

# SELENIUM

---

*The ART of Browser (HTML+DOM) Testing  
...the automated way!*

*Presented By: Mabeth Borres of GA-SEI31*

# WHY WEB TEST AUTOMATION?

---

- Would you love a world where the computer does the testing of your web application for you? Particularly tedious form validations? Or, mundane site navigation links testing?

# SELENIUM... OR FIRSTLY, 'JAVASCRIPTTESTRUNNER'

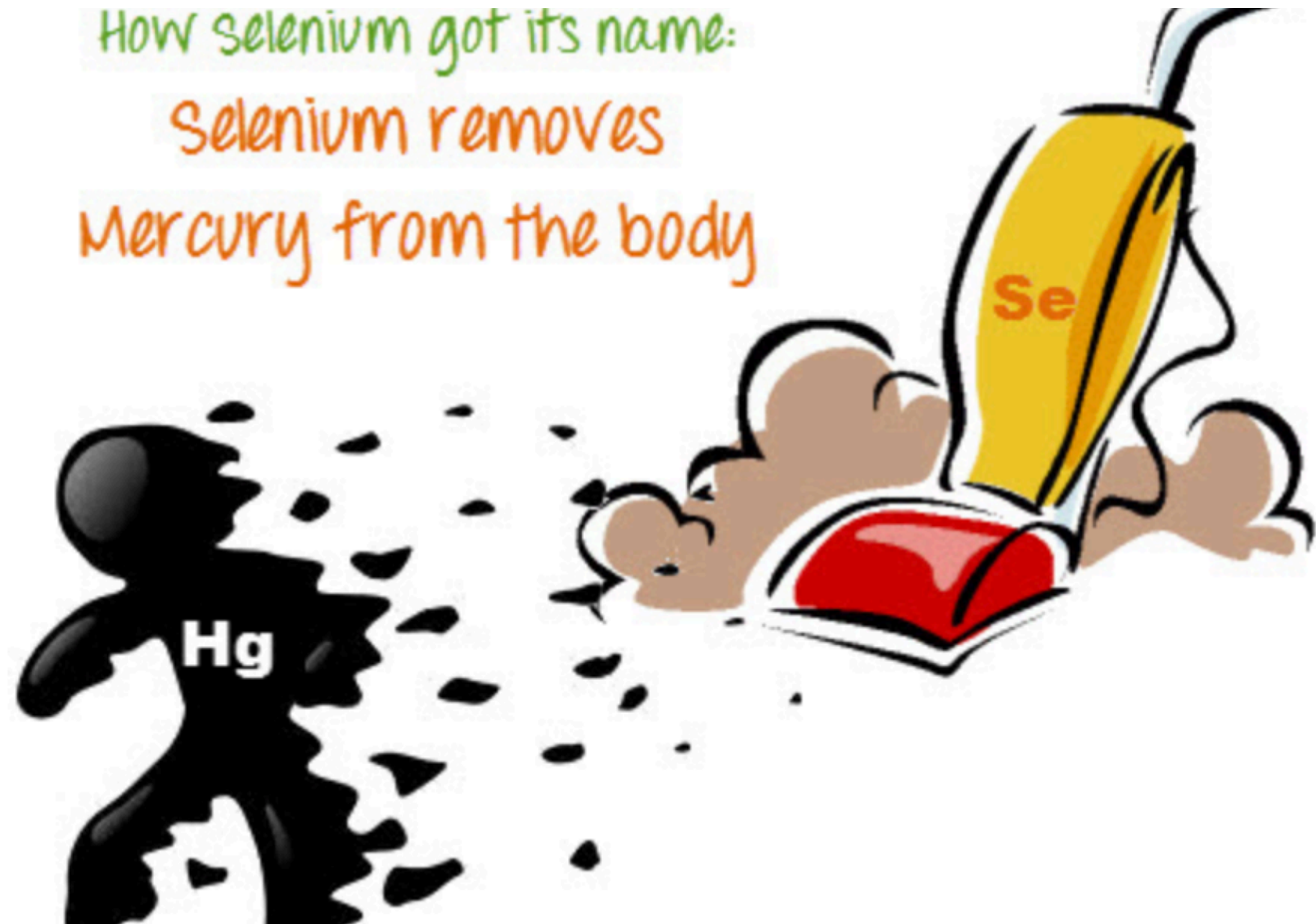
---

- Open-source browser automation tool developed by Jason Huggins (of THOUGHTWORKS) in 2004
- THOUGHTWORKS Product
- Originally named 'JavaScriptTestRunner' - HTML + DOM testing

# SO, WHY THE NAME 'SELENIUM' ?

---

- Selenium is a well-known antidote for Mercury poisoning.



