Hands On WebDriver: Training

Alan Richardson

@eviltester

alan@compendiumdev.co.uk

www.seleniumsimplified.com
www.eviltester.com
www.compendiumdev.co.uk

Install

- IntelliJ IDE
- Maven
- Firefox
- FirePath plugin

Application Under Test

- Tracks
  - getontracks.org
- bitnami install
  - Create admin user
  - Username: “user”
  - Password: “password”
  - Makes it easier to use the sample code

Section: Create Project and Install WebDriver
New Project, Pom.xml

Create New Project

- Add Junit & WebDriver to the project

What is Maven?

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

pom.xml

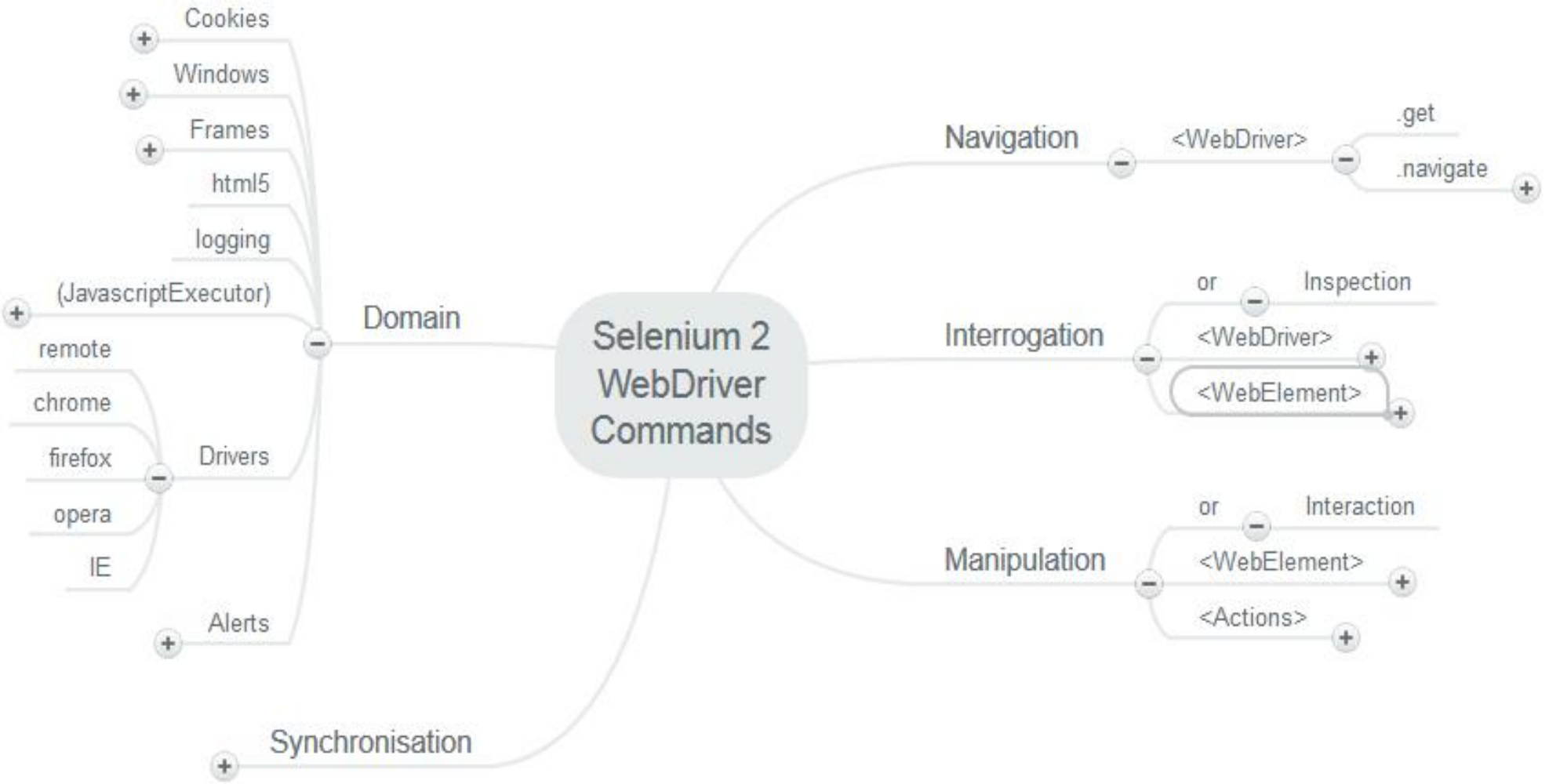
```
<dependencies>
    <dependency>
        <groupId>org.hamcrest</groupId>
        <artifactId>hamcrest-all</artifactId>
        <version>1.3</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-server</artifactId>
        <version>2.32.0</version>
    </dependency>
</dependencies>
```

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”

Section: WebDriver is an API

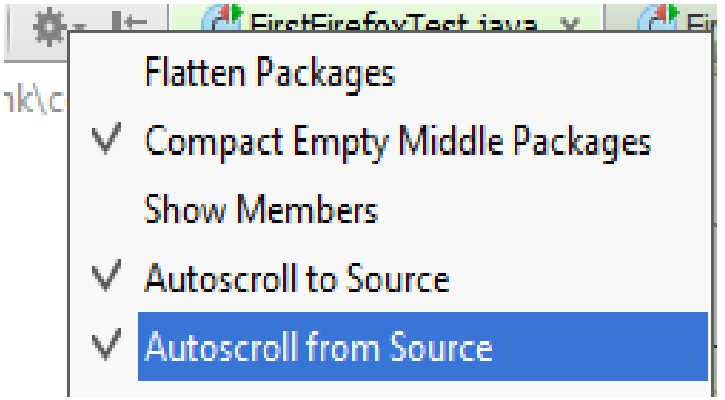
API Overview



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

WebDriver API Learning Tips

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



Navigation Shortcut Keys

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

Section: My First Test Login

Login Tests

- Tests
  - We want to be able to login as admin user
  - An invalid User can not login

Create a Login Test: Demo & Exercise: Follow along

- Package: com.seleniumsimplified.tracks.login
- ClassName: LoginTest
- Method: aUserCanLogin

Covers: Java Packages, Java Classes, @Test, import, test src folder vs main, Junit & Maven Naming, Browser inspection, code completion, run test from IDE,

WebDriver API Used:

WebDriver, new FirefoxDriver(), .get(), WebElement, findElement, By.id, By.name, sendKeys, click, assertThat, is(), .getTitle(), .close(), .quit(),

Demo

- 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

Exercise: Create a Fail to Logon Test

- Create a new test method
- Type in wrong password
- Find the error message element
- Check text is an error message
- New WebDriver API method: getText()

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 the xpath findElements code
- Can sometimes recompile and hot swap the code into the running debug session.

