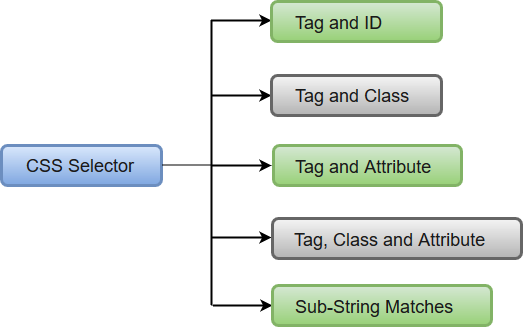
**[next →](https://www.javatpoint.com/selenium-webdriver-locating-strategies)**[**← prev**](https://www.javatpoint.com/selenium-webdriver-locating-strategies)

Locating Strategies- (By CSS)

* CSS **stands for Cascading Style Sheets**. It is a Style Sheet Language which is used to describe the look and formatting of a document written in markup language.
* locating web elements through CSS involves use of CSS Selector which identifies an element based on the combination of HTML tag, id, class and attributes.

You can also refer to our CSS tutorial: <https://www.javatpoint.com/css-selector>

In WebDriver, CSS Selector works in six modes to identify and locate web elements.



1. [Tag and ID](https://www.javatpoint.com/webdriver-locating-strategies-by-css-tag-and-id)
2. [Tag and Class](https://www.javatpoint.com/webdriver-locating-strategies-by-css-tag-and-class)
3. [Tag and Attribute](https://www.javatpoint.com/webdriver-locating-strategies-by-css-tag-and-attribute)
4. [Tag, Class and Attribute](https://www.javatpoint.com/webdriver-locating-strategies-by-css-tag-class-and-attribute)
5. [Sub-String Matches](https://www.javatpoint.com/webdriver-locating-strategies-by-css-sub-string-matches)

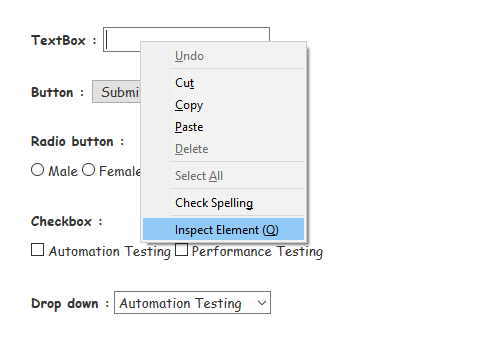
Locating Strategies- (By CSS- Tag and ID)

In this section, you will learn how to locate a particular web element using CSS - Tag and ID Selector.

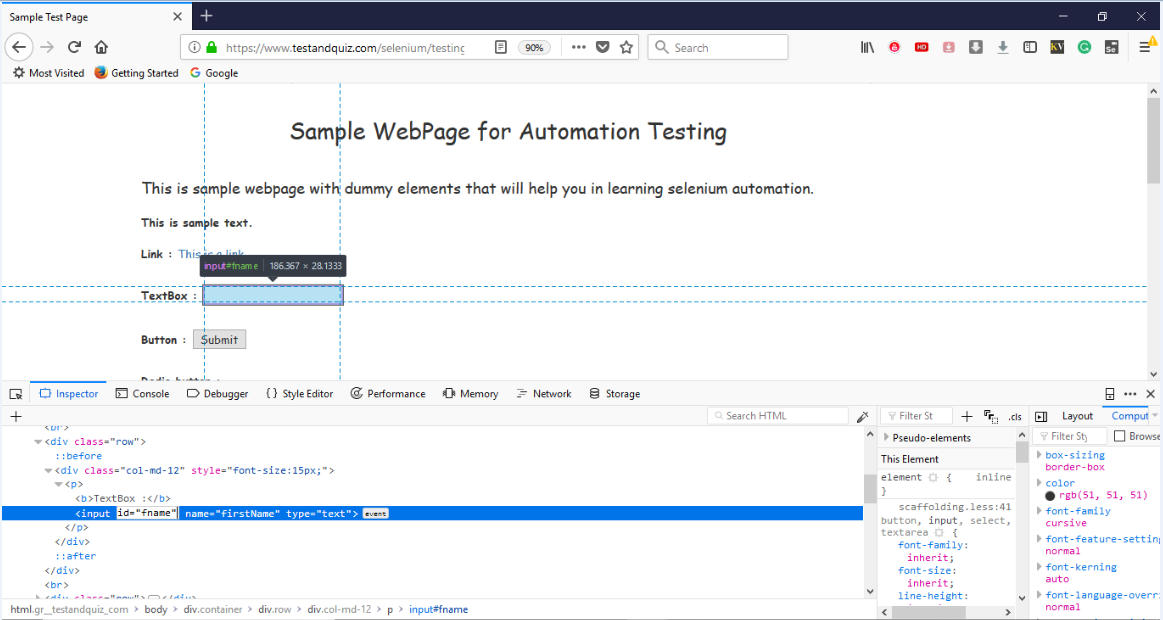
As we know that locating a particular web element involves inspection of its HTML codes.

Follow the steps given below to locate the Textbox on the sample web page.

