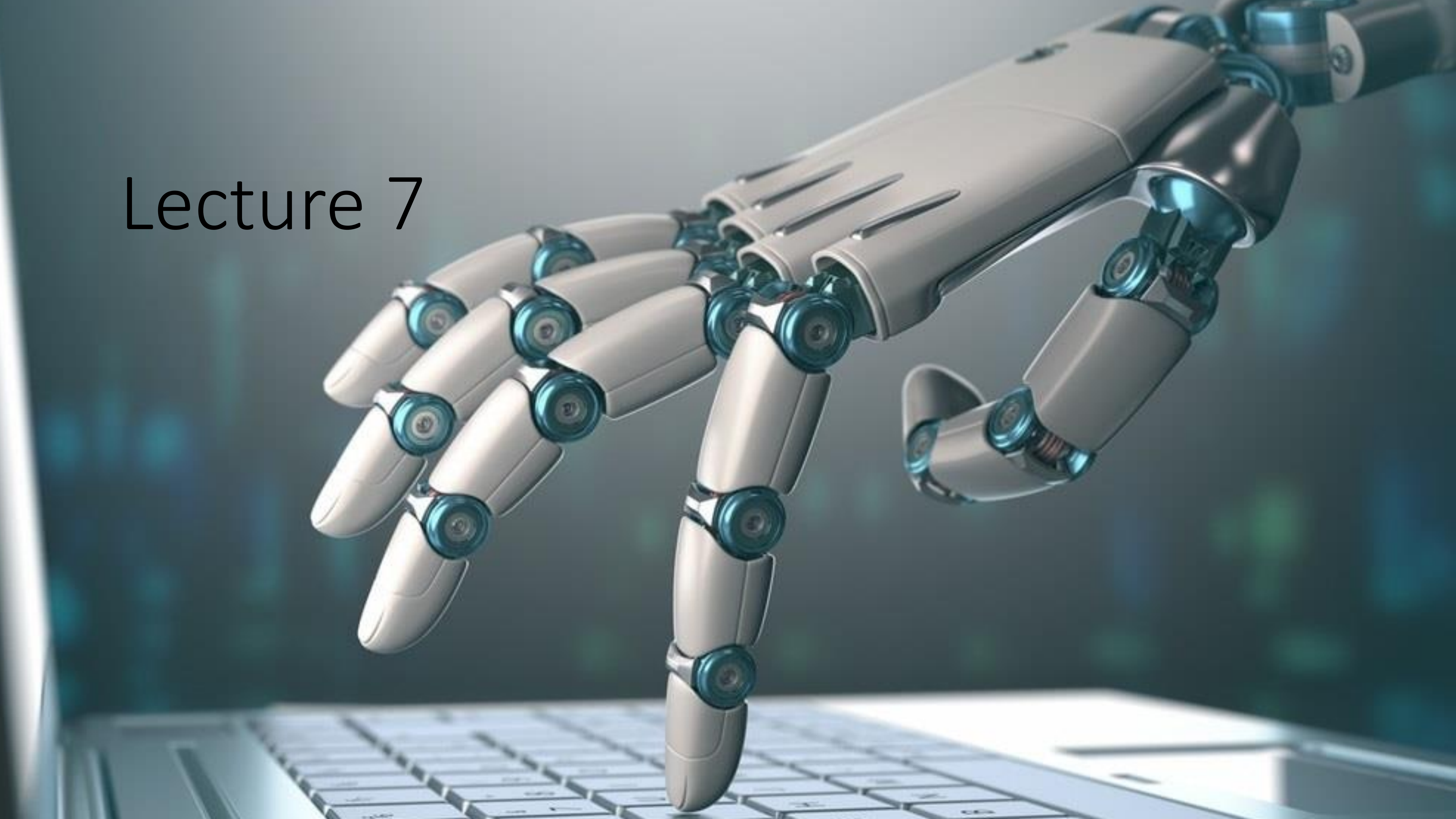
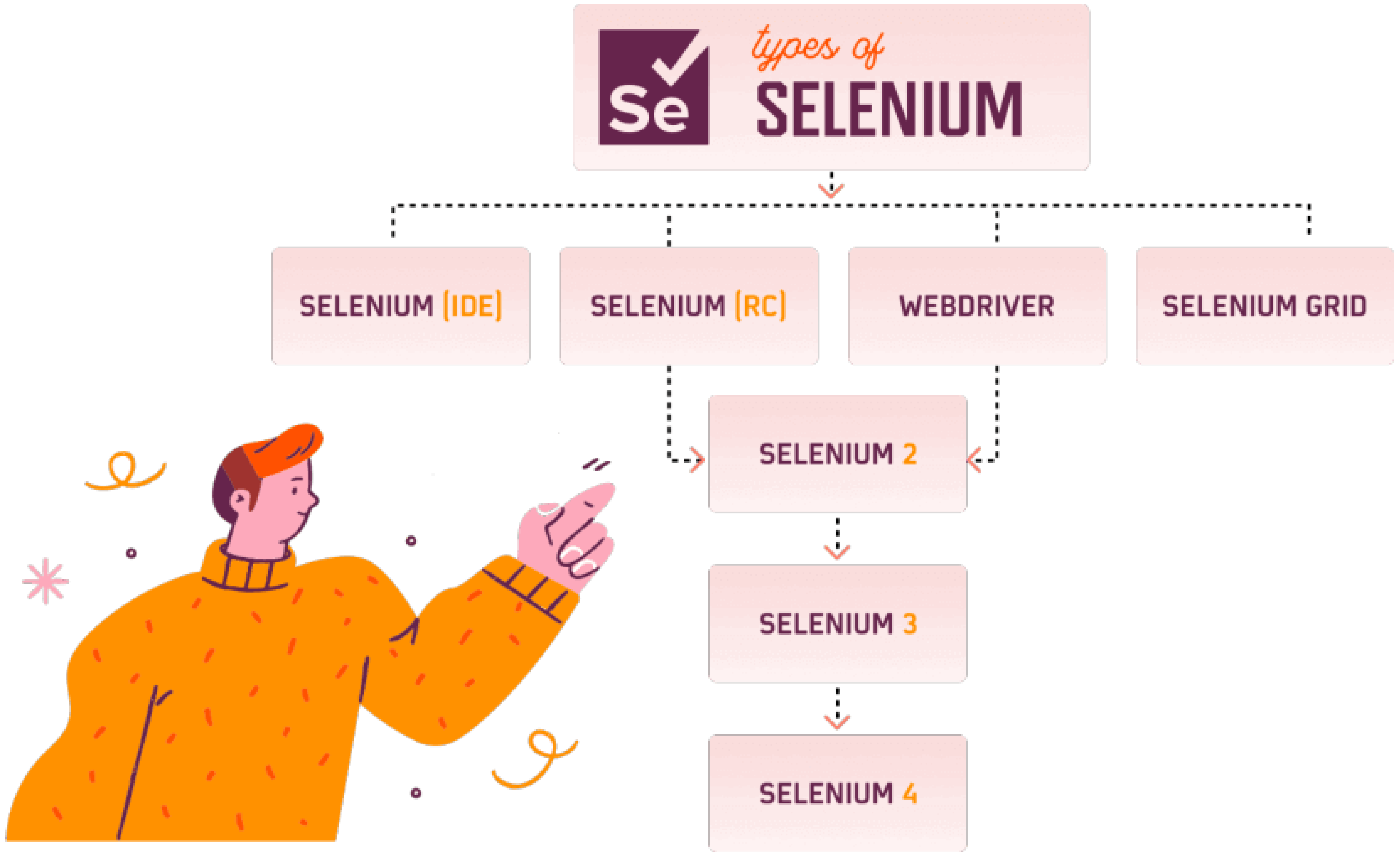


Lecture 7

Agenda

- **Selenium Web driver**
- Locators
- Xpath
- Css selector
- iframe
- Practice

Selenium



It all started in 2004 with Jason Huggins, an engineer at Thoughtworks. While testing, he realized that manual testing was inefficient. So, he created a program using Javascript to automate repetitive tasks. Selenium Core is the name of this open-source program used for automating different web applications.

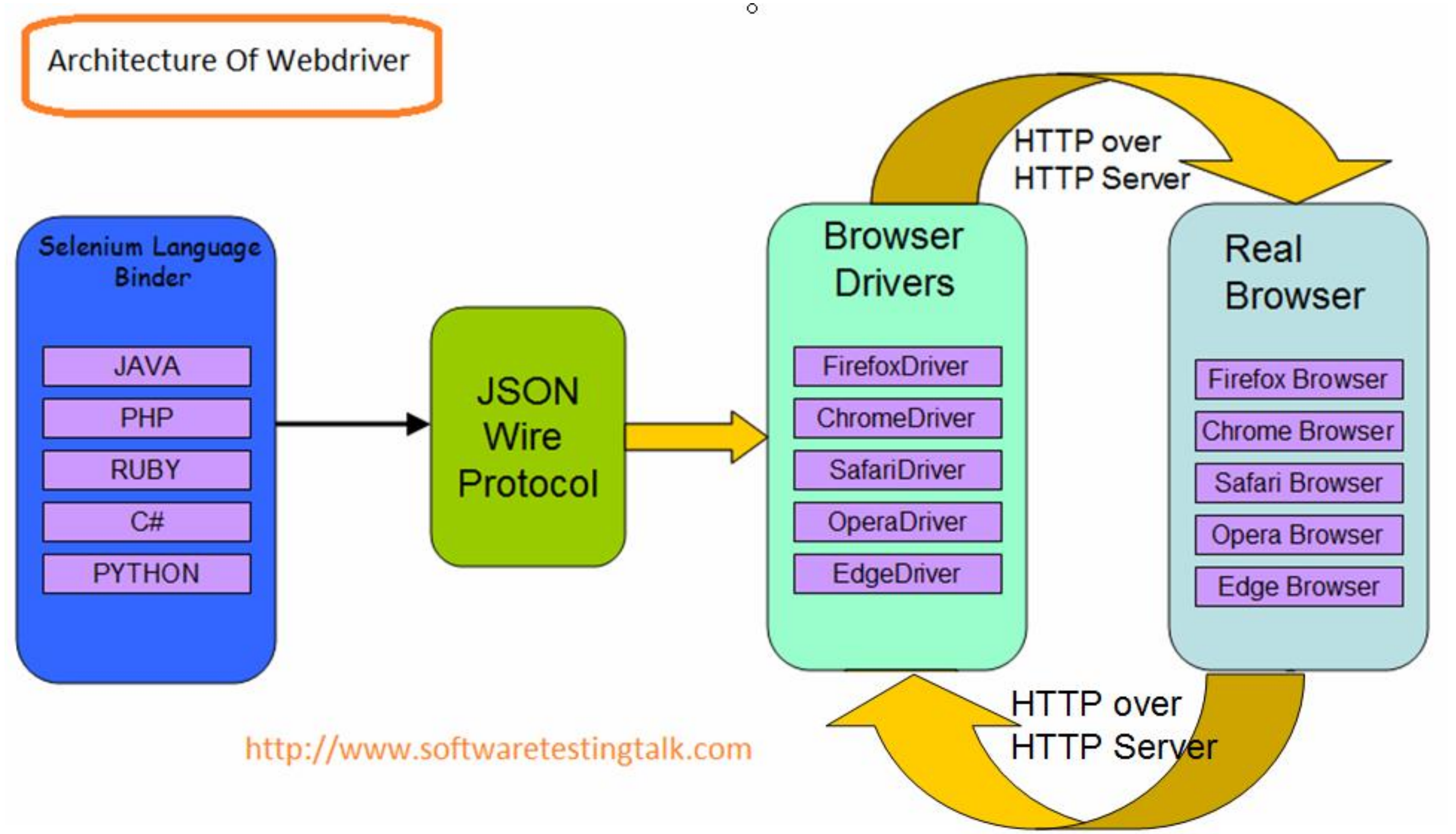
Selenium WebDriver

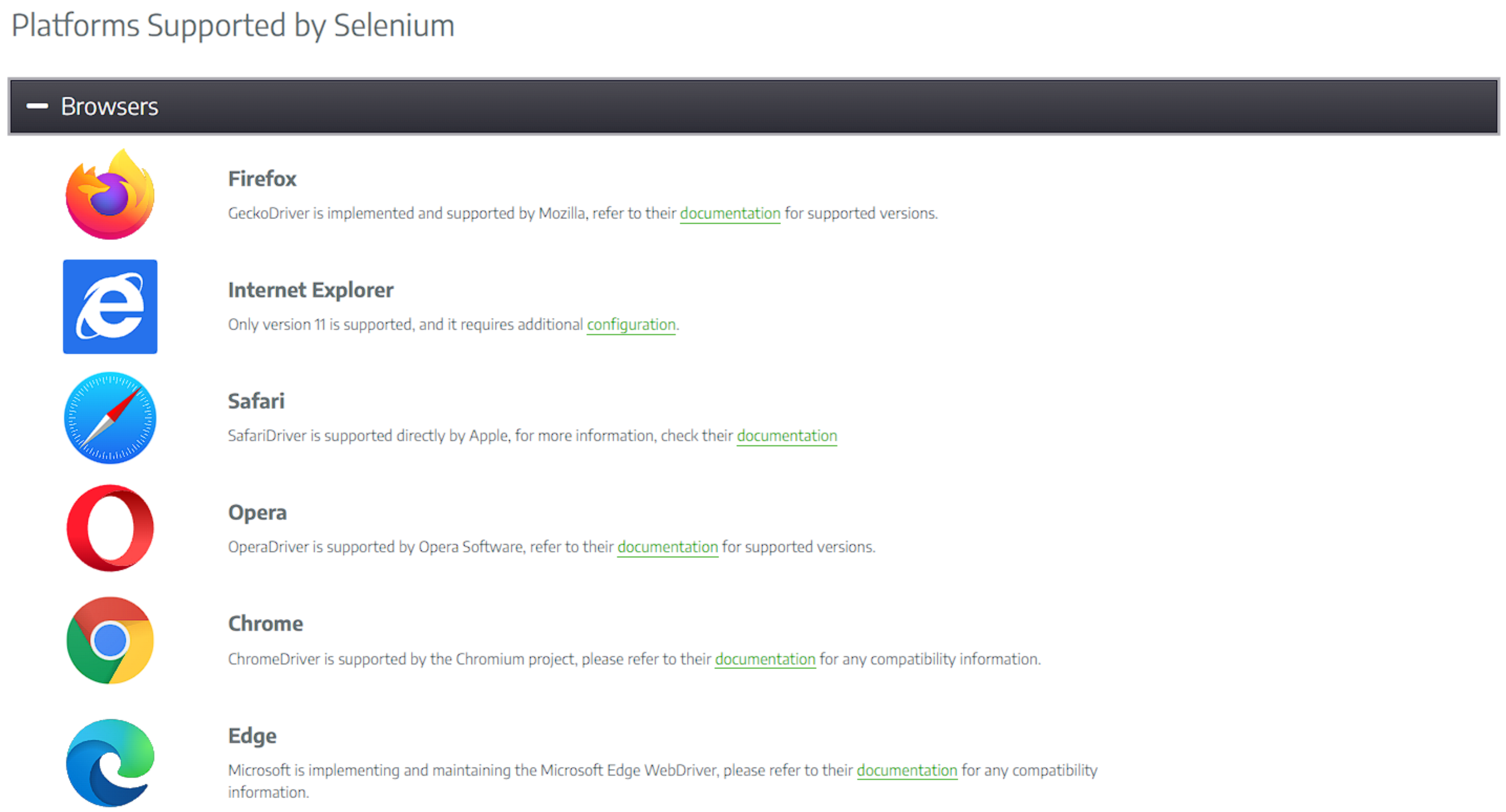
If you want to

- create robust, browser-based regression automation suites and tests
- scale and distribute scripts across many environments

Selenium-WebDriver makes direct calls to the browser using each browser's native support for automation.

Selenium WebDriver





Platforms Supported by Selenium

Browsers

Firefox

GeckoDriver is implemented and supported by Mozilla, refer to their documentation for supported versions.

Internet Explorer

Only version 11 is supported, and it requires additional configuration.

Safari

SafariDriver is supported directly by Apple, for more information, check their documentation

Opera

OperaDriver is supported by Opera Software, refer to their documentation for supported versions.

