

Generating reliable object selector using Spy Web utility

This article shows you some examples of generating reliable object selectors using Spy Web Utility.

Spy Web Utility provides an intelligent object capturing capability on websites. To learn more about this utility, see [Spy Web Utility](#).

There are two widely used selectors: CSS and XPath. Locators are object attributes that are used to build up XPath or CSS selectors. Locators help find and identify elements on the web page under test.

Based on the selected method, you can edit the Selected Locator to adjust the current selector of an object or manually add a selector using either an XPath or a CSS selector.

With the CSS or XPath mode of the Selection Method, you can further input and edit XPath or CSS objects to identify objects on web UI.

Below are some web element locators:

Web element locators	Description
Id	Select elements with the specified @id attribute.
Name	Select the first element with the specified @name attribute.

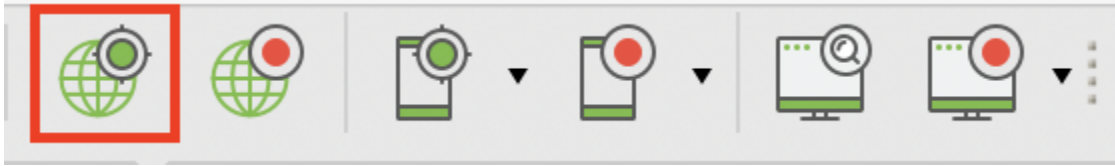
Link text	Select links (anchor tag) element which contains text matching the specified link text.
Partial link text	Select links (anchor tag) element which contains text matching the specified partial link text.
Tag name	Locate elements using a tag name.
Class name	Locate elements using a class name.
CSS	Select elements using CSS selectors.
XPath	Locate elements using an XPath expression.

How to find object locators?

You can capture objects, get web element XPath or CSS locators, and manually input XPath or CSS object selectors using Spy Web Utility. Spy Web Utility provides instant feedback by auto-detecting and highlighting all elements that match with the provided selector.

In this section, we will illustrate how to use the Spy Web Utility to find object locators.

1. Open or create a new project. From the main toolbar, click on the Spy Web icon.



The Object Spy window displays.

-
2. In the URL section, input the URL of the website you want to test. We use our demo page in this example: `http://demoaut.katalon.com/`



