    - 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

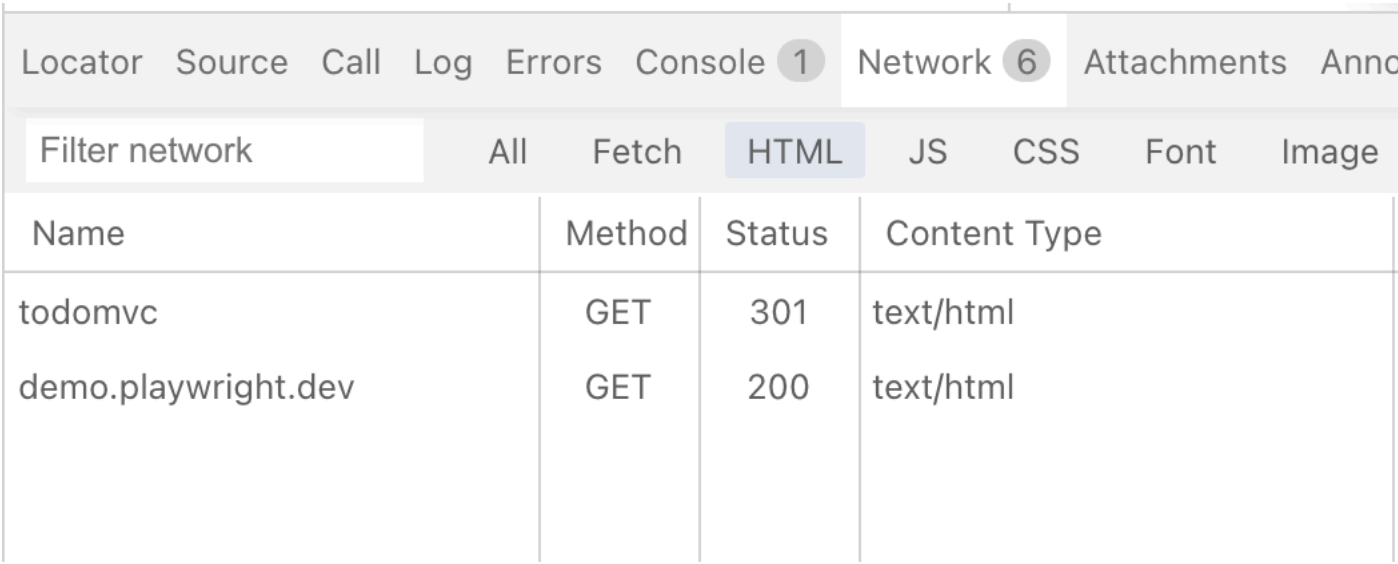
Release notes

Version 1.47

Network Tab improvements

The Network tab in the UI mode and trace viewer has several nice improvements:

- filtering by asset type and URL
- better display of query string parameters
- preview of font assets



`--tsconfig` CLI option

By default, Playwright will look up the closest tsconfig for each imported file using a heuristic. You can now specify a single tsconfig file in the command line, and Playwright will use it for all imported files, not only test files:

```
# Pass a specific tsconfig
npx playwright test --tsconfig tsconfig.test.json
```

APIRequestContext now accepts URLSearchParams and string as query parameters

You can now pass `URLSearchParams` and `string` as query parameters to APIRequestContext:

```
test('query params', async ({ request }) => {
  const searchParams = new URLSearchParams();
  searchParams.set('userId', 1);
  const response = await request.get(
      'https://jsonplaceholder.typicode.com/posts',
      {
        params: searchParams // or as a string: 'userId=1'
      }
  );
  // ...
});
```

Miscellaneous

- The `mcr.microsoft.com/playwright:v1.47.0` now serves a Playwright image based on Ubuntu 24.04 Noble. To use the 22.04 jammy-based image, please use `mcr.microsoft.com/playwright:v1.47.0-jammy` instead.
- New option `behavior` in page.removeAllListeners(), browser.removeAllListeners() and browserContext.removeAllListeners() to wait for ongoing listeners to complete.
- TLS client certificates can now be passed from memory by passing `cert` and `key` as buffers instead of file paths.
- Attachments with a `text/html` content type can now be opened in a new tab in the HTML report. This is useful for including third-party reports or other HTML content in the Playwright test report and distributing it to your team.
- `noWaitAfter` in locator.selectOption() was deprecated.
- We've seen reports of WebGL in Webkit misbehaving on GitHub Actions `macos-13`. We recommend upgrading GitHub Actions to `macos-14`.

Browser Versions

- Chromium 129.0.6668.29
- Mozilla Firefox 130.0
- WebKit 18.0

This version was also tested against the following stable channels:

- Google Chrome 128
- Microsoft Edge 128

Version 1.46

TLS Client Certificates

Playwright now allows you to supply client-side certificates, so that server can verify them, as specified by TLS Client Authentication.

The following snippet sets up a client certificate for `https://example.com`:

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

export default defineConfig({
  // ...
  use: {
    clientCertificates: [{
      origin: 'https://example.com',
      certPath: './cert.pem',
      keyPath: './key.pem',
      passphrase: 'mysecretpassword',
    }],
  },
  // ...
});
```

You can also provide client certificates to a particular test project or as a parameter of browser.newContext() and apiRequest.newContext().

`--only-changed` cli option

New CLI option `--only-changed` will only run test files that have been changed since the last git commit or from a specific git "ref". This will also run all test files that import any changed files.

```sh
# Only run test files with uncommitted changes
npx playwright test --only-changed

# Only run test files changed relative to the "main" branch
npx playwright test --only-changed=main
```

Component Testing: New `router` fixture

This release introduces an experimental `router` fixture to intercept and handle network requests in component testing. There are two ways to use the router fixture:

- Call `router.route(url, handler)` that behaves similarly to page.route().
- Call `router.use(handlers)` and pass MSW library request handlers to it.

Here is an example of reusing your existing MSW handlers in the test.

```ts
import { handlers } from '@src/mocks/handlers';

test.beforeEach(async ({ router }) => {
  // install common handlers before each test
  await router.use(...handlers);
});

test('example test', async ({ mount }) => {
  // test as usual, your handlers are active
  // ...
});
```

This fixture is only available in component tests.

UI Mode / Trace Viewer Updates

- Test annotations are now shown in UI mode.
- Content of text attachments is now rendered inline in the attachments pane.
- New setting to show/hide routing actions like route.continue().
- Request method and status are shown in the network details tab.
- New button to copy source file location to clipboard.
- Metadata pane now displays the `baseURL`.

Miscellaneous

- New `maxRetries` option in apiRequestContext.fetch() which retries on the `ECONNRESET` network error.
- New option to box a fixture to minimize the fixture exposure in test reports and error messages.
- New option to provide a custom fixture title to be used in test reports and error messages.

Browser Versions

- Chromium 128.0.6613.18
- Mozilla Firefox 128.0
- WebKit 18.0

This version was also tested against the following stable channels:

- Google Chrome 127
- Microsoft Edge 127

Version 1.45

Clock

Utilizing the new Clock API allows to manipulate and control time within tests to verify time-related behavior. This API covers many common scenarios, including:

- testing with predefined time;
- keeping consistent time and timers;
- monitoring inactivity;
- ticking through time manually.

```
// Initialize clock and let the page load naturally.
await page.clock.install({ time: new Date('2024-02-02T08:00:00') });
await page.goto('http://localhost:3333');

// Pretend that the user closed the laptop lid and opened it again at 10am,
// Pause the time once reached that point.
await page.clock.pauseAt(new Date('2024-02-02T10:00:00'));

// Assert the page state.
await expect(page.getByTestId('current-time')).toHaveText('2/2/2024,
10:00:00 AM');
```

```
// Close the laptop lid again and open it at 10:30am.
await page.clock.fastForward('30:00');
await expect(page.getByTestId('current-time')).toHaveText('2/2/2024,
10:30:00 AM');
```

See the clock guide for more details.

Test runner

- New CLI option `--fail-on-flaky-tests` that sets exit code to `1` upon any flaky tests. Note that by default, the test runner exits with code `0` when all failed tests recovered upon a retry. With this option, the test run will fail in such case.
- New environment variable `PLAYWRIGHT_FORCE_TTY` controls whether built-in `list`, `line` and `dot` reporters assume a live terminal. For example, this could be useful to disable tty behavior when your CI environment does not handle ANSI control sequences well. Alternatively, you can enable tty behavior even when to live terminal is present, if you plan to post-process the output and handle control sequences.

```
# Avoid TTY features that output ANSI control sequences
PLAYWRIGHT_FORCE_TTY=0 npx playwright test

# Enable TTY features, assuming a terminal width 80
PLAYWRIGHT_FORCE_TTY=80 npx playwright test
```

- New options testConfig.respectGitIgnore and testProject.respectGitIgnore control whether files matching `.gitignore` patterns are excluded when searching for tests.
- New property `timeout` is now available for custom expect matchers. This property takes into account `playwright.config.ts` and `expect.configure()`.

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

export const expect = baseExpect.extend({
  async toHaveAmount(locator: Locator, expected: number, options?: {
timeout?: number }) {
    // When no timeout option is specified, use the config timeout.
    const timeout = options?.timeout ?? this.timeout;
    // ... implement the assertion ...
  },
});
```

Miscellaneous

- Method locator.setInputFiles() now supports uploading a directory for `<input type=file webkitdirectory>` elements.

```
await page.getByLabel('Upload
directory').setInputFiles(path.join(__dirname, 'mydir'));
```

- Multiple methods like locator.click() or locator.press() now support a `ControlOrMeta` modifier key. This key maps to `Meta` on macOS and maps to `Control` on Windows and Linux.

```
// Press the common keyboard shortcut Control+S or Meta+S to trigger
a "Save" operation.
await page.keyboard.press('ControlOrMeta+S');
```

- New property `httpCredentials.send` in apiRequest.newContext() that allows to either always send the `Authorization` header or only send it in response to `401 Unauthorized`.
- New option `reason` in apiRequestContext.dispose() that will be included in the error message of ongoing operations interrupted by the context disposal.
- New option `host` in browserType.launchServer() allows to accept websocket connections on a specific address instead of unspecified `0.0.0.0`.
- Playwright now supports Chromium, Firefox and WebKit on Ubuntu 24.04.
- v1.45 is the last release to receive WebKit update for macOS 12 Monterey. Please update macOS to keep using the latest WebKit.

Browser Versions

- Chromium 127.0.6533.5
- Mozilla Firefox 127.0
- WebKit 17.4

This version was also tested against the following stable channels:

- Google Chrome 126
- Microsoft Edge 126

Version 1.44

New APIs

Accessibility assertions

- expect(locator).toHaveAccessibleName() checks if the element has the specified accessible name:

```
const locator = page.getByRole('button');
await expect(locator).toHaveAccessibleName('Submit');
```

- expect(locator).toHaveAccessibleDescription() checks if the element has the specified accessible description:

```
const locator = page.getByRole('button');
await expect(locator).toHaveAccessibleDescription('Upload a photo');
```

- expect(locator).toHaveRole() checks if the element has the specified ARIA role:

```
const locator = page.getByTestId('save-button');
await expect(locator).toHaveRole('button');
```

Locator handler

- After executing the handler added with page.addLocatorHandler(), Playwright will now wait until the overlay that triggered the handler is not visible anymore. You can opt-out of this behavior with the new `noWaitAfter` option.
- You can use new `times` option in page.addLocatorHandler() to specify maximum number of times the handler should be run.
- The handler in page.addLocatorHandler() now accepts the locator as argument.
- New page.removeLocatorHandler() method for removing previously added locator handlers.

```
const locator = page.getByText('This interstitial covers the button');
await page.addLocatorHandler(locator, async overlay => {
  await overlay.locator('#close').click();
}, { times: 3, noWaitAfter: true });
// Run your tests that can be interrupted by the overlay.
// ...
await page.removeLocatorHandler(locator);
```

Miscellaneous options

- `multipart` option in `apiRequestContext.fetch()` now accepts `FormData` and supports repeating fields with the same name.

  ```
  const formData = new FormData();
  formData.append('file', new File(['let x = 2024;'], 'f1.js', { type:
  'text/javascript' }));
  formData.append('file', new File(['hello'], 'f2.txt', { type:
  'text/plain' }));
  context.request.post('https://example.com/uploadFiles', {
    multipart: formData
  });
  ```

- `expect(callback).toPass({ intervals })` can now be configured by `expect.toPass.intervals` option globally in testConfig.expect or per project in testProject.expect.
- `expect(page).toHaveURL(url)` now supports `ignoreCase` option.
- testProject.ignoreSnapshots allows to configure per project whether to skip screenshot expectations.

Reporter API

- New method suite.entries() returns child test suites and test cases in their declaration order. suite.type and testCase.type can be used to tell apart test cases and suites in the list.
- Blob reporter now allows overriding report file path with a single option `outputFile`. The same option can also be specified as `PLAYWRIGHT_BLOB_OUTPUT_FILE` environment variable that might be more convenient on CI/CD.
- JUnit reporter now supports `includeProjectInTestName` option.

Command line

- `--last-failed` CLI option to for running only tests that failed in the previous run.

First run all tests:

```
$ npx playwright test

Running 103 tests using 5 workers
...
2 failed
  [chromium] › my-test.spec.ts:8:5 › two
─────────────────────────────────────────────────────────────────
  [chromium] › my-test.spec.ts:13:5 › three
─────────────────────────────────────────────────────────────────
101 passed (30.0s)
```

Now fix the failing tests and run Playwright again with `--last-failed` option:

```
$ npx playwright test --last-failed

Running 2 tests using 2 workers
  2 passed (1.2s)
```

Browser Versions

- Chromium 125.0.6422.14
- Mozilla Firefox 125.0.1
- WebKit 17.4

This version was also tested against the following stable channels:

- Google Chrome 124
- Microsoft Edge 124

Version 1.43

New APIs

- Method browserContext.clearCookies() now supports filters to remove only some cookies.

```
// Clear all cookies.
await context.clearCookies();
// New: clear cookies with a particular name.
await context.clearCookies({ name: 'session-id' });
// New: clear cookies for a particular domain.
await context.clearCookies({ domain: 'my-origin.com' });
```

- New mode `retain-on-first-failure` for testOptions.trace. In this mode, trace is recorded for the first run of each test, but not for retires. When test run fails, the trace file is retained, otherwise it is removed.

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

export default defineConfig({
  use: {
```

```
    trace: 'retain-on-first-failure',
  },
});
```

- New property testInfo.tags exposes test tags during test execution.

```
test('example', async ({ page }) => {
  console.log(test.info().tags);
});
```

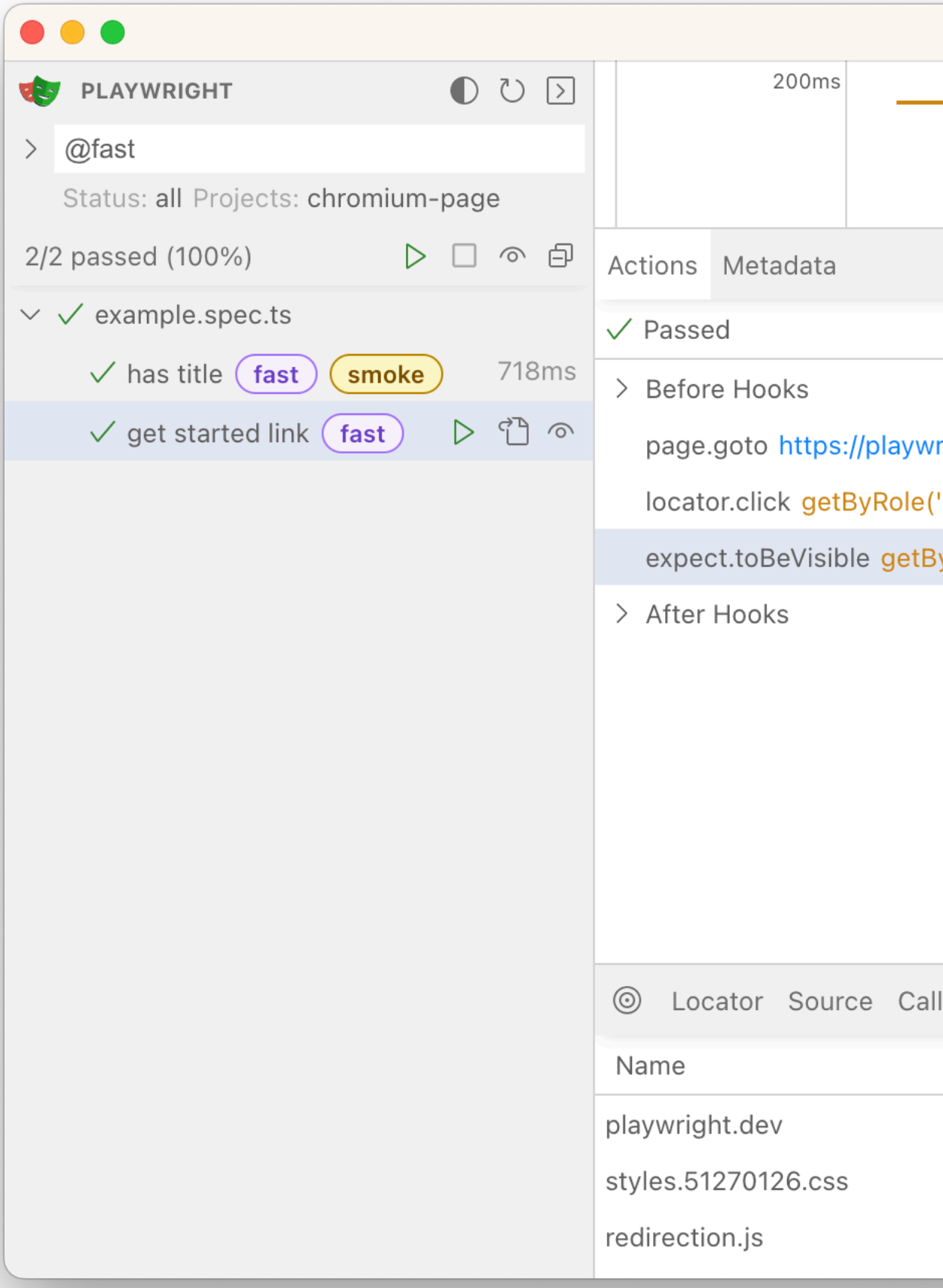
- New method locator.contentFrame() converts a Locator object to a FrameLocator. This can be useful when you have a Locator object obtained somewhere, and later on would like to interact with the content inside the frame.

