Comprehensive XPath Cheatsheet for Selenium

Table of Contents

---

1. XPath Basics

Absolute vs Relative XPath

Absolute XPath (starts from root - NOT RECOMMENDED)

/html/body/div[1]/div[2]/form/input

Relative XPath (starts from anywhere - RECOMMENDED)

//input[@id='username']
//div[@class='container']//button

Basic Syntax

Syntax	Description	Example
`//`	Select nodes anywhere in document	`//input`
`/`	Select direct child	`//div/input`

`.`	Current node	`.//button`
`..`	Parent node	`//input/..`
`@`	Select attribute	`//input[@name='email']`
`*`	Any element	`//*[@id='submit']`
`|`	Union (OR)	`//input | //button`

---

2. XPath Axes

Forward Axes

Axis	Description	Example
`child::`	Direct children	`//div/child::input`
`descendant::`	All descendants	`//form/descendant::input`
`following::`	All nodes after closing tag	`//h1/following::div`
`following-sibling::`	Siblings after current node	`//label/following-sibling::input`

Backward Axes

Axis	Description	Example
`parent::`	Parent node	`//input/parent::div`
`ancestor::`	All ancestors	`//input/ancestor::form`
`preceding::`	All nodes before opening tag	`//div/preceding::h1`
`preceding-sibling::`	Siblings before current node	`//input/preceding-sibling::label`

Other Axes

Axis	Description	Example

`self::`	Current node itself	`//input/self::input`
`attribute::`	Attributes of current node	`//input/attribute::name`
`descendant-or-self::`	Current node + descendants	`//form/descendant-or-self::*`
`ancestor-or-self::`	Current node + ancestors	`//input/ancestor-or-self::div`

Axis Shortcuts

// = /descendant-or-self::node()/
@ = attribute::
. = self::node()
.. = parent::node()

---

3. Predicates & Filters

Basic Predicates

```
// First element
//div[1]
//button[first()]

// Last element
//div[last()]

// Second last
//div[last()-1]

// Position less than 3
//li[position() < 3]

// Elements with specific attribute
//input[@type='text']

// Multiple conditions (AND)
//input[@type='text' and @name='username']

// Multiple conditions (OR)
//input[@type='text' or @type='email']
```

```
// NOT condition
//input[not(@disabled)]
```

Attribute Filters

```
// Exact match
//div[@class='container']

// Contains
//div[contains(@class, 'btn')]

// Starts with
//input[starts-with(@id, 'user')]

// Ends with (XPath 2.0)
//input[ends-with(@name, 'email')]

// Multiple classes
//div[contains(@class, 'btn') and contains(@class, 'primary')]
```

---

4. XPath Functions

String Functions

```
// contains() - Most used
//button[contains(text(), 'Submit')]
//div[contains(@class, 'error')]

// starts-with()
//input[starts-with(@id, 'user')]
//a[starts-with(@href, 'https://')]

// normalize-space() - Removes extra whitespace
//button[normalize-space(text())='Submit']
//div[normalize-space()='Welcome User']

// concat() - Concatenate strings
//div[@class=concat('btn', '-', 'primary')]

// substring()
//div[substring(@id, 1, 4)='user']
```

```
// string-length()
//input[string-length(@value) > 5]

// translate() - Replace characters
//input[translate(@name, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'abcdefghijklmnopqrstuvwxyz')='username']
```

Boolean Functions

```
// not()
//input[not(@disabled)]
//div[not(contains(@class, 'hidden'))]

// true() and false()
//input[@required=true()]
```

Number Functions

```
// count() - Count elements
//div[count(.//input) > 2]

// sum()
//div[sum(.//input/@value) > 100]

// number()
//input[@tabindex=number('1')]
```

Node Functions

```
// text() - Text content
//button[text()='Submit']
//div[text()='Welcome']

// name() - Element name
//*[name()='input']

// local-name()
//*[local-name()='input']

// position()
//li[position()=2]
```

```
// last()
//li[last()]
```

---

5. Operators

Comparison Operators

```
// Equal
//input[@type='text']

// Not equal
//input[@type!='hidden']

// Greater than
//input[@tabindex > 0]

// Less than
//div[@data-count < 10]

// Greater than or equal
//input[@maxlength >= 5]

// Less than or equal
//input[@minlength <= 20]
```

Logical Operators

```
// AND
//input[@type='text' and @required]

// OR
//input[@type='text' or @type='email']

// NOT (using not() function)
//input[not(@disabled)]
```

