Xpath Locators:

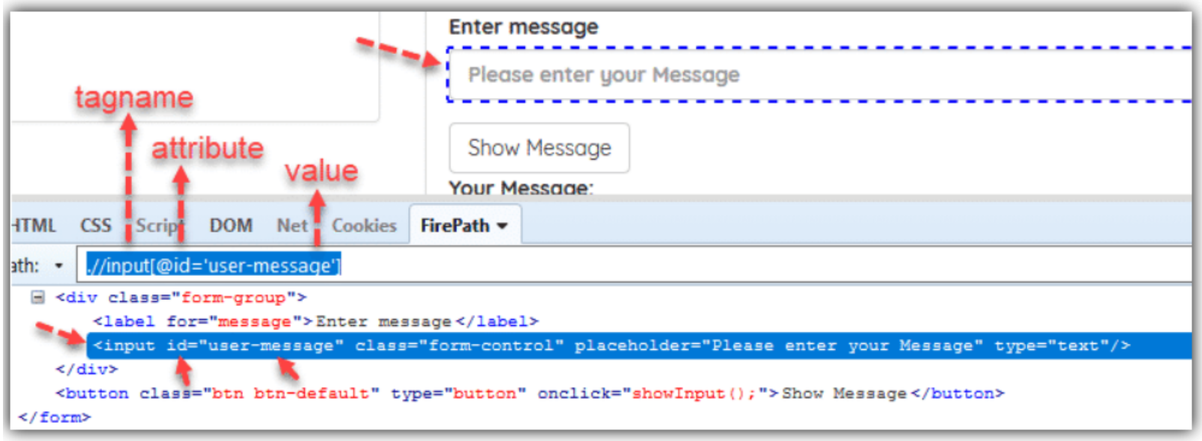
Writing Smart XPaths for Dynamic Elements

1. Tag – Attribute – Value Trio
2. Contains
3. Starts-with
4. Chained XPaths Declarations
5. XPath with “or” Statement
6. XPath with “and” Statement
7. XPath Text
8. Ancestor
9. Following
10. Child
11. Preceding
12. Following-sibling
13. Descendant
14. Parent
15. Locate an Element inside Array of Elements

Basic:

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

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


Enter message
Please enter your Message
Show Message
Your Message:
tagname
attribute
value
HTML CSS Script DOM Net Cookies FirePath
ath:

```
.//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>
```

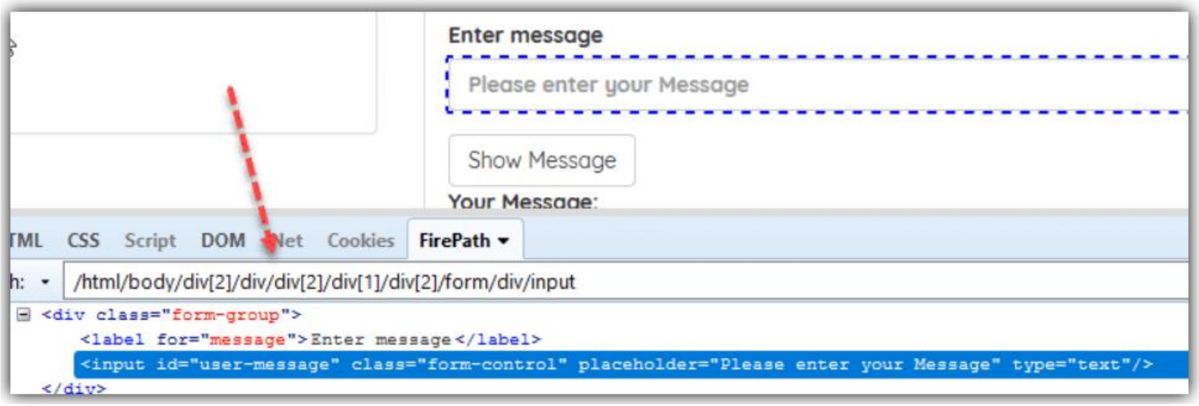
You can also use **class**, **name**, **link text**, and the other attributes to locate an element with XPath as shown above.

Writing Smart XPaths for Complex and Dynamic Elements

Tag – Attribute – Value Trio

Syntax: //tag[@attribute='value']

Example: *//input[@id, 'user-message']*


Enter message
Please enter your Message
Show Message
Your Message:
TML CSS Script DOM Net Cookies FirePath

```
/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>
```

Examples:

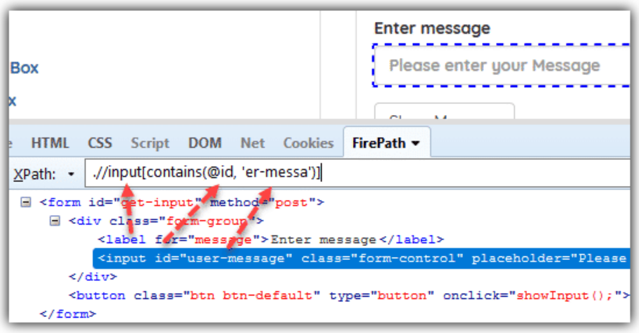
```
//input[@type='send text']

//label[@id='clkBtn']

//input[@value='SEND']

//*[@class='swtestacademy']
--> "*" means, search "swtestacademy" class for all tags.

//a[@href='http://www.swtestacademy.com/']

//img[@src='cdn.medianova.com/images/img_59c4334feaa6d.png']
```

Contains

It is very handy XPath Selenium locator and sometimes it saves the life of a test automation engineer. When an attribute of an element is dynamic, then you can use contains() for the constant part of the web element but also you can use contains() in any condition when you need.

