BDD

Lesson – 7 Selenium Web Driver

Lesson Objectives

- Introduction To Web Driver
- Writing first Web Driver Test
- Locating UI Elements-Developers Tools
- Navigation API
- Interrogation API
- WebElement API
- Why synchronization is important
- Using Explicit & Implicit Wait
- JavaScript Executor

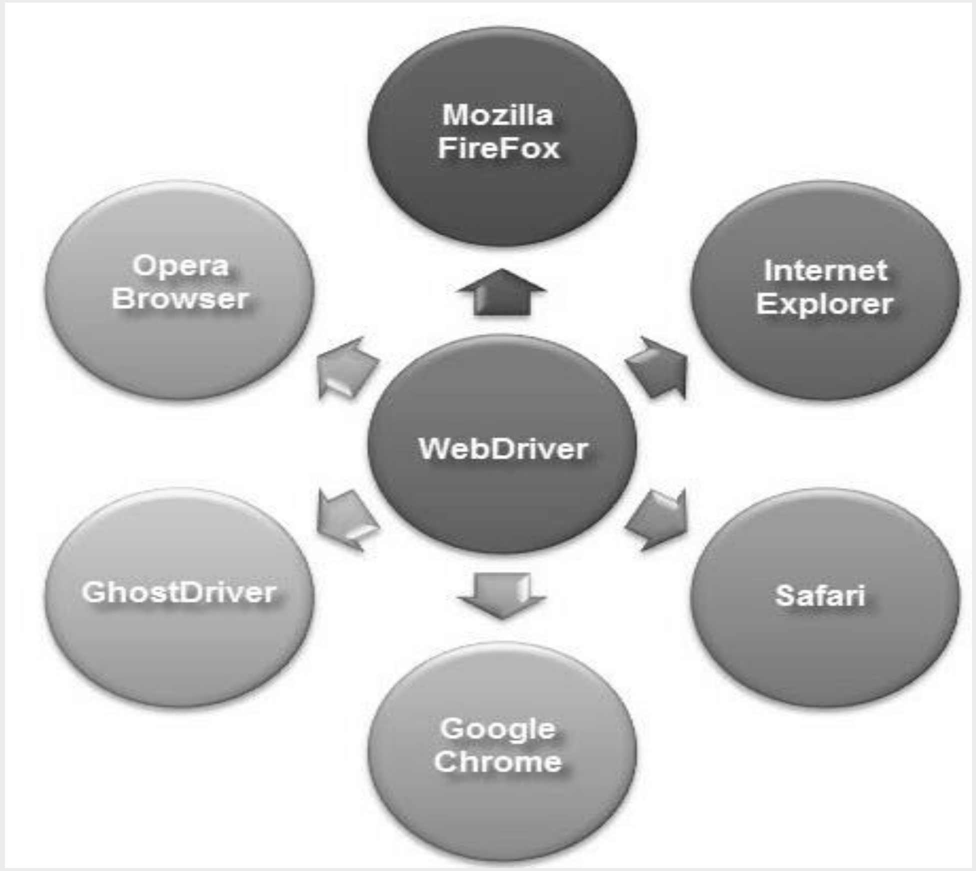
Lesson Objectives (Contd.)

- Selenium: How it works
- Different drivers
  - Chrome
  - Firefox
  - Internet Explorer
  - Headless Browser
    - Ghost Driver and Phantom JS
  - Mobile Browsers
    - Selendriod & Appium
  - Remote Web Driver
  - Capabilities
  - Profile setting
  - Selenium Grid

