**Locators:**

Locators are static methods of ‘By’ class. ‘By’ is an abstract class in org.openqa.selenium package.

Why locators?

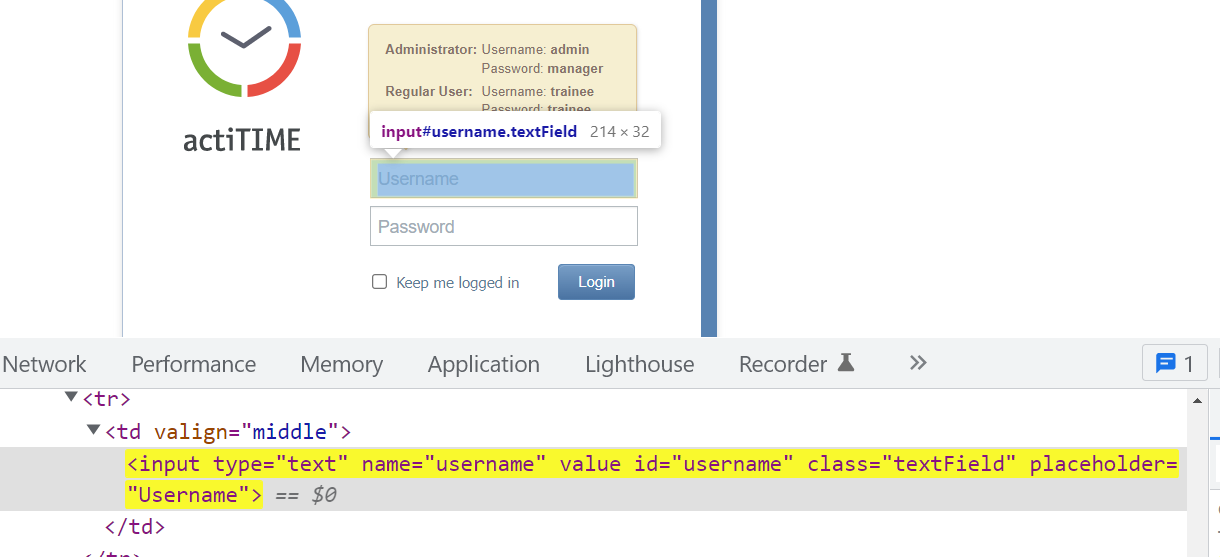
In order to perform any operations on specific web element we should first locate the element on the web page. Hence in order to locate the element on web page we go for locators.

Types of locators:

1. id locator: If we have an attribute as ‘id’ in the element node in HTML tree structure, we go for id locator.

Usage: driver.findElement(By.id(“id\_attribute\_value”));

Ex:

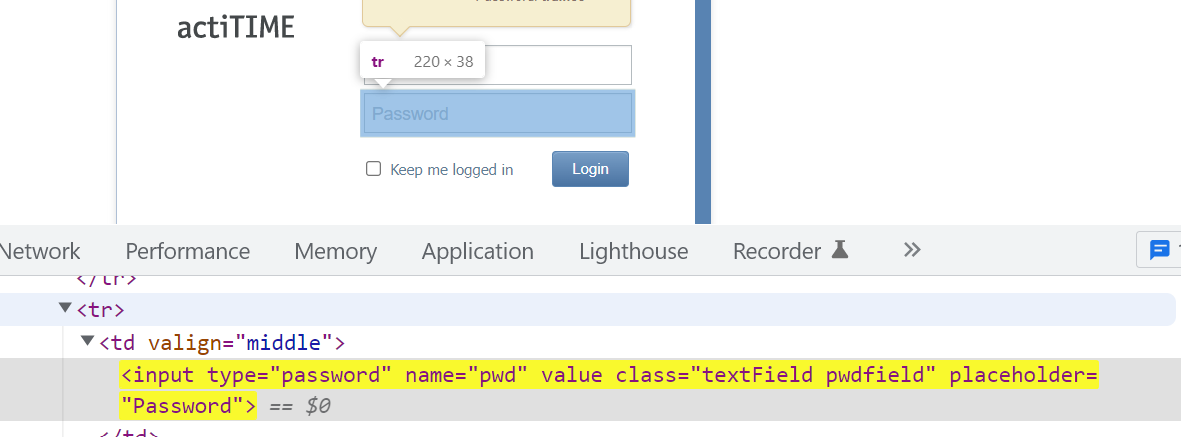


WebElement usernameTF = driver.findElement(By.id(“username”));

1. name locator: If we have an attribute as ‘name’ in the element node in HTML tree structure, we go for name locator.

Usage: driver.findElement(By.name(“name\_attribute\_value”));

Ex:



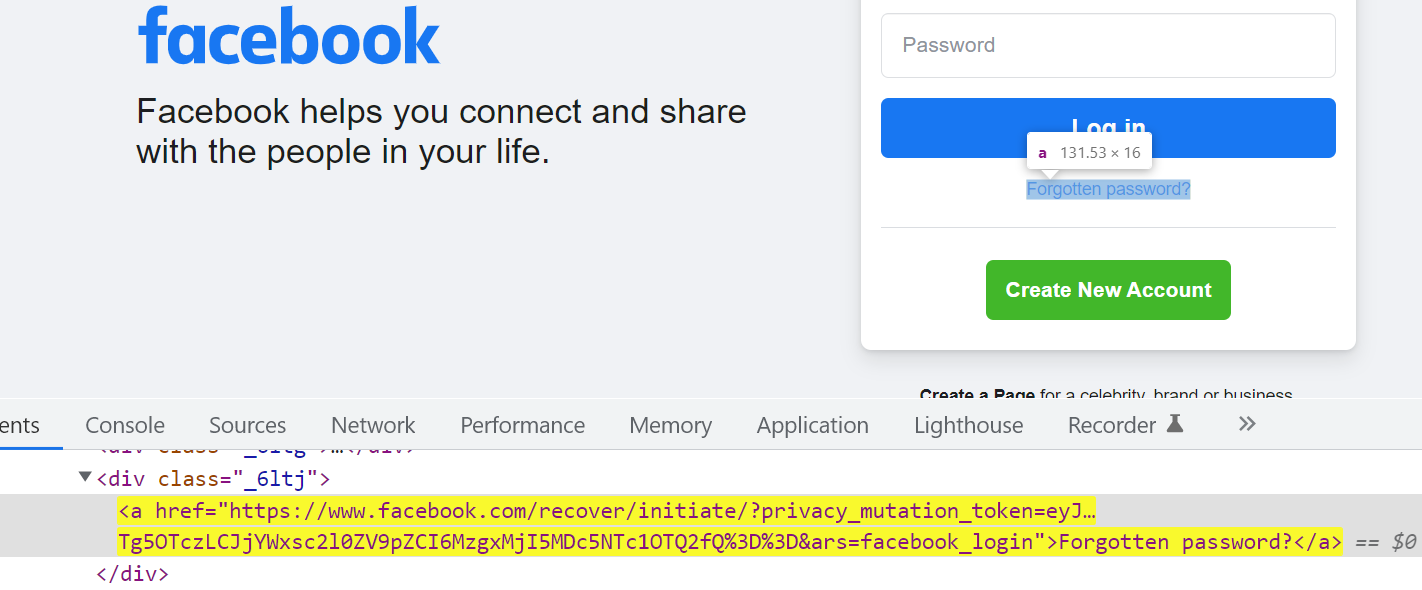
WebElement passwordTF = driver.findElement(By.name(“pwd”));

1. linkText locator: If the element is a link and the element node is having text on it then we go for linkText locator.

Usage: driver.findElement(By.linkText(“text\_on\_the\_link”));

Note: Links have the tag name <a>

Ex:



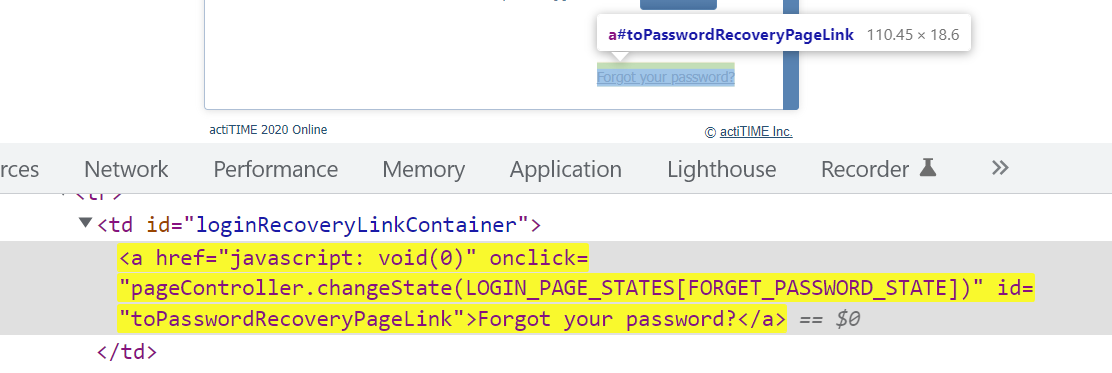
WebElement link = driver.findElement(By.linkText(“Forgotten

password?”));

1. partialLinkText locator: If the element is a link and the element node in HTML has lengthy text on it then we for partialLinkText locator.

Usage: driver.findElement(By.partialLinkText(“partial\_text\_on\_the\_link”));

Ex:

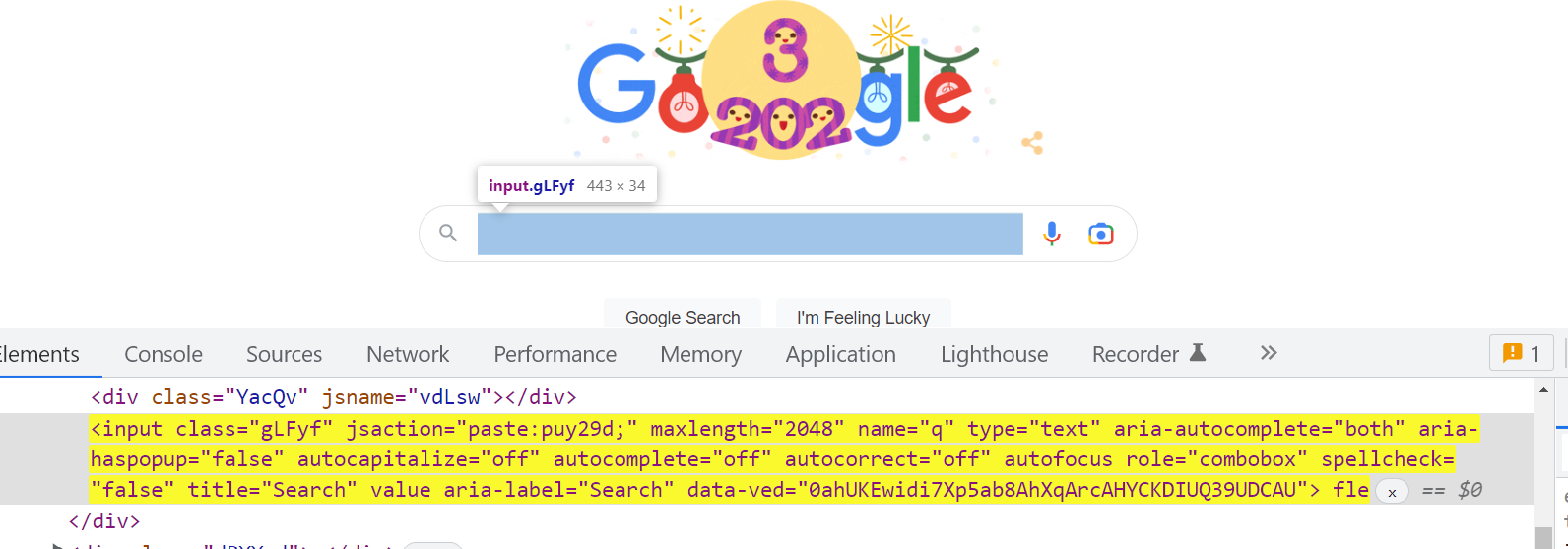


WebElement link = driver.findElement(By.partialLinkText(“Forgot”));

1. className locator: If the element node in HTML has an attribute as ‘class’ then we go for className locator.