Chrome

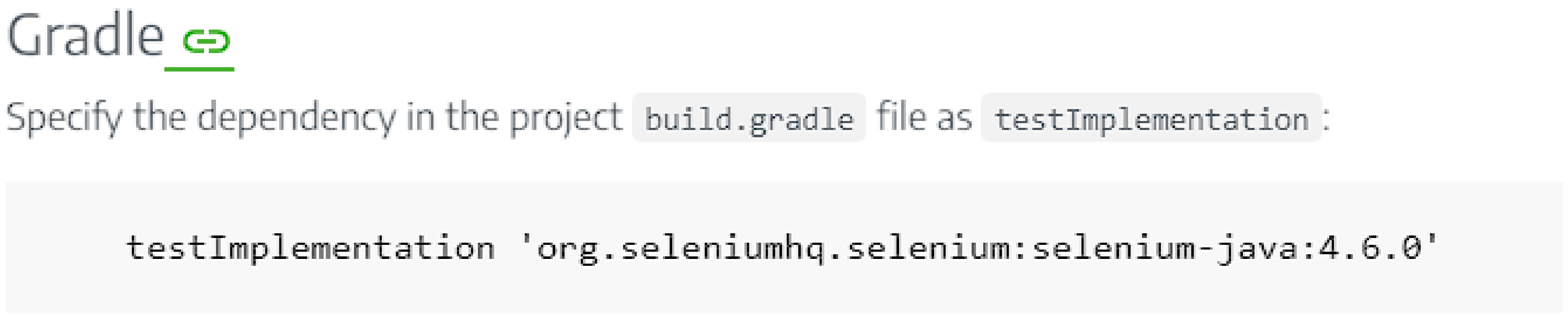
ChromeDriver is supported by the Chromium project, please refer to their documentation for any compatibility information.

Edge

Microsoft is implementing and maintaining the Microsoft Edge WebDriver, please refer to their documentation for any compatibility information.

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

Build.gradle

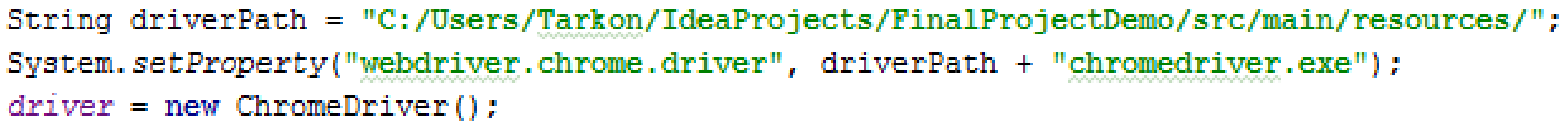


Gradle

Specify the dependency in the project `build.gradle` file as `testImplementation`:

```
testImplementation 'org.seleniumhq.selenium:selenium-java:4.6.0'
```

How to launch webdriver

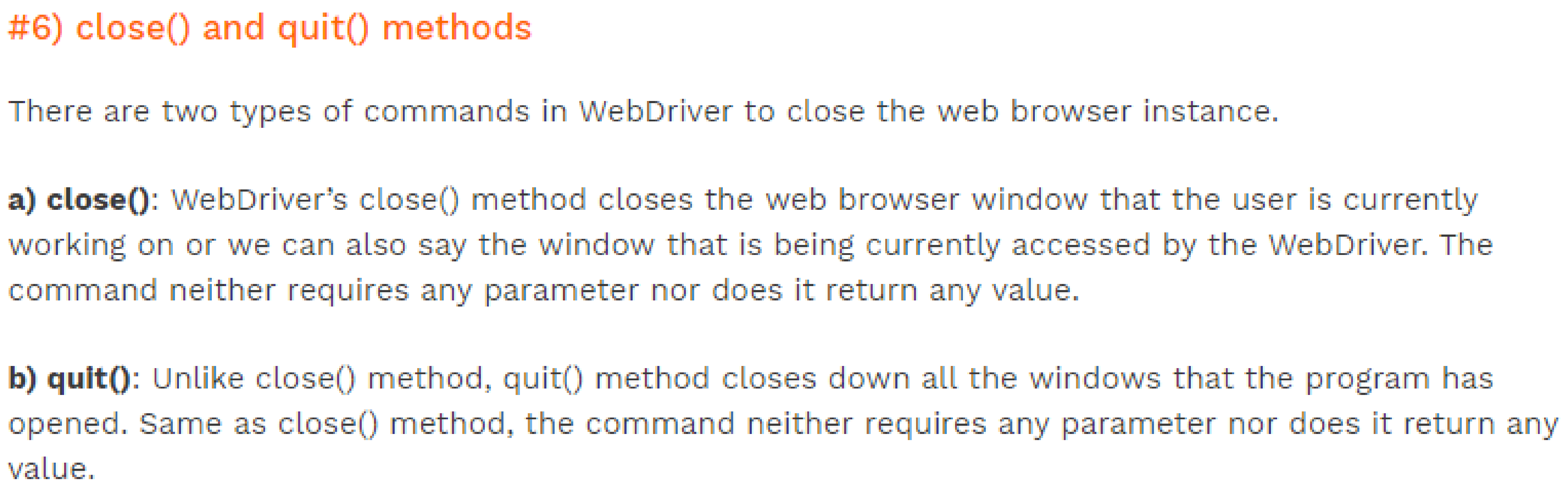


```
String driverPath = "C:/Users/Tarkon/IdeaProjects/FinalProjectDemo/src/main/resources/";
System.setProperty("webdriver.chrome.driver", driverPath + "chromedriver.exe");
driver = new ChromeDriver();
```

Webdriver top commands

- #1) get()
- #2) getCurrentUrl()
- #3) findElement(By, by) and click()
- #4) isEnabled()
- #5) findElement(By, by) with sendKeys()
- #6) findElement(By, by) with getText()
- #7) Submit()
- #8) findElements(By, by)

Never Forget to CLOSE or QUIT driver



#6) close() and quit() methods

There are two types of commands in WebDriver to close the web browser instance.

a) close(): WebDriver's close() method closes the web browser window that the user is currently working on or we can also say the window that is being currently accessed by the WebDriver. The command neither requires any parameter nor does it return any value.

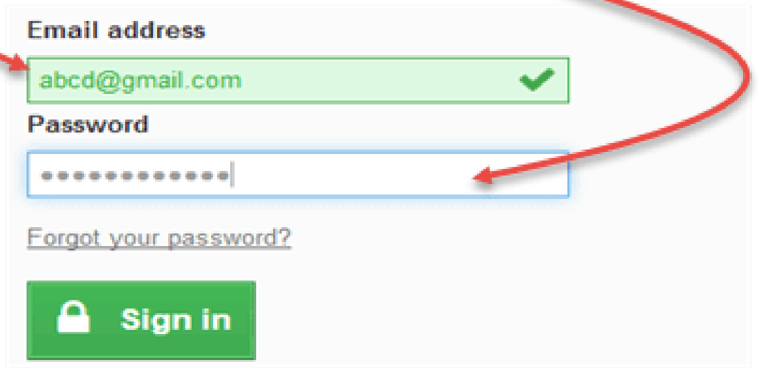
b) quit(): Unlike close() method, quit() method closes down all the windows that the program has opened. Same as close() method, the command neither requires any parameter nor does it return any value.

Simple action that can be done with WebElements

- Click();
- sendKeys();
- Submit();
- SelectBy();

sendKeys();

```
// Get the WebElement corresponding to the Email Address(TextField)
WebElement email = driver.findElement(By.id("email")); 1

// Retrieve the WebElement corresponding to the Password Field
WebElement password = driver.findElement(By.name("passwd")); 2

email.sendKeys("abcd@gmail.com"); 3

password.sendKeys("abcdefghlkjl"); 4
```



Email address
abcd@gmail.com
Password
•••••••••••••

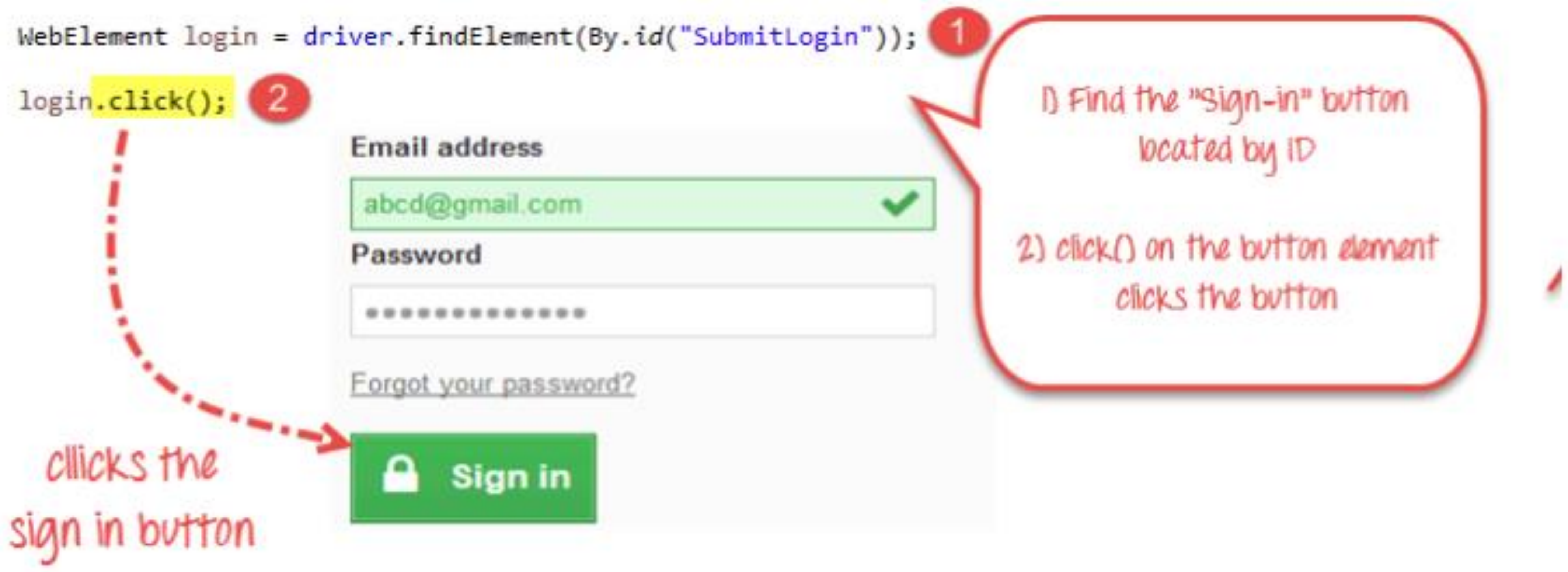
1) Find the "Email Address" Text Field using id locator

2) Find the "Password" Field using name locator

3) Enter text into the "Email Address"

4) Enter password into the "Password" using sendKeys()

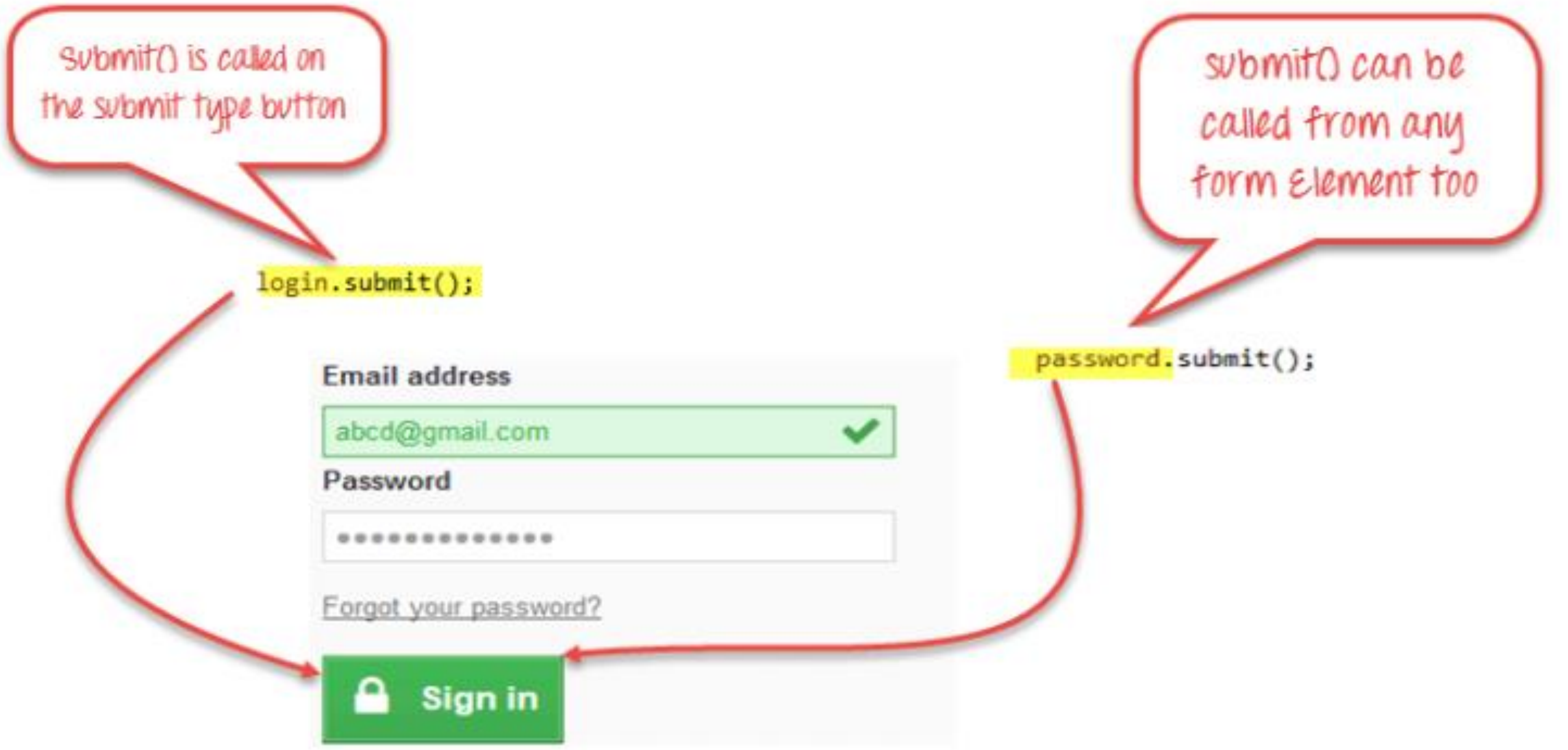
click();



```
WebElement login = driver.findElement(By.id("SubmitLogin"));
login.click();
```

1
2
Email address
abcd@gmail.com
Password
Forgot your password?
Sign in
1) Find the "Sign-in" button located by ID
2) click() on the button element clicks the button
clicks the sign in button

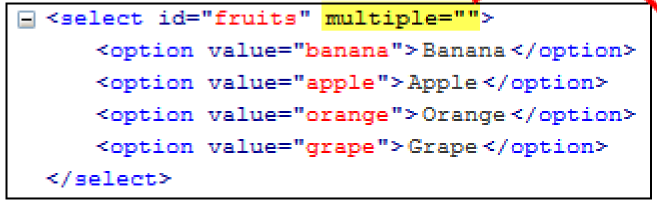
submit();


Submit() is called on the submit type button
submit() can be called from any form Element too
Email address
abcd@gmail.com
Password
Forgot your password?
Sign in

```
login.submit();
password.submit();
```