4.1: Selenium 2.0 – Web Driver

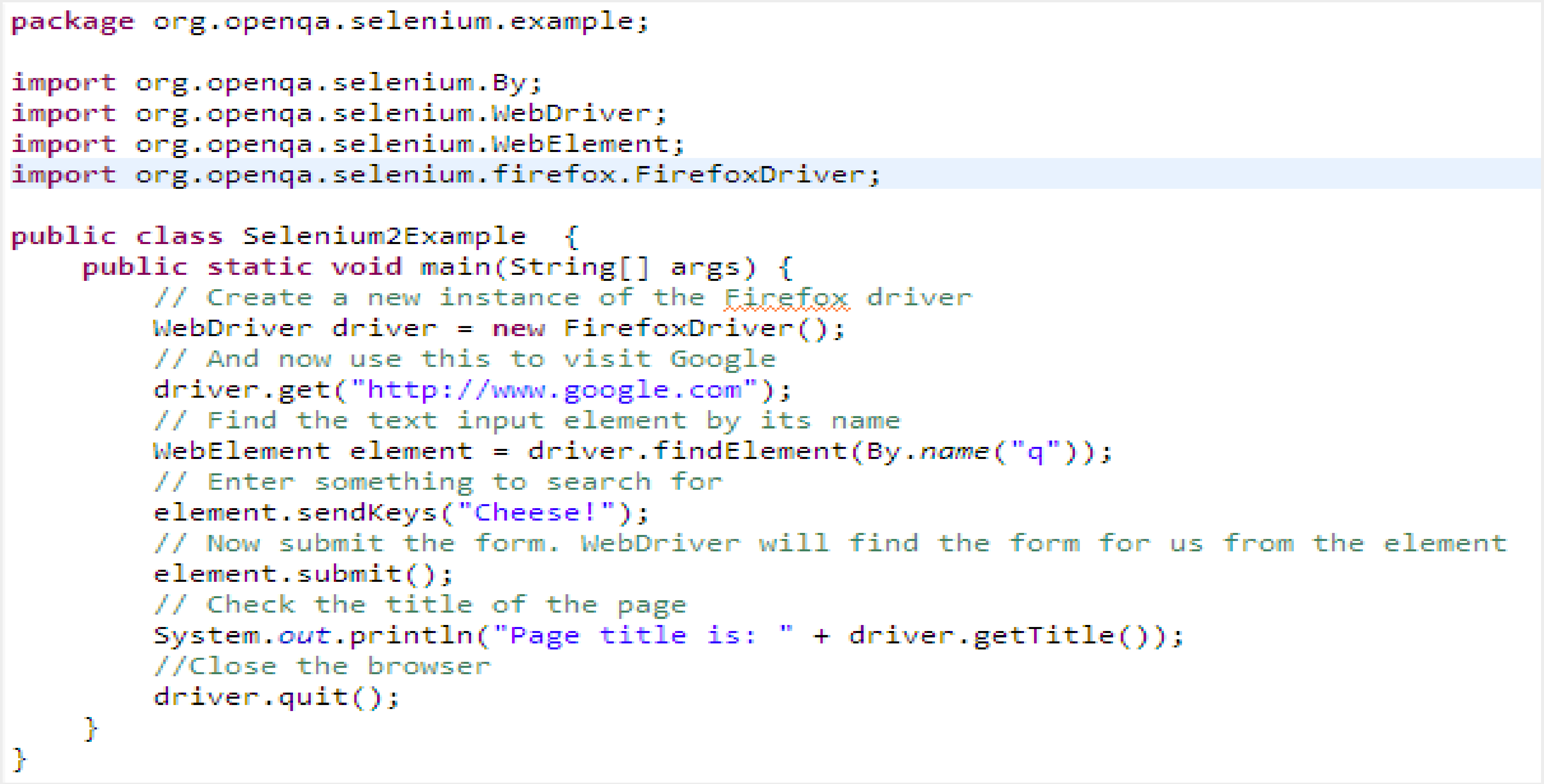
Introduction To Web Driver

- Web automation framework that allows you to execute your tests against different browsers, not just Firefox (unlike Selenium IDE).
- Provides a simpler, more concise programming interface
- Selenium-WebDriver was developed to better support dynamic web pages where elements of a page may change without the page itself being reloaded
- Supply a well-designed object-oriented API that provides improved support for modern advanced web-app testing problems.



5.1: Testing Web Applications Using Web Driver API

Writing first Web Driver Test(Java)



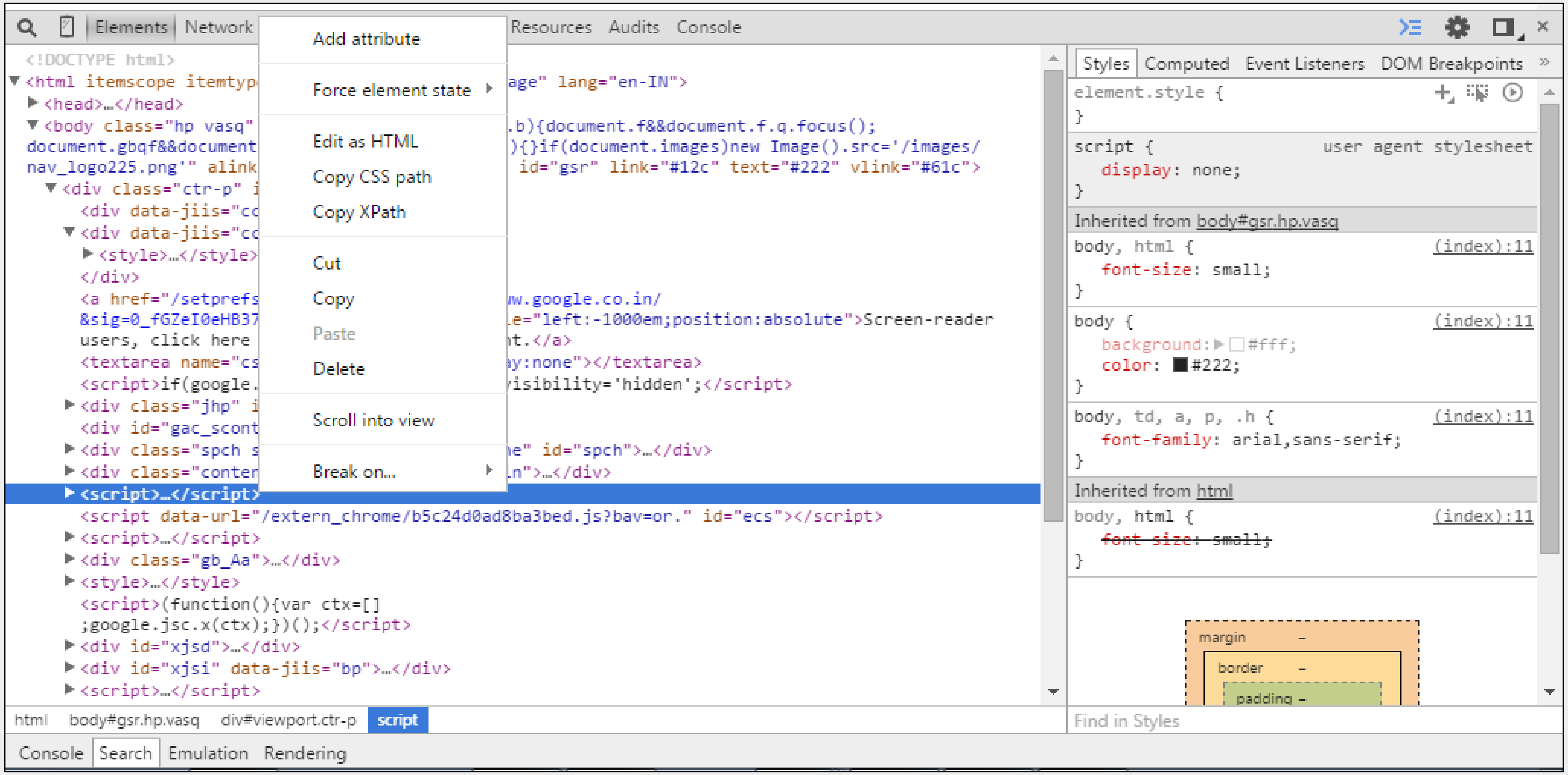
```
package org.openqa.selenium.example;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Selenium2Example  {
    public static void main(String[] args) {
        // Create a new instance of the Firefox driver
        WebDriver driver = new FirefoxDriver();
        // And now use this to visit Google
        driver.get("http://www.google.com");
        // Find the text input element by its name
        WebElement element = driver.findElement(By.name("q"));
        // Enter something to search for
        element.sendKeys("Cheese!");
        // Now submit the form. WebDriver will find the form for us from the element
        element.submit();
        // Check the title of the page
        System.out.println("Page title is: " + driver.getTitle());
        //Close the browser
        driver.quit();
    }
}
```

5.1: Testing Web Applications Using Web Driver API

Locating UI Elements-Developers Tools



```
<!DOCTYPE html>
▼<html itemscope itemtyp                       age" lang="en-IN">
  ▶<head>…</head>
  ▼<body class="hp vasq"                        .b){document.f&&document.f.q.focus();
  document.gbqf&&document                       ){}if(document.images)new Image().src='/images/
  nav_logo225.png'" alink                        id="gsr" link="#12c" text="#222" vlink="#61c">
    ▼<div class="ctr-p" i
       <div data-jiis="cc
      ▼<div data-jiis="cc
        ▶<style>…</style>
       </div>
       <a href="/setprefs                       w.google.co.in/
       &sig=0_fGZeI0eHB37                       e="left:-1000em;position:absolute">Screen-reader
       users, click here                        t.</a>
       <textarea name="cs                       y:none"></textarea>
       <script>if(google.                       visibility='hidden';</script>
      ▶<div class="jhp" i
       <div id="gac_scont
      ▶<div class="spch s                       e" id="spch">…</div>
      ▶<div class="conter                       n">…</div>
      ▶<script>…</script>
       <script data-url="/extern_chrome/b5c24d0ad8ba3bed.js?bav=or." id="ecs"></script>
      ▶<script>…</script>
      ▶<div class="gb_Aa">…</div>
      ▶<style>…</style>
       <script>(function(){var ctx=[]
       ;google.jsc.x(ctx);})();</script>
      ▶<div id="xjsd">…</div>
      ▶<div id="xjsi" data-jiis="bp">…</div>
      ▶<script>…</script>
```

```
element.style {
}
script {                          user agent stylesheet
   display: none;
}
Inherited from body#gsr.hp.vasq
body, html {                      (index):11
   font-size: small;
}
body {                            (index):11
   background: ▶ ☐ #fff;
   color: ■ #222;
}
body, td, a, p, .h {              (index):11
   font-family: arial,sans-serif;
}
Inherited from html
body, html {                      (index):11
   ~~font-size: small;~~
}
```