```
const locator = page.locator('iframe[name="embedded"]');
// ...
const frameLocator = locator.contentFrame();
await frameLocator.getByRole('button').click();
```

- New method frameLocator.owner() converts a FrameLocator object to a Locator. This can be useful when you have a FrameLocator object obtained somewhere, and later on would like to interact with the `iframe` element.

```
const frameLocator = page.frameLocator('iframe[name="embedded"]');
// ...
const locator = frameLocator.owner();
await expect(locator).toBeVisible();
```

UI Mode Updates


PLAYWRIGHT
@fast
Status: all Projects: chromium-page
2/2 passed (100%)
example.spec.ts
has title
fast
smoke
718ms
get started link
fast
200ms
Actions
Metadata
Passed
Before Hooks
page.goto https://playwr
locator.click getByRole('
expect.toBeVisible getBy
After Hooks
Locator
Source
Call
Name
playwright.dev
styles.51270126.css
redirection.js

- See tags in the test list.
- Filter by tags by typing `@fast` or clicking on the tag itself.
- New shortcuts:
  - "F5" to run tests.
  - "Shift F5" to stop running tests.
  - "Ctrl `" to toggle test output.

Browser Versions

- Chromium 124.0.6367.8
- Mozilla Firefox 124.0
- WebKit 17.4

This version was also tested against the following stable channels:

- Google Chrome 123
- Microsoft Edge 123

Version 1.42

New APIs

- New method page.addLocatorHandler() registers a callback that will be invoked when specified element becomes visible and may block Playwright actions. The callback can get rid of the overlay. Here is an example that closes a cookie dialog when it appears:

```
// Setup the handler.
await page.addLocatorHandler(
    page.getByRole('heading', { name: 'Hej! You are in control of your
cookies.' }),
    async () => {
      await page.getByRole('button', { name: 'Accept all' }).click();
    });
// Write the test as usual.
await page.goto('https://www.ikea.com/');
await page.getByRole('link', { name: 'Collection of blue and white'
}).click();
await expect(page.getByRole('heading', { name: 'Light and easy'
})).toBeVisible();
```

- `expect(callback).toPass()` timeout can now be configured by `expect.toPass.timeout` option globally or in project config
- electronApplication.on('console') event is emitted when Electron main process calls console API methods.

```
electronApp.on('console', async msg => {
  const values = [];
  for (const arg of msg.args())
    values.push(await arg.jsonValue());
  console.log(...values);
});
await electronApp.evaluate(() => console.log('hello', 5, { foo: 'bar' }));
```

- New syntax for adding tags to the tests (@-tokens in the test title are still supported):

```
test('test customer login', {
  tag: ['@fast', '@login'],
}, async ({ page }) => {
  // ...
});
```

Use `--grep` command line option to run only tests with certain tags.

```
npx playwright test --grep @fast
```

- `--project` command line flag now supports '*' wildcard:

```
npx playwright test --project='*mobile*'
```

- New syntax for test annotations:

```
test('test full report', {
  annotation: [
    { type: 'issue', description:
'https://github.com/microsoft/playwright/issues/23180' },
    { type: 'docs', description: 'https://playwright.dev/docs/test-
annotations#tag-tests' },
  ],
}, async ({ page }) => {
  // ...
});
```

- page.pdf() accepts two new options `tagged` and `outline`.

Announcements

- ⚠️ Ubuntu 18 is not supported anymore.

Browser Versions

- Chromium 123.0.6312.4
- Mozilla Firefox 123.0
- WebKit 17.4

This version was also tested against the following stable channels:

- Google Chrome 122
- Microsoft Edge 123

Version 1.41

New APIs

- New method page.unrouteAll() removes all routes registered by page.route() and page.routeFromHAR(). Optionally allows to wait for ongoing routes to finish, or ignore any errors from them.
- New method browserContext.unrouteAll() removes all routes registered by browserContext.route() and browserContext.routeFromHAR(). Optionally allows to wait for ongoing routes to finish, or ignore any errors from them.
- New option `style` in page.screenshot() and locator.screenshot() to add custom CSS to the page before taking a screenshot.
- New option `stylePath` for methods expect(page).toHaveScreenshot() and expect(locator).toHaveScreenshot() to apply a custom stylesheet while making the screenshot.
- New `fileName` option for Blob reporter, to specify the name of the report to be created.

Browser Versions

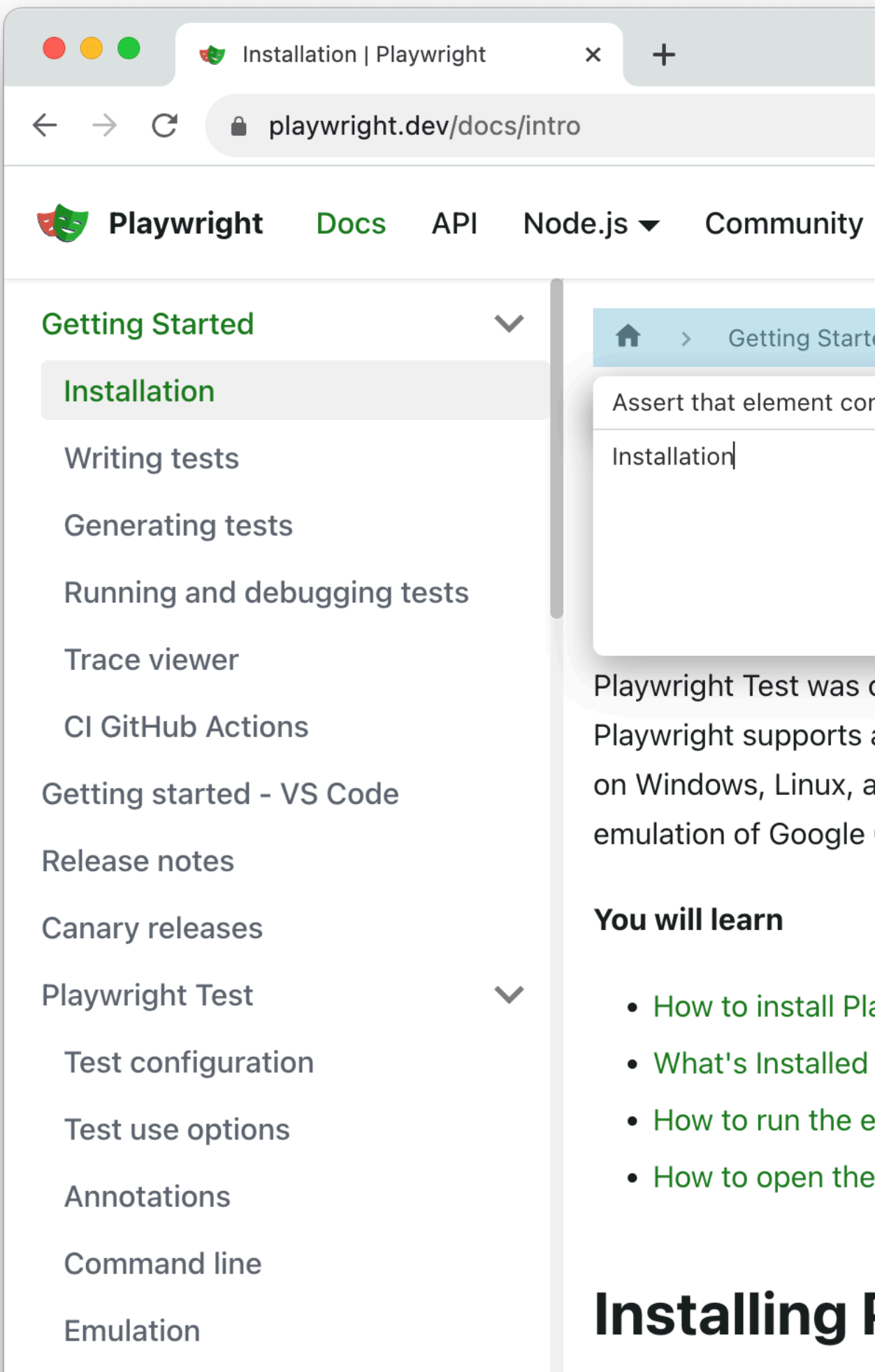
- Chromium 121.0.6167.57
- Mozilla Firefox 121.0
- WebKit 17.4

This version was also tested against the following stable channels:

- Google Chrome 120
- Microsoft Edge 120

Version 1.40

Test Generator Update


Installation | Playwright
playwright.dev/docs/intro
Playwright
Docs
API
Node.js
Community
Getting Started
Installation
Writing tests
Generating tests
Running and debugging tests
Trace viewer
CI GitHub Actions
Getting started - VS Code
Release notes
Canary releases
Playwright Test
Test configuration
Test use options
Annotations
Command line
Emulation
Getting Start
Assert that element cor
Installation
Playwright Test was
Playwright supports
on Windows, Linux,
emulation of Google
You will learn
How to install Pl
What's Installed
How to run the e
How to open the
Installing

New tools to generate assertions:

- "Assert visibility" tool generates expect(locator).toBeVisible().
- "Assert value" tool generates expect(locator).toHaveValue().
- "Assert text" tool generates expect(locator).toContainText().

Here is an example of a generated test with assertions:

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

test('test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  await page.getByRole('link', { name: 'Get started' }).click();
  await
expect(page.getByLabel('Breadcrumbs').getByRole('list')).toContainText('Ins
tallation');
  await expect(page.getByLabel('Search')).toBeVisible();
  await page.getByLabel('Search').click();
  await page.getByPlaceholder('Search docs').fill('locator');
  await expect(page.getByPlaceholder('Search
docs')).toHaveValue('locator');
});
```

New APIs

- Option `reason` in page.close(), browserContext.close() and browser.close(). Close reason is reported for all operations interrupted by the closure.
- Option `firefoxUserPrefs` in browserType.launchPersistentContext().

Other Changes

- Methods download.path() and download.createReadStream() throw an error for failed and cancelled downloads.
- Playwright docker image now comes with Node.js v20.

Browser Versions

- Chromium 120.0.6099.28
- Mozilla Firefox 119.0
- WebKit 17.4

This version was also tested against the following stable channels:

- Google Chrome 119
- Microsoft Edge 119

Version 1.39

Add custom matchers to your expect

You can extend Playwright assertions by providing custom matchers. These matchers will be available on the expect object.

test.spec.ts
```
import { expect as baseExpect } from '@playwright/test';
export const expect = baseExpect.extend({
  async toHaveAmount(locator: Locator, expected: number, options?: {
timeout?: number }) {
    // ... see documentation for how to write matchers.
  },
});

test('pass', async ({ page }) => {
  await expect(page.getByTestId('cart')).toHaveAmount(5);
});
```

See the documentation [for a full example](#).

Merge test fixtures

You can now merge test fixtures from multiple files or modules:

fixtures.ts
```
import { mergeTests } from '@playwright/test';
import { test as dbTest } from 'database-test-utils';
import { test as a11yTest } from 'a11y-test-utils';

export const test = mergeTests(dbTest, a11yTest);
```
test.spec.ts
```
import { test } from './fixtures';

test('passes', async ({ database, page, a11y }) => {
  // use database and a11y fixtures.
});
```

Merge custom expect matchers

You can now merge custom expect matchers from multiple files or modules:

fixtures.ts
```
import { mergeTests, mergeExpects } from '@playwright/test';
import { test as dbTest, expect as dbExpect } from 'database-test-utils';
import { test as a11yTest, expect as a11yExpect } from 'a11y-test-utils';

export const test = mergeTests(dbTest, a11yTest);
export const expect = mergeExpects(dbExpect, a11yExpect);
```
test.spec.ts
```
import { test, expect } from './fixtures';

test('passes', async ({ page, database }) => {
  await expect(database).toHaveDatabaseUser('admin');
  await expect(page).toPassA11yAudit();
});
```

Hide implementation details: box test steps

You can mark a test.step() as "boxed" so that errors inside it point to the step call site.

```
async function login(page) {
  await test.step('login', async () => {
    // ...
  }, { box: true });  // Note the "box" option here.
}
Error: Timed out 5000ms waiting for expect(locator).toBeVisible()
  ... error details omitted ...

  14 |   await page.goto('https://github.com/login');
> 15 |   await login(page);
     |         ^
  16 | });
```

See test.step() documentation for a full example.

New APIs

- expect(locator).toHaveAttribute()

Browser Versions

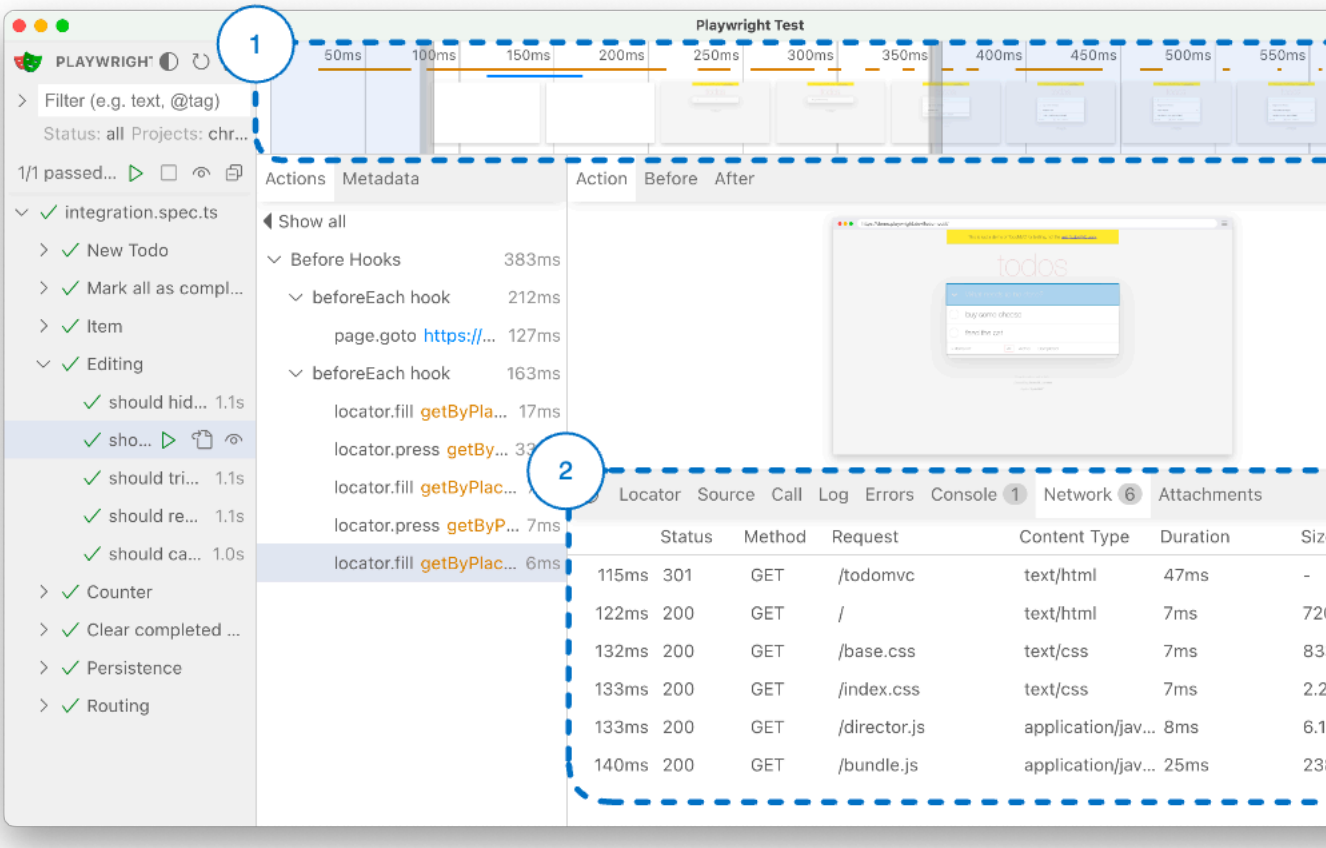
- Chromium 119.0.6045.9
- Mozilla Firefox 118.0.1
- WebKit 17.4

This version was also tested against the following stable channels:

- Google Chrome 118
- Microsoft Edge 118

Version 1.38

UI Mode Updates



1. Zoom into time range.
2. Network panel redesign.

New APIs

- browserContext.on('weberror')
- locator.pressSequentially()
- The reporter.onEnd() now reports `startTime` and total run `duration`.

