**FindElement():**

1. findElement() is used to **get the address of first matching element on the webpage**.
2. Returntype is **WebElement Interface**.
3. If findElement() method is **not able to find the element** on the webpage, we get **NoSuchElementException**.

--------------------------------------- \*\*\*\*\* --------------------------------------------------

**FindElemets():**

1. findElements() is used to **get the address of all the matching elements on the webpage**.
2. ReturnType is **List<WebElement>**
3. If findElements() method is **not able to find the elements** on the webpage, then we **get empty List**.

--------------------------------------- \*\*\*\*\* --------------------------------------------------

**INSPECTION METHODS LOCATORS:**

Before performing action on any element, first we should locate the element on webpage.

Hence to find its address we have **findElement() method and findElements() method** inside WebDiver interface.

1. findElement() method of WebDriver interface is used to get the address of first matching element on the webpage.
2. Return type of findElement() method is WebElement interface.
3. If findElement() method is not able to find the element on the webpage, then we get NoSuchElementException.
4. **findElement() method takes locators as its argument.**

**Locators:**

Locators are static methods **of abstract ‘By’ class.**

(OR)

Locators are **used to locate one or more elements** on the webpage.

(OR)

Locators act as argument for findElement() and findElements().

**Types of Locators:**

1. tagName(String arg)
2. id(String arg)
3. name(String arg)
4. className(String arg)
5. linkText(String arg)
6. partialLinkText(String arg)
7. cssSelector(String arg)
8. xpath(String arg)

**EXAMPLE:**

**HTML Code to create link:**

<html>

<body>

<a href='https://www.google.com' id='i1' name='n1' class='c1' value='v1' title=’t1’>GOOGLY</a>

</body>

</html>

-----------------\*\*\*\*\*-----------------

In the above code, to create a link we use <a> tag (Anchor Tag) and mandatory attribute called 'href', but other attributes like id, name, class, value, title are optional. So, in selenium we can use features like tag, attributes, id, name and class and text to identify the element on the webpage.

So according to above e.g.

1. tagName("a") ---------- We use tagName of the element.
2. id("i1") ---------- We use id of the element.
3. name("n1") ---------- We use name of the element.
4. className("c1") ---------- We use class of the element.
5. linkText("Googly") ---------- Here we use text but make sure element is 'link' only.
6. partialLinkText("Go") ---------- Here we use a part of the complete linktext.
7. **cssSelector** ---------- Here We can identify any element by giving **both tag and attribute**.

**Syntax:**

**tagName[AttributeName='AttributeValue']**

Example:a[href='https://www.google.com']

Example:a[id='i1']

Example:a[name='n1']

Example:a[class='c1']

Example:a[value='v1']

Example:a[title='t1']

1. **xpath:**

xpath is the **path travelled in the HTML tree to find an element**.

(OR)

xpath is **one of the locator which covers all possible ways** to find an element.

There are **Two Types** of xpath:

1. Absolute xpath
2. Relative xpath

**1. Absolute xpath:**

Means **complete path** we travel **from start(html)** to element --> It is **achieved by'/'(Single Forward Slash)** --> **immediate child/immediate descendant**.

**Note:**

In Realtime applications, **if we use absolute xpath**, it will become very lengthy. So**, we use Relative xpath.**

**2.Relative xpath:**

It is the **shortest path** --> **achieved by '//'(Double Forward Slash**) --> **Any Child/Any descendant.**

**Relative xpath- Shortest Path of element in HTML Tree.** // means any child or any descendant.

**Ex:**

html

body

div [1]

input A [1]

input B [2]

div [2]

input V [1]

input D [2]

**Elements Relative xpath**

ABVD//input

AB//div [1]/input

A//div [1]/input [1]

AV//input [1]

BA//div [1]/input [2] |//div [2]/input [1]

AD//div [1]/input [1] |//div [2]/input [2]

--------------------------------------- \*\*\*\*\* --------------------------------------------------

**Cases of Relative xpath:**

**Case-1:** **xpath by unique attribute:**

**Syntax:**

**//tagName[@AttrName='AttrValue']**

**Examples:**

1. //input[@name='q’] (Google.com)
2. //a[@href='https://www.facebook.com/Praveendraomusic/'] (<http://www.yuvadhwaja.in/index.html>)
3. //img[@src='images/Kunwar singh\_kannada.jpg']
4. //input[@name='pwd'] (actitime.com)

--------------------------------------- \*\*\*\*\* --------------------------------------------------

**Case-2: xpath by text() function:**

**Syntax:**

**//tagName[text()='textValue']**

**Examples:**

1. //div[text()='login']
2. //small[text()='Now']
3. //a[text()='Gmail']

--------------------------------------- \*\*\*\*\* --------------------------------------------------

<div fn='Deepika' ln='padukone'>

<div fn='Deepika' ln='Rai'>

<div fn='Aishwarya' ln='Rai'>

//div[@fn='Deepika'] //2

//div[@ln='Rai'] //2

//div[@fn='Deepika' and @ln='Rai'] //1

In this Situation we go for **case-3** (i.e., we can get element **by multiple attributes**).