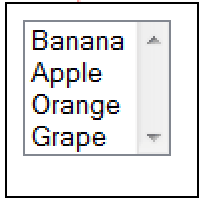
selectByN();

page source



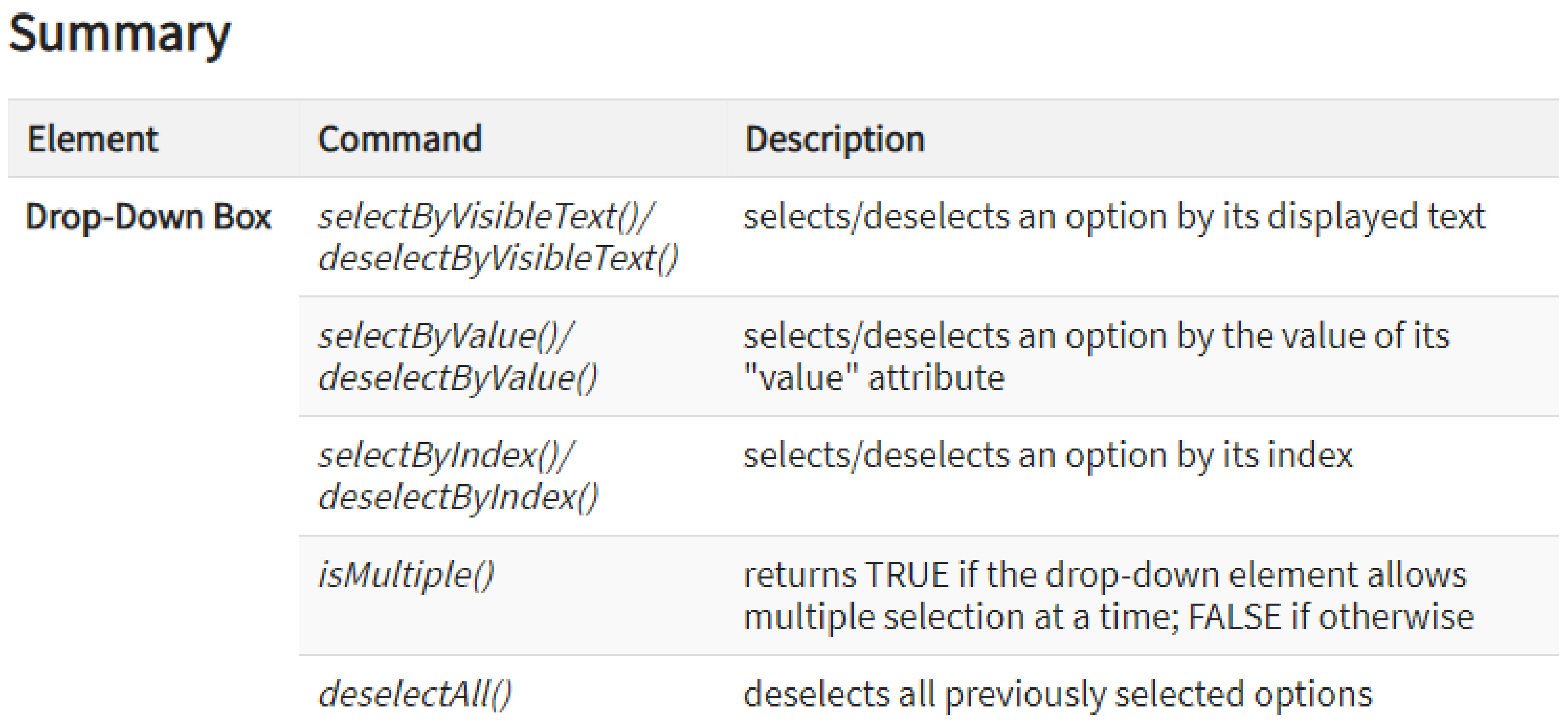
```
<select id="fruits" multiple="">
    <option value="banana">Banana</option>
    <option value="apple">Apple</option>
    <option value="orange">Orange</option>
    <option value="grape">Grape</option>
</select>
```

HTML page



```
//Selecting Items in a Multiple SELECT elements
driver.get("http://jsbin.com/osebed/2");
Select fruits = new Select(driver.findElement(By.id("fruits")));
fruits.selectByVisibleText("Banana");
fruits.selectByIndex(1);
```

selectByN();



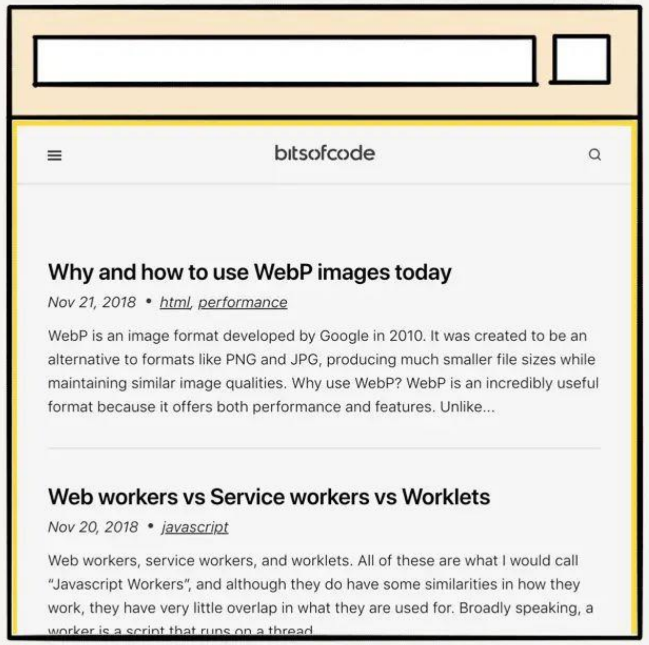
Summary

Element	Command	Description
Drop-Down Box	*selectByVisibleText()/ deselectByVisibleText()*	selects/deselects an option by its displayed text
	selectByValue()/ deselectByValue()	selects/deselects an option by the value of its "value" attribute
	selectByIndex()/ deselectByIndex()	selects/deselects an option by its index
	isMultiple()	returns TRUE if the drop-down element allows multiple selection at a time; FALSE if otherwise
	deselectAll()	deselects all previously selected options

Headless browser

- Headless testing is a way of running browser UI tests without the *head,* which in this case means that there’s no browser UI, no GUI of any sorts. This is useful since when running tests, especially in a CI environment, there is nobody “watching” the visuals, so there is no need to have the extra overhead of the browser GUI.

Headless browser



bitsofcode

Why and how to use WebP images today

Nov 21, 2018 • *html, performance*

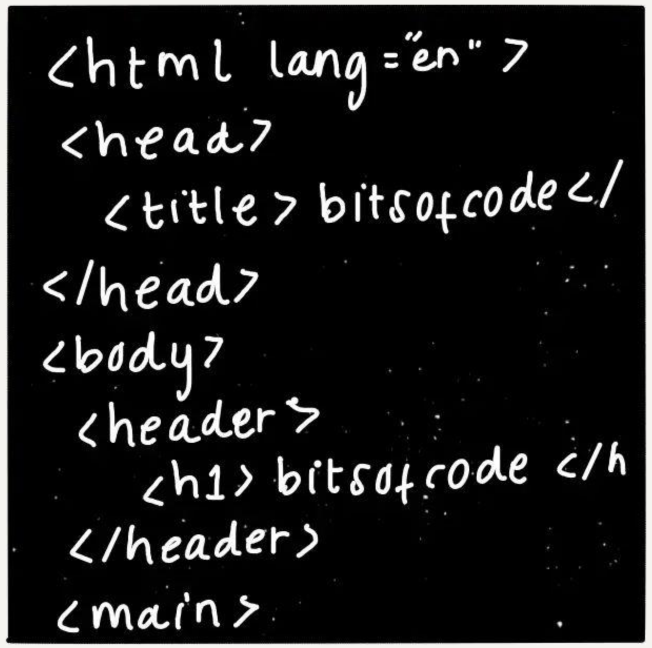
WebP is an image format developed by Google in 2010. It was created to be an alternative to formats like PNG and JPG, producing much smaller file sizes while maintaining similar image qualities. Why use WebP? WebP is an incredibly useful format because it offers both performance and features. Unlike...

Web workers vs Service workers vs Worklets

Nov 20, 2018 • *javascript*

Web workers, service workers, and worklets. All of these are what I would call "Javascript Workers", and although they do have some similarities in how they work, they have very little overlap in what they are used for. Broadly speaking, a worker is a script that runs on a thread.

headful



```
<html lang="en">
 <head>
   <title> bitsofcode</
</head>
<body>
  <header>
     <h1> bitsofcode </h
  </header>
 <main>
```

head*less*

Headless browser advantages

- Able to run far more instances simultaneously than non-headless drivers.
- Can make use of large amounts of factory-generated or manually created test variables in Data-Driven Testing
- Run-time can be reduced by up to 50% for most tasks.
- Can be executed without taking up the screen context of a computer
- Can be used for scrapping
- Less chances of failure due to ‘human intervention.’
- Any desired screenshots are still stored just like in regular automation testing.

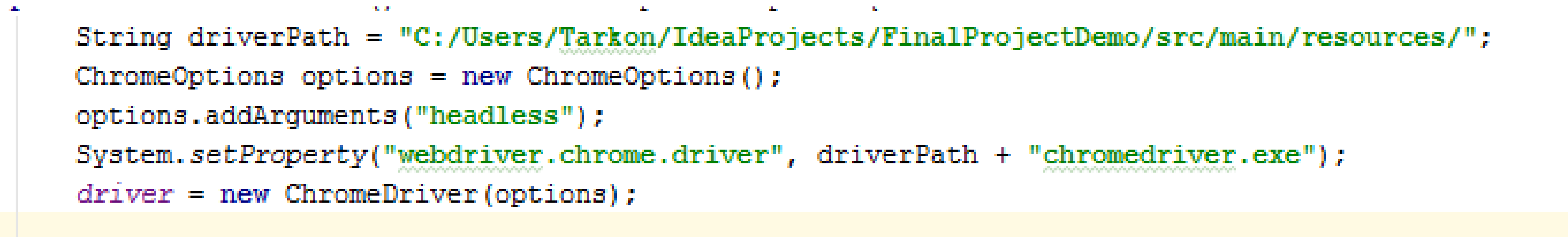
Headless browser disadvantages

- Hard to debug inconsistent failures on locating elements due to page loading too fast
- Unintended interactions (losing the benefit of automated UI testing vs integration or unit testing) due to speed/headless state of browser
- When you run headless tests, you are not really mimicking the real user experience.
- Cosmetic bugs can’t be identified while doing headless browser testing, such as the location of a button, or the color of a web element, etc.

Headless browser list

- Firefox -> geckodriver
- Chrome
- PhantomJS
- ZombieJS
- HtmlUnit
- Pupeteer

Using headless browser

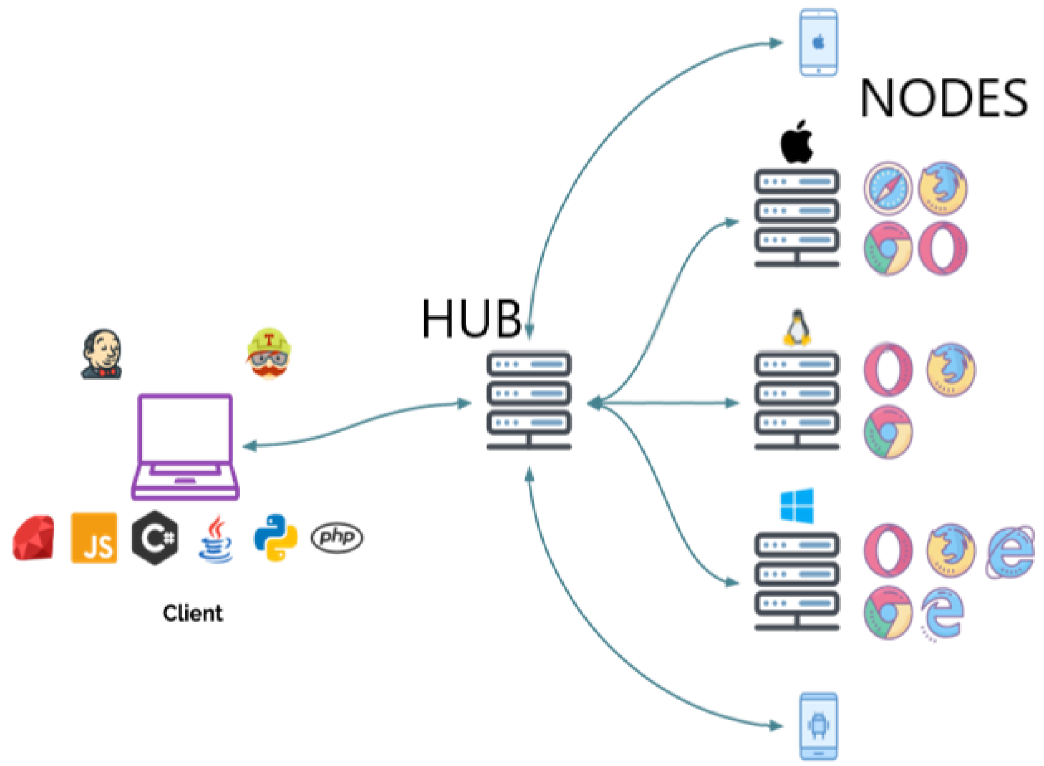


```
String driverPath = "C:/Users/Tarkon/IdeaProjects/FinalProjectDemo/src/main/resources/";
ChromeOptions options = new ChromeOptions();
options.addArguments("headless");
System.setProperty("webdriver.chrome.driver", driverPath + "chromedriver.exe");
driver = new ChromeDriver(options);
```

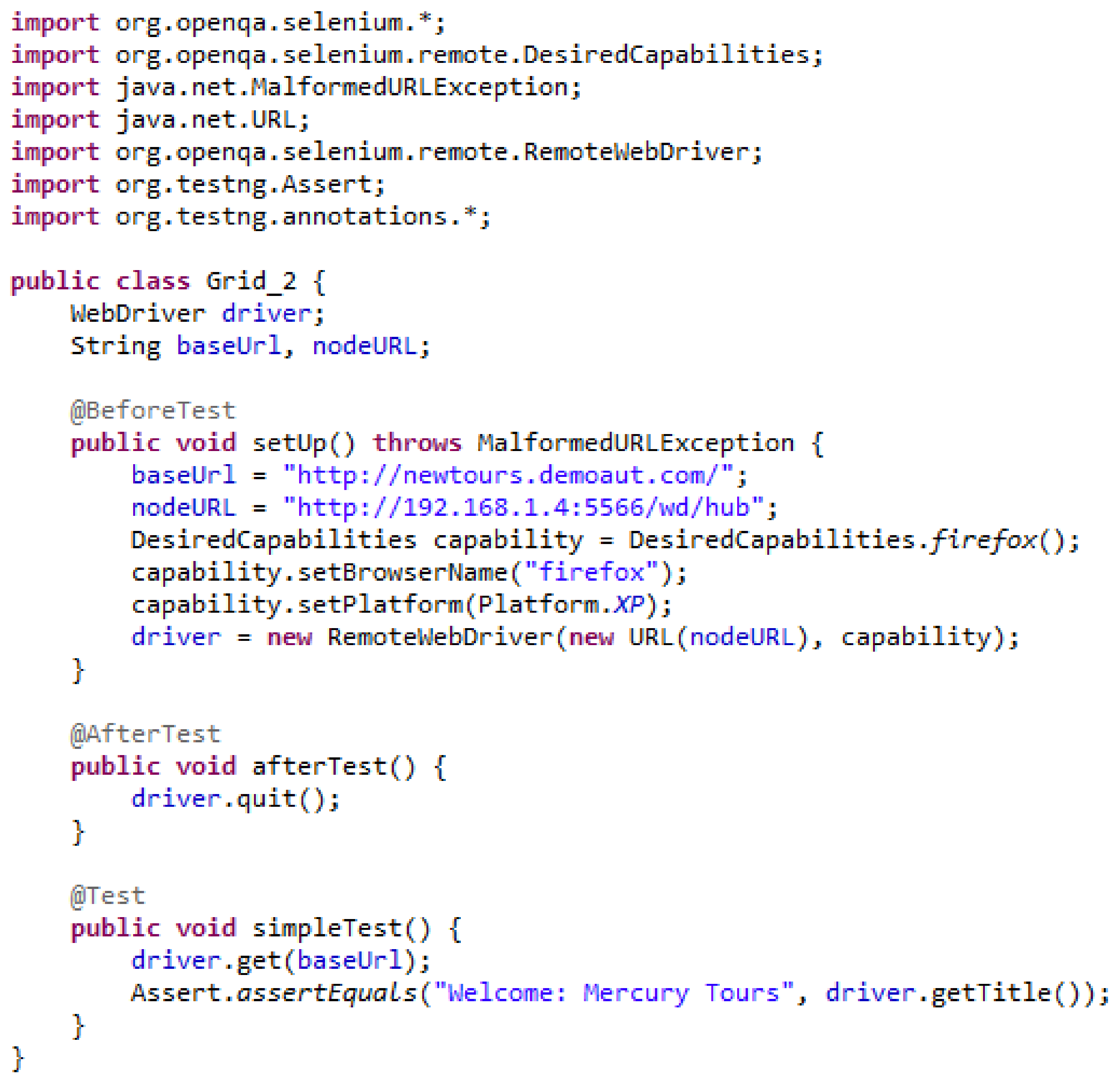
Selenium Grid

- Selenium Grid is a server that allows tests to use web browser instances running on remote machines. With Selenium Grid, one server acts as the hub.
- A grid consists of a single **hub**, and one or more **nodes**.