Deprecations

- The following methods were deprecated: page.type(), frame.type(), locator.type() and elementHandle.type(). Please use locator.fill() instead which is much faster. Use locator.pressSequentially() only if there is a special keyboard handling on the page, and you need to press keys one-by-one.

Breaking Changes: Playwright no longer downloads browsers automatically

> **Note**: If you are using `@playwright/test` package, this change does not affect you.

Playwright recommends to use `@playwright/test` package and download browsers via `npx playwright install` command. If you are following this recommendation, nothing has changed for you.

However, up to v1.38, installing the `playwright` package instead of `@playwright/test` did automatically download browsers. This is no longer the case, and we recommend to explicitly download browsers via `npx playwright install` command.

v1.37 and earlier

`playwright` package was downloading browsers during `npm install`, while `@playwright/test` was not.

v1.38 and later

`playwright` and `@playwright/test` packages do not download browsers during `npm install`.

Recommended migration

Run `npx playwright install` to download browsers after `npm install`. For example, in your CI configuration:

```
- run: npm ci
- run: npx playwright install --with-deps
```

Alternative migration option - not recommended

Add `@playwright/browser-chromium`, `@playwright/browser-firefox` and `@playwright/browser-webkit` as a dependency. These packages download respective browsers during `npm install`. Make sure you keep the version of all playwright packages in sync:

```
// package.json
{
  "devDependencies": {
    "playwright": "1.38.0",
    "@playwright/browser-chromium": "1.38.0",
    "@playwright/browser-firefox": "1.38.0",
    "@playwright/browser-webkit": "1.38.0"
  }
}
```

Browser Versions

- Chromium 117.0.5938.62
- Mozilla Firefox 117.0
- WebKit 17.0

This version was also tested against the following stable channels:

- Google Chrome 116
- Microsoft Edge 116

Version 1.37

New `npx playwright merge-reports` tool

If you run tests on multiple shards, you can now merge all reports in a single HTML report (or any other report) using the new `merge-reports` CLI tool.

Using `merge-reports` tool requires the following steps:

1. Adding a new "blob" reporter to the config when running on CI:

   playwright.config.ts

   ```
   export default defineConfig({
     testDir: './tests',
     reporter: process.env.CI ? 'blob' : 'html',
   });
   ```

   The "blob" reporter will produce ".zip" files that contain all the information about the test run.

2. Copying all "blob" reports in a single shared location and running `npx playwright merge-reports`:

```
npx playwright merge-reports --reporter html ./all-blob-reports
```

Read more in our documentation.

📚 Debian 12 Bookworm Support

Playwright now supports Debian 12 Bookworm on both x86_64 and arm64 for Chromium, Firefox and WebKit. Let us know if you encounter any issues!

Linux support looks like this:

	Ubuntu 20.04	Ubuntu 22.04	Debian 11	Debian 12
Chromium	✅	✅	✅	✅
WebKit	✅	✅	✅	✅
Firefox	✅	✅	✅	✅

UI Mode Updates

- UI Mode now respects project dependencies. You can control which dependencies to respect by checking/unchecking them in a projects list.
- Console logs from the test are now displayed in the Console tab.

Browser Versions

- Chromium 116.0.5845.82
- Mozilla Firefox 115.0
- WebKit 17.0

This version was also tested against the following stable channels:

- Google Chrome 115
- Microsoft Edge 115

Version 1.36

🏝️ Summer maintenance release.

Browser Versions

- Chromium 115.0.5790.75
- Mozilla Firefox 115.0
- WebKit 17.0

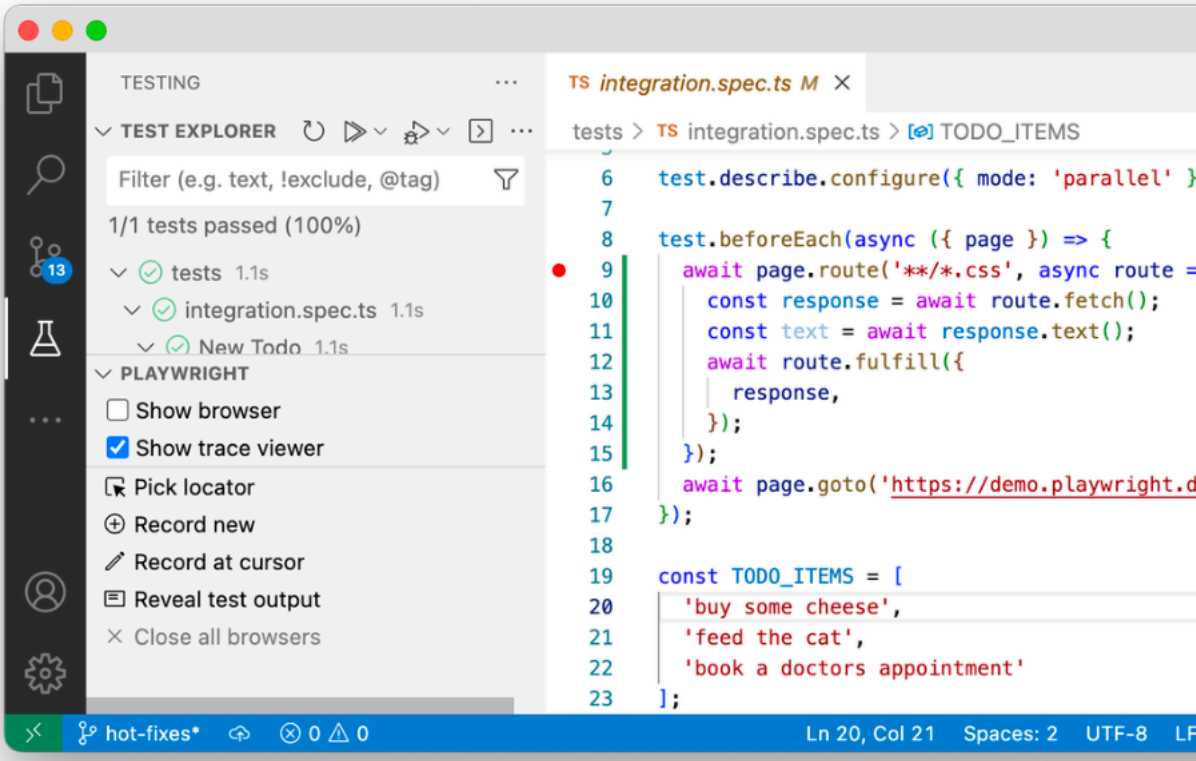
This version was also tested against the following stable channels:

- Google Chrome 114
- Microsoft Edge 114

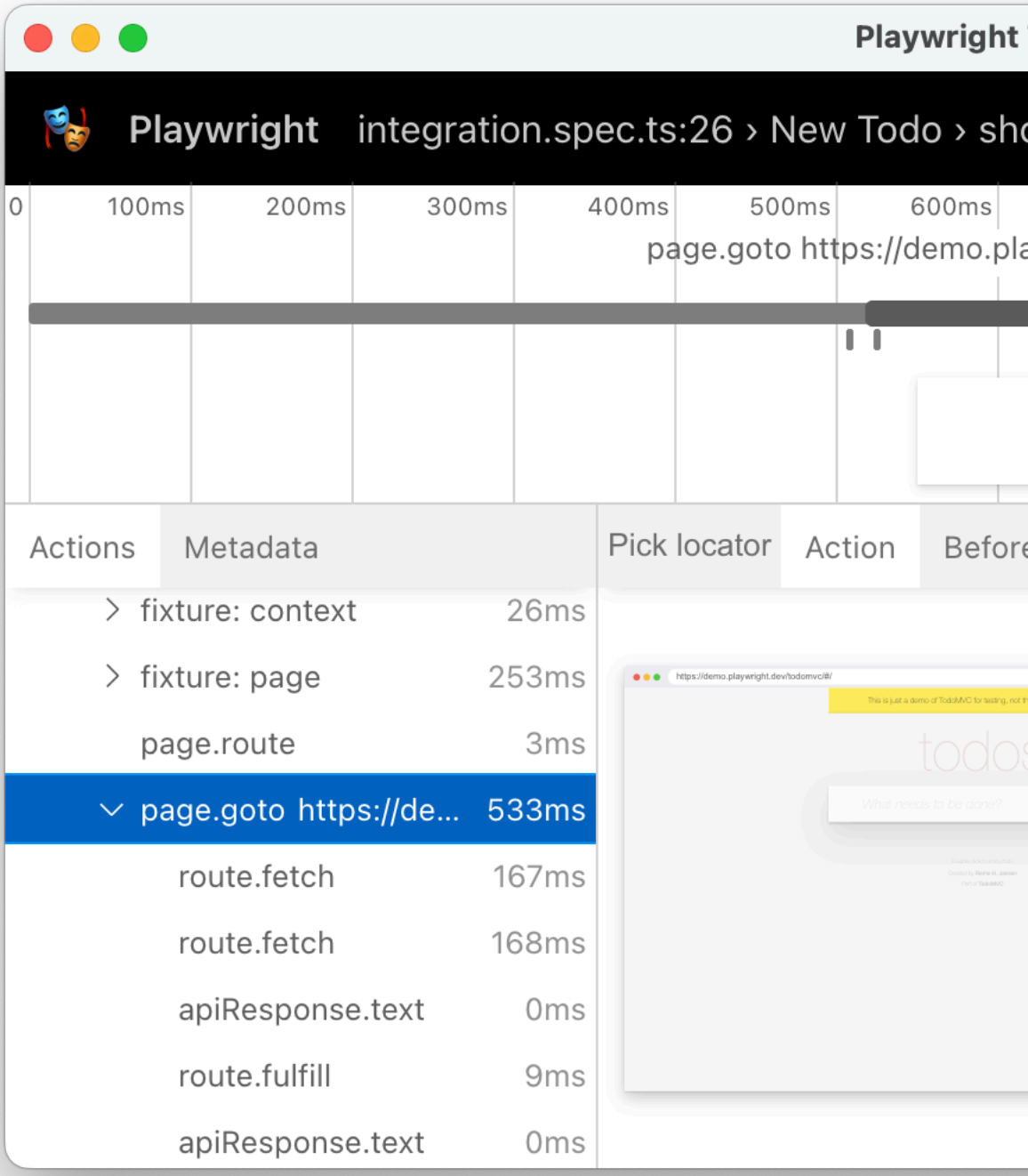
Version 1.35

Highlights

- UI mode is now available in VSCode Playwright extension via a new "Show trace viewer" button:



- UI mode and trace viewer mark network requests handled with page.route() and browserContext.route() handlers, as well as those issued via the API testing:



- New option `maskColor` for methods page.screenshot(), locator.screenshot(), expect(page).toHaveScreenshot() and expect(locator).toHaveScreenshot() to change default masking color:

```
await page.goto('https://playwright.dev');
await expect(page).toHaveScreenshot({
```

```
  mask: [page.locator('img')],
  maskColor: '#00FF00', // green
});
```

- New `uninstall` CLI command to uninstall browser binaries:

```
$ npx playwright uninstall # remove browsers installed by this installation
$ npx playwright uninstall --all # remove all ever-install Playwright browsers
```

- Both UI mode and trace viewer now could be opened in a browser tab:

```
$ npx playwright test --ui-port 0 # open UI mode in a tab on a random port
$ npx playwright show-trace --port 0 # open trace viewer in tab on a random port
```

⚠️ Breaking changes

- `playwright-core` binary got renamed from `playwright` to `playwright-core`. So if you use `playwright-core` CLI, make sure to update the name:

```
$ npx playwright-core install # the new way to install browsers when using playwright-core
```

  This change **does not** affect `@playwright/test` and `playwright` package users.

Browser Versions

- Chromium 115.0.5790.13
- Mozilla Firefox 113.0
- WebKit 16.4

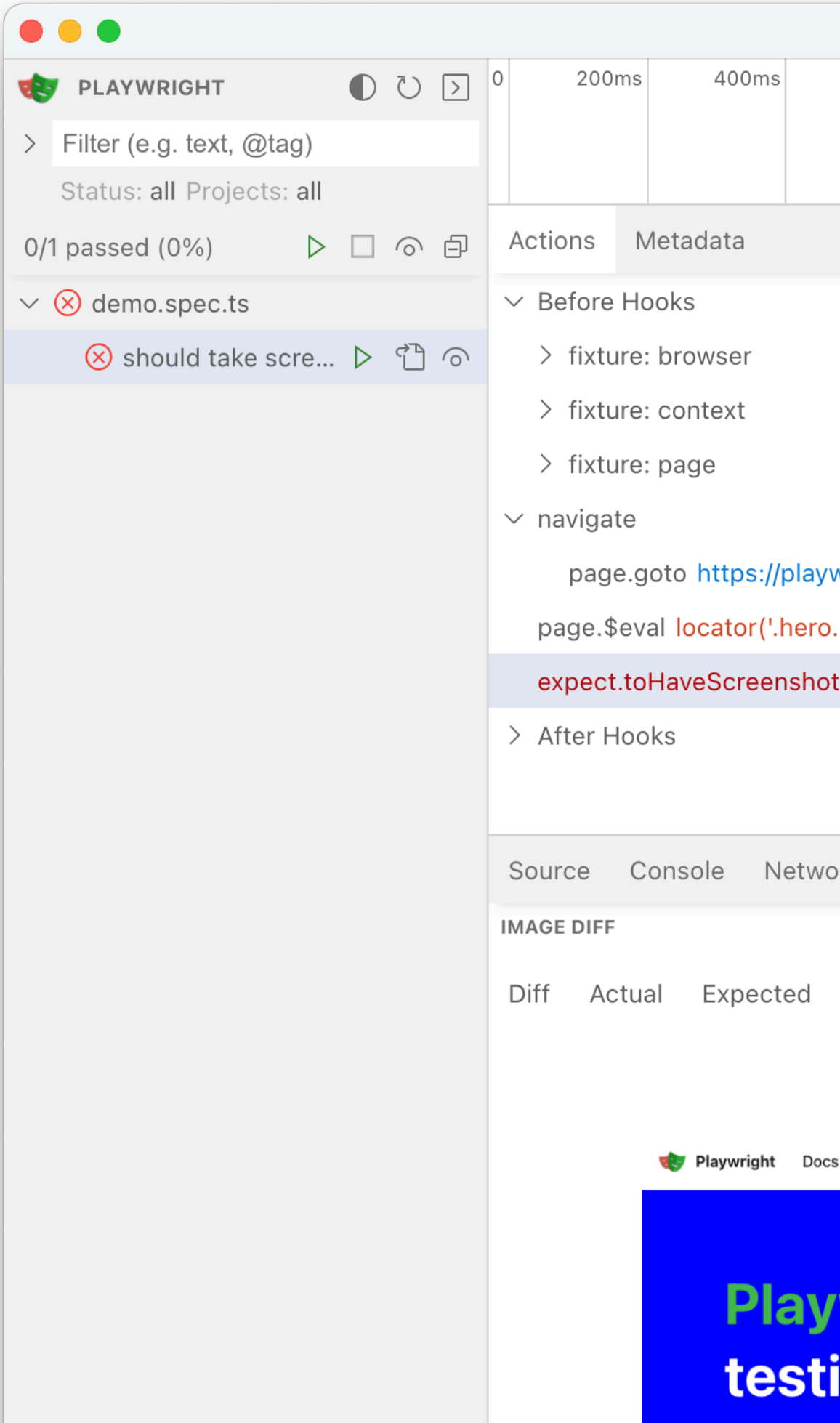
This version was also tested against the following stable channels:

- Google Chrome 114
- Microsoft Edge 114

Version 1.34

Highlights

- UI Mode now shows steps, fixtures and attachments:


PLAYWRIGHT
Filter (e.g. text, @tag)
Status: all Projects: all
0/1 passed (0%)
demo.spec.ts
should take scre...
0
200ms
400ms
Actions
Metadata
Before Hooks
fixture: browser
fixture: context
fixture: page
navigate
page.goto https://playw
page.$eval locator('.hero.
expect.toHaveScreenshot
After Hooks
Source
Console
Netwo
IMAGE DIFF
Diff
Actual
Expected
Playwright
Docs
Play
testi

- New property testProject.teardown to specify a project that needs to run after this and all dependent projects have finished. Teardown is useful to cleanup any resources acquired by this project.

  A common pattern would be a `setup` dependency with a corresponding `teardown`:

  playwright.config.ts

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

export default defineConfig({
  projects: [
    {
      name: 'setup',
      testMatch: /global.setup\.ts/,
      teardown: 'teardown',
    },
    {
      name: 'teardown',
      testMatch: /global.teardown\.ts/,
    },
    {
      name: 'chromium',
      use: devices['Desktop Chrome'],
      dependencies: ['setup'],
    },
    {
      name: 'firefox',
      use: devices['Desktop Firefox'],
      dependencies: ['setup'],
    },
    {
      name: 'webkit',
      use: devices['Desktop Safari'],
      dependencies: ['setup'],
    },
  ],
});
```

- New method `expect.configure` to create pre-configured expect instance with its own defaults such as `timeout` and `soft`.