* Open URL: <https://www.testandquiz.com/selenium/testing.html>
* Right click on the Textbox on the sample web page and select Inspect Element



* It will launch a window containing all the specific codes involved in the development of the Text box.



* Take a note of its Tag and value of its id attribute.



The Java Syntax for locating a web element through CSS - Tag and ID Selector is written as:

1. driver.findElement(By.cssSelector("Tag#Value of id attribute"))

Therefore, for locating the Textbox on the sample web page, we will use the input tag along with the value of its id attribute:

1. driver.findElement(By.cssSelector("input#fname"))

Similarly, for locating the Submit button on the sample web page, we will use the button tag along with the value of its id attribute:

1. driver.findElement(By.cssSelector("button#idOfButton"))

We have created a sample script for you to get a better understanding of how to use CSS - Tag and ID Selector. We have embedded comments in each section of code which will guide you through whole automation process.

1. **import** org.openqa.selenium.By;
2. **import** org.openqa.selenium.WebDriver;
3. **import** org.openqa.selenium.firefox.FirefoxDriver;
4. **import** org.openqa.selenium.remote.DesiredCapabilities;
6. **public** **class** SampleOne {
8. **public** **static** **void** main(String[] args) {
10. // System Property for Gecko Driver
11. System.setProperty("webdriver.gecko.driver","D:\\GeckoDriver\\geckodriver.exe" );
13. // Initialize Gecko Driver using Desired Capabilities Class
14. DesiredCapabilities capabilities = DesiredCapabilities.firefox();
15. capabilities.setCapability("marionette",**true**);
16. WebDriver driver= **new** FirefoxDriver(capabilities);

19. // Launch Website
20. driver.navigate().to("https://www.testandquiz.com/selenium/testing.html");
22. // Click on the textbox and send value
23. driver.findElement(By.cssSelector("input#fname")).sendKeys("JavaTpoint");
25. // Click on the Submit button using click() command
26. driver.findElement(By.cssSelector("button#idOfButton")).click();

29. //  Close the Browser
30. driver.close();
32. }
34. }

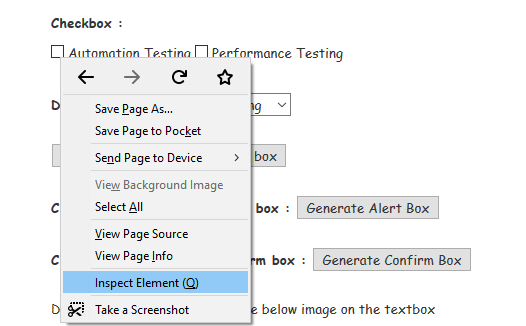
Locating Strategies- (By CSS-Tag and Class)

In this section, you will learn how to locate a particular web element using CSS - Tag and Class Selector.

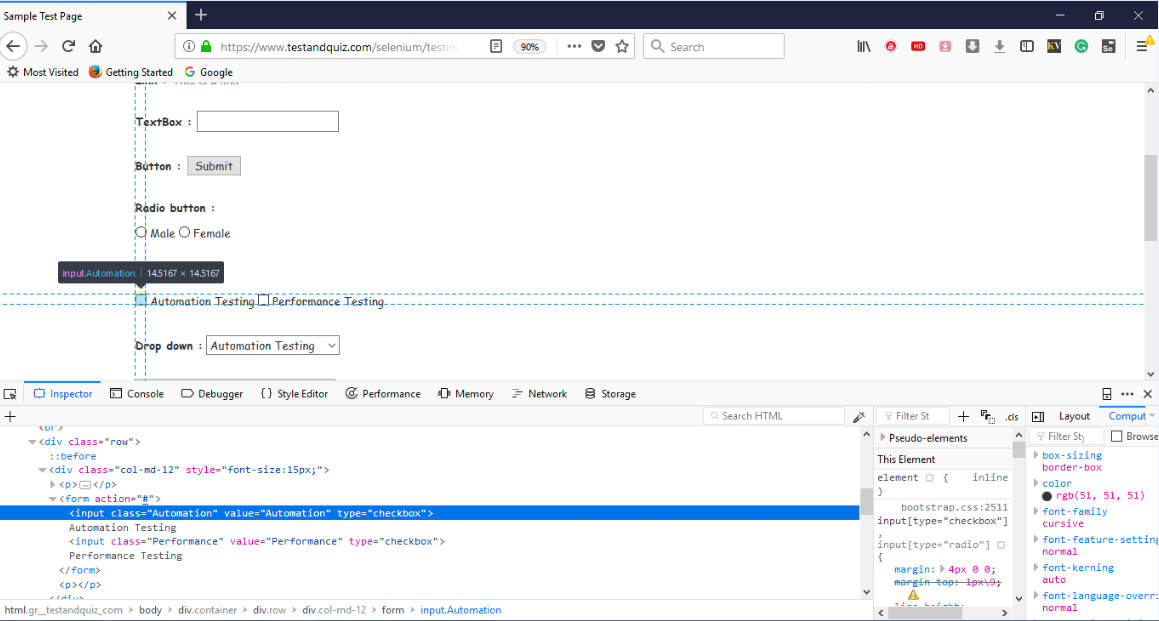
As we know that locating a particular web element involves inspection of its HTML codes.