Object Spy

URL



Start ▾



New page



New frame



New object



Delete



Save

CAPTURED OBJECTS

OBJECT PROPERTIES

Object Name

Selection Method 

☒ XPath

☐ Attributes

☐ CSS

☐ Image

Name	Value

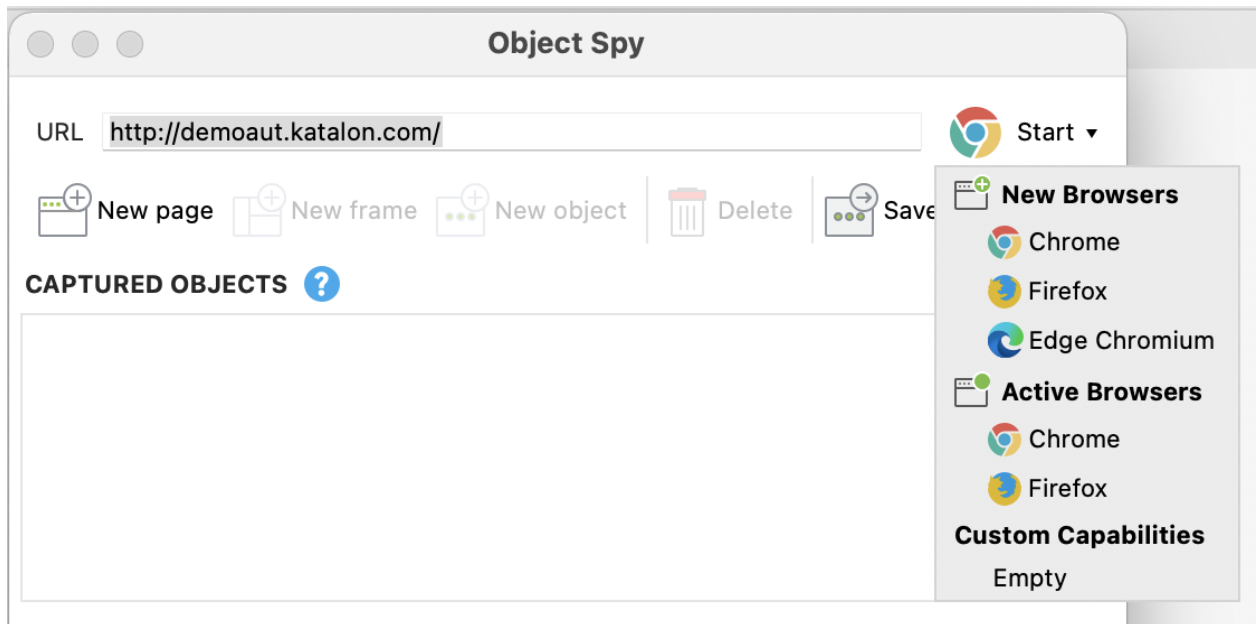
Selected Locator

Verify and Highlight

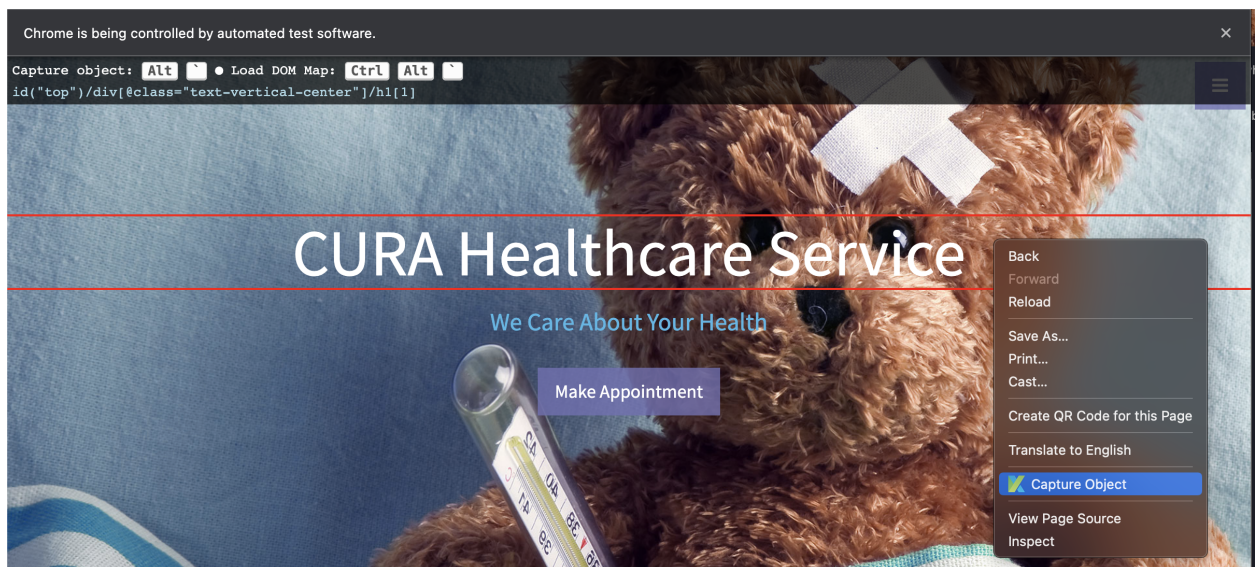
Add Screenshot



3. To choose a browser, click on the dropdown icon of the Start button. For this example, we choose a new Chrome browser.



4. Click Start. Katalon Studio opens a new Chrome browser and navigates to the demo website.
5. To capture test objects, first, hover the mouse over them. The focused web object is highlighted in red.



Then, by pressing <Alt + `> or right-clicking and choosing Capture Object, the focused object is captured and added to the Captured Objects list. Katalon Studio automatically captures all available properties of the objects. You can change the folder name and edit the value of any properties.

Object Spy

URL

http://demoaut.katalon.com/

 Start ▾

 New page

 New frame

 New object

 Delete

 Save

CAPTURED OBJECTS

▼

 Page_CURA Healthcare Service

 h1_CURA Healthcare Service

OBJECT PROPERTIES

Object Name

h1_CURA Healthcare Service

Selection Method

☒ XPath

☐ Attributes

☐ CSS

☐ Image

Name	Value
xpath:idRelative	//header[@id='top']/div/h1
xpath:neighbor	(.//*[normalize-space(text()) and normalize...
xpath:neighbor	(.//*[normalize-space(text()) and normalize...
xpath:neighbor	(.//*[normalize-space(text()) and normalize...
xpath:neighbor	(.//*[normalize-space(text()) and normalize...
xpath:position	//h1
xpath:customAttributes	//h1[(text() = 'CURA Healthcare Service' or...

Selected Locator

//header[@id='top']/div/h1

Verify and Highlight

Add Screenshot

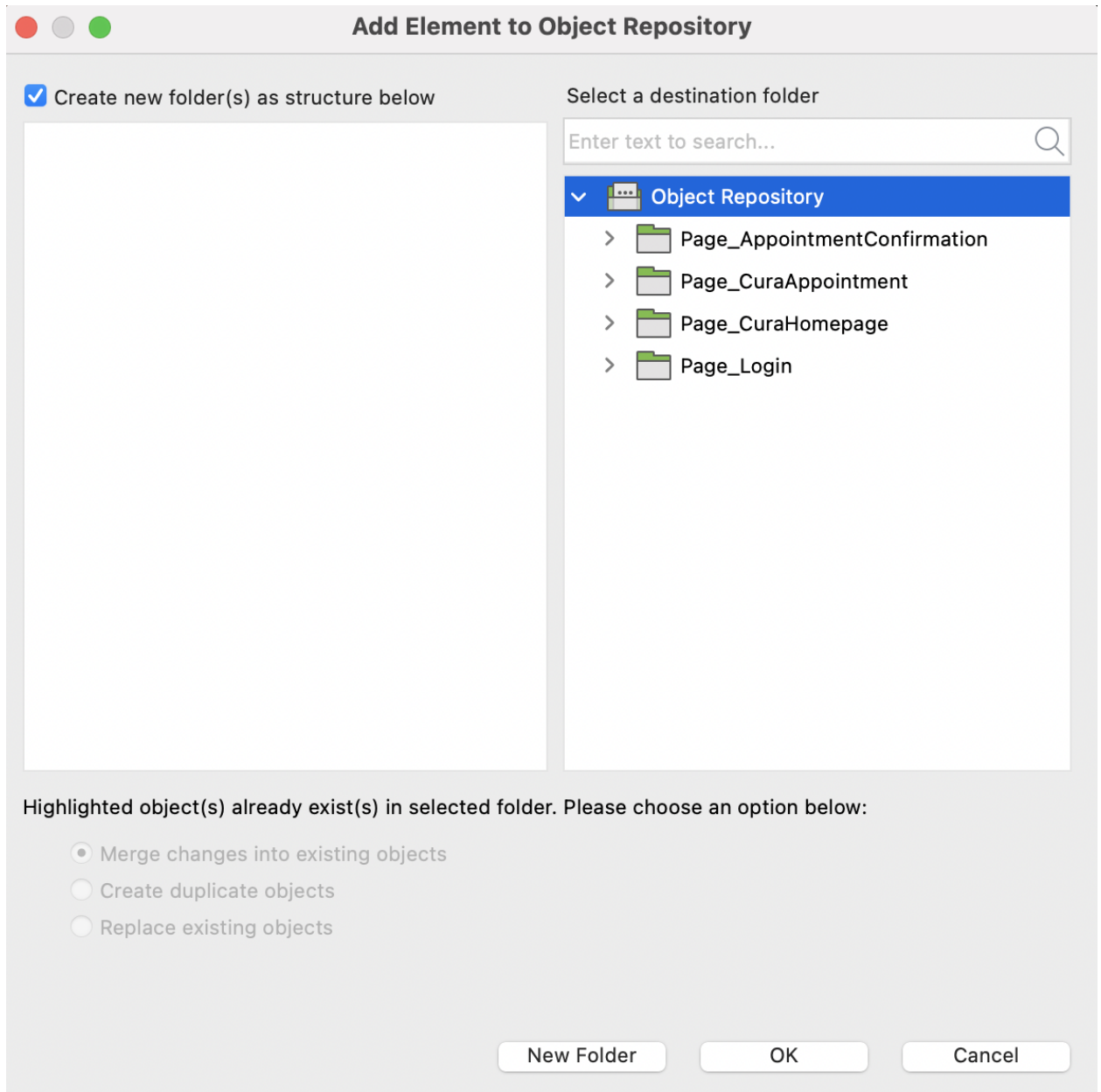
In the Selection Method section, you can choose XPath, Attributes, CSS, or Image to locate captured objects.

