45426: Teste e Qualidade de Software

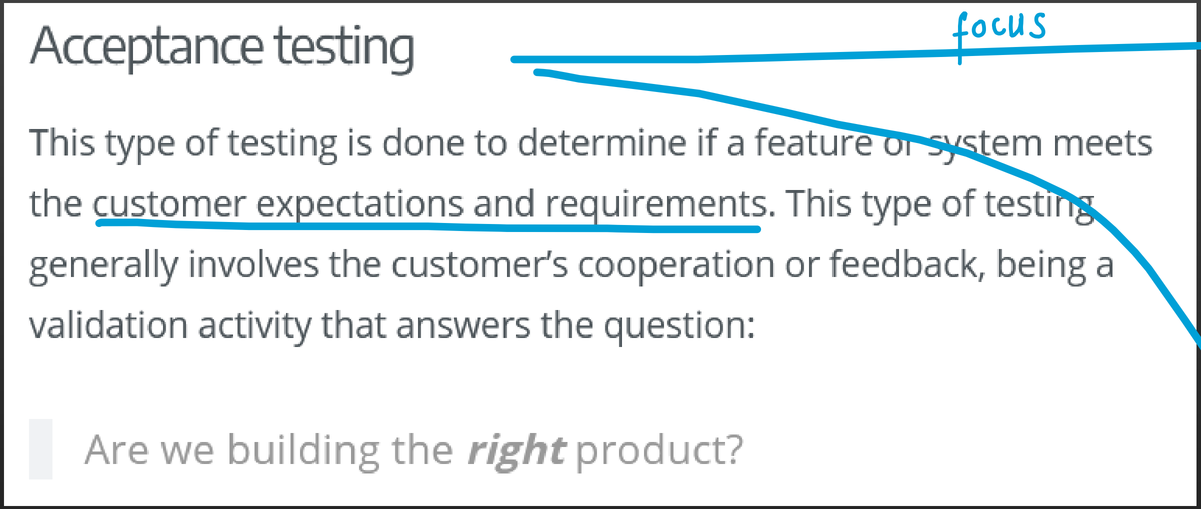
UAT with web automation using the Selenium framework

Ilídio Oliveira

v2024-03-05

universidade de aveiro
departamento de eletrónica,
telecomunicações e informática
deti

User acceptance testing (UAT)



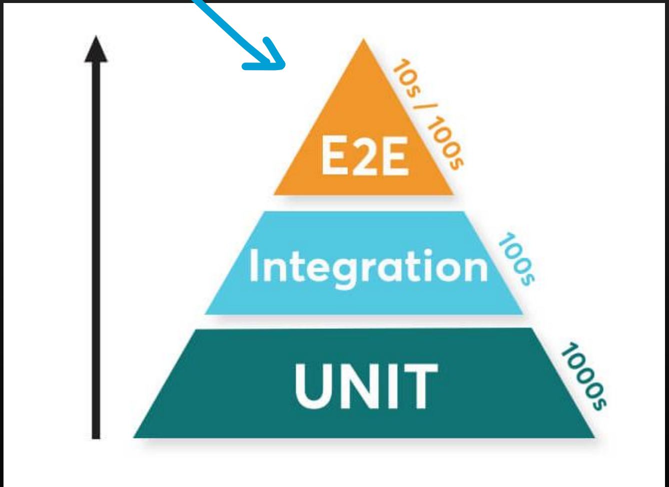
Acceptance testing

This type of testing is done to determine if a feature or system meets the customer expectations and requirements. This type of testing generally involves the customer's cooperation or feedback, being a validation activity that answers the question:

> Are we building the ***right*** product?

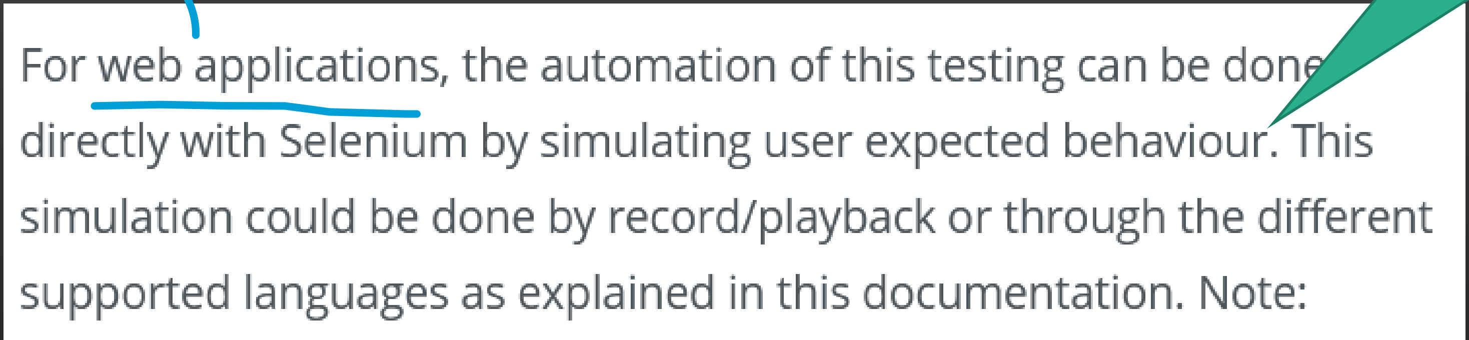
focus

level

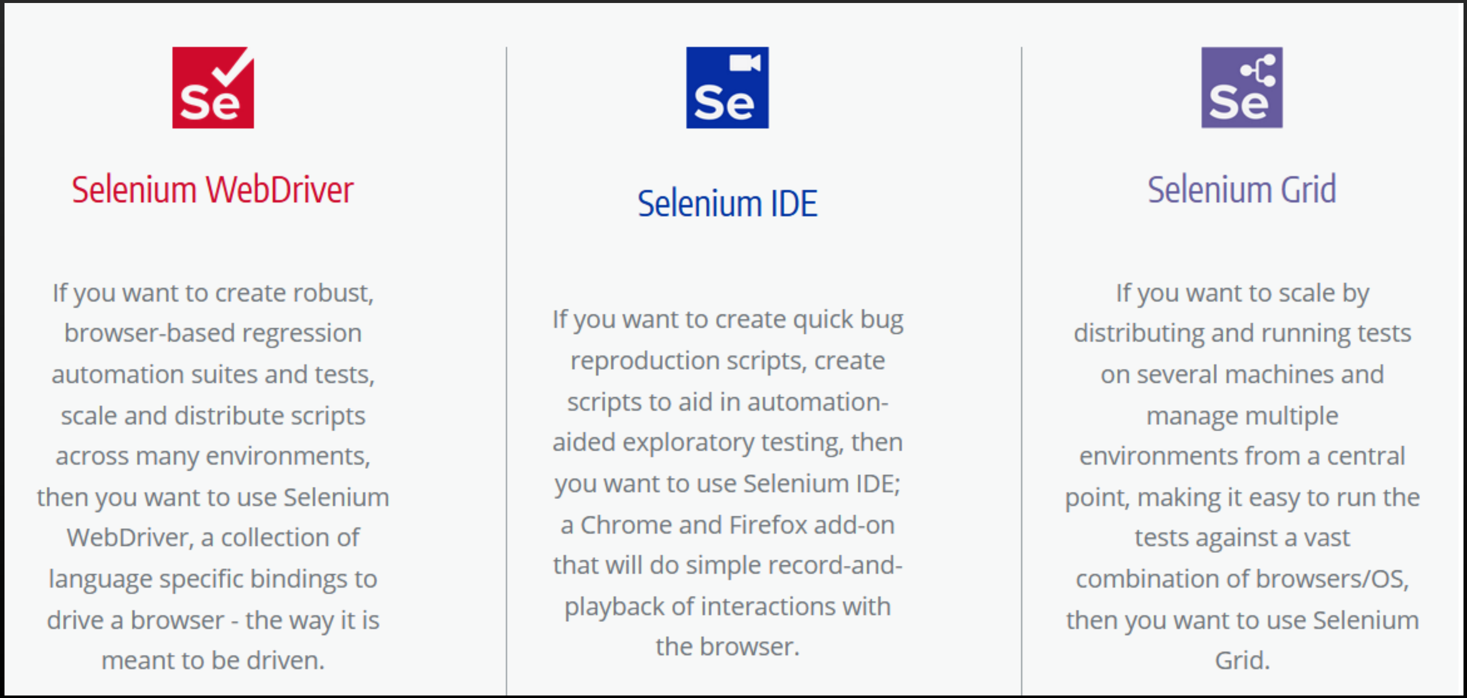


Web UAT ⊂ UAT ⊂ Functional testing

…by controlling the browser programmatically.



