Hands On WebDriver: Training One Day

Alan Richardson

@eviltester

http://www.compendiumdev.co.uk/contact

www.SeleniumSimplified.com
www.EvilTester.com
www.CompendiumDev.co.uk
www.JavaForTesters.com

Materials & Code etc.

- http://unow.be/at/se2014
  - Main Slides
  - Exercise Slides
  - Link to Source Code Repo
  - Link to test pages

+ Discount on Online Selenium 2 WebDriver with Java Training course – discount ends 30th June

Alan Richardson

uk.linkedin.com/in/eviltester

Independent Test Consultant
& Custom Training

Contact Alan

http://compendiumdev.co.uk/contact

Blogs and Websites

- CompendiumDev.co.uk
- SeleniumSimplified.com
- EvilTester.com
- JavaForTesters.com
- Twitter: **@eviltester**

Online Training Courses

- Technical Web Testing 101
  Unow.be/at/techwebtest101
- Intro to Selenium
  Unow.be/at/startwebdriver
- Selenium 2 WebDriver API
  Unow.be/at/webdriverapi

Videos

youtube.com/user/EviltesterVideos

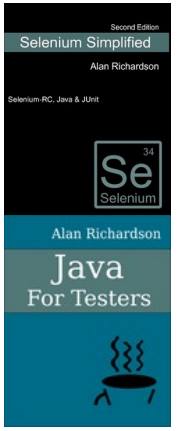
Books

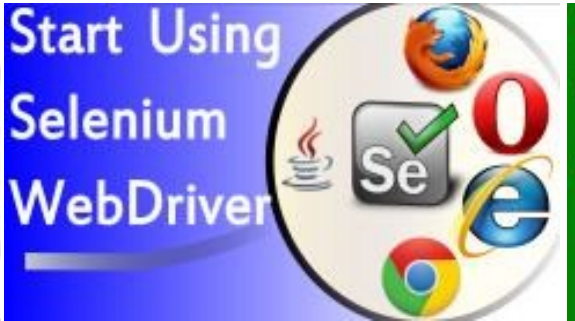
Selenium Simplified

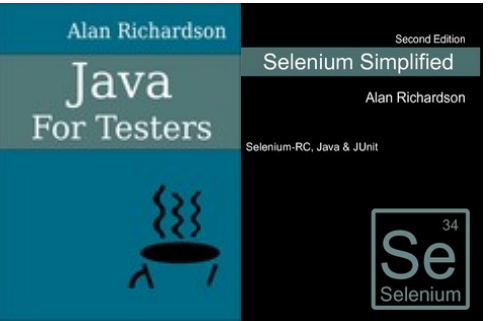
Unow.be/rc/selsimp

Java For Testers

leanpub.com/javaForTesters







Overview

- Hints and Tips
- Installation
- Basic WebDriver API capabilities
- Simple interrogation and navigation
- Synchronization strategies
- AJAX applications.
- Tools
- Location strategies using CSS and XPath.
- Introduction to abstraction approaches

Approach

- Presentation
- Demos
- Exercises
  - Multiple Tracks depending on your level

Status Check

- Java Experience?
- Automation Experience?
- Selenium Experience?
- WebDriver Experience?

Introduction & Overview

Ancient History

- Selenium 1.0
  - Javascript through a proxy
  - Flat API “selenium.”
  - Server based

- WebDriver
  - Object Oriented API, Interface which each driver implements
  - Driver calls browser directly
  - Has a server component Remote WebDriver

Installation

- Java SDK
- Maven
- IntelliJ IDE
- Firefox
- FirePath plugin

seleniumsimplified.com/get-started/

What is Maven?

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

pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.hamcrest</groupId>
        <artifactId>hamcrest-all</artifactId>
        <version>1.3</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-server</artifactId>
        <version>2.40.0</version>
    </dependency>
</dependencies>
```

Java Language Levels

- Set in IntelliJ Project Settings
- And in the pom.xml

```
<!-- I have added the build section to support importing into
 IntelliJ automatically without throwing errors about wrong Java
 Version. This basically says the source requires at least Java
1.7
 and use a compiler that outputs Java 1.7 -->
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.1</version>
            <configuration>
                <source>1.7</source>
                <target>1.7</target>
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Benefits of Maven

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

MyFirstTest.java

```
package com.eviltester.webdriver;

import org.junit.Assert;
import org.junit.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class MyFirstTest {

 @Test
 public void startWebDriver(){

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

   Assert.assertTrue("Expected Selenium Simplified",
                     driver.getTitle().
                     startsWith("Selenium Simplified"));

   driver.close();
   driver.quit();
 }
}
```

Recommended Applications For Exercises Overview

- Beginner: Selenium Test Pages
  - https://github.com/eviltester/seleniumTestPages
  - http://seleniumsimplified.com/testpages/
- Intermediate:
  - http://todomvc.com/
- Advanced
  - http://google-gruyere.appspot.com/
  - http://demo.redmine.org/

Source Code For Examples etc.

- https://xp-dev.com/
  - svn/HandsOnWebDriverOneDayCourseJava/trunk
- http://compendiumdev.co.uk/
  - files/conferences/se2014/handson20140504.zip

