**Introduction:**

Selenium is an open-source and a portable automated software testing tool for testing web applications. It has capabilities to operate across different browsers and operating systems. Selenium is not just a single tool but a set of tools that helps testers to automate web-based applications more efficiently.



## The Origins of Selenium

[Selenium](https://saucelabs.com/resources/topics/selenium) originated in 2004 by Jason Huggins at [ThoughtWorks](https://www.thoughtworks.com/en-us). Jason was trying to figure out an easy way to [automate testing](https://saucelabs.com/resources/topics/automated-testing) of a time and expense web application.

When asked what name should be given to this tool, Jason gave a flippant, on-the-spot answer: the dominant product at the time was Mercury’s QuickTest Professional, so anyone who hated it and wanted to be “cured” would likely seek out the cure to Mercury poisoning… [which is Selenium](https://pubmed.ncbi.nlm.nih.gov/29124976/)!

## Selenium Tools & Description:

**Selenium IDE**

Selenium **I**ntegrated **D**evelopment **E**nvironment (IDE) is a Firefox plugin that lets testers to record their actions as they follow the workflow that they need to test.

**Selenium RC**

Selenium **R**emote **C**ontrol (RC) was the flagship testing framework that allowed more than simple browser actions and linear execution. It makes use of the full power of programming languages such as Java, C#, PHP, Python, Ruby and PERL to create more complex tests.

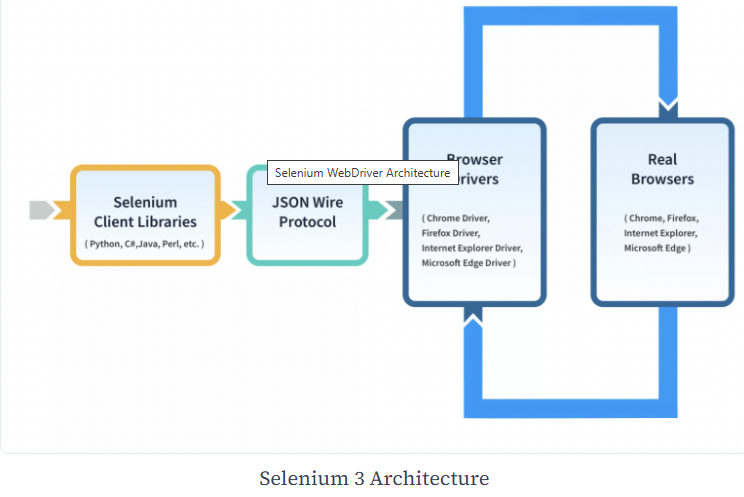
**Selenium WebDriver**

Selenium WebDriver is the successor to Selenium RC which sends commands directly to the browser and retrieves results.

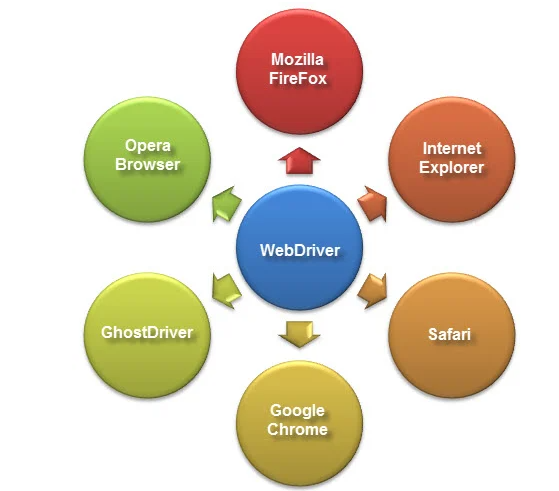
**Selenium Grid**

Selenium Grid is a tool used to run parallel tests across different machines and different browsers simultaneously which results in minimized execution time.

**Selenium Web Driver Architecture:**



**The Browsers Supported by Selenium**



**HTMLUnit is the only browser that WebDriver can directly automate,** meaning that no other separate component is needed to install or run while the test is being executed. For other browsers, a separate program is needed. That program is called the **Driver Server**.