For web applications, the automation of this testing can be done directly with Selenium by simulating user expected behaviour. This simulation could be done by record/playback or through the different supported languages as explained in this documentation. Note:



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 and Firefox 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

Browser plugin

Record/replay tests

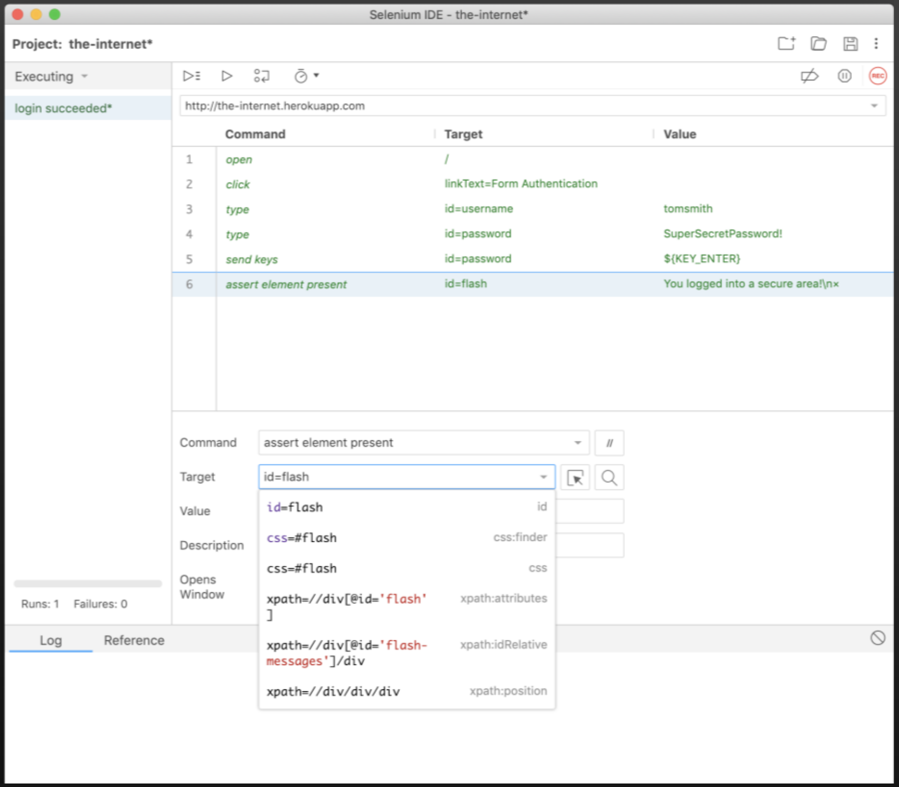
Developer-friendly:

- Interactive web elements picker
- Step-by-step execution
- Can export the test as Java
- These tests can be run in different cloud services

...

Similar alternative:

https://www.katalon.com/katalon-recorder-ide/



http://the-internet.herokuapp.com

	Command	Target	Value
1	open	/	
2	click	linkText=Form Authentication	
3	type	id=username	tomsmith
4	type	id=password	SuperSecretPassword!
5	send keys	id=password	${KEY_ENTER}
6	assert element present	id=flash	You logged into a secure area!\n×

Command: assert element present

Target: id=flash

Value

Description

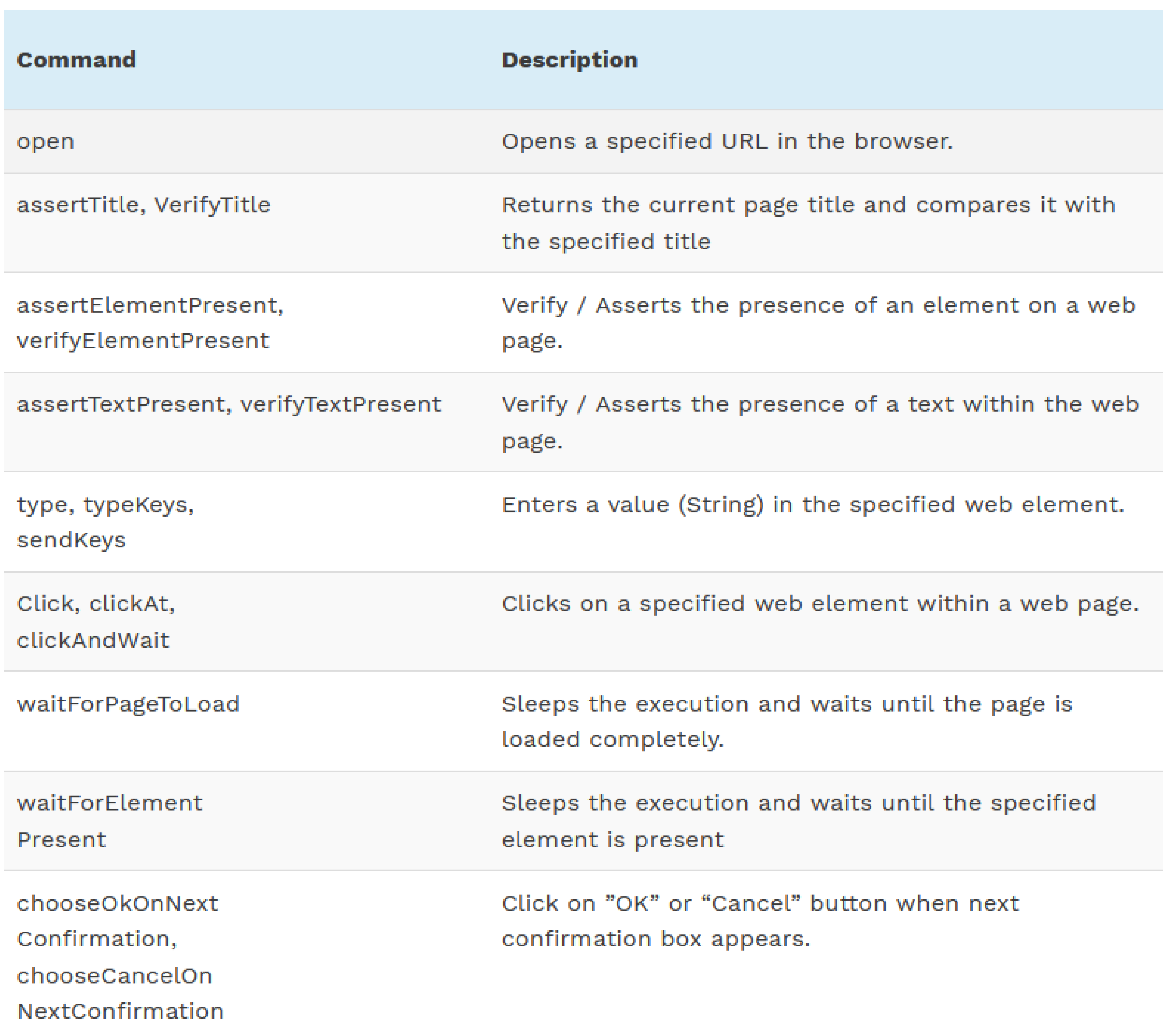
Opens Window

```
id=flash                          id
css=#flash                        css:finder
css=#flash                        css
xpath=//div[@id='flash'           xpath:attributes
]
xpath=//div[@id='flash-           xpath:idRelative
messages']/div
xpath=//div/div/div               xpath:position
```

