What is XPath?

* XPath stands for XML Path Language
* XPath uses "path like" syntax to identify and navigate nodes in an XML document
* XPath contains over 200 built-in functions
* XPath can be used to navigate through elements and attributes in an XML document.

XPath Path Expressions

* XPath uses path expressions to select nodes or node-sets in an XML document.
* These path expressions look very much like the path expressions you use with traditional computer file systems:



## XPath Terminology

### **Nodes**

In XPath, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document nodes.

XML documents are treated as trees of nodes. The topmost element of the tree is called the root element.

Look at the following XML document:

<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  <book>  
    <title lang="en">Harry Potter</title>  
    <author>J K. Rowling</author>  
    <year>2005</year>  
    <price>29.99</price>  
  </book>  
</bookstore>

Example of nodes in the XML document above:

<bookstore> (root element node)  
  
<author>J K. Rowling</author> (element node)  
  
lang="en" (attribute node)

### **Atomic values**

Atomic values are nodes with no children or parent.

Example of atomic values:

J K. Rowling  
  
"en"

### **Items**

Items are atomic values or nodes.

## Relationship of Nodes

### **Parent**

Each element and attribute has one parent.

In the following example; the book element is the parent of the title, author, year, and price:

<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>

### **Children**

Element nodes may have zero, one or more children.

In the following example; the title, author, year, and price elements are all children of the book element:

<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>

### **Siblings**

Nodes that have the same parent.

In the following example; the title, author, year, and price elements are all siblings:

<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>

### **Ancestors**

A node's parent, parent's parent, etc.

In the following example; the ancestors of the title element are the book element and the bookstore element:

<bookstore>  
  
<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>  
  
</bookstore>

### **Descendants**

A node's children, children's children, etc.

In the following example; descendants of the bookstore element are the book, title, author, year, and price elements:

<bookstore>  
  
<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>  
  
</bookstore>

|  |  |
| --- | --- |
| **Expression** | **Description** |
| *nodename* | Selects all nodes with the name "*nodename*" |
| / | Selects from the root node |
| // | Selects nodes in the document from the current node that match the selection no matter where they are |
| . | Selects the current node |
| .. | Selects the parent of the current node |
| @ | Selects attributes |

|  |  |
| --- | --- |
| **Path Expression** | **Result** |
| bookstore | Selects all nodes with the name "bookstore" |
| /bookstore | Selects the root element bookstore  **Note:** If the path starts with a slash ( / ) it always represents an absolute path to an element! |
| //bookstore/book | Selects all book elements that are children of bookstore |
| //book | Selects all book elements no matter where they are in the document |
| //bookstore//book | Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element |
| //@lang | Selects all attributes that are named lang |

|  |  |
| --- | --- |
| **Path Expression** | **Result** |
| //bookstore/book[1] | Selects the first book element that is the child of the bookstore element.  **Note:** In IE 5,6,7,8,9 first node is[0], but according to W3C, it is [1]. To solve this problem in IE, set the SelectionLanguage to XPath:  *In JavaScript: xml*.setProperty("SelectionLanguage","XPath"); |
| //bookstore/book[last()] | Selects the last book element that is the child of the bookstore element |
| //bookstore/book[last()-1] | Selects the last but one book element that is the child of the bookstore element |
| //bookstore/book[position()<3] | Selects the first two book elements that are children of the bookstore element |
| //title[@lang] | Selects all the title elements that have an attribute named lang |
| //title[@lang='en'] | Selects all the title elements that have a "lang" attribute with a value of "en" |
| //bookstore/book[price>35.00] | Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00 |
| //bookstore/book[price>35.00]/title | Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00 |

Selecting Unknown Nodes

XPath wildcards can be used to select unknown XML nodes.

|  |  |
| --- | --- |
| **Wildcard** | **Description** |
| \* | Matches any element node |
| @\* | Matches any attribute node |
| node() | Matches any node of any kind |

In the table below we have listed some path expressions and the result of the expressions:

|  |  |
| --- | --- |
| **Path Expression** | **Result** |
| /bookstore/\* | Selects all the child element nodes of the bookstore element |
| //\* | Selects all elements in the document |
| //title[@\*] | Selects all title elements which have at least one attribute of any kind |

Selecting Several Paths

By using the | operator in an XPath expression you can select several paths.

