Release notes

Version 1.39

Evergreen browsers update.

Browser Versions

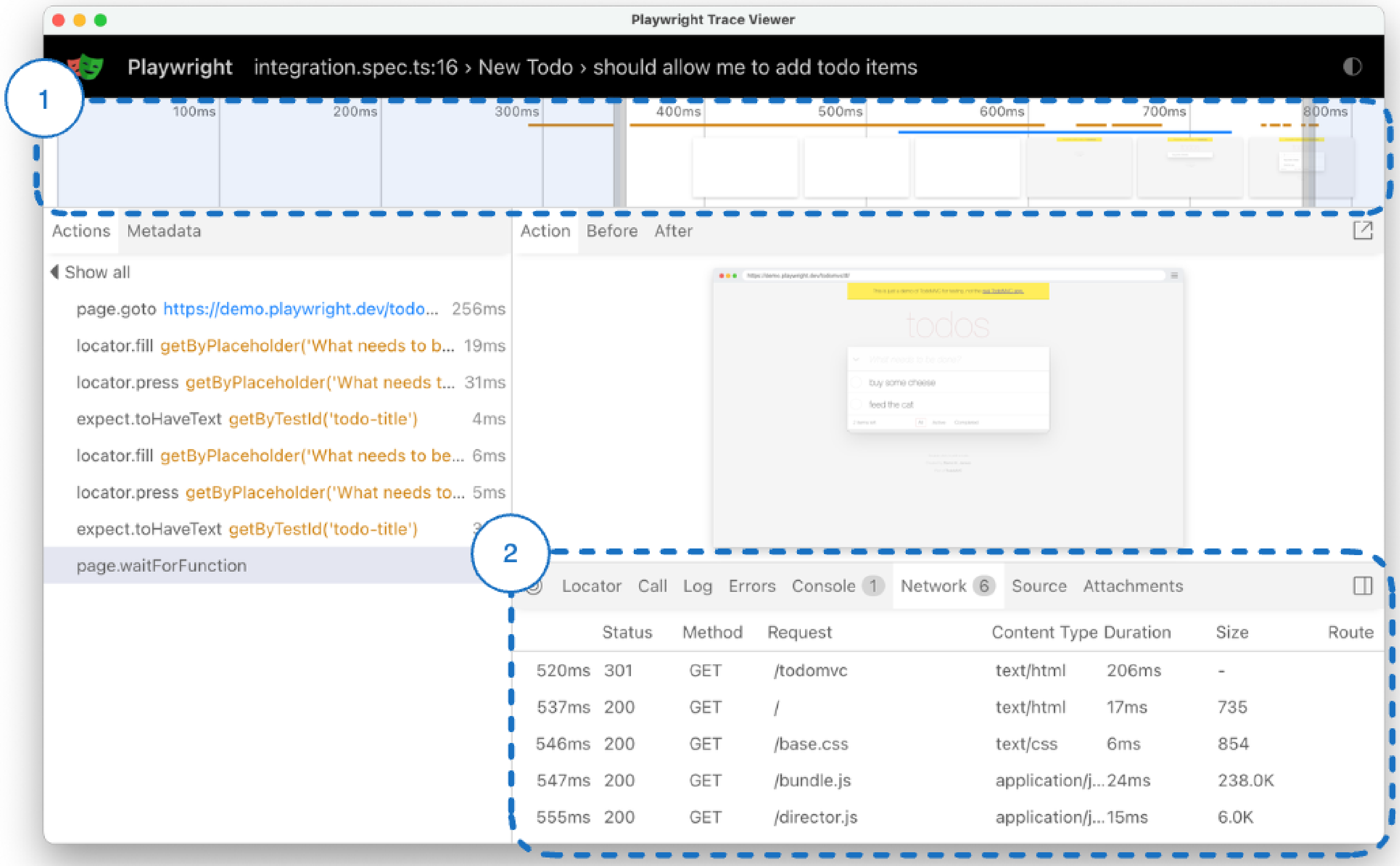
- Chromium 119.0.6045.9
- Mozilla Firefox 118.0.1
- WebKit 17.4

This version was also tested against the following stable channels:

- Google Chrome 118
- Microsoft Edge 118

Version 1.38

Trace Viewer Updates



```
page.goto https://demo.playwright.dev/todo...  256ms
locator.fill getByPlaceholder('What needs to b...  19ms
locator.press getByPlaceholder('What needs t...  31ms
expect.toHaveText getByTestId('todo-title')  4ms
locator.fill getByPlaceholder('What needs to be...  6ms
locator.press getByPlaceholder('What needs to...  5ms
expect.toHaveText getByTestId('todo-title')  3
page.waitForFunction
```

