Appium Fundamentals

```
>npm -v
>node -v
>nvm -v
>java —version
>appium -v
>brew -v
>mvn -v
>git -v
```

>Installation required for appium in MAC:
Android Studio & tools
Eclipse
JDK
Xcode. ->only installed for git in Mac
Install appium through CLI
Appium inspector

```
>vi ~/.bash_profile          => Here we set & save all environment variables but is
saved for specific terminal session temporarily and
                        not inside shell(zsh)
>source ~/.bash_profile

>vi ~/.zshrc                            => Here we set & save all environment variables
but is saved permanently within shell(zsh)
>source ~/.zshrc

>open /usr/libexec
```

```
>echo $PATH
/opt/homebrew/opt/openjdk/bin:/opt/homebrew/opt/openjdk@17/bin:/opt/
homebrew/opt/php@8.1/sbin:/opt/homebrew/opt/openjdk@17/bin:/opt/
homebrew/opt/php@7.1/sbin:/opt/homebrew/opt/php@7.1/bin:/opt/homebrew/
opt/php@8.1/bin:/opt/homebrew/opt/php@8.1/bin:/opt/homebrew/opt/
openjdk@17/bin:/Users/krishnabros/.jenv/shims:/Users/krishnabros/.jenv/bin:/
Users/krishnabros/.pyenv/shims:/Users/krishnabros/.pyenv/bin:/Library/
Frameworks/Python.framework/Versions/3.9/bin:/Users/krishnabros/.nvm/
versions/node/v22.6.0/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/
bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/
com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/
com.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/
com.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/
```

```
Applications/VMware Fusion Tech Preview.app/Contents/Public:/usr/local/go/
bin:/opt/salt/bin:/usr/local/sbin:/Applications/Visual Studio Code.app/Contents/
Resources/app/bin:/Users/krishnabros/Library/Application Support/JetBrains/
Toolbox/scripts
```

```
>echo $PATH
/opt/homebrew/opt/openjdk/bin:/opt/homebrew/opt/openjdk@17/bin:/opt/
homebrew/opt/php@8.1/sbin:/opt/homebrew/opt/openjdk@17/bin:/opt/
homebrew/opt/php@7.1/sbin:/opt/homebrew/opt/php@7.1/bin:/opt/homebrew/
opt/php@8.1/bin:/opt/homebrew/opt/php@8.1/bin:/opt/homebrew/opt/
openjdk@17/bin:/Users/krishnabros/.jenv/shims:/Users/krishnabros/.jenv/bin:/
Users/krishnabros/.pyenv/shims:/Users/krishnabros/.pyenv/bin:/Library/
Frameworks/Python.framework/Versions/3.9/bin:/Users/krishnabros/.nvm/
versions/node/v22.6.0/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/
bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/
com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/
com.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/
com.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/
Applications/VMware Fusion Tech Preview.app/Contents/Public:/usr/local/go/
bin:/opt/salt/bin:/usr/local/sbin:/Applications/Visual Studio Code.app/Contents/
Resources/app/bin:/Users/krishnabros/Library/Application Support/JetBrains/
Toolbox/scripts:opt/homebrew/Cellar/git/2.46.0/bin/git:opt/homebrew/Cellar/git/
2.46.0/bin:/usr/local/bin:/Users/krishnabros/Library/Android/sdk/platform-tools:/
Users/krishnabros/Library/Android/sdk/tools:/Users/krishnabros/Library/
Android/sdk/tools/bin:/Users/krishnabros/Library/Android/sdk/emulator
```

```
export JAVA_HOME=$(/usr/libexec/java_home)
export PATH=$PATH:$JAVA_HOME/bin
export M2_HOME=/Users/krishnabros/Documents/apache-maven-3.8.8
export PATH=$PATH:$M2_HOME/bin
export ANDROID_HOME=/Users/krishnabros/Library/Android/sdk
export PATH=$PATH:$ANDROID_HOME/platform-tools
export PATH=$PATH:$ANDROID_HOME/tools
export PATH=$PATH:$ANDROID_HOME/tools/bin
export PATH=$PATH:$ANDROID_HOME/emulator
```

>unset [variable_name]. => To unset/ remove environment variable

>Command + Shift + .   => to unhide / hide hidden files in MAC

>/usr/libexec/java_home -V.   => You can find the java installed in your system through this command