In the table below we have listed some path expressions and the result of the expressions:

|  |  |
| --- | --- |
| **Path Expression** | **Result** |
| //book/title | //book/price | Selects all the title AND price elements of all book elements |
| //title | //price | Selects all the title AND price elements in the document |
| /bookstore/book/title | //price | Selects all the title elements of the book element of the bookstore element AND all the price elements in the document |

XPath Axes

An axis represents a relationship to the context (current) node, and is used to locate nodes relative to that node on the tree.

|  |  |
| --- | --- |
| **AxisName** | **Result** |
| ancestor | Selects all ancestors (parent, grandparent, etc.) of the current node |
| ancestor-or-self | Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself |
| attribute | Selects all attributes of the current node |
| child | Selects all children of the current node |
| descendant | Selects all descendants (children, grandchildren, etc.) of the current node |
| descendant-or-self | Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself |
| following | Selects everything in the document after the closing tag of the current node |
| following-sibling | Selects all siblings after the current node |
| namespace | Selects all namespace nodes of the current node |
| parent | Selects the parent of the current node |
| preceding | Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes |
| preceding-sibling | Selects all siblings before the current node |
| self | Selects the current node |

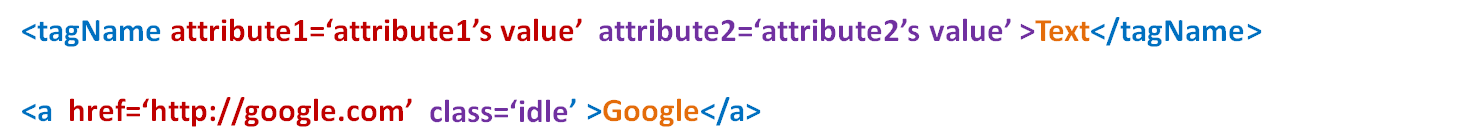
### **Examples**

|  |  |
| --- | --- |
| **Example** | **Result** |
| child::book | Selects all book nodes that are children of the current node |
| attribute::lang | Selects the lang attribute of the current node |
| child::\* | Selects all element children of the current node |
| attribute::\* | Selects all attributes of the current node |
| child::text() | Selects all text node children of the current node |
| child::node() | Selects all children of the current node |
| descendant::book | Selects all book descendants of the current node |
| //ancestor::book | Selects all book ancestors of the current node |
| //ancestor-or-self::book | Selects all book ancestors of the current node - and the current as well if it is a book node |
| child::\*/child::price | Selects all price grandchildren of the current node |

Xpath in Selenium Webdriver

**Table of content**1 Xpath  
2 Parent Reference xpath  
3 Group Index xpath  
4 text() xpath  
5 Wild card in xpath  
6 Dependent & Independent xpath  
7 Contains function  
8 Starts-With function  
9 Last()  
10 Position Function in xpath  
11 CaSe in-sensitive Xpath  
12 Length function  
13 Relational Xpath  
14 Axes in xpath  
15 Forward Axes  
16 Reverse Axes  
17 Normalize Space xpath

**Xpath Syntax**  
  
  
  
****

**HTML code Syntax**  
  
****  
  
HTML code can have n-number of attributes, text and closing tag is not mandatory for a few elements  
  
*There are two kinds of xpaths*  
  1. Absolute XPath  
  2. Relative XPath  
  
**Absolute Xpath**  
  
****  
  
**/ -**point to the first node on the HTML document, it is HTML tag  
Note: we are not going to focus on absolute XPath.  
  
**Relative Xpath**  
  
  
  
****  
  
**// -**points to any node in the webpage  
  
**tagName -**tag name is nothing but the name which is present after the < (angular bracket)  
  
**attribute -**whatever is present inside < and > bracket except tagname is an attribute, any number of attributes can present in HTML code  
  
**attribute's value -**it is corresponding value to the attribute, sometimes for boolean attribute developers may not specify any value; in those cases, HTML takes 'true' as default value.  
  
