22. XPath Axes and Relative Locators

1. Every element has a tagname in the HTML line.
2. Every element has a certain attributes.
3. Every attribute has certain value.
4. With the help of attribute value we can locate an element by using locator strategies

Scenario - 1

- What if suppose an element is not having any attributes how can we locate this elment

Scenario - 2

- Even though if the elements are having attributes if they values of attributes are dynamically changing then how can we locate this element

Solution

1. Locate an element whose values of attributes are static which is near to element not having attributes using Xpath.
2. Located element is a **Base Node** or **Current Node** or **Self Node** or **Context Node**
3. From that base element Traverse or Move or Navigate to element through the DOM from top to bottom, bottom to top which is not having any attributes or values of attributes changing

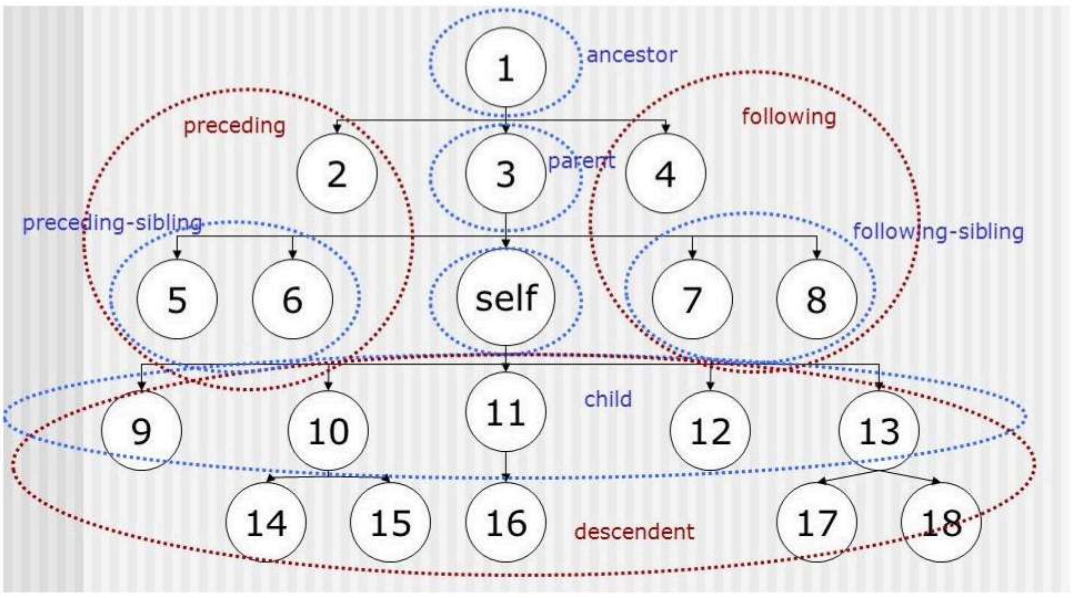
XPath Axes

- To move or traverse from base element in DOM from top to bottom, bottom to top we need to use keywords which are called Xpath Axes.
- In the process of moving, Keywords or Xpath Axes are used to search for the multiple nodes in the XML document from the current node context.
- Xpath axes or keywords should be used along with Xpath (directly jump to the element in the DOM).

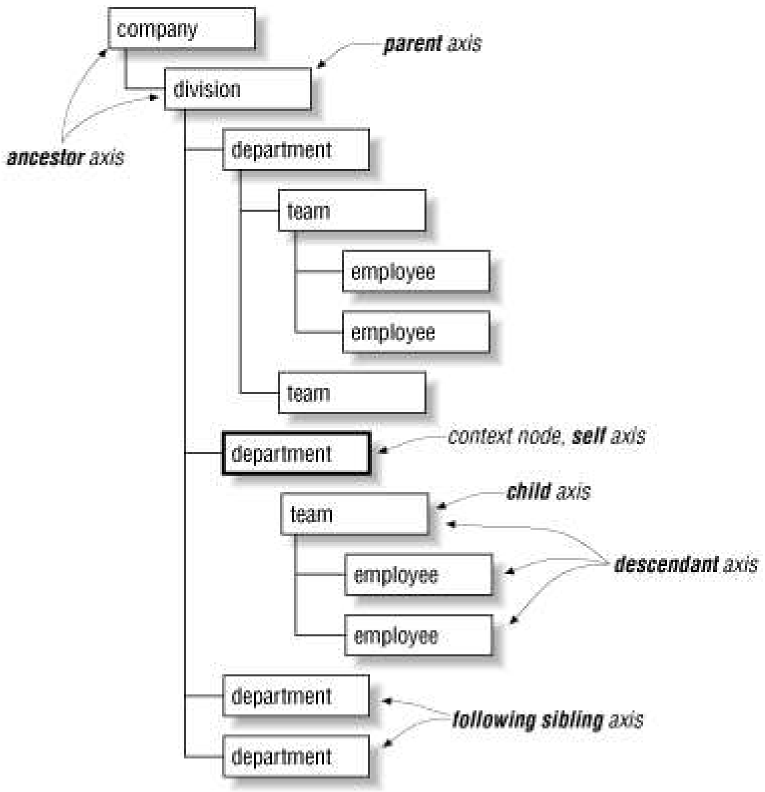
Keywords in XPath Axes

- self
- child, parent
- ancestor, descendant
- following, following-sibling
- preceding, preceding-sibling

Relationship of Nodes



Main purpose of Relationship of Nodes is if we want to identify some element suppose 1 which is not having any attribute in that we can identify nearby element **2 or 3 or 4 or selfnode** from there we can navigate and find that element so that kind of traversing is possible with XPath Axes.
Every Number is representing Node/Element in DOM structure/webpage/HTML.Whichever element we are taking as Basis and that is Self node or Current node.We can make any node as a self node.