Syntax: //tag[contains(@attribute, 'value')]

Example: *//input[contains(@id, 'er-messa')]*



Box
x
Enter message
Please enter your Message
HTML CSS Script DOM Net Cookies FirePath
XPath:

```
.//input[contains(@id, 'er-messa')]
```

```
<form id="get-input" method="post">
  <div class="form-group">
      <label for="message">Enter message</label>
      <input id="user-message" class="form-control" placeholder="Please
  </div>
  <button class="btn btn-default" type="button" onclick="showInput();">
</form>
```

Examples:

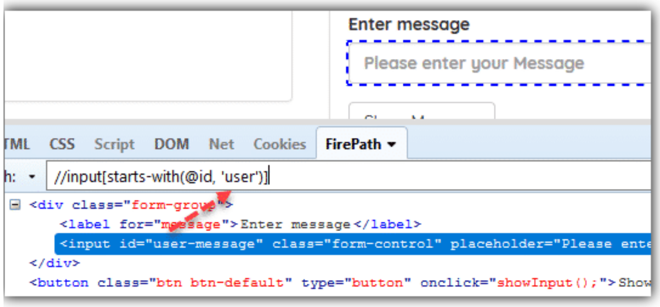
```
//*[contains(@name,'btnClk')]
--> It searches "btnClk" for all name attributes in the DOM.

//*[contains(text(),'here')]
--> It searches the text "here" in the DOM.

//*[contains(@href,'swtestacademy.com')]
--> It searches "swtestacademy.com" link in the DOM.
```

Starts-with

This method checks the starting text of an attribute. It is very handy to use when the attribute value changes dynamically but also you can use th method for non-changing attribute values.

Syntax: //tag[starts-with(@attribute, 'value')]

Example: *//input[starts-with(@id, 'user')]*



Enter message
Please enter your Message
TML CSS Script DOM Net Cookies FirePath
h:

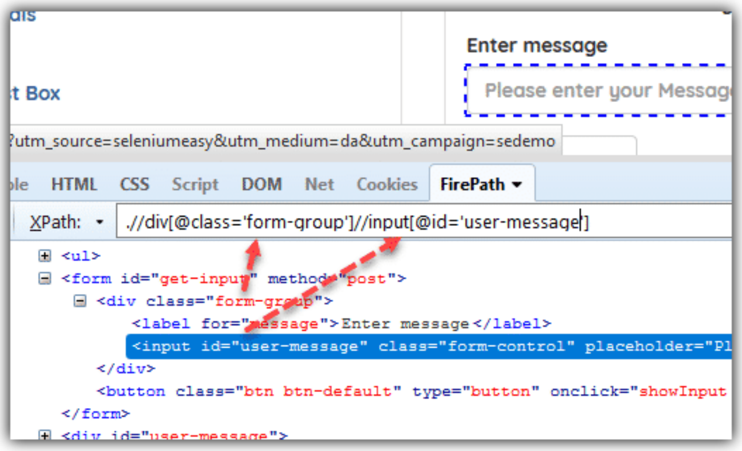
```
//input[starts-with(@id, 'user')]
```

```
<div class="form-group">
    <label for="message">Enter message</label>
    <input id="user-message" class="form-control" placeholder="Please ente
</div>
<button class="btn btn-default" type="button" onclick="showInput();">Show
```

Chained Declarations

We can chain multiple relative XPath declarations with **"//" double slash** to find an element location as shown below.

Example: *//div[@class='form-group']//input[@id='user-message']*


als
t Box
Enter message
Please enter your Messag
?utm_source=seleniumeasy&utm_medium=da&utm_campaign=sedemo
le HTML CSS Script DOM Net Cookies FirePath
XPath:

```
.//div[@class='form-group']//input[@id='user-message']
```

```
⊞ <ul>
⊟ <form id="get-input" method="post">
   ⊟ <div class="form-group">
        <label for="message">Enter message</label>
        <input id="user-message" class="form-control" placeholder="Pl
     </div>
     <button class="btn btn-default" type="button" onclick="showInput
  </form>
⊞ <div id="user-message">
```

Operator "or"