https://www.seleniumhq.org/docs/07_selenium_grid.jsp

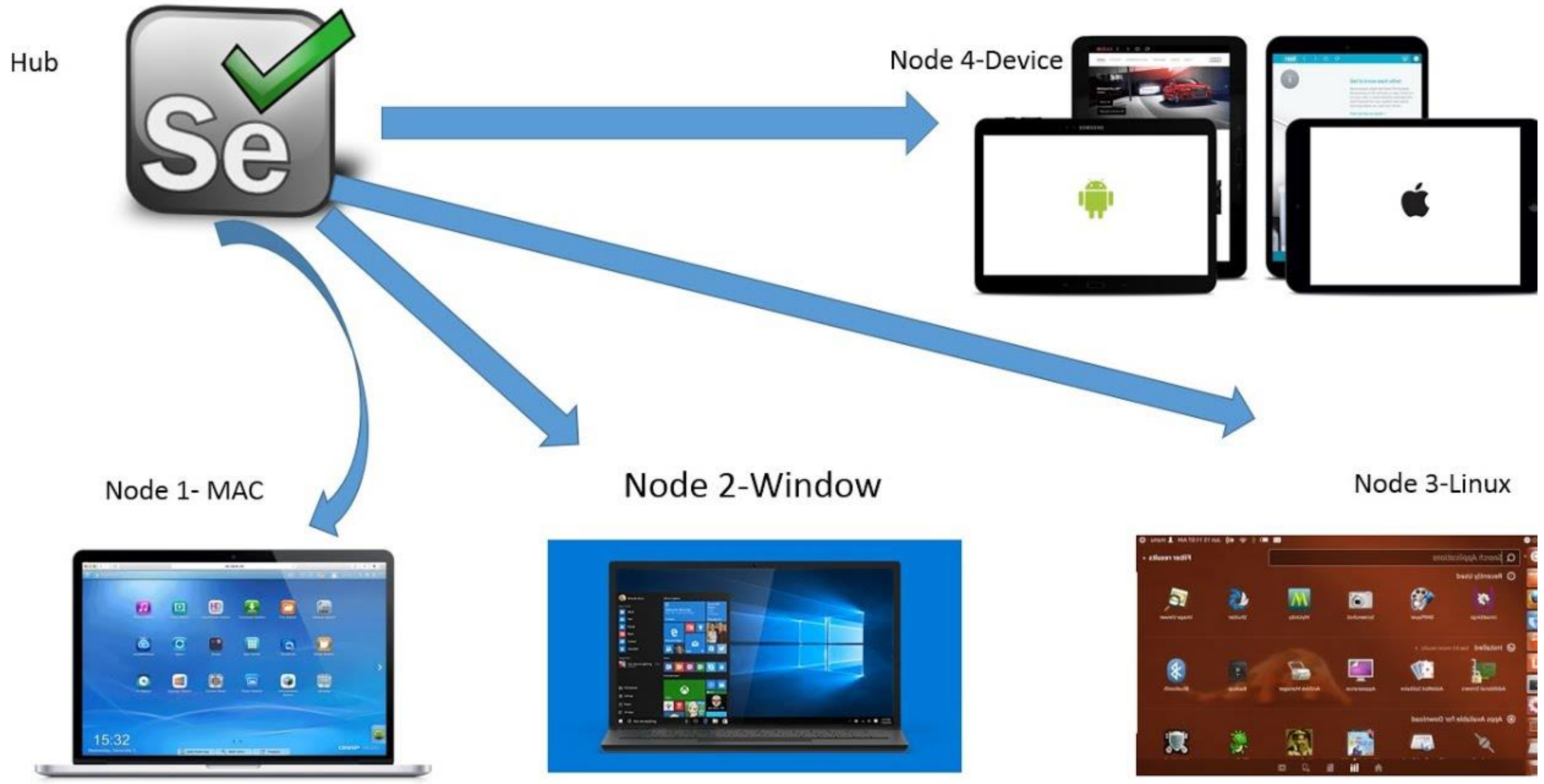


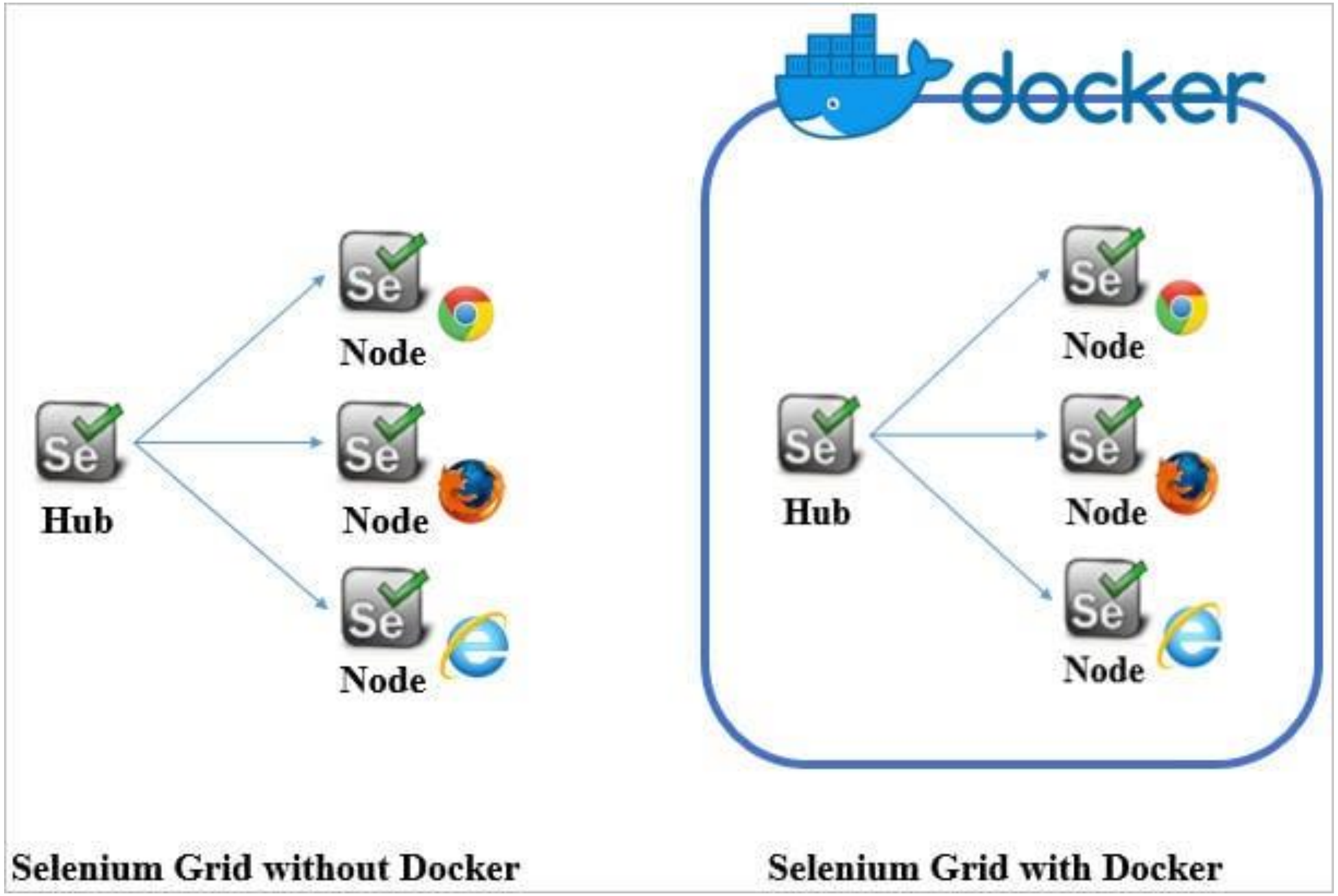
Selenium Grid



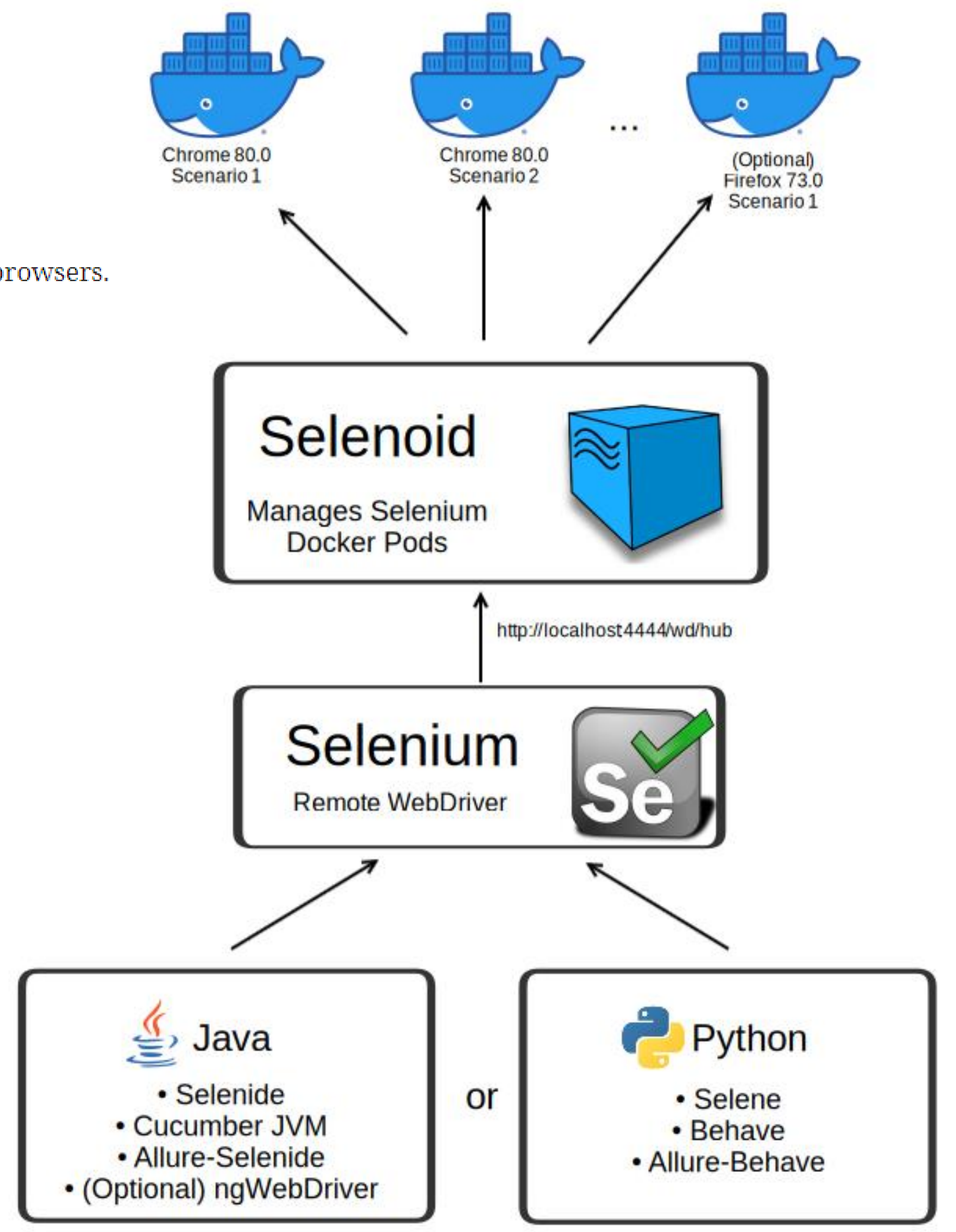
```java
import org.openqa.selenium.*;
import org.openqa.selenium.remote.DesiredCapabilities;
import java.net.MalformedURLException;
import java.net.URL;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.Assert;
import org.testng.annotations.*;

public class Grid_2 {
    WebDriver driver;
    String baseUrl, nodeURL;

    @BeforeTest
    public void setUp() throws MalformedURLException {
        baseUrl = "http://newtours.demoaut.com/";
        nodeURL = "http://192.168.1.4:5566/wd/hub";
        DesiredCapabilities capability = DesiredCapabilities.firefox();
        capability.setBrowserName("firefox");
        capability.setPlatform(Platform.XP);
        driver = new RemoteWebDriver(new URL(nodeURL), capability);
    }

    @AfterTest
    public void afterTest() {
        driver.quit();
    }

    @Test
    public void simpleTest() {
        driver.get(baseUrl);
        Assert.assertEquals("Welcome: Mercury Tours", driver.getTitle());
    }
}
```

Selenium Grid




docker
Node
Node
Node
Hub
Node
Node
Node
Hub
Selenium Grid without Docker
Selenium Grid with Docker

Selenoid is a powerful Golang implementation of original Selenium hub code. It is using Docker to launch browsers. Please refer to GitHub repository if you need source code.

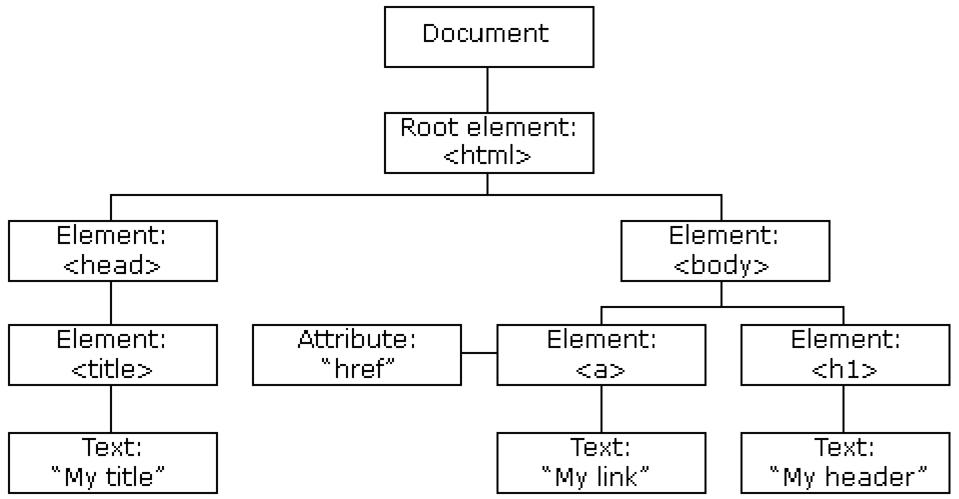


Agenda

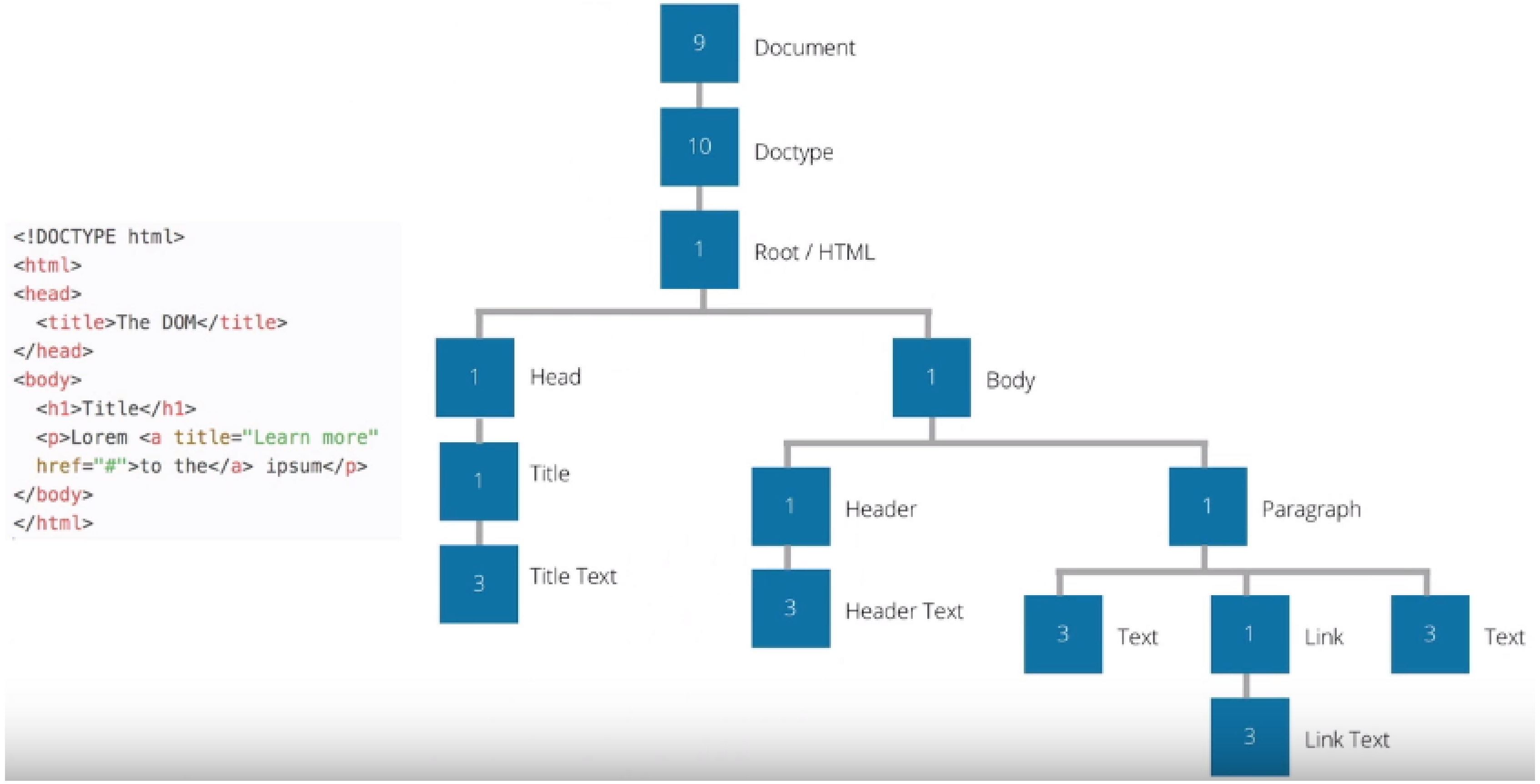
- Selenium Web driver
- **Locators**
- Xpath
- Css selector
- Iframe
- Practice