Usage: driver.findElement(By.className(“class\_attribute\_value”));

Ex:



WebElement searchBar = driver.findElement(By.className(“gLFyf”));

1. tagName locator: Identifying an element or elements using tag name of the element node in HTML can be done using tagName locator.

Usage: driver.findElement(By.tagName(“tag\_name\_of\_element”));

Ex:

WebElement textbox = driver.findElement(By.tagName(“input”));

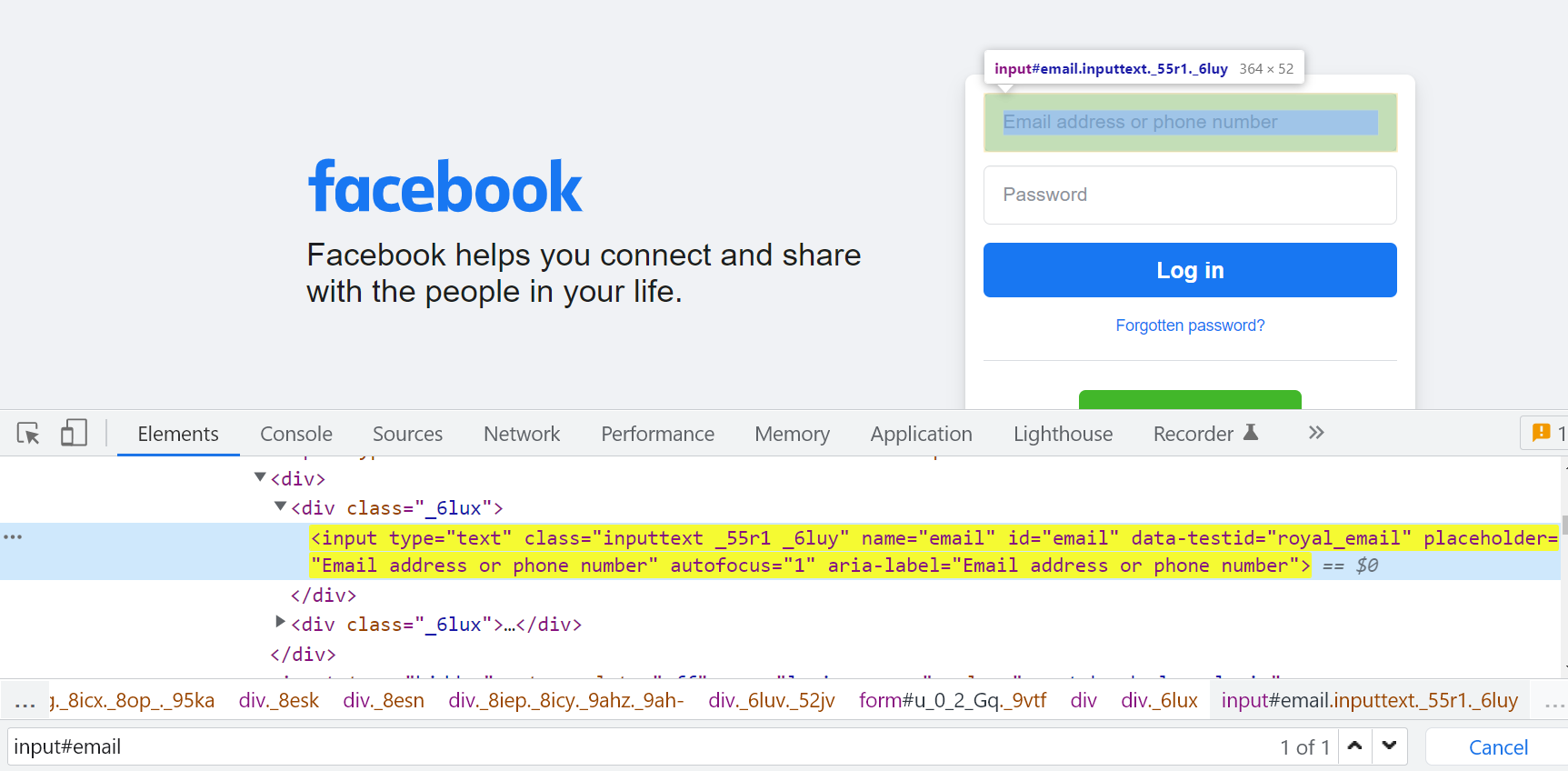
Note: There can be multiple elements having same tag name and class name in the HTML tree structure.

1. cssSelector locator: We use cssSelector locator if we have attributes with unique attribute values in the element node in HTML.

Syntaxes:

* For id attribute : tagname#id\_attribute\_value

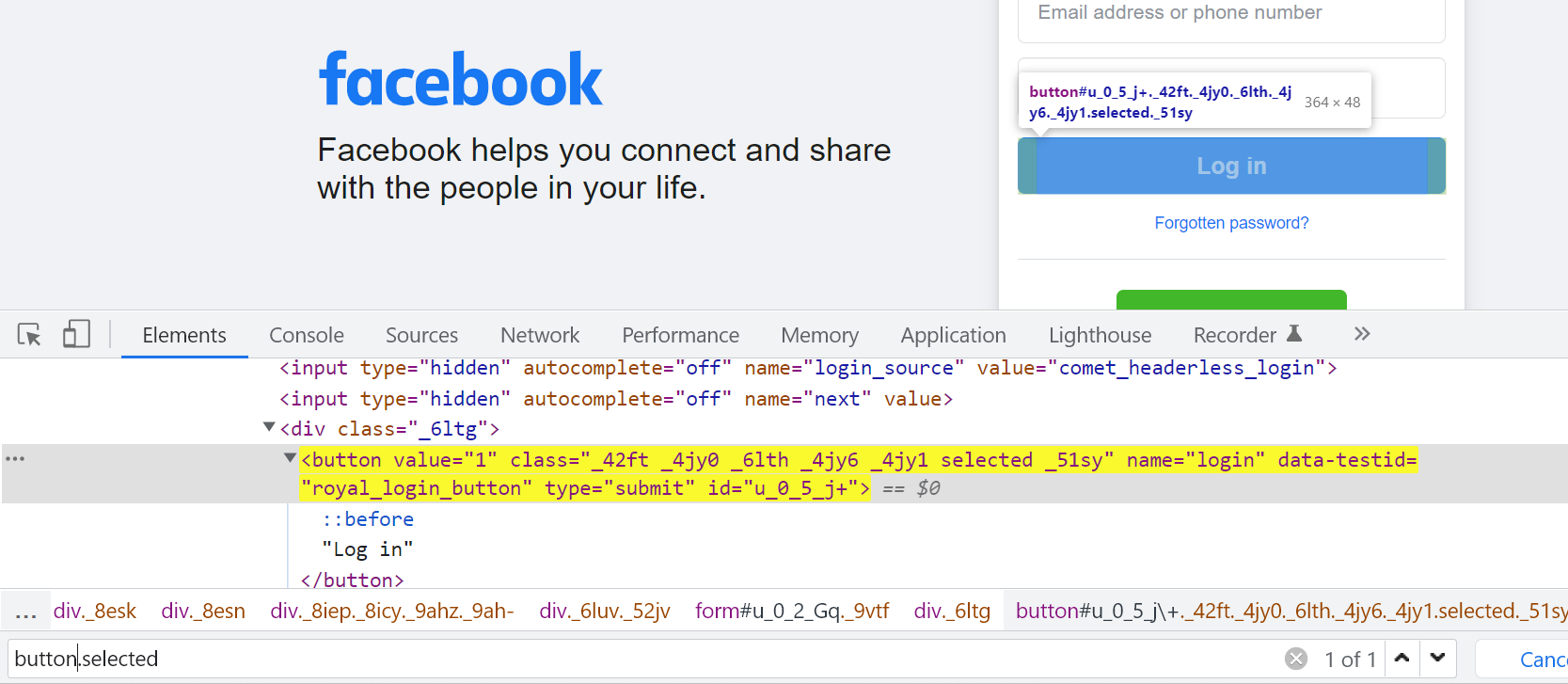
Ex:



input#email

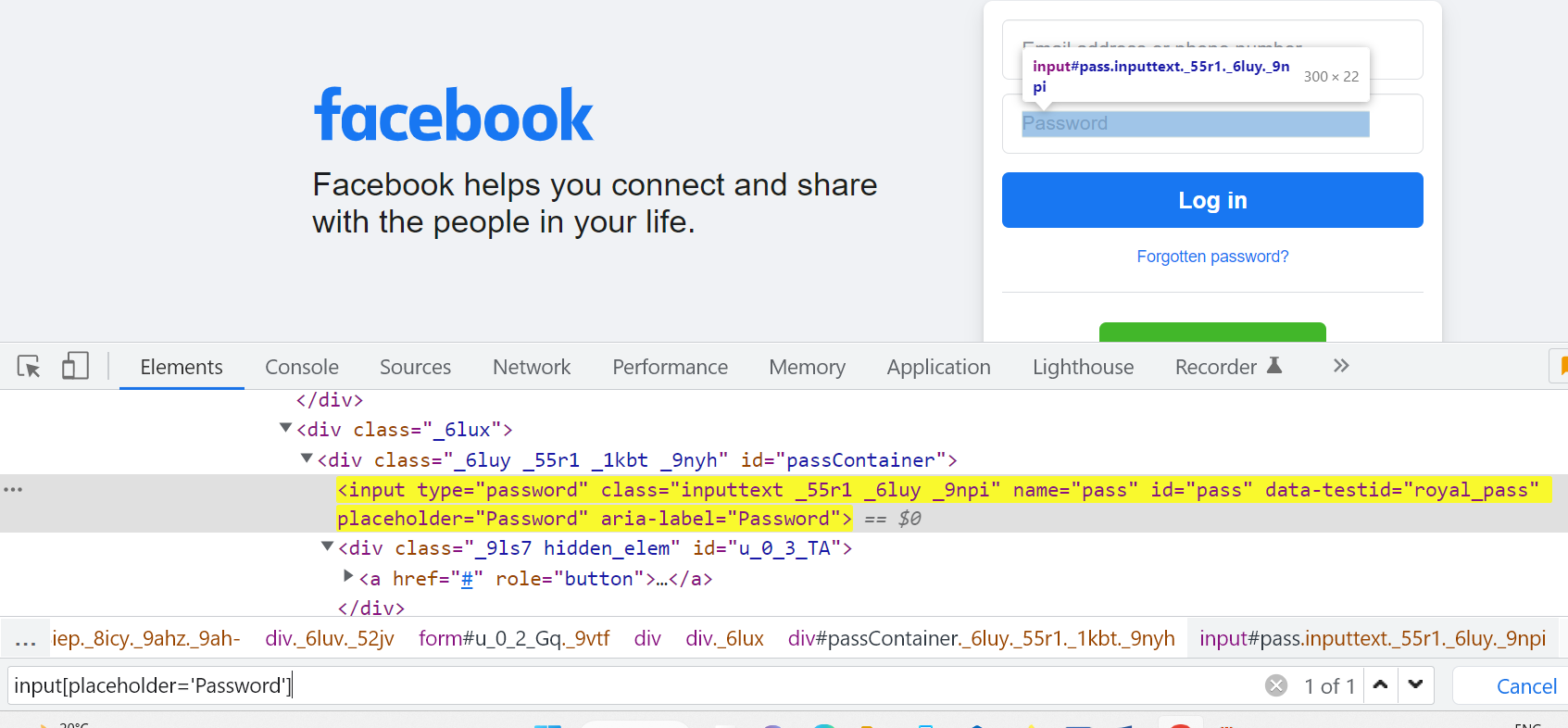
* For class attribute : tagname.class\_attribute\_value

Ex:

button.selected

* Generic syntax : tagname[attribute\_name=’attribute\_value’]

Ex:



