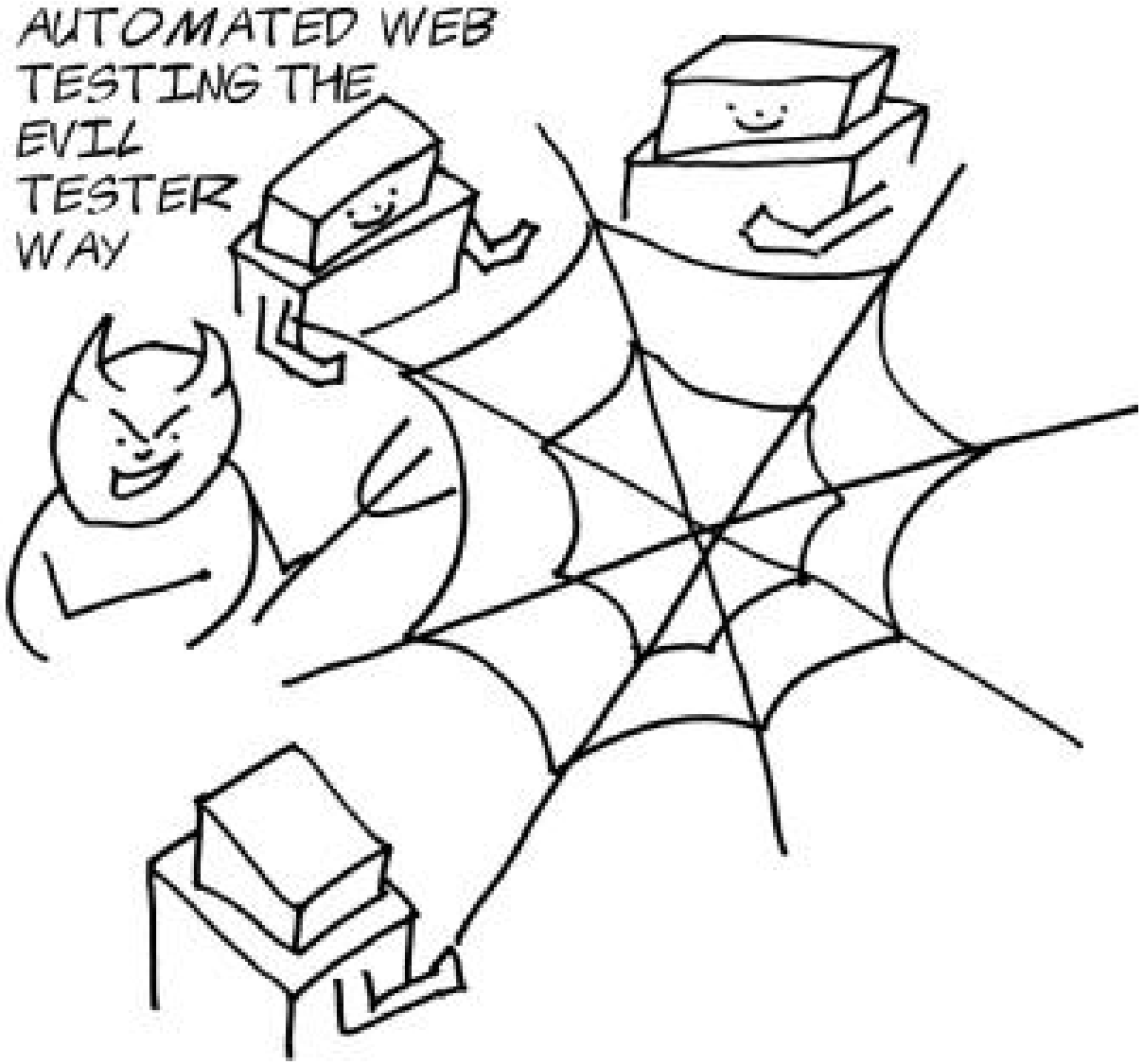
# Selenium 2 : WebDriver Basics

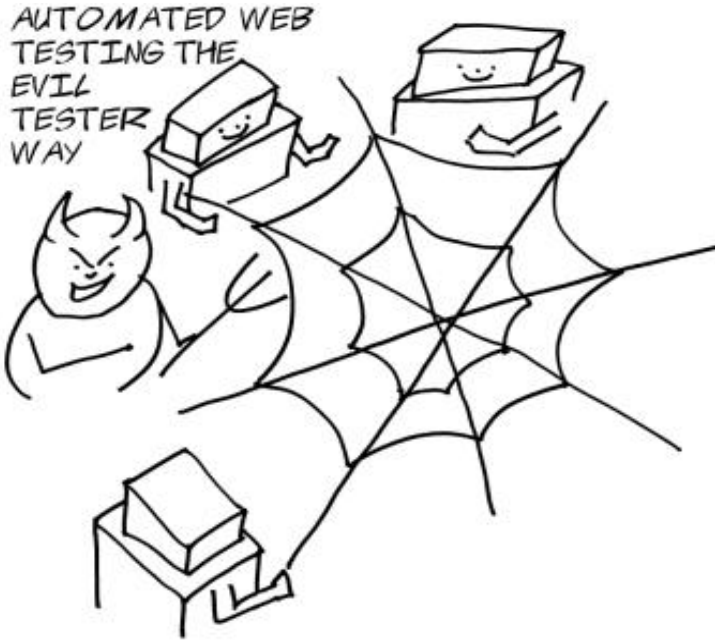
Alan Richardson



# Copyright


- All contents copyright Compendium Developments 2016
- Author: Alan Richardson
- This is the slide deck released to support the online course “Selenium 2 WebDriver With Java”
- Permission is granted to print this out for the sole use of the purchaser and student of the course. This slide deck should not be distributed.

# Overview

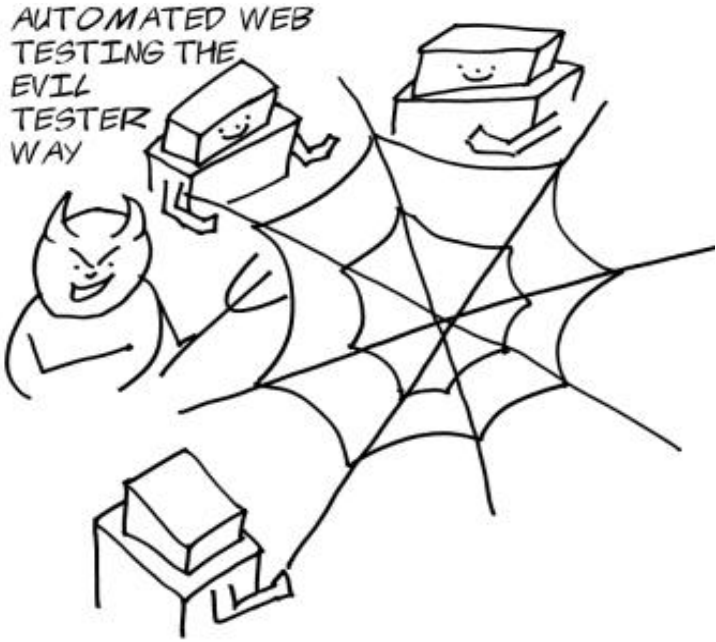


# Aims

- *Brief* overview of Selenium 2.0 WebDriver
  - Differences between Selenium and WebDriver
- Practical Automation Examples: in Java using WebDriver
  - Basics
  - HTML
  - AJAX and JavaScript
- Self-Learning Strategies

All the Basics
to get you started

# Overview of Selenium 2.0 WebDriver



# Ancient History

- Selenium 1.0
  - Javascript through a proxy
  - Flat API “selenium.”
  - Server based
  - Easy to use
- WebDriver
  - Object Oriented API, Interface which each driver implements
  - Driver calls browser directly
  - Harder

# Selenium 2.x

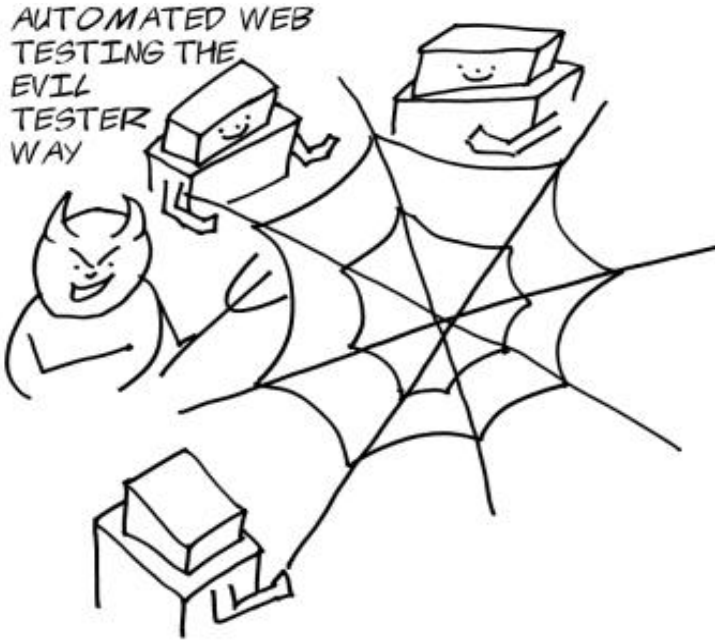
- Selenium 1.0 and WebDriver Merged
- Selenium 1.0 API fixed and stable
- WebDriver API standardisation underway
- WebDriver API now simpler

# Selenium 1.0 vs Selenium 2.0 Differences

- Selenium-RC Server
- Flat API
- Well Documented
- Easy for beginners

- Direct Browser Access
- Object API
- Forums and Source Code
- Better for experienced coders *(but becoming simpler)*

# About the tools



# We will use

- Java SDK
- Maven
- Firefox
- IntelliJ
- Firefox Plugins
  - Firebug
  - Firepath

# Java SDK

- Java SDK/JDK, not the JRE
- SDK has the compiler
- 'javac -version'
- SDK downloaded from oracle
- http://www.oracle.com/technetwork/java/javase/downloads/index.html
- JRE download from Java.com

www.java.com/en/

Search

Java

Download Help

# JAVA + YOU, DOWNLOAD TODAY!

Free Java Download

» What is Java? » Do I have Java? » Need Help?

About Java

Java Developer Conference | Java + Alice | Java + Greenfoot | Java + BlueJ | Oracle Academy | Java Magazine

Select Language | About Java | Support | Developers | Feedback
Privacy | Terms of Use | Trademarks | Disclaimer

ORACLE

www.oracle.com/technetwork/java/javase/downloads/index.html

ORACLE

Sign In/Register Help Country Communities I am a... I want to... Search

Products Solutions Downloads Store Support Training Partners About

Oracle Technology Network > Java > Java SE > Downloads

- Java SE
- Java EE
- Java ME
- Java SE Support
- Java SE Advanced & Suite
- Java Embedded
- Java DB
- Web Tier
- Java Card
- Java TV
- New to Java
- Community
- Java Magazine

Overview | Downloads | Documentation | Community | Technologies | Training

## Java SE Downloads

DOWNLOAD

Java Platform (JDK) 8u65 / 8u66

DOWNLOAD

NetBeans with JDK 8

### Java Platform, Standard Edition

**Java SE 8u65 / 8u66**

Java SE 8u65 includes important security fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release. Java SE 8u66 is a patch-set update, including all of 8u65 plus additional features (described in the release notes).

Learn more

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Products

JDK

DOWNLOAD

Server JRE

Java SDKs and

- Java SE
- Java EE and Glas
- Java ME
- Java Card
- NetBeans IDE
- Java Mission Con

Java Resour

- Java APIs
- Technical Articles
- Demos and Videc
- Forums
- Java Magazine
- Java.net
- Developer Trainir
- Tutorials
- Java.com

www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.h

ORACLE

Sign In/Register Help Country Communities I am a... I want to... Search

Products Solutions Downloads Store Support Training Partners About O

Oracle Technology Network > Java > Java SE > Downloads

- Java SE
- Java EE
- Java ME
- Java SE Support
- Java SE Advanced & Suite
- Java Embedded
- Java DB
- Web Tier
- Java Card
- Java TV
- New to Java
- Community
- Java Magazine

Overview | Downloads | Documentation | Community | Technologies | Training

## Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 8u65 Checksum
JDK 8u66 Checksum

### Java SE Development Kit 8u65

**You must accept the Oracle Binary Code License Agreement for Java SE to download this software.**

Accept License Agreement  Decline License Agreement

| Product / File Description | File Size | Download |
|---|---|---|
| Linux ARM v6/v7 Hard Float ABI | 77.69 MB | jdk-8u65-linux-arm32-vfp-hflt.tar.gz |
| Linux ARM v8 Hard Float ABI | 74.66 MB | jdk-8u65-linux-arm64-vfp-hflt.tar.gz |
| Linux x86 | 154.67 MB | jdk-8u65-linux-i586.rpm |
| Linux x86 | 174.84 MB | jdk-8u65-linux-i586.tar.gz |

**Java SDKs and**

- Java SE
- Java EE and Glas
- Java ME
- Java Card
- NetBeans IDE
- Java Mission Cor

**Java Resour**

- Java APIs
- Technical Articles
- Demos and Videc
- Forums
- Java Magazine
- Java.net
- Developer Trainir
- Tutorials
- Java.com

# Environment Variables

- Need to add JDK bin folder to the path
- Need to add JAVA_HOME environment variable
  - Points to root folder of JDK

# Maven

- Maven is a java tool we use for dependency management
- pom.xml to define dependencies
- Also a build tool
- maven.apache.org
- Install by following the instructions on the site

https://maven.apache.org

Apache Maven Project
http://maven.apache.org/

Maven™

Apache / Maven / Welcome to Apache Maven

Last Published: 2016-01-13

MAIN

- Welcome
- License
- Download
- Install
- Configure
- Run
- IDE Integration

ABOUT MAVEN

- What is Maven?
- Features
- FAQ
- Support and Training

# Welcome to Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

If you think that Maven could help your project, you can find out more information about in the "About Maven" section of the navigation. This includes an in-depth description of what Maven is, a list of some of its main features, and a set of frequently asked questions about what Maven is.