The HTML DOM (Document Object Model)

- When a web page is loaded, the browser creates a Document Object Model of the page
- The HTML DOM model is constructed as a tree of Objects
- In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements
- With the document object model, JavaScript gets all the power it needs to create dynamic HTML
- Document Object Model or DOM is an essential component of web development and **automation**



HTML DOM example



```
<!DOCTYPE html>
<html>
<head>
  <title>The DOM</title>
</head>
<body>
  <h1>Title</h1>
  <p>Lorem <a title="Learn more"
  href="#">to the</a> ipsum</p>
</body>
</html>
```

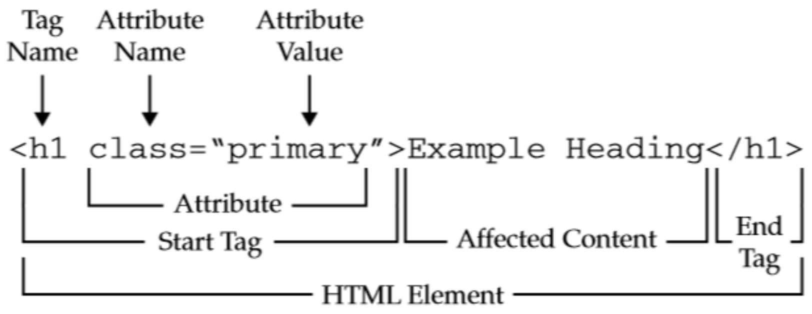
HTML (Hyper Text Markup Language)

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page
- All HTML documents must start with a document type declaration: <!DOCTYPE html>
- Begins with <html> and ends with </html>
- The visible part of the HTML document is between <body> and </body>

HTML syntax overview

A graphical overview of the HTML markup syntax shown so far is presented here:

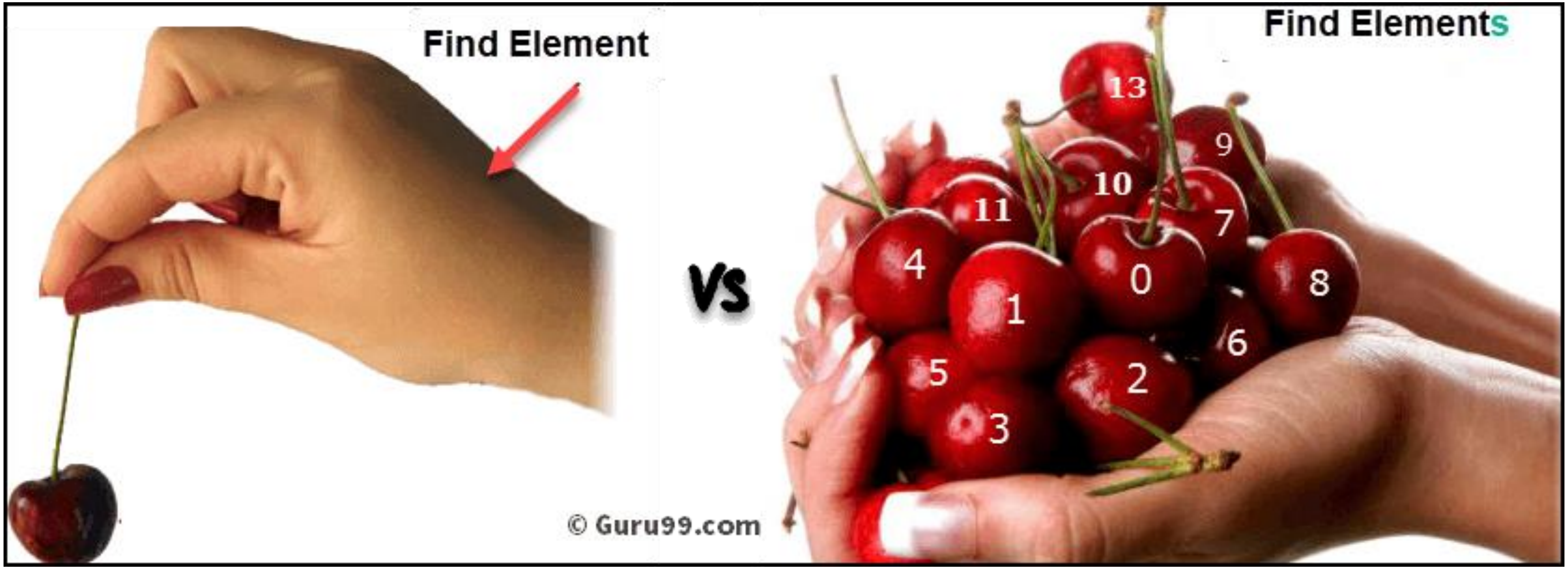


Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h *n*> ... </h*n*>	Delimits a level *n* heading
<b> ... </b>	Set ... in boldface
<i> ... </i>	Set ... in italics
<center> ... </center>	Center ... on the page horizontally
<ul> ... </ul>	Brackets an unordered (bulleted) list
<ol> ... </ol>	Brackets a numbered list
<li> ... </li>	Brackets an item in an ordered or numbered list
<br>	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a horizontal rule
<img src="...">	Displays an image here
<a href="..."> ... </a>	Defines a hyperlink

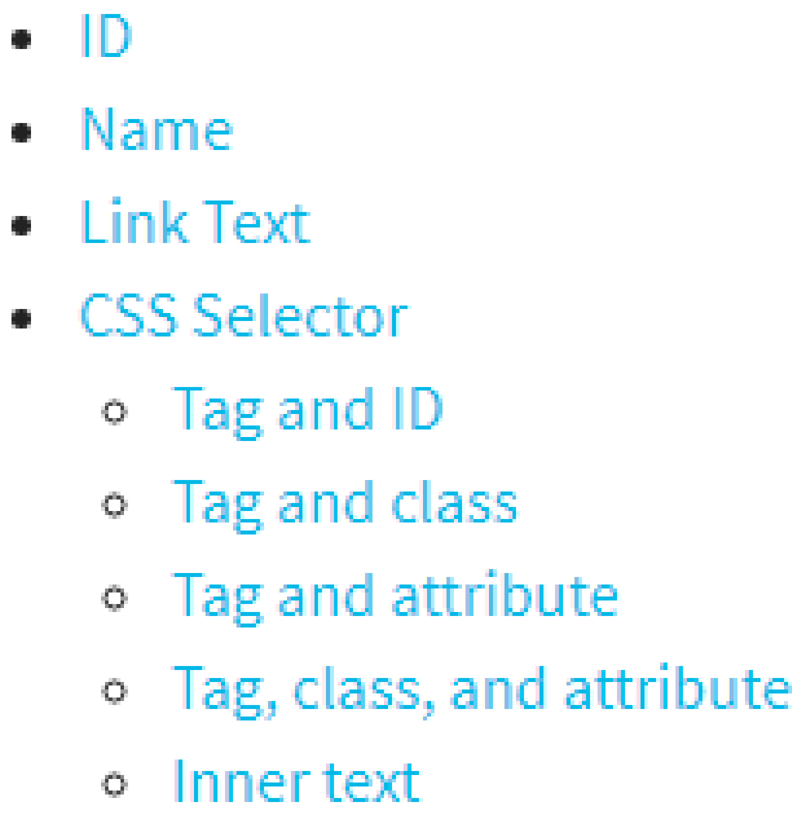
Locators – what it is?

- Locator is a command that tells Selenium which GUI elements (say Text Box, Buttons, Check Boxes etc) its needs to operate on. Identification of correct GUI elements is a prerequisite to creating an automation script. But accurate identification of GUI elements is more difficult than it sounds. Sometimes, you end up working with incorrect GUI elements or no elements at all! Hence, Selenium provides a number of Locators to precisely locate a GUI element

findElement vs findElements

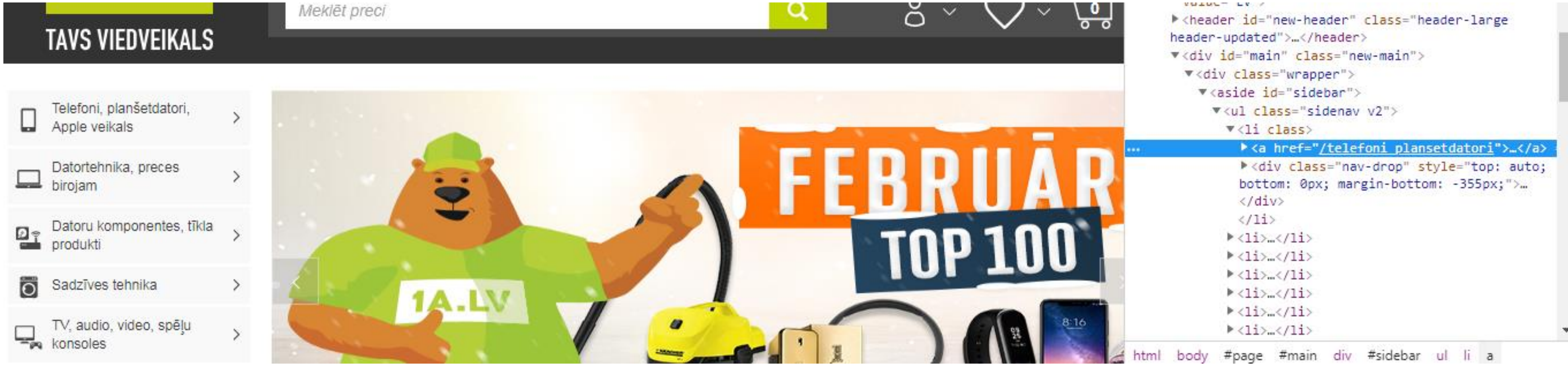


Locators



- ID
- Name
- Link Text
- CSS Selector
    - Tag and ID
    - Tag and class
    - Tag and attribute
    - Tag, class, and attribute
    - Inner text

How to look?


TAVS VIEDVEIKALS
Meklēt preci
Telefoni, planšetdatori, Apple veikals
Datortehnika, preces birojam
Datoru komponentes, tīkla produkti
Sadzīves tehnika
TV, audio, video, spēļu konsoles
FEBRUĀR
TOP 100
1A.LV

```
▶<header id="new-header" class="header-large
header-updated">…</header>
▼<div id="main" class="new-main">
  ▼<div class="wrapper">
    ▼<aside id="sidebar">
      ▼<ul class="sidenav v2">
        ▼<li class>
          ▶<a href="/telefoni_plansetdatori">…</a>
          ▶<div class="nav-drop" style="top: auto;
          bottom: 0px; margin-bottom: -355px;">…
          </div>
          </li>
        ▶<li>…</li>
        ▶<li>…</li>
        ▶<li>…</li>
        ▶<li>…</li>
        ▶<li>…</li>
        ▶<li>…</li>
```

html body #page #main div #sidebar ul li a

Agenda

- Selenium Web driver
- Locators
- **Xpath**
- Css selector
- Iframe
- Practice

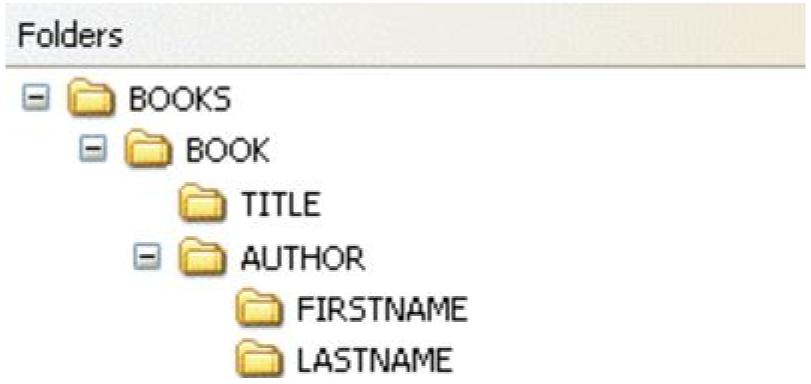
What is Xpath

- XPath stands for XML Path Language
- XPath uses "path like" syntax to identify and navigate nodes in an XML document
- XPath contains over 200 built-in functions
- XPath is a major element in the XSLT standard
- XPath is a W3C recommendation

XPath Path Expressions

XPath uses path expressions to select nodes or node-sets in an XML document.

These path expressions look very much like the path expressions you use with traditional computer file systems:



How XPath Works

The XPath specification is the foundation for a variety of specifications, including XSLT and linking/addressing specifications such as `XPointer`. So an understanding of XPath is fundamental to a lot of advanced XML usage. This section provides an introduction to XPath in the context of XSLT.

XPath Expressions

In general, an XPath expression specifies a pattern that selects a set of XML nodes. XSLT templates then use those patterns when applying transformations. (`XPointer`, on the other hand, adds mechanisms for defining a **point** or a **range** so that XPath expressions can be used for addressing).

The nodes in an XPath expression refer to more than just elements. They also refer to text and attributes, among other things. In fact, the XPath specification defines an abstract document model that defines seven kinds of nodes:

- Root
- Element
- Text
- Attribute
- Comment
- Processing instruction
- Namespace

The root element of the XML data is modeled by an **element** node. The XPath root node contains the document's root element as well as other information relating to the document.

