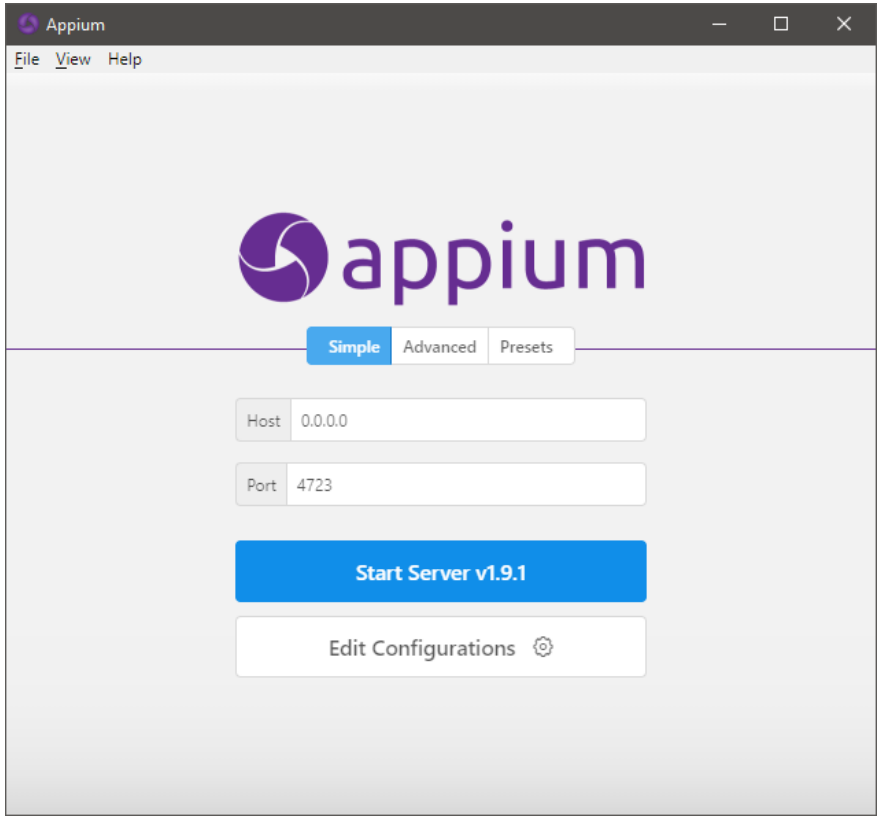
4. Installations & Env. Perp.

JOHN BRYCE
Leading in IT Education
a matrix company

d. Appium Desktop

"Appium Desktop" is a GUI wrapper around the Appium server that can be downloaded for any platform. It comes bundled with everything required to run the Appium server, so you don't need to worry about Node.js. It also comes with an Inspector, which enables you to check out the hierarchy of your app. This can come in handy when writing tests.

Download link:
https://github.com/appium/appium-desktop/releases/
For Windows, select:
appium-desktop-setup-<version>.exe



5. Android SDK Tools

JOHN BRYCE
Leading in IT Education
a matrix company

a. Android Debug Bridge (ADB)

Documentation: https://developer.android.com/studio/command-line/adb
Location on PC: %ANDROID_HOME%\platform-tools**adb.exe**

Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device. The adb command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device.

Selected examples:

adb devices - Query for devices

adb install <path_to_apk> - Install an app

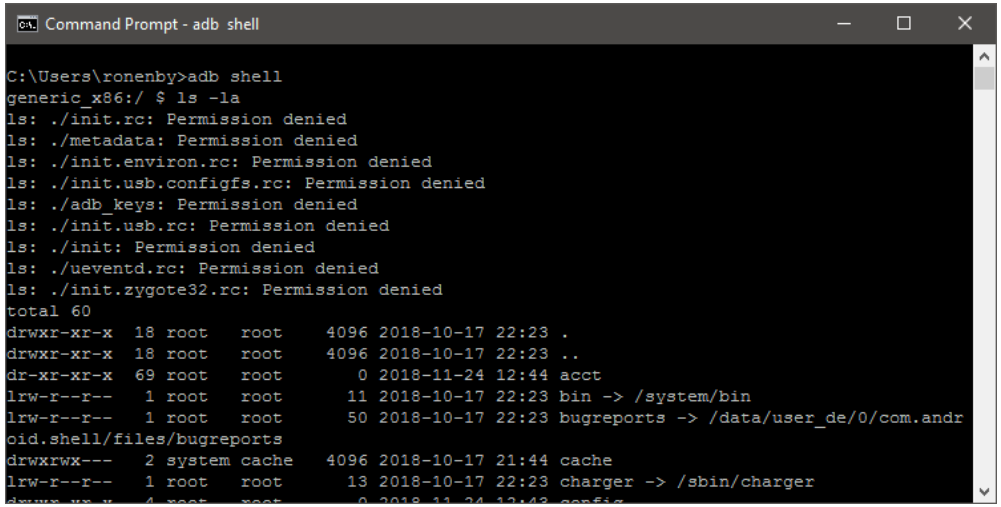
adb pull <remote> <local> - Copy files from device

adb push <local> <remote> - Copy files to device

adb shell - Issue shell commands

There are much more options…

See the official documentation.



