Skip to main content

PlaywrightDocsAPI
Node.js

- Node.js
- Python
- Java
- .NET

Community
Search⌘K

- Getting Started
  - Installation
  - Writing tests
  - Generating tests
  - Running and debugging tests
  - Trace viewer
  - Setting up CI
- Getting started - VS Code
- Release notes
- Canary releases
- Playwright Test
  - Test configuration
  - Test use options
  - Annotations
  - Command line

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`:

```
const { chromium } = require('playwright');

(async () => {
  const pathToExtension = require('path').join(__dirname, 'my-extension');
  const userDataDir = '/tmp/test-user-data-dir';
  const browserContext = await
chromium.launchPersistentContext(userDataDir, {
    headless: false,
    args: [
      `--disable-extensions-except=${pathToExtension}`,
      `--load-extension=${pathToExtension}`
    ]
  });
  let [backgroundPage] = browserContext.backgroundPages();
  if (!backgroundPage)
    backgroundPage = await browserContext.waitForEvent('backgroundpage');

  // Test the background page as you would any other page.
  await browserContext.close();
})();
```

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:

fixtures.ts

```
import { test as base, chromium, type BrowserContext } from
'@playwright/test';
import path from 'path';

export const test = base.extend<{
  context: BrowserContext;
  extensionId: string;
}>({
  context: async ({ }, use) => {
    const pathToExtension = path.join(__dirname, 'my-extension');
    const context = await chromium.launchPersistentContext('', {
      headless: false,
      args: [
        `--disable-extensions-except=${pathToExtension}`,
        `--load-extension=${pathToExtension}`,
      ],
    });
    await use(context);
    await context.close();
  },
  extensionId: async ({ context }, use) => {
    /*
    // for manifest v2:
    let [background] = context.backgroundPages()
    if (!background)
      background = await context.waitForEvent('backgroundpage')
    */

    // for manifest v3:
    let [background] = context.serviceWorkers();
    if (!background)
      background = await context.waitForEvent('serviceworker');

    const extensionId = background.url().split('/')[2];
    await use(extensionId);
  },
});
export const expect = test.expect;
```

Then use these fixtures in a test:

```
import { test, expect } from './fixtures';

test('example test', async ({ page }) => {
  await page.goto('https://example.com');
  await expect(page.locator('body')).toHaveText('Changed by my-extension');
});

test('popup page', async ({ page, extensionId }) => {
  await page.goto(`chrome-extension://${extensionId}/popup.html`);
  await expect(page.locator('body')).toHaveText('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:

fixtures.ts

```
// ...

const pathToExtension = path.join(__dirname, 'my-extension');
const context = await chromium.launchPersistentContext('', {
  headless: false,
  args: [
    `--headless=new`,
    `--disable-extensions-except=${pathToExtension}`,
    `--load-extension=${pathToExtension}`,
  ],
});
// ...
```

[Previous
Browsers]

[Next
Clock]