Frame

At every point of time, page exposes its current frame tree via the page.main_frame and frame.child_frames methods.

Frame object's lifecycle is controlled by three events, dispatched on the page object:

- page.on("frameattached") - fired when the frame gets attached to the page. A Frame can be attached to the page only once.
- page.on("framenavigated") - fired when the frame commits navigation to a different URL.
- page.on("framedetached") - fired when the frame gets detached from the page. A Frame can be detached from the page only once.

An example of dumping frame tree:

Sync Async

```python
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://www.theverge.com")
    dump_frame_tree(page.main_frame, "")
    browser.close()

def dump_frame_tree(frame, indent):
    print(indent + frame.name + '@' + frame.url)
    for child in frame.child_frames:
        dump_frame_tree(child, indent + "    ")

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

Methods

add_script_tag

Added in: v1.8

Returns the added tag when the script's onload fires or when the script content was injected into frame.

Adds a `<script>` tag into the page with the desired url or content.

Usage

```
frame.add_script_tag()
frame.add_script_tag(**kwargs)
```

Arguments

- `content` str *(optional)*

  Raw JavaScript content to be injected into frame.

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

  Path to the JavaScript file to be injected into frame. If `path` is a relative path, then it is resolved relative to the current working directory.

- `type` str *(optional)*

  Script type. Use 'module' in order to load a Javascript ES6 module. See script for more details.

- `url` str *(optional)*

  URL of a script to be added.

Returns

- ElementHandle

add_style_tag

Added in: v1.8

Returns the added tag when the stylesheet's onload fires or when the CSS content was injected into frame.

Adds a `<link rel="stylesheet">` tag into the page with the desired url or a `<style type="text/css">` tag with the content.

Usage

```
frame.add_style_tag()
frame.add_style_tag(**kwargs)
```

Arguments

- `content` str *(optional)*

  Raw CSS content to be injected into frame.

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

  Path to the CSS file to be injected into frame. If `path` is a relative path, then it is resolved relative to the current working directory.

- `url` str *(optional)*

  URL of the `<link>` tag.

Returns

- ElementHandle

content

Added in: v1.8

Gets the full HTML contents of the frame, including the doctype.

Usage

```
frame.content()
```

Returns

- str

drag_and_drop

Added in: v1.13

Usage

```
frame.drag_and_drop(source, target)
frame.drag_and_drop(source, target, **kwargs)
```

Arguments

- `source` str

  A selector to search for an element to drag. If there are multiple elements satisfying the selector, the first will be used.

- `target` str

  A selector to search for an element to drop onto. If there are multiple elements satisfying the selector, the first will be used.

- `force` bool *(optional)*

  Whether to bypass the actionability checks. Defaults to `false`.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `source_position` Dict *(optional)* Added in: v1.14

  - `x` float
  - `y` float

  Clicks on the source element at this point relative to the top-left corner of the element's padding box. If not specified, some visible point of the element is used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `target_position` Dict *(optional)* Added in: v1.14

  - `x` float
  - `y` float

  Drops on the target element at this point relative to the top-left corner of the element's padding box. If not specified, some visible point of the element is used.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

- `trial` bool *(optional)*

  When set, this method only performs the actionability checks and skips the action. Defaults to `false`. Useful to wait until the element is ready for the action without performing it.

evaluate

Added in: v1.8

Returns the return value of `expression`.

If the function passed to the frame.evaluate() returns a Promise, then frame.evaluate() would wait for the promise to resolve and return its value.

If the function passed to the frame.evaluate() returns a non-Serializable value, then frame.evaluate() returns `undefined`. Playwright also supports transferring some additional values that are not serializable by `JSON`: `-0`, `NaN`, `Infinity`, `-Infinity`.

Usage

Sync **Async**

```
result = frame.evaluate("([x, y]) => Promise.resolve(x * y)", [7, 8])
print(result) # prints "56"
```

A string can also be passed in instead of a function.

Sync **Async**

```
print(frame.evaluate("1 + 2")) # prints "3"
x = 10
print(frame.evaluate(f"1 + {x}")) # prints "11"
```

ElementHandle instances can be passed as an argument to the frame.evaluate():

Sync **Async**

