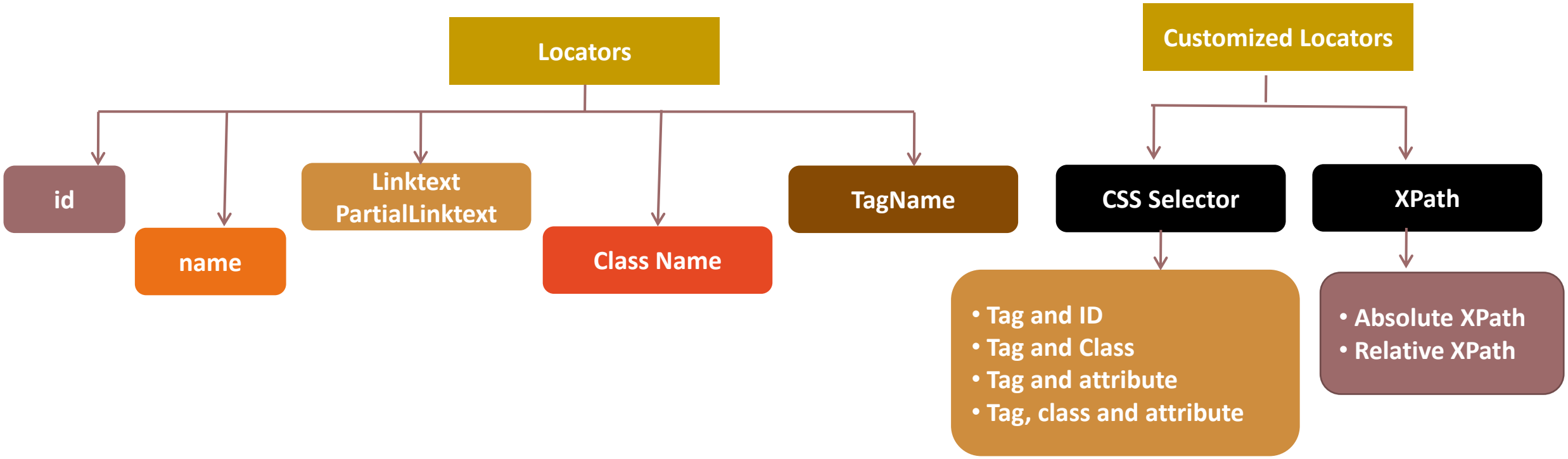
Selenium Locators

Types of Locators

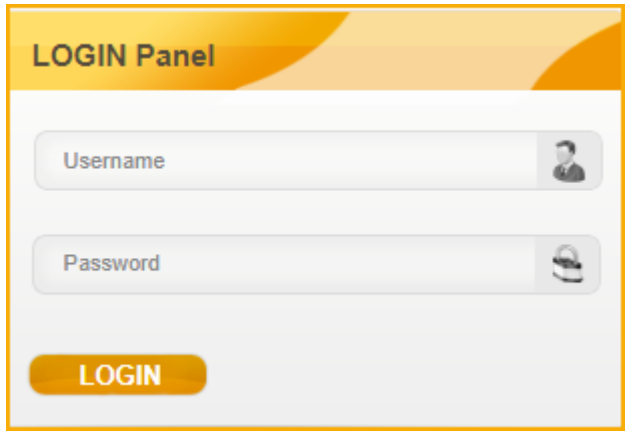
- We can identify various elements on the web using **Locators**.
- Locators are addresses that identify a web element uniquely within the page.