Navigation API

driver.get("URL")

- Required to navigate to a page
- E.g.: driver.get("http://www.google.com");
- WebDriver will wait until the page has fully loaded before returning control to your test or script
- to ensure page is fully loaded then wait commands can be used

driver.navigate().to("URL")

- E.g.: driver.navigate().to("http://www.google.com");
- Other Navigate commands
- driver.navigate().refresh();
- driver.navigate().forward();
- driver.navigate().back();

Interrogation API

driver.getTitle()

- Get the title of the current page

driver. getCurrentUrl()

- Get the current URL of the browser

driver.getPageSource()

- Get the source code of the page

Syntax:

```
        public void testTitleReliability() {
        driver.get("https://www.google.com");
                 boolean title = driver.getTitle().contains("Google");
                 if(title)
                String currentURL = driver.getCurrentUrl();
                (If you want to verify a particular text is present or not on the page,do as below)
                boolean b = driver.getPageSource().contains("your text");
                System.out.println("Expected title is present ");
                 else if(!title)
        System.out.println(" Expected title is not present"); ");
        }
```

WebElement API

findElement:

- Used to locate single element and return WebElement object of first occurrences element on web page
- If element not found, throw s exception NoSuchElementException
- Syntax: findElement(By by)

Example:

```
WebElement element = driver.findElement(By.id("Home"));
element.click();
```

WebElement API

findElements:

- Used to find multiple element on webpage, e.g.: count total number of row in table
- Returns List of WebElement object of all occurrences of element
- If element not found, returns empty List of WebElement object
- Syntax: List element = findElements(By by)

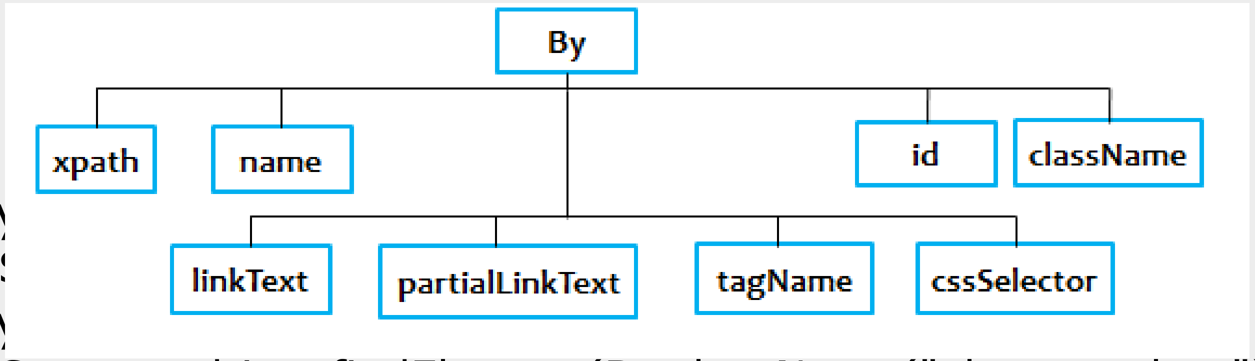
Example:

```
List {WebElement} element = driver.findElement(By.xpath("//table/tr"));
```

WebElement API (Cont..)

By:

- A collection of factory functions for creating webdriver.Locator instances



B

-

B class name

- Syntax : driver.findElement(By.className("element class"))

By name: Locates elements whose name attribute has the given value

- Syntax : driver.findElement(By.name("element name"))

WebElement API (Cont..)

- By XPath: Locates elements matching a XPath selector.
  - For example, given the selector "//div", WebDriver will search from the document root regardless of whether the locator was used with a WebElement
  - Syntax: driver.findElement(By.xpath("xpath expression"))
- By linkText :Locates link elements whose visible text matches the given string
  - Syntax : driver.findElement(By.link("link text"))
- By partialLinkText: Locates link elements whose visible text contains the given substring
  - Syntax : driver.findElement(By. partialLinkText("link text"))
- By tagName: Locates elements with a given tag name.
  - The returned locator is equivalent to using the getElementsByTagName DOM function
  - Syntax : driver.findElement(By.tagName("element html tag name"))
- By CSS Selector: Locates elements with a given tag name.
  - Syntax : driver.findElement(By.cssSelector("css selector"))

5.1: Testing Web Applications Using Web Driver API

WebElement API (Cont..)

Click():

- For Example: Login button is available on login screen
- Syntax:
- WebElement click = driver.findElement(By.xpath("//*[@id='btnLogOn']"));
  click.click();

Scenarios where Click() is used:

- “Check / Uncheck “ a checkbox
- Select a radio button

WebElement API (Cont..)

Clear():

- Function sets the value property of the element to an empty string ('')

Syntax:

```
        driver.findElement(By.xpath("//*[@id=“textBox”]")).clear();
```

SendKeys():

- Method is used to simulate typing into an element, which may set its value

Syntax:

```
        driver.findElement(By.id(“NameTextBox")).sendKeys(“Rahul”);
```

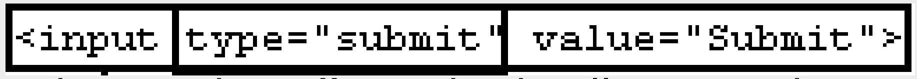
5.1: Testing Web Applications Using Web Driver API

WebElement API (Cont..)

- SendKeys():
  - Scenarios where sendKeys() is used:
    - Sending special characters (Enter , F5, Ctrl, Alt etc..)
    - Key events to WebDriver
    - Uploading a file

    Syntax:

    ```
    //Sending Ctrl+A
     driver.findElement(By.xpath("//body")).sendKeys(Keys.chord(Keys.CONTROL,
    "a"));
    //Sending pagedown key from keyboard
    driver.findElement(By.id("name")).sendKeys(Keys.PAGE_DOWN);
    //Sending space key
    driver.findElement(By.id("name")).sendKeys(Keys.SPACE);
    ```
- Submit():
  - If form has submit button which has type = “submit” instead of type = "button" then .submit() method will work



```
<input type="submit" value="Submit">
```

  - If button Is not Inside <form> tag then .submit() method will not work.