**Text -**text is the value present inside > and <  
  
Now let form the xpath for the above html code.  
**//a[@class='idle']**

**Example-- //book[@category="cooking"]**

//button[@value="Strawberry"]

## Xpath with Tagname

tagname : //button

//button

## Xpath with Index

Syntax for Xpath with Index : //tagName[index]

**Xpath for the elements :**  
Bluberry button- //button[1]  
Banana button - //button[2]  
Strawberry button -//button[3]

## Xpath with Attribute

Xpath with Attribute ://tagName[@attribute='attribute value']

Xpath with multiple Attributes://tagName[@attrib='attrib value'][@attrib2='attrib2 value']...

Xpath with Attribute and Index://tagName[@attribute='attribute value'][index]

//button[@name='banana']

//button[@id="firstButton"] 01\_xpatth%20\_locator1.html

## Xpath With Parent Reference

We cannot expect an HTML element to have different or uniques properties all the time. Sometimes there is a chance that every element may have the same kind of attributes, In those cases, we cannot use Xpath with Attribute in selenium webdriver  
  
To handle such kind of cases we may need to take help of the parent element to find our actual element  
  
Store the below code in HTML file and open it in chrome

<html>

<body>

<div id="berry">

<button type="button">Blueberry</button><br><br>

<button type="button">Banana</button><br><br>

<button type="button">Strawberry</button><br><br>

</div>

<div id="fruit">

<button type="button">Apple</button><br><br>

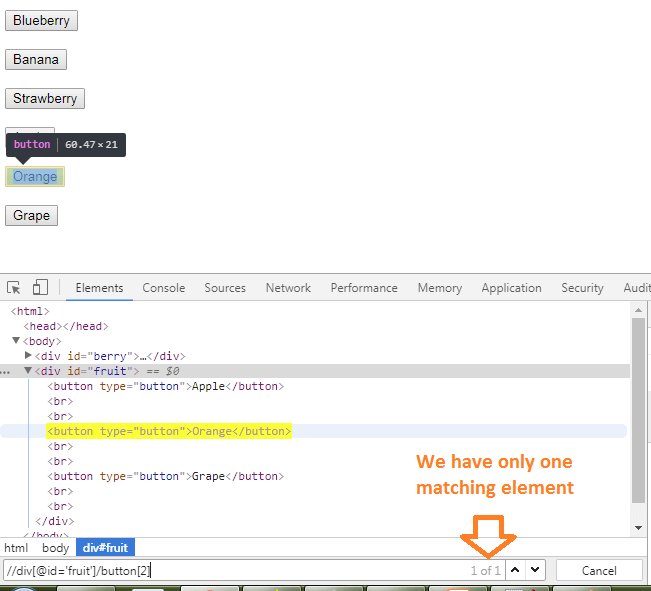
<button type="button">Orange</button><br><br>

<button type="button">Grape</button><br><br>

</div>

</body>

</html>

**The syntax for Xpath with parent and child**  
  
**parent-child-xpath**  
  
For Orange element we have to refetr it parent div which has id attribute as **fruit** Xpath for the Orange: //div[@id='fruit']/button[2]  
  
  
  
We have only one match for the XPath we have written.  
  
**Explanation for Xpath : //div[@id='fruit']/button[2]**  
// - look for any node which has 'div' as tagname and id as fruit, look for immediate child(/) node which has tagname as button and at the index of 2.