**How to Install Selenium:**

## Selenium WebDriver Installation

Selenium installation is a 3-step process:

**Step 1:** Install Java SDK  
**Step 2:** Install Eclipse  
**Step 3:** Install Selenium Webdriver Files  
**Step 4:** Configure Eclipse IDE with WebDriver

## Difference between Selenium and UFT

Following is the main difference between UFT and Selenium:

| **HP UFT (QTP)** | **Selenium** |
| --- | --- |
| It is commercial tool by Micro Focus and hence it requires a license and is expensive | Its an [open source testing tool](https://www.guru99.com/best-open-source-testing-tools.html)  hence it does not require  license and is free |
| It is used for testing client-server applications. It can test web- based as well as desktop applications | Using Selenium only web  applications can be automated |
| QTP tests can only be developed in  [QTP](https://www.guru99.com/quick-test-professional-qtp-tutorial.html)IDE | Selenium has the option of using  wide range of IDEs like  Visual Studio, Eclipse, Netbeans |
| HP UFT only supports VB script | Selenium supports JAVA,  .NET, Ruby, Perl, PHP and  many other  [programming languages](https://www.guru99.com/best-programming-language.html) |
| HP UFT comes with built in object repository.  Object repository development and  maintenance is quite easy in HP ALM | Selenium dose not have such  built in object repository,  but object can be managed  by using UI element user extension |
| HP UFT integrates with test management  tool like [HP Quality Center](https://www.guru99.com/hp-alm-free-tutorial.html) | There is no such tool that  integrates with Selenium |
| All types of dialog box is supported by UFT Testing | It supports dialog box partially |
| It supports different environments like SAP, Oracle, .NET but user need to purchase add-on license for them | It supports addition of plug-ins to include features that are not provided by core Selenium |
| Automation testing is expensive due to licensing cost | Web automation testing with  Selenium incurs less cost |
| HP QTP offers very good technical support | It has no official support,  since it is an open source |
| HP QTP supports only[VBScript](https://www.guru99.com/vbscript-tutorials-for-beginners.html). Hence  Object Oriented Approach to Test Automation becomes challenging | Selenium supports mature  Object Oriented Language  like Java |
| HP QTP is more user friendly and scripts are  developed quickly. | It is a low level tool with less  functionality. Script requires  more time to develop and maintain |
| Backward compatibility is powerful.  Latest version of HP ALM supports code  that was developed 5 years back | With new Selenium release  the API changes.  So Test Scripts need to be  updated |

**Selenium API:**

<https://www.selenium.dev/selenium/docs/api/java/index.html>

https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/package-summary.html

**Important Interfaces:**

**WebDriver: (Interface)**

https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebDriver.html

**WebElement: (Interface)**

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html>

**TakesScreenshot :Interface**

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/TakesScreenshot.html>

# JavascriptExecutor

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/JavascriptExecutor.html>

# Alert:

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Alert.html>

# Capabilities:

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Capabilities.html>

# Action:

Interface representing a single user-interaction action.

https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/interactions/Action.html

**Important Classes:**

**By: (Class)**

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.html>

**Cookie: (Class)**

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Cookie.html>

**Dimension:(Class)**

https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Dimension.html

**Keys: (Enum Class)**

https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Keys.html

**Point:**

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Point.html>

**Dimension:**

https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Dimension.html

**Events:**

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/events/package-summary.html>

**Keys:**

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Keys.html>

**Actions:**

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/interactions/Actions.html>

The user-facing API for emulating complex user gestures.

**Locators of Selenium WebDriver:**

* *Selenium supports 8 different types of locators namely id, name, className, tagName, linkText, partialLinkText, CSS selector and xpath.*
* *Using id is one of the most reliable and fast methods of element recognition. Usually, the id is always unique on a given web page.*
* *CSS selector and XPath can identify dynamic elements on a web page. A combination of different attributes and tag names can be used with CSS and xpath to identify any given element.*
* ID
* Name
* ClassName
* LinkText
* Partial LinkText
* TagName
* CSS Selector
* XPath

