Evaluating JavaScript

Introduction

Playwright scripts run in your Playwright environment. Your page scripts run in the browser page environment. Those environments don't intersect, they are running in different virtual machines in different processes and even potentially on different computers.

The page.evaluate() API can run a JavaScript function in the context of the web page and bring results back to the Playwright environment. Browser globals like `window` and `document` can be used in `evaluate`.

Sync Async

```
href = page.evaluate('() => document.location.href')
```

If the result is a Promise or if the function is asynchronous evaluate will automatically wait until it's resolved:

Sync Async

```
status = page.evaluate("""async () => {
  response = await fetch(location.href)
  return response.status
}""")
```

Evaluation Argument

Playwright evaluation methods like page.evaluate() take a single optional argument. This argument can be a mix of Serializable values and JSHandle or ElementHandle instances. Handles are automatically converted to the value they represent.

Sync Async

```
# A primitive value.
page.evaluate('num => num', 42)

# An array.
page.evaluate('array => array.length', [1, 2, 3])

# An object.
page.evaluate('object => object.foo', { 'foo': 'bar' })

# A single handle.
button = page.evaluate_handle('window.button')
page.evaluate('button => button.textContent', button)

# Alternative notation using elementHandle.evaluate.
button.evaluate('(button, from) => button.textContent.substring(from)', 5)

# Object with multiple handles.
button1 = page.evaluate_handle('window.button1')
button2 = page.evaluate_handle('.button2')
page.evaluate("""o => o.button1.textContent + o.button2.textContent""",
    { 'button1': button1, 'button2': button2 })

# Object destructuring works. Note that property names must match
# between the destructured object and the argument.
# Also note the required parenthesis.
page.evaluate("""
    ({ button1, button2 }) => button1.textContent + button2.textContent""",
    { 'button1': button1, 'button2': button2 })

# Array works as well. Arbitrary names can be used for destructuring.
# Note the required parenthesis.
page.evaluate("""
    ([b1, b2]) => b1.textContent + b2.textContent""",
    [button1, button2])

# Any non-cyclic mix of serializables and handles works.
page.evaluate("""
    x => x.button1.textContent + x.list[0].textContent + String(x.foo)""",
    { 'button1': button1, 'list': [button2], 'foo': None })
```

Right:

Sync Async

```
data = { 'text': 'some data', 'value': 1 }
# Pass |data| as a parameter.
result = page.evaluate("""data => {
  window.myApp.use(data)
}""", data)
```

Wrong:

Sync Async

```
data = { 'text': 'some data', 'value': 1 }
result = page.evaluate("""() => {
  // There is no |data| in the web page.
  window.myApp.use(data)
}""")
```