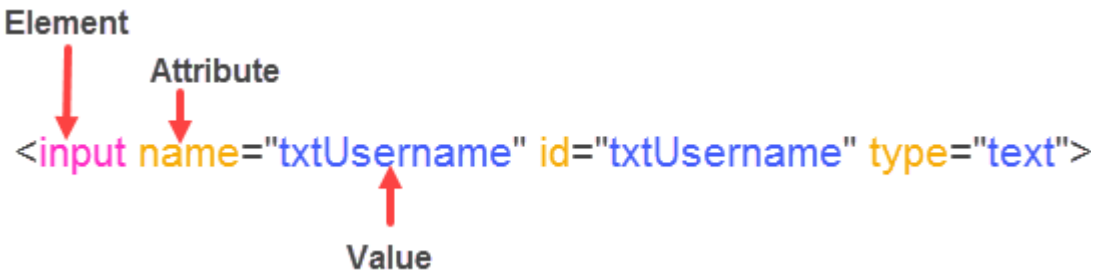
Locators

- id
- name
- linkText
- Partial LinkText
- class
- TagName

HTML Structure

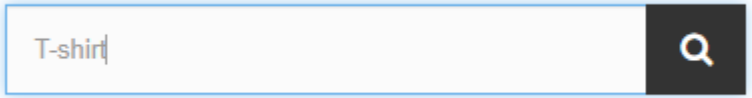



```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
▶<head>…</head>
▼<body>
  ▼<div id="wrapper">
    ▼<div id="content">
      ▶<style type="text/css">…</style>
      ▼<div id="divLogin">
        ▶<div id="divLogo">…</div>
        ▼<form id="frmLogin" method="post" action="/index.php/auth/validateCredentials">
            <div id="logInPanelHeading">LOGIN Panel</div>
          ▼<div id="divUsername" class="textInputContainer">
              <input name="txtUsername" id="txtUsername" type="text">
              <span class="form-hint">Username</span>
            </div>
          ▼<div id="divPassword" class="textInputContainer">
              <input name="txtPassword" id="txtPassword" type="password">
              <span class="form-hint">Password</span>
            </div>
            <div id="divLoginHelpLink"></div>
          ▼<div id="divLoginButton">
              <input type="submit" name="Submit" class="button" id="btnLogin" value="LOGIN">
            </div>
          </form>
        </div>
      </div>
    </div>
  </body>
</html>
```



ID

http://automationpractice.com/index.php

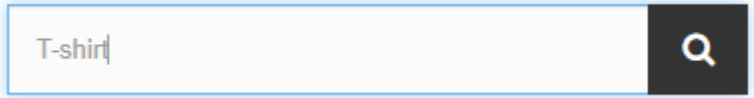



```
<input class="search_query form-control ac_input" type="text" id=
"search_query_top" name="search_query" placeholder="Search" value
autocomplete="off"> == $0
```

driver.find_element(By.ID,"search_query_top").send_keys("T-shirt")

Name

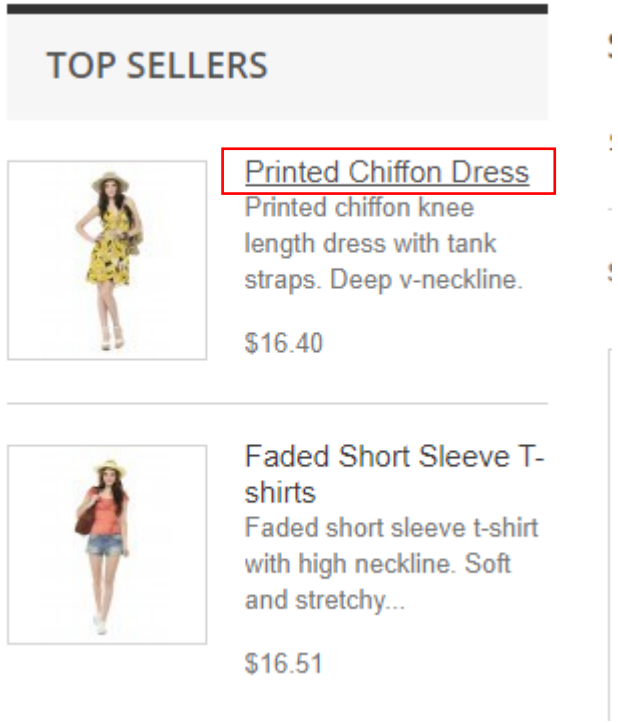
http://automationpractice.com/index.php




```
▶<button type="submit" name="submit_search" class="btn btn-default
button-search">…</button> == $0
```

driver.find_element(By.NAME,"submit_search").click()

Link Text / Partial Link Text

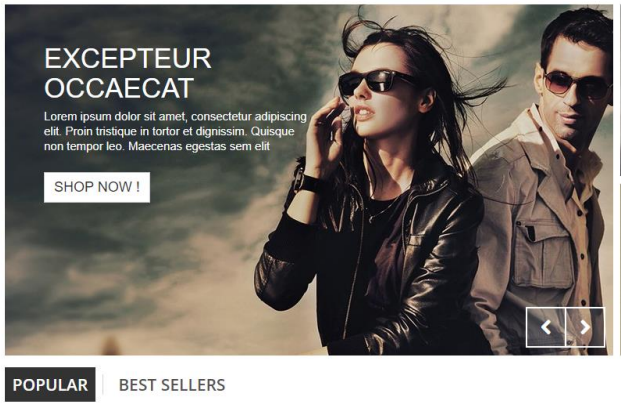



```
<a class="product-name" href="http://automationpractice.com/index.php?id_product=7&controller=product" title>
                                Printed Chiffon Dress
                            </a> == $0
```

driver.find_element(By.LINK_TEXT,"Printed Chiffon Dress").click()
driver.find_element(By.PARTIAL_LINK_TEXT,"Chiffon Dress").click()