6. In the Selected Locator section, you can manually input your XPath, then click Verify and Highlight. Katalon finds elements that match your XPath locator and highlights them on the web page.

A single locator might identify many elements because a web page can have many elements that use the same formats and styles. The next section will guide you on how to create unique locators to find complex or dynamic elements. See below: [Working with XPath selector](#).

https://demo-store.katalon.com/

7. To save objects to the Objects Repository, click on Save. Then, select a folder to add the captured objects into. Click OK when done.



Working with XPath selector

The following guide shows how to leverage Katalon Studio Spy Web Utility to locate web elements with XPath selector.

These XPath axes methods are used to find complex or dynamic elements.

In the examples below, we are using our demo page:

<http://demoaut.katalon.com/>

You can find our web sample project available to be downloaded here: [Katalon Web Automation Sample Project](#).

Checking Multiple Attributes

You can identify an element with the combination of multiple attributes to build a unique locator. This method is useful when an element has multiple attributes, but using only one attribute cannot create a unique locator.

For example, you can identify the login button with multiple attributes, like `@id` and `@type`, to locate the Login button.

XPath:

```
//*[@id='btn-login'][@type='submit']
```

The screenshot displays the Katalon Studio interface with the 'Elements' tab selected. The DOM tree shows the structure of the login page, with the login button highlighted. The XPath selector `//*[@id='btn-login'][@type='submit']` is entered in the search bar at the bottom. The 'Object Spy' window on the right shows the properties of the selected element, including its name, selection method, and various XPath expressions.

Object Spy

URL: <https://katalon-demo-cura.herokuapp.com/>

CAPTURED OBJECTS

- Page_CURA Healthcare Service
 - button_Login

OBJECT PROPERTIES

Object Name	button_Login
Selection Method	XPath
Name	Value
xpath:attributes	//button[@id='btn-login']
xpath:idRelative	//section[@id='login']/div/div/div[2]/form/div
xpath:neighbor	(.//*[normalize-space(text()) and normalize-space()='username'])
xpath:neighbor	(.//*[normalize-space(text()) and normalize-space()='password'])
xpath:neighbor	(.//*[normalize-space(text()) and normalize-space()='login'])
xpath:position	//button
xpath:customAttributes	//button[@id='btn-login' and @type='submit']

Selected Locator

```
//*[@id='btn-login'][@type='submit']
```

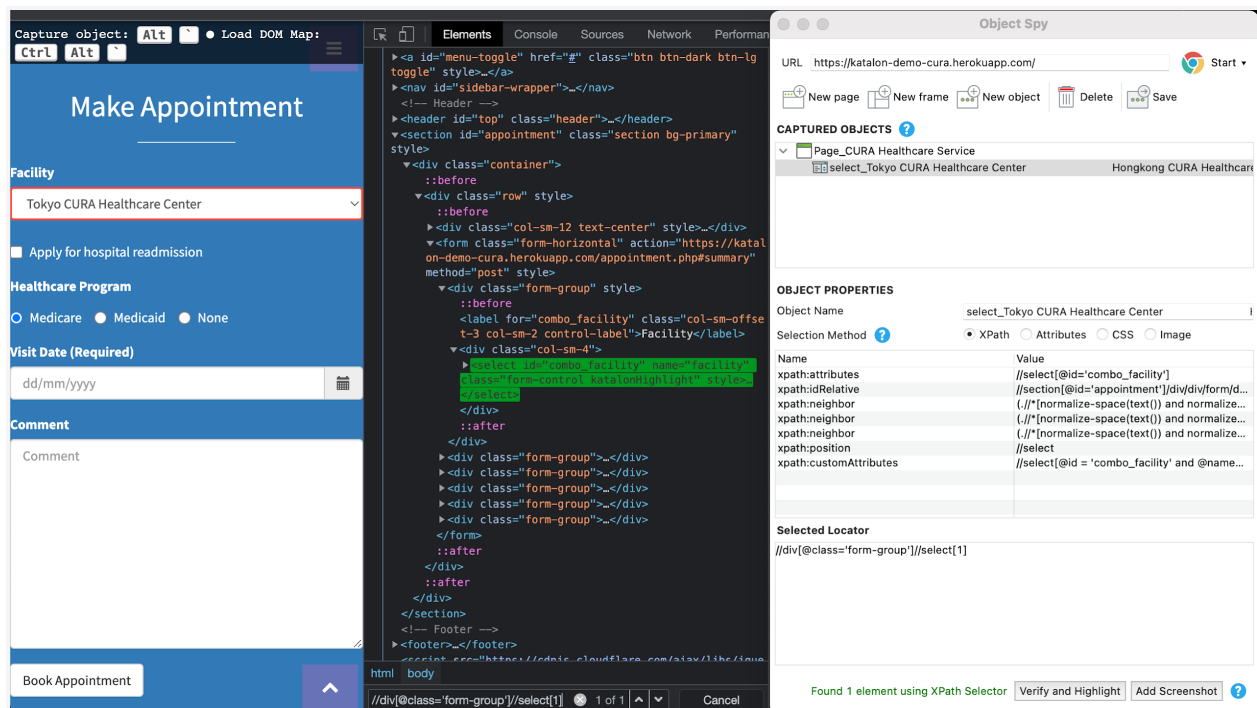
Found 1 element using XPath Selector [Verify and Highlight] [Add Screenshot]

Specify a given tag name in terms of the index value you wish to locate. Use this when there are more than one element present in the DOM with similar attributes and it becomes difficult to search them.