Junit Summary

- Test Execution and Categorisation Framework
- @Test
  - Annotate a method with @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"));

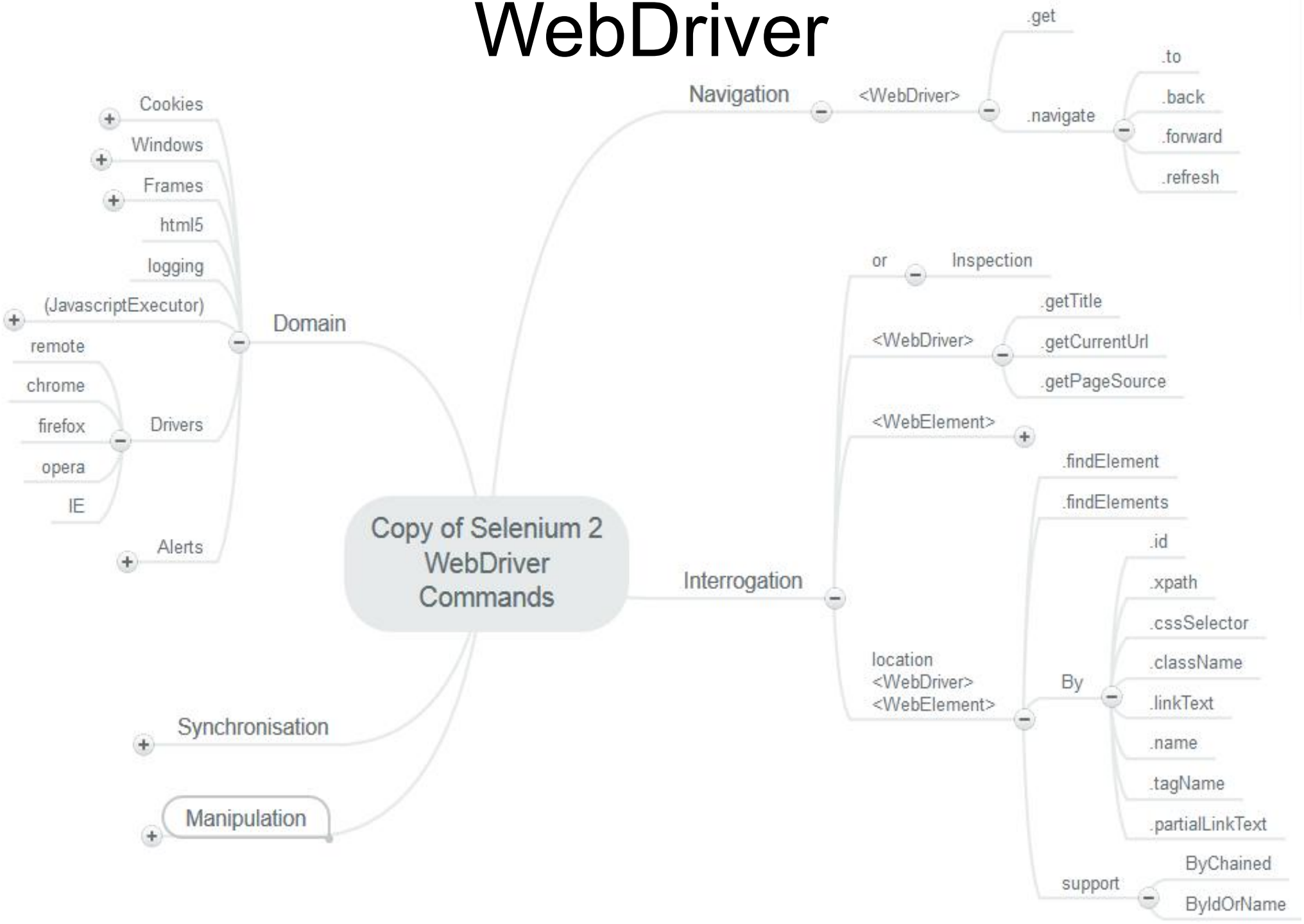
- is
- not
- *Number*
  - greaterThan,
  - greaterThanOrEqualTo,
  - lessThan,
  - lessThanOrEqualTo - test ordering
- *Text*
  - equalToIgnoringCase
  - equalToIgnoringWhiteSpace
  - containsString,
  - endsWith,
  - startsWith
- *Collections*
  - hasEntry,
  - hasKey,
  - hasValue - test a map contains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

Used WebDriver API Summary

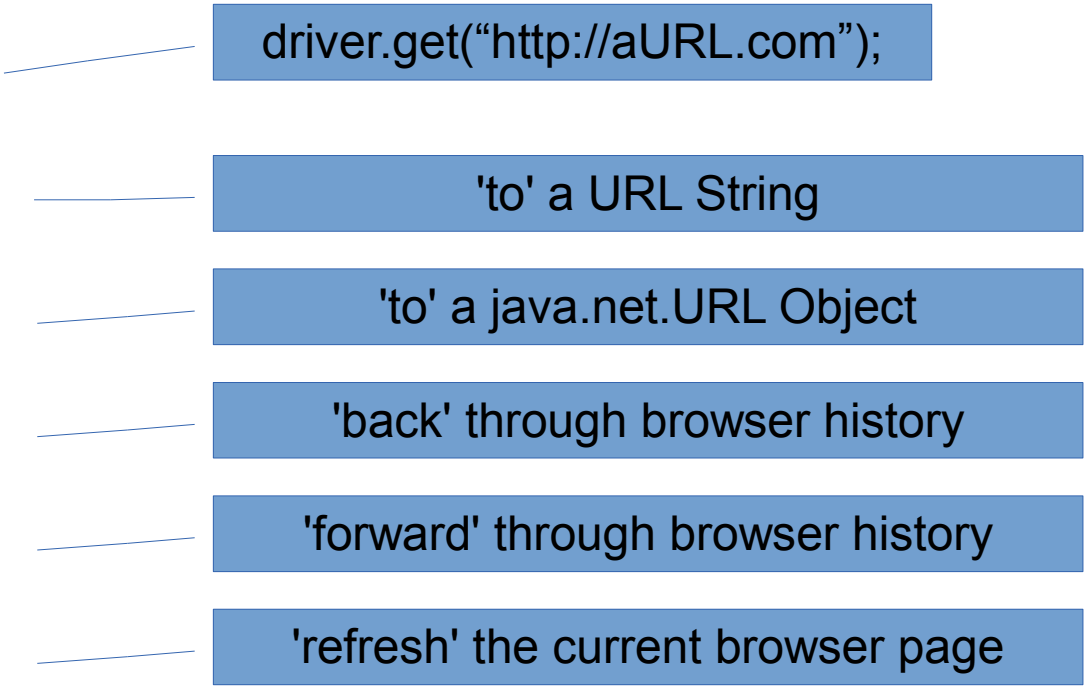
- WebDriver driver = new FirefoxDriver();
  - Create a new instance of a driver and browser
- driver.get(aURL);
  - Open a web page in the browser
- findElement, By.id, By.name
  - WebElement element = driver.findElement(By.id(“anid));
  - Find a WebElement using id or name attribute,
  - **findElement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

WebDriver


Copy of Selenium 2 WebDriver Commands
Navigation
<WebDriver>
.get
.navigate
.to
.back
.forward
.refresh
Domain
Cookies
Windows
Frames
html5
logging
(JavascriptExecutor)
Drivers
remote
chrome
firefox
opera
IE
Alerts
Interrogation
or
Inspection
<WebDriver>
.getTitle
.getCurrentUrl
.getPageSource
<WebElement>
location
<WebDriver>
<WebElement>
.findElement
.findElements
By
.id
.xpath
.cssSelector
.className
.linkText
.name
.tagName
.partialLinkText
support
ByChained
ByIdOrName
Synchronisation
Manipulation

