#### 安装

要用 JavaScript（Node.js）编写 Appium 测试，我们需要选择一个与 Appium 兼容的客户端库。维护得最好的库和 Appium 团队推荐使用的库是 WebdriverIO，所以让我们使用它。

安装：

npm i --save-dev webdriverio

引入：

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

#### 链接appium 配置

// 应用启动配置

const capabilities = {

platformName: 'Android', //设备平台

'appium:automationName': 'UiAutomator2', //自动化测试引擎

'appium:deviceName': 'Android', //设备名称

'appium:appPackage': 'com.android.chrome', // 应用包名

'appium:appActivity': 'com.google.android.apps.chrome.Main', // 应用的入口 Activity

"appium:noReset": true, // 是否保留会话数据 默认false

"appium:udid": 'your\_device\_udid', // 设备的唯一识别符号

};

// Appium配置

const wdOpts = {

hostname: process.env.APPIUM\_HOST || 'localhost', // appium服务器地址

port: parseInt(process.env.APPIUM\_PORT, 10) || 4723, // appium服务端口号

capabilities,

};

// 设备实例

const driver = await remote(wdOpts);

#### 常用api

Appium 命令：

**1、获取当前上下文**

driver.getContext();

返回：表示当前上下文的字符串，无上下文返回null

1. **切换上下文**

driver.switchContext(name);

| **参数** | **类型** | **说明** |
| --- | --- | --- |
| name | string | 表示可用上下文的字符串 |

1. **获取可用的上下文**

driver.getContexts();

返回：表示可用上下文的字符串数组，例如“WEBVIEW”或“NATIVE”

1. **暂停**

暂停执行一段特定时间

driver.pause(milliseconds);

| **参数** | **类型** | **说明** |
| --- | --- | --- |
| milliseconds | number | 时间（毫秒） |

##### WebDriver 协议:

**1、删除会话**

driver.deleteSession();

命令会关闭与当前会话关联的任何顶级浏览上下文，终止连接，最后关闭当前会话。

1. **重启会话**

browser.reloadSession();

1. **后退**

browser.back();

  命令使浏览器在当前顶级浏览上下文的联合会话历史记录中向后移动一步。这相当于按下浏览器 chrome 中的后退按钮

1. **向前**

browser.forward();

命令使浏览器在当前顶级浏览上下文的联合会话历史记录中向前遍历一步。

1. **获取当前窗口句柄**

browser.getWindowHandle();

命令返回当前顶级浏览上下文的窗口句柄。它可以用作 Switch To Window 的参数。

1. **刷新**

browser.refresh();

 命令使浏览器重新加载当前顶级浏览上下文中的页面。

1. **关闭窗口**

browser.closeWindow();

关闭窗口命令关闭当前顶级浏览上下文。完成后，如果没有其他顶级浏览上下文打开，则 WebDriver 会话本身将关闭。

1. **切换到窗口**

browser.switchToWindow(handle);

| **参数** | **类型** | **说明** |
| --- | --- | --- |
| handle | string | 表示窗口句柄的字符串，应该是 getWindowHandles 调用中返回的字符串之一 |

切换到窗口命令用于选择当前会话的当前顶级浏览上下文，即用于处理命令的浏览上下文。

1. **获取所有窗口句柄**

browser.getWindowHandles();

获取窗口句柄命令返回每个打开的顶级浏览上下文的窗口句柄列表。

1. **切换到框架（iframe）**

browser.switchToFrame(id);

| **参数** | **类型** | **说明** |
| --- | --- | --- |
| id | 数字，对象，空 | 三种可能类型之一：  null：这代表顶级浏览上下文（即不是 iframe）；  数字：代表与框架相对应的窗口对象的索引；  对象：一个使用接收的 Element 对象findElement； |

命令用于选择当前顶级浏览上下文或当前浏览上下文的子浏览上下文作为后续命令的当前浏览上下文。

**11、返回到iframe父级**

browser.switchToParentFrame();

