



We are on a mission to address the digital skills gap for 10 Million+ young professionals, train and empower them to forge a career path into future tech

Locators

, 2022



Contents

- Web Inspector
- Locators
- Types of Locators

Web Inspector



What is Web/Browser Inspector

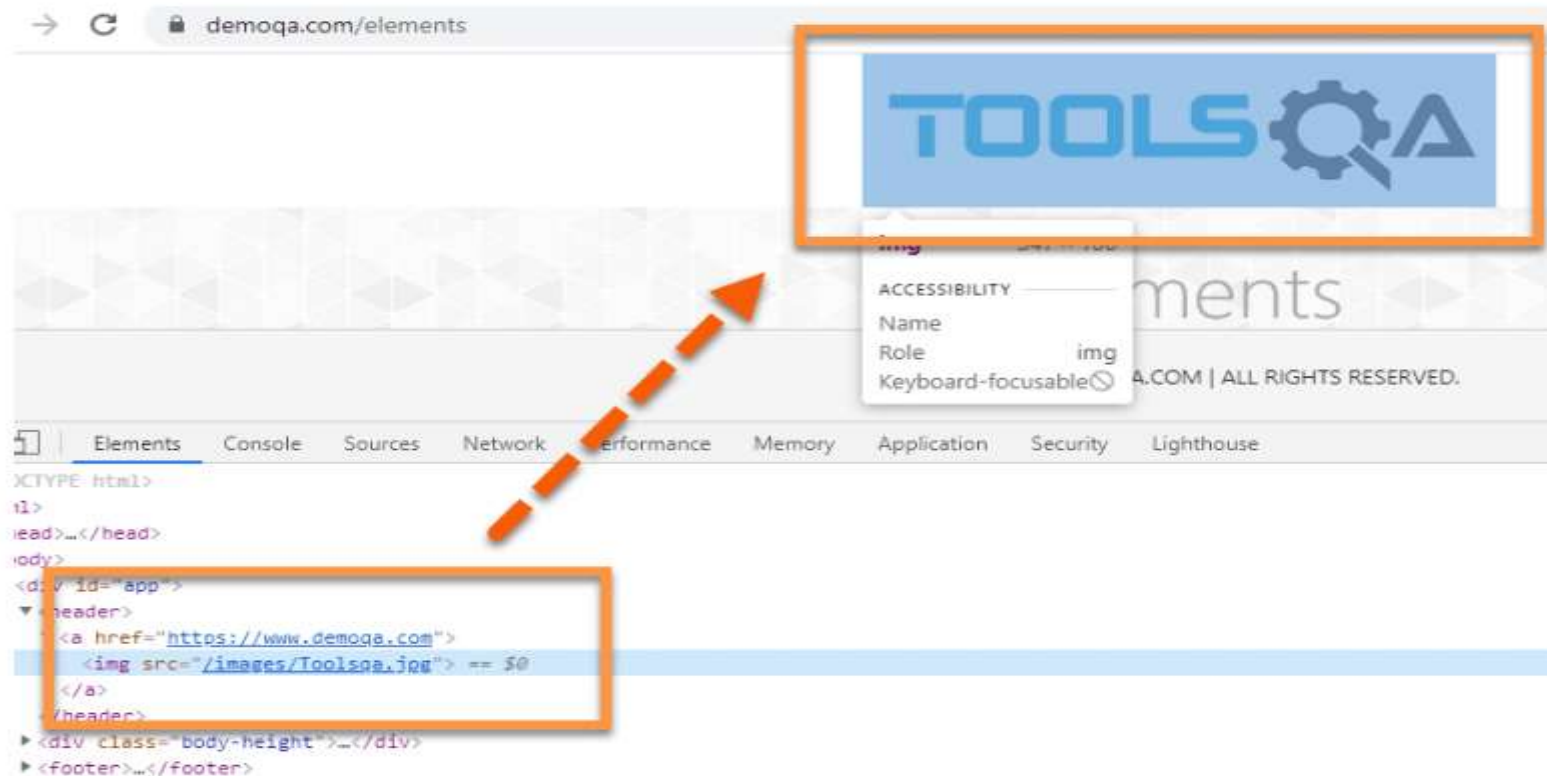
- a web inspector or a browser inspector helps to identify/locate what's inside the web page in a browser.
- The inspector shows the CSS, HTML, and JavaScript of the web page so that the developer or the tester can analyze the web page and use the browser developer tools' features to run a few checks.
- Example:
 - Checking out the color of some text on the web page.
 - For this, I can change the color code within the browser through the web inspector on that particular line and see live on the web page.
- **These changes done on the web inspector are temporary and will disappear once we close the tab.**

Inspecting Web Browser Elements

- We can open up the web Inspector using the following options:
 - 1) Right Click on the Web page and Select Inspect.
 - 2) Press Ctrl+Shift+C.
- How to inspect elements in Google Chrome?
 - 1) Launch the Google Chrome web browser and open up the inspect panel by pressing F12 or the methods described above (right-click -> inspect).
 - 2) Once you open up the inspect panel, you would notice that there are multiple tabs available on the top, such as Elements, Console, Sources, Network, and Performance. However, the Elements tab is the one which consists of all the HTML properties belonging to the current web page.

Inspecting Web Browser Elements

3) Now, if we hover the mouse pointer over the HTML tags in the DOM, it will highlight the corresponding elements it represents on the webpage.

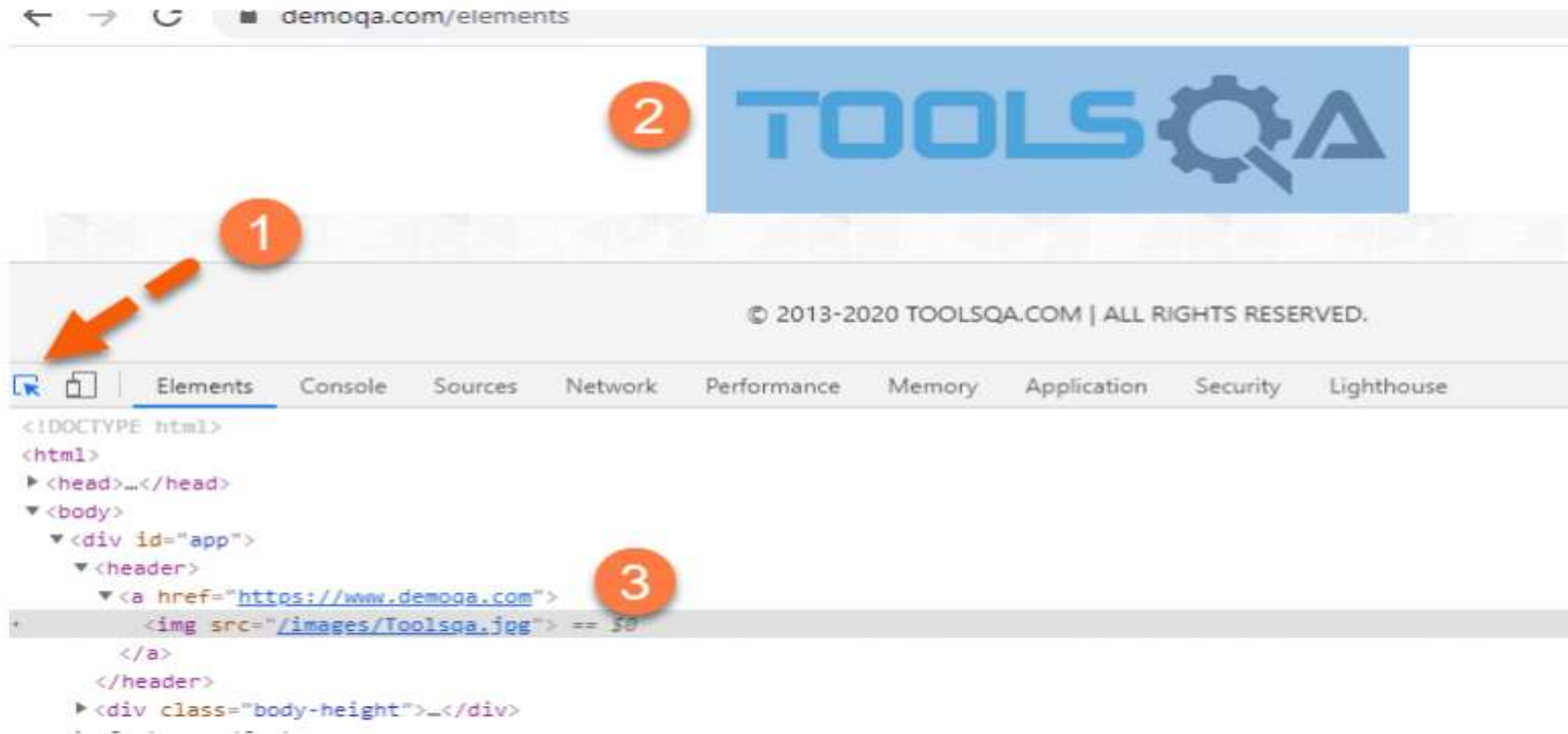


Inspecting Web Browser Elements

4) Now, the main point is, how do we find the web element in the DOM.

- Click on the "Mouse Icon" arrow (as designated by Marker 1 in below screenshot) and then select the web element on the web page.
- It will automatically highlight the corresponding HTML element in the DOM.
- Suppose we want to find the HTML elements corresponding to the banner image (as shown below by marker 3).
- When we select the mouse point and click on the banner image, it will automatically highlight the corresponding HTML element, as shown by marker 2

Inspecting Web Browser Elements



Locators



Locators

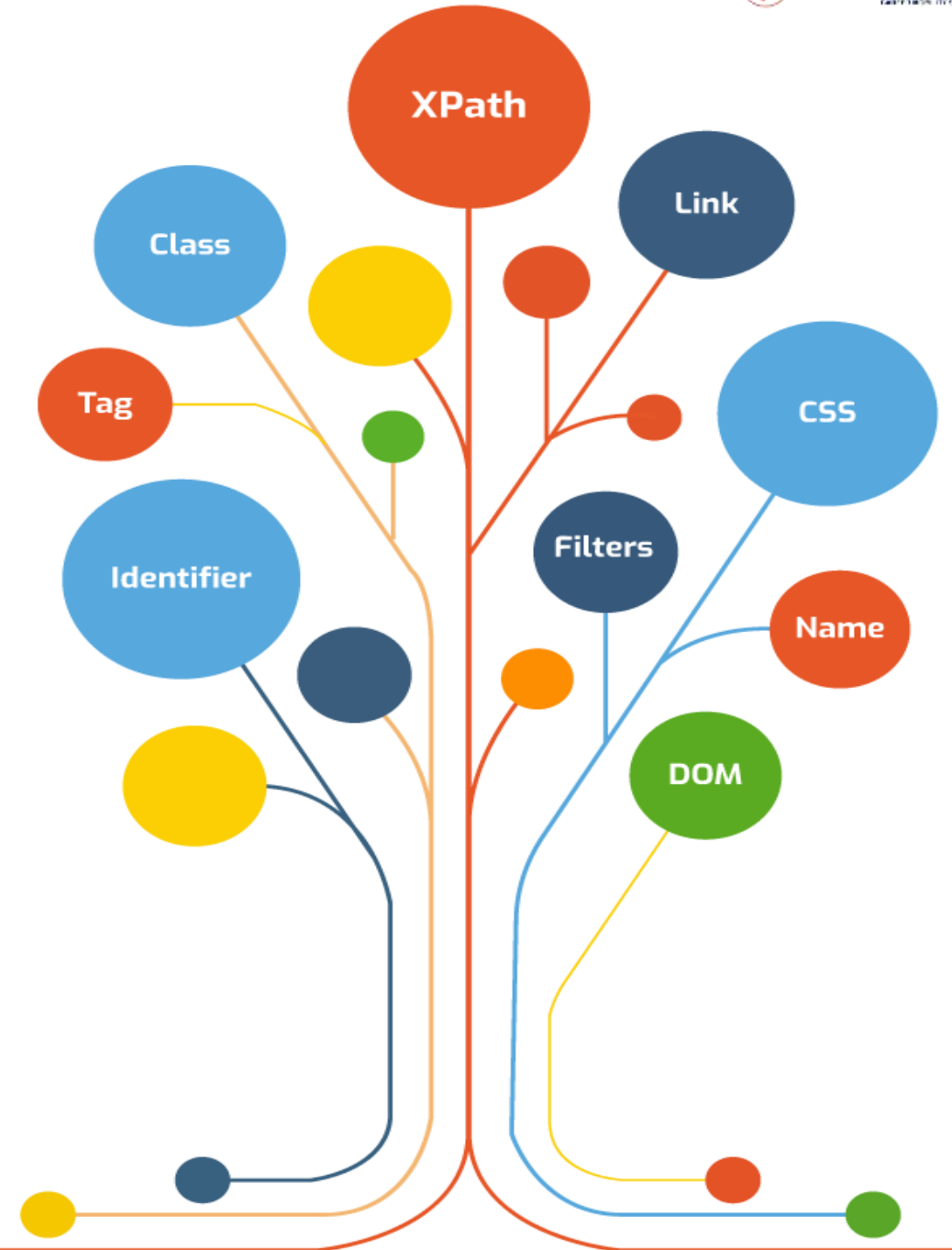
Introduction to Locators

- Identification of the correct GUI element on a web page is pre-requisite for creating any successful automation script.
- Locating an element is one of the most important tasks in selenium tests but there is no fixed strategy that you can follow to identify an element (object) uniquely on the web page.
- Locating any element on a web page is complex task.
- It totally depends on the user's ideas and their comfort level in locating web elements.
- To recognize the element (Object) uniquely on the web page, Selenium gives us some locator strategies.