```
body_handle = frame.evaluate("document.body")
html = frame.evaluate("([body, suffix]) => body.innerHTML + suffix", [body_handle,
"hello"])
body_handle.dispose()
```

Arguments

- `expression` str

  JavaScript expression to be evaluated in the browser context. If the expression evaluates to a function, the function is automatically invoked.

- `arg` EvaluationArgument *(optional)*

  Optional argument to pass to `expression`.

Returns

- Serializable

evaluate_handle

Added in: v1.8

Returns the return value of `expression` as a JSHandle.

The only difference between frame.evaluate() and frame.evaluate_handle() is that frame.evaluate_handle() returns JSHandle.

If the function, passed to the frame.evaluate_handle(), returns a Promise, then frame.evaluate_handle() would wait for the promise to resolve and return its value.

Usage

Sync **Async**

```
a_window_handle = frame.evaluate_handle("Promise.resolve(window)")
a_window_handle # handle for the window object.
```

A string can also be passed in instead of a function.

Sync **Async**

```
a_handle = page.evaluate_handle("document") # handle for the "document"
```

JSHandle instances can be passed as an argument to the frame.evaluate_handle():

Sync **Async**

```
a_handle = page.evaluate_handle("document.body")
result_handle = page.evaluate_handle("body => body.innerHTML", a_handle)
print(result_handle.json_value())
result_handle.dispose()
```

Arguments

- `expression` str

  JavaScript expression to be evaluated in the browser context. If the expression evaluates to a function, the function is automatically invoked.

- `arg` EvaluationArgument *(optional)*

  Optional argument to pass to `expression`.

Returns

- JSHandle

frame_element

Added in: v1.8

Returns the `frame` or `iframe` element handle which corresponds to this frame.

This is an inverse of element_handle.content_frame(). Note that returned handle actually belongs to the parent frame.

This method throws an error if the frame has been detached before `frameElement()` returns.

Usage

Sync Async

```
frame_element = frame.frame_element()
content_frame = frame_element.content_frame()
assert frame == content_frame
```

Returns

- ElementHandle

frame_locator

Added in: v1.17

When working with iframes, you can create a frame locator that will enter the iframe and allow selecting elements in that iframe.

Usage

Following snippet locates element with text "Submit" in the iframe with id `my-frame`, like `<iframe id="my-frame">`:

Sync Async

```
locator = frame.frame_locator("#my-iframe").get_by_text("Submit")
locator.click()
```

Arguments

- `selector` str

  A selector to use when resolving DOM element.

Returns

- FrameLocator

get_by_alt_text

Added in: v1.27

Allows locating elements by their alt text.

Usage

For example, this method will find the image by alt text "Playwright logo":

```
<img alt='Playwright logo'>
```

Sync Async

```
page.get_by_alt_text("Playwright logo").click()
```

Arguments

- `text` str|Pattern

Text to locate the element for.

- `exact` bool *(optional)*

  Whether to find an exact match: case-sensitive and whole-string. Default to false. Ignored when locating by a regular expression. Note that exact match still trims whitespace.

Returns

- Locator

get_by_label

Added in: v1.27

Allows locating input elements by the text of the associated `<label>` or `aria-labelledby` element, or by the `aria-label` attribute.

Usage

For example, this method will find inputs by label "Username" and "Password" in the following DOM:

```
<input aria-label="Username">
<label for="password-input">Password:</label>
<input id="password-input">
```

Sync **Async**

```
page.get_by_label("Username").fill("john")
page.get_by_label("Password").fill("secret")
```

Arguments

- `text` str|Pattern

  Text to locate the element for.

- `exact` bool *(optional)*

Whether to find an exact match: case-sensitive and whole-string. Default to false. Ignored when locating by a regular expression. Note that exact match still trims whitespace.

Returns

- Locator

get_by_placeholder

Added in: v1.27

Allows locating input elements by the placeholder text.

Usage

For example, consider the following DOM structure.

```
<input type="email" placeholder="name@example.com" />
```

You can fill the input after locating it by the placeholder text:

Sync **Async**

```
page.get_by_placeholder("name@example.com").fill("playwright@microsoft.com")
```

Arguments

- `text` str|Pattern

  Text to locate the element for.

- `exact` bool *(optional)*

  Whether to find an exact match: case-sensitive and whole-string. Default to false. Ignored when locating by a regular expression. Note that exact match still trims whitespace.

Returns

- Locator

get_by_role

Added in: v1.27

Allows locating elements by their ARIA role, ARIA attributes and accessible name.

Usage

Consider the following DOM structure.