切换到父框架命令将当前浏览上下文设置为当前浏览上下文的父级，以便将来的命令可以执行。

**12、执行动作**

browser.performActions(actions);

| **姓名** | **类型** | **细节** |
| --- | --- | --- |
| actions | 目的[] | 对象列表，每个对象代表一个输入源及其相关操作 |

执行操作命令用于执行复杂的用户操作。

示例：手势滑动 从窗口（0，10）滑动到（0，500）

await browser.performActions([{

        type: 'pointer',

        id: 'finger1',

        parameters: { pointerType: 'touch' },

        actions: [

            { type: 'pointerMove', duration: 0, x: 0, y: 10},

            { type: 'pointerDown', button: 0 },

            { type: 'pause', duration: 100 },

            { type: 'pointerMove', duration: 1000, origin: 'viewport', x: 0, y: 500},

            { type: 'pointerUp', button: 0 }

        ]

    }]);

##### 浏览器：

1. **选择器**

$$与$

$$命令是一种简短而方便的方法来获取页面上的多个元素。返回一个对象数组。

$ 命令是一种简短而方便的方法，用于获取页面上的单个元素。

1. **Url**

browser.url(url);

| **参数** | **类型** | **说明** |
| --- | --- | --- |
| url地址 | String | 要导航到的 URL |

协议绑定以加载浏览器的 URL。

1. **获取浏览器窗口大小**

browser.getWindowSize();

返回浏览器窗口大小 { width: 1280, height: 767 }

1. **等到**

browser.waitUntil(condition, { timeout, timeoutMsg, interval });

| **参数** | **类型** | **说明** |
| --- | --- | --- |
| condition | Function | 等待直到返回真值的条件 |
| Options（选修的） | WaitUntilOptions | 命令选项 |
| options.timeout （选修的） | Number | 超时时间（毫秒）（默认值：5000） |
| options.timeoutMsg（选修的） | String | 当 waitUntil 超时时抛出的错误消息 |
| options.interval（选修的） | Number | 条件检查之间的间隔（默认值：500） |

如果您想要等待某件事，此 wait 命令就是您的万能武器。它等待一个条件，并等待直到该条件满足并返回真值。

1. **执行脚本**

browser.execute(script, arguments);

| **参数** | **类型** | **说明** |
| --- | --- | --- |
| script | String，Function | 要执行的脚本。 |
| Arguments（选修的） | 参数 | 脚本参数 |

将一段 JavaScript 代码注入页面，在当前选定框架的上下文中执行。执行的脚本被认为是同步的，并且脚本的执行结果将返回给客户端。

script 参数以函数体的形式定义要执行的脚本。该函数返回的值将返回给客户端。该函数将使用提供的 args 数组调用，并且可以按照指定的顺序通过参数对象访问这些值。

参数可以是任何 JSON 基元、数组或 JSON 对象。定义 WebElement 引用的 JSON 对象将转换为相应的 DOM 元素。同样，脚本结果中的任何 WebElement 都将作为 WebElement JSON 对象返回到客户端。

示例：

获取页面滚动的高度

 const scrollPosition = await browser.execute(() => {

        // 优先使用 window.scrollY，如果不支持则回退到 document.documentElement.scrollTop

        return (window.scrollY || document.documentElement.scrollTop) || document.body.scrollTop;

    });

##### 元素：

1. **添加值**

$(selector).addValue(value);

向给定选择器找到的输入或文本区域元素添加一个值。

1. **清除值**

$(selector).clearValue();

清除输入或文本区域元素的值。使用此命令之前，请确保您可以与元素交互。您无法清除已禁用或处于只读模式的输入元素。

1. **点击**

$(selector).click({ button, x, y, skipRelease });

| **参数** | **类型** | **说明** |
| --- | --- | --- |
| Options（选修的） | ClickOptions | 点击选项（可选） |
| options.button | string，number | 可以是 [0, "left", 1, "middle", 2, "right"] 之一（可选） 默认值 0 |
| options.x（选修的） | number | 号码（可选）  距离元素中心x方向像素值 默认值0 |
| options.y（选修的） | number | 号码（可选）  距离元素中心y方向像素值 默认值 0 |
| options.skipRelease（选修的） | boolean | 布尔值（可选）跳过发送releaseActions |