Navigation Annotated

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



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")

```
@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());
}
```

Takes a String which could be the ID or the Name

Have to instantiate, not use statically

Used WebElement API Summary

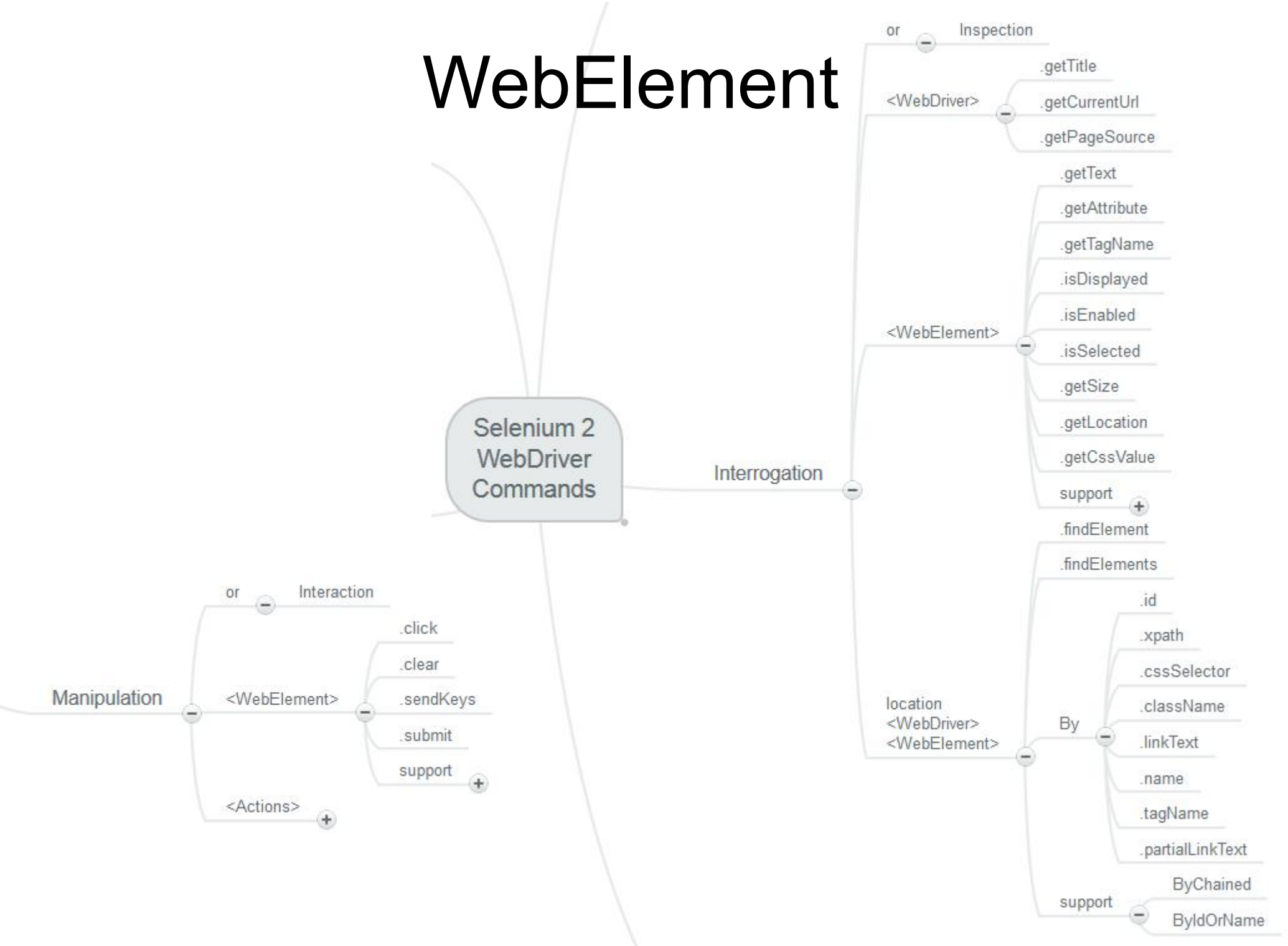
- WebElement, findElement,
  - WebElement element = driver.findElement(By.id(“anid));
  - Find a WebElement using id or name attribute,
  - **findElement can be chained**
- By.id, By.name
  - Locators to 'find' the elements

- 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


Selenium 2 WebDriver Commands
Interrogation
or
Inspection
<WebDriver>
.getTitle
.getCurrentUrl
.getPageSource
<WebElement>
.getText
.getAttribute
.getTagName
.isDisplayed
.isEnabled
.isSelected
.getSize
.getLocation
.getCssValue
support
location
<WebDriver>
<WebElement>
.findElement
.findElements
By
.id
.xpath
.cssSelector
.className
.linkText
.name
.tagName
.partialLinkText
support
ByChained
ByIdOrName
Manipulation
or
Interaction
<WebElement>
.click
.clear
.sendKeys
.submit
support
<Actions>