Follow the steps given below to locate the Textbox on the sample web page1

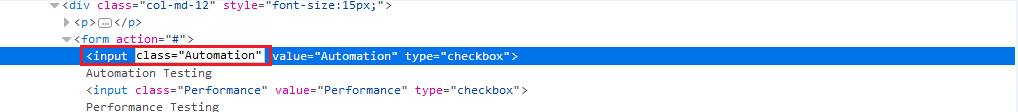
* Open URL: <https://www.testandquiz.com/selenium/testing.html>
* Right click on the "Automation Testing" checkbox on the sample web page and select Inspect Element.



* It will launch a window containing all the specific codes involved in the development of the desired check box.



* Take a note of its Tag and value of its Class attribute.



The Java Syntax for locating a web element through CSS - Tag and Class Selector is written as:

1. driver.findElement(By.cssSelector("Tag.Value of Class attribute"))

Therefore, for locating the "Automation Testing" Checkbox on the sample web page we will use the input tag along with the value of its Class attribute:

1. driver.findElement(By.cssSelector("input.Automation"))

We have created a sample script for you to get a better understanding of how to use CSS - Tag and Class Selector. We have embedded comments in each section of code which will guide you through whole automation process.

1. **import** org.openqa.selenium.By;
2. **import** org.openqa.selenium.WebDriver;
3. **import** org.openqa.selenium.firefox.FirefoxDriver;
4. **import** org.openqa.selenium.remote.DesiredCapabilities;
6. **public** **class** SampleTwo {
8. **public** **static** **void** main(String[] args) {
10. // System Property for Gecko Driver
11. System.setProperty("webdriver.gecko.driver","D:\\GeckoDriver\\geckodriver.exe" );
13. // Initialize Gecko Driver using Desired Capabilities Class
14. DesiredCapabilities capabilities = DesiredCapabilities.firefox();
15. capabilities.setCapability("marionette",**true**);
16. WebDriver driver= **new** FirefoxDriver(capabilities);
18. // Launch Website
19. driver.navigate().to("https://www.testandquiz.com/selenium/testing.html");


23. // Locate the checkbox by cssSelector and check it using click() function
24. driver.findElement(By.cssSelector("input.Automation")).click();

27. // Close the Browser
28. driver.close();
30. }
32. }

[**next →← prev**](https://www.javatpoint.com/selenium-webdriver-locating-strategies-by-css)

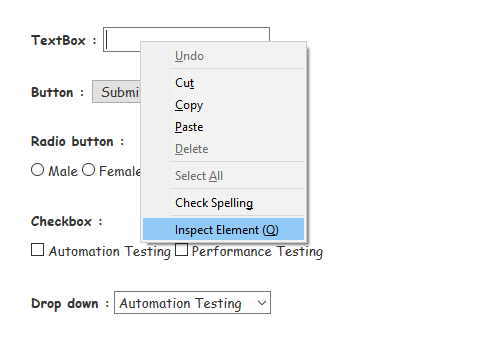
# Locating Strategies- (By CSS-Tag and Attribute)

In this section, you will learn how to locate a particular web element using CSS - Tag and Attribute Selector.

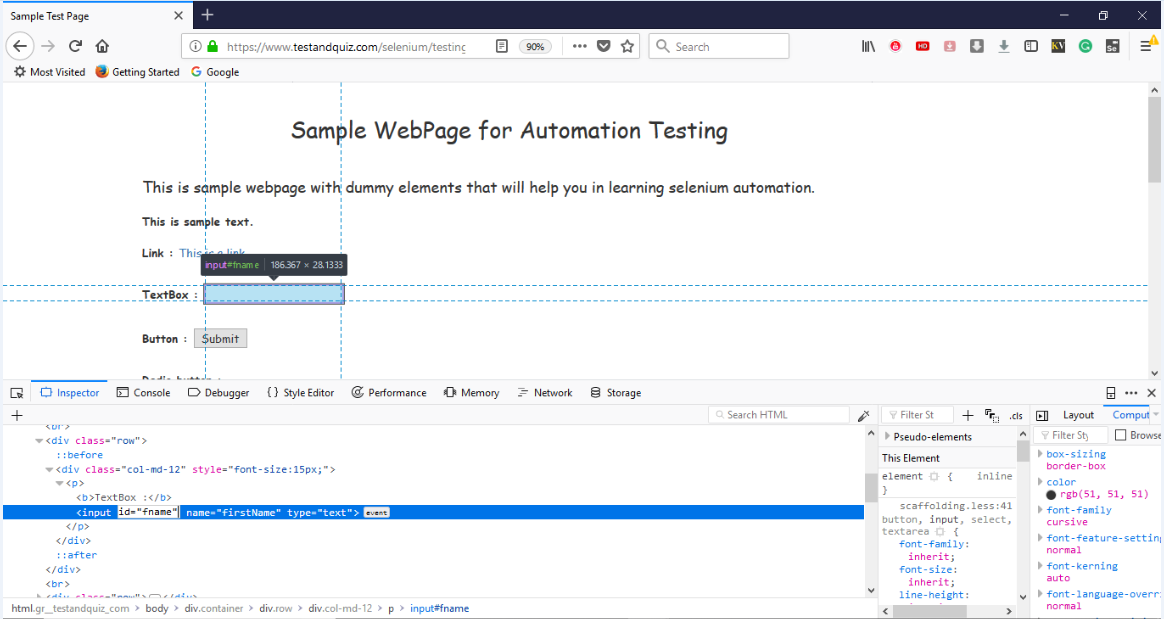
As we know that locating a particular web element involves inspection of its HTML codes.