Arithmetic Operators

```
// Addition
//div[@data-value=5+5]
```

```
// Subtraction
//li[position()=last()-1]

// Multiplication
//div[@width=10*2]

// Division
//div[@height=100 div 2]

// Modulo
//li[position() mod 2 = 0]  // Even positions
```

---

6. Common Patterns for Selenium

Finding Elements by Common Attributes

```
// By ID
//*[@id='username']

// By Name
//input[@name='email']

// By Class (single)
//*[@class='btn-primary']

// By Class (contains - for multiple classes)
//*[contains(@class, 'btn-primary')]

// By Type
//input[@type='password']

// By Placeholder
//input[@placeholder='Enter email']

// By Value
//input[@value='Submit']

// By Title
//a[@title='Home Page']

// By Href
//a[@href='/login']
```

```
// By Data attributes
//*[@data-test='login-button']
//*[@data-testid='submit']
```

Text-Based Selection

```
// Exact text match
//button[text()='Login']
//span[text()='Welcome']

// Contains text
//button[contains(text(), 'Submit')]
//div[contains(text(), 'Error')]

// Text with whitespace normalized
//button[normalize-space()='Submit Form']

// Partial text match (case-insensitive using translate)
//button[contains(translate(text(), 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'abcdefghijklmnopqrstuvwxyz'), 'submit')]
```

Parent-Child Relationships

```
// Direct child
//form/input
//div[@class='container']/button

// Any descendant
//form//input
//div[@class='container']//button

// Parent from child
//input[@id='username']/..
//button[@type='submit']/parent::div

// Ancestor
//input[@id='username']/ancestor::form
//button/ancestor::div[@class='modal']
```

Sibling Navigation

```
// Following sibling
```

```
//label[@for='username']/following-sibling::input
//div[@class='error']/following-sibling::div[1]

// Preceding sibling
//input[@id='password']/preceding-sibling::label

// All siblings
//li[@class='active']/following-sibling::li
```

Index-Based Selection

```
// First element
(//input)[1]
//div[@class='item'][1]

// Last element
(//input)[last()]
//div[@class='item'][last()]

// Second element
(//input)[2]

// Third from last
(//input)[last()-2]

// Range of elements
//li[position() >= 2 and position() <= 4]
```

---

7. Dynamic Elements

Handling Dynamic IDs

```
// ID starts with
//input[starts-with(@id, 'user')]

// ID ends with
//input[substring(@id, string-length(@id) - string-length('name') +1) = 'name']

// ID contains
//input[contains(@id, 'username')]

// Multiple ID parts
```

```
//*[contains(@id, 'user') and contains(@id, 'input')]
```

Handling Dynamic Classes

```
// Class contains
//*[contains(@class, 'btn-primary')]

// Multiple class conditions
//*[contains(@class, 'btn') and contains(@class, 'primary')]

// Class starts with
//*[starts-with(@class, 'btn-')]
```

Dynamic Attributes

```
// Attribute contains
//*[contains(@data-id, 'user')]

// Multiple dynamic attributes
//*[contains(@id, 'submit') and contains(@class, 'btn')]

// Attribute exists (regardless of value)
//*[@data-test]
```

Using Multiple Attributes

```
// Two attributes
//input[@type='text' and @name='username']

// Three or more attributes
//button[@type='submit' and @class='btn-primary' and text()='Login']

// Attribute combination with text
//span[@class='label' and contains(text(), 'Email')]
```

---

8. Advanced Techniques

Excluding Elements

```
// Exclude by attribute
//div[not(@class='hidden')]
```

```
// Exclude by text
//li[not(text()='Home')]

// Exclude specific element
//input[not(@type='hidden')]

// Exclude multiple conditions
//div[not(@class='hidden' or @style='display: none;')]
```

Chaining Conditions

```
// AND conditions
//input[@type='text' and @required and @name='email']

// OR conditions
//input[@type='text' or @type='email' or @type='password']

// Mixed AND/OR
//input[(@type='text' or @type='email') and @required]

// Nested predicates
//div[@class='form-group'][.//input[@required]]
```

Working with Tables

```
// All table rows
//table//tr

// Specific column in all rows
//table//tr/td[2]

// Row by text content
//table//tr[td[text()='John Doe']]

// Cell by row and column
//table//tr[2]/td[3]

// Row containing specific text in any cell
//table//tr[td[contains(text(), 'Active')]]

// Row where first cell equals and get third cell
//table//tr[td[1][text()='John']]/td[3]
```

Working with Lists

```
// All list items
//ul/li

// List item containing text
//ul/li[contains(text(), 'Item')]

// Nth list item
//ul/li[3]

// First list item
//ul/li[1]

// Last list item
//ul/li[last()]

// Even list items
//ul/li[position() mod 2 = 0]

// Odd list items
//ul/li[position() mod 2 = 1]
```

