# Selenium

Selenium is a suite of tools for automating web browsers

# Outline Selenium

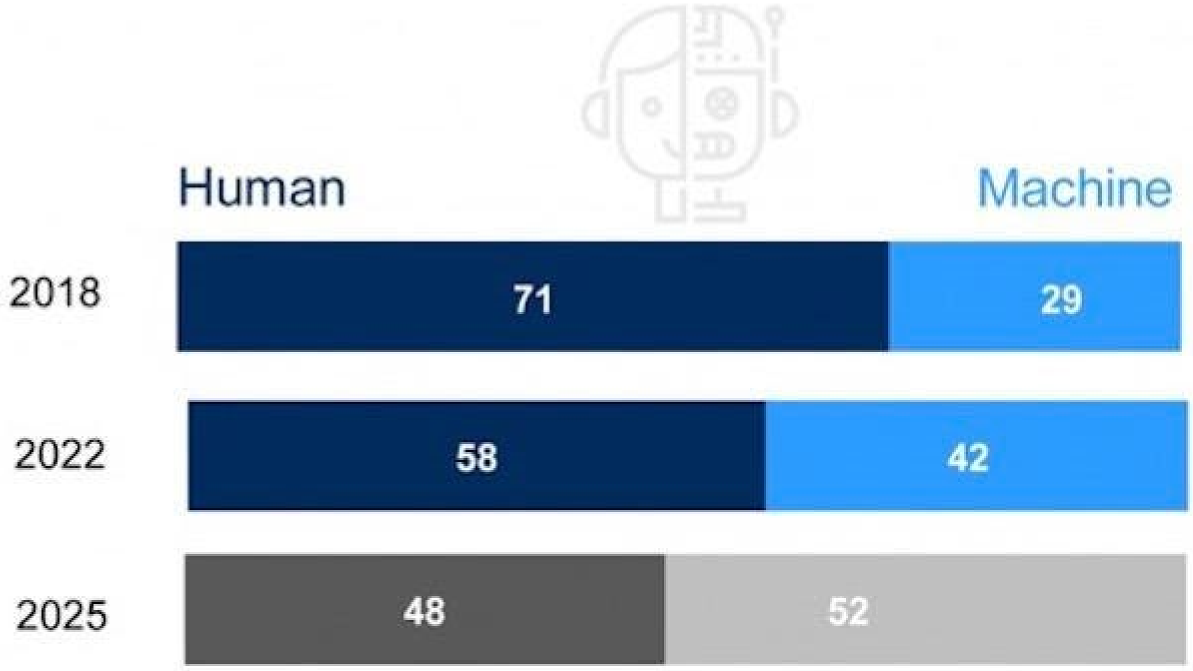
1. What is Selenium
2. Features of Selenium
3. Components of Selenium
4. Browsers | OS | Languages supported

# Overview of Test Automation

- In 2018

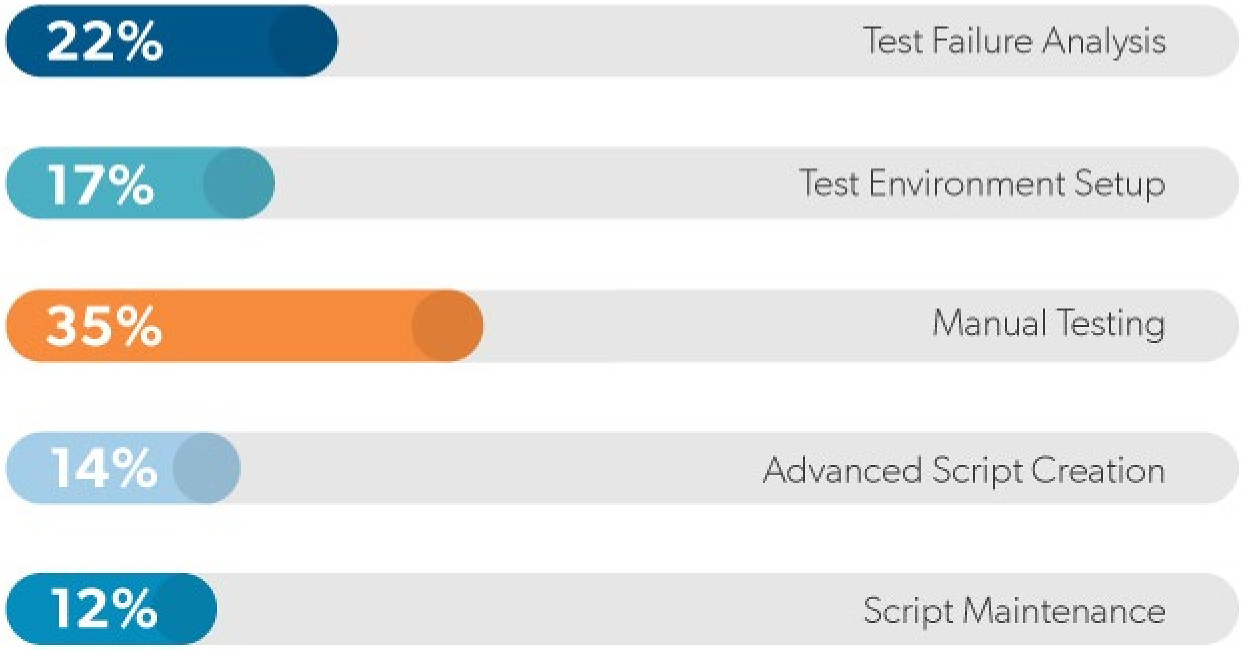
Rate of automation

Division of labour as share of hours spent (%)



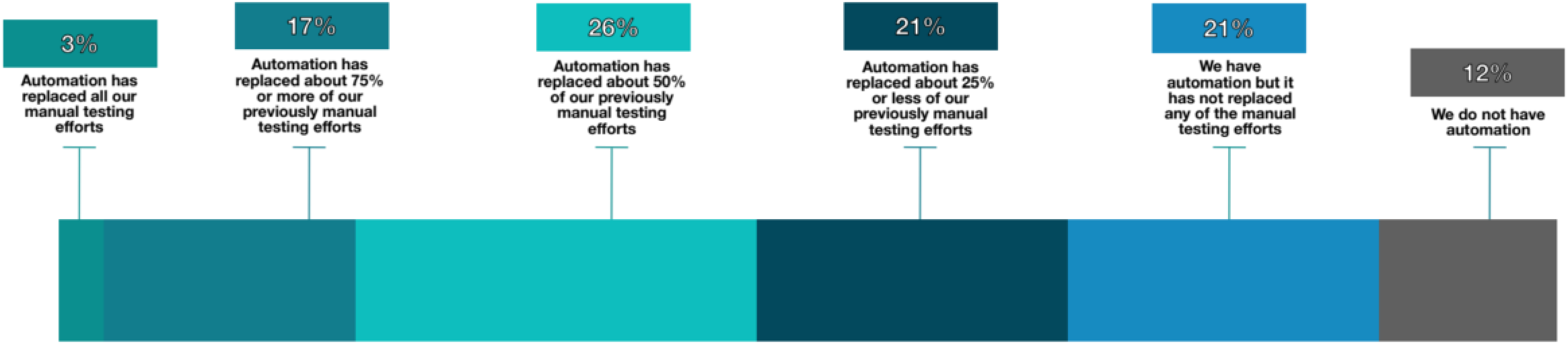
Source: Future of Jobs Report 2018, World Economic Forum