Runs: 1   Failures: 0

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

Frequent commands

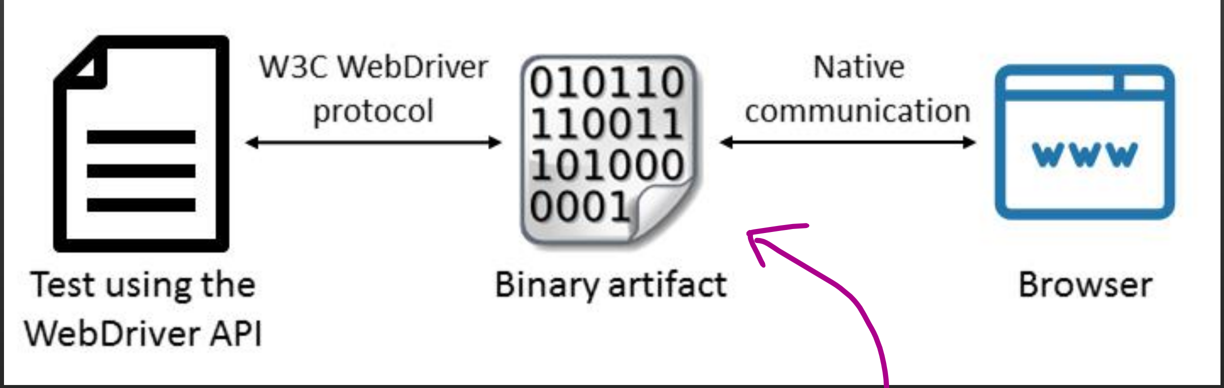


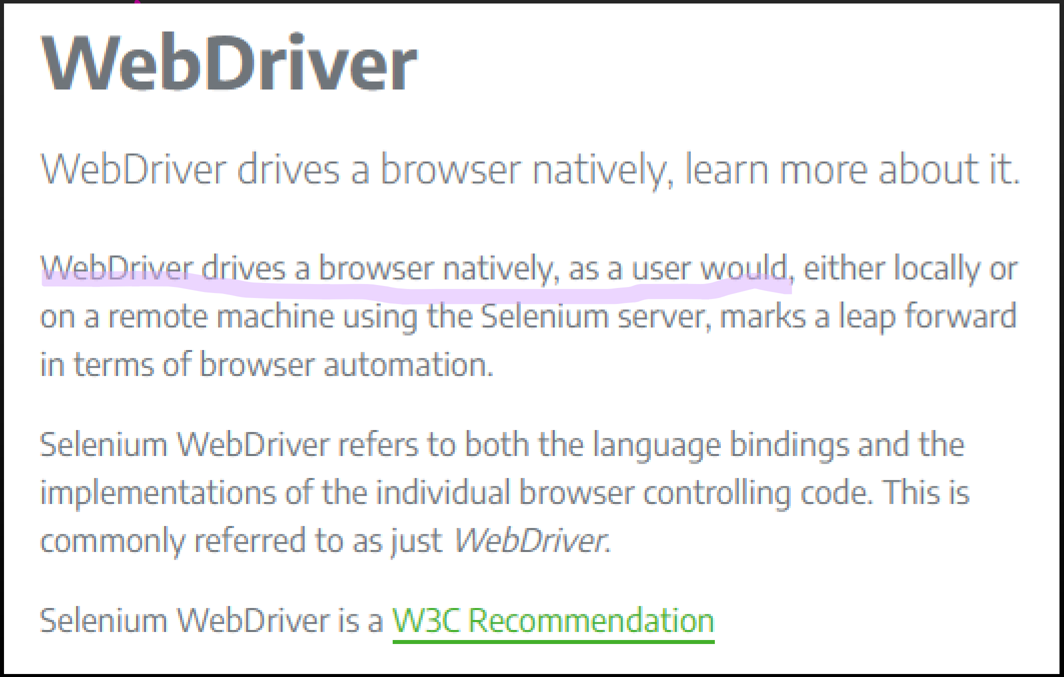
Command	Description
open	Opens a specified URL in the browser.
assertTitle, VerifyTitle	Returns the current page title and compares it with the specified title
assertElementPresent, verifyElementPresent	Verify / Asserts the presence of an element on a web page.
assertTextPresent, verifyTextPresent	Verify / Asserts the presence of a text within the web page.
type, typeKeys, sendKeys	Enters a value (String) in the specified web element.
Click, clickAt, clickAndWait	Clicks on a specified web element within a web page.
waitForPageToLoad	Sleeps the execution and waits until the page is loaded completely.
waitForElement Present	Sleeps the execution and waits until the specified element is present
chooseOkOnNext Confirmation, chooseCancelOn NextConfirmation	Click on "OK" or "Cancel" button when next confirmation box appears.

Step-by-step introduction with examples available from: https://www.softwaretestinghelp.com/selenium-ide-script-selenium-tutorial-3/

Selenium WebDriver for testing

Selenium WebDriver





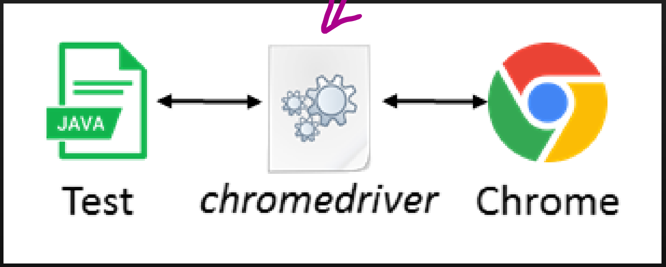
WebDriver

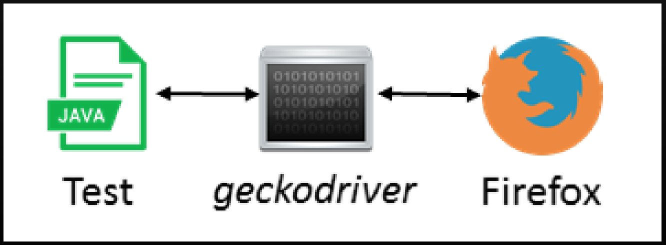
WebDriver drives a browser natively, learn more about it.

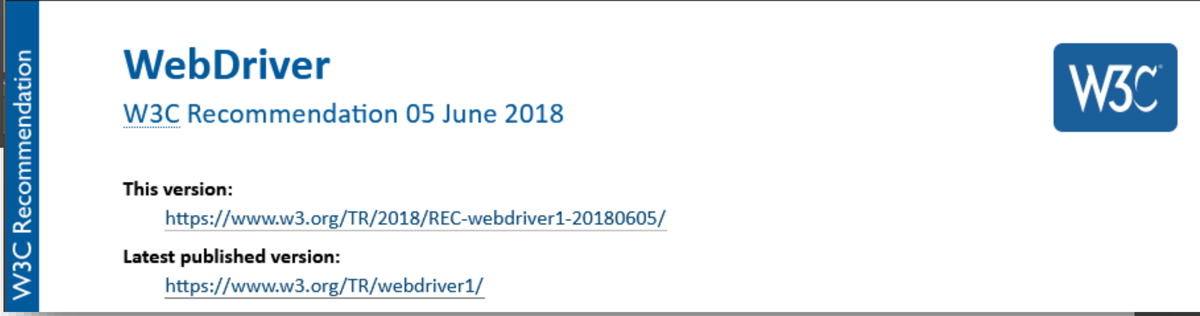
WebDriver drives a browser natively, as a user would, either locally or on a remote machine using the Selenium server, marks a leap forward in terms of browser automation.

Selenium WebDriver refers to both the language bindings and the implementations of the individual browser controlling code. This is commonly referred to as just *WebDriver*.

