Browser

- extends: EventEmitter

A Browser is created via browser_type.launch(). An example of using a Browser to create a Page:

Sync **Async**

```
from playwright.sync_api import sync_playwright, Playwright

def run(playwright: Playwright):
    firefox = playwright.firefox
    browser = firefox.launch()
    page = browser.new_page()
    page.goto("https://example.com")
    browser.close()

with sync_playwright() as playwright:
    run(playwright)
```

Methods

close

Added in: v1.8

In case this browser is obtained using browser_type.launch(), closes the browser and all of its pages (if any were opened).

In case this browser is connected to, clears all created contexts belonging to this browser and disconnects from the browser server.

> **NOTE**
>
> This is similar to force quitting the browser. Therefore, you should call browser_context.close() on any BrowserContext's you explicitly created earlier with

browser.new_context() **before** calling browser.close().

The Browser object itself is considered to be disposed and cannot be used anymore.

Usage

```
browser.close()
```

new_browser_cdp_session

Added in: v1.11

> **NOTE**
>
> CDP Sessions are only supported on Chromium-based browsers.

Returns the newly created browser session.

Usage

```
browser.new_browser_cdp_session()
```

Returns

- CDPSession

new_context

Added in: v1.8

Creates a new browser context. It won't share cookies/cache with other browser contexts.

> **NOTE**
>
> If directly using this method to create BrowserContexts, it is best practice to explicitly close the returned context via browser_context.close() when your code is done with the BrowserContext, and before calling browser.close(). This will ensure the `context` is closed gracefully and any artifacts—like HARs and videos—are fully flushed and saved.

Usage

Sync | Async

```
browser = playwright.firefox.launch() # or "chromium" or "webkit".
# create a new incognito browser context.
context = browser.new_context()
# create a new page in a pristine context.
page = context.new_page()
page.goto("https://example.com")

# gracefully close up everything
context.close()
browser.close()
```

Arguments

- `accept_downloads` bool *(optional)*

  Whether to automatically download all the attachments. Defaults to `true` where all the downloads are accepted.

- `base_url` str *(optional)*

  When using page.goto(), page.route(), page.wait_for_url(), page.expect_request(), or page.expect_response() it takes the base URL in consideration by using the `URL()` constructor for building the corresponding URL. Unset by default. Examples:

  - baseURL: `http://localhost:3000` and navigating to `/bar.html` results in `http://localhost:3000/bar.html`
  - baseURL: `http://localhost:3000/foo/` and navigating to `./bar.html` results in `http://localhost:3000/foo/bar.html`
  - baseURL: `http://localhost:3000/foo` (without trailing slash) and navigating to `./bar.html` results in `http://localhost:3000/bar.html`

- `bypass_csp` bool *(optional)*

  Toggles bypassing page's Content-Security-Policy. Defaults to `false`.

- `color_scheme` "light"|"dark"|"no-preference"|"null" *(optional)*

Emulates `'prefers-colors-scheme'` media feature, supported values are `'light'`, `'dark'`, `'no-preference'`. See page.emulate_media() for more details. Passing `'null'` resets emulation to system defaults. Defaults to `'light'`.

- `device_scale_factor` float *(optional)*

  Specify device scale factor (can be thought of as dpr). Defaults to `1`. Learn more about emulating devices with device scale factor.

- `extra_http_headers` Dict[str, str] *(optional)*

  An object containing additional HTTP headers to be sent with every request. Defaults to none.

- `forced_colors` "active"|"none"|"null" *(optional)*

  Emulates `'forced-colors'` media feature, supported values are `'active'`, `'none'`. See page.emulate_media() for more details. Passing `'null'` resets emulation to system defaults. Defaults to `'none'`.

- `geolocation` Dict *(optional)*

  - `latitude` float

    Latitude between -90 and 90.

  - `longitude` float

    Longitude between -180 and 180.

  - `accuracy` float *(optional)*

    Non-negative accuracy value. Defaults to `0`.

- `has_touch` bool *(optional)*

  Specifies if viewport supports touch events. Defaults to false. Learn more about mobile emulation.

- `http_credentials` Dict *(optional)*

  - `username` str

  - `password` str

- `origin` str *(optional)*

  Restrain sending http credentials on specific origin (scheme://host:port).