# The most time-consuming activities within a test cycle



https://research.aimultiple.com/test-automation-statistics/

# Effect of automated testing on effort reduction



# Automated tests adoption

In 2022 these percentages of testers claimed to use automated testing or scripting for:

1. Functional and regression testing: 73%
2. Unit testing: 45%
3. CI/CD: 44%
4. Load and stress testing: 31%

- In 2022,
    1. 55% of companies claimed to pursue automated testing strategies cite quality improvement, and ~30% cite time to market as their primary driver
    2. ~42% of companies indicated that test automation is a key part of QA

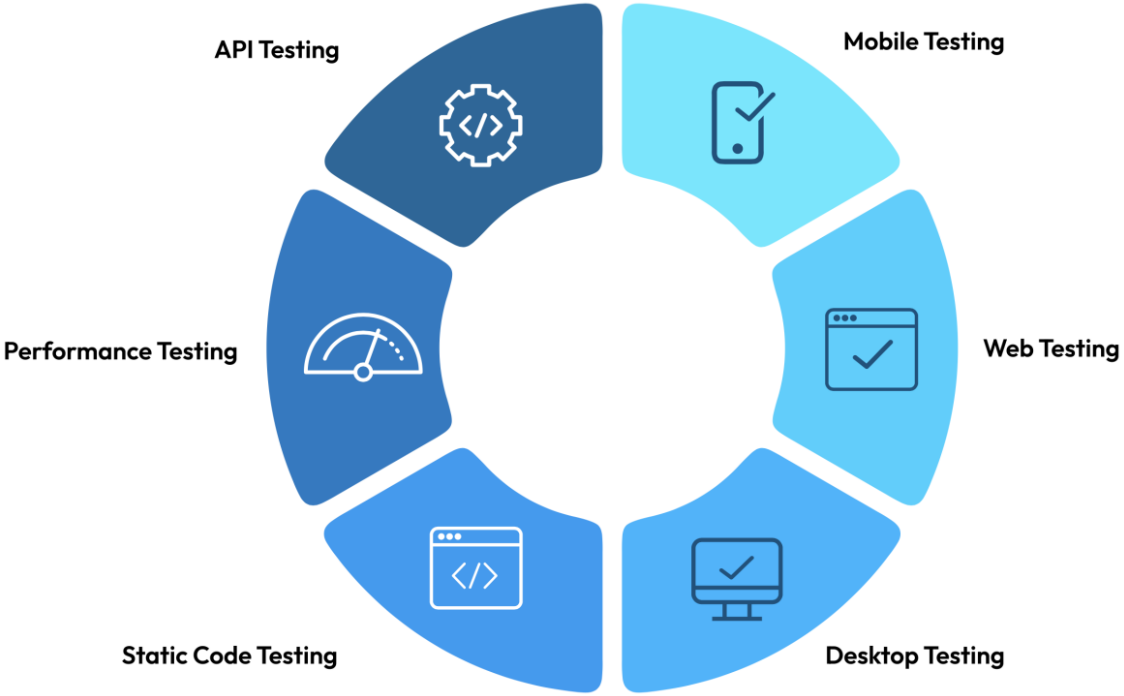
https://research.aimultiple.com/test-automation-statistics/

# Test development models/frameworks

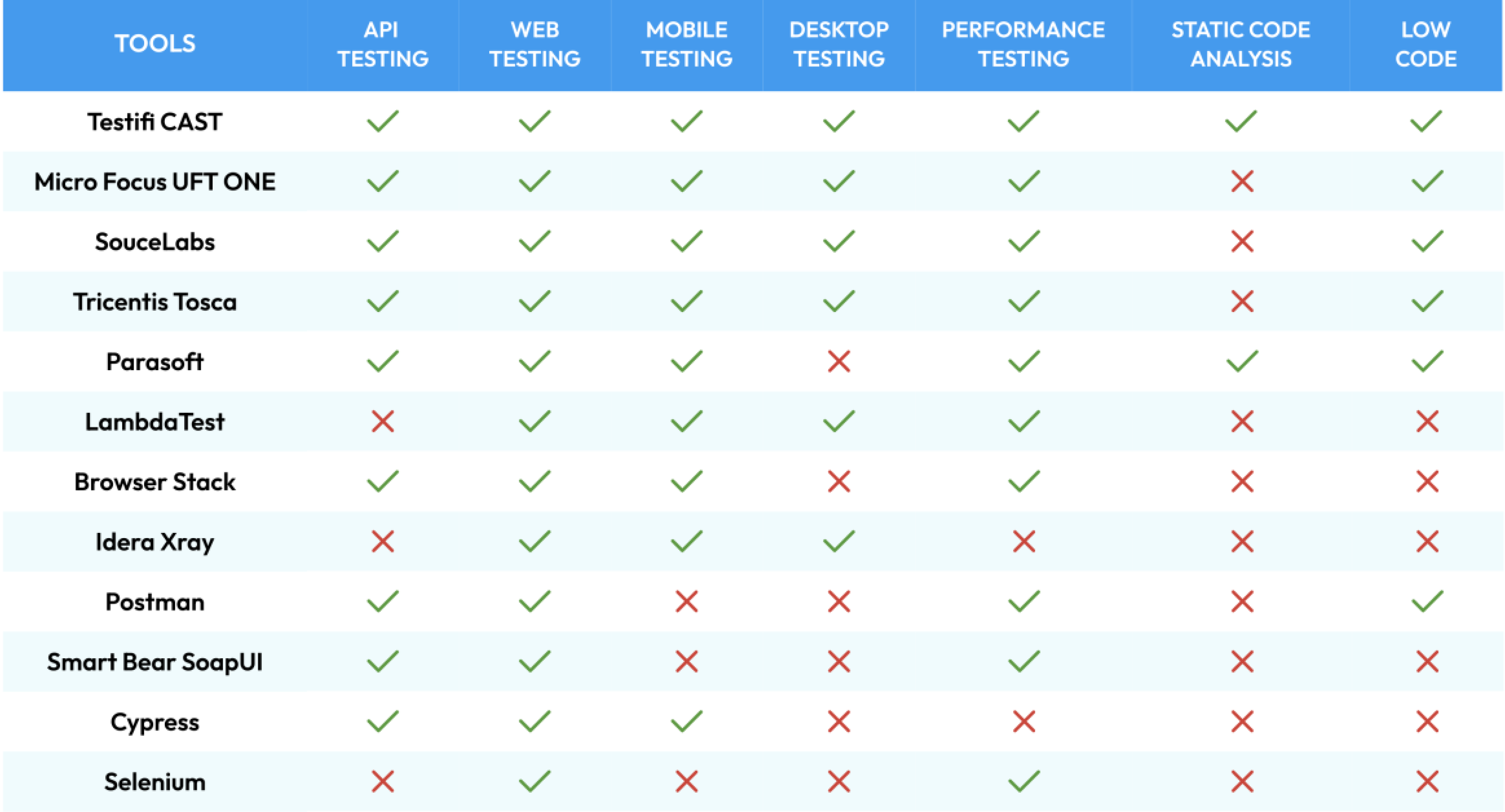
1. In 2022 test-driven development as a development framework was used 18% of the time.
2. In 2022 DevOps as a development model was used 38% of the time.
3. API testing is an essential skill that the tester should be familiar with. In a survey, 97% of responders indicated that API testing is crucial for their operational success.
4. In a survey, 99% of responders indicated that functional test automation and test scripting are crucial for them.
5. **Market size & growth**
6. The global test market exceeded $40 billion in 2020.
7. The global test market is expected to grow between 7% to 12% CAGR between 2021 and 2025.

