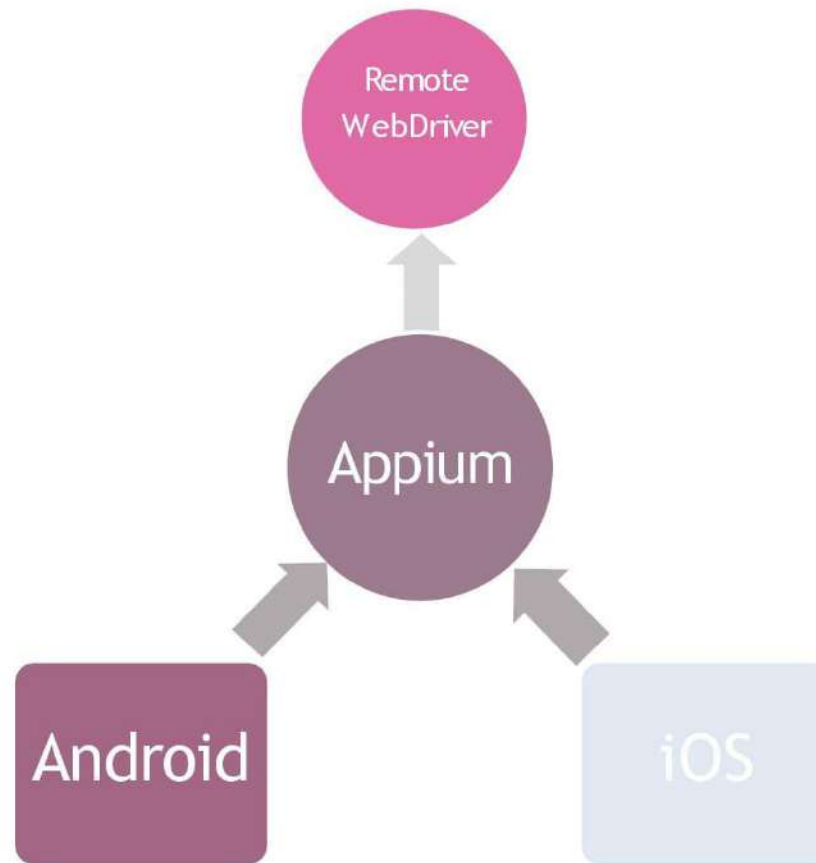


MOBILE APP AUTOMATION USING APPIUM

LOKESH KUMART

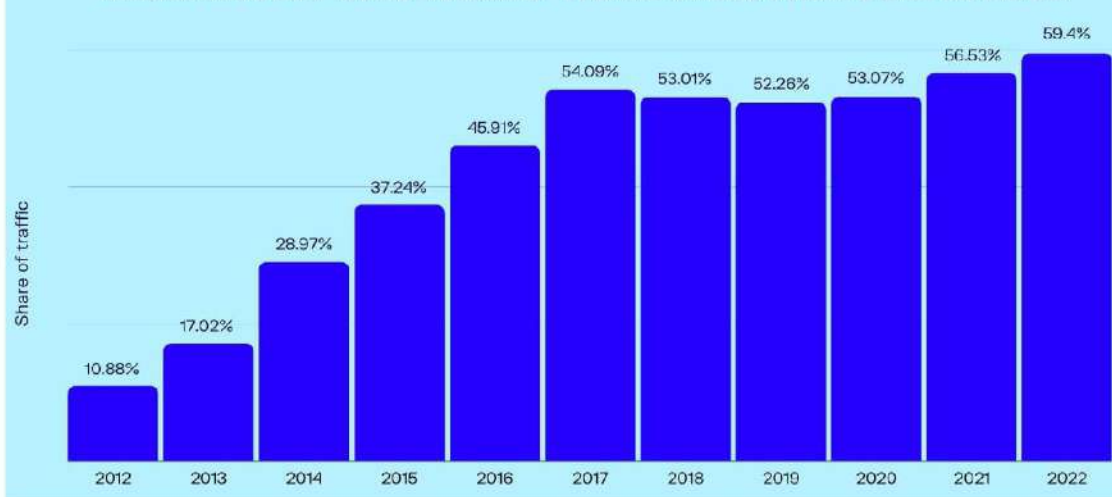


PRE-REQUISITES FOR LEARNING APPIUM



WHY SHOULD I KNOW MOBILE AUTOMATION?

Global Mobile Phone Website Traffic Share From 2012 to 2022



55%

of online booking comes from tablets ¹



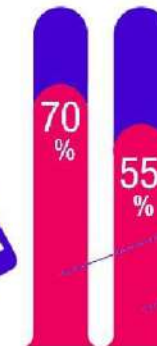
96%

of online shoppers use their tablets around the house during leisure time ²



68%

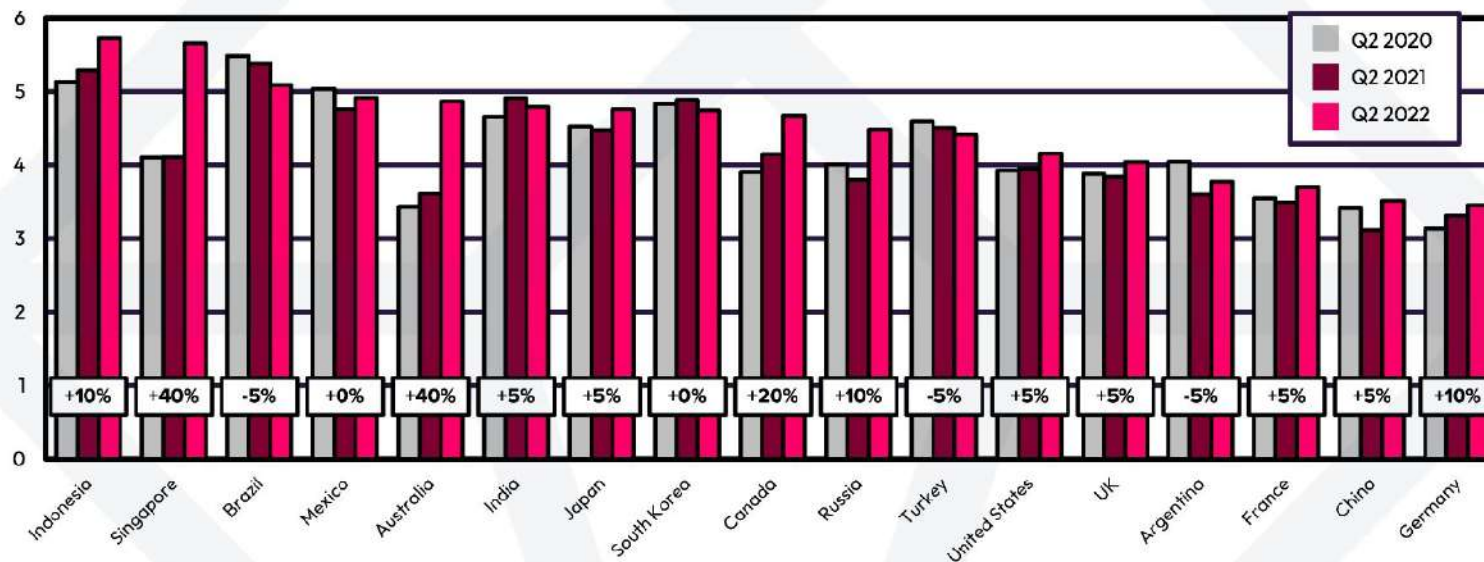
of smartphone users visit mobile sites while on-the-go ³



- Over 70% claimed that a mobile optimized website is important to them ⁴
- 55% will skip over a website that is not mobile optimized ⁵

WHY SHOULD I KNOW MOBILE AUTOMATION?

Avg. Daily Hours Spent in Apps Select Markets

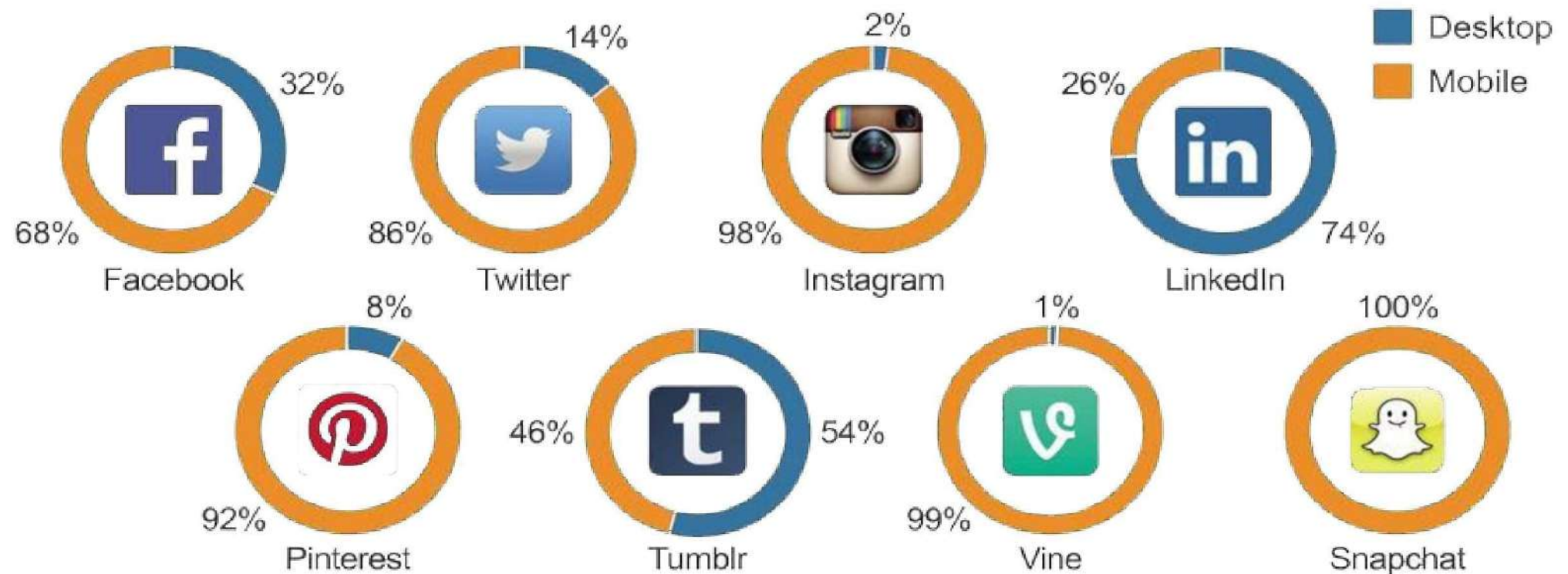


Note: Android phones. All estimates from data.ai.
Growth based on 2-year % change (Q1 2022 vs. Q1 2020).