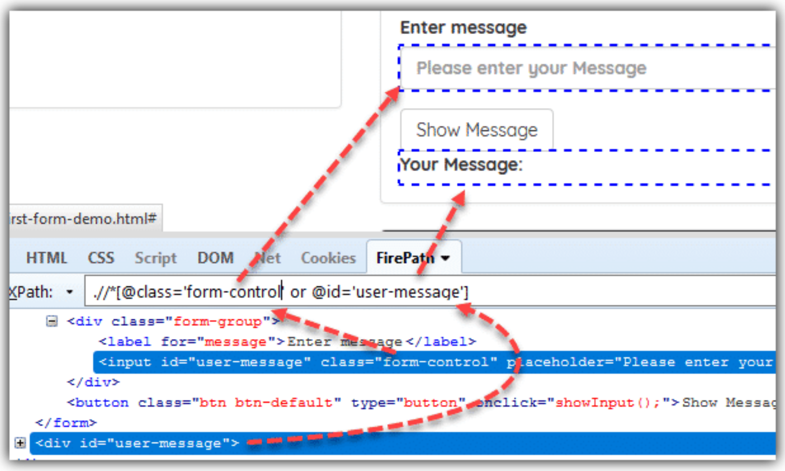
In this method, we use two interrogation conditions such as A and B and return a result-set as shown below:

A	B	Result
False	False	No Element
True	False	Returns A
False	True	Returns B
True	True	Returns Both

"or" is case-sensitive, you should not use capital "OR".

Syntax: //tag[XPath Statement-1 or XPath Statement-2]

Example: *//*[@id='user-message' or @class='form-control']*


Enter message
Please enter your Message
Show Message
Your Message:
rst-form-demo.html#
HTML CSS Script DOM Net Cookies FirePath
XPath:

```
.//*[@class='form-control' or @id='user-message']
```

```
⊟ <div class="form-group">
     <label for="message">Enter message</label>
     <input id="user-message" class="form-control" placeholder="Please enter your
  </div>
  <button class="btn btn-default" type="button" onclick="showInput();">Show Messag
</form>
⊞ <div id="user-message">
```

Operator "and"

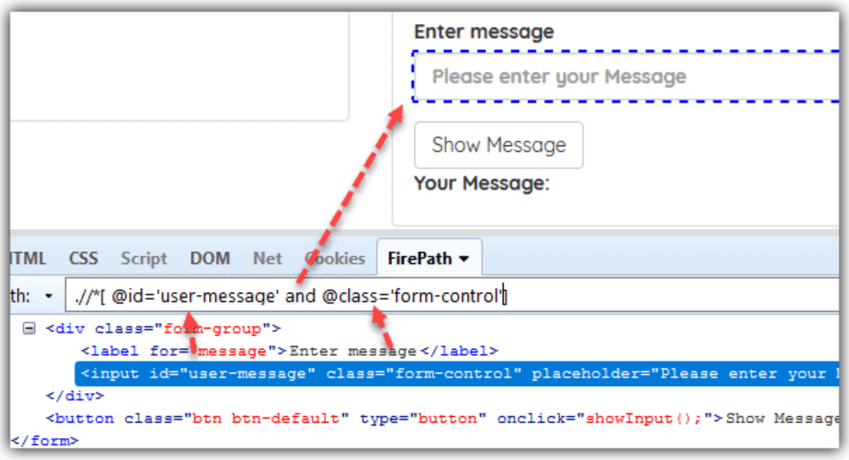
In this method, we use two interrogation conditions such as A and B and return a result-set as shown below:

A	B	Result
False	False	No Element
True	False	No Element
False	True	No Element
True	True	Returns Both

"and" is case-sensitive, you should not use capital "AND".

Syntax: //tag[XPath Statement-1 and XPath Statement-2]

Example: *//*[@id='user-message' and @class='form-control']*


Enter message
Please enter your Message
Show Message
Your Message:
TML CSS Script DOM Net Cookies FirePath

```
th: .//*[ @id='user-message' and @class='form-control']
```

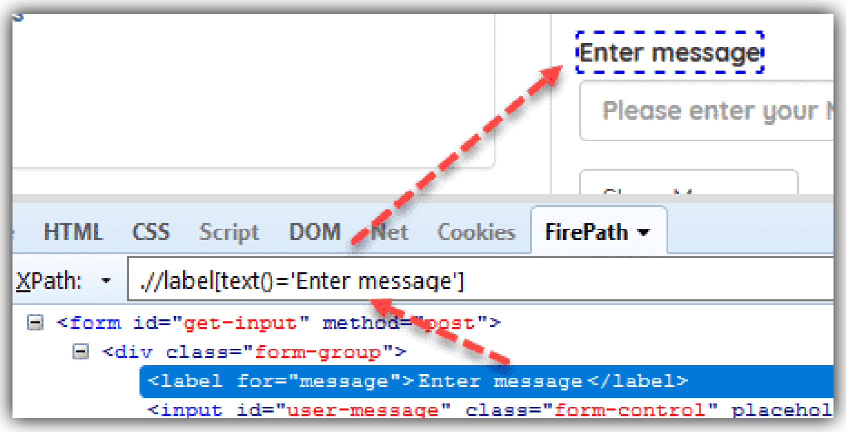
```
⊟ <div class="form-group">
      <label for="message">Enter message</label>
      <input id="user-message" class="form-control" placeholder="Please enter your
   </div>
   <button class="btn btn-default" type="button" onclick="showInput();">Show Message
</form>
```

Text

We can find an element with its exact text.

