Chrome extensions

Introduction

> **NOTE**
>
> Extensions only work in Chrome / Chromium launched with a persistent context. Use custom browser args at your own risk, as some of them may break Playwright functionality.

The following is code for getting a handle to the background page of a Manifest v2 extension whose source is located in `./my-extension`:

Sync Async

```python
from playwright.sync_api import sync_playwright, Playwright

path_to_extension = "./my-extension"
user_data_dir = "/tmp/test-user-data-dir"


def run(playwright: Playwright):
    context = playwright.chromium.launch_persistent_context(
        user_data_dir,
        headless=False,
        args=[
            f"--disable-extensions-except={path_to_extension}",
            f"--load-extension={path_to_extension}",
        ],
    )
    if len(context.background_pages) == 0:
        background_page = context.wait_for_event('backgroundpage')
    else:
        background_page = context.background_pages[0]

    # Test the background page as you would any other page.
```

```
    context.close()


with sync_playwright() as playwright:
    run(playwright)
```

Testing

To have the extension loaded when running tests you can use a test fixture to set the context. You can also dynamically retrieve the extension id and use it to load and test the popup page for example.

First, add fixtures that will load the extension:

conftest.py

```
from typing import Generator
from pathlib import Path
from playwright.sync_api import Playwright, BrowserContext
import pytest


@pytest.fixture()
def context(playwright: Playwright) -> Generator[BrowserContext, None, None]:
    path_to_extension = Path(__file__).parent.joinpath("my-extension")
    context = playwright.chromium.launch_persistent_context(
        "",
        headless=False,
        args=[
            f"--disable-extensions-except={path_to_extension}",
            f"--load-extension={path_to_extension}",
        ],
    )
    yield context
    context.close()


@pytest.fixture()
def extension_id(context) -> Generator[str, None, None]:
    # for manifest v2:
```

```
    # background = context.background_pages[0]
    # if not background:
    #     background = context.wait_for_event("backgroundpage")

    # for manifest v3:
    background = context.service_workers[0]
    if not background:
        background = context.wait_for_event("serviceworker")

    extension_id = background.url.split("/")[2]
    yield extension_id
```

Then use these fixtures in a test:

test_foo.py

```
from playwright.sync_api import expect, Page


def test_example_test(page: Page) -> None:
    page.goto("https://example.com")
    expect(page.locator("body")).to_contain_text("Changed by my-extension")


def test_popup_page(page: Page, extension_id: str) -> None:
    page.goto(f"chrome-extension://{extension_id}/popup.html")
    expect(page.locator("body")).to_have_text("my-extension popup")
```

Headless mode

> **DANGER**
>
> `headless=new` mode is not officially supported by Playwright and might result in unexpected behavior.

By default, Chrome's headless mode in Playwright does not support Chrome extensions. To overcome this limitation, you can run Chrome's persistent context with a new headless mode by

using the following code:

conftest.py

```
path_to_extension = Path(__file__).parent.joinpath("my-extension")
context = playwright.chromium.launch_persistent_context(
    "",
    headless=False,
    args=[
        "--headless=new",
        f"--disable-extensions-except={path_to_extension}",
        f"--load-extension={path_to_extension}",
    ],
)
```