This site is separated into the following sections, depending on how you'd like to use Maven:

| Use | Download, Install, Run Maven | Configure, Use Maven and Maven Plugins |
|---|---|---|
| | Information for those needing to build a project that uses Maven | Information for those wanting to use Maven to build their project, including a "10 minute test" that gives a practical overview of Maven's main features in just 10 minutes and plugin list for more information on each plugin |

# How to install

- 3.3.9
  - Download 'binary' zip, not the 'source' zip
  - Unarchive
  - Add to a folder with no spaces e.g.
    - C:\mvn\maven_3_3_9
  - Add \bin folder to path
- Older
  - As above, but need an M2_HOME variable pointing to root folder for mavn install

# Firefox

- Install Firefox
  - https://www.mozilla.org/en-GB/firefox/new/
- Drivers are installed for WebDriver by default so no need to install any extra code.
- This is the first browser you automate against

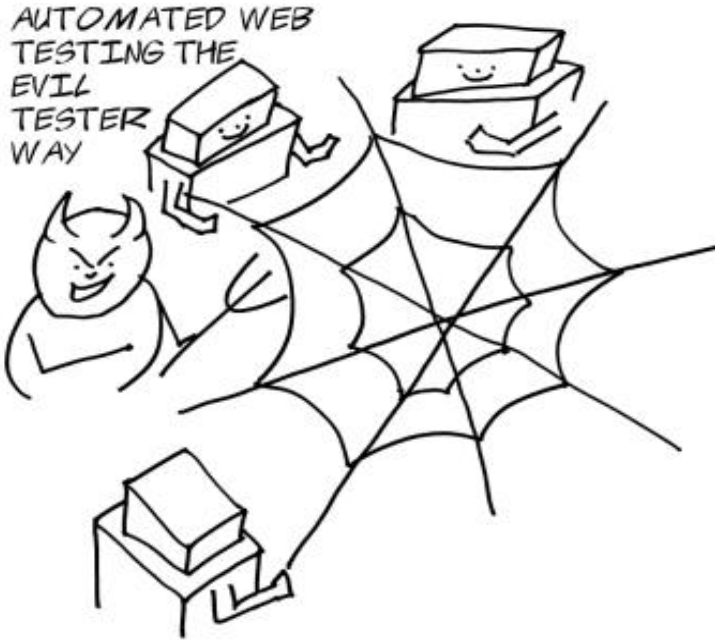
# IntelliJ

- https://www.jetbrains.com/idea/
- Install the Community Edition
  - Free
  - Open Source

# Firefox Plugins

- Firebug
  - Like the developer tools
- Firepath
  - Allows CSS and Xpath selectors to be used

# Create a Project



# Create a project structure

- using maven
  - IDE
  - Or By Hand (not recommended)

# What is Maven?

- A build tool
- Standard Folder Structure
- Java project tasks simpler
  - mvn clean compile
  - mvn package
  - mvn test
- http://maven.apache.org

# Maven

- Create project structure with Maven
  - mvn archetype:generate
    -DgroupId=com.seleniumsimplified.webdriver
    -DartifactId=webdriverbasics
    -DarchetypeArtifactId=maven-archetype-quickstart
    -DinteractiveMode=false
- http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html
- http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html

# In the IDE

- Import project
- Amend pom junit version to 4.11
- Compile & Install

# Amend the basic test

```
package com.eviltester.com;

import org.junit.Test;
import static junit.framework.Assert.assertTrue;

public class AppTest
{
    @Test
    public void testApp()
    {
        assertTrue( true );
    }
}
```

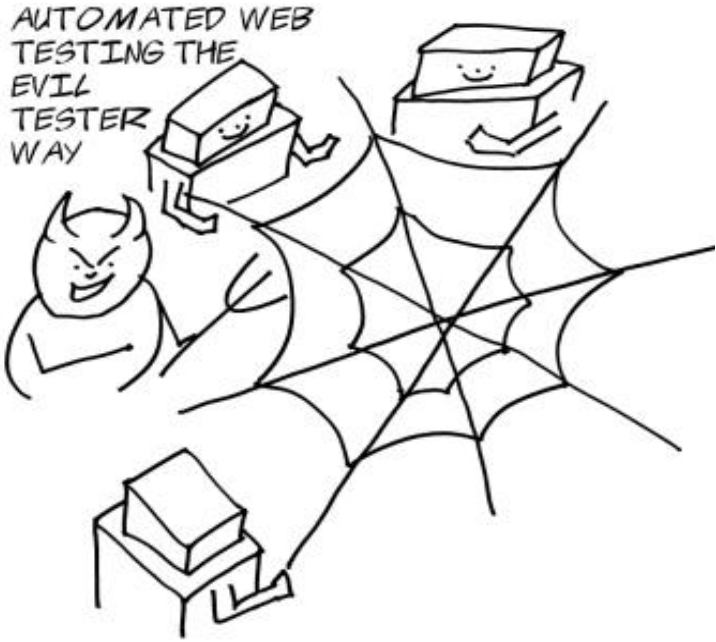
# Add Selenium to the pom.xml

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>

    <artifactId>selenium-server</artifactId>

    <version>2.53.0</version>

</dependency>
```

# Benefits of Maven

- Standard directory structure
- Supported by Selenium and some of the drivers
- Easy to update and install
- Simple to add to Continous Integration
  - “mvn test”

# Begin Coding



# In Selenium 2.0 The Driver is King

- Selenium 1.0
  - The Server is King
- Selenium 2.0
  - The Driver is King

# Add HtmlUnitDriver to the pom.xml

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>htmlunit-driver</artifactId>
    <version>2.20</version>
</dependency>
```

# First Test

```
@Test
public void driverIsTheKing(){
    WebDriver driver = new HtmlUnitDriver(true);
    driver.get("http://www.compendiumdev.co.uk/selenium");
    assertTrue(driver.getTitle().startsWith(
                  "Selenium Simplified"));
    }
```

# First Test Explained

Junit test

Core Interface to understand

A headless Implementation of a Driver

```
@Test
public void driverIsTheKing(){
    WebDriver driver = new HtmlUnitDriver(true);
    driver.get("http://www.compendiumdev.co.uk/selenium");
    assertTrue(driver.getTitle().startsWith(
                "Selenium Simplified"));
    }
```

Most basic navigation command

A JUnit assert

Return page title as a string

# Exercise

- Familiarise yourself with your IDE
  - IntelliJ
    - use the project config to setup “autoscroll”
    - experiment with
      - code completion commands ctlr+space
      - Param completion ctrl+p
      - Javadoc Ctrl+Q
- Experiment and play a little with the test, e.g.
  - change URL,
  - check for different text

# FirefoxDriver vs Marionette

- Up till Firefox v 45 we use FirefoxDriver
  - Built into WebDriver
  - You can download Firefox v 45
  - https://www.mozilla.org/en-US/firefox/organizations/all/
- For Firefox > 46 we use Marionette
  - A separate download
  - https://developer.mozilla.org/en-US/docs/Mozilla/QA/Marionette/WebDriver

# Use a real browser with FirefoxDriver when Firefox version <= 45

```
@Test
public void firefoxIsSupportedByWebDriver(){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.compendiumdev.co.uk/selenium");
    AssertTrue( driver.getTitle().
                startsWith("Selenium Simplified"));
    driver.close();
    driver.quit();
}
```

# Real Browser Explored

Implementations exist for Firefox, Chrome, IE Opera, etc.

```
@Test
public void firefoxIsSupportedByWebDriver(){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.compendiumdev.co.uk/selenium");
    assertEquals(true,
                 driver.getTitle().
                 startsWith("Selenium Simplified"));
    driver.close();
    driver.quit();
}
```

close() and quit() physical browsers

# To use Marionette

- Download the latest executable
  - https://developer.mozilla.org/en-US/docs/Mozilla/QA/Marionette/WebDriver
  - Unarchive to a directory
  - Rename to wires.exe
  - Add wires.exe to the Path

# Use a real browser with Marionette when Firefox version > 46 and Marionette in the Path

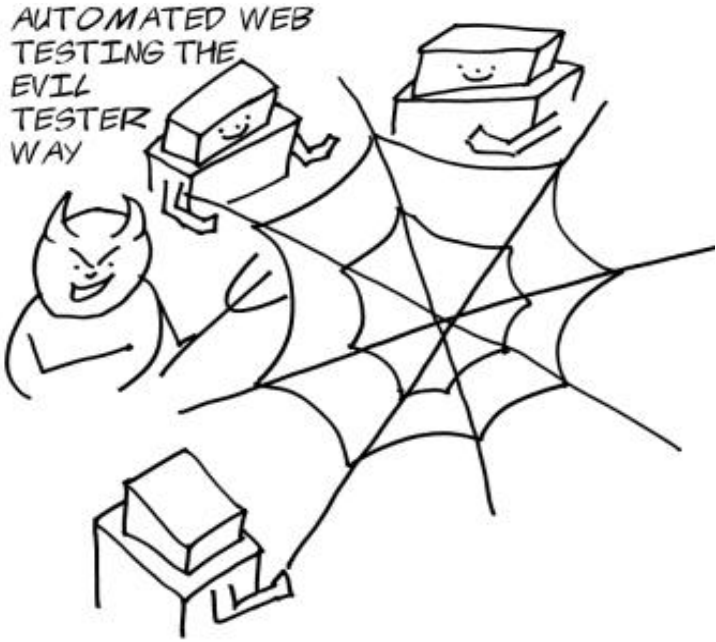
```
@Test
public void firefoxIsSupportedByWebDriver(){
    WebDriver driver = new MarionetteDriver();
    driver.get("http://www.compendiumdev.co.uk/selenium");
    AssertTrue( driver.getTitle().
                startsWith("Selenium Simplified"));
    driver.close();
    driver.quit();
}
```

# Use a real browser with Marionette when Firefox version > 46 and Marionette not in the Path

```
@Test

public void firefoxIsSupportedByWebDriver(){

    String currentDir = System.getProperty("user.dir");

    String marionetteDriverLocation = currentDir +
                   "/../tools/marionette/wires.exe";

    System.setProperty("webdriver.gecko.driver",
                      marionetteDriverLocation);

    WebDriver driver = new MarionetteDriver();

    driver.get("http://www.compendiumdev.co.uk/selenium");

    AssertTrue( driver.getTitle().

                 startsWith("Selenium Simplified"));

    driver.close();

    driver.quit();

}
```

Set the location of the wires.exe in the code

# Basic Automation Knowledge



What are the basics that we need to know so that we can automate the web?

# Basic Knowledge

- Open a browser
- Navigate to page
- Read the page title
- Read the url
- Get text from the page
- Click on links
- Fill in forms
- Click on buttons to submit forms

# Basic Knowledge Chunked

- *Open a browser*
- *Navigate to page*

} **Navigate**

- *Read the page title*
- *Read the url*
- *Get text from the page*

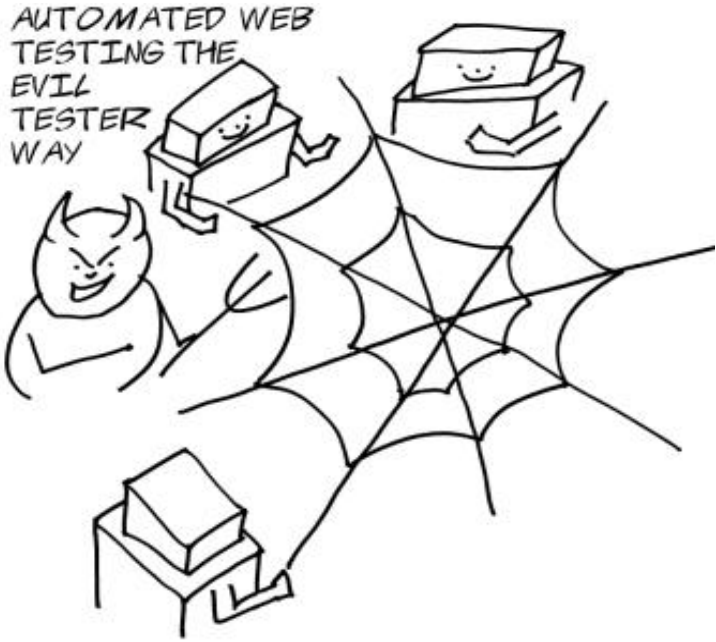
} **Interrogate**

- *Click on links*
- *Fill in forms*
- *Click on buttons*

} **Manipulate**

**& Synchronize**

# JUnit



# Junit Basics

- Add to pom.xml

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.10</version>
  <scope>test</scope>
</dependency>
```

- junit.org
- @Test

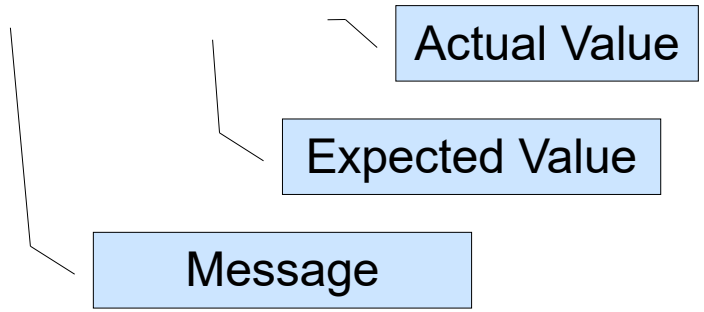
# Assertions

- assertTrue(“description”, expected, actual)
  - AssertFalse(...)
- assertEquals(“description”, expected, actual)

- Used to check results

# Junit Example

```
import org.junit.Test;

import static junit.framework.Assert.assertEquals;

public class JUnitExampleTest {

    @Test
    public void aBasicJUnitTest(){

        assertEquals("2+2=4", 4, 2+2);

