Pages

Pages

Each BrowserContext can have multiple pages. A Page refers to a single tab or a popup window within a browser context. It should be used to navigate to URLs and interact with the page content.

Sync **Async**

```
page = context.new_page()

# Navigate explicitly, similar to entering a URL in the browser.
page.goto('http://example.com')
# Fill an input.
page.locator('#search').fill('query')

# Navigate implicitly by clicking a link.
page.locator('#submit').click()
# Expect a new url.
print(page.url)
```

Multiple pages

Each browser context can host multiple pages (tabs).

- Each page behaves like a focused, active page. Bringing the page to front is not required.
- Pages inside a context respect context-level emulation, like viewport sizes, custom network routes or browser locale.

Sync **Async**

```
# create two pages
page_one = context.new_page()
```

```
page_two = context.new_page()

# get pages of a browser context
all_pages = context.pages
```

Handling new pages

The `page` event on browser contexts can be used to get new pages that are created in the context. This can be used to handle new pages opened by `target="_blank"` links.

Sync **Async**

```
# Get page after a specific action (e.g. clicking a link)
with context.expect_page() as new_page_info:
    page.get_by_text("open new tab").click() # Opens a new tab
new_page = new_page_info.value

new_page.wait_for_load_state()
print(new_page.title())
```

If the action that triggers the new page is unknown, the following pattern can be used.

Sync **Async**

```
# Get all new pages (including popups) in the context
def handle_page(page):
    page.wait_for_load_state()
    print(page.title())

context.on("page", handle_page)
```

Handling popups

If the page opens a pop-up (e.g. pages opened by `target="_blank"` links), you can get a reference to it by listening to the `popup` event on the page.

This event is emitted in addition to the `browserContext.on('page')` event, but only for popups relevant to this page.

Sync **Async**

```python
# Get popup after a specific action (e.g., click)
with page.expect_popup() as popup_info:
    page.get_by_text("open the popup").click()
popup = popup_info.value

popup.wait_for_load_state()
print(popup.title())
```

If the action that triggers the popup is unknown, the following pattern can be used.

Sync **Async**

```python
# Get all popups when they open
def handle_popup(popup):
    popup.wait_for_load_state()
    print(popup.title())

page.on("popup", handle_popup)
```