SOCIAL NETWORK ACTIVITY SURVEY

Social Network Activity: Mobile vs. Desktop

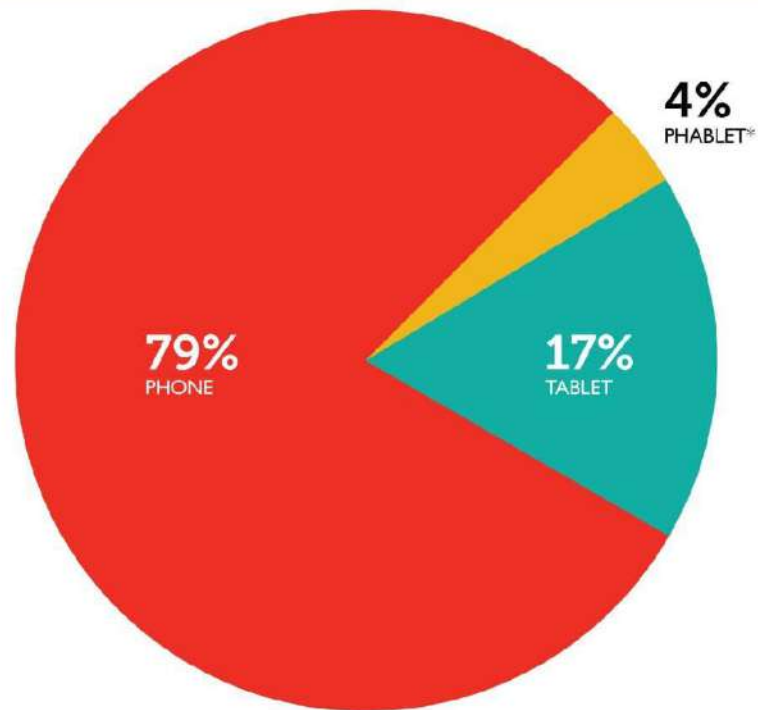
% of time spent on social networks in the United States, by platform*



KEY CHALLENGES

- The number of models and types of smartphones on the market today is already large, with more always entering the market.
- Each device tends to have differences, and deciding which devices to test can be a challenge.
- While everyone wants to support as many devices as possible, adding devices can greatly increase testing time as well as cost.
- In addition to the ever-increasing number of devices on the market, there are also a number of OS platforms, and many companies are developing for an increasing number of them.
- While continuous integration has become standard for most web development projects, the same becomes for mobile development as well.

DEVICES - MARKET DISTRIBUTION



*Devices with a screen size > 5in but < 7in
(primarily driven by the Samsung Galaxy Note and Galaxy Note 2, which comprise more than 99% of the category)

ANDROID DEVICE FRAGMENTATION

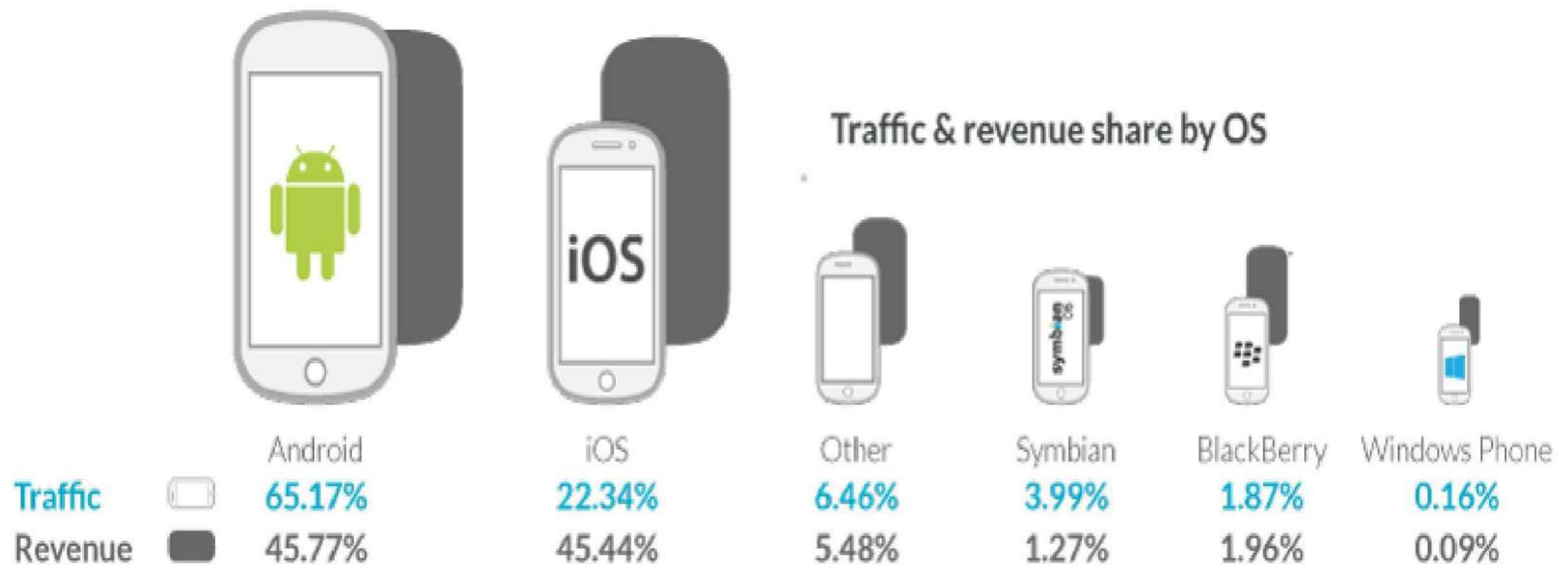


REAL DEVICES VS. EMULATORS

#	Parameter	Real Device	Emulator
1	Cost Effective	Very expensive considering the type of smart phones and range of testing	Very low or no cost (GenyMotion)
2	Chipset	Chipsets in particular can also provide vastly different user experiences between high and low end devices	Push notifications, geo-location, orientation, and other functionality are impossible to test adequately without a real device.
3	Battery	Each real devices have different battery specifications and discharge rates.	Battery cannot be emulated
4	Screen size, Resolution	Multitude of screen sizes & resolutions	Emulator may still not give the information on brightness, saturation etc.
5	Phone interruptions + Behaviour	Swipe, Pinch /Expand, Tap /Double Tap, Long Press	An emulator cannot adequately show how network related interruption events will react with your application
6	Parallel Execution	Not possible /Limited	Relatively easy to build and run



OS MARKET SHARE



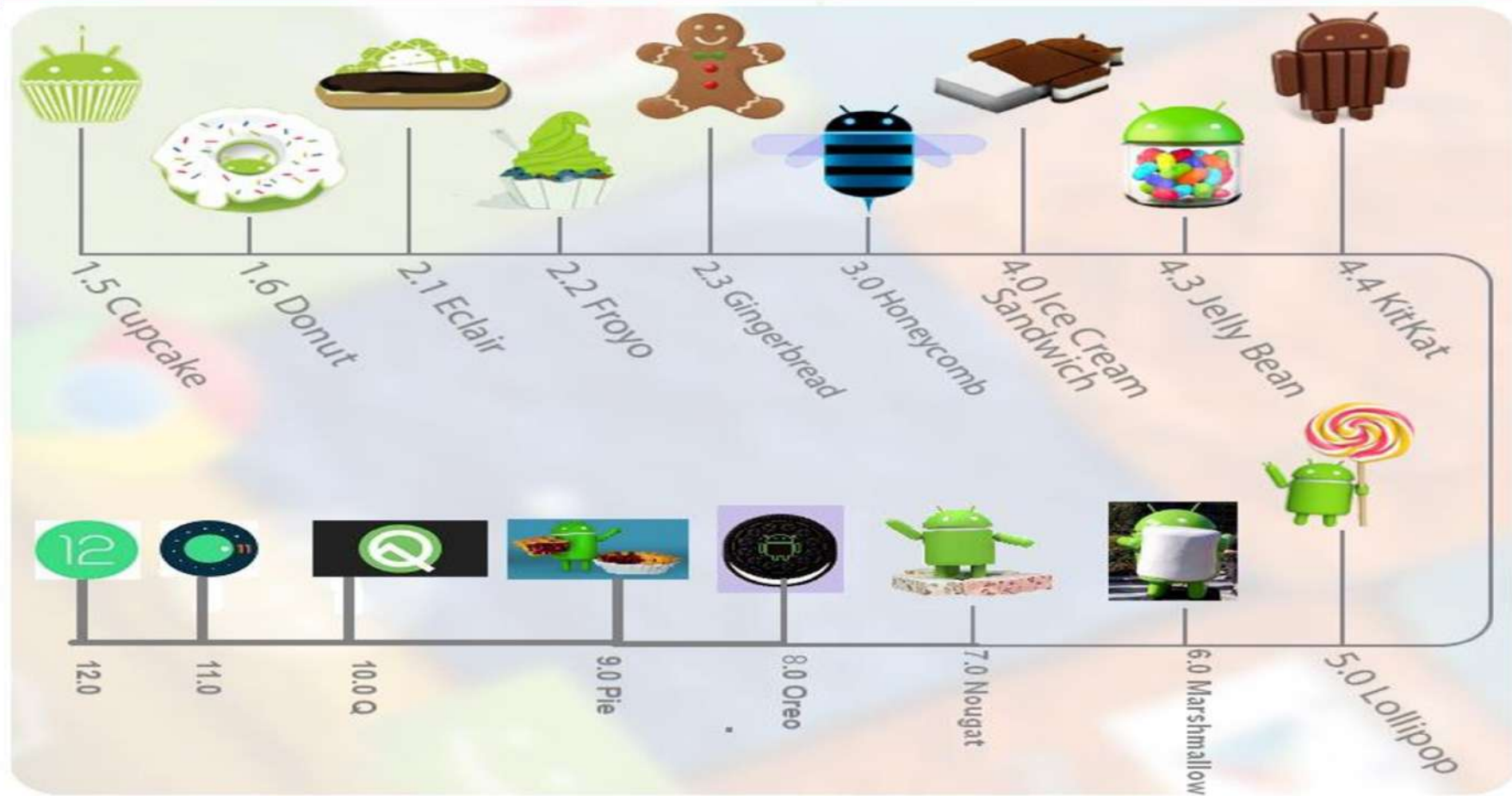
MOBILE AUTOMATION - PLATFORM & APPLICATIONS



