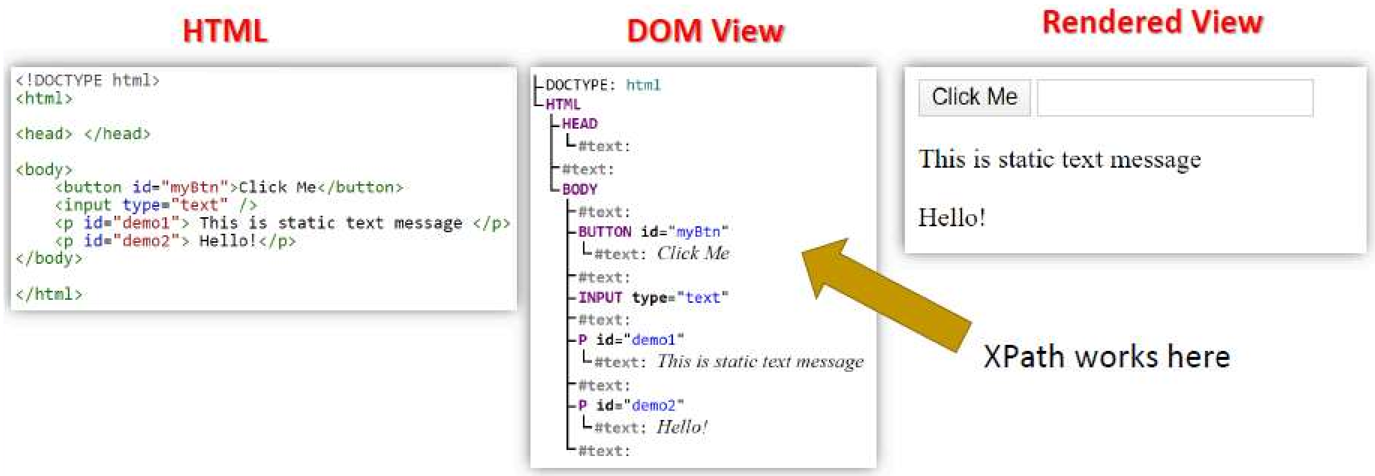
21. Customized Locators - XPath and XPath Options

Xpath or XML Path

- Xpath finds dynamically changing elements on complex webpages.
- Xpath is an **address or syntax or language** for finding any element on the web page using **XML path expression** on **HTML DOM Structure.**
- XPath can be used to navigate through **nodes** to find elements in HTML DOM Structure.

Document Object Model (DOM)

- DOM is an api interface provided by browser. Browser generates DOM when a web page is loaded.



```
<!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>
```

```
├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:
```

Other locators may fail to find elements but **XPATH (Most preferred, Effective)** will definitely identify elements whichever it is,wherever it is on the webpage.

Types of xpath

Absolute XPATH	Relative XPATH (Mostly Preferred)
Start with /	Start with //
starts from root html node so long XPATH if long web page	Starts from middle of HTML DOM Structure so no long XPATH even though long web page
We use only tags/nodes	We use tags/nodes and atleast one attribute
Syntax /parentElement/childElement/grandchildElement	**Syntax** //tagname[@attribute='value'] **or** //*[@attribute='value']
Ex: (1). /html/body/nav/div/div[2]/ul[3]/li[1]/a (2). /html/body/div[1]/div/div[3]/div[1]/img	Ex: (1). //*[@id="header-navbar"]/ul[3]/li[1]/a (2). //*[@id="divLogo"]/img

Why Relative XPATH is most preferred or Why Absolute XPATH is unstable

- If a developer introduces a new element or changes the location or path of element then the absolute xpath will be broken.So Absolute XPATH is Unstable and Relative XPATH is preferred.