    }
}
```



# More JUnit

- Avoid repeating code in your @Test methods
- Before and after each @Test
  - @Before
  - @After
- Once per class
  - @BeforeClass
  - @AfterClass
  - Declare methods as static

# JUnit Exercises

- Create a test with:
  - assertTrue, assertFalse, assertEquals
- Use @Before to assign a value to a field and use it in a test
- Use @BeforeClass to assign a value to a field and use it in a test
- Explore the documentation at junit.org
- Explore the source for JUnit in your IDE
- How do you use assertThat?

This is for bonus points, research required.

# JUnit Exercises Sample Answers

- Junit asserts
  - junit.sourceforge.net/javadoc/org/junit/Assert.html
- assertThat
  - Uses 'matchers'
  - junit.sourceforge.net/javadoc/org/junit/matchers/JUnitMatchers.html
  - junit.sourceforge.net/javadoc/org/hamcrest/core/package-summary.html

# A little more Hamcrest with JUnit 4.10

- http://code.google.com/p/hamcrest/
- JUnit 4.10 has a cut down version with core
- So how do we use more?
- junit-dep
  - Easy to use 1.1 with JUnit 10
  - Require a maven exclusion to use > 1.1
- e.g.

*NavigationBasicsTest.java navigateWithRefresh*

- *assertThat(endTime, greaterThan(startTime));*

# Hamcrest Summary

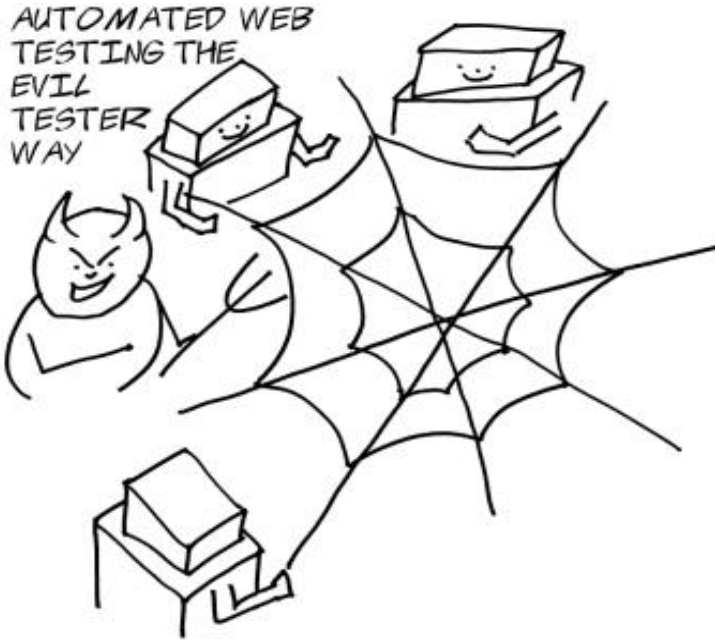
- Create readable assertions
- assertThat( actual, is(expected value));
  - Check that an actual value is as expected
- Look at the code to see what methods are available:
  - is, not, greaterThan, lessThan, endsWith, startsWith, containsString, etc.
- Read the docs
- http://code.google.com/p/hamcrest/wiki/Tutorial#A_tour_of_common_matchers

# Hamrest Matchers CheatSheet

- is
- not
- *Number*
  - greaterThan,
  - greaterThanOrEqualTo,
  - lessThan,
  - lessThanOrEqualTo - test ordering
- *Text*
  - equalToIgnoringCase
  - equalToIgnoringWhiteSpace
  - containsString,
  - endsWith,
  - startsWith
- *Collections*
  - hasEntry,
  - hasKey,
  - hasValue - test a map contains an entry, key or value
  - hasItem,
  - hasItems - test a collection contains elements
  - hasItemInArray - test an array contains an element
- notNullValue,
- nullValue - test for null

- http://code.google.com/p/hamcrest/wiki/Tutorial#A_tour_of_common_matchers

# IntelliJ IDE Tips



# Project Panel

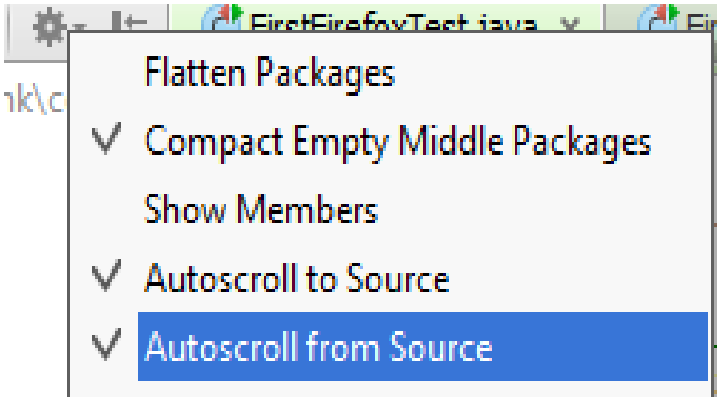
- Use the Cog dropdown menu
- Show Members
- AutoScroll to Source
- AutoScroll from Source

# Navigation Shortcut Keys

- Find Symbol
  - Shift + Cntrl + Alt + N
- Find Class
  - Cntrl + N
- Find File
  - Shift + Ctnrl + N

# WebDriver API Learning Tips

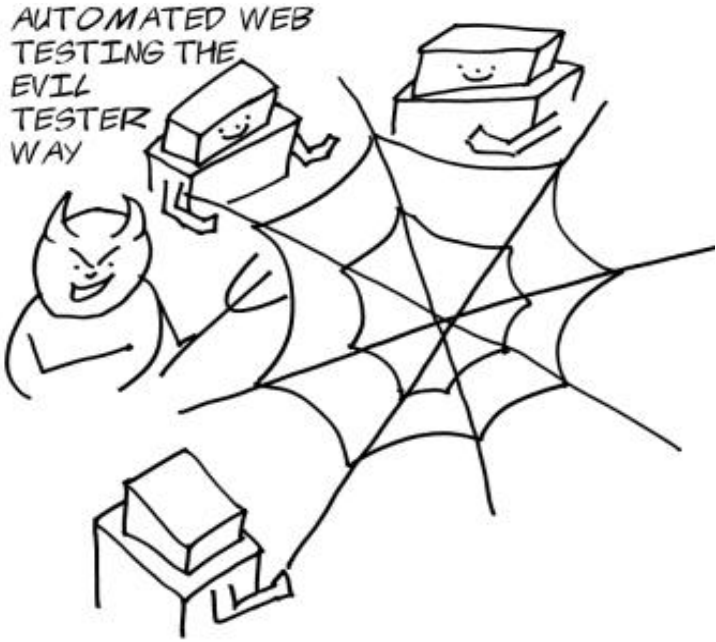
- Use Code Completion
- ctrl+Q for Documentation of Commands
- ctrl+click to view the actual WebDriver code
- AutoScroll



# Basics

- Use code completion as much as possible
- Learn short cut keys
- Remove any code syntax errors as fast as possible because they interfere with code completion

# Navigation



# Navigation

- driver
  - .get(<url>)
  - .*navigate*
    - .to(<url>)
    - .to(URL)
    - .back()
    - .forward()
    - .refresh()

Simple set of direct navigation commands which also include the traditional shortcut back/forward paths.

In practice I tend to use .get(...) more than .navigate.to(...)

# Navigation Annotated

- driver
  - .get(<url>)
  - *.navigate*
    - .to(<url>)
    - .to(URL)
    - .back()
    - .forward()
    - .refresh()

driver.get(“http://aURL.com”);

'to' a URL String

'to' a java.net.URL Object

'back' through browser history

'forward' through browser history

'refresh' the current browser page

# A Note on Code

- In the slides you will see:
  - ```
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.compendiumdev.co.uk/selenium");
    ```
- In the code you will see:
  - ```
    driver =
      Driver.get("http://www.compendiumdev.co.uk/selenium");
    ```
- This was for speed of execution to allow sharing the browser between tests.

# Navigation Example Test

```
private static WebDriver driver;

@BeforeClass
public static void createDriver(){

        driver = new FirefoxDriver();
}


@Test
public void navigateWithNavigateTo(){

   driver.navigate().to(
           "http://www.compendiumdev.co.uk/selenium/search.php");

   assertTrue(driver.getTitle().
                 startsWith("Selenium Simplified Search Engine"));
}

@AfterClass
public static void quitDriver(){

        driver.quit();
}
```

driver needs to be static because of @BeforeClass and @AfterClass

Execute this method once per class. Before any of the @Test methods have run.

Execute this method once per class. After all @Test methods have run.

# Navigation Exercise

- Use the following URLs;
- navigate() .to(), .forward(), back(), .refresh()
- Assert on titles to check navigation worked
- http://compendiumdev.co.uk

| File Path | Title |
|---|---|
| /selenium | "Selenium Simplified" |
| /selenium/search.php | "Selenium Simplified Search Engine" |
| /selenium/basic_html_form.html | "HTML Form Elements" |
| /selenium/basic_web_page.html | "Basic Web Page Title" |
| /selenium/refresh.php | "Refreshed Page on ([0-9]{10})" |

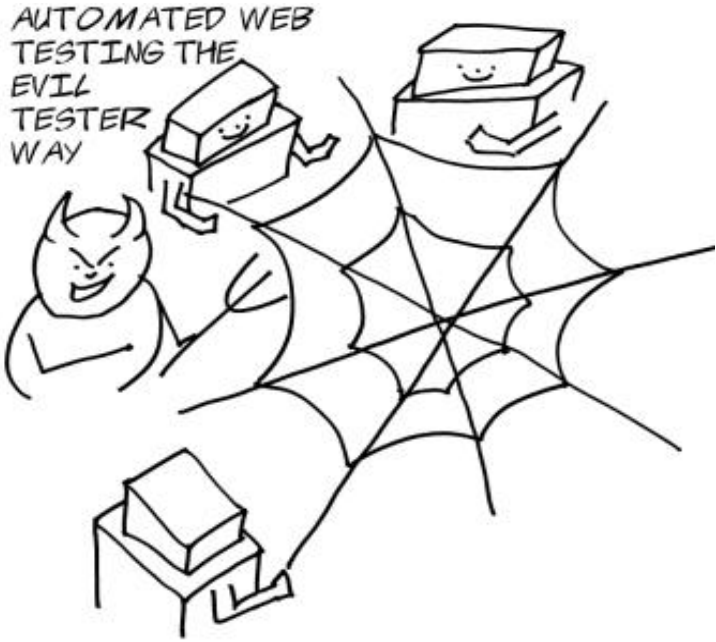
# Does Navigation Require Synchronisation?

- Navigation blocks until the HTML of the page is loaded
- This does not mean the page is ready
  - for manipulation or
  - full interrogation
- Sometimes synchronisation is required on navigation

# What about clicking?

- Normally we navigate by clicking:
  - on links
  - on buttons
  - by submitting forms
- This is navigation as a side-effect
  - It requires synchronisation
  - We will cover this later in Manipulation

# Interrogation



# Interrogation

- Driver
- Dom (WebElement based)

# Driver level Interrogation Methods

- driver
  - .getTitle()
  - .getCurrentUrl()
  - .getPageSource()

All return String.

Pretty obvious what each method does.

Be wary of using getPageSource it may not return what you expect.

# Driver Level Interrogation Exercise

- Create a test which:
  - Visits
    - www.compendiumdev.co.uk/selenium/basic_web_page.html
  - Assert title equals “Basic Web Page Title”
  - Assert current URL is what you expect
  - Check page source contains “A paragraph of text”

# DriverInterrogateTest

```
@Test
public void driverLevelPageInterrogateMethods(){

    WebDriver driver;

    final String theTestPageURL =
      "http://www.compendiumdev.co.uk/selenium/basic_web_page.html";

    driver = new FirefoxDriver();
    driver.navigate().to(theTestPageURL);

    assertEquals(driver.getTitle(), "Basic Web Page Title");
    assertEquals(driver.getCurrentUrl(), theTestPageURL);

    String pageSource = driver.getPageSource();
    assertTrue(pageSource.contains("A paragraph of text"));

   driver.close();
    driver.quit();

}
```

# Be Careful with getPageSource

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
    <head>
        <title>Basic Web Page Title</title>
    </head>
    <body>
         <p id="para1" class="main">A paragraph of text</p>
        <p id="para2" class="sub">Another paragraph of text</p>
    </body>
</html>
```

**File on server**

Line separation

Additional attributes

Attribute Ordering

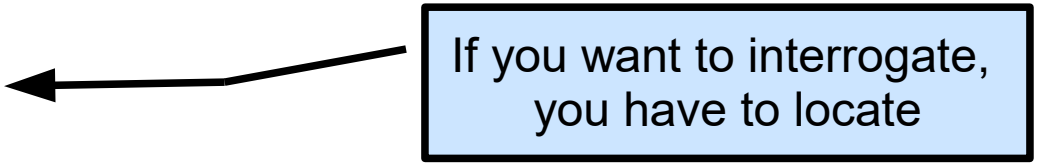
**Firefox Driver Differences**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
         <title>Basic Web Page Title</title>
    </head>
    <body>
        <p class="main" id="para1">A paragraph of text</p>
         <p class="sub" id="para2">Another paragraph of text</p>

</body></html>
```

**String returned by method**

# Dom Element Interrogation Approach

- Find the Dom Element (WebElement)
  - .findElement
  - .findElements




- Use the WebElement Object methods:
  - .getText()
  - .getAttribute()
  - .getTagName()
  - .isEnabled()
  - .isSelected()
  - .isDisplayed()
  - .getSize()
  - .getLocation()
  - .getCssValue()

# WebElement Interrogation Example

```
@Test
public void WebElementInterrogationBasics(){

    final WebDriver driver;
    final String theTestPageURL =
            "http://www.compendiumdev.co.uk" +
            "/selenium/basic_web_page.html";

    driver = new FirefoxDriver();

    driver.navigate().to(theTestPageURL);

    WebElement para1 = driver.findElement(By.id("para1"));

    assertEquals(para1.getText(),"A paragraph of text");

    driver.close();
}
```

Find an Element By using a locator strategy, in this case the id of the element.

WebElement represents an object in the Dom

Inspect the WebElement using the exposed methods.

# Driver .findElement(By)

```
driver.findElement(By.id("para1"))
```

- By.id
- By.xpath
- By.cssSelector
- By.className
- By.linkText
- By.name
- By.tagName
- By.partialLinkText

We find an element By using a locator strategy.

e.g. by using the id, by evaluating an xpath expression, by executing a css selector, etc.

# ExampleFirstFindByTest

```
public class ExampleFirstFindByTest {

    static WebDriver driver;

    @BeforeClass
    public static void createDriverAndVisitTestPage(){
        driver = new FirefoxDriver();
        driver.get("http://www.compendiumdev.co.uk/" +
                "selenium/find_by_playground.php");
    }

    @Test
    public void findByID(){
        WebElement cParagraph = driver.findElement(By.id("p3"));
        assertEquals("This is c paragraph text",
                        cParagraph.getText());
    }

    @AfterClass
    public static void closeBrowser(){
        driver.quit();
    }
}
```

# First find By Exercises

- Using http://www.compendiumdev.co.uk
  - /selenium/find_by_playground.php
- findElement and then assert
  - e.g. getText(), getAttribute("id")
- Create a test for each By
  - By.id
  - By.linkText
  - By.name
  - By.partialLinkText
  - By.className

Inspect the DOM or look at the page source to identify locator approaches

If multiple elements match then findElement will return the first.

"NoSuchElementException" means that the locator didn't match anything in the DOM.

# First findElement & By Sample Answer Explored

- With multiple tests it becomes clear why we use @BeforeClass and @AfterClass
- Note @Test(expected = …)
  - Import the correct exceptions

# .findElements

- .findElement only returns 1 WebElement
- When a By can return more elements then .findElement returns the first in the list
- .findElements does not throw an exception if it can't match anything, it returns an empty list

- .findElements returns the full list of matching WebElements
  - e.g. `driver.findElements(By.className("normal"));`

# FindElements Example Test

```
@Test
public void returnAListOfElementsByClassName(){

    List<WebElement> elements;
    elements = driver.findElements(
                            By.className("normal"));

    Set<String> foundTags = new HashSet<String>();

    for(WebElement e : elements){
        foundTags.add(e.getTagName());
    }

    assertTrue("expected p tag", foundTags.contains("p"));
    assertTrue("expected ul tag", foundTags.contains("ul"));
    assertTrue("expected li tag", foundTags.contains("li"));
    assertTrue("expected a tag", foundTags.contains("a"));
    assertFalse("did not expect div tag", foundTags.contains("div"));
}
```

Need to use a List<WebElement>

There are many items in the DOM with this className

Iterate around them to find the matching tags

Assert that we found the expected elements

# FindElements Exercises

- Assert that there are 18 div elements
- Assert that there are 25 <a> which href to a para

*Hint: List has a .size() method*

- For Bonus points: Assert that there are 16 nested paragraphs and 41 paras in total

# FindElements sample Answer

- Did you notice findElements does not throw an exception if no match?
- //div
- //a and //a[@href]
- //p[contains(.,"nested")]

# Chaining .findElement (s)

findElement(By.id(“.”)).findElement(By.name(“.”))

e.g.

