CSS-Selector:

htmltag[attribute=’value’]

input[id='username']

id:

htmltag#valueOfID

or

#valueOFID

Classname:

Htmltag.classname

Or

.classname

Parent to child :

Css-expression > specify the child tag

Xpath:

* Absolute xpath - > /html/body/../../../../../.././
* **Relateive xpath-> //htmlTag[@attribute=’value’]**
* Xpath by position

Basic :

//input[@id='username']

Functions in xpath :

text() :

//htmltag[text()=’completeText’]

* Example : //h3[text()='Careers | Synechron']

contains(arg1,arg2):

* arg1 🡪 attribute or function
* arg2 🡪 should be a partial value of attribute or text
* Example :

//h3[contains(text(),'Working')]

//img[contains(@src,'timer.png')]

starts-with(arg1,arg2)

* arg1 🡪 attribute or function
* arg2 🡪 should be a partial value of attribute or text
* //button[starts-with(@id,'ext-gen')]
* //h3[starts-with(text(),'Synechron')]
* //div[starts-with(@id,'ext-gen')]
* //\*[starts-with(@id,'ext-gen')]

Regular Expressions :

* AND
  + //htmltag[@attribute1=’value1’ AND @attribute2=’value2’ AND text()=’texttoSearch’]
  + Example:
    - //input[@class='textField' and @id='username']
* OR
  + //htmltag[@attribute1=’value1’ OR @attribute2=’value2’ OR text()=’texttoSearch’]
    - //input[@class='textField' or @id='username']

Traversing from parent to child :

If child is immediate child :

* //a[@id=’loginButton’]/div

Not immediate child :

* //td[@id=’loginButtonContainer’]//div

Traversing from child to parent:

//parentTAG[parentHtmlTAG[xpathofElement]]

NOTE : While handling dependent and independent elements , always write xpath for independent elements FIRST.

* Traversing from child to parent
  + //tr[th[text()='Directed by']]//a
* //div[div[div[h3[text()='Super Saver Bali (4N)']]]]//p[contains(@class,'latoBold')]

Xpath using axes Functions

Traversing between siblings:

* following-sibling
  + //htmltag[@attribure=’value’]/following-sibling::siblingTAG
  + //tr[th[text()='Directed by']]//a
* OR
  + //th[text()='Directed by']/following-sibling::td/a
* preceding-sibling
  + //tr[th[text()='Directed by']]/preceding-sibling::tr
* Following
  + //tr[th[text()='Directed by']]/following::tr
* Preceding
  + //tr[th[text()='Directed by']]/preceding::tr
* Ancestor
  + //h3[text()='Super Saver Bali (4N)']/ancestor::div[@class='boxShadow bdr packageListing pointer packageDetailsBox']//p[contains(@class,'latoBold')]
* Parent
  + //th[text()='Directed by']/parent::tr
* Child
  + //a[@id='loginButton']/child::div

WebDriver Setup :

* JAVA Software
* Eclipse
* Maven installation
* Added dependencies
* Downloaded driver executables
* Basic Test – launching
* findElement
* findElements
* getText()
* getAttribute(String)
* implicit wait
* webdriverwait
* fluentwait
* writing reusable functions
* handling dropdown using select
* handling multiselect dropdown using select
* handling customized dropdown
* Actions
  + Keyboard action
  + Mouse movement
  + Drag and drop
  + Right click
  + Double click
* Switch to
  + Alerts / popups
  + Window
  + Frame
* Hidden division popup
* Authentication popup
* File download popup
* File upload popup
* AutoIT
* Data driven testing
  + How to read properties files
  + How to read text file
  + How to use in test
  + How to use POI – reading excel
* Capturing Screenshot - TBD
* Executing Java script - TBD
* Desired Capabilities – TBD
* Executing Tests on different Brower using WebDriverManager

FrameWorks:

* Function Driven Automation Framework
* Data Driven Automation Framework
* Keyword Driven Automation Framework
* Hybrid Driven Automation Framework
* Page Object Model
* TestNG

TestNG :

* Annotations
  + @Test
  + @BeforeSuite
  + @AfterSuite
  + @BeforeClass
  + @AfterClass
  + @BeforeTest
  + @AfterTest
  + @BeforeMethod
  + @AfterMethod
  + @Dataprovider
* Order of execution
  + Using priority
  + Using dependsOnMethods
* Reading Data
  + From xml file
  + From dataprovider
* Groups in TestNG
  + Include
  + Exclude
* Reports
* Executing Tests in Parallel
* Configuring Extent Reports in selenium
* Taking Screen shots in selenium
* Executing javascript in selenium
* POM
  + Created a project and sample code
  + Added compiler plugin to maven
  + Added surefire plugin to maven
  + Created profiles for different testng xml file
* Created Batch file for test execution
* GRID
  + Downloaded latest GRID jar file
  + Created hub
  + Copied json sample file and modified as required
  + Regestred Node to hub
  + Updated driver creation logic
  + Executed Code on Distributed Environment