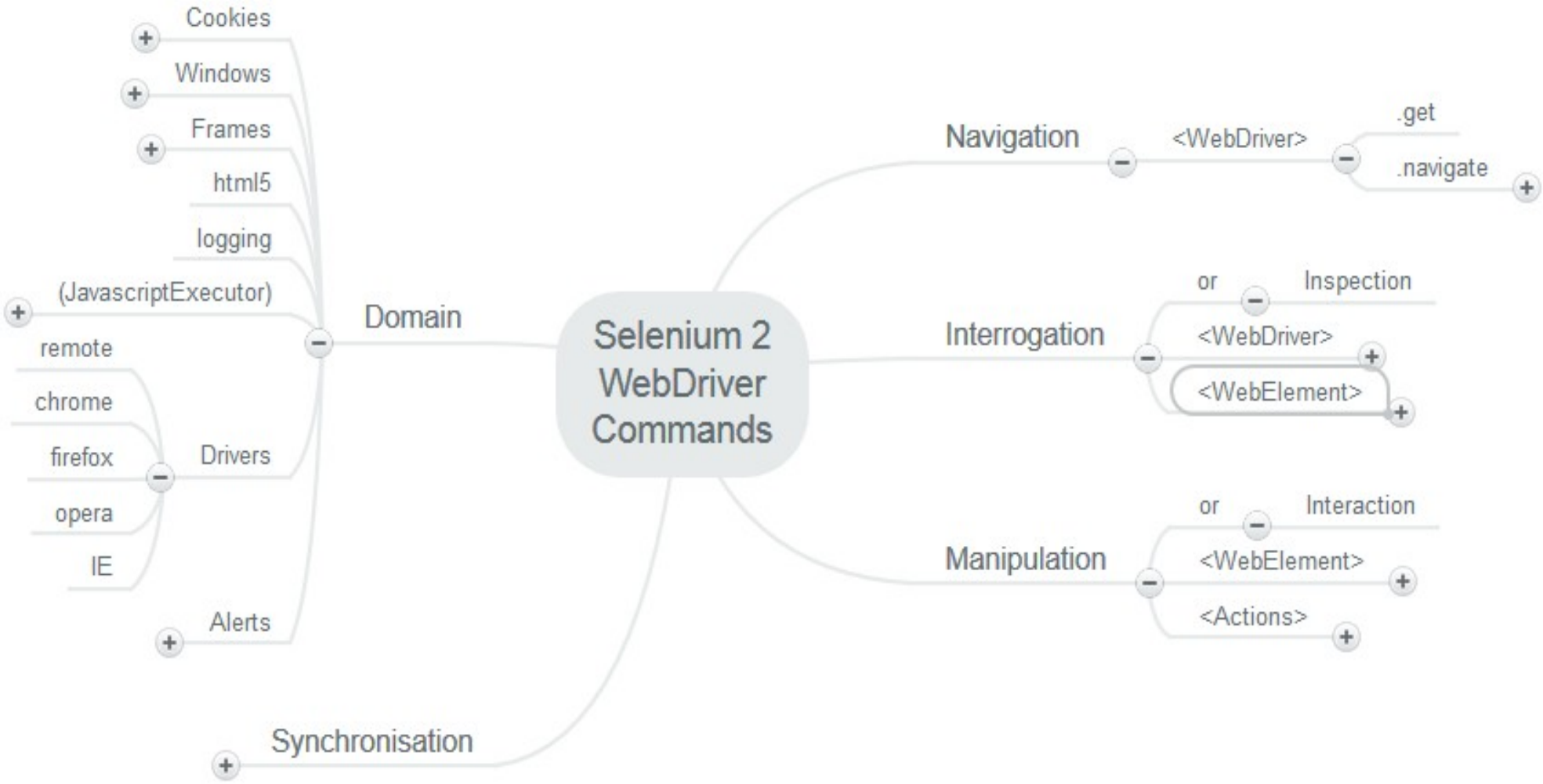
http://unow.be/at/se2014

https://xp-dev.com/svn/HandsOnWebDriverOneDayCourseJava/trunk/

http://compendiumdev.co.uk/files/conferences/se2014/handson20140504.zip

Section: WebDriver is an API

API Overview



http://www.mindmeister.com/280141421/selenium-2-webdriver-commands

http://unow.be/at/apimindmap

Basic Knowledge

- *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

WebDriver Basics

- WebDriver
  - Think of the driver as the browser
  - Interface with implementations – FirefoxDriver, ChromeDriver etc.
  - High Level commands
    - get, getTitle, getCurrentUrl, close, quit
  - Functional Groupings
    - Navigate, Manage
  - Locate
    - FindElement, FindElements

WebElement Basics

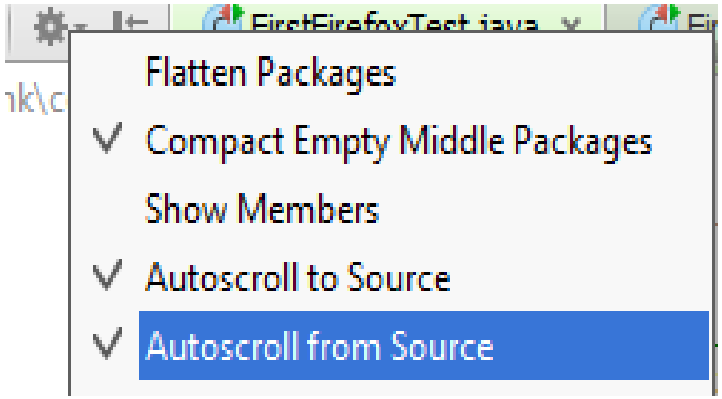
- The DOM elements
  - <a>, <p>, <ul>, etc.
- Found By.Id, By.Name, etc.
- Interrogate
  - .getAttribute, .getValue, isEnabled etc.
- Manipulate
  - SendKeys, click, etc.

JUnit Basics

- Test Execution Framework
- @Test
  - Annotate a method with @Test
- Can Run @Test methods from the IDE
- Use Assertions to check results
  - Assert.assertTrue(true);
  - Assert.assertEquals(1+1, 2);

WebDriver API Learning Tips

- Use Code Completion (ctrl+space)
- ctrl+Q for Documentation of Commands (ctrl+j on mac)
- ctrl+click to view the actual WebDriver code (cmd+click on mac)
- AutoScroll



IntelliJ Navigation Shortcut Keys

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

on mac use Cmd instead of Cntrl

Browser Tools Overview

- Firefox
  - Inspect
  - Inspect with Firebug
    - FirePath
- Chrome Inspect

Section: My First Test

Demo

- http://seleniumsimplified.com/testpages/basic_web_page.html
  - Get Page, Check Title, Find para1, Check Text
- How to run a test Class
- How to run a test method
- How to set a breakpoint
- How to debug a test Method
- How to use evaluate

MyFirstExtendedTest.java

```
@Test
public void getBasicPageAndCheckDetails(){

  WebDriver driver = new FirefoxDriver();

  driver.get(
    "http://seleniumsimplified.com/testpages/basic_web_page.html"
       );

  Assert.assertTrue("Expected Different Title",
                    driver.getTitle().
                       equals("Basic Web Page Title"));

  WebElement para1 = driver.findElement(By.id("para1"));
  Assert.assertTrue(
             "A paragraph of text".equals(para1.getText()));

  driver.close();
  driver.quit();
}
```

Used WebDriver API Summary

- WebDriver driver = new FirefoxDriver();
  - Create a new instance of a driver and browser
- driver.get(aURL);
  - Open a web page in the browser
- findElement, By.id, By.name
  - WebElement element = driver.findElement(By.id(“anid));
  - Find a WebElement using id or name attribute,
  - **findElement can be chained**
- driver.getTitle()
  - Get the title of the current page
- driver.close(), driver.quit()
  - Close the current browser window,
  - quit the browser and driver
- Junit Used
  - @Test
  - Assert.assertTrue
  - Assert.assertEquals

My First Test Exercises

IntelliJ & JUnit

Java & Maven Summary

- Packages organise the Java classes
- Java Classes Start with Uppercase letters
- Methods start with lowercase letters
- Add tests in the src/test/java folder hierarchy
- Add Test into the class name to automatically run from maven as a test
- Import classes from other packages to use them
- Code completion helps, be careful what you import

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

IntelliJ Evaluate Expression

- In Debug mode use Evaluate to experiment with code constructs in situ.
  - e.g. when writing xpath findElements code
- Can sometimes recompile and hot swap the code into the running debug session.