Class Name

http://automationpractice.com/index.php

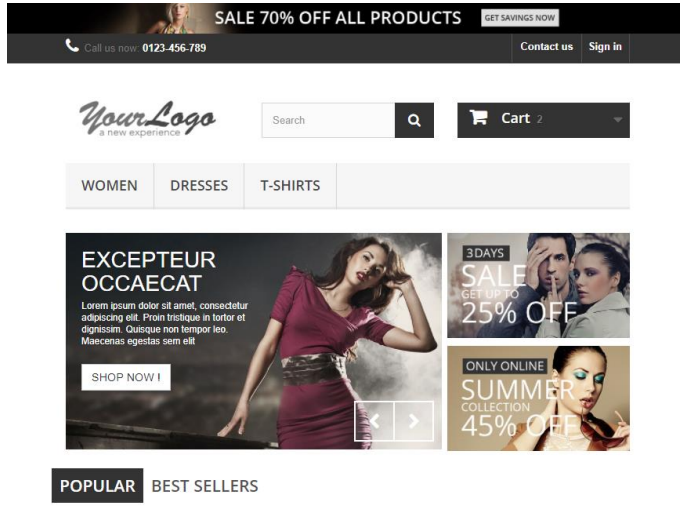



```
<ul id="homeslider" style="max-height: 448px; width: 515%; position: relative; left: -960px;"> == $0
  <li class="homeslider-container bx-clone" style="float: left; list-style: none; position: relative; width: 480px;">…</li>
  <li class="homeslider-container" style="float: left; list-style: none; position: relative; width: 480px;">…</li>
  <li class="homeslider-container" style="float: left; list-style: none; position: relative; width: 480px;">…</li>
  <li class="homeslider-container" style="float: left; list-style: none; position: relative; width: 480px;">…</li>
  <li class="homeslider-container bx-clone" style="float: left; list-style: none; position: relative; width: 480px;">…</li>
</ul>
```

```
sliders=driver.find_elements(By.CLASS_NAME,"homeslider-container")
print(len(sliders))
```

TagName

http://automationpractice.com/index.php




```
links=driver.find_elements(By.TAG_NAME,"a")
print(len(links))
```

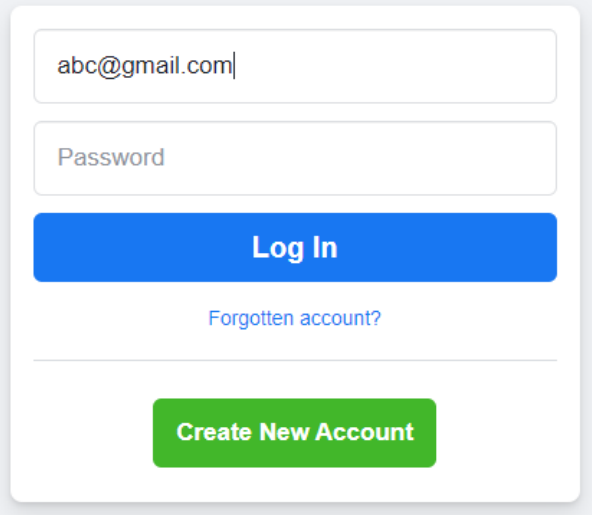
CSS Selectors

CSS Selector - Cascading Style Sheets

- Tag & ID (OR) #id
- Tag & class (OR) .class
- Tag & attribute (OR) [attribute=value]
- Tag , class & attribute

CSS Selector – *Tag* and *ID*

https://www.facebook.com/

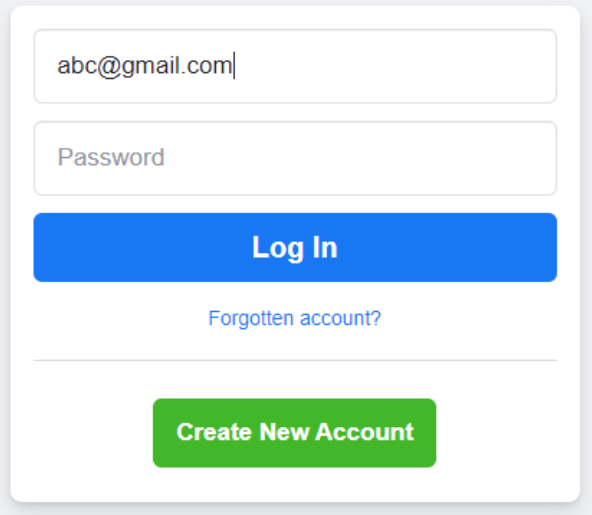



```
<input type="text" class="inputtext _55r1 _6luy" name="email" id=
"email" data-testid="royal_email" placeholder="Email address or phone
number" autofocus="1" aria-label="Email address or phone number"> == $0
```