Selenium WebDriver is a W3C Recommendation







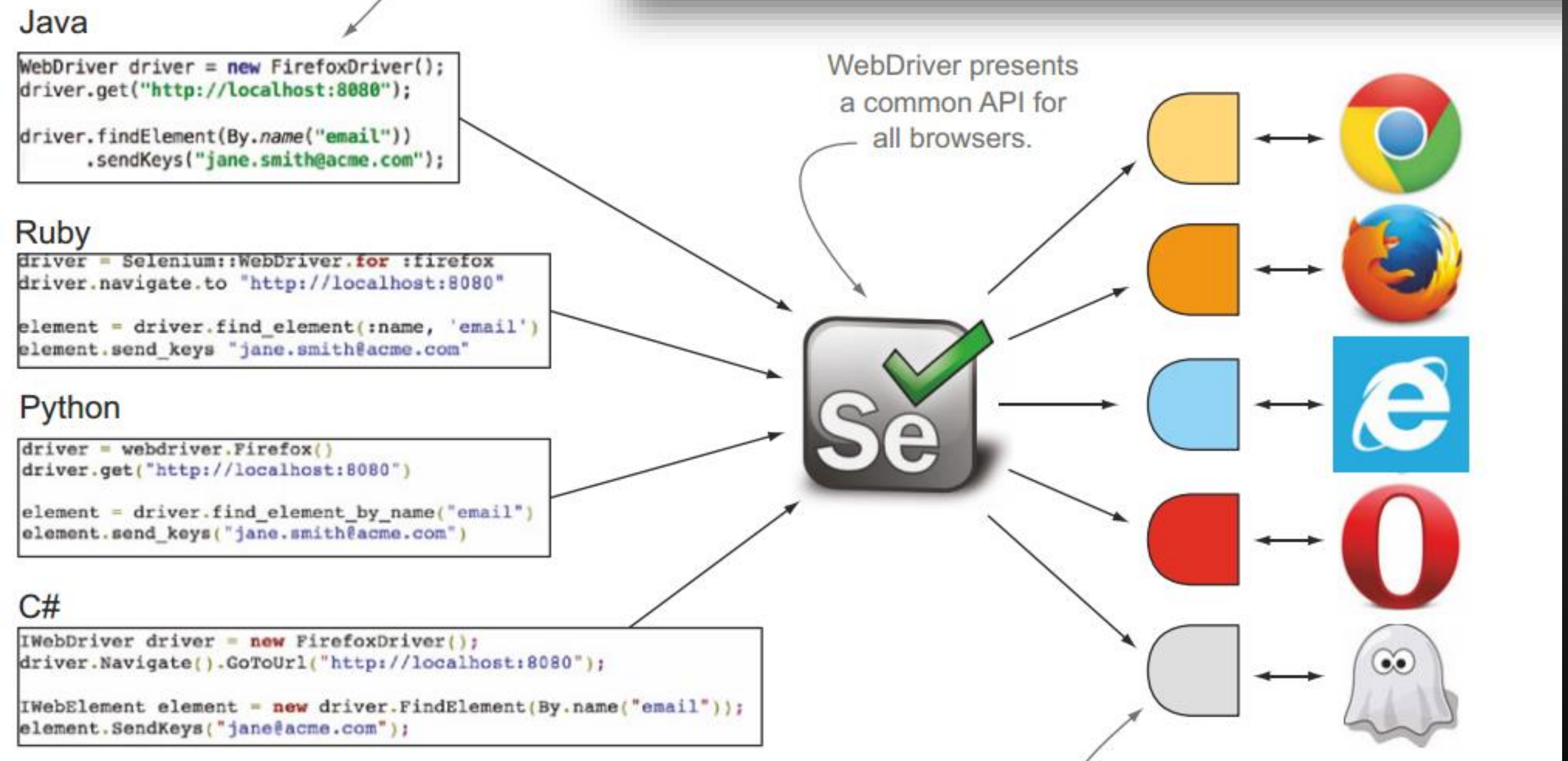
WebDriver

W3C Recommendation 05 June 2018

This version:
https://www.w3.org/TR/2018/REC-webdriver1-20180605/

Latest published version:
https://www.w3.org/TR/webdriver1/

Test code can be in different languages.



Java

```
WebDriver driver = new FirefoxDriver();
driver.get("http://localhost:8080");

driver.findElement(By.name("email"))
      .sendKeys("jane.smith@acme.com");
```

Ruby

```
driver = Selenium::WebDriver.for :firefox
driver.navigate.to "http://localhost:8080"

element = driver.find_element(:name, 'email')
element.send_keys "jane.smith@acme.com"
```

Python

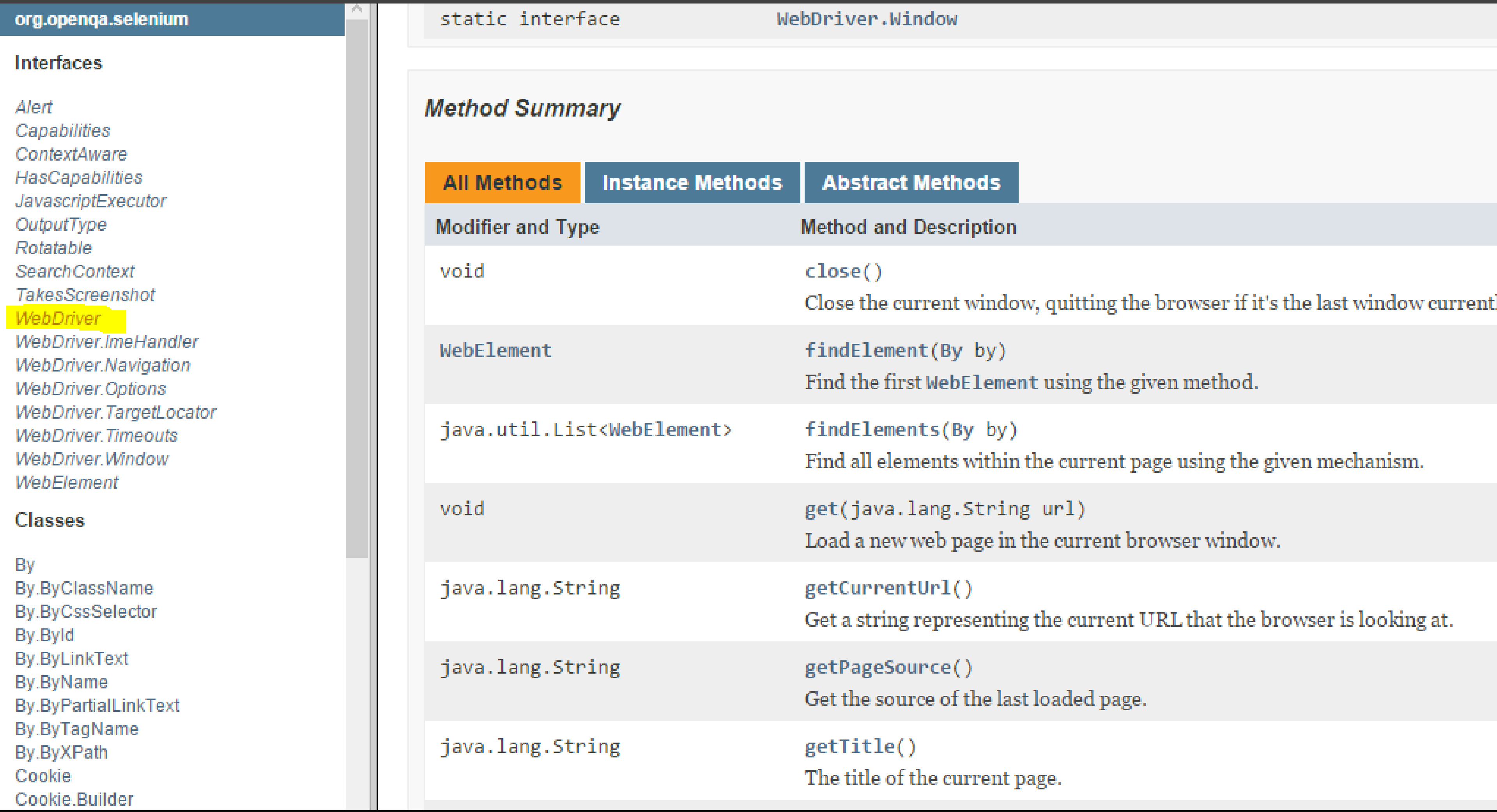
```
driver = webdriver.Firefox()
driver.get("http://localhost:8080")

element = driver.find_element_by_name("email")
element.send_keys("jane.smith@acme.com")
```

C#

```
IWebDriver driver = new FirefoxDriver();
driver.Navigate().GoToUrl("http://localhost:8080");

IWebElement element = new driver.FindElement(By.name("email"));
element.SendKeys("jane@acme.com");
```

WebDriver presents a common API for all browsers.