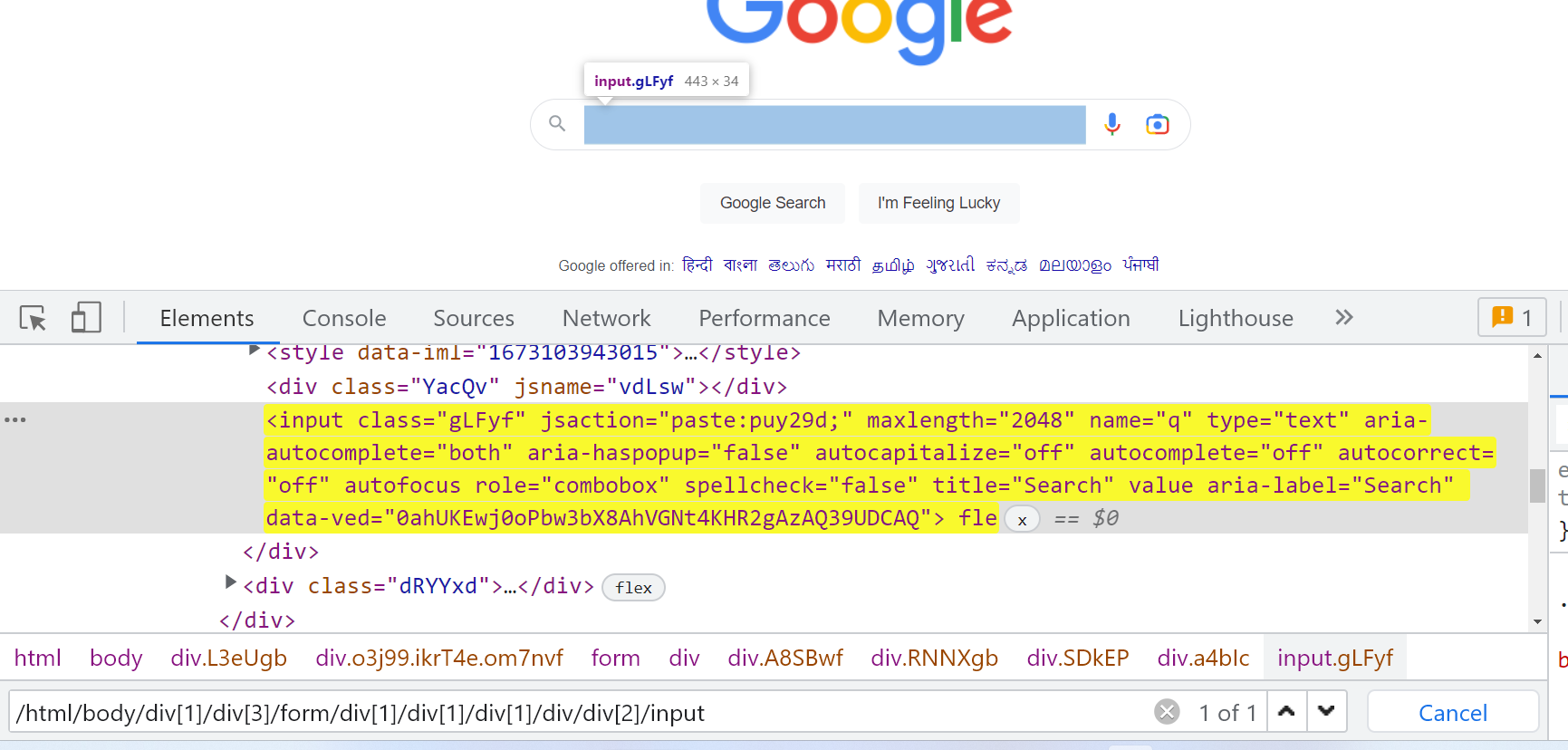
input[placeholder=’Password’]

1. xpath locator: It is one of the locators in WebDriver tool which is used to navigate entire html document & identify the element based on the web element’s attribute or visible text(inner text).

We have two types in xpath, namely,

1. Absolute xpath: The complete path from the root of html tree structure till the element is called absolute path. We use ‘/’ for traversing.

Ex:



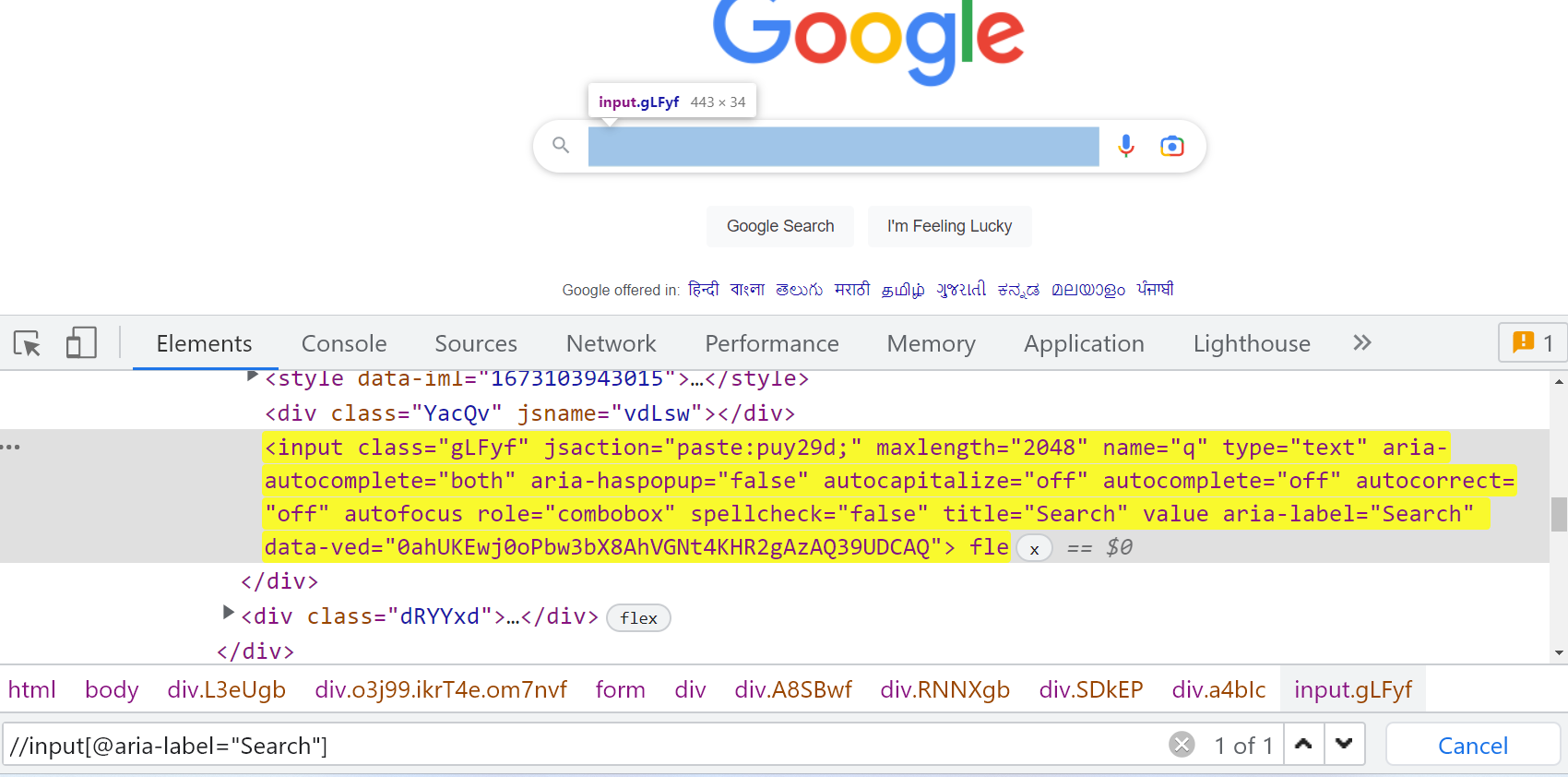
/html/body/div[1]/div[3]/form/div[1]/div[1]/div[1]/div/div[2]/input

Drawbacks:

* Absolute xpath can be lengthy when we have more elements in the web page.
* Traversing from root node to the element node is tedious.
* In Agile, due to frequent requirement changes GUI also changes which might have effect on absolute xpath. Hence absolute xpath is not used in real time.

1. Relative xpath: The path written directly to the web element using attribute or visible text on the element node is called relative xpath.

We use ‘//’ for traversing.

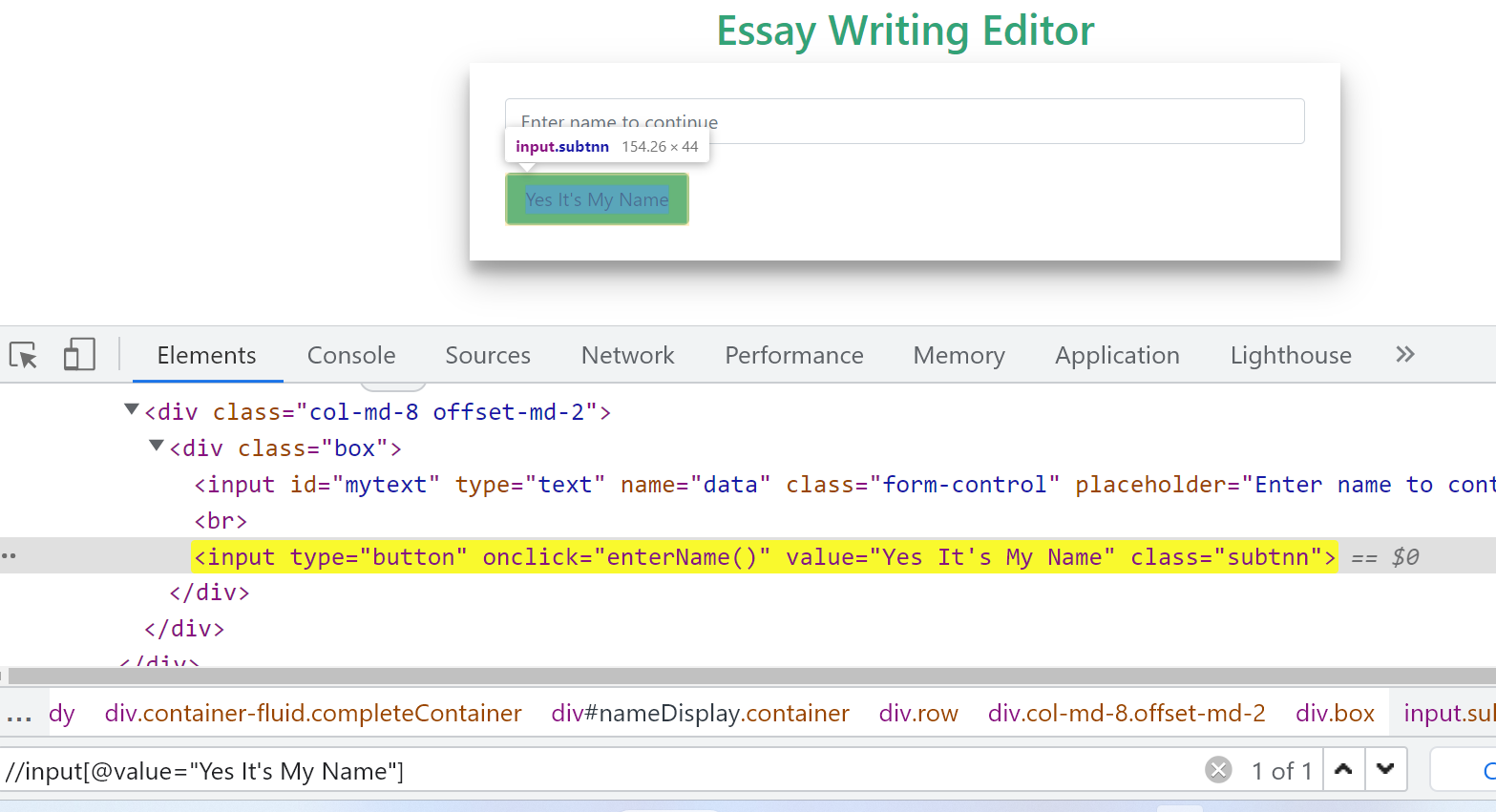
Ex: 

//input[@aria-label=’Search’]

Relative xpath is categorized in to two types:

1. Basic Relative xpath
2. Advanced Relative xpath
3. Basic Relative xpath: It is categorized in to three types
4. Xpath by attributes: Xpath is written directly to the element using unique attribute.