## Xpath with Group Index

Sometimes we may have to handle the elements with the XPath index, but the index may give more than one match, which are under different parents; in these situations, the index might not help you. We may have to use Group index in this kind of scenarios  
  
Group index puts all matches into a list and gives indexes them. So here we will not have any duplicates matches

Syntax : (//tagName)[index]

We have to use parenthesis to make an xpath into group XPath after it indexes the XPath  
  
**Store below HTML code into HTML file :**

<html>

<body>

<div id="fruit"><br><br><br>

<button type="button">Blueberry</button><br><br>

<button type="button" >Banana</button><br><br>

<button type="button">Strawberry</button><br><br>

</div>

<div id="fruit">

<button type="button">Apple</button><br><br>

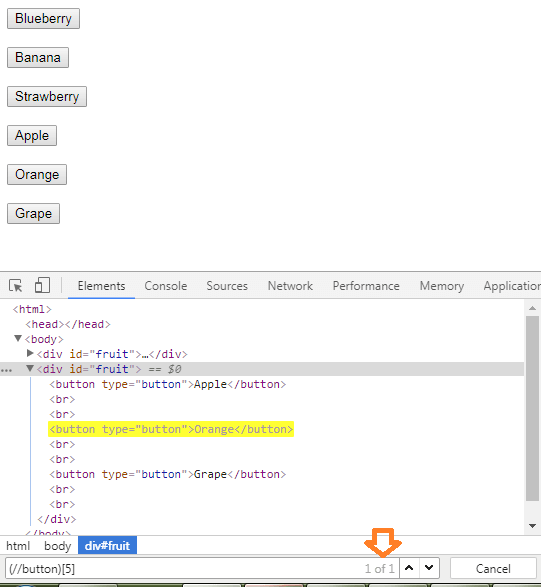
<button type="button" >Orange</button><br><br>

<button type="button">Grape</button><br><br>

</div>

</body>

</html>

Let's write XPath for Orange : **(//button)[5]**  
  


## text() function in Xpath

There will be situations, where you may not be able to use any HTML property other than text present in the element  
  
text() function helps us to find the element based on the text present in the element, text () function is case sensitive

<button type="button">Blueberry</button><br><br>​

In the above code, the text is Blueberry, and we can write XPath using text () like below

xpath with text : //button[text()='Bluberry']

Note: we use **@** sign for attributes, functions do not need **@** sign  
  
We can also match element(s) which have text in them with below XPath

xpath with text ://button[text()]

://button/text()

## Wild card Character with Xpath in Selenium python

\* -is the one of most used wild card character with xpath in selenium webdriver, we can use it instead of the tag name and attribute  
  
//\* - matches all the elements present in the HTML (including HTML)  
  
//div/\* - matches all the immediate element(s) inside the **div** tag  
  
//input[@\*] - matches all the element(s) with **input** tag and have at least one attribute, attribute value may or may not present  
  
//\*[@\*] - matches all the element(s) which have at least one attribute.

## contains() function in Xpath

contains() function helps user to fins the element with partial values, or dynamically chaning values, contains verifies matches with portion of the value

contains function ://xpath[contains(@attribute, 'attribute value')]

//xpath[contains(@text(), 'attribute value')]

Example of below html:

<html>

<body>

<div id="fruit"><br><br><br><br><br><br><br><br><br><br><br>

<button type="button">Blue berry1234</button><br><br>

<button type="button" >Banana</button><br><br>

<button type="button">Straw</button><br><br>

<button type="button">berry</button><br><br>

<button type="button">Straw berry</button><br><br>

</div>

</body>

</html>

Xpath for the Blueberry : **//button[contains(text(),'Blue')]**  
Xpath for the Banana : **//button[contains(text(),'Ban')]**  
  
***More Complex items:***  
In the same way, if you try to find XPath for Straw berry with **//button[contains(text(),'Straw')],** it finds the element with text Straw as well.  
  
If you try with berry, you may get 'berry' element. So how to find the Straw berry button.  
  