Each supported browser has a specific driver that implements the common API.



org.openqa.selenium

Interfaces

Alert
Capabilities
ContextAware
HasCapabilities
JavascriptExecutor
OutputType
Rotatable
SearchContext
TakesScreenshot
WebDriver
WebDriver.ImeHandler
WebDriver.Navigation
WebDriver.Options
WebDriver.TargetLocator
WebDriver.Timeouts
WebDriver.Window
WebElement

Classes

By
By.ByClassName
By.ByCssSelector
By.ById
By.ByLinkText
By.ByName
By.ByPartialLinkText
By.ByTagName
By.ByXPath
Cookie
Cookie.Builder

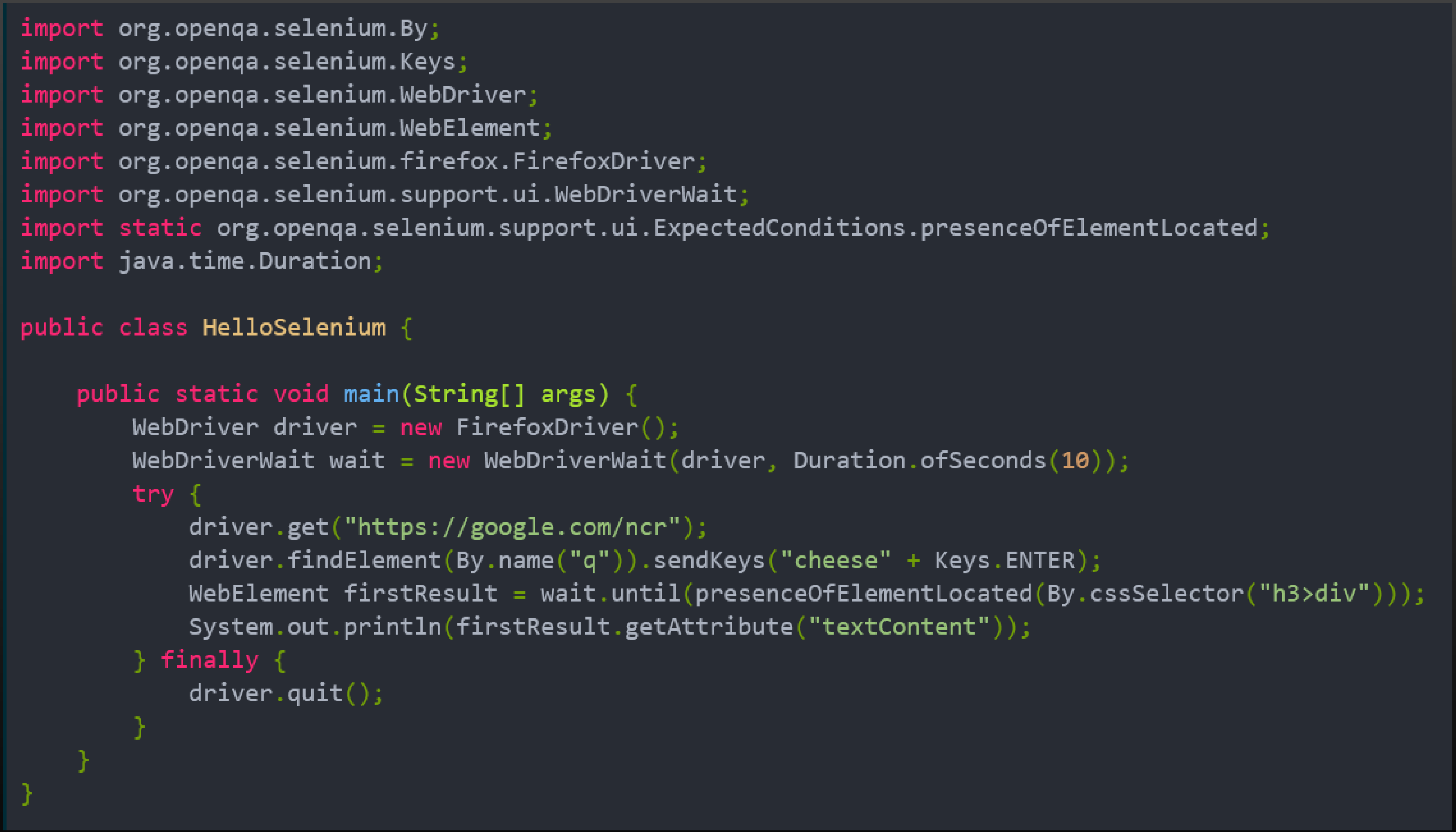
`static interface`	`WebDriver.Window`

Method Summary

Modifier and Type	Method and Description
`void`	`close()` Close the current window, quitting the browser if it's the last window current
`WebElement`	`findElement(By by)` Find the first `WebElement` using the given method.
`java.util.List<WebElement>`	`findElements(By by)` Find all elements within the current page using the given mechanism.
`void`	`get(java.lang.String url)` Load a new web page in the current browser window.
`java.lang.String`	`getCurrentUrl()` Get a string representing the current URL that the browser is looking at.
`java.lang.String`	`getPageSource()` Get the source of the last loaded page.
`java.lang.String`	`getTitle()` The title of the current page.

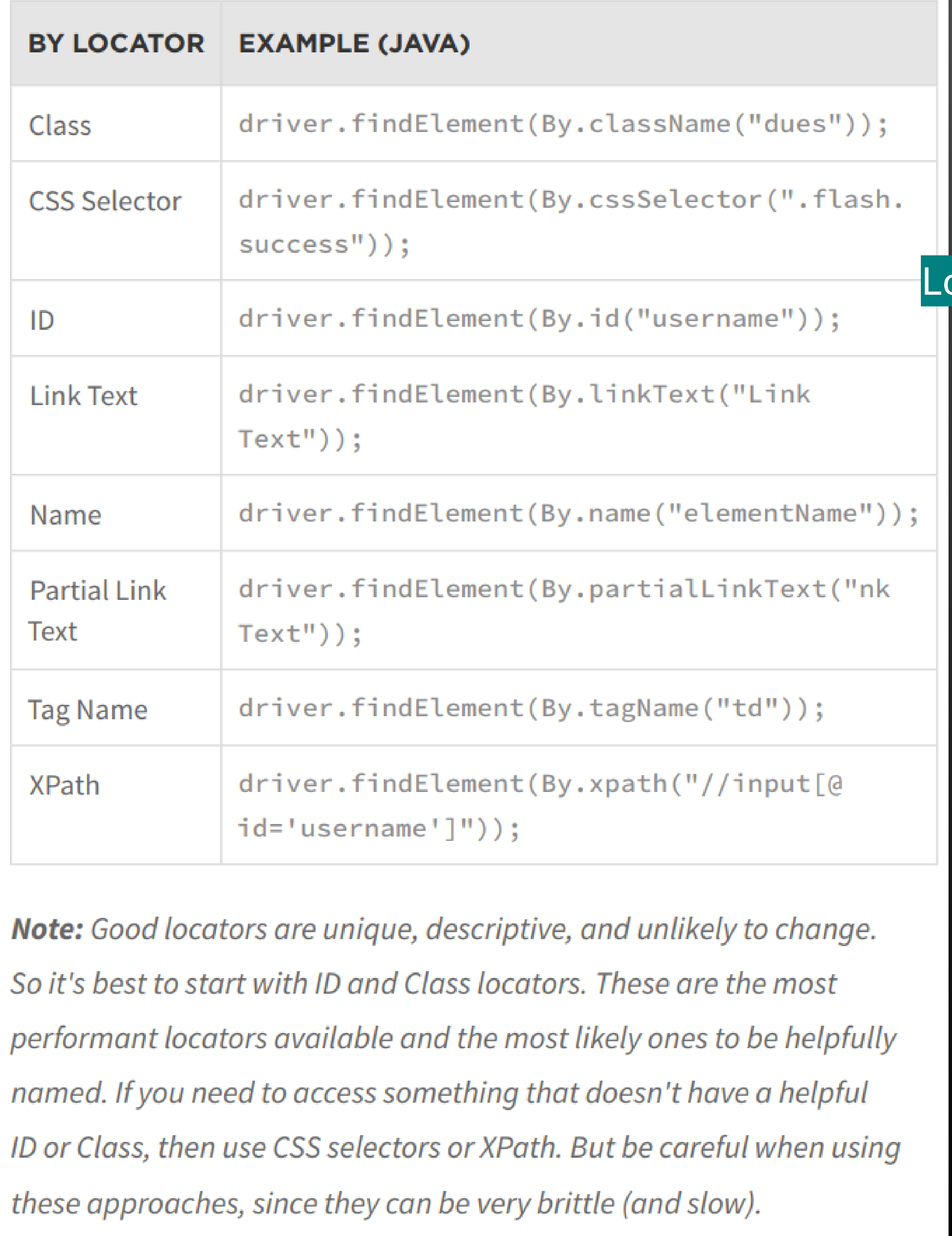
https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/WebDriver.html