Note

- Every node can be **Base Node** or **Current Node** or **Self Node** or **context Node.**
- self is optional for **Base Node** or **Current Node** or **Self Node** or **context Node.**

XPath Axes Description

Axes	Description	Syntax
Child	Traverse all **child element** of the **current html tag**	//*[attribute='value']/child::tagname
Parent	Traverse **parent element** of the **current html tag**	//*[attribute='value']/parent::tagname
Following	Traverse **all element** that comes **after the current tag**	//*[attribute='value']/following::tagname
Preceding	Traverse **all nodes** that comes **before the current html tag.**	//*[attribute='value']/preceding::tagname
Following-sibling	Traverse from **current Html tag** to **Next sibling Html tag**.	//current html tag[@attribute ='value']/following-sibling:: sibling tag[@attribute ='value']
Preceding-sibling	Traverse from **current Html tag** to **previous sibling Html tag.**	//current html tag[@attribute ='value']/preceding-sibling:: previous tag[@attribute ='value']
Ancestor	Traverse all the **ancestor elements** (grandparent, parent, etc.) of **the current html tag.**	//*[attribute='value']/ancestor::tagname
Descendant	Traverse all **descendent element** (child node, grandchild node, etc.) of the **current Html tag.**	//*[attribute='value']/descendant::tagname

Xpathaxes.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
options = webdriver.ChromeOptions()
```

```
options.add_experimental_option("detach", True)
driver = webdriver.Chrome(options=options)
driver.get("https://testautomationpractice.blogspot.com/")
driver.maximize_window()
driver.implicitly_wait(10)
```

Self

```
self_element = driver.find_element(By.XPATH, "//input[@id='input1']/self::input")
print("Self:", self_element.get_attribute("name"))
```

Parent

```
parent_element = driver.find_element(By.XPATH, "//input[@id='input1']/parent::div")
print("Parent:", parent_element.get_attribute("class"))
```

Child

```
child_element = driver.find_element(By.XPATH, "//div[@id='section1']/child::p")
print("Child:", child_element.text)
```

Ancestor

```
ancestor_element = driver.find_element(By.XPATH,
"//input[@id='input1']/ancestor::div[@class='container']")
print("Ancestor:", ancestor_element.get_attribute("class"))
```

Descendant

```
descendant_element = driver.find_element(By.XPATH, "//div[@class='container']/descendant::h4[1]")
print("Descendant:", descendant_element.text)
```

Following

```
following_element = driver.find_element(By.XPATH,
"//input[@id='input1']/following::input[@id='input2']")
print("Following:", following_element.get_attribute("name"))
```

Following-sibling

```
following_sibling_element = driver.find_element(By.XPATH,
"//input[@id='input1']/following-sibling::button")
print("Following sibling:", following_sibling_element.text)
```

Preceding

```
preceding_element = driver.find_element(By.XPATH,
"//input[@id='input2']/preceding::input[@id='input1']")
print("Preceding:", preceding_element.get_attribute("name"))
```

Preceding-sibling

```
preceding_sibling_element = driver.find_element(By.XPATH,
"//input[@id='input1']/preceding-sibling::p")
print("Preceding sibling:", preceding_sibling_element.text)
```

Relative locators (Friendly Locators)

- ➔ In Selenium 4, Relative Locators (previously called Friendly Locators) allow you to locate web elements relative to other elements on a web page.
- ➔ This is particularly helpful when a direct locator (like id or class) is not available or dynamic.
- ➔ **List of relative locators**
  - ◆ **above()** ⇒ Finds an element that is located above a specified element.
  - ◆ **below()** ⇒ Finds an element that is located below a specified element.

- **to_left_of()** ⇒ Finds an element that is located to the left of a specified element.
- **to_right_of()** ⇒ Finds an element that is located to the right of a specified element.
- **near()** ⇒ Finds an element that is located near (within a 50-pixel radius by default) a specified element.

RelativeLocators.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.relative_locator import locate_with
options = webdriver.ChromeOptions()
options.add_experimental_option("detach", True)
driver = webdriver.Chrome(options=options)
driver.get("https://testautomationpractice.blogspot.com/")
driver.maximize_window()
driver.implicitly_wait(10)
```

Reference Element

```
reference_element = driver.find_element(By.ID, "btn2")
print("Reference Element Text: ", reference_element.text)
```

Above the Reference Element

```
element_above =
driver.find_element(locate_with(By.TAG_NAME, "button").above(reference_element))
print("Above Element Text: ", element_above.text)
```

Below the Reference Element

```
element_below =
driver.find_element(locate_with(By.TAG_NAME, "button").below(reference_element))
print("Below Element Text: ", element_below.text)
```

To the Left of the Reference Element

```
element_left = driver.find_element(locate_with(By.TAG_NAME, "input").to_left_of(reference_element))
element_left.send_keys("T-shirts")
```

To the Right of the Reference Element

```
elements_right =
driver.find_elements(locate_with(By.TAG_NAME,"h4").to_right_of(reference_element))
print(len(elements_right))
for ele in elements_right:
  print(ele.text)
```

Near the Reference Element

```
elements_near = driver.find_elements(locate_with(By.TAG_NAME,"h4").near(reference_element))
print(len(elements_near))
for ele in elements_near:
  print(ele.text)
```