```html
<h3>Sign up</h3>
<label>
  <input type="checkbox" /> Subscribe
</label>
<br/>
<button>Submit</button>
```

You can locate each element by it's implicit role:

Sync **Async**

```python
expect(page.get_by_role("heading", name="Sign up")).to_be_visible()

page.get_by_role("checkbox", name="Subscribe").check()

page.get_by_role("button", name=re.compile("submit", re.IGNORECASE)).click()
```

Arguments

- `role`
  "alert"|"alertdialog"|"application"|"article"|"banner"|"blockquote"|"button"|"caption"|"cell"|"checkbox"|"code"|"columnheader"|"combobox"|"complementary"|"contentinfo"|"definition"|"deletion"|"dialog"|"directory"|"document"|"emphasis"|"feed"|"figure"|"form"|"generic"|"grid"|"gridcell"|"group"|"heading"|"img"|"insertion"|"link"|"list"|"listbox"|"listitem"|"log"|"main"|"marquee"|"math"|"meter"|"menu"|"menubar"|"menuitem"|"menuitemcheckbox"|"menuitemradio"|"navigation"|"none"|"note"|"option"|"paragraph"|"presentation"|"progressbar"|"radio"|"radiogroup"|"region"|"row"|"rowgroup"|"rowheader"|"scrollbar"|"search"|"searchbox"|"separator"|"slider"|"spinbutton"|"status"|"strong"|"subscript"|"superscript"|"switch"|"tab"|"table"|"tablist"|"tabpanel"|"term"|"textbox"|"time"|"timer"|"toolbar"|"tooltip"|"tree"|"treegrid"|"treeitem"

  Required aria role.

- `checked` bool *(optional)*

  An attribute that is usually set by `aria-checked` or native `<input type=checkbox>` controls.

  Learn more about `aria-checked`.

- `disabled` bool *(optional)*

  An attribute that is usually set by `aria-disabled` or `disabled`.

  > **NOTE**
  >
  > Unlike most other attributes, `disabled` is inherited through the DOM hierarchy. Learn more about `aria-disabled`.

- `exact` bool *(optional)* Added in: v1.28

  Whether `name` is matched exactly: case-sensitive and whole-string. Defaults to false. Ignored when `name` is a regular expression. Note that exact match still trims whitespace.

- `expanded` bool *(optional)*

  An attribute that is usually set by `aria-expanded`.

  Learn more about `aria-expanded`.

- `include_hidden` bool *(optional)*

  Option that controls whether hidden elements are matched. By default, only non-hidden elements, as defined by ARIA, are matched by role selector.

  Learn more about `aria-hidden`.

- `level` int *(optional)*

  A number attribute that is usually present for roles `heading`, `listitem`, `row`, `treeitem`, with default values for `<h1>-<h6>` elements.

  Learn more about `aria-level`.

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

Option to match the accessible name. By default, matching is case-insensitive and searches for a substring, use `exact` to control this behavior.

Learn more about accessible name.

- `pressed` bool *(optional)*

  An attribute that is usually set by `aria-pressed`.

  Learn more about `aria-pressed`.

- `selected` bool *(optional)*

  An attribute that is usually set by `aria-selected`.

  Learn more about `aria-selected`.

Returns

- Locator

Details

Role selector **does not replace** accessibility audits and conformance tests, but rather gives early feedback about the ARIA guidelines.

Many html elements have an implicitly defined role that is recognized by the role selector. You can find all the supported roles here. ARIA guidelines **do not recommend** duplicating implicit roles and attributes by setting `role` and/or `aria-*` attributes to default values.

get_by_test_id

Added in: v1.27

Locate element by the test id.

Usage

Consider the following DOM structure.