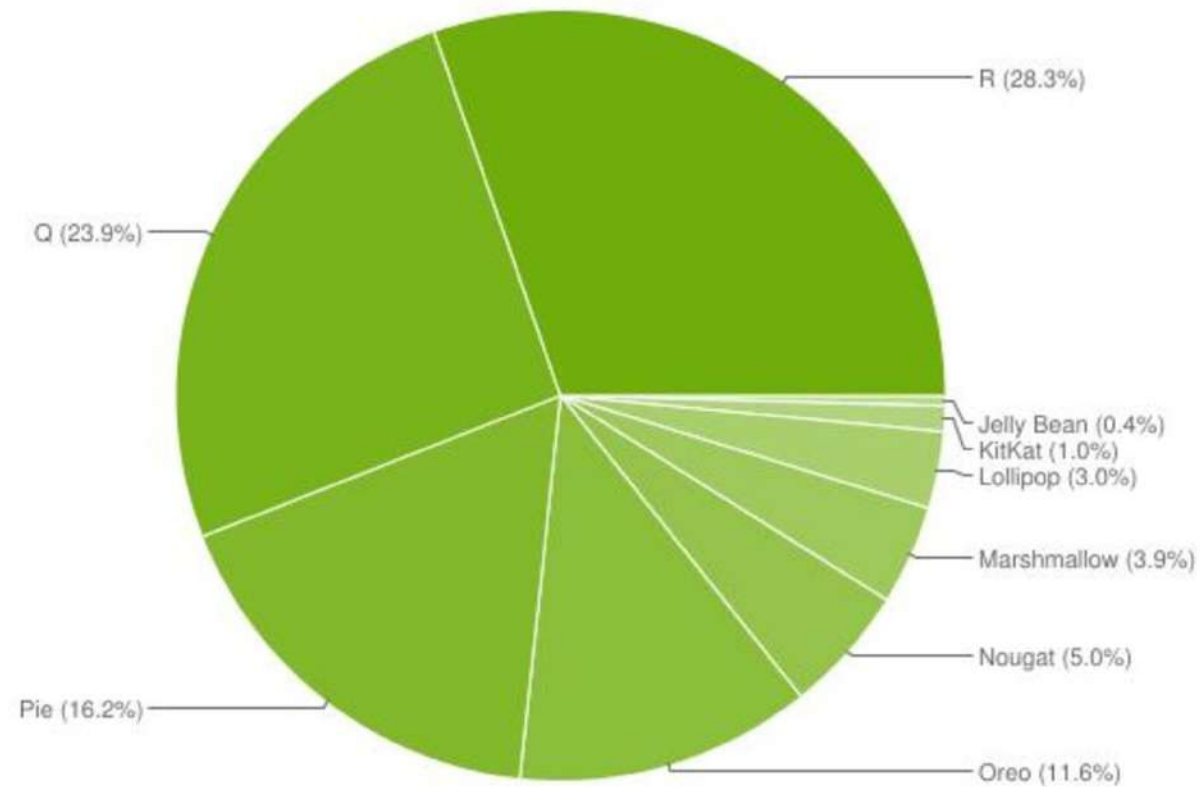
BENEFITS OF MOBILEAUTOMATION

1. Higher test coverage
2. Faster time to market
3. Reusability of code
4. Improved accuracy
5. Eliminate Human error

ANDROID VERSIONS

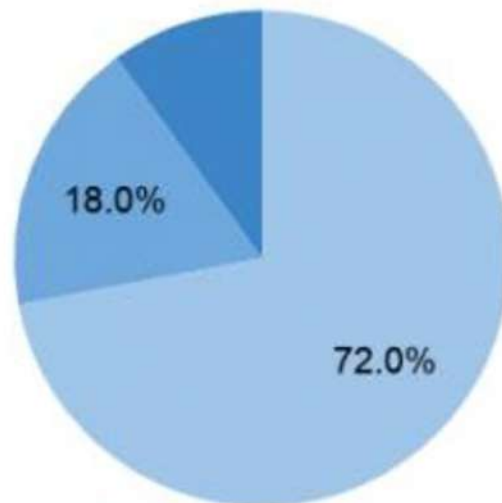


ANDROID VERSIONS DISTRIBUTION



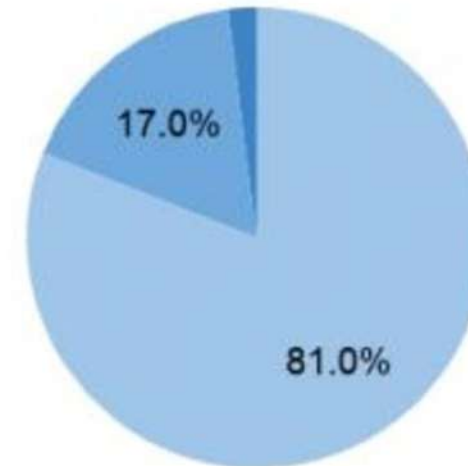
IOSVERSIONS DISTRIBUTION

72% of all devices use iOS 14



● iOS 14 ● iOS 13
● Earlier OS Versions

81% of all devices introduced in the last four years use iOS 14




● iOS 14 ● iOS 13
● Earlier OS Versions

NATIVE VS. HYBRID VS. MOBILE



TOOLS & FEATURES - A COMPARISON STUDY

Tools		Rule 1	Rule 2	Rule 3	Rule 4
Calabash-ios		✗	✗	✗	✓
Frank		✗	✗	✗	✗
UIAutomation		✗	✗	✗	✗
ios-driver		✓	✓	✓	✗
KeepItFunctional		✓	✗	✗	✗
Calabash-android		✗	✗	✗	✗
MonkeyTalk		✗	✗	✗	✗
Robotium		✗	✗	✓	✗
UiAutomator		✗	✗	✗	✗
Selendroid		✗	✓	✓	✗
Appium	 	✓	✓	✓	✓

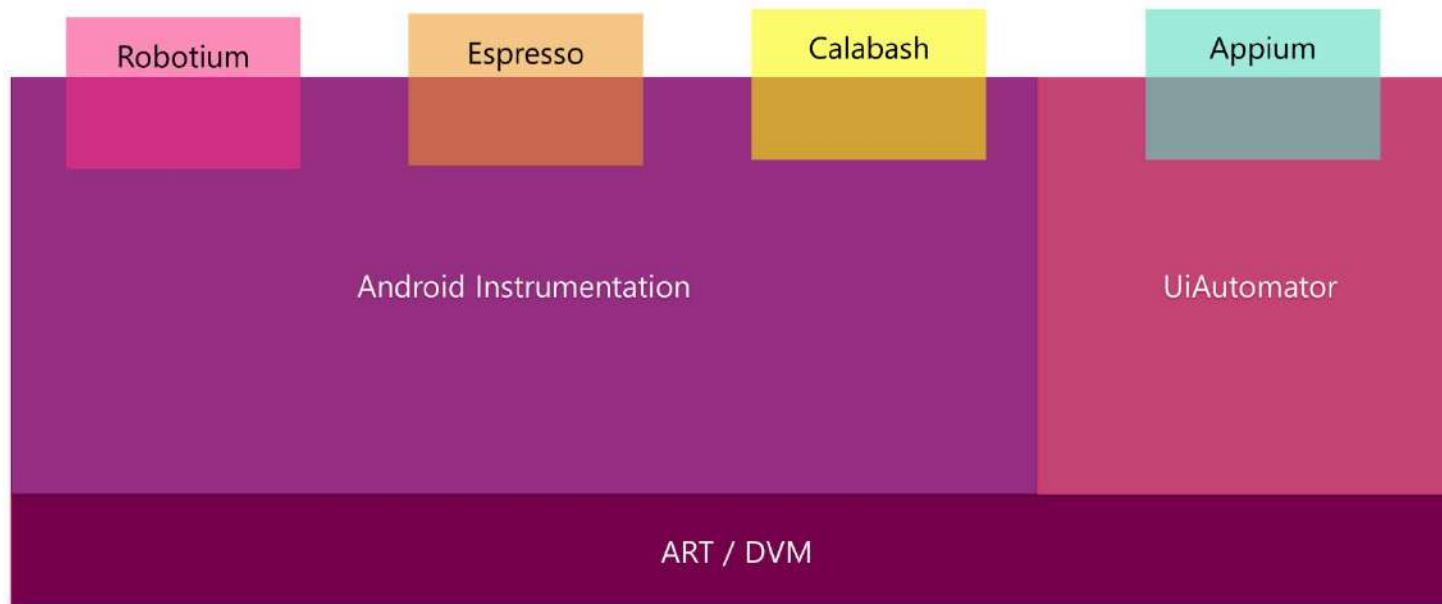
The app, which is to be automated is of same code as in production /store

The tool scripting supported by any language and can be of any frameworks

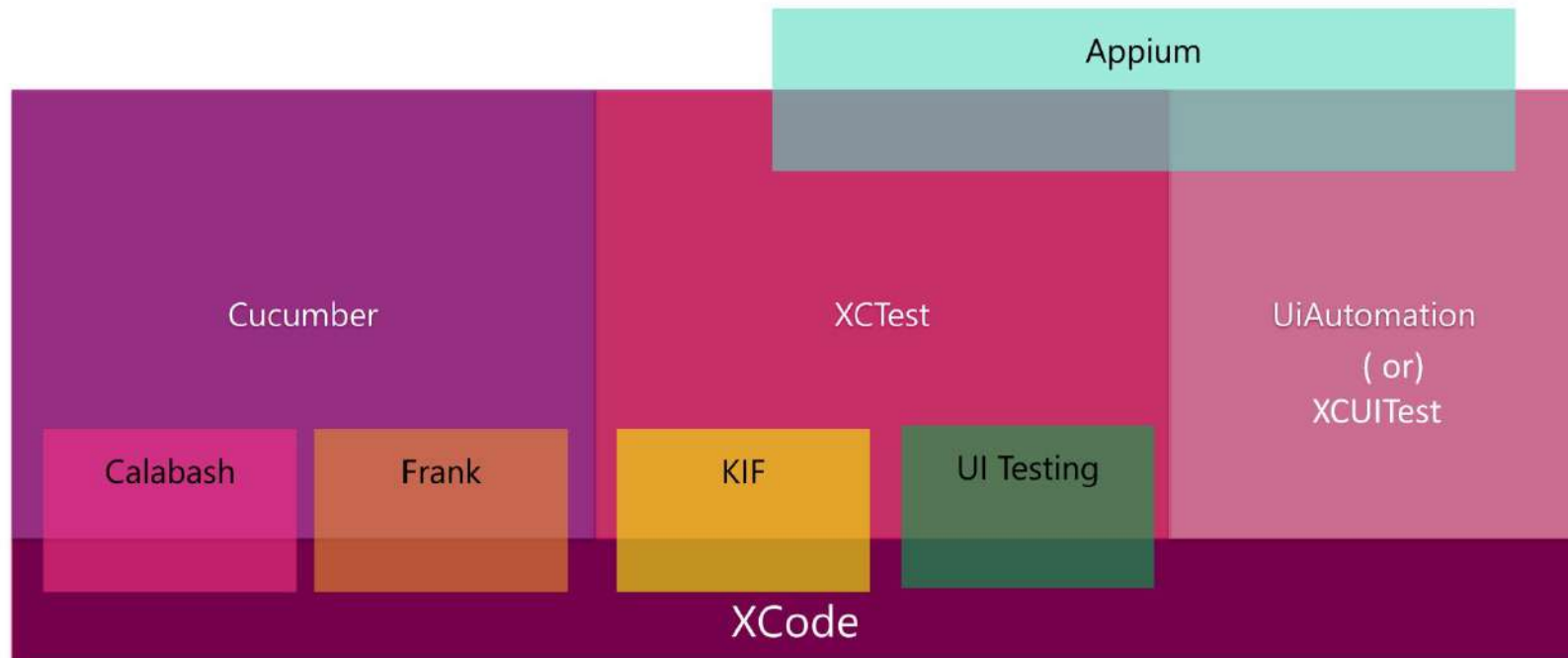
The tool using a standard automation specification an a large and thriving open-source community effort d API.

The tool has a large and thriving open-source community effort for support.

FAMILY OF ANDROID TEST FRAMEWORKS



FAMILY OF IOSTEST FRAMEWORKS



WHAT IS APPIUM?

Appium is an open source, cross-platform test automation tool for native, hybrid and mobile web apps, tested on simulators (iOS), emulators (Android), and real devices (iOS, Android). Appium opens up the possibility of true cross-platform native mobile automation.

- You don't have to recompile your app or modify it in any way, due to use of standard automation APIs on all platforms.
- You can write tests with your favourite dev tools using any WebDriver-compatible language such as Java, Objective-C, JavaScript with Node.js (in promise, call-back or generator flavors), PHP, Python, Ruby, C#, Clojure, or Perl with the Selenium WebDriver API and language-specific client libraries.





In 2014, Sauce Labs released Appium 1.0 and ended the year with over **135 contributors** and **5,000 commits**.



Native and hybrid app testing is trending up on Sauce!



In 2013-2014 versus 2014-2015, we saw an **increase of 1023% of mobile tests run** on our platform year over year.