单击一个元素。

click这会为所选元素发出WebDriver命令，当未传递任何选项时，该命令通常会滚动到所选元素，然后单击该元素。传递选项对象时，它会使用操作类而不是 webdriver click，从而提供传递按钮类型、坐标等附加功能。默认情况下，使用选项时，执行单击操作后会发送释放操作命令，传递option.skipRelease=true以跳过此操作。

注意：如果您有固定位置的元素（例如固定的页眉或页脚），在视口内滚动后会覆盖所选元素，则点击将在给定的坐标处发出，但将由您的固定（覆盖）元素接收。在这些情况下，会引发以下错误：

Element is not clickable at point (x, x). Other element would receive the click: ..."

要解决此问题，请尝试找到覆盖元素并通过execute命令将其移除，以免干扰点击。您还可以尝试使用scroll适合您场景的偏移量自行滚动到元素。

1. **获取html**

$(selector).getHTML(includeSelectorTag);

| **姓名** | **类型** | **细节** |
| --- | --- | --- |
| includeSelectorTag 选修的 | Boolean | 如果为 true，则包含选择器元素标签（默认值：true） |

通过选择器获取指定 DOM 元素的源代码。

1. **获取位置**

$(selector).getLocation();

确定元素在页面上的位置。点 (0, 0) 指的是页面的左上角。

1. **获取尺寸**

$(selector).getSize();

获取 DOM 元素的宽度和高度。

1. **获取文本**

$(selector).getText();

从 DOM 元素获取文本内容。确保您要请求文本的元素是可交互的 ，否则您将获得一个空字符串作为返回值。如果元素被禁用或不可见，而您仍想接收文本内容，请使用getHTML 作为解决方法。

1. **获取值**

$(selector).getValue();

获取给定选择器找到的或<select>文本的值<input>。如果通过给定选择器找到多个元素，则返回值数组。对于复选框或单选按钮类型的输入，请使用 isSelected。

1. **是否可点击**

$(selector).isClickable();

当满足以下条件时，元素被视为可点击：

1、该元素存在

2、元素被显示

3、该元素未被禁用

4、该元素位于视口内

5、该元素可以滚动到视口中

6、元素的中心不与其他元素重叠

否则返回 false。

1. **是否被展示**

$(selector).isDisplayed();

如果所选的 DOM 元素已显示（即使元素位于视口之外），则返回 true。如果您想要验证元素是否也不在视口内，请使用 isDisplayedInViewport 命令。

1. **是否存在**

$(selector).isExisting();

如果元素存在于 DOM 中，则返回 true。

1. **滚动到可见区域**

$(selector).scrollIntoView();

滚动元素到视口中。

1. **设定值**

$(selector).setValue(value);

给输入框设定值。

1. **等待显示**

$(selector).waitForDisplayed({ timeout, reverse, timeoutMsg, interval });

| **姓名** | **类型** | **细节** |
| --- | --- | --- |
| options 选修的 | WaitForOptions | waitForEnabled 选项（可选） |
| options.timeout 选修的 | Number | 时间（毫秒）（默认值：500） |
| options.reverse 选修的 | Boolean | 如果为真则等待相反的结果（默认值：假） |
| options.timeoutMsg 选修的 | String | 如果存在，它将覆盖默认错误消息 |
| options.interval 选修的 | Number | 检查间隔（默认值waitforInterval） |

等待元素在指定的毫秒数内显示或不显示。

1. **等待存在**

$(selector).waitForExist({ timeout, reverse, timeoutMsg, interval });

| **姓名** | **类型** | **细节** |
| --- | --- | --- |
| options选修的 | WaitForOptions | waitForEnabled 选项（可选） |
| options.timeout选修的 | Number | 时间（毫秒）（默认值：500） |
| options.reverse选修的 | Boolean | 如果为真则等待相反的结果（默认值：假） |
| options.timeoutMsg选修的 | String | 如果存在，它将覆盖默认错误消息 |
| options.interval选修的 | Number | 检查间隔（默认值waitforInterval） |