## ***What is Selenium?***

* Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms.
* It is quite similar to HP Quick Test Pro (QTP now UFT) only that Selenium focuses on automating web-based applications.
* Testing done using Selenium tool is usually referred as Selenium Testing.

**It has four components.**

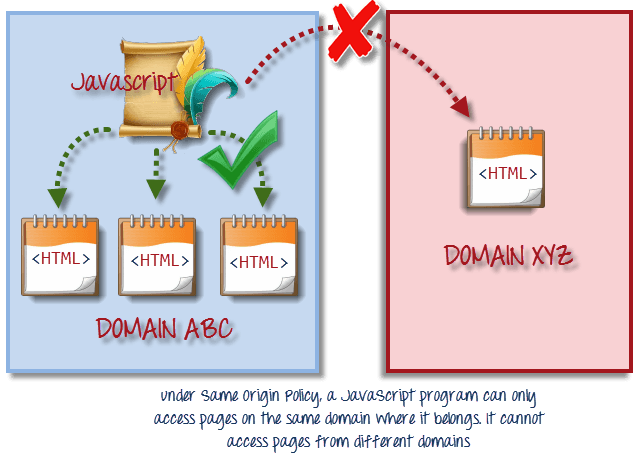
* Selenium Integrated Development Environment (IDE)
* Selenium Remote Control (RC)
* WebDriver
* Selenium Grid



* Selenium was **created by Jason Huggins in 2004**. An engineer at ThoughtWorks, he was working on a web application that required frequent testing.
* He created a[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)program that would automatically control the browser's actions. He named this program as the "**JavaScriptTestRunner**.
* Seeing potential in this idea to help automate other web applications, he made JavaScriptRunner open-source which was later re-named as **Selenium Core**.

## ***The Same Origin Policy Issue***

**Same Origin policy prohibits JavaScript code from accessing elements from a domain that is different from where it was launched**.



* Testers needed to install local copies of both Selenium Core (a JavaScript program) and the web server containing the web application being tested so they would belong to the same domain.

## ***Birth of Selenium Remote Control (Selenium RC)***

* ThoughtWork's engineer, **Paul Hammant**, decided to create a server that will act as an HTTP proxy to "trick" the browser into believing that Selenium Core and the web application being tested come from the same domain.
* This system became known as the **Selenium Remote Control** or **Selenium 1**.

## ***Birth of Selenium Grid***

* Selenium Grid was developed by **Patrick Lightbody**to address the need of minimizing test execution times as much as possible.
* He initially called the system "**Hosted QA**."
* It was capable of capturing browser screenshots during significant stages, and also of **sending out Selenium commands to different machines simultaneously.**

## ***Birth of Selenium IDE***

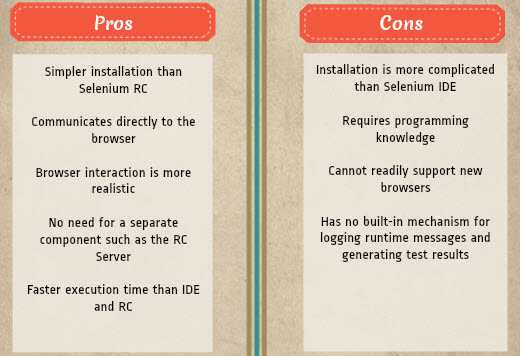
* **Shinya Kasatani** of Japan created **Selenium IDE**, a Firefox extension that can automate the browser through a record-and-playback feature.
* He came up with this idea to further increase the speed in creating test cases.
* He donated Selenium IDE to the Selenium Project in 2**006**.

## ***Brief Introduction WebDriver***

* The WebDriver proves itself to be **better than both Selenium IDE and Selenium RC** in many aspects.
* It implements a more modern and stable approach in automating the browser's actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for Automation.
* **It controls the browser by directly communicating with it.**

The supported languages are the same as those in Selenium RC.

* Java
* C#
* PHP
* Python
* Perl
* Ruby



## Selenium Grid

Selenium Grid is a tool**used together with Selenium RC to run parallel tests** across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

**Features:**

* Enables **simultaneous running of tests**in **multiple browsers and environments.**
* **Saves time** enormously.
* Utilizes the **hub-and-nodes** concept. The hub acts as a central source of Selenium commands to each node connected to it.