Dom Element Interrogation Approach

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

If you want to interrogate, you have to locate

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

WebElement Manipulation

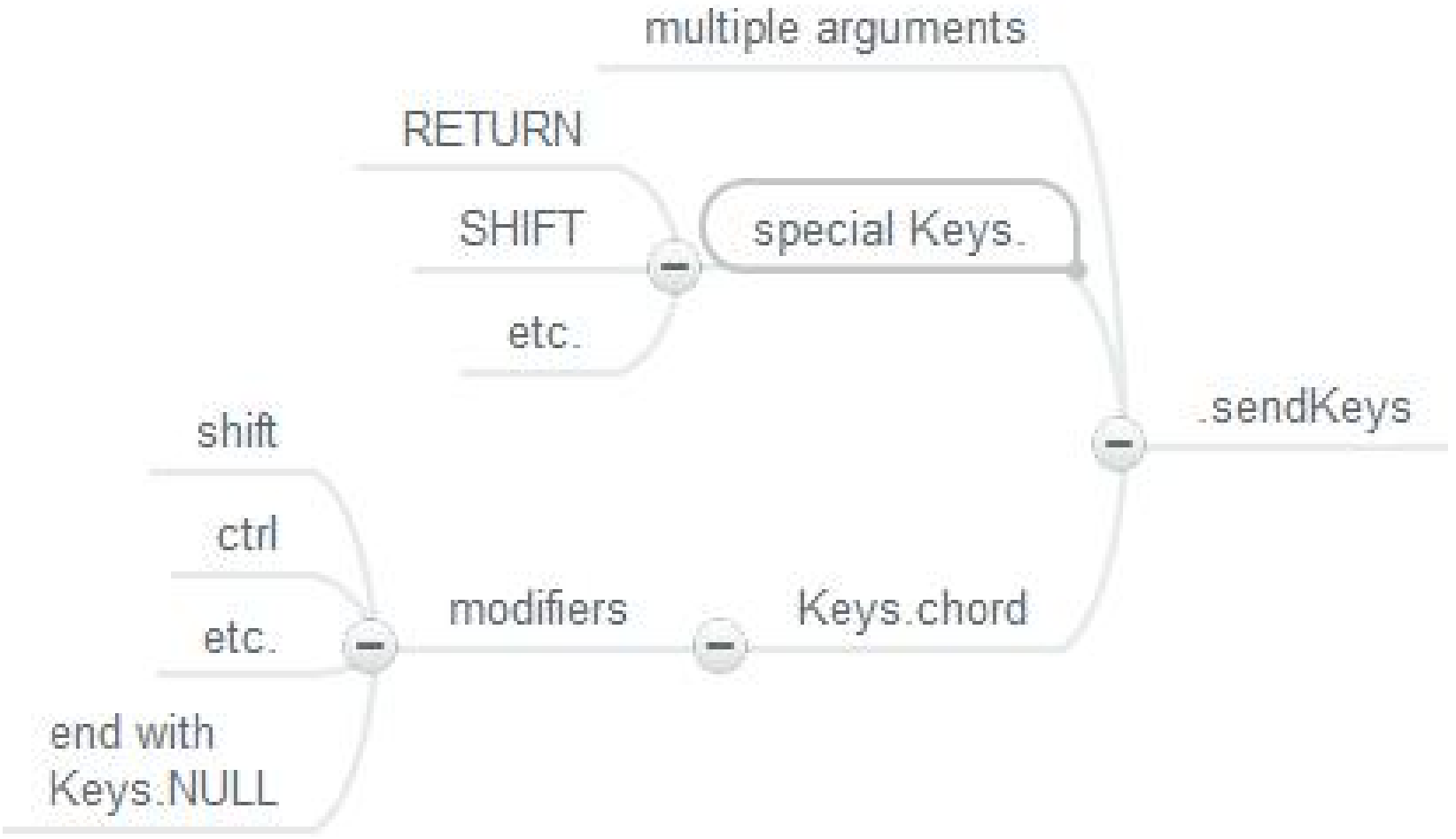
- .click()
- .clear()
  - Clear a field
- .sendKeys(String) *actually (CharSequence)*
  - Sends the appropriate events to a WebElement keyup, keydown, etc.
  - Helper Keys class (org.openqa.selenium.Keys)
- .submit()
  - Submit a form

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


multiple arguments
RETURN
SHIFT
etc.
special Keys.
.sendKeys
shift
ctrl
etc.
modifiers
Keys.chord
end with
Keys.NULL

Section: Refactor Tests To POJO Abstraction Layers

We need to refactor

- We have duplicate code for login across all tests
- We have identified a number of pages we need to add to an abstraction layer
  - Login, Dashboard, Calendar

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

Login Test Refactoring We Could do

- Refactor from
  - Basic test with all details embedded in it, to...
  - Domain Objects
    - A Tracks system/environment
  - Page Objects
    - LoginPage
    - Page redirected to After Login
  - Support Objects
    - DriverProvider

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)

Demo

- Create the Test We want to see

Review and Evaluate the Code

- Why WebDriverManager?
- Review, are the objects at the right level?
  - Do they do too much?
  - Do they do too little?
- Why a get() on Login Page Object?
- Any other questions?

Create the Code with Code Completion

- WebDriverManager
  - main/java
  - com.seleniumsimplified.tracks.webdriver
- Tracks
  - main/java
  - com.seleniumsimplified.tracks.domain.site
- LoginPage, TracksDashboardHomePage
  - main/java
  - com.seleniumsimplified.tracks.pageobjects

Use the code from existing tests

Guided Demo

- Follow along as I refactor the test into page objects

Exercise

- Amend the fail to login test to use the new pageObjects and domain objects

Debrief

- Issues?
- Questions?
- Comments?

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

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

Section: Make Tests Robust with Basic Synchronisation

ExpectedConditions
WebDriverWait

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

Exercise: Basic Synchronisation with ExpectedConditions

- Wait for the login field to be visible before you try to find it and use it
- Use an ExpectedConditions method instead of an assert on getTitle

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?

Exercise: Refactor waits into Page Objects

- We need to continually refactor
- Moving synchronisation into the page objects helps make new tests robust

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);
```

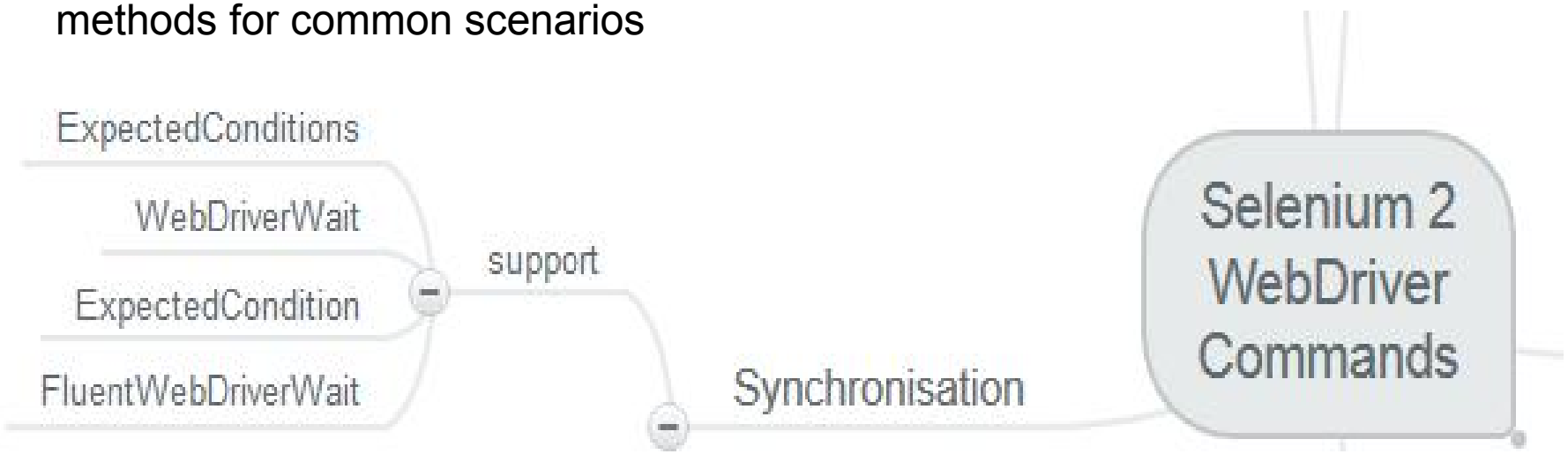
If an element can not be found then a findElement would take 10 seconds before throwing an Exception.

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

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



Section: Junit
@Before, @After

Tests are getting a tad cluttered

- At a test level we have setup code, and test code, so it is harder to see what the test does
  - e.g. create driver, quit driver
  - Are not part of the test, they are clutter

Solution: @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

Guided Demo Refactor Login Test

- Amend Login Test to use
- @BeforeClass
  - Setup the stuff shared by all tests in a class
- @Before
  - Setup the common test preconditions
- @After
  - Clean up after the test
- @AfterClass
  - Clean up driver and global setup

Exercise Refactor Fail to Login Test

- Refactor the logout test to use @BeforeClass, @Before, @AfterClass, @After

Exercise: Create a Logout Method

- Discus: How might we implement a logout method?

- Now. Create a logout method to allow us to logout after each test without closing the browser after each test
  - Use it in the LoginTest @After method

Debrief

- Now the @Test method concentrates on the 'test' not the setup and teardown
  - Easier to read?
- More obvious what tests to add to what class
  - Do they share the same preconditions? @Before, @BeforeClass
- How did you implement the logout method?
- Any other observations or questions about our tests?

Section: Plan the Testing

Survey the site

Exercise: Now that we can login – plan the testing

- Survey the site
- What functionality is there?
- What looks hard to automate?
- What looks suitable for data driven testing?
- What if we automate first can help later testing?

Survey Notes – Functionality Home Screen

- Projects List
  - Checkbox to complete action – Ajax, move into completed actions list
  - Drop down for delete/ defer
  - Active projects list – delete, edit
  - Ajax for star item
  - Show description – ajax
  - [P] Edit project – change to edit project screen
  - List display shared with Tickler – common components
  - Etc.
- Side component
  - New Task (shared across pages)
  - Various project and context lists

Survey Notes - Possible Automation Complexity

- Drop down navigation menus – hover events, slow drawing
- Ajax in screens for display and amend
- Collapsing screen elements

Data Driven

- Helps identify entities & CRUD functionality
- Next Actions
- Projects
- Notes
- Users

Possible Automation Ordering

- Create
  - Users – to test login & admin
  - Projects – to seed users, test lists etc
  - Next Actions – to seed projects, test lists etc
  - Notes – to add to projects
- Edit
  - Projects, Next Actions, Notes, Users
- Delete
  - Next Actions, Projects, Notes

Section: Automate Drop Down Menu

Risk: Drop Down Navigation Menus

- Target the risk of drop down navigation menus first
- Discussion
  - Given our existing level of knowledge, how might we do it?

Exercise

- Use current knowledge to click on the View \ Calendar drop down menu
  - e.g. findElement, By.*x,* sendKeys, click, ExpectedConditions, new WebDriverWait etc.

- Remember to use @BeforeClass, @Before, @After, @AfterClass and existing abstraction layers

Section: Automate Drop Down Menu with Simulated User Actions For Hover

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();
```

