MS Business Centre
Innovating Solutions

Mobile Testing with Appium:
Android and iOS Apps

Presented by:

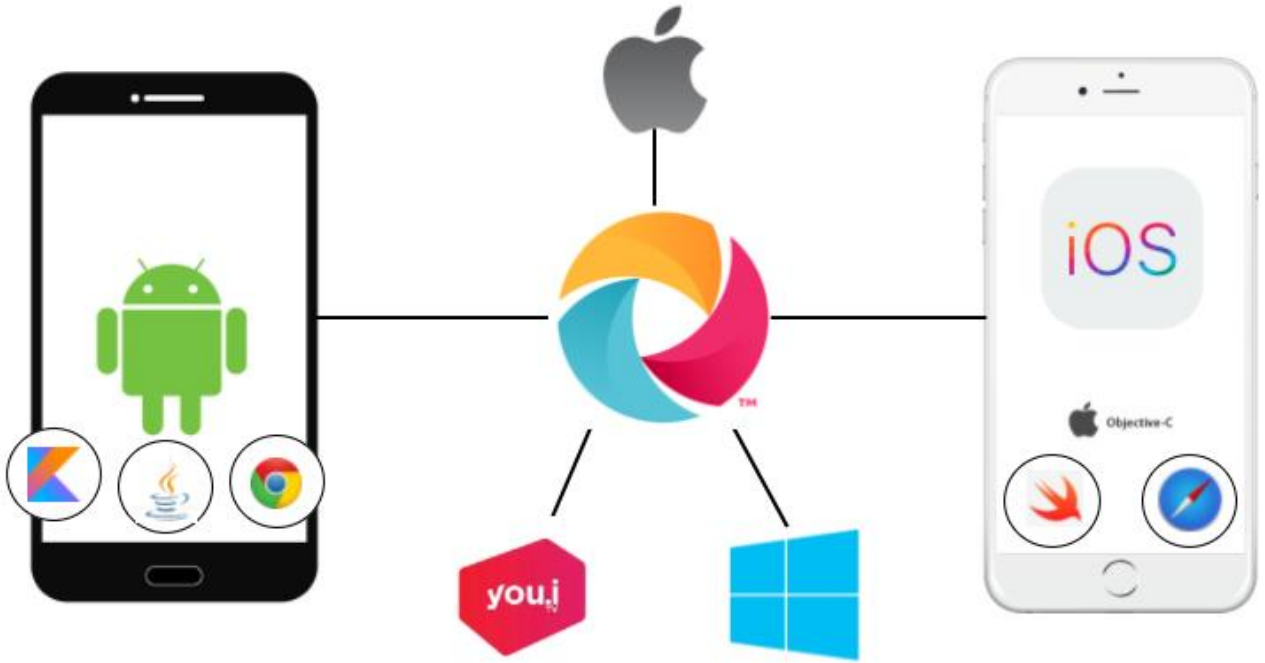
Parth Naik

Agenda

- Introduction to Appium
- Appium Philosophy
- Appium Architecture
- How appium works?
- Installation
- Appium Desired Capabilities
- Selector Strategies
- Demo