- **Syntax:**
- driver.findElement(By.xpath("//input[@name='Company']")).submit();

WebElement API (Cont..)

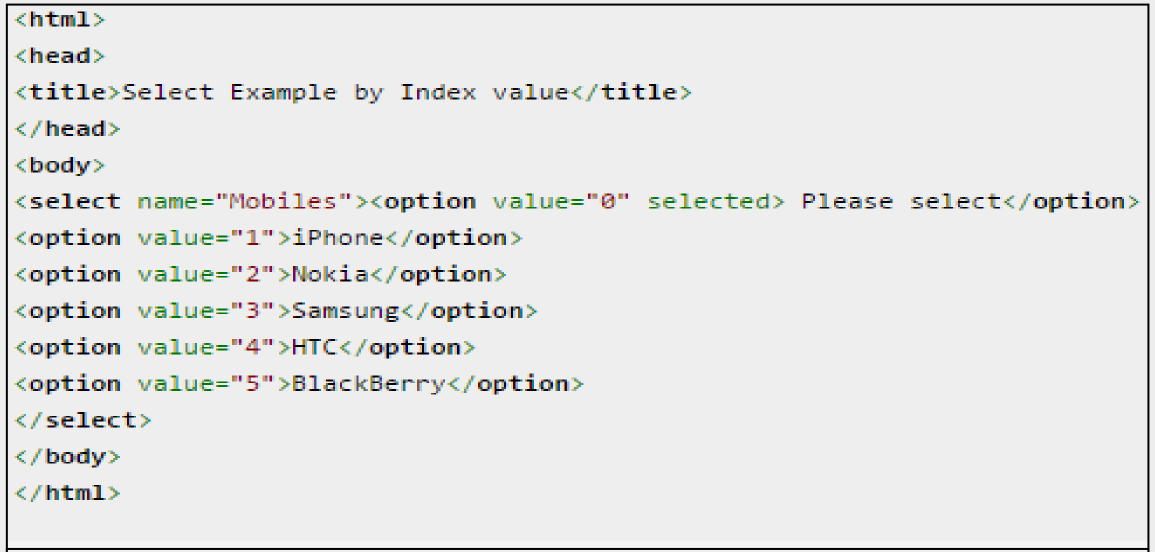
Select:

- WebDriver’s support classes
- Used to work with Dropdowns

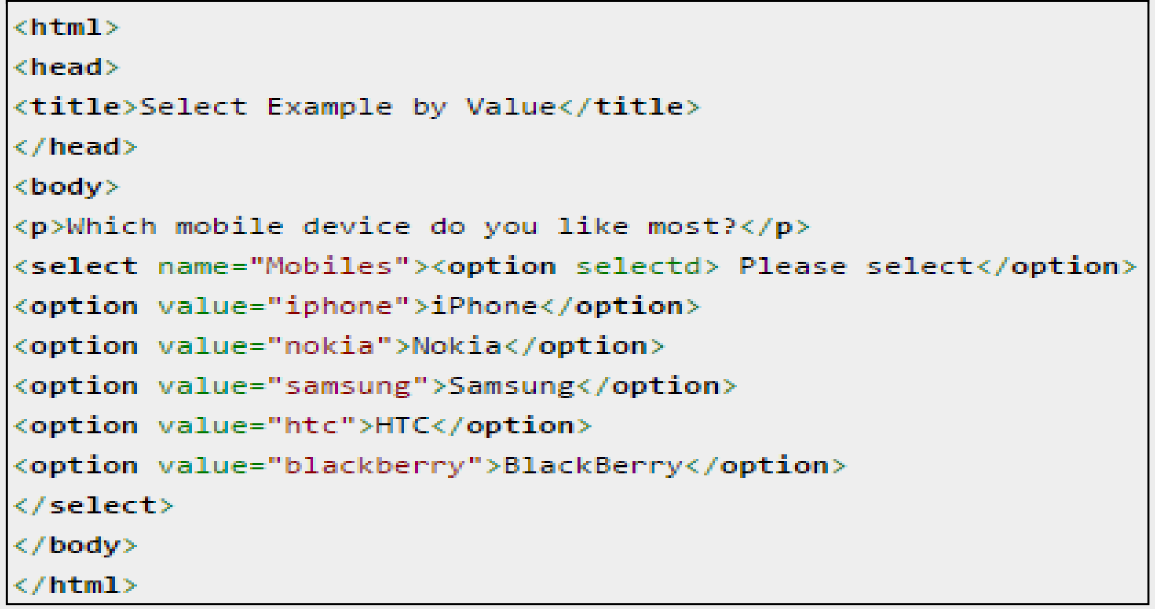
- Method Name: selectByIndex
- Syntax: select.selectByIndex(Index);

- Method Name: selectByValue
- Syntax: select.selectByValue(Value);

- Method Name: selectByVisibleText
- Syntax: select.selectByVisibleText(Text);



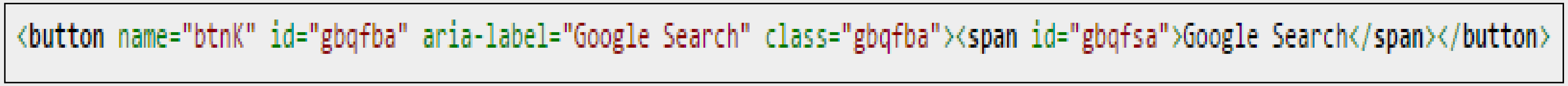
```
<html>
<head>
<title>Select Example by Index value</title>
</head>
<body>
<select name="Mobiles"><option value="0" selected> Please select</option>
<option value="1">iPhone</option>
<option value="2">Nokia</option>
<option value="3">Samsung</option>
<option value="4">HTC</option>
<option value="5">BlackBerry</option>
</select>
</body>
</html>
```



```
<html>
<head>
<title>Select Example by Value</title>
</head>
<body>
<p>Which mobile device do you like most?</p>
<select name="Mobiles"><option selectd> Please select</option>
<option value="iphone">iPhone</option>
<option value="nokia">Nokia</option>
<option value="samsung">Samsung</option>
<option value="htc">HTC</option>
<option value="blackberry">BlackBerry</option>
</select>
</body>
</html>
```

WebElement API (Cont..)