```
const slowExpect = expect.configure({ timeout: 10000 });
await slowExpect(locator).toHaveText('Submit');

// Always do soft assertions.
const softExpect = expect.configure({ soft: true });
```

- New options `stderr` and `stdout` in testConfig.webServer to configure output handling:

  playwright.config.ts

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

export default defineConfig({
  // Run your local dev server before starting the tests
```

```
  webServer: {
    command: 'npm run start',
    url: 'http://127.0.0.1:3000',
    reuseExistingServer: !process.env.CI,
    stdout: 'pipe',
    stderr: 'pipe',
  },
});
```

- New locator.and() to create a locator that matches both locators.

```
const button =
page.getByRole('button').and(page.getByTitle('Subscribe'));
```

- New events browserContext.on('console') and browserContext.on('dialog') to subscribe to any dialogs and console messages from any page from the given browser context. Use the new methods consoleMessage.page() and dialog.page() to pin-point event source.

⚠️ Breaking changes

- `npx playwright test` no longer works if you install both `playwright` and `@playwright/test`. There's no need to install both, since you can always import browser automation APIs from `@playwright/test` directly:

  automation.ts

```
import { chromium, firefox, webkit } from '@playwright/test';
/* ... */
```

- Node.js 14 is no longer supported since it reached its end-of-life on April 30, 2023.

Browser Versions

- Chromium 114.0.5735.26
- Mozilla Firefox 113.0
- WebKit 16.4

This version was also tested against the following stable channels:

- Google Chrome 113
- Microsoft Edge 113

Version 1.33

Locators Update

- Use locator.or() to create a locator that matches either of the two locators. Consider a scenario where you'd like to click on a "New email" button, but sometimes a security settings dialog shows up instead. In this case, you can wait for either a "New email" button, or a dialog and act accordingly:

```
const newEmail = page.getByRole('button', { name: 'New email' });
const dialog = page.getByText('Confirm security settings');
await expect(newEmail.or(dialog)).toBeVisible();
if (await dialog.isVisible())
  await page.getByRole('button', { name: 'Dismiss' }).click();
await newEmail.click();
```

- Use new options `hasNot` and `hasNotText` in locator.filter() to find elements that **do not match** certain conditions.

```
const rowLocator = page.locator('tr');
await rowLocator
    .filter({ hasNotText: 'text in column 1' })
    .filter({ hasNot: page.getByRole('button', { name: 'column 2
button' }) })
    .screenshot();
```

- Use new web-first assertion expect(locator).toBeAttached() to ensure that the element is present in the page's DOM. Do not confuse with the expect(locator).toBeVisible() that ensures that element is both attached & visible.

New APIs

- locator.or()
- New option `hasNot` in locator.filter()
- New option `hasNotText` in locator.filter()
- expect(locator).toBeAttached()
- New option `timeout` in route.fetch()
- reporter.onExit()

⚠️ Breaking change

- The `mcr.microsoft.com/playwright:v1.33.0` now serves a Playwright image based on Ubuntu Jammy. To use the focal-based image, please use `mcr.microsoft.com/playwright:v1.33.0-focal` instead.

Browser Versions

- Chromium 113.0.5672.53
- Mozilla Firefox 112.0
- WebKit 16.4

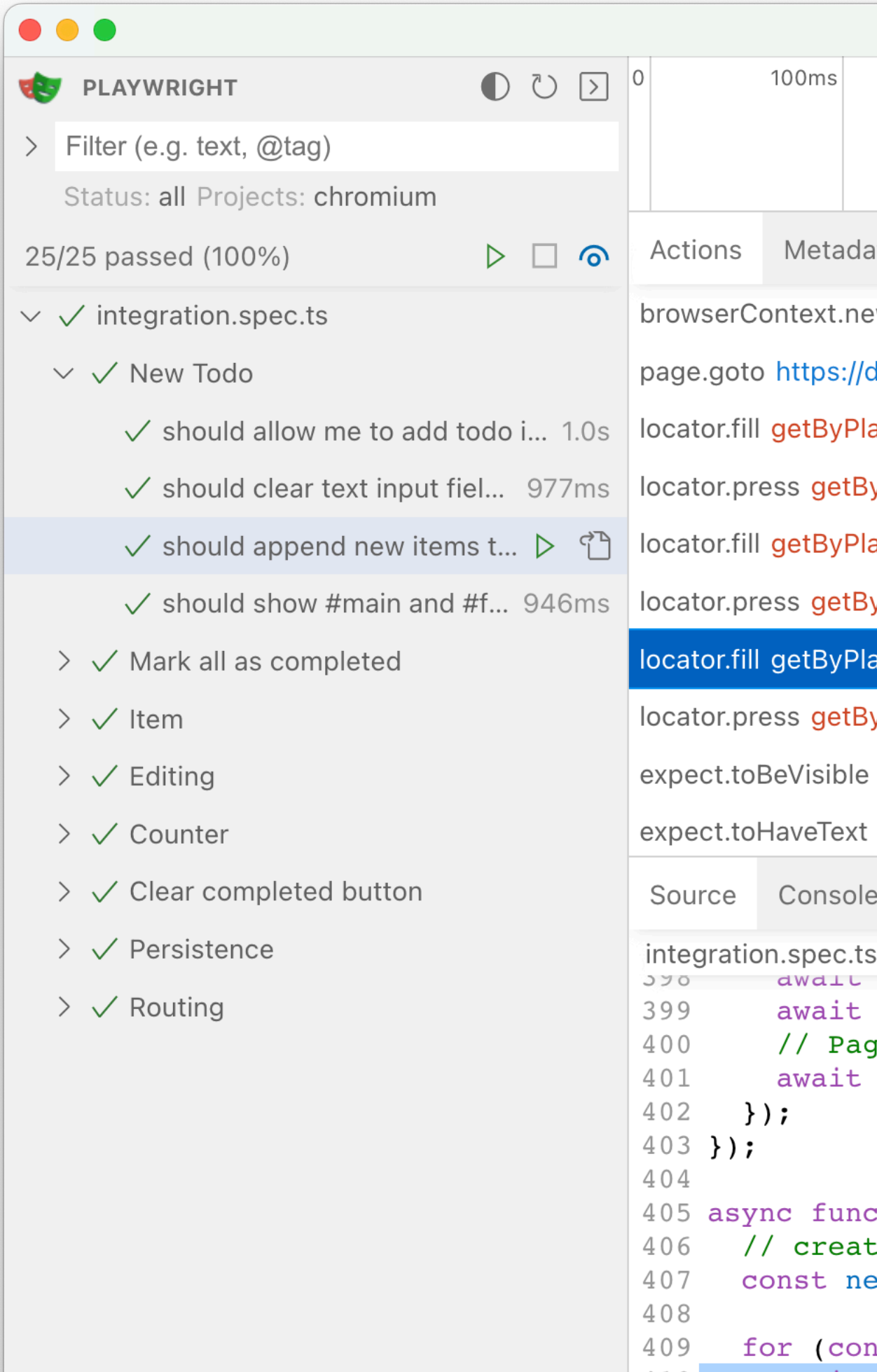
This version was also tested against the following stable channels:

- Google Chrome 112
- Microsoft Edge 112

Version 1.32

Introducing UI Mode (preview)

New UI Mode lets you explore, run and debug tests. Comes with a built-in watch mode.


PLAYWRIGHT
Filter (e.g. text, @tag)
Status: all Projects: chromium
25/25 passed (100%)
integration.spec.ts
New Todo
should allow me to add todo i... 1.0s
should clear text input fiel... 977ms
should append new items t...
should show #main and #f... 946ms
Mark all as completed
Item
Editing
Counter
Clear completed button
Persistence
Routing
0
100ms
Actions
Metada
browserContext.ne
page.goto https://d
locator.fill getByPla
locator.press getBy
locator.fill getByPla
locator.press getBy
locator.fill getByPla
locator.press getBy
expect.toBeVisible
expect.toHaveText
Source
Console
integration.spec.ts
399 await
400 // Pag
401 await
402 });
403 });
404
405 async func
406 // creat
407 const ne
408
409 for (con

Engage with a new flag `--ui`:

```
npx playwright test --ui
```

New APIs

- New options `updateMode` and `updateContent` in page.routeFromHAR() and browserContext.routeFromHAR().
- Chaining existing locator objects, see locator docs for details.
- New property testInfo.testId.
- New option `name` in method tracing.startChunk().

⚠️ Breaking change in component tests

Note: **component tests only**, does not affect end-to-end tests.

- `@playwright/experimental-ct-react` now supports **React 18 only**.
- If you're running component tests with React 16 or 17, please replace `@playwright/experimental-ct-react` with `@playwright/experimental-ct-react17`.

Browser Versions

- Chromium 112.0.5615.29
- Mozilla Firefox 111.0
- WebKit 16.4

This version was also tested against the following stable channels:

- Google Chrome 111
- Microsoft Edge 111

Version 1.31

New APIs

- New property testProject.dependencies to configure dependencies between projects.

  Using dependencies allows global setup to produce traces and other artifacts, see the setup steps in the test report and more.

  playwright.config.ts

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

  export default defineConfig({
    projects: [
      {
        name: 'setup',
        testMatch: /global.setup\.ts/,
  ```

```
    },
    {
      name: 'chromium',
      use: devices['Desktop Chrome'],
      dependencies: ['setup'],
    },
    {
      name: 'firefox',
      use: devices['Desktop Firefox'],
      dependencies: ['setup'],
    },
    {
      name: 'webkit',
      use: devices['Desktop Safari'],
      dependencies: ['setup'],
    },
  ],
});
```

- New assertion expect(locator).toBeInViewport() ensures that locator points to an element that intersects viewport, according to the intersection observer API.

```
const button = page.getByRole('button');

// Make sure at least some part of element intersects viewport.
await expect(button).toBeInViewport();

// Make sure element is fully outside of viewport.
await expect(button).not.toBeInViewport();

// Make sure that at least half of the element intersects viewport.
await expect(button).toBeInViewport({ ratio: 0.5 });
```

Miscellaneous

- DOM snapshots in trace viewer can be now opened in a separate window.
- New method `defineConfig` to be used in `playwright.config`.
- New option `Route.fetch.maxRedirects` for method route.fetch().
- Playwright now supports Debian 11 arm64.
- Official docker images now include Node 18 instead of Node 16.

⚠ Breaking change in component tests

Note: **component tests only**, does not affect end-to-end tests.

`playwright-ct.config` configuration file for component testing now requires calling `defineConfig`.

```
// Before

import { type PlaywrightTestConfig, devices } from
'@playwright/experimental-ct-react';
const config: PlaywrightTestConfig = {
  // ... config goes here ...
};
export default config;
```

Replace `config` variable definition with `defineConfig` call:

```
// After

import { defineConfig, devices } from '@playwright/experimental-ct-react';
export default defineConfig({
  // ... config goes here ...
});
```

Browser Versions

- Chromium 111.0.5563.19
- Mozilla Firefox 109.0
- WebKit 16.4

This version was also tested against the following stable channels:

- Google Chrome 110
- Microsoft Edge 110

Version 1.30

Browser Versions

- Chromium 110.0.5481.38
- Mozilla Firefox 108.0.2
- WebKit 16.4

This version was also tested against the following stable channels:

- Google Chrome 109
- Microsoft Edge 109

Version 1.29

New APIs

- New method route.fetch() and new option `json` for route.fulfill():

```
await page.route('**/api/settings', async route => {
  // Fetch original settings.
  const response = await route.fetch();

  // Force settings theme to a predefined value.
  const json = await response.json();
  json.theme = 'Solorized';

  // Fulfill with modified data.
  await route.fulfill({ json });
});
```

- New method locator.all() to iterate over all matching elements:

```
// Check all checkboxes!
const checkboxes = page.getByRole('checkbox');
for (const checkbox of await checkboxes.all())
  await checkbox.check();
```

- locator.selectOption() matches now by value or label:

```
<select multiple>
  <option value="red">Red</div>
  <option value="green">Green</div>
  <option value="blue">Blue</div>
</select>
await element.selectOption('Red');
```

- Retry blocks of code until all assertions pass:

```
await expect(async () => {
  const response = await page.request.get('https://api.example.com');
  await expect(response).toBeOK();
}).toPass();
```

  Read more in our documentation.

- Automatically capture **full page screenshot** on test failure:

  playwright.config.ts