Syntax: //tagname[@attributeName=’attributeValue’]

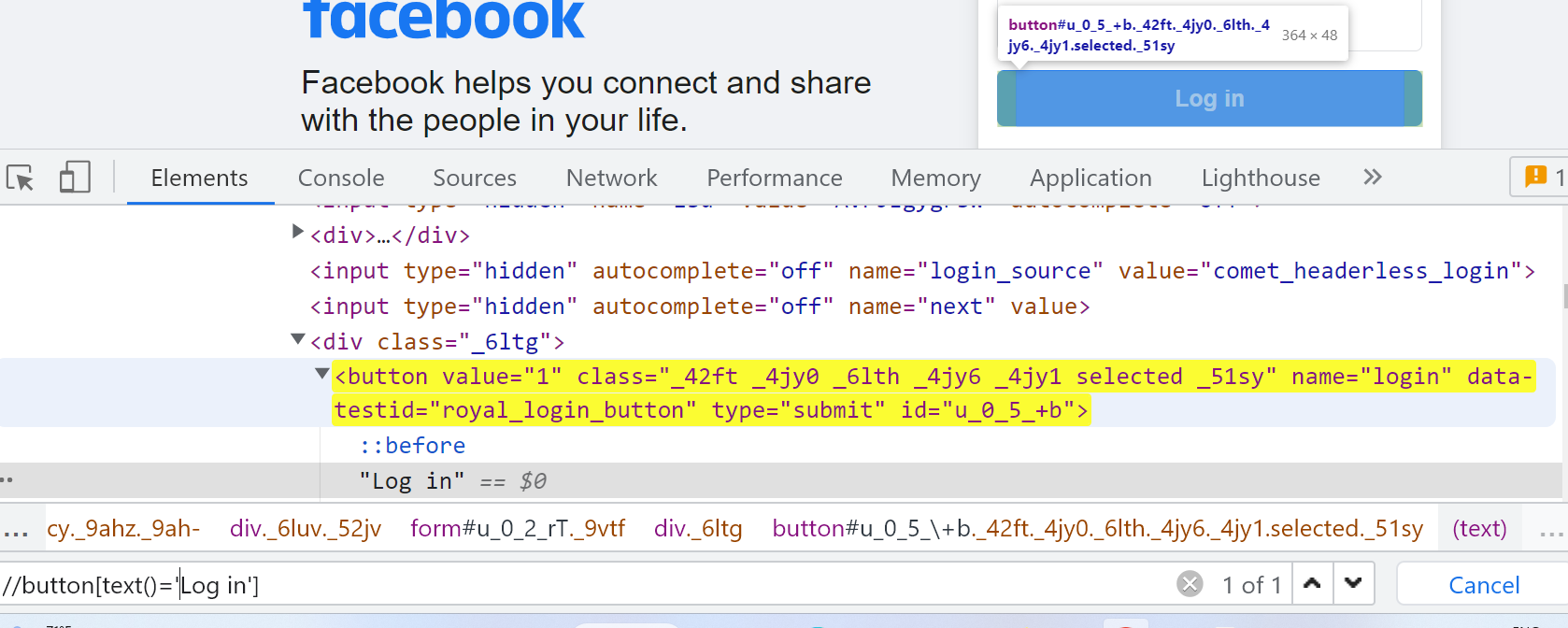
Ex: 

//input[@value=”Yes It’s My Name”]

1. Xpath by text(): Xpath is written directly to the element using the visible text or inner text on the element node.

Syntax: //tagname[text()=’text\_value’]

or //tagname[.=’text\_value’]

Ex: 

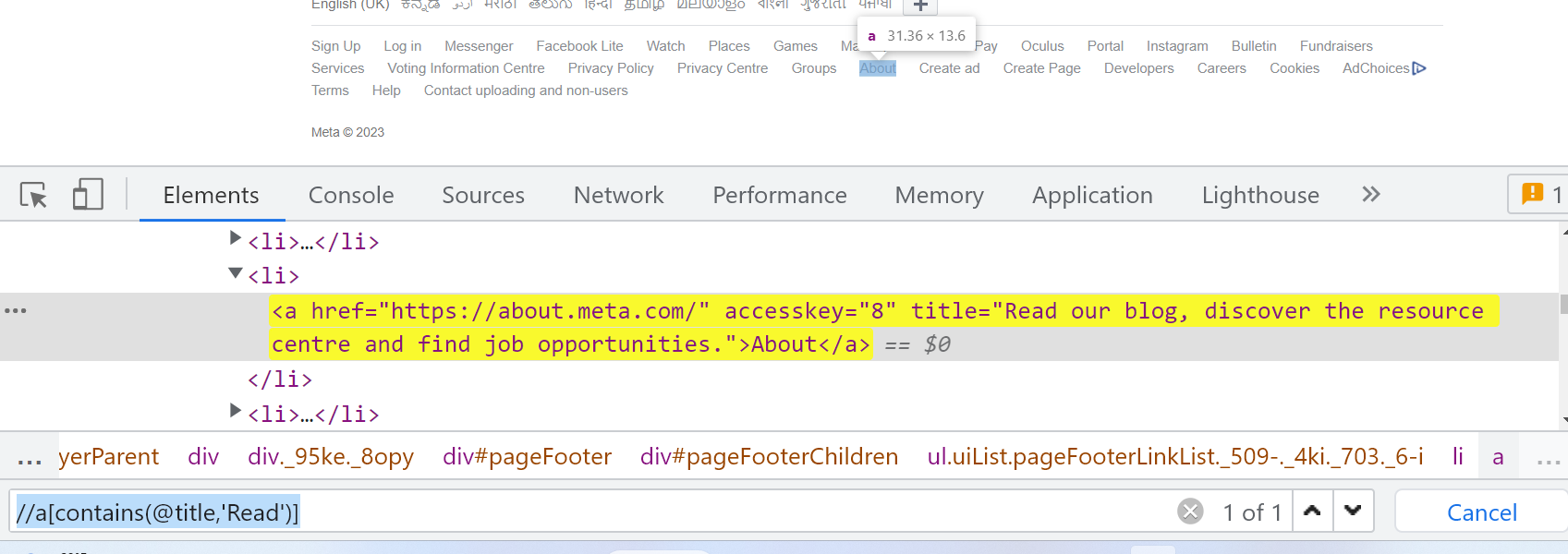
//button[text()=’Log in’]

1. Xpath by contains(): Xpath is written directly to the element using using partial visible text or partial attribute value on the element node.

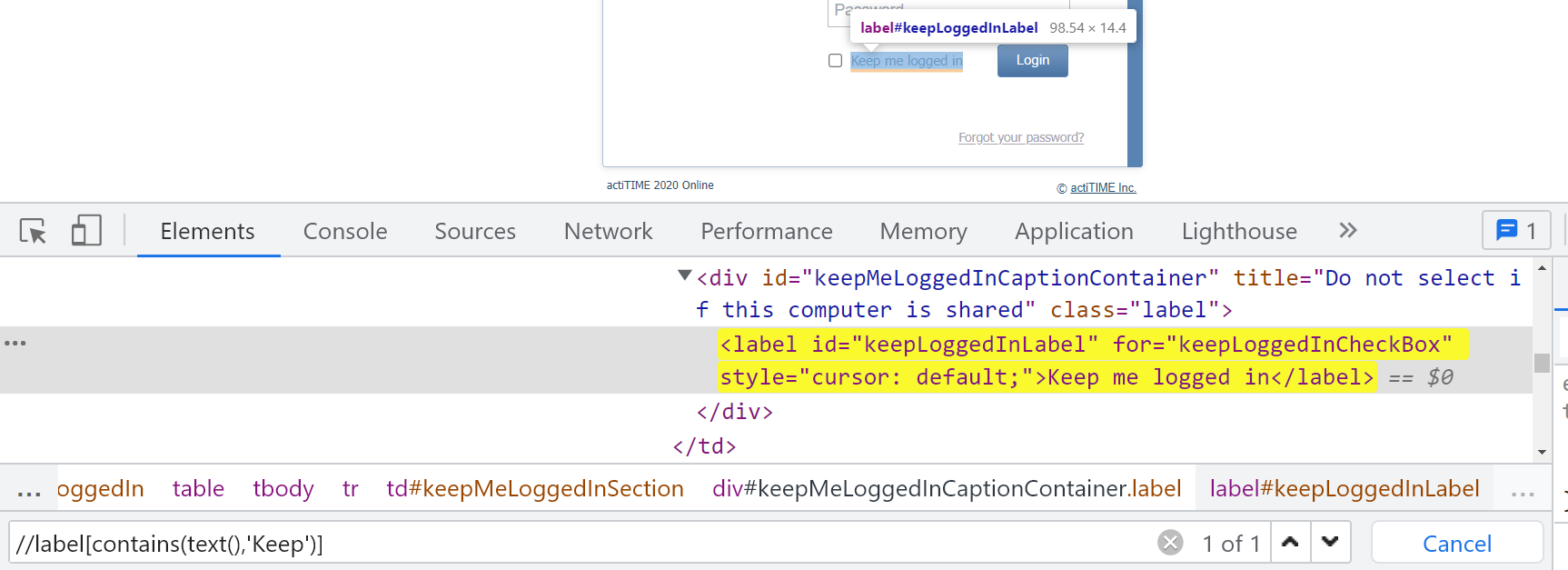
Syntax: //tagname[contains(@attributeName,’attributeValue’)]

//tagname[contains(text(),’textValue’)]

* contains() handles lengthy attribute values and texts

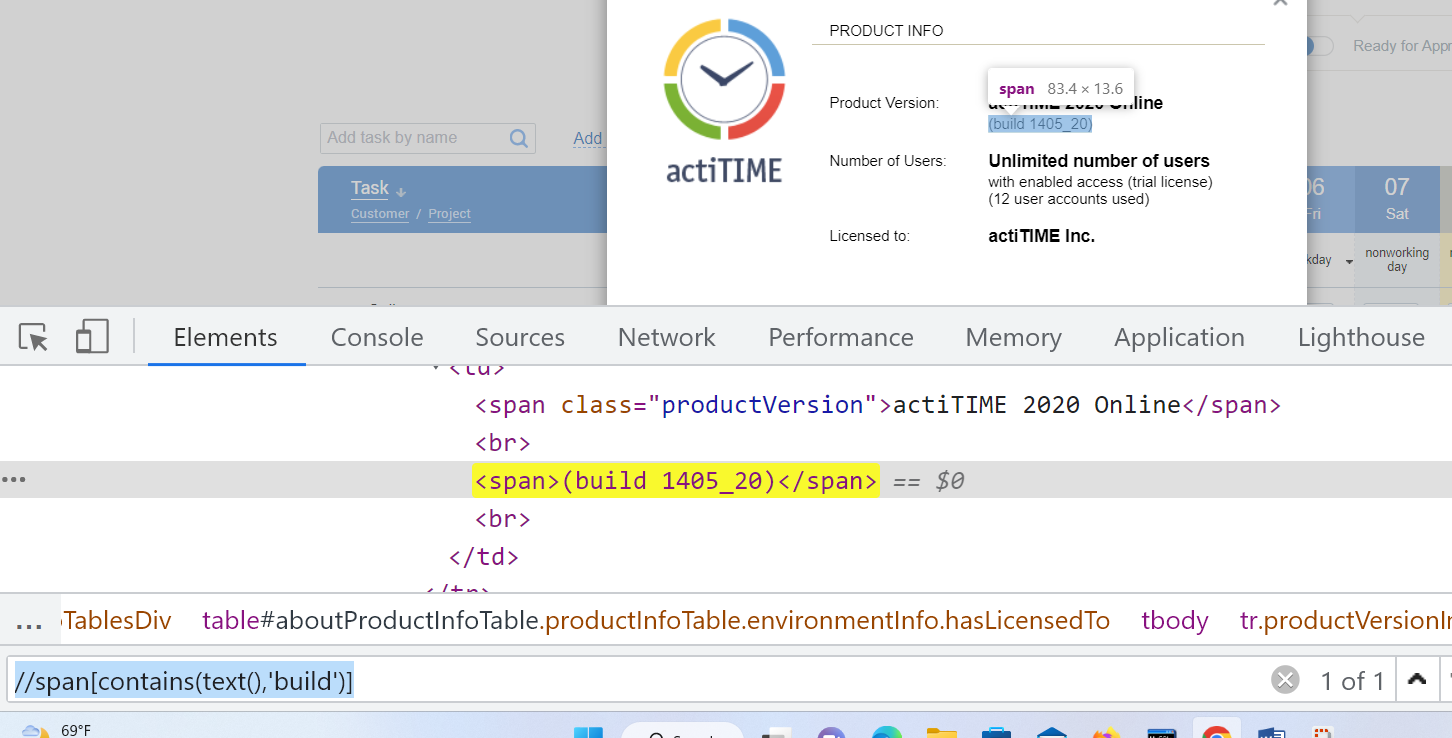
Ex: 