JUnit @Before, @After

- @BeforeClass,
  - Run once before any @Test in the class is run
  - static
- @AfterClass
  - Run once after all @Test in the class have run
  - static
- @Before,
  - Run before each @Test
- @After
  - Run after each @Test

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

Hamcrest Matchers CheatSheet

e.g. assertThat(driver.getTitle(), is(not("bob"));

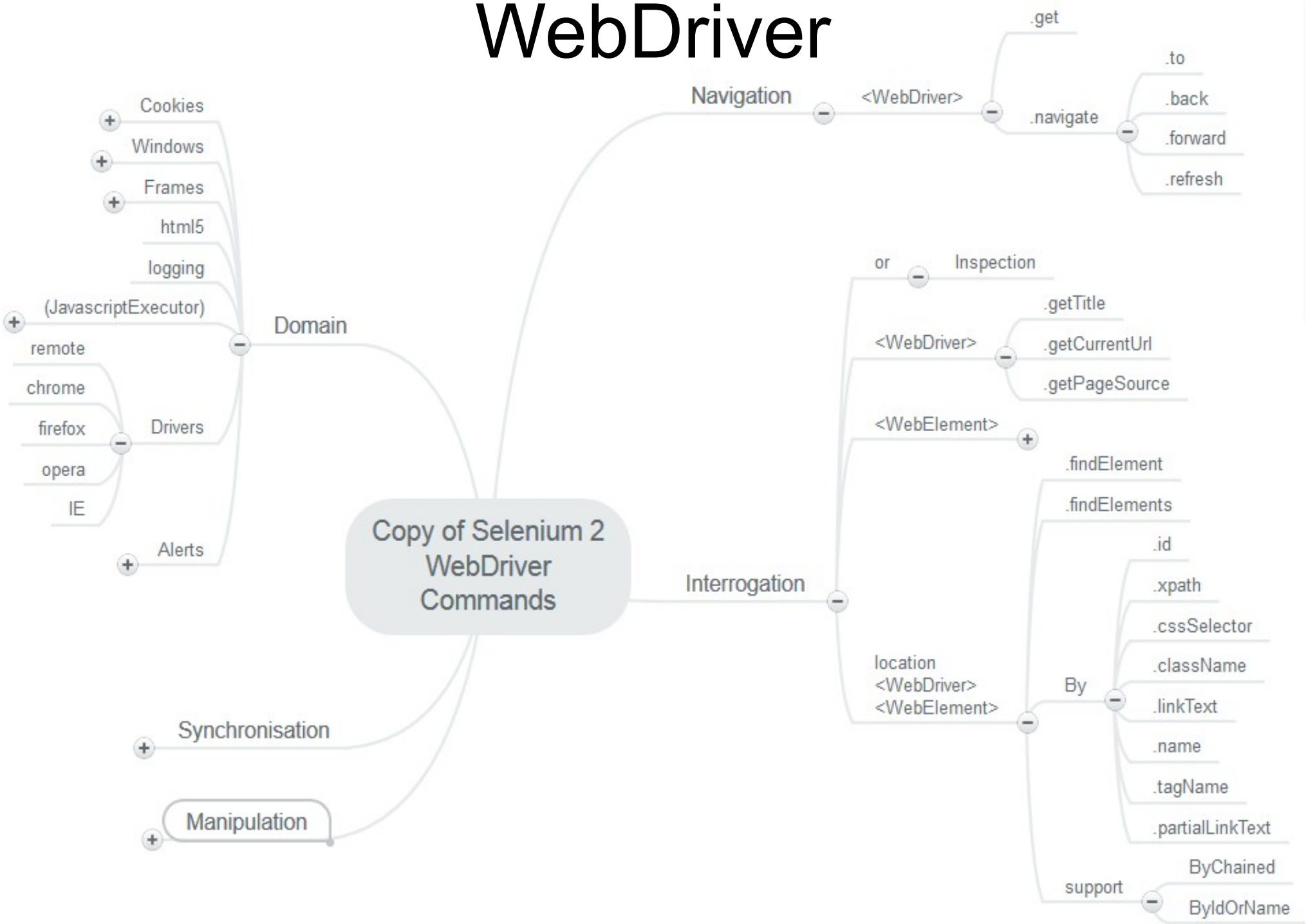
- is
- not
- *Number*
  - greaterThan,
  - greaterThanOrEqualTo,
  - lessThan,
  - lessThanOrEqualTo - test ordering
- *Text*
  - equalToIgnoringCase
  - equalToIgnoringWhiteSpace
  - containsString,
  - endsWith,
  - startsWith
- *Collections*
  - hasEntry,
  - hasKey,
  - hasValue - test a map contains 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

Hamcrest & JUnit Exercises

Navigation & Interrogation

WebDriver



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

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.

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

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.