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

Exercise

- Create a new @Test method in the same class as the click test
- Use Actions to simulate Hover and Click

- *Remember you can run an individual test by right clicking on the method name*

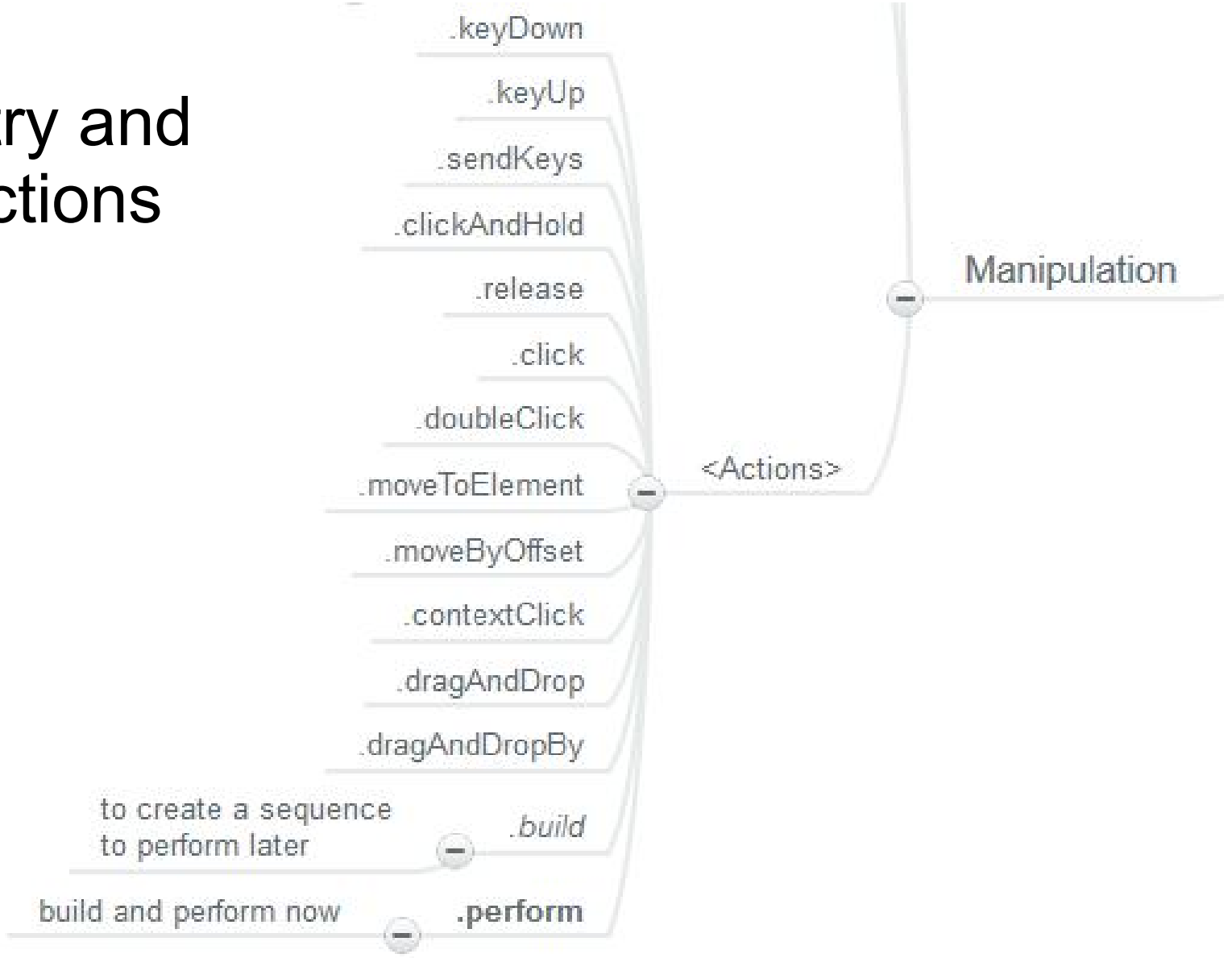
- *Try moving the mouse around when you run the Actions test, then try doing that when you run the click test*

Debrief

- 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 API Summary Slides

Note: I try and avoid actions



Section: Refactor Navigation code into a NavigationMenu Page Object

Exercise: Refactor the NavigationMenu code

- Create a page object to represent the navigation menu
- Use this in your tests to navigate using a menu and sub menu item
- Use your new object to navigate to
  - View \ Calendar
  - View \ Feeds
  - Organize \ Contexts

Section: Support Classes

Slow Loading Component

Another Support Class

- Support Classes you already know
  - ExpectedConditions
  - WebDriverWait
- These are not core WebDriver but are useful to know
- Get in the habit of looking at the source

Quick Demo of Support Classes Code

- Click through a WebDriver class
- Have IntelliJ options on
  - Autoscroll to source
  - Autoscroll from source
- Quick look through the classes
- New items are added with every release