Syntax: //tag[text()='text value']

Example: *.//label[text()='Enter message']*



```
XPath: .//label[text()='Enter message']
```

```
⊟ <form id="get-input" method="post">
    ⊟ <div class="form-group">
          <label for="message">Enter message</label>
          <input id="user-message" class="form-control" placehol
```

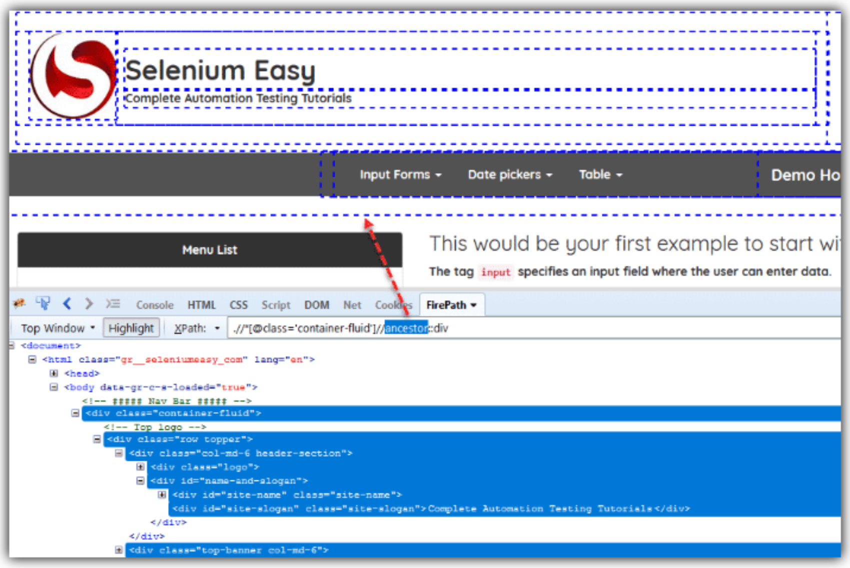
Ancestor

It finds the element before the ancestor statement and set it as a top node and then starts to **find the elements in that node**. In below example,

1- First, it finds the class which id is "container-fluid"

2- Then, starts to find div elements in that node.

Example: *//*[@class='container-fluid']//ancestor::div*



This would be your first example to start wi

The tag `input` specifies an input field where the user can enter data.

```
.//*[@class='container-fluid']//ancestor::div
```

```
<document>
  <html class="gr__seleniumeasy_com" lang="en">
    <head>
    <body data-gr-c-s-loaded="true">
      <!-- ##### Nav Bar ##### -->
      <div class="container-fluid">
        <!-- Top logo -->
        <div class="row topper">
          <div class="col-md-6 header-section">
            <div class="logo">
            <div id="name-and-slogan">
              <div id="site-name" class="site-name">
              <div id="site-slogan" class="site-slogan">Complete Automation Testing Tutorials</div>
            </div>
          </div>
          <div class="top-banner col-md-6">
```

You can select specific div groups by changing div depths as shown below.

.//[@class='container-fluid']//ancestor::div[1] – Returns 13 nodes*
.//[@class='container-fluid']//ancestor::div[2] – Returns 7 nodes*
.//[@class='container-fluid']//ancestor::div[3] – Returns 5 nodes*
.//[@class='container-fluid']//ancestor::div[4] – Returns 3 nodes*
.//[@class='container-fluid']//ancestor::div[5] – Returns 1 node*

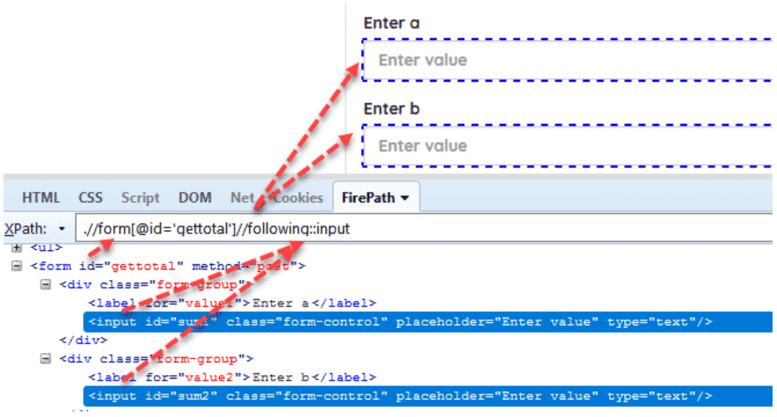
Following

Starts to locate elements **after the given parent node**. It finds the element before the following statement and set as the top node and then starts to find **all elements after that node**. In below example,

1- First, it finds the form which id is "gettotal"

2- Then, starts to find all input elements after that node.