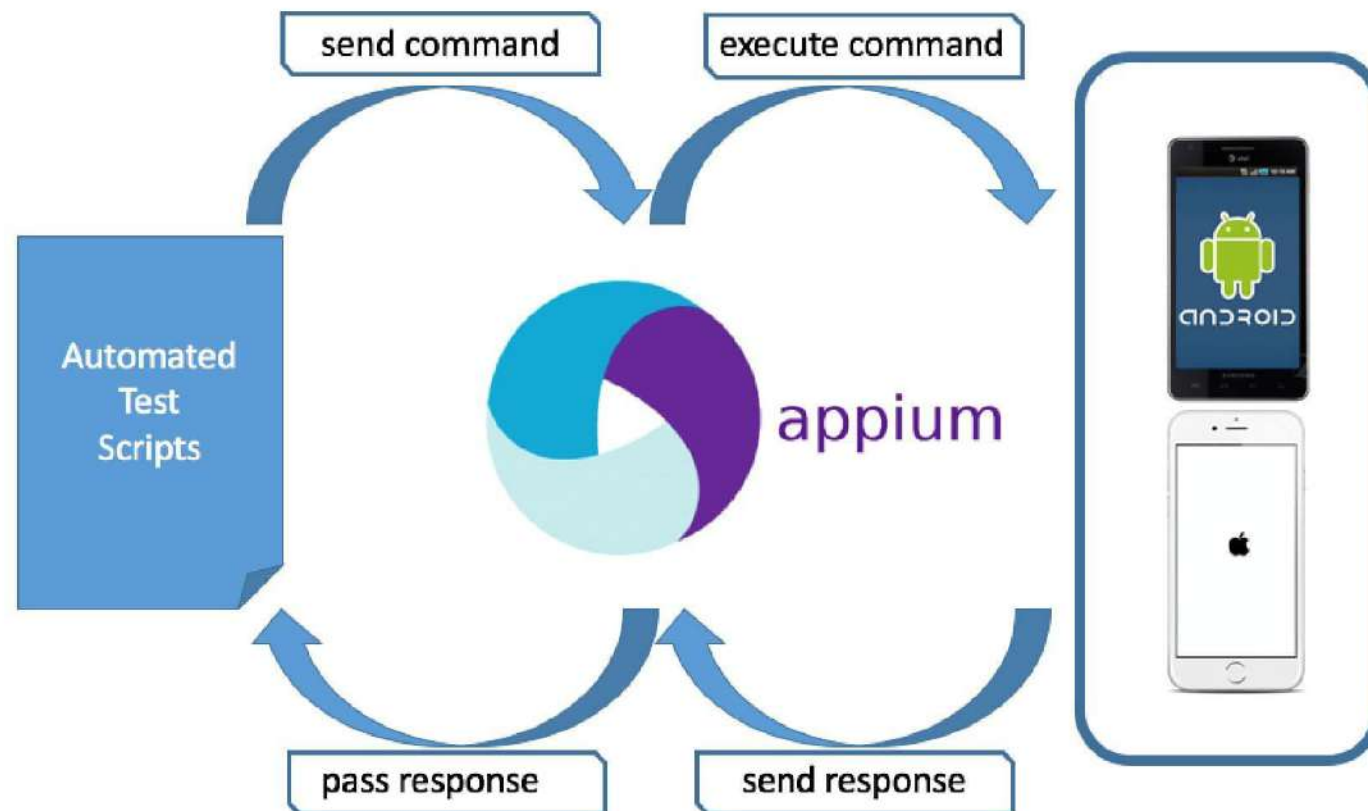


Android native + hybrid mobile tests run on Sauce **increased 993%** year over year (2013-2014 versus 2014-2015)



iOS tests increased a staggering **1760%**!

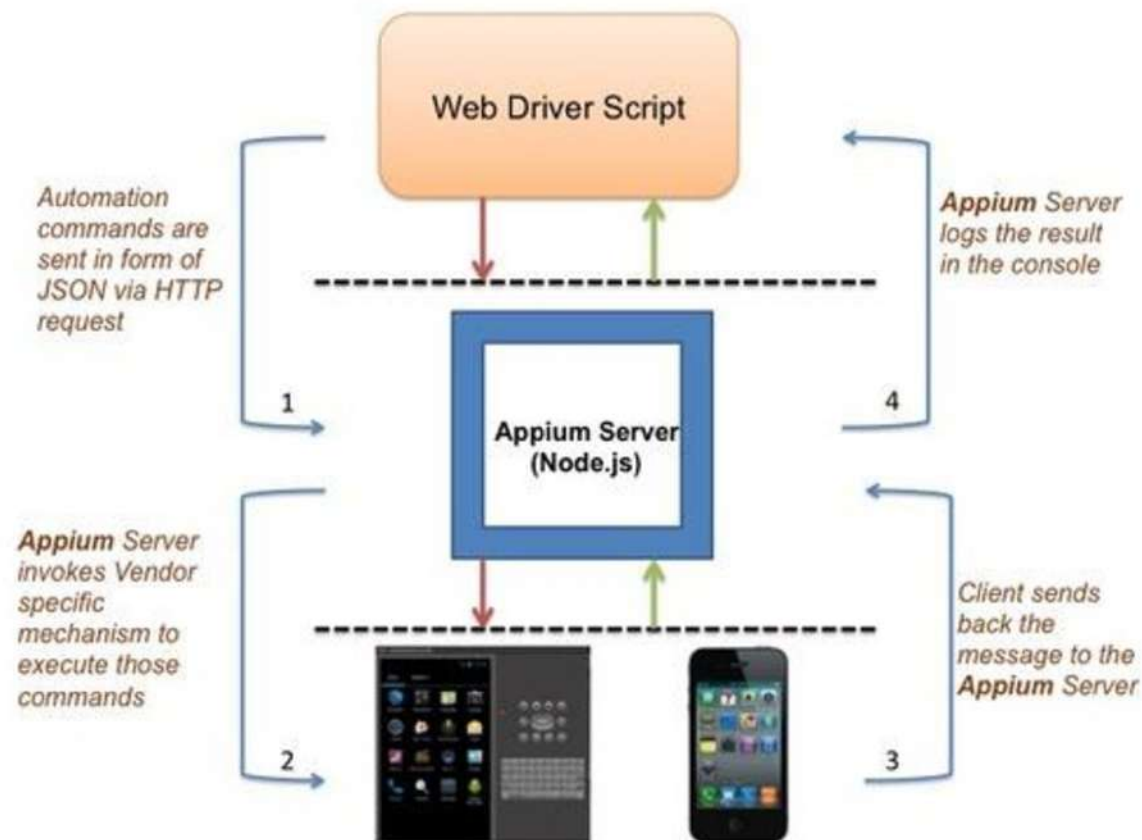
HIGH LEVEL - WORKFLOW OF APPIUM - Native App



WORKFLOW OF APPIUM - Native App

1. From Web-driver, Automation Commands are sent in form of JSON via HTTP request to Appium Server.
2. Appium Server invokes Vendor specific mechanism to execute those commands on the Mobile-Device.
3. Mobile Device (Client) sends back the message to the Appium Server.
4. Appium Server logs the result in the console of the Web Driver.

WORKFLOW OF APPIUM - Native App

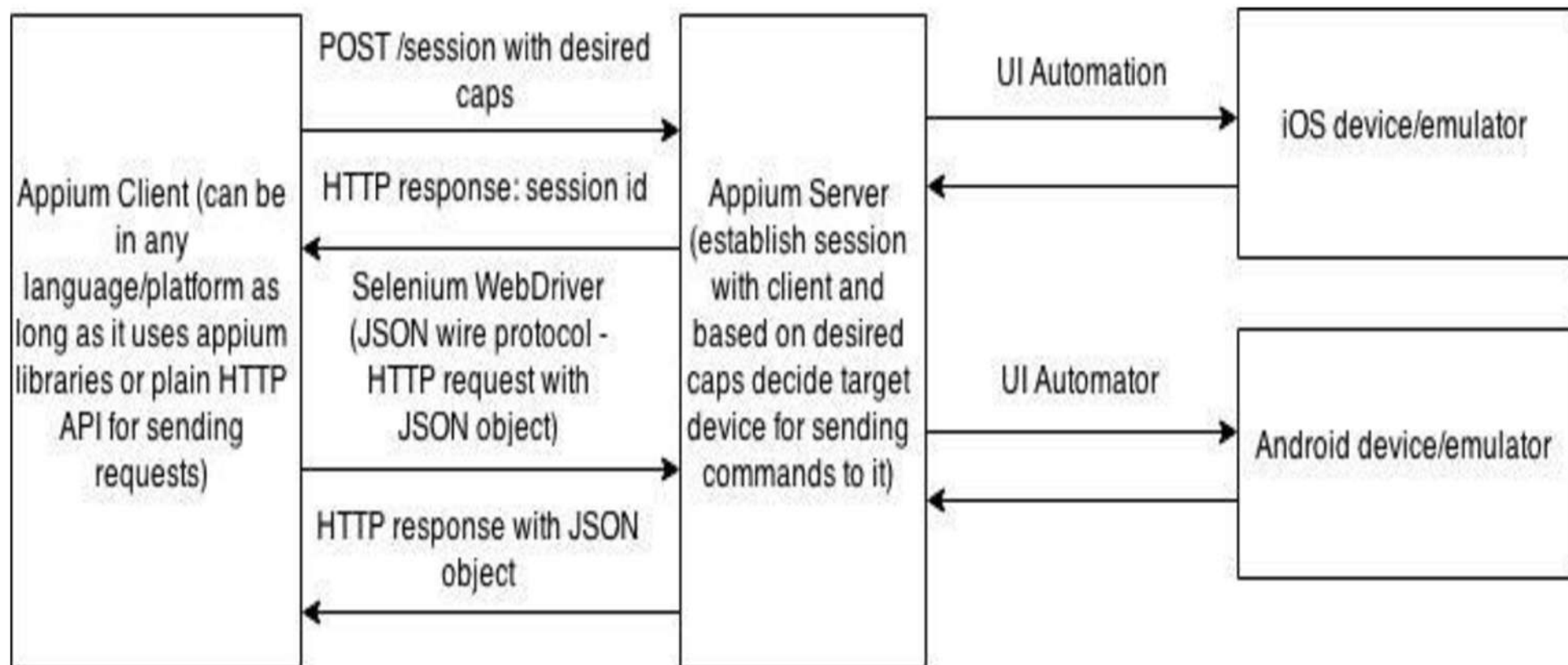


VENDOR PROVIDED FRAMEWORKS IN APPIUM

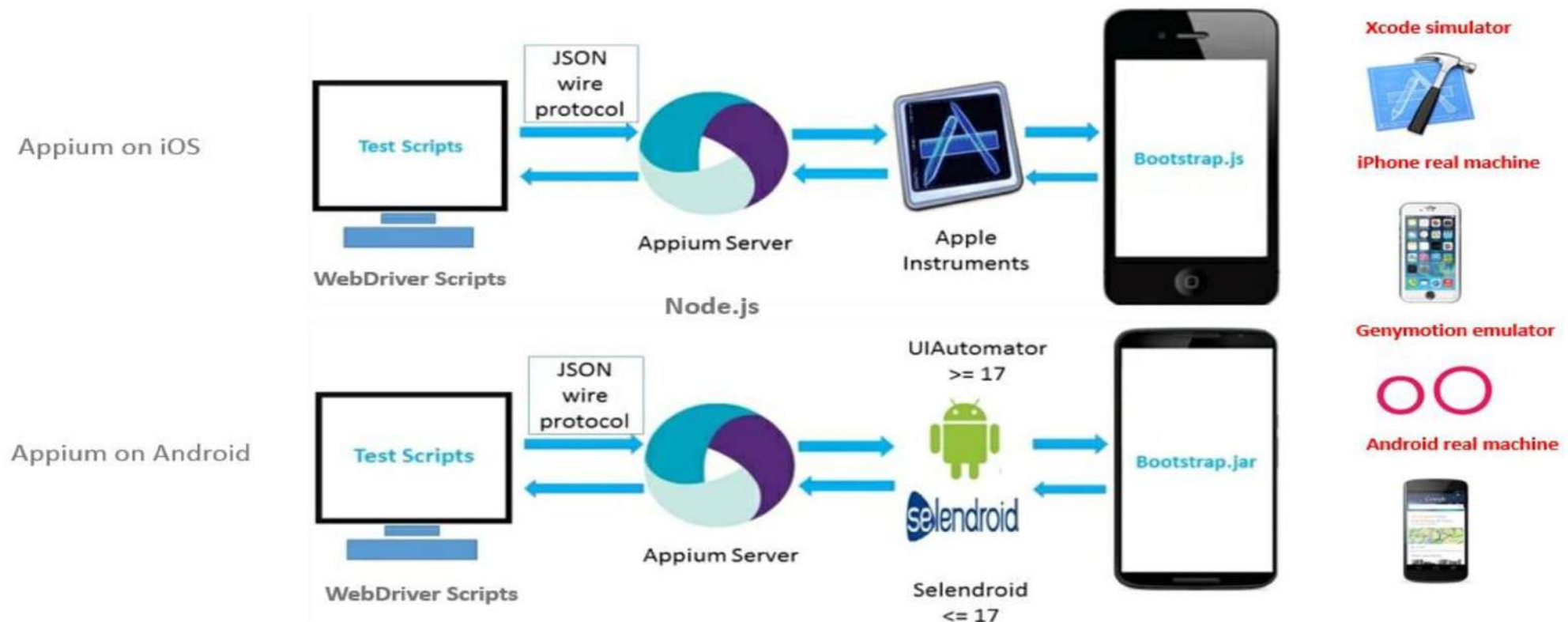
The vendor-provided frameworks used are:

- iOS v9.3 and above: Apple's **XCUITest (UIAutomation)**
- iOS v9.2 and lower: Apple's **XCTest**
- Android v4.2 to v6.0: Google's **UiAutomator1**
- Android v4.2 and above: Google's **UiAutomator2**
- Android v10.0 and above: Google's **Espresso**
- Android v2.3 to v4.1: Google's Instrumentation, **Selendroid**

WORKFLOW OF APPIUM - Native App

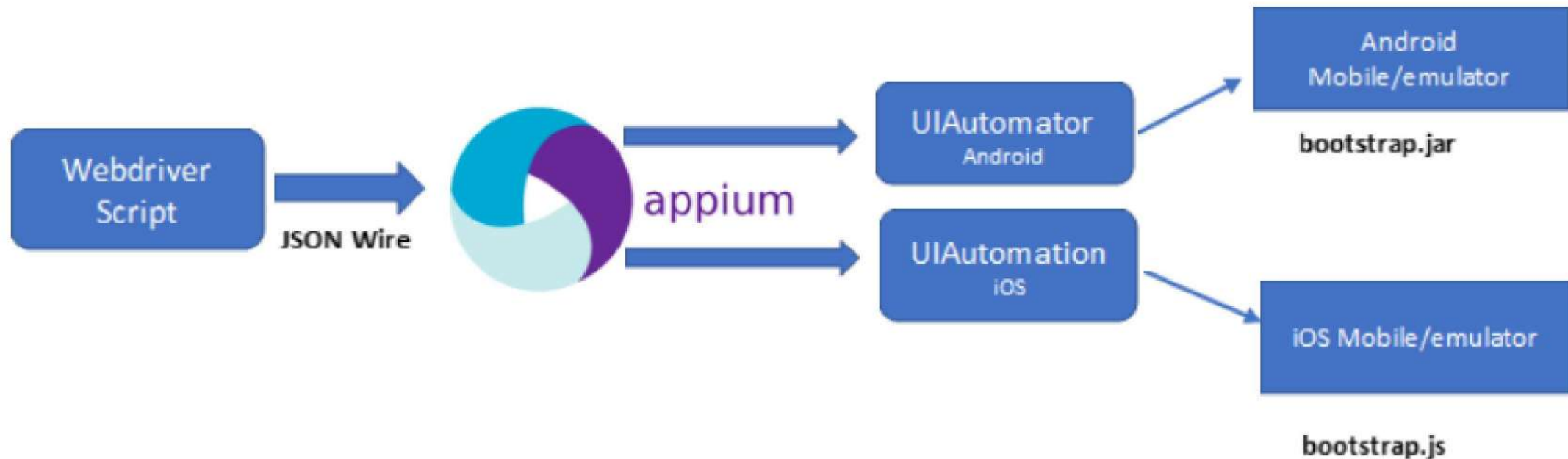


APPIUM - WAY OF WORKING - Native App

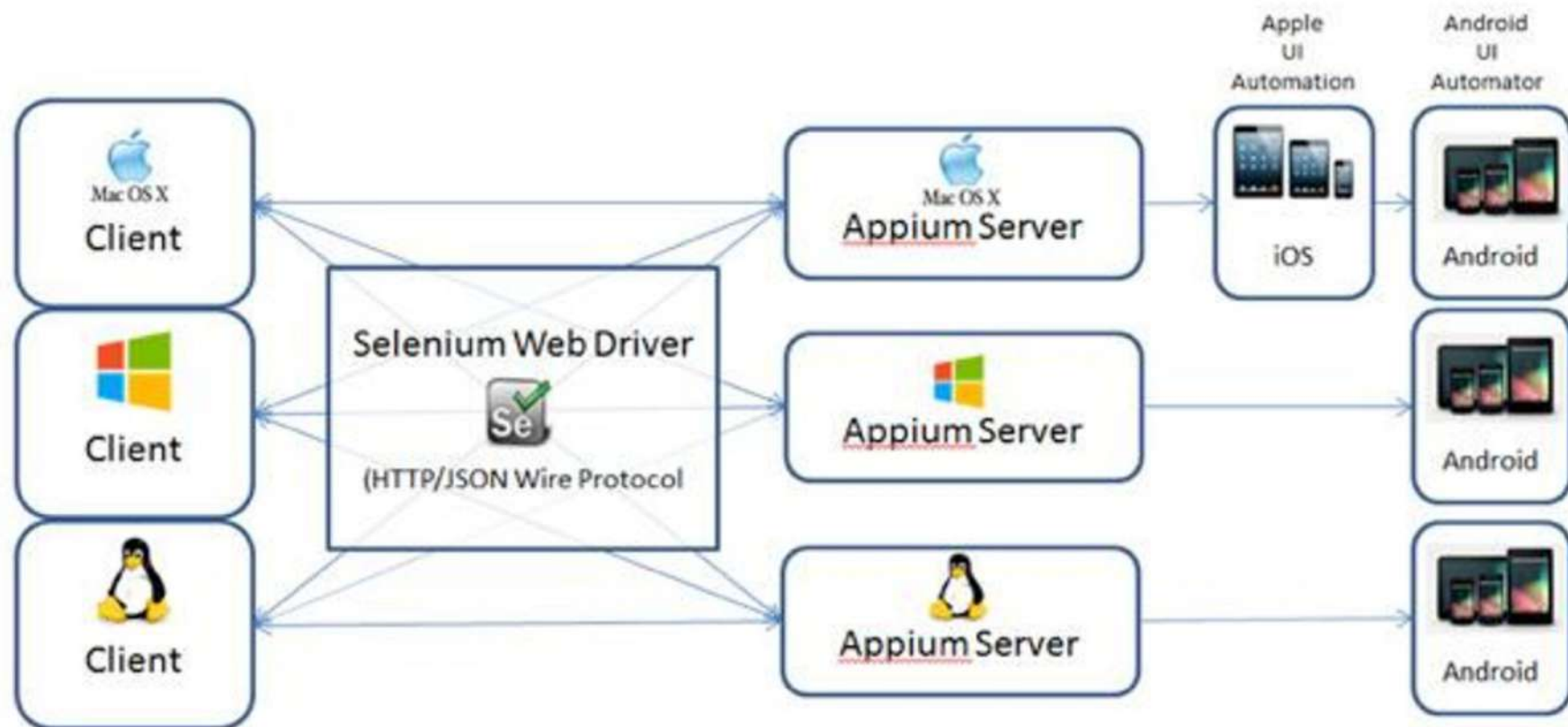