//a[contains(@title,’Read’)]



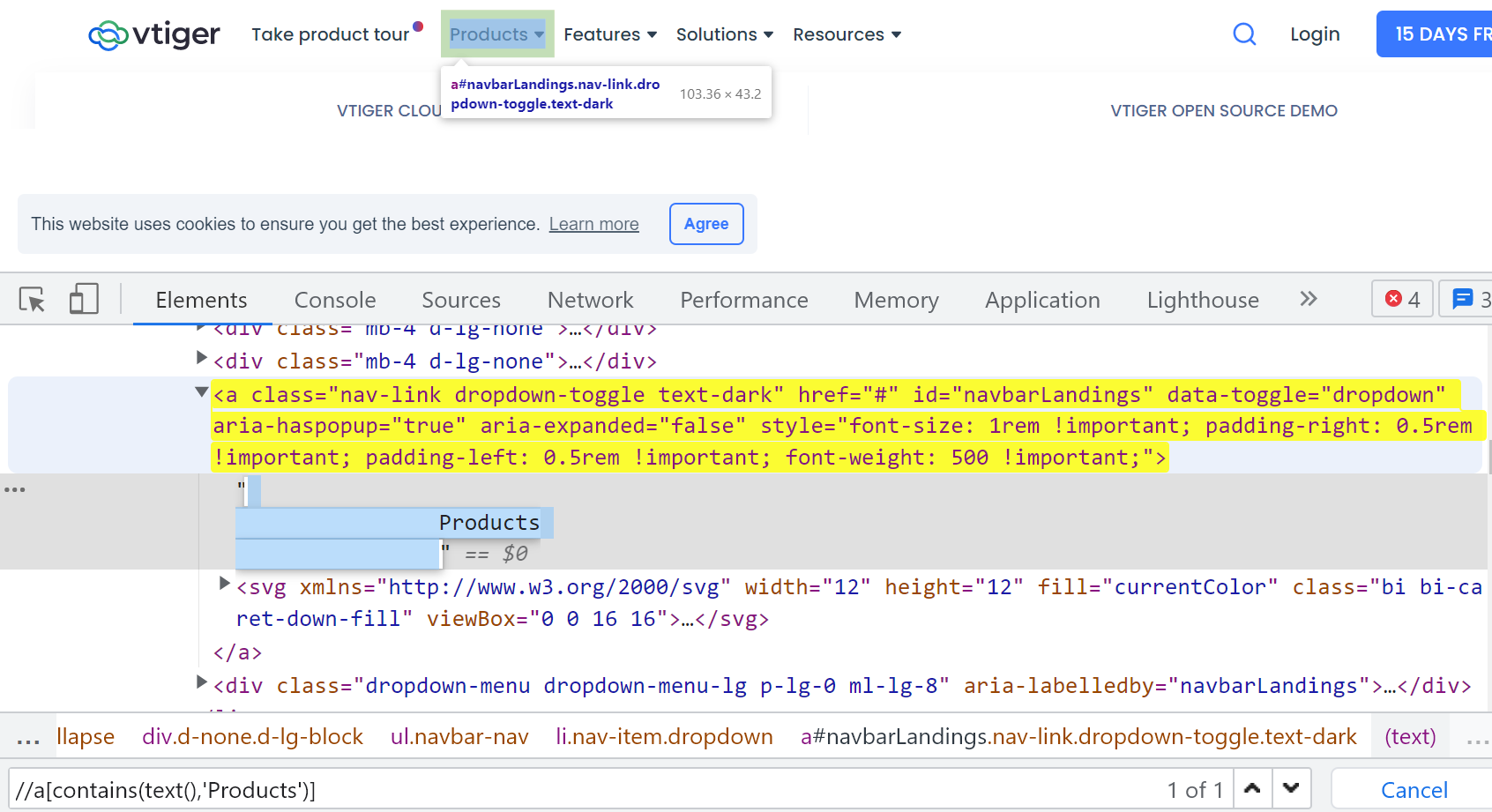
//label[contains(text(),’Keep’)]

* It handles partially changing elements

Ex: 

//span[contains(text(),’build’)]

* It handles non-breakable spaces

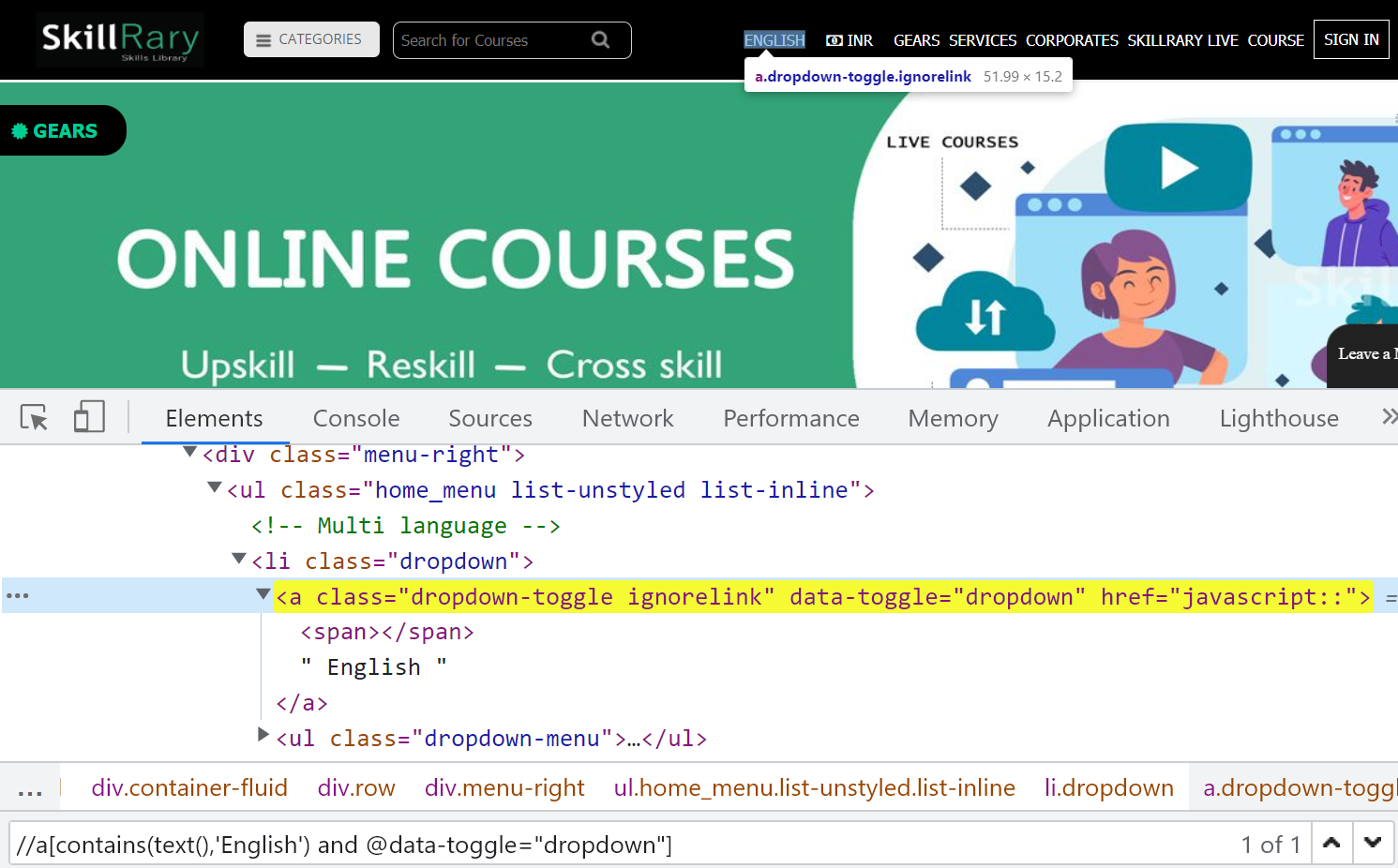
Ex: 

//a[contains(text(),’Products’)]

Note: Non-breakable spaces are the spaces given by the developer/web designer while developing a web page using the command ‘&nbsp;’ to differentiate elements having same inner texts.

Xpath Keywords:

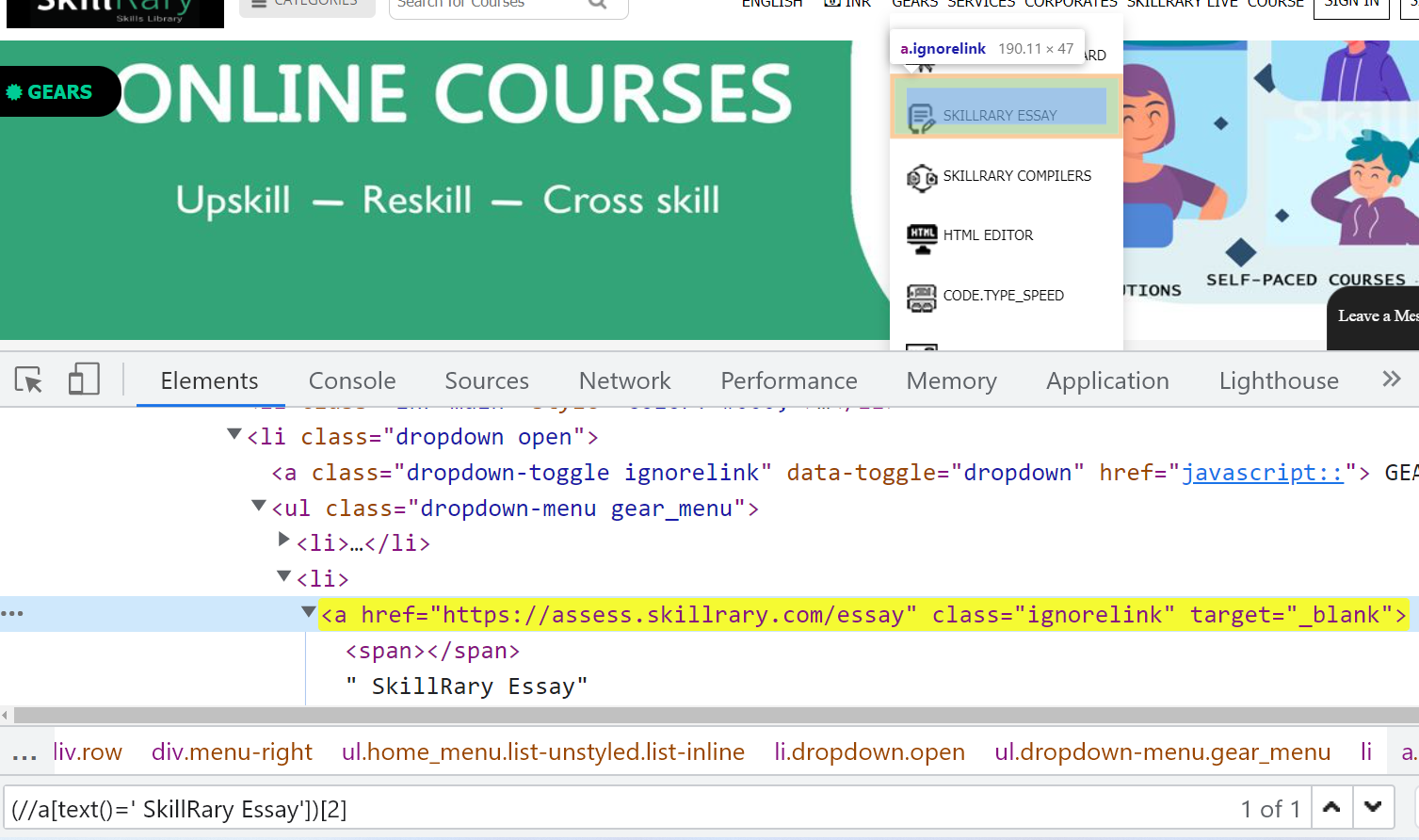
1. and: matches both the attributes
2. or: matches any one of the attributes

Ex: 

//a[contains(text(),’English’) and @data-toggle=’dropdown’]

1. Advanced Relative xpath:
2. Xpath by group index: When xpath written to node fetches multiple matching elements, index of the element is specified to locate the element.