Working with Forms

```
// All inputs in a form
//form[@id='loginForm']//input

// Required fields only
//form//input[@required]

// All buttons in form
//form[@name='register']//button

// Form by its action
//form[@action='/submit']

// Form field by label
//label[text()='Email']/following-sibling::input
//label[contains(text(), 'Password')]/../input
```

Working with Dropdowns

// Select element
//select[@id='country']

// Option by text
//select[@id='country']/option[text()='India']

// Option by value
//select[@id='country']/option[@value='IN']

// Selected option
//select[@id='country']/option[@selected]

// All options
//select[@id='country']/option

---

9. Best Practices

DO's

✅ Use relative XPath
//button[@id='submit']

✅ Use unique attributes (id, name, data-testid)
//*[@data-testid='login-button']

✅ Be specific but flexible
//div[@class='modal']//button[text()='Close']

✅ Use contains() for partial matches
//div[contains(@class, 'error-message')]

✅ Combine multiple attributes
//input[@type='text' and @name='username']

✅ Use text() wisely
//button[normalize-space(text())='Submit']

✅ Parent-child relationships when needed
//form[@id='login']//button[@type='submit']

DON'Ts

❌ Avoid absolute XPath
/html/body/div[1]/div[2]/form/input[1]

❌ Avoid fragile index-based XPath
//div[1]/div[3]/div[2]/input[1]

❌ Don't rely solely on position
//div[5]/input[2]

❌ Avoid overly long XPath
//html/body/div/div/div/div/form/div/input

❌ Don't use hardcoded indices with dynamic content
//table/tr[15]/td[3]

❌ Avoid = for class with multiple values
//div[@class='btn btn-primary'] // May fail
Use: //div[contains(@class, 'btn-primary')] // Better

---

10. Real-World Examples

Login Form

```
// Username field
//input[@id='username']
//input[@name='username']
//input[@placeholder='Enter username']

// Password field
//input[@type='password']
//input[@id='password']

// Login button
//button[@type='submit']
//button[text()='Login']
//input[@value='Login']

// Remember me checkbox
//input[@type='checkbox' and @name='remember']

// Forgot password link
```

```
//a[contains(text(), 'Forgot Password')]
```

E-commerce Product Page

```
// Product title
//h1[@class='product-title']
//div[@class='product-info']//h1

// Product price
//span[@class='price']
//*[contains(@class, 'product-price')]

// Add to cart button
//button[contains(text(), 'Add to Cart')]
//button[@data-action='add-to-cart']

// Product quantity input
//input[@type='number' and @name='quantity']

// Size selector
//select[@id='size']
//button[contains(@class, 'size-option')][text()='M']

// Product rating
//div[@class='rating']//span[contains(@class, 'stars')]
```

Data Table

```
// Header row
//table[@id='dataTable']//thead//tr

// All data rows
//table[@id='dataTable']//tbody//tr

// Specific row by index
//table[@id='dataTable']//tbody//tr[5]

// Row containing specific text
//table//tr[td[contains(text(), 'John Doe')]]

// Get cell value from row with specific data
//table//tr[td[1][text()='ID-123']]/td[3]
```

```
// All checkboxes in table
//table//tbody//input[@type='checkbox']

// Checkbox for specific row
//table//tr[td[text()='John Doe']]//input[@type='checkbox']
```

Navigation Menu

```
// Main menu
//nav[@id='main-menu']//ul

// Menu item by text
//nav//a[text()='Home']
//nav//li[contains(text(), 'Products')]

// Active menu item
//nav//li[contains(@class, 'active')]

// Submenu
//nav//li[a[text()='Products']]//ul

// Dropdown menu item
//nav//li[contains(@class, 'dropdown')]//a[text()='Settings']
```

Modal/Dialog

```
// Modal container
//div[contains(@class, 'modal') and contains(@class, 'show')]

// Modal title
//div[@class='modal-header']//h3
//div[contains(@class, 'modal')]//h2

// Modal close button
//button[@data-dismiss='modal']
//button[contains(@class, 'close')]
//div[@class='modal-header']//button

// Modal confirm button
//div[@class='modal-footer']//button[text()='Confirm']
```

Search and Filters

```
// Search input
//input[@type='search']
//input[@placeholder='Search...']

// Search button
//button[@type='submit' and contains(@class, 'search')]

// Filter checkboxes
//div[@class='filters']//input[@type='checkbox']

// Category filter
//select[@name='category']

// Price range filter
//input[@name='min-price']
//input[@name='max-price']

// Apply filters button
//button[text()='Apply Filters']
```