Automation sample (not as a Test)



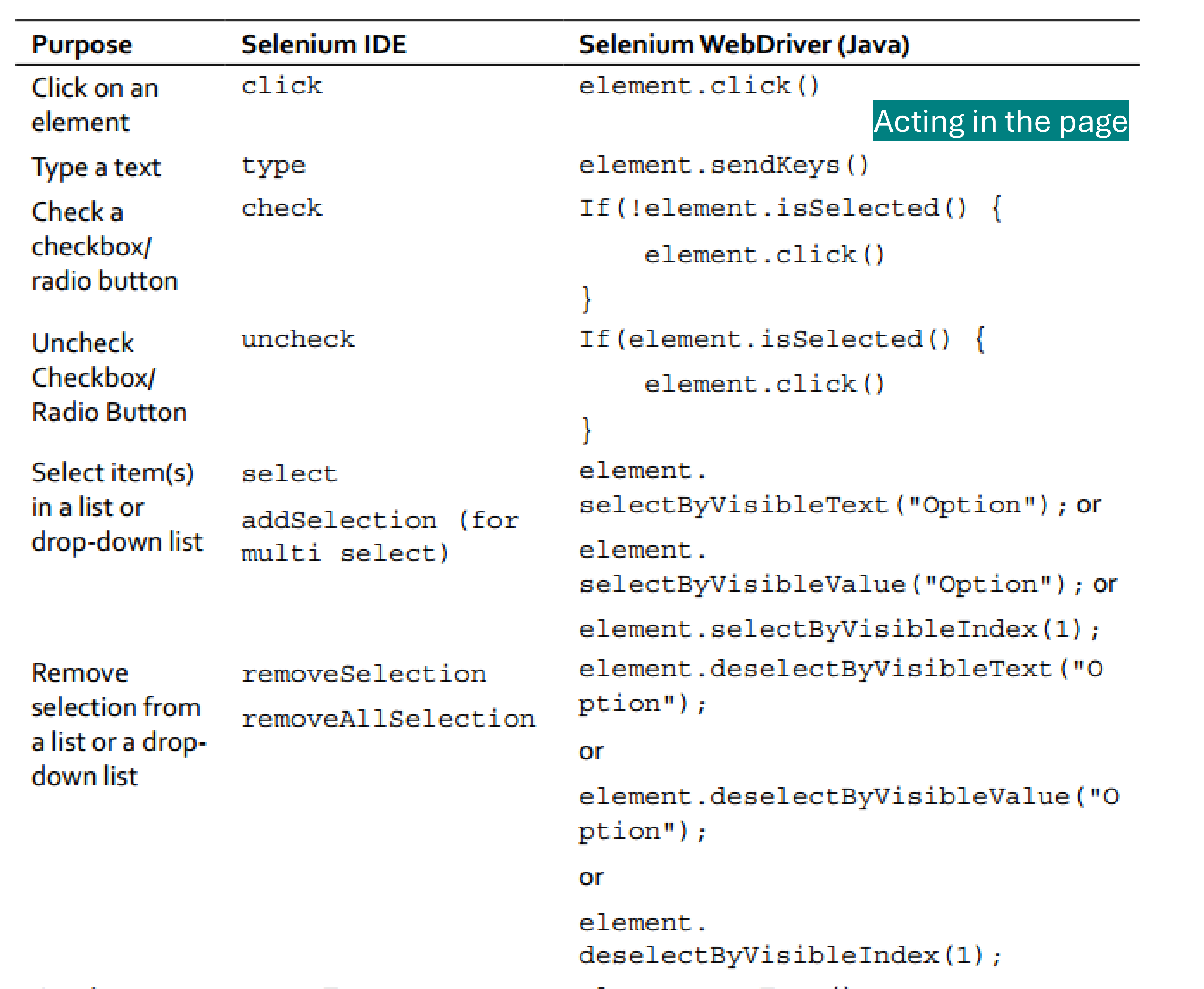
```java
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import static org.openqa.selenium.support.ui.ExpectedConditions.presenceOfElementLocated;
import java.time.Duration;

public class HelloSelenium {

    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
        try {
            driver.get("https://google.com/ncr");
            driver.findElement(By.name("q")).sendKeys("cheese" + Keys.ENTER);
            WebElement firstResult = wait.until(presenceOfElementLocated(By.cssSelector("h3>div")));
            System.out.println(firstResult.getAttribute("textContent"));
        } finally {
            driver.quit();
        }
    }
}
```

Locating elements in the page



BY LOCATOR	EXAMPLE (JAVA)
Class	`driver.findElement(By.className("dues"));`
CSS Selector	`driver.findElement(By.cssSelector(".flash.success"));`
ID	`driver.findElement(By.id("username"));`
Link Text	`driver.findElement(By.linkText("Link Text"));`
Name	`driver.findElement(By.name("elementName"));`
Partial Link Text	`driver.findElement(By.partialLinkText("nk Text"));`
Tag Name	`driver.findElement(By.tagName("td"));`
XPath	`driver.findElement(By.xpath("//input[@id='username']"));`

Note: *Good locators are unique, descriptive, and unlikely to change. So it's best to start with ID and Class locators. These are the most performant locators available and the most likely ones to be helpfully named. If you need to access something that doesn't have a helpful ID or Class, then use CSS selectors or XPath. But be careful when using these approaches, since they can be very brittle (and slow).*



Acting in the page

Purpose	Selenium IDE	Selenium WebDriver (Java)
Click on an element	`click`	`element.click()`
Type a text	`type`	`element.sendKeys()`

Check a checkbox/ radio button	`check`	`If(!element.isSelected() {`
		`    element.click()`
		`}`

Uncheck Checkbox/ Radio Button	`uncheck`	`If(element.isSelected() {`
		`    element.click()`
		`}`

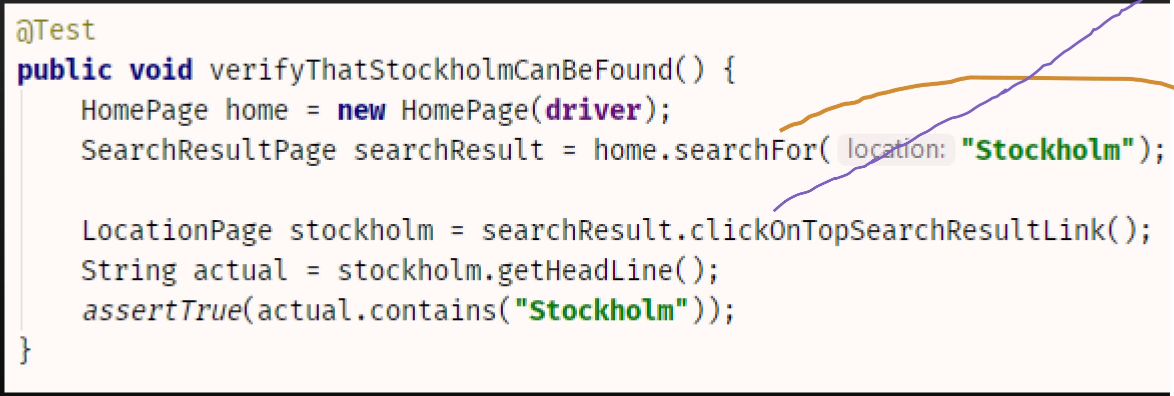
Select item(s) in a list or drop-down list	`select`
	`addSelection (for multi select)`
		`element.selectByVisibleText("Option");` or
		`element.selectByVisibleValue("Option");` or
		`element.selectByVisibleIndex(1);`

Remove selection from a list or a drop-down list	`removeSelection`
	`removeAllSelection`
		`element.deselectByVisibleText("Option");`
		or
		`element.deselectByVisibleValue("Option");`
		or
		`element.deselectByVisibleIndex(1);`