Syntax: (xpath expression)[index]

Ex: 

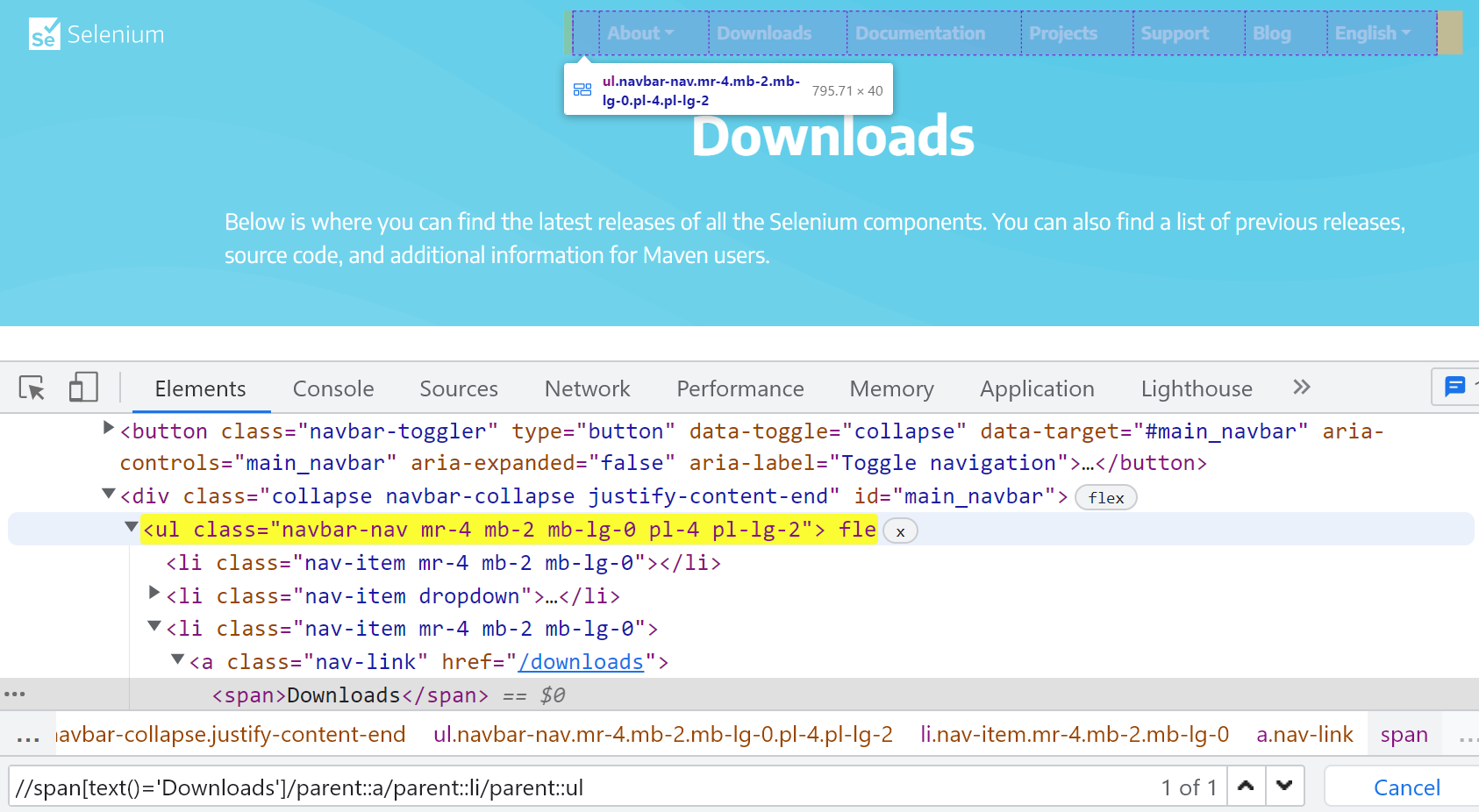
(//a[text()=’ SkillRary Essay])[2]

1. Xpath by axes:

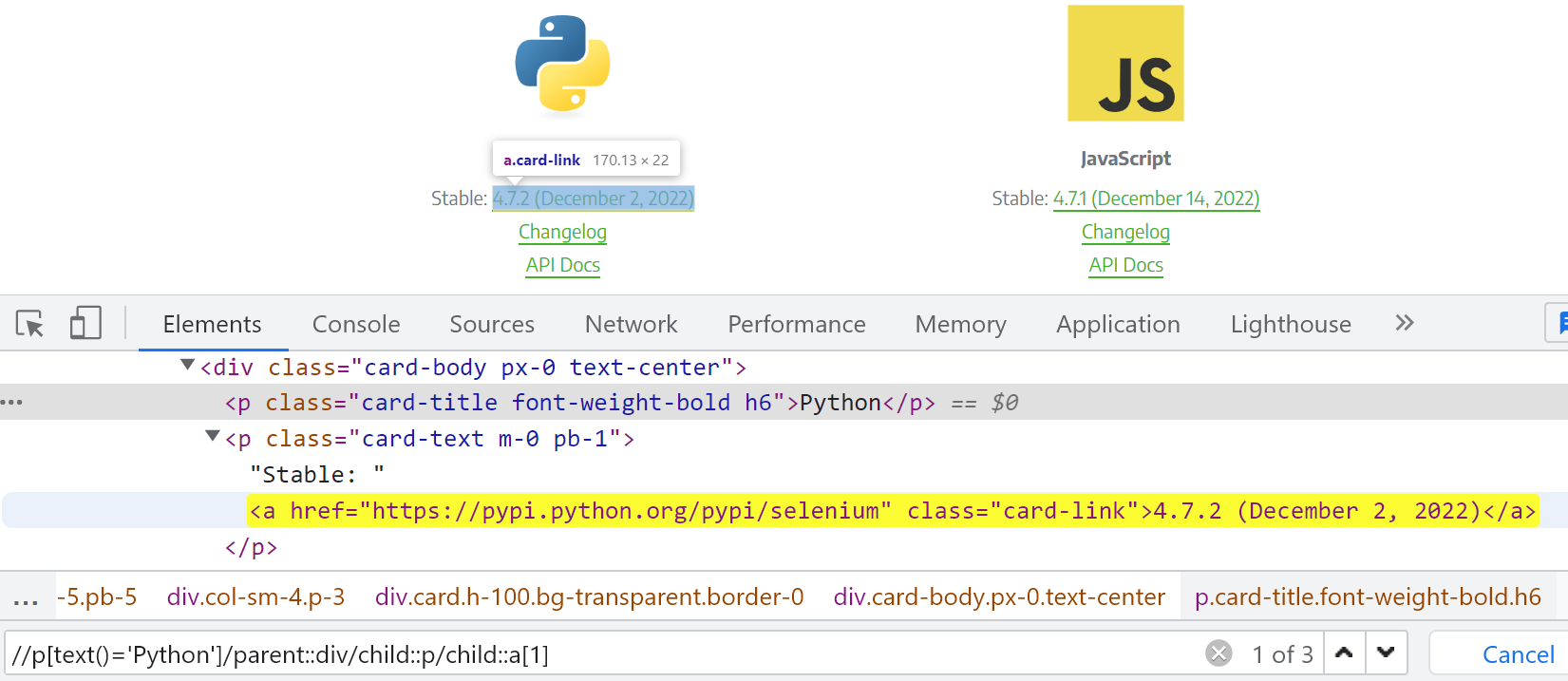
Xpath has different axes which helps in optimizing the xpath expression. All the xpath by axes have same usage as follows:

/axes\_name::tagname

1. parent axes: Traversing to immediate parent of current node.

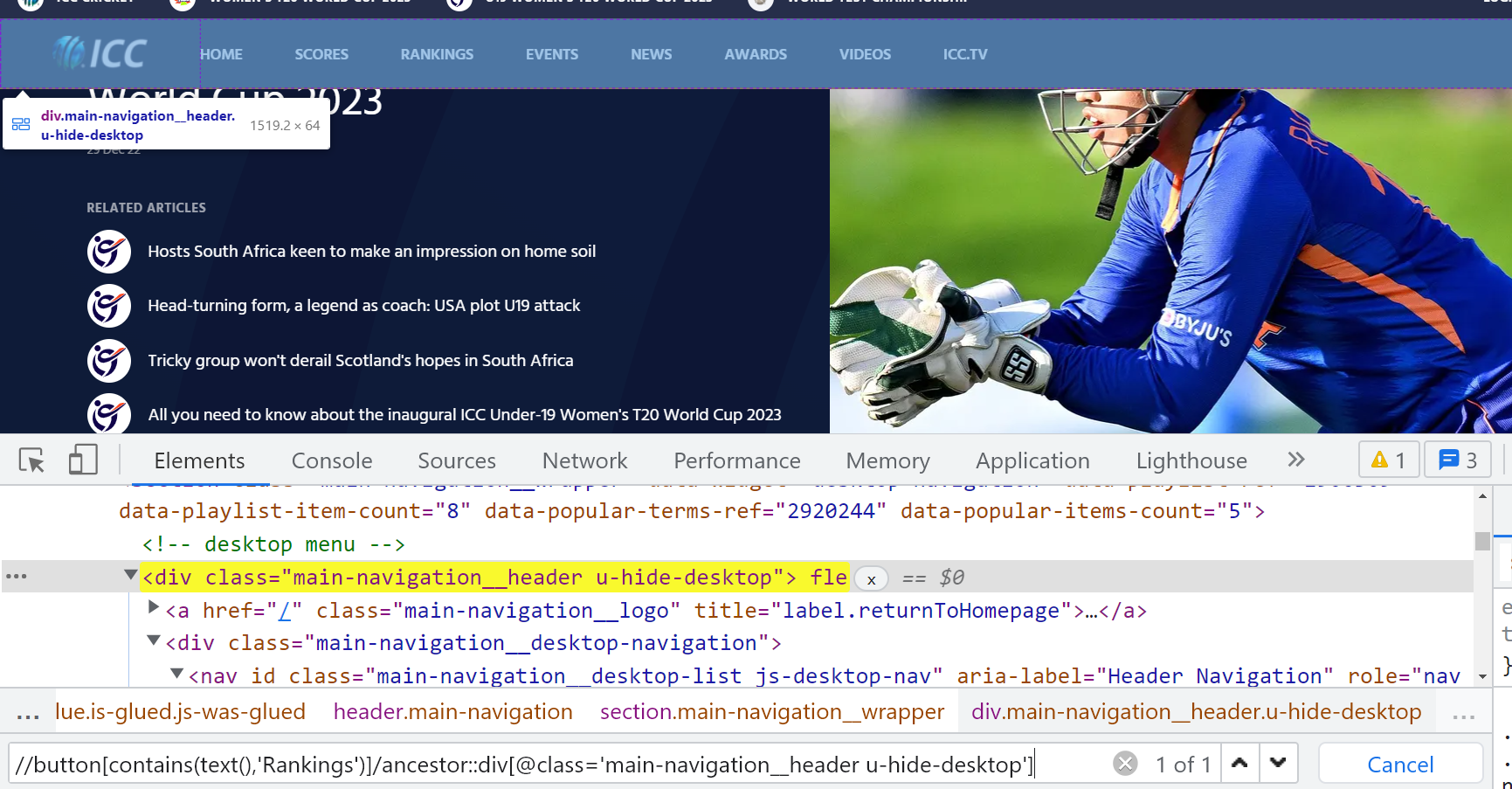
Ex: Traversing from Downloads to header navigation 

1. child axes: Traversing to immediate child of the current node.