```
import { defineConfig } from '@playwright/test';
export default defineConfig({
  use: {
    screenshot: {
      mode: 'only-on-failure',
      fullPage: true,
    }
  }
});
```

Miscellaneous

- Playwright Test now respects `jsconfig.json`.
- New options `args` and `proxy` for androidDevice.launchBrowser().
- Option `postData` in method route.continue() now supports Serializable values.

Browser Versions

- Chromium 109.0.5414.46
- Mozilla Firefox 107.0
- WebKit 16.4

This version was also tested against the following stable channels:

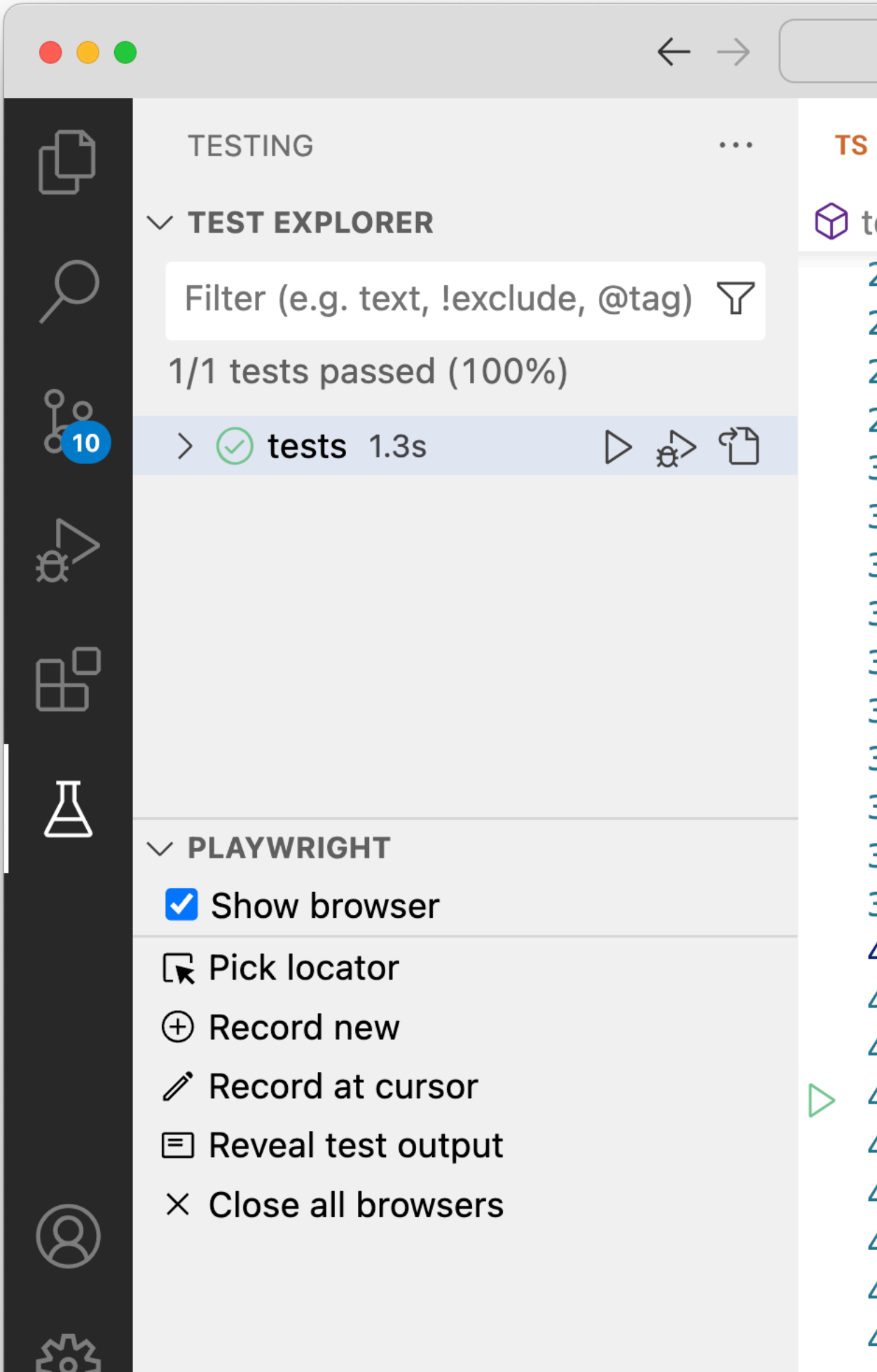
- Google Chrome 108

- Microsoft Edge 108

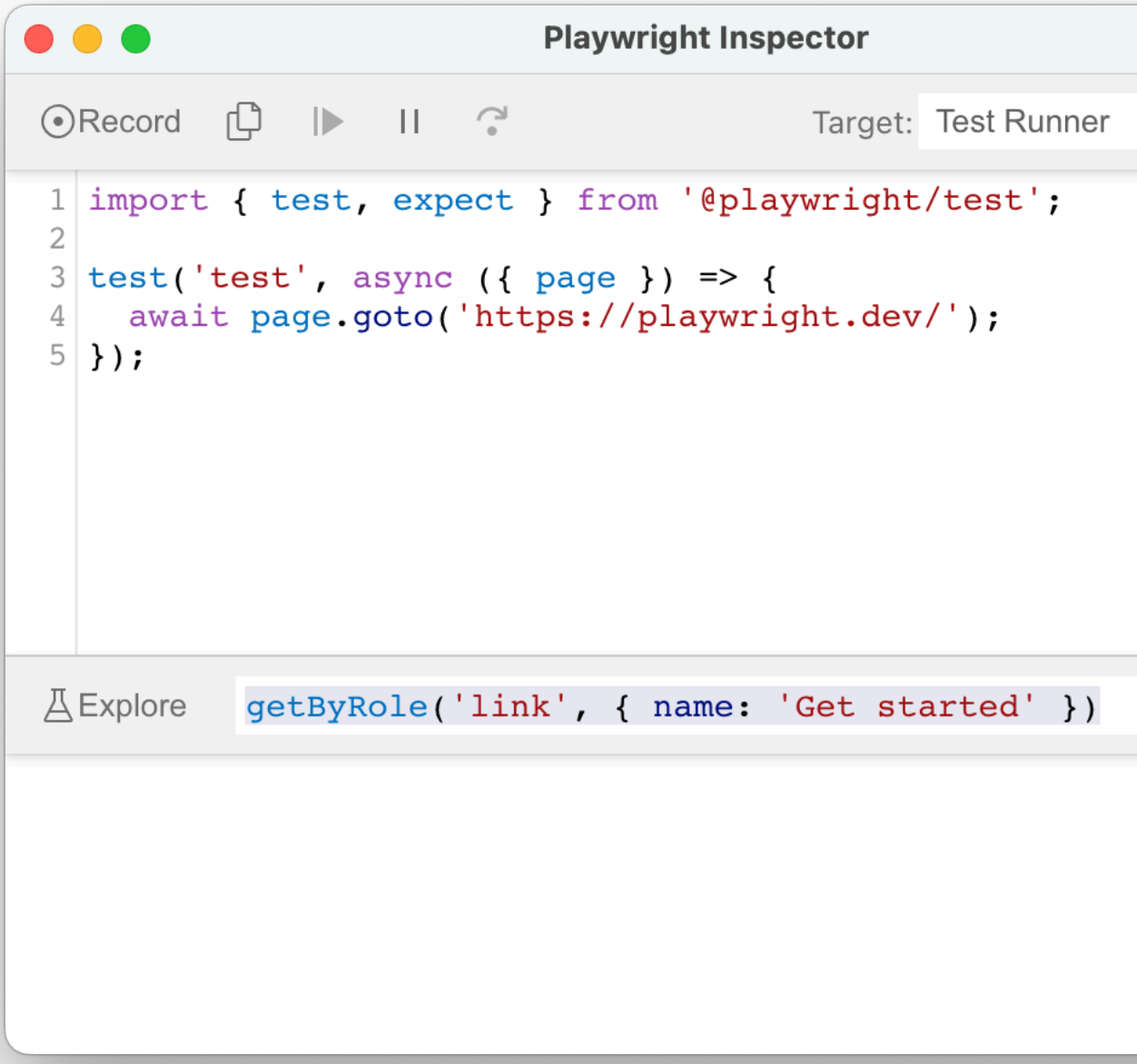
Version 1.28

Playwright Tools

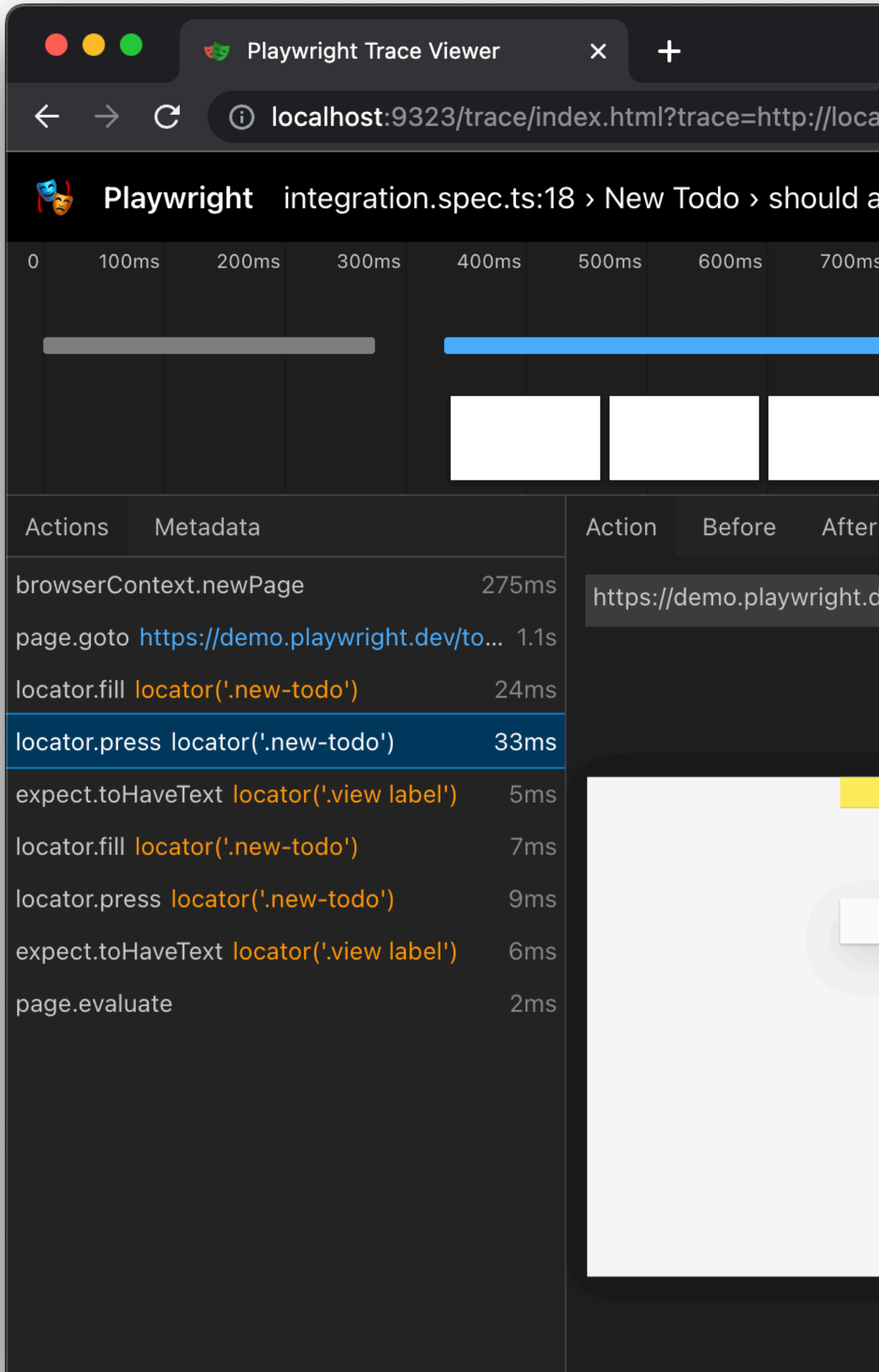
- **Record at Cursor in VSCode.** You can run the test, position the cursor at the end of the test and continue generating the test.


TESTING
TEST EXPLORER
Filter (e.g. text, !exclude, @tag)
1/1 tests passed (100%)
tests 1.3s
PLAYWRIGHT
Show browser
Pick locator
Record new
Record at cursor
Reveal test output
Close all browsers

- **Live Locators in VSCode.** You can hover and edit locators in VSCode to get them highlighted in the opened browser.
- **Live Locators in CodeGen.** Generate a locator for any element on the page using "Explore" tool.



- **Codegen and Trace Viewer Dark Theme.** Automatically picked up from operating system settings.


Playwright Trace Viewer
localhost:9323/trace/index.html?trace=http://loca
Playwright
integration.spec.ts:18 › New Todo › should a
0
100ms
200ms
300ms
400ms
500ms
600ms
700ms
Actions
Metadata
Action
Before
After
browserContext.newPage 275ms
page.goto https://demo.playwright.dev/to... 1.1s
locator.fill locator('.new-todo') 24ms
locator.press locator('.new-todo') 33ms
expect.toHaveText locator('.view label') 5ms
locator.fill locator('.new-todo') 7ms
locator.press locator('.new-todo') 9ms
expect.toHaveText locator('.view label') 6ms
page.evaluate 2ms
https://demo.playwright.d

Test Runner

- Configure retries and test timeout for a file or a test with test.describe.configure().

```
// Each test in the file will be retried twice and have a timeout of
20 seconds.
test.describe.configure({ retries: 2, timeout: 20_000 });
test('runs first', async ({ page }) => {});
test('runs second', async ({ page }) => {});
```

- Use testProject.snapshotPathTemplate and testConfig.snapshotPathTemplate to configure a template controlling location of snapshots generated by expect(page).toHaveScreenshot() and expect(value).toMatchSnapshot().

  playwright.config.ts

```
import { defineConfig } from '@playwright/test';
export default defineConfig({
  testDir: './tests',
  snapshotPathTemplate:
'{testDir}/__screenshots__/{testFilePath}/{arg}{ext}',
});
```

New APIs

- locator.blur()
- locator.clear()
- android.launchServer() and android.connect()
- androidDevice.on('close')

Browser Versions

- Chromium 108.0.5359.29
- Mozilla Firefox 106.0
- WebKit 16.4

This version was also tested against the following stable channels:

- Google Chrome 107
- Microsoft Edge 107

Version 1.27

Locators

With these new APIs writing locators is a joy:

- page.getByText() to locate by text content.
- page.getByRole() to locate by ARIA role, ARIA attributes and accessible name.
- page.getByLabel() to locate a form control by associated label's text.

- page.getByTestId() to locate an element based on its `data-testid` attribute (other attribute can be configured).
- page.getByPlaceholder() to locate an input by placeholder.
- page.getByAltText() to locate an element, usually image, by its text alternative.
- page.getByTitle() to locate an element by its title.

```
await page.getByLabel('User Name').fill('John');

await page.getByLabel('Password').fill('secret-password');

await page.getByRole('button', { name: 'Sign in' }).click();

await expect(page.getByText('Welcome, John!')).toBeVisible();
```

All the same methods are also available on Locator, FrameLocator and Frame classes.

Other highlights

- `workers` option in the `playwright.config.ts` now accepts a percentage string to use some of the available CPUs. You can also pass it in the command line:

  ```
  npx playwright test --workers=20%
  ```

- New options `host` and `port` for the html reporter.

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

  export default defineConfig({
    reporter: [['html', { host: 'localhost', port: '9223' }]],
  });
  ```

- New field `FullConfig.configFile` is available to test reporters, specifying the path to the config file if any.
- As announced in v1.25, Ubuntu 18 will not be supported as of Dec 2022. In addition to that, there will be no WebKit updates on Ubuntu 18 starting from the next Playwright release.

Behavior Changes

- expect(locator).toHaveAttribute() with an empty value does not match missing attribute anymore. For example, the following snippet will succeed when `button` **does not** have a `disabled` attribute.

  ```
  await expect(page.getByRole('button')).toHaveAttribute('disabled',
  '');
  ```

- Command line options `--grep` and `--grep-invert` previously incorrectly ignored `grep` and `grepInvert` options specified in the config. Now all of them are applied together.

Browser Versions

- Chromium 107.0.5304.18
- Mozilla Firefox 105.0.1
- WebKit 16.0

This version was also tested against the following stable channels:

- Google Chrome 106
- Microsoft Edge 106

Version 1.26

Assertions

- New option `enabled` for expect(locator).toBeEnabled().
- expect(locator).toHaveText() now pierces open shadow roots.
- New option `editable` for expect(locator).toBeEditable().
- New option `visible` for expect(locator).toBeVisible().

Other highlights

- New option `maxRedirects` for apiRequestContext.get() and others to limit redirect count.
- New command-line flag `--pass-with-no-tests` that allows the test suite to pass when no files are found.
- New command-line flag `--ignore-snapshots` to skip snapshot expectations, such as `expect(value).toMatchSnapshot()` and `expect(page).toHaveScreenshot()`.

Behavior Change

A bunch of Playwright APIs already support the `waitUntil: 'domcontentloaded'` option. For example:

```
await page.goto('https://playwright.dev', {
  waitUntil: 'domcontentloaded',
});
```

Prior to 1.26, this would wait for all iframes to fire the `DOMContentLoaded` event.

To align with web specification, the `'domcontentloaded'` value only waits for the target frame to fire the `'DOMContentLoaded'` event. Use `waitUntil: 'load'` to wait for all iframes.

Browser Versions

- Chromium 106.0.5249.30
- Mozilla Firefox 104.0
- WebKit 16.0

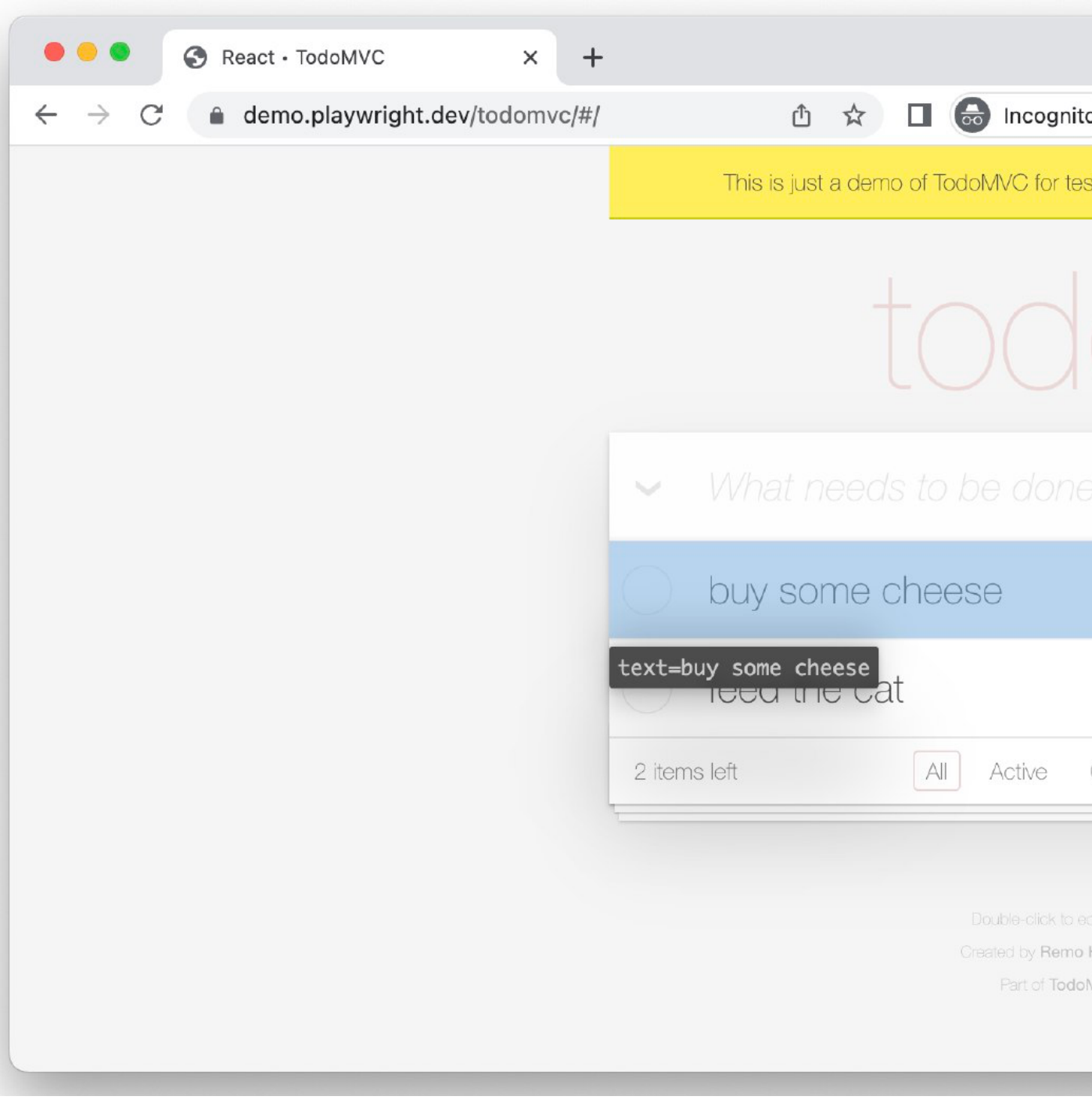
This version was also tested against the following stable channels:

- Google Chrome 105
- Microsoft Edge 105

Version 1.25

VSCode Extension

- Watch your tests running live & keep devtools open.
- Pick selector.
- Record new test from current page state.



Test Runner

- test.step() now returns the value of the step function:

```
test('should work', async ({ page }) => {
  const pageTitle = await test.step('get title', async () => {
    await page.goto('https://playwright.dev');
    return await page.title();
  });
  console.log(pageTitle);
});
```

- Added test.describe.fixme().
- New `'interrupted'` test status.
- Enable tracing via CLI flag: `npx playwright test --trace=on`.