- getText():
  - Get the text content from a DOM-element found by given selector
  - Make sure the element you want to request the text from is interact able otherwise empty string is returned
    - Syntax:
    - WebElement TxtBoxContent = driver.findElement(By.id(“WebelementID”));
    - TxtBoxContent.getText();
- getAttribute():
  - getText() will only get the inner text of an element
  - To get the value, you need to use getAttribute(“attribute name”)
  - Attribute name can be class, id, name, status etc



```
<button name="btnK" id="gbqfba" aria-label="Google Search" class="gbqfba"><span id="gbqfsa">Google Search</span></button>
```

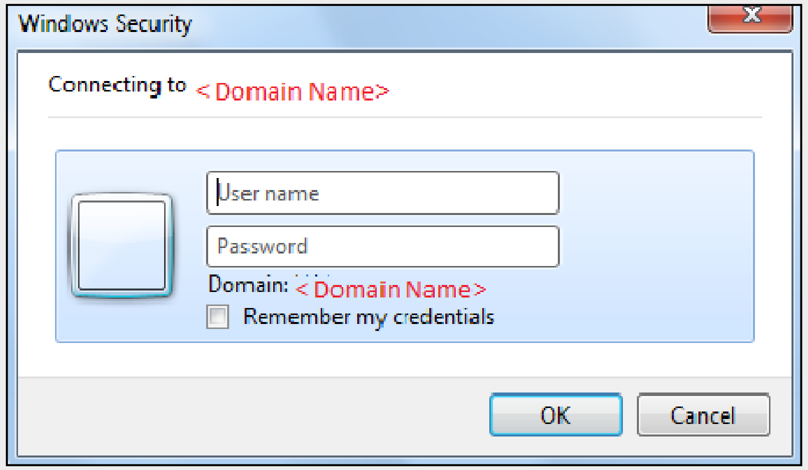
- Syntax:
  - WebElement TxtBoxContent = driver.findElement(By.id(WebelementID));
    System.out.println("Printing " + TxtBoxContent.getAttribute(“class"));

5.1: Testing Web Applications Using Web Driver API

Handling Popup Dialogs and Alerts

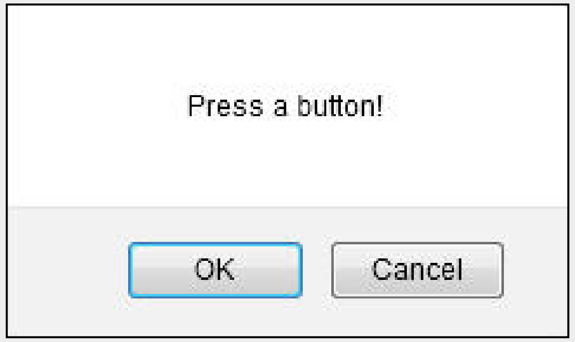
Two types of alerts:

- Windows based alert pop ups
  - Selenium will not be able to recognize it, since it is an OS-level dialog



Web based alert pop ups

- Can be Alert box/ Pop up box/ confirmation Box/ Prompt/ Authentication Box
- Alert interface gives us following methods to deal with the alert
  - accept() : To accept the alert
  - dismiss() : To dismiss the alert
  - getText() : To get the text of the alert
  - sendKeys() : To write some text to the alert



5.1: Testing Web Applications Using Web Driver API

Windows

Multiple windows are handled by switching the focus from one window to another

Syntax:

```
    // Opening site
        driver.findElement(By.xpath("//img[@alt='SeleniumMasterLogo']")).click();
    // Storing parent window reference into a String Variable
        String Parent_Window = driver.getWindowHandle();
    // Switching from parent window to child window
        for (String Child_Window : driver.getWindowHandles())
        {
                driver.switchTo().window(Child_Window);
    // Performing actions on child window
                driver.findElement(By.id("dropdown_txt")).click();
                driver.findElement(By.xpath("//*[@id='anotherItemDiv']")).click();
        }
    //Switching back to Parent Window
        driver.switchTo().window(Parent_Window);
    //Performing some actions on Parent Window
        driver.findElement(By.className("btn_style")).click();
```

Alerts

- Present in the **org.openqa.selenium.Alert** package
- Syntax:

```
Alert simpleAlert = driver.switchTo().alert();      //switch from main window to an alert
String alertText = simpleAlert.getText();            //To get the text present on alert
System.out.println("Alert text is " + alertText);
//Simple alert
simpleAlert.accept();                                 //To click on 'Ok'/'Yes' on Alert

                    OR
//Confirmation Alert
simpleAlert. dismiss();                             //To click on 'Cancel'/'No' on Alert

                    OR
//Prompt Alerts
simpleAlert. sendKeys("Accepting the alert"); //Send some text to the alert
```

Why synchronization is important

- “Mechanism which involves more than one components to work parallel with each other”
- Every time user performs an operation on the browser, one of the following happens:
  - The request goes all the way to server and entire DOM is refreshed when response comes back
  - The request hits the server and only partial DOM gets refreshed (Ajax requests or asynchronous JavaScript calls)
  - The request is processed on the client side itself by JavaScript functions
- So if we think about the overall workflow, there is a need of certain synchronization that happens between the client(aka. browser) and the server (the url)

Using Explicit & Implicit Wait

- Implicit Wait

- Element Synchronization
  - Default element existence timeout can be set
  - Below statement will set the default object synchronization timeout as 20
  - Means that selenium script will wait for maximum 20 seconds for element to exist
  - If Web element does not exist within 20 seconds, it will throw an exception

- driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

Using Explicit & Implicit Wait

- Explicit Wait
  - Specific condition synchronization
- Instruct selenium to wait until element is in expected condition
  - Syntax:

```
WebDriverWait w = new WebDriverWait(driver,20);
w.ignoring(NoSuchElementException.class);
WebElement P = null;
//below statement will wait until element becomes visible
P=w.until(ExpectedConditions.visibilityOfElementLocated(By.id("x")));
//below statement will wait until element becomes clickable.
P= w.until(ExpectedConditions.elementToBeClickable(By.id("ss")));
```

JavaScript Executor

- An interface which provides mechanism to execute Javascript through selenium driver
- Provides “executescript” & "executeAsyncScript" methods, to run JavaScript in the context of the currently selected frame or window
- Used to enhance the capabilities of the existing scripts by performing Javascript injection into our application under test
- Package
  import org.openqa.selenium.JavascriptExecutor;

  Syntax

  ```
  JavascriptExecutor js = (JavascriptExecutor) driver;
  js.executeScript(Script,Arguments);
  ```

  Script - The JavaScript to execute
  Arguments - The arguments to the script.(Optional)

JavaScript Executor(Scenarios)

- How to generate Alert Pop window in selenium?
- Code:

```
JavascriptExecutor js = (JavascriptExecutor)driver
Js.executeScript("alert('hello world');");
```

- How to click a button in Selenium WebDriver using JavaScript?
  Code:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
js.executeScript("arguments[0].click();", element);
```

- How to refresh browser window using Javascript ?
- Code:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
driver.executeScript("history.go(0)");
```

