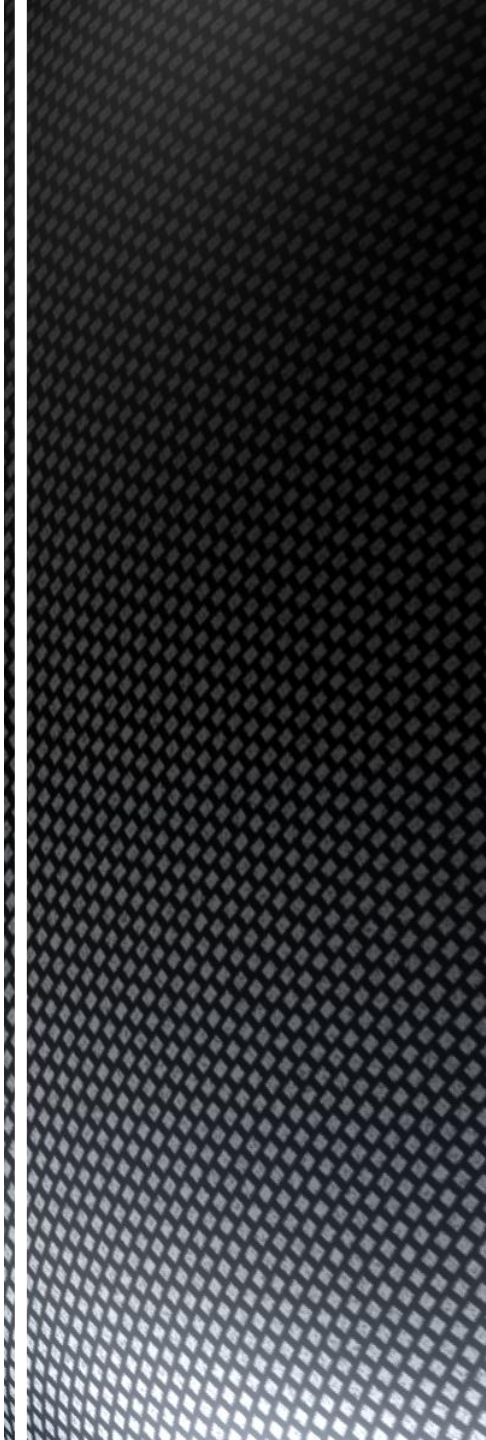
Accessibility test automation

- accessibility-developer-tools (ADT)
- Selenium
- Scala
- Cucumber & Gherkin



What is Web Accessibility ?

- Web accessibility means that people with disabilities can use the Web.

to w3.org for full description

Why Web Accessibility is important ?

- The Web is an increasingly important resource in many aspects of life;
- An accessible Web can help people with disabilities more actively participate in society.

to w3.org for full description

Is Web Accessibility expensive ?

- Many accessibility features are easily implemented if they are planned from the beginning of Web site development or redesign;
- Fixing inaccessible Web sites can require significant effort.