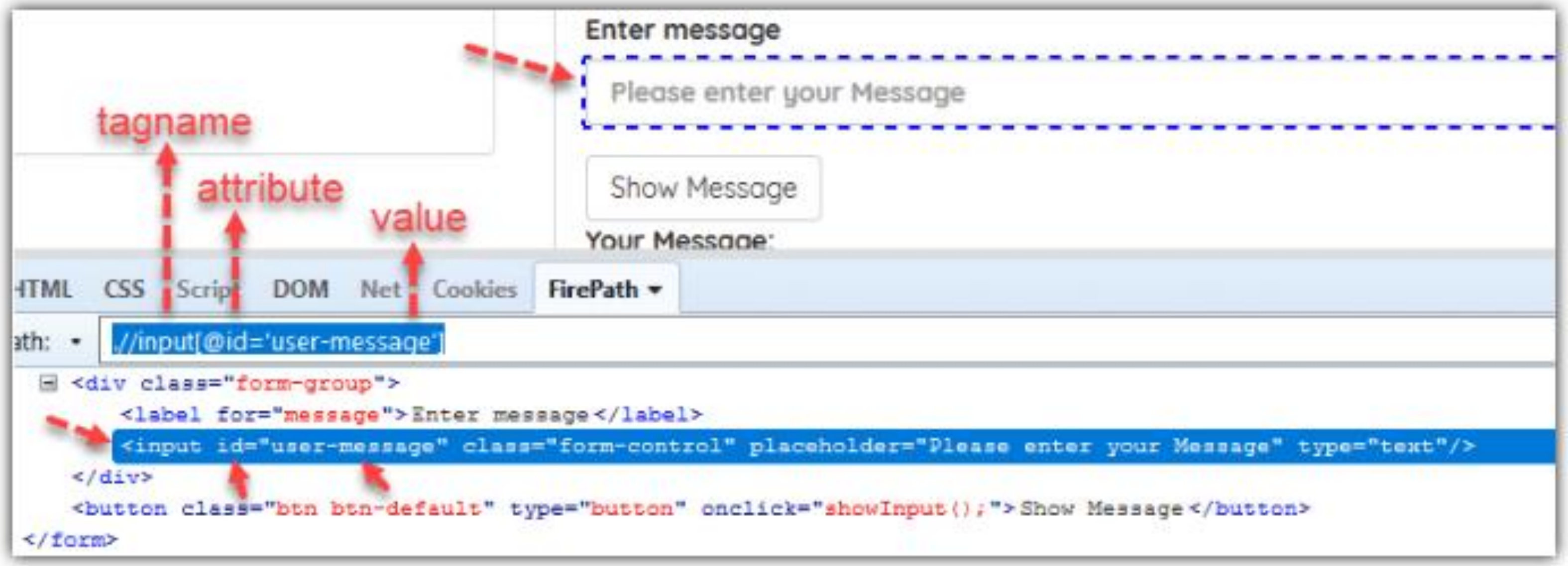
Xpath what is is?

XPath Selenium Selectors

We can find the location of any element on a web page using XML path expressions. The basic syntax for XPath is shown below:

Syntax = //tagname[@attribute='Value']

Example = //input[@id='user-message']



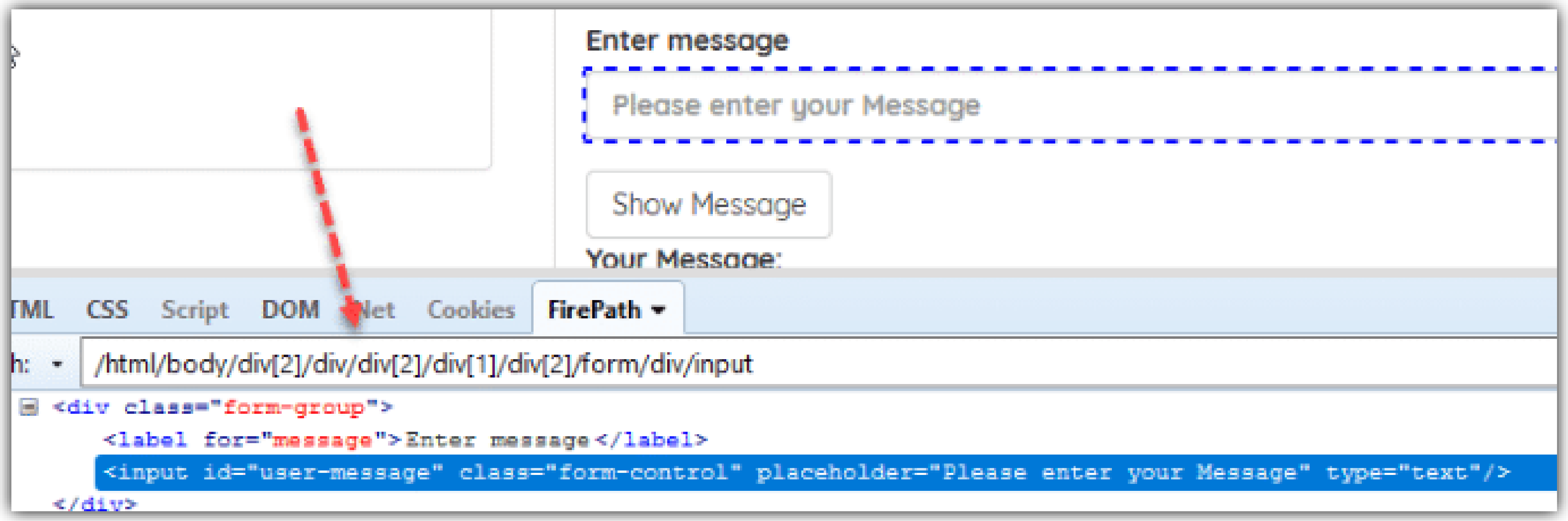
```
.//input[@id='user-message']
<div class="form-group">
    <label for="message">Enter message</label>
    <input id="user-message" class="form-control" placeholder="Please enter your Message" type="text"/>
  </div>
  <button class="btn btn-default" type="button" onclick="showInput();">Show Message</button>
</form>
```

DO NOT USE

Absolute XPath

- It is a direct way to locate an element.
- It is very brittle.
- Starts with single slash "/" that means starting to search from the root.

Example: */html/body/div[2]/div/div[2]/div[1]/div[2]/form/div/input*

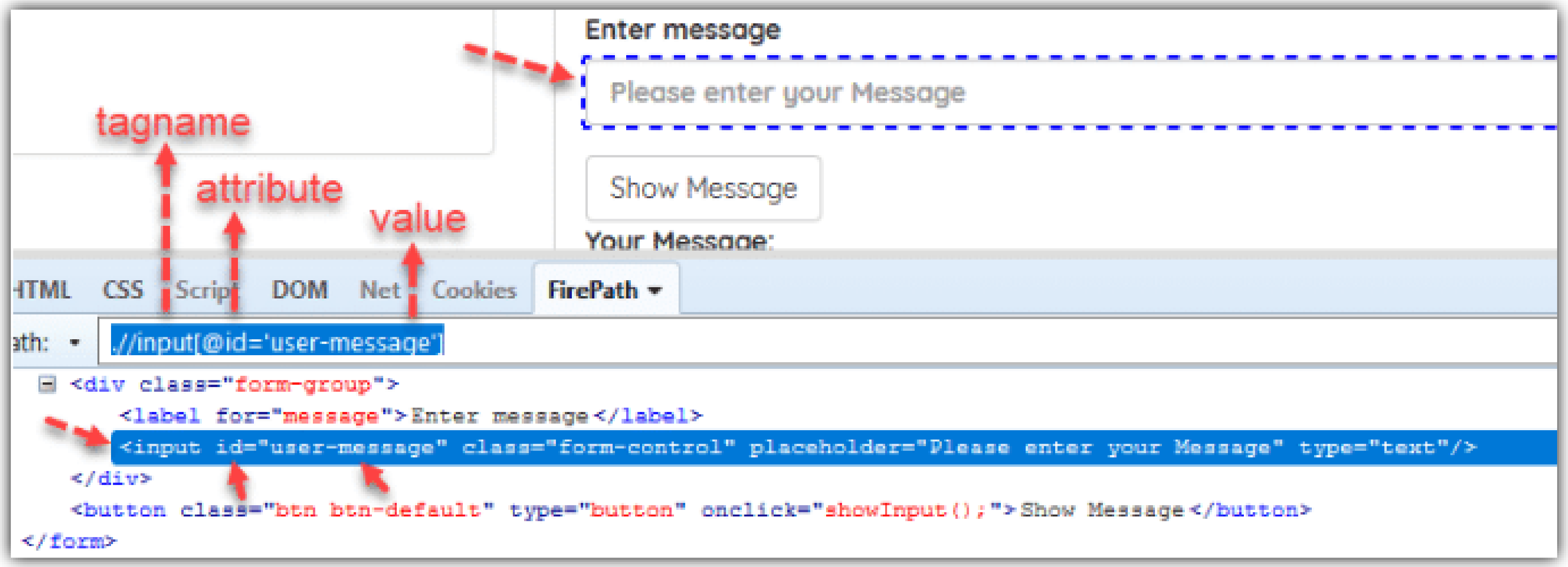


```
/html/body/div[2]/div/div[2]/div[1]/div[2]/form/div/input
<div class="form-group">
    <label for="message">Enter message</label>
    <input id="user-message" class="form-control" placeholder="Please enter your Message" type="text"/>
</div>
```

Common Xpath locators

Syntax = //tagname[@attribute='Value']

Example = //input[@id='user-message']



```
.//input[@id='user-message']
<div class="form-group">
    <label for="message">Enter message</label>
    <input id="user-message" class="form-control" placeholder="Please enter your Message" type="text"/>
</div>
<button class="btn btn-default" type="button" onclick="showInput();">Show Message</button>
</form>
```

Agenda

- Selenium Web driver
- Locators
- Xpath
- **Css selector**
- Iframe
- Practice

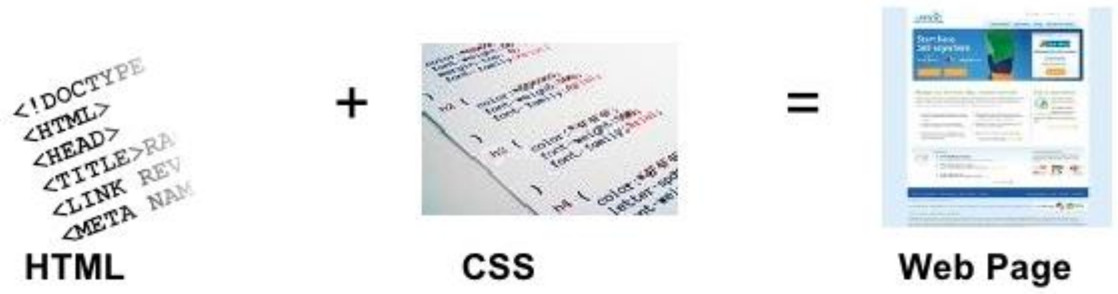
CSS (Cascading Style Sheets)

- body {
- background-color: lightblue;
- }

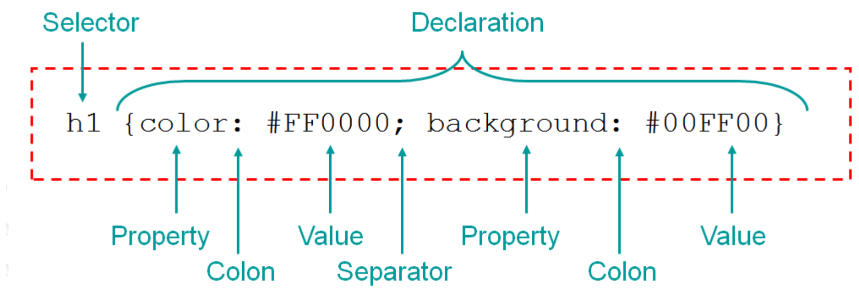
- h1 {
- color: white;
- text-align: center;
- }

- p {
- font-family: verdana;
- font-size: 20px;
- }

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS describes **how HTML elements are to be displayed**
- Contains the rules for the presentation of HTML.
- CSS was introduced to keep the presentation information separate from HTML markup (content).
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files**



CSS syntax overview



The element Selector

```
p {
  text-align: center;
  color: red;
}
```

The class Selector

```
.className {
  text-align: center;
  color: red;
}
```

The id Selector

```
#myId {
  text-align: center;
  color: red;
}
```

Grouping Selectors

```
h1, h2 {
  text-align: center;
  color: red;
}
```

Property	Description	Example
color	Sets the foreground color of an element	body{color: #FCC9814;}
background-color	Sets the background color of an element	body{background-color: green;}
background-image	Inserts a background image	body{background-image: url (“earth.gif”);}
background-repeat	Specifies how background image will repeat itself. It may repeat horizontally (repeat-x) vertically (repeat-y) or (repeat) or may not be repeated (no-repeat)	body{background-repeat: repeat-x;}

Cssselector what is is?

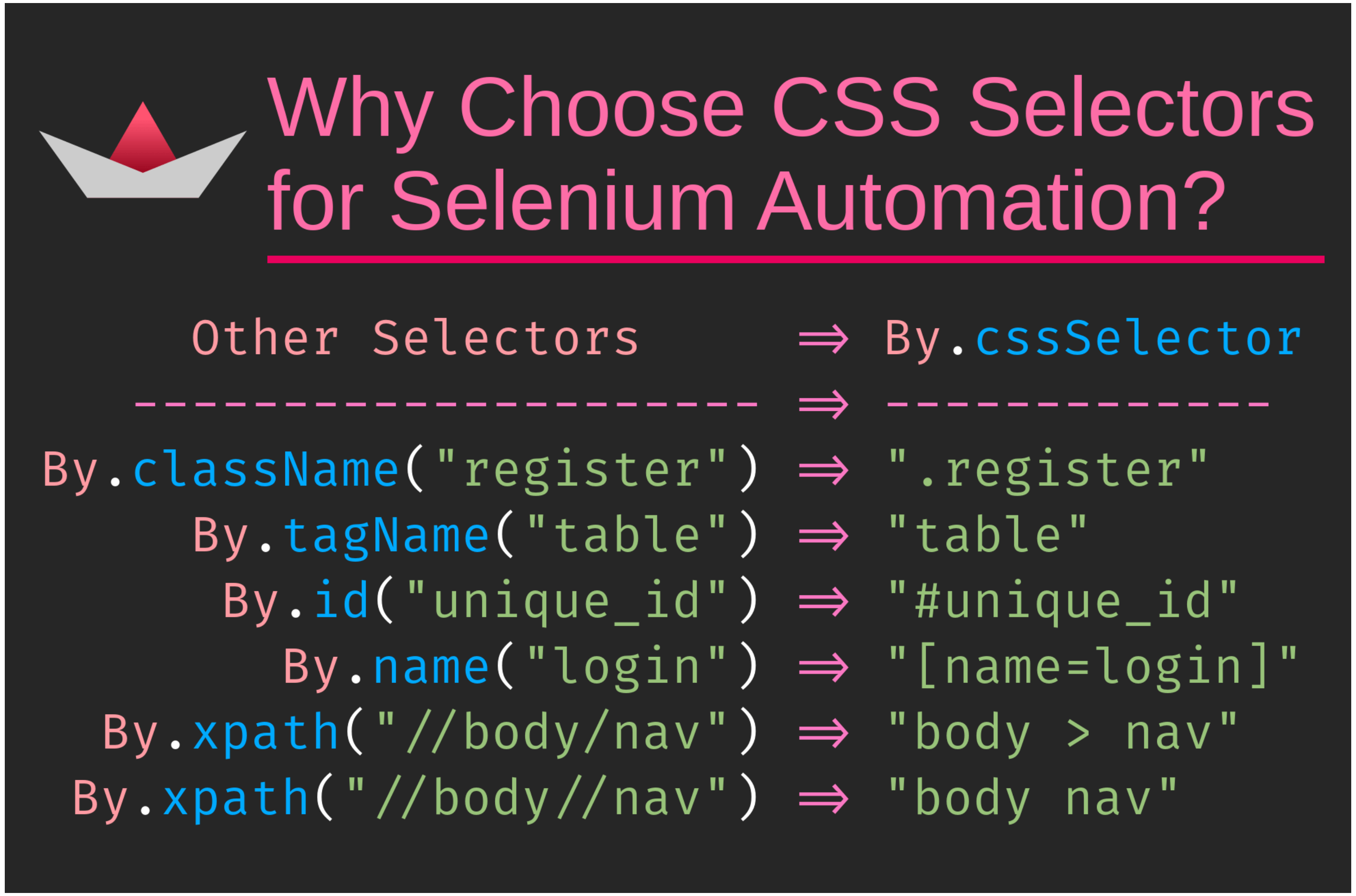
- As we all know, CSS stands for **Cascading Style Sheets**. By using CSS selectors, we can find or select HTML elements on the basis of their id, class or other attributes.

Why to use it?

Using CSS selectors to locate elements has some benefits:

- It’s faster
- More readable
- And used more often

Why to use it?