5. Android SDK Tools

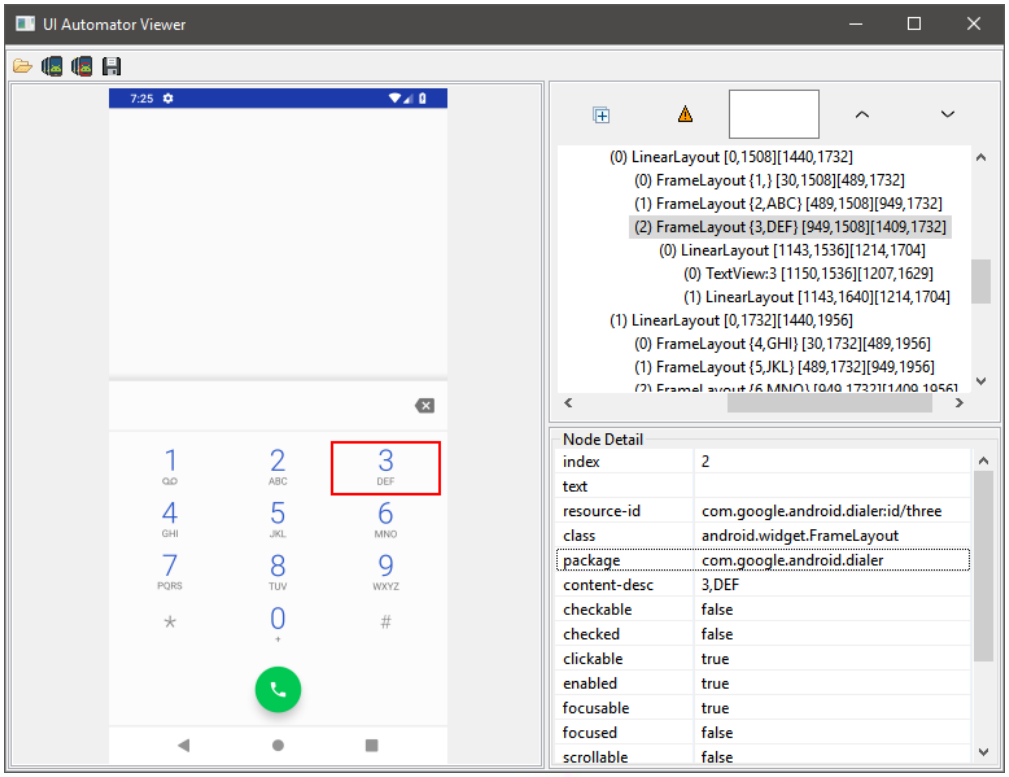
JOHN BRYCE
Leading in IT Education
a matrix company

b. UI Automator viewer

Documentation: https://developer.android.com/training/testing/ui-automator

Location on PC: %ANDROID_HOME%\tools\bin\uiautomatorviewer.bat

The uiautomatorviewer tool provides a convenient GUI to scan and analyze the UI components currently displayed on an Android device. You can use this tool to inspect the layout hierarchy and view the properties of UI components that are visible on the foreground of the device.



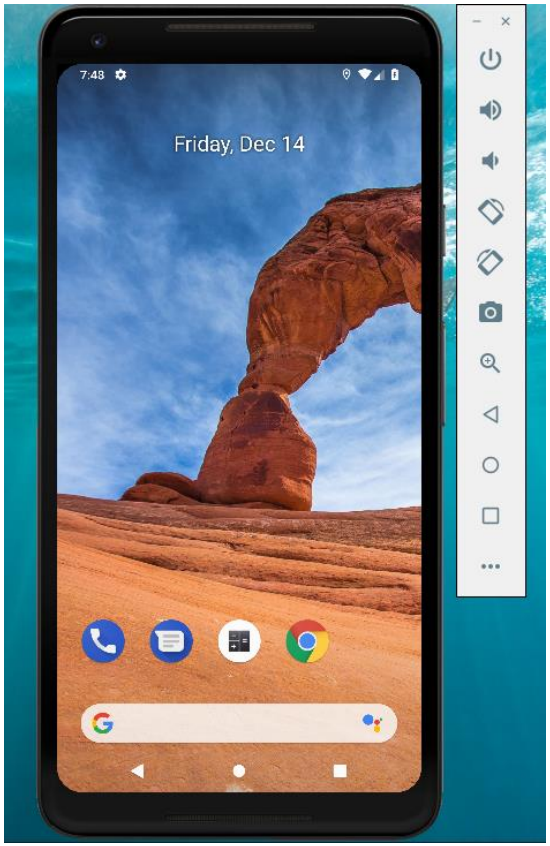
5. Android SDK Tools

JOHN BRYCE
Leading in IT Education
a matrix company

c. Android Emulator

Documentation:
https://developer.android.com/studio/run/emulator

Location on PC: %ANDROID_HOME%\emulator**emulator.exe**



The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.
The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.

6. Prepare Your Device

JOHN BRYCE
Leading in IT Education
a matrix company

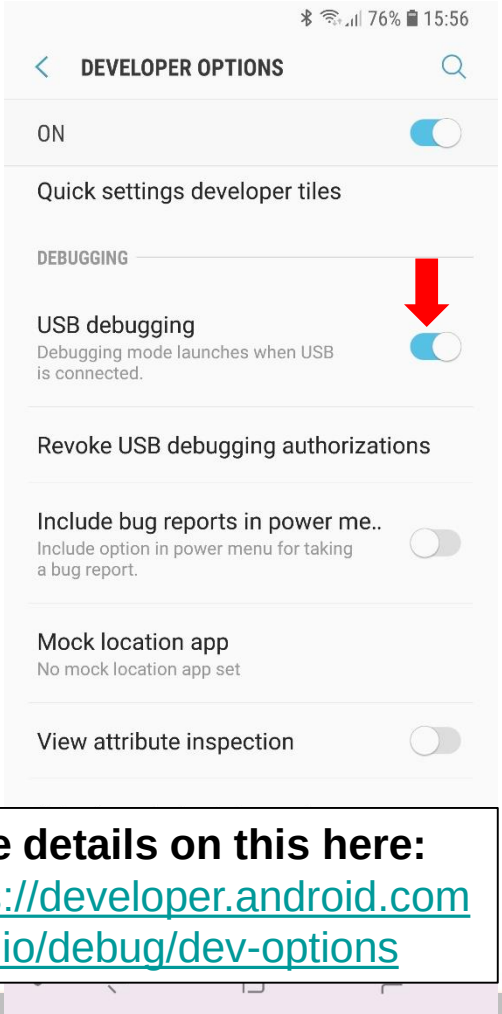
Mobile apps can be tested on a physical (real) device or on a virtual (emulated) device.

a. Prepare a Physical Android Device for Automation

If you want to control and automate apps on a physical device, it must be connected to the automation PC with a USB cable, or via Wi-Fi.

On Android devices, you also need to enable a special “USB Debugging” mode:

1. Enable "Developer Options" on the device:
   "Settings“ menu -> "About Device" -> Tap **7 times** on "Build Number" to enable developer options.
2. Inside "Developer Options“ menu -> Enable "USB Debugging"
3. Verify device successfully connected with "**adb devices**" command line17



NOTE: most devices also require a special USB/ADB **driver** software that has to be installed on the automation PC.