>To install homebrew:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

>Run these two commands in your terminal to add Homebrew to your PATH:

1. (echo; echo 'eval "$(/opt/homebrew/bin/brew shellenv)"') >> /Users/krishnabros/.zprofile
2. eval "$(/opt/homebrew/bin/brew shellenv)"

>Below path by default added to env PATH

```
==> HOMEBREW_PREFIX=/opt/homebrew
HOMEBREW_CELLAR=/opt/homebrew/Cellar
HOMEBREW_REPOSITORY=/opt/homebrew
INFOPATH=/opt/homebrew/share/info:
```

>To check homebrew version:

```
brew -v
```

>To check the path of homebrew in Mac:

```
which brew          =>/opt/homebrew/bin/brew
```

>To check path of git in Mac:

```
which git           =>/usr/bin/git
```

>To check path of maven in Mac:

```
which mvn           => /Users/krishnabros/Documents/apache-maven-3.8.8/bin/mvn
```

>To check path of java in Mac:

```
which java          => /usr/bin/java
```

```
>which node         => /usr/local/bin/node
```

```
>which npm          => /usr/local/bin/npm
```

```
>which appium    =>/usr/local/bin/appium
```

>Location of “node_modules” folder:
In Mac: /usr/local/lib/node_modules
In Windows: C:Users/krishna/AppData/Roaming/npm/node_modules

>Location of “npm” folder:
In Mac: /usr/local/lib/node_modules/npm
In Windows: C:Users/krishna/AppData/Roaming/npm

>Location of “nvm” folder:

In Mac: /usr/local/lib/node_modules/nvm
In Windows: C:Users/krishna/AppData/Roaming/npm/nvm

>To install nvm:

```
brew install nvm (or)
brew reinstall nvm
```

>To set path of nvm as per the previous nvm install command result:

```
mkdir ~/.nvm
vi ~/.profile       => add the below line into this file
export NVM_DIR="$HOME/.nvm"
  [ -s "/opt/homebrew/opt/nvm/nvm.sh" ] && \. "/opt/homebrew/opt/nvm/nvm.sh"  # This loads nvm
  [ -s "/opt/homebrew/opt/nvm/etc/bash_completion.d/nvm" ] && \. "/opt/homebrew/opt/nvm/etc/bash_completion.d/nvm"  # This loads nvm bash_completion
source ~/.profile     => to save the changes
```

```
nvm -v   => to check version
```

```
echo $(/usr/libexec/java_home)         =>/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home
echo $JAVA_HOME      =>/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home
echo $JDK_HOME      =>/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home
```

>To install appium: [open /usr/local/lib/node_modules/appium]

```
npm install -g appium@next
appium -v
appium
```

Commands to remember :
https://rahulshettyacademy.com/blog/index.php/2021/07/25/get-started-with-appium/#t-1603642568346

```
>printenv.  = > To list all environment variable
>echo $PATH    => TO print specific environment variable
>which git => to display path of the software installed
>open /usr/libexec/java_home.   => to navigate to the specific folder/ file path
>vi ~/.bash_profile.   => To set permanent environment variable. To save ESC + :!wq + ENTER
>export [variable_name]=[variable_value]
>source ~/.bash_profile    => to execute .bash_profile by either restarting the terminal window or using this command
>vi ~/.zshrc                              => Here we set & save all environment variables but is saved permanently within shell(zsh)
```

>source ~/.zshrc

>unset [variable_name]. => To unset/ remove environment variable

>To remove duplicate values in $PATH variables in MAC, execute below commands

```
typeset –U path
echo $PATH | tr ':' '\n'
```

>UiAutomator2:

-UiAutomator is a UI testing framework introduced by Google to facilitate automation on an Android device or emulator.
Appium leverages this UiAutomator with its own wrapper and came up with UiAutomator2 Driver to automate the Android devices.

