**XPath**

**1.Introduction to XPath**

**1.1 What is XPath?**

* XPath is a query language used to navigate HTML documents.
* In automation testing, XPath helps identify specific elements on a webpage, enabling automated actions and validations.
* It allows for locating elements within a document using various criteria like element names, attributes, text content, and their relationships.

**1.2 Importance of XPath in Automation Testing**

* XPath is essential in automation testing as it provides a reliable way to locate elements on a web page.
* Unlike CSS selector, XPath offers more complex and versatile methods of finding elements.
* Making it particularly useful for handling complex page structures and dynamic content.

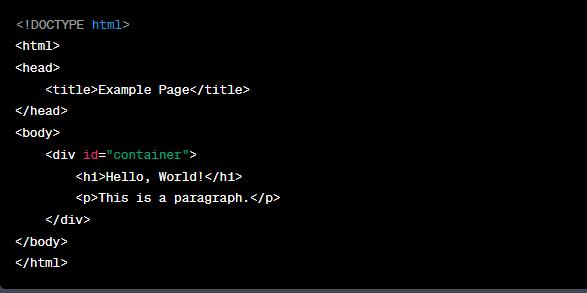
**2. XPath Syntax and Expressions**

**2.1 Absolute vs. Relative XPath**

* **Absolute XPath**: Absolute XPath is an XPath expression that specifies the complete path to the desired element from the root(<html>) of the document.

**Example:**

Consider the following HTML snippet:



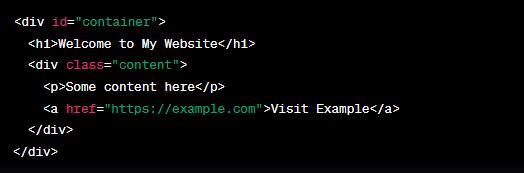
Let’s say you want to select the `<p>` element using an absolute XPath expression.

The absolute XPath for the `<p>` element would be: `/html/body/div/p`.

* **Relative XPath**: Relative XPath is a way to locate elements on a web page by specifying a path that starts from a particular element rather than from the root of the document.

Example:

Consider the following HTML snippet:



Locating the "Welcome to My Website" Heading:

**XPath:   //div[@id='container']/h1**

Locating the "Some content here" Paragraph:

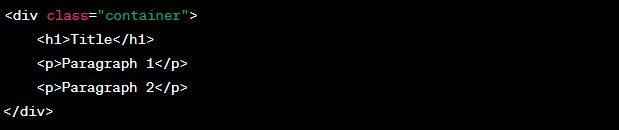
**XPath:  //div[@class='content']/p**

**2.2 Prefer 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.
* if an element is removed or added in the DOM, the relative XPath is not impacted

Here are a few examples to illustrate how to use Relative XPath:

Suppose you have the following HTML snippet:



* **Selecting an Element by Tag Name:**

Let's say you want to select the <h1> element. The relative XPath expression would be:

**//h1**

**Explanation:**

The // at the beginning indicates that the search should start from the root of the document, but then we directly specify the h1 tag without including the entire path.

* **Selecting Elements by Class:**

If you want to select all <p> elements with the class "container", the relative XPath expression would be:

**//div[@class=’container’]/p**

**Explanation:**

We start by locating the <div> element with the class "container" and then navigate to its child <p> elements.

* **Selecting Nth Element:**

To select the second <p> element, you can use:

**//p [2**]

**Explanation:**

[2] specifies that you want the second <p> element among all the <p> elements.

* **Selecting Element by Text:**

Suppose you want to select the <p> element containing the text "Paragraph 2". You can use:

**//P [text () =’paragraph 2’]**

**Explanation:**

text () = 'Paragraph 2' specifies the condition that the text of the <p> element should match "Paragraph 2".

* **Selecting Element by Partial Text:**

If you want to select the <p> element containing the word "Paragraph" (regardless of the number), you can use:

**//P [contains (text (), ‘paragraph’)]**

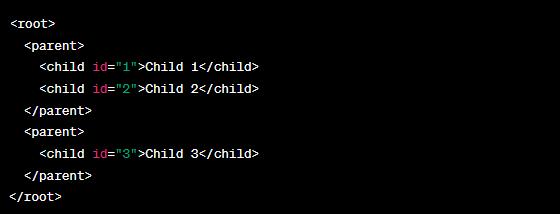
**Explanation:**

contains (text (), 'Paragraph') checks if the text of the <p> element contains the substring "Paragraph".

* Relative XPath expressions are useful because they adapt to changes in the structure of the document and make your queries more resilient to minor modifications.
* However, they might become less effective when dealing with more complex and deeply nested structures, where specific paths might be required.