	Status	Method	Request	Content Type	Duration	Size	Route
520ms	301	GET	/todomvc	text/html	206ms	-	
537ms	200	GET	/	text/html	17ms	735	
546ms	200	GET	/base.css	text/css	6ms	854	
547ms	200	GET	/bundle.js	application/j...	24ms	238.0K	
555ms	200	GET	/director.js	application/j...	15ms	6.0K	

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

New APIs

- browser_context.on("weberror")
- locator.press_sequentially()

Deprecations

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

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

Highlights

- New --full-page-screenshot command line flag allows taking a full page screenshot on failure.
- It is now possible to override the context options for a single test by using the browser_context_args marker.
- `pytest-playwright` is now also getting published on Anaconda

📚 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	✅	✅	✅	✅

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

- New option `mask_color` for methods page.screenshot() and locator.screenshot() to change default masking color.

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

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

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

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

  ```
  button = page.get_by_role("button").and_(page.get_by_title("Subscribe"))
  ```

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

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:

```python
new_email = page.get_by_role("button", name="New email")
dialog = page.get_by_text("Confirm security settings")
expect(new_email.or_(dialog)).is_visible()
if (dialog.is_visible()):
  page.get_by_role("button", name="Dismiss").click()
new_email.click()
```

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

```python
row_locator = page.locator("tr")
row_locator.filter(has_not_text="text in column 1").filter(
    has_not=page.get_by_role("button", name="column 2 button")
).screenshot()
```

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

New APIs

- locator.or_()
- New option `has_not` in locator.filter()
- New option `has_not_text` in locator.filter()
- expect(locator).to_be_attached()
- New option `timeout` in route.fetch()

⚠ Breaking change

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

New APIs

- Custom expect message, see test assertions documentation.
- New options `update_mode` and `update_content` in page.route_from_har() and browser_context.route_from_har().
- Chaining existing locator objects, see locator docs for details.
- New option `name` in method tracing.start_chunk().

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

```
from playwright.sync_api import expect

locator = page.get_by_role("button")
```

```python
# Make sure at least some part of element intersects viewport.
expect(locator).to_be_in_viewport()

# Make sure element is fully outside of viewport.
expect(locator).not_to_be_in_viewport()

# Make sure that at least half of the element intersects viewport.
expect(locator).to_be_in_viewport(ratio=0.5)
```

Miscellaneous

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

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():

  ```python
  def handle_route(route: Route):
    # Fetch original settings.
    response = route.fetch()

    # Force settings theme to a predefined value.
    json = response.json()
    json["theme"] = "Solorized"

    # Fulfill with modified data.
    route.fulfill(json=json)


  page.route("**/api/settings", handle_route)
  ```

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

  ```python
  # Check all checkboxes!
  checkboxes = page.get_by_role("checkbox")
  for checkbox in checkboxes.all():
    checkbox.check()
  ```

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

  ```html
  <select multiple>
    <option value="red">Red</div>
    <option value="green">Green</div>
    <option value="blue">Blue</div>
  </select>
  ```

  ```python
  element.select_option("Red")
  ```

Miscellaneous

- 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

- **Live Locators in CodeGen.** Generate a locator for any element on the page using "Explore" tool.

New APIs

- locator.blur()
- locator.clear()

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.get_by_text() to locate by text content.
- page.get_by_role() to locate by ARIA role, ARIA attributes and accessible name.
- page.get_by_label() to locate a form control by associated label's text.
- page.get_by_test_id() to locate an element based on its `data-testid` attribute (other attribute can be configured).
- page.get_by_placeholder() to locate an input by placeholder.
- page.get_by_alt_text() to locate an element, usually image, by its text alternative.
- page.get_by_title() to locate an element by its title.

```
page.get_by_label("User Name").fill("John")

page.get_by_label("Password").fill("secret-password")

page.get_by_role("button", name="Sign in").click()

expect(page.get_by_text("Welcome, John!")).to_be_visible()
```

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

Other highlights

- 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).to_have_attribute() 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.

  ```
  expect(page.get_by_role("button")).to_have_attribute("disabled", "")
  ```

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).to_be_enabled().
- expect(locator).to_have_text() now pierces open shadow roots.
- New option `editable` for expect(locator).to_be_editable().
- New option `visible` for expect(locator).to_be_visible().

Other highlights

- New option `max_redirects` for api_request_context.get() and others to limit redirect count.
- Python 3.11 is now supported.