Example: *.//form[@id='gettotal']//following::input*



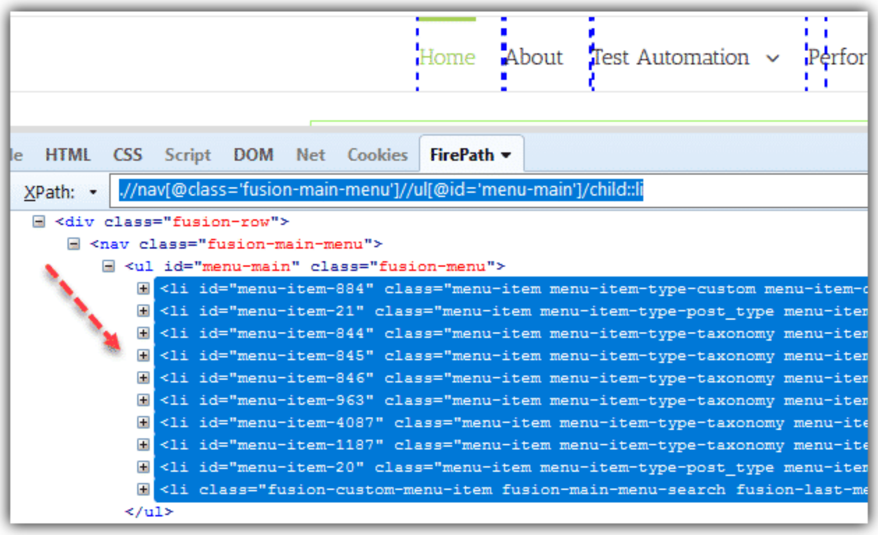
XPath: .//form[@id='gettotal']//following::input

```
<ul>
<form id="gettotal" method="post">
  <div class="form-group">
    <label for="value1">Enter a</label>
    <input id="sum1" class="form-control" placeholder="Enter value" type="text"/>
  </div>
  <div class="form-group">
    <label for="value2">Enter b</label>
    <input id="sum2" class="form-control" placeholder="Enter value" type="text"/>
```

Child

Selects all children elements of the current node.

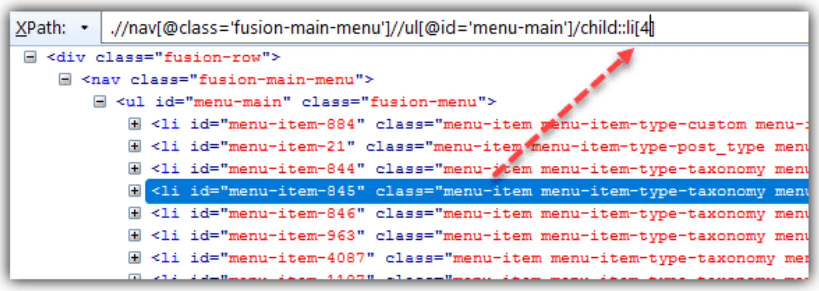
Example: *//nav[@class='fusion-main-menu']//ul[@id='menu-main']/child::li*



```
.//nav[@class='fusion-main-menu']//ul[@id='menu-main']/child::li
```

```
⊟ <div class="fusion-row">
    ⊟ <nav class="fusion-main-menu">
        ⊟ <ul id="menu-main" class="fusion-menu">
            ⊞ <li id="menu-item-884" class="menu-item menu-item-type-custom menu-item-c
            ⊞ <li id="menu-item-21" class="menu-item menu-item-type-post_type menu-iter
            ⊞ <li id="menu-item-844" class="menu-item menu-item-type-taxonomy menu-iter
            ⊞ <li id="menu-item-845" class="menu-item menu-item-type-taxonomy menu-iter
            ⊞ <li id="menu-item-846" class="menu-item menu-item-type-taxonomy menu-iter
            ⊞ <li id="menu-item-963" class="menu-item menu-item-type-taxonomy menu-iter
            ⊞ <li id="menu-item-4087" class="menu-item menu-item-type-taxonomy menu-ite
            ⊞ <li id="menu-item-1187" class="menu-item menu-item-type-taxonomy menu-ite
            ⊞ <li id="menu-item-20" class="menu-item menu-item-type-post_type menu-iter
            ⊞ <li class="fusion-custom-menu-item fusion-main-menu-search fusion-last-me
          </ul>
```

You can also **select the required "li" element by using li[1], li[2], li[3]**, etc. syntax as shown below.



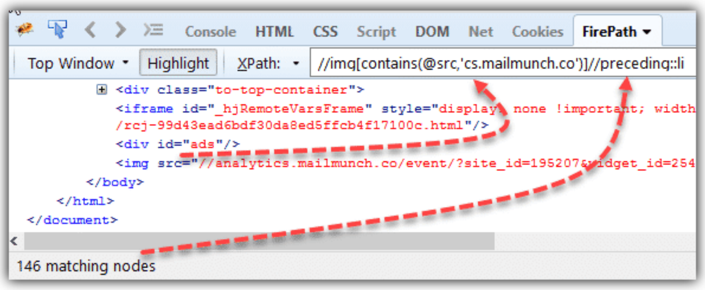
```
.//nav[@class='fusion-main-menu']//ul[@id='menu-main']/child::li[4]
```

```
⊟ <div class="fusion-row">
    ⊟ <nav class="fusion-main-menu">
        ⊟ <ul id="menu-main" class="fusion-menu">
            ⊞ <li id="menu-item-884" class="menu-item menu-item-type-custom menu-i
            ⊞ <li id="menu-item-21" class="menu-item menu-item-type-post_type menu
            ⊞ <li id="menu-item-844" class="menu-item menu-item-type-taxonomy menu
            ⊞ <li id="menu-item-845" class="menu-item menu-item-type-taxonomy menu
            ⊞ <li id="menu-item-846" class="menu-item menu-item-type-taxonomy menu
            ⊞ <li id="menu-item-963" class="menu-item menu-item-type-taxonomy menu
            ⊞ <li id="menu-item-4087" class="menu-item menu-item-type-taxonomy me
```

Preceding

Select all nodes that come before the current node. I give an example on swtestacademy. We will find all "li" elements in the homepage. First, we will locate the bottom element, then use preceding with "li" to find all "li" elements as shown below.