```
WebElement element = driver.findElement(By.id("div1")).
                            findElement(By.name("pName3")).
                            findElement(By.tagName("a"));
```

- Can use any By locator strategy

- Cannot Chain .findElements() as it returns a List of Web Elements

```
List<WebElement>
```

# Chaining with ByChained

- ByChained is a support class

```
import org.openqa.selenium.support.pagefactory.ByChained;
```

- ByChained extends By (*it is a By*)
- Instantiate it and pass in the By objects

```
element = driver.findElement(
                    new ByChained(
                        By.id("div1"),
                        By.name("pName9"),
                        By.tagName("a")));
```

Have to instantiate, not use statically

Takes a list of By Objects

# Other By Support Classes

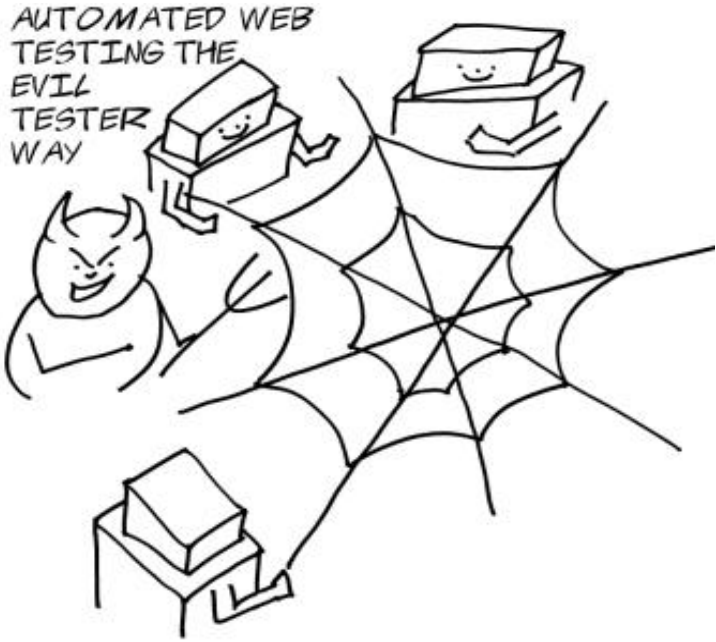
- ByIdOrName("string to match")

Takes a String which could be the ID or the Name

```
@Test
public void findByIdOrNameMatchOnName(){

    WebElement element;
    element = driver.findElement(
                        new ByIdOrName("pName2"));

    assertEquals("expected a match on name",
            "This is b paragraph text",
            element.getText());
}
```

Have to instantiate, not use statically

# CSS Selectors



# CSS Selectors

- A CSS Selector matches a set of elements
- Used to style HTML from CSS Stylesheets
- Reused in Selenium to match DOM elements
- Useful References
  - https://developer.mozilla.org/en-US/docs/CSS
  - http://reference.sitepoint.com/css/selectorref
  - http://net.tutsplus.com/tutorials/html-css-techniques/the-30-css-selectors-you-must-memorize/
  - http://www.quirksmode.org/css/contents.html
  - http://www.w3schools.com/cssref/css_selectors.asp
  - http://css-tricks.com/attribute-selectors/

# Basics of CSS Selectors

- *
  - match any element
- #id
  - match an id e.g. #p4
- .class
  - match a class e.g. “.normal”
- tag
  - match tags
- [attribute]
  - Match on the attribute name

# CSS Attribute Matching

- tag[attribute]
  - match tags with an attribute
- tag[attribute=”value”]
  - match tags with a specific attribute value
- tag[attr1='val1'][attr2='val2']
  - match tag using multiple attribute values
- tag[attribute*=”alu”]
  - Partial match anywhere on value
- tag[attribute^=”val”]
  - Partial match start of value
- tag[attribute$=”lue”]
  - Partial match end of value
- tag[attribute~=”value”]
  - Match on space separated values
- tag[a='val'], tag[b='val']
  - , is an 'or' grouping

# By.cssSelector

- By.cssSelector(<a css selector string>)

```
@Test
public void findElementsUsingCSSTag(){

    List<WebElement> elements;

    elements = driver.findElements(
                    By.cssSelector("p"));

    assertEquals("expected a different number",
                    41,
                    elements.size());
}
```

# Useful Tools For CSS and XPath

- Firefox
  - Install FireBug and FirePath plugins
- Chrome
  - Developer tools are supposed to allow search using xpath or css (sometimes this breaks between releases)

# CSS Basic Exercises

- Use By.cssSelector as replacement for
  - By.id, By.name, By.className, By.tagName

| **Replace** | **Assert** |
|---|---|
| By.id(“p31”) | getAttribute(“name”) == “pName31” |
| By.name(“ulName1”) | getAttribute(“id”) == “ul1” |
| By.className(“specialDiv”) | getAttribute(“id”) == “div1” |
| By.tagName(“li”) | getAttribute(“name”) == “liName1” |

  - Create test first using By.id (etc.)
  - Check test works
  - Replace By.id (etc.) with By.cssSelector

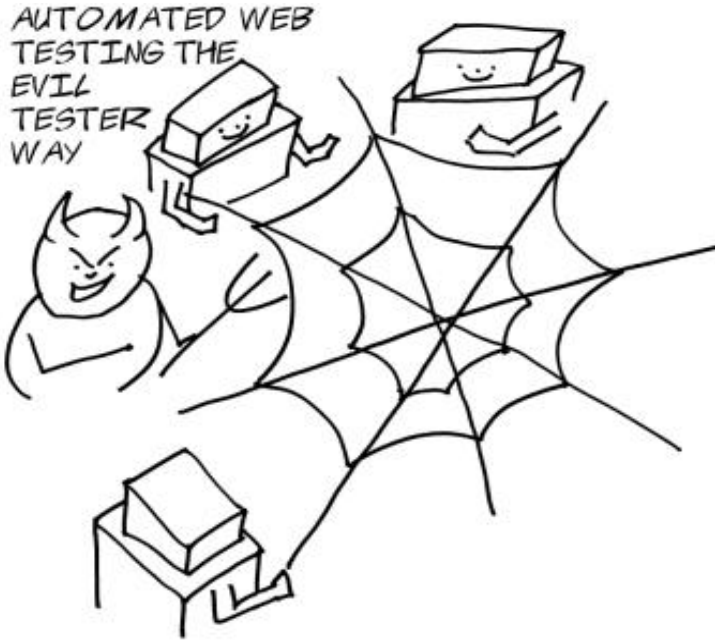
# CSS Basics – Sample Answers

- I have two sets of sample answers because there are so many ways of fulfilling the result
  - FindByCSSSelectorBasicExercisesTest
  - FindByCSSSelectorBasicExercisesFullAnswersTest

# CSS Selectors – Some Paths

- A > B – B directly under A e.g. <A><B/></A>
- A B – descendant
  - selectors separated by space i.e. “this then that”
  - Any degree of separation
  - e.g. “div li” would match but “div > li” would not
- A + B – B siblings of an A
- B:first-child - every B which is first child of something
  - For more selectors see the references

# XPath



# XPath

- XML is an XML Query Language
- XPath is often slower than CSS
- XPath has more functions than CSS
- Xpath can navigate up and down the DOM

- References
  - http://www.w3schools.com/xpath/
  - http://www.w3schools.com/xpath/xpath_functions.asp

# By.xpath

Match p elements anywhere in the DOM which have a name of 'pName5'

```
@Test
  public void findSpecificPara(){

      WebElement element;
      element = driver.findElement(

              By.xpath("//p[@name='pName5']"));

      assertEquals("Expected matching id",
                   "p5",
                   element.getAttribute("id"));
  }
```

# Xpath Selector Basics

- // - match anywhere
- / - match from root
- //* - any element
- //tag – named element
- //*[@attribute] – match if it has attribute
- //*[@attribute='value'] – attribute with value
- . for content matching
- .. for traversing back up the path e.g. //div/p/a/..
- Operators: and, or etc
  - w3schools.com/xpath/xpath_operators.asp
- Xpath functions e.g. contains, starts-with
  - w3schools.com/xpath/xpath_functions.asp

# XPath Basic Exercises

- Use By.xpath as replacement for
  - By.id, By.name, By.className, By.tagName

| **Replace** | **Assert** |
|---|---|
| By.id(“p31”) | getAttribute(“name”) == “pName31” |
| By.name(“ulName1”) | getAttribute(“id”) == “ul1” |
| By.className(“specialDiv”) | getAttribute(“id”) == “div1” |
| By.tagName(“li”) | getAttribute(“name”) == “liName1” |

  - Create test first using By.id (etc.)
  - Check test works
  - Replace By.id (etc.) with By.xpath

# XPath Basics – Sample Answers

- I cheated and copy pasted the CSS test – what did you do?
- FindByXPathSelectorBasicExercisesFullAnswersTest
  - You may not get access to all xpath functions e.g.
    - //*[ends-with(@name,'ulName1')]
    - //*[matches(@name,'ulName1')]
    - Depends on Browser support for xpath
      - Currently Xpath 1.0
      - http://www.w3.org/TR/xpath/

# Xpath, CSS, By.*revisited*

## *The Challenge*

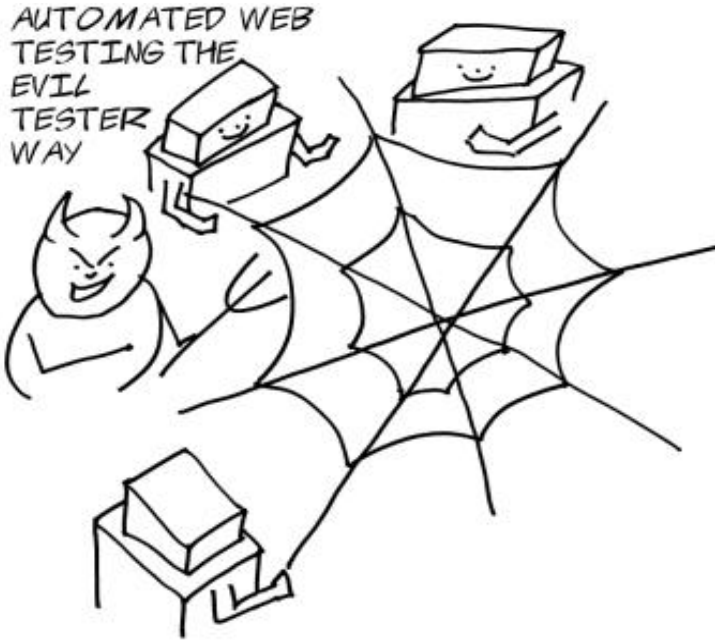
# Access any DOM elements

- DOM access isn't just restricted to body tags
- Can also be used on elements in the head
- Challenge
  - Assume driver.getTitle does not exist
  - Write tests which replicate the getTitle functionality using .findElement, .getText, and whatever locator strategy you want (By.id, By.name, etc.)
  - Can you do it with Xpath? Can you do it with CSS?
    - GetTitleReplacementExerciseTest

# Additional References

- Xpath, CSS Rosetta Stone
  - http://www.simple-talk.com/dotnet/.net-framework/xpath,-css,-dom-and-selenium-the-rosetta-stone
  - http://bit.ly/RDJ3Wb

# Manipulation



# WebElement Manipulation

- .click()
- .clear()
  - Clear a field
- .sendKeys(String) *actually (CharSequence)*
  - Sends the appropriate events to a WebElement keyup, keydown, etc.
  - Helper Keys class (org.openqa.selenium.Keys)
- .submit()
  - Submit a form

# Manipulation 1st Try

- compendiumdev.co.uk/selenium/basic_ajax.html
- Manually, select “server”, select “java”, submit, check submitted details
- automate with .click and previous interrogation learning
- Investigate ...

# Introducing WebDriverWait

- WebDriver has a helper class WebDriverWait which can help us synchronise our tests
- Used in conjunction with another helper class ExpectedConditions
- We can write simple synchronisation statements

```
new WebDriverWait(driver,10).
          until(
                ExpectedConditions.titleIs("HTML Form Elements"));
```

# WebDriverWait Example

```
@Test
public void exampleUsingExpectedConditions(){

    WebDriver driver;

    driver = driver.get("http://compendiumdev.co.uk/selenium/" +
                        "basic_html_form.html");



    new WebDriverWait(driver,10).until(
            ExpectedConditions.titleIs("HTML Form Elements"));


    assertEquals("HTML Form Elements", driver.getTitle());
}
```

Construct a WebDriverWait with the driver and a timeout in seconds.

ExpectedConditions has a lot of common conditions that you would otherwise have to code yourself.

**Explore the methods**

I don't really need this assert because WebDriverWait will throw a TimeoutException if the ExpectedCondition is not met

# Manipulation Example Test

```
public class ManipulateExampleTest {

    static WebDriver driver;

    @BeforeClass
    public static void createDriverAndVisitTestPage(){
        driver = new FirefoxDriver();
        driver = driver.get( "http://www.compendiumdev.co.uk" +
                                "/selenium/basic_html_form.html");
    }

    @Test
    public void simpleInteraction(){
        WebElement checkBox1 = driver.findElement(
                By.cssSelector("input[value='cb1']"));

        assertFalse("Starts not selected",
                checkBox1.isSelected());

        checkBox1.click();

        assertTrue("Click selects",
                checkBox1.isSelected());
    }


    @AfterClass
    public static void closeBrowser(){
        driver.quit();
    }
}
```

# Manipulation Exercises

Using http://compendiumdev.co.uk/selenium/basic_html_form.html

1) Submit form and assert page title changes
2) Clear, then type comments, submit form and check output
3) Submit form with radio 2 selected
4) Submit form with only checkbox 1 selected
5) Submit form with drop down item 5 selected
6) Submit form with multiple select 1, 2 & 3
7) Bonus points, submit with a file, and check name on output

# Synchronisation Problems?

- Changing DOM state =
  risk of synchronisation problems

- For these examples would waiting for the page title would be 'good enough'?
  - See submitFormWithDropDownFiveUsingKeyboardFullText
  - Can change between browser versions

```
new WebDriverWait(driver,10).
      Until(
         ExpectedConditions.titleIs("Processed Form Details"));
```

# Manipulation Exercises Review

- How many ways to submit a form?
  - Which is 'best'?
- Is .click robust on checkbox?
  - What could go wrong?
- Options for checking an element is missing?
- Strategies for working with Drop Down?
- Does multiple select work the same way as the real world?
- Any repeated code?
  - What did/could you do about that?

# Manipulation Summary

- Filename fields use .sendKeys
- Checkbox and radio items
  - .isSelected, .click
- Can clear text or text area
- Cannot clear a checkbox, multiselect, drop down, or radiobutton
- Simple manipulation has a very simple set of methods

# More on SendKeys

- Keys.chord to send Key Sequences
- Shift, CTRL etc start a modifier sequence
- Keys.NULL ends a modifier sequence

```
aTextArea.sendKeys("this text", "and this text");

commentTextArea.sendKeys(
      Keys.chord(Keys.SHIFT, "bob", Keys.NULL, " Dobbs"));

assertEquals("BOB Dobbs", commentTextArea.getText())
```

# Support Class for Select

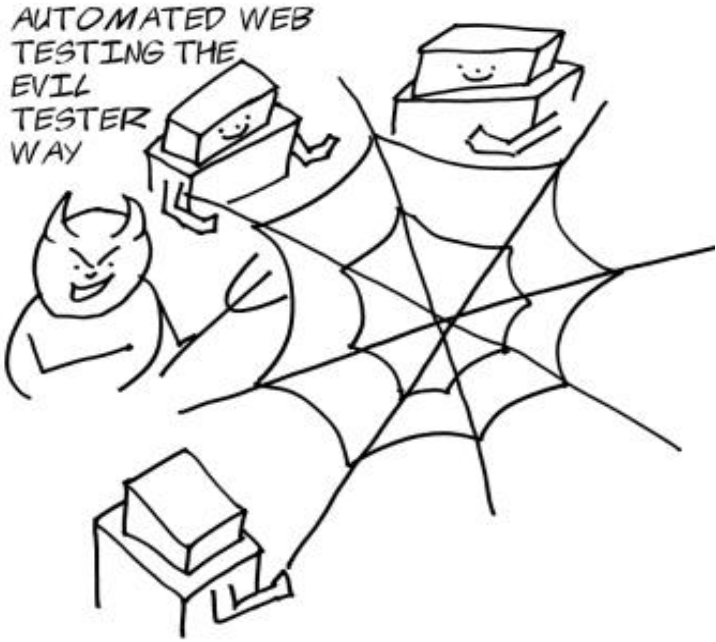
- Select
  - .isMultiple
  - .getOptions
  - .getAllSelectedOptions
  - .getFirstSelectedOption
  - .selectByVisibleText
  - .deselectAll
  - etc.

- Makes working with selects a little easier

