

[Skip to main content](#)



[PlaywrightDocsAPI](#)

[Node.js](#)

- [Node.js](#)
- [Python](#)
- [Java](#)
- [.NET](#)

[Community](#)

Search⌕

- [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](#)

- [Emulation](#)
  - [Fixtures](#)
  - [Global setup and teardown](#)
  - [Parallelism](#)
  - [Parameterize tests](#)
  - [Projects](#)
  - [Reporters](#)
  - [Retries](#)
  - [Sharding](#)
  - [Timeouts](#)
  - [TypeScript](#)
  - [UI Mode](#)
  - [Web server](#)
- [Guides](#)
  - [Library](#)
  - [Accessibility testing](#)
  - [Actions](#)
  - [Assertions](#)
  - [API testing](#)
  - [Authentication](#)
  - [Auto-waiting](#)
  - [Best Practices](#)
  - [Browsers](#)
  - [Chrome extensions](#)
  - [Clock](#)
  - [Components \(experimental\)](#)
  - [Debugging Tests](#)
  - [Dialogs](#)
  - [Downloads](#)
  - [Evaluating JavaScript](#)
  - [Events](#)
  - [Extensibility](#)
  - [Frames](#)
  - [Handles](#)
  - [Isolation](#)
  - [Locators](#)
  - [Mock APIs](#)
  - [Mock browser APIs](#)
  - [Navigations](#)
  - [Network](#)
  - [Other locators](#)
  - [Page object models](#)
  - [Pages](#)
  - [Screenshots](#)
  - [Visual comparisons](#)
  - [Test generator](#)
  - [Trace viewer](#)
  - [Videos](#)
  - [WebView2](#)
- [Migration](#)

- [Integrations](#)
- [Supported languages](#)
- 
- Guides
- Downloads

On this page

# Downloads

## Introduction

For every attachment downloaded by the page, [page.on\('download'\)](#) event is emitted. All these attachments are downloaded into a temporary folder. You can obtain the download url, file name and payload stream using the [Download](#) object from the event.

You can specify where to persist downloaded files using the `downloadsPath` option in [browserType.launch\(\)](#).

NOTE

Downloaded files are deleted when the browser context that produced them is closed.

Here is the simplest way to handle the file download:

```
// Start waiting for download before clicking. Note no await.
const downloadPromise = page.waitForEvent('download');
await page.getByText('Download file').click();
const download = await downloadPromise;

// Wait for the download process to complete and save the downloaded file
somewhere.
await download.saveAs('/path/to/save/at/' + download.suggestedFilename());
```

## Variations

If you have no idea what initiates the download, you can still handle the event:

```
page.on('download', download => download.path().then(console.log));
```

Note that handling the event forks the control flow and makes the script harder to follow. Your scenario might end while you are downloading a file since your main control flow is not awaiting for this operation to resolve.

NOTE

For uploading files, see the [uploading files](#) section.

[Previous](#)  
[Dialogs](#)

[Next](#)  
[Evaluating JavaScript](#)

- [Introduction](#)

## Learn

- [Getting started](#)
- [Playwright Training](#)
- [Learn Videos](#)
- [Feature Videos](#)

## Community

- [Stack Overflow](#)
- [Discord](#)
- [Twitter](#)
- [LinkedIn](#)

## More

- [GitHub](#)
- [YouTube](#)
- [Blog](#)
- [Ambassadors](#)

Copyright © 2024 Microsoft