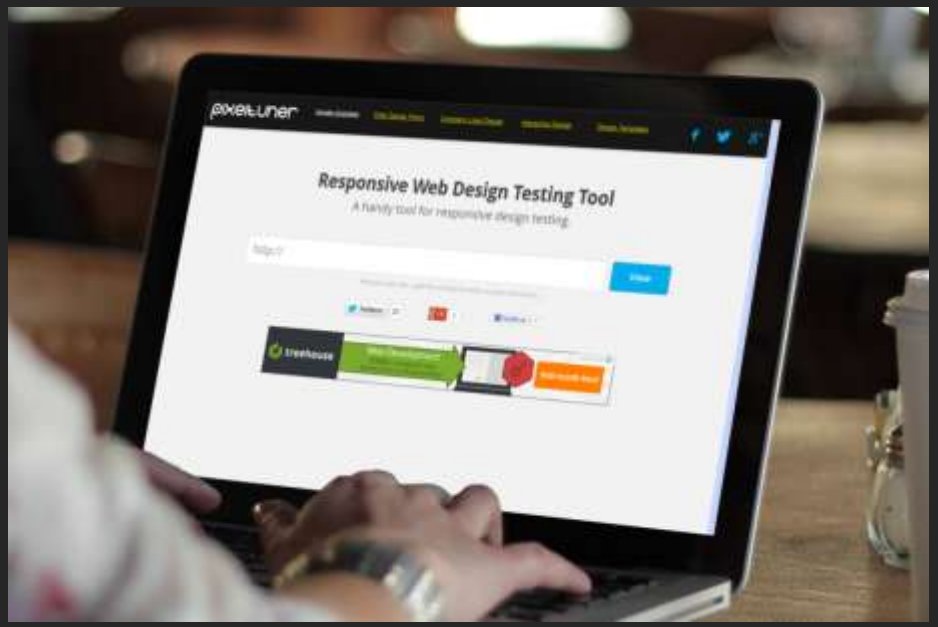
Automated Website Testing Using Selenium

Stuart Dickerson
Fall 2016

Our goals:

1. what Selenium is (for those who don't)
2. how to create and automate a browser
3. where Selenium fits in the continuous integration loop
4. how to scale Selenium

Web UI's = Diversity



We are trying to build applications that run right in a dozen browsers, look good on thousands of different devices, load fast enough for impatient commuters in the subway, and are still accessible enough that blind people can use them.

JavaScript Fatigue Fatigue by Quincy Larson

Introducing Selenium

- What exactly is Selenium?
  - An API for accessing a browser from code
- How does it work?
  - code communicates to a browser via an API
  - write code that mimics user activity

A little background

- 2004: created at ThoughtWorks by Jim Huggins and open sourced
- 2005: evolves into Selenium Remote Control
- 2007: Jim goes to work at Google; Simon Stewart develops WebDriver
- 2008: Selenium Grid is created to replace Remote Control
- 2009: Both projects merge at next Google Test Automation Conference

*"you can cure mercury poisoning by taking **selenium** supplements"*

That being said…

- Project evolved to support many
  - browsers

    Chrome, Firefox, IE, Edge, Safari & headless browsers (mobile versions as well)
  - programming languages

    Ruby, Java, C#, Perl, PHP, Python, JavaScript & Groovy
  - platforms

    Linux, Windows, iOS & Android
- A truly universal open source community 

Selenium Offerings

- Selenium IDE
  - test recorder & playback tool
    (not very versatile)
- WebDriver
  - the heart of Selenium
- Grid
  - scales tests to many browsers and platforms

Goal 1: Accomplished!

1. **what Selenium is (you all now know)**
2. how to create and automate a browser
3. where Selenium fits in the continuous integration loop
4. how to scale Selenium

WebDriver

1. create an instance
2. set the browser that you want
   - add browser capabilities (optional)
3. manipulate the DOM
4. quit the instance

DOM manipulation

- locate the desired element
  - (in order or performance)
    id, name, CSS or Xpath
- perform actions on the element
  - click, mouse, send keystrokes or set attributes
- check the result of the action


DEMO

Browsers...

- Chrome
  - fastest and most lightweight
- Firefox
  - new stand-alone driver (geckodriver) with Selenium 3 (currently in Beta)
- Internet Explorer
  - enterprise controlled security settings render it useless
- Edge
  - new stand-alone driver (WebDriver *how original!*) with Selenium 3 (currently in Beta) but only for C# and JavaScript

Browser gotchas…

- New browser versions can break your Selenium WebDriver
- Be cautious of automatic browser upgrades

WebDriver tips...