Why Choose CSS Selectors for Selenium Automation?

```
      Other Selectors         ⇒ By.cssSelector
    ------------------------  ⇒ -------------
By.className("register")  ⇒ ".register"
     By.tagName("table")  ⇒ "table"
      By.id("unique_id")  ⇒ "#unique_id"
        By.name("login")  ⇒ "[name=login]"
   By.xpath("//body/nav")  ⇒ "body > nav"
  By.xpath("//body//nav")  ⇒ "body nav"
```

How to use it?

The generic way of locating elements by their attribute in CSS Selectors is

```
css = element_name[<attribute_name>='<value>']
```

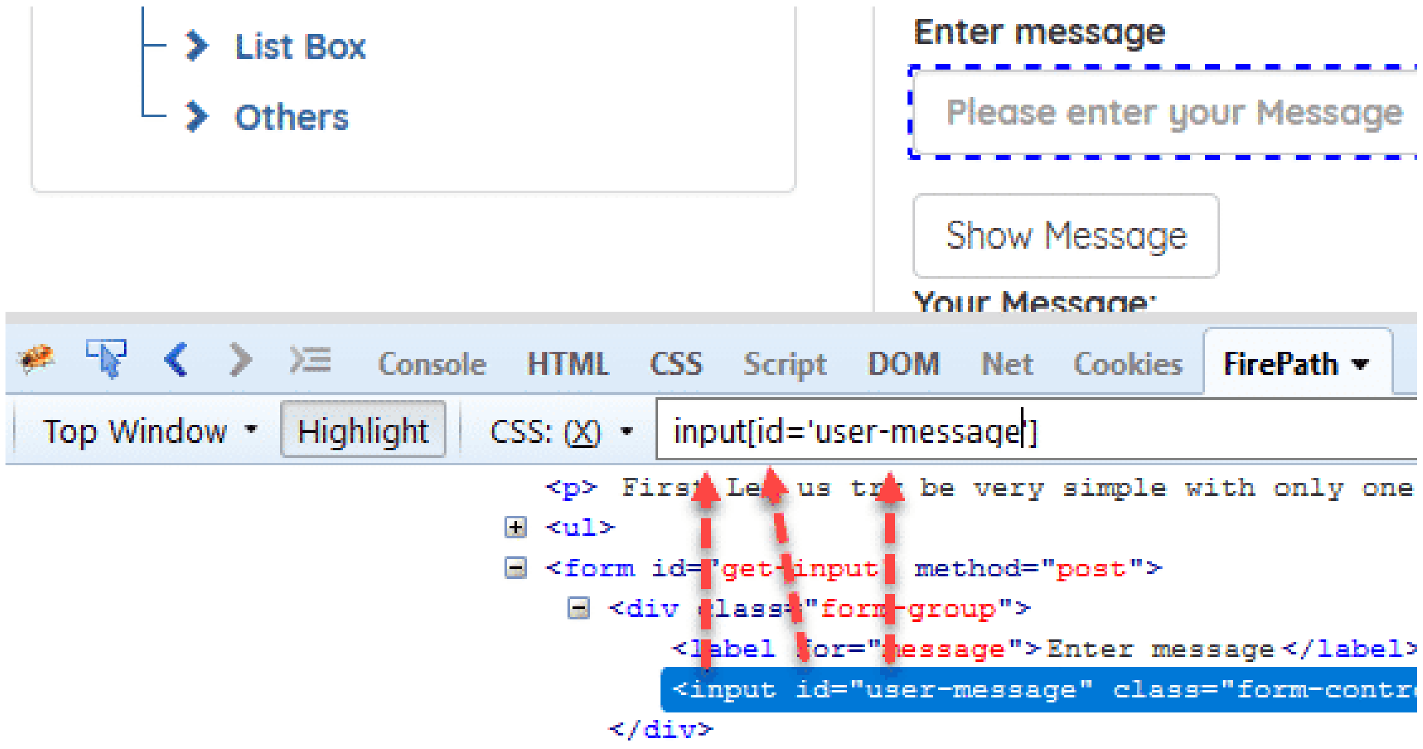
e.g. `css=input[name='first_name']`

CSS selectors

You can use **tag[attribute='value']** syntax.
*(Note: For XPath we use **tag[@attribute='value']** for this.)*

Example

Syntax: input[id='user-message']

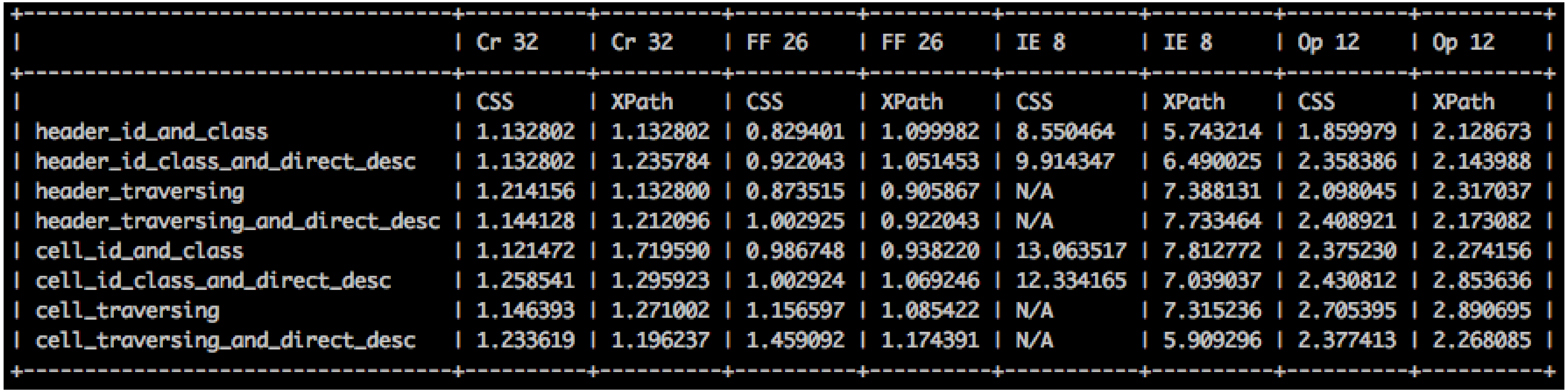


```
input[id='user-message']
```

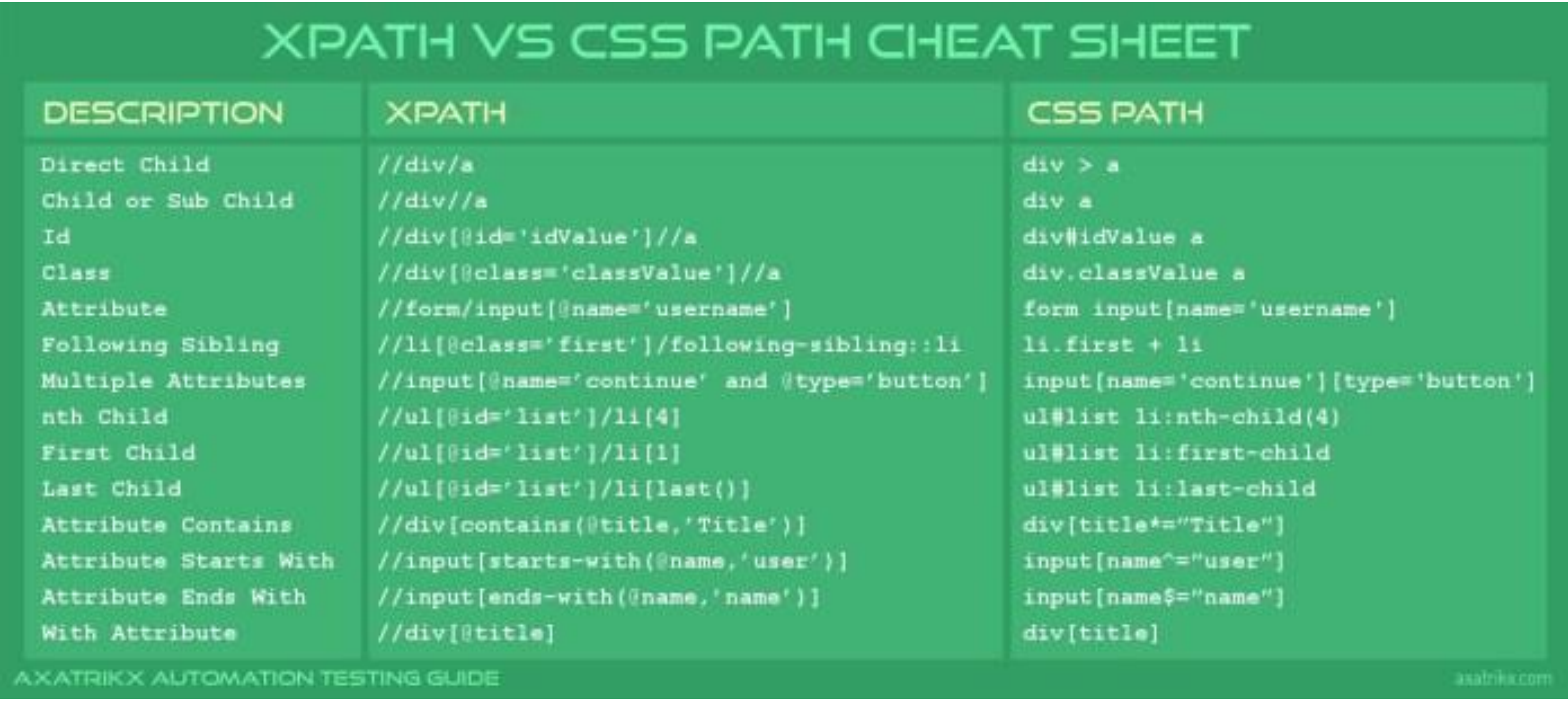
```
<p> First Let us try be very simple with only one
<ul>
<form id="get-input" method="post">
  <div class="form-group">
    <label for="message">Enter message</label>
    <input id="user-message" class="form-contr
  </div>
```

XPath vs CSSselector

- CSS is faster and simpler than Xpath particularly in case of IE browser where Xpath works very slowly.



```
+-----------------------------------+----------+----------+----------+----------+-----------+----------+----------+----------+
|                                   | Cr 32    | Cr 32    | FF 26    | FF 26    | IE 8      | IE 8     | Op 12    | Op 12    |
+-----------------------------------+----------+----------+----------+----------+-----------+----------+----------+----------+
|                                   | CSS      | XPath    | CSS      | XPath    | CSS       | XPath    | CSS      | XPath    |
| header_id_and_class               | 1.132802 | 1.132802 | 0.829401 | 1.099982 | 8.550464  | 5.743214 | 1.859979 | 2.128673 |
| header_id_class_and_direct_desc   | 1.132802 | 1.235784 | 0.922043 | 1.051453 | 9.914347  | 6.490025 | 2.358386 | 2.143988 |
| header_traversing                 | 1.214156 | 1.132800 | 0.873515 | 0.905867 | N/A       | 7.388131 | 2.098045 | 2.317037 |
| header_traversing_and_direct_desc | 1.144128 | 1.212096 | 1.002925 | 0.922043 | N/A       | 7.733464 | 2.408921 | 2.173082 |
| cell_id_and_class                 | 1.121472 | 1.719590 | 0.986748 | 0.938220 | 13.063517 | 7.812772 | 2.375230 | 2.274156 |
| cell_id_class_and_direct_desc     | 1.258541 | 1.295923 | 1.002924 | 1.069246 | 12.334165 | 7.039037 | 2.430812 | 2.853636 |
| cell_traversing                   | 1.146393 | 1.271002 | 1.156597 | 1.085422 | N/A       | 7.315236 | 2.705395 | 2.890695 |
| cell_traversing_and_direct_desc   | 1.233619 | 1.196237 | 1.459092 | 1.174391 | N/A       | 5.909296 | 2.377413 | 2.268085 |
+-----------------------------------+----------+----------+----------+----------+-----------+----------+----------+----------+
```



XPATH VS CSS PATH CHEAT SHEET

DESCRIPTION	XPATH	CSS PATH
Direct Child	//div/a	div > a
Child or Sub Child	//div//a	div a
Id	//div[@id='idValue']//a	div#idValue a
Class	//div[@class='classValue']//a	div.classValue a
Attribute	//form/input[@name='username']	form input[name='username']
Following Sibling	//li[@class='first']/following-sibling::li	li.first + li
Multiple Attributes	//input[@name='continue' and @type='button']	input[name='continue'][type='button']
nth Child	//ul[@id='list']/li[4]	ul#list li:nth-child(4)
First Child	//ul[@id='list']/li[1]	ul#list li:first-child
Last Child	//ul[@id='list']/li[last()]	ul#list li:last-child
Attribute Contains	//div[contains(@title,'Title')]	div[title*="Title"]
Attribute Starts With	//input[starts-with(@name,'user')]	input[name^="user"]
Attribute Ends With	//input[ends-with(@name,'name')]	input[name$="name"]
With Attribute	//div[@title]	div[title]