https://research.aimultiple.com/test-automation-statistics/

# Top 7 key Features for automation testing tools



https://research.aimultiple.com/test-automation-platforms/



| TOOLS | API TESTING | WEB TESTING | MOBILE TESTING | DESKTOP TESTING | PERFORMANCE TESTING | STATIC CODE ANALYSIS | LOW CODE |
|---|---|---|---|---|---|---|---|
| Testifi CAST | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Micro Focus UFT ONE | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |
| SouceLabs | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |
| Tricentis Tosca | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |
| Parasoft | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| LambdaTest | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ |
| Browser Stack | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ |
| Idera Xray | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| Postman | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ |
| Smart Bear SoapUI | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ |
| Cypress | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Selenium | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ |

https://research.aimultiple.com/test-automation-platforms/

# Low code

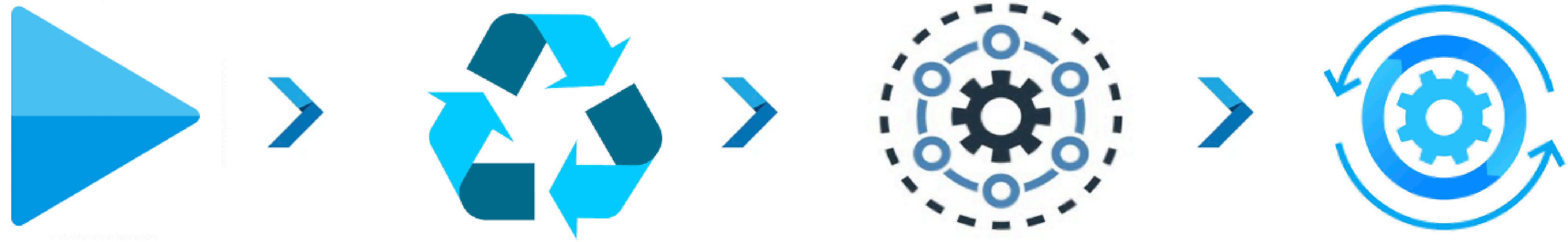
- No-code or codeless automated testing has a record and playback design. So using it doesn't require extensive coding and programming knowledge. According to a survey of 3,300 IT professionals, 66% of respondents preferred codeless platforms to accelerate digital transformation and improve responsiveness to business objectives.[11]

Low-code test automation tools can:

- Accelerate development by 5 to 10 times [12]
- Shorten development cycle
- Support digital transformation
- Lessen reliance on hard-to-hire technical skills.

# How does codeless automated testing work?

AIMultiple



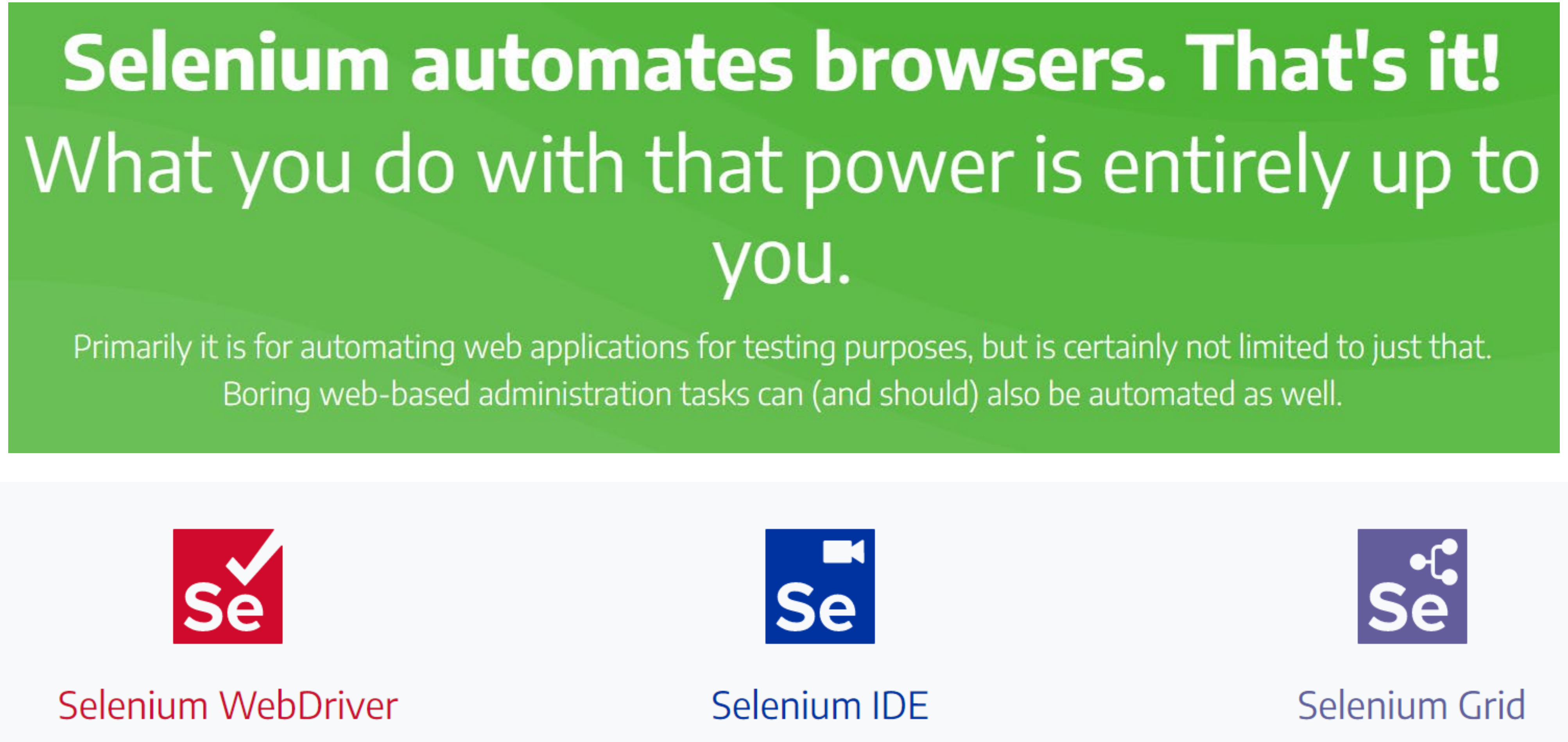
Automation with Record and Playback

Automation with Reusable Function

Automation with Frameworks

Codeless Automation

# https://www.selenium.dev/



**Selenium automates browsers. That's it!**

What you do with that power is entirely up to you.

Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should) also be automated as well.

Selenium WebDriver

Selenium IDE

Selenium Grid

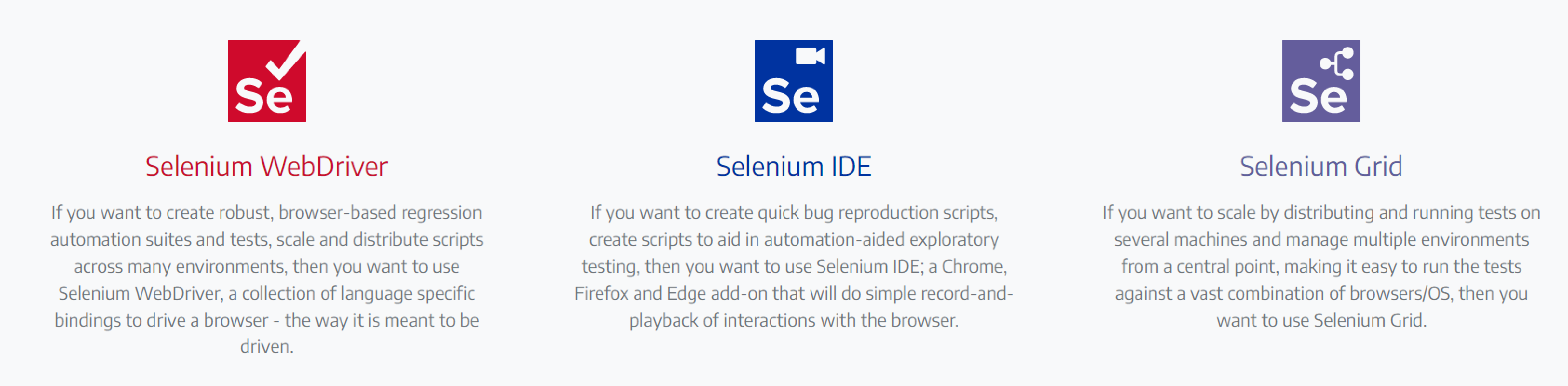
# Advantages of Selenium

**Selenium is an Open Source Software**

- **Selenium supports various programming languages to write programs (Test scripts)** [Java,
- C#, Python, Ruby, JavaScript, and Kotlin
- **Selenium supports various operating systems**
- **Selenium supports various Browsers (Mozilla Firefox, Google Chrome, IE, Opera, Safari etc**
- **Selenium supports Parallel Test Execution**

# Drawbacks of Selenium

- Supports only web-based applications
- Difficult to use, takes more time to create Test cases
- Difficult to Setup Test Environment
- Limited support for Image Testing
- No Built-in Reporting facility



### Selenium WebDriver

If you want to create robust, browser-based regression automation suites and tests, scale and distribute scripts across many environments, then you want to use Selenium WebDriver, a collection of language specific bindings to drive a browser - the way it is meant to be driven.

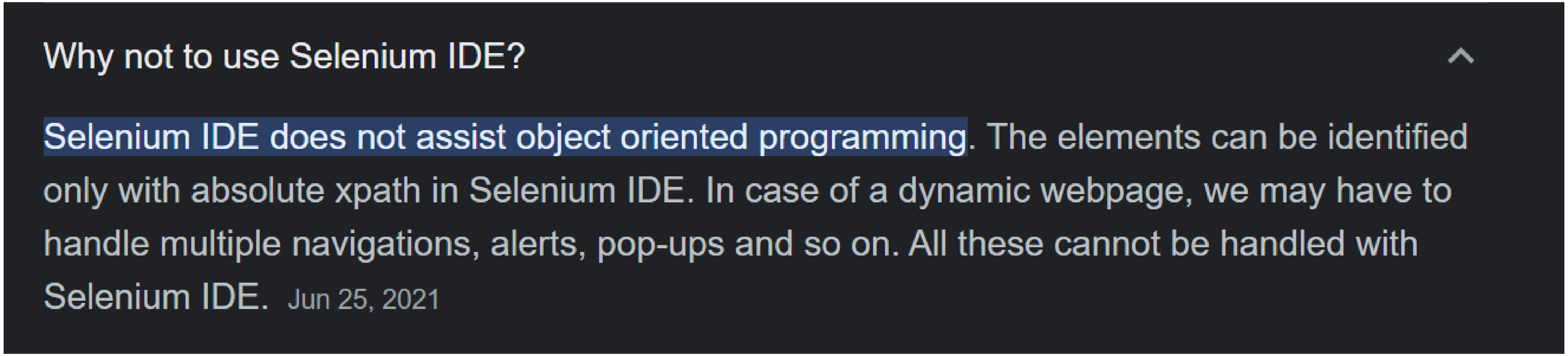
### Selenium IDE

If you want to create quick bug reproduction scripts, create scripts to aid in automation-aided exploratory testing, then you want to use Selenium IDE; a Chrome, Firefox and Edge add-on that will do simple record-and-playback of interactions with the browser.

### Selenium Grid

If you want to scale by distributing and running tests on several machines and manage multiple environments from a central point, making it easy to run the tests against a vast combination of browsers/OS, then you want to use Selenium Grid.