```
new Select(WebElement)
```

- Try repeating the previous select 1,2 and 3 exercise using Select

# User Interactions



# User Interactions

May more accurately simulate user interaction

- ***new Actions(driver)***
- Actions Sequence
  - .keyDown
  - .keyUp
  - .sendKeys
  - .clickAndHold
  - .release
  - .click
  - .doubleClick
  - .moveToElement
- .moveByOffset
- .contextClick
- .dragAndDrop
- .dragAndDropBy
- *.build()*
- ***.perform()***

Build if you want to store and reuse the action sequence, otherwise just perform

# User Interactions Example - Cautionary

```
Actions actions = new Actions(driver);

actions.click(multiSelectOptions.get(0)).
        click(multiSelectOptions.get(1)).
        click(multiSelectOptions.get(2)).perform();

clickSubmitButton();
```

- *Sometimes this works, sometimes it doesn't. But you don't need User Interactions to achieve the effect. Use carefully*
- Previously all 3 were selected, which didn't mirror real life
- Here, only 1 is selected, which does mirror real life

# User Interactions Demo

- Demo explaining commands
- http://compendiumdev.co.uk/selenium/basic_html_form.html
- Click on input[value='cb1']
- Show build vs perform
- Drill in to code
- Explain commands, chaining
- Explain native vs Javascript events

# Use User Interactions Carefully

- Can vary between machines & browsers
- Can vary between browser versions
- If plain WebDriver works then use it
  - Use User Interactions for 'complex' situations
  - A last resort
- If you are having problems then check the forums, you might not be the only one

# User Interactions Exercises

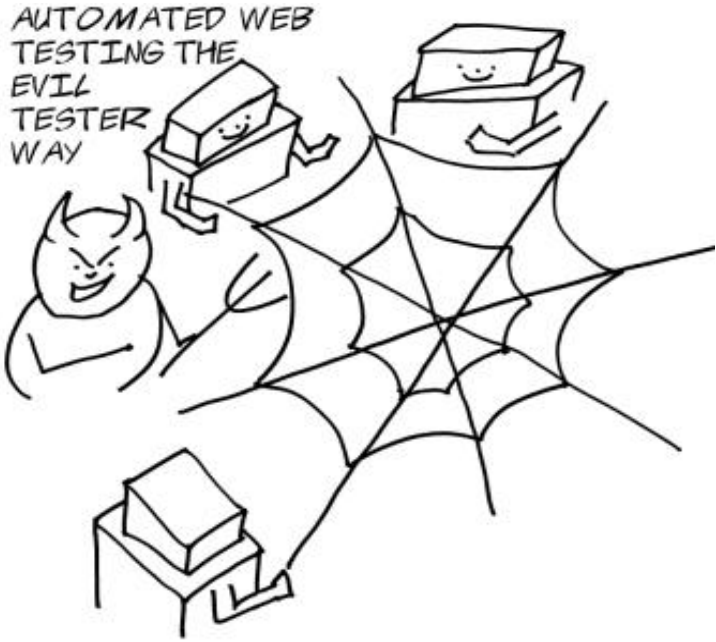
http://compendiumdev.co.uk/selenium/gui_user_interactions.html

- Move draggable1 on to droppable1
  - assert droppable1 text change to “Dropped!”
- Drag and Drop draggable2 to droppable1
  - Assert droppable 1 text change to “Get Off Me!”
- Press control+B and assert for text change on draggable1 to “Bwa! Ha! Ha!”
- Challenges:
  - Control+Space and red squares say “Let GO!!” can you assert for this?
  - Can you draw something in the canvas?

# User Interactions Exercises

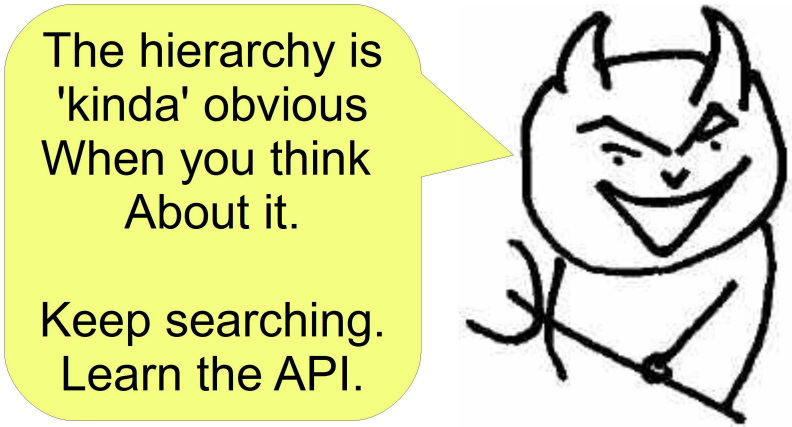
- How did you get on with Control+B
  - Unfortunate key choice?
- How did you get on with Control+space?
  - Could trigger but couldn't assert?
  - Yup, can't use KeyDown with space
- How didyou get on with drawing?
  - If you want an additional challenge, try drawing a picture.

# Alerts



# Alerts

- Handle Alerts with
  - driver.switchTo().alert()
    - .getText()
    - .dismiss()
    - .accept()
    - .sendKeys(String)
  - .alert() returns an Alert object
- Does alert exist?
  - Switch to it and catch the exception to check



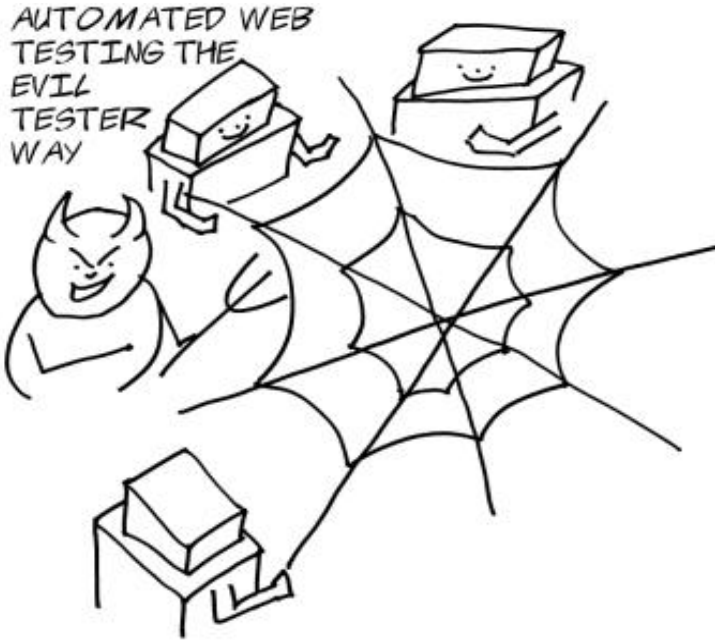
# Alert Exercises

- Using http://compendiumdev.co.uk/selenium/alerts.html
- For each alert, assert against getText()
- Accept and dismiss alert
- Accept and dismiss confirm (check text on page changes)
- Accept and dismiss prompt (check text on page changes)
- Change text on prompt and accept and dismiss

# Alerts Retrospective

- Not too hard
- Firefox throws an exception if you send keys to an alert that doesn't expect it
- Chrome does not

# Frames & Windows



# Frames

- driver.switchTo().
  - frame(“framename”)
  - frame(int)
  - frame(WebElement)
  - defaultContent()
    - _top
- When you switch your commands act on the new <body>

“topNav”

“menu” “content” “related”

“footer”

# Frames Example

```
@Test
public void switchToFrameExample(){
    WebDriver driver =
         Driver.get("http://www.compendiumdev.co.uk/selenium/frames");

    assertEquals("Frameset Example Title (Example 6)",
                 driver.getTitle());

    driver.switchTo().frame("menu");

    driver.findElement(
          By.cssSelector("a[href='frames_example_1.html']")).click();

    String titleForExample1 = "Frameset Example Title (Example 1)";

    new WebDriverWait(driver,Driver.DEFAULT_TIMEOUT_SECONDS).
            until(ExpectedConditions.titleIs(titleForExample1));

    assertEquals(titleForExample1,driver.getTitle());
}
```

Switch to the menu frame.

If I don't switch then I'll get a NoSuchElementException

Had to Synchronise on the page change.

# Frames Exercises

- click on content.“load green page” then on “Green Page”.”Back to original page” and assert for the presence of “<h1>Content</h1>”
- click on “menu”.”iFrames Example” then “Example 5” in the iframe, then “Content”.”Load Main Frames Page”

# Frames Answers And FAQs

- Cover Frames Exercises Test
- Cover Frame FAQ Test

# Frames Exercises Refactored

- To Show ExpectedConditions static import
- To show wait being created in @Before
  - Usage as wait.until
- Wait can return the element
- To show Hamcrest matchers
  - AssertEquals to AssertThat e.g.
    - assertThat(driver.getTitle(), is("Frameset Example Title (Example 6)"));
- Mention other refactorings we could do
  - Page objects
  - Remove comments
  - Hint: Try and remove direct driver from your tests

# Windows

- Each Browser window has a unique handle
  - e.g {b3e5c07c-fac7-492f-a86d-d2fa57608185}
- driver.getWindowHandle() returns handle for current window
- driver.getWindowHandles() returns a Set<String> of all window handles
- driver.switchTo.window(String handle) switches control to the chosen window

# Window Example

```
@Test
public void switchToNewWindow(){

    WebDriver driver = Driver.get(
               "http://www.compendiumdev.co.uk/selenium/frames");

    assertEquals("Expected only 1 current window", 1,
                   driver.getWindowHandles().size());

    String framesWindowHandle = driver.getWindowHandle();

    driver.switchTo().frame("content");
    driver.findElement(By.cssSelector(
         "a[href='http://www.seleniumsimplified.com']")).click();

    assertEquals("Expected a New Window opened", 2,
                 driver.getWindowHandles().size());


    Set<String> myWindows = driver.getWindowHandles();
    String newWindowHandle="";

    for(String aHandle : myWindows){
        if(!framesWindowHandle.contentEquals(aHandle)){
            newWindowHandle = aHandle; break;
        }
    }

    driver.switchTo().window(newWindowHandle);

    assertTrue("Expected Selenium Simplified site",
               driver.getTitle().contains("Selenium Simplified"));
    }
```

We start with one window open

Remember the current window handle

Clicking on this Link opens a new window

Find the new window handle

Switch to the new window

Driver commands now act on new window

# Window Exercises

- Using http://compendiumdev.co.uk/selenium/frames/index.html
- amend the example (WindowsExampleTest) to switch back to the frame window and demonstrate that you switched control
- Two of the links on the content frame can be switched to by name, (eviltester.com and Compendium Developments) open both of them and switch between all three windows (two of them using switchTo("name"))
  - Close each window when done with it

# Window Exercises: Answers

- Switch between windows
  - Driver quit at start
  - Explain iterating around handles
  - Switch to and close so only 1 window open
- switchTobyName
  - Use Firebug to show named window
    - <a id="compdev" ...
  - driver.close(); to close the window

# Manage Current Window

- Driver.manage.window()
  - .maximise
  - .getSize
  - .setSize
    - xSize use Dimension
  - .getPosition
  - .setPosition
    - xPosition use Point

# Manage Window Example

```
@Test
public void manageWindow(){

    WebDriver driver = Driver.get(
                "http://www.compendiumdev.co.uk/selenium/frames");

    driver.manage().window().setPosition(new Point(10,20));

    Point pos = driver.manage().window().getPosition();

    assertEquals(10, pos.getX());
    assertEquals(20, pos.getY());

    driver.manage().window().setSize(new Dimension(350,400));

    Dimension winSizes = driver.manage().window().getSize();

    assertEquals(350, winSizes.getWidth());
    assertEquals(400, winSizes.getHeight());
}
```

Position moves The window around

Position uses a Point object

Position uses X & Y co-ords

Size... Resizes the browser window

Size uses Dimensions i.e width and height

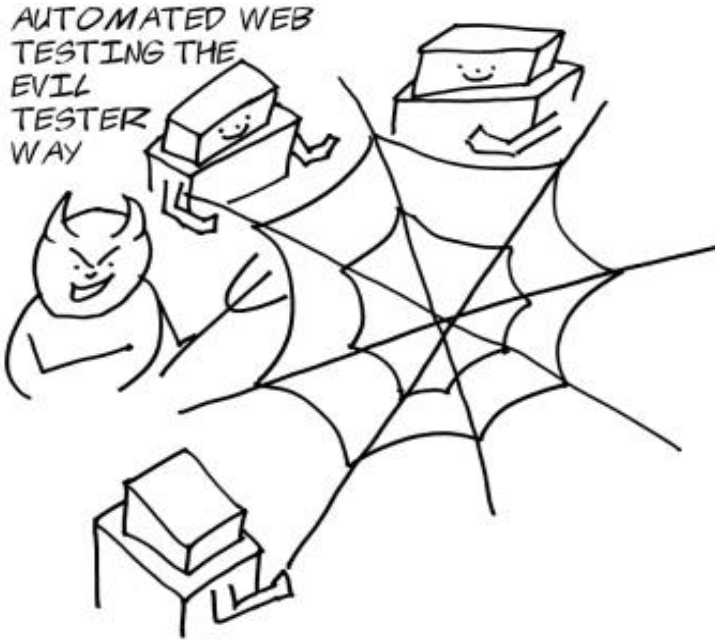
# Window Manage Exercise

- Using
  - http://www.compendiumdev.co.uk/selenium/bounce.html
- Maximise the window
- Reduce the window to half size
- Move the reduced window into the center of the screen
- Just for fun, make it smaller and bounce it around the screen

# Window Manage Exercise Debrief

- Asserting can be hard when you don't know the size of the screen

# Synchronisation Strategies



# WebDriverWait Basics

# Poor Synchronisation Leads to...

- Test intermittency
- Slow test execution
- False positives/negatives
- Moaning

# Exercise: Feel The Pain

- Write a test which,
  - on compendiumdev.co.uk/selenium/basic_ajax.html
  - Select “Server” from “combo1”
  - Select “Java” from “combo2”
  - Click “Code In It” button
  - Assert that the loaded page has "_valuelanguage_id" text of 23

If it doesn't work when you run it, then break point and step through and get it to work.

# My Failing Test

It works in Debug mode!

```
WebDriver driver;

driver = Driver.get("http://compendiumdev.co.uk/selenium/" +
                        "basic_ajax.html");

// select Server
WebElement categorySelect = driver.findElement(By.id("combo1"));

categorySelect.findElement(By.cssSelector("option[value='3']")).click();

// then select Java
WebElement languageSelect = driver.findElement(By.id("combo2"));

languageSelect.findElement(By.cssSelector("option[value='23']")).click();

// Submit the form
WebElement codeInIt = driver.findElement(By.name("submitbutton"));
codeInIt.click();

WebElement languageWeUsed = driver.findElement(By.id("_valuelanguage_id"));
assertEquals("Expected Java code", "23",languageWeUsed.getText());
```

# The Step Through Heuristic

If it works when I step through, but not when I run it, then I probably have a Synchronisation problem.

Also known as:

- It works on my machine
- It works in this browser

# Construct WebDriverWait

- We can construct a WebDriverWait with a driver and a timeout
  - `new WebDriverWait(driver,10)`
- Additionally add a 'sleep' time in milliseconds between each 'apply'
  - `new WebDriverWait(driver, 10, 50)`

# WebDriverWait

wait.until(ExpectedCondition);

- WebDriverWait takes an ExpectedCondition
- ExpectedCondition implements “apply”
- “apply” returns
  - null or Boolean (false) for false
  - An object, or Boolean (true) for true
- wait.until stops when !null or true or timeout time reached

# ExpectedConditions

- Support has some common expected conditions
- ExpectedConditions
  - .titleIs(String)
  - .titleContains(String)
  - .presenceOfElementLocated(By)
  - etc.

# Exercise: Use ExpectedConditions

- Add WebDriverWait and ExpectedConditions to create as many synchronisation points as you need to get the test passing

# Exercise Retrospective

- What ExpectedConditions did you use?
- Which Waits did you add?
- Which Waits did you need?
- Do the waits increase the execution time of the test?

# Exercise Example Answers

- I used two waits, one after clicking combo1 and one after clicking button
- I 'needed' both, but page title failure was more intermittent than the combo failure

Are there any risks with these answers?