```
<button data-testid="directions">Itinéraire</button>
```

You can locate the element by it's test id:

Sync **Async**

```
page.get_by_test_id("directions").click()
```

Arguments

- `test_id` str|Pattern

  Id to locate the element by.

Returns

- Locator

Details

By default, the `data-testid` attribute is used as a test id. Use selectors.set_test_id_attribute() to configure a different test id attribute if necessary.

get_by_text

Added in: v1.27

Allows locating elements that contain given text.

See also locator.filter() that allows to match by another criteria, like an accessible role, and then filter by the text content.

Usage

Consider the following DOM structure:

```
<div>Hello <span>world</span></div>
<div>Hello</div>
```

You can locate by text substring, exact string, or a regular expression:

Sync **Async**

```
# Matches <span>
page.get_by_text("world")

# Matches first <div>
page.get_by_text("Hello world")

# Matches second <div>
page.get_by_text("Hello", exact=True)

# Matches both <div>s
page.get_by_text(re.compile("Hello"))

# Matches second <div>
page.get_by_text(re.compile("^hello$", re.IGNORECASE))
```

Arguments

- `text` str|Pattern

  Text to locate the element for.

- `exact` bool *(optional)*

  Whether to find an exact match: case-sensitive and whole-string. Default to false. Ignored when locating by a regular expression. Note that exact match still trims whitespace.

Returns

- Locator

Details

Matching by text always normalizes whitespace, even with exact match. For example, it turns multiple spaces into one, turns line breaks into spaces and ignores leading and trailing whitespace.

Input elements of the type `button` and `submit` are matched by their `value` instead of the text content. For example, locating by text `"Log in"` matches `<input type=button value="Log in">`.

get_by_title

Added in: v1.27

Allows locating elements by their title attribute.

Usage

Consider the following DOM structure.

```
<span title='Issues count'>25 issues</span>
```

You can check the issues count after locating it by the title text:

Sync **Async**

```
expect(page.get_by_title("Issues count")).to_have_text("25 issues")
```

Arguments

- `text` str|Pattern

  Text to locate the element for.

- `exact` bool *(optional)*

  Whether to find an exact match: case-sensitive and whole-string. Default to false. Ignored when locating by a regular expression. Note that exact match still trims whitespace.

Returns

- Locator

goto

Added in: v1.8

Returns the main resource response. In case of multiple redirects, the navigation will resolve with the response of the last redirect.

The method will throw an error if:

- there's an SSL error (e.g. in case of self-signed certificates).

- target URL is invalid.
- the `timeout` is exceeded during navigation.
- the remote server does not respond or is unreachable.
- the main resource failed to load.

The method will not throw an error when any valid HTTP status code is returned by the remote server, including 404 "Not Found" and 500 "Internal Server Error". The status code for such responses can be retrieved by calling response.status.

> **NOTE**
>
> The method either throws an error or returns a main resource response. The only exceptions are navigation to `about:blank` or navigation to the same URL with a different hash, which would succeed and return `null`.

> **NOTE**
>
> Headless mode doesn't support navigation to a PDF document. See the upstream issue.

Usage

```
frame.goto(url)
frame.goto(url, **kwargs)
```

Arguments

- `url` str

  URL to navigate frame to. The url should include scheme, e.g. `https://`.

- `referer` str *(optional)*

  Referer header value. If provided it will take preference over the referer header value set by page.set_extra_http_headers().

- `timeout` float *(optional)*

  Maximum operation time in milliseconds, defaults to 30 seconds, pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_navigation_timeout(), browser_context.set_default_timeout(), page.set_default_navigation_timeout() or page.set_default_timeout() methods.

- `wait_until` "load"|"domcontentloaded"|"networkidle"|"commit" *(optional)*

  When to consider operation succeeded, defaults to `load`. Events can be either:

  - `'domcontentloaded'` - consider operation to be finished when the `DOMContentLoaded` event is fired.
  - `'load'` - consider operation to be finished when the `load` event is fired.
  - `'networkidle'` - **DISCOURAGED** consider operation to be finished when there are no network connections for at least `500` ms. Don't use this method for testing, rely on web assertions to assess readiness instead.
  - `'commit'` - consider operation to be finished when network response is received and the document started loading.

Returns

- NoneType|Response

is_enabled

Added in: v1.8

Returns whether the element is enabled.

Usage

```
frame.is_enabled(selector)
frame.is_enabled(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- bool

locator

Added in: v1.14

The method returns an element locator that can be used to perform actions on this page / frame. Locator is resolved to the element immediately before performing an action, so a series of actions on the same locator can in fact be performed on different DOM elements. That would happen if the DOM structure between those actions has changed.

