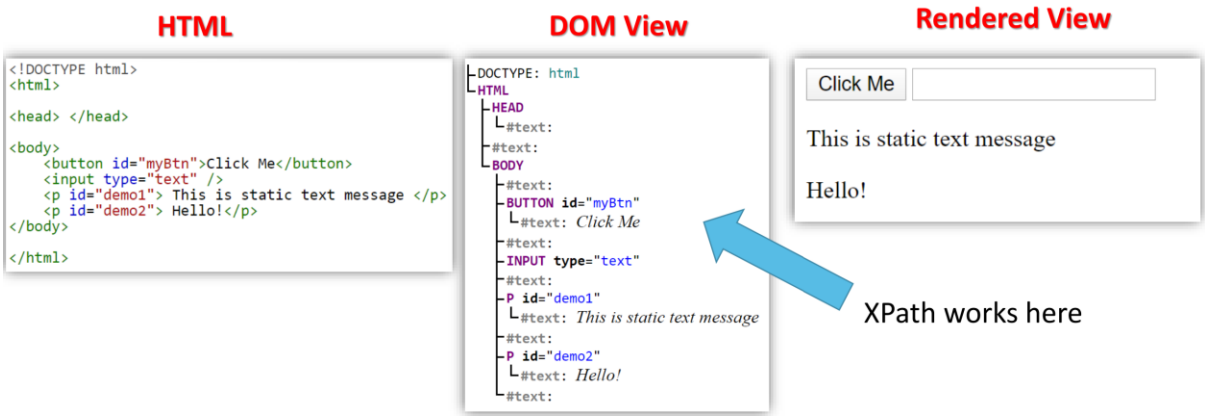
XPath

- **XPath** stands for XML Path Language.
- It is a syntax used to navigate through elements and attributes in an XML document.
- In web automation, XPath is used to locate elements on a webpage by their structure and attributes.

DOM (Document Object Model)

- **DOM** represents the structured content of a webpage, with each element as a node in a tree-like structure.
- XPath allows us to locate specific nodes (elements) within the DOM.



Types of XPath

1. Absolute XPath (Full XPath)

- An **Absolute XPath** provides the full path from the root of the document to the target element.
- It starts with a single /, which represents the root node.
- Example: /html/body/header/div/div/div[2]/div/input

2. Relative XPath (Partial XPath)

- A **Relative XPath** is a more flexible way of finding an element. It directly jumps to the element using attributes, without starting from the root.
- It starts with //, which allows XPath to search anywhere in the document.
- Example: //*[@name="search"]

Which XPath Should Be Preferred?

- **Relative XPath** is preferred because it is shorter, easier to maintain, and less likely to break if the webpage structure changes.

Differences Between Absolute and Relative XPaths

Absolute XPath	Relative XPath
Starts with /	Starts with //
Represents the full path from the root node	Does not need the full path, can directly target elements
Does not use attributes	Primarily works with attributes like id, class, etc.
Traverses through every node in the DOM	Jumps directly to the element using attributes

XPath Usage:

https://demowebshop.tricentis.com/

1. Absolute XPath

- Refers to the complete path from the root element to the target element in the DOM structure.
- **XPath Format**: /html/body/div/div[2]/div[1]
- **Example XPath**: /html[1]/body[1]/div[4]/div[1]/div[1]/div[1]/a[1]/img[1]
- **WebElement Example**: Logo

Notes: Absolute XPath is very specific and less reliable because even a small change in the webpage structure can break the XPath.

2. Relative XPath

- A shorter, more flexible version of XPath that begins from any node in the DOM, rather than the root.
- **XPath Format**: //tagname[@attribute='value']
- **Example XPath**: //img[@alt='Tricentis Demo Web Shop']
- **WebElement Example**: Logo

Notes: Relative XPath is more commonly used as it adapts better to minor changes in the page structure.

3. XPath with Single Attribute

- Selects elements based on one attribute and its value.
- **XPath Format**: //tagname[@attribute='value']
- **Example XPath**: //input[@id='small-searchterms']
- **WebElement Example**: Search box

4. XPath with Multiple Attributes

- Selects elements by matching multiple attributes and their respective values.
- **XPath Format**: //tagname[@attribute1='value1'][@attribute2='value2']
- **Example XPath**: //input[@value='Search'][@type='submit']
- **WebElement Example**: Search button

Notes: Multiple attributes help make the XPath more specific and reliable.

5. XPath with *and* Operator

- Uses the ***and*** operator to combine multiple attribute conditions that must all be true for the element to be selected.
- **XPath Format**: //tagname[@attribute1='value1' and @attribute2='value2']
- **Example XPath**: //input[@value='Search' and @type='submit']
- **WebElement Example**: Search button

6. XPath with *or* Operator

- Uses the ***or*** operator to combine multiple attribute conditions, where only one of the conditions needs to be true for the element to be selected.
- **XPath Format**: //tagname[@attribute1='value1' or @attribute2='value2']
- **Example XPath**: //input[@value='Search' or @type='submit']
- **WebElement Example**: Search button

7. XPath with contains() Function

- Matches elements that contain a specific substring within an attribute.
- **XPath Format**: //*[contains(@class, 'btn')]
- **Example XPath**: //h2//a[contains(@href,'computer')]

- **WebElement Example**: Selects products whose name contains "computer" under Featured Products.

Note: The contains() function is useful for partial matches.

8. XPath with starts-with() Function

- Matches elements whose attribute values start with a specified string.
- **XPath Format**: //*[starts-with(@id, 'user')]
- **Example XPath**: //h2//a[starts-with(@href,'/build')]
- **WebElement Example**: Selects products whose name starts with "build" under Featured Products.

Note: The starts-with() function is helpful for dynamic elements whose IDs or classes are partially consistent.

9. XPath with text() Function

- Selects elements based on the exact text content of the element.
- **XPath Format**: //*[text()='Login']
- **Example XPath 1**: //*[text()="Register"] – Selects the Register link.
- **Example XPath 2**: //*[contains(text(), "Fiction")] – Selects all product titles under Books that contain "Fiction".

10. XPath with last() Function

- Selects the last element in a set of matching nodes.
- **XPath Format**: //input[last()]
- **Example XPath**: //div[@class='column follow-us']//li[last()]
- **WebElement Example**: Selects Google+ under the FOLLOW US footer menu.

Note: The last() function is useful when you want the last occurrence of an element.

11. XPath with position() Function

- Selects an element based on its position in the list of matching nodes.
- **XPath Format**: //input[position()=2]
- **Example XPath**: //div[@class='column follow-us']//li[position()=2]

- **WebElement Example**: Selects the 2nd option under the FOLLOW US footer menu.

Dynamic XPath


```
<button name="start" class="start" style="">START</button>
```

```
<button name="stop" class="stop" style="">STOP</button>
```

XPath	Technique
//button[@name='start']	Exception
//button[@name='stop']	Exception
//button[@name='start' or @name='stop']	or
//button[contains(@name,'st')]	contains()
//button[starts-with(@name,'st')]	starts-with()
//button[text()='STOP' or text()='START'];	text() - innertext