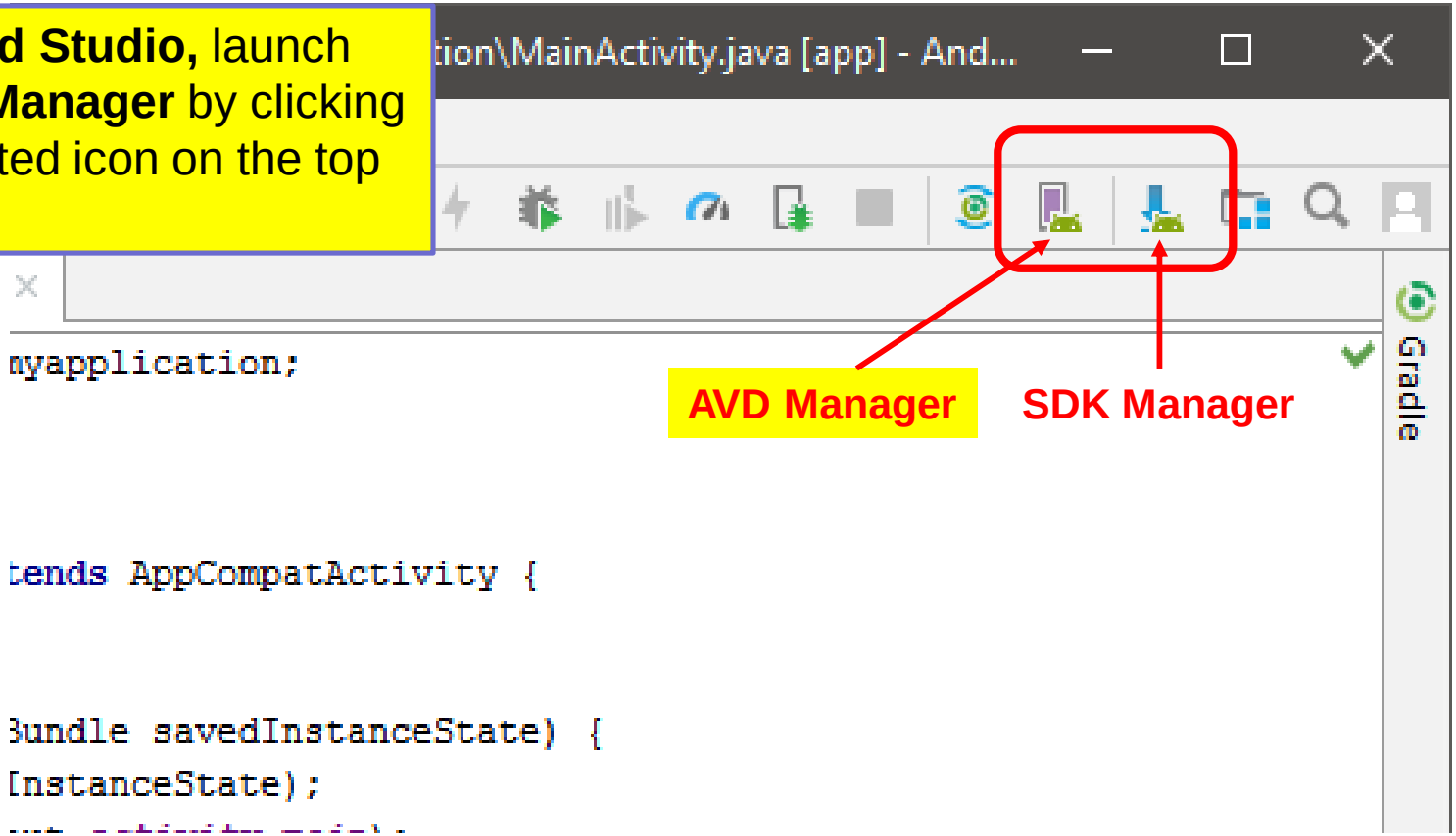
More details on this here: https://developer.android.com/studio/debug/dev-options

6. Prepare Your Device

JOHN BRYCE
Leading in IT Education
a matrix company

b. Create Android Virtual Device (AVD)

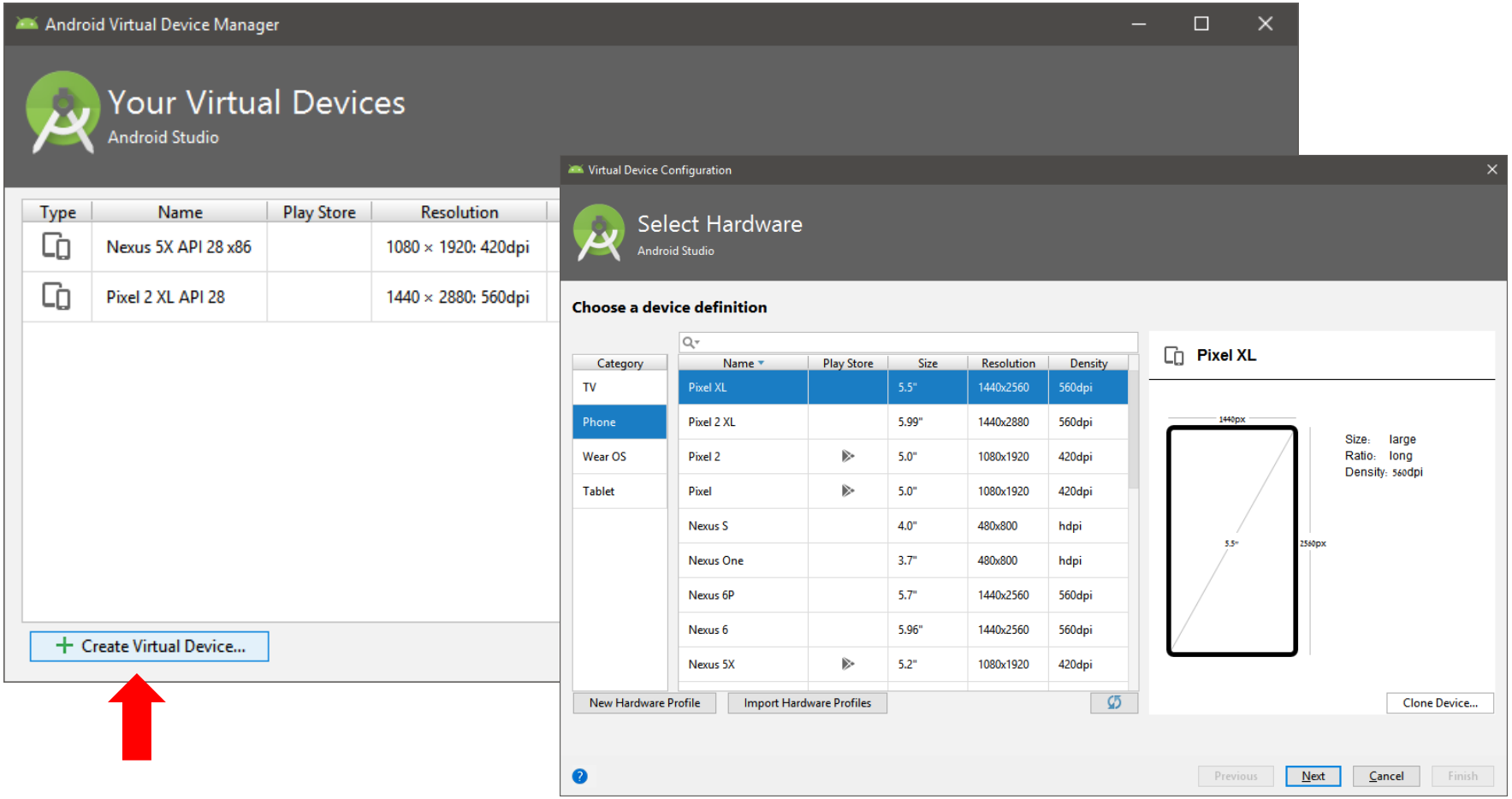
In **Android Studio,** launch the **AVD Manager** by clicking the indicated icon on the top toolbar.