```
new WebDriverWait(driver,10).until(
        ExpectedConditions.invisibilityOfElementLocated(
                By.id("ajaxBusy")));

new WebDriverWait(driver,10).until(
        ExpectedConditions.presenceOfElementLocated(
                By.cssSelector("option[value='23']")));

new WebDriverWait(driver,10).until(
        ExpectedConditions.visibilityOfElementLocated(
                By.cssSelector("option[value='23']")));

new WebDriverWait(driver,10).until(
        ExpectedConditions.elementToBeClickable(
                By.cssSelector("option[value='23']")));

new WebDriverWait(driver,10).until(
     ExpectedConditions.titleIs("Processed Form Details"));
```

Wait for Ajax Image to be invisible

Wait for option to be present

Wait for option to be visible

Wait for option to be clickable

This was a very intermittent failure

# Custom ExpectedConditions

# Custom ExpectedCondition

```
new WebDriverWait(driver,10).until(
        new SelectContainsText(By.id("combo2"),"Java")
);
```

- Why?
  - ExpectedConditions doesn't have what you need
  - You want to make your tests read well for your usage scenario
  - You want to pass additional values to the apply method
- ... create a Custom ExpectedCondition

# Custom ExpectedCondition Example

I made it private because
It is local to my test, normally
this would be public

You can return either
Boolean or WebElement.
I chose Boolean for this.

```
private class SelectContainsText implements ExpectedCondition<Boolean> {

    private String textToFind;
    private By findBy;

    public SelectContainsText(final By comboFindBy, final String textToFind) {
        this.findBy = comboFindBy;
        this.textToFind = textToFind;
    }

    @Override
    public Boolean apply(WebDriver webDriver) {

        WebElement combo = webDriver.findElement(this.findBy);
        List<WebElement> options = combo.findElements(By.tagName("option"));

        for(WebElement anOption : options){
            if(anOption.getText().equals(this.textToFind))
                return true;
        }

        return false;
    }
}
```

Pass in whatever you need
in the constructor

Override apply, this is
called by WebDriverWait

Implement your checking code
using the passed in WebDriver

# Adhoc Waits Created Inline Example

Create ExpectedCondition on the fly and @Override the apply method

If not Boolean then null == false

WebDriver automatically passed through to the apply method

```
new WebDriverWait(driver,10).until(

        new ExpectedCondition<Boolean>() {

            @Override
            public Boolean apply(WebDriver webDriver) {

                return webDriver.findElement(

                        By.cssSelector("option[value='23']"))
                            .isDisplayed();
            }
        }
);
```

Add your synchronisation code in the apply method

Anonymous classes cannot have custom constructors so use for simple waits

# Wrap Adhoc Wait in a method

Anonymous class wrapped in a method can use final params in the class. Bypasses need for a constructor.

```
private ExpectedCondition<WebElement>
        optionWithValueDisplayed(final String value) {

   return new ExpectedCondition<WebElement>() {

      @Override
      public WebElement apply(WebDriver webDriver) {
         return webDriver.findElement(
                  By.cssSelector("option[value='" + value + "']"));
      }

   };
}
```

Returning a WebElement for future use.

# Waiting Exercises

- Create WebDriverWait with an inline ExpectedCondition that returns a WebElement, not a Boolean, assert on the text of that WebElement
- Create a custom ExpectedCondition
  - for titleDoesNotContain(<String>)
  - use /selenium/basic_redirect.html

# Waiting Exercises Answers

# ExpectedCondition & Function

- An ExpectedCondition implements Function applying to WebDriver objects
  - Function<WebDriver, T>
  - The apply acts on a WebDriver but it can return any Object e.g. Boolean, WebElement
- WebDriverWait will also take a Predicate<WebDriver>

# ExpectedCondition Example

This really doesn't wait for anything,
it is an illustration of a principle

```
@Test
public void returnAStringFromWebDriverWaitExpectedCondition(){

    String theTime = new WebDriverWait(driver,10,100).

            until(new ExpectedCondition<String>() {

                @Override
                public String apply(WebDriver driver) {
                    return "01:01:04";
                }
            }
            );

    assertEquals("Expected a different time",
                    "01:01:04", theTime);
}
```

This ExpectedCondition
will return a String.

This could be written as
new Function<WebDriver, String>()

# ExpectedCondition As Method Example

```
@Test
public void returnAStringFromWebDriverWaitExpectedConditionAsMethod(){

    String theTime = new WebDriverWait(driver,10,100).
            until(timeHasChangedTo("01:01:04")
            );

    assertEquals("Expected a different time", "01:01:04", theTime);
}

private ExpectedCondition<String> timeHasChangedTo
                                        (final String timeToReturn) {
    return new ExpectedCondition<String>() {
        @Override
        public String apply(WebDriver driver) {
            return timeToReturn;
        }
    };
}
```

# Fluently Using WebDriverWait

# Use WebDriverWait Fluently

- Additional methods
  - .pollingEvery
  - .ignoring
  - .withTimeout
  - .withMessage
  - .ignoreAll
- Allow us to create WebDriverWait fluently

# Use WebDriverWait Fluently

```
new WebDriverWait(driver, 1).

    pollingEvery(100, TimeUnit.MILLISECONDS).

    ignoring(IllegalStateException.class).

    withTimeout(5, TimeUnit.SECONDS).

    withMessage("will wait 5 seconds").

     until(
        new ExpectedCondition<Boolean>() {
            @Override
            public Boolean apply(WebDriver webDriver) {
                throw new IllegalStateException();
            }
        }
     );
```

Override polling time

Ignore Exceptions

Override timeout

Include text in Timeout Message

# FluentWait

# FluentWait

- WebDriverWait extends FluentWait so everything you have seen is really FluentWait
- Main difference between FluentWait and WebDriverWait:
  - FluentWait until can apply to anything
    - e.g. WebElement, String, Time, Files etc.
    - and can return anything
  - WebDriverWait until applies to WebDriver
    - and can return anything

# FluentWait with WebElement Example

Declare that a WebElement will be passed to the wait

```
countdown = driver.findElement(
                    By.id("javascript_countdown_time"));

 new FluentWait<WebElement>(countdown).

         withTimeout(10, TimeUnit.SECONDS).
         pollingEvery(100,TimeUnit.MILLISECONDS).

         until(new Function<WebElement, Boolean>() {

             @Override
             public Boolean apply(WebElement element) {
                 return element.getText().endsWith("04");
             }
         }
         );
```

Pass in WebElement

Configure wait fluently

Declare returning a Boolean

# FluentWait Exercise

- Using javascript_countdown.html
  - Create a fluent wait that returns a String with the time when the last two chars are “04”
    - Assert the returned time is 01:01:04
  - Do the same with a WebDriverWait

# Implicit or Explicit

# Waiting Approaches

- Explicit wait
  - WebDriverWait
    - ExpectedConditions
    - ExpectedCondition (Class or inline)
  - FluentWait
- Implicit wait
  - .findElement has an implicit wait
  - Default is 0
  - Can amend the global default

If an element can not be found then a findElement would take 10 seconds before throwing an Exception.

```
driver.manage().timeouts()
            .implicitlyWait(10, TimeUnit.SECONDS);
```

# Implicit or Explicit?

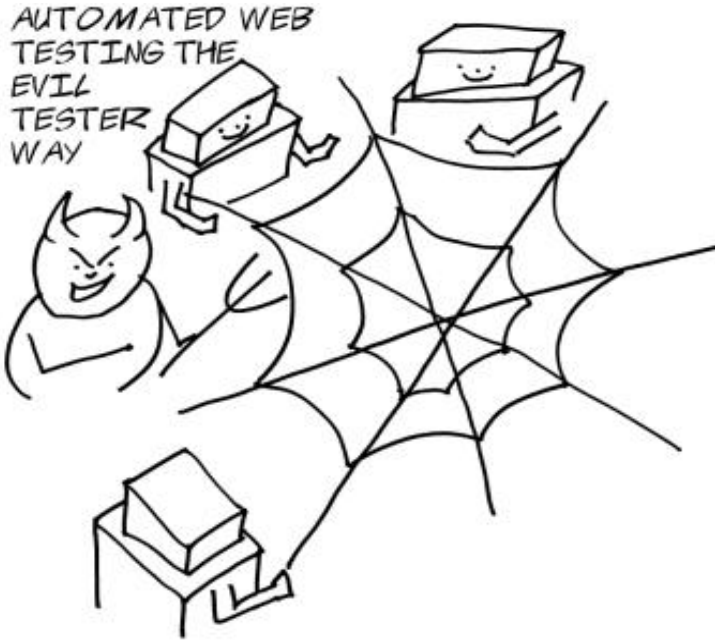
- Implicit can make initial tests faster to write
  - you don't worry about synchronisation when writing tests
- It can be harder to add synchronisation later
  - You have to identify a source of intermittency
- If you start with implicit then you can expose synchronisation problems by gradually reducing the implicit wait time

# Refactoring WebDriverWait

# To make “waits” readable

- Wrap an Inline wait in a method
  - e.g. ExpectedConditions
- Wrap an ExpectedCondition class in a method
- Override toString
- Create a 'wait' variable
  - assign new WebDriverWait(...) in @Before
- Import ExpectedConditions statically
  - Refactor to static import
- Wait for a 'thing' then use it

# Cookies



# Cookies

- Inspect
  - driver.manage
    - .getCookies()
    - .getCookieNamed(“name”)
- Interact
  - driver.manage
    - .addCookie(Cookie)
    - .deleteAllCookies
    - .deleteCookie(Cookie)
    - .deleteCookieNamed(“name”)

# Cookies Example

```
@Test
public void visitSearchPageAndCheckNoLastSearchCookie(){

    WebDriver driver;

     driver = Driver.get("http://compendiumdev.co.uk/selenium/search.php");


    driver.manage().deleteAllCookies();

    driver.navigate().refresh();

    Cookie aCookie = driver.manage().getCookieNamed("SeleniumSimplifiedLastSearch");

    assertEquals("Should be no last search cookie", null, aCookie);
}
```

Delete all cookies for current domain

Refresh page to get an initial set of cookies

Find a named cookie
Return null if not found

# Cookie.Builder

- Can use constructor
  - new Cookie
- Or can use Cookie.Builder

```
Cookie aNewCookie = new Cookie.Builder(
                                aCookie.getName(),
                                String.valueOf(2)).
                            domain(aCookie.getDomain()).
                            path( aCookie.getPath()).
                            expiresOn(aCookie.getExpiry()).
                            isSecure(aCookie.isSecure()).
                            build();
```

# Cookies Exercises

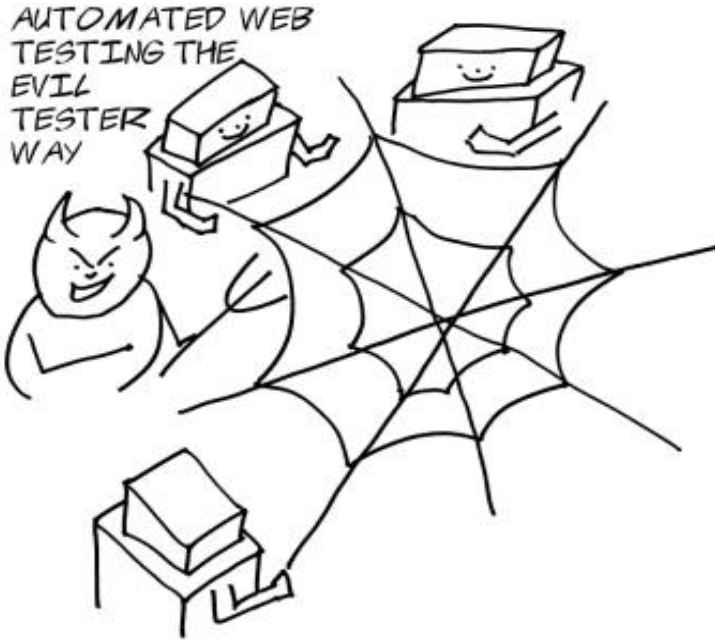
- http://compendiumdev.co.uk/selenium/search.php
- Uses the cookies:
  - seleniumSimplifiedSearchLastVisit
  - seleniumSimplifiedSearchNumVisits
  - SeleniumSimplifiedLastSearch
- Write tests to
  - Search and check number of visits == 1
  - Set visits cookie to 42, check next search is 43
    - By cloning a cookie
    - And creating a cookie from scratch

Remember to experiment with Cookie Builder and just the constructor

# Cookies Answers Debrief

- Creating cookies by scratch requires analysing the application to see what it needs
  - Cross browser issues if you get it wrong

# JavaScript



# Why use Javascript?

- Workarounds
- Custom synchronisation
- Make the app more testable
  - Adjust hidden fields
  - Amend values
- Simulate hard to reach conditions
- Test Flash & HTML 5

# Javascript Execution

- Cast WebDriver to JavascriptExecutor
  - .executeScript(script, args...)
  - .executeAsyncScript(script, args...)
- Arguments are accessed using
  - arguments[index] e.g. "document.title=arguments[0]"
- Return values are converted to Java types
  - Html Element = WebElement, decimal = Double, non-decimal = Long, boolean = Boolean, array = List<Object>, else String or null
- Runs in an anonymous function

# (JavascriptExecutor) Example

```
@Test
public void callAJavaScriptFunctionOnThePage(){

    WebDriver driver = Driver.get(
            "http://www.compendiumdev.co.uk/selenium/canvas_basic.html");


    JavascriptExecutor js =(JavascriptExecutor)driver;


    int actionsCount = driver.findElements(
                         By.cssSelector("#commandlist li")).size();
    assertEquals("By default app has 2 actions listed",
                 2, actionsCount);

    js.executeScript("draw(1, 150,150,40, '#FF1C0A');");

    actionsCount = driver.findElements(
                      By.cssSelector("#commandlist li")).size();
    assertEquals("Calling draw should add an action",
                  3, actionsCount);
}
```

Test page is at compendiumdev.co.uk/selenium/canvas_basic.html

Cast driver to JavascriptExecutor to access the JavaScript methods

Execute the 'draw' Function in the page

# Explore code to learn

- How do we know to cast to JavascriptDriver
  - Examine WebDriver, Examine Driver
- Because it is on Driver not WebDriver there is a risk that the driver may not implement it
  - But they all tend to extend RemoteWebDriver so changes are slim that they don't

# JavascriptExecutor Exercises

- Call the draw function with arguments from the .executeScript method & assert it drew e.g.
  - .executeScript(“document.title=arguments[0]”, “bob”);
- Execute a script which adds two arguments & assert the result
  - “return 10;” will return 10
- Pass a WebElement as an argument, and use Jquery to 'hide' it, then assert that it is hidden
  - e.g to hide body using JQuery: $('body').hide();
- Create a test which adds Javascript to the page, demonstrate that it persists beyond the anonymous function execution

# Javascript Exercise Answers

# Execute Async JavaScript

- When executeAsyncScript is called, WebDriver adds an additional final argument, a callback function to signal that async execution has finished
  - "var callback = arguments[arguments.length – 1];"
- Any argument you pass to the callback function will be returned to WebDriver
  - HTML Element == WebElement, number == Long etc.
- Call it expecting an Object and cast appropriately
- SetScriptTimeout
  - driver.manage().timeouts().setScriptTimeout(10, TimeUnit.SECONDS);

# Execute Async example

- Http://compendiumdev.co.uk/selenium/basic_ajax.html
- Will use executeScript to extend jquery hide
  - To call a callback when something is hidden
- Then the executeAsyncScript will make our webdriver callback the callback used in JQUery
- *Look at help docs to see other example test usages*

# Execute Async Example