Announcements

- 🎁 We now ship Ubuntu 22.04 Jammy Jellyfish docker image: `mcr.microsoft.com/playwright:v1.34.0-jammy`.
- 🪦 This is the last release with macOS 10.15 support (deprecated as of 1.21).
- 🪦 This is the last release with Node.js 12 support, we recommend upgrading to Node.js LTS (16).
- ⚠️ Ubuntu 18 is now deprecated and will not be supported as of Dec 2022.

Browser Versions

- Chromium 105.0.5195.19
- Mozilla Firefox 103.0
- WebKit 16.0

This version was also tested against the following stable channels:

- Google Chrome 104
- Microsoft Edge 104

Version 1.24

🌍 Multiple Web Servers in `playwright.config.ts`

Launch multiple web servers, databases, or other processes by passing an array of configurations:

playwright.config.ts

```
import { defineConfig } from '@playwright/test';
export default defineConfig({
  webServer: [
    {
      command: 'npm run start',
      url: 'http://127.0.0.1:3000',
```

```
      timeout: 120 * 1000,
      reuseExistingServer: !process.env.CI,
    },
    {
      command: 'npm run backend',
      url: 'http://127.0.0.1:3333',
      timeout: 120 * 1000,
      reuseExistingServer: !process.env.CI,
    }
  ],
  use: {
    baseURL: 'http://localhost:3000/',
  },
});
```

🐂 Debian 11 Bullseye Support

Playwright now supports Debian 11 Bullseye on x86_64 for Chromium, Firefox and WebKit. Let us know if you encounter any issues!

Linux support looks like this:

| | Ubuntu 20.04 | Ubuntu 22.04 | Debian 11 | :--- | :---: | :---: | :---: | :---: | | Chromium | ✅ | ✅ | ✅ | | WebKit | ✅ | ✅ | ✅ | | Firefox | ✅ | ✅ | ✅ |

🕵️ Anonymous Describe

It is now possible to call test.describe() to create suites without a title. This is useful for giving a group of tests a common option with test.use().

```
test.describe(() => {
  test.use({ colorScheme: 'dark' });

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

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

🧩 Component Tests Update

Playwright 1.24 Component Tests introduce `beforeMount` and `afterMount` hooks. Use these to configure your app for tests.

For example, this could be used to setup App router in Vue.js:

src/component.spec.ts

```
import { test } from '@playwright/experimental-ct-vue';
import { Component } from './mycomponent';

test('should work', async ({ mount }) => {
```

```
  const component = await mount(Component, {
    hooksConfig: {
      /* anything to configure your app */
    }
  });
});
```

playwright/index.ts

```
import { router } from '../router';
import { beforeMount } from '@playwright/experimental-ct-vue/hooks';

beforeMount(async ({ app, hooksConfig }) => {
  app.use(router);
});
```

A similar configuration in Next.js would look like this:

src/component.spec.jsx

```
import { test } from '@playwright/experimental-ct-react';
import { Component } from './mycomponent';

test('should work', async ({ mount }) => {
  const component = await mount(<Component></Component>, {
    // Pass mock value from test into `beforeMount`.
    hooksConfig: {
      router: {
        query: { page: 1, per_page: 10 },
        asPath: '/posts'
      }
    }
  });
});
```

playwright/index.js

```
import router from 'next/router';
import { beforeMount } from '@playwright/experimental-ct-react/hooks';

beforeMount(async ({ hooksConfig }) => {
  // Before mount, redefine useRouter to return mock value from test.
  router.useRouter = () => hooksConfig.router;
});
```

Version 1.23

Network Replay

Now you can record network traffic into a HAR file and re-use this traffic in your tests.

To record network into HAR file:

```
npx playwright open --save-har=github.har.zip https://github.com/microsoft
```

Alternatively, you can record HAR programmatically:

```
const context = await browser.newContext({
  recordHar: { path: 'github.har.zip' }
});
```

```
// ... do stuff ...
await context.close();
```

Use the new methods page.routeFromHAR() or browserContext.routeFromHAR() to serve matching responses from the HAR file:

```
await context.routeFromHAR('github.har.zip');
```

Read more in our documentation.

Advanced Routing

You can now use route.fallback() to defer routing to other handlers.

Consider the following example:

```
// Remove a header from all requests.
test.beforeEach(async ({ page }) => {
  await page.route('**/*', async route => {
    const headers = await route.request().allHeaders();
    delete headers['if-none-match'];
    await route.fallback({ headers });
  });
});

test('should work', async ({ page }) => {
  await page.route('**/*', async route => {
    if (route.request().resourceType() === 'image')
      await route.abort();
    else
      await route.fallback();
  });
});
```

Note that the new methods page.routeFromHAR() and browserContext.routeFromHAR() also participate in routing and could be deferred to.

Web-First Assertions Update

- New method expect(locator).toHaveValues() that asserts all selected values of `<select multiple>` element.
- Methods expect(locator).toContainText() and expect(locator).toHaveText() now accept `ignoreCase` option.

Component Tests Update

- Support for Vue2 via the `@playwright/experimental-ct-vue2` package.
- Support for component tests for create-react-app with components in `.js` files.

Read more about component testing with Playwright.

Miscellaneous

- If there's a service worker that's in your way, you can now easily disable it with a new context option `serviceWorkers`:

  playwright.config.ts

  ```
  export default {
    use: {
      serviceWorkers: 'block',
    }
  };
  ```

- Using `.zip` path for `recordHar` context option automatically zips the resulting HAR:

  ```
  const context = await browser.newContext({
    recordHar: {
      path: 'github.har.zip',
    }
  });
  ```

- If you intend to edit HAR by hand, consider using the `"minimal"` HAR recording mode that only records information that is essential for replaying:

  ```
  const context = await browser.newContext({
    recordHar: {
      path: 'github.har',
      mode: 'minimal',
    }
  });
  ```

- Playwright now runs on Ubuntu 22 amd64 and Ubuntu 22 arm64. We also publish new docker image `mcr.microsoft.com/playwright:v1.34.0-jammy`.

⚠️ Breaking Changes ⚠️

WebServer is now considered "ready" if request to the specified url has any of the following HTTP status codes:

- `200-299`
- `300-399` (new)
- `400`, `401`, `402`, `403` (new)

Version 1.22

Highlights

- Components Testing (preview)

  Playwright Test can now test your React, Vue.js or Svelte components. You can use all the features of Playwright Test (such as parallelization, emulation & debugging) while running components in real browsers.

  Here is what a typical component test looks like:

App.spec.tsx

```
import { test, expect } from '@playwright/experimental-ct-react';
import App from './App';

// Let's test component in a dark scheme!
test.use({ colorScheme: 'dark' });

test('should render', async ({ mount }) => {
  const component = await mount(<App></App>);

  // As with any Playwright test, assert locator text.
  await expect(component).toContainText('React');
  // Or do a screenshot 🚀
  await expect(component).toHaveScreenshot();
  // Or use any Playwright method
  await component.click();
});
```

Read more in our documentation.

- Role selectors that allow selecting elements by their ARIA role, ARIA attributes and accessible name.

```
// Click a button with accessible name "log in"
await page.locator('role=button[name="log in"]').click();
```

Read more in our documentation.

- New locator.filter() API to filter an existing locator

```
const buttons = page.locator('role=button');
// ...
const submitButton = buttons.filter({ hasText: 'Submit' });
await submitButton.click();
```

- New web-first assertions expect(page).toHaveScreenshot() and expect(locator).toHaveScreenshot() that wait for screenshot stabilization and enhances test reliability.

  The new assertions has screenshot-specific defaults, such as:

  - disables animations
  - uses CSS scale option

```
await page.goto('https://playwright.dev');
await expect(page).toHaveScreenshot();
```

The new expect(page).toHaveScreenshot() saves screenshots at the same location as expect(value).toMatchSnapshot().

Version 1.21

Highlights

- New role selectors that allow selecting elements by their ARIA role, ARIA attributes and accessible name.

  ```
  // Click a button with accessible name "log in"
  await page.locator('role=button[name="log in"]').click();
  ```

  Read more in our documentation.

- New `scale` option in page.screenshot() for smaller sized screenshots.
- New `caret` option in page.screenshot() to control text caret. Defaults to `"hide"`.
- New method `expect.poll` to wait for an arbitrary condition:

  ```
  // Poll the method until it returns an expected result.
  await expect.poll(async () => {
    const response = await page.request.get('https://api.example.com');
    return response.status();
  }).toBe(200);
  ```

  `expect.poll` supports most synchronous matchers, like `.toBe()`, `.toContain()`, etc. Read more in our documentation.

Behavior Changes

- ESM support when running TypeScript tests is now enabled by default. The `PLAYWRIGHT_EXPERIMENTAL_TS_ESM` env variable is no longer required.
- The `mcr.microsoft.com/playwright` docker image no longer contains Python. Please use `mcr.microsoft.com/playwright/python` as a Playwright-ready docker image with pre-installed Python.
- Playwright now supports large file uploads (100s of MBs) via locator.setInputFiles() API.

Browser Versions

- Chromium 101.0.4951.26
- Mozilla Firefox 98.0.2
- WebKit 15.4

This version was also tested against the following stable channels:

- Google Chrome 100
- Microsoft Edge 100

Version 1.20

Highlights

- New options for methods page.screenshot(), locator.screenshot() and elementHandle.screenshot():

    - Option `animations: "disabled"` rewinds all CSS animations and transitions to a consistent state
    - Option `mask: Locator[]` masks given elements, overlaying them with pink `#FF00FF` boxes.
- `expect().toMatchSnapshot()` now supports anonymous snapshots: when snapshot name is missing, Playwright Test will generate one automatically:

```
expect('Web is Awesome <3').toMatchSnapshot();
```

- New `maxDiffPixels` and `maxDiffPixelRatio` options for fine-grained screenshot comparison using `expect().toMatchSnapshot()`:

```
expect(await page.screenshot()).toMatchSnapshot({
  maxDiffPixels: 27, // allow no more than 27 different pixels.
});
```

  It is most convenient to specify `maxDiffPixels` or `maxDiffPixelRatio` once in testConfig.expect.

- Playwright Test now adds testConfig.fullyParallel mode. By default, Playwright Test parallelizes between files. In fully parallel mode, tests inside a single file are also run in parallel. You can also use `--fully-parallel` command line flag.

  playwright.config.ts

```
export default {
  fullyParallel: true,
};
```

- testProject.grep and testProject.grepInvert are now configurable per project. For example, you can now configure smoke tests project using `grep`:

  playwright.config.ts

```
export default {
  projects: [
    {
      name: 'smoke tests',
      grep: /@smoke/,
    },
  ],
};
```

- Trace Viewer now shows API testing requests.
- locator.highlight() visually reveals element(s) for easier debugging.

Announcements

- We now ship a designated Python docker image `mcr.microsoft.com/playwright/python`. Please switch over to it if you use Python. This is the last release that includes Python inside our javascript `mcr.microsoft.com/playwright` docker image.

- v1.20 is the last release to receive WebKit update for macOS 10.15 Catalina. Please update macOS to keep using latest & greatest WebKit!

Browser Versions

- Chromium 101.0.4921.0
- Mozilla Firefox 97.0.1
- WebKit 15.4

This version was also tested against the following stable channels:

- Google Chrome 99
- Microsoft Edge 99

Version 1.19

Playwright Test Update

- Playwright Test v1.19 now supports *soft assertions*. Failed soft assertions

  do not terminate test execution, but mark the test as failed.

  ```
  // Make a few checks that will not stop the test when failed...
  await expect.soft(page.locator('#status')).toHaveText('Success');
  await expect.soft(page.locator('#eta')).toHaveText('1 day');

  // ... and continue the test to check more things.
  await page.locator('#next-page').click();
  await expect.soft(page.locator('#title')).toHaveText('Make another
  order');
  ```

  Read more in our documentation

- You can now specify a **custom expect message** as a second argument to the `expect` and `expect.soft` functions, for example:

  ```
  await expect(page.locator('text=Name'), 'should be logged
  in').toBeVisible();
  ```

  The error would look like this:

  ```
      Error: should be logged in

      Call log:
        - expect.toBeVisible with timeout 5000ms
        - waiting for "getByText('Name')"


        2 |
        3 | test('example test', async({ page }) => {
      > 4 |   await expect(page.locator('text=Name'), 'should be logged
  in').toBeVisible();
          |
  ```

```
^
      5 | });
      6 |
```

Read more in our documentation

- By default, tests in a single file are run in order. If you have many independent tests in a single file, you can now run them in parallel with test.describe.configure().

Other Updates

- Locator now supports a `has` option that makes sure it contains another locator inside:

```
await page.locator('article', {
  has: page.locator('.highlight'),
}).click();
```

Read more in locator documentation

- New locator.page()
- page.screenshot() and locator.screenshot() now automatically hide blinking caret
- Playwright Codegen now generates locators and frame locators
- New option `url` in testConfig.webServer to ensure your web server is ready before running the tests
- New testInfo.errors and testResult.errors that contain all failed assertions and soft assertions.

Potentially breaking change in Playwright Test Global Setup

It is unlikely that this change will affect you, no action is required if your tests keep running as they did.

We've noticed that in rare cases, the set of tests to be executed was configured in the global setup by means of the environment variables. We also noticed some applications that were post processing the reporters' output in the global teardown. If you are doing one of the two, learn more

Browser Versions

- Chromium 100.0.4863.0
- Mozilla Firefox 96.0.1
- WebKit 15.4

This version was also tested against the following stable channels:

- Google Chrome 98
- Microsoft Edge 98

Version 1.18

Locator Improvements

- locator.dragTo()
- `expect(locator).toBeChecked({ checked })`
- Each locator can now be optionally filtered by the text it contains:

```
await page.locator('li', { hasText: 'my item'
}).locator('button').click();
```

Read more in locator documentation

Testing API improvements

- `expect(response).toBeOK()`
- `testInfo.attach()`
- `test.info()`

Improved TypeScript Support

1. Playwright Test now respects `tsconfig.json`'s `baseUrl` and `paths`, so you can use aliases
2. There is a new environment variable `PW_EXPERIMENTAL_TS_ESM` that allows importing ESM modules in your TS code, without the need for the compile step. Don't forget the `.js` suffix when you are importing your esm modules. Run your tests as follows:

```
npm i --save-dev @playwright/test@1.18.0-rc1
PW_EXPERIMENTAL_TS_ESM=1 npx playwright test
```

Create Playwright

The `npm init playwright` command is now generally available for your use:

```
# Run from your project's root directory
npm init playwright@latest
# Or create a new project
npm init playwright@latest new-project
```

This will create a Playwright Test configuration file, optionally add examples, a GitHub Action workflow and a first test `example.spec.ts`.

New APIs & changes

- new `testCase.repeatEachIndex` API
- `acceptDownloads` option now defaults to `true`

Breaking change: custom config options

Custom config options are a convenient way to parametrize projects with different values. Learn more in this guide.

Previously, any fixture introduced through test.extend() could be overridden in the testProject.use config section. For example,

```
// WRONG: THIS SNIPPET DOES NOT WORK SINCE v1.18.

// fixtures.js
const test = base.extend({
  myParameter: 'default',
});

// playwright.config.js
module.exports = {
  use: {
    myParameter: 'value',
  },
};
```

The proper way to make a fixture parametrized in the config file is to specify `option: true` when defining the fixture. For example,

```
// CORRECT: THIS SNIPPET WORKS SINCE v1.18.

// fixtures.js
const test = base.extend({
  // Fixtures marked as "option: true" will get a value specified in the
config,
  // or fallback to the default value.
  myParameter: ['default', { option: true }],
});

// playwright.config.js
module.exports = {
  use: {
    myParameter: 'value',
  },
};
```

Browser Versions

- Chromium 99.0.4812.0
- Mozilla Firefox 95.0
- WebKit 15.4

This version was also tested against the following stable channels:

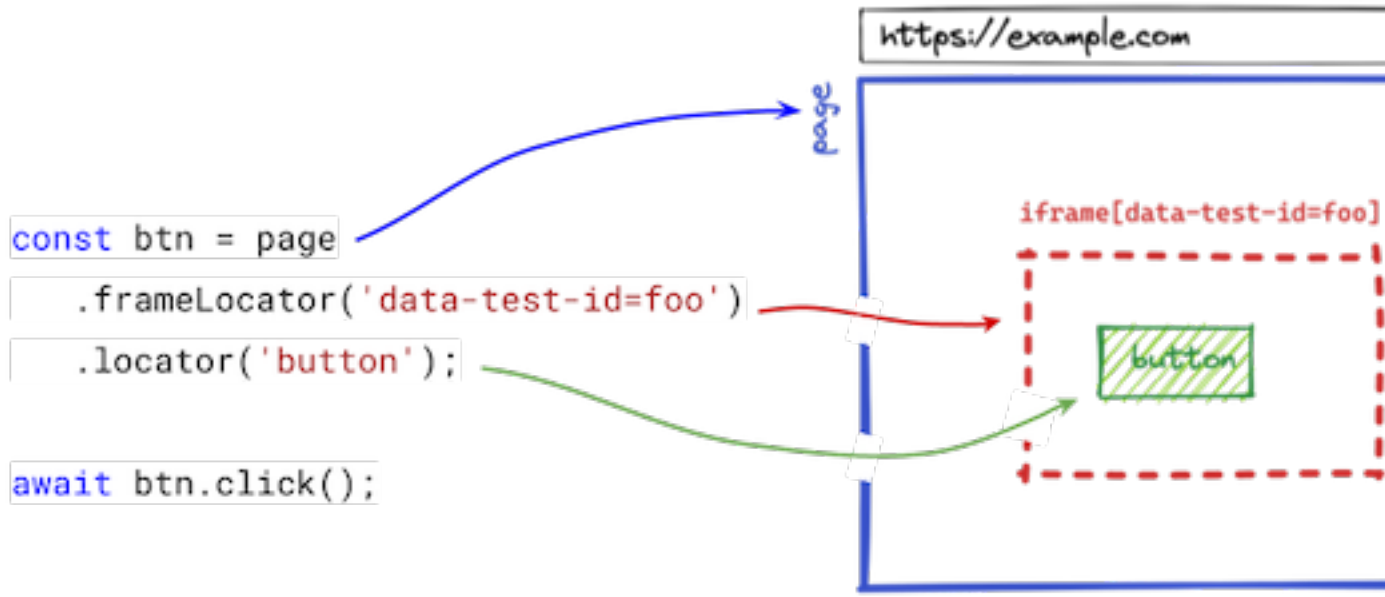
- Google Chrome 97
- Microsoft Edge 97

Version 1.17

Frame Locators

Playwright 1.17 introduces frame locators - a locator to the iframe on the page. Frame locators capture the logic sufficient to retrieve the `iframe` and then locate elements in that

iframe. Frame locators are strict by default, will wait for `iframe` to appear and can be used in Web-First assertions.



Frame locators can be created with either page.frameLocator() or locator.frameLocator() method.