We can combne more than one contains functions like : **//xpath[contains(text(), 'text1')][contains(text(), 'text2')]**  
Xpath for Strawberry is : **//button[contains(text(),'Straw')][contains(text(), 'berry')]**  
  
Not only for text you can apply contains function for other properties as well Eg : **//button[contains(@type,'but')]**

## starts-with function xpath

Starts-with function matches the elements which property starting value

**syntax ://xpath[starts-with(@attribute,'starting value')]**

**<button type="button">Straw berry</button><br><br>**

**Xpath for the Strawberry element ://button[starts-with(text(), 'Straw')]**

## ends-with function xpath

Ends-with function matches the elements properties ending value

**syntax ://xpath[ends-with(@attribute,'ending value')]**

**<button type="button">Straw berry</button><br><br>**

Xpath for the Strawberry element : **//button[ends-with(text(), 'berry')]**

## Last() function in Xpath

By default, automation tools take the first instance of the match; also, if we want to achieve the first element, we can use index [1]. But in some pages, we may not be able to see how many matches are present when the page is loading or on a dynamic page.  
  
last() function in Xpath helps the user to find the last match of the element.

**last function : //xpath[last()]**

*take the example of below HTML code*

**<html>**

**<body>**

**<div id="fruit"><br><br><br>**

**<button type="button">Blueberry</button><br><br>**

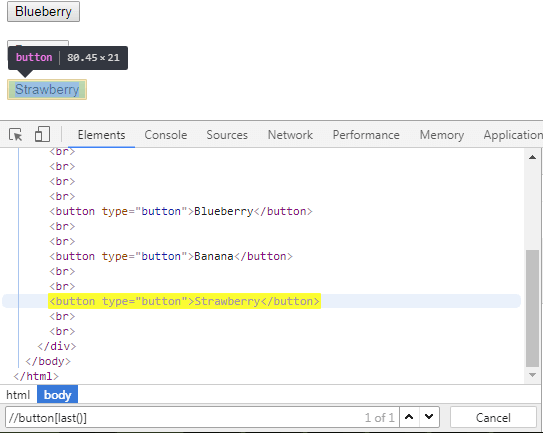
**<button type="button" >Banana</button><br><br>**

**<button type="button">Strawberry</button><br><br>**

**</div>**

**</body>**

**</html>**

In the above, if we want to write xpath for the last element, it is easy we can say use index [3], but if the application is very large or dynamic, we cannot say how many elements are going to present.  
  
So let's use last function in xpath : **//button[last()]** - points to the Strawberry button.  
  


## Position function in Xpath

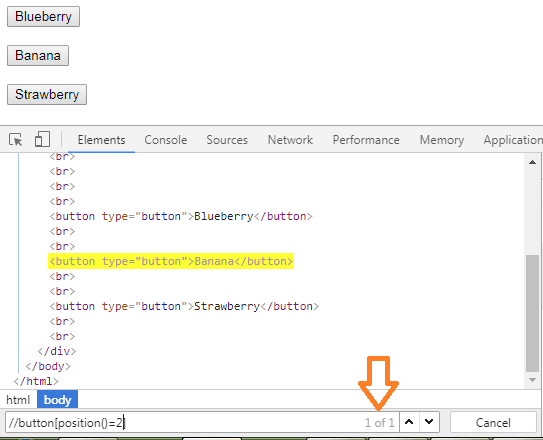
Position function helps the user to get the match at a particular index, using position we can get elements which are less than the position or greater than the position as well.

**position function ://xpath[position()=2]**

**://xpath[position()<2]**

**://xpath[position()>2]**

**://xpath[position()=<2]] ...**

**Example : //button[position()=2]**

## CaSe in-sensitive Xpath in selenium webdriver

Sometimes we may have a situation where we have found the element based on the attribute. We can use @ method for an attribute, but if the attribute values change every time lower to upper case or mix case value when page refreshes, in this @ method may not help us.  
  
