

[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
- Page object models

On this page

Page object models

Introduction

Large test suites can be structured to optimize ease of authoring and maintenance. Page object models are one such approach to structure your test suite.

A page object represents a part of your web application. An e-commerce web application might have a home page, a listings page and a checkout page. Each of them can be represented by page object models.

Page objects **simplify authoring** by creating a higher-level API which suits your application and **simplify maintenance** by capturing element selectors in one place and create reusable code to avoid repetition.

Implementation

We will create a `PlaywrightDevPage` helper class to encapsulate common operations on the `playwright.dev` page. Internally, it will use the `page` object.

- TypeScript
- JavaScript
- Library

playwright-dev-page.ts

```
import { expect, type Locator, type Page } from '@playwright/test';

export class PlaywrightDevPage {
  readonly page: Page;
  readonly getStartedLink: Locator;
  readonly gettingStartedHeader: Locator;
  readonly pomLink: Locator;
  readonly tocList: Locator;

  constructor(page: Page) {
    this.page = page;
    this.getStartedLink = page.locator('a', { hasText: 'Get started' });
    this.gettingStartedHeader = page.locator('h1', { hasText:
'Installation' });
    this.pomLink = page.locator('li', {
      hasText: 'Guides',
    }).locator('a', {
```

```

        hasText: 'Page Object Model',
    });
    this.tocList = page.locator('article div.markdown ul > li > a');
}

async goto() {
    await this.page.goto('https://playwright.dev');
}

async getStarted() {
    await this.getStartedLink.first().click();
    await expect(this.gettingStartedHeader).toBeVisible();
}

async pageObjectModel() {
    await this.getStarted();
    await this.pomLink.click();
}
}

```

playwright-dev-page.js

```

const { expect } = require('@playwright/test');

exports.PlaywrightDevPage = class PlaywrightDevPage {
    /**
     * @param {import('@playwright/test').Page} page
     */
    constructor(page) {
        this.page = page;
        this.getStartedLink = page.locator('a', { hasText: 'Get started' });
        this.gettingStartedHeader = page.locator('h1', { hasText:
'Installation' });
        this.pomLink = page.locator('li', {
            hasText: 'Guides',
        }).locator('a', {
            hasText: 'Page Object Model',
        });
        this.tocList = page.locator('article div.markdown ul > li > a');
    }

    async goto() {
        await this.page.goto('https://playwright.dev');
    }

    async getStarted() {
        await this.getStartedLink.first().click();
        await expect(this.gettingStartedHeader).toBeVisible();
    }

    async pageObjectModel() {
        await this.getStarted();
        await this.pomLink.click();
    }
};

```

models/PlaywrightDevPage.js

```

class PlaywrightDevPage {
    /**
     * @param {import('playwright').Page} page
     */
    constructor(page) {
        this.page = page;
    }
}

```

```

    this.getStartedLink = page.locator('a', { hasText: 'Get started' });
    this.gettingStartedHeader = page.locator('h1', { hasText:
'Installation' });
    this.pomLink = page.locator('li', {
      hasText: 'Playwright Test',
    }).locator('a', {
      hasText: 'Page Object Model',
    });
    this.tocList = page.locator('article div.markdown ul > li > a');
  }
  async getStarted() {
    await this.getStartedLink.first().click();
    await expect(this.gettingStartedHeader).toBeVisible();
  }

  async pageObjectModel() {
    await this.getStarted();
    await this.pomLink.click();
  }
}
module.exports = { PlaywrightDevPage };

```

Now we can use the PlaywrightDevPage class in our tests.

- TypeScript
- JavaScript
- Library

example.spec.ts

```

import { test, expect } from '@playwright/test';
import { PlaywrightDevPage } from './playwright-dev-page';

test('getting started should contain table of contents', async ({ page })
=> {
  const playwrightDev = new PlaywrightDevPage(page);
  await playwrightDev.goto();
  await playwrightDev.getStarted();
  await expect(playwrightDev.tocList).toHaveText([
    `How to install Playwright`,
    `What's Installed`,
    `How to run the example test`,
    `How to open the HTML test report`,
    `Write tests using web first assertions, page fixtures and locators`,
    `Run single test, multiple tests, headed mode`,
    `Generate tests with Codegen`,
    `See a trace of your tests`
  ]);
});

test('should show Page Object Model article', async ({ page }) => {
  const playwrightDev = new PlaywrightDevPage(page);
  await playwrightDev.goto();
  await playwrightDev.pageObjectModel();
  await expect(page.locator('article')).toContainText('Page Object Model is
a common pattern');
});

```

example.spec.js

```

const { test, expect } = require('@playwright/test');
const { PlaywrightDevPage } = require('./playwright-dev-page');

```

```

test('getting started should contain table of contents', async ({ page })
=> {
  const playwrightDev = new PlaywrightDevPage(page);
  await playwrightDev.goto();
  await playwrightDev.getStarted();
  await expect(playwrightDev.tocList).toHaveText([
    `How to install Playwright`,
    `What's Installed`,
    `How to run the example test`,
    `How to open the HTML test report`,
    `Write tests using web first assertions, page fixtures and locators`,
    `Run single test, multiple tests, headed mode`,
    `Generate tests with Codegen`,
    `See a trace of your tests`
  ]);
});

test('should show Page Object Model article', async ({ page }) => {
  const playwrightDev = new PlaywrightDevPage(page);
  await playwrightDev.goto();
  await playwrightDev.pageObjectModel();
  await expect(page.locator('article')).toContainText('Page Object Model is
a common pattern');
});

```

example.spec.js

```

const { PlaywrightDevPage } = require('./playwright-dev-page');

// In the test
const page = await browser.newPage();
await playwrightDev.goto();
await playwrightDev.getStarted();
await expect(playwrightDev.tocList).toHaveText([
  `How to install Playwright`,
  `What's Installed`,
  `How to run the example test`,
  `How to open the HTML test report`,
  `Write tests using web first assertions, page fixtures and locators`,
  `Run single test, multiple tests, headed mode`,
  `Generate tests with Codegen`,
  `See a trace of your tests`
]);

```

[Previous](#)

[Other locators](#)

[Next](#)

[Pages](#)

- [Introduction](#)
- [Implementation](#)

Learn

- [Getting started](#)
- [Playwright Training](#)

- [Learn Videos](#)
- [Feature Videos](#)

Community

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

More

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

Copyright © 2024 Microsoft