JavaScript Executor(Scenarios)

- How to get innertext of the entire webpage in Selenium?
  Code:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
String sText = js.executeScript("return document.documentElement.innerText;").toString();
```

- How to get the Title of our webpage ?
  Code:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
String sText = js.executeScript("return document.title;").toString();
```

JavaScript Executor(Scenarios)

- How to perform Scroll on application using Selenium?
  Code:
  ```
  JavascriptExecutor js = (JavascriptExecutor)driver; //Vertical scroll - down by 50 pixels
  js.executeScript("window.scrollBy(0,50)");
  ```

- Note: for scrolling till the bottom of the page we can use the code:
  ```
  js.executeScript("window.scrollBy(0,document.body.scrollHeight)");
  ```

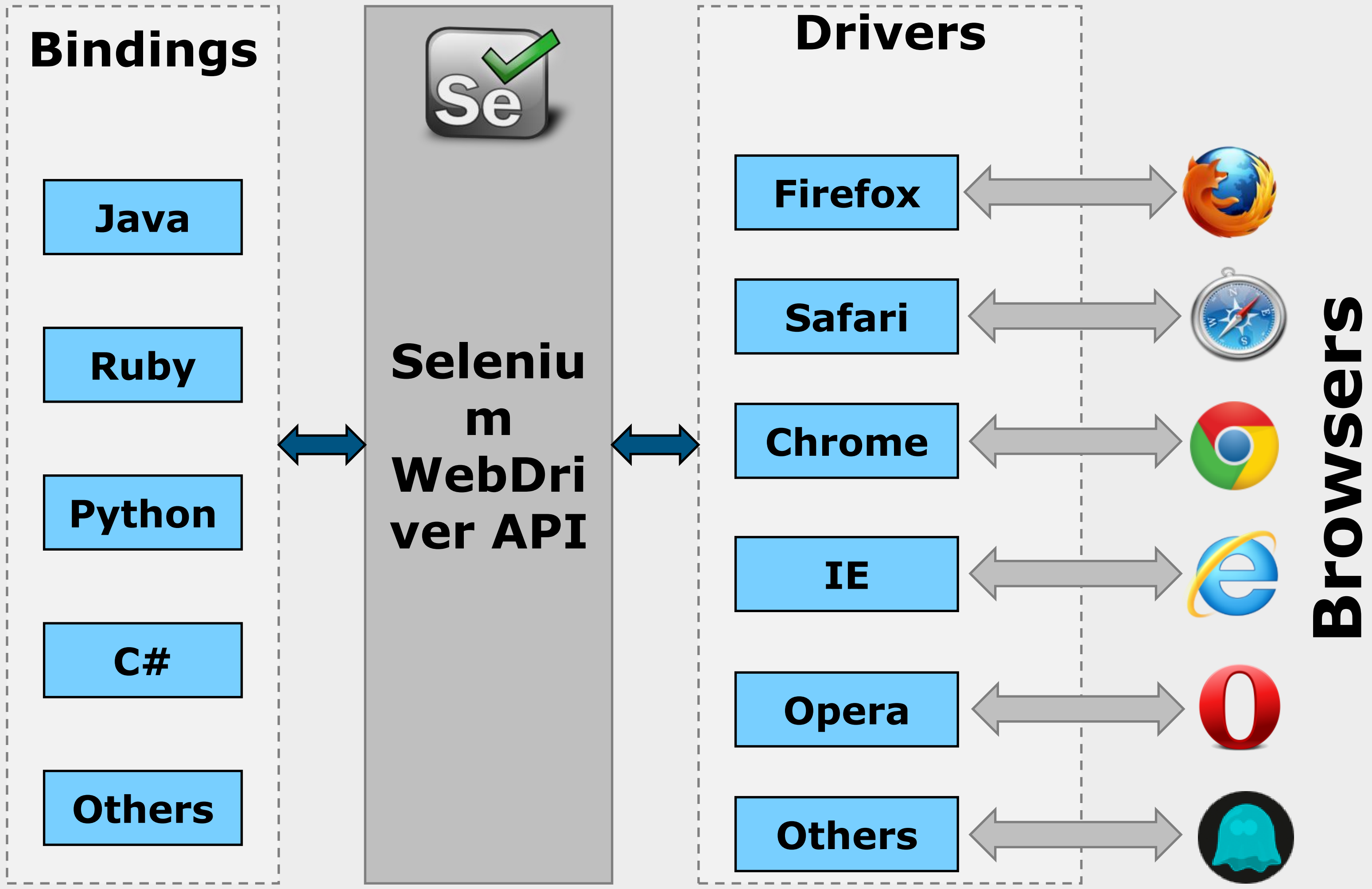
5.1: Testing Web Applications Using Web Driver API

JavaScript Executor(Scenarios)

- How to click on a Sub Menu which is only visible on mouse hover on Menu?
  Code:
  ```
  JavascriptExecutor js = (JavascriptExecutor)driver;
   //Hover on Automation Menu on the Menu Bar
    js.executeScript("$('ul.menus.menu-secondary.sf-js-enabled.sub-menu li').hover()");
  ```

- How to navigate to different page using Javascript?
  Code:
  ```
  JavascriptExecutor js = (JavascriptExecutor)driver;   //Navigate to new Page
  js.executeScript("window.location = 'https://www.facebook.com/uftHelp'");
  ```

Selenium - How it works?



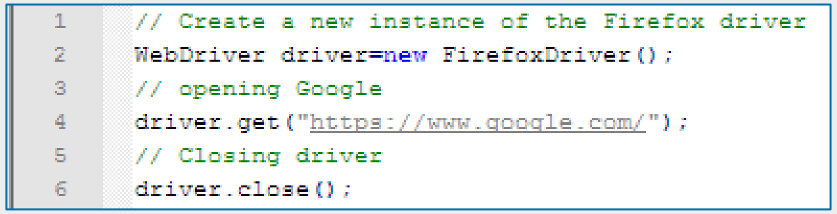
Different Drivers

Firefox:
WebDriver driver = new FirefoxDriver();
IE:
WebDriver driver = new InternetExplorerDriver();
Chrome:
WebDriver driver = new ChromeDriver();
Safari:
WebDriver driver = new SafariDriver();
Opera:
WebDriver driver = new OperaDriver()
GhostDriver and PhantomJs:
WebDriver driver = new PhantomJSDriver()

7.1: Selenium Web Driver – Advance

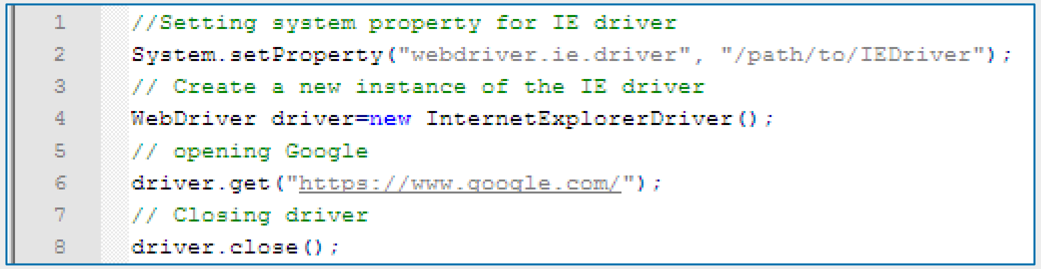
Different Drivers (Contd.)