# Selenium IDE vs WebDriver

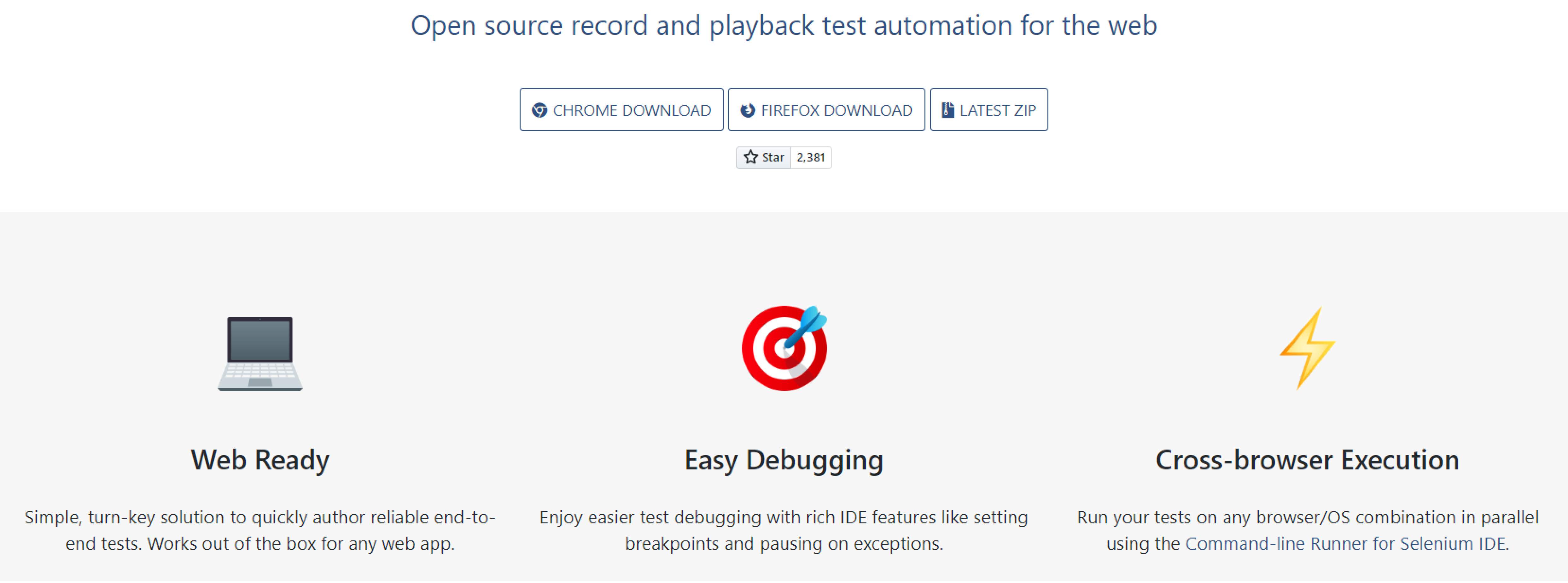


Why not to use Selenium IDE?

Selenium IDE does not assist object oriented programming. The elements can be identified only with absolute xpath in Selenium IDE. In case of a dynamic webpage, we may have to handle multiple navigations, alerts, pop-ups and so on. All these cannot be handled with Selenium IDE. Jun 25, 2021

# Selenium IDE

https://www.selenium.dev/selenium-ide/



Open source record and playback test automation for the web

### Web Ready

Simple, turn-key solution to quickly author reliable end-to-end tests. Works out of the box for any web app.

### Easy Debugging

Enjoy easier test debugging with rich IDE features like setting breakpoints and pausing on exceptions.

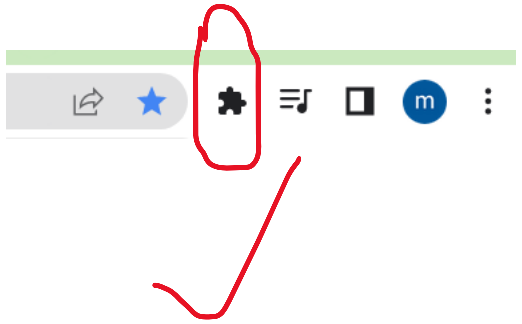
### Cross-browser Execution

Run your tests on any browser/OS combination in parallel using the Command-line Runner for Selenium IDE.

# Selenium IDE

- Add the extension to your browser
- Try some online samples at:
- Try form authentication

[https://the-internet.herokuapp.com/](https://the-internet.herokuapp.com/)



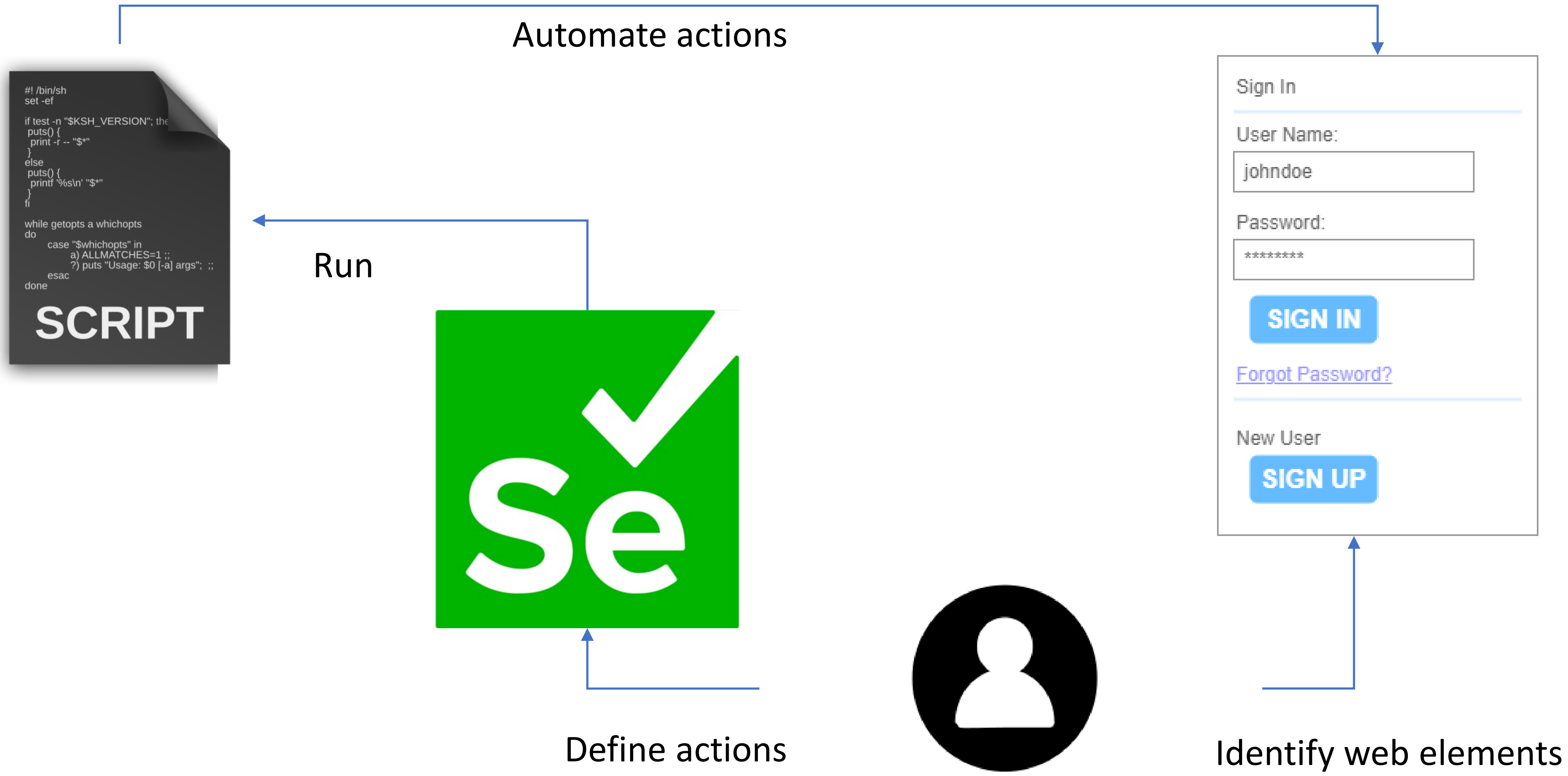
# Selenium IDE

Selenium IDE is part of the Selenium suite and was developed to speed up the creation of automation scripts. It’s a rapid prototyping tool and can be used by engineers with no programming knowledge whatsoever.