For example, you want to locate the first dropdown in a page.

XPath:

```
//div[@class='form-group']//select[1]
```



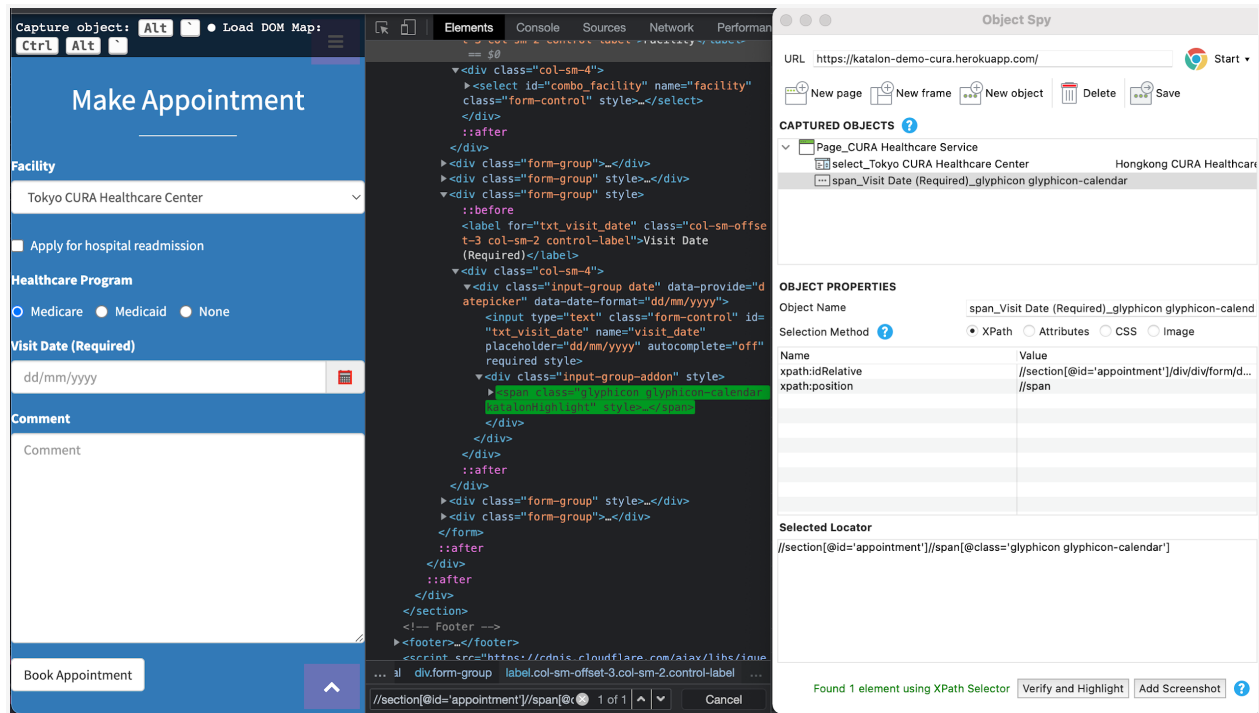
Chained XPath

You can use multiple XPath expressions and chain them.

For example, use the chained XPath to find the calendar icon in the appointment section.

XPath:

```
//section[@id='appointment']//span[@class='glyphicon glyphicon-calendar']
```



Contains()

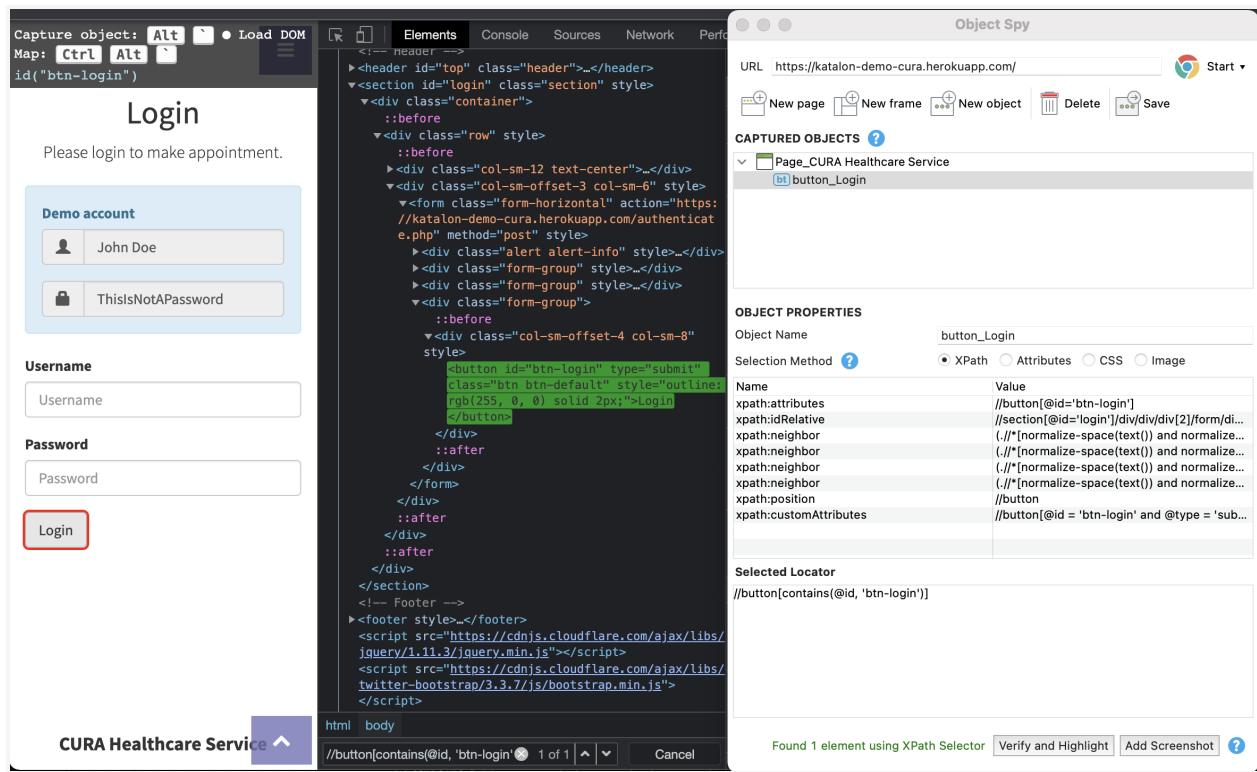
You can use the `Contains()` method to detect dynamic elements that contain static values.