6. Prepare Your Device

JOHN BRYCE
Leading in IT Education
a matrix company

b. Create Android Virtual Device (AVD)



6. Prepare Your Device

JOHN BRYCE
Leading in IT Education
a matrix company

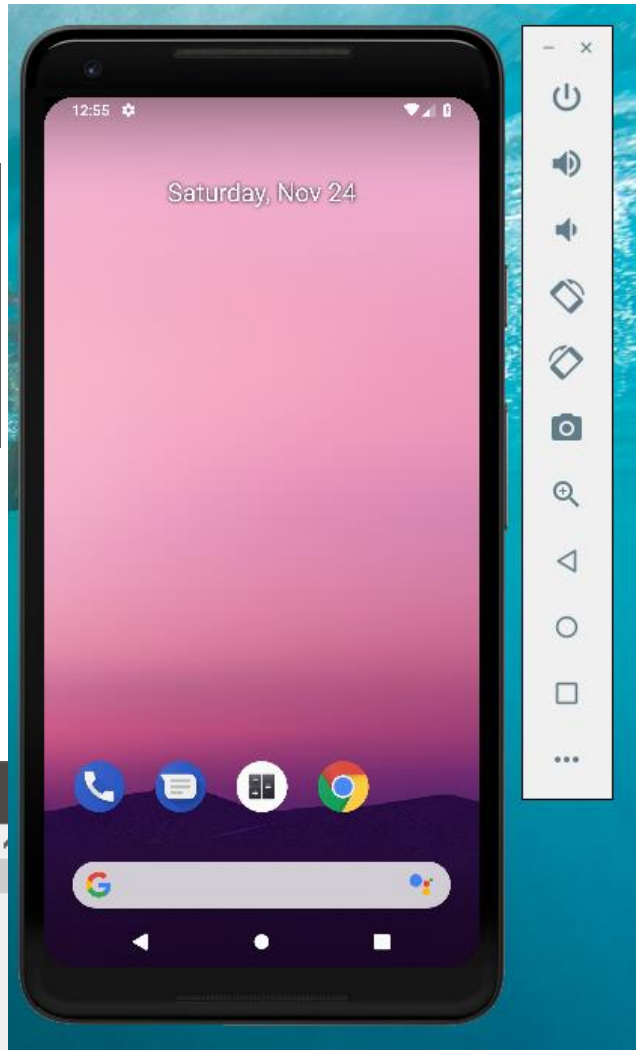
b. Create Android Virtual Device (AVD)

Launch the AVD:

C:\Windows\System32\cmd.exe - emulator.exe -avd Pixel_2_XL_API_28

```
C:\Android\sdk\emulator>emulator.exe -avd Pixel_2_XL_API_28
emulator: WARNING: Crash service did not start

HAX is working and emulator runs in fast virt mode.
```



Verify the virtual device is available via ADB – execute “adb devices” and verify the list of devices is not empty and shows your device:

Select Command Prompt

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\ronenby>adb devices
List of devices attached
emulator-5554   device
```

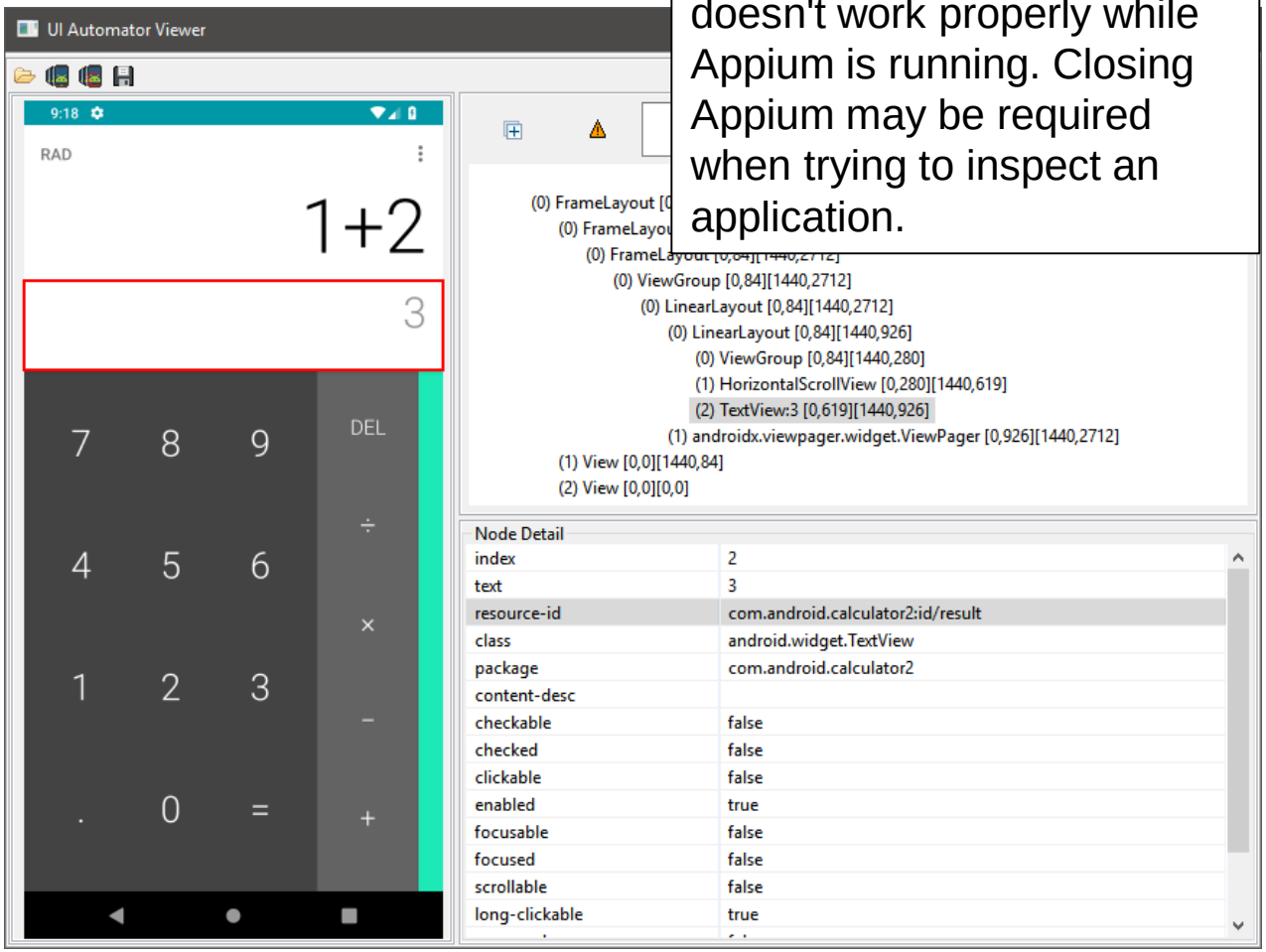
7. Inspect Mobile App Elements

JOHN BRYCE
Leading in IT Education
a matrix company

a. With uiautomatorviewer.bat

Just as we can use Chrome developer tools (F12) to inspect the HTML DOM hierarchy, to locate individual web elements and calculate their XPath expressions or CSS selectors for web automation with Selenium – So we need some kind of a tool that enables us to inspect the Mobile app elements for automation with Appium

Here we present 2 options:



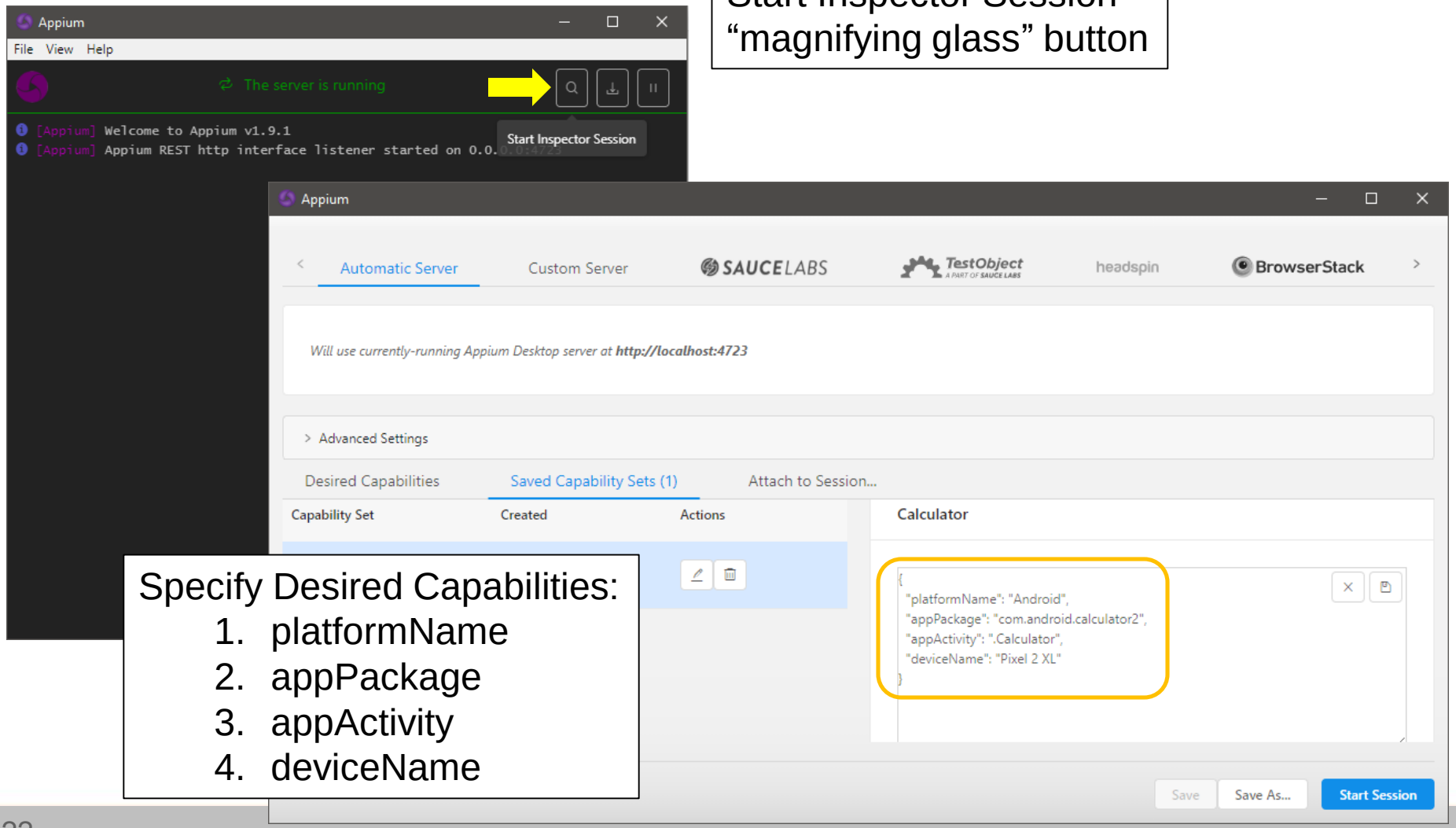
Note: uiautomatorviewer doesn't work properly while Appium is running. Closing Appium may be required when trying to inspect an application.