Example: *//img[contains(@src,'cs.mailmunch.co')]//preceding::li*



```
//img[contains(@src,'cs.mailmunch.co')]//preceding::li
```

```
        ⊞ <div class="to-top-container">
          <iframe id="_hjRemoteVarsFrame" style="display: none !important; width
          /rcj-99d43ead6bdf30da8ed5ffcb4f17100c.html"/>
          <div id="ads"/>
          <img src="//analytics.mailmunch.co/event/?site_id=195207&widget_id=254
      </body>
   </html>
</document>
```

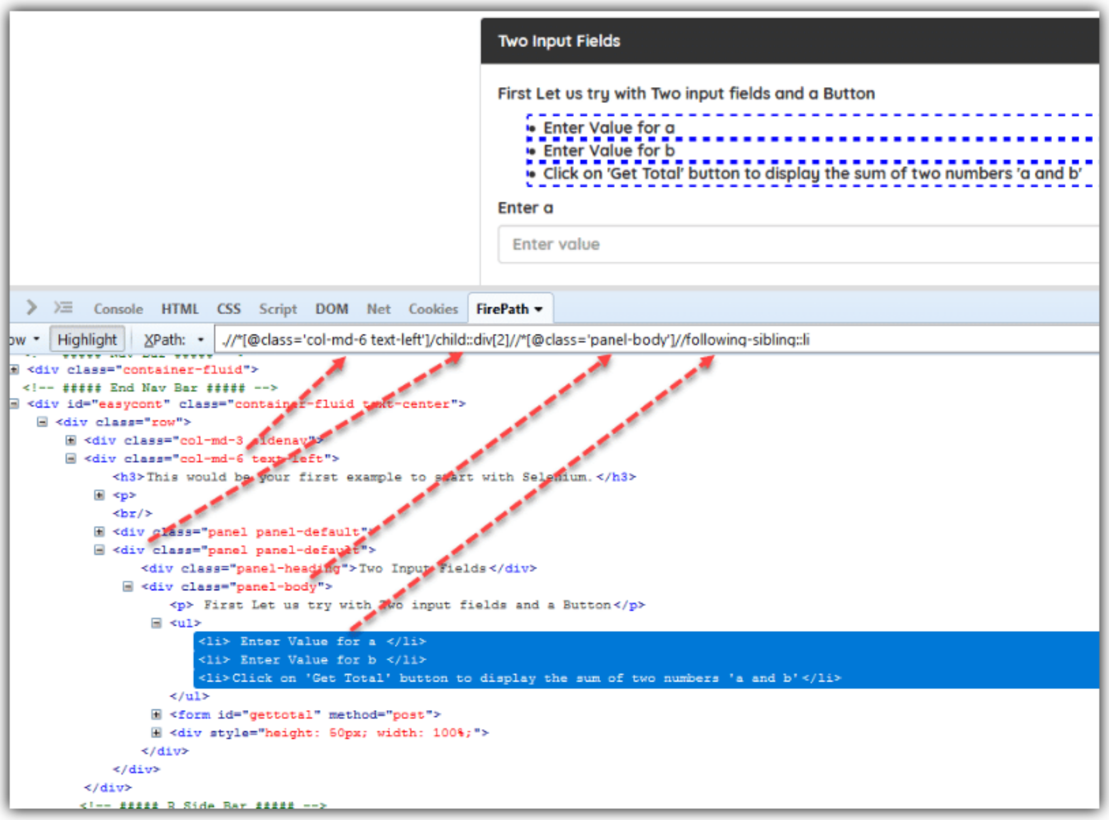
146 matching nodes

Also, you can use [1], [2], etc. to select a specific element in the preceding element list.

Following-sibling

Select the following siblings of the context node.

Example: *//*[@class='col-md-6 text-left']/child::div[2]//*[@class='panel-body']//following-sibling::li*



First Let us try with Two input fields and a Button

- Enter Value for a
- Enter Value for b
- Click on 'Get Total' button to display the sum of two numbers 'a and b'