Locators

What is Locator?

- Locator in selenium is like an address that is used to identify the web element uniquely on the web page.
- In other words, Locator is a unique command used to identify web elements placed within a web page.
- And almost all UI automation tools provide the capability to use locators for the identification of HTML elements on a web page.
- It is very important to use the correct locator for quick execution testing.



Types of Locators

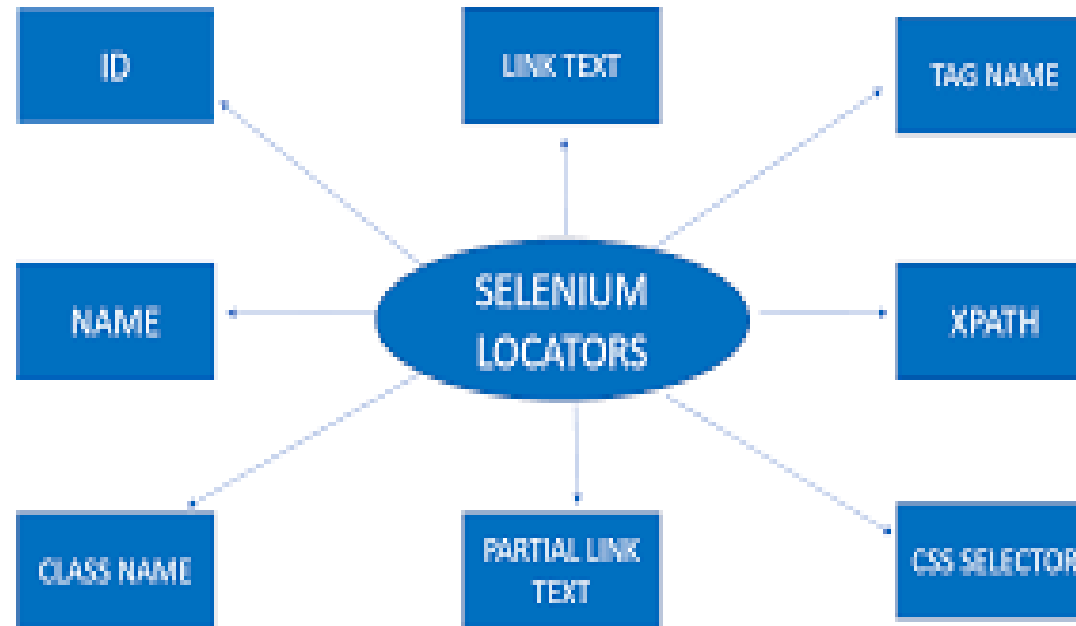


Types of Locators

Selenium Locators Types

- There are various types of locators, using which we can identify a web element uniquely on the Webpage.
- Selenium supports the following locators:

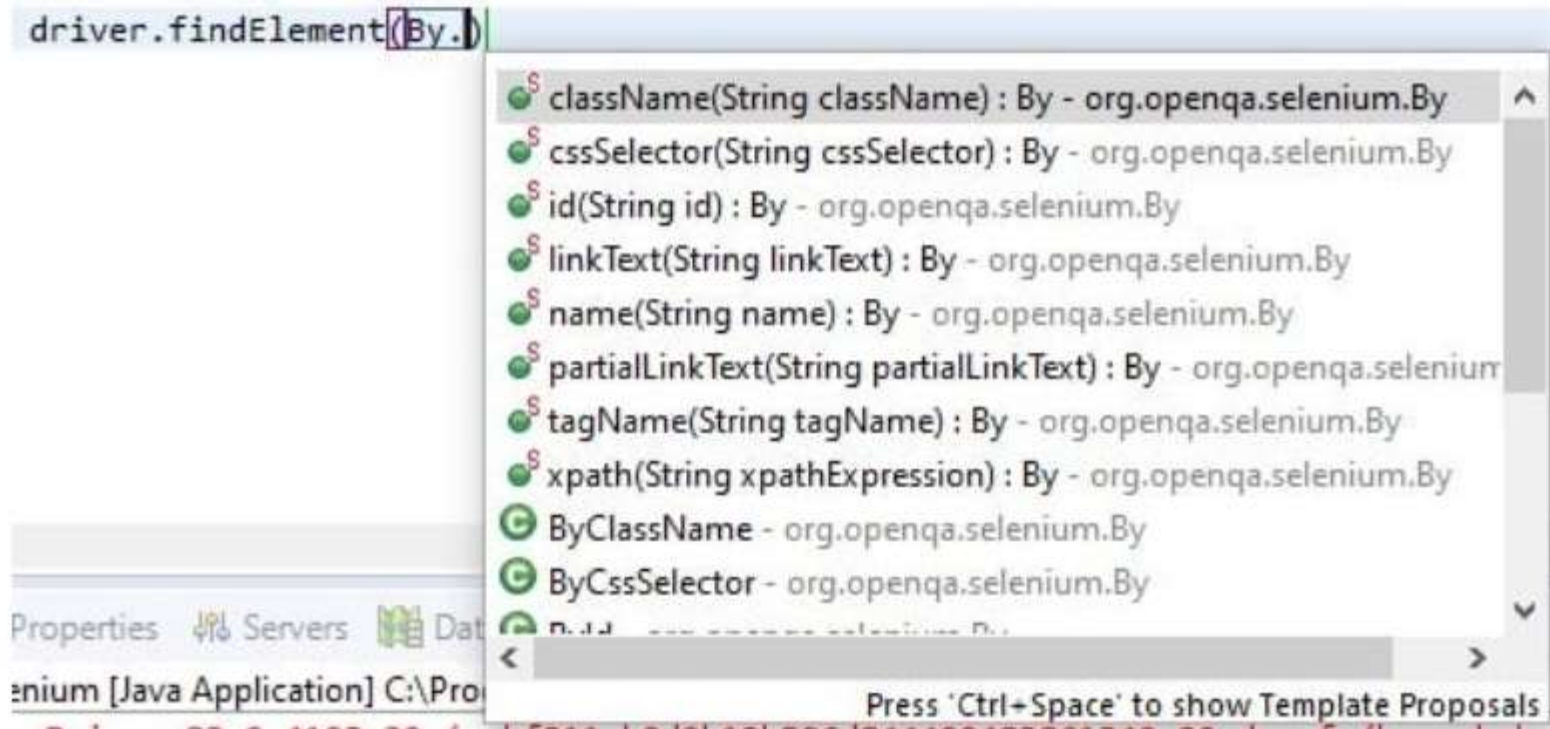
1. Id
2. Name
3. Class Name
4. HTML Tag Name
5. Linked Text
6. Partial Link Text
7. Xpath
8. CSS



Types of Locators

Selenium Locators Types

- You can quickly identify all the supported locators by Selenium by browsing all the visible methods on the "By" class, as shown below:



Locators

Id

- ID locator is the safest and fastest locator to find the location of an element based on the value of “ID” attribute on a web page.
- Since ID is unique for each element on the web page so it can be easily identified. It is always the first choice for testers. It is mostly an account number, college name, username, password, or sign in button that will be unique.
- The general syntax to use id locator is as follows:

WebElement Element = driver.FindElement(By.Id(“Attribute value”));

where,

driver ➡ Object reference variable of the WebDriver object.

WebElement ➡ Return type of the findElement() method.

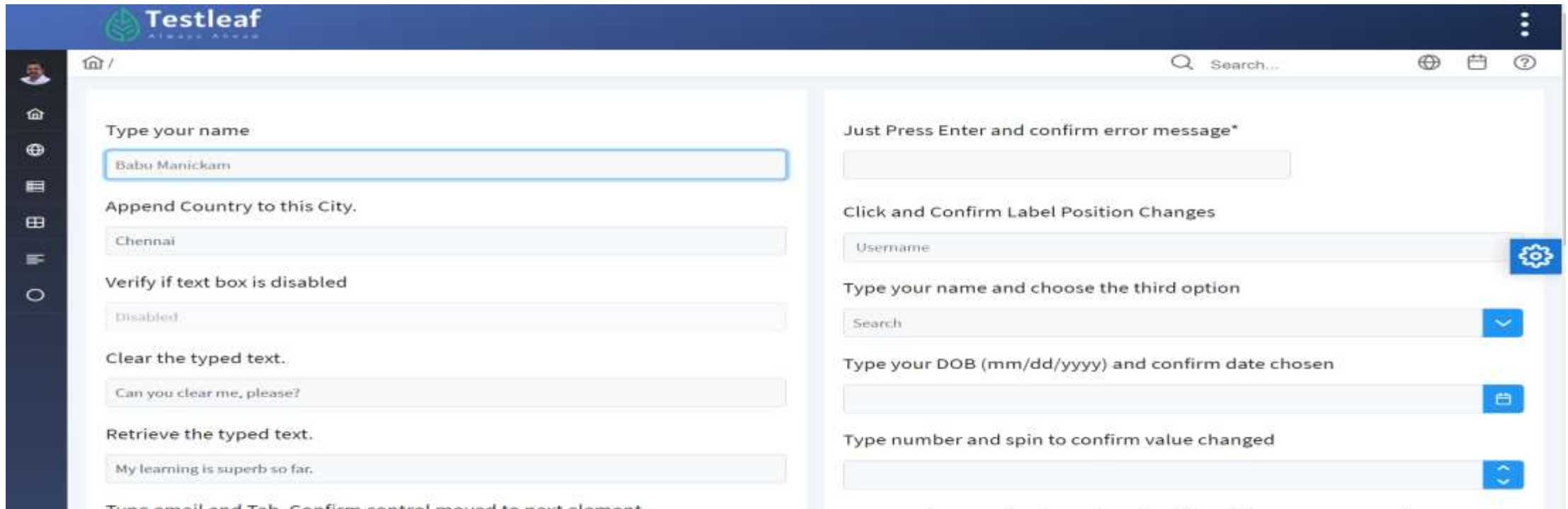
element ➡ Variable used to store the returning web element.

By ➡ By is a class that extends java.lang.Object. It provides a mechanism that is used to locate elements within a document.

Locators

Id Example

- Launch google chrome browser and navigate to the following testing website Url “https://www.leafground.com/input.xhtml” you can see the text box **Type your name**.
- We will locate the Type your name text box with the help of Id Locator.