Ex: traversing from Python to version 

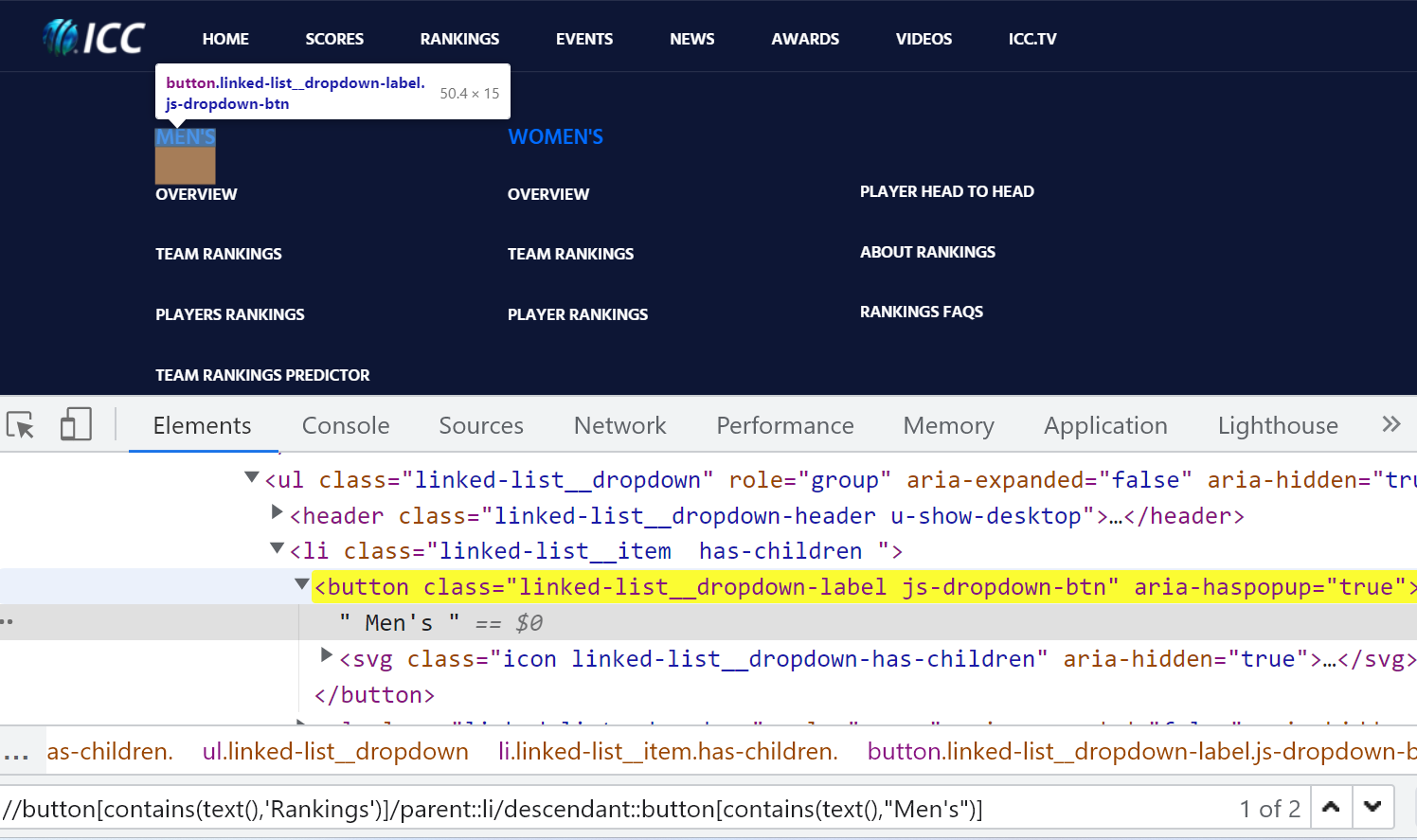
1. ancestor axes: Traversing to any parent node from the current node.

Ex: Traversing from Rankings button to header navigation bar



1. descendant axes: Traversing to any child node from the current node

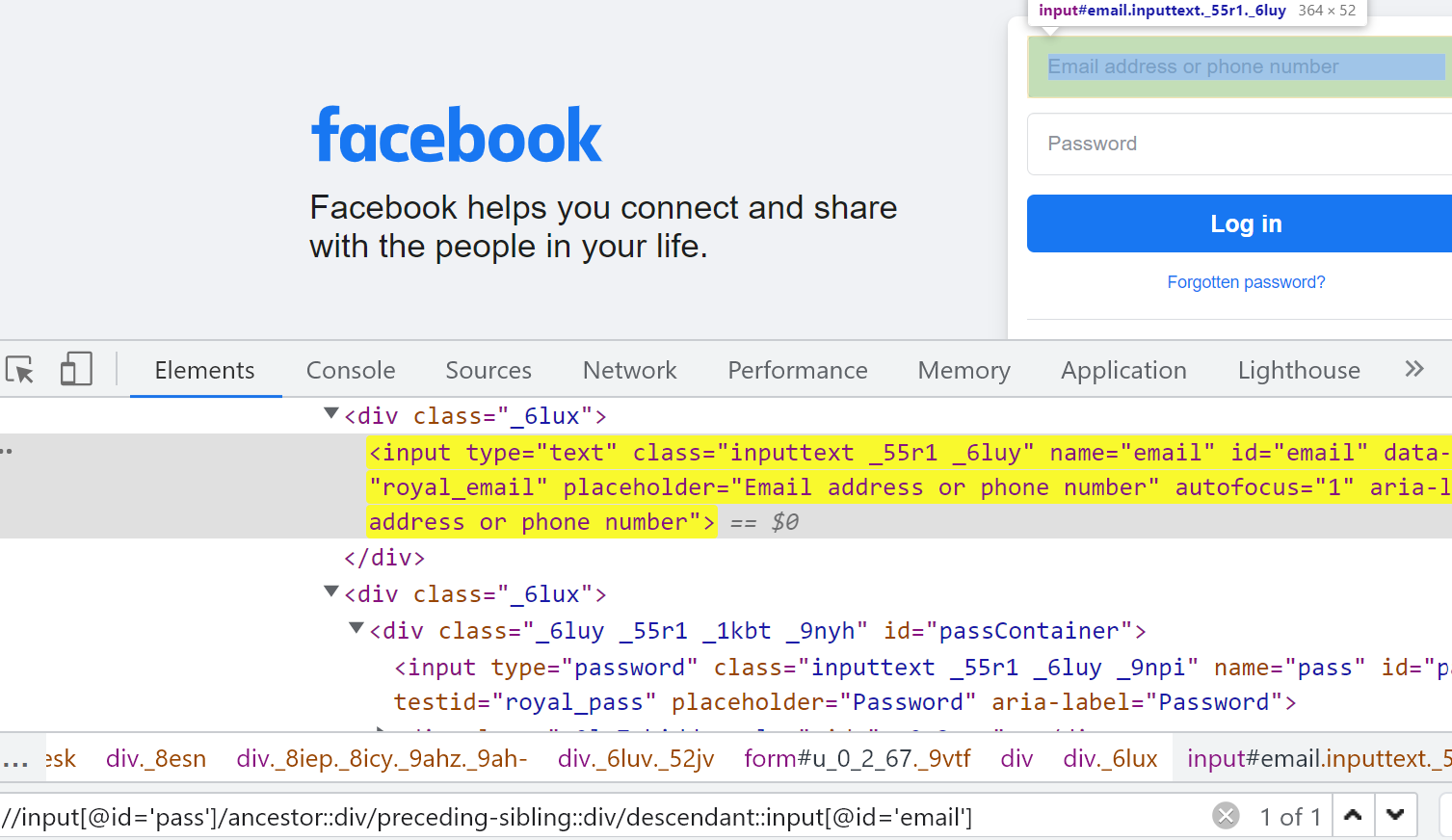
Ex: Traversing from Rankings button to Men’s section



1. Sibling functions:

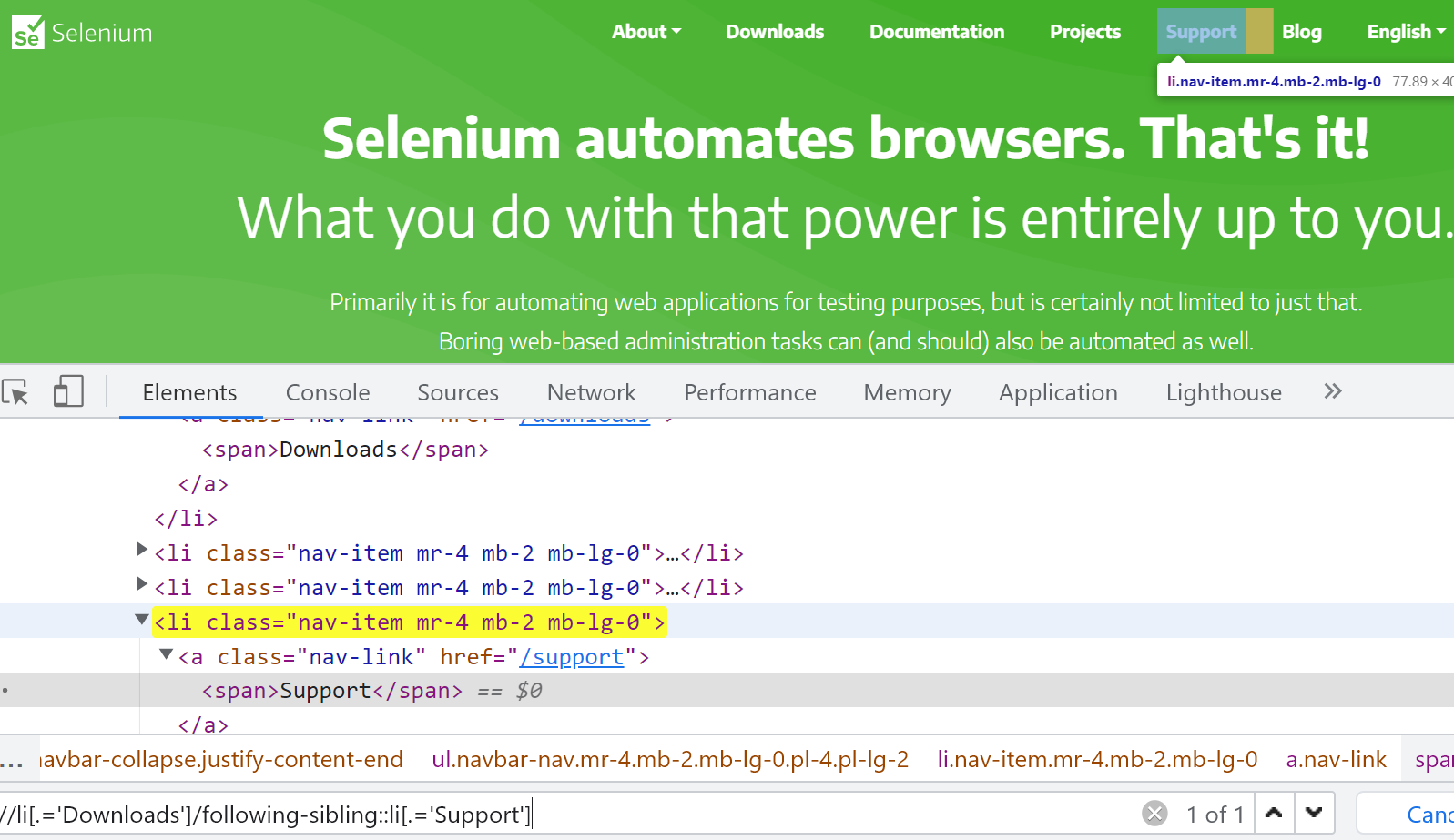
* preceding-sibling axes: Traversing to the previous nodes from the current node which are at the same level, having common parent.