  Credentials for HTTP authentication. If no origin is specified, the username and password are sent to any servers upon unauthorized responses.

- `ignore_https_errors` bool *(optional)*

  Whether to ignore HTTPS errors when sending network requests. Defaults to `false`.

- `is_mobile` bool *(optional)*

  Whether the `meta viewport` tag is taken into account and touch events are enabled. isMobile is a part of device, so you don't actually need to set it manually. Defaults to `false` and is not supported in Firefox. Learn more about mobile emulation.

- `java_script_enabled` bool *(optional)*

  Whether or not to enable JavaScript in the context. Defaults to `true`. Learn more about disabling JavaScript.

- `locale` str *(optional)*

  Specify user locale, for example `en-GB`, `de-DE`, etc. Locale will affect `navigator.language` value, `Accept-Language` request header value as well as number and date formatting rules. Defaults to the system default locale. Learn more about emulation in our emulation guide.

- `no_viewport` bool *(optional)*

  Does not enforce fixed viewport, allows resizing window in the headed mode.

- `offline` bool *(optional)*

  Whether to emulate network being offline. Defaults to `false`. Learn more about network emulation.

- `permissions` List[str] *(optional)*

  A list of permissions to grant to all pages in this context. See browser_context.grant_permissions() for more details. Defaults to none.

- `proxy` Dict *(optional)*

- `server` str

  Proxy to be used for all requests. HTTP and SOCKS proxies are supported, for example `http://myproxy.com:3128` or `socks5://myproxy.com:3128`. Short form `myproxy.com:3128` is considered an HTTP proxy.

- `bypass` str *(optional)*

  Optional comma-separated domains to bypass proxy, for example `".com, chromium.org, .domain.com"`.

- `username` str *(optional)*

  Optional username to use if HTTP proxy requires authentication.

- `password` str *(optional)*

  Optional password to use if HTTP proxy requires authentication.

Network proxy settings to use with this context. Defaults to none.

> **NOTE**
> For Chromium on Windows the browser needs to be launched with the global proxy for this option to work. If all contexts override the proxy, global proxy will be never used and can be any string, for example `launch({ proxy: { server: 'http://per-context' } })`.

- `record_har_content` "omit"|"embed"|"attach" *(optional)*

  Optional setting to control resource content management. If `omit` is specified, content is not persisted. If `attach` is specified, resources are persisted as separate files and all of these files are archived along with the HAR file. Defaults to `embed`, which stores content inline the HAR file as per HAR specification.

- `record_har_mode` "full"|"minimal" *(optional)*

  When set to `minimal`, only record information necessary for routing from HAR. This omits sizes, timing, page, cookies, security and other types of HAR information that are not used when replaying from HAR. Defaults to `full`.

- `record_har_omit_content` bool *(optional)*

Optional setting to control whether to omit request content from the HAR. Defaults to `false`.

- `record_har_path` Union[str, pathlib.Path] *(optional)*

  Enables HAR recording for all pages into the specified HAR file on the filesystem. If not specified, the HAR is not recorded. Make sure to call browser_context.close() for the HAR to be saved.

- `record_har_url_filter` str|Pattern *(optional)*

- `record_video_dir` Union[str, pathlib.Path] *(optional)*

  Enables video recording for all pages into the specified directory. If not specified videos are not recorded. Make sure to call browser_context.close() for videos to be saved.

- `record_video_size` Dict *(optional)*

  - `width` int

    Video frame width.

  - `height` int

    Video frame height.

  Dimensions of the recorded videos. If not specified the size will be equal to `viewport` scaled down to fit into 800x800. If `viewport` is not configured explicitly the video size defaults to 800x450. Actual picture of each page will be scaled down if necessary to fit the specified size.

- `reduced_motion` "reduce"|"no-preference"|"null" *(optional)*

  Emulates `'prefers-reduced-motion'` media feature, supported values are `'reduce'`, `'no-preference'`. See page.emulate_media() for more details. Passing `'null'` resets emulation to system defaults. Defaults to `'no-preference'`.

- `screen` Dict *(optional)*

  - `width` int

    page width in pixels.

  - `height` int

page height in pixels.

Emulates consistent window screen size available inside web page via `window.screen`. Is only used when the `viewport` is set.

- `service_workers` "allow"|"block" *(optional)*

  Whether to allow sites to register Service workers. Defaults to `'allow'`.

