Question:

what is difference between / , ./ , // , .//

**Locator in selenium:**

1. id: use to locat unique element in webpage
   1. it is also use for locat single element
2. name: use to locat element by name attribute
3. linktext: use to locat element by complete text/value
4. partial linktext: use to locat element by partial text/value
5. className: use to locat element by className
   1. it is also use for locat multiple element
6. TagName: use to locat element by html tags
   1. it is also use for locat multiple element
7. Css Selector: This is a combination of multiple selector like or we can also use single selector
   1. -> a) Tag and id //tag#id
   2. b) Tag and class //tag.className
   3. c) Tag and attribute //tag[attribute = value]
   4. d) Tag, class and attribute //tag.className[attribute = value]
   5. driver.findElement(By.cssSelector("#id"); // find element by id
   6. driver.findElement(By.cssSelector(".className"); // find element by class
   7. driver.findElement(By.cssSelector("[attribute = value]"); //find element by attribute
8. XPath: it is use to locat element by using xpath expression

---------------------------------------------------------------------------------------------------------------------------------------------------

==> **XPath:**

- XPath is stands for XML Path

- It is a query-based syntax for XML schema and HTML documents that testers use to traverse elements within the document.

- Expressions refer to the values used in XPath, and an expression typically contains the chaining of multiple elements with a specific syntax or structure.

- xpath is locator strategies for finding the element through there address/location

- Symbol for select attribute in xpath = '@'

- Types of Nodes in xpath:

- Element Node

- Text Node

- Attribute Node

- Document Node

- Others

- Relationship of Nodes or XPath Axes:

- Parents

- Children

- Siblings

- Ancestors

- Descendants

* **Types of XPath:**

1) Absolute Xpath:

- It has a xpath from root element

eg: /html/body/div/div/section/div/div/input

- Direct way to locate an element

- Starts from root Node and single slash (/)

- Not reliable :

- As soon as any change happen in any element the xpath won't be work

2) Relative XPath Or Dynamic XPath:

- Syntax for creation of xpath

//tag[@attribute='value']

- It starts from the node of your choice

- Shorter than Absolute Xpath

- More reliable and shorter

- It start from any node and with double slash(//) thats why it access any element in hierachy

eg: //div[@id="row1"]/input

---------------------------------------------------------------------------------------------------------------------------------------------------

**Difference Between Absolute XPath and Relative XPath:**

Absolute XPath | Relative XPath

1: It is Starts from root node | 1: It is directly jump to element on DOM

2: It is starts with single back slash '/' | 2: It is starts with double slash '//'

3: In this we only use tags/nodes | 3: In this we use attributes.

---------------------------------------------------------------------------------------------------------------------------------------------------

**Advantages of XPath**

- XPath supports all major test automation libraries and programming languages

- It provides bidirectional flow which means traversal can be both ways

- Supports both XML and HTML documents.

- XPath expression works at any document level, with no restriction to start the traversal at a specific level

- It is the declarative expression not procedural; this helps to use any indexes and different types of axes/functions efficiently to select the specific node

- XPath selector is compatible with old-age browsers as well as modern browsers

- XPath locator provides Axes and methods. You can use them in XPath expression to solve complex locator problems.

**Disadvantages of XPath**

- The performance of the XPath varies from application to application. As a matter of fact, experts say that XPath locators are comparatively slow, which can affect the overall performance of the test suite.

- It cannot solve the modern world shadow DOM problem.

- XPath is associated with more than one element in the DOM tree, and it tends to break as and when new element-level changes are introduced.

- Usage of indexes (specific nodes with numbers) within the XPath causes high maintenance (Ex: //div[1]).

- If XPath gets complicated more it becomes untidy and readability decreases

---------------------------------------------------------------------------------------------------------------------------------------------------

-->Q) When do we use . (dot) in front of XPath?

=> When we need to create a relative location path, starting at the context node

---------------------------------------------------------------------------------------------------------------------------------------------------

**--> '//' vs './/'**

1) // :

- XPath expression: "//div"

- Result: Selects all div elements anywhere in the document.