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();
```

SlowLoadableCompnent

- 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

Demo: Convert the LoginPage to SlowLoadableComponent

- Demo where I conver the LoginPage to SlowLoadableComponent

Exercise: Convert TracksDashboardHomePage

- Convert the TracksDashboardHomePage page object to use SlowLoadableComponent
- Implement the load and isLoaded method
- Make any necessary changes to tests

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

Should Navigation Menu use SlowLoadableComponent?

- Discussion

Section: Shared Component

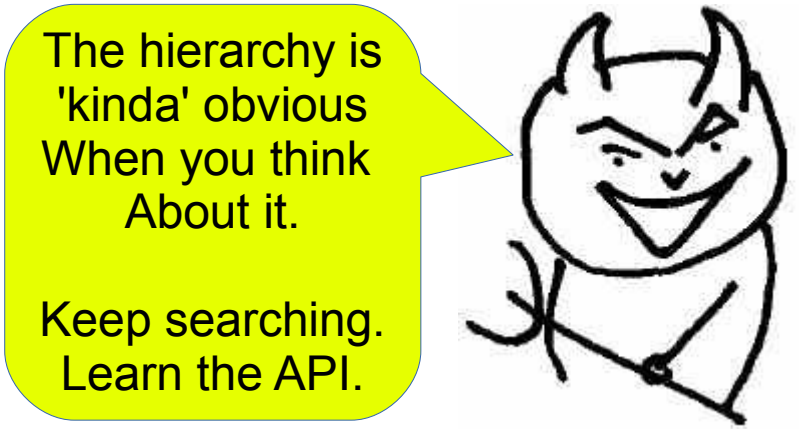
Add Next Action

Exercise: Manually add a next action

- Investigate the form
- Create a 'next action' manually so you know what you are automating

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



Exercise: Create a test to add a next action

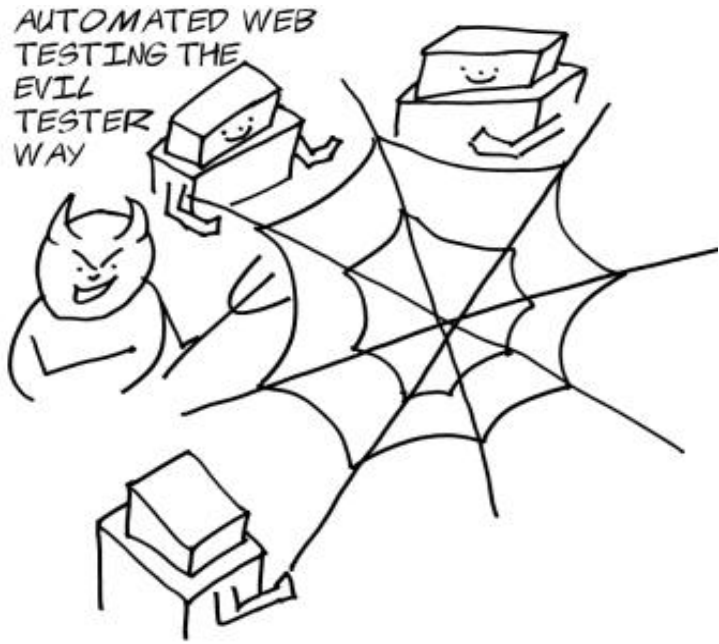
- Create a test which adds a next action
- What you need to know:
  - To handle an alert
    - driver.switchTo().alert().accept();
  - This isn't as easy as it looks, try to keep any automation solutions simple.
    - Hint – you can't click the “Add Action” button when a calendar is visible

Debrief

- What was hard?
- What did you do to allow you to submit the form?
- What is wrong with our test?
  - Repeatable?
  - Maintainable?
  - Extendable?
  - Robust?

Discuss: How can we check that a next action has been added?

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

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

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)

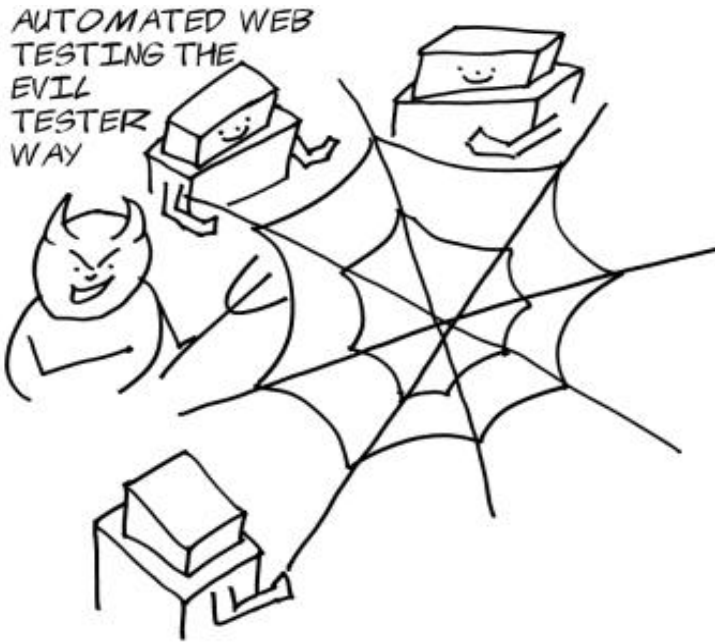
Exercise: Use FirePath to experiment with CSS Selectors

- Select all links
- Select the div with id “toggle_forms”
- Select the anchor with the title “Hide new action form (Alt+N)”
- Experiment with other constructs
- Can you build a css selector which finds all the next actions for a context?

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

XPath



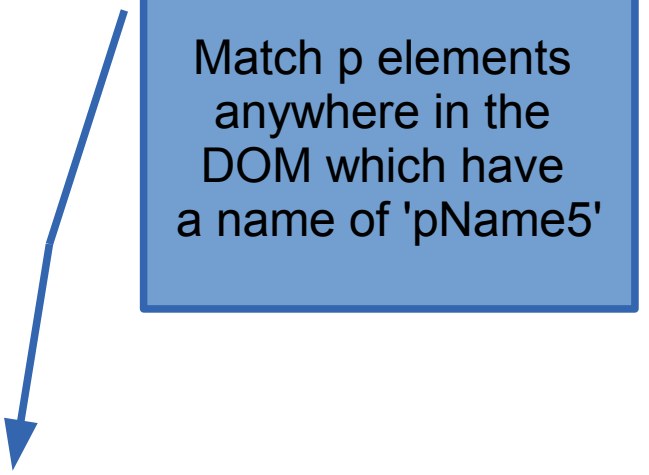
XPath

- 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"));
 }
```



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

Exercise: Use FirePath to experiment with XPath Selectors

- Select all links
- Select the div with id “toggle_forms”
- Select the anchor with the title “Hide new action form (Alt+N)”
- Experiment with other constructs
- Can you build an xpath selector which finds all the next actions for a context?

Before you write code to check if action is added you'll need to know

- How to chain an xpath findElement
- How to write a custom ExpectedCondition

Chain xpath in findElement

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

Exercise: Create a method to count the number of next actions

- Create a method to count the number of next actions in a list given the name of a context
  - Find the context on the page
  - Count the number of items in the context list
  - You may need to use a combination of css selector and xpath selector
    - parentWebElement =

      webElementFoundByCSS.findElement(By.xpath("../.."));
    - parentWebElement.findElements(
      By.cssSelector("div > div"));
- Refactor the count code into a private method

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.