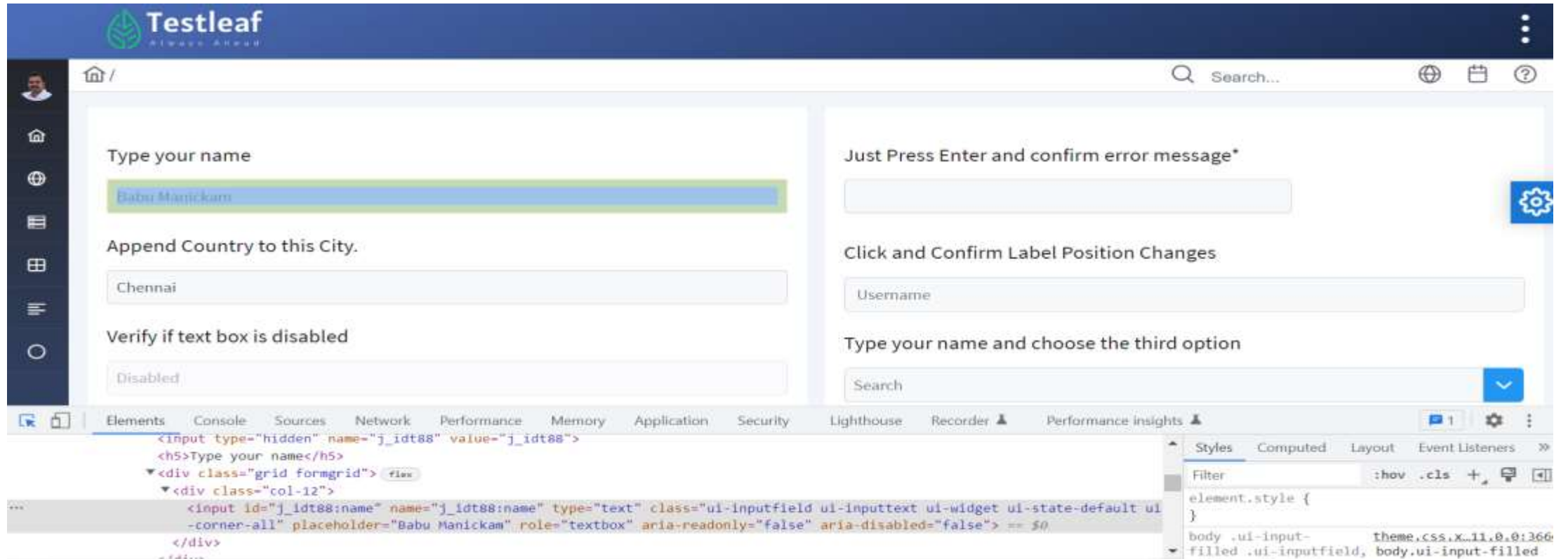


The screenshot shows the Testleaf website interface. The header includes the Testleaf logo and a search bar. The main content area is divided into two columns. The left column contains several text input fields with labels: "Type your name" (containing "Babu Manickam"), "Append Country to this City." (containing "Chennai"), "Verify if text box is disabled" (containing "Disabled"), "Clear the typed text." (containing "Can you clear me, please?"), "Retrieve the typed text." (containing "My learning is superb so far."), and "Type email and Tab. Confirm control moved to next element". The right column contains: "Just Press Enter and confirm error message*", "Click and Confirm Label Position Changes" (containing "Username" with a settings icon), "Type your name and choose the third option" (containing "Search" with a dropdown arrow), "Type your DOB (mm/dd/yyyy) and confirm date chosen" (containing a date picker icon), and "Type number and spin to confirm value changed" (containing a spinner icon).

Locators

Id Example

- You will notice that there are input tag, class, name and id. Use Id `j_idt88:name` to locate the text box element Type your name.



The screenshot shows the Testleaf application interface with a sidebar on the left and a main content area. The main content area contains three sections: "Type your name" with a text input field containing "Babu Manickam", "Append Country to this City" with a text input field containing "Chennai", and "Verify if text box is disabled" with a disabled text input field containing "Disabled". On the right side, there are three more sections: "Just Press Enter and confirm error message*" with a text input field, "Click and Confirm Label Position Changes" with a text input field containing "Username", and "Type your name and choose the third option" with a text input field containing "Search". The browser developer tools are open at the bottom, showing the HTML structure of the "Type your name" input field. The HTML code is as follows:

```
<input type="hidden" name="j_idt88" value="j_idt88">
<h5>Type your name</h5>
<div class="grid formgrid">
  <div class="col-12">
    <input id="j_idt88:name" name="j_idt88:name" type="text" class="ui-inputfield ui-inputtext ui-widget ui-state-default ui-corner-all" placeholder="Babu Manickam" role="textbox" aria-readonly="false" aria-disabled="false"> == $0
  </div>
</div>
```

The Styles panel on the right shows the default styles for the input field, including the placeholder text "Babu Manickam".

Java Selenium Program – Id Locator

```
package com.mysamp1;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

import io.github.bonigarcia.wdm.WebDriverManager;

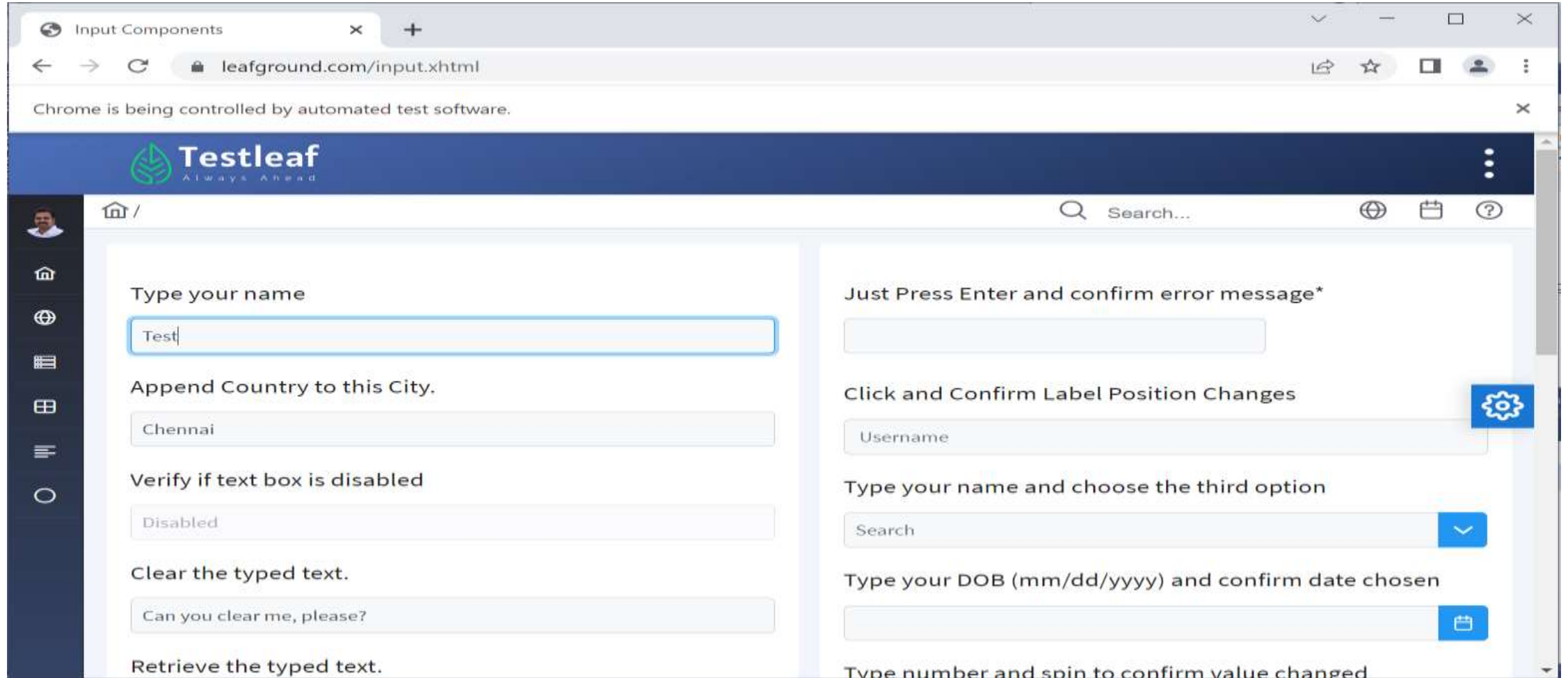
public class navigate {

    public static void main(String[] args) {
        WebDriverManager.chromedriver().setup();
        ChromeOptions co=new ChromeOptions();
        WebDriver driver=new ChromeDriver(co);
        driver.get("https://www.leafground.com/input.xhtml");
        driver.manage().window().maximize();
        driver.findElement(By.id("j_idt88:name")).sendKeys("Ajay");

    }

}
```

Selenium Program – Id Locator - Output



The screenshot shows a web browser window with the URL `leafground.com/input.xhtml`. The page is titled "Input Components" and displays a collection of form elements for testing purposes. The interface includes a dark blue header with the "Testleaf" logo and a search bar. A sidebar on the left contains navigation icons. The main content area is divided into two columns of input components:

- Left Column:**
 - "Type your name" with a text input field containing "Test".
 - "Append Country to this City." with a text input field containing "Chennai".
 - "Verify if text box is disabled" with a disabled text input field containing "Disabled".
 - "Clear the typed text." with a text input field containing "Can you clear me, please?".
 - "Retrieve the typed text." (no input field visible).
- Right Column:**
 - "Just Press Enter and confirm error message*" with a text input field.
 - "Click and Confirm Label Position Changes" with a text input field containing "Username" and a blue gear icon.
 - "Type your name and choose the third option" with a dropdown menu showing "Search".
 - "Type your DOB (mm/dd/yyyy) and confirm date chosen" with a date picker input field.
 - "Type number and spin to confirm value changed" (no input field visible).

Locators

Name

- Every element in the webpage has many attributes, and a Name is one among them. It is a very efficient strategy to locate an element by name.
- Name locator is the second safest and fastest locator to locate an element based on the value of “name” attribute on the web page. The name cannot be unique for each element at all times.
- If there are multiple elements with the same name on a web page then Selenium will always perform the action on the first matching element.
- Username and Password fields are examples that can be identified with “name” attribute. The general syntax to use name locator is as follows:

Locators

Name

- The general syntax to use name locator is as follows:

WebElement Element = driver.FindElement(By.Name("Attribute value"));

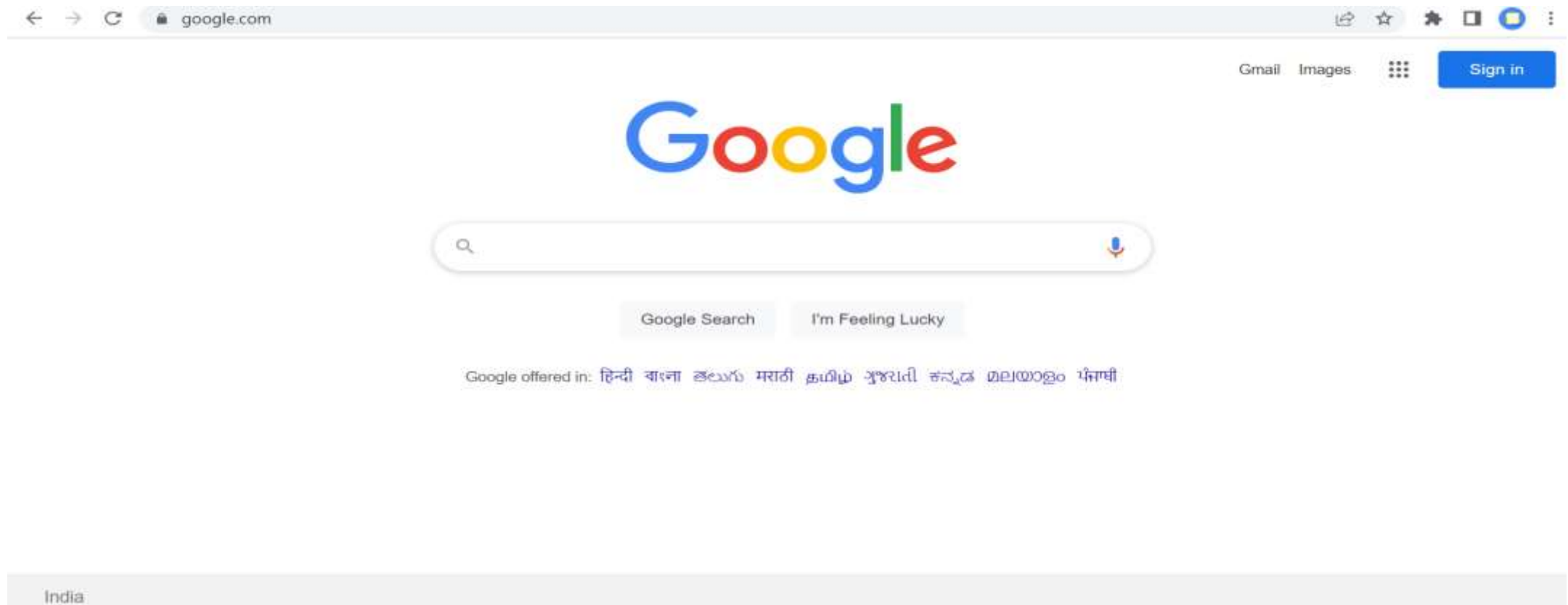
- Example

WebElement Element = driver.FindElement(By.Name("username"));