XPATHLocator.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
driver.get("https://demowebshop.tricentis.com/")
driver.maximize_window()
driver.implicitly_wait(10)
```

Absolute XPath (full XPath)

```
logo = driver.find_element(By.XPATH, "/html[1]/body/div[4]/div[1]/div[1]/div[1]/a/img")
print("Logo presence:", logo.is_displayed())  # true
```

Relative XPath (partial XPath)

```
logo = driver.find_element(By.XPATH, "//img[@alt='Tricentis Demo Web Shop']")
print("Logo presence:", logo.is_displayed())  # true
```

XPath with single attribute

```
driver.find_element(By.XPATH, "//input[@id='small-searchterms']").send_keys("T-shirts")
```

XPath with multiple attributes

```
driver.find_element(By.XPATH, "//input[@id='small-searchterms'][@value='Search
store']").send_keys("T-shirts")
driver.quit()
```

XPATH Options

XPath with or operator

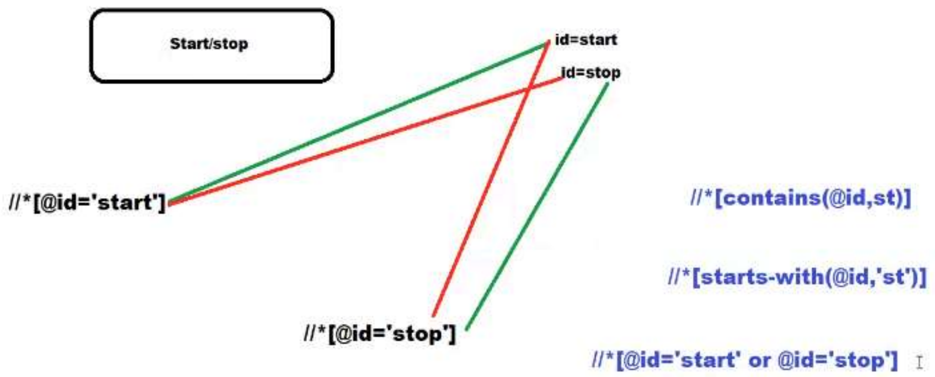
- XPath uses **or** operator to combine multiple attribute conditions, where only one of the conditions needs to be true for the element to be selected.

XPath with and operator

- XPath uses **and** operator to combine multiple attribute conditions that must all be true for the element to be selected.

XPath with contains() & XPath with starts-with()

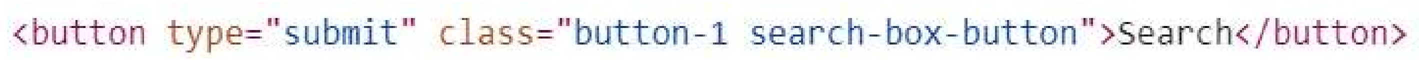
- contains() and starts-with() are used for finding dynamically changing elements.



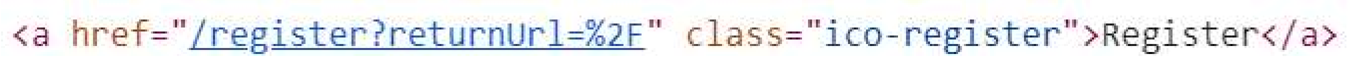
➔ **Example**

- ◆ `driver.find_element(By.XPATH,"//input[contains(@id,'email')]").send_keys("T-shirts")`
- ◆ `driver.find_element(By.XPATH,"//button[starts-with(@id,'login')]").click()`

XPath with text()



```
<button type="submit" class="button-1 search-box-button">Search</button>
```



```
<a href="/register?returnUrl=%2F" class="ico-register">Register</a>
```

Inner text : Search **and** **Linktext :** Register

➔ **text()** method identifies an element by using Inner Text or Visible Text.

➔ Link Text is also an Inner Text or Visible Text.

➔ **Example**

- ◆ `driver.find_element(By.XPATH,"//a[text()='Forgotten password?']").click()`

- driver.find_element(By.XPATH,"//a[contains(text(),'Forgotten password?')]").click()

XPath with last() Function

➔ Selects the last element in a set of matching nodes.

➔ **XPath Format** ⇒ //input[last()]

➔ The last() function is useful when you want the last occurrence of an element.

XPath with position() Function

➔ Selects an element based on its position in the list of matching nodes.

➔ **XPath Format** ⇒ //input[position()=2]

XPATHOptions.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
driver.get("https://demowebshop.tricentis.com/")
driver.maximize_window()
driver.implicitly_wait(10)
```