*Selenium is often being compared to Mercury's QuickTest Pro (QTP)*

# MERCURY INTERACTIVE\*

## ➤ Evolution



MERCURY INTERACTIVE

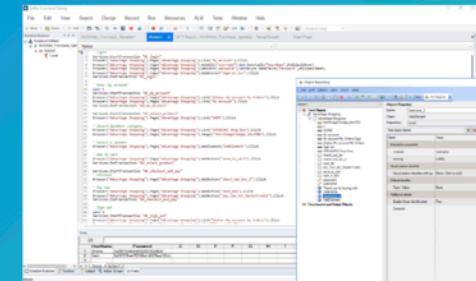
MICRO  
FOCUS

Unified Functional Testing (UFT)

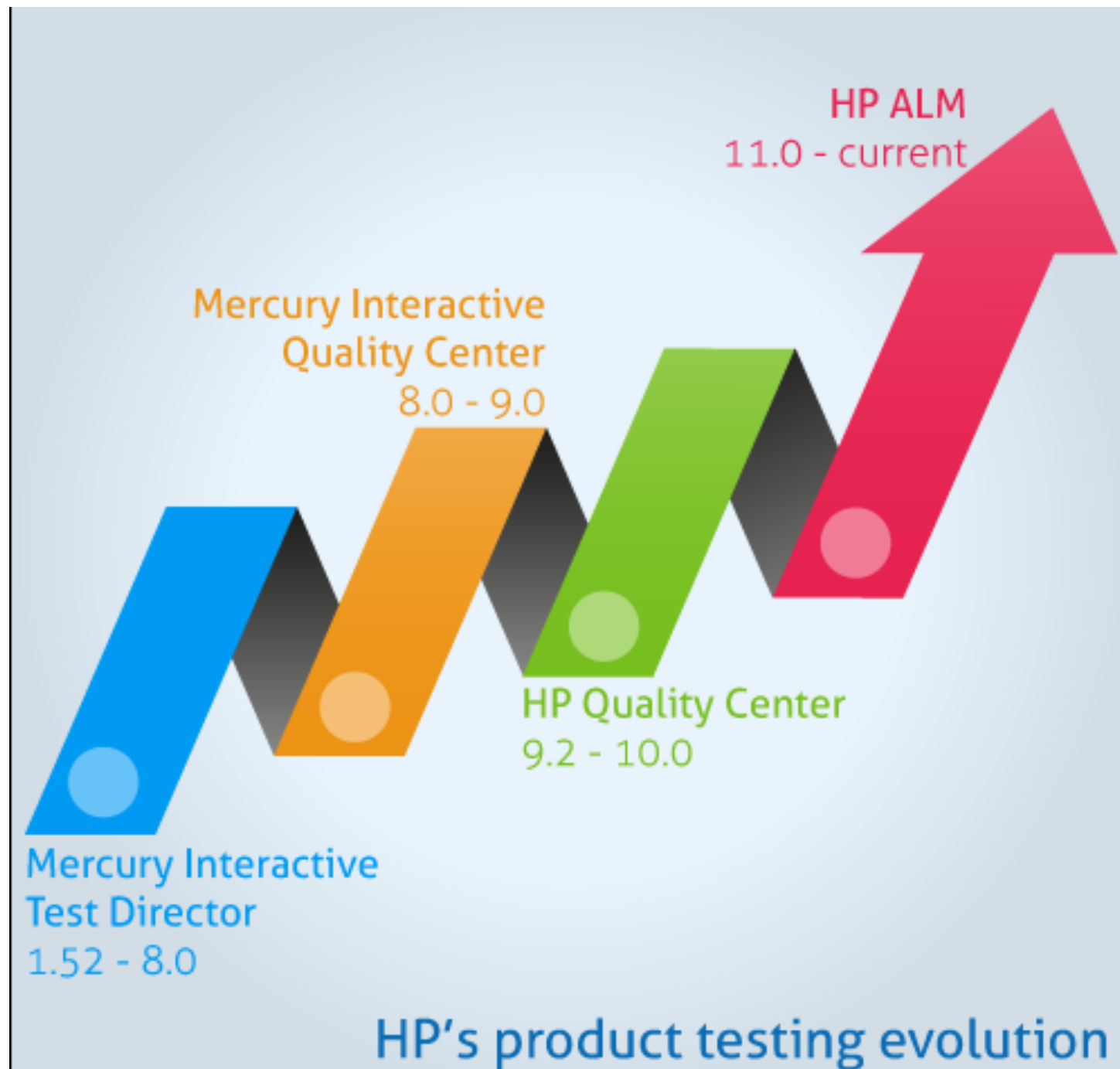
## Functional Test Automation Software

Intelligent test automation for web, mobile, API, hybrid, RPA, and enterprise apps

Free Trial



\*Presenter's test automation work experience



# ADVANTAGES OF SELENIUM OVER QUICKTEST PRO

.....

## 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 <b>different browsers</b>	Can only run tests in <b>Firefox, Internet Explorer and Chrome</b>
Supports <b>various operating systems</b>	Can only be used in <b>Windows</b>
Supports <b>mobile devices</b>	QTP Supports Mobile app test automation (iOS & Android) using HP solution called - HP Mobile Center
Can execute tests <b>while the browser is minimized</b>	Needs to have the application under test to be visible on the desktop
Can execute tests <b>in parallel.</b>	Can only execute in parallel but using Quality Center which is again a paid product.

*WINNING POINTS: COST - FLEXIBILITY - PARALLEL TESTING*



# WHY OTHERS CHOOSE QUICKTEST PRO

.....