Locators

Name Example

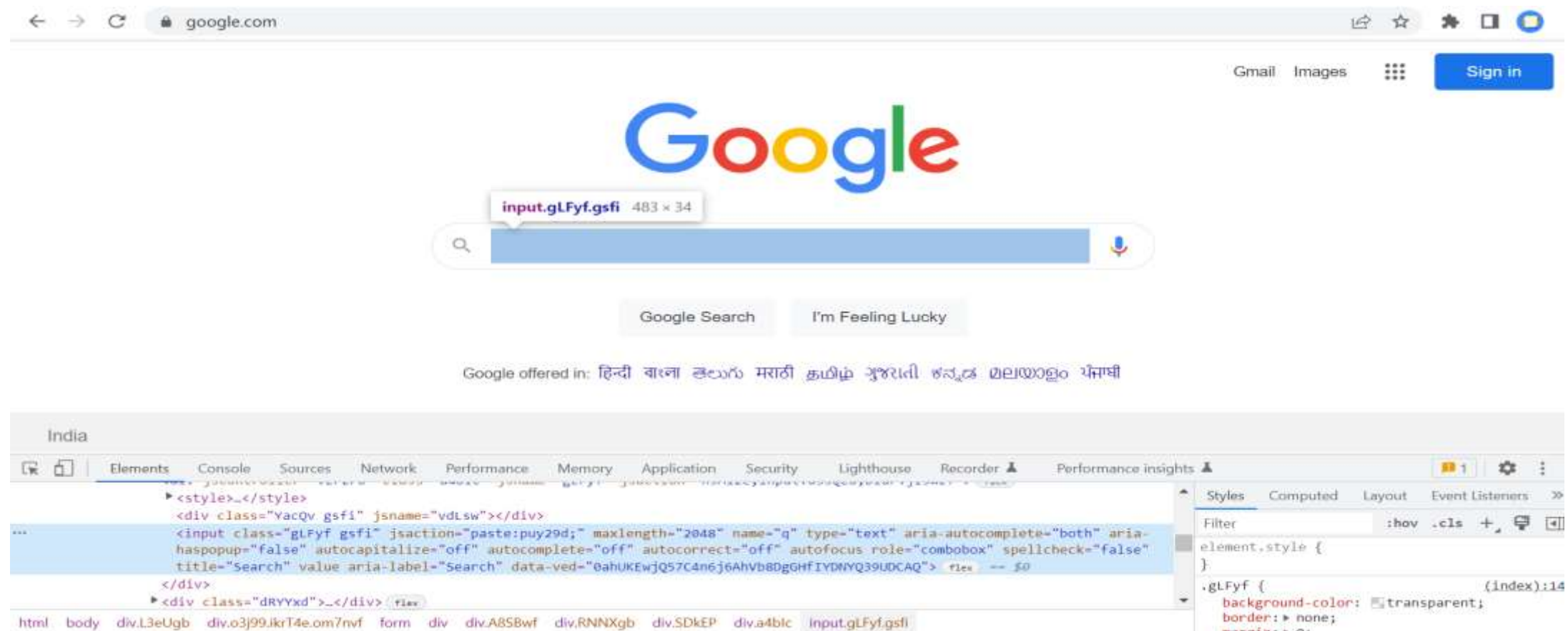
- Launch google chrome browser and navigate to google website. You can see search text box.
- Now will locate the search text box with the help of Name Locator.



Locators

Name Example

- You will notice that there are input tag, class, name and id. Use q to locate the search text box.



The screenshot shows the Google homepage. The search bar is highlighted with a tooltip displaying the locator `input.gLFyf.gsfi` and its dimensions `483 x 34`. Below the page, the Chrome DevTools 'Elements' panel is open, showing the HTML structure of the search bar. The HTML snippet is as follows:

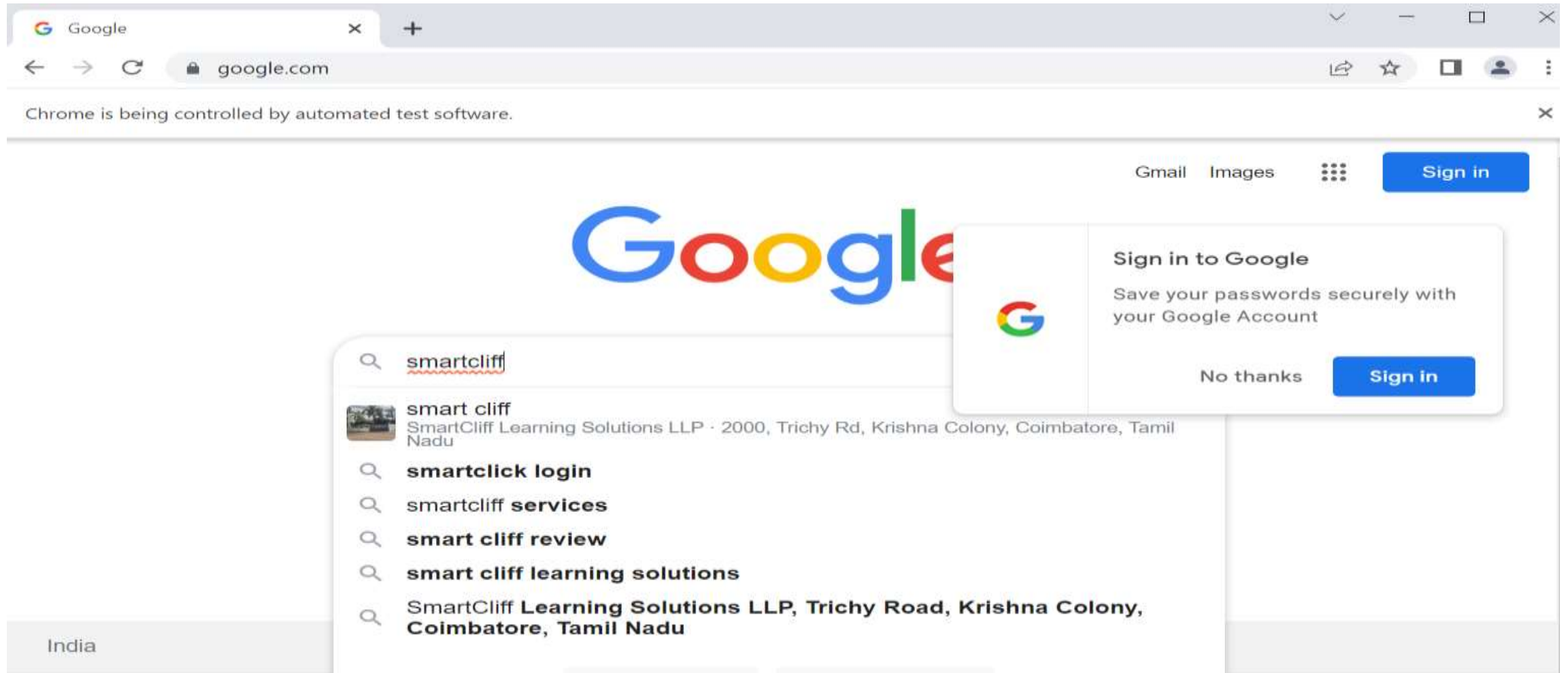
```
<div class="YacQv gsfi" jsname="vdisw"></div>
...
<input class="gLFyf gsfi" jsaction="paste:puy29d;" maxlength="2048" name="q" type="text" aria-autocomplete="both" aria-haspopup="false" autocapitalize="off" autocomplete="off" autocorrect="off" autofocus role="combobox" spellcheck="false" title="Search" value="" data-ved="0ahUKEWjQ57C4n6j6AhVb8DgGHfIYDNyQ39UDCAQ">
</div>
<div class="dRYYxd"></div>
```

The 'Styles' panel on the right shows the default styles for the `.gLFyf` class, including `background-color: transparent` and `border: none`.

Selenium Program – Name Locator

```
public class TestScript {  
    public static void main(String[] args)  
    {  
        WebDriverManager.chromedriver().setup();  
        ChromeOptions co=new ChromeOptions();  
        WebDriver driver=new ChromeDriver(co);  
        driver.get("https://www.google.com");  
        WebElement E1 = driver.FindElement(By.Name("q"));  
        E1.SendKeys("smartcliff");  
    }  
}
```

Selenium Program – Name Locator - Output



Locators

Class Name

- ClassName allows to find web elements based on the class values of the DOM.
- For example, if we want to identify or perform any operation on the form element, we can use the class to identify it.
- To identify it successfully, we need to make sure that the class name value that we are using for locating the web element, i.e., web form, is unique, and any other class doesn't have the same value.
- If any other class contains the same value as this, then Selenium will select whichever element comes first while scrapping through the web page.
- The general syntax to use id locator is as follows:

WebElement Element = driver.FindElement(By.ClassName("Attribute value"));

Locators

ClassName Example

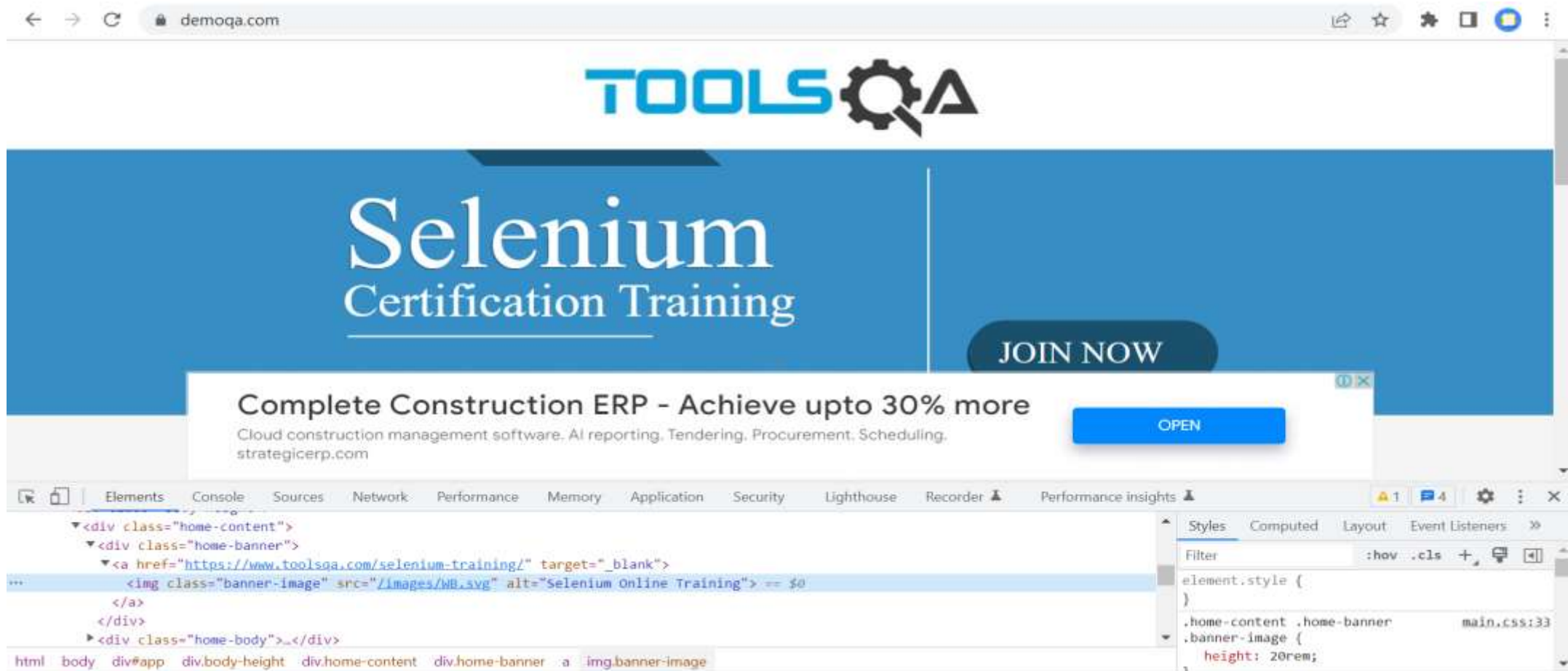
- Launch google chrome browser and navigate to toolsqa website <https://www.demoqa.com/>. You can see banner image.
- Now will locate the search text box with the help of ClassName Locator.