Ex: Traversing from password text field to username text field



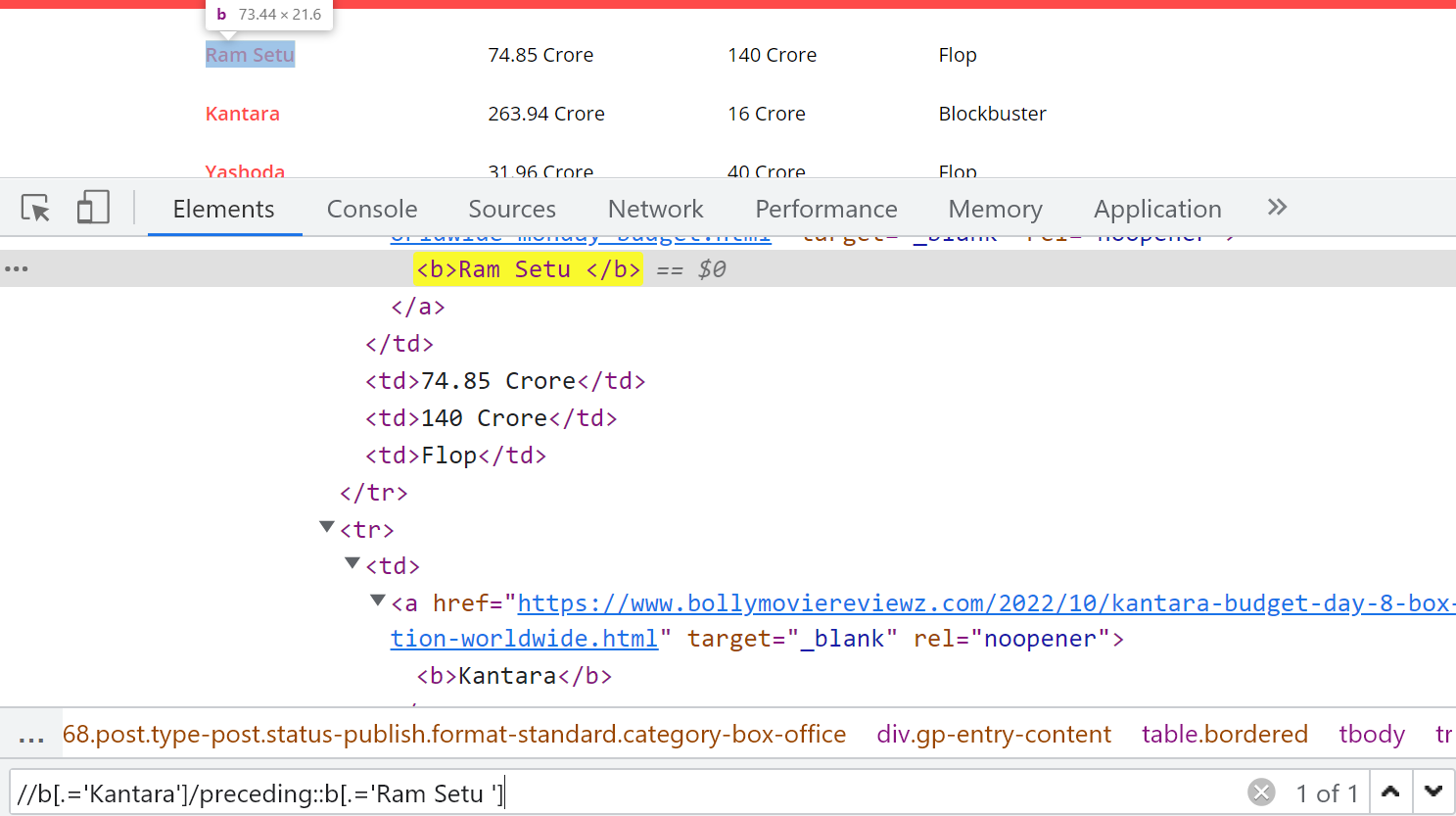
* following-sibling axes: Traversing to the next nodes from current node which are at the same level, having common parent.

Ex: Traversing from Downloads to Support in header



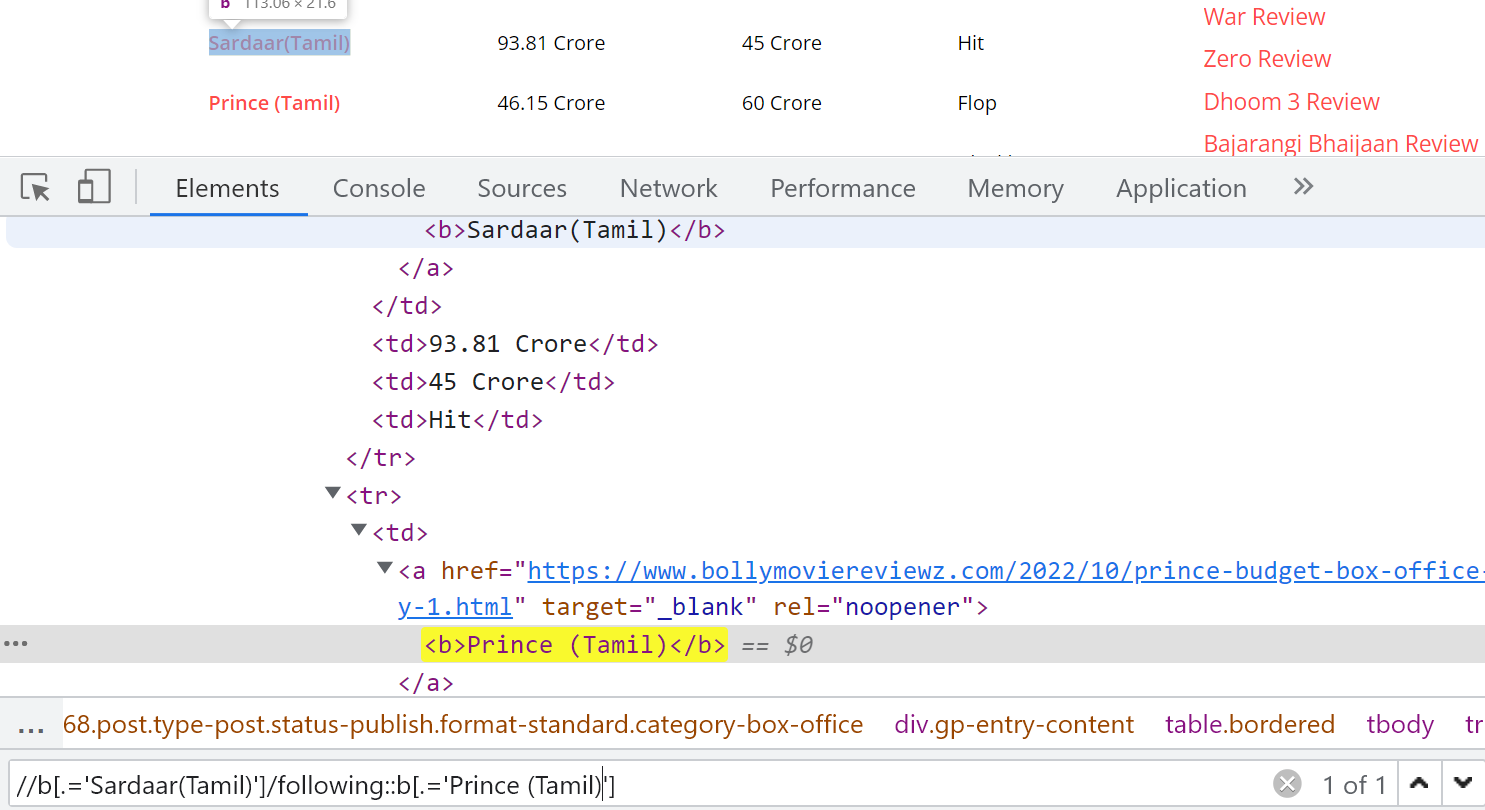
1. preceding axes: Traversing to the previous nodes from current node which are at the same level having different parents.

Ex: Traversing from Kantara to Ram Setu



1. following axes: Traversing to the next nodes from current node which are at the same level having different parents.

Ex: Traversing from Sardar to Prince



Xpath functions:

1. text() : To fetch visible text/inner text on the element node.
2. contains(): To get partial text/partial attribute value on the element node.
3. normalize-space(): To ignore unwanted preceding and following spaces in the text/attribute value on the element node.

Syntax: //tagname[normalize-space(@attributename)=’attribute\_value’]

//tagname[normalize-space(text())=’text\_value’]

Ex:

* <input type=” text “ id=” userID “ />

//input[normalize-space(@type)=’text’]

* <span> GEARS </span>

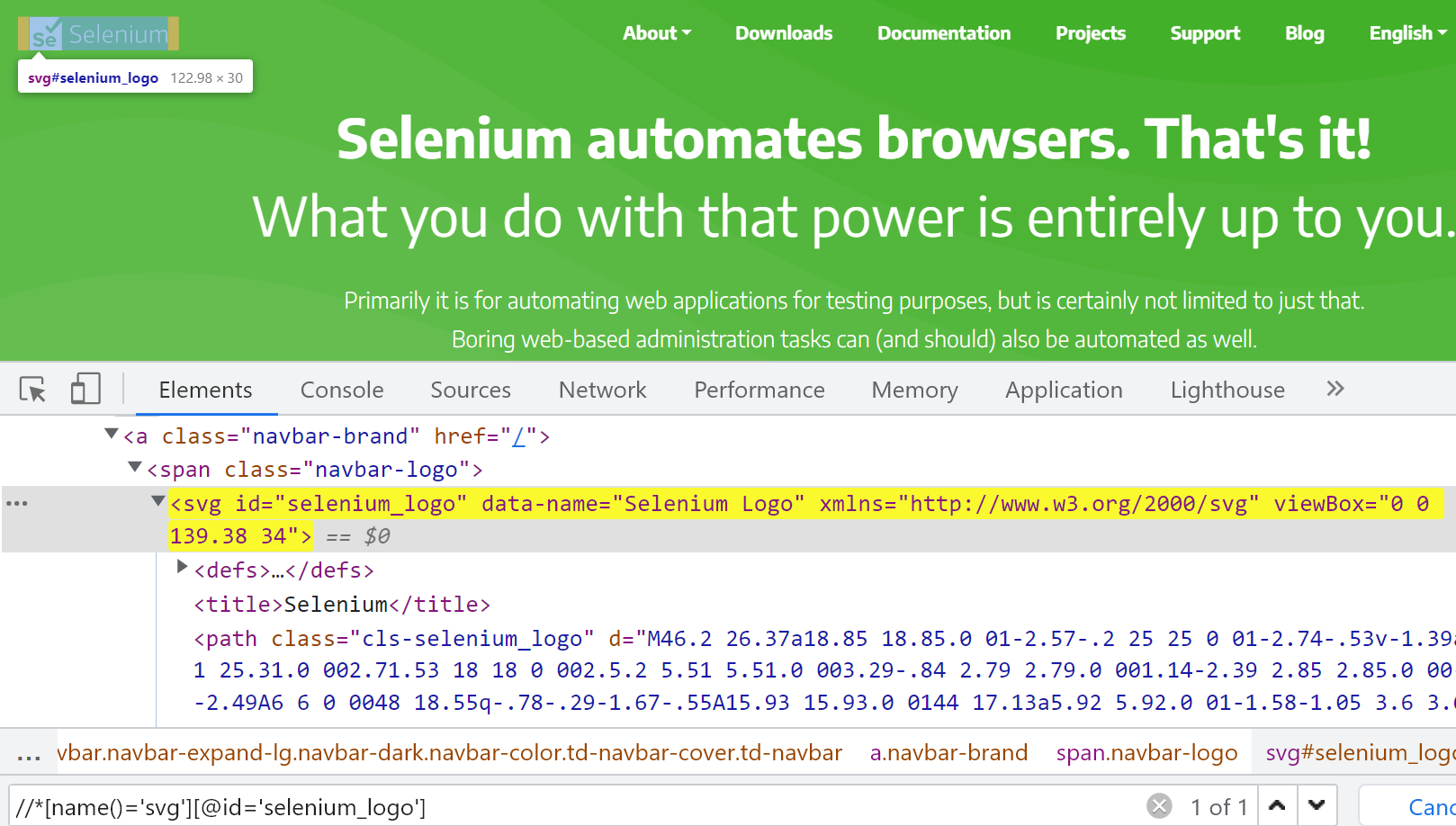
//span[normalize-space(text())=’GEARS’]

1. last(): To fetch the last node in the list of element nodes in web tables.
2. name(): To identify svg tags.

svg tags are graphical elements like circle, symbols – chrome symbol, firefox symbol which cannot be handled using normal xpath format.

Syntax: //\*[name()=’svg’]

Identify Selenium logo in selenium.dev



What is dynamic xpath?

* Xpath which is getting created during runtime is called dynamic xpath.
* Dynamic xpath can be used to multiple elements by changing the variable value in runtime.
* Dynamic xpath reduces number of elements in Object Repository.
* Ex: //a[text()=”+orgName+”]

Here orgName is the dynamic data.