** From version 1.6, Appium added support to XCUITest for iOS10 / Xcode 8, and UiAutomator2 support for Android*

APPIUM - WAY OF WORKING - Native App

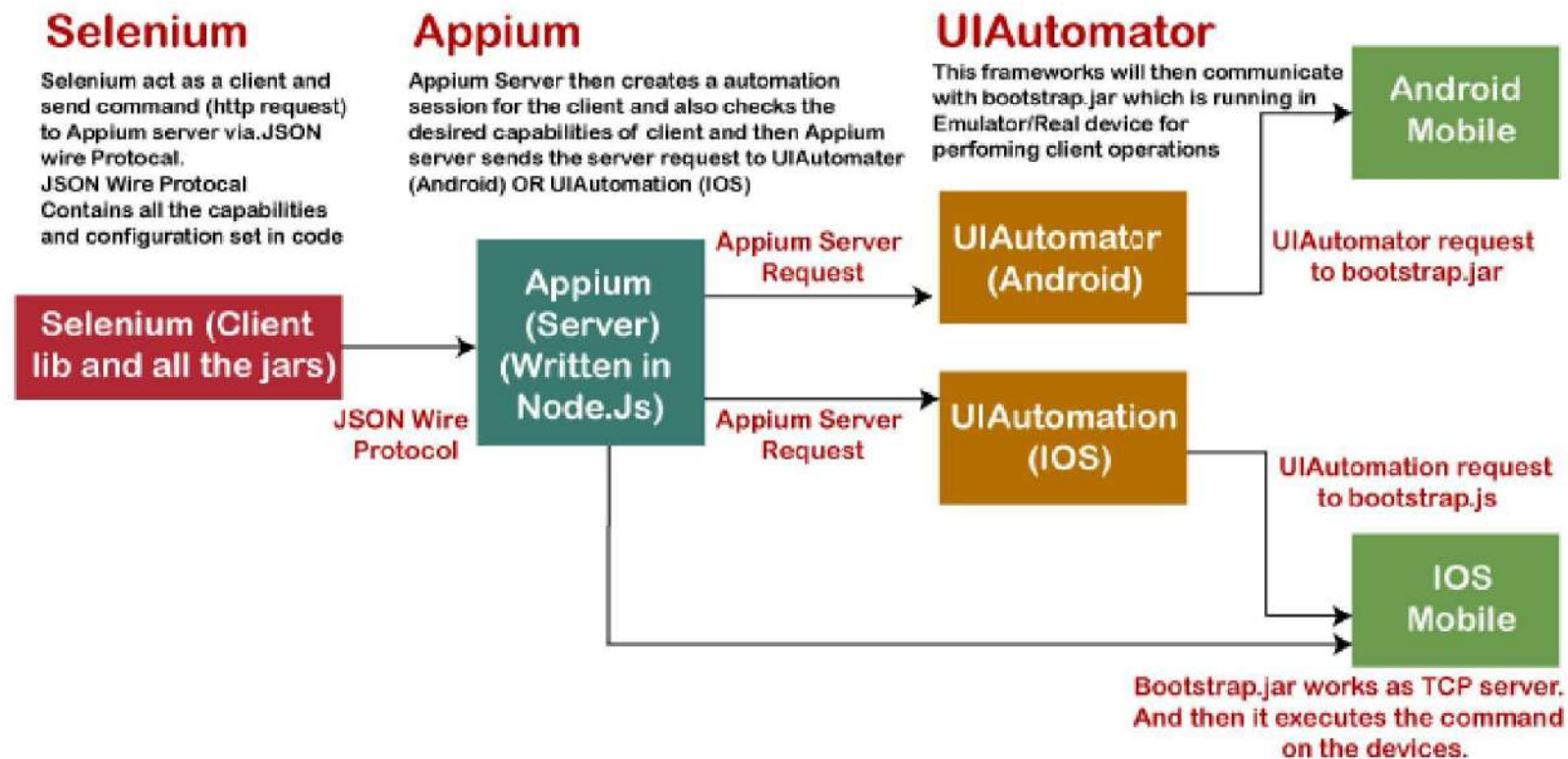


APPIUM - WAY OF WORKING - Native App

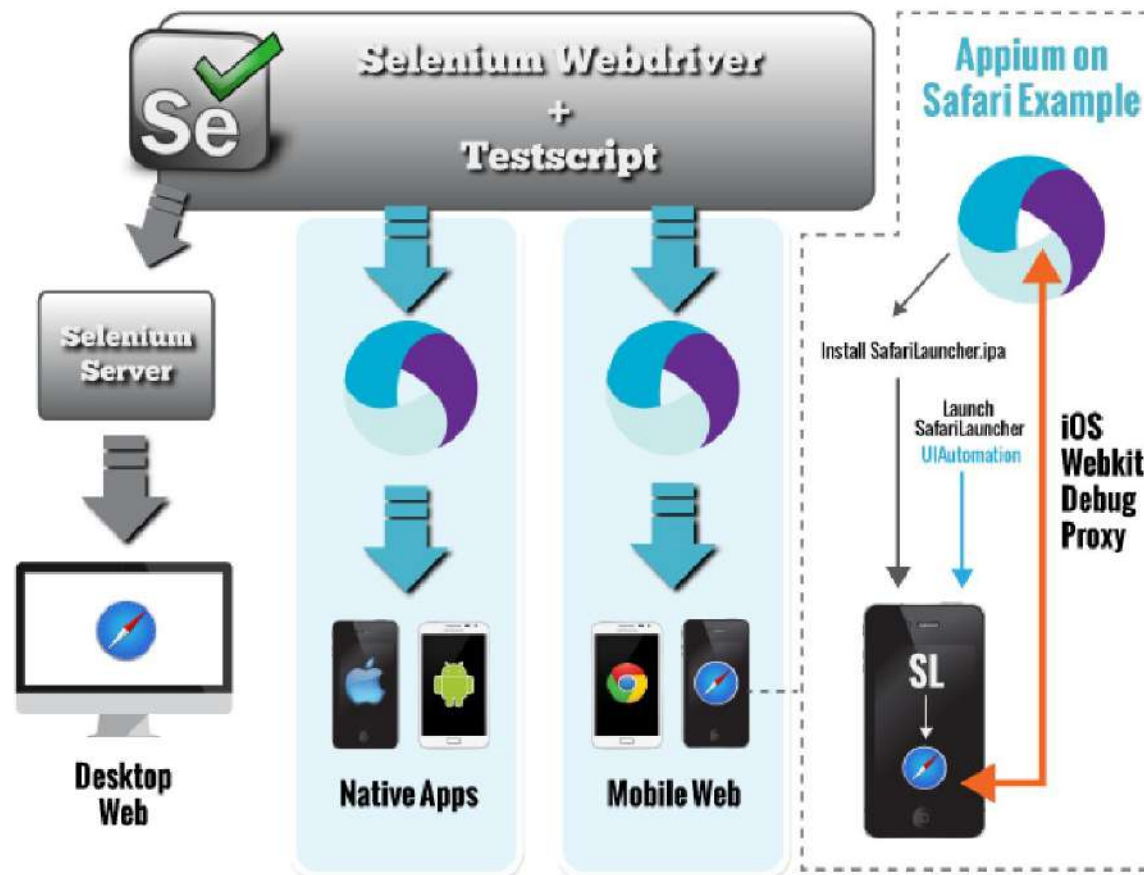


APPIUM - WAY OF WORKING - Native App

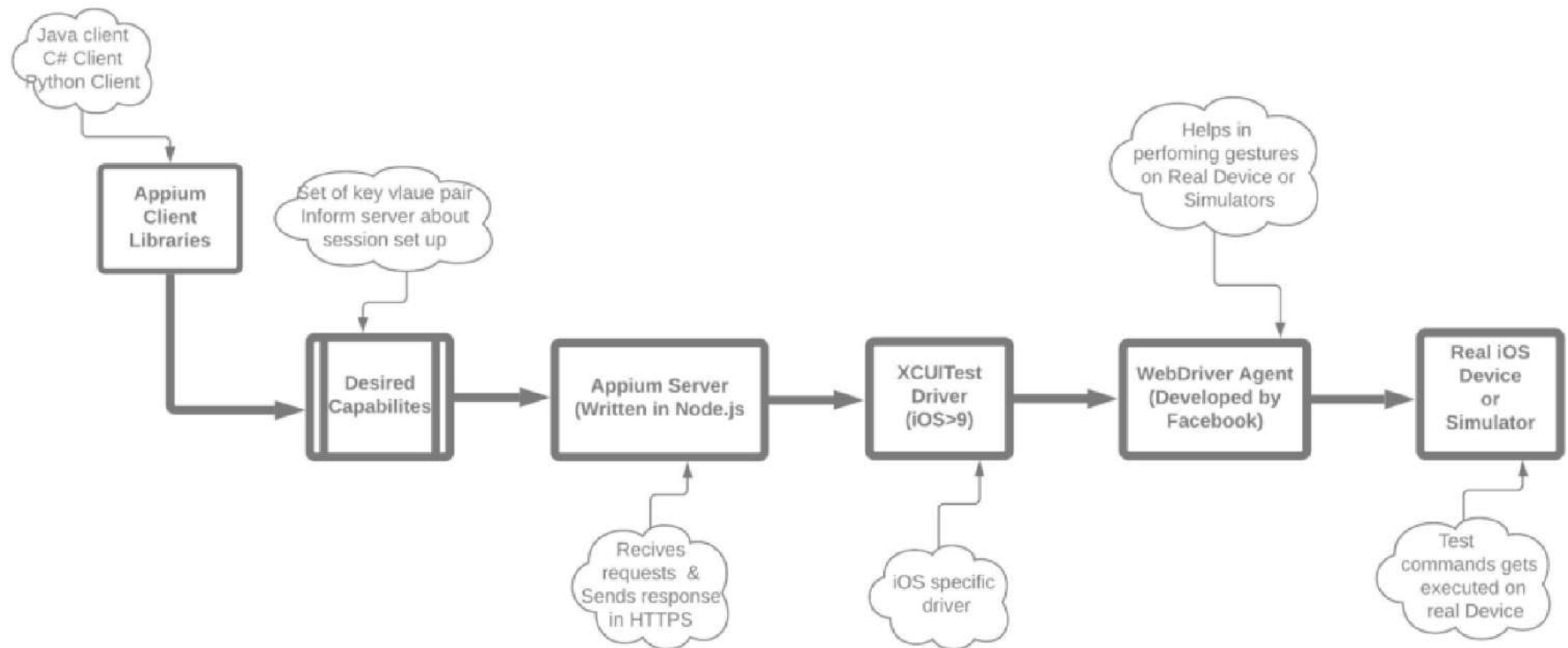
Appium Architecture



APPIUM - WAY OF WORKING - Mobile Web



APPIUM - iOS-WAY OF WORKING - Native App



APPIUM - REFERENCE LIBRARY USED

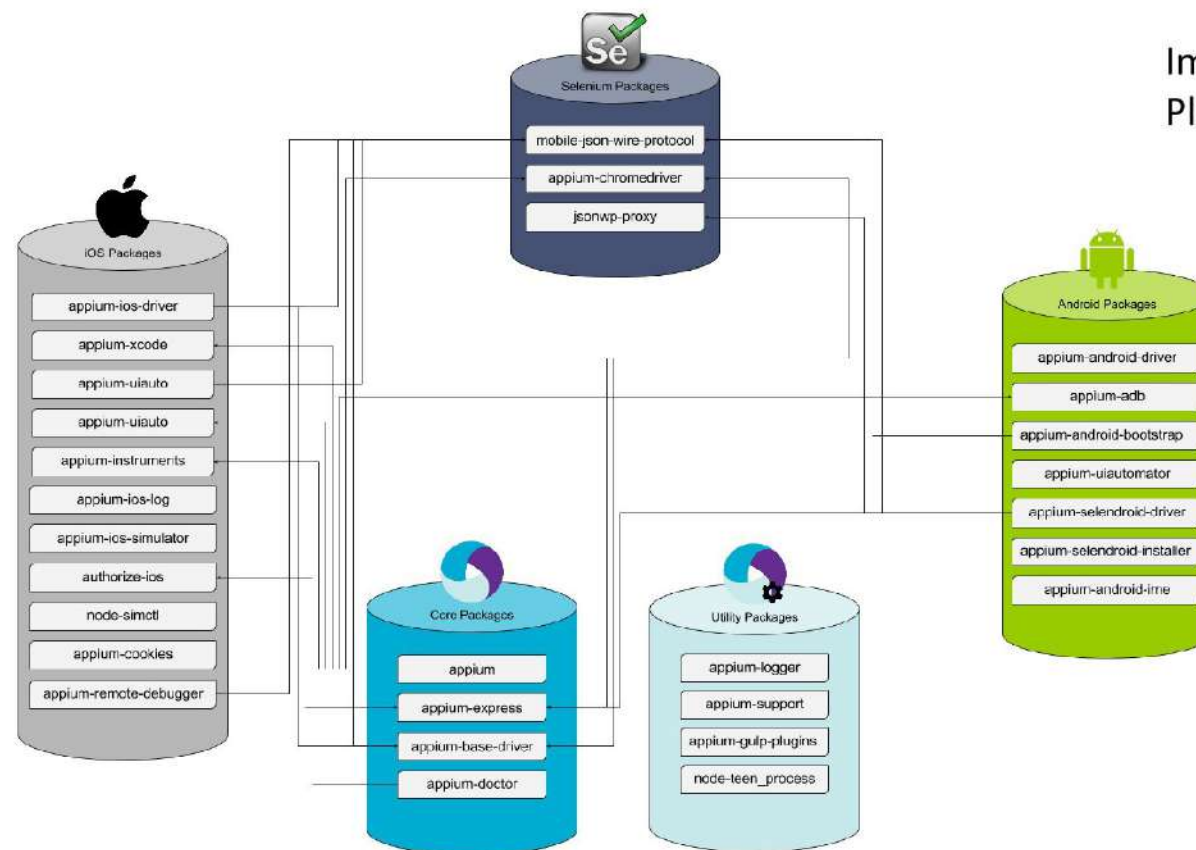
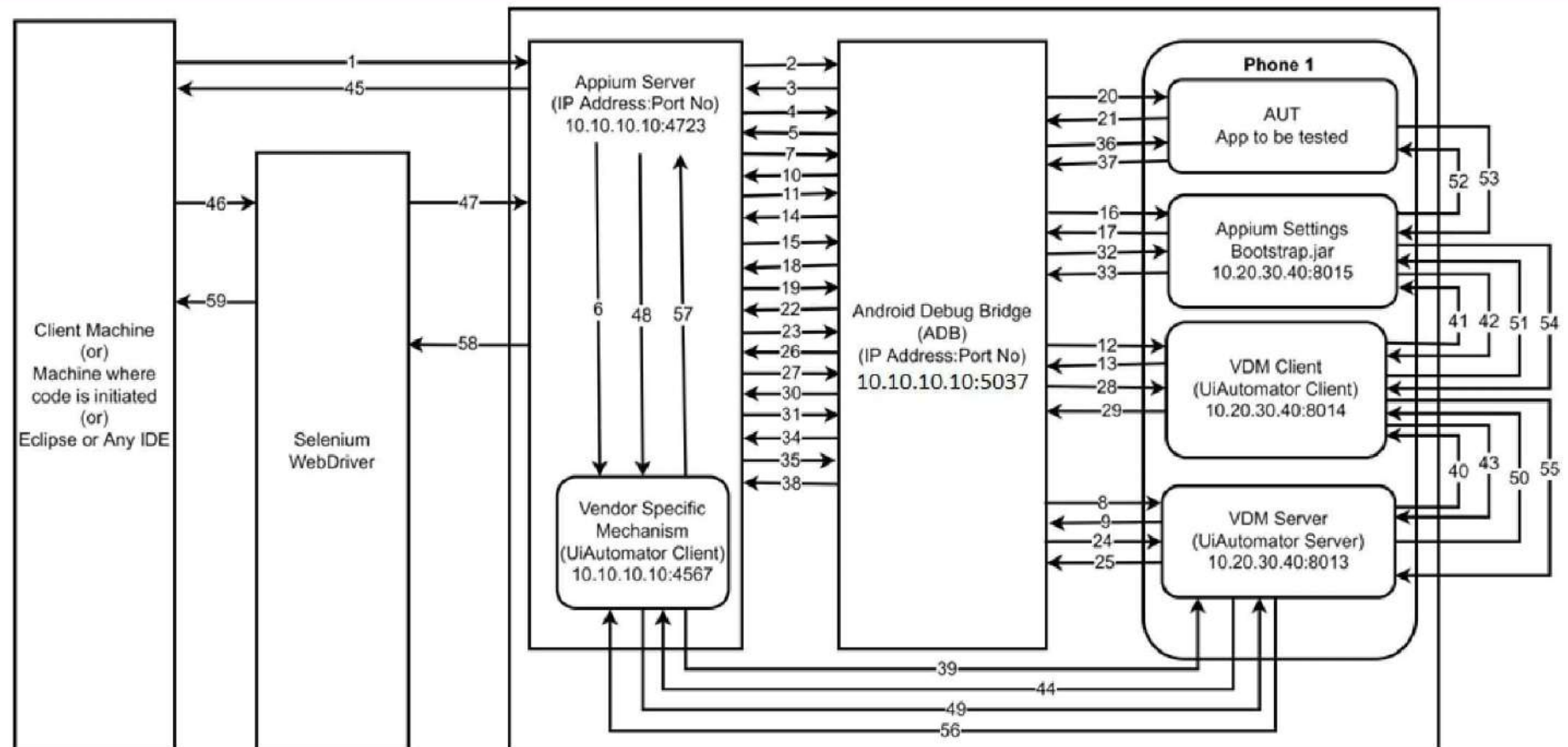