**2.3 XPath Axes:**

* XPath axes are a set of methods that allow you to traverse the hierarchical structure of an HTML document to locate elements based on their relationships with other elements.
* There are several axes available in XPath, and they provide different ways to navigate through the document tree. Here are the main XPath axes explained with examples:

****

* **Ancestor Axis** (ancestor: :):

**XPath:**  //child[@id='3']/ancestor: parent

* **Parent Axis** (parent: :):

**XPath:**  //child[@id='1']/parent: parent

* **Child Axis** (child: :):

**XPath:**  //parent/child

* **Descendant Axis** (descendant: :):

**XPath:** //parent[@id='1']/descendant: child

* **Following Axis** (following: :):

**XPath:**  //parent[@id='1']/following: child

* **Preceding Axis** (preceding: :):

**XPath:** //child[@id='2']/preceding: child

* **Following-Sibling Axis** (following-sibling: :):

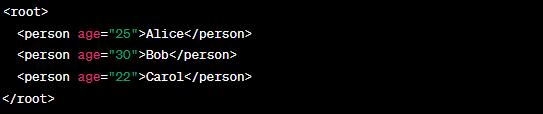
**XPath:** //child[@id='1']/following-sibling: child

* **Preceding-Sibling Axis** (preceding-sibling: :):

**XPath:** //child[@id='3']/preceding-sibling: child

**2.4 XPath predicates:**

* XPath predicates are used to filter elements or nodes in an XML or HTML document based on certain conditions.
* Predicates are added to XPath expressions within square brackets [...], and they allow you to select only the nodes that satisfy the given condition.



* Selecting Elements with Specific Attribute Values:

**XPath:**   //person[@age='30']

* Selecting Elements with Numeric Attributes:

**XPath:**  //person[number(@age) > 25]

* Selecting Elements with Text Content:

**XPath:** //person [text () ='Alice']

* Selecting Nth Element:

**XPath:**   //person [2]

**2.5 Common XPath Functions:**

* XPath functions enhance your ability to navigate and manipulate the document's structure and content.

Here are some common XPath functions and their usage:

* **text ()**:

This function returns the text content of the selected node.

**XPath**:   //p/text ()

* **count ():**

This function counts the number of nodes that match a given XPath expression.

**XPath**: count(//p)

* **concat ():**

This function concatenates two or more strings.

**XPath**: concat ("Hello, ", "world!")

* **contains ():**

This function checks if a given string contains a specific substring.

**XPath:** //div [contains (@class, "important")]

* **starts-with ():**

This function checks if a given string starts with a specific substring.

**XPath:** //img [starts-with (@src, "https://example.com")]

* **position ():**

This function returns the position of the current node in its parent's list of child nodes.

**XPath:** //li [position () = 2]

* **last ():**

This function returns the position of the last node in the context.

**XPath:** //tr [last ()]

* **sum ():**

This function calculates the sum of numeric values in a selected set of nodes.

**XPath:** sum(//price)

* **normalize-space ():**

This function removes leading and trailing whitespace and collapses internal whitespace into a single space.

**XPath:** normalize-space(//p)

**3. Handling complex scenarios**

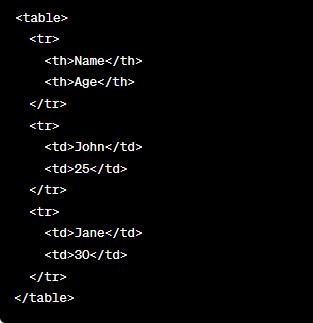
**3.1 Handling Complex Scenarios with XPath:**

* Handling complex scenarios with XPath involves using the XPath language to navigate and select elements within an XML or HTML document.
* XPath provides a way to address specific elements based on their structure, attributes, or relationships with other elements.

Let's go through examples of handling complex scenarios involving tables, dynamic dropdowns, and nested elements.

**1. Tables:**

Consider an HTML table like this:



**Example:** Selecting all ages from the table

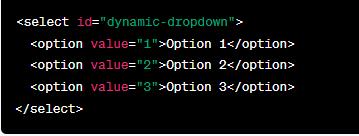
**XPath:** //table/tr [position () > 1]/td [2]/text ()

**Explanation:**

* //table: Select any table in the document.
* /tr [position () > 1]: Select table rows (excluding the header row).
* /td [2]: Select the second cell (age) in each row.
* /text (): Extract the text content from the selected cell.

**2. Dynamic Dropdowns:**

Consider a dynamic dropdown menu like this:



**Example:** Selecting the currently selected option

**XPath:** `//select[@id='dynamic-dropdown']/option[@selected]/text () `

**Explanation:**

* `//select[@id='dynamic-dropdown’] `: Select the dropdown element with the given ID.
* `/option[@selected] `: Select the option element with the "selected" attribute.
* `/text () `: Extract the text content of the selected option.

**3. Nested Elements:**

Consider a nested structure like this:



**Example:** Selecting text from nested paragraphs within a specific parent

**XPath:** //div[@class='parent’] [1] //p/text ()

**Explanation:**

* //div[@class='parent’] [1]: Select the first parent div element.
* //p/text (): Select the text content of all nested paragraph elements under the selected parent.