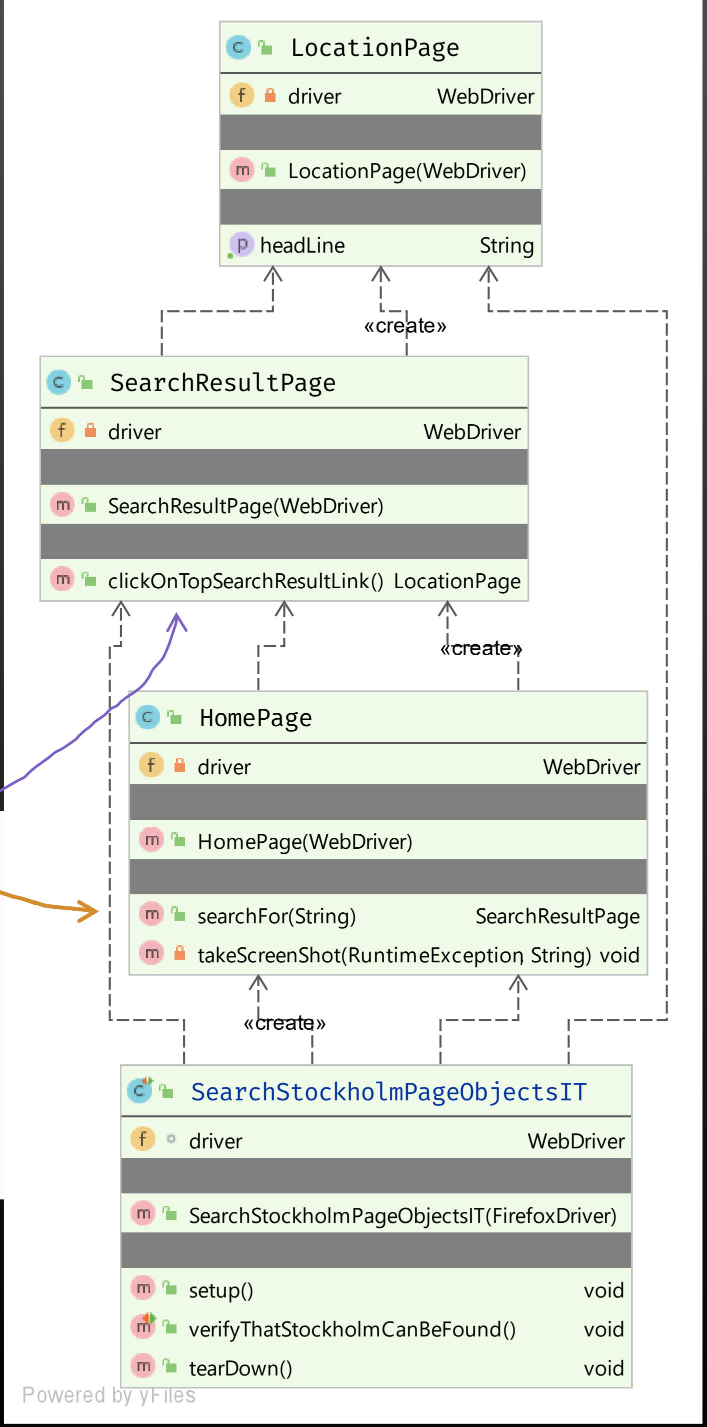
Page Object Model

Create Classes to map the Pages

Encapsulate WebDriver low-level interaction in the target class method



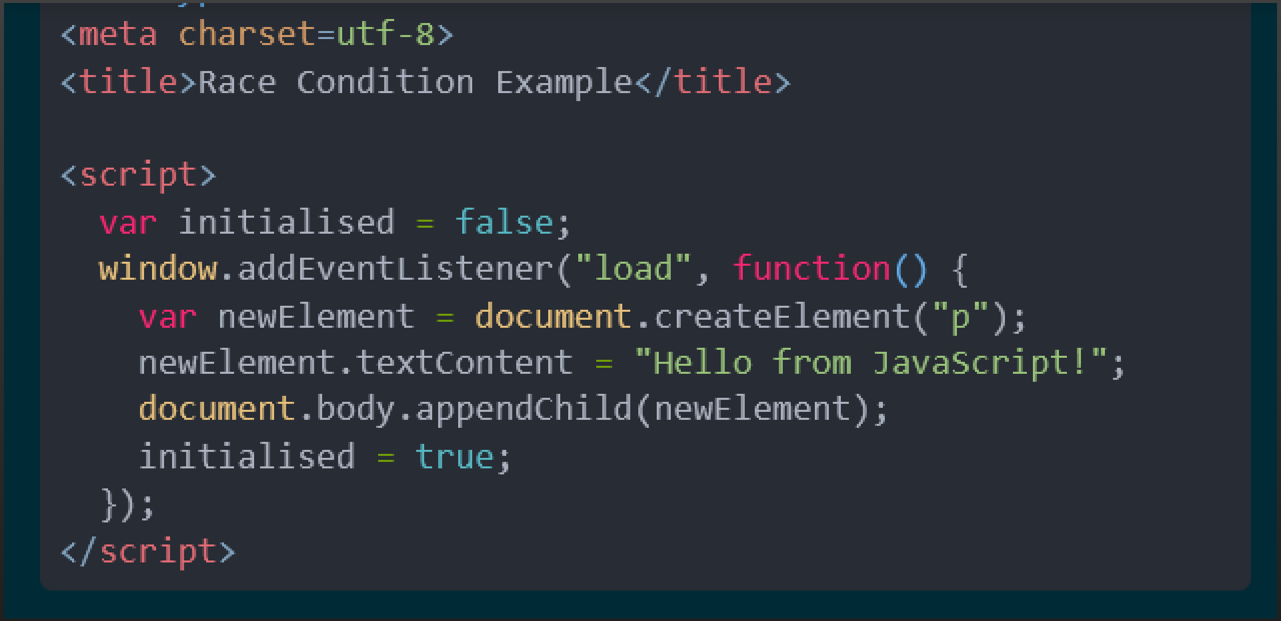
```
@Test
public void verifyThatStockholmCanBeFound() {
    HomePage home = new HomePage(driver);
    SearchResultPage searchResult = home.searchFor( location: "Stockholm");

    LocationPage stockholm = searchResult.clickOnTopSearchResultLink();
    String actual = stockholm.getHeadLine();
    assertTrue(actual.contains("Stockholm"));
}
```



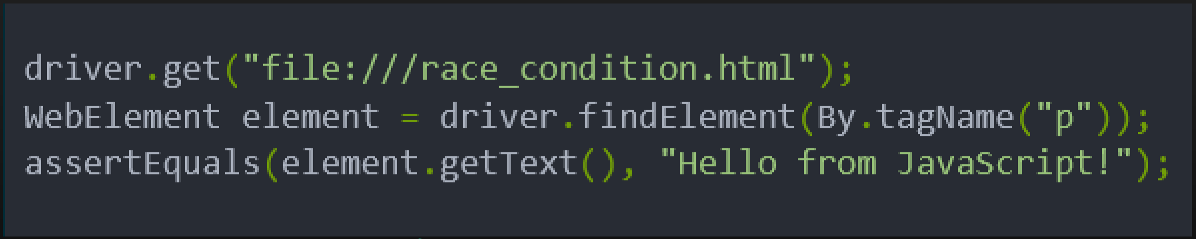
LocationPage
driver WebDriver
LocationPage(WebDriver)
headLine String
«create»
SearchResultPage
driver WebDriver
SearchResultPage(WebDriver)
clickOnTopSearchResultLink() LocationPage
«create»
HomePage
driver WebDriver
HomePage(WebDriver)
searchFor(String) SearchResultPage
takeScreenShot(RuntimeException, String) void
«create»
SearchStockholmPageObjectsIT
driver WebDriver
SearchStockholmPageObjectsIT(FirefoxDriver)
setup() void
verifyThatStockholmCanBeFound() void
tearDown() void
Powered by yFiles

I Oliveira

Race conditions in tests



```
<meta charset=utf-8>
<title>Race Condition Example</title>

<script>
  var initialised = false;
  window.addEventListener("load", function() {
    var newElement = document.createElement("p");
    newElement.textContent = "Hello from JavaScript!";
    document.body.appendChild(newElement);
    initialised = true;
  });
</script>
```



```
driver.get("file:///race_condition.html");
WebElement element = driver.findElement(By.tagName("p"));
assertEquals(element.getText(), "Hello from JavaScript!");
```