.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"));`

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

Basic WebElement API Summary

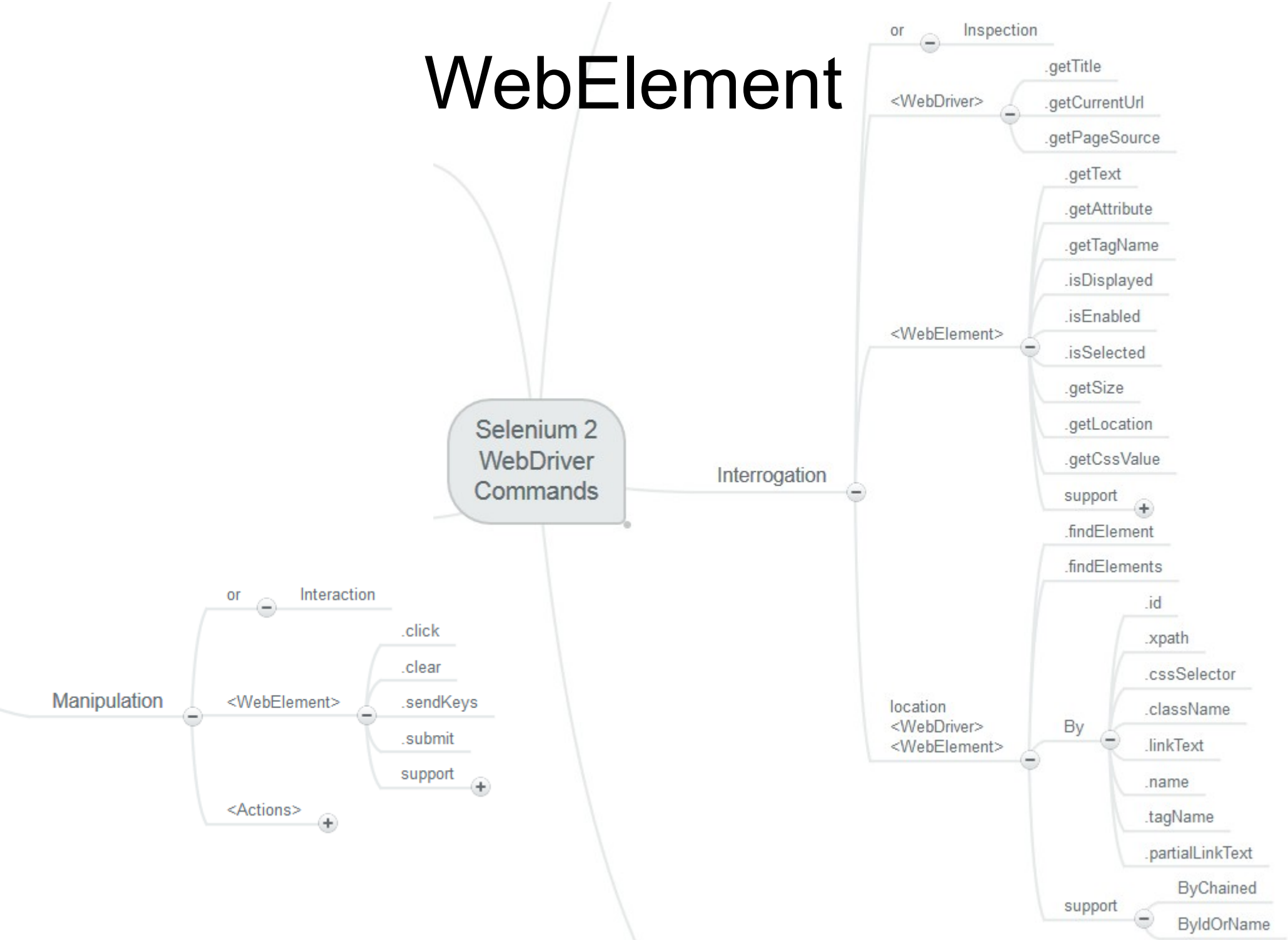
- WebElement, findElement,
  - WebElement element = driver.findElement(By.id(“anid));
  - Find a WebElement using id or name attribute,
  - **findElement can be chained**
- By.id, By.name
  - Locators to 'find' the elements

- element.sendKeys(“type this”);
  - Send keys to a web element to type something
- element.click()
  - Click on a web element

element.getText

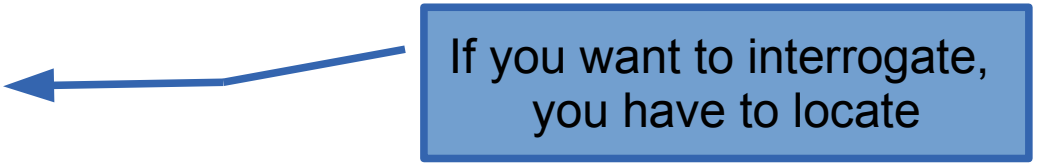
  - Get the text of the current element

WebElement



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()

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

Navigation & Interrogation Exercises

Manipulation

WebElement Manipulation

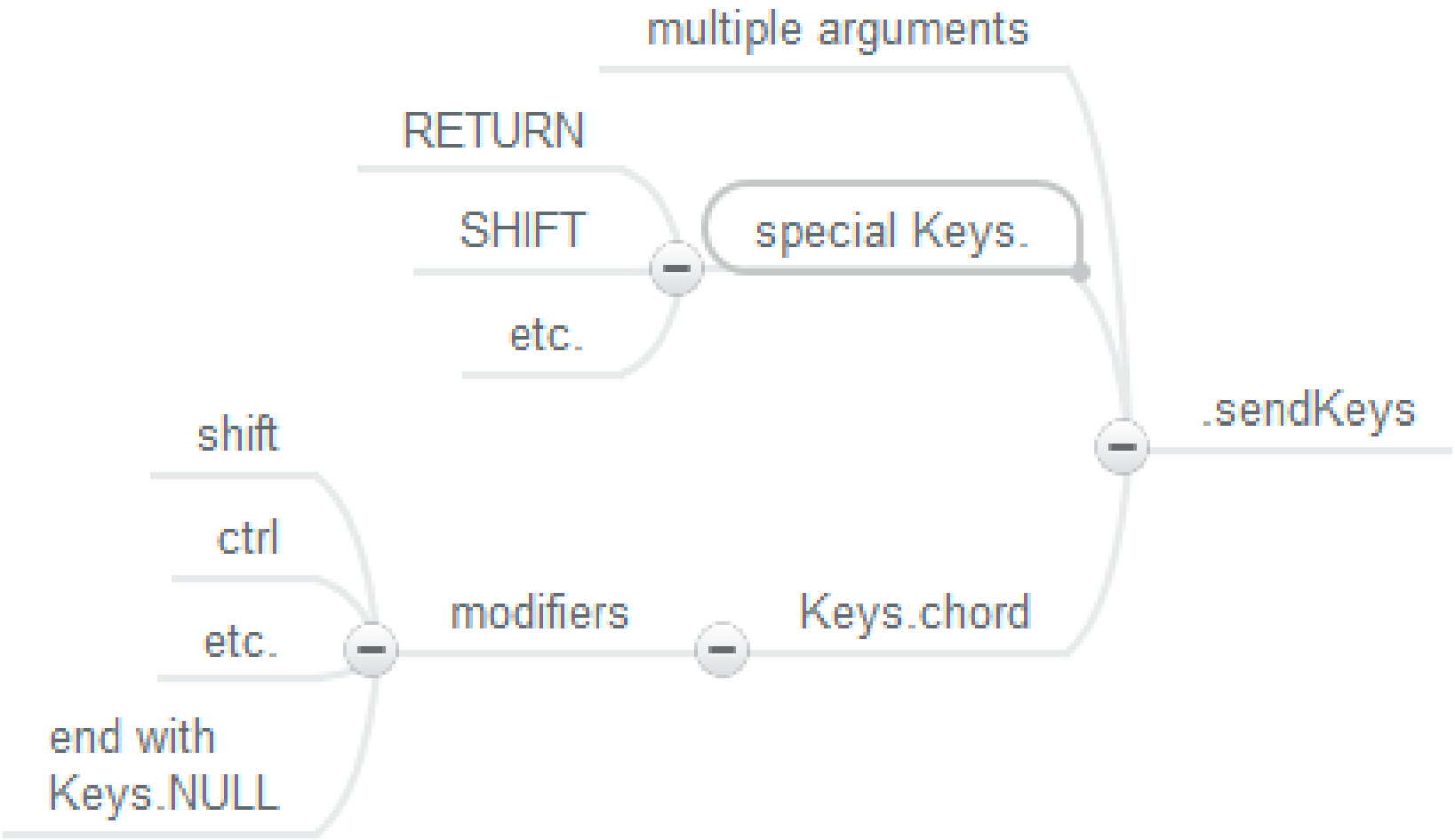
- .click()
- .clear()
  - Clear a field
- .sendKeys(String) *actually (CharSequence)*
  - Sends the appropriate events to a WebElement keyup, keydown, etc.
  - Helper Keys class (org.openqa.selenium.Keys)
- .submit()
  - Submit a form