Supported functions

- Support for both Chrome and Firefox
- Improved locator functionality
- Parallel execution of tests using Selenium command line runner
- Provision for control flow statements
- Automatically waits for the page to load
- Supports embedded JavaScript code-runs
- IDE has a debugger which allows step execution, adding breakpoints
- Support for code exports

# Typical Selenium Flow



Automate actions

Run

Define actions

Identify web elements

```
#! /bin/sh
set -ef

if test -n "$KSH_VERSION"; the
 puts() {
  print -r -- "$*"
 }
else
 puts() {
  printf '%s\n' "$*"
 }
fi

while getopts a whichopts
do
        case "$whichopts" in
                a) ALLMATCHES=1 ;;
                ?) puts "Usage: $0 [-a] args";  ;;
        esac
done
```

Sign In

User Name: johndoe

Password: ********

# What is Selenium WebDriver?

- Selenium WebDriver is currently the most widely used component in the Selenium tool suite. Selenium WebDriver: Selenium 2 integrated with WebDriver API provides an understandable programming interface. JAVA and C# languages are mostly preferred to work with Selenium

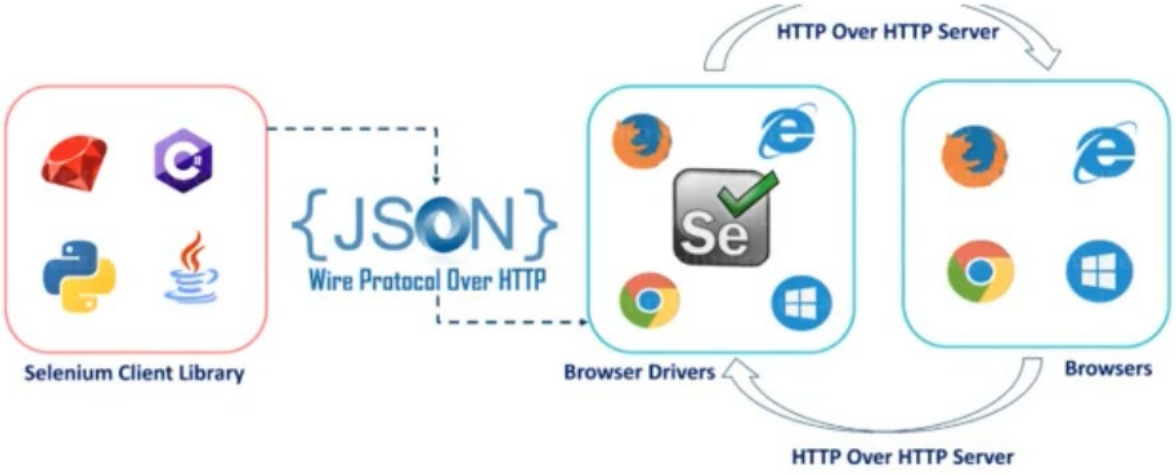
# WebDriver

- If you are beginning with desktop website or mobile website test automation, then you are going to be using WebDriver APIs. WebDriver uses browser automation APIs provided by browser vendors to control the browser and run tests. This is as if a real user is operating the browser. Since WebDriver does not require its API to be compiled with application code; It is not intrusive. Hence, you are testing the same application which you push live

# Selenium 3’s architecture

Selenium 3’s architecture consists of five layers.

1. **Selenium Client Library**
2. **Selenium API**
3. **JSON Wire protocol**
4. **Browser Drivers**
5. **Browser**



# WebDriver Interface

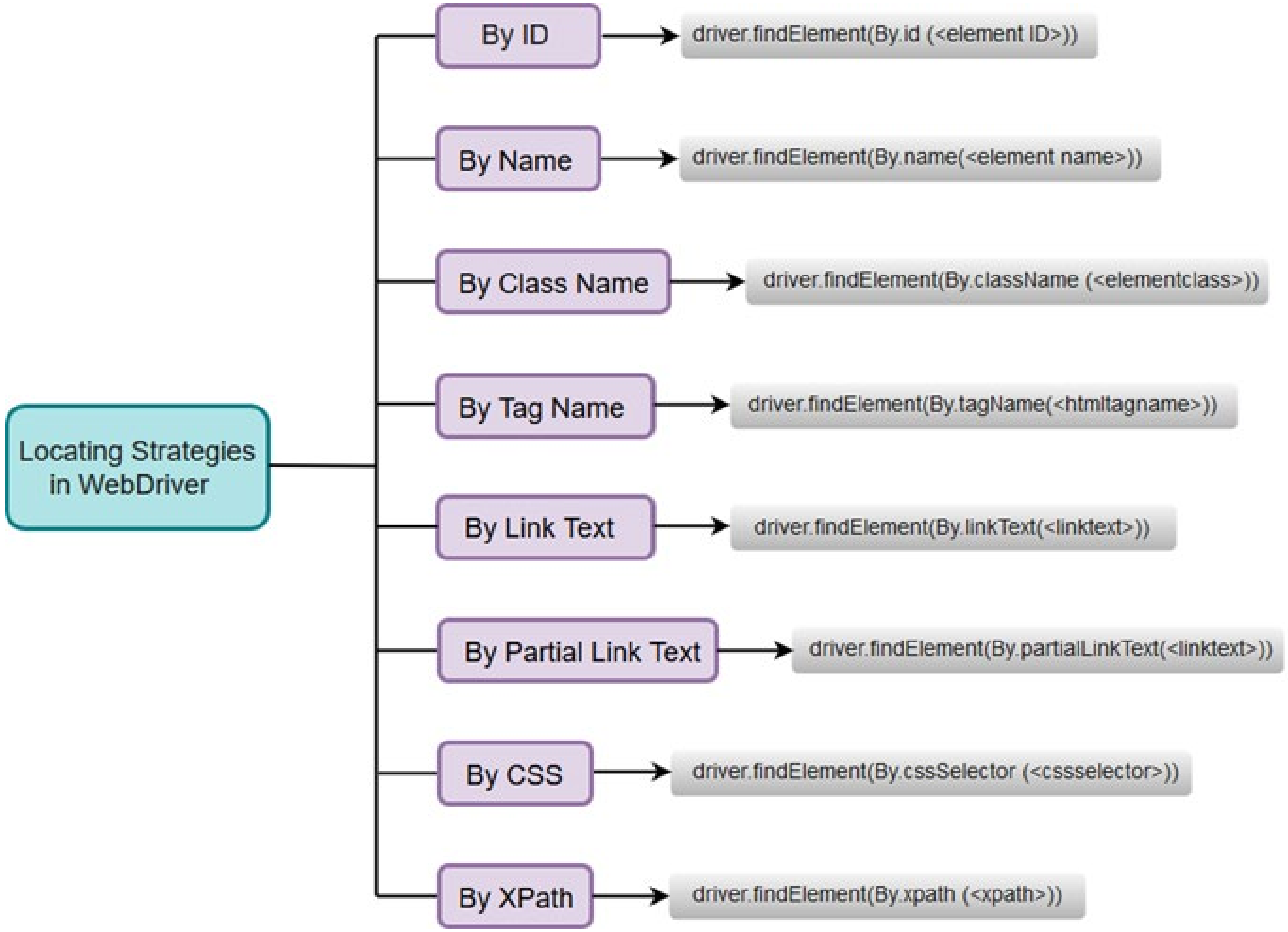
**WebDriver is an Interface which provides Set of Browser Automation methods with empty bodies (Abstract methods)** Classes like

- ChromeDriver,
- FirefoxDriver,
- MicrosoftEdgeDriver ,
- SafariDriver

# Selenium Web Driver Locators

- As part of Automation, Selenium Performs actions (such as click, typing) on the Page HTML Elements.