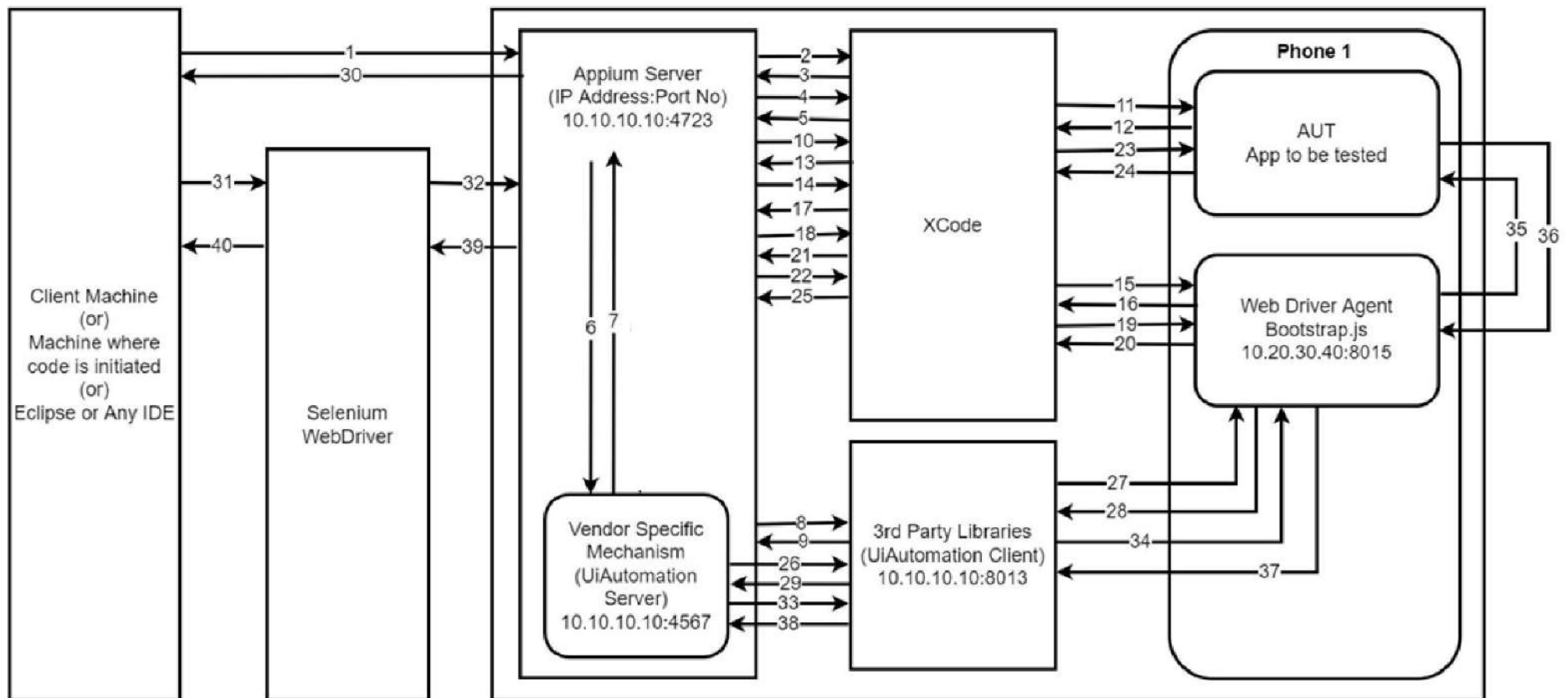
Image is just for reference.
Pls don't get confused

* for the sake of simplicity we miss on dependency relationships of packages within framework container and its community used utilities

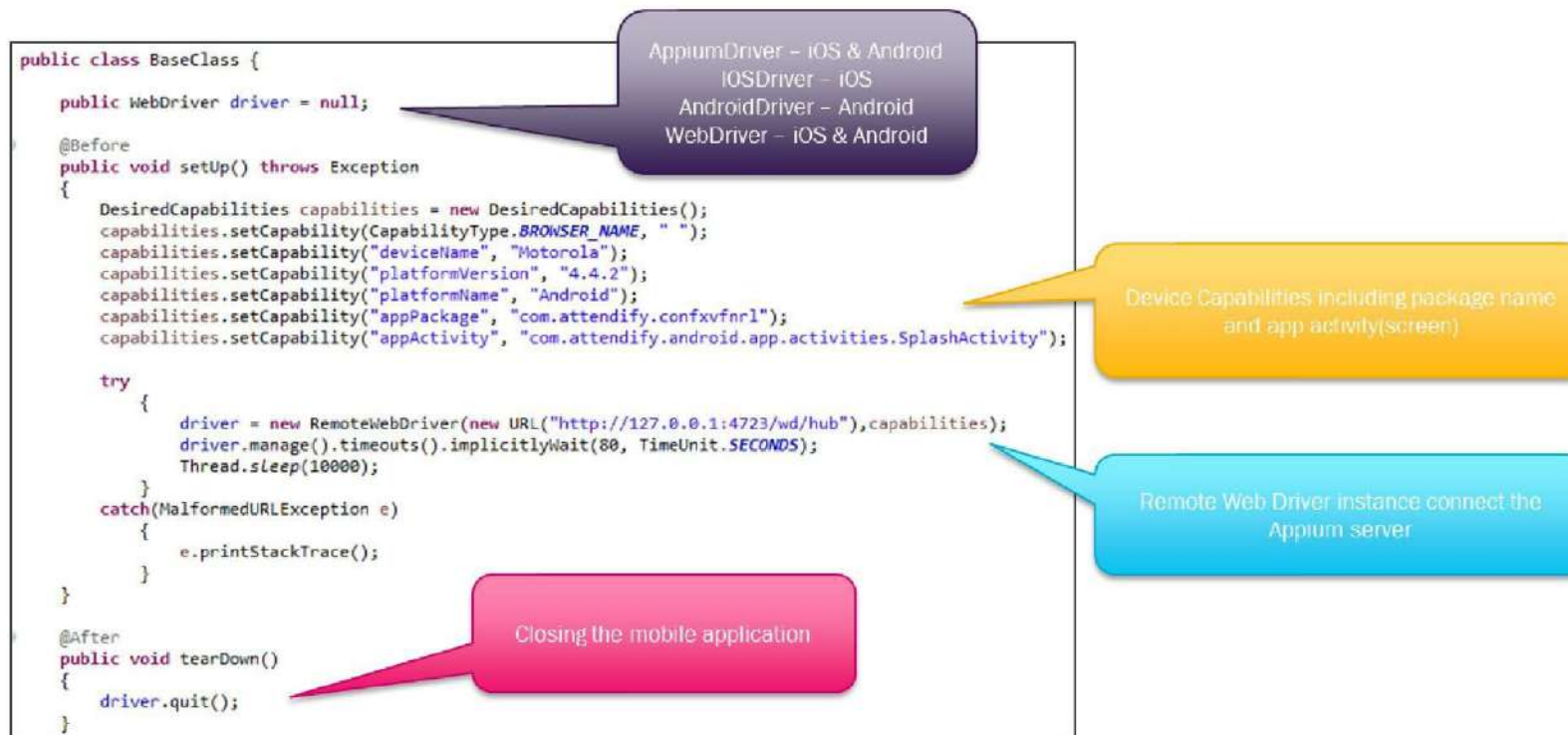
ANDROID ARCHITECHTURE - DETAILED - Native App



iOS ARCHITECHTURE - DETAILED - Native App



CODE STRUCTURE OVERVIEW



UI ELEMENT INSPECTORS



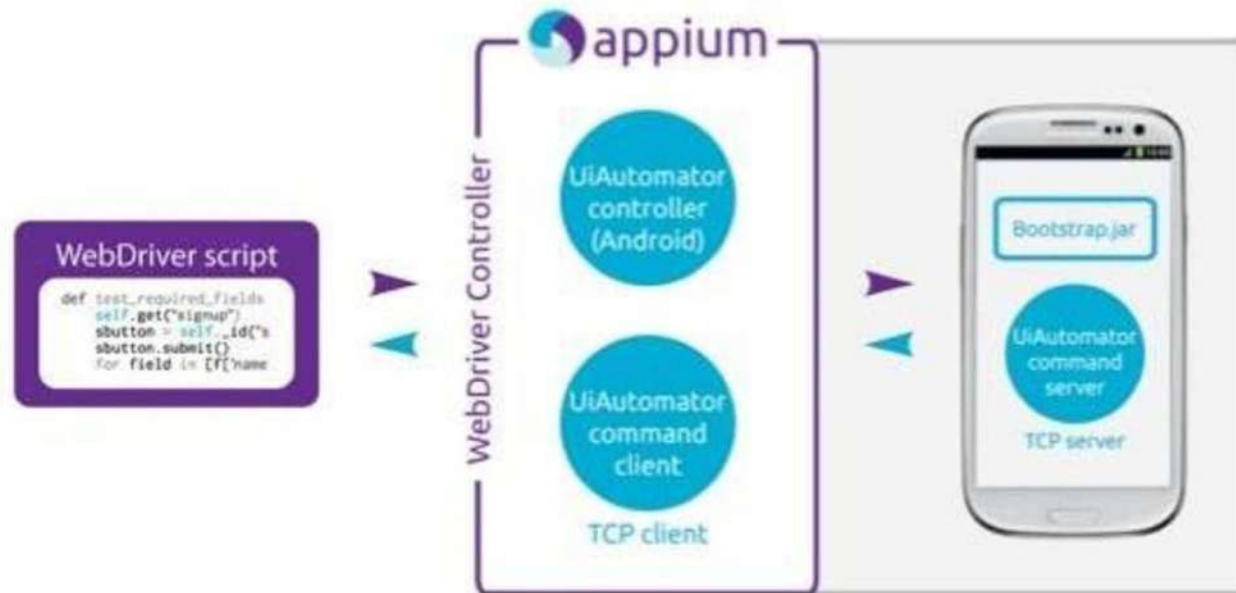
UI LOCATORS - NATIVE APP/HYBRID APP (NATIVE VIEW)

id 	name 	linkText 
className 	tagName 	Partial LinkText 
xpath 	css 	Accessibility Id 