```
const locator = page.frameLocator('#my-iframe').locator('text=Submit');
await locator.click();
```

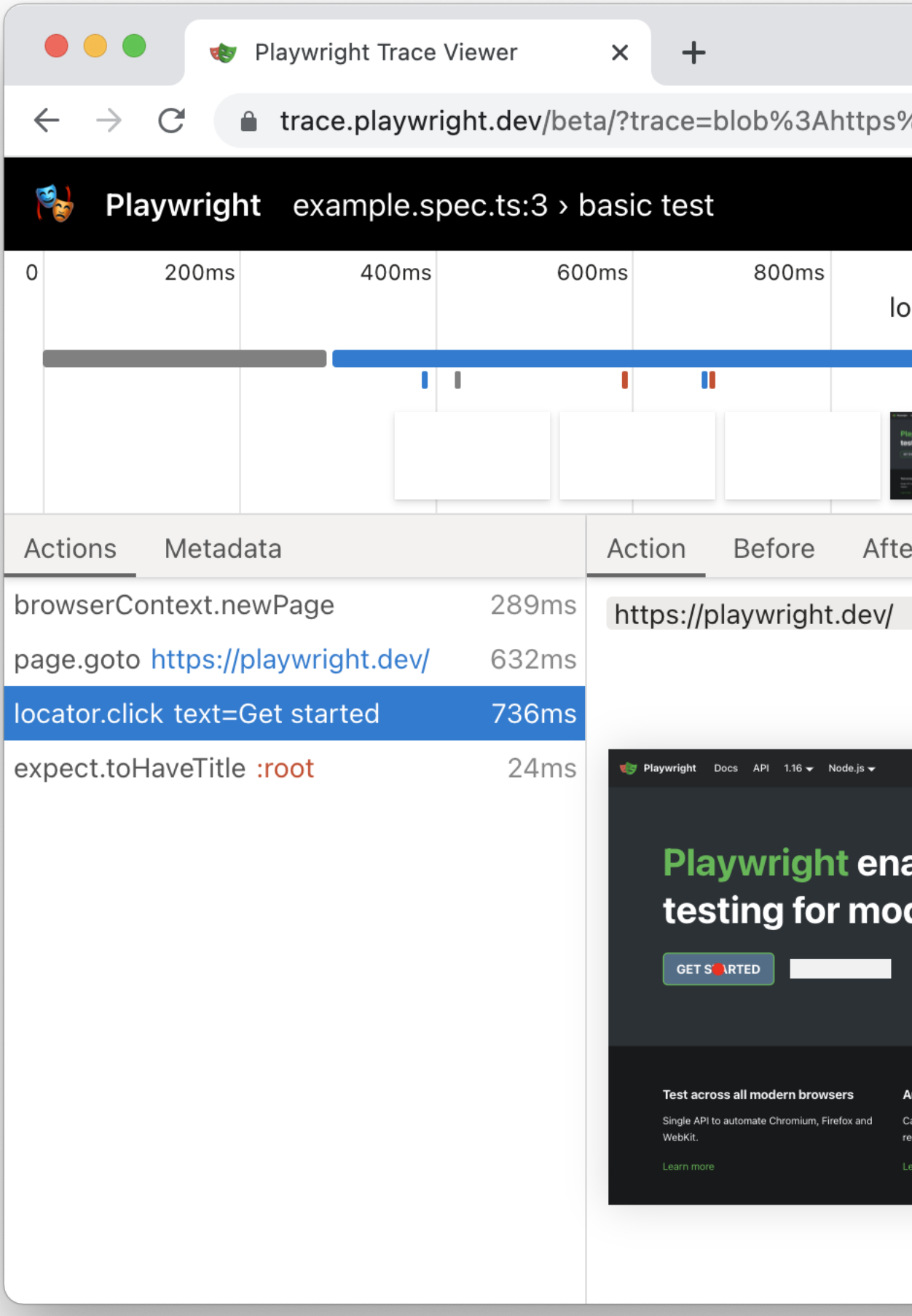
Read more at our documentation.

Trace Viewer Update

Playwright Trace Viewer is now **available online** at https://trace.playwright.dev! Just drag-and-drop your `trace.zip` file to inspect its contents.

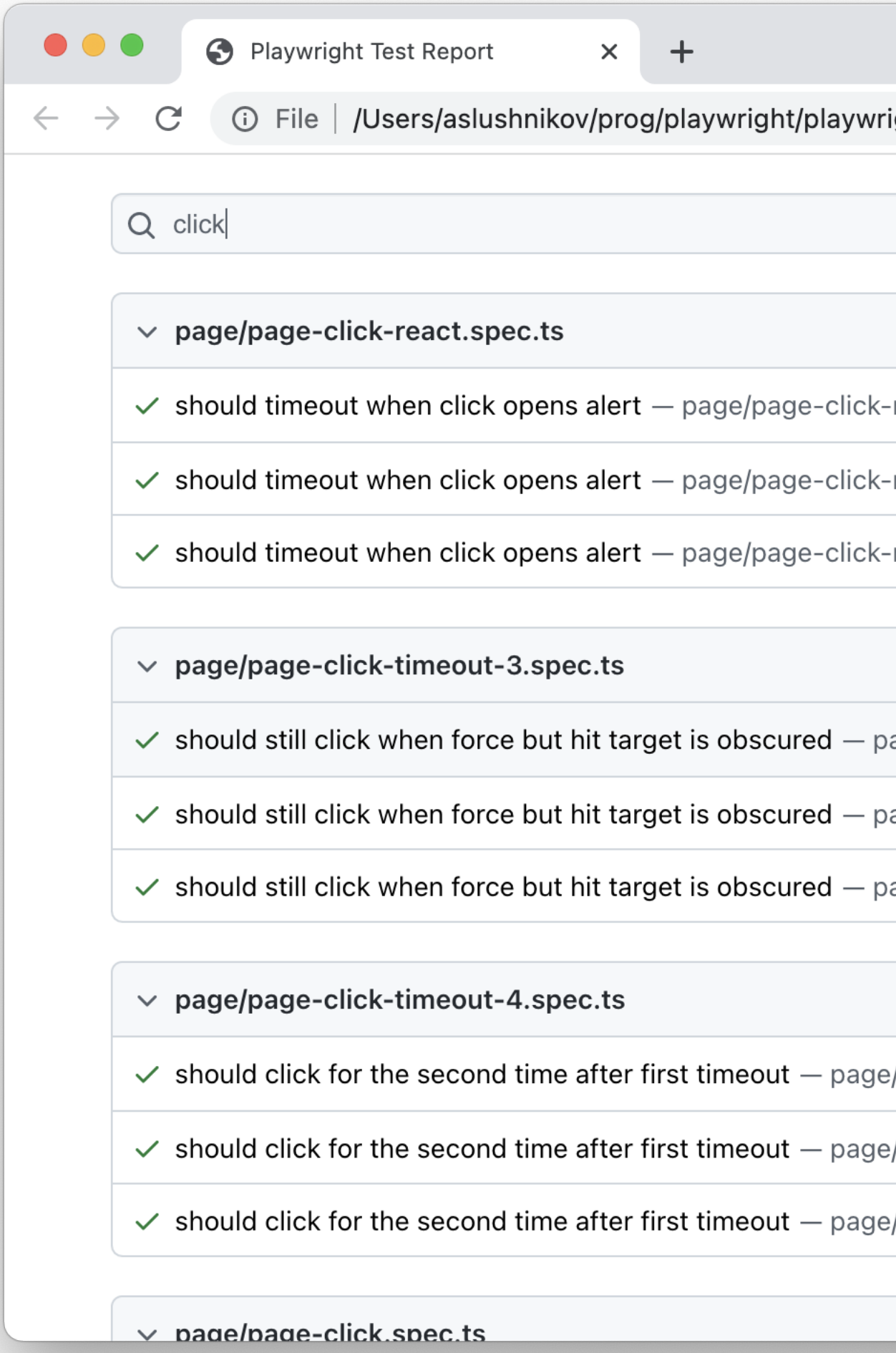
> **NOTE**: trace files are not uploaded anywhere; trace.playwright.dev is a progressive web application that processes traces locally.

- Playwright Test traces now include sources by default (these could be turned off with tracing option)
- Trace Viewer now shows test name
- New trace metadata tab with browser details
- Snapshots now have URL bar


Playwright Trace Viewer
trace.playwright.dev/beta/?trace=blob%3Ahttps%
Playwright example.spec.ts:3 › basic test
0
200ms
400ms
600ms
800ms
Actions
Metadata
Action
Before
browserContext.newPage 289ms
page.goto https://playwright.dev/ 632ms
locator.click text=Get started 736ms
expect.toHaveTitle :root 24ms
https://playwright.dev/
Playwright
Docs
API
1.16
Node.js
Playwright ena
testing for mod
GET STARTED
Test across all modern browsers
Single API to automate Chromium, Firefox and WebKit.
Learn more

HTML Report Update

- HTML report now supports dynamic filtering
- Report is now a **single static HTML file** that could be sent by e-mail or as a slack attachment.


Playwright Test Report
File
/Users/aslushnikov/prog/playwright/playwri
click
page/page-click-react.spec.ts
should timeout when click opens alert — page/page-click-
should timeout when click opens alert — page/page-click-
should timeout when click opens alert — page/page-click-
page/page-click-timeout-3.spec.ts
should still click when force but hit target is obscured — pa
should still click when force but hit target is obscured — pa
should still click when force but hit target is obscured — pa
page/page-click-timeout-4.spec.ts
should click for the second time after first timeout — page/
should click for the second time after first timeout — page/
should click for the second time after first timeout — page/
page/page-click.spec.ts

Ubuntu ARM64 support + more

- Playwright now supports **Ubuntu 20.04 ARM64**. You can now run Playwright tests inside Docker on Apple M1 and on Raspberry Pi.
- You can now use Playwright to install stable version of Edge on Linux:

```
npx playwright install msedge
```

New APIs

- Tracing now supports a `'title'` option
- Page navigations support a new `'commit'` waiting option
- HTML reporter got new configuration options
- `testConfig.snapshotDir` option
- `testInfo.parallelIndex`
- `testInfo.titlePath`
- `testOptions.trace` has new options
- `expect.toMatchSnapshot` supports subdirectories
- `reporter.printsToStdio()`

Version 1.16

🎭 Playwright Test

API Testing

Playwright 1.16 introduces new API Testing that lets you send requests to the server directly from Node.js! Now you can:

- test your server API
- prepare server side state before visiting the web application in a test
- validate server side post-conditions after running some actions in the browser

To do a request on behalf of Playwright's Page, use **new page.request API**:

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

test('context fetch', async ({ page }) => {
  // Do a GET request on behalf of page
  const response = await page.request.get('http://example.com/foo.json');
  // ...
});
```

To do a stand-alone request from node.js to an API endpoint, use **new `request` fixture**:

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

test('context fetch', async ({ request }) => {
  // Do a GET request on behalf of page
  const response = await request.get('http://example.com/foo.json');
```

```
  // ...
});
```

Read more about it in our API testing guide.

Response Interception

It is now possible to do response interception by combining API Testing with request interception.

For example, we can blur all the images on the page:

```
import { test, expect } from '@playwright/test';
import jimp from 'jimp'; // image processing library

test('response interception', async ({ page }) => {
  await page.route('**/*.jpeg', async route => {
    const response = await page._request.fetch(route.request());
    const image = await jimp.read(await response.body());
    await image.blur(5);
    await route.fulfill({
      response,
      body: await image.getBufferAsync('image/jpeg'),
    });
  });
  const response = await page.goto('https://playwright.dev');
  expect(response.status()).toBe(200);
});
```

Read more about response interception.

New HTML reporter

Try it out new HTML reporter with either `--reporter=html` or a `reporter` entry in `playwright.config.ts` file:

```
$ npx playwright test --reporter=html
```

The HTML reporter has all the information about tests and their failures, including surfacing trace and image artifacts.

should render proper landing page

landing.spec.ts:3

✕ RUN

Errors

```
Error: Snapshot comparison failed:

Expected: /Users/aslushnikov/prog/tmp/test-results/landing-
Received: /Users/aslushnikov/prog/tmp/test-results/landing-
    Diff: /Users/aslushnikov/prog/tmp/test-results/landing-

  4 |   await page.goto('https://playwright.dev');
  5 |   // await page.$eval('.hero', e => e.remove());
> 6 |   expect(await page.screenshot()).toMatchSnapshot('fo
    |                                   ^
  7 | });
  8 |

    at /Users/aslushnikov/prog/tmp/landing.spec.ts:6:35
    at WorkerRunner._runTestWithBeforeHooks (/Users/aslushn
```

Test Steps

- ✓ Before Hooks
- ✓ page.goto(https://playwright.dev) — landing.spec.ts:4
- ✓ page.screenshot — landing.spec.ts:6
- ✕ expect.toMatchSnapshot — landing.spec.ts:6
- ✓ After Hooks

Image mismatch

Read more about our reporters.

🎭 Playwright Library

locator.waitFor

Wait for a locator to resolve to a single element with a given state. Defaults to the `state: 'visible'`.

Comes especially handy when working with lists:

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

test('context fetch', async ({ page }) => {
  const completeness = page.locator('text=Success');
  await completeness.waitFor();
  expect(await page.screenshot()).toMatchSnapshot('screen.png');
});
```

Read more about locator.waitFor().

Docker support for Arm64

Playwright Docker image is now published for Arm64 so it can be used on Apple Silicon.

Read more about Docker integration.

🎭 Playwright Trace Viewer

- web-first assertions inside trace viewer
- run trace viewer with `npx playwright show-trace` and drop trace files to the trace viewer PWA
- API testing is integrated with trace viewer
- better visual attribution of action targets

Read more about Trace Viewer.

Browser Versions

- Chromium 97.0.4666.0
- Mozilla Firefox 93.0
- WebKit 15.4

This version of Playwright was also tested against the following stable channels:

- Google Chrome 94
- Microsoft Edge 94

Version 1.15

🎭 Playwright Library

🖱️ Mouse Wheel

By using mouse.wheel() you are now able to scroll vertically or horizontally.

📜 New Headers API

Previously it was not possible to get multiple header values of a response. This is now possible and additional helper functions are available:

- request.allHeaders()
- request.headersArray()
- request.headerValue()
- response.allHeaders()
- response.headersArray()
- response.headerValue()
- response.headerValues()

🌈 Forced-Colors emulation

Its now possible to emulate the `forced-colors` CSS media feature by passing it in the browser.newContext() or calling page.emulateMedia().

New APIs

- page.route() accepts new `times` option to specify how many times this route should be matched.
- page.setChecked() and locator.setChecked() were introduced to set the checked state of a checkbox.
- request.sizes() Returns resource size information for given http request.
- tracing.startChunk() - Start a new trace chunk.
- tracing.stopChunk() - Stops a new trace chunk.

🎭 Playwright Test

🤝 `test.parallel()` run tests in the same file in parallel

```
test.describe.parallel('group', () => {
  test('runs in parallel 1', async ({ page }) => {
  });
  test('runs in parallel 2', async ({ page }) => {
  });
});
```

By default, tests in a single file are run in order. If you have many independent tests in a single file, you can now run them in parallel with test.describe.parallel(title, callback).

🛠️ Add `--debug` CLI flag

By using `npx playwright test --debug` it will enable the Playwright Inspector for you to debug your tests.

Browser Versions

- Chromium 96.0.4641.0
- Mozilla Firefox 92.0
- WebKit 15.0

Version 1.14

🎭 Playwright Library

⚡ New "strict" mode

Selector ambiguity is a common problem in automation testing. **"strict" mode** ensures that your selector points to a single element and throws otherwise.