During such kind of situations, we must ignore the case(UPPER/lower). Below is the syntax to match the elements by ignoring case, the translate method helps us to perform this.

**Syntax :// tagname[text(), 'sourceCaseContent', 'targetCaseContent'), 'value']**

**tagname** - is the HTML tag used for the element like the label, a, span, div..  
  
**text()** - text() value present in the element.  
  
**sourceCaseContent** - We have to pass the Letter(s) which all we want to convert 'ABCD' so on.. we can also give only a few letters like 'agk' (sourceCaseContent could be UPPER/lower case)  
  
**targetCaseContent** - We have to pass the Letter(s) to which we want to convert the SourceCaseContent, it could be in any case  
  
**value** - the target value which we want to compare it could be any value, but it should be in target case (if the target case is UPPER then the value also should be in upper

**Html code : <label id="aBcK" name='p'>SEleNiuM</label>**

**Xpath ://label[contains(translate(text(),'CDEILMNSU','cdeilmnsu'),'selenium')]**

[HtmlUnitDriver for headless execution in Selenium webdriver](https://chercher.tech/java/headless-browsers-selenium-webdriver#htmlunit)

## Attribute value's Length xpath in selenium

We can [find the element](https://chercher.tech/java/find-elements-in-selenium) based on the attribute value/text length in selenium, the string-length() method helps us to form the xpath based on the element's attribute length.

**yntax :// tagname[string-length(@attibute's name/text)= expectedLength]**

**tagname** - is the HTML tag used for the element like the label, a, span, div..  
  
**@attribute's name** - any attribute present in the element like id, name, src, href... text **()** - text() value present in the element.  
  
**expectedLength**- numeric expected length of the attribute value or text value

**Html code : <label id="twinkie" name='p'>selenium</label>**

**Xpath ://label[string-length(@id) = 7]**

**Xpath ://label[string-length(text()) = 8]**

**twinkie - 7 letters**

**selenium - 8 letters**

## Relational value Xpath in selenium webdriver

We can form xpath based on the numeric attribute value / text present in the element with relational operators. For example, we can find the elements which have numeric text greater than 40 or less than 70 like so

**Syntax :// tagname[@atrribute/function > expectedValue]**

**@atrribute/function**- should result in numeric value  
  
**expectedValue** - Must be a numeric value

**Html code : <label id="50" name='p'>30</label>**

**Xpath 1 ://label[text()>20]**

**Xpath 2 ://label[@id()<70]**

**Xpath 3 ://label[@id()<70][text()>20]**

## Axes in Xpath in selenium webdriver

XPath axes are used to identify elements that periodically change or refreshes it attributes by their relationship like parent, child, sibling, based on the independent element, whose properties do not change.  
  
Axes refer to the node on which elements are lying relative to an independent element. We could traverse in both forward and reverse directions.  
  
**Forward Axis :**

* *self*
* *attribute*
* *child*
* *descendant*
* *descendant-or-self*
* *following-sibling*
* *following*

