# Test use options

## Introduction

In addition to configuring the test runner you can also configure Emulation, Network and Recording for the Browser or BrowserContext. These options are passed to the `use: {}` object in the Playwright config.

### Basic Options

Set the base URL and storage state for all tests:

playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  use: {
    // Base URL to use in actions like `await page.goto('/')`.
    baseURL: 'http://127.0.0.1:3000',

    // Populates context with given storage state.
    storageState: 'state.json',
  },
});
```

| Option | Description |
| --- | --- |
| testOptions.baseURL | Base URL used for all pages in the context. Allows navigating by using just the path, for example `page.goto('/settings')`. |
| testOptions.storageState | Populates context with given storage state. Useful for easy authentication, learn more. |

### Emulation Options

With Playwright you can emulate a real device such as a mobile phone or tablet. See our guide on projects for more info on emulating devices. You can also emulate the `"geolocation"`, `"locale"` and `"timezone"` for all tests or for a specific test as well as set the `"permissions"` to show notifications or change the `"colorScheme"`. See our Emulation guide to learn more.

playwright.config.ts

```
import { defineConfig } from '@playwright/test';
```

```
export default defineConfig({
  use: {
    // Emulates `'prefers-colors-scheme'` media feature.
    colorScheme: 'dark',

    // Context geolocation.
    geolocation: { longitude: 12.492507, latitude: 41.889938 },

    // Emulates the user locale.
    locale: 'en-GB',

    // Grants specified permissions to the browser context.
    permissions: ['geolocation'],

    // Emulates the user timezone.
    timezoneId: 'Europe/Paris',

    // Viewport used for all pages in the context.
    viewport: { width: 1280, height: 720 },
  },
});
```

| Option | Description |
| --- | --- |
| testOptions.colorScheme | Emulates `'prefers-colors-scheme'` media feature, supported values are `'light'`, `'dark'`, `'no-preference'` |
| testOptions.geolocation | Context geolocation. |
| testOptions.locale | Emulates the user locale, for example `en-GB`, `de-DE`, etc. |
| testOptions.permissions | A list of permissions to grant to all pages in the context. |
| testOptions.timezoneId | Changes the timezone of the context. |
| testOptions.viewport | Viewport used for all pages in the context. |

## Network Options

Available options to configure networking:

playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  use: {
    // Whether to automatically download all the attachments.
    acceptDownloads: false,

    // An object containing additional HTTP headers to be sent with every
request.
    extraHTTPHeaders: {
      'X-My-Header': 'value',
    },

    // Credentials for HTTP authentication.
    httpCredentials: {
      username: 'user',
      password: 'pass',
    },

    // Whether to ignore HTTPS errors during navigation.
    ignoreHTTPSErrors: true,
```

```
    // Whether to emulate network being offline.
    offline: true,

    // Proxy settings used for all pages in the test.
    proxy: {
      server: 'http://myproxy.com:3128',
      bypass: 'localhost',
    },
  },
});
```

| Option | Description |
| --- | --- |
| testOptions.acceptDownloads | Whether to automatically download all the attachments, defaults to `true`. Learn more about working with downloads. |
| testOptions.extraHTTPHeaders | An object containing additional HTTP headers to be sent with every request. All header values must be strings. |
| testOptions.httpCredentials | Credentials for HTTP authentication. |
| testOptions.ignoreHTTPSErrors | Whether to ignore HTTPS errors during navigation. |
| testOptions.offline | Whether to emulate network being offline. |
| testOptions.proxy | Proxy settings used for all pages in the test. |

NOTE

You don't have to configure anything to mock network requests. Just define a custom Route that mocks the network for a browser context. See our network mocking guide to learn more.

## Recording Options

With Playwright you can capture screenshots, record videos as well as traces of your test. By default these are turned off but you can enable them by setting the `screenshot`, `video` and `trace` options in your `playwright.config.js` file.