- Resulting nodes: <div id="main">, <div>, <div id="secondary">

2) ".//":

- Assume you are at the <div id="main">.

- XPath expression: ".//div"

- Result: Selects all div elements within the subtree rooted at the current context node (the <div id="main">).

- Resulting nodes: <div>

---------------------------------------------------------------------------------------------------------------------------------------------------

--> **Position and index**

- If multiple element present with same tag or attribute so to select/locat that element we use position() function and index

1) **index:**

- index is start from 1 to go on

- Syntax: tag[index]

eg: h5[2]

2) **position():**

- It is a function or property where we can pass index to locat element

- Syntax: tag[position() = index]

eg: h5[position() = 2]

- The main difference between position and index is we can use multiple comparator operator with position() function

-- Equal operator (=) -> //h5[position()=2]

-- Not Equal operator (!=) -> //h5[position()!=2]

-- Less than operator (<) -> //h5[position()<2]

-- Less than or equal to operator (<=) -> //h5[position()<=2]

-- Greater than operator (>) -> //h5[position()>2]

-- Greater than or equal to (>=) -> //h5[position()>=2]

==> This operator we cannot use with index we can only use with position() function.

3) **last():**

- last() is a function to select last matching element

- Syntax: tag[last()]

eg: h5[last()] --> it is select last h5 element in webpage

- we can also perform some substraction/operation using last function

eg: h5(last()-1] --> it is select second last h5 element in webpage

--> **Xpath Expression indexing:**

- if we dont have any id or attribute for locat and xpath pointing to multiple location so we can use index to identify that xpath

- Syntax: (xpath-Expression)[index]

eg: (//div[@class="row"])[2]

|-> xpath //div[@class="row"] return multiple row so if we want to 2 row so we mention [2] index

---------------------------------------------------------------------------------------------------------------------------------------------------

--> **Different XPath functions:**

- This function is use for Dynamic XPath generator

1) Last() : finding the last element of webpage

2) Position() : use to find element by index

3) Contains() : Find the element by particular content present in that tag or not --> eg: //body[contains(@class,'page-id-400')] --> It check mentioned class are present in body-class or not if present than select body element

4) Starts-with() : find the element from start-with particular text() --> eg: //h1[starts-with(text(),'Log in')]

5) Text() : To find the element by text() --> eg: //h1[text()='log in']

6) not() : It is use for negation purpose --> eg: //p[not(starts-with(text(),'This page is'))] --> it highlight all <p> tag element whose value not starts-with "This page is"

---------------------------------------------------------------------------------------------------------------------------------------------------

--> **XPath Operators:**

- In xpath there is following operator: =,!=,<,>,and,or,div(division)

- or: or operator is use for if we want to select element by checking one of them condition satisfied

- and: and operator is use when we want to select element by checking if and only if all condition is satisfied

---------------------------------------------------------------------------------------------------------------------------------------------------

--> **XPath Wildcards character '\*':**

- In XPath '\*' is work as a wildcards character

- It will replace by any element based on condition

- Syntax: //\*[@attribute='value']

eg: //button[@\*='add\_btn'] --> it select element whose attribute value is add\_btn

---------------------------------------------------------------------------------------------------------------------------------------------------

--> **XPath Axes:**

- XPath axes are those axes that are used to search for the multiple nodes in the XML document form the current node context.

- These method are mainly used when the web element is not identified with the help of ID , name, className, link text, css selector

- In many cases there are many element present in webpage without attibute, so it is difficult to identify element using old xpath formula thats why we use Axes to locat element

- Syntax : axesname::nodetest[predicate]

1) ancestor : select all ancestor node of current node like parent, grand-parent etc

2) descendant :select all descendant node of current node like children, grand-children etc

3) parent : select parent node of current node --> eg: //div[@id='row1']/parent::div