Locators

ClassName Example

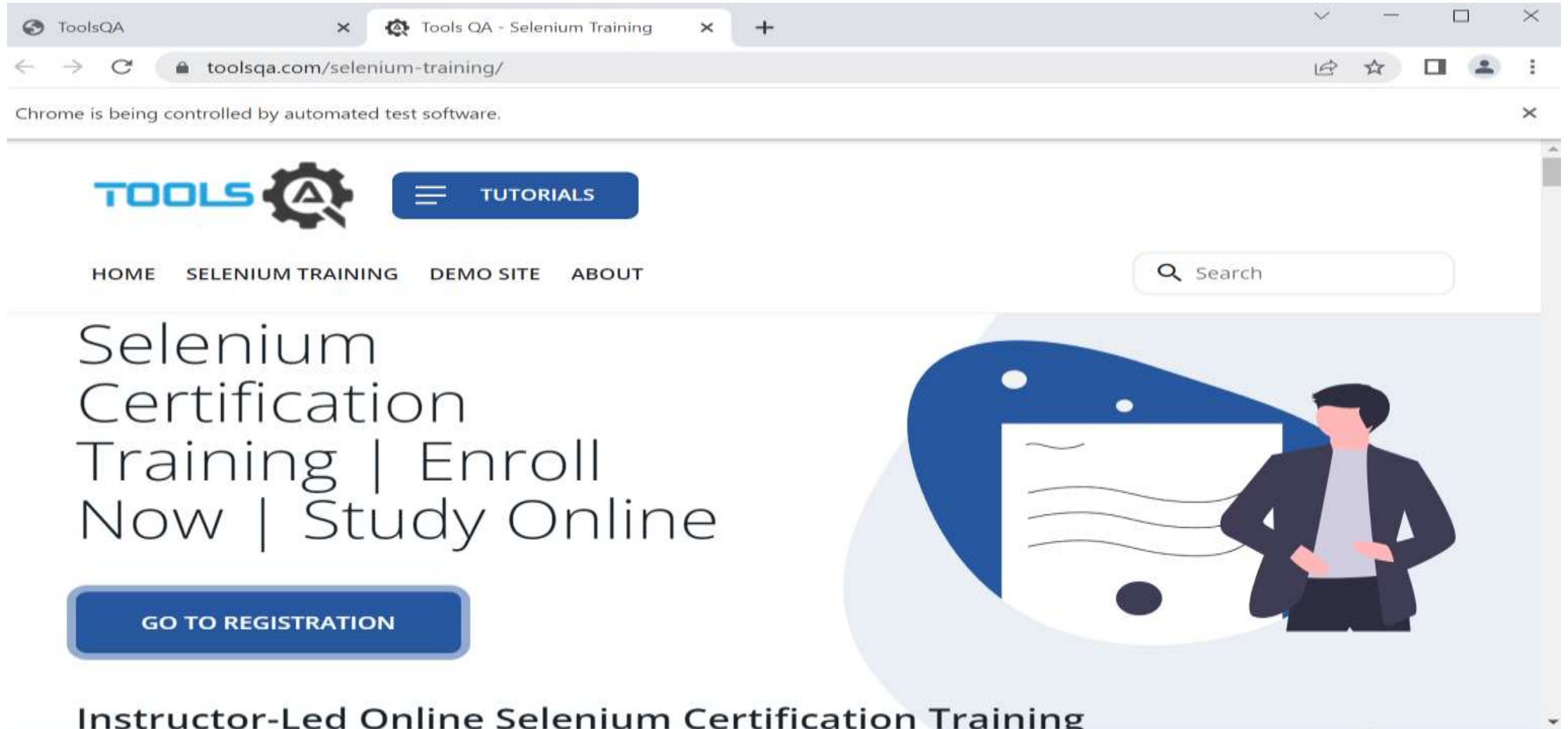
- You will notice that the form image tag with class = "banner-image".



Selenium Program – ClassName Locator

```
public class TestScript {  
    public static void main(String[] args)  
    {  
        WebDriverManager.chromedriver().setup();  
        ChromeOptions co=new ChromeOptions();  
        WebDriver driver=new ChromeDriver(co);  
        driver.get("https://demoqa.com/");  
        WebElement E1 = driver.FindElement(By.ClassName("banner-image"));  
        E1.click();  
    }  
}
```

Selenium Program – ClassName Locator - Output



Locators

TagName

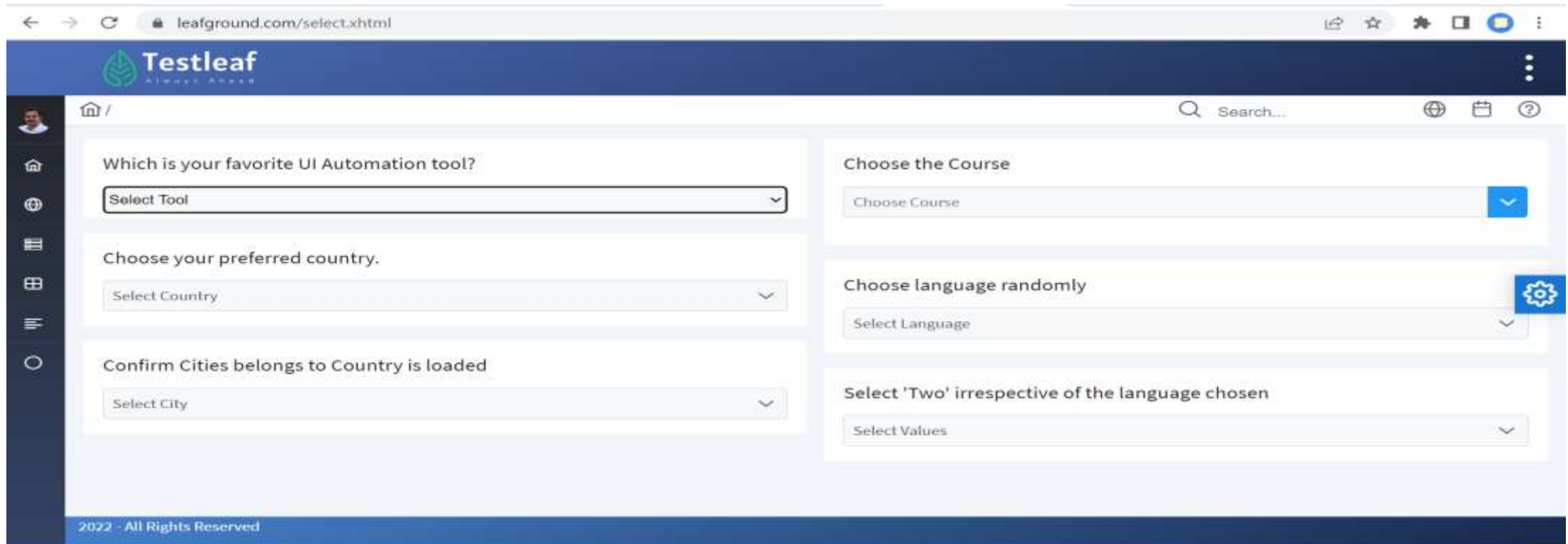
- This locator type uses tag names to identify elements in a web page. The tag name is the HTML tag, such as `input`, `div`, `anchor tag`, `button`, etc.
- TagName locator is used to finding an element from the group elements like checkboxes, drop-downs, etc. It locates an element by its tag name.
- The same tag will be present multiple times on a web page, so it's advisable to use `findElements()` method when using the TagName locator.
- The general syntax to use id locator is as follows:

`WebElement Element = driver.FindElement(By.TagName("Attribute value"));`

Locators

TagName Example

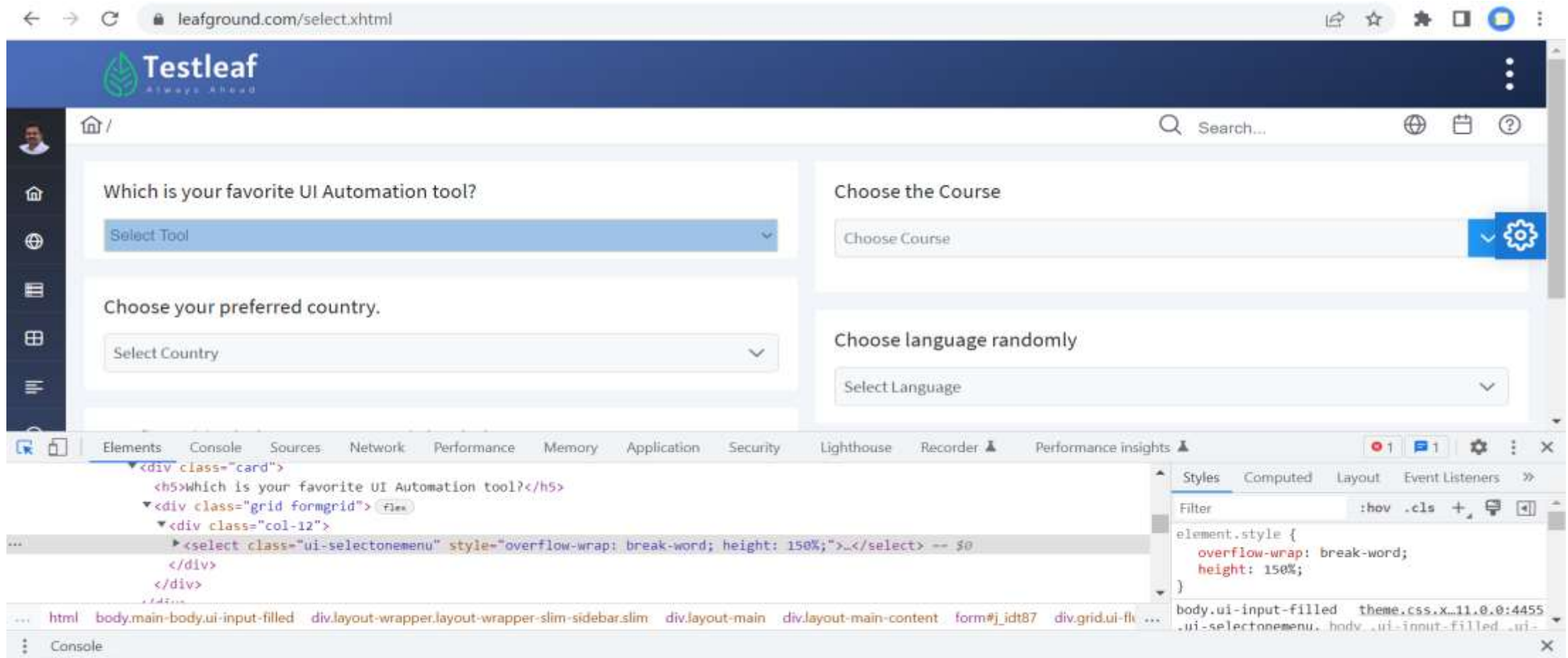
- Launch google chrome browser and navigate to <https://www.leafground.com/select.xhtml>. You can see select tool drop-down list.
- Now will locate the select tool drop-down list with the help of TagName Locator.