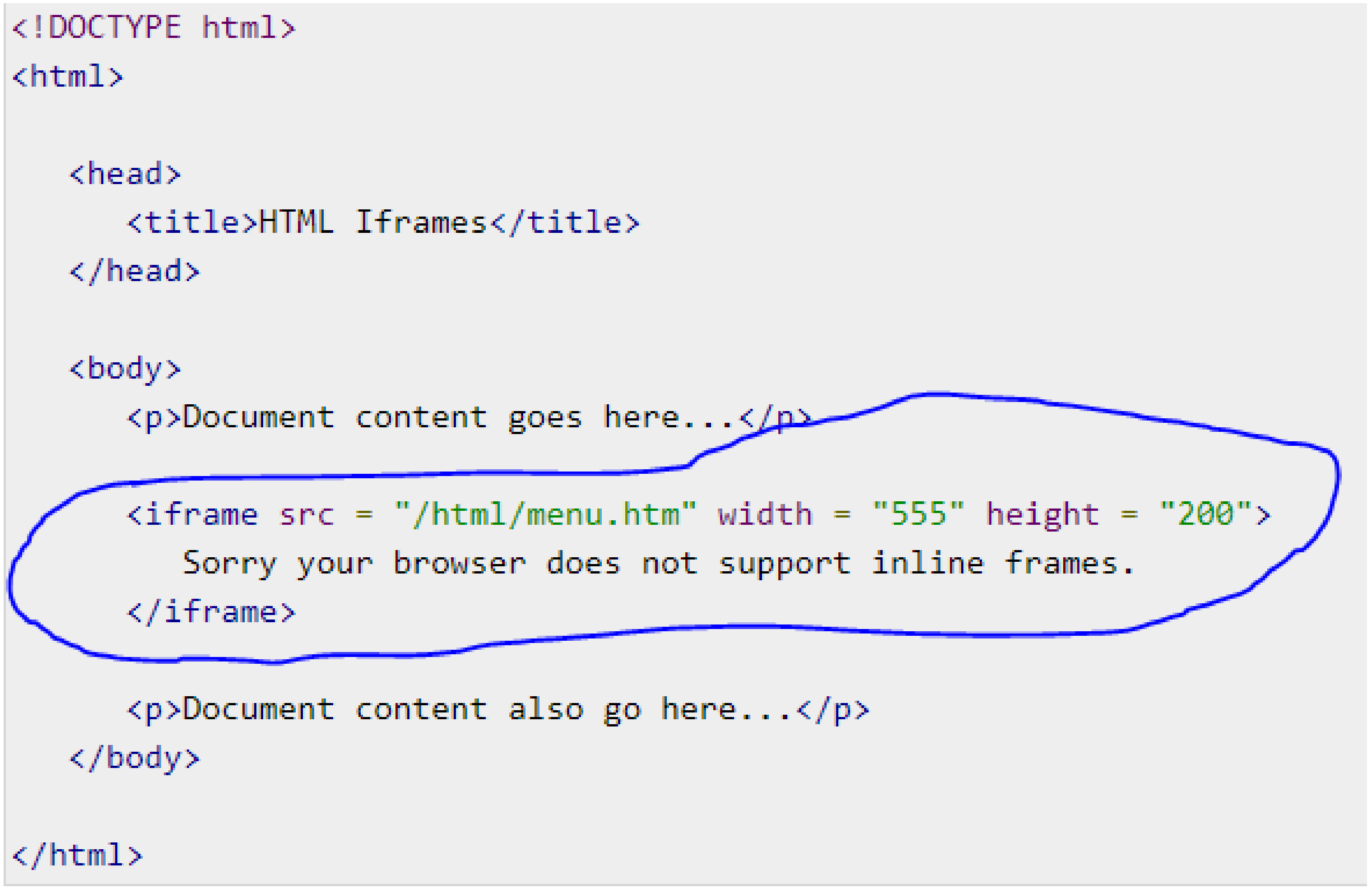
AXATRIKX AUTOMATION TESTING GUIDE

axatrikx.com

Agenda

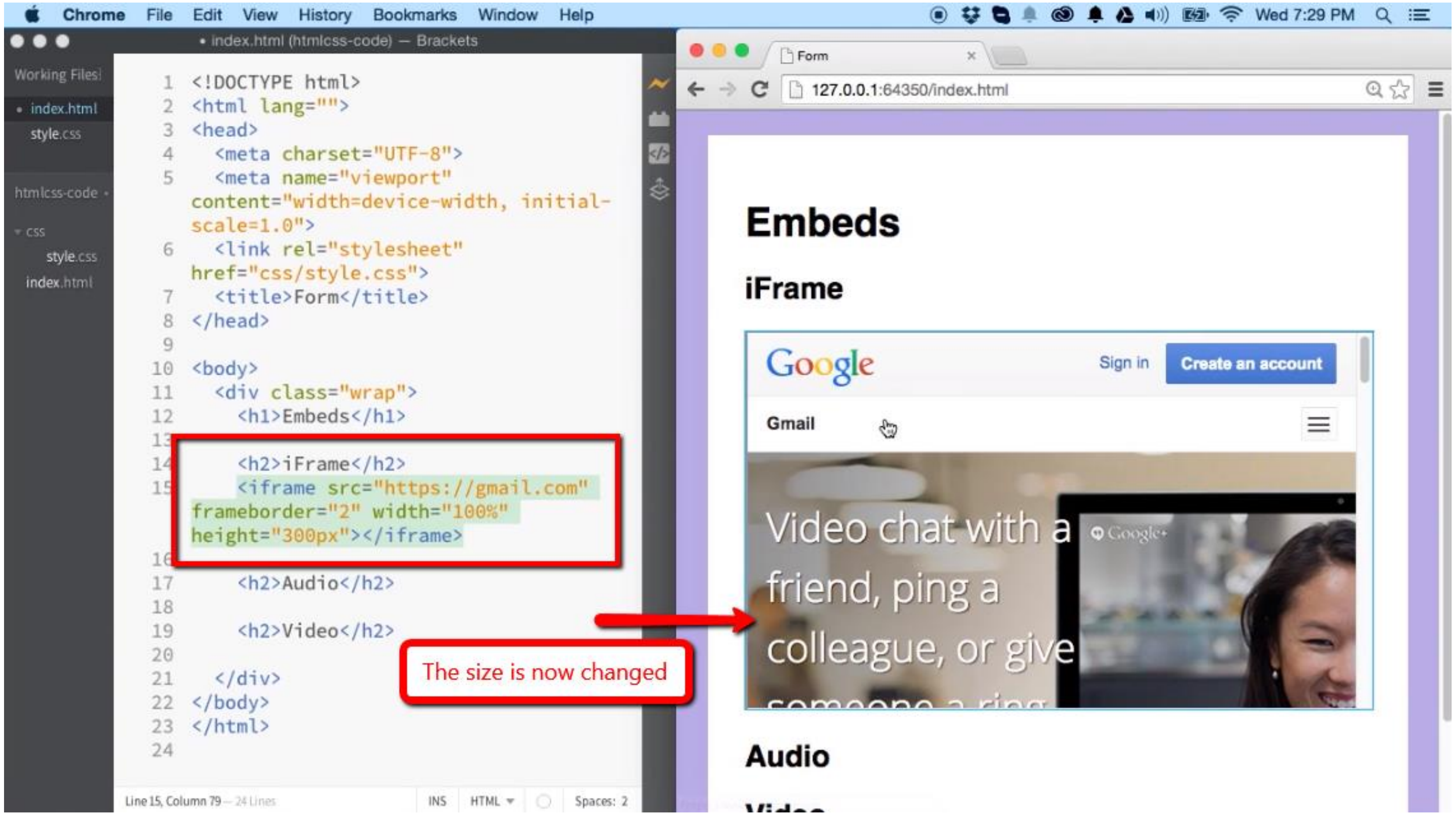
- Selenium Web driver
- Locators
- Xpath
- Css selector
- **Iframe + Pop up and alerts**
- Practice

Iframe



```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Iframes</title>
   </head>

   <body>
      <p>Document content goes here...</p>

      <iframe src = "/html/menu.htm" width = "555" height = "200">
         Sorry your browser does not support inline frames.
      </iframe>

      <p>Document content also go here...</p>
   </body>

</html>
```

Iframe

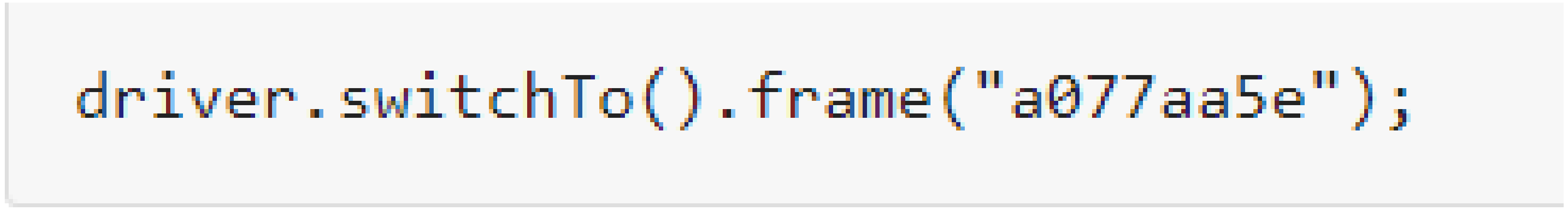


```
<!DOCTYPE html>
<html lang="">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
  <link rel="stylesheet"
href="css/style.css">
  <title>Form</title>
</head>

<body>
  <div class="wrap">
    <h1>Embeds</h1>

    <h2>iFrame</h2>
    <iframe src="https://gmail.com"
frameborder="2" width="100%"
height="300px"></iframe>

    <h2>Audio</h2>

    <h2>Video</h2>

  </div>
</body>
</html>
```

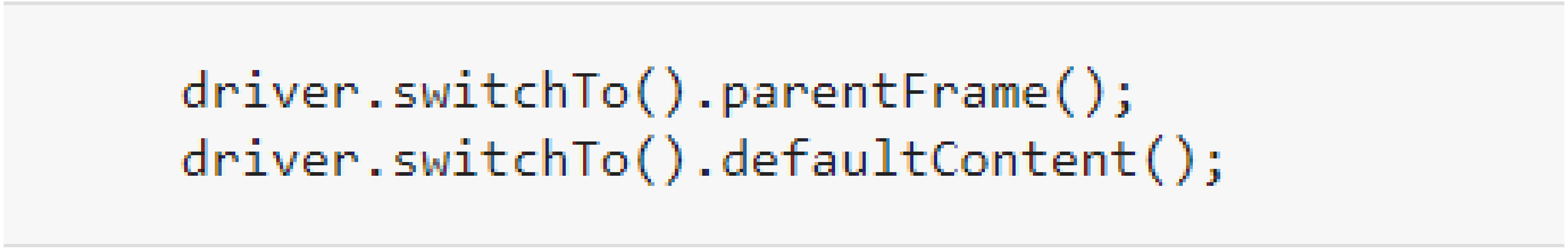
The size is now changed

Chrome File Edit View History Bookmarks Window Help
Wed 7:29 PM
• index.html (htmlcss-code) — Brackets
Working Files
index.html
style.css
htmlcss-code
css
style.css
index.html
Line 15, Column 79 — 24 Lines
INS
HTML
Spaces: 2
Form
127.0.0.1:64350/index.html
Embeds
iFrame
Google
Sign in
Create an account
Gmail
Video chat with a friend, ping a colleague, or give someone a ring
Audio
Video

Iframe



```
driver.switchTo().frame("a077aa5e");
```

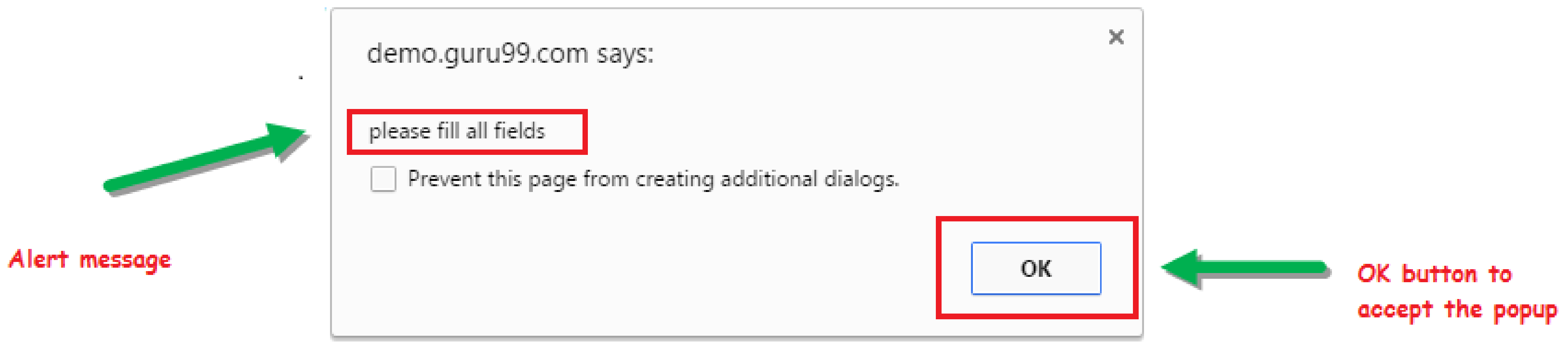


```
    driver.switchTo().parentFrame();
    driver.switchTo().defaultContent();
```

Iframe

```
/*for(int i=0; i<=size; i++){
    driver.switchTo().frame(i);
    int total=driver.findElements(By.xpath("html/body/a/img")).size();
        System.out.println(total);
    driver.switchTo().defaultContent(); //switching back from the iframe
 }*/

        //Commented the code for finding the index of the element
    driver.switchTo().frame(0); //Switching to the frame
        System.out.println("********We are switched to the iframe*******");
        driver.findElement(By.xpath("html/body/a/img")).click();

        //Clicking the element in line with Advertisement
    System.out.println("*********We are done***************");
        }
    }
```

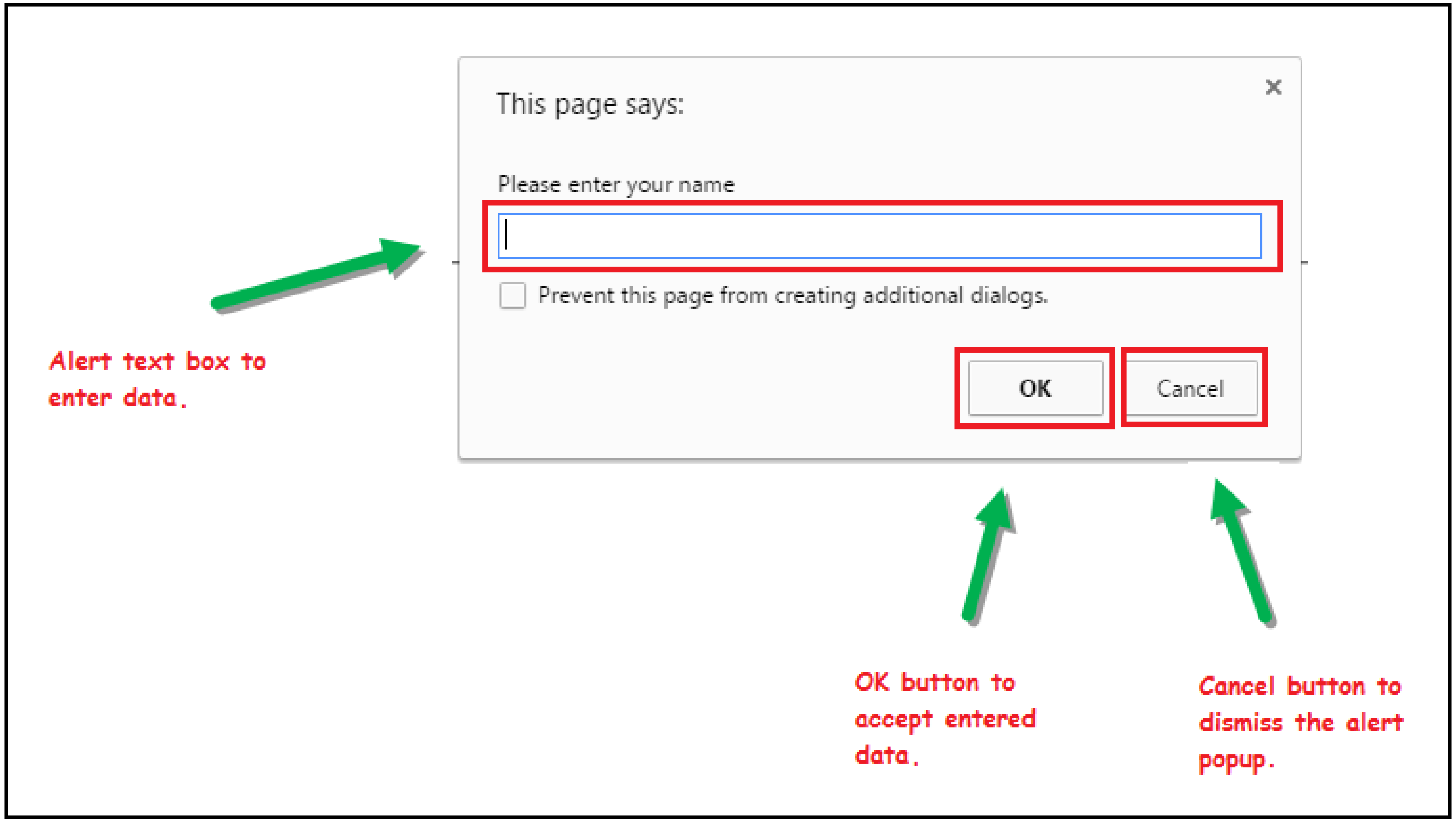
Pops and Alerts


demo.guru99.com says:

please fill all fields

Prevent this page from creating additional dialogs.

Pops and Alerts



This page says:

Please enter your name

Prevent this page from creating additional dialogs.

×
OK
Cancel
Alert text box to enter data.
OK button to accept entered data.
Cancel button to dismiss the alert popup.

Pops and Alerts

1) void dismiss() **// To click on the 'Cancel' button of the alert.**

```
driver.switchTo().alert().dismiss();
```

2) void accept() **// To click on the 'OK' button of the alert.**

```
driver.switchTo().alert().accept();
```

3) String getText**() // To capture the alert message.**

```
driver.switchTo().alert().getText();
```

4) void sendKeys(String stringToSend) **// To send some data to alert box.**

```
driver.switchTo().alert().sendKeys("Text");
```

Agenda

- Selenium Web driver
- Locators
- Xpath
- Css selector
- Iframe
- **Practice**

Reminder

- Setup Webdriver location
- Create driver
- Open url
- FindElement
- DO SOME MAGIC
- Close or Quit driver

Practice

Scenario demo

1. Go to https://www.ss.com/lv/
2. Click on “Suņi, kucēni”
3. Enter “Vecums 1”

Practice 2

- Go to https://www.ss.com/lv/
- Open cars
- Enter price 6000 - 10000
- year from 2001
- engine max 3.0
- colour – Balta
- Click submit

Practice 3

- Go to https://www.aliexpress.com/
- In search write: tattoo
- Press search
- Set min price 10
- Set max price 20
- Press ok

Practice 4

- Go to https://www.facebook.com/ or any other social network
- Login with your credentials
- Press Login
- Open your profile

Bonus Task of PAIN

1. Open https://220.lv
2. Click on menu item “Auto preces”
3. Click on category “Riepas”
4. Verify page title is “PLAŠĀKAIS AUTO PREČU KLĀSTS ! IENĀC INTERNETA VEIKALĀ 220.LV ! | 220.lv”
5. Click on “Vasaras riepas”
6. Search tires by following input “Riepas platums 215 Profils 60 Ražotājs Bridgestone”
7. Add to cart

THANK YOU FOR YOUR ATTENTION