Follow the steps given below to locate the Textbox on the sample web page.

* Open URL: <https://www.testandquiz.com/selenium/testing.html>
* Right click on the Textbox on the sample web page and select Inspect Element



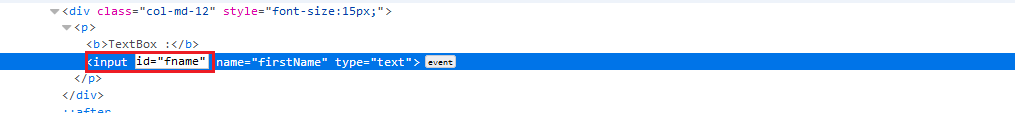
* It will launch a window containing all the specific codes involved in the development of the Text box.



* Take a note of its Tag and Attribute.

#### Note: You can choose attributes such as id, class and name along with their values when you are locating through CSS - Tag and Attribute Selector.

* Here, we will take a note of its tag and value of its id attribute.



The Java Syntax for locating a web element through CSS - Tag and Attribute Selector is written as:

1. driver.findElement(By.cssSelector("Tag[Attribute=value]"))

Therefore, for locating the Textbox on the sample web page we will use the input tag with id attribute:

1. driver.findElement(By.cssSelector("input[id=fname]"))

Similarly, for locating the Submit button on the sample web page we will use the button tag with id attribute:

1. driver.findElement(By.cssSelector("button[id=idOfButton]"))

We have created a sample script for you to get a better understanding of how to use CSS - Tag and Attribute Selector. We have embedded comments in each section of code which will guide you through whole automation process.

1. **import** org.openqa.selenium.By;
2. **import** org.openqa.selenium.WebDriver;
3. **import** org.openqa.selenium.firefox.FirefoxDriver;
4. **import** org.openqa.selenium.remote.DesiredCapabilities;
6. **public** **class** SampleThree {
8. **public** **static** **void** main(String[] args) {
10. // System Property for Gecko Driver
11. System.setProperty("webdriver.gecko.driver","D:\\GeckoDriver\\geckodriver.exe" );
13. // Initialize Gecko Driver using Desired Capabilities Class
14. DesiredCapabilities capabilities = DesiredCapabilities.firefox();
15. capabilities.setCapability("marionette",**true**);
16. WebDriver driver= **new** FirefoxDriver(capabilities);

19. // Launch Website
20. driver.navigate().to("https://www.testandquiz.com/selenium/testing.html");
22. // Click on the textbox and send value
23. driver.findElement(By.cssSelector("input[id=fname]")).sendKeys("Selenium Tutorials");
25. // Click on the Submit button using click() command
26. driver.findElement(By.cssSelector("button[id=idOfButton]")).click();

29. // Close the Browser
30. driver.close();
32. }
34. }

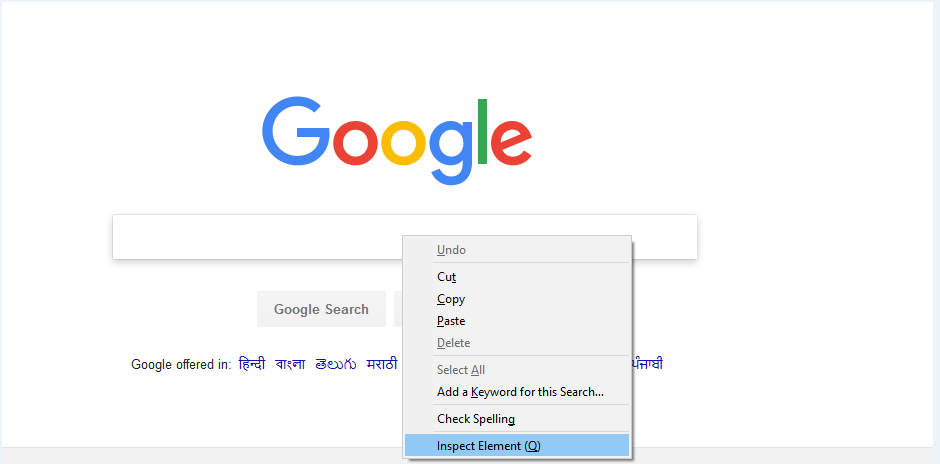
Locating Strategies- (By CSS-Tag, Class and Attribute)

In this section, you will learn how to locate a particular web element using CSS - Tag, Class and Attribute Selector.