```
@Test
public void waitInBrowserForTimeSample(){
    driver.manage().timeouts().setScriptTimeout(10, TimeUnit.SECONDS);

    long start = System.currentTimeMillis();
        ((JavascriptExecutor) driver).executeAsyncScript(
            "window.setTimeout(arguments[arguments.length - 1], 500);");

    System.out.println(
            "Elapsed time: " + (System.currentTimeMillis() - start));

    assertTrue("Elapsed time should be greater than 500 milli",
            (System.currentTimeMillis() - start) > 500);

}
```

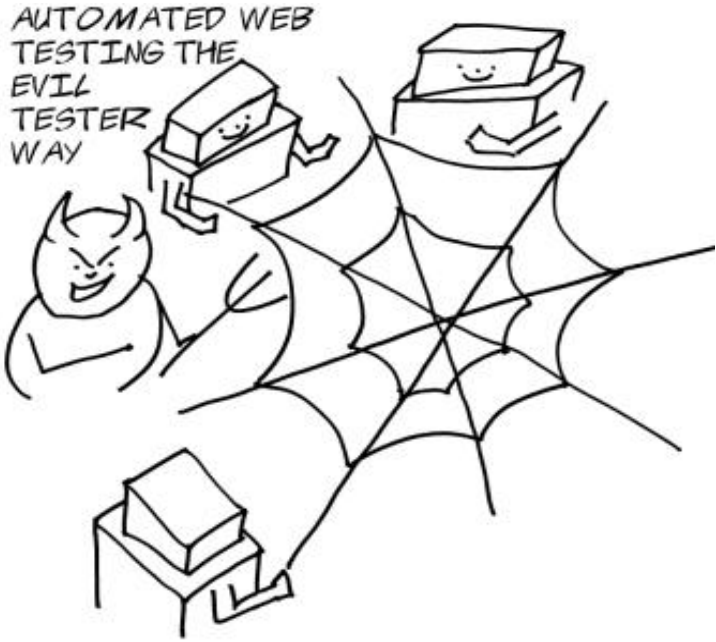
# Execute Async Exercise

- Look at the JavaDoc help for executeAsyncScript
  - Create a test for the sleep sample code
  - Use the XMLHttpRequest sample to call /selenium/ajaxselect.php which is the server method which responds to the Ajax call on the app. You will need to add an id parameter to the call e.g. id=2

# Execute Async Exercise Debrief

- Did you remember to add setScriptTimeout?
- Any syntax errors in the javadoc code?

# Introducing Different Browsers



# Browsers Overview

- Firefox Driver – currently built in
- HtmlUnit Driver – currently built in
- ChromeDriver – separate .exe
- OperaDriver – separate .exe available through maven
- IEDriver – seperate .exe
- RemoteDriver – currently built in, requires a server to connect to e.g. saucelabs or grid
- Various mobile drivers and safari driver

# Firefox Driver

- Currently part of deployed Jars
  - Effectively 'built in'
  - Easy to get started with
  - Can be slow to startup (at least for me)

```
aDriver = new FirefoxDriver();
```

# Firefox Driver Profile

- Driver Profiles allow us to initialise a driver with additional capabilities and specific configuration

```
FirefoxProfile profile = new FirefoxProfile();
WebDriver driver = new FirefoxDriver(profile);
```

- Profile level methods

```
profile.setEnableNativeEvents(true);
```

- Set browser preferences

```
profile.setPreference("extensions.firebug.currentVersion", "1.6.2");
```

- Load extensions

```
profile.addExtension(new File(extensionPath));
```

# FirefoxDriver Useful Links

- Firefox Preferences
  - about:config
  - http://www.timeatlas.com/5_minute_tips/general/introduction_to_firefox_preferences#.UIvbL4az728
- Firefox Driver Documentation
  - http://code.google.com/p/selenium/wiki/FirefoxDriver

# Browser Capabilities

- Generic browser control mechanism
- e.g. Set Proxies

```
org.openqa.selenium.Proxy proxy = new org.openqa.selenium.Proxy();
        proxy.setHttpProxy("localhost:8080")
             .setFtpProxy("localhost:8080")
             .setSslProxy("localhost:8080");

        DesiredCapabilities capabilities = new DesiredCapabilities();

        capabilities.setCapability(CapabilityType.PROXY, proxy);
```

# HtmlUnitDriver

- http://htmlunit.sourceforge.net/
- DesiredCapabilities.setJavascriptEnabled seems to have been added for HTMLUnit

# ChromeDriver

- http://code.google.com/p/selenium/wiki/ChromeDriver
- Download the driver server
  - set "webdriver.chrome.driver" to the location
- Command line switches
  - http://peter.sh/experiments/chromium-command-line-switches/
  - Pass in via options.addArguments
- ChromeDriver.log is useful debugging tool

# ChromeDriver Example

Calculate the location of Chromedriver relative to my test code

```
String currentDir = System.getProperty("user.dir");

String chromeDriverLocation =
   currentDir + "/../tools/chromedriver/chromedriver.exe";

System.setProperty("webdriver.chrome.driver",
                   chromeDriverLocation);

ChromeOptions options = new ChromeOptions();
options.addArguments("disable-plugins");
options.addArguments("disable-extensions");

aDriver = new ChromeDriver(options);
```

Set the property so that the driver can be found

Disable the plugins and extensions to prevent them interfering with the tests

Get the driver

# Opera Driver

- Documentation and Downloads
  - http://code.google.com/p/selenium/wiki/OperaDriver
  - https://github.com/operasoftware/operadriver
  - http://mvnrepository.com/artifact/com.opera/operadriver
- Add to maven

```
<dependency>
    <groupId>com.opera</groupId>
    <artifactId>operadriver</artifactId>
    <version>1.1</version>
</dependency>
```

- Issues? then check for an up to date driver

# Config Opera Driver

- Capabilities
- OperaProfile
  - opera:config
  - http://www.opera.com/support/usingopera/operaini

```
OperaProfile profile = new OperaProfile();

// switching off Javascript will cause the opera driver to fail
profile.preferences().set("Extensions", "Scripting", 0);

WebDriver opera = new OperaDriver(profile);
```

# IE Driver

- http://code.google.com/p/selenium/wiki/InternetExplorerDriver
- Download the server executable
  - http://code.google.com/p/selenium/downloads/list
  - Set “webdriver.ie.driver” to the location of the driver executable

```
String currentUserDir = System.getProperty("user.dir");
String IEDriverLocation = currentUserDir +
                "/../tools/iedriver_32/IEDriverServer.exe";

System.setProperty("webdriver.ie.driver", IEDriverLocation);

aDriver = new InternetExplorerDriver();
```

# IE Driver on Path

- Drivers change
- IE Driver used to be add to path
- This method will be deprecated in favour of properties
- Pointing this out because we have to learn to read the code and the log messages output by Selenium

# Try Different Browsers

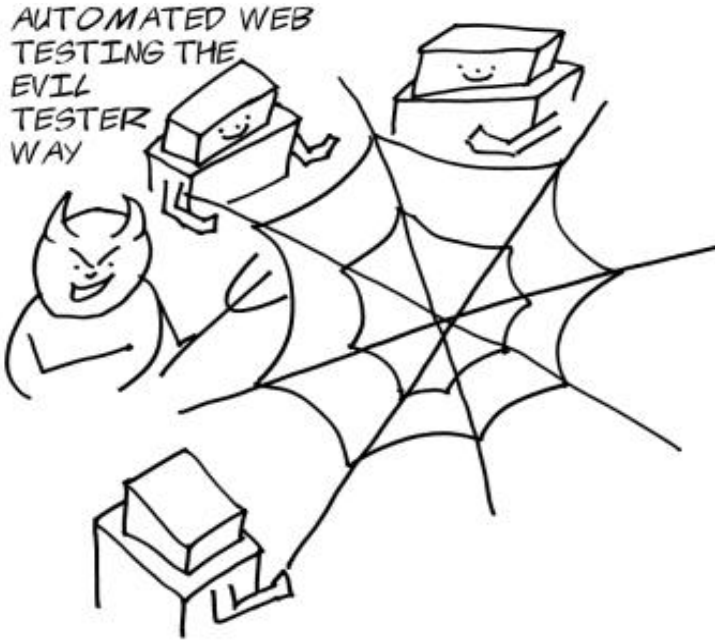
- Some tests will fail due to:
  - Driver issues
    - Bugs in the drivers
  - Driver differences
    - different exceptions thrown,
    - new exceptions thrown
  - Incompatible driver/browser versions
  - Driver scope
    - Some functionality not implemented
  - Documentation differences
    - Look at log output or read source

… etc.

# Different Browsers Exercises

- Convert some of the other slides to use different browsers.
- As you do this, think of what approaches you could have for controlling which browser your tests use – this will prepare you for our discussion on Driver Manager and Supplier Classes

# RemoteWebDriver



# Remote Driver

- When server is running on another machine
- e.g. SauceLabs.com

Remote driver configured by capabilities

```
DesiredCapabilities capabilities;

   capabilities = DesiredCapabilities.firefox();
   capabilities.setCapability("version", "5");
   capabilities.setCapability("platform", Platform.XP);

try {
      String sauceURL = System.getenv("SAUCELABS_URL");
      aDriver = new RemoteWebDriver(
                        new URL(sauceURL),
                        capabilities);

} catch (MalformedURLException e) {
   e.printStackTrace();
}
```

Watch out for UnsupportedCommandException during your tests

# RemoteWebDriver Profiles

- Can set firefox profiles on RemoteWebDriver

```
FirefoxProfile fp = new FirefoxProfile();
// set something on the profile...
DesiredCapabilities dc = DesiredCapabilities.firefox();
dc.setCapability(FirefoxDriver.PROFILE, fp);
WebDriver driver = new RemoteWebDriver(dc);
```

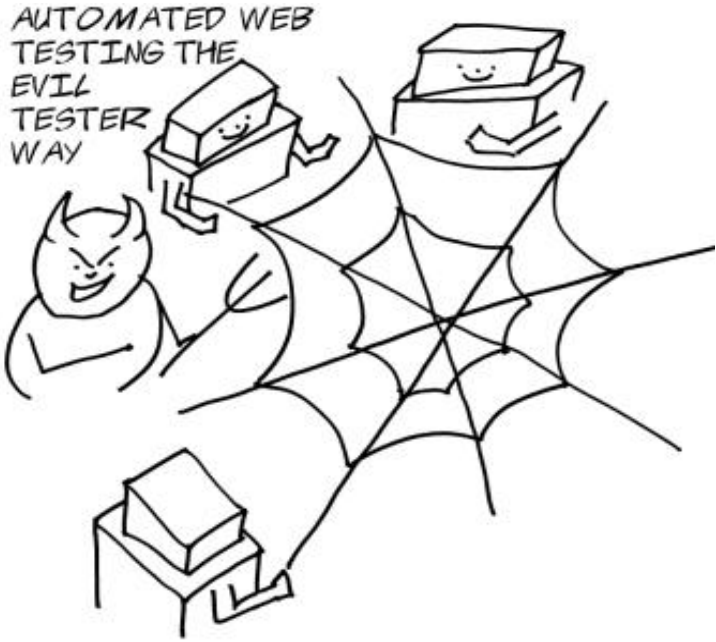
- Can set Chrome options on RemoteWebDriver

```
ChromeOptions options = new ChromeOptions();
// set some options
DesiredCapabilities dc = DesiredCapabilities.chrome();
dc.setCapability(ChromeOptions.CAPABILITY, options);
WebDriver driver = new RemoteWebDriver(dc);
```

# Saucelabs

- saucelabs.com
- Sign up for free
- Free plan good enough for this course, and a lot of things
- Records video
- Manual Testing

# Selenium Grid



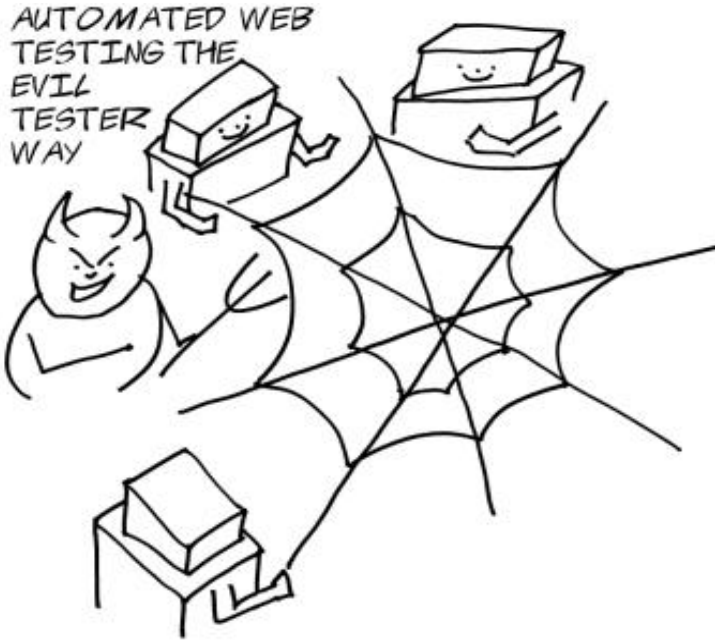
# Selenium Grid

- http://code.google.com/p/selenium/wiki/Grid2
- Build into the selenium-server-standalone .jar
- -role hub
  - -role node (register nodes to the hub)
- Configure via command line or **json file**

# Selenium Grid Exercise

- Run tests against localhost
  - Download the most recent selenium-server-standalone-x.xx.x.jar
  - Start up a localhost hub and a localhost node
  - Run your tests against it
- Create a node on a separate machine e.g. a virtual machine and run a test on that through a localhost hub

# Driver Managers and Suppliers - Approaches



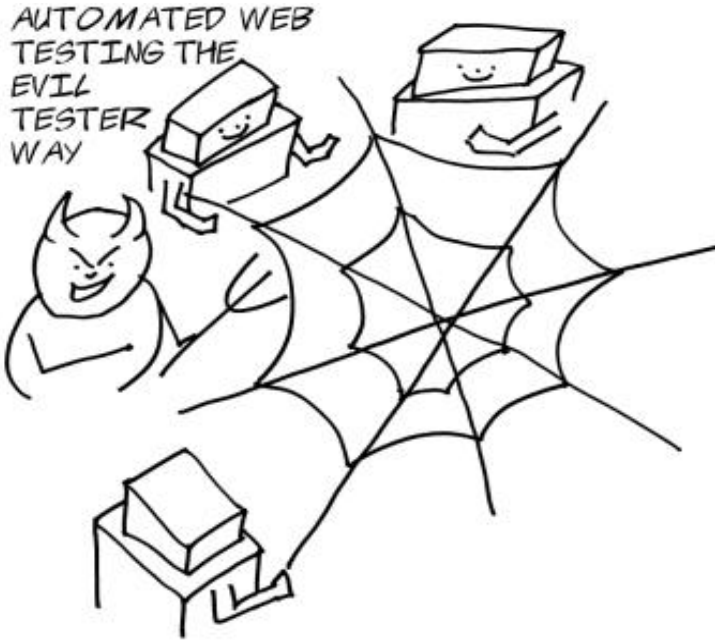
# Driver Manager Classes

- Typically we create different Driver Manager Classes
  - Manager / Supplier / Provider / Factory
- Created one for Selenium Simplified
- Created one for this course
- Example in Selenium Test code
  - https://code.google.com/p/selenium/source/browse/trunk/java/client/test/org/openqa/selenium/testing/drivers/WebDriverBuilder.java

# Driver Manager Exercises

- Create a simple Driver class which when you call get() returns an instantiation of WebDriver based on a system property selenium2Basics.driver

# Taking Screenshots



# Taking Screenshots

- Cast WebDriver to TakesScreenshot
  - getScreenshotAs(OutputType...)
    - OutputType.FILE
    - OutputType.BASE64
    - OutputType.BYTES

Creates a temp file

Good for remote driver usage

- Check for capability
  - `if(((HasCapabilities)driver).getCapabilities().is(CapabilityType.TAKES_SCREENSHOT)){...}`
- Cast with Exception
  - 
```
try{
   TakesScreenshot snapper = (TakesScreenshot)driver;
}catch(ClassCastException e){...}
```

# Takes Screenshot Example