Introduction to Appium

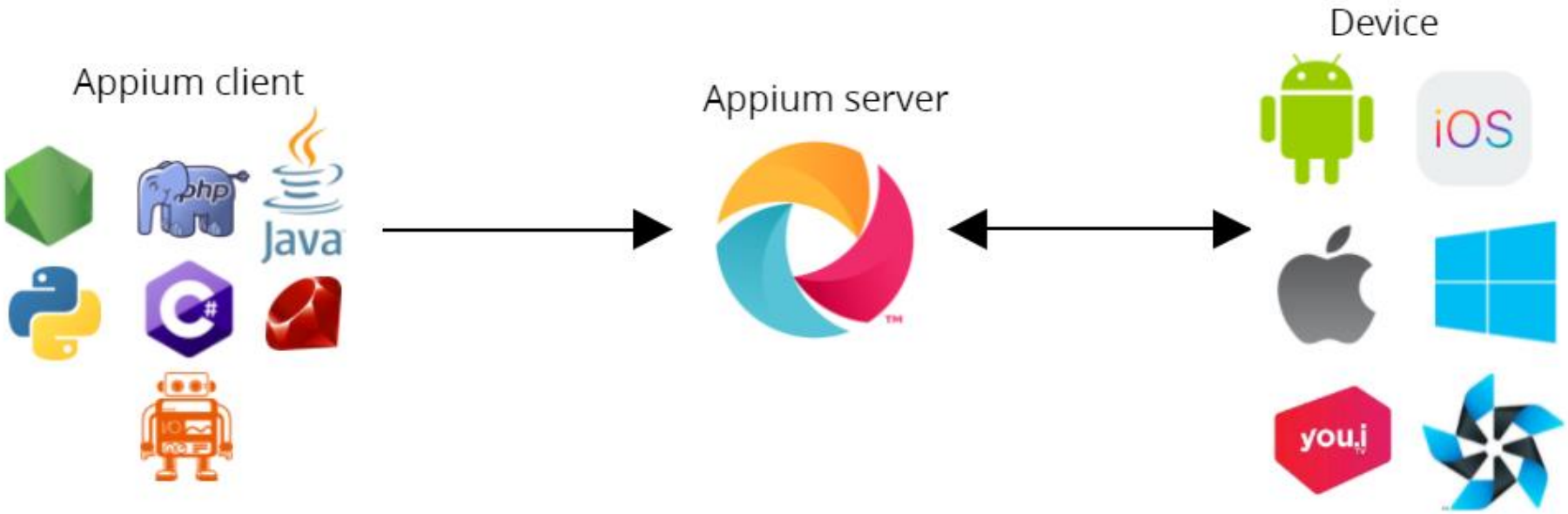
- Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS mobile, Android mobile, and Windows desktop platforms.
- Appium is "cross-platform": it allows you to write tests against multiple platforms (iOS, Android, Windows), using the same API.
- Mobile web apps are web apps accessed using a mobile browser. Appium supports Safari on iOS and Chrome or the built-in 'Browser' app on Android.
- Hybrid apps have a wrapper around a "webview“ - a native control that enables interaction with web content.



Appium Philosophy

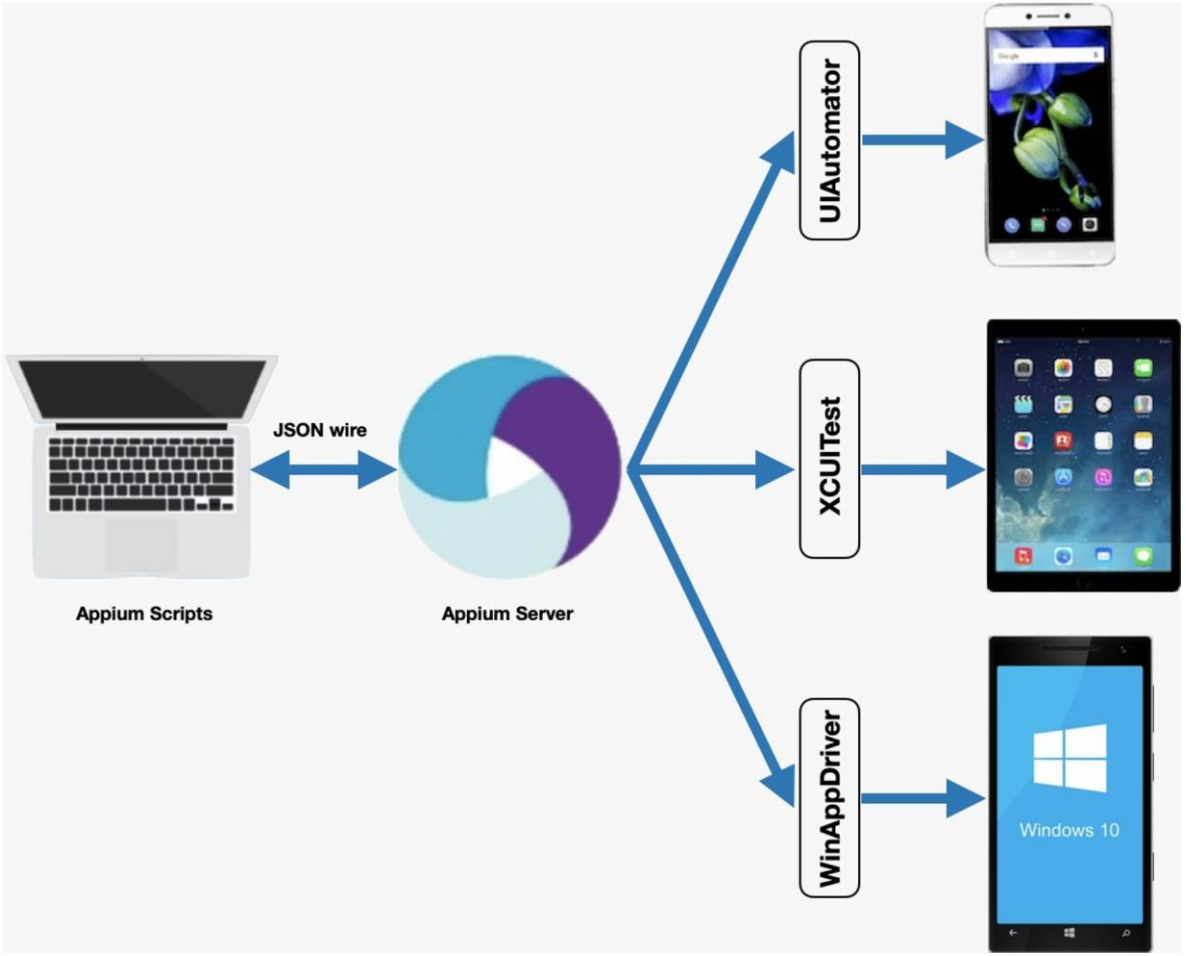
Appium was designed to meet mobile automation needs according to a philosophy outlined by the following four tenets:

1. You shouldn't have to recompile your app or modify it in any way in order to automate it.
2. You shouldn't be locked into a specific language or framework to write and run your tests.
3. A mobile automation framework shouldn't reinvent the wheel when it comes to automation APIs.
4. A mobile automation framework should be open source, in spirit and practice as well as in name!



Appium Architecture

- Appium is a client-server architecture. The Appium server communicates with the client through the HTTP JSONWire Protocol using JSON objects.
- Once it receives the request, it creates a session and returns the session ID, which will be used for communication so that all automation actions will be performed in the context of the created session.
- Appium uses the UIAutomator test framework to execute commands on real Android devices and emulators.
- Appium uses the XCUITest test framework to execute commands on real Apple mobile devices and emulators.
- Appium uses WinAppDriver to execute commands for Windows Desktop apps.



How appium works?

- You write your tests using one of Appium client libraries.
- Your tests calls the Webdriver API.
- The Webdriver sends the request in form of Json via http request to the Appium server.
- The Appium server, under the hood invokes vendor specific mechanisms to execute the test commands.
- The client (devices or emulators) responds back to Appium server.
- Appium server logs the results in console.

Installation

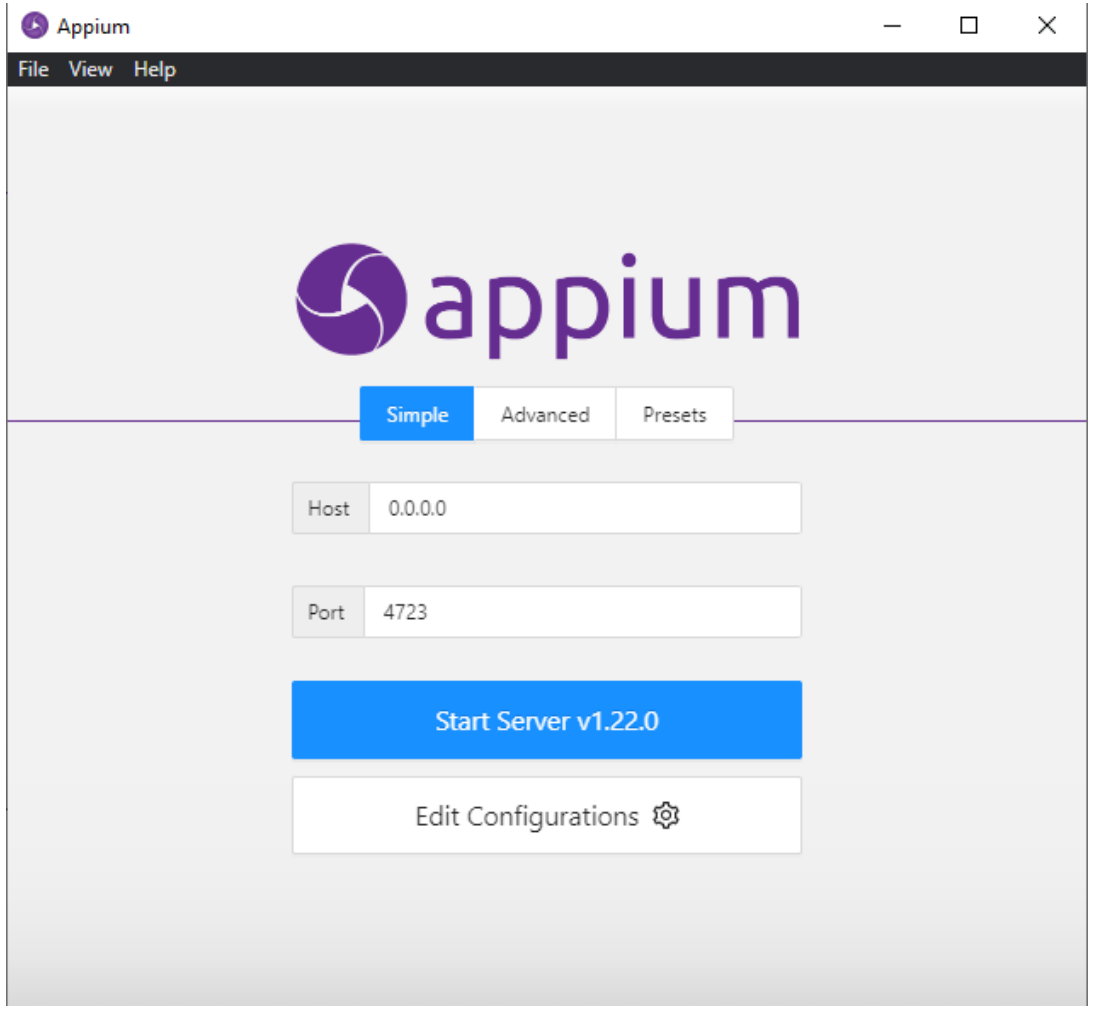
- **Installation via NPM**: Appium is a server written in Node.js. It can be built and installed from source or installed directly from NPM:

  $ npm install -g appium
  $ appium

- **Installation via Desktop App Download**: Simply download the latest version of Appium Desktop from the releases page.

- **iOS Requirements**:
  1. Mac OS X 10.10 or higher, 10.11.1 recommended
  2. XCode >= 7.1.1 recommended (Xcode 8 ? brew install carthage)
  3. Apple Developer Tools (iPhone simulator SDK, command line tools)

- **Android Requirements:**
  1. Android SDK API >= 17



Appium Desired Capabilities

- Desired Capabilities are keys and values encoded in a JSON object, sent by Appium clients to the server when a new automation session is requested.

- They tell the Appium drivers all kinds of important things about how you want your test to work. 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.

- Desired Capabilities can be scripted in the WebDriver test or set within the Appium Server GUI (via an Inspector Session)

- Some important capabilities are demonstrated in the following example:

```
{
    "platformName": "iOS",
    "platformVersion": "11.0",
    "deviceName": "iPhone 7",
    "automationName": "XCUITest",
    "app": "/path/to/my.app"
}
```

Selector Strategies

Strategy	Description
Accessibility ID	Read a unique identifier for a UI element. For XCUITest it is the element's `accessibility-id` attribute. For Android it is the element's `content-desc` attribute.
Class name	For IOS it is the full name of the XCUI element and begins with XCUIElementType. For Android it is the full name of the UIAutomator2 class (e.g.: android.widget.TextView)
ID	Native element identifier. `resource-id` for android; `name` for iOS.
Name	Name of element
XPath	Search the app XML source using xpath (not recommended, has performance issues)
Image	Locate an element by matching it with a base 64 encoded image file
Android UiAutomator (UiAutomator2 only)	Use the UI Automator API, in particular the UiSelector class to locate elements. In Appium you send the Java code, as a string, to the server, which executes it in the application's environment, returning the element or elements.
Android View Tag (Espresso only)	Locate an element by its view tag
Android Data Matcher (Espresso only)	Locate an element using Espresso DataMatcher
IOS UIAutomation	When automating an iOS application, Apple's Instruments framework can be used to find elements

Demo

Important Links:

- https://github.com/appium/appium
- https://appium.io/downloads.html
- https://github.com/appium/appium-desktop/releases
- https://github.com/appium/appium-inspector/releases
- https://appium.io/docs/en/about-appium/intro/
- https://appiumpro.com/editions/4-using-appium-for-testing-mobile-web-apps
- https://caps.cloudgrey.io/

Any Questions?

Thank You