Behavior Change

A bunch of Playwright APIs already support the `wait_until: "domcontentloaded"` option. For example:

```
page.goto("https://playwright.dev", wait_until="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 `wait_until="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

Announcements

- 🎁 We now ship Ubuntu 22.04 Jammy Jellyfish docker image: `mcr.microsoft.com/playwright/python:v1.34.0-jammy`.
- 🪦 This is the last release with macOS 10.15 support (deprecated as of 1.21).
- ⚠ 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

🐂 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	✅	✅	✅	

New introduction docs

We rewrote our Getting Started docs to be more end-to-end testing focused. Check them out on playwright.dev.

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:

```
context = browser.new_context(record_har_path="github.har.zip")
# ... do stuff ...
context.close()
```

```
context = await browser.new_context(record_har_path="github.har.zip")
# ... do stuff ...
await context.close()
```

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

```
context.route_from_har("github.har.zip")
```

```
await context.route_from_har("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
def remove_header_handler(route: Route) -> None:
    headers = route.request.all_headers()
    if "if-none-match" in headers:
        del headers["if-none-match"]
    route.fallback(headers=headers)

page.route("**/*", remove_header_handler)
```

```
# Abort all images
def abort_images_handler(route: Route) -> None:
    if route.request.resource_type == "image":
        route.abort()
    else:
        route.fallback()

page.route("**/*", abort_images_handler)
```

```
# Remove a header from all requests
async def remove_header_handler(route: Route) -> None:
    headers = await route.request.all_headers()
    if "if-none-match" in headers:
        del headers["if-none-match"]
    await route.fallback(headers=headers)

await page.route("**/*", remove_header_handler)

# Abort all images
async def abort_images_handler(route: Route) -> None:
    if route.request.resource_type == "image":
        await route.abort()
    else:
        await route.fallback()

await page.route("**/*", abort_images_handler)
```

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

Web-First Assertions Update

- New method expect(locator).to_have_values() that asserts all selected values of `<select multiple>` element.
- Methods expect(locator).to_contain_text() and expect(locator).to_have_text() now accept `ignore_case` option.

Miscellaneous

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

  ```
  context = browser.new_context(service_workers="block")
  ```

```
page = context.new_page()
```

```
context = await browser.new_context(service_workers="block")
page = await context.new_page()
```

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

  ```
  context = browser.new_context(record_har_path="github.har.zip")
  ```

  ```
  context = await browser.new_context(record_har_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:

  ```
  context = browser.new_context(record_har_mode="minimal",
  record_har_path="har.har")
  ```

  ```
  context = await browser.new_context(record_har_mode="minimal",
  record_har_path="har.har")
  ```

- Playwright now runs on Ubuntu 22 amd64 and Ubuntu 22 arm64.

Version 1.22

Highlights

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

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

  Read more in our documentation.

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

```
buttons = page.locator("role=button")
# ...
submit_button = buttons.filter(has_text="Submit")
submit_button.click()
```

- Codegen now supports generating Pytest Tests

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"
  page.locator("role=button[name='log in']").click()
  ```

  ```
  # 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"`.

Behavior Changes

- 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.set_input_files() 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 element_handle.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.
- 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

Highlights

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

  ```
  page.locator("article", has=page.locator(".highlight")).click()
  ```

  ```
  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

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

API Testing

Playwright for Python 1.18 introduces new API Testing that lets you send requests to the server directly from Python! 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**:

```
# Do a GET request on behalf of page
res = page.request.get("http://example.com/foo.json")
```

```
# Do a GET request on behalf of page
res = await page.request.get("http://example.com/foo.json")
```

Read more in our documentation.

Web-First Assertions

Playwright for Python 1.18 introduces Web-First Assertions.

Consider the following example:

```
from playwright.sync_api import Page, expect

def test_status_becomes_submitted(page: Page) -> None:
    # ..
    page.locator("#submit-button").click()
    expect(page.locator(".status")).to_have_text("Submitted")
```

```
from playwright.async_api import Page, expect

async def test_status_becomes_submitted(page: Page) -> None:
    # ..
    await page.locator("#submit-button").click()
    await expect(page.locator(".status")).to_have_text("Submitted")
```

Playwright 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 pass this timeout as an option.

Read more in our documentation.

Locator Improvements

- locator.drag_to()
- Each locator can now be optionally filtered by the text it contains:

  ```
  page.locator("li", has_text="my item").locator("button").click()
  ```

  ```
  await page.locator("li", has_text="my item").locator("button").click()
  ```

  Read more in locator documentation

New APIs & changes

- `accept_downloads` option now defaults to `True`.
- `sources` option to embed sources into traces.

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.frame_locator() or locator.frame_locator() method.