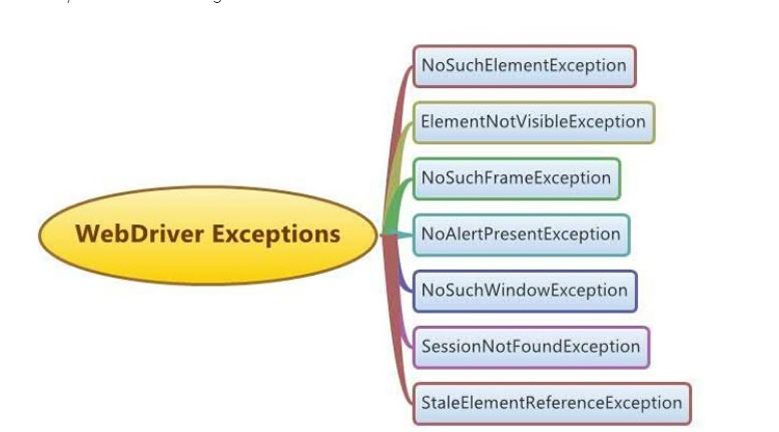
**Best Practices For Selenium Locators**

Choosing the correct locator for recognizing a web element is quite essential in Selenium. Listed below are some of the best practices that a quality engineer needs to follow to make efficient use of locators in the ***Selenium WebDriver based automation framework***.

* *Do not use****dynamic attribute****values to locate elements, as they may change frequently and result in breakage of locator script. It also severely affects the****maintainability, reliability****, and****efficiency****of the automation script.*
* ***ID****and****name****attributes take precedence over other locators if your web page contains unique****ID****and****name****, then it's always advisable to use them instead of****XPath****as they are faster and more efficient.*
* *While using****locators****, make sure that your locator points precisely to the required element. If the needed scenario needs to perform some operation on a single element, then make sure that your locator exactly matches to only one element. If the locator points to several different elements, then it may cause breakage in your script.*
* *Never use locators to locate auto-generated elements on the web page. Sometimes in a dynamic web environment, element attribute properties are generated at run time. One should avoid these elements as they may cause breakage during script execution*.
* *While working with****XPath****or****CSS locators****, one should avoid directly using the one generated by the****Chrome Dev Tools****. It may seem one of the easiest ways to generate****XPath****, but in the long run, it induces reliability issues, code breakage, maintainability issues, etc. It may look tempting to use these, but you would be better off creating your customized****XPath****in the longer run.*

|  |  |
| --- | --- |
| **XPath locators in Selenium** WebDriver are used to identify elements on a web page.  These locators allow complex and flexible navigation of the web page's Document Object Model (DOM). There are several types of XPath locators,  each useful in different situations.  **● XPath locator by ID:**This locator allows you to identify an element by its id attribute.  Example:  driver.findElement(By.xpath("//\*[@id='username']"));  Note: XPath is a wildcard in the snippets helping to select unknown XML nodes  **●  XPath locator by class name:** This locator can identify elements based on their class attribute.  Example:  driver.findElement(By.xpath("//\*[@class='login-button']"));  **● XPath locator by name:**This locator identifies elements by their name attribute.  Example:  driver.findElement(By.xpath("//\*[@name='password']"));  **● XPath locator by tag name:**This locator can identify elements by their HTML  tag name.  Example:  driver.findElement(By.xpath("//p"));  **● XPath locator by text:** This locator identifies elements based on their inner text.  Example:  driver.findElement(By.xpath("//\*[text()='Submit']"));  **● XPath locator using contains:**This locator can identify elements based on a  substring of one of their attribute values.  Example:  driver.findElement(By.xpath("//\*[contains(@href,'google.com')]"));  **● XPath locator using starts-with:**This locator identifies elements whose attribute  values start with a particular string.  Example:  driver.findElement(By.xpath("//\*[starts-with(@id,'user')]"));  **● XPath locator using ends-with:** This locator can identify elements whose attribute  values end with a particular string.  Example:  driver.findElement(By.xpath("//\*[ends-with(@id,'name')]")); |  |
| **Xpath Axes Methods:**  <https://www.headspin.io/blog/using-xpath-in-selenium-effectively> |  |

**WebDriver Exceptions:**

****

**Detailed Info on Exceptions:**

**https://www.testquality.com/blog/tpost/ci5pvsurc1-different-types-of-selenium-webdriver-co**

|  |  |
| --- | --- |
| **TestNG**  (Documentation)  <https://testng.org/> |  |
|  |  |
|  |  |