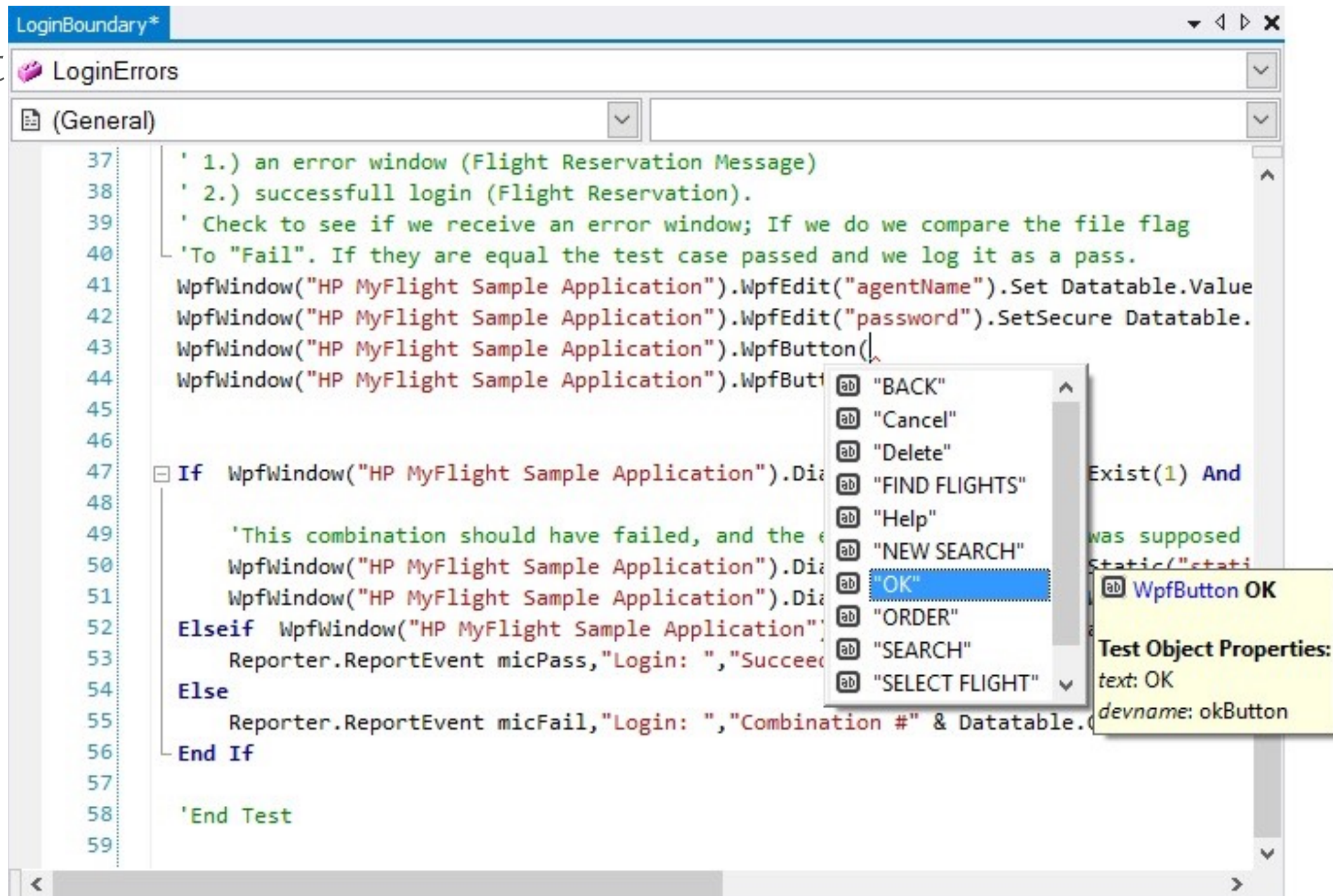
## Advantages of QTP over Selenium

QTP	Selenium
Can test <b>both web and desktop applications</b>	Can only test web applications
Comes with a <b>built-in object repository</b>	Has no built-in object repository
<b>Automates faster than Selenium</b> because it is a fully featured IDE.	Automates at a slower rate because it does not have a native IDE and only third party IDE can be used for development
Data-driven testing is easier to perform because <b>it has built-in global and local data tables.</b>	Data-driven testing is more cumbersome since you have to rely on the programming language's capabilities for setting values for your test data
<b>Can access controls within the browser</b> (such as the Favorites bar, Address bar, Back and Forward buttons, etc.)	Cannot access elements outside of the web application under test
Provides professional <b>customer support</b>	No official user support is being offered.
Has native capability to <b>export test data</b> into external formats	Has no native capability to export runtime data onto external formats
Parameterization Support is built	Parameterization can be done via programming but is difficult to implement.
Test Reports are generated automatically	No native support to generate test /bug reports.

*Catch: IDE or Integrated Development Environment*

# VBSCRIPT – VISUAL BASIC SCRIPTING (QTP)

- JavaScript
- Rules
- Over
- VBScript,
- Peace!



*Just like any other language... LEARN its Data Types, Syntax, Libraries, Operators, etc*