Example:

Use the `Contains()` method to find the login button that contains the id: "btn-login".

XPath:

```
//button[contains(@id, 'btn-login')]
```



Last()

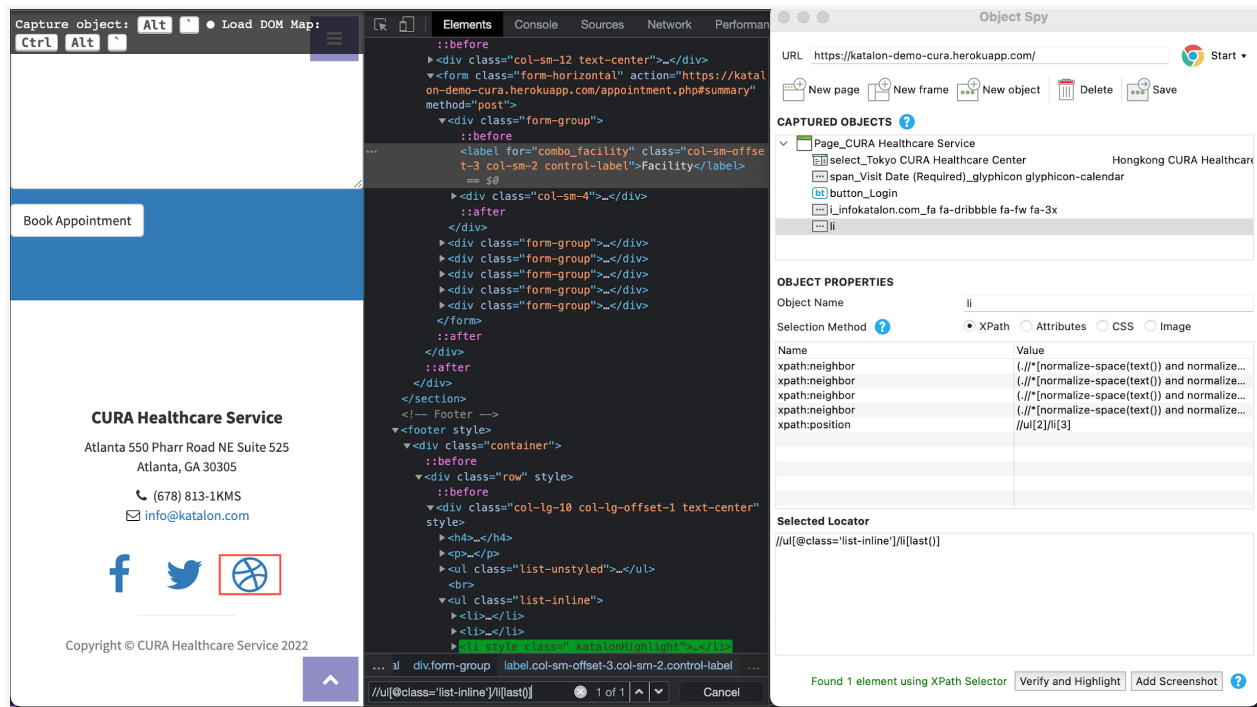
Last() is a method used in an XPath expression. It is used to get the very last node.

Example:

There are 3 social icon links, and you want to get the 3rd item by using Last().

XPath:

```
//ul[@class='list-inline']/li[last()]
```



Start-with()