XPath with 'or' operator

```
driver.find_element(By.XPATH, "//input[@id='small-searchterms' or
@value='xyz']").send_keys("T-shirts")
```

XPath with 'and' operator

```
driver.find_element(By.XPATH, "//input[@id='small-searchterms' and @value='Search
store']").send_keys("T-shirts")
```

XPath with contains()

```
product_contains = driver.find_elements(By.XPATH, "//h2//a[contains(@href, 'computer')]")
print("Number of computer-related products:", len(product_contains))  # 4
```

XPath with starts-with()

```
product_startwith = driver.find_elements(By.XPATH, "//h2//a[starts-with(@href, '/build')]")
print("Number of build products:", len(product_startwith))  # 3
```

XPath with text()

```
register_link = driver.find_element(By.XPATH, "//a[text()='Register']")
print("Register link presence:", register_link.is_displayed())  # true
```

XPath with last()

```
googleplus_linktext = driver.find_element(By.XPATH, "//div[@class='column follow-us']//li[last()]").text
print(googleplus_linktext)  # Google+
```

XPath with position()

```
twitter_text = driver.find_element(By.XPATH, "//div[@class='column follow-us']//li[position()=2]").text
print(twitter_text)  # Twitter
driver.quit()
```

XPath with Normalize space()

➔ All leading, trailing white spaces are removed in the string.

➔ Within the string, any sequence of whitespace characters is replaced with a single space.

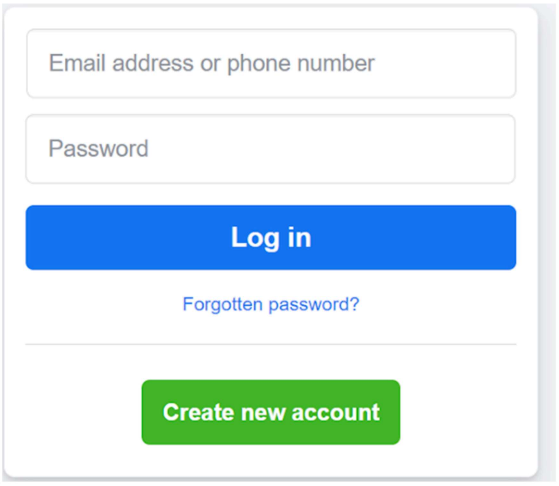
➔ Removes all new lines and tabs present in a string

➔ Removes all leading and trailing white spaces from the text content or attribute value.

➔ Replaces any sequence of whitespace characters within the text content or attribute value with a single space.

➔ Eliminates all newlines and tab characters from the text content or attribute value.

- **Syntax**
  - **//tagname[normalize-space(@attribute) = "attribute value"]** ⇒ Processes the attribute value of the @attribute and compares it with "attribute value
  - **//tagname[normalize-space(.)]** ⇒ Processes the text content of the current node after normalizing spaces.
  - **//tagname[normalize-space(' Sample ') = 'Sample']** ⇒ Normalizes the literal string 'Sample ' (removes extra spaces) and checks if it equals 'Sample'.



```
▼<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">
 </div>
▼<div class="_6lux">
  ▼<div class="_6luy _55r1 _1kbt" id="passContainer">
     <input type="password" class="inputtext _55r1 _6luy _9npi" name="pass" id="pass"
     data-testid="royal_pass" placeholder="Password" aria-label="Password">
```

➔ **Example**

- **//input[normalize-space(@class)="inputtext _55r1 _6luy"]**
- **//a[normalize-space()="Contact uploading and non-users"]** ⇒ footer section

SelectorsHub

1. SelectorsHub is XPath and CSS Selector plugin for popular browsers like Chrome, Firefox and Edge.
2. It assists in auto-generating, validating, and debugging XPath, CSS Selectors, and other locators such as SVG and iframe elements for web elements in automation testing.
3. It is highly useful for testers and developers working with Selenium or other web automation frameworks.