7. Inspect Mobile App Elements

JOHN BRYCE
Leading in IT Education
a matrix company

b. With Appium Desktop (1)

Start Inspector Session - “magnifying glass” button



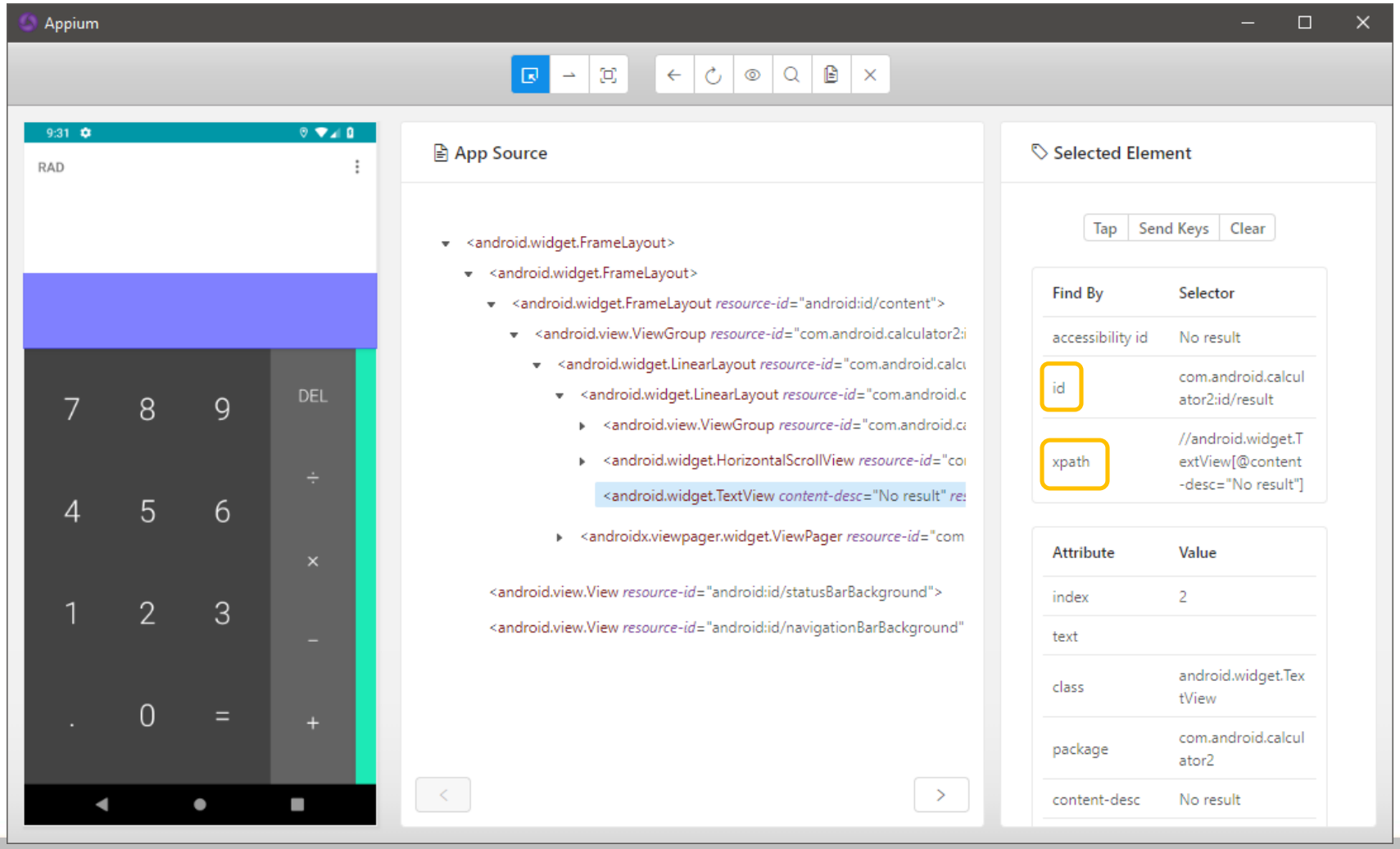
Specify Desired Capabilities:

1. platformName
2. appPackage
3. appActivity
4. deviceName

7. Inspect Mobile App Elements

JOHN BRYCE
Leading in IT Education
a matrix company

b. With Appium Desktop (2)



8. Appium Coding

JOHN BRYCE
Leading in IT Education
a matrix company

a. Maven Dependency

To start working with Appium in the context of a Java project, you need to add the “java-client” Maven dependency (in the **pom.xml** file):

https://mvnrepository.com/artifact/io.appium/java-client