## **A Comparison between Selenium and QTP(now UFT)**

**Quick Test Professional(QTP)** is a proprietary automated testing tool previously owned by the company **Mercury Interactive**before it was **acquired by Hewlett-Packard in 2006**. The Selenium Tool Suite has many advantages over QTP as detailed below -

***Advantages of Selenium over QTP***

|  |  |
| --- | --- |
| **Selenium** | **QTP** |
| **Open source**,**free to use**, and**free of charge.** | **Commercial**. |
| **Highly extensible** | Limited add-ons |
| Can run tests across **different browsers** | Can only run tests in **Firefox**, **Internet Explorer**and **Chrome** |
| Supports **various operating systems** | Can only be used in **Windows** |
| Supports **mobile devices** | QTP Supports Mobile app test automation (iOS & Android) using HP solution called - HP Mobile Center |
| Can execute tests **while** the **browser is minimized** | Needs to have the application under test to be visible on the desktop |
| Can execute tests **in parallel**. | Can only execute in parallel but using Quality Center which is again a paid product. |

# ***Locators for Selenium***

* **Id** *Select element with the specified* ***@id*** *attribute.*
* **Name** *Select first element with the specified* ***@name*** *attribute.*
* **Linktext** *Select link (anchor tag) element which contains text matching the specified link text*
* **Partial Linktext** *Select link (anchor tag) element which contains text matching the specified partial link text*
* **Tag Name** *Locate Element using a Tag Name .*
* **Class name***Locate Element using a class Name ..*
* **Css** *Select the element using css selectors.*
* **Xpath** *Locate an element using an XPath expression.*

**Locating an Element By ID**:

Example 1:

<**div** id="toolbar">.....</**div**>

Example 2:

<input id="email" class="required" **type**="text"/>

**Locating an Element By Name:**

<input name="register" **class**="required" type="text"/> WebElement **register**= driver.findElement(By.name("register"));

**Locating an Element By LinkText:**

<**a** href="[http://www.seleniumhq.org">Downloads</**a**> WebElement](http://www.seleniumhq.org/) download = driver.findElement(By.linkText("Downloads"));

**Locating an Element By Partial LinkText**:

<**a** href="seleniumhq.org">Download selenium server</**a**> WebElement download = driver.findElement(By.PartialLinkText("Download"));

**Locating an Element By TagName**:

**Select** **select** = new **Select**(driver.findElement(**By**.tagName("select"))); **select**.selectByVisibleText("Nov"); or **select**.selectByValue("11");

**Locating an Element By Class Name**

WebElement classtest =driver.findElement(By.className(“sample”));

**CSS Selector:**

WebElement CheckElements = driver.findElements(By.cssSelector("input[id=email']"));

**XPath Selector:**  
XPath is designed to allow the navigation of XML documents, with the purpose of selecting individual elements, attributes, or some other part of an XML document for specific processing

There are two types of xpath

1. Native Xpath, it is like directing the xpath to go in direct way. like  
Example:  
html/head/body/table/tr/td

Here the advantage of specifying native path is, finding an element is very easy as we are mention the direct path. But if there is any change in the path (if some thing has been added/removed) then that xpath will break.

2. Relative Xpath.  
In relative xpath we will provide the relative path, it is like we will tell the xpath to find an element by telling the path in between.  
Advantage here is, if at all there is any change in the html that works fine, until unless that particular path has changed. Finding address will be quite difficult as it need to check each and every node to find that path.  
Example:  
//table/tr/td

**Example Syntax to work with Image**

    xpath=//img[@alt='image alt text goes here']

**Example syntax to work with table**

    xpath=//table[@id='table1']//tr[4]/td[2]

    xpath=(//table[@class='nice'])//th[text()='headertext']/

**Example syntax to work with anchor tag**

    xpath=//a[contains(@href,'href goes here')]

    xpath=//a[contains(@href,'#id1')]/@class

**Example syntax to work with input tags**

    xpath=//input[@name='name2' **and** @value='yes']