  - `'allow'`: Service Workers can be registered.
  - `'block'`: Playwright will block all registration of Service Workers.

- `storage_state` Union[str, pathlib.Path]|Dict *(optional)*

  - `cookies` List[Dict]

    - `name` str
    - `value` str
    - `domain` str

      Domain and path are required. For the cookie to apply to all subdomains as well, prefix domain with a dot, like this: ".example.com"

    - `path` str

      Domain and path are required

    - `expires` float

      Unix time in seconds.

    - `httpOnly` bool
    - `secure` bool
    - `sameSite` "Strict"|"Lax"|"None"

      sameSite flag

    Cookies to set for context

  - `origins` List[Dict]

    - `origin` str
    - `localStorage` List[Dict]
      - `name` str
      - `value` str

    localStorage to set for context

  Learn more about storage state and auth.

  Populates context with given storage state. This option can be used to initialize context with logged-in information obtained via browser_context.storage_state().

- `strict_selectors` bool *(optional)*

  If set to true, enables strict selectors mode for this context. In the strict selectors mode all operations on selectors that imply single target DOM element will throw when more than one element matches the selector. This option does not affect any Locator APIs (Locators are always strict). Defaults to `false`. See Locator to learn more about the strict mode.

- `timezone_id` str *(optional)*

  Changes the timezone of the context. See ICU's metaZones.txt for a list of supported timezone IDs. Defaults to the system timezone.

- `user_agent` str *(optional)*

  Specific user agent to use in this context.

- `viewport` NoneType|Dict *(optional)*

    - `width` int

      page width in pixels.

    - `height` int

      page height in pixels.

  Sets a consistent viewport for each page. Defaults to an 1280x720 viewport. `no_viewport` disables the fixed viewport. Learn more about viewport emulation.

Returns

- BrowserContext

new_page

Added in: v1.8

Creates a new page in a new browser context. Closing this page will close the context as well.

This is a convenience API that should only be used for the single-page scenarios and short snippets. Production code and testing frameworks should explicitly create browser.new_context() followed by the browser_context.new_page() to control their exact life times.

Usage

```
browser.new_page()
browser.new_page(**kwargs)
```

Arguments

- `accept_downloads` bool *(optional)*

  Whether to automatically download all the attachments. Defaults to `true` where all the downloads are accepted.

- `base_url` str *(optional)*

  When using page.goto(), page.route(), page.wait_for_url(), page.expect_request(), or page.expect_response() it takes the base URL in consideration by using the `URL()` constructor for building the corresponding URL. Unset by default. Examples:

  - baseURL: `http://localhost:3000` and navigating to `/bar.html` results in `http://localhost:3000/bar.html`
  - baseURL: `http://localhost:3000/foo/` and navigating to `./bar.html` results in `http://localhost:3000/foo/bar.html`
  - baseURL: `http://localhost:3000/foo` (without trailing slash) and navigating to `./bar.html` results in `http://localhost:3000/bar.html`

- `bypass_csp` bool *(optional)*

  Toggles bypassing page's Content-Security-Policy. Defaults to `false`.

- `color_scheme` "light"|"dark"|"no-preference"|"null" *(optional)*

Emulates `'prefers-colors-scheme'` media feature, supported values are `'light'`, `'dark'`, `'no-preference'`. See page.emulate_media() for more details. Passing `'null'` resets emulation to system defaults. Defaults to `'light'`.

- `device_scale_factor` float *(optional)*

  Specify device scale factor (can be thought of as dpr). Defaults to `1`. Learn more about emulating devices with device scale factor.

- `extra_http_headers` Dict[str, str] *(optional)*

  An object containing additional HTTP headers to be sent with every request. Defaults to none.

- `forced_colors` "active"|"none"|"null" *(optional)*

  Emulates `'forced-colors'` media feature, supported values are `'active'`, `'none'`. See page.emulate_media() for more details. Passing `'null'` resets emulation to system defaults. Defaults to `'none'`.

- `geolocation` Dict *(optional)*

  - `latitude` float

    Latitude between -90 and 90.

  - `longitude` float

    Longitude between -180 and 180.

  - `accuracy` float *(optional)*

    Non-negative accuracy value. Defaults to `0`.