Pass `strict: true` into your action calls to opt in.

```
// This will throw if you have more than one button!
await page.click('button', { strict: true });
```

📍 New Locators API

Locator represents a view to the element(s) on the page. It captures the logic sufficient to retrieve the element at any given moment.

The difference between the Locator and ElementHandle is that the latter points to a particular element, while Locator captures the logic of how to retrieve that element.

Also, locators are **"strict" by default**!

```
const locator = page.locator('button');
await locator.click();
```

Learn more in the documentation.

🧩 Experimental React and Vue selector engines

React and Vue selectors allow selecting elements by its component name and/or property values. The syntax is very similar to attribute selectors and supports all attribute selector operators.

```
await page.locator('_react=SubmitButton[enabled=true]').click();
await page.locator('_vue=submit-button[enabled=true]').click();
```

Learn more in the react selectors documentation and the vue selectors documentation.

✨ New `nth` and `visible` selector engines

- `nth` selector engine is equivalent to the `:nth-match` pseudo class, but could be combined with other selector engines.
- `visible` selector engine is equivalent to the `:visible` pseudo class, but could be combined with other selector engines.

```
// select the first button among all buttons
await button.click('button >> nth=0');
// or if you are using locators, you can use first(), nth() and last()
await page.locator('button').first().click();

// click a visible button
await button.click('button >> visible=true');
```

🎭 Playwright Test

✅ Web-First Assertions

`expect` now supports lots of new web-first assertions.

Consider the following example:

```
await expect(page.locator('.status')).toHaveText('Submitted');
```

Playwright Test will be re-testing the node with the selector `.status` until fetched Node has the `"Submitted"` text. It will be re-fetching the node and checking it over and over, until the condition is met or until the timeout is reached. You can either pass this timeout or configure it once via the `testProject.expect` value in test config.

By default, the timeout for assertions is not set, so it'll wait forever, until the whole test times out.

List of all new assertions:

- `expect(locator).toBeChecked()`
- `expect(locator).toBeDisabled()`
- `expect(locator).toBeEditable()`
- `expect(locator).toBeEmpty()`
- `expect(locator).toBeEnabled()`
- `expect(locator).toBeFocused()`
- `expect(locator).toBeHidden()`
- `expect(locator).toBeVisible()`
- `expect(locator).toContainText(text, options?)`
- `expect(locator).toHaveAttribute(name, value)`
- `expect(locator).toHaveClass(expected)`
- `expect(locator).toHaveCount(count)`
- `expect(locator).toHaveCSS(name, value)`
- `expect(locator).toHaveId(id)`
- `expect(locator).toHaveJSProperty(name, value)`
- `expect(locator).toHaveText(expected, options)`
- `expect(page).toHaveTitle(title)`

- `expect(page).toHaveURL(url)`
- `expect(locator).toHaveValue(value)`

⛓ Serial mode with `describe.serial`

Declares a group of tests that should always be run serially. If one of the tests fails, all subsequent tests are skipped. All tests in a group are retried together.

```
test.describe.serial('group', () => {
  test('runs first', async ({ page }) => { /* ... */ });
  test('runs second', async ({ page }) => { /* ... */ });
});
```

Learn more in the documentation.

🐾 Steps API with `test.step`

Split long tests into multiple steps using `test.step()` API:

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

test('test', async ({ page }) => {
  await test.step('Log in', async () => {
    // ...
  });
  await test.step('news feed', async () => {
    // ...
  });
});
```

Step information is exposed in reporters API.

🌎 Launch web server before running tests

To launch a server during the tests, use the `webServer` option in the configuration file. The server will wait for a given url to be available before running the tests, and the url will be passed over to Playwright as a `baseURL` when creating a context.

playwright.config.ts

```
import { defineConfig } from '@playwright/test';
export default defineConfig({
  webServer: {
    command: 'npm run start', // command to launch
    url: 'http://127.0.0.1:3000', // url to await for
    timeout: 120 * 1000,
    reuseExistingServer: !process.env.CI,
  },
});
```

Learn more in the documentation.

Browser Versions

- Chromium 94.0.4595.0
- Mozilla Firefox 91.0
- WebKit 15.0

Version 1.13

Playwright Test

- ⚡ **Introducing Reporter API** which is already used to create an Allure Playwright reporter.
- ⛵ **New `baseURL` fixture** to support relative paths in tests.

Playwright

- 🖐 **Programmatic drag-and-drop support** via the page.dragAndDrop() API.
- 🔎 **Enhanced HAR** with body sizes for requests and responses. Use via `recordHar` option in browser.newContext().

Tools

- Playwright Trace Viewer now shows parameters, returned values and `console.log()` calls.
- Playwright Inspector can generate Playwright Test tests.

New and Overhauled Guides

- Intro
- Authentication
- Chrome Extensions
- Playwright Test Annotations
- Playwright Test Configuration
- Playwright Test Fixtures

Browser Versions

- Chromium 93.0.4576.0
- Mozilla Firefox 90.0
- WebKit 14.2

New Playwright APIs

- new `baseURL` option in browser.newContext() and browser.newPage()
- response.securityDetails() and response.serverAddr()
- page.dragAndDrop() and frame.dragAndDrop()
- download.cancel()
- page.inputValue(), frame.inputValue() and elementHandle.inputValue()
- new `force` option in page.fill(), frame.fill(), and elementHandle.fill()
- new `force` option in page.selectOption(), frame.selectOption(), and elementHandle.selectOption()

Version 1.12

⚡ Introducing Playwright Test

Playwright Test is a **new test runner** built from scratch by Playwright team specifically to accommodate end-to-end testing needs:

- Run tests across all browsers.
- Execute tests in parallel.
- Enjoy context isolation and sensible defaults out of the box.
- Capture videos, screenshots and other artifacts on failure.
- Integrate your POMs as extensible fixtures.

Installation:

```
npm i -D @playwright/test
```

Simple test `tests/foo.spec.ts`:

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

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  const name = await page.innerText('.navbar__title');
  expect(name).toBe('Playwright');
});
```

Running:

```
npx playwright test
```

👉 Read more in Playwright Test documentation.

🥷 Introducing Playwright Trace Viewer

Playwright Trace Viewer is a new GUI tool that helps exploring recorded Playwright traces after the script ran. Playwright traces let you examine:

- page DOM before and after each Playwright action
- page rendering before and after each Playwright action
- browser network during script execution

Traces are recorded using the new browserContext.tracing API:

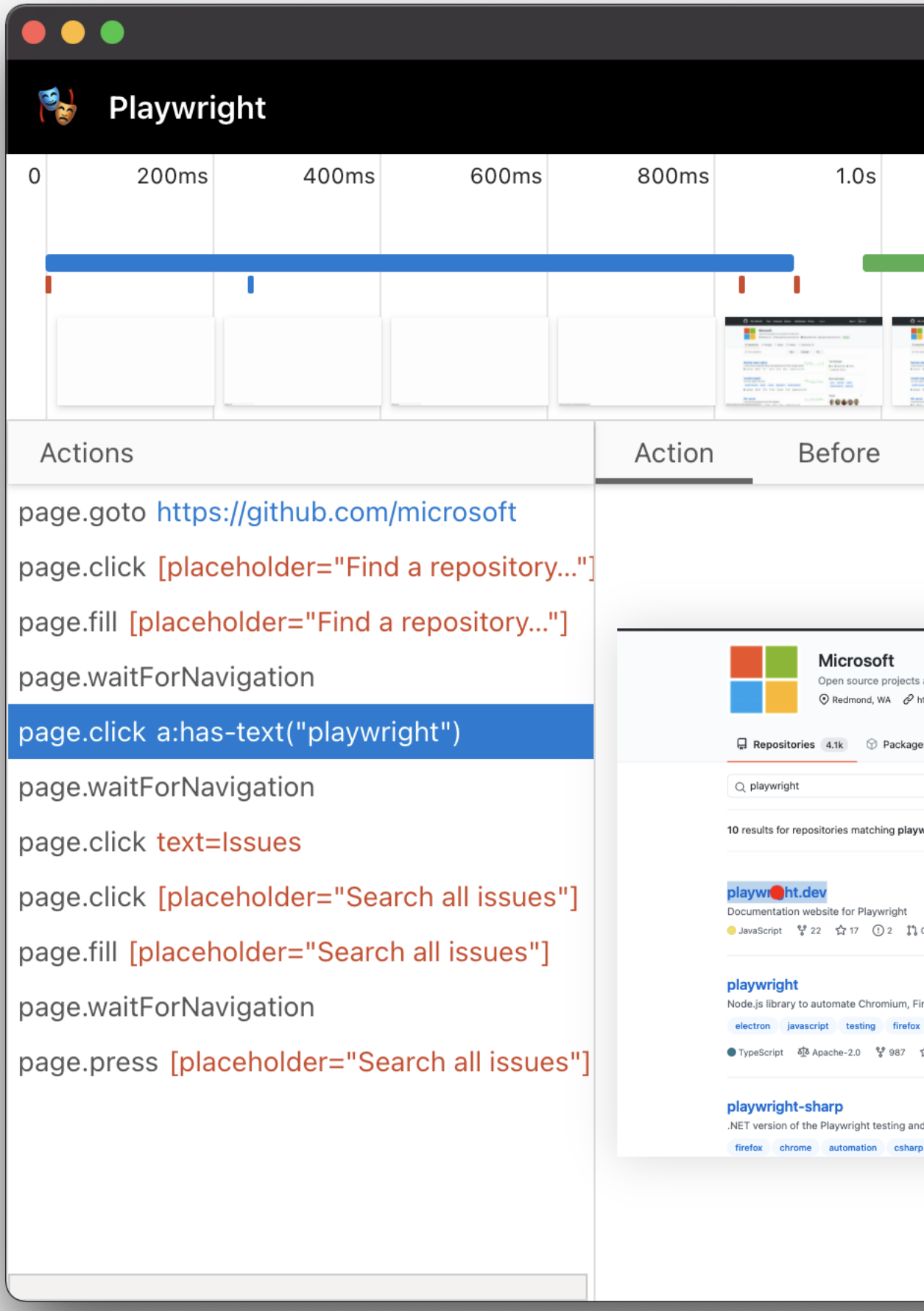
```
const browser = await chromium.launch();
const context = await browser.newContext();

// Start tracing before creating / navigating a page.
await context.tracing.start({ screenshots: true, snapshots: true });

const page = await context.newPage();
await page.goto('https://playwright.dev');
```

```
// Stop tracing and export it into a zip archive.
await context.tracing.stop({ path: 'trace.zip' });
```

Traces are examined later with the Playwright CLI:

```
npx playwright show-trace trace.zip
```

That will open the following GUI:


Playwright
0
200ms
400ms
600ms
800ms
1.0s
Actions
Action
Before
page.goto https://github.com/microsoft
page.click [placeholder="Find a repository..."]
page.fill [placeholder="Find a repository..."]
page.waitForNavigation
page.click a:has-text("playwright")
page.waitForNavigation
page.click text=Issues
page.click [placeholder="Search all issues"]
page.fill [placeholder="Search all issues"]
page.waitForNavigation
page.press [placeholder="Search all issues"]
Microsoft
Repositories
playwright
10 results for repositories matching
playwright.dev
Documentation website for Playwright
JavaScript
playwright
Node.js library to automate Chromium, Fir
electron
javascript
testing
firefox
TypeScript
Apache-2.0
playwright-sharp
.NET version of the Playwright testing and
firefox
chrome
automation

👉 Read more in trace viewer documentation.

Browser Versions

- Chromium 93.0.4530.0
- Mozilla Firefox 89.0
- WebKit 14.2

This version of Playwright was also tested against the following stable channels:

- Google Chrome 91
- Microsoft Edge 91

New APIs

- `reducedMotion` option in page.emulateMedia(), browserType.launchPersistentContext(), browser.newContext() and browser.newPage()
- browserContext.on('request')
- browserContext.on('requestfailed')
- browserContext.on('requestfinished')
- browserContext.on('response')
- `tracesDir` option in browserType.launch() and browserType.launchPersistentContext()
- new browserContext.tracing API namespace
- new download.page() method

Version 1.11

📹 New video: Playwright: A New Test Automation Framework for the Modern Web (slides)

- We talked about Playwright
- Showed engineering work behind the scenes
- Did live demos with new features ✨
- **Special thanks** to applitools for hosting the event and inviting us!

Browser Versions

- Chromium 92.0.4498.0
- Mozilla Firefox 89.0b6
- WebKit 14.2

New APIs

- support for **async predicates** across the API in methods such as page.waitForRequest() and others
- new **emulation devices**: Galaxy S8, Galaxy S9+, Galaxy Tab S4, Pixel 3, Pixel 4
- new methods:
  - page.waitForURL() to await navigations to URL

    - video.delete() and video.saveAs() to manage screen recording
- new options:
    - `screen` option in the browser.newContext() method to emulate `window.screen` dimensions
    - `position` option in page.check() and page.uncheck() methods
    - `trial` option to dry-run actions in page.check(), page.uncheck(), page.click(), page.dblclick(), page.hover() and page.tap()

Version 1.10

- Playwright for Java v1.10 is **now stable**!
- Run Playwright against **Google Chrome** and **Microsoft Edge** stable channels with the new channels API.
- Chromium screenshots are **fast** on Mac & Windows.

Bundled Browser Versions

- Chromium 90.0.4430.0
- Mozilla Firefox 87.0b10
- WebKit 14.2

This version of Playwright was also tested against the following stable channels:

- Google Chrome 89
- Microsoft Edge 89

New APIs

- browserType.launch() now accepts the new `'channel'` option. Read more in our documentation.

Version 1.9

- Playwright Inspector is a **new GUI tool** to author and debug your tests.
    - **Line-by-line debugging** of your Playwright scripts, with play, pause and step-through.
    - Author new scripts by **recording user actions**.
    - **Generate element selectors** for your script by hovering over elements.
    - Set the `PWDEBUG=1` environment variable to launch the Inspector
- **Pause script execution** with page.pause() in headed mode. Pausing the page launches Playwright Inspector for debugging.
- **New has-text pseudo-class** for CSS selectors. `:has-text("example")` matches any element containing `"example"` somewhere inside, possibly in a child or a descendant element. See more examples.
- **Page dialogs are now auto-dismissed** during execution, unless a listener for `dialog` event is configured. Learn more about this.
- Playwright for Python is **now stable** with an idiomatic snake case API and pre-built Docker image to run tests in CI/CD.

Browser Versions

- Chromium 90.0.4421.0
- Mozilla Firefox 86.0b10
- WebKit 14.1

New APIs

- page.pause().

Version 1.8

- Selecting elements based on layout with `:left-of()`, `:right-of()`, `:above()` and `:below()`.
- Playwright now includes command line interface, former playwright-cli.

```
npx playwright --help
```

- page.selectOption() now waits for the options to be present.
- New methods to assert element state like page.isEditable().

New APIs

- elementHandle.isChecked().
- elementHandle.isDisabled().
- elementHandle.isEditable().
- elementHandle.isEnabled().
- elementHandle.isHidden().
- elementHandle.isVisible().
- page.isChecked().
- page.isDisabled().
- page.isEditable().
- page.isEnabled().
- page.isHidden().
- page.isVisible().
- New option `'editable'` in elementHandle.waitForElementState().

Browser Versions

- Chromium 90.0.4392.0
- Mozilla Firefox 85.0b5
- WebKit 14.1

Version 1.7

- **New Java SDK**: Playwright for Java is now on par with JavaScript, Python and .NET bindings.
- **Browser storage API**: New convenience APIs to save and load browser storage state (cookies, local storage) to simplify automation scenarios with authentication.

- **New CSS selectors**: We heard your feedback for more flexible selectors and have revamped the selectors implementation. Playwright 1.7 introduces new CSS extensions and there's more coming soon.
- **New website**: The docs website at playwright.dev has been updated and is now built with Docusaurus.
- **Support for Apple Silicon**: Playwright browser binaries for WebKit and Chromium are now built for Apple Silicon.

New APIs

- browserContext.storageState() to get current state for later reuse.
- `storageState` option in browser.newContext() and browser.newPage() to setup browser context state.

Browser Versions

- Chromium 89.0.4344.0
- Mozilla Firefox 84.0b9
- WebKit 14.1

Previous
Getting started - VS Code

Next
Canary releases

- Version 1.47
- Version 1.46
- Version 1.45
- Version 1.44
- Version 1.43
- Version 1.42
- Version 1.41
- Version 1.40
- Version 1.39
- Version 1.38
- Version 1.37
- Version 1.36
- Version 1.35
- Version 1.34
- Version 1.33
- Version 1.32
- Version 1.31
- Version 1.30
- Version 1.29
- Version 1.28
- Version 1.27
- Version 1.26

- Version 1.25
- Version 1.24
- Version 1.23
- Version 1.22
- Version 1.21
- Version 1.20
- Version 1.19
- Version 1.18
- Version 1.17
- Version 1.16
- Version 1.15
- Version 1.14
- Version 1.13
- Version 1.12
- Version 1.11
- Version 1.10
- Version 1.9
- Version 1.8
- Version 1.7

Learn

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

Community

- Stack Overflow
- Discord
- Twitter
- LinkedIn

More

- GitHub
- YouTube
- Blog
- Ambassadors