```
<dependency>
   <groupId>io.appium</groupId>
   <artifactId>java-client</artifactId>
   <version>6.1.0</version>
</dependency>
```

8. Appium Coding

JOHN BRYCE
Leading in IT Education
a matrix company

b. Desired Capabilities (1)

http://appium.io/docs/en/writing-running-appium/caps/

- Desired capabilities are a set of keys and values sent to the Appium server to tell the server what kind of automation session we're interested in starting up.
- There are also various capabilities which can modify the behavior of the server during automation.
- Each Appium client builds capabilities in a way specific to the client's language, but at the end of the day, they are sent over to Appium as JSON objects.

8. Appium Coding

JOHN BRYCE
Leading in IT Education
a matrix company

b. Desired Capabilities (2)

Java:

```
DesiredCapabilities capabilities = DesiredCapabilities.android();
capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, Platform.ANDROID);
capabilities.setCapability(MobileCapabilityType.BROWSER_NAME, "");
capabilities.setCapability(AndroidMobileCapabilityType.APP_PACKAGE, APP_PACKAGE_NAME);
capabilities.setCapability(AndroidMobileCapabilityType.APP_ACTIVITY, APP_ACTIVITY_NAME);
capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "Pixel 2 XL");
```

JSON:

```
{
  "platformName": “Android",
  "browserName": "",
  "appPackage": "com.android.calculator2",
  "appActivity": ".Calculator",
  “deviceName": “Pixel 2 XL"
}
```

NOTE:
On Android, desired capabilities must specify one of the following:

- **“appPackage”** & “**appActivity”** (details of the activity to launch)
- **“app”** (path to APK file to install)

“**browserName**” capability must be an empty string if you wish to automate a native app, or set to “Chrome” if you wish to run automation on the mobile Chrome browser – similar to a regular Selenium test, but on a mobile device.

8. Appium Coding

JOHN BRYCE
Leading in IT Education
a matrix company

c. UiAutomator (1)

http://appium.io/docs/en/drivers/android-uiautomator/

- Appium's older support for automating **Android** apps is via the **UiAutomator** driver.
- The way to start a session using the UiAutomator driver is to include the **platformName** capability in your new session request, with the value **Android**. Of course, you must also include appropriate **platformVersion**, **deviceName**, and **app** capabilities, at a minimum. In the case of this driver, **no automationName** capability should be used.
- It is highly recommended to also set the **appPackage** and **appActivity** capabilities in order to let Appium know exactly which package and activity should be launched for your application. Otherwise, Appium will try to determine these automatically from your app manifest.

8. Appium Coding

JOHN BRYCE
Leading in IT Education
a matrix company

c. UiAutomator2 (2)

http://appium.io/docs/en/drivers/android-uiautomator2/

- Appium's **flagship** support for automating **Android** apps is via the **UiAutomator2** driver. This driver leverages Google's UiAutomator2 technology to facilitate automation on a device or emulator.
- The way to start a session using the UiAutomator2 driver is to include the **automationName** capability in your new session request, with the value **UiAutomator2**. Of course, you must also include appropriate **platformName** (=**Android**), **platformVersion**, **deviceName**, and **app** capabilities, at a minimum.
- It is highly recommended to also set the **appPackage** and **appActivity** capabilities in order to let Appium know exactly which package and activity should be launched for your application. Otherwise, Appium will try to determine these automatically from your app manifest.

8. Appium Coding

JOHN BRYCE
Leading in IT Education
a matrix company

d. Appium API

http://appium.io/docs/en/about-appium/api/

In addition to all functionality supported by Selenium WebDriver API, Appium extends it, and provides **A LOT** of additional, mobile-related, options:

- Status
- Execute Mobile Command
- Session
  - Create
  - End
  - Get Session Capabilities
  - Go Back
  - Screenshot
  - Source
  - Timeouts
    - Timeouts
    - Implicit Wait
    - Async Script
  - Orientation
    - Get Orientation
    - Set Orientation
  - Geolocation
    - Get Geolocation
    - Set Geolocation
  - Logs
    - Get Log Types
    - Get Logs
  - Settings
    - Update Settings
    - Get Device Settings
- Device
  - Activity
    - Start Activity
    - Current Activity
    - Current Package
  - App
    - Install App
    - Is App Installed
    - Launch App
    - Background App
    - Close App
    - Reset App
    - Remove App
    - Get App Strings
    - End Test Coverage
  - Files
    - Push File
    - Pull File
    - Pull Folder
  - Interactions
    - Shake
    - Lock
    - Unlock
    - Is Locked
    - Rotate
  - Keys
    - Press keycode
    - Long press keycode
    - Hide Keyboard
    - Is Keyboard Shown
  - Network
    - Toggle Airplane Mode
    - Toggle Data
    - Toggle WiFi
    - Toggle Location Services
    - Send SMS
    - GSM Call
    - GSM Signal
    - GSM Voice
  - Performance Data
    - Get Performance Data
    - Performance Data Types
  - Screen Recording
    - Start Screen Recording
    - Stop Screen Recording
  - Simulator
    - Perform Touch ID
    - Toggle Touch ID Enrollment
  - System
    - Open Notifications
    - System Bars
    - System Time
  - Authentication
    - Finger Print
- Context
  - Get Context
  - Get All Contexts
  - Set Context
- Interactions
  - Mouse
    - Move To
    - Click
    - Double Click
    - Button Down
    - Button Up
  - Touch
    - Single Tap
    - Double Tap
    - Move
    - Touch Down
    - Touch Up
    - Long Press
    - Scroll
    - Flick
    - Multi Touch Perform
    - Touch Perform
  - W3C Actions

8. Appium Coding

JOHN BRYCE
Leading in IT Education
a matrix company

e. Code!

```java
public class MobileCalculatorTest {

    private final static String APP_PACKAGE_NAME = "com.android.calculator2";
    private final static String APP_ACTIVITY_NAME = ".Calculator";

    private AndroidDriver<MobileElement> driver;

    @BeforeClass
    public void before() throws MalformedURLException {
        DesiredCapabilities capabilities = DesiredCapabilities.android();
        capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "UiAutomator2");
        capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, Platform.ANDROID);
        capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "Pixel 2 XL");
        capabilities.setCapability(MobileCapabilityType.BROWSER_NAME, "");
        capabilities.setCapability(AndroidMobileCapabilityType.APP_PACKAGE, APP_PACKAGE_NAME);
        capabilities.setCapability(AndroidMobileCapabilityType.APP_ACTIVITY, APP_ACTIVITY_NAME);

        driver = new AndroidDriver<MobileElement>(new URL("http://0.0.0.0:4723/wd/hub"), capabilities);
    }

    @Test
    public void mobileCalculatorTest() {

        driver.findElement(By.id("com.android.calculator2:id/digit_1")).click();
        driver.findElement(By.id("com.android.calculator2:id/op_add")).click();
        driver.findElement(By.id("com.android.calculator2:id/digit_2")).click();
        MobileElement resultBox = driver.findElement(By.id("com.android.calculator2:id/result"));
        String result = resultBox.getText();

        System.out.println("Result: " + result);

        Assert.assertEquals(3, Integer.parseInt(result));
    }
```

8. Appium Coding

JOHN BRYCE
Leading in IT Education
a matrix company

f. Page Object Design Pattern

Remember the importance of the Page Object design pattern!
It's relevant not only for web applications automation, but for mobile apps as well, and in fact for all GUI applications!

While working on an Appium project, you should employ all the techniques and best practices you learned with regards to "regular" Selenium web automation projects.