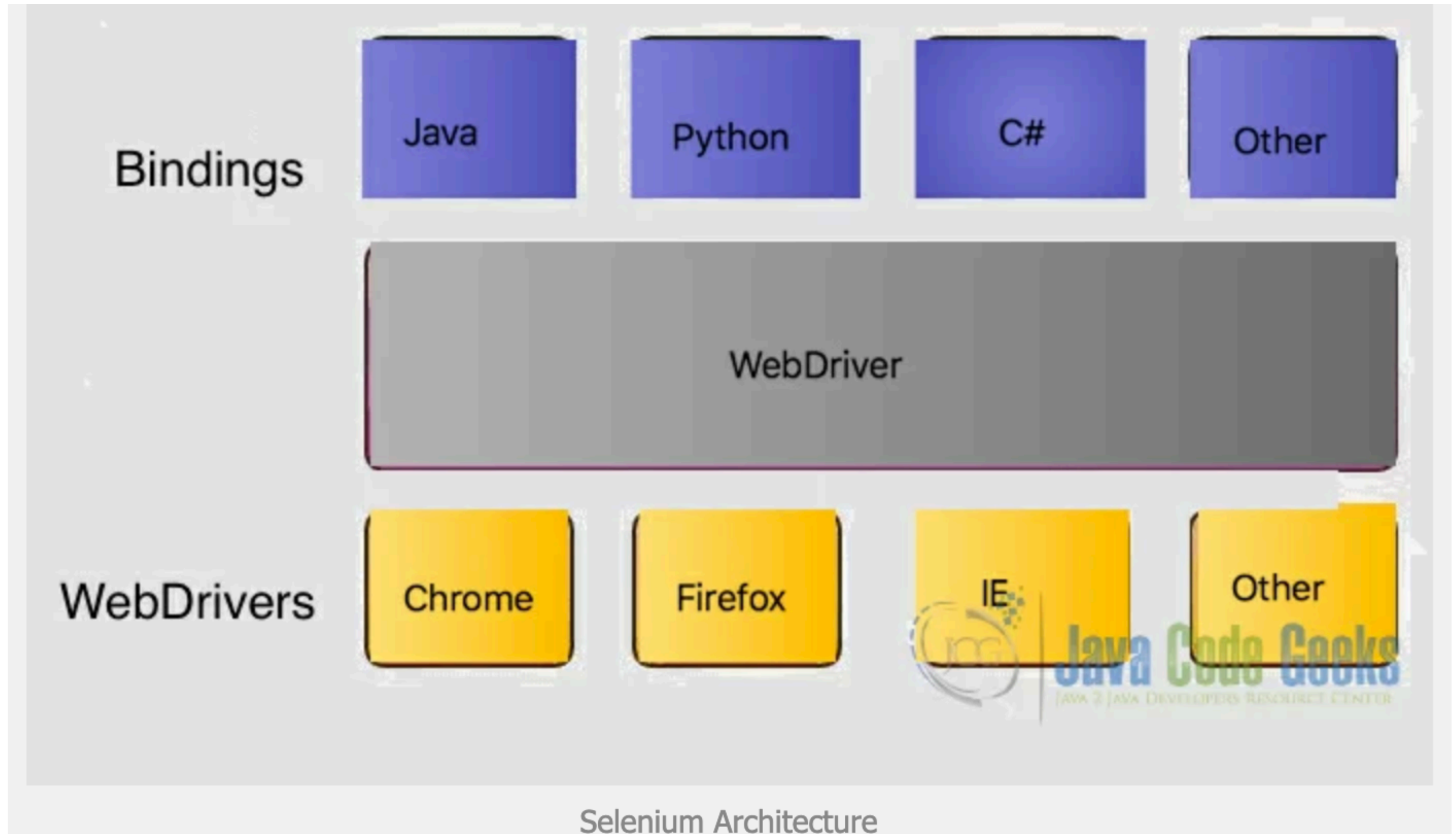
# SELENIUM TOOL SUITE

---

- **Selenium IDE**, a Browser add-on that you can only use in creating relatively simple test cases and test suites.
- **Selenium Remote Control**, also known as **Selenium 1**, which is the first Selenium tool that allowed users to use programming languages in creating complex tests.
- **WebDriver**, the newer breakthrough that allows your test scripts to communicate directly to the browser, thereby controlling it from the OS level.
- **Selenium Grid** is also a tool that is used with Selenium RC to execute parallel tests across different browsers and operating systems.

# SELENIUM 2 ARCHITECTURE

---



*Used Java for this Demo*

# SELENIUM IDE

---

- Demo
- Easiest to install and use
- Test Prototyping
- Knowledge of HTML and DOM
- Test: <https://mmborres.github.io/index.html>

# SELENIUM IDE TEST

## ► Script

The screenshot displays the Selenium IDE interface for a project named "Test Pamper My Pet". The URL bar shows "https://mmborres.github.io/pamper\_my\_pet\_deploy/". The test script is titled "Test Login - Add To Cart - Update - Remove". The script consists of the following commands:

Line	Command	Target	Value
23	click	css=input	
24	type	css=input	1
25	click	css=input	
26	click	id=1	
27	// click	xpath=//button[contains(.,'Add to Cart')]	
28	mouse over	css=.fa-shopping-cart	
29	mouse out	css=.fa-shopping-cart	
30	wait for element present	xpath=//i[contains(.,' Cart')]	10000
31	click	css=.fa-shopping-cart	
32	wait for element present	css=.fa-trash	10000
33	click	css=.fa-trash	
34	click	linkText=View Order History	
35	click	linkText=Open	
36	click	linkText=Logout	
37	close		