- `has_touch` bool *(optional)*

  Specifies if viewport supports touch events. Defaults to false. Learn more about mobile emulation.

- `http_credentials` Dict *(optional)*

  - `username` str

  - `password` str

- `origin` str *(optional)*

  Restrain sending http credentials on specific origin (scheme://host:port).

  Credentials for HTTP authentication. If no origin is specified, the username and password are sent to any servers upon unauthorized responses.

- `ignore_https_errors` bool *(optional)*

  Whether to ignore HTTPS errors when sending network requests. Defaults to `false`.

- `is_mobile` bool *(optional)*

  Whether the `meta viewport` tag is taken into account and touch events are enabled. isMobile is a part of device, so you don't actually need to set it manually. Defaults to `false` and is not supported in Firefox. Learn more about mobile emulation.

- `java_script_enabled` bool *(optional)*

  Whether or not to enable JavaScript in the context. Defaults to `true`. Learn more about disabling JavaScript.

- `locale` str *(optional)*

  Specify user locale, for example `en-GB`, `de-DE`, etc. Locale will affect `navigator.language` value, `Accept-Language` request header value as well as number and date formatting rules. Defaults to the system default locale. Learn more about emulation in our emulation guide.

- `no_viewport` bool *(optional)*

  Does not enforce fixed viewport, allows resizing window in the headed mode.

- `offline` bool *(optional)*

  Whether to emulate network being offline. Defaults to `false`. Learn more about network emulation.

- `permissions` List[str] *(optional)*

  A list of permissions to grant to all pages in this context. See browser_context.grant_permissions() for more details. Defaults to none.

- `proxy` Dict *(optional)*

- `server` str

  Proxy to be used for all requests. HTTP and SOCKS proxies are supported, for example `http://myproxy.com:3128` or `socks5://myproxy.com:3128`. Short form `myproxy.com:3128` is considered an HTTP proxy.

- `bypass` str *(optional)*

  Optional comma-separated domains to bypass proxy, for example `".com, chromium.org, .domain.com"`.

- `username` str *(optional)*

  Optional username to use if HTTP proxy requires authentication.

- `password` str *(optional)*

  Optional password to use if HTTP proxy requires authentication.

Network proxy settings to use with this context. Defaults to none.

> **NOTE**
> For Chromium on Windows the browser needs to be launched with the global proxy for this option to work. If all contexts override the proxy, global proxy will be never used and can be any string, for example `launch({ proxy: { server: 'http://per-context' } })`.

- `record_har_content` "omit"|"embed"|"attach" *(optional)*

  Optional setting to control resource content management. If `omit` is specified, content is not persisted. If `attach` is specified, resources are persisted as separate files and all of these files are archived along with the HAR file. Defaults to `embed`, which stores content inline the HAR file as per HAR specification.

- `record_har_mode` "full"|"minimal" *(optional)*

  When set to `minimal`, only record information necessary for routing from HAR. This omits sizes, timing, page, cookies, security and other types of HAR information that are not used when replaying from HAR. Defaults to `full`.

- `record_har_omit_content` bool *(optional)*

Optional setting to control whether to omit request content from the HAR. Defaults to `false`.

- `record_har_path` Union[str, pathlib.Path] *(optional)*

  Enables HAR recording for all pages into the specified HAR file on the filesystem. If not specified, the HAR is not recorded. Make sure to call browser_context.close() for the HAR to be saved.

- `record_har_url_filter` str|Pattern *(optional)*

- `record_video_dir` Union[str, pathlib.Path] *(optional)*

  Enables video recording for all pages into the specified directory. If not specified videos are not recorded. Make sure to call browser_context.close() for videos to be saved.

- `record_video_size` Dict *(optional)*

  - `width` int

    Video frame width.

  - `height` int

    Video frame height.

  Dimensions of the recorded videos. If not specified the size will be equal to `viewport` scaled down to fit into 800x800. If `viewport` is not configured explicitly the video size defaults to 800x450. Actual picture of each page will be scaled down if necessary to fit the specified size.

- `reduced_motion` "reduce"|"no-preference"|"null" *(optional)*

  Emulates `'prefers-reduced-motion'` media feature, supported values are `'reduce'`, `'no-preference'`. See page.emulate_media() for more details. Passing `'null'` resets emulation to system defaults. Defaults to `'no-preference'`.

