**Xpath (8 with formula)**

**What is Xpath?**

1. Xpath or XML path is a query language for selecting nodes from XML documents. XPath is one of the locators supported by selenium webdriver. XPath is used to locate the elements. Using XPath, we could navigate through elements and attributes in an XML document to locate web elements such as textbox, button, checkbox, Image etc., in a web page.
2. **Syntax** of Xpath=//tagname[@attribute=’value’] // Here, //---Select current Node. Tagname ---Tagname of the particular node. @-----Selects Attribute. Attribute ---Attribute name of the node. Value----Value of the Attribute.
3. **Types of Xpath ---**Absolute xpath and 2) Relative xpath
4. **Absolute xpath--**It is called complete or full XPath. It starts from <html> tag or it starts from single forward slash(/). It begins with top HTML node and ends with input node and it is prefixed with a “/”. **Example -/HTML/body/div/div[@id=’Email’]**

* **Advantage** —it identifies the element very fast.
* **Disadvantage**—If some other tag added or removed in between , then this path will no longer works.

1. **Relative Xpath --**It begins from the current location and is prefixed with a “//”. Path starts with the node of your choice.**Example: //span[@class=’Email’].or //\*[@class=’features -box’]/**
2. **It starts with double forward slash(//), It can search the element anywhere at the webpage.**

* **Advantage** – no need to write long xpath. you will choose the element from where the relative path should begin. The chosen element is less likely to change .This is a safer approach. This xpath will select all the elements present in DOM.

### **What does / and // means in xpath?**

**One forward slash:** / means absolute xpath. It begins with a root path and it is prefixed with a “/”.

**Double forward slash:** // means relative xpath. It begins from the current location and is prefixed with a “//”. Q..W**hat is the difference between Absolute Path and Relative Path?**

**…………………………………….**

1. **What is CSS?--CSS stands for Cascading Style Sheets.** It is a language that describes the style of an HTML document. **It is mainly used for designing, coloring, setting layout.. There are 3 ways we can use CSS: InlineCall, InternalCall, ExternalCall. External Style Sheets can save a lot of work. This Sheets are stored in CSS files.** It is a way of identifying the element on web page or finding address of an element on the web page. Using CSS is best practice when you compare to xpath.

**Advantage of CSS over Xpath are**

* CSS Selector will not change browser to browser as xpath will change.
* CSS is native to browsers and Xpath is not
* Xpath can traverse up the document from a child element to parents.
* Xpath engines are different in each browser, hence make them inconsistent.
* It is used to find html element based on various attribute like name.Id , class and many others.
* By **using below method** we can identify the elements uniquely

1)TagName 2) Id 3) ClassName 4)Attributes 5) Contains 6)Starts-with 7)Ends with 8)Using Siblings.

1. **Why CSS selector is faster than XPath?**

CSS selectors perform far better than Xpath and it is well documented in Selenium community—Because--- Xpath engines are different in each browser, hence make them inconsistent. Xpath tend to become complex and hence make hard to read in my opinion.

**Xpath Formula: It has 8 formula.**

<input type=text placeholder="First Name">   
<input type=text placeholder="First Name" id="welcome to bittech">

<span type=text placeholder="First Name"> hello,this is xpath class </span>  
<span type=text placeholder="First Name"> Hi,this is xpath class </span>  
<span> This is xpath class </span>  
    <div>  
    <div>  
<a> <div> <div> </a>  
  
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
**Xpath formula:**  
**1: If i need 1 property then:**  
Syntax: //TagName[@PropertyName='pValue']  
EX: //input[@type='text']

**2: If i need more then 1 property under same tag:**Syntax: //tagName[@pName='pValue' and @pName='pValue']  
Ex: //input[@type='text' and @id='welcome to bittech']

**3: If i want to use part of value:**Syntax: //tagName[contains(@pName,'part of pValue')]   
EX: //input[contains(@id,'bittech')

**4: If i want to add 2 tag:**Syntax: //TagName[@PropertyName='pValue']//tagName[@pName='pValue' and @pName='pValue']  
EX: //input[@type='text']//span[@type='text' and @placeholder='First Name']

**5: If i want to use a text between 2 tag:**  
Syntax: //tagname[text()='Text value']  
EX: //span[text()=' This is xpath class ']

**6: If i want to use part of text between 2 tag:**  
Syntax: //tagName[contains(text(),'Part of Text')]  
EX: //span[contains(text(),'xpath')]