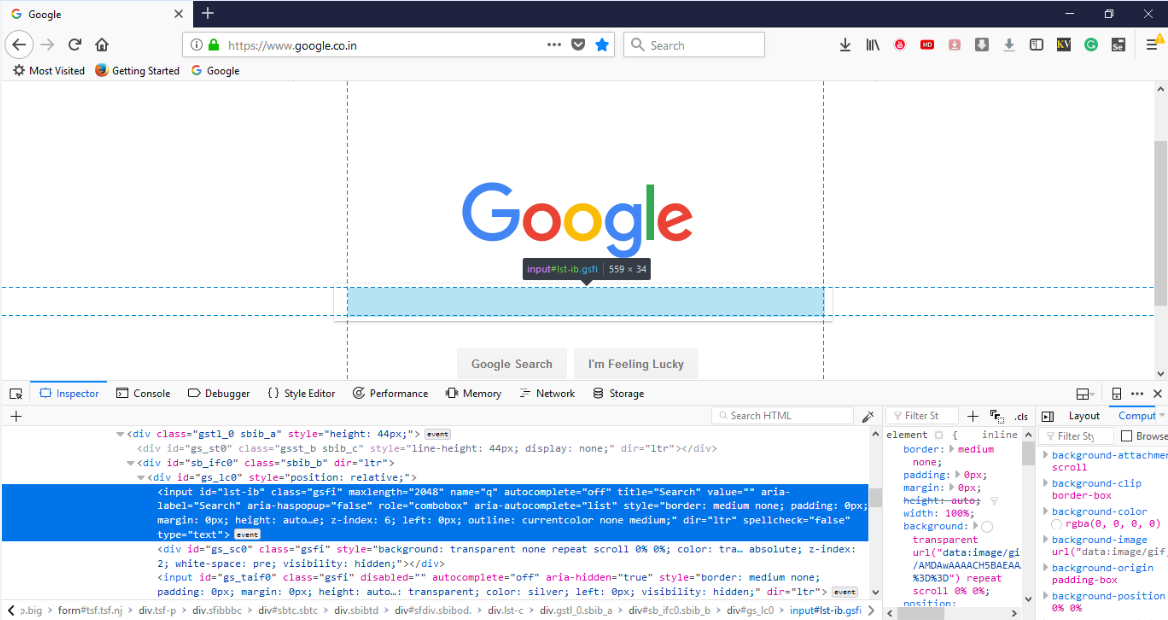
Under this scenario, we will inspect the search text box of one of the most popular search engine i.e. Google.

As we know that locating a particular web element involves inspection of its HTML codes. Therefore, follow the steps given below to locate the textbox provided on the initial web page of Google Search Engine.

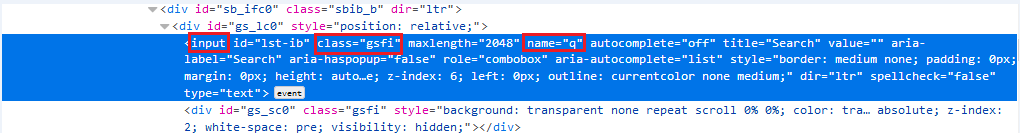
* Open URL: <https://www.google.co.in/>
* Right click on the Textbox on the sample web page and select Inspect Element



* It will launch a window containing all the specific codes involved in the development of the textbox.



* Take a note of its Tag, Class and Attribute.



The Java Syntax for locating a web element through CSS - Tag, Class and Attribute Selector is written as:

1. driver.findElement(By.cssSelector("tag.class[attribute=value]"))

Therefore, for locating the Textbox on the Google Search Engine page we will use the input tag, Class and Name attribute:

1. driver.findElement(By.cssSelector("input.gsfi[name=q]"))

Similarly, for locating the Google Search button on the Google Search Engine page we will use its respective input tag, Class and Name attribute:

1. driver.findElement(By.cssSelector("input.jsp[name=btnK]"))

We have created a sample script for you to get a better understanding of how to use CSS - Tag, Class and Attribute Selector. We have embedded comments in each section of code which will guide you through whole automation process.

1. **import** org.openqa.selenium.By;
2. **import** org.openqa.selenium.WebDriver;
3. **import** org.openqa.selenium.firefox.FirefoxDriver;
4. **import** org.openqa.selenium.remote.DesiredCapabilities;
6. **public** **class** SampleFour {
8. **public** **static** **void** main(String[] args) {
10. // System Property for Gecko Driver
11. System.setProperty("webdriver.gecko.driver","D:\\GeckoDriver\\geckodriver.exe" );
13. // Initialize Gecko Driver using Desired Capabilities Class
14. DesiredCapabilities capabilities = DesiredCapabilities.firefox();
15. capabilities.setCapability("marionette",**true**);
16. WebDriver driver= **new** FirefoxDriver(capabilities);

19. // Launch Website
20. driver.navigate().to("www.google.co.in");
22. // Click on the textbox and send value
23. driver.findElement(By.cssSelector("input.gsfi[name=q]")).sendKeys("javaTpoint Tutorials");
25. // Click on the Google Search button using click() command
26. driver.findElement(By.cssSelector("input.jsp[name=btnK]")).click();

29. // Close the Browser
30. driver.close();
32. }
34. }

# Locating Strategies- (By CSS- Sub-String Matches)

In this section, you will learn how to locate a particular web element using CSS - Sub-String Matches Technique.