- Use only explicit waits

```
driver.wait(until.(some condition))
```

- Pre-populate cookies on the driver

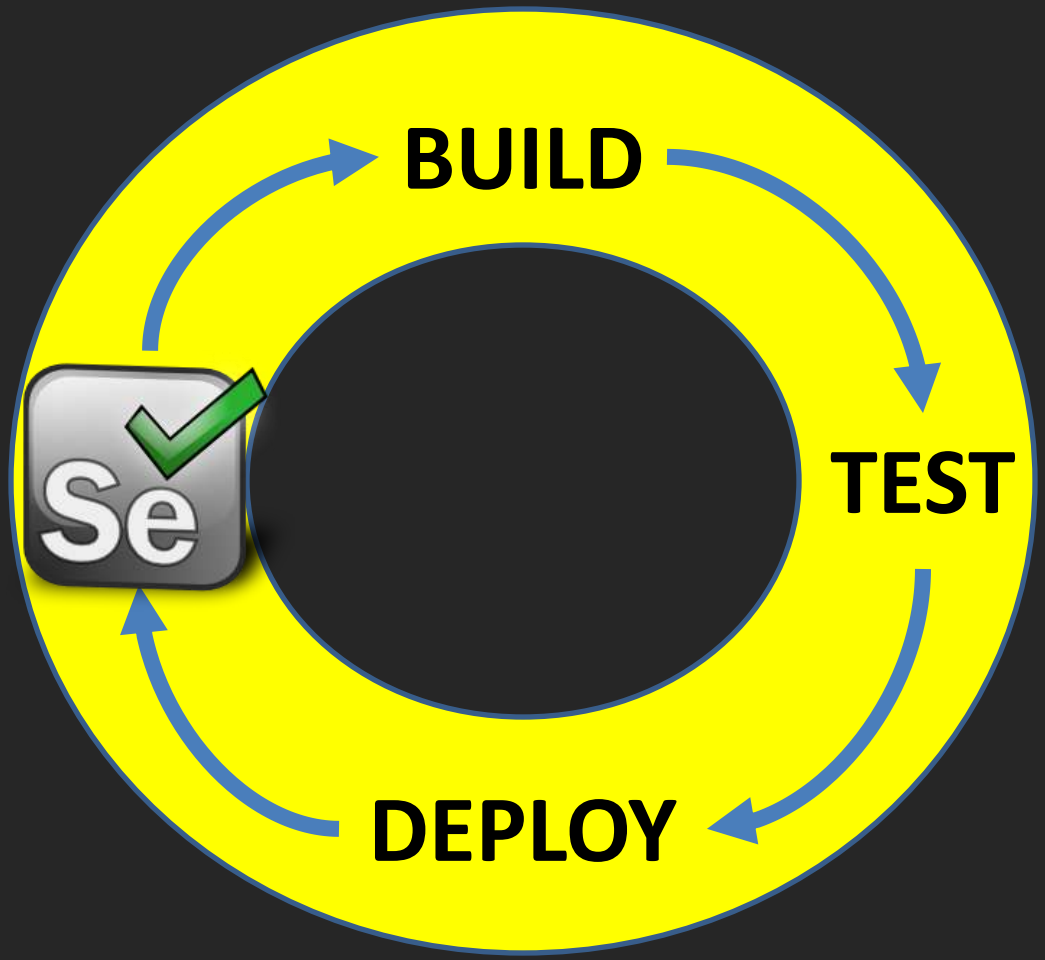
```
driver.manage().addCookie("test", "cookie-1");
```

- Disable images if not needed
- Maximize browser to prevent timeouts

Goal 2: Accomplished!

1. what Selenium is (you all now know)
2. **how to create and automate a browser**
3. where Selenium fits in the continuous integration loop
4. how to scale Selenium

Continuous Integration & Deployment



Testing frameworks

- Selenium integrates with almost any testing framework
  - mocha, TestNG, msTest, JUnit
- Your language of choice
- Create your own!