Below the script table, there are input fields for Command, Target, Value, and Description. The status bar at the bottom indicates "Runs: 1 Failures: 0". The log shows the following messages:

- 34.click on linkText=View Order History OK
- 35.Trying to find linkText=Open... OK
- 36.click on linkText=Logout OK
- 37.close OK
- 'Test Login - Add To Cart - Update - Remove' completed successfully

*Record once and reuse whenever*

# SELENIUM WEBDRIVER

---

- Demo
- Download Required Drivers
- Language Knowledge: ex. Java
- Identify Elements of the Page:
  - XPath
  - ID
  - CSS Selector
  - Classname



**TestNG** is a testing [framework](#) for the [Java programming language](#) created by [Cédric Beust](#) and inspired by [JUnit](#) and [JUnit4](#). The design goal of TestNG is to cover a wider range of test categories: unit, functional, end-to-end, integration, etc., with more powerful and easy-to-use functionalities.



# WEBDRIVER IN JAVA + TESTNG

## ► Java

The screenshot displays the Eclipse IDE environment. The Package Explorer on the left shows the project structure: FirstTest > src > automation > automation.testNG.TestMemGame1\_CustomPage. The main editor shows the code for MinHighScore\_NegativeTest\_NonIntegerValue.java. The code includes imports for Selenium WebDriver, TestNG annotations, and a test method testHighScore\_NonInteger(). The IDE also shows the TestNG Results window at the bottom, indicating that the tests passed successfully.

```
6
7 import org.testng.annotations.BeforeTest;
8 import org.openqa.selenium.By;
9 import org.openqa.selenium.WebDriver;
10 import org.openqa.selenium.firefox.FirefoxDriver;
11 import org.testng.annotations.AfterTest;
12
13 public class MinHighScore_NegativeTest_NonIntegerValue {
14     WebDriver driver = null;
15
16     @Test
17     public void testHighScore_NonInteger() throws Exception {
18         driver.findElement(By.id("minHiScore")).click();
19         driver.findElement(By.id("minHiScore")).sendKeys("ZZ");
20         Thread.sleep(2000);
21         driver.findElement(By.xpath("//input[@value='Ready? Game On!']")).click();
22         String actual = driver.switchTo().alert().getText();
23         System.out.println(actual);
24         Assert.assertEquals("Minimum High Score must be a positive number.", actual);
25         driver.switchTo().alert().accept();
26         driver.close();
27     }
28 }
```

Search:  ☒ Passed: 9 ☒ Failed: 0 ☒ Skipped: 0 Tests: 1/1 Methods: 9 (95906 ms)

Default suite ( 9/0/0/0 ) (33.722 s)

- Default test ( 33.722 s)
  - automation.testNG.TestMemGame1\_CustomPage.A\_GameLevel\_Positive\_Test
    - testGameLevel\_Positive (6.79 s)
  - automation.testNG.TestMemGame1\_CustomPage.MinHighScore\_NegativeTest\_NegativeValue
    - testHighScore\_Negative (2.709 s)
  - automation.testNG.TestMemGame1\_CustomPage.MinHighScore\_NegativeTest\_ZeroValue
    - testHighScore\_Zero (2.901 s)
  - automation.testNG.TestMemGame1\_CustomPage.GameLevel\_NegativeTest\_MoreThanMaxValue
    - tesGameLevel\_MoreThanMax (2.799 s)
  - automation.testNG.TestMemGame1\_CustomPage.A\_MinHighScore\_Positive\_Test
    - testHighScore\_Positive (7.072 s)
  - automation.testNG.TestMemGame1\_CustomPage.MinHighScore\_NegativeTest\_NonIntegerValue
    - testHighScore\_NonInteger (2.869 s)
  - automation.testNG.TestMemGame1\_CustomPage.GameLevel\_NegativeTest\_NegativeValue
    - tesGameLevel\_NegativeValue (2.814 s)
  - automation.testNG.TestMemGame1\_CustomPage.GameLevel\_NegativeTest\_NonIntegerValue
    - tesGameLevel\_NonIntegerValue (2.805 s)
  - automation.testNG.TestMemGame1\_CustomPage.GameLevel\_NegativeTest\_ZeroValue
    - tesGameLevel\_ZeroValue (2.963 s)

Create Java Class, Run as TestNG