Firefox Driver:



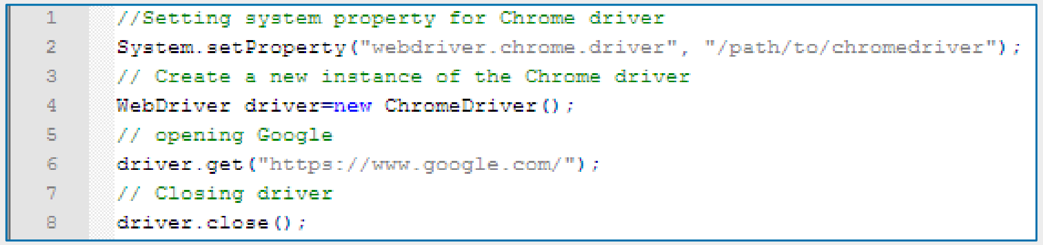
```java
// Create a new instance of the Firefox driver
WebDriver driver=new FirefoxDriver();
// opening Google
driver.get("https://www.google.com/");
// Closing driver
driver.close();
```

IE



```java
//Setting system property for IE driver
System.setProperty("webdriver.ie.driver", "/path/to/IEDriver");
// Create a new instance of the IE driver
WebDriver driver=new InternetExplorerDriver();
// opening Google
driver.get("https://www.google.com/");
// Closing driver
driver.close();
```

Ch



```java
//Setting system property for Chrome driver
System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");
// Create a new instance of the Chrome driver
WebDriver driver=new ChromeDriver();
// opening Google
driver.get("https://www.google.com/");
// Closing driver
driver.close();
```

Different Drivers (Contd.)

Headless Browser

- Web browser without a graphical user interface
- Normally, interaction with a website are done with mouse and keyboard using a browser with a GUI
- While most headless browser provides an API to manipulate the page/DOM, download resources etc.
- So instead of, for example, actually clicking an element with the mouse, a headless browser allows you to click an element by code
- Headers, Local storage and Cookies work the same way
- List of Headless Browsers
  - PhanthomJS
  - HtmlUnit
  - TrifleJS
  - Splash

Different Drivers (Contd.)

PhanthomJS (Headless Browser)

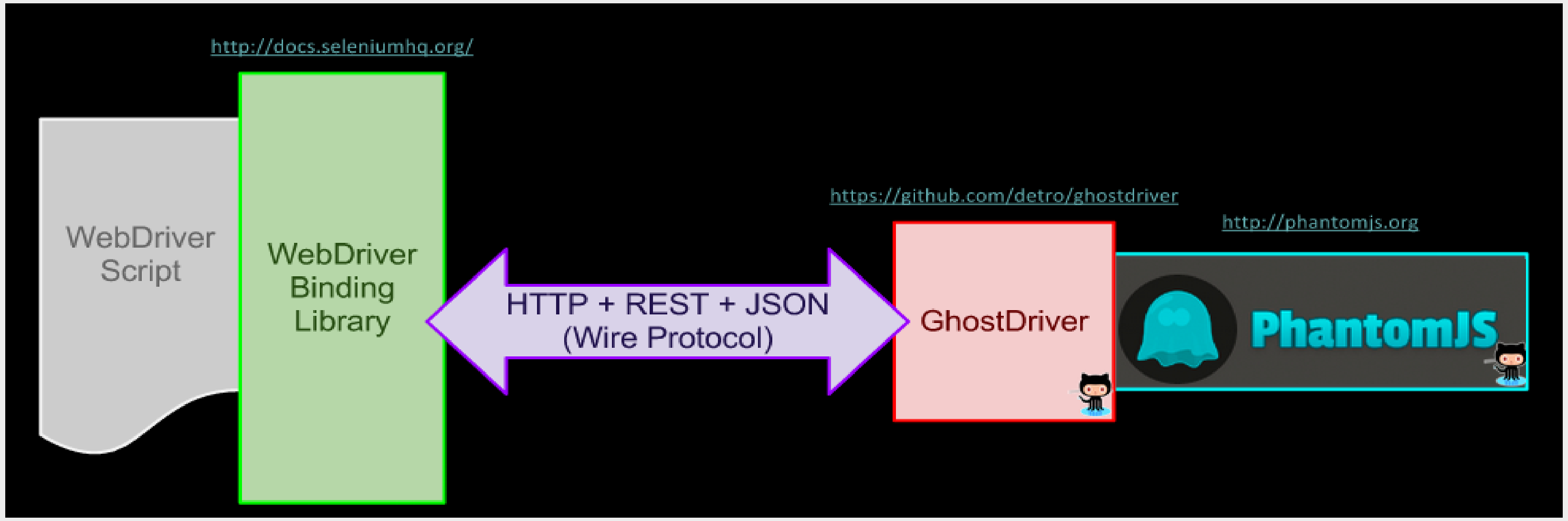
- HEADLESS WEBSITE TESTING
  - Headless Web Kit with JavaScript API
- SCREEN CAPTURE
  - Programmatically capture web contents, including SVG and Canvas
- PAGE AUTOMATION
  - Access and manipulate webpages with the standard DOM API, or with usual libraries like jQuery
- Example of interacting with a page using PhantomJS:

```
page.evaluate(function() {
        //Fill in form on page
        document.getElementById('Name').value = 'John Doe';
        document.getElementById('Email').value =
  'john.doe@john.doe';
        //Submit
        $('#SubmitButton').click();
    });
```

Different Drivers (Contd.)

GhostDriver(Headless Browser)

- Pure JavaScript implementation of the WebDriver Wire Protocol for PhantomJS Remote
- WebDriver that uses PhantomJS as back-end


http://docs.seleniumhq.org/
WebDriver Script
WebDriver Binding Library
HTTP + REST + JSON (Wire Protocol)
https://github.com/detro/ghostdriver
GhostDriver
http://phantomjs.org
PhantomJS

Different Drivers (Contd.)