Enter a

Enter value

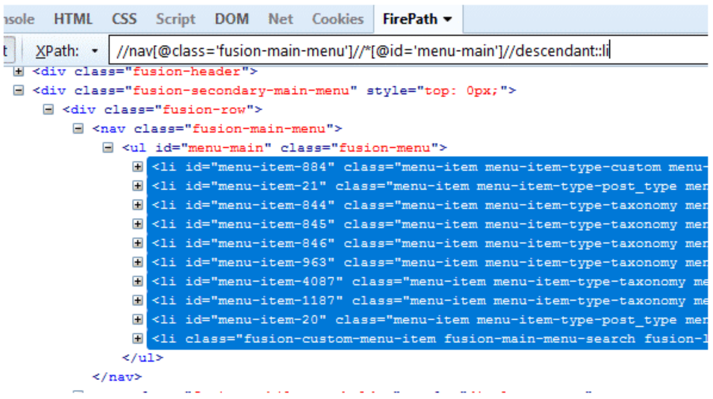
```
.//*[@class='col-md-6 text-left']/child::div[2]//*[@class='panel-body']//following-sibling::li
```

```
<div class="container-fluid">
 <!-- ##### End Nav Bar ##### -->
<div id="easycont" class="container-fluid text-center">
    <div class="row">
        <div class="col-md-3 sidenav">
        <div class="col-md-6 text-left">
            <h3>This would be your first example to start with Selenium.</h3>
            <p>
            <br/>
            <div class="panel panel-default">
            <div class="panel panel-default">
                <div class="panel-heading">Two Input Fields</div>
                <div class="panel-body">
                    <p> First Let us try with Two input fields and a Button</p>
                    <ul>
                        <li> Enter Value for a </li>
                        <li> Enter Value for b </li>
                        <li>Click on 'Get Total' button to display the sum of two numbers 'a and b'</li>
                    </ul>
                    <form id="gettotal" method="post">
                    <div style="height: 50px; width: 100%;">
                </div>
            </div>
        </div>
      <!-- ##### R Side Bar ##### -->
```

Descendant

Identifies and returns all the element descendants to current element which means traverse down under the current element's node. Below, the XPath returns all "li" elements under the "menu-main".

Example: *//nav[@class='fusion-main-menu']//*[@id='menu-main']//descendant::li*



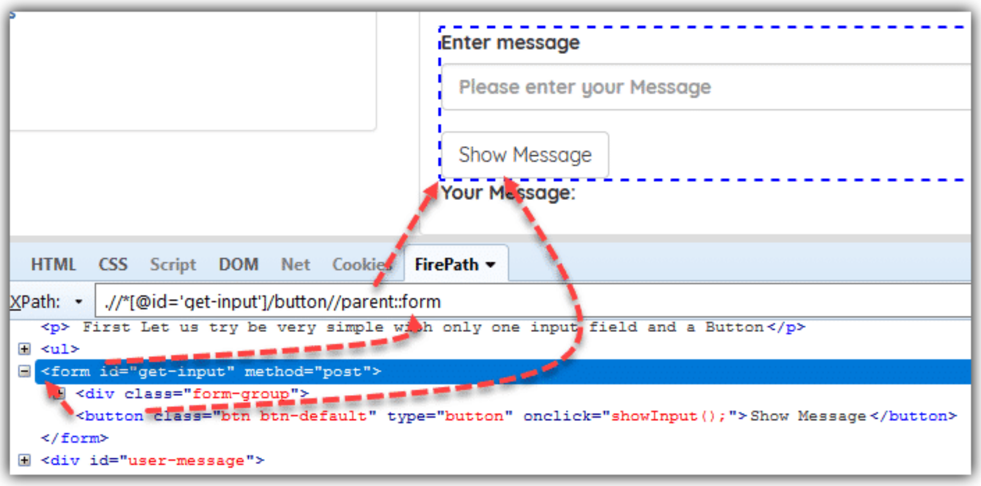
```
//nav[@class='fusion-main-menu']//*[@id='menu-main']//descendant::li
```

```
<div class="fusion-header">
<div class="fusion-secondary-main-menu" style="top: 0px;">
    <div class="fusion-row">
        <nav class="fusion-main-menu">
            <ul id="menu-main" class="fusion-menu">
                <li id="menu-item-884" class="menu-item menu-item-type-custom menu-
                <li id="menu-item-21" class="menu-item menu-item-type-post_type men
                <li id="menu-item-844" class="menu-item menu-item-type-taxonomy men
                <li id="menu-item-845" class="menu-item menu-item-type-taxonomy men
                <li id="menu-item-846" class="menu-item menu-item-type-taxonomy men
                <li id="menu-item-963" class="menu-item menu-item-type-taxonomy men
                <li id="menu-item-4087" class="menu-item menu-item-type-taxonomy me
                <li id="menu-item-1187" class="menu-item menu-item-type-taxonomy me
                <li id="menu-item-20" class="menu-item menu-item-type-post_type men
                <li class="fusion-custom-menu-item fusion-main-menu-search fusion-l
            </ul>
        </nav>
```

Parent

Returns the parent of the current node as shown in the below example.