```
locator = page.frame_locator("my-frame").locator("text=Submit")
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

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.

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

Version 1.16

🎭 Playwright Library

`locator.wait_for`

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:

```
order_sent = page.locator("#order-sent")
order_sent.wait_for()
```

Read more about locator.wait_for().

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

- run trace viewer with `npx playwright show-trace` and drop trace files to the trace viewer PWA
- 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

🖱️ Mouse Wheel

By using `Page.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.all_headers()
- Request.headers_array()
- Request.header_value(name: str)
- Response.all_headers()
- Response.headers_array()
- Response.header_value(name: str)
- Response.header_values(name: str)

🌈 Forced-Colors emulation

Its now possible to emulate the `forced-colors` CSS media feature by passing it in the context options or calling Page.emulate_media().

New APIs

- Page.route() accepts new `times` option to specify how many times this route should be matched.
- Page.set_checked(selector: str, checked: bool) and Locator.set_checked(selector: str, checked: bool) was introduced to set the checked state of a checkbox.
- Request.sizes() Returns resource size information for given http request.
- BrowserContext.tracing.start_chunk() - Start a new trace chunk.
- BrowserContext.tracing.stop_chunk() - Stops a new trace chunk.

Browser Versions

- Chromium 96.0.4641.0
- Mozilla Firefox 92.0
- WebKit 15.0

Version 1.14

⚡ 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!
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**!

```
locator = page.locator("button")
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.

```
page.locator("_react=SubmitButton[enabled=true]").click()
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
button.click("button >> nth=0")
# or if you are using locators, you can use first, nth() and last
page.locator("button").first.click()

# click a visible button
button.click("button >> visible=true")
```

Browser Versions

- Chromium 94.0.4595.0
- Mozilla Firefox 91.0
- WebKit 15.0

Version 1.13

Playwright

- 🖖 **Programmatic drag-and-drop support** via the page.drag_and_drop() API.

- 🔎 **Enhanced HAR** with body sizes for requests and responses. Use via `recordHar` option in browser.new_context().

Tools

- Playwright Trace Viewer now shows parameters, returned values and `console.log()` calls.

New and Overhauled Guides

- Intro
- Authentication
- Chrome Extensions

Browser Versions

- Chromium 93.0.4576.0
- Mozilla Firefox 90.0
- WebKit 14.2

New Playwright APIs

- new `baseURL` option in browser.new_context() and browser.new_page()
- response.security_details() and response.server_addr()
- page.drag_and_drop() and frame.drag_and_drop()
- download.cancel()
- page.input_value(), frame.input_value() and element_handle.input_value()
- new `force` option in page.fill(), frame.fill(), and element_handle.fill()
- new `force` option in page.select_option(), frame.select_option(), and element_handle.select_option()

Version 1.12

🧟‍♂️ 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 browser_context.tracing API:

```
browser = chromium.launch()
context = browser.new_context()

# Start tracing before creating / navigating a page.
context.tracing.start(screenshots=True, snapshots=True)

page.goto("https://playwright.dev")

# Stop tracing and export it into a zip archive.
context.tracing.stop(path = "trace.zip")
```

Traces are examined later with the Playwright CLI:

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

That will open the following GUI:

👉 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.emulate_media(), browser_type.launch_persistent_context(), browser.new_context() and browser.new_page()
- browser_context.on("request")
- browser_context.on("requestfailed")
- browser_context.on("requestfinished")

- browser_context.on("response")
- `tracesDir` option in browser_type.launch() and browser_type.launch_persistent_context()
- new browser_context.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.expect_request() and others
- new **emulation devices**: Galaxy S8, Galaxy S9+, Galaxy Tab S4, Pixel 3, Pixel 4
- new methods:
  - page.wait_for_url() to await navigations to URL
  - video.delete() and video.save_as() to manage screen recording
- new options:
  - `screen` option in the browser.new_context() 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.

```
playwright --help
```

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

New APIs

- element_handle.is_checked().
- element_handle.is_disabled().
- element_handle.is_editable().
- element_handle.is_enabled().
- element_handle.is_hidden().
- element_handle.is_visible().
- page.is_checked().
- page.is_disabled().
- page.is_editable().
- page.is_enabled().
- page.is_hidden().
- page.is_visible().
- New option `'editable'` in element_handle.wait_for_element_state().

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

- browser_context.storage_state() to get current state for later reuse.
- `storageState` option in browser.new_context() and browser.new_page() to setup browser context state.

Browser Versions

- Chromium 89.0.4344.0
- Mozilla Firefox 84.0b9
- WebKit 14.1