**7: If i want to convert from absolute to relative xpath:**  
Syntax: /tagName/tagName/tagName//tagname[text()='Text value']  
Ex;/input/input/span//span[text()='This is xpath class']a

**8: If i need to index any tag:**  
syntax: //tagName[text()='Text Value']/div[number of index]  
Ex: //span[text()=' This is xpath class ']/div[2]

* Target tag must have siblings with same tag name
* catch the parent
* use /TargetTagname[1] (mention index number)

**Java Scripts [Current name -ECMA Script 6 -2015]**

1. JavaScript is the programming language of HTML and the Web. JavaScript is client-side scripting language .It is used to create interactive effects within web browsers
2. **To use JavaScript we can make** -- Web or mobile application ,Real time applications like chats and video streaming services .With JavaScript we can build the backend for our web and mobile applications
3. JavaScript is case sensitive ,It contains many statements, all ending with a semicolon ;
4. Not sensitive to whitespace or line breaks,Write one -line comments using //
5. Write multiline comments using /\*/ and It runs from top to bottom

**JavaScript Syntax**

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

var x, y, z;          // How to declare variables  
x = 5; y = 6;      // How to assign values  
z = x + y;         // How to compute values

**JavaScript Values**

**The JavaScript syntax defines two types of values: Fixed values and variable values.**

**Fixed values are called literals. Variable values are called variables.**

**JavaScript Literals**

The most important rules for writing fixed values are:

**Numbers** are written with or without decimals:

**Strings** are text, written within double or single quotes:

"John Doe"  
'John Doe'

**JavaScript Variables**

In a programming language, **variables** are used to **store** data values.

JavaScript uses the var keyword to **declare** variables.

An **equal sign** is used to **assign values** to variables.

**In this example**, x is defined as a variable. Then, x is assigned (given) the value 6:

var x;  
x = 6;

**JavaScript Operators**

JavaScript uses **arithmetic operators** ( + - \* / ) to **compute** values:

(5 + 6) \* 10  
JavaScript uses an **assignment operator** ( = ) to **assign** values to variables:

var x, y;  
x = 5;  
y = 6;

**JavaScript Expressions**

An expression is a combination of values, variables, and operators, which computes to a value.

The computation is called an evaluation.

For example, 5 \* 10 evaluates to 50:

5 \* 10

Expressions can also contain variable values:

x \* 10

The values can be of various types, such as numbers and strings.

For example, "John" + " " + "Doe", evaluates to "John Doe":

"John" + " " + "Doe"

**JavaScript Keywords**

JavaScript **keywords** are used to identify actions to be performed.

The var keyword tells the browser to create variables:

var x, y;  
x = 5 + 6;  
y = x \* 10;

**JavaScript Comments**

Not all JavaScript statements are "executed".

Code after double slashes // or between /\* and \*/ is treated as a comment.

Comments are ignored, and will not be executed:

var x = 5;   // I will be executed  
  
// var x = 6;   I will NOT be executed

You will learn more about comments in a later chapter.

**JavaScript Character Set**

JavaScript uses the **Unicode** character set.

Unicode covers (almost) all the characters, punctuations, and symbols in the world.

For a closer look, please study our [Complete Unicode Reference](https://www.w3schools.com/charsets/ref_html_utf8.asp)

**JavaScript Identifiers**

Identifiers are names.

In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).

The rules for legal names are much the same in most programming languages.

In JavaScript, the first character must be a letter, or an underscore (\_), or a dollar sign ($).

Subsequent characters may be letters, digits, underscores, or dollar signs.

Numbers are not allowed as the first character.  
This way JavaScript can easily distinguish identifiers from numbers.

**JavaScript is Case Sensitive**

All JavaScript identifiers are **case sensitive**.

The variables lastName and lastname, are two different variables:

var lastname, lastName;  
lastName = "Doe";  
lastname = "Peterson";

JavaScript does not interpret **VAR** or **Var** as the keyword **var**.

**JavaScript and Camel Case**

Historically, programmers have used different ways of joining multiple words into one variable name:

**Hyphens**:

first-name, last-name, master-card, inter-city.

Hyphens are not allowed in JavaScript. They are reserved for subtractions.

**Underscore:**

first\_name, last\_name, master\_card, inter\_city.

**Upper Camel Case (Pascal Case):**

FirstName, LastName, MasterCard, InterCity.

**Lower Camel Case:**

JavaScript programmers tend to use camel case that starts with a lowercase letter:

firstName, lastName, masterCard, interCity.