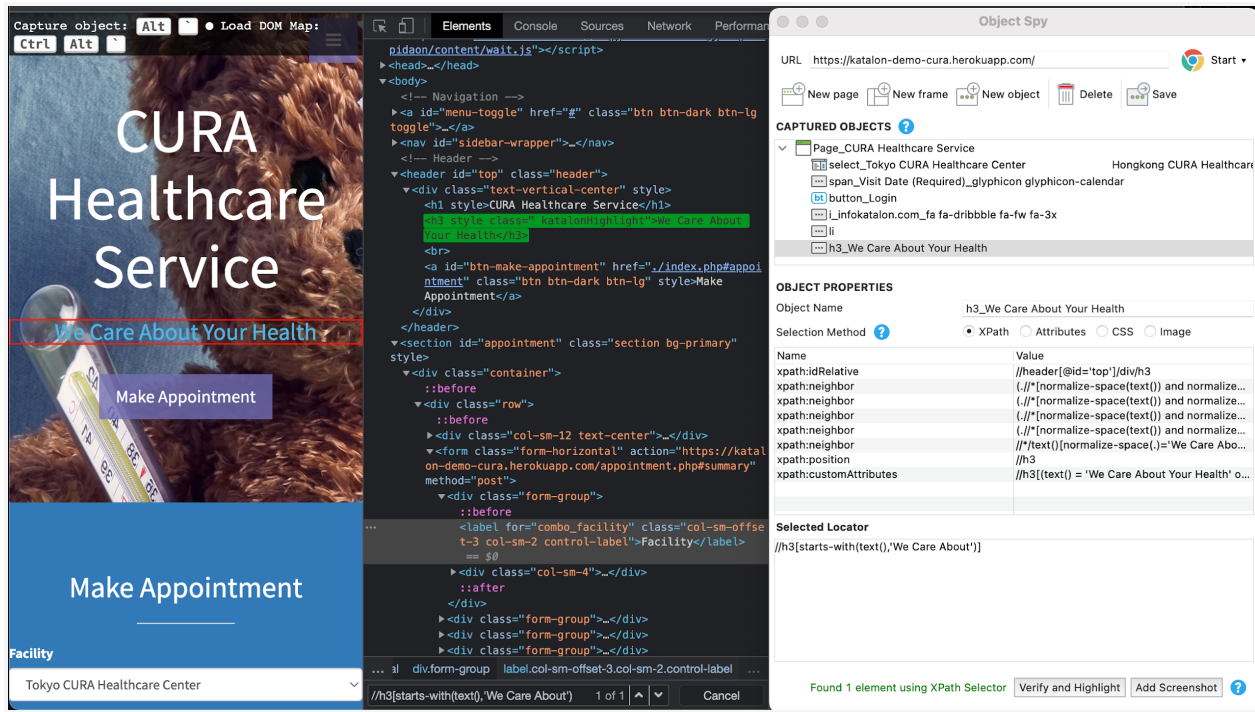
The Start-with method finds the element using the starting text of an attribute. This method is useful when the first part of the attribute value is fixed (static), and the rest is dynamic.

Example:

To find the line: "We Care About Your Health", use the Starts-with() method to find the line that starts with "We Care About".

XPath:

```
//h3[starts-with(text(),'We Care About')]
```



XPath axes methods

These XPath axes methods are used to find complex or dynamic elements.

Following

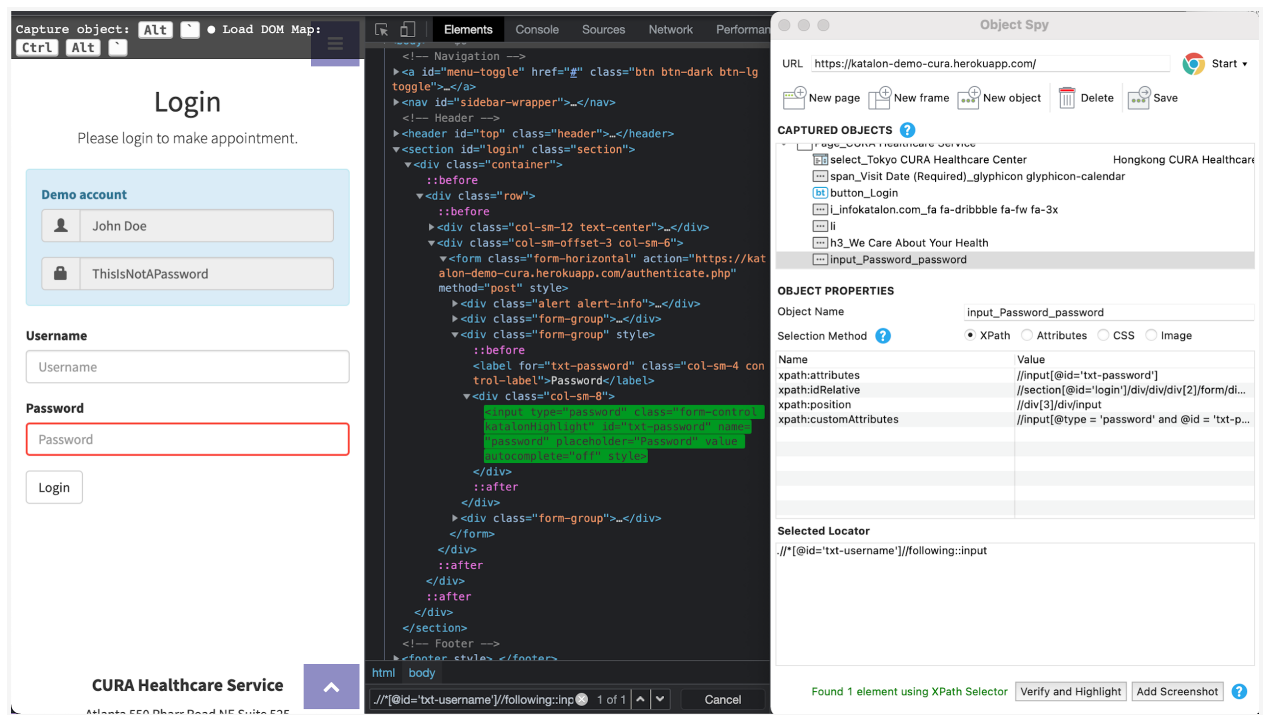
Selects all elements in the document following the current node().

Example:

By using the following method, you can identify the Password text box, which is located after the Username name field.