Alert Messages

```
// Success message
//div[contains(@class, 'alert-success')]
//div[contains(@class, 'success') and contains(text(), 'successful')]

// Error message
//div[contains(@class, 'alert-danger')]
//span[contains(@class, 'error-message')]

// Warning message
//div[contains(@class, 'alert-warning')]

// Info message
//div[contains(@class, 'alert-info')]

// Toast notification
//div[contains(@class, 'toast')]
```

File Upload

```
// File input
//input[@type='file']
```

// Upload button
//button[contains(text(), 'Upload')]

// File name display
//span[@class='file-name']

// Remove file button
//button[contains(@class, 'remove-file')]

Pagination

// Current page
//li[contains(@class, 'active')]//a
//span[@class='current-page']

// Next button
//a[contains(text(), 'Next')]
//button[@aria-label='Next page']

// Previous button
//a[contains(text(), 'Previous')]
//button[@aria-label='Previous page']

// Specific page number
//a[text()='5']
//button[text()='3']

// Last page
//a[contains(@class, 'last-page')]

Selenium Code Examples

Python (Selenium)

```
from selenium import webdriver
from selenium.webdriver.common.by import By

# Find single element
element = driver.find_element(By.XPATH, "//input[@id='username']")

# Find multiple elements
```

```
elements = driver.find_elements(By.XPATH, "//div[@class='item']")

# Click element
driver.find_element(By.XPATH, "//button[text()='Submit']").click()

# Send keys
driver.find_element(By.XPATH, "//input[@name='email']").send_keys("test@example.com")

# Get text
text = driver.find_element(By.XPATH, "//div[@class='message']").text

# Get attribute
value = driver.find_element(By.XPATH, "//input[@id='username']").get_attribute("value")

# Wait for element
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

element = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.XPATH, "//div[@id='result']"))
)
```

Java (Selenium)

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

// Find single element
WebElement element = driver.findElement(By.xpath("//input[@id='username']"));

// Find multiple elements
List<WebElement> elements = driver.findElements(By.xpath("//div[@class='item']"));

// Click element
driver.findElement(By.xpath("//button[text()='Submit']")).click();

// Send keys
driver.findElement(By.xpath("//input[@name='email']")).sendKeys("test@example.com");

// Get text
String text = driver.findElement(By.xpath("//div[@class='message']")).getText();

// Get attribute
```

```
String value = driver.findElement(By.xpath("//input[@id='username']")).getAttribute("value");

// Wait for element
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement element = wait.until(ExpectedConditions.presenceOfElementLocated(
    By.xpath("//div[@id='result']")
));
```

Quick Reference Card

Most Commonly Used XPaths

1. //tagname[@attribute='value']
2. //tagname[text()='text']
3. //tagname[contains(@attribute, 'value')]
4. //tagname[contains(text(), 'text')]
5. //tagname[starts-with(@attribute, 'value')]
6. //parent/child
7. //parent//descendant
8. //element/following-sibling::element
9. //element/parent::element
10. //element[1]

Performance Tips

1. **Use ID when available** - Fastest locator
2. **Avoid `//` at the start if possible** - Use specific parent
3. **Don't use too many `//`** - Be specific with hierarchy
4. **Use `contains()` sparingly** - Exact matches are faster
5. **Index-based selection is fast** - But fragile
6. **Combine attributes** - More specific = faster
7. **Use `@data-testid`** - Purpose-built for testing
8. **Cache complex XPaths** - Store in variables/constants

Troubleshooting XPath Issues

Element Not Found

- Verify XPath in browser DevTools ($x in console)
- Check if element is in iframe
- Wait for element to be present
- Verify element is not hidden
- Check if element is in shadow DOM

Multiple Elements Match

- Add more specific attributes
- Use index [1], [2], etc.
- Add parent context
- Use unique attributes

XPath Changes Frequently

- Avoid using indices
- Use stable attributes (data-*, id, name)
- Use text content if stable
- Use relative relationships
- Work with developers to add test IDs

Browser DevTools Testing

Chrome/Edge DevTools Console

```
// Test XPath
$x("//input[@id='username']")

// Test CSS Selector
$("input#username")

// Count elements
$x("//div[@class='item']").length

// Get element properties
$x("//input[@id='username']")[0].value
```

Firefox DevTools Console

```
// Same as Chrome
$x("//input[@id='username']")
```

Created by: QACanCode
Last Updated: 2025
Version: 1.0

For more automation resources, visit [qacancode.com]