SauceCon Online 2020

# 1 Codebase, 2 Mobile Platforms: How to Test iOS and Android by Just Writing Tests Once

François Martin

KARAKUN

# François Martin

## Full Stack Software Engineer



francois.martin@karakun.com

martinfrancois

/in/françoismartin

@fmartin_



Working group:
ISTQB® Mobile Application Testing Certification

# Poll

Q: Did you do any mobile app development previously?
A: Yes, native (or hybrid that compiles to native) / Yes, hybrid (WebView-based, like Ionic) / No

Q: Have you done any automated mobile testing before?
A: Yes / No

Q: If you answered yes to the previous question, which platforms did you target?
A: Android / iOS / Both (cross-platform)

Q: Have you written any automated end-to-end-tests (desktop or mobile) before?
A: Yes, using WebdriverIO v6 / Yes, using WebdriverIO v5 / Yes, using Selenium / Yes, using a different framework / No

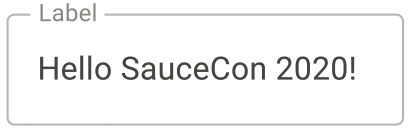
# Native vs Hybrid

- Native
  - 2 separate codebases (iOS, Android)
  - Develop everything twice

- Hybrid (JS: NativeScript, React Native; Java: Gluon Mobile; C#: Xamarin)
  - 1 Codebase
  - Compiles to both platforms (iOS, Android)

# How do hybrid frameworks work?

**Text Field Example:**



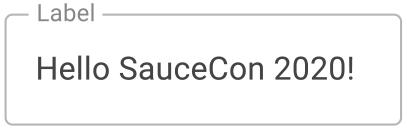

Android:
```
<EditText android:text="Hello SauceCon 2020!"/>
```

iOS (SwiftUI):
```
TextField("Hello SauceCon 2020!")
```

Hybrid (NativeScript):
```
<TextField text="Hello SauceCon 2020!"/>
```

# How do hybrid frameworks work?

**Text Field Example:**




Hybrid (NativeScript): `<TextField text="Hello SauceCon 2020!"/>`

```
function TextField(text: string) {
  if (isAndroid) {
    return <EditText android:text=text/>
  } else if(isIos) {
    return TextField(text)
  }
}
```

Android: `<EditText android:text="Hello SauceCon 2020!"/>`

iOS (SwiftUI): `TextField("Hello SauceCon 2020!")`

# Automated mobile E2E testing

- Quite different to testing websites
- Initial setup more involved
  - Testing on iOS requires a Mac (without Sauce Labs RDC, but recommended)
  - Testing iOS real devices (without Sauce Labs RDC) especially difficult to setup
- Not headless (so far)
- Most difficult part: Selectors
  - Web: id, class...
  - Mobile: accessibilityId
- Stable automation requires developing the app designed for testability
- A lot of platform-specific code necessary

# Selector Strategies


XPath

# Selector Strategies (WebdriverIO)

accessibilityId:

```
$("~Login")
```

UiSelector (Android):

```
$('android=new UiSelector().className("android.widget.Button").text("Login")')
```

predicate (iOS):

```
$("-ios predicate string:type == 'XCUIElementTypeButton' && label == 'Login'")
```

# Applying the hybrid approach

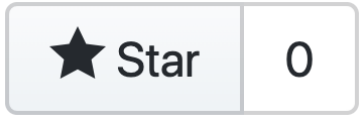
```
// utils.ts
function getButtonWithText(text: string): WebdriverIO.Element {
  if (browser.isAndroid) {
    return $(`android=new UiSelector().className("android.widget.Button").text("${text}")`);
  } else if (browser.isIOS) {
    return $(`-ios predicate string:type == 'XCUIElementTypeButton' && label == '${text}'`);
  }
}

// login.spec.ts (Test)
getButtonWithText("Login");
```

# Introducing: wdio-mobile-utils

- Free and open source
- Fluent API, TypeScript support
- Supports WebdriverIO v5 and v6
- Makes you forget about XPath (not guaranteed)
- Automated releases (continuous delivery)
- Pull Requests and issues (especially fixes) welcome!

```
npm install --save-dev wdio-mobile-utils
```

https://github.com/martinfrancois/wdio-mobile-utils

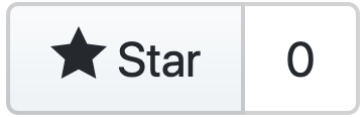


# But wait, there‘s more!

**Introducing wdio-mobile-utils-demo**

- Demo application built with NativeScript
- E2E Tests using WebdriverIO v5 (currently) and **wdio-mobile-utils**
- Scripts for E2E tests out-of-the-box
  - iOS, Android (real devices, simulator / emulator), cross-platform
  - Sauce Labs Virtual Cloud for iOS
  - Sauce Labs RDC for Android and iOS
- Builds for iOS real devices, simulators and Android on Travis CI
  - Tested on Sauce Labs triggered by Travis CI

https://github.com/martinfrancois/wdio-mobile-utils-demo



# Cross-platform selectors

UiSelector (Android):

```
$('android=new UiSelector().className("android.widget.Button").text("Login")')
```

predicate (iOS):

```
$("-ios predicate string:type == 'XCUIElementTypeButton' && label == 'Login'")
```

wdio-mobile-utils:

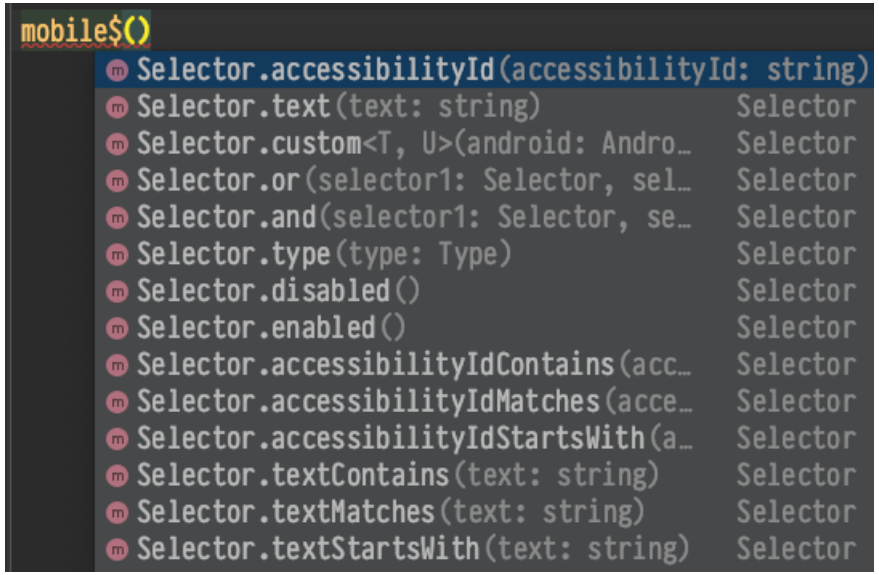
```
mobile$(Selector.and(Selector.type(Type.BUTTON), Selector.text('Login')));
```

Alternative:

```
mobile$(
  Selector.and(
    Selector.type(Type.BUTTON),
    Selector.text('Login')
  )
)
```

# Cross-platform selectors

IDE Code Completion (Example: Jetbrains IntelliJ IDEA):



# Utilities (excerpt)

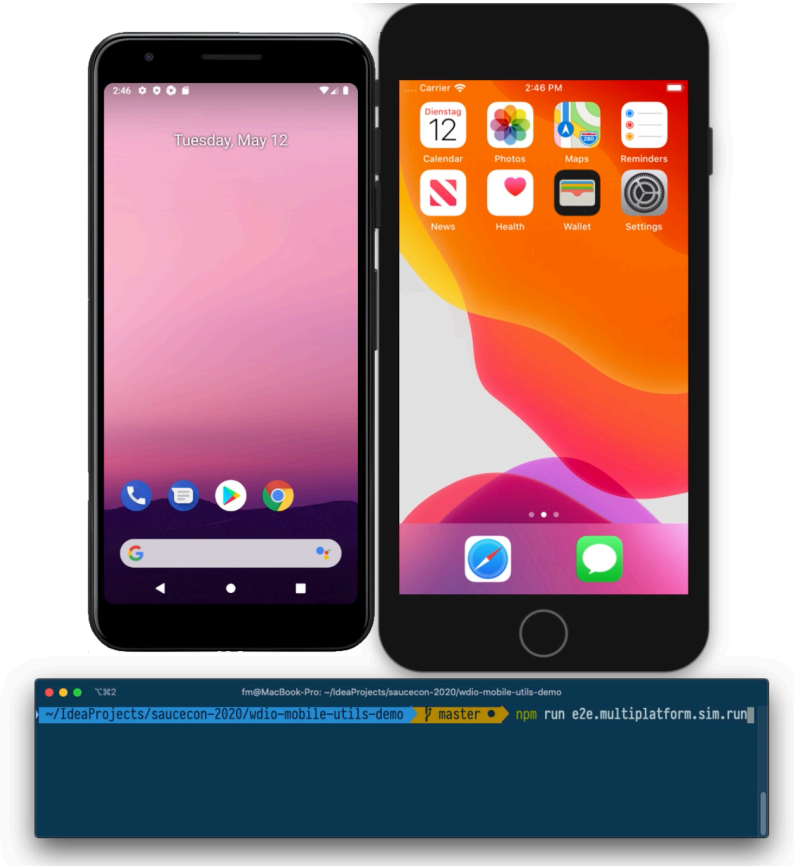
```
// Alert
isAlertDisplayed()
waitForAlertDisplayed()
acceptAlert()
dismissAlert()

// App Utilities
isAppState(APP_RUNNING_STATE.FOREGROUND, wait, appId, bundleId)
isBrowserAppState(APP_RUNNING_STATE.BACKGROUND, wait)
openApp(wait, appId, bundleId)

// Gestures
swipeUp()
swipeDown()
swipeRight()
swipeLeft()
swipeRightOnElement(element)
swipeLeftOnElement(element)
```