driver.find_element(By.CSS_SELECTOR,"#email").send_keys("abc@gmail.com")
(or)
driver.find_element(By.CSS_SELECTOR,"input#email").send_keys("abc@gmail.com")

CSS Selector – *Tag* and Class

https://www.facebook.com/

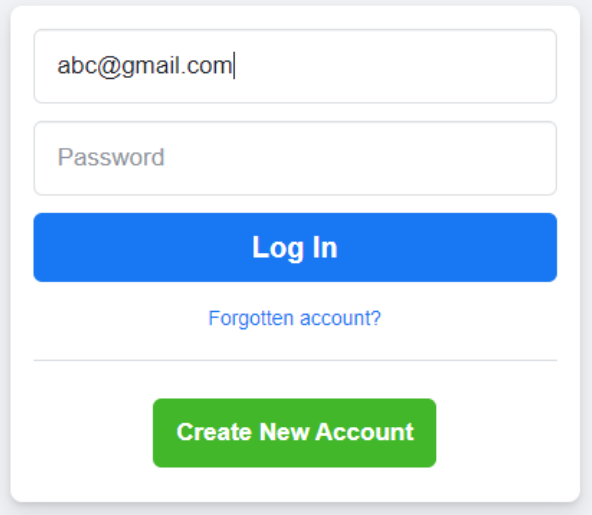



```
<input type="text" class="inputtext _55r1 _6luy" name="email" id=
"email" data-testid="royal_email" placeholder="Email address or phone
number" autofocus="1" aria-label="Email address or phone number"> == $0
```

driver.find_element(By.CSS_SELECTOR,".inputtext").send_keys("abc@gmail.com")
(or)
driver.find_element(By.CSS_SELECTOR,"input.inputtext").send_keys("abc@gmail.com")

CSS Selector – *Tag* and Attribute

https://www.facebook.com/

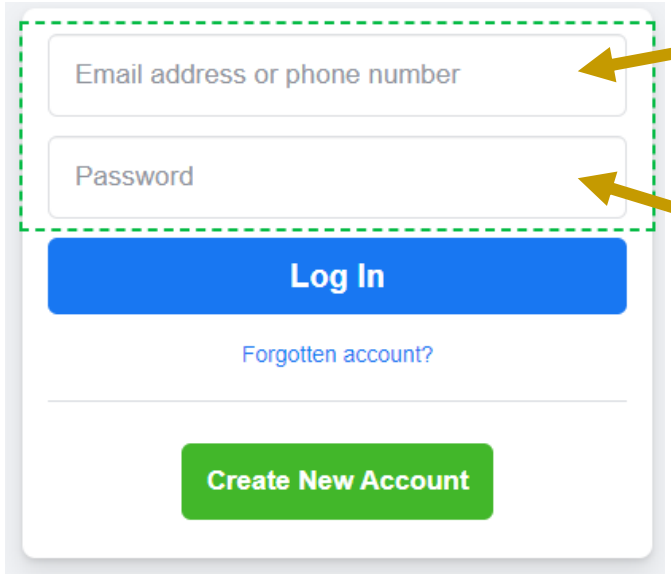



```
<input type="text" class="inputtext _55r1 _6luy" name="email" id=
"email" data-testid="royal_email" placeholder="Email address or phone
number" autofocus="1" aria-label="Email address or phone number"> == $0
```

driver.find_element(By.CSS_SELECTOR,"[name=email]").send_keys("abc@gmail.com")
(or)
driver.find_element(By.CSS_SELECTOR,"input[name=email]").send_keys("abc@gmail.com")

CSS Selector - *Tag, class* and *attribute*

https://www.facebook.com/




```
▼<div class="_6lux">
    <input type="text" class="inputtext _55r1 _6luy" name=
    "email" id="email" data-testid="royal_email" placeholder=
    "Email address or phone number" autofocus="1" aria-label=
    "Email address or phone number" style>
  </div>
▼<div class="_6lux">
    <input type="password" class="inputtext _55r1 _6luy" name=
    "pass" id="pass" data-testid="royal_pass" placeholder=
    "Password" aria-label="Password">
  </div>
```

```
driver.find_element(By.CSS_SELECTOR,"input.inputtext[data-testid=royal_email]").send_keys("abc@gmail.com") #Email
driver.find_element(By.CSS_SELECTOR,"input.inputtext[data-testid=royal_pass]").send_keys("abc") #Password
```

XPath

XPath

1. **What Is XPath?**
2. **Types Of XPaths**
   - Absolute
   - Relative
3. **How to capture XPath?**
4. **Writing Dynamic XPath by different ways:**
   - Using ‘OR’ & ‘AND’
   - Using Contains()
   - Using Starts-With()
   - Using Text()
   - Chained XPath

What is XPath?