4) following-sibling : select all sibling after current node

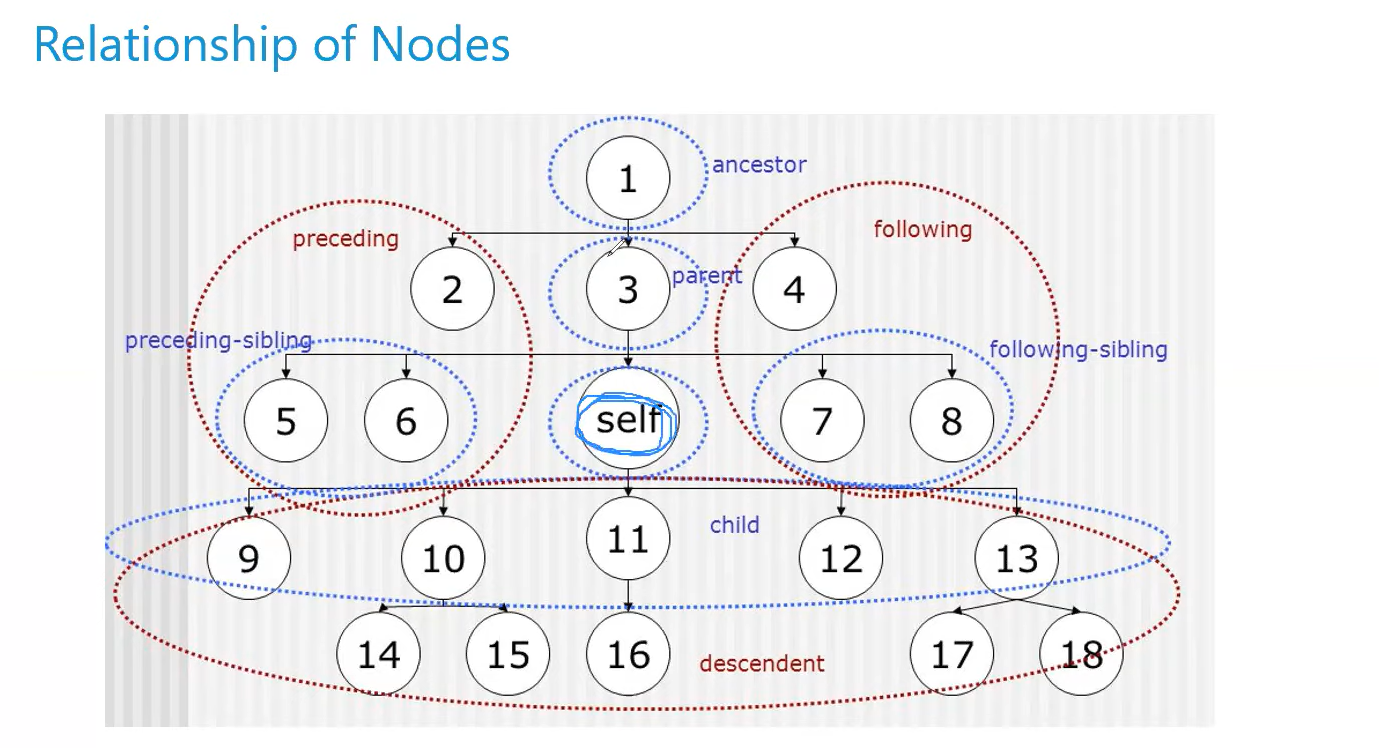
5) preceding-sibling : select all sibling before current node --> eg: //li[text()='Verify text saved']/parent::ol/preceding-sibling::h5[1]