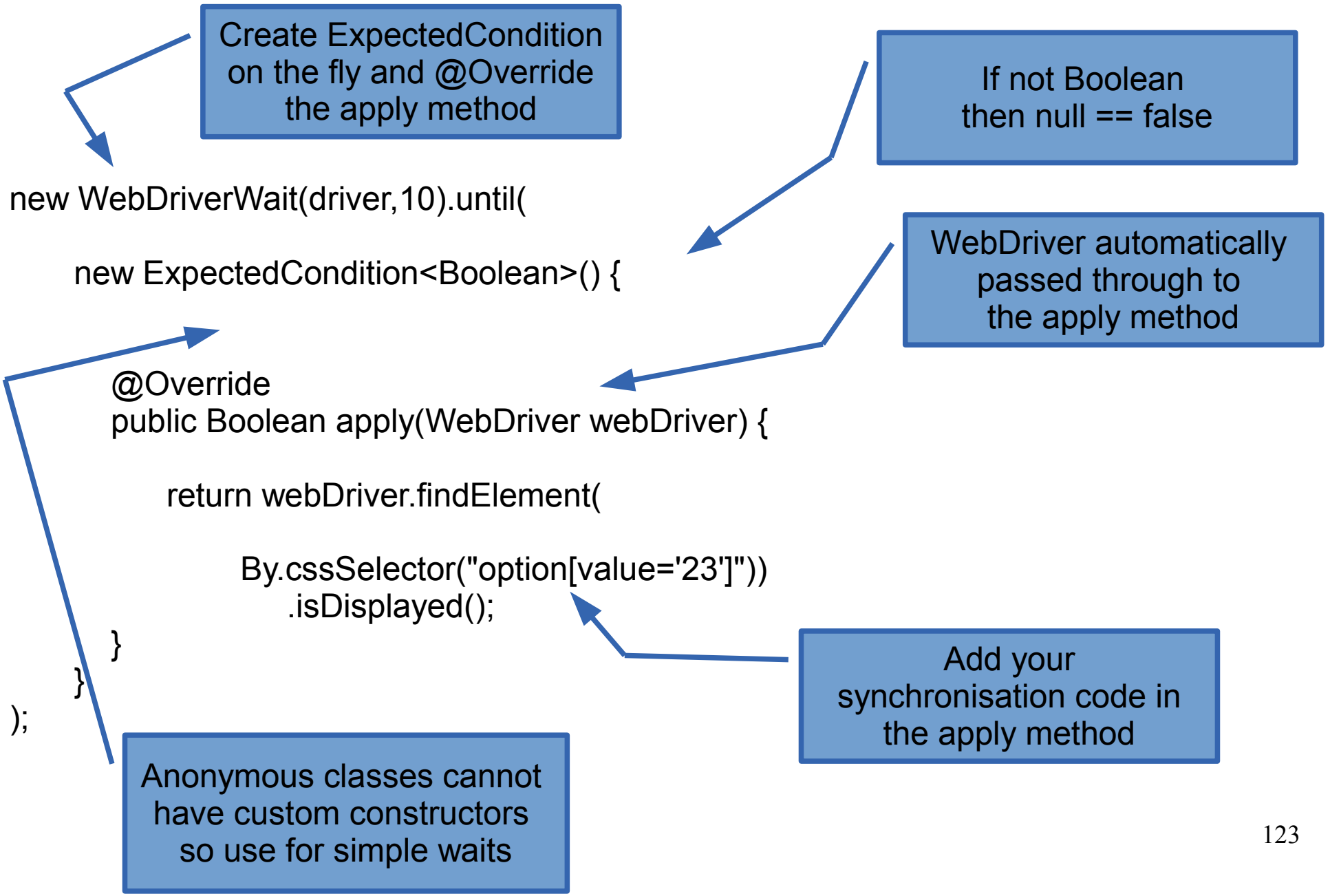
```
private class SelectContainsText implements ExpectedCondition<Boolean> {

    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



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.

Exercise: Write code to wait for action count to increase

- Use a custom ExpectedCondition to check the count goes up, after you add a next Action

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

    assertThat(newCurrentActionsCount, is(greaterThan(currentActionsForContext)));
```

Section: Random Data

Random Or Generated Test Data

- Not strictly WebDriver but important concept
- Generate random valid domain objects
- Use current dates
- Etc.

Useful Reminder about Java Dates

- http://stackoverflow.com/questions/1404210/java-date-vs-calendar

```
DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");

Calendar cal = Calendar.getInstance();

System.out.println(dateFormat.format(cal.getTime()));
```

Reminder about Java Random

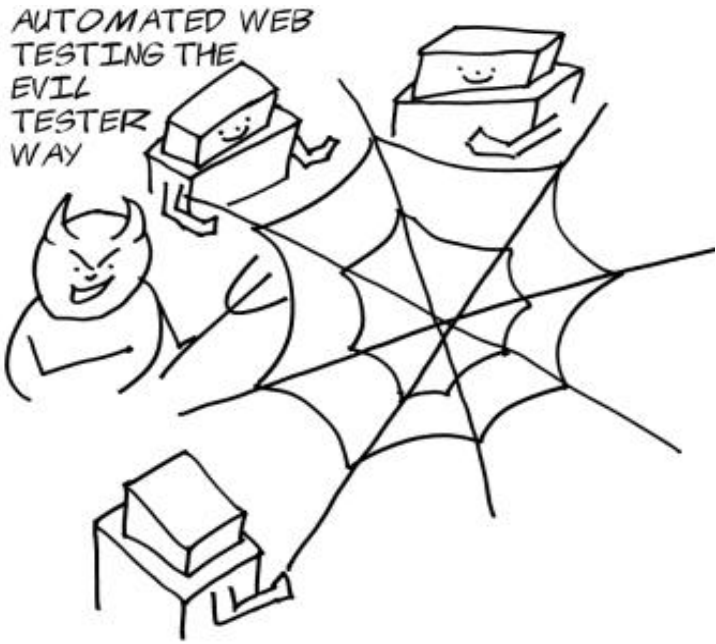
- Java util.Random

```
//a random number between 1 and 10
Random myRandom = new Random();
System.out.println(myRandom.nextInt(10) + 1);
```

- Apply same principle to strings

http://stackoverflow.com/questions/41107/how-to-generate-a-random-alpha-numeric-string-in-java

Cookies

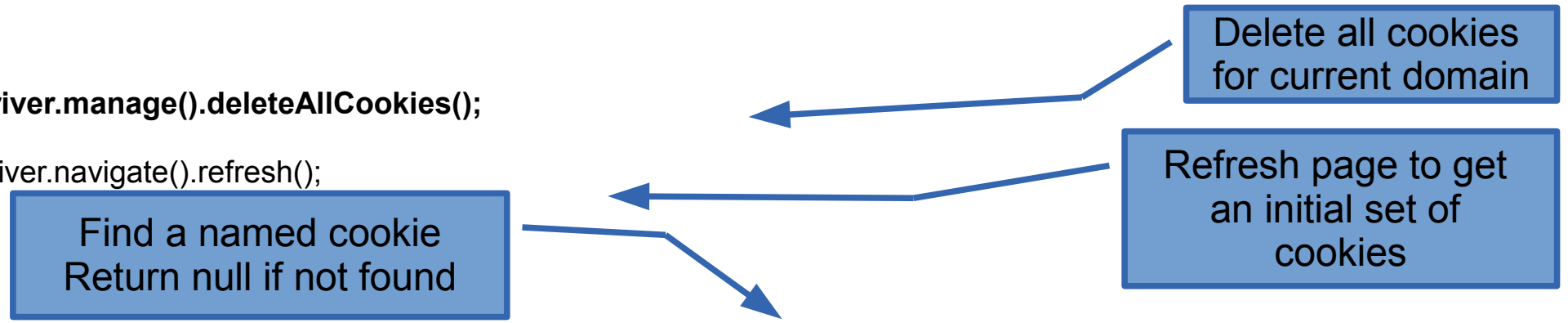


Cookies

- 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);
}
```



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();
```