- XPath is defined as **XML path**.
- **It is a syntax or language for finding any element on the web page using XML path expression**.
- XPath is used to find the location of any element on a webpage using **HTML** DOM structure.
- XPath can be used to navigate through elements and attributes in DOM.

DOM – Document Object Model

- DOM is an API Interface provided by browser.
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

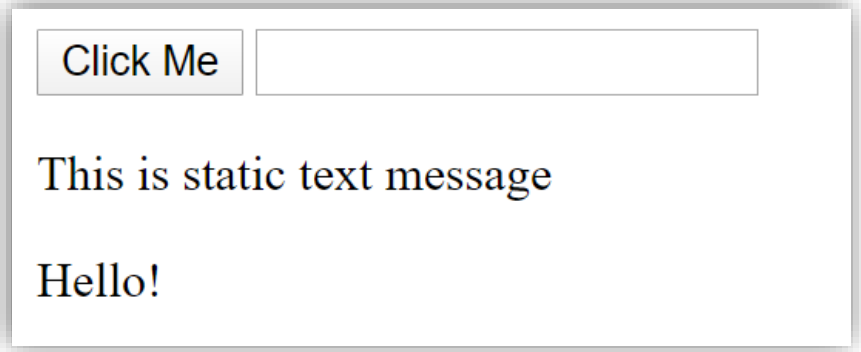
HTML

```
<!DOCTYPE html>
<html>

<head> </head>

<body>
    <button id="myBtn">Click Me</button>
    <input type="text" />
    <p id="demo1"> This is static text message </p>
    <p id="demo2"> Hello!</p>
</body>

</html>
```

DOM View

```
DOCTYPE: html
HTML
  HEAD
    #text:
  #text:
  BODY
    #text:
    BUTTON id="myBtn"
      #text: Click Me
    #text:
    INPUT type="text"
    #text:
    P id="demo1"
      #text: This is static text message
    #text:
    P id="demo2"
      #text: Hello!
    #text:
```

Rendered View



XPath works here

Absolute XPath

- It is the direct way to find the element.
- The disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.
- It begins with the single forward slash(/) ,which means you can select the element from the root node.
- Below is the example of an absolute XPath expression of the element
- Ex:

Absolute Xpath : /html[1]/body[1]/div[1]/div[1]/header[1]/div[3]/div[1]/div[1]/div[1]/a[1]/img[1]

Relative XPath

- Relative XPath the path starts from the middle of the HTML DOM structure.
- It starts with the double forward slash (//), which means it can search the element anywhere at the webpage.
- You can start from the middle of the HTML DOM structure and no need to write long XPath.

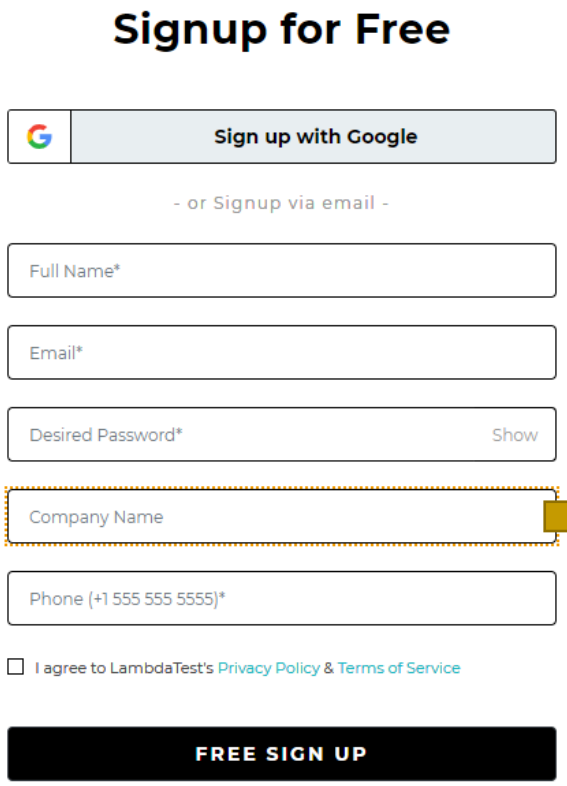
Ex:

Relative Xpath : //img[@class='logo img-responsive']

Syntax for Relative XPath

- XPath contains the path of the element situated at the web page. Standard syntax for creating XPath is.
- **// :** Select current node.
- **Tagname:** Tagname of the particular node.
- **@:** Select attribute.
- **Attribute:** Attribute name of the node.
- **Value:** Value of the attribute.
- Xpath=//tagname[@attribute='value']

XPath with OR



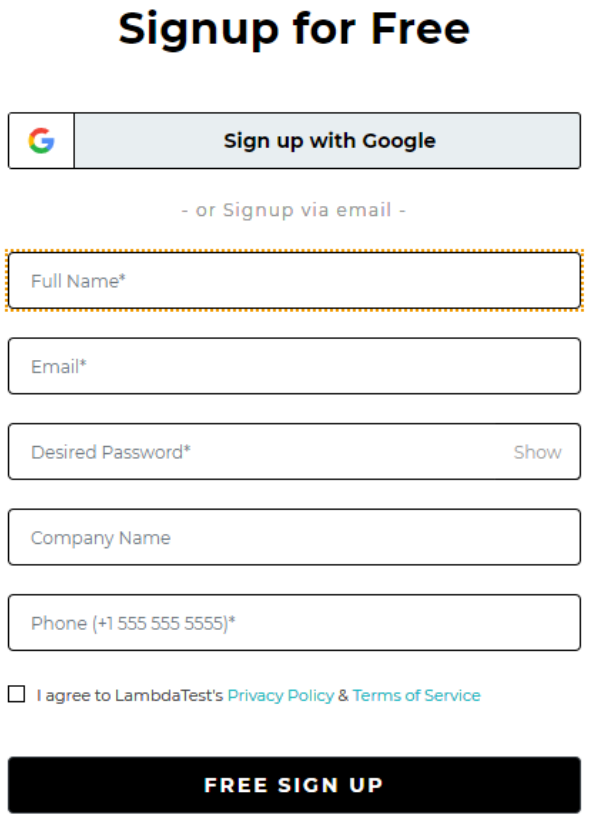
https://accounts.lambdatest.com/register

```
▼<div class="form-group">
    <input type="text" placeholder="Company Name" name=
    "organization_name" value class="form-control " style
    xpath="1"> == $0
  </div>
```

driver.find_element(By.XPATH,"//input[@name='organization_name' or @placeholder='Organization']").send_keys("abc")

XPath with AND

https://accounts.lambdatest.com/register

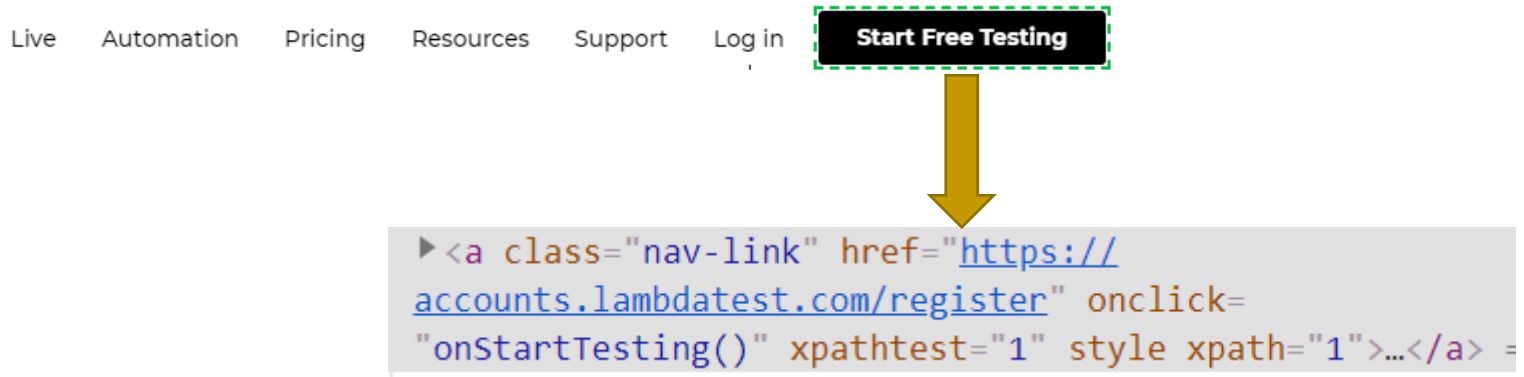


```
▼<div class="form-group">
    <input type="text" placeholder="Full Name*"
    name="name" value required="required" class=
    "form-control " xpath="1"> == $0
  </div>
```

```
driver.find_element(By.XPATH,"//input[@name='name' and @placeholder='Full Name*']").send_keys("abc")
```

