Webdriver snippet

Instantiate Specific Browser Or Client

You may need this code in every selenium script. So I thought of keeping it here handy for the popular ones.  
  
**Firefox Driver**

WebDriver driver = new FirefoxDriver();

**Chrome Driver**

WebDriver driver = new ChromeDriver();

**Safari Driver**

WebDriver driver = new SafariDriver();

**Internet Explorer Driver**  
**WebDriver driver = new InternetExplorerDriver();**  
**Android Driver**

Check If An Element Exists

You may need to perform a action based on a specific web element being present on the web page. You can use below code snippet to check if a element with id “element-id” exists on web page. 

driver.findElements(By.id("element-id")).size()!=0

How To Check If An Element Is Visible With WebDriver

The above mentioned method may be good to check if a elemet exists on page. However sometimes a element may be not visible, therefore you can not perform any action on it. You can check whether an element is visible or not using below code.

WebElement element = driver.findElement(By.id("element-id"));

if(element instanceof RenderedWebElement) {

System.out.println("Element visible");

} else {

System.out.println("Element Not visible");

}

Refresh Page

This may be required often. Just a simple refresh of the page equivalent to a browser refresh.

driver.navigate().refresh();

Navigate Back And Forward On The Browser

Navigating the history of borwser can be easily done using below two methods. The names are self explanatory.

//Go back to the last visited page

driver.navigate().back();

//go forward to the next page

driver.navigate().forward();

Wait For Element To Be Available

The application may load some elements late and your script needs to stop for the element to be available for next action. You can perform this check using below code.  
In below code, the script is going to wait maximum 30 seconds for the element to be available. Feel free to change the maximum number per your application needs.

WebDriverWait wait = new WebDriverWait(driver, 30);

WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("id123")));

Focus On A Input Element On Page

Doing focus on any element can be easily done by clicking the mouse on the required element. However when you are using selenium you may need to use this workaround instead of mouse click you can send some empty keys to a element you want to focus.

WebElement element = driver.findElement(By.id("element-id"));

//Send empty message to element for setting focus on it.

element.sendKeys("");

Overwrite Current Input Value On Page

The sendKeys method on WebElement class will append the value to existing value of element. If you want to clear the old value. You can use clear() method. 

WebElement element = driver.findElement(By.id("element-id"));

element.clear();

element.sendKeys("new input value");

Mouseover Action To Make Element Visible Then Click

When you are dealing with highly interactive multi layered menu on a page you may find this useful. In this scenario, an element is not visible unless you click on the menu bar. So below code snippet will accomplish two steps of opening a menu and selecting a menu item easily.

Actions actions = new Actions(driver);

WebElement menuElement = driver.findElement(By.id("menu-element-id"));

actions.moveToElement(menuElement).moveToElement(driver.findElement(By.xpath("xpath-of-menu-item-element"))).click();

Extract CSS Attribute Of An Element

This can be really helpful for getting any CSS property of a web element.  
For example, to get background color of an element use below snippet

String bgcolor = driver.findElement(By.id("id123")).getCssValue("background-color");

and to get text color of an element use below snippet

String textColor = driver.findElement(By.id("id123")).getCssValue("color");

Find All Links On The Page

A simple way to extract all links from a web page.

List link = driver.findElements(By.tagName("a"));

Take A Screenshot On The Page

The most useful one. Selenium can capture the screenshot of any error you want to record. You may want to add this code in your exception handling logic.

File snapshot =((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

Execute A JavaScript Statement On Page

If you love JavaScript, you are going to love this. This simple JavascriptExecutor can run any javascript code snippet on browser during your testing. In case you are not able to find a way to do something using web driver, you can do that using JS easily.  
Below code snippet demonstrates how you can run a alert statement on the page you are testing.

JavascriptExecutor jsx = (JavascriptExecutor) driver;

jsx.executeScript("alert('hi')");

Upload A File On A Page

Uploading a file is a common use case. As of now there is not webdriver way to do this, however this can be easily done with the help of JavascriptExecutor and little bit of JS code. 

String filePath = "path\\to\\file\for\\upload";

JavascriptExecutor jsx = (JavascriptExecutor) driver;

jsx.executeScript("document.getElementById('fileName').value='" + filePath + "';");

Scroll Up, Down Or Anywhere On A Page

Scrolling on any web page is required almost always. You may use below snippets to do scrolling in any direction you need.

JavascriptExecutor jsx = (JavascriptExecutor) driver;

//Vertical scroll - down by 100 pixels

jsx.executeScript("window.scrollBy(0,100)", "");

//Vertical scroll - up by 55 pixels (note the number is minus 55)

jsx.executeScript("window.scrollBy(0,-55)", "");

//Horizontal scroll - right by 20 pixels

jsx.executeScript("window.scrollBy(20,0)", "");

//Horizontal scroll - left by 95 pixels (note the number is minus 95)

jsx.executeScript("window.scrollBy(-95,0)", "");

Get HTML Source Of A Element On Page

If you want to extract the HTML source of any element, you can do this by some simple Javascript code.

JavascriptExecutor jsx = (JavascriptExecutor) driver;

String elementId = "element-id";

String html =(String) jsx.executeScript("return document.getElementById('" + elementId + "').innerHTML;");

Switch Between Frames In Java Using Webdriver

Multiple iframes are very common in recent web applications. You can have your webdriver script switch between different iframes easily by below code sample

WebElement frameElement = driver.findElement(By.id("id-of-frame"));

driver.switchTo().frame(frameElement);

Web driver is really a powerful functional testing tool. I have not looked into any other libraries and tools after I have started using Webdriver ( selenium ) . The learning curve is really short for any java developer.   
  
Earlier version of Selenium required a separate remote server to run the scripts. The latest version does not even require that. You can directly run a selenium webdriver script using webdriver library. This makes life of a developer even more easy.  
  
For a quick start you may want to try the Selenium IDE. It has recording feature, that can record user actions on browser and then generate code in any supported language like Java, Python, Ruby and .NET  
  
I hope you will like these code snippets. I have used all of them at multiple places and have them handy.   
  
Can you think about any other common scenario that you used in web driver? Don’t forget to share with us in comments section below.