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())
```

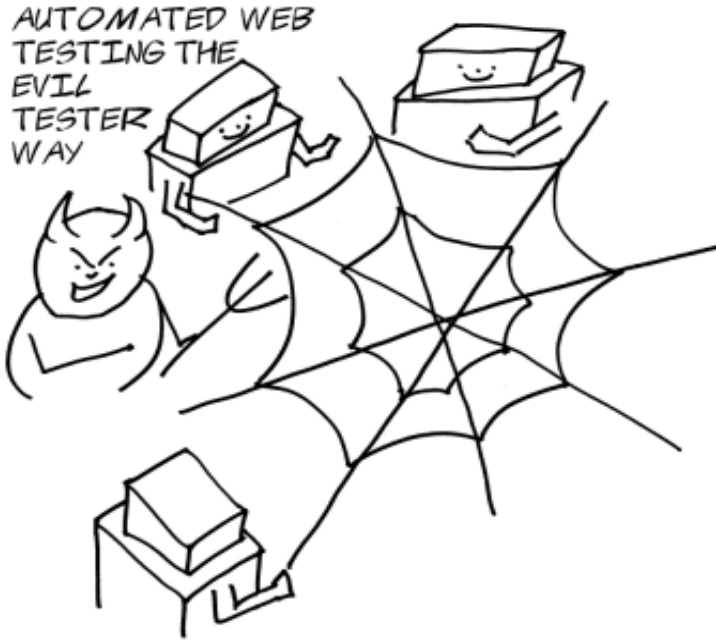
SendKeys



Manipulation Exercises

Location Strategies

CSS Selectors



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 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

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
- B:nth-child(n) – the nth child of B e.g. n==3, 3rd
  - For more selectors see the references

By.cssSelector

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

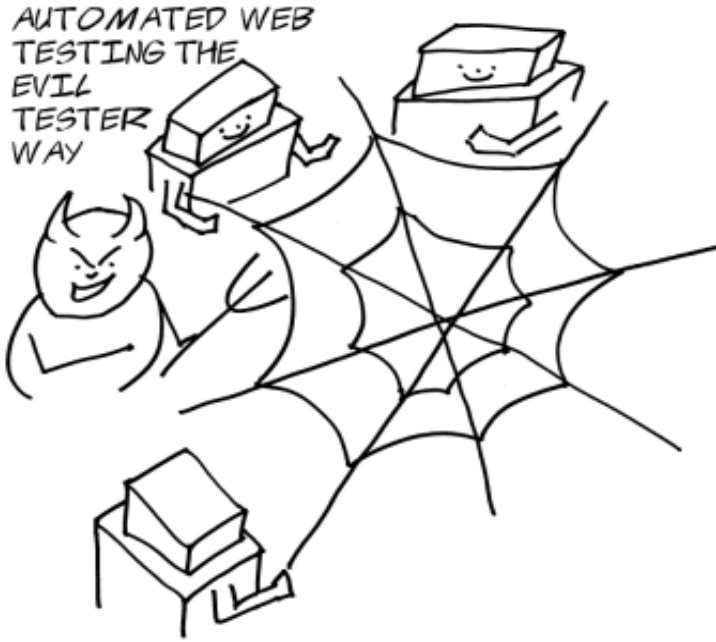
```
@Test
public void findElementsUsingCSSTag(){

    List<WebElement> elements;

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

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

CSS Selectors

- CSS Selectors are faster than Xpath
- CSS Selectors can't traverse back up the dom
- Try and use CSS Selectors by default for complicated selection queries