XPath with contains()

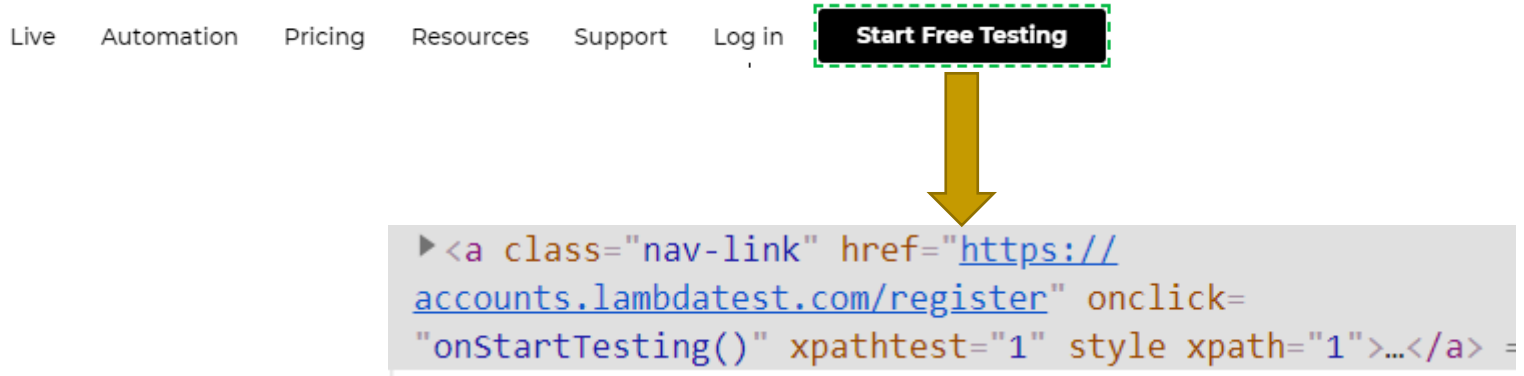
https://www.lambdatest.com/



driver.find_element(By.XPATH,"//a[contains(text(), 'Testing')]")
driver.find_element(By.XPATH,"//a[contains(@id, 'value')]")

XPath with starts-with()

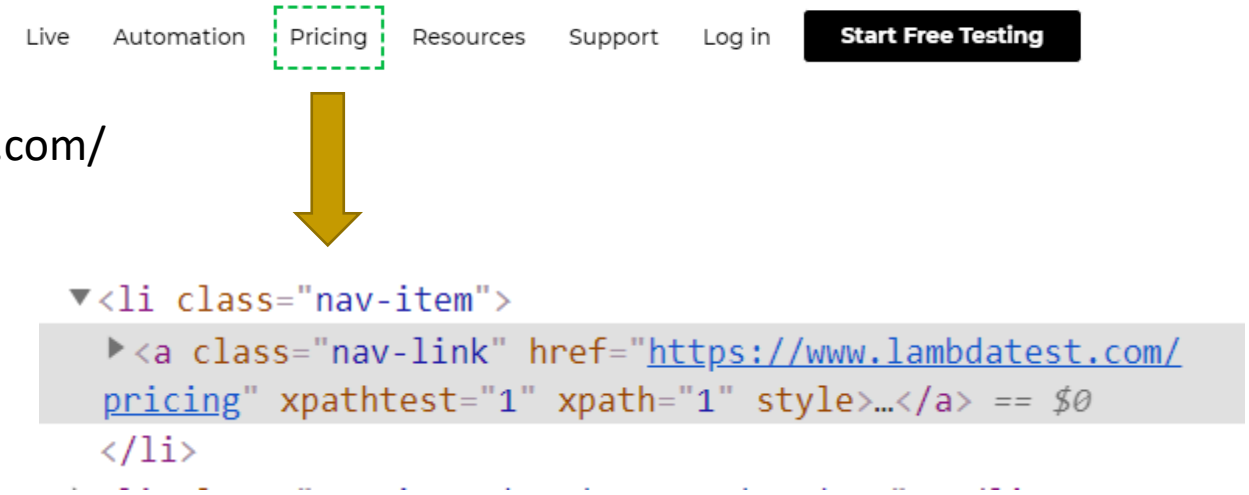
https://www.lambdatest.com/



driver.find_element(By.XPATH,"//a[starts-with(text(), 'Start')]")
driver.find_element(By.XPATH,"//a[starts-with(@name,'value')]")