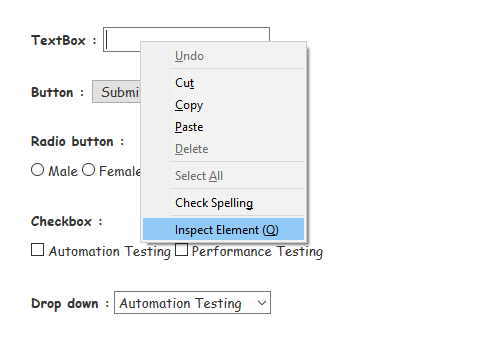
WebDriver provides an interesting feature of allowing partial string matches using ^, $ and\*.

### 1. Starts with (^):

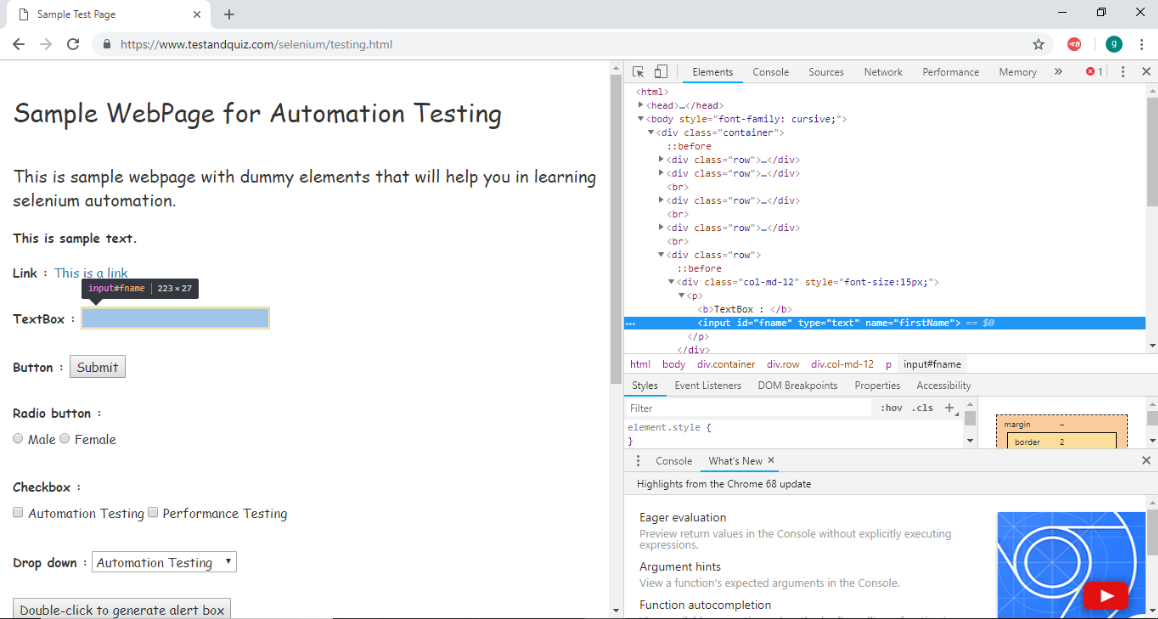
To select and locate a web element, we would use ^ which means 'starts with'.

Follow the steps given below to locate the Textbox using partial string matching technique.

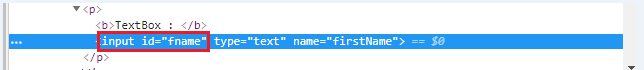
* Open URL: <https://www.testandquiz.com/selenium/testing.html>
* Right click on the Textbox on the sample web page and select Inspect Element



* It will launch a window containing all the specific codes involved in the development of the Text box.



* Take a note of its tag and id attribute.



The Java Syntax for locating a web element through CSS - Sub-String Matches Technique using ^ operator is written as:

1. driver.findElement(By.cssSelector("Tag[attribute^=prefix of the string]"))

Therefore, for locating the Text box on the sample web page we will use the input tag along with the id attribute:

1. driver.findElement(By.cssSelector("input[id^='fna']"))

Similarly, for locating the Submit button on the sample web page we will use the button tag along with the id attribute:

1. driver.findElement(By.cssSelector("button[id^='idOf']"))

We have created a sample script for you to get a better understanding of how to locate a web element through CSS - Sub-String Matches Technique using ^ operator.

1. **import** org.openqa.selenium.By;
2. **import** org.openqa.selenium.WebDriver;
3. **import** org.openqa.selenium.chrome.ChromeDriver;
5. **public** **class** SampleFive {
7. **public** **static** **void** main(String[] args) {
9. // System Property for Chrome Driver
10. System.setProperty("webdriver.chrome.driver","D:\\ChromeDriver\\chromedriver.exe");
12. // Instantiate a ChromeDriver class.
13. WebDriver driver=**new** ChromeDriver();
15. // Launch Website
16. driver.navigate().to("https://www.testandquiz.com/selenium/testing.html");
18. // Click on the textbox and send value
19. driver.findElement(By.cssSelector("input[id^='fna']")).sendKeys("JavaTpoint JMeter Tutorial");
21. // Click on the Submit button using click() command
22. driver.findElement(By.cssSelector("button[id^='idOf']")).click();
24. // Close the Browser
25. driver.close();

