Automating Native Apps and Web Apps Using JavaScript with Appium

Introduction to Native Apps and Web Apps Automation Using Appium

Native Apps Automation Using Appium with JavaScript

Web Apps Automation Using Appium with JavaScript

Q&A and Recap

Learning Objectives

By the end of this lesson, you will be able to:

Understand what Appium is and its role in test automation.
Set up Appium for automating native and web applications using JavaScript.
Create and execute basic automation scripts for native and web applications.
Differentiate between automating native apps and web apps.
Debug and optimize Appium scripts.

Introduction

Appium is a popular open-source tool used for automating mobile and web applications. It supports a wide range of platforms, including Android, iOS, and Windows. With JavaScript, Appium enables developers to write versatile and reusable automation scripts. This capability makes Appium essential for quality assurance teams looking to streamline testing processes for both native and web apps.

Why Use Appium with JavaScript?

Cross-Platform Support: Automates apps across different platforms.
Code Reusability: JavaScript scripts can be reused across Android and iOS.
Community Support: A large developer community for Appium and JavaScript ensures rich resources for troubleshooting and learning.

Key Concepts and Definitions

What is Appium?

Appium is an automation framework for mobile and web applications that interacts with devices using the WebDriver protocol.

Native Apps vs. Web Apps

> Native Apps: Built specifically for mobile platforms (e.g., Android, iOS) using SDKs.
> Web Apps: Mobile-optimized websites accessed via a browser.

Key Components in Appium Automation

1. Appium Server: A Node.js-based server that communicates with devices.
2. Desired Capabilities: A set of key-value pairs that specify the test setup.
3. Appium Client Libraries: Language bindings (e.g., Appium-JavaScript) used to write test scripts.

Appium's Architecture

Appium operates in a client-server architecture:

> Client: Your automation scripts in JavaScript.
> Server: Appium server running locally or remotely.

Step-by-Step Explanation

Step 1: Prerequisites

Ensure the following are installed:

1. Node.js: For running JavaScript and the Appium server.
2. Appium: Installed via npm: `npm install -g appium`
3. Appium Inspector: A GUI tool for inspecting app elements.
4. Device Setup: Either an emulator (Android/iOS) or a real device.
5. WebDriverIO: A JavaScript-based test automation framework.

Step 2: Setting Up the Environment

1. Install Appium WebDriverIO:

```
bash
Copy code
npm install webdriverio @wdio/appium-service
```

2.
3. Initialize WebDriverIO:

```
bash
Copy code
npx wdio config
```

4.
   Choose Appium as the service.
   Specify test framework (e.g., Mocha or Jasmine).
5. Launch Appium Server:

```
bash
Copy code
appium
```

6.

Step 3: Writing the Test Script

Create a JavaScript file, e.g., `test.js`, and include the following:

```
javascript
Copy code
const { remote } = require('webdriverio');

// Define Desired Capabilities
const options = {
    path: '/wd/hub',
    port: 4723,
    capabilities: {
        platformName: 'Android',
        platformVersion: '11.0',
        deviceName: 'emulator-5554',
```

```
        app: '/path/to/your/app.apk', // Replace with the app's path
        automationName: 'UiAutomator2',
    }
};

(async () => {
    const driver = await remote(options);

    // Interact with elements
    const button = await driver.$('~Login');
    await button.click();

    const input = await driver.$('~Username');
    await input.setValue('testuser');

    // Take a screenshot
    await driver.saveScreenshot('./screenshot.png');

    await driver.deleteSession();
})();
```

Step 4: Running the Test Script

Run the script using Node.js:

```
bash
Copy code
node test.js
```

Step 5: Debugging with Appium Inspector

Launch the Appium Inspector to inspect app elements and generate locators.

Examples

Example 1: Automating a Web App

```
javascript
Copy code
```

```
const { remote } = require('webdriverio');

const options = {
    capabilities: {
        browserName: 'chrome',
        platformName: 'Android',
        deviceName: 'emulator-5554',
    }
};

(async () => {
    const driver = await remote(options);

    await driver.url('https://example.com');
    const searchBox = await driver.$('input[name="q"]');
    await searchBox.setValue('Appium with JavaScript');
    await searchBox.keys('Enter');

    const result = await driver.$('h3');
    console.log(await result.getText());

    await driver.deleteSession();
})();
```

Practice Exercises

Exercise 1: Automate a Login Screen

Locate the username, password fields, and the login button.
Enter dummy credentials and click the login button.
Verify that the user is redirected to the dashboard.

Exercise 2: Test Navigation in a Web App

Open a website in a mobile browser.
Perform actions like clicking links, filling out forms, and navigating between pages.

Exercise 3: Capture Screenshots

Automate a mobile app or website to take a screenshot of each step.

Summary

Appium is a robust tool for automating both native and web apps.
JavaScript with WebDriverIO simplifies scripting and test management.
Native apps require application files (.apk/.ipa) and specific configurations, whereas web apps rely on browser automation.
Debugging and inspecting elements are key to efficient test automation.