Locators

TagName Example

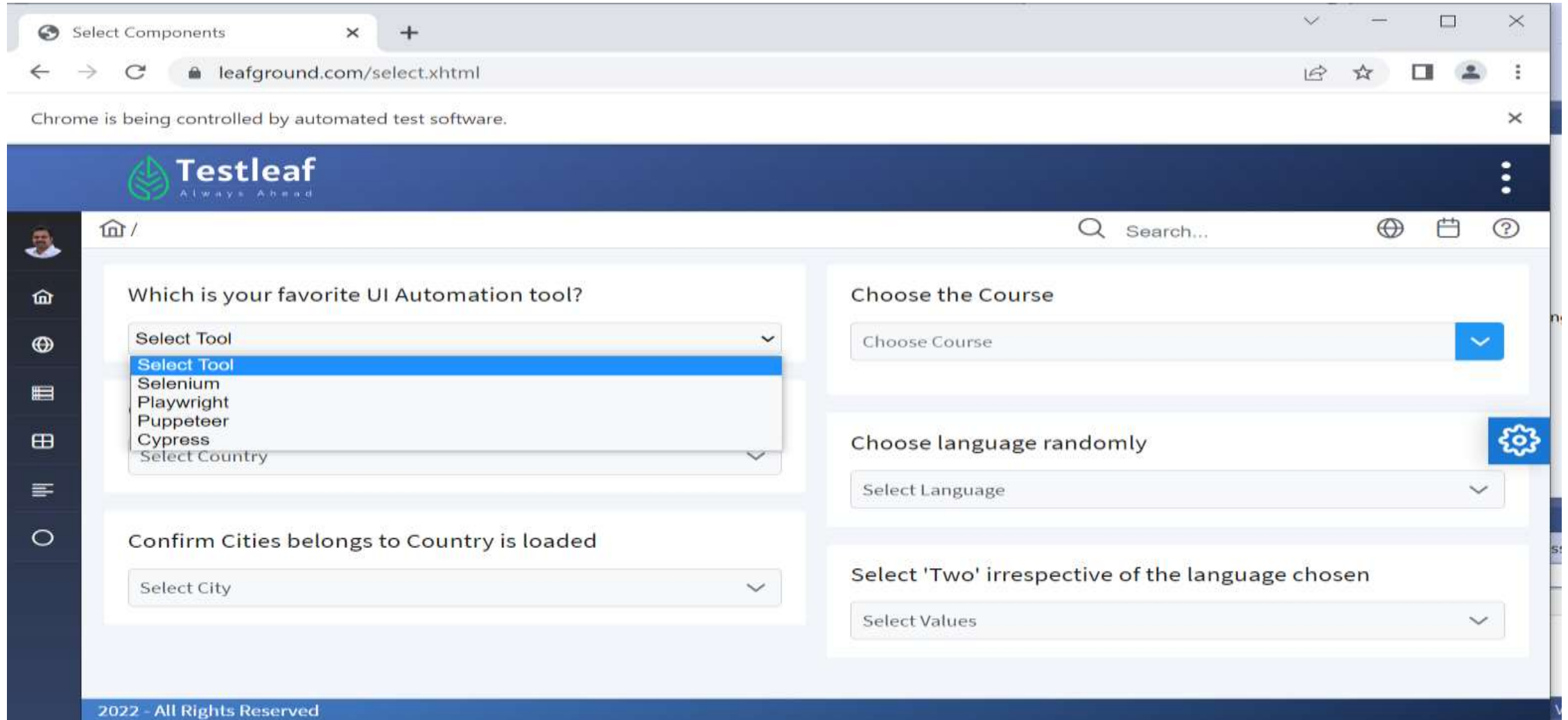
- You will notice that the select tool drop-down list with tag = "select".



Selenium Program – TagName Locator

```
public class TestScript {  
    public static void main(String[] args)  
    {  
        WebDriverManager.chromedriver().setup();  
        ChromeOptions co=new ChromeOptions();  
        WebDriver driver=new ChromeDriver(co);  
        driver.get("https://www.leafground.com/select.xhtml");  
        WebElement E1 = driver.FindElement(By.TagName("select"));  
        E1.click();  
    }  
}
```


Selenium Program – TagName Locator - Output



Locators

LinkText

- This locator finds an element by the link text. If there is one unique element on the web page, you can easily find an element with the link text.
- The LinkText helps you in identifying the hyperlinks on a webpage. You must use anchor tag (<a>) for completing this action. If you want to create hyperlinks you must use both anchor tags and then LinkText.
- The general syntax to use id locator is as follows:

WebElement Element = driver.FindElement(By.LinkText("Attribute value"));

Selenium Program – LinkText Locator

```
public class TestScript {  
    public static void main(String[] args)  
    {  
        WebDriverManager.chromedriver().setup();  
        ChromeOptions co=new ChromeOptions();  
        WebDriver driver=new ChromeDriver(co);  
        driver.get("https://www.google.com/");  
        WebElement E1 = driver.findElement(By.LinkText("Gmail"));  
  
        E1.click();    }  
    }  
}
```

Locators

PartialLinkText

- This locator locates an element by a partial match of its link text and then performs actions on it. It works in the same way as the link text.
- We can use this technique to use only one piece of text that should be an anchor tag to locate a web element.
- The general syntax to use id locator is as follows:

WebElement Element = driver.FindElement(By.PartialLinkText("Attribute value"));

Selenium Program – PartialLinkText Locator

```
public class TestScript {  
    public static void main(String[] args)  
    {  
        WebDriverManager.chromedriver().setup();  
        ChromeOptions co=new ChromeOptions();  
        WebDriver driver=new ChromeDriver(co);  
        driver.get("https://www.google.com/");  
        WebElement E1 = driver.findElement(By.partialLinkText("Gm"));  
  
        E1.click();    }  
    }  
}
```

Types of Locators

Xpath

Locator – Xpath

- XPath stands for XML Path.
- It's a query language that helps identify elements from an XML document.
- It uses expressions that navigate into an XML document in a way that can be traced from the start to the intended element like forming a path from the start



Xpath

Types of XPath

Xpath Types

- Absolute Xpath
- Relative Xpath
- The node is selected by following a path or steps. We can use XPath to find the location of any element on a webpage.



Selecting Nodes in XPath:

Expression	Description
/	Selects from the root node
//	Selects elements relative to reference node that match the selection anywhere inside the HTML DOM
@	Selects attributes

Xpath

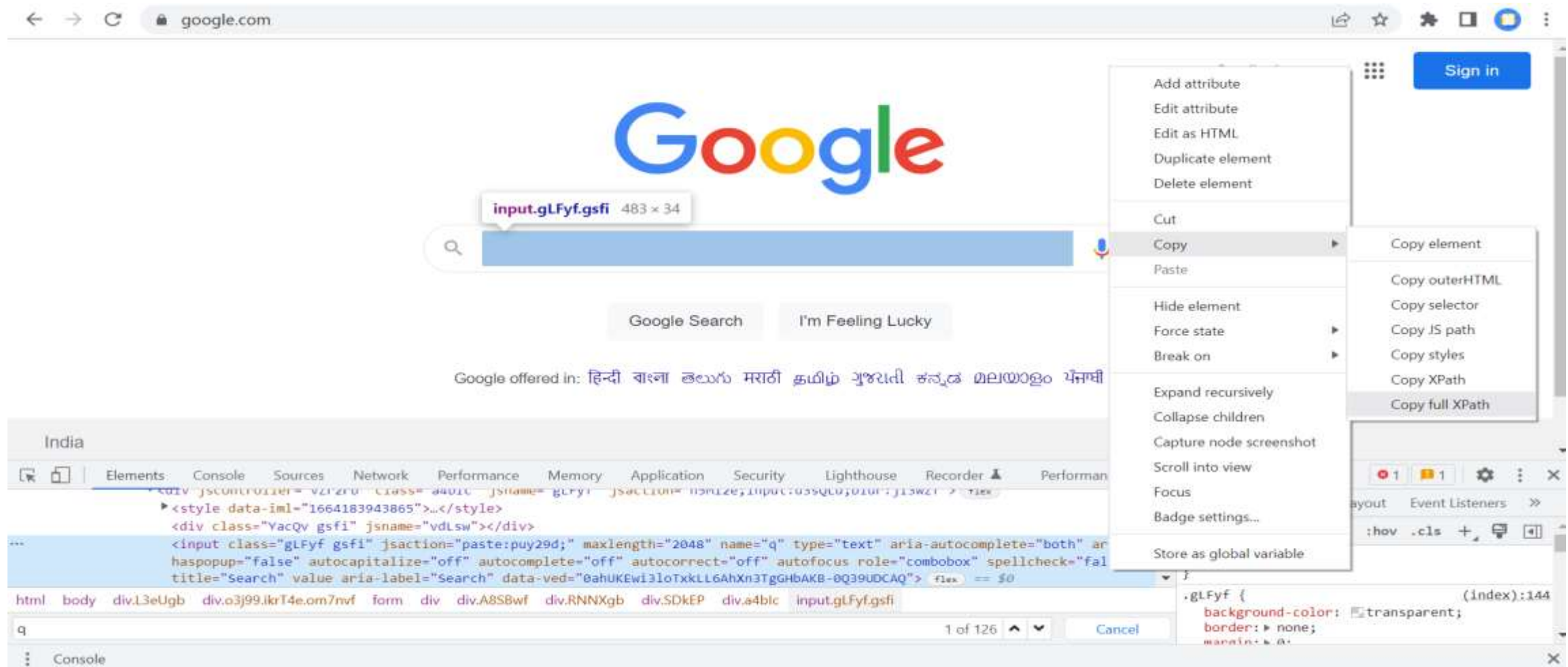
Absolute Xpath

- Absolute xpath has the complete path beginning from the root to the element which we want to identify.
- An absolute xpath starts with the / symbol.
- An absolute xpath is lengthy and difficult to maintain.
- if any changes are made in the path of the element then this XPath gets failed. So, This is the main disadvantage of absolute XPath.
- WebElement Searchbar =

```
driver.findElement(By.XPath("/html/body/div[1]/div[3]/form/div[1]/div[1]/div[1]/div/div[2]/input"));
```

Xpath

Absolute Xpath



Selenium Program – Absolute Xpath Locator

```
public class TestScript {  
    public static void main(String[] args)  
    {  
        WebDriverManager.chromedriver().setup();  
        ChromeOptions co=new ChromeOptions();  
        WebDriver driver=new ChromeDriver(co);  
        driver.get("https://www.google.com/");  
        WebElement E1 =  
        driver.findElement(By.xpath("/html/body/div[1]/div[3]/form/div[1]/div[1]/div[1]/div/div[2]/textarea"));  
        E1.click();    }  
    }  
}
```