- `screen` Dict *(optional)*

  - `width` int

    page width in pixels.

  - `height` int

page height in pixels.

Emulates consistent window screen size available inside web page via `window.screen`. Is only used when the `viewport` is set.

- `service_workers` "allow"|"block" *(optional)*

  Whether to allow sites to register Service workers. Defaults to `'allow'`.

  - `'allow'`: Service Workers can be registered.
  - `'block'`: Playwright will block all registration of Service Workers.

- `storage_state` Union[str, pathlib.Path]|Dict *(optional)*

  - `cookies` List[Dict]

    - `name` str
    - `value` str
    - `domain` str

      Domain and path are required. For the cookie to apply to all subdomains as well, prefix domain with a dot, like this: ".example.com"

    - `path` str

      Domain and path are required

    - `expires` float

      Unix time in seconds.

    - `httpOnly` bool
    - `secure` bool
    - `sameSite` "Strict"|"Lax"|"None"

      sameSite flag

    Cookies to set for context

  - `origins` List[Dict]

    - `origin` str
    - `localStorage` List[Dict]
      - `name` str
      - `value` str

    localStorage to set for context

  Learn more about storage state and auth.

  Populates context with given storage state. This option can be used to initialize context with logged-in information obtained via browser_context.storage_state().

- `strict_selectors` bool *(optional)*

  If set to true, enables strict selectors mode for this context. In the strict selectors mode all operations on selectors that imply single target DOM element will throw when more than one element matches the selector. This option does not affect any Locator APIs (Locators are always strict). Defaults to `false`. See Locator to learn more about the strict mode.

- `timezone_id` str *(optional)*

  Changes the timezone of the context. See ICU's metaZones.txt for a list of supported timezone IDs. Defaults to the system timezone.

- `user_agent` str *(optional)*

  Specific user agent to use in this context.

- `viewport` NoneType|Dict *(optional)*

    - `width` int

      page width in pixels.

    - `height` int

      page height in pixels.

  Sets a consistent viewport for each page. Defaults to an 1280x720 viewport. `no_viewport` disables the fixed viewport. Learn more about viewport emulation.

Returns

- Page

start_tracing

Added in: v1.11

> **NOTE**
>
> This API controls Chromium Tracing which is a low-level chromium-specific debugging tool. API to control Playwright Tracing could be found here.

You can use browser.start_tracing() and browser.stop_tracing() to create a trace file that can be opened in Chrome DevTools performance panel.

Usage

Sync | **Async**

```
browser.start_tracing(page, path="trace.json")
page.goto("https://www.google.com")
browser.stop_tracing()
```

Arguments

- `page` Page *(optional)*

  Optional, if specified, tracing includes screenshots of the given page.

- `categories` List[str] *(optional)*

  specify custom categories to use instead of default.

- `path` Union[str, pathlib.Path] *(optional)*

  A path to write the trace file to.

- `screenshots` bool *(optional)*

  captures screenshots in the trace.

stop_tracing

Added in: v1.11

> **NOTE**
>
> This API controls Chromium Tracing which is a low-level chromium-specific debugging tool. API to control Playwright Tracing could be found here.

Returns the buffer with trace data.

Usage

```
browser.stop_tracing()
```

Returns

- bytes

Properties

browser_type

Added in: v1.23

Get the browser type (chromium, firefox or webkit) that the browser belongs to.

Usage

```
browser.browser_type
```

Returns

- BrowserType

contexts

Added in: v1.8

Returns an array of all open browser contexts. In a newly created browser, this will return zero browser contexts.

Usage

Sync | Async

```
browser = pw.webkit.launch()
print(len(browser.contexts())) # prints `0`
context = browser.new_context()
print(len(browser.contexts())) # prints `1`
```

Returns

- List[BrowserContext]

is_connected

Added in: v1.8

Indicates that the browser is connected.

Usage

```
browser.is_connected()
```

Returns

- bool

version

Added in: v1.8

Returns the browser version.

Usage

```
browser.version
```

Returns

- str

Events

on("disconnected")

Added in: v1.8

Emitted when Browser gets disconnected from the browser application. This might happen because of one of the following:

- Browser application is closed or crashed.
- The browser.close() method was called.

Usage

```
browser.on("disconnected", handler)
```

Event data

- Browser