Handles

Introduction

Playwright can create handles to the page DOM elements or any other objects inside the page. These handles live in the Playwright process, whereas the actual objects live in the browser. There are two types of handles:

- JSHandle to reference any JavaScript objects in the page
- ElementHandle to reference DOM elements in the page, it has extra methods that allow performing actions on the elements and asserting their properties.

Since any DOM element in the page is also a JavaScript object, any ElementHandle is a JSHandle as well.

Handles are used to perform operations on those actual objects in the page. You can evaluate on a handle, get handle properties, pass handle as an evaluation parameter, serialize page object into JSON etc. See the JSHandle class API for these and methods.

API reference

- JSHandle
- ElementHandle

Here is the easiest way to obtain a JSHandle.

Sync Async

```
js_handle = page.evaluate_handle('window')
#  Use jsHandle for evaluations.
```

Element Handles

DISCOURAGED

The use of ElementHandle is discouraged, use Locator objects and web-first assertions instead.

When ElementHandle is required, it is recommended to fetch it with the page.wait_for_selector() or frame.wait_for_selector() methods. These APIs wait for the element to be attached and visible.

Sync **Async**

```
# Get the element handle
element_handle = page.wait_for_selector('#box')

# Assert bounding box for the element
bounding_box = element_handle.bounding_box()
assert bounding_box.width == 100

# Assert attribute for the element
class_names = element_handle.get_attribute('class')
assert 'highlighted' in class_names
```

Handles as parameters

Handles can be passed into the page.evaluate() and similar methods. The following snippet creates a new array in the page, initializes it with data and returns a handle to this array into Playwright. It then uses the handle in subsequent evaluations:

Sync **Async**

```
# Create new array in page.
my_array_handle = page.evaluate_handle("""() => {
  window.myArray = [1];
  return myArray;
}""")

# Get current length of the array.
length = page.evaluate("a => a.length", my_array_handle)

# Add one more element to the array using the handle
page.evaluate("(arg) => arg.myArray.push(arg.newElement)", {
  'myArray': my_array_handle,
  'newElement': 2
})

# Release the object when it's no longer needed.
my_array_handle.dispose()
```

Handle Lifecycle

Handles can be acquired using the page methods such as page.evaluate_handle(), page.query_selector() or page.query_selector_all() or their frame counterparts frame.evaluate_handle(), frame.query_selector() or frame.query_selector_all(). Once created, handles will retain object from garbage collection unless page navigates or the handle is manually disposed via the js_handle.dispose() method.

API reference

- JSHandle
- ElementHandle
- element_handle.bounding_box()
- element_handle.get_attribute()
- element_handle.inner_text()
- element_handle.inner_html()
- element_handle.text_content()
- js_handle.evaluate()
- page.evaluate_handle()
- page.query_selector()
- page.query_selector_all()

Locator vs ElementHandle

CAUTION

We only recommend using ElementHandle in the rare cases when you need to perform extensive DOM traversal on a static page. For all user actions and assertions use locator instead.

The difference between the Locator and ElementHandle is that the latter points to a particular element, while Locator captures the logic of how to retrieve that element.

In the example below, handle points to a particular DOM element on page. If that element changes text or is used by React to render an entirely different component, handle is still pointing to that very stale DOM element. This can lead to unexpected behaviors.

Sync **Async**

```
handle = page.query_selector("text=Submit")
handle.hover()
handle.click()
```

With the locator, every time the locator is used, up-to-date DOM element is located in the page using the selector. So in the snippet below, underlying DOM element is going to be located twice.

Sync **Async**

```
locator = page.get_by_text("Submit")
locator.hover()
locator.click()
```