JavaScript and Selenium JavaScriptExecutor

What is JavaScript?

JavaScript is the preferred language inside the browser to interact with HTML dom. This means that a Browser has JavaScript implementation in it and understands the JavaScript commands. You can disable it using browser options in your browser.

JavaScript was also the language that was used by earlier Selenium versions, it is still used by Selenium webdriver to perform some actions. For e.g. Selenium has Xpath implemented in JavaScript for IE, to overcome the lack of Xpath engine in IE. Understanding JavaScript is fun and it can enable you to do lot of cool things that otherwise you may find tricky. Lets understand how

WebDriver gives you a method called Driver.executeScript which executes the JavaScript in context of the loaded browser page.

***What can JavaScript and Selenium JavaScriptExecutor do for us?***

First thing to know that the ***JavaScriptExecutor***comes separately and also comes under the WebDriver but both do the same thing. With in the WebDriver, it is named as ***ExecuteScript***. Below you will find the examples for both. You can use any of it.

Well practically anything you want to do in browser. Lets see how, we try to parallel selenium commands with JavaScript.

**Evaluating Xpaths**

You can use javascripts to find an element by Xpath. This is a brilliant way to skip internal Xpath engines. Here is what you have to do

|  |  |
| --- | --- |
| 1 | WebElement value = (WebElement) Driver.executeScript("return document.evaluate( '//body//div/iframe' ,document, null, XPathResult.FIRST\_ORDERED\_NODE\_TYPE, null ).singleNodeValue;"); |

Notice the command document.evaluate(). This is the Xpath evaluator in javascript. Signature of the function is

|  |  |
| --- | --- |
| 1 | document.evaluate( xpathExpression, contextNode, namespaceResolver, resultType, result ); |

* ***xpathExpression:****A string containing the XPath expression to be evaluated.*
* ***contextNode:****A node in the document against which the xpathExpression should be evaluated, including any and all of its child nodes. The document node is the most commonly used.*
* ***NamespaceResolver:****A function that will be passed any namespace prefixes contained within xpathExpression which returns a string representing the namespace URI associated with that prefix.*
* ***resultType:****A constant that specifies the desired result type to be returned as a result of the evaluation. The most commonly passed constant is XPathResult.ANY\_TYPE which will return the results of the XPath expression as the most natural type.*
* ***result:****If an existing XPathResult object is specified, it will be reused to return the results. Specifying null will create a new XPathResult object*

*Finding Element*

Using selenium we can find any element on page using something similar to

|  |  |
| --- | --- |
| 1 | Driver.findElementById("Id of the element"); |

we can do the same thing using JavaScript like so

|  |  |
| --- | --- |
| 1  2  3 | WebElement searchbox = null;    searchbox = (WebElement) ff.executeScript("return document.getElementById('gsc-i-id1');", searchbox); |

you will get the element in the searchbox variable.

**Finding elements** Selenium command

|  |  |
| --- | --- |
| 1 | List\&lt;WebElement\&gt; elements = List\&lt;WebElement\&gt; Driver.findElementByTagName("div"); |

**using JavaScript**

|  |  |
| --- | --- |
| 1 | List elements = (List') ff.executeScript("return document.getElementsByTagName('div');", searchbox); |

***Changing style attribute of the element*s**

You can change the style property of elements to change the rendered view of the element. For e.g this is how you can create a border around the about me element on toolsQa home page. See the below images of same element before and after creating border

Here is the code.

|  |  |
| --- | --- |
| 1 | Driver.executeScript("document.getElementById('text-4').style.borderColor = 'Red'"); |

Coloring elements can also help us take screenshots with visual markers to identify problematic elements.

***Getting Element Attributes***

You can get any value of valid attributes of the element. For eg

|  |  |
| --- | --- |
| 1 | Driver.findElementById("gsc-i-id1").getAttribute("Class"); |

this code will get the value of class attribute of an element with id = gsc-i-id1.

In Javascript same thing can be executed using

|  |  |
| --- | --- |
| 1 | String className = Driver.executeScript("return document.getElementById('gsc-i-id1').getAttribute('class');")); |

***Total Frames in Browser***

Lets say you want to know  total number of frames inside a webpage, including the Iframes. You cant do it using selenium directly, you may need to create your own logic to parse and find frames. However in JavaScript is simple and is done like this.

|  |  |
| --- | --- |
| 1 | ff.executeScript("document.frames.length;"); |

Be ware of this one. It only works in IE, cant get it to work for Mozilla.

***Adding an Element in DOM***

Just for fun if you want to add an element in the DOM. Lets add a button html object

|  |  |
| --- | --- |
| 1  2  3 | Driver.executeScript("var btn=document.createElement('BUTTON');"                         + "document.body.appendChild(btn);"); |

***Size of Window***

Finding the size of inner browser window. It is the size of the window in which you see the web page.

|  |  |
| --- | --- |
| 1  2  3 | Driver.executeScript("return window.innerHeight;")    Driver.executeScript("return window.innerWidth;") |

***Navigating to different Page***

|  |  |
| --- | --- |
| 1 | ff.executeScript("window.location = 'http://yahoo.com'"); |

What is JavaScriptExecutor?

***JavaScriptExecutor***is an interface which provides mechanism to execute Javascript through selenium driver. It provides “***executescript***” & “***executeAsyncScript***” methods, to run JavaScript in the context of the currently selected frame or window.

***Generate Alert Pop Window***

|  |  |
| --- | --- |
| 1  2 | JavascriptExecutor js = (JavascriptExecutor)driver;  Js.executeScript("alert('hello world');"); |

***Click Action***

|  |  |
| --- | --- |
| 1  2 | JavascriptExecutor js = (JavascriptExecutor)driver;  js.executeScript("arguments[0].click();", element); |

***Refresh Browser***

|  |  |
| --- | --- |
| 1  2 | JavascriptExecutor js = (JavascriptExecutor)driver;  driver.executeScript("history.go(0)"); |

***Get InnerText of a Webpage***

|  |  |
| --- | --- |
| 1  2 | JavascriptExecutor js = (JavascriptExecutor)driver;  string sText =  js.executeScript("return document.documentElement.innerText;").toString(); |

***Get Title of a WebPage***

|  |  |
| --- | --- |
| 1  2 | JavascriptExecutor js = (JavascriptExecutor)driver;  string sText =  js.executeScript("return document.title;").toString(); |

***Scroll Page***

|  |  |
| --- | --- |
| 1  2  3 | JavascriptExecutor js = (JavascriptExecutor)driver;    //Vertical scroll - down by 150  pixels    js.executeScript("window.scrollBy(0,150)"); |

Similarly you can execute practically any JavaScript command using Selenium. I hope this blog gives you an idea of what can be done and you can take it forward by playing around with different JavaScripts.