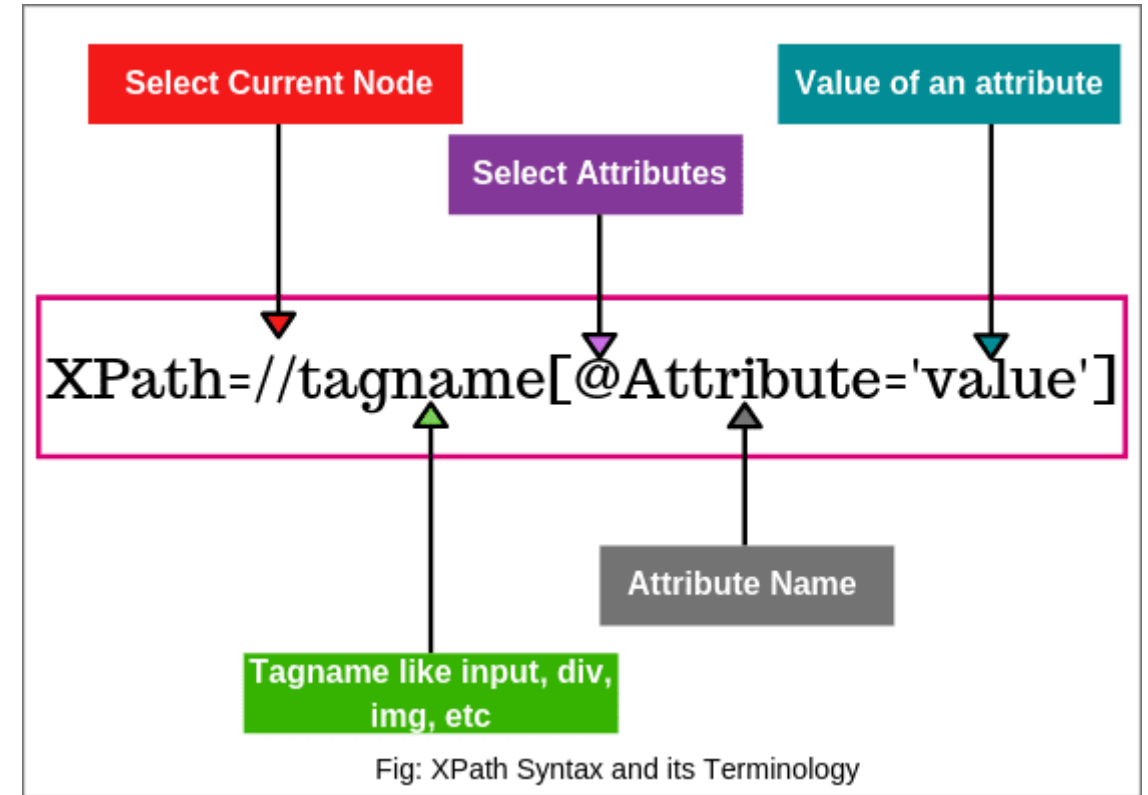
Selenium Program – Absolute Xpath Locator - Output



Xpath

Relative Xpath

- The relative xpath starts by referring to the element that we want to identify and not from the root node.
- A relative xpath starts with the “//”symbol
- Even if an element is removed or added in the DOM, the relative xpath is not impacted
- It can search elements anywhere on the webpage



Selenium Program – Relative Xpath Locator

```
public class TestScript {  
    public static void main(String[] args)  
    {  
        WebDriverManager.chromedriver().setup();  
        ChromeOptions co=new ChromeOptions();  
        WebDriver driver=new ChromeDriver(co);  
        driver.get("https://www.google.com/");  
        WebElement E1 = driver.findElement(By.xpath("//*[@id=\"APjFqb\"]"));  
        E1.click();  
    }  
}
```

Selenium Program – Absolute Xpath Locator - Output



Xpath

Xpath - Contains

- contains() is a function within Xpath expression which is used to search for the web elements that contain a particular text
- We can extract all the elements that match the given text value using the XPath contains() function throughout the webpage
- Contains in XPath has ability to find the element with partial text

Syntax :

```
//<HTML tag>[contains(@attribute_name,'attribute_value')]
```

Eg:

```
driver.get("https://www.google.com/");
```

```
WebElement E1 = driver.findElement(By.xpath("//textarea[contains(@name,'q')]"));
```

```
E1.click();
```

Xpath

XPath - Starts-with

- starts-with() method is used when we know about the initial partial attribute value or initial partial text associated with the web element.
- User can also use this method to locate web elements those consists of both static(initial) and dynamic(trailing) values.

Syntax:

XPath: //tagname[starts-with(@attribute, 'value')]

Example:

- //input[starts-with(@placeholder, 'Organization')]
- //input[starts-with(@name, 'organization')]

Xpath

XPath - Ends-with

- ends-with() method checks the ending text of an attribute and finds elements whose attribute changes dynamically on refresh.

Syntax:

XPath: //tagname[ends-with(@attribute, 'value')]

Example:

```
//input[ends-with(@placeholder, 'Organization')]
```

```
//input[ends-with(@name, 'organization')]
```

Xpath

XPath - “OR” Statement

- In OR expression, two conditions are used, whether 1st condition OR 2nd condition should be true.
- It is also applicable if any one of the condition is true or maybe both.
- Means any one condition should be true to find the element.

Syntax:

XPath: //tagname[XPath statement-1 or XPath statement-2]

Example:

```
//input[@value = 'Log In' or @type = 'submit']
```

Xpath

XPath - “AND” Statement

- In AND expression, two conditions are used, both conditions should be true to find the element
- It fails to find the element if any one condition is false.

Syntax:

XPath: //tagname[XPath statement-1 and XPath statement-2]

Example:

```
driver.get("https://www.google.com/");  
WebElement E1 = driver.findElement(By.xpath("//*[@name='q' and @class='gLFyf']"));  
E1.click();
```

Xpath

XPath – Parent

- Parent in Selenium is a method used to retrieve the parent node of the current node selected in the web page
- It is very useful in the situation when you select an element and need to get the parent element using Xpath.

Example:

```
Xpath=//*[@id='rt-feature']//parent::div
```

```
Xpath=//*[@title="50"]//parent::store
```

Xpath

Xpath – Ancestor

- The ancestor axis selects all ancestor's element (parent, grandparent, great-grandparents, etc.) of the current node.
- This axis always contains the root node (unless the current node is the root node).

Syntax:

```
//ancestor::tagName
```


Xpath – Ancestor (Contd.)

Example:

- Let us consider the login button as current node. First, find the Xpath of current node.

XPath(Current node): //input[@id = 'u_0_a']

- Now, we will find XPath of parent and grandparent of current node.

XPath(Parent node): //input[@id = 'u_0_a']//ancestor::label

XPath(Grandparent node): //input[@id = 'u_0_a']//ancestor::td

- In both cases, 1 of 1 node is matched by using “ancestor” axis.

XPath – Sibling

- A Sibling in Selenium Webdriver is a function used to fetch a web element which is a sibling to the parent element.
- If the parent element is known then the web element can be easily found or located that can use the sibling attribute of the Xpath expression in selenium webdriver

Example:

- 1) `//div[@class='canvas- graph'] //a[@href ='/ accounting.html'] [i[@class='icon-
usd']] /following-sibling::h4 Xpath=//*[title="50"]/parent::store`
- 2) `//input[@id = 'u_0_5'] /following-sibling::label`

XPath – Sibling

Example:

```
<ul class="right"> <li><a href="https://www.educba.com/feature">Home</a></li>  
<li><a href="https://www.educba.com/Blogs">Automation</a></li>  
<li><a href="https://www.educba.com/product">Product</a></li>  
<li><a href="https://www.educba.com/pricing">Pricing</a></li>  
<li><a href="https://www.educba.com/support/">Support</a></li>  
<li class="sign-out"><a href="https://accounts.educba.com/login">Login</a></li>  
<li class="login"><a href="https://accounts.educba.com/login">Free Sign in</a> </li>  
  
//li[@class='sign-out']//following-sibling::li"  
  
//li[@class='sign-out']//preceding-sibling::li"
```

Xpath

Advantages of Xpath

- Queries are easy to type and read.
- Query strings are easily embedded in programs, scripts, and XML or HTML attributes.
- Queries are easily parsed.
- You can specify any path that can occur in an XML document and any set of conditions for the nodes in the path.
- You can uniquely identify any node in an XML document
- Queries return any number of results, including zero.
- Query conditions can be evaluated at any level of a document and are not expected to navigate from the top node of a document.
- Queries do not return repeated nodes

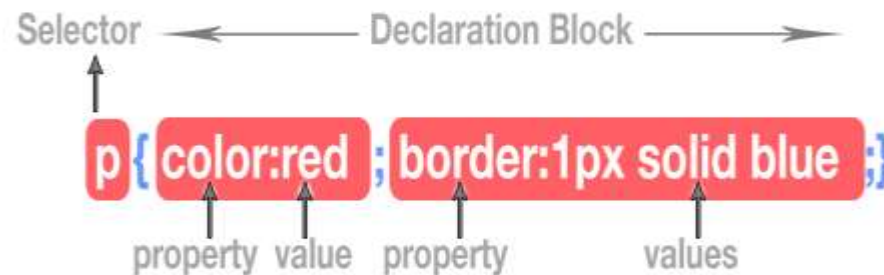
Xpath

Advantages of Xpath (Contd.)

- For programmers, queries are declarative, not procedural. They say what should be found, not how it should be found. This is important because a query optimizer must be free to use indexes or other structures to find results efficiently.
- XPath is designed to be used in many contexts. It is applicable to providing links to nodes, for searching repositories, and for many other applications

CSS Selector

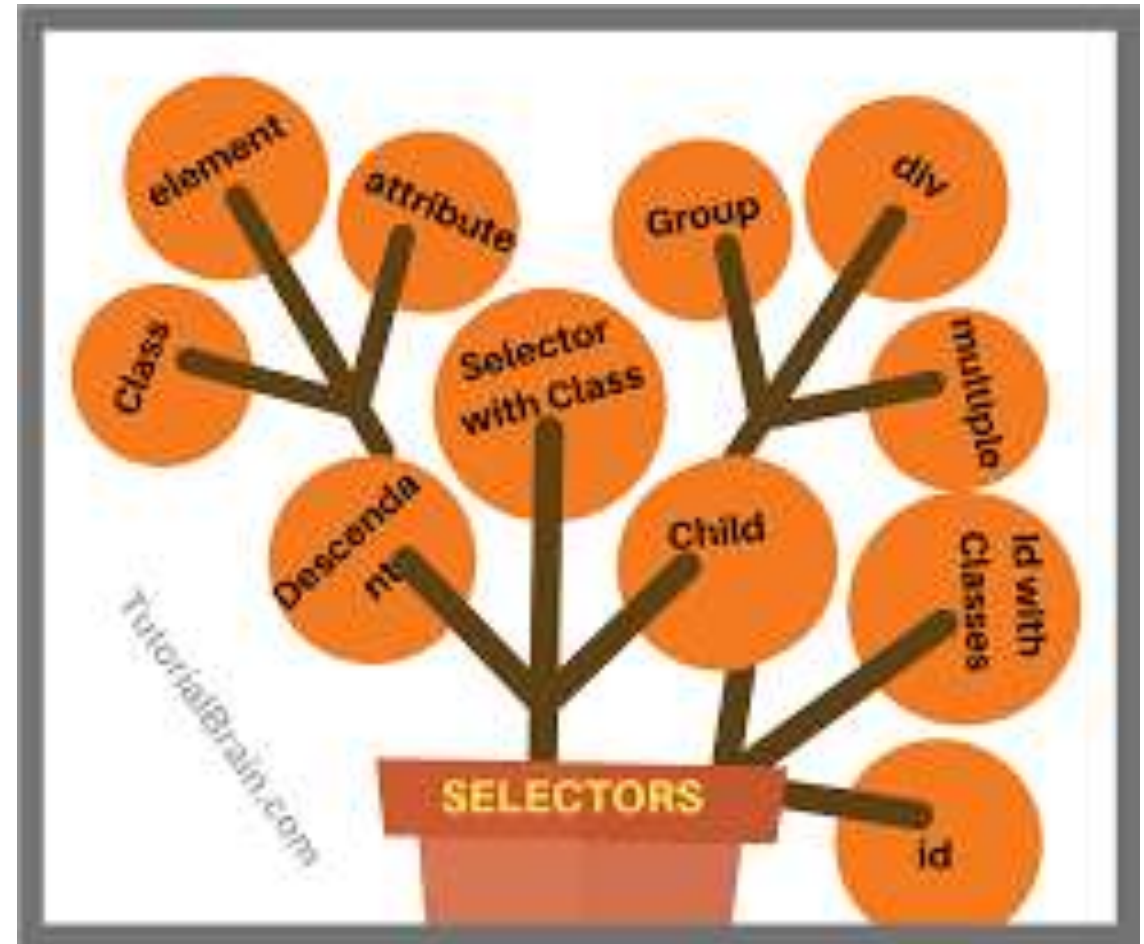
- CSS Selectors in Selenium are string patterns used to identify an element based on a combination of HTML tag, id, class, and attributes. CSS Selectors in Selenium have many formats, but we will only focus on the most common ones.
- `css = element_name[<attribute_name>=<value>']`
- `WebElement firstName = driver.findElement(By.cssSelector("input[name='first_name']"));`



CSS Selector

CSS Selector (Contd.)