Example: *.//*[@id='get-input']/button//parent::form*



XPath: .//*[@id='get-input']/button//parent::form

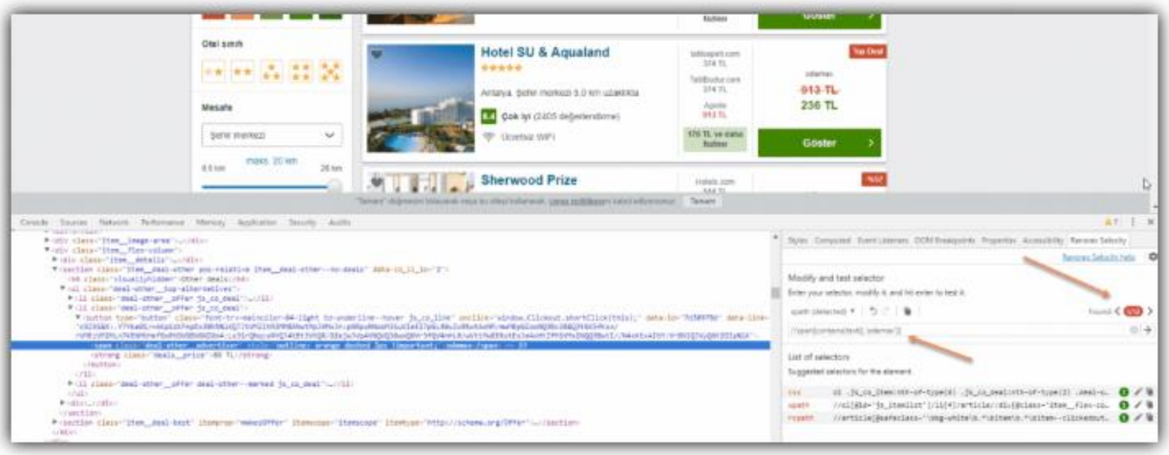
```
  <p> First Let us try be very simple wish only one input field and a Button</p>
⊞ <ul>
⊟ <form id="get-input" method="post">
    ⊞ <div class="form-group">
      <button class="btn btn-default" type="button" onclick="showInput();">Show Message</button>
  </form>
⊞ <div id="user-message">
```

Locate an Element inside Array of Elements

In Trivago website, lets search "Antalya" keyword. Then, find the first Odamax hotel with XPath.

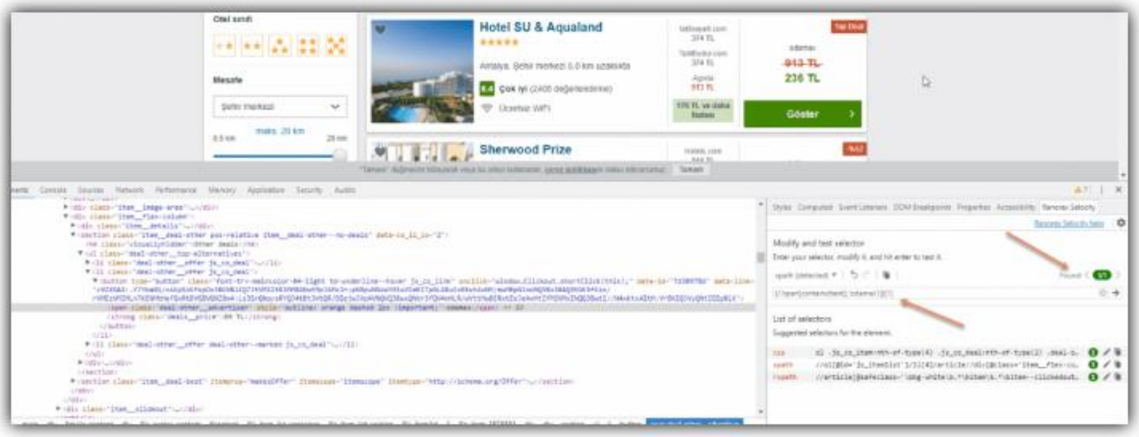
First, we can find all Odamax hotels by using its text with below XPath:

//span[contains(text(),'odamax')]



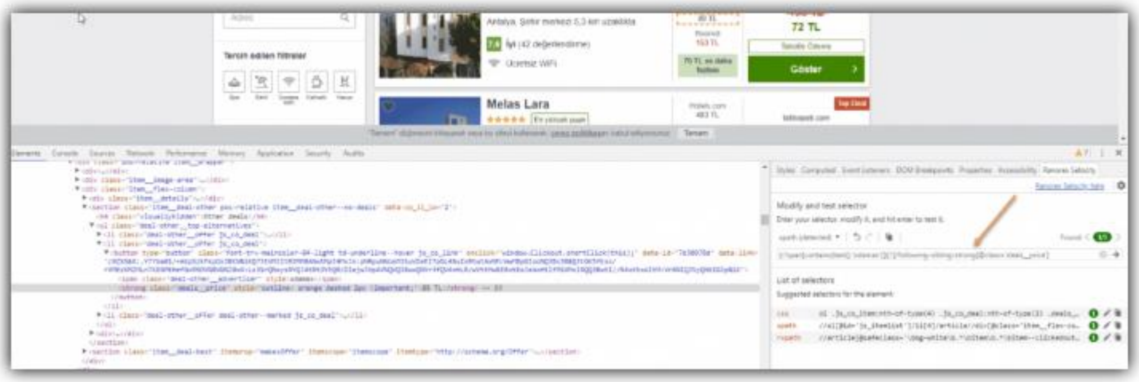
Above XPath returns many Odamax hotel's, we can select the first one with below XPath expression:

(//span[contains(text(),'odamax')])[1]



You can also continue to search and find the related hotel's price element with below XPath:

(//span[contains(text(),'odamax')])[1]/following-sibling::strong[@class='deals__price']


72 TL
Göster
Melas Lara
Modify and test selector
List of selectors