6) Self : From where we are starting to locate element this nod is self node

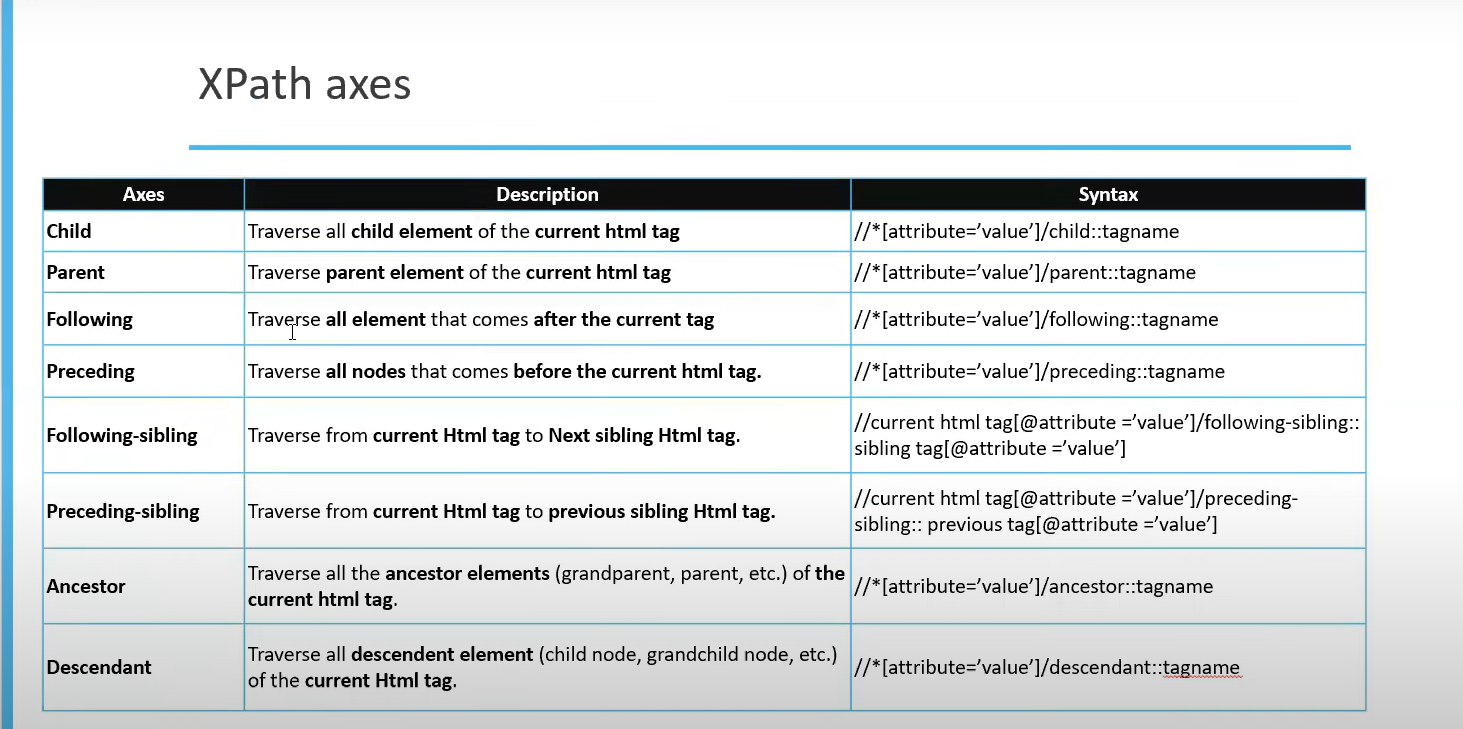
7) Child : select child node of current node

8) Following : Select all node after current node

9) Preceding : Select all node before current node



Self node syntax: //\*[attribute=’value’]



---------------------------------------------------------------------------------------------------------------------------------------------------

--> **Combine XPath expression:**

- If we want to combine two or more xpath expression than we use pipe '|'

- Syntax: xpath1-expression | xpath2-expression

eg: //h2 | //h5 --> it will select all element with h2 and h5 element

//div[@id='row1']/button | //div[@id='row1']/input

---------------------------------------------------------------------------------------------------------------------------------------------------

===> Note: Writing XPath for <svg> and it child element is different

you can find through wildcard value

eg: //\*[@y='11']

---------------------------------------------------------------------------------------------------------------------------------------------------

--> **XPath vs CSS Selector**

* 1. Xpath is faster than css selector
  2. css selector are more user-friendly than xpath
  3. css selector only traverse forword direction but xpath traverse both forword and backword direction.
  4. xpath provide functions() who can identify/locat element by the function but css-selector are not support function
  5. Xpath is good when we dealing with complex web structure or when precise element selection is required. But css selector are used when target element are easily identified.