**Reverse Axis :**

* *parent*
* *ancestor*
* *preceding-sibling*
* *preceding*
* *ancestor-or-self*

***Save below code as HTML file***

**<html>**

**<head></head>**

**<body>**

**<div name='username'>**

**<label id="user">Username</label>**

**<input id="username" type="text">**

**</div>**

**<div name='password'>**

**<label id="pass">Password</label>**

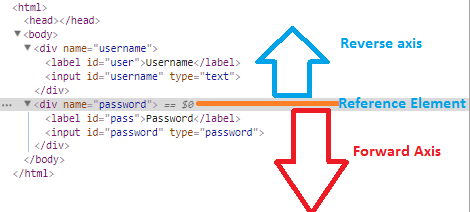
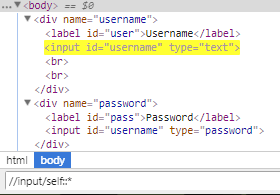
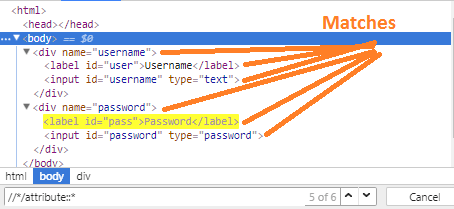
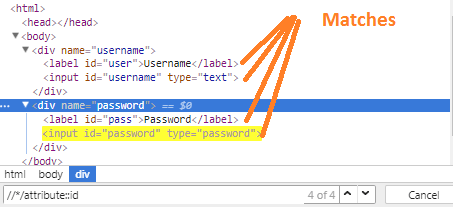
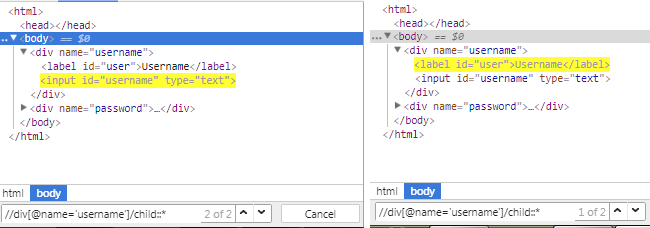
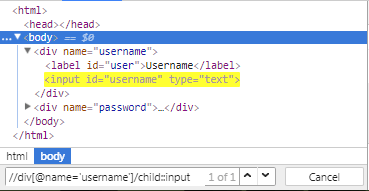
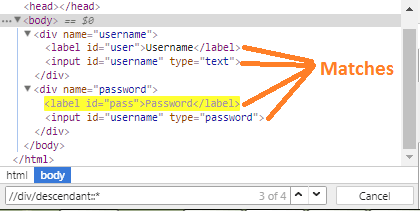
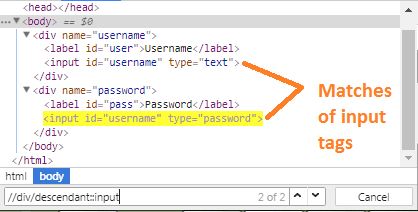
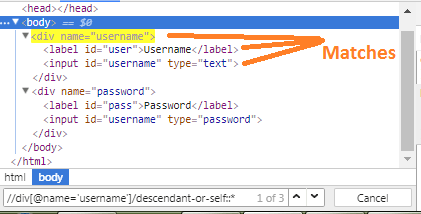
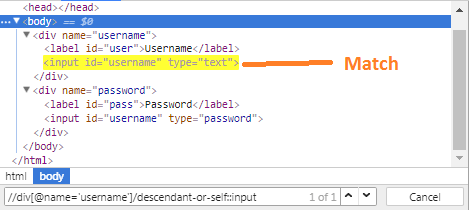
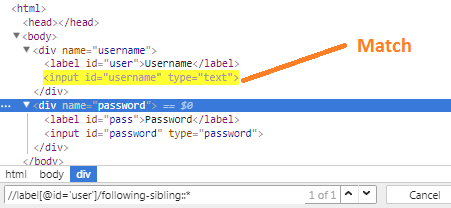
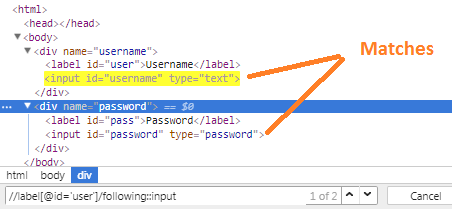
**<input id="username" type="password">**