**Case-3: xpath by multiple attributes:**

**Syntax:**

**UniqueElement(and):**

**//tagName[@attr1='attrVal' and @attr2='attrVal' and ........]**

**ManyElements(or):**

**//tagName[@attr1='attrVal' or @attr2='attrVal' or........]**

**Examples:**

1. //div[@class='\_4IiNRh \_2mtkou' and @title='Core i3'] (flipkart.com)
2. //div[@class='\_1GEhLw' and text()='Core i7'] (flipkart.com)

--------------------------------------- \*\*\*\*\* --------------------------------------------------

<div id='ui-id-1'>

<div id='ui-id-2'>

<div id='ui-id-3'>

<div id='ui-id-4'>

<div id='ui-id-5'>

//div[@id='ui-id-1'] | //div[@id='ui-id-2'] |

//div[@id='ui-'id-1' or @id='ui-id-2' or ... ]

Raghavendra equals Raghavendra

Raghavendra **contains** Raghav

In this Situation we go for **case-4** (i.e., we can get element **by contains() function**).

**Case-4: xpath by contains() function:**

**Syntax:**

**//tagName[contains(@attrName,'AttrValue')]**

Write xpath expression to match all the above five

1.Handle Partially Dynamic Elements

Example: //div[contains(@id,'ui-id')]

2.Avoid Writing Lengthy text

//p[contains(text(),'Uttara Karnataka')]

3.Avoid Spaces present in the values

//div[contains(text(),'Login')]

--------------------------------------- \*\*\*\*\* --------------------------------------------------

**Case-5: xpath by axis(Relationship):**

**1.Travel from Parent --> Child:**

**parent element/descendant::child element**

**2.Travel from Child --> Parent:**

**child element/ancestor::parent element**

**3.Travel from one element --> next element of same parent:**

**element/following-sibling::next element**

**4.Travel from one element --> previous element of same parent:**

**element/preceding-sibling::previous element**

**Ex:**

html

body

table

tbody

tr id=t1

td A

td B

td C

td D

tr id=t2

td E

td F

td G

td H

1. html to body-->//html/descendant::body

2. html to tbody-->//html/descendant::tbody

3. html to B-->//html/descendant::td[text()='B']

4. tbody to G-->//tbody/descendant::td[text()='G']

5. t1 to ABCD -->//tr[@id='t1']/descendant::td

E to t2--> //td[text()='E']/ancestor::tr[@id='t2']

t1 to body-->//tr[@id='t1']/ancestor::body

C to html-->//td[text()='C']/ancestor::html

A to B --> //td[text()='A']/following-sibling::td[text()='B']

A to B,C,D --> //td[text()='A']/following-sibling::td

F to H --> //td[text()='F']/following-sibling::td[text()='H']

A to C,D-->//td[text()='A']/following-sibling::td[text()='C' or text()='D']

G to F-->//td[text()='G']/preceding-sibling::td[text()='F']

D to -->//td[text()='D']/preceding-sibling::td[text()='A']

H to F,G-->//td[text()='H']/preceding-sibling::td[text()='F' or text()='G'] cousin

B to F-->//td[text()='B']/ancestor::tr/following-sibling::tr[@id='t2']/descendant::td[text()='F']uncle

G to t1 -->//td[text()='G']/ancestor::tr/preceding-sibling::tr

//td[text()='G']/ancestor::tbody/descendant::tr[@id='t1']

--------------------------------------- \*\*\*\*\* --------------------------------------------------

**Case-6: xpath by Dependent and independent element:**

**(OR)**

**xpath by reference of surrounding unique element:**

(Selenium.dev)

Ruby Download

Java Download

Python Download

C# Download

JS Download

In our example

1.Inspect Independent Element

//td[.='Java']

2.Travel back until dependent element

within boundary

//td[.='Java']/

3.Inspect dependent and write continuously

//td[.='Java']/..//a[.='Download']

(or)

Alternate way to reach from Java ----> Download

//td[.='Java']/following-sibling::td[@data-label='Links']/descendant::a[.='Download']

//td[.='Java']/following-sibling::td[@data-label='Links']/a[.='Download']

//td[text()='Java']/following-sibling::td[@data-label='Links']/descendant::a[text()='Download']

//td[text()='Java']/ancestor::tr/descendant::a[text()='Download']

//td[text()='Java']/..//a[text()='Download']

(flipkart.com)

//div[text()='Motorola']/preceding-sibling::div[@class='\_1p7h2j']

--------------------------------------- \*\*\*\*\* --------------------------------------------------

**case-7: xpath by group index:**

What is the difference btw //input, input[1], (//input)[1] ??

//input-matches with all the inputs on the webpage

//input[1]-matches with all the first inputs of their parent element(tag)

(//input)[1]-matches with the very first input on the webpage

(//input)[57]-matches 57th input

(//input)[last()] -matches last input

(//input)[last()-1]-matches second last(input before the last one)

(//input)[position() > last()-5]-matches last 5 inputs

(//input)[position() mod 2 = 0]-matches all the even inputs

(//input)[position() mod 2 = 1]-matches all the odd inputs

--------------------------------------- \*\*\*\*\* --------------------------------------------------