Mobile Browsers

- Mobile web browsers differ greatly in terms of features offered an operating systems supported
- Best can display most websites and offer page zoom and keyboard shortcuts, while others can only display websites optimizes for mobile devices
- Appium and Selendroid are the two cross browser mobile automation tool

Appium(Mobile Browser)

- Open-source tool for automating native, mobile web, and hybrid applications on iOS and Android platforms, which is handled by a Appium node.js server.
- Cross-platform which allows to write tests against multiple platforms (iOS, Android), using the same API
- Enables code reuse between iOS and Android test suites

Different Drivers (Contd.)

Selendroid

Test automation framework which is used Android native & hybrid applications (apps) and mobile web

- Full compatibility with the JSON Wire Protocol
- No modification of app under test required in order to automate it
- Testing the mobile web using built in Android driver webview app
- UI elements can be found by different locator types
- Selendroid can interact with multiple Android devices (emulators or hardware devices) at the same time
- Existing emulators are started automatically
- Supports hot plugging of hardware devices
- Full integration as a node into Selenium Grid for scaling and parallel testing

Remote Web Driver

Implementation class of the WebDriver interface that a test script developer can use to execute their test scripts via the Remote WebDriver server on a remote machine

Needs to be configured so that it can run your tests on a separate machine

If **driver is not Remote WebDriver**, communication to the web browser is local. So driver will be used as below:

```
Webdriver driver = new FirefoxDriver();
```

using driver will access Firefox on the local machine, directly

If **Remote WebDriver**, needs location Selenium Server (Grid) web browser

For example,

```
WebDriver driver = new RemoteWebDriver(new
URL("http://localhost:4444/wd/hub"),    DesiredCapabilities.firefox());
```

Capabilities and Profile Setting

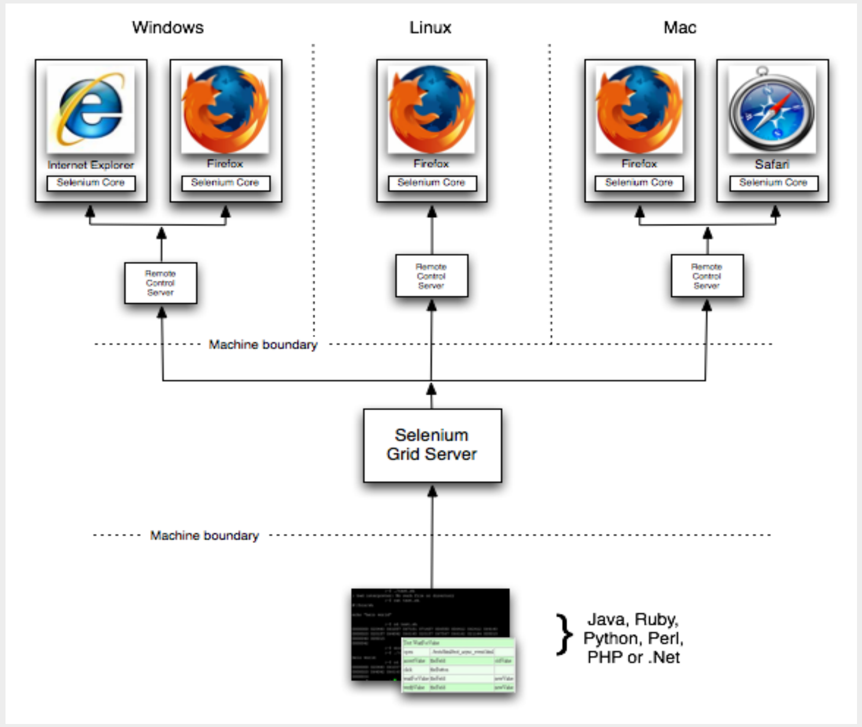
Capabilities describes a series of key/value pairs that encapsulate aspects of a browser
DesiredCapabilities set properties for the WebDriver
Profile used to create custom Firefox profile and use it with desired capabilities

Example to use Firefox profile with desired capabilities:

```
DesiredCapabilities dc=DesiredCapabilities.firefox();
FirefoxProfile profile = new FirefoxProfile();
dc.setCapability(FirefoxDriver.PROFILE, profile);
Webdriver driver =  new FirefoxDriver(dc);
```

Selenium Grid

Allows you run your tests on different machines against different browsers in parallel. That is, running multiple tests at the same time against different machines running different browsers and operating systems. Essentially, Selenium-Grid support distributed test execution. It allows for running your tests in a distributed test execution environment.



Summary

In this lesson, you have learnt

- Multiple windows are handled by switching the focus from one window to another.
- By is a collection of factory functions for creating webdriver.Locator instances.
- Alert contains methods for dismissing, accepting, inputting, and getting text from alert prompts.
- Explicit synchronization points are inserted in the script using WebDriverWait class.
- Each and every time when there is need to match speed of the application and speed of test execution we have to use thread.sleep().
- The implicit wait will not wait for the entire time that is specified, rather it will only wait, until the entire page is loaded.

Summary

In this lesson, you have learnt

- An interface which provides mechanism to execute Javascript through selenium driver
- Used to click on a Sub Menu which is only visible on mouse hover on Menu
- Used to to get innertext of the entire webpage in Selenium
- Used to navigate to different page using Javascript
- Used to click a button in Selenium WebDriver using JavaScript

Summary

In this lesson, you have learnt

In this lesson, you have understood that how the selenium works and interacts with client server.

Different drivers that are available for selenium-Chrome , IE ,Firefox ,headless browsers and mobile browsers.

In remote webdriver the server will always run on the machine with the browser you want to test. And ,there are two ways to user the server: command line or configured in code.

Summary

Capabilities gives facility to set the properties of browser. Such as to set BrowserName, Platform, Version of Browser.

There are two reasons why you might want to use Selenium-Grid.

- To run your tests against multiple browsers, multiple versions of browser, and browsers running on different operating systems.
- To reduce the time it takes for the test suite to complete a test pass.

Review Question

Question 1

- Select which is NOT an Explicit Wait
  - VisibilityOfElementLocated
  - ElementToBeClickable
  - PageLoadTimeout
  - None of the above

Question 2: True/False

- The syntax is correct:
- Syntax : driver.findElement(By. PartialLinkText("link text"));

Question 3: Fill in the Blanks

- findElements is used to find _______ element on webpage

Review Question

Question 4: True/False

- The syntax is correct:

  Syntax :

  JavascriptExecutor js = (JavascriptExecutor)driver;

Question 5: Fill in the Blanks

- An interface which provides mechanism to execute ____________ through selenium driver

Review Question

Question 6:

- PhanthomJS is
- Chrome Driver
- Mobile Browser
- Headless Browser
- Firefox Driver

Question 7: True/False

- Firefox saves your information such as cookies and browser history in a file called your profile.

Question 8: Fill in the Blanks

- Client driver will send the program that is written in eclipse IDE as __________ and send it to Selenium server