# Demo Alert Cross-platform

Command in wdio-mobile-demo-utils:

```
npm run e2e.multiplatform.sim.run
```

**login.spec.ts**



# Demo Alert Cross-platform

Command in wdio-mobile-demo-utils:

```
npm run e2e.multiplatform.sim.run
```

**login.spec.ts**

# Deeplink

wdio-demo-utils:

```
openDeeplink('/my/deeplink', appId, bundleId)
```

Android Implementation:

```
browser.closeApp();
browser.execute('mobile: deepLink', {
    url: path,
    package: appId,
});
```

iOS Implementation:

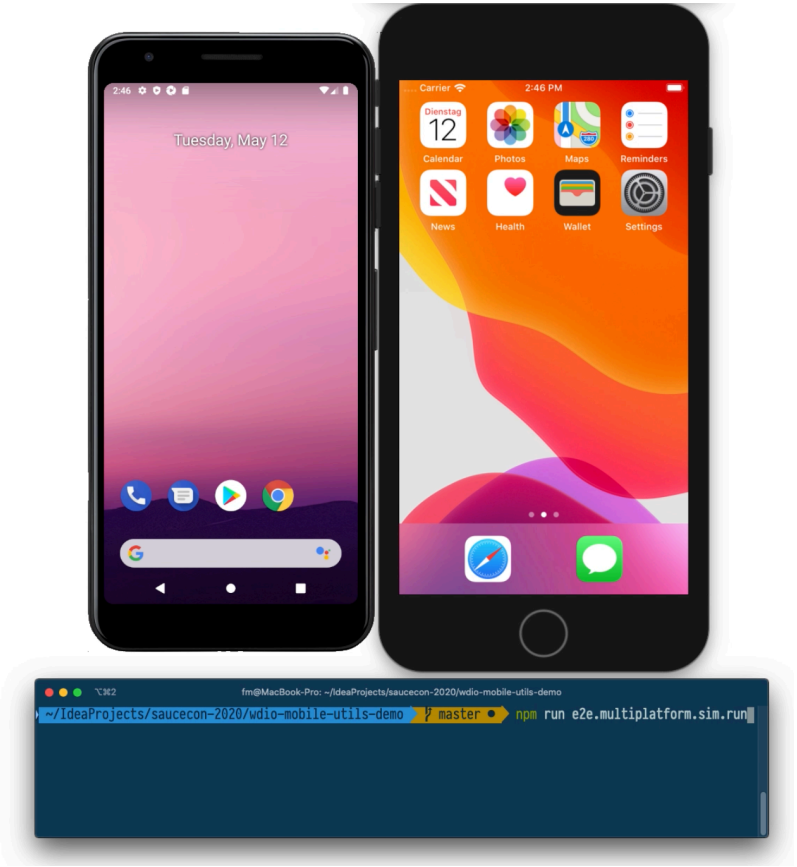
```
// imagine 64 lines of code here
```

# Demo Deeplink Cross-platform

Command in wdio-mobile-demo-utils:

```
npm run e2e.multiplatform.sim.run
```

**home.spec.ts**



# Demo Deeplink Cross-platform

Command in wdio-mobile-demo-utils:

`npm run e2e.multiplatform.sim.run`

**home.spec.ts**

# Thanks for your attention!

## Any questions?

For any further questions after the talk, join and ask in the Slack Channel:
**#1-codebase-2-mobile-platforms**

François Martin

francois.martin@karakun.com

martinfrancois

/in/françoismartin

@fmartin_

# Useful Links and References

- Appium Capabilities
- Appium Capabilities for iOS
- WebdriverIO Documentation
- WebdriverIO Guide on Pageobjects
- WebdriverIO TypeScript Guide
- Appium and Deeplinks
- Opening Deeplinks using Appium
- Locator Strategies in Appium
- Picking the right locator strategy