- The Locators are the way to identify an *HTML* element on a web page.
  Selenium WebDriver uses any of the below locators to identify the element on the page and performs the Action
  - ID
  - Xpath
  - CSS Selector
  - name
  - Class Name
  - Tag Name
  - Link Text
  - Partial Link Text



Locating Strategies in WebDriver

By ID

```
driver.findElement(By.id (<element ID>))
```

By Name

```
driver.findElement(By.name(<element name>))
```

By Class Name

```
driver.findElement(By.className (<elementclass>))
```

By Tag Name

```
driver.findElement(By.tagName(<htmltagname>))
```

By Link Text

```
driver.findElement(By.linkText(<linktext>))
```

By Partial Link Text

```
driver.findElement(By.partialLinkText(<linktext>))
```

By CSS

```
driver.findElement(By.cssSelector (<cssselector>))
```

By XPath

```
driver.findElement(By.xpath (<xpath>))
```

https://www.javatpoint.com/selenium-webdriver-locating-strategies

# First Selenium Script (java)

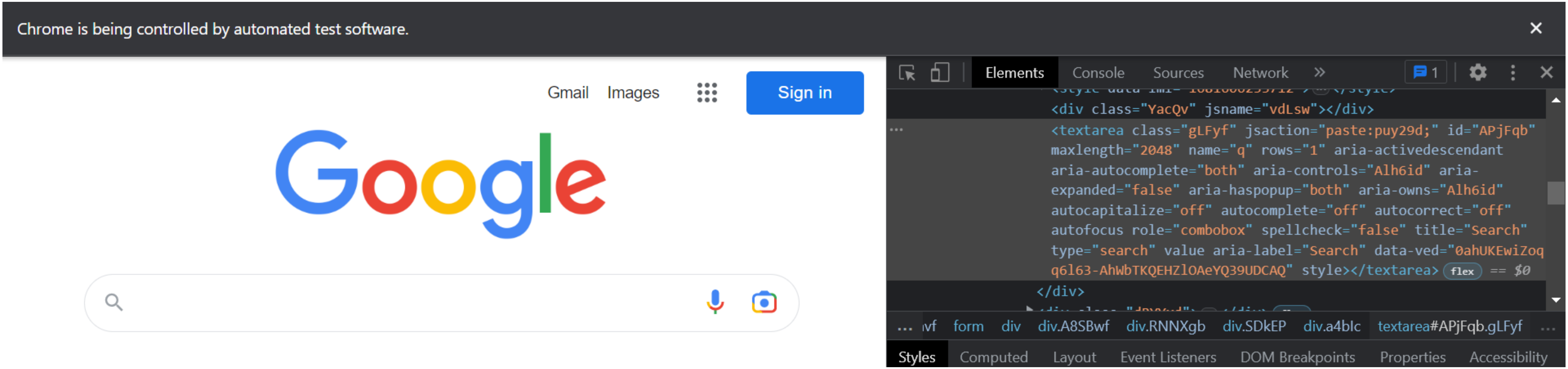
- Use Eclipse or STS
- Create a java project
- Add selenium jars to your project [ use new folder]
  - https://www.selenium.dev/downloads/
- Download Chrome Driver
  - https://chromedriver.chromium.org/downloads [ check your chrome version]

- Write a script to open selenium website and close ; use Chrome driver

# Sample Java : Open a browser

```
public class BorwaserTest {
public static void main(String[] args) {
System.setProperty("webdriver.chrome.driver", ".\\Driver\\chromedriver.exe");
WebDriver driver=new ChromeDriver();
driver.get("https://www.google.com");
driver.manage().window().maximize();
WebElement searchKey = driver.findElement(By.name("q"));
searchKey.sendKeys("selenium");
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(30));
}
}
```

# Finding web elements

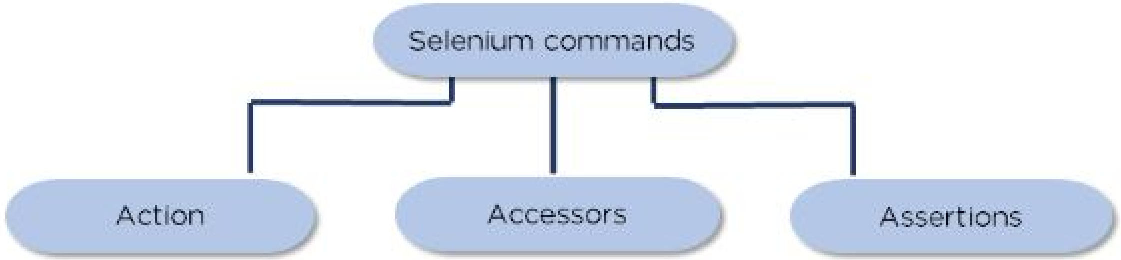

Chrome is being controlled by automated test software.

```
<div class="YacQv" jsname="vdLsw"></div>
<textarea class="gLFyf" jsaction="paste:puy29d;" id="APjFqb"
maxlength="2048" name="q" rows="1" aria-activedescendant
aria-autocomplete="both" aria-controls="Alh6id" aria-
expanded="false" aria-haspopup="both" aria-owns="Alh6id"
autocapitalize="off" autocomplete="off" autocorrect="off"
autofocus role="combobox" spellcheck="false" title="Search"
type="search" value aria-label="Search" data-ved="0ahUKEwiZoq
q6l63-AhWbTKQEHZlOAeYQ39UDCAQ" style></textarea> flex == $0
</div>
```

```
… ıvf  form  div  div.A8SBwf  div.RNNXgb  div.SDkEP  div.a4bIc  textarea#APjFqb.gLFyf  …
```

- While you are on the search field go to inspect
- Use the locator `input[name='q']` to ensure there is one element

# Selenium Commands



# Actions

- Actions - Commands that interact directly with the application:
- clickAndWait(),
- typeAndWait()

# Assessors

Enable the user to store certain values to a user-defined variable:

- storeTitle()

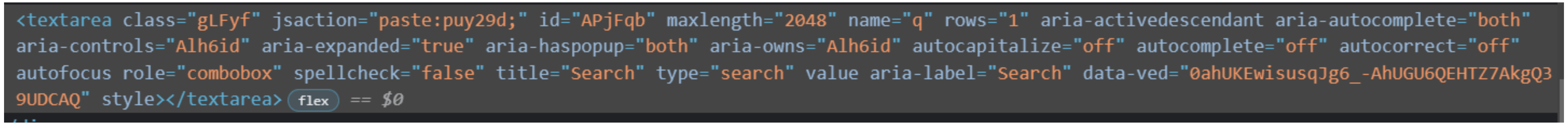
# Assertions

Assertions - Verifies the current state of the application along with an expected state. There are different types of assertions:

1. Assert command makes sure that the test execution is terminated in case of failure
2. Verify command ensures script execution even if the verification has failed
3. WaitFor command waits for a specific condition to be met before executing the next test step

# Creating Test Cases Manually

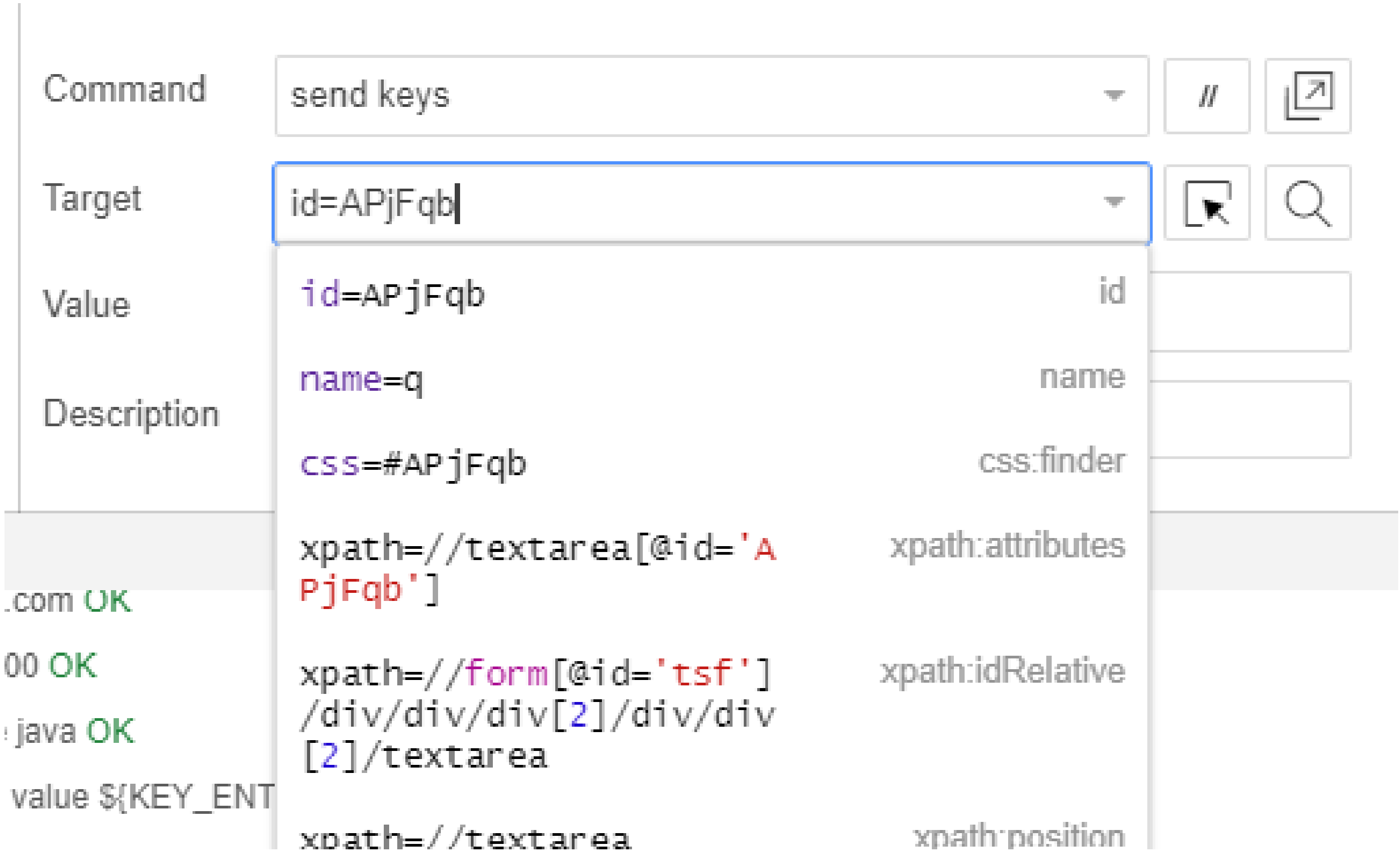
- Inspect while your run Selenium IDE



```
<textarea class="gLFyf" jsaction="paste:puy29d;" id="APjFqb" maxlength="2048" name="q" rows="1" aria-activedescendant aria-autocomplete="both"
aria-controls="Alh6id" aria-expanded="true" aria-haspopup="both" aria-owns="Alh6id" autocapitalize="off" autocomplete="off" autocorrect="off"
autofocus role="combobox" spellcheck="false" title="Search" type="search" value aria-label="Search" data-ved="0ahUKEwisusqJg6_-AhUGU6QEHTZ7AkgQ3
9UDCAQ" style></textarea> flex == $0
```

https://www.google.com/search?q=selenium+ide+search+with+chrome&rlz=1C1CHBF_enCA1014CA1014&sxsrf=APwXEdddfwogwx_4GBi4yFh_r4Qu_0lKaQ:1681670042411&source=lnms&tbm=vid&sa=X&ved=2ahUKEwjajoeMha_-AhXwVaQEHReWCcQQ_AUoAnoECAIQBA&biw=1024&bih=481&dpr=1.88#fpstate=ive&vld=cid:24d8f337,vid:m4KpTvEz3vg

# Multiple Locators for the same Element



Command: send keys

Target: id=APjFqb

Value

Description

| | |
|---|---|
| id=APjFqb | id |
| name=q | name |
| css=#APjFqb | css:finder |
| xpath=//textarea[@id='APjFqb'] | xpath:attributes |
| xpath=//form[@id='tsf']/div/div/div[2]/div/div[2]/textarea | xpath:idRelative |
| xpath=//textarea | xpath:position |

.com OK

00 OK

java OK

value ${KEY_ENT

As your usecase is to open the browser and search for a given word within Google Home Page, the *Search Box* HTML is as follows:

```
<input class="gLFyf gsfi" maxlength="2048" name="q" type="text" jsaction="paste:puy29d" aria-autocomplete="both" aria-haspopup="false" autocapitalize="off" autocomplete="off" autocorrect="off" autofocus="" role="combobox" spellcheck="false" title="Search" value="" aria-label="Search" data-ved="0ahUKEwiq_ZvUyJvrAhXhzjgGHXBjBx4Q39UDCAQ">
```

So to identify the `<input>` box you can use either of the following Locator Strategies:

- *Name*:

```
driver.FindElement(By.Name("q")).SendKeys("gbqfq");
```

- *CssSelector*:

```
driver.FindElement(By.CssSelector("[name='q']")).SendKeys("APPLES");
```

- *XPath*:

```
driver.FindElement(By.XPath("//*[@name='q']")).SendKeys("APPLES");
```