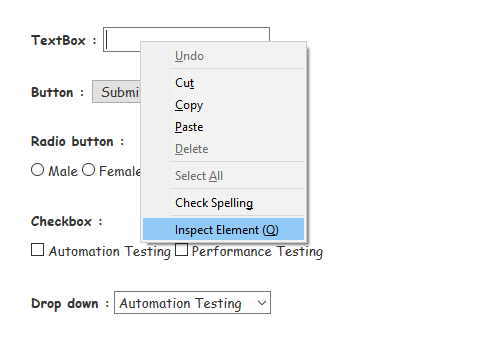
28. }
30. }

### 2. Ends with ($):

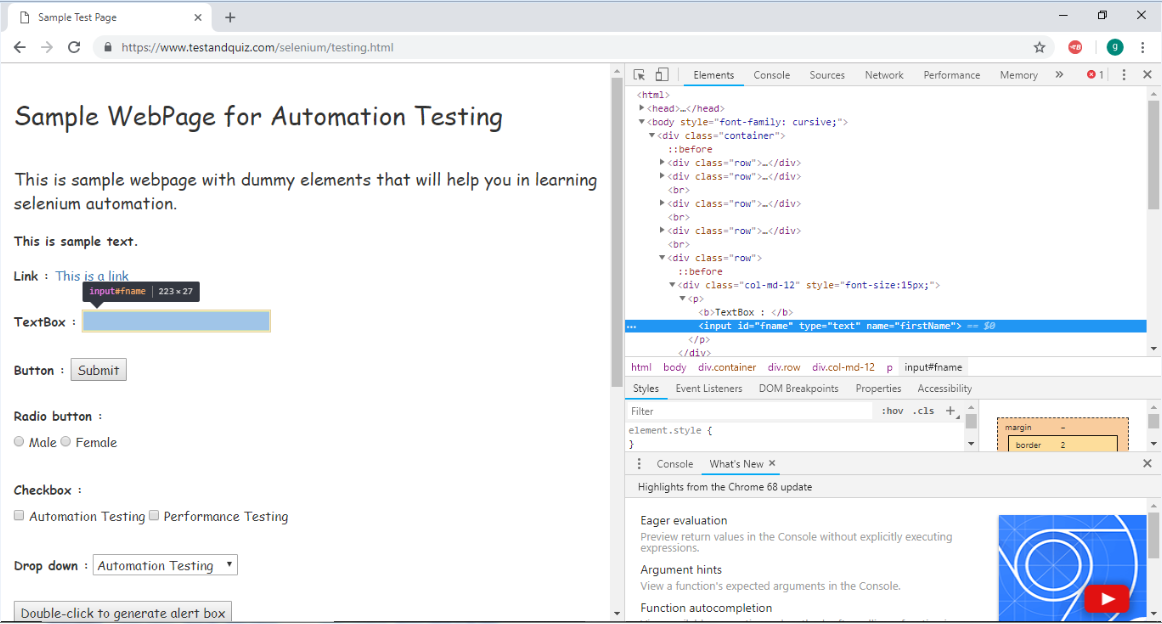
To select and locate a web element, we would use $ which means 'ends with'.

Follow the steps given below to locate the Textbox using partial string matching technique.

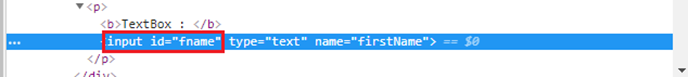
* Open URL: <https://www.testandquiz.com/selenium/testing.html>
* Right click on the Textbox on the sample web page and select Inspect Element



* It will launch a window containing all the specific codes involved in the development of the Text box.



* Take a note of its tag and id attribute.



The Java Syntax for locating a web element through CSS - Sub-String Matches Technique using $ operator is written as:

1. driver.findElement(By.cssSelector("Tag[attribute$=suffix of the string]"))

Therefore, for locating the Text box on the sample web page we will use the input tag along with the id attribute:

1. driver.findElement(By.cssSelector("input[id$='me']"))

Similarly, for locating the Submit button on the sample web page we will use the button tag along with the id attribute:

1. driver.findElement(By.cssSelector("button[id$='on']"))

We have created a sample script for you to get a better understanding of how to locate a web element through CSS - Sub-String Matches Technique using $ operator.

1. **import** org.openqa.selenium.By;
2. **import** org.openqa.selenium.WebDriver;
3. **import** org.openqa.selenium.chrome.ChromeDriver;
5. **public** **class** SampleSix {
7. **public** **static** **void** main(String[] args) {
9. // System Property for Chrome Driver
10. System.setProperty("webdriver.chrome.driver","D:\\ChromeDriver\\chromedriver.exe");
12. // Instantiate a ChromeDriver class.
13. WebDriver driver=**new** ChromeDriver();
15. // Launch Website
16. driver.navigate().to("https://www.testandquiz.com/selenium/testing.html");
18. // Click on the textbox and send value
19. driver.findElement(By.cssSelector("input[id$='me']")).sendKeys("JavaTpoint Data Structure Tutorial");
21. // Click on the Submit button using click() command
22. driver.findElement(By.cssSelector("button[id$='on']")).click();
24. //  Close the Browser
25. driver.close();