Trace files, screenshots and videos will appear in the test output directory, typically `test-results`.

playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  use: {
    // Capture screenshot after each test failure.
    screenshot: 'only-on-failure',

    // Record trace only when retrying a test for the first time.
    trace: 'on-first-retry',

    // Record video only when retrying a test for the first time.
    video: 'on-first-retry'
  },
});
```

| Option | Description |
| --- | --- |
| testOptions.screenshot | Capture screenshots of your test. Options include `'off'`, `'on'` and `'only-on-failure'` |

| Option | Description |
| --- | --- |
| testOptions.trace | Playwright can produce test traces while running the tests. Later on, you can view the trace and get detailed information about Playwright execution by opening Trace Viewer. Options include: `'off'`, `'on'`, `'retain-on-failure'` and `'on-first-retry'` |
| testOptions.video | Playwright can record videos for your tests. Options include: `'off'`, `'on'`, `'retain-on-failure'` and `'on-first-retry'` |

## Other Options

playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  use: {
    // Maximum time each action such as `click()` can take. Defaults to 0 (no limit).
    actionTimeout: 0,

    // Name of the browser that runs tests. For example `chromium`, `firefox`, `webkit`.
    browserName: 'chromium',

    // Toggles bypassing Content-Security-Policy.
    bypassCSP: true,

    // Channel to use, for example "chrome", "chrome-beta", "msedge", "msedge-beta".
    channel: 'chrome',

    // Run browser in headless mode.
    headless: false,

    // Change the default data-testid attribute.
    testIdAttribute: 'pw-test-id',
  },
});
```

| Option | Description |
| --- | --- |
| testOptions.actionTimeout | Timeout for each Playwright action in milliseconds. Defaults to `0` (no timeout). Learn more about timeouts and how to set them for a single test. |
| testOptions.browserName | Name of the browser that runs tests. Defaults to 'chromium'. Options include `chromium`, `firefox`, or `webkit`. |
| testOptions.bypassCSP | Toggles bypassing Content-Security-Policy. Useful when CSP includes the production origin. Defaults to `false`. |
| testOptions.channel | Browser channel to use. Learn more about different browsers and channels. |
| testOptions.headless | Whether to run the browser in headless mode meaning no browser is shown when running tests. Defaults to `true`. |
| testOptions.testIdAttribute | Changes the default `data-testid` attribute used by Playwright locators. |

## More browser and context options

Any options accepted by browserType.launch() or browser.newContext() can be put into `launchOptions` or `contextOptions` respectively in the `use` section.

playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  use: {
    launchOptions: {
      slowMo: 50,
    },
  },
});
```

However, most common ones like `headless` or `viewport` are available directly in the `use` section - see basic options, emulation or network.

## Explicit Context Creation and Option Inheritance

If using the built-in `browser` fixture, calling browser.newContext() will create a context with options inherited from the config:

playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  use: {
    userAgent: 'some custom ua',
    viewport: { width: 100, height: 100 },
  },
});
```

An example test illustrating the initial context options are set:

```
test('should inherit use options on context when using built-in browser
fixture', async ({
  browser,
}) => {
  const context = await browser.newContext();
  const page = await context.newPage();
  expect(await page.evaluate(() => navigator.userAgent)).toBe('some custom
ua');
  expect(await page.evaluate(() => window.innerWidth)).toBe(100);
  await context.close();
});
```

## Configuration Scopes

You can configure Playwright globally, per project, or per test. For example, you can set the locale to be used globally by adding `locale` to the `use` option of the Playwright config, and then override it for a specific project using the `project` option in the config. You can also override it for a specific test by adding `test.use({})` in the test file and passing in the options.

playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  use: {
    locale: 'en-GB'
  },
});
```

You can override options for a specific project using the `project` option in the Playwright config.

```
import { defineConfig, devices } from '@playwright/test';

export default defineConfig({
  projects: [
    {
      name: 'chromium',
      use: {
        ...devices['Desktop Chrome'],
        locale: 'de-DE',
      },
    },
  ],
});
```

You can override options for a specific test file by using the `test.use()` method and passing in the options. For example to run tests with the French locale for a specific test:

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

test.use({ locale: 'fr-FR' });

test('example', async ({ page }) => {
  // ...
});
```

The same works inside a describe block. For example to run tests in a describe block with the French locale:

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

test.describe('french language block', () => {

  test.use({ locale: 'fr-FR' });

  test('example', async ({ page }) => {
    // ...
  });
});
```

Previous
Test configuration

Next
Annotations

Learn

- Getting started
- Playwright Training
- Learn Videos
- Feature Videos

Community

- Stack Overflow
- Discord
- Twitter
- LinkedIn

More

- GitHub
- YouTube
- Blog
- Ambassadors