-We need UiAutomator2 driver is required to automate android devices.
-We need xcuitest driver is required to automate iPhone devices

>appium driver list =>command shows all drivers which appium provides

```
✔ Listing available drivers
- uiautomator2 [not installed]
- xcuitest [not installed]
- espresso [not installed]
- mac2 [not installed]
- windows [not installed]
- safari [not installed]
- gecko [not installed]
- chromium [not installed]
```

>sudo appium driver install uiautomator2   => command to install appium uiautomator2 driver for automating android devices.

```
- automationName: UiAutomator2
- platformNames: ["Android"]
```

>sudo appium driver install xcuitest   => command to install appium xcuitest driver for automating iPhone devices.

```
- automationName: XCUITest
- platformNames: ["iOS","tvOS"]
```

>Location of “main.js” file:
-In Mac: /usr/local/lib/node_modules/appium/build/lib/main.js

-In Windows: C:Users/krishna/AppData/Roaming/npm/node_modules/appium/build/lib/main.js

>’main.js’ file is responsible for invoking the appium server

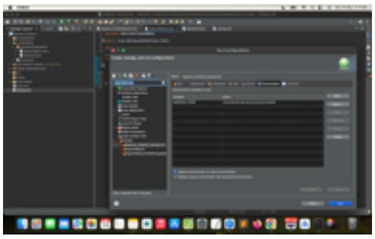
>AppiumServiceBuilder object used to programmatically start & stop the appium server in appium code

```
//code to start & stop server
            AppiumDriverLocalService service = new AppiumServiceBuilder()
                        .withAppiumJS(new File("//usr//local//lib//node_modules//
appium//build//lib//main.js"))
                        .withIPAddress(“127.0.0.1")
                        .usingPort(4723)
                        .build();

            service.start();
            service.stop();
```

OR

```
AppiumServiceBuilder builder = new AppiumServiceBuilder();
            builder.withAppiumJS(new File("//usr//local//lib//node_modules//
appium//build//lib//main.js"));
            builder.withIPAddress("127.0.0.1");
            builder.usingPort(4723);
            builder.build();

AppiumDriverLocalService service =
AppiumDriverLocalService.buildService(builder);
service.start();
service.stop();
```

>To solve the issue of programmatically starting & stopping Appium server, we need to address ANDROID_HOME environment variable issue by adding this variable within the testNG run configuration of the test case/ java class under environment variable tab as below
ANDROID_HOME=/Users/krishnabros/Library/Android/sdk



Command + Shift + 3.   => To take screenshot in Mac

>To see active ports listening in Mac

sudo lsof -i -n -P | grep TCP

>Installing appium doctor.   => Attempts to diagnose and fix common Node, iOS and Android configuration issues before starting Appium.
>npm install @appium/doctor —force —location=global
>sudo npm install appium-doctor -g

>nappium-doctor

>Locator Strategy(Java Example):
As you may expect, there are many different element locator strategies available to you, including:

- ○ Accessibility ID
- ○ Class name
- ○ ID
- ○ Name
- ○ XPath
- ○ Image (Recently Introduced)
- ○ Android UiAutomator (UiAutomator2 only)
- ○ Android View Tag (Espresso only)
- ○ IOS UIAutomation

https://kobiton.com/blog/appium-element-locator-strategies/

>Install appium inspector application

>Set capabilities in appium inspector
https://appium.io/docs/en/2.1/guides/caps/

>By.Id, .classname, .xpath etc are from selenium library
>AppiumBy.accessibilityId, .androidUIAutomator etc are from appium(java-client) library

//locators: xpath, id, classname, accessibilityId, androidUIAutomator

>Always appium server needed to be turned on through Terminal for Appium Inspector to work or connect properly with device.

```
>longPress, scroll, swipe, pinch, drag gestures documentation
https://github.com/appium/appium-uiautomator2-driver/blob/master/docs/android-mobile-gestures.md
```

```
>adb     =>. Android Debug Bridge =>  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
>adb devices.       =>to list all the connected devices/ emulators showing from Android Studio
List of devices attached
emulator-5554        device
```

```
>adb shell

>adb shell pm list packages.       =>to list all the packages

>adb start-server
>adb kill-server

>emulator -list-avds       => to list all created emulator devices in Android Studio

adb shell dumpsys package com.android.webview
```

```
>To get information about the currently open application
>To find app package and app activity of android app
>App Package and App Activity
adb shell dumpsys window | grep -E 'mCurrentFocus'    => Mac
adb shell dumpsys window | find 'mCurrentFocus'          => Windows
```

```
Activity activity = new Activity(String appPackage, String appActivity)
```