DEMO

Test execution

- Locally
- Continuous Integration (CI) Server

  Jenkins, TFS, TeamCity
- Headless browsers
- Dispatch to Selenium Grid

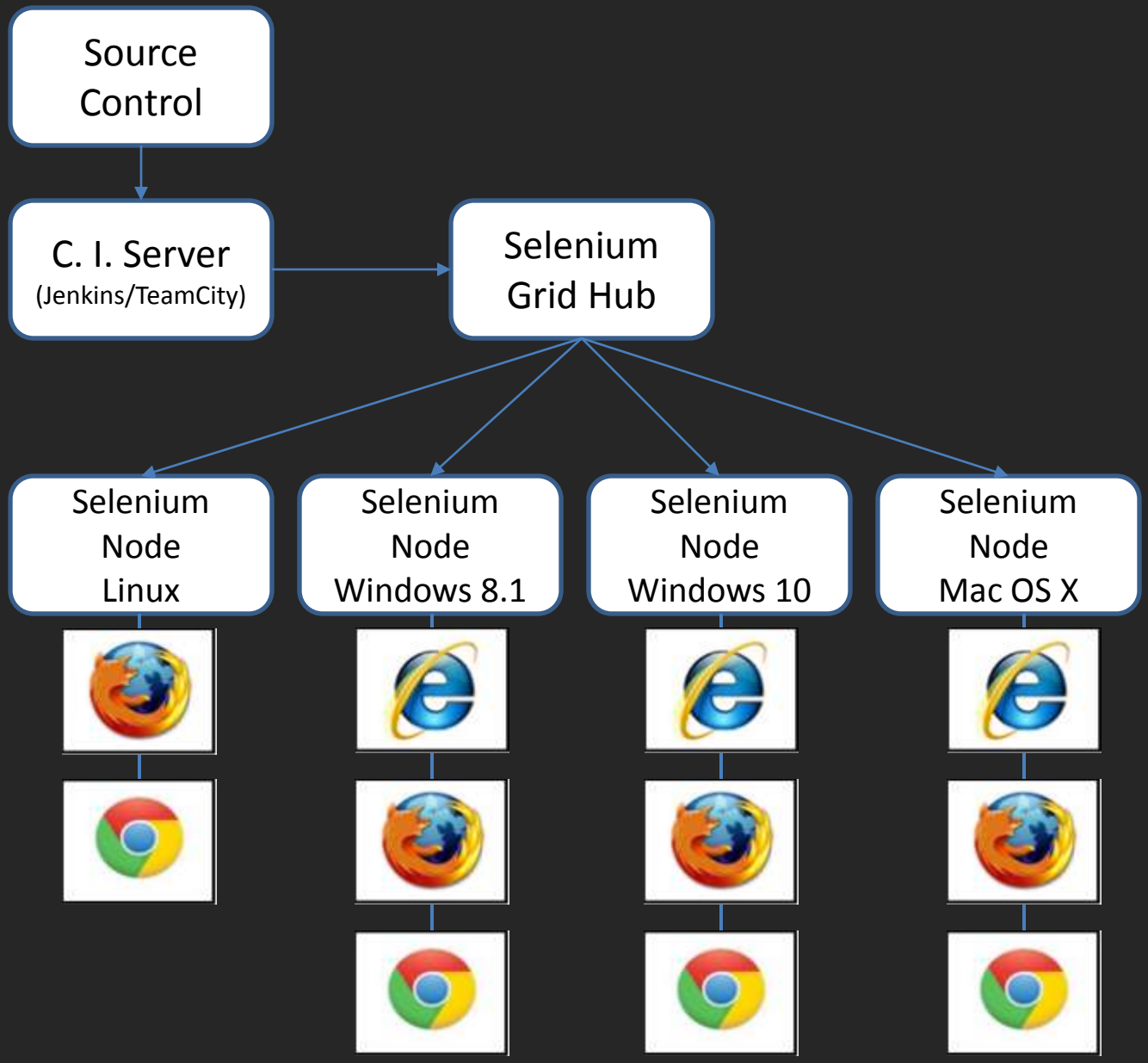
Goal 3: Accomplished!

1. what Selenium is (you all now know)
2. how to create and automate a browser
3. **where Selenium fits in the continuous integration loop**
4. how to scale Selenium

Selenium Grid

- Different platforms with different browsers
- Run multiple tests at the same time
- Reduces the time it takes for your test suite to complete

Speed up test execution!


Source Control
C. I. Server
(Jenkins/TeamCity)
Selenium Grid Hub
Selenium Node Linux
Selenium Node Windows 8.1
Selenium Node Windows 10
Selenium Node Mac OS X

How to set it up

- Stand-alone .jar file
- Create a Hub
  - Update the hub's JSON config
  - Run the .jar
- Create a Node
  - Update the node's JSON config
  - Run the .jar


DEMO
Time

Grid gotchas…

- No definitive documentation of hub to node ratios/setup
  - Hub can be a bottleneck
  - More hubs with fewer nodes and browsers
  - Keep hub and node on same server
- Test exceptions don’t release resources
  - Browser instances left open
  - Nodes require restart

Good test writing tips…

- Write atomic and autonomous tests
- Small tests focused on one thing
- Group like tests together in small batches
  - TestCatory in vstestConsole.exe
- Run test groups in parallel
  - Test Runner or CI Server

Cloud Grid providers

- Sauce Labs
- BrowserStack
- CrossBrowserTesting
- bitbar (formerly TestDroid)
- many others…

Docker implementation

- Open source containers
  - Hub
  - Chrome Node with 1 browser instance
  - Firefox Node with 1 browser instance
- Linux CentOS 7.x or Ubuntu 14.x

https://hub.docker.com/r/selenium/

Selenium Grid Extras

- Created by Groupon
  https://github.com/groupon/Selenium-Grid-Extras
- Extra features:
  - Ability to restart a node after a set number of test executions
  - Automatically upgrade WebDriver
  - Ability to record tests
  - Take OS screenshots
  - more…

Goal 4: Accomplished!

1. what Selenium is (you all now know)
2. how to create and automate a browser
3. where Selenium fits in the continuous integration loop
4. **how to scale Selenium**

Other uses for Selenium

- Website monitoring
  - New Relic
- Boring web-based administration tasks
- Performance testing

Final thoughts…

- Easy to learn
- Works the same across all languages
- Great starting point for novice developers

https://github.com/stuartdga/SeleniumDemoJS