**</div>**

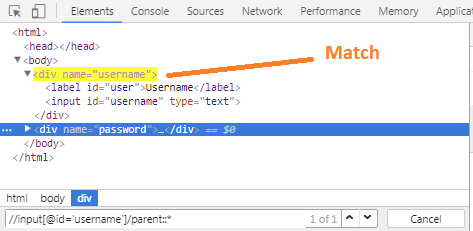
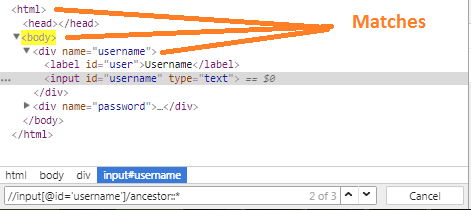
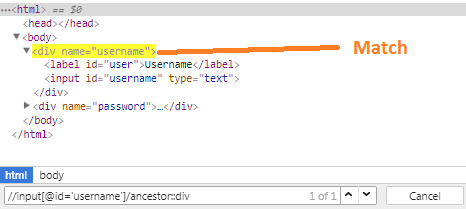
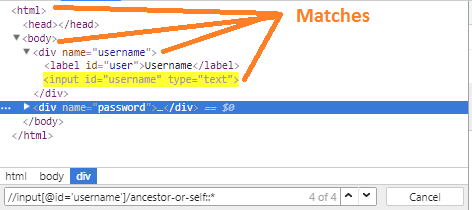
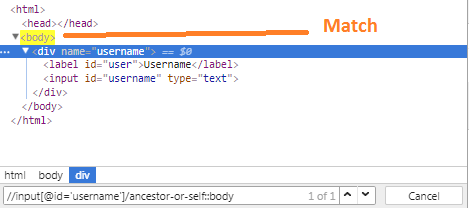
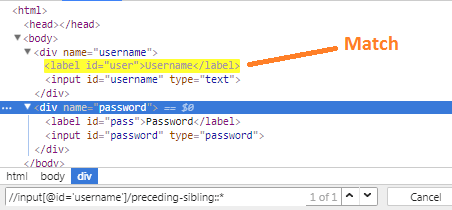
**</body>**

**</html>**

## Forward Axes

The forward axis in XPath helps to [find the element](https://chercher.tech/java/find-elements-in-selenium)/node after the current or reference element (helps to find an element in the code which is below the current element in HTML file).  
  
We can narrow down the matches by adding more details about the HTML element like tagnames, attributes.  
  
  
  
**self :**It specifies the current element  
  
  
  
**attribute :**It specifies the attributes of the current element.  
  
  
  
Now let's narrow down to the element(s), which have an attribute id.  
  
  
**child :**It specifies all child elements of the current element.  
  
  
  
We can match the element with a particular tag or ids, below one matches the child element(s) which have input as a tag.  
  
  
**descendant :**It specifies all the children and grandchildren elements  
  
  
  
Narrow down to the descendant who has input as a tag  
  
  
**descendant-or-self :** It specifies current or all the children and grandchildren elements  
  
  
  
Narrow down to the child who has input as a tag; if the current element has an input tag, then this xpath matches that as well.  
  
  
**following-sibling :** It specifies the following siblings of the current element. Siblings are at the same level as the current element and share its parent.  
  
  
  
**following :** It specifies all elements that come after the current element, which includes elements of other div's as well.  
  


## Reverse Axes

Reverse axis in xpath helps to [find the element](https://chercher.tech/java/find-elements-in-selenium)/node before the current or reference element (helps to find an element in the code which is above the current element in HTML file)  
  
We can narrow down the matches by adding more details about the HTML element like tagnames, attributes.  
  
**parent :**It specifies the parent of the current element.  
  
  
  
**ancestor :** It specifies the ancestors of the current element/nodes, which include the parents up to the root HTML.  
  
  
  
Narrow the matches to the element(s) which has **div** as HTML tag by navigating to ancestors   
  
**ancestor-or-self :**It specifies the current element or all elements that come before the current element  
  
  
  
Let's narrow down our search to the element which has tag as a body from the current element by navigating the reverse axis; if the current element is a body, then it matches the current element itself.  
  
  
**preceding-sibling :**It specifies the element that comes before the current element  
  
  
  
**preceding :** It specifies all elements that come before the current element (i.e. before it's opening tag).  
  