XPath:

```
//*[@id='txt-username']/following::input
```



Ancestor

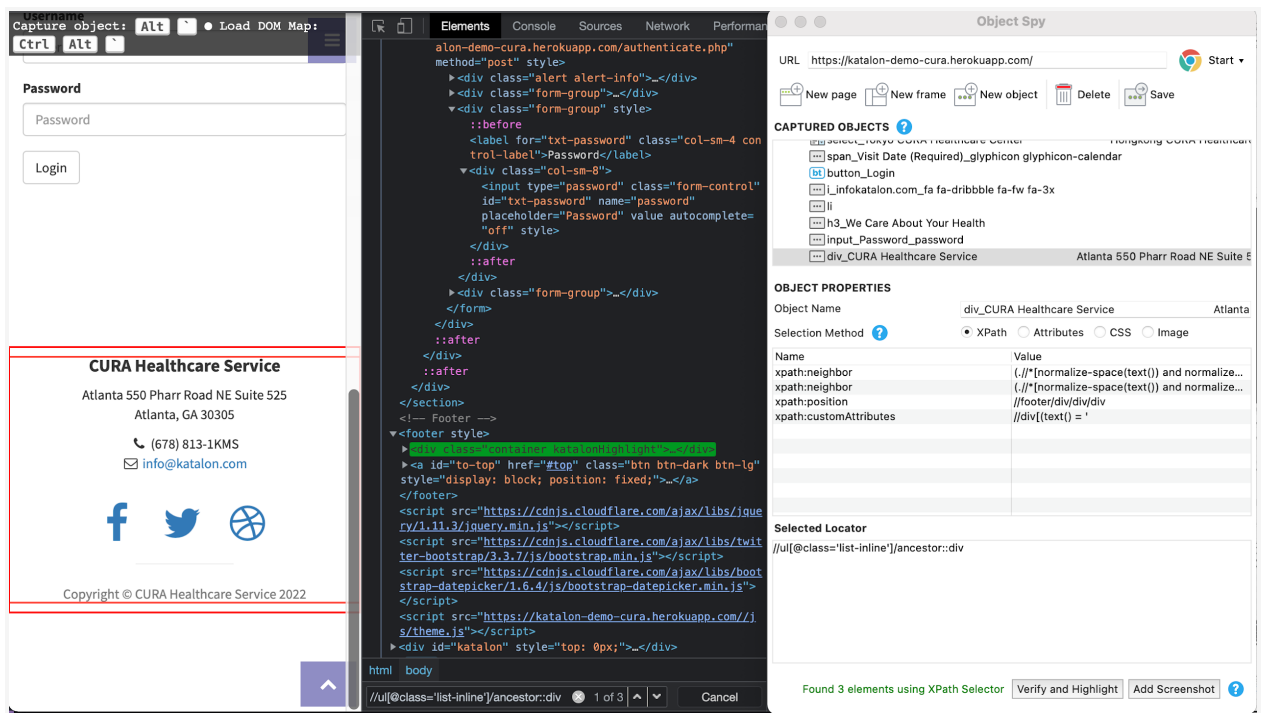
This method selects all ancestors (parent, grandparent, etc.) of the current node.

Example:

In the screenshot below, you want to get the ancestors of the `ul` tag highlighted in red.

XPath:

```
//*[@class='list-inline']/ancestor::div
```



Child

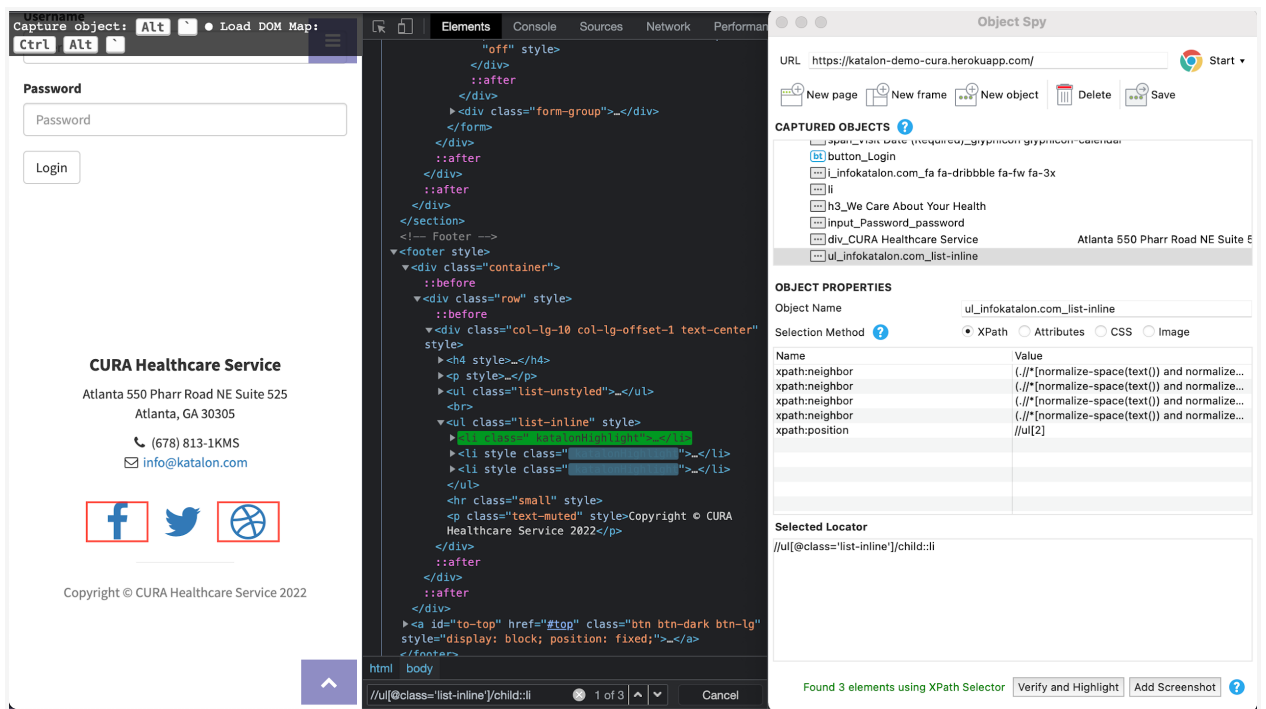
Select all children of the current node.

Example:

Using the child method, you can identify all social links as shown in the below screenshot.