- Tag and ID
- Tag and class
- Tag and attribute
- Tag, class, and attribute
- Inner text



CSS Selector

CSS Selector (Contd.)

- Tag and ID
 - To locate elements by Tag and ID, you have to use the following components:
 - Html tag: It provides the tag we wish to locate (e.g. input tag).
 - #: It is used to represent the ID attribute.
 - Keep in mind that when you wish to locate an element via ID through a CSS selector, it must have a hash sign on the same. For other attributes, we need not use the hash sign.
 - Value of the ID attribute: This represents the value of the ID we are using to locate the element.
 - Syntax: `css=(Html tag)(# (value of the ID attribute))`
 - Example: `driver.findElement(By.cssSelector("input#userpassword"))`

CSS Selector

CSS Selector (Contd.)

- Tag and Class

- Syntax:

`css=(HTML tag)(.)(Value of Class attribute)`

- Example:

- `driver.findElement(By.cssSelector("button.submit-btn"))`

CSS Selector

CSS Selector (Contd.)

- Tag, Class and Attribute

- Syntax:

css=(HTML tag>)(.)(Class attribute value)([attribute=Value of attribute])

- Example: `driver.findElement(By.cssSelector("button.submit-btn[data-callback=\"onSubmit\"]"));`
- Here is the DOM structure:

```
<button data-sitekey="6LceAqQaAAAAAO0LclgLnXy3gH_M3X5aDrqUihHw" data-callback="onSubmit" data-amplitude="R_signup" type="submit" class="btn btn-dark submit-btn g-recaptcha" css="1">Free Sign Up</button>
```

CSS Selector

Wild (*, ^ and \$) in CSS for classes

- Selenium CSS selector in Selenium helps in matching multiple strings through the use of multiple patterns like ^, \$, *.
- Wildcard selectors in CSS are used for selecting multiple elements simultaneously.

a. Starts-With in CSS Selector

- The Starts-With helps locate elements when we try to match elements with a string that starts with a designated value.
- **Syntax** `css=(HTML tag)([attribute^=start of the string])`

CSS Selector

Wild (*, ^ and \$) in CSS for classes

Example

- Here is the DOM structure for locating the WebElement:

```
<input type="email" name="email" value="" placeholder="Email" required="required"
      autofocus="autofocus" class="form-control mt-3 form-control-lg">
```

- Here how the CSS [attribute^=value] Selector is used for locating the desired WebElement:

```
driver.findElement(By.cssSelector("input[name^='em']"));
```

CSS Selector

Wild (*, ^ and \$) in CSS for classes

b. Ends-With in CSS Selector

- This helps locate elements when we try to match elements with a string that ends with a designated value.

- **Syntax** `css=(HTML tag)([attribute$=end of the string])`

- **Example**

- Here is the DOM structure of the element:

```
<input type="email" name="email" value="" placeholder="Email" required="required"
autofocus="autofocus" class="form-control mt-3 form-control-lg">
```

- Here is how Ends-With in CSS Selector is used for locating the required WebElement:

```
driver.findElement(By.cssSelector("input[name$='ail']"));
```

CSS Selector

Wild (*, ^ and \$) in CSS for classes

c. Contains in CSS Selector

- This helps locate elements when we try to match elements with a string that containing a designated value.

- **Syntax** `css=(HTML tag)([attribute*=partial string])`

- **Example**

- Here is the DOM structure of the element:

```
<input type="email" name="email" value="" placeholder="Email" required="required"
autofocus="autofocus" class="form-control mt-3 form-control-lg">
```

- Here is how Contains in CSS Selector is used for locating the required WebElement:

```
driver.findElement(By.cssSelector("input[class*='control']"));
```

CSS Selector

Advantages of CSS Selector

- It's faster than XPath.
- It's much easier to learn and implement
- You have a high chance of finding your elements.
- It's compatible with most browsers to date.

Disadvantages of CSS Selector

- The browser does not support CSS or the CSS selectors you need.
- Support one selector type.
 - 1) Traversing the DOM is not possible with CSS as that of the Xpath.
 - 2) Browser Compatibility issue. We have to determine which style is supported and which is not.
 - 3) Cross browser issue. Since some selectors behave differently in different browsers.

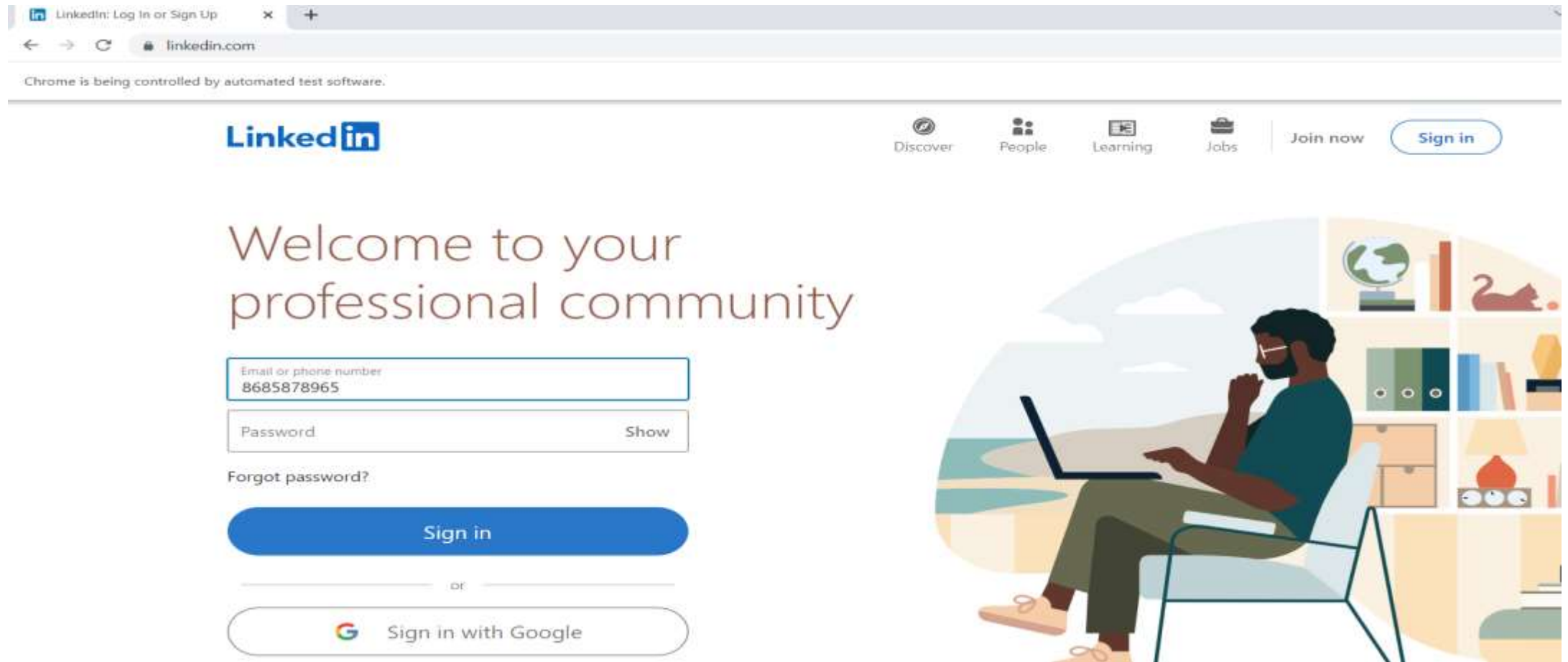
CSS Selector

Example Program 1

```
public class TestScript {  
    public static void main(String[] args)  
    {  
        System.setProperty("webdriver.chrome.driver", "C:\\workspace\\ChromeDriver\\chromedriver.exe");  
        WebDriver driver = new ChromeDriver();  
        driver.get("https://www.linkedin.com");  
        WebElement ele = driver.findElement(By.cssSelector("input#session_key"));  
        ele.sendKeys("8685878965");  
        String str = ele.getAttribute("value");  
        System.out.println("Attribute value: " + str);  
        driver.quit();  
    }  
}
```

CSS Selector

Output



CSS Selector

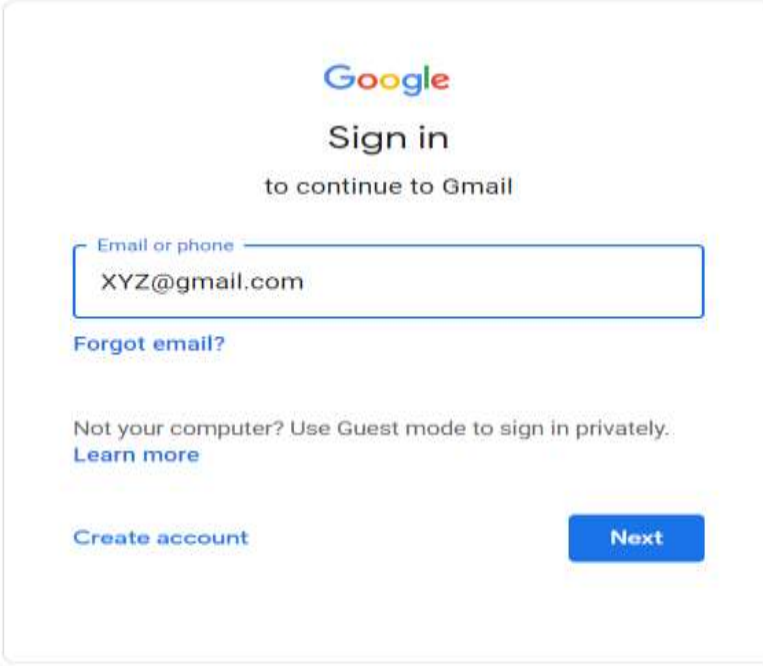
Example Program 2

```
public class TestScript {  
    public static void main(String[] args)  
    {  
        System.setProperty("webdriver.chrome.driver", "C:\\workspace\\ChromeDriver\\chromedriver.exe");  
        WebDriver driver = new ChromeDriver();  
        driver.get("https://www.gmail.com");  
        WebElement ele = driver.findElement(By.CssSelector("input[id='yDmH0d']"));  
        ele.SendKeys("XYZ@gmail.com");  
    }  
}
```

CSS Selector

Output

Chrome is being controlled by automated test software.



The screenshot shows the Google Sign-in interface. At the top is the Google logo, followed by the text "Sign in to continue to Gmail". Below this is a text input field with the placeholder "Email or phone" and the value "XYZ@gmail.com". To the left of the input field is a blue link "Forgot email?". Below the input field is the text "Not your computer? Use Guest mode to sign in privately." followed by a blue link "Learn more". At the bottom left is a blue link "Create account", and at the bottom right is a blue button labeled "Next". At the very bottom of the page, there is a footer with "English (United Kingdom)" and a dropdown arrow, followed by links for "Help", "Privacy", and "Terms".

Google

Sign in
to continue to Gmail

Email or phone

XYZ@gmail.com

[Forgot email?](#)

Not your computer? Use Guest mode to sign in privately.
[Learn more](#)

[Create account](#) [Next](#)

English (United Kingdom) ▼ [Help](#) [Privacy](#) [Terms](#)

CSS Selector

Practice

- 1) Perform Automation Testing for <https://my.indiamart.com/> application using Java Selenium.
 - i)Launch the application in chrome browser
 - ii)Select the wheat Flour link
 - iii)Select Wheat grain
 - iv)Move to link Chennai and Select
 - v)Select and click the first product in resulting page
 - vi)Switch to the child window
 - vii)Get the latest price.
 - viii)Switch to parent window

Practice

- 1) Perform Automation Testing for <https://www.woodenstreet.com/> application using Java Selenium.
 - i)Launch the application in chrome browser
 - ii)Select the living storage link
 - iii)Select the TV Unit link
 - iv)Apply the filters
 - Price Under 9999
 - Honey Finish in Finish
 - v)Select the first option from the link
 - vi)Add it to the cart
 - vii)Move to the homepage
 - viii)Select the Study & Office
 - ix)Move to Office furniture
 - x)Select Workstation
 - xi) Apply Sort by Latest filter and with Storage in Storage.
 - xii)Add it to the cart.

THANK YOU