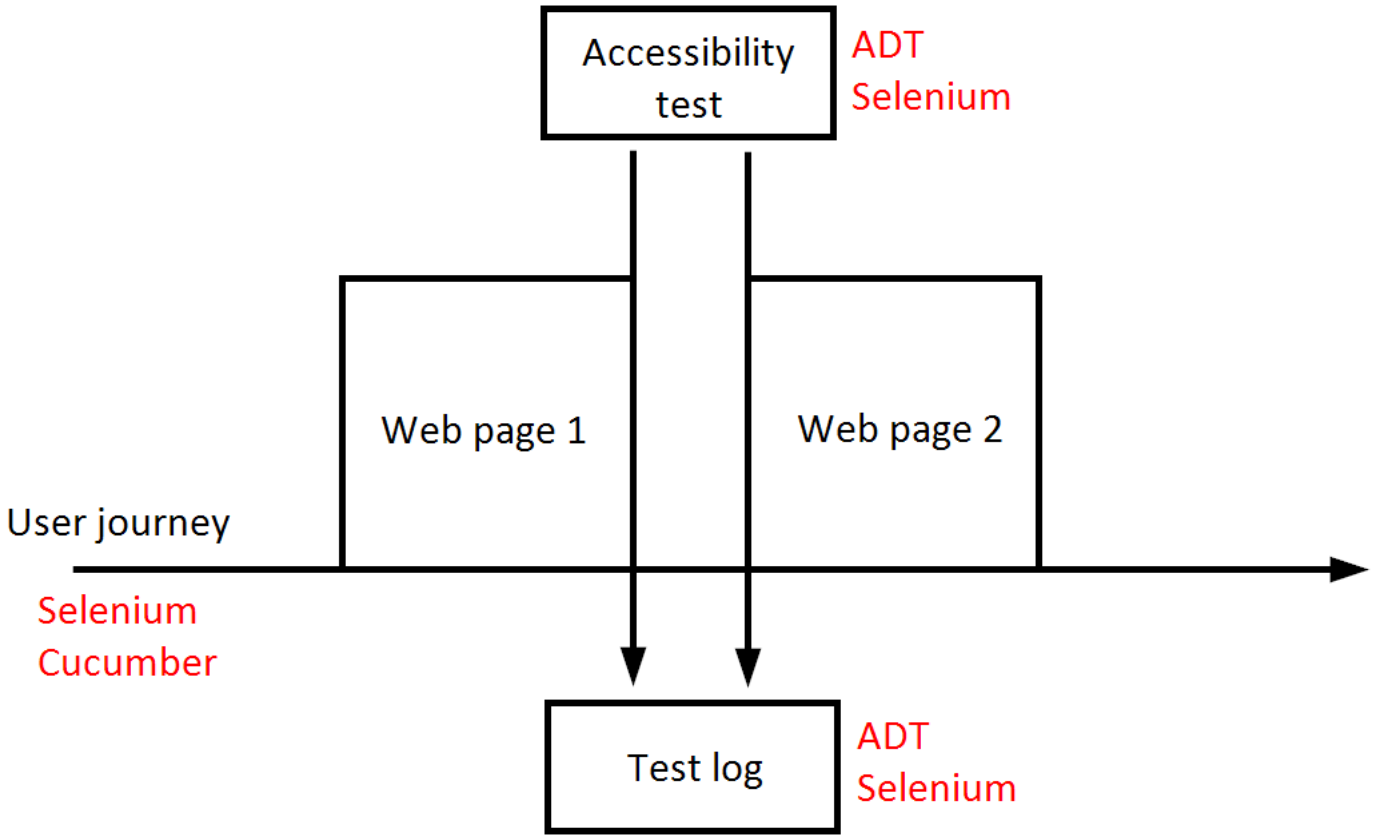
to w3.org for full description

Testing the Accessibility of a Web Site

- There are evaluation tools that help with evaluation (one of which is ADT);
- However, no tool alone can determine if a site meets accessibility guidelines;
- Knowledgeable human evaluation is required to determine if a site is accessible.

to w3.org for full description

Accessibility test automation example



Tools used in this example

- ADT (to GitHub);
- Selenium (to seleniumhq.org);
- Scala (to scala-lang.org);
- Cucumber (to cucumber.io).

What we need from accessibility-developer-tools

As written in the readme, ADT is: “…a library of accessibility-related testing and utility code.”

The library and utility code from the project is compiled into a single executable JavaScript file:

axs_testing.js

(to axs_testing.js raw content)

What we need from Selenium

We use a library of audit rules compiled into a single JavaScript. From Selenium we need a mechanism for executing JavaScript.

Therefore we can use:

JavascriptExecutor Interface

(to selenium.googlecode.com)

What we need from Scala

To use Selenium we must use a programming language that is supported by Selenium;

Scala is not listed, but Scala is compatible with Java. Scala classes are Java classes, and vice versa (to more information).

What we need from Cucumber

It is optional to use Cucumber as you can achieve similar results by writing Scala code for Selenium;

Cucumber is used in this example to organise the test suite and make the user journey readable by a larger audience.

Checklist for Implementation

- User journey
- Code for journey
- Code for test execution
- Code for logs

User journey

A user journey/story/feature would look like:

```
@suite
Feature: Accessible user journey
  As a tester
  I want to test a web page for accessibility
  So that I can be sure that it does not contain critical accessibility issues

  Scenario: Test accessibility for Google.com
    Given user navigates to 'http://google.com' web page
    Then accessibility test is executed on the open page
```

Code for journey

Our journey consists of a step that opens the given URL in a browser:

```
Given( """^user navigates to '(.*)' web page$""") {
   (userUrl: String) =>
     withCurrentDriver { implicit webDriver =>
       webDriver.get(userUrl)
   }
 }
```

Code for test execution

Tests are executed with the help of JavascriptExecutor:

```
Then( """^accessibility test is executed on the open page$""") {
  () => withCurrentDriver { implicit webDriver =>
    val cache = collection.mutable.Map[String, String]()
    val jse = webDriver.asInstanceOf[JavascriptExecutor]
    def getUrlSource(arg: String): String = cache get arg match {
      case Some(result) => result
      case None =>
        val result: String = scala.io.Source.fromURL(arg).mkString
        cache(arg) = result
        result
    }
    jse.executeScript(getUrlSource("https://raw.githubusercontent.com/GoogleChrome/" +
       "accessibility-developer-tools/stable/dist/js/axs_testing.js"))
    val report = jse.executeScript("var results = axs.Audit.run();return axs.Audit.createReport(results);")
    println(report)
  }
}
```

Code for logs

The report is generated with the help of a JavaScript code which is held within axs_testing.js (snippet visible in the previous slide as well):

```
val report = jse.executeScript("var results = axs.Audit.run();return
axs.Audit.createReport(results);")
    println(report)
```

Demo

Demo

Thank you

Extra bits:

Accessibility-driver repo on GitHub

(driver that piggybacks on existing journeys)

Accessibility-developer-tools wiki on GitHub

(detailed info about each error type from logs)

Presenter: Kristaps Melderis