Exercise: Manual CSS Selectors

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

```
@Test
 public void findSpecificPara(){

    WebElement element;
    element = driver.findElement(

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

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

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

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

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

- Note browsers tend to use Xpath 1.0

Chain XPath & CSS in findElement

- Remember we can 'chain' findElement
  - elem.findElement(By.id('me')).findElement(By.id('this');
- You can use a combination of css selector and xpath selector
  - parentWebElement = webElementFoundByCSS.findElement(By.xpath(".."));
  - parentWebElement.findElements(By.cssSelector("div > div"));

Exercise: Manual XPath Selectors

Exercise: CSS Basic Exercises

Synchronisation

Discussion

- Why wait?
- What conditions might you have waited for? e.g. visible... what else?
- Why wait for title instead of assert?
- Can we make the code cleaner?

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

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

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

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
- Can make tests increasingly slower
- Implicit can make test failures slower to report

Basic Synchronisation with ExpectedConditions

- Wait for the page to be in the correct state before working with it
  - Why? ElementNotFound exception
  - More robust across browsers
- WebDriverWait
  - A wait class which handles element not found exceptions
- ExpectedConditions
  - A class of static helper methods
  - e.g. new WebDriverWait(driver,10).until(
    ExpectedConditions.*somecondition*(
    By.id("user_login")));

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.

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

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

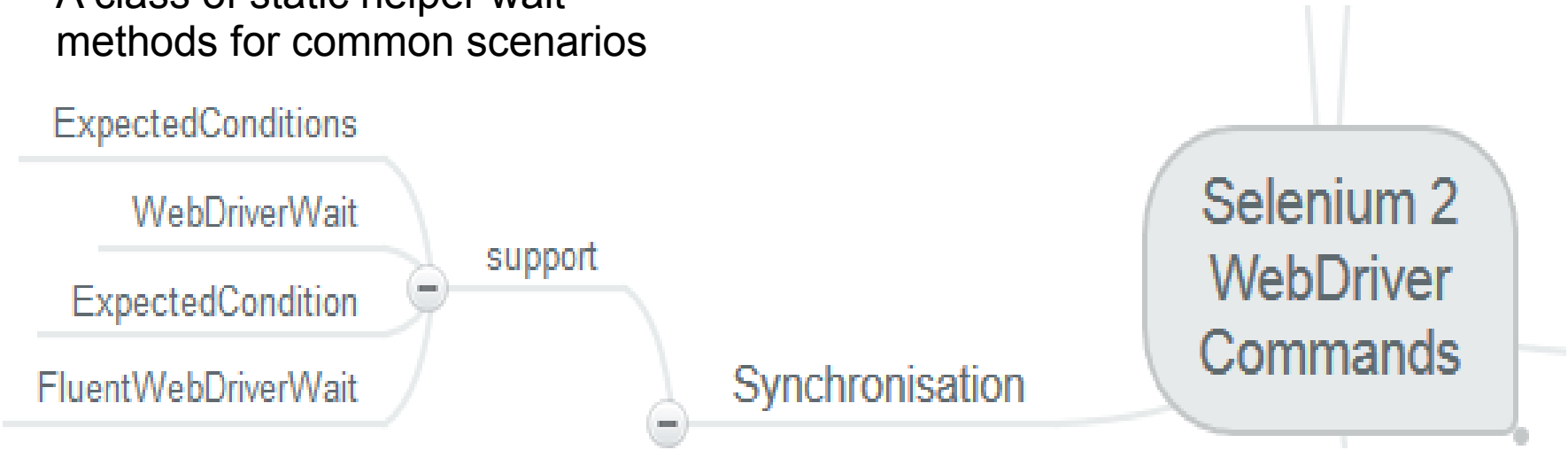
Explore the methods

Synchronisation API Summary

- new WebDriverWait(driver, seconds)
  - Create a wait class
- .until
  - Wait until a specific condition happens
  - until can return an element
- ExpectedConditions
  - A class of static helper wait methods for common scenarios

e.g.

- .titleIs(expectedValue)
  - True when title of page is expectedValue
- visibilityOfElementLocated
  - Returns an element when element located is visible



Section:
Custom Expected Condition

Custom ExpectedCondition

```
new WebDriverWait(driver,10).until(
        contextItemCountChanges(currentActionsForContext, "in tutorial"));
);
```

- 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 anything e.g.
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.

Synchronisation Exercises

Abstraction

Abstraction Layers

- Abstraction Layers are
  - not a framework
  - contextual for an application, or an application framework
  - Essential for:
    - Maintenance
    - Speed of writing tests
    - Ease of understanding tests

Abstraction Layers Categorised

1) **Data**
   - Generic Data Abstractions e.g. email, postcode
2) **Physical**
   - Physical layout of your application e.g. pages, components
   - Navigation across pages
3) Domain
   - Your application Entities domain e.g. user, account
4) Logical
   - User actions, workflows

Hints

- Create the test you want to see
  - Write the test as if the abstraction layers already existed
  - So you can:
    - See if it is readable, at the right level etc.
    - Use code completion to create classes (IntelliJ: Alt+Enter)

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 or SlowLoadableComponent
  - Basic objects
- Modelling Decisions Required

Example Test Using Page Object

```
@Test
public void basicHtmlPageTest(){

    WebDriver driver = new FirefoxDriver();
    BasicHtmlPage page =
                    new BasicHtmlPage(driver);

    page.get();

    assertEquals(page.getParagraphOne(),
              "A paragraph of text");
    assertEquals(page.getParagraphTwo(),
              "Another paragraph of text");
}
```

Example Page Object

```
public class BasicHtmlPage {
    private final WebDriver driver;
    private String paragraphTwo;

    public BasicHtmlPage(WebDriver driver) {
        this.driver = driver;
    }

    public void get() {
        driver.get(
"http://seleniumsimplified.com/testpages/basic_web_page.html");
    }

    public String getParagraphOne() {
        return driver.findElement(By.id("para1")).getText();
    }

    public String getParagraphTwo() {
        return driver.findElement(By.id("para2")).getText();
    }
}
```

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

Page Object Factory

- Annotate WebElements with @FindBy

```
@FindBy(how= How.ID, using="combo1")
private WebElement aComboElement;
```

- Initialise the annotated WebElement(s) using a PageFactory

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

Slow Loadable Component

- If our page objects extend SlowLoadableComponent then we have an interface for 'waiting' for free
- Instead of:

```
TracksDashboardHomePage dashboard =
     new TracksDashboardHomePage(driver);

new WebDriverWait(driver,10).until(
     ExpectedConditions.titleIs(
         dashboard.EXPECTED_TITLE));
```

- We have to do:

```
TracksDashboardHomePage dashboard =
     new TracksDashboardHomePage(driver);

 dashboard.get();
```

SlowLoadableComponent

- Public interface
  - get()
    - loads and waits for component to be available