XPath:

```
//ul[@class='list-inline']/child::li
```



Preceding

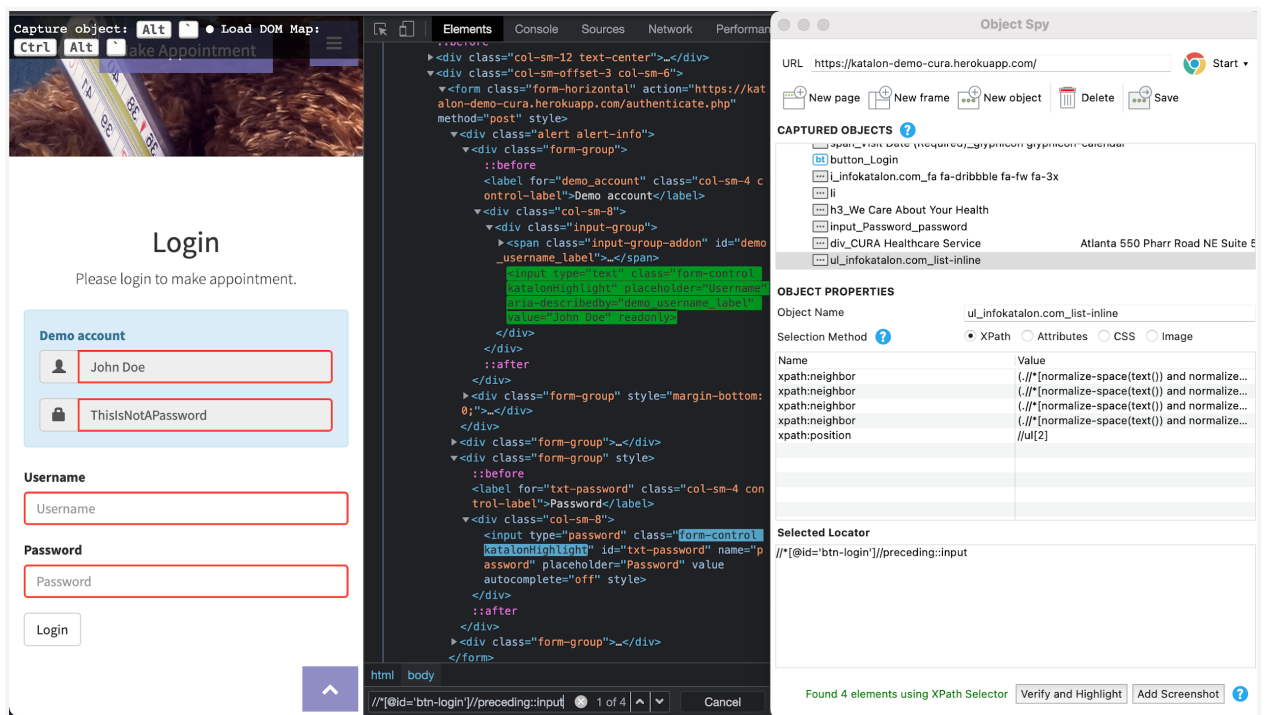
Select all nodes that come before the current node.

Example:

Using the preceding method, you can identify all nodes that come before the Login button.

XPath:

```
//*[ @id='btn-login']//preceding::input
```



Following-sibling

Select the following siblings of the context node. Siblings are at the same level as the current node, as shown in the screen below. It finds the element after the current node.

Example:

By using the following-sibling method, you can identify the Password text box which located after the Username name field.

XPath:

```
./.*[@id='txt-username']//following::input
```

Capture object: **Alt** Load DOM Map:

Ctrl **Alt**

Login

Please login to make appointment.

Demo account

John Doe

ThisIsNotAPassword

Username

Username

Password

Password

Login

Elements

Console

Sources

Network

Performance

```
<div class="col-sm-12 text-center"></div>
<div class="col-sm-offset-3 col-sm-6">
  <form class="form-horizontal" action="https://katalon-demo-cura.herokuapp.com/authenticate.php" method="post" style=
    <div class="alert alert-info">
      <div class="form-group">
        ::before
        <label for="demo_account" class="col-sm-4 control-label">Demo account</label>
        <div class="col-sm-8">
          <div class="input-group">
            <span class="input-group-addon" id="demo_username_label"></span>
            <input type="text" class="form-control" placeholder="Username" aria-describedby="demo_username_label" value="John Doe" readonly>
          </div>
        </div>
      </div>
    </div>
    <div class="form-group" style="margin-bottom: 0;"></div>
  <div class="form-group"></div>
  <div class="form-group" style=
    ::before
    <label for="txt-password" class="col-sm-4 control-label">Password</label>
    <div class="col-sm-8">
      <input type="password" class="form-control" id="txt-password" name="password" placeholder="Password" value="ThisIsNotAPassword" style="background-color: #f0f0f0;">
    </div>
    </div>
  </div>
  <div class="form-group"></div>
</form>
```

html body

//*[@id='txt-username']/following::input

1 of 1

Cancel

Object Spy

URL <https://katalon-demo-cura.herokuapp.com/>

New page New frame New object Delete Save

CAPTURED OBJECTS

alert_Visit Cura (required)_gryphicon_gryphicon-card-ruler

button_Login

ul_infokatalon.com_fa-fa-dribbble-fa-fw-fa-3x

h3_We Care About Your Health

input_Password_password

div_CURA Healthcare Service

ul_infokatalon.com_list-inline

Atlanta 550 Pharr Road NE Suite E

OBJECT PROPERTIES

Object Name

Selection Method ☒ XPath ☐ Attributes ☐ CSS ☐ Image

Name	Value
xpath:neighbor	(.//*[normalize-space(text()) and normalize...]
xpath:neighbor	(.//*[normalize-space(text()) and normalize...]
xpath:neighbor	(.//*[normalize-space(text()) and normalize...]
xpath:neighbor	(.//*[normalize-space(text()) and normalize...]
xpath:position	//ul[2]

Selected Locator

//*[@id='txt-username']/following::input

Found 1 element using XPath Selector