Ex:

```
Activity activity = new Activity("io.appium.android.apis", "io.appium.android.apis.preference.PreferenceDependencies");
driver.startActivity(activity).    => but startActivity() is deprecated above Appium 2.0.0, so use below one
(OR)
((JavascriptExecutor) driver).executeScript("mobile: startActivity", ImmutableMap.of(
                          "intent", "io.appium.android.apis/io.appium.android.apis.preference.PreferenceDependencies"
));
```

https://github.com/appium/appium-uiautomator2-driver/#mobile-startactivity

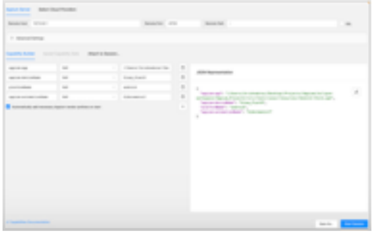
>To install app into android emulator present in android studio

```
adb install /Users/krishnabros/Documents/RahulShettyCourse/AppiumCourse/General-Store.apk
```

>For Appium inspector session to start
-Start Appium server using ‘appium’ command in terminal
-Start Android studio emulator
-Then only Appium inspector session gets started

>Appium Inspector

```
{
  "appium:app": "//Users//krishnabros//Desktop//Projects//Appium//eclipse-workspace//Appium_Project2//src//test//java//resources//General-Store.apk",
  "appium:deviceName": "Vinay_Pixel8",
  "platformName": "android",
  "appium:automationName": "UiAutomator2"
}
```

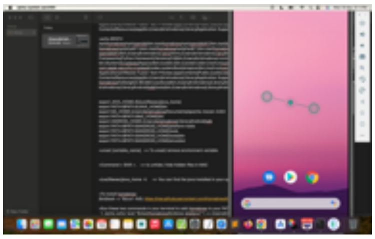


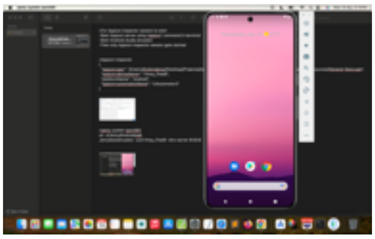
>To solve internet / wifi connection in Android emulator in Mac:

>qemu system aarch64(Android Emulator):

```
cd ~/Library/Android/sdk
./emulator/emulator -avd Vinay_Pixel8 -dns-server 8.8.8.8
```

Qemu present in below path
/Users/krishnabros/Library/Android/sdk/emulator/qemu/darwin-aarch64/qemu-system-aarch64





To capture screenshot in Mac:
Command + Shift + 3   -> open image -> copy image

>brew install Carthage.   =>In order to launch web driver agent on your Mac OS and not required on android
>which Carthage
/opt/homebrew/bin/Carthage

>xcode-select --install.   =>to download command line tools for Xcode

>Command + Shift + G.   => In Mac finder, we can directly navigate to specific folder path

>Webdriver Agent will help launching iOS app in iPhone devices.

>iPhone 15 Pro Max capabilities:

```
{
  "appium:app": "/Users/krishnabros/Library/Developer/Xcode/DerivedData/UIKitCatalog-bjtytzoitisvauerejdwyprtaolr/Build/Products/Debug-iphonesimulator/UIKitCatalog.app",
  "appium:automationName": "XCUITest",
  "appium:deviceName": "iPhone 15 Pro Max",
  "appium:platformVersion": "17.5",
  "platformName": "iOS"
}
```

OR

```
{
  "appium:app": "/Users/krishnabros/Desktop/Projects/Appium/Appium-Demo-Apps/UIKitCatalog.app",
  "appium:automationName": "XCUITest",
  "appium:deviceName": "iPhone 15 Pro Max",
  "appium:platformVersion": "17.5",
  "platformName": "iOS"
}
```

OR

```
{
  "app": "/Users/krishnabros/Desktop/Projects/Appium/Appium-Demo-Apps/TestApp 3.app",
  "automationName": "XCUITest",
  "deviceName": "iPhone 15 Pro Max",
  "platformVersion": "17.5",
  "platformName": "iOS"
}
```

>Locators to identify objects/ elements in iOS apps

○ id

○ accessibilityId

○ Xpath

○ classname

○ iOSClassChain

○ iOSPredicateString

GitHub token: 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s