```
extends SlowLoadableComponent<PageObjectClassName>
```

- Call super constructor in constructor

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

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

Why do this?

- Synchronise on load
  - Page Load – don't do anything until page is loaded
  - Component Load – so we don't try to engage with the component until it is loaded
  - Get in the habit of synchronising
  - Interface makes it easy to extend
  - 'forced' to think about synchronisation
  - Encourages more comprehensive checks on 'ready'
- No impact
  - If we don't call .get() we don't trigger the wait

Summary

- Page Objects can be POJO
- Domain Objects are POJO
- Create abstraction layers
- Write failing code and use code completion
- Never add asserts into your abstraction layer
- Write methods so they read as native language
- Refactor in small increments
- Keep refactoring

Page Object Exercises

Bonus Sections

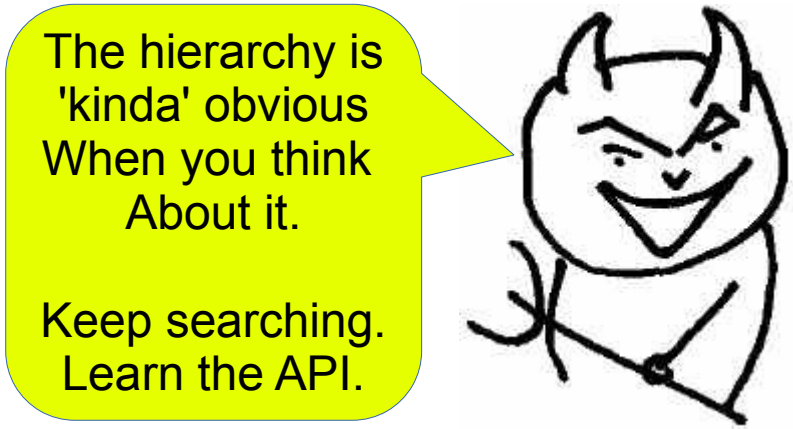
Support Classes as Abstractions

Examine the WebDriver source code for examples to build on.

- Select
- ExpectedConditions

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



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

More About Actions

- Actions uses GWT – operating system mouse move etc. so is more realistic
- You can interfere with the test if you move mouse and keyboard
- Actions can be brittle

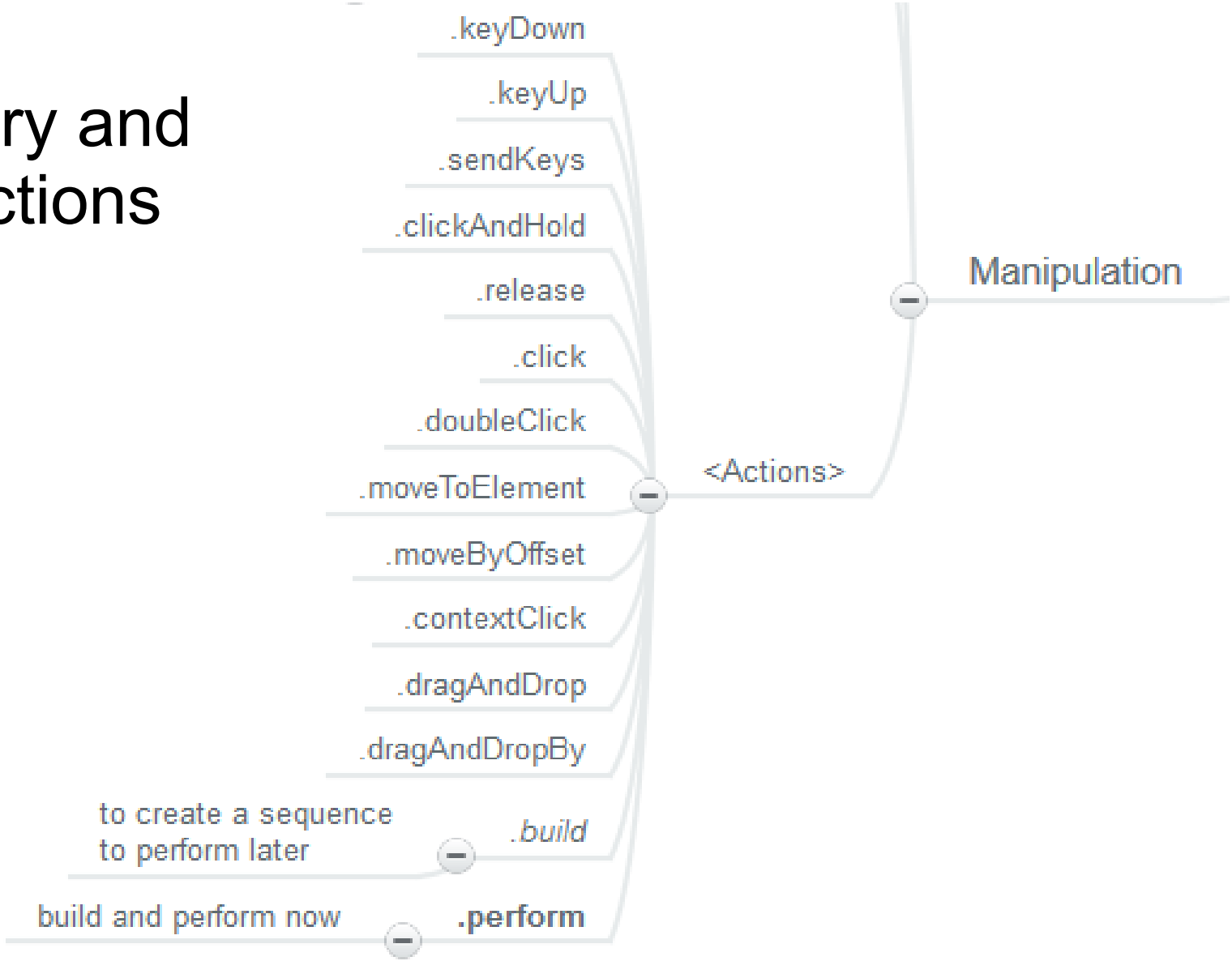
Actions to simulate user Actions

- Hover, etc.
- new Actions(driver).
  ChainOfActions.
  build().perform();
- e.g. a chain of click actions

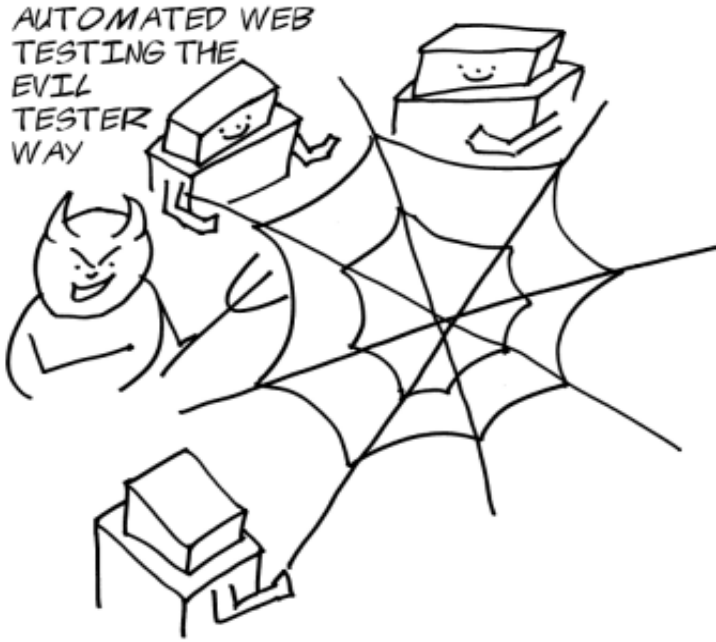
```
Actions actions = new Actions(driver);