XPath with Text()

https://www.lambdatest.com/



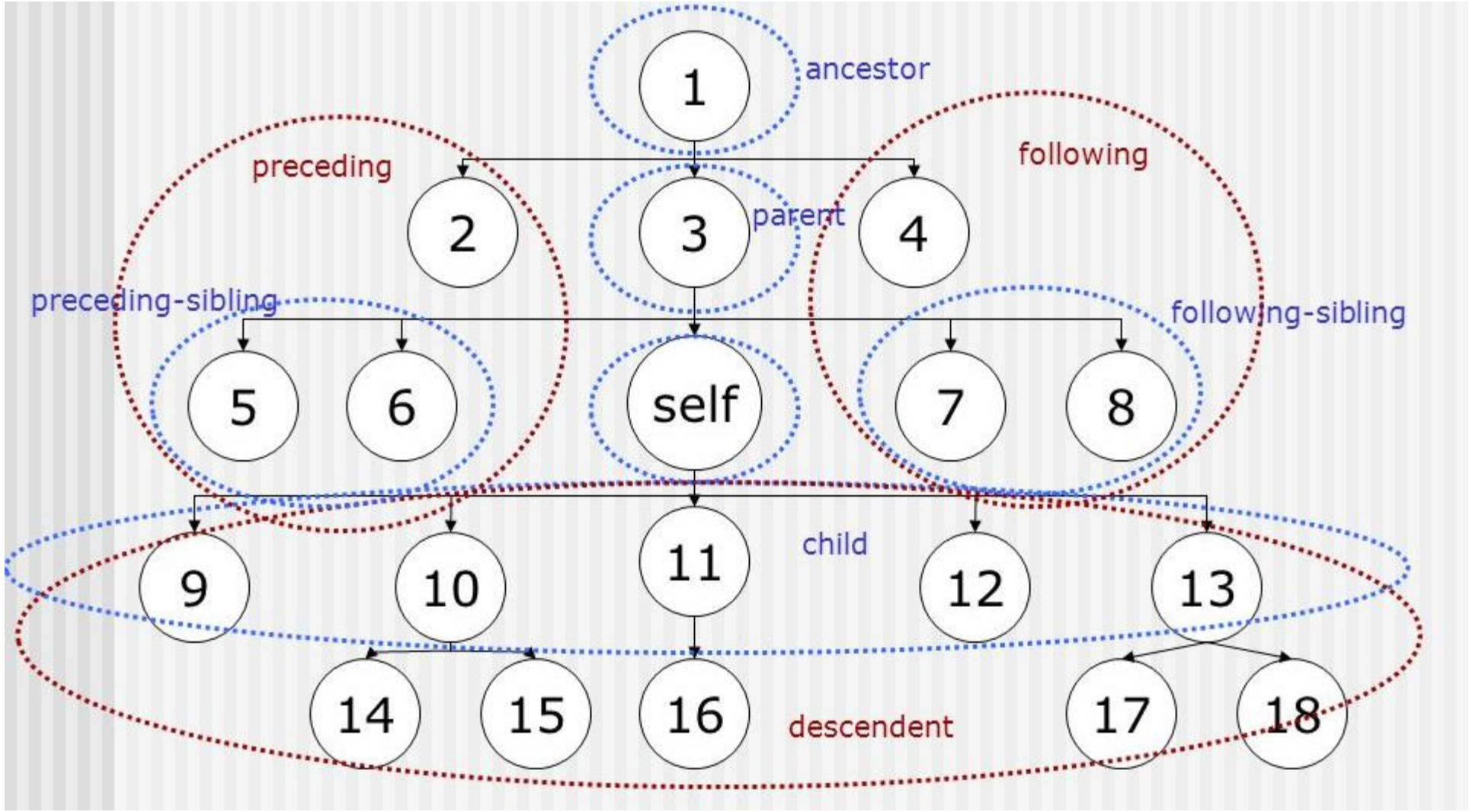
driver.find_element(By.XPATH,"//a[text()='Pricing']")

XPath Axes

XPath axes

- XPath axes are those axes that are used to search for the multiple nodes in the XML document from the current node context.
- These methods are mainly used when the web element is not identified with the help of ID, name, class name, link text, CSS selector and XPath, etc. locators.

Relationship of Nodes



XPath axes

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