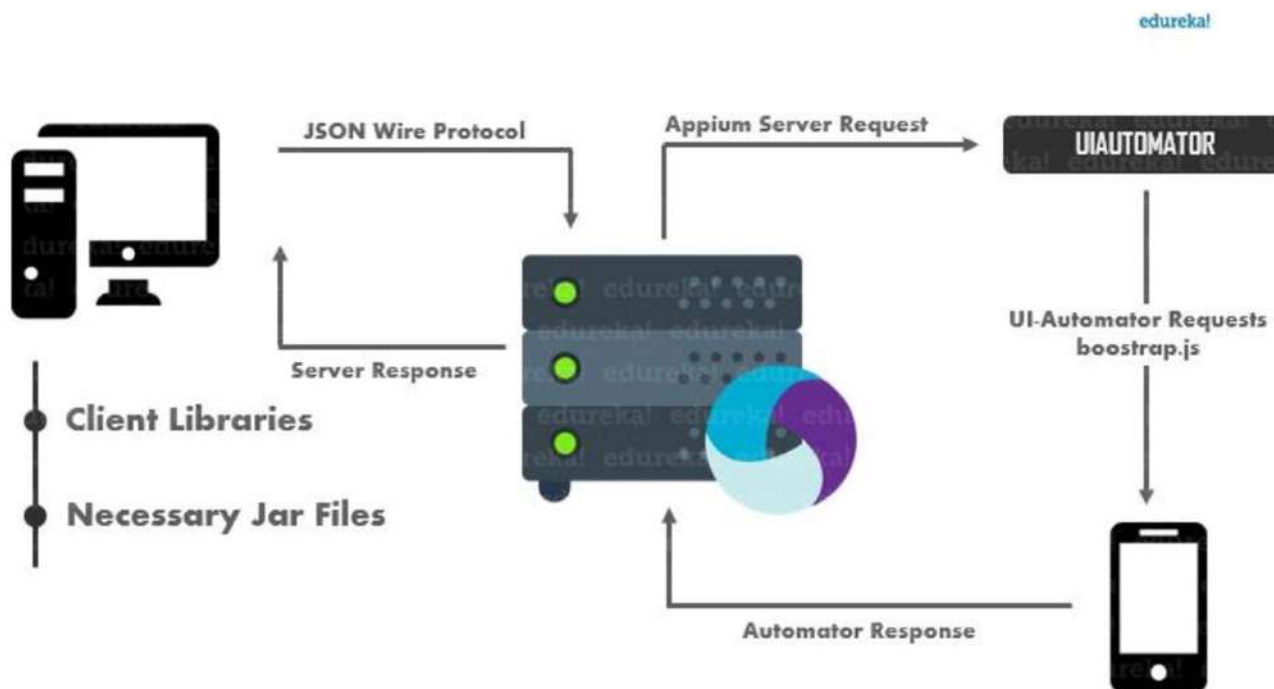
UI LOCATORS - MOBILE WEB/HYBRID APP (WEB VIEW)

id ✓	name ✓	linkText ✓
className ✓	tagName ✓	Partial LinkText ✓
xpath ✓	css ✓	Accessibility Id ✗

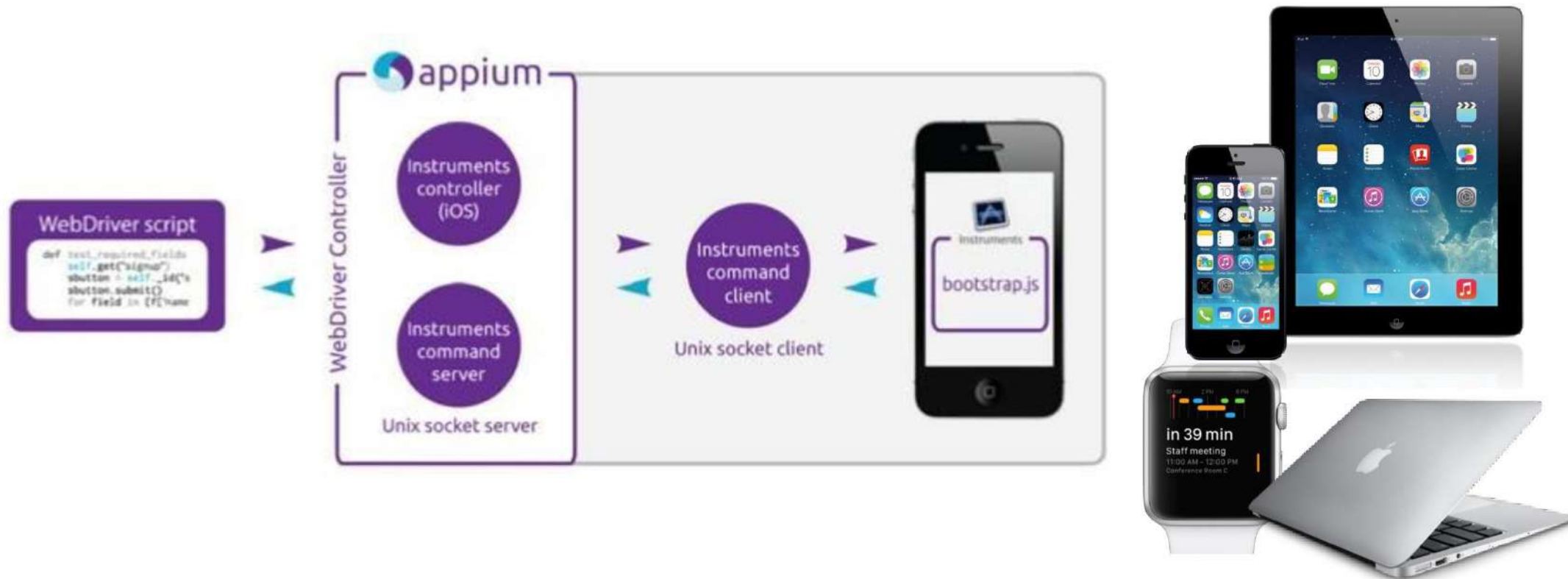
APPIUM ARCHITECTURE [ANDROID]



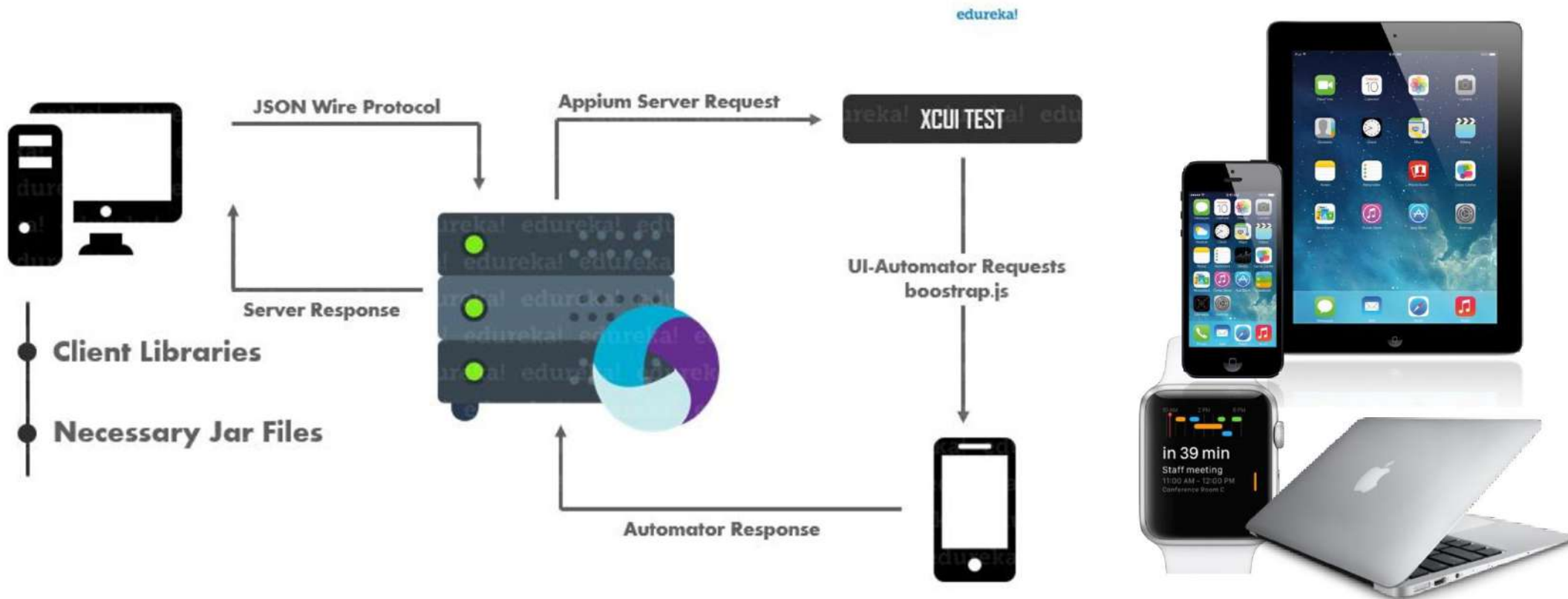
APPIUM ARCHITECTURE [ANDROID]



APPIUM ARCHITECTURE [IOS]



APPIUM ARCHITECTURE [IOS]





ANDROID DEBUG BRIDGE

It is a command line tool that lets you communicate with an emulator instance or connected Android-powered device. It is a client-server program that includes three components:

- A client, which runs on your development machine. You can invoke a client from a shell by issuing an adb command. Other Android tools such as DDMS also create adb clients.
- A server, which runs as a background process on your development machine. The server manages communication between the client and the adb daemon running on an emulator or device.
- A daemon, which runs as a background process on each emulator or device instance.

ENABLING ADB DEBUGGING

On Android 4.2 and higher, the Developer options screen is hidden by default. To make it visible, go to **Settings > About phone** and tap **Build number** seven times.

When you connect a device running Android 4.2.2 or higher to your computer, the system shows a dialog asking whether to accept an RSA key that allows debugging through this computer.

This security mechanism protects user devices because it ensures that USB debugging and other adb commands cannot be executed unless you're able to unlock the device and acknowledge the dialog.

ENABLING ADB DEBUGGING



ADB COMMANDS

Category	Command	Description
Target Device	-d	Direct an adb command to the only attached USB device.
	-e	Direct an adb command to the only running emulator instance.
General	devices	Prints a list of all attached emulator/device instances.
	version	Prints the adb version number.

ADB COMMANDS

Category	Command	Description
Debug	logcat	Prints log data to the screen.
Data	install	Pushes an Android application (specified as a full path to an .apk file) to an emulator/device.
	pull	Copies a specified file from an emulator/device instance to your development computer.
	push	Copies a specified file from your development computer to an emulator/device instance.

ADB COMMANDS

Category	Command	Description
server	start-server	Checks whether the adb server process is running and starts it, if not.
	kill-server	Terminates the adb server process.
Shell	shell	Starts a remote shell in the target emulator/device instance.

DEVICE CONNECTION STATE

State	Description
offline	The instance is not connected to adb or is not responding.
device	<p>The instance is now connected to the adb server.</p> <p>Note that this state does not imply that the Android system is fully booted and operational, since the instance connects to adb while the system is still booting.</p> <p>However, after boot-up, this is the normal operational state of an emulator/device instance.</p>
no device / blank	There is no emulator/device connected.
unauthorized	Debugging permission is not granted

CONNECT TO WIRELESS

Step	Step Description
1	Connect your Android device and adb host computer to a common Wi-Fi network accessible to both.
2	Connect the device to the host computer with a USB cable.
3	Set the target device to listen for a TCP/IP connection on port 5555. <code>\$ adb tcpip 5555</code>
4	Disconnect the USB cable from the target device.
5	Find the IP address of the Android device. For example, on a Nexus device, you can find the IP address at <code>Settings > About tablet (or About phone) > Status > IP address</code> . Or, on an Android Wear device, you can find the IP address at <code>Settings > Wi-Fi Settings > Advanced > IP address</code> .
6	Connect to the device, identifying it by IP address. <code>\$ adb connect <device-ip-address></code>

APPIUM SCRIPT EXECUTION IN DEVICE



THANK YOU!