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

Actions API Summary Slides

Note: I try and avoid actions



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

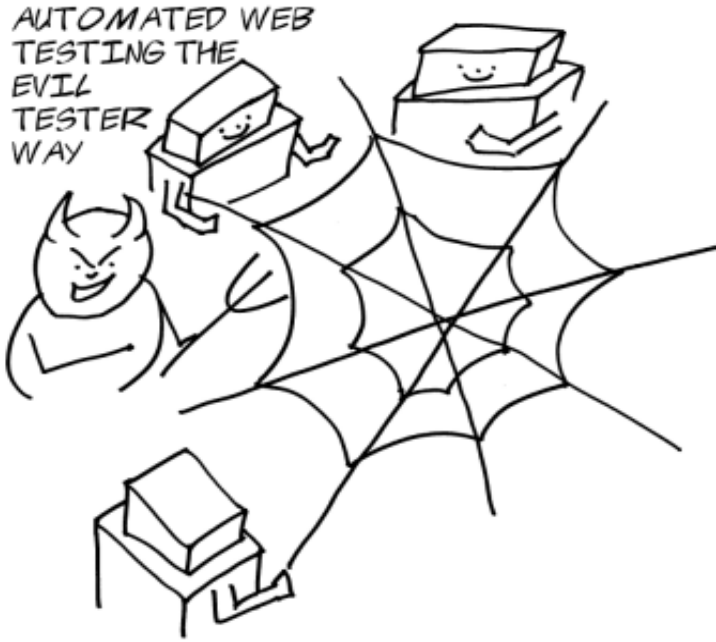
Find a named cookie
Return null if not found

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();
```

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

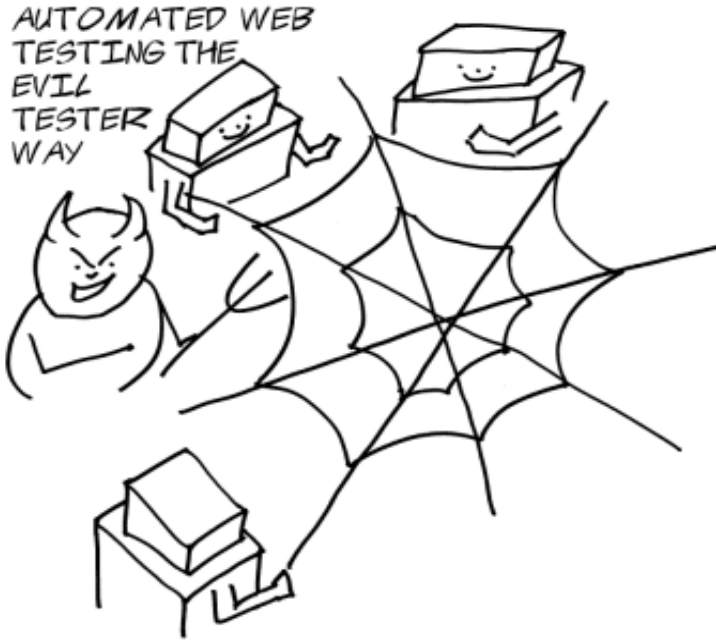
Execute Async JavaScript

- 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

```
@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);

}
```

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

```
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

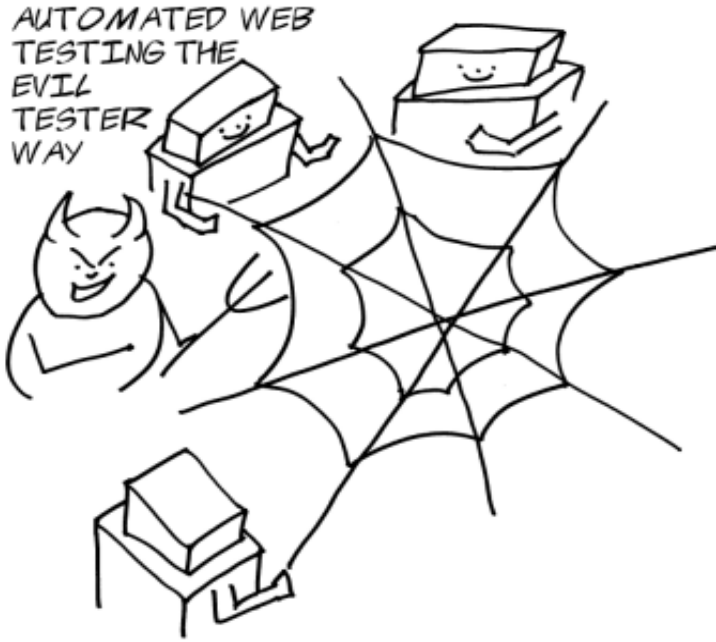
- http://code.google.com/p/selenium/wiki/InternetExplorerDriver
- Download the server executable
  - http://code.google.com/p/selenium/downloads/list
  - Set “webdriver.ie.driver” to the location of the driver executable

```
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

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);
```

Grid Systems

- saucelabs.com
- browserstack.com

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

Section: Any Time Left?

Any Questions?

Time for Free For All Automation?

- Pick a section and try and automated it
- Refactor some of the code into page objects or domain objects
- Try generating test data
- Try some of the advanced exercises

Reference Section

Basic Practice Pages

- http://SeleniumSimplified.com/testpages/
  - 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://github.com/eviltester/seleniumTestPages

Advanced Practice Pages

- compendiumdev.co.uk/selenium/Showcase/Showcase.html
- Source available at https://github.com/eviltester/simpleGWTShowCaseClone

Learn More Java

- Java For Testers by Alan Richardson
- 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

- Practice on some simple apps:
  - http://todomvc.com/
  - http://google-gruyere.appspot.com/
  - http://www.redmine.org/
  - http://getontracks.org/

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/

Alan Richardson

uk.linkedin.com/in/eviltester

Independent Test Consultant
& Custom Training

Contact Alan

http://compendiumdev.co.uk/contact

Blogs and Websites

- CompendiumDev.co.uk
- SeleniumSimplified.com
- EvilTester.com
- JavaForTesters.com
- Twitter: **@eviltester**

Online Training Courses

- Technical Web Testing 101
  Unow.be/at/techwebtest101
- Intro to Selenium
  Unow.be/at/startwebdriver
- Selenium 2 WebDriver API
  Unow.be/at/webdriverapi

Videos

youtube.com/user/EviltesterVideos

Books

Selenium Simplified

Unow.be/rc/selsimp

Java For Testers

leanpub.com/javaForTesters