```
import io.appium.java_client.MobileElement;
import io.appium.java_client.pagefactory.*;

@AndroidFindBy(someStrategy)
@iOSFindBy(someStrategy)
MobileElement someElement;

@AndroidFindBy(someStrategy) //for the crossplatform mobile native
@iOSFindBy(someStrategy) //testing
List<MobileElement> someElements;
```

9. Good To Know

JOHN BRYCE
Leading in IT Education
a matrix company

How to Start Appium Server from Java Code:

http://www.automationtestinghub.com/3-ways-to-start-appium-server-from-java/

```
AppiumDriverLocalService service =
AppiumDriverLocalService.buildDefaultService();
service.start();
// your tests...
service.stop();
```

How to Find App Package and Activity names using “adb shell”:

http://toolsqa.com/mobile-automation/appium/how-to-find-apppackage-and-appactivity-for-apk-file/

```
adb shell
dumpsys window windows | grep –E  'mCurrentFocus|mFocusedApp'
```

10. Exercise – Google Maps

Configure desired capabilities to launch **Google Maps** app:

Package name: com.google.android.apps.maps

Activity name: com.google.android.maps.MapsActivity

Test steps (after the app was launched):

Set test parameters: **String locationName; String addressShouldContain;**

1. If 'Sign in' offer screen is shown - click 'SKIP'
2. Click top search bar
3. Type the value of **locationName** into the 'Search here' box
4. Read the location name and address of the **first search result**, and print them
5. Verify the first result address contains the value of **addressShouldContain**

Examples:

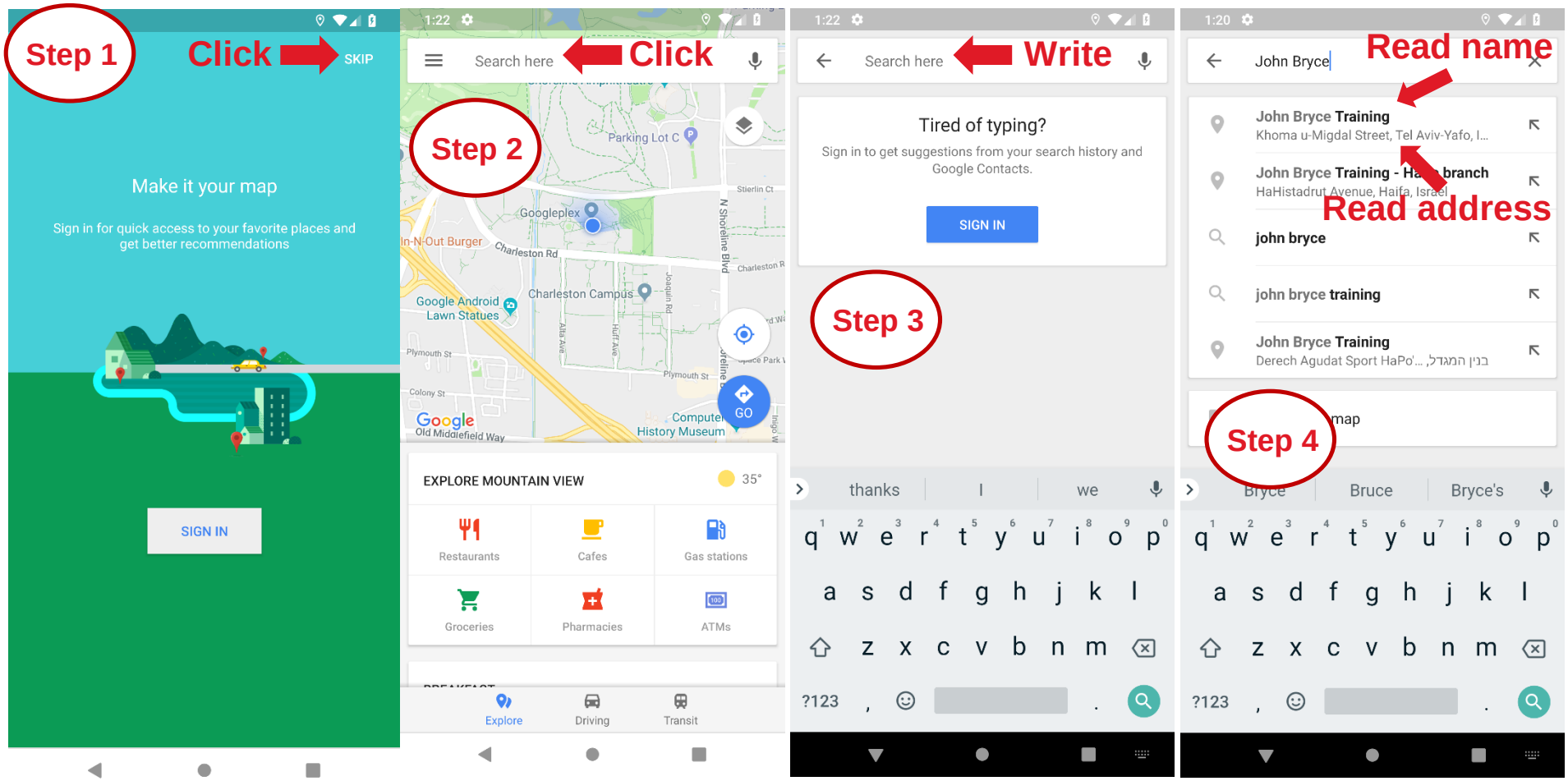
locationName = "John Bryce"; addressShouldContain = "Tel Aviv"

locationName = "Statue of Liberty"; addressShouldContain = "New York"

locationName = "Louvre"; addressShouldContain = "Paris"

10. Exercise – Google Maps

JOHN BRYCE
Leading in IT Education
a matrix company



11. External Resources

JOHN BRYCE
Leading in IT Education
a matrix company

Introduction to Appium (Official)
http://appium.io/docs/en/about-appium/intro/?lang=en

Appium - Getting Started (Official)
http://appium.io/docs/en/about-appium/getting-started/?lang=en

Appium Tutorial for Beginners (Android & iOS)
http://toolsqa.com/mobile-automation/appium/appium-tutorial/

Appium for Android Tutorial
https://nishantverma.gitbooks.io/appium-for-android/content/

APPIUM Tutorial for Android & iOS Mobile Apps Testing
https://www.guru99.com/introduction-to-appium.html

Implement Page Object Model using Appium & Java for Android Tests
https://blog.testproject.io/2018/07/31/page-object-model-appium-java-android/

JOHN BRYCE
Leading in IT Education
a matrix company

Thank You!