```
@Test
public void gotoPage() throws IOException {

    WebDriver driver = new FirefoxDriver();
    driver.get("http://seleniumsimplified.com");

    TakesScreenshot snapper = (TakesScreenshot)driver;
    File tempScreenshot = snapper.getScreenshotAs(OutputType.FILE);

    System.out.println(tempScreenshot.getAbsolutePath());

    File myScreenshotDirectory = new File("C:\\temp\\screenshots\\");
    myScreenshotDirectory.mkdirs();

    File myScreenshot =
        new File(myScreenshotDirectory, "gotoPageScreen.png");

    FileUtils.moveFile(tempScreenshot, myScreenshot);
    assertThat(myScreenshot.length(), is(greaterThan(0L)));

    driver.get("file://" + myScreenshot.getAbsolutePath());

}
```

Cast Driver to TakesScreenshot

Create screenshot in temporary file

Create folder to store screenshot

Setup location to store screenshot

Move and rename temp file

Display file in browser for a manual check

# TakesScreenshot Exercises 1

- Can we take a screenshot?
  - Create a firefox driver
    - use getCapabilities to check that it can create a screenshot
    - Cast directly to snapper and catch any ClassCastException
  - Create an HtmlUnitDriver
    - use getCapabilities to check that it can not create a screenshot
    - Cast directly to snapper and catch any ClassCastException

# TakesScreenshot Exercises 2

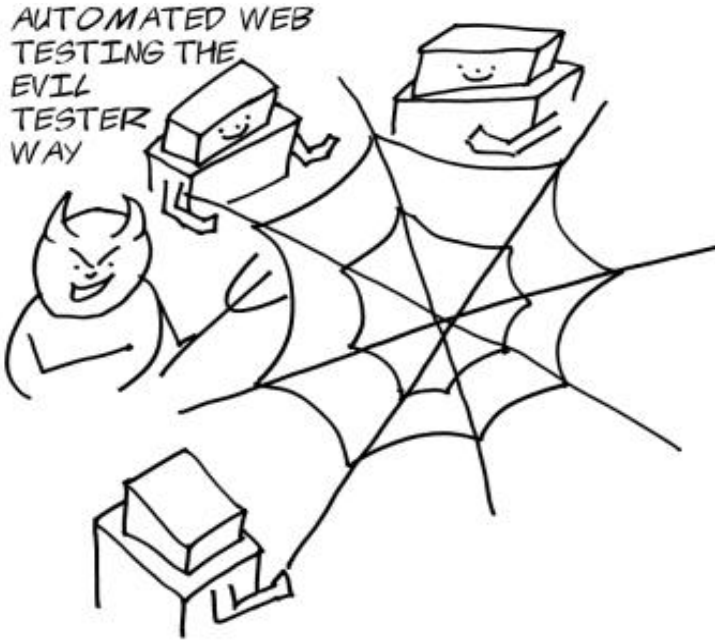
- Ensure driver has the capability to take a screenshot then
  - Save screenshot to file with a random name
  - Save BASE64 string to file with random name
  - Save BYTE to a file with random name

Hint: Base64 object can decode before writing to a FileOutputStream

Hint: FileOutputStream can write a byte[]

# TakesScreenshot Exercise Debrief

# Page Objects



# Page Objects?

- An abstraction around pages
  - Or page components
- Remove driver.xxxx from tests
- Model the physical domain
- Numerous ways of doing this
  - Page Factory
  - Extends LoadableComponent
  - Basic objects
- Modelling Decisions Required

# Refactor to Page Object Heuristics

- Convert local abstractions to Page Abstractions
- Replace comments with abstractions
- Make Decisions about the model based on the test readability and method reuse
- Build only what you need as you need it

# Page Object Heuristics

- Construct the page object with a WebDriver
- Limit the methods to physical actions
- Expose Constants as public static final
- Don't navigate to the page in the constructor, either use a navigation object or a get() method

# Example: refactor summary

- SynchronisationWithWebDriverWaitTest
- Local Methods become page objects
  - 2 pages: BasicAjaxPage, FormProcessorPage
- Move synchronisation code on ajax into page
  - As an expected condition
  - And in the SelectCategory method
- Created an enum for Category
- Moved title synchronisation into page object
- Tests now easy to expand
- TODO: fix failing test, language Enum

# Exercise Using Initial Page Objects

- Using refactorExampleOne package
- Investigate and fix the failing javascript test
  - If you get stuck doing this then @ignore it until you see the answer
- Refactor Language into an enum,
  - implement all Languages in the enum

# Exercise Debrief

- Application Bug: Javascript on Ajax call has different value than first loading
  - Fixed by selected Server then Web
  - Alternative fix – change Enum to initial value and detect on change
- Multiple CPP in language – handled by enum naming

# LoadableComponent

- Extend to create page objects
  - Extends LoadableComponent<PageObject>
    - e.g. LoadableComponent<BasicAjaxPageObject>
  - Implement
    - Load
      - Do the work to load the page
    - IsLoaded
      - Throw an error if component/page is not loaded
  - Receive “get” for free
  - Good for when you want to call get and do nothing if already loaded

# LoadableComponent Example

- Add
  - `extends LoadableComponent<BasicAjaxPageObject>`
  - `extends LoadableComponent<ProcessedFormPage>`
    - But is ProcessedFormPage a loadable component?
- Implement methods
- Remove get from BasicAjaxPageObject because it now comes from LoadableComponent

# SlowLoadableComponent

- For those instances where you may need to wait for a component to load
- Get will now poll the isLoaded for a specified time

# SlowLoadableCompnent

- Public interface

```
extends SlowLoadableComponent<PageObjectClassName>
```

  - get()
    - loads and waits for component to be available
- Call super constructor in constructor

```
public PageObjectClassName(WebDriver driver) {
    super(new SystemClock(), 10);
    this.driver = driver;
}
```

- Implement load and isLoaded
  - I often leave 'load' empty because I navigate to the page
  - isLoaded throws a new Error if not loaded

# Page Object Models 2.0 PageFactory

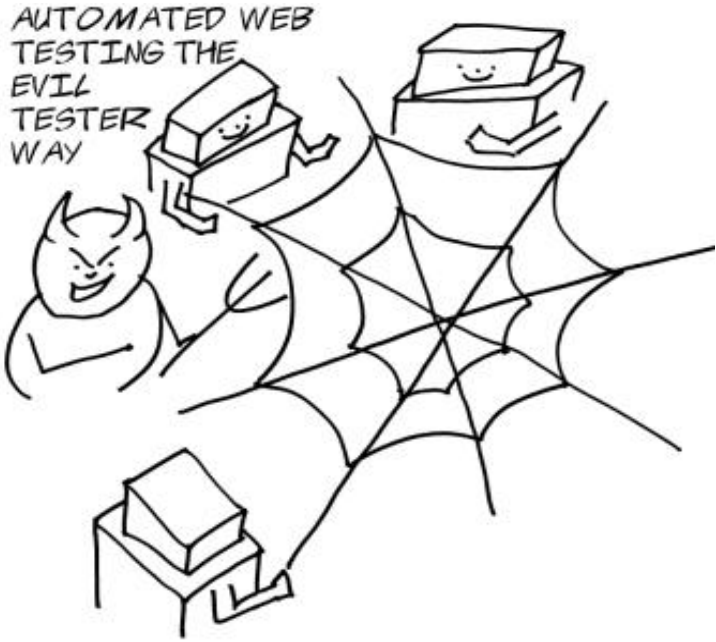
- Annotate WebElements with @FindBy

```
@FindBy(how= How.ID, using="combo1")
```

- Initialise the annotated WebElement(s) using a PageFactory

```
public BasicAjaxPageObject(WebDriver webDriver) {
    driver = webDriver;
    PageFactory.initElements(driver, this);
}
```

# Summary Slides



# WebDriver Summary Sheet

**WebDriver Interact**

```
= new
<driverClass>()
.close()
 .quit()
```

## Navigate

**WebDriver**

```
.get("URL")
.navigate
 .to(Url)
 .to("URL")
 .back()
 .forward()
 .refresh()
```

## Inspect

**WebDriver**

```
.getTitle()
.
getCurrentUrl()
.
getPageSource()
```

**WebElement**

```
.getText()
.
getAttribute("name"
)
.getTagName()
.isEnabled()
.isSelected()
.isDisplayed()
.getSize()
.getLocation()
.getCssValue()
```

**Finding elements**

**WebDriver**

```
WebElement =
  .findElement(BY)
List<WebElement> =
  .findElements(BY)
```

**By**

```
 .id("an_id")
 .xpath("xpath")
 .cssSelector("css")
 .className("class")
 .linkText("text")
 .name("name"
 .tagName("a_tag")

.partialLinkText("t")
;
```

**Support**

```
ByChaining(By, By)
ByIdOrName("idName")
```

## Interact

**WebElement**

```
.click()
.submit()
.clear()
.sendKeys(String)
.sendKeys(Keys.x)
```

**SwitchTo.**

```
.alert()
   .getText()
   .accept()
   .dismiss()

.sendKeys(String)
.frame(...)
```

**support.ui.Select**

```
.<methods>()
```

**Cookies**

**WebDriver**

```
   .manage()
      .deleteAllCookies()
      .addCookie(Cookie)
      .deleteCookie(Cookie)
   .deleteCookieNamed(String)
```

**Actions**

```
.keyUp() etc.
.perform()
```

**(JavascriptExecutor)**

```
.executeScript
```

## Synchronise

**WebDriverWait**

```
(driver, timeout in
Seconds)
.until(ExpectedCondition)
```

**ExpectedConditions**

```
.titleIs(String)
… a lot of helper methods
```

**Cookies**

**WebDriver**

```
   .manage()
      .getCookieNamed(String)
      .getCookies()
```

# Selectors

**XPATH Selectors**

```
// - match anywhere
/ - match from root
//* - any element
//tag
//*[@attribute]
//*[@attribute="value"]
//tag[@attribute="value"]
//tag1/tag2       (child)
//tag1//tag2      (any descendant)
//tag1/../..      (.. to go up)
//*[.='element textl]
```

**[@at1="a" and @at2="b"]**

and, or, =, !=, <, >, >=, <-

**Functions**

```
contains(@a,"alu")
starts-with(@a,"v")
ends-with(@a,"e")
```

**Indexing**

//tag[1]

**CSS Selectors**

```
* - any
#id
.class
tag
[attribute]
[attribute="value"]
tag[attribute="value"]
tag[attr1='val1'][attr2='val2']
tag[att1='val1'], orThisTag
```

**= (' or ")**

```
*="anywhere in value"
^="start of value"
$="end of value"
~="spaceSeperatedValue"
```

**Paths**

```
A > B (child)
A B (any descendant)
A + B (A's B sibling)
tag:first-child
```

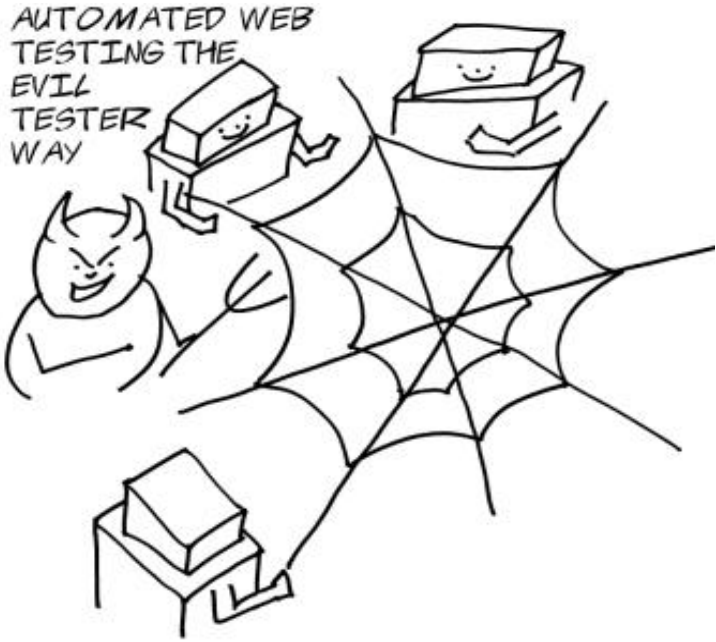
CSS References

- http://reference.sitepoint.com/css/selectorref
- http://net.tutsplus.com/tutorials/html-css-techniques/the-30-css-selectors-you-must-memorize/
- http://www.quirksmode.org/css/contents.html
- http://www.w3schools.com/cssref/css_selectors.asp

XPath References

- http://www.w3schools.com/xpath/
- http://www.simple-talk.com/dotnet/.net-framework/xpath,-css,-dom-and-selenium-the-rosetta-stone/

# End Notes



# End Points

- You will encounter problems we haven't covered here
  - e.g. use of xpaths – I had to convert to css selectors for one site I was testing
- You ***will*** need to download and read the Selenium 2.0 source code as you test

# This is just the start

- Explore the API in your IDE.
- Look in the code.
- Ask yourself “what is that for?”

# Basic Practice Pages

- http://compendiumdev.co.uk/selenium/
  - alert.html
  - basic_ajax.html
  - basic_web_page.html
  - basic_html_form.html
  - ajaxselect.php
  - calculate.php
  - find_by_playground.php
  - refresh.php
  - search.php

- Source available at https://bitbucket.org/ajrichardson/seleniumtestpages

# Advanced Practice Pages

- http://compendiumdev.co.uk/selenium/
  - Showcase/Showcase.html
- Source available at https://bitbucket.org/ajrichardson/simplegwtshowcaseclone

# Learn More Java

- Agile Java by Robert Martin
- Effective Java by Joshua Bloch
- Implementation Patterns by Kent Beck
- Growing Object-Oriented Software, Guided by Tests by Steve Freeman and Nat Pryce

# Live pages to challenge yourself

gwt.google.com/samples/Showcase/Showcase.html

- Try the above it offers many challenges

www.primefaces.org/showcase/ui/home.jsf

# Alan Richardson

- www.eviltester.com
- www.compendiumdev.co.uk
- www.seleniumsimplified.com
- @eviltester
- alan@compendiumdev.co.uk