Might cause intermittent results…

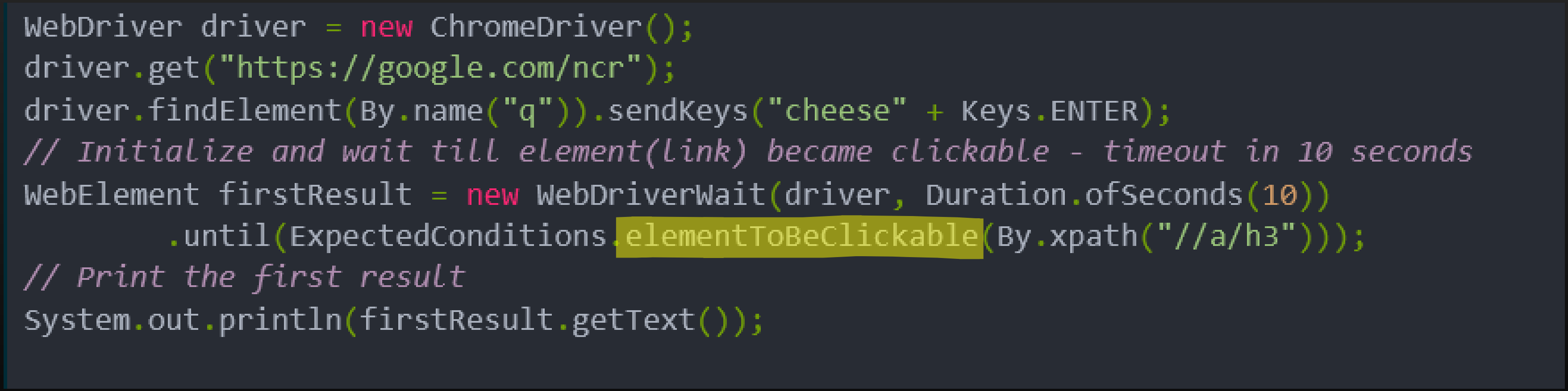
https://www.selenium.dev/documentation/en/webdriver/waits/

Explicit waits

Normal instruction set available on the WebElement (e.g.: *WebElement.click* and *WebElement.sendKeys)* are guaranteed to be synchronous

When employing a wait, you are using what is commonly referred to as an *explicit wait*:

- allow your code to halt program execution until the *condition* resolves (or times out).



```java
WebDriver driver = new ChromeDriver();
driver.get("https://google.com/ncr");
driver.findElement(By.name("q")).sendKeys("cheese" + Keys.ENTER);
// Initialize and wait till element(link) became clickable - timeout in 10 seconds
WebElement firstResult = new WebDriverWait(driver, Duration.ofSeconds(10))
        .until(ExpectedConditions.elementToBeClickable(By.xpath("//a/h3")));
// Print the first result
System.out.println(firstResult.getText());
```

https://www.selenium.dev/documentation/en/webdriver/waits/

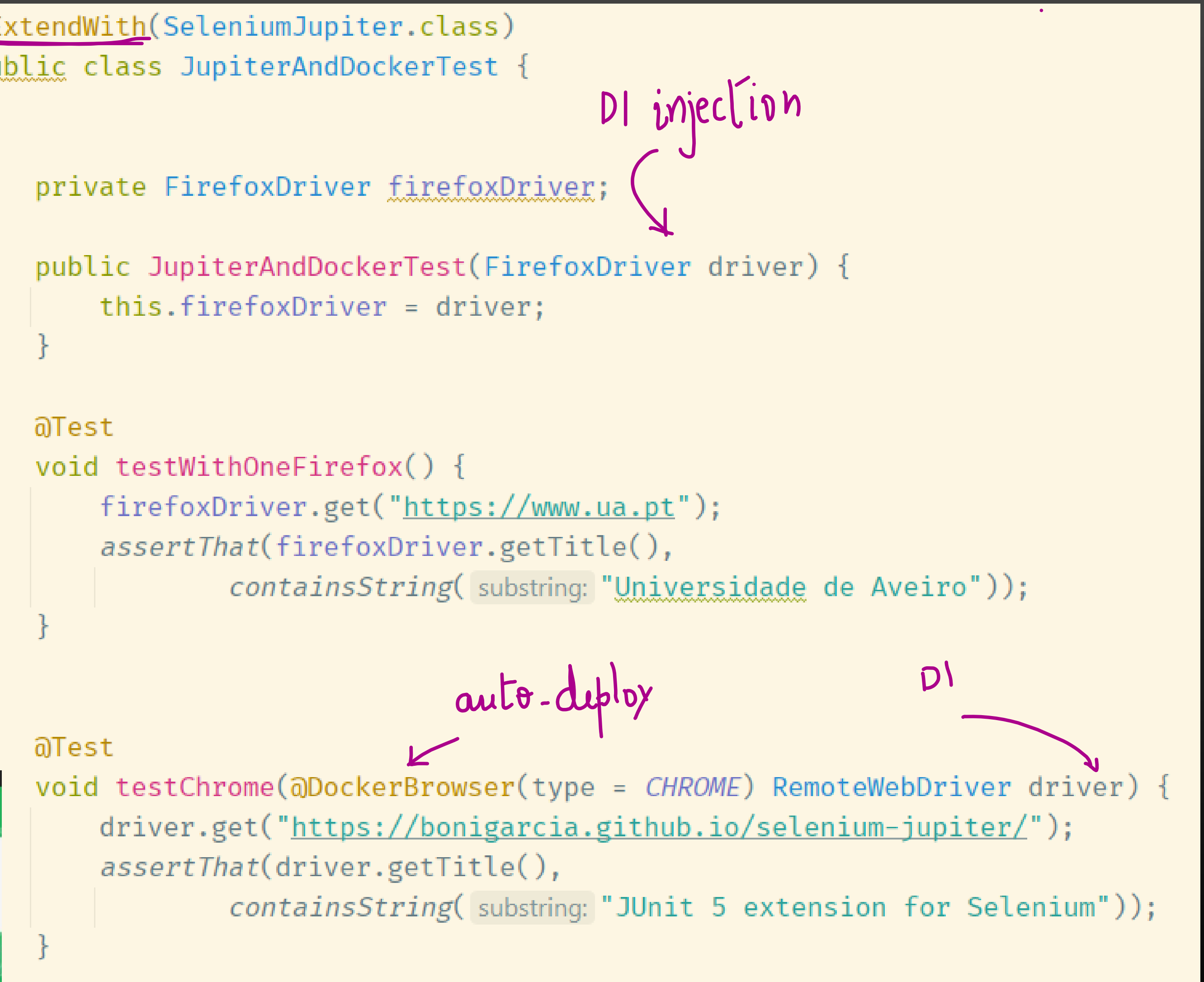
Selenium-Jupiter (JUnit 5 extension)

Get browser (driver) instance with DI (at constructor or parameter levels)

Automatic connect/close browser

Can retrieve browser driver automatically

Auto-deploy containers and run tests “remotely“ (RemoteWebDriver mode)



```java
@ExtendWith(SeleniumJupiter.class)
public class JupiterAndDockerTest {


    private FirefoxDriver firefoxDriver;

    public JupiterAndDockerTest(FirefoxDriver driver) {
        this.firefoxDriver = driver;
    }

    @Test
    void testWithOneFirefox() {
        firefoxDriver.get("https://www.ua.pt");
        assertThat(firefoxDriver.getTitle(),
                containsString( substring: "Universidade de Aveiro"));
    }


    @Test
    void testChrome(@DockerBrowser(type = CHROME) RemoteWebDriver driver) {
        driver.get("https://bonigarcia.github.io/selenium-jupiter/");
        assertThat(driver.getTitle(),
                containsString( substring: "JUnit 5 extension for Selenium"));
    }
```

DI injection
auto-deploy
DI

Selenium Conf
VIRTUAL 2020
Intermediate Workshop
Developing Selenium tests with JUnit 5
Boni García
Assistant Professor @ Universidad Carlos III de Madrid
@boni_gg
This session brought to you by
BrowserStack
2020.seleniumconf.in | @seleniumconf | 10-12 September, 2020

https://bonigarcia.github.io/selenium-jupiter/

Suggested links

Very comprehensive tutorials

Selenium doc: https://www.selenium.dev/

Selenium-Jupiter doc: https://bonigarcia.github.io/selenium-jupiter/