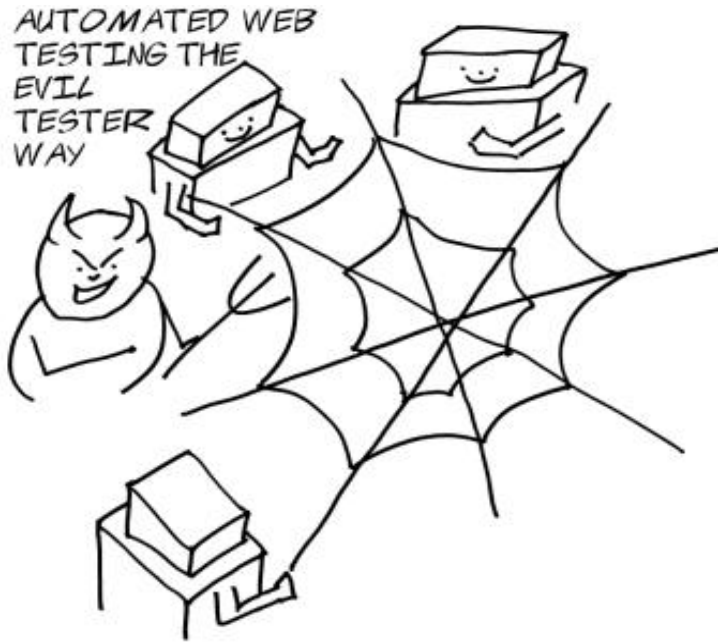
Cookies Demo

- Demo – using FireBug to examine the Cookies
- See changes when we logout

Exercise: Cookies

- Write a test that confirms you are automatically logged out when all cookies are deleted.

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

Tracks Javascript Example

```
@Test
public void javascriptAndJquery(){
    WebElement toggleNotes =
        driver.findElement(By.cssSelector("#toggle-notes-nav"));
    toggleNotes.click();

    JavascriptExecutor javascript = (JavascriptExecutor) driver;

    if((Long)javascript.executeScript("return jQuery.active")!=0)
            System.out.println("JQuery is not running");

    if((Boolean)javascript.executeScript(
                    "return jQuery.active=true")!=false)
        System.out.println("JQuery is not running");
    }
}
```

- If JQuery were still running no console output would happen

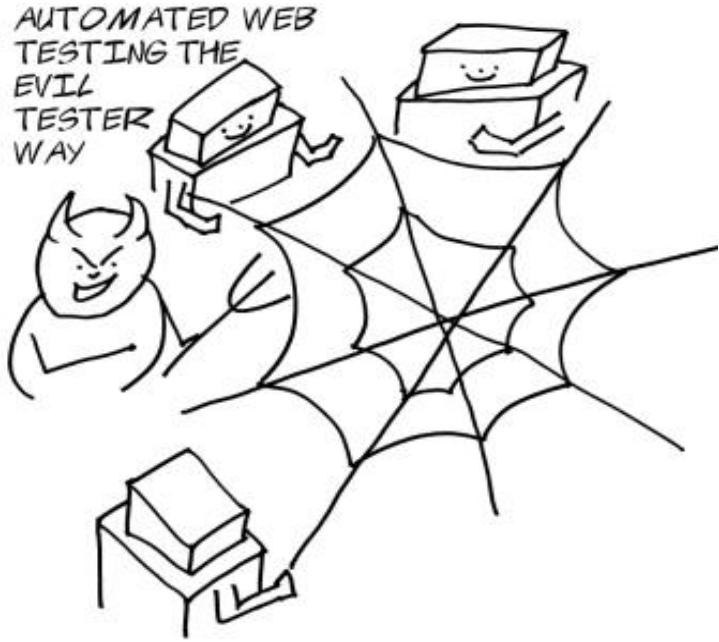
Execute Async JavaScript

- 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 (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

```
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);
```

Calculate the location of Chromedriver relative to my test code

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

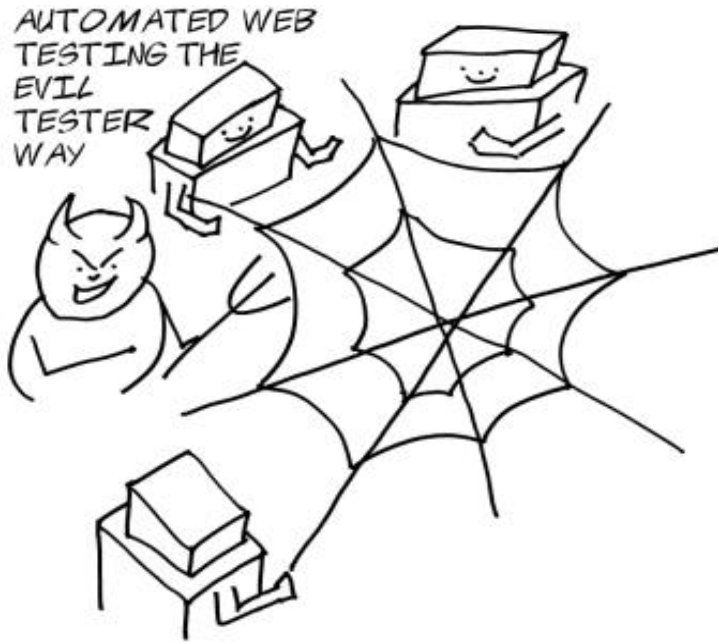
- http://code.google.com/p/selenium/wiki/InternetExplorerDriver
- Download the server executable
  - http://code.google.com/p/selenium/downloads/list
  - Set “webdriver.ie.driver” to the location of the driver executable

```
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

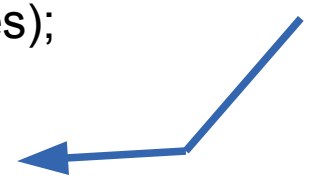
RemoteWebDriver



Remote Driver

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

```
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();
}
```



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

Section: Any Time Left?

Any Questions?

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

Alan Richardson Online

- Selenium Blog
  - SeleniumSimplified.com
- Testing Blog
  - EvilTester.com
- Testing Papers and Tools
  - CompendiumDev.co.uk
- Profile
  - uk.linkedin.com/in/eviltester
- Free: Technical Web Testing 101
  - Unow.be/at/udemy101
- Free: Intro to Selenium
  - Unow.be/at/udemystart
- Selenium 2 WebDriver API
  - Unow.be/at/udemyapi
- **Selenium Simplified Book**
  - Unow.be/rc/selsimp

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