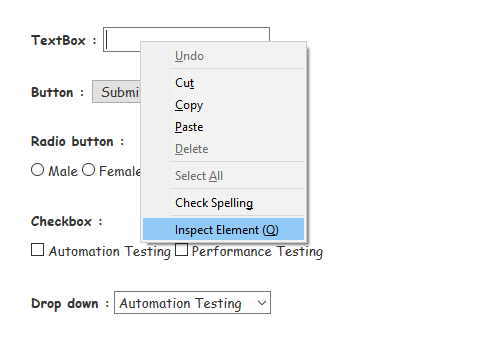
28. }
30. }

### 3. Contains (\*):<

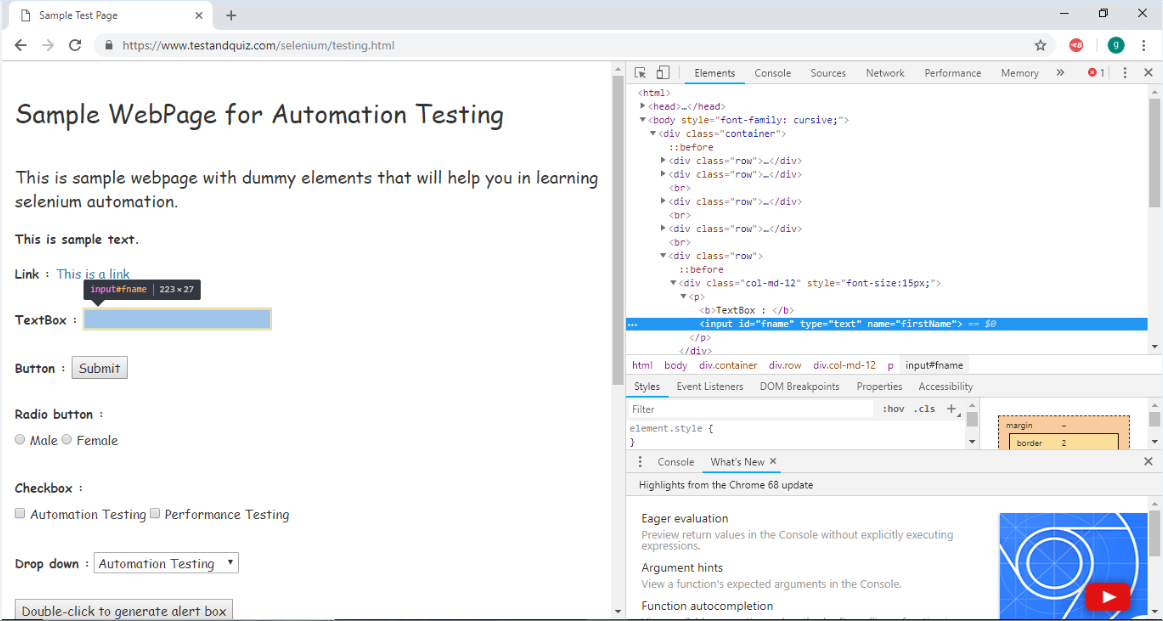
To select and locate a web element, we would use \* which means 'sub-string'.

Follow the steps given below to locate the Textbox using partial string matching technique.

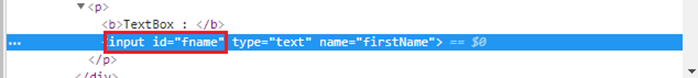
* Open URL: <https://www.testandquiz.com/selenium/testing.html>
* Right click on the Textbox on the sample web page and select Inspect Element



* It will launch a window containing all the specific codes involved in the development of the Text box.



* Take a note of its tag and id attribute.



The Java Syntax for locating a web element through CSS - Sub-String Matches Technique using \* operator is written as:

* 1. driver.findElement(By.cssSelector("Tag[attribute\*=sub-string]"))

Therefore, for locating the Text box on the sample web page we will use the input tag along with the id attribute:

* 1. driver.findElement(By.cssSelector("input[id\*='id']"))

We can also use 'contains()' in place of \* operator

* 1. driver.findElement(By.cssSelector("input:contains('id')"))

We have created a sample script for you to get a better understanding of how to locate a web element through CSS - Sub-String Matches Technique using $ operator.

* 1. **import** org.openqa.selenium.By;
  2. **import** org.openqa.selenium.WebDriver;
  3. **import** org.openqa.selenium.chrome.ChromeDriver;
  5. **public** **class** SampleSeven {
  7. **public** **static** **void** main(String[] args) {
  9. // System Property for Chrome Driver
  10. System.setProperty("webdriver.chrome.driver","D:\\ChromeDriver\\chromedriver.exe");
  12. // Instantiate a ChromeDriver class.
  13. WebDriver driver=**new** ChromeDriver();
  15. // Launch Website
  16. driver.navigate().to("https://www.testandquiz.com/selenium/testing.html");
  18. // Click on the textbox and send value
  19. driver.findElement(By.cssSelector("input[id\*='id']")).sendKeys("JavaTpoint C++ Tutorial");
  21. // Close the Browser
  22. driver.close();

  25. }
  27. }