Learn more about locators.

Learn more about locators.

Usage

```
frame.locator(selector)
frame.locator(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to use when resolving DOM element.

- `has` Locator *(optional)*

  Matches elements containing an element that matches an inner locator. Inner locator is queried against the outer one. For example, `article` that has `text=Playwright` matches `<article><div>Playwright</div></article>`.

Note that outer and inner locators must belong to the same frame. Inner locator must not contain FrameLocators.

- `has_not` Locator *(optional)* Added in: v1.33

  Matches elements that do not contain an element that matches an inner locator. Inner locator is queried against the outer one. For example, `article` that does not have `div` matches `<article><span>Playwright</span></article>`.

  Note that outer and inner locators must belong to the same frame. Inner locator must not contain FrameLocators.

- `has_not_text` str|Pattern *(optional)* Added in: v1.33

  Matches elements that do not contain specified text somewhere inside, possibly in a child or a descendant element. When passed a [string], matching is case-insensitive and searches for a substring.

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

  Matches elements containing specified text somewhere inside, possibly in a child or a descendant element. When passed a [string], matching is case-insensitive and searches for a substring. For example, `"Playwright"` matches `<article><div>Playwright</div></article>`.

Returns

- Locator

set_content

Added in: v1.8

This method internally calls document.write(), inheriting all its specific characteristics and behaviors.

Usage

```
frame.set_content(html)
frame.set_content(html, **kwargs)
```

Arguments

- `html` str

  HTML markup to assign to the page.

- `timeout` float *(optional)*

  Maximum operation time in milliseconds, defaults to 30 seconds, pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_navigation_timeout(), browser_context.set_default_timeout(), page.set_default_navigation_timeout() or page.set_default_timeout() methods.

- `wait_until` "load"|"domcontentloaded"|"networkidle"|"commit" *(optional)*

  When to consider operation succeeded, defaults to `load`. Events can be either:

  - `'domcontentloaded'` - consider operation to be finished when the `DOMContentLoaded` event is fired.
  - `'load'` - consider operation to be finished when the `load` event is fired.
  - `'networkidle'` - **DISCOURAGED** consider operation to be finished when there are no network connections for at least `500` ms. Don't use this method for testing, rely on web assertions to assess readiness instead.
  - `'commit'` - consider operation to be finished when network response is received and the document started loading.

title

Added in: v1.8

Returns the page title.

Usage

```
frame.title()
```

Returns

- str

wait_for_function

Added in: v1.8

Returns when the `expression` returns a truthy value, returns that value.

Usage

The frame.wait_for_function() can be used to observe viewport size change:

Sync **Async**

```
from playwright.sync_api import sync_playwright, Playwright

def run(playwright: Playwright):
    webkit = playwright.webkit
    browser = webkit.launch()
    page = browser.new_page()
    page.evaluate("window.x = 0; setTimeout(() => { window.x = 100 }, 1000);")
    page.main_frame.wait_for_function("() => window.x > 0")
    browser.close()

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

To pass an argument to the predicate of `frame.waitForFunction` function:

Sync **Async**

```
selector = ".foo"
frame.wait_for_function("selector => !!document.querySelector(selector)",
selector)
```

Arguments

- `expression` str

  JavaScript expression to be evaluated in the browser context. If the expression evaluates to a function, the function is automatically invoked.

- `arg` EvaluationArgument *(optional)*

  Optional argument to pass to `expression`.

- `polling` float|"raf" *(optional)*

  If `polling` is `'raf'`, then `expression` is constantly executed in `requestAnimationFrame` callback. If `polling` is a number, then it is treated as an interval in milliseconds at which the function would be executed. Defaults to `raf`.

- `timeout` float *(optional)*

  Maximum time to wait for in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- JSHandle

wait_for_load_state

Added in: v1.8

Waits for the required load state to be reached.

This returns when the frame reaches a required load state, `load` by default. The navigation must have been committed when this method is called. If current document has already reached the required state, resolves immediately.

Usage

Sync Async

```
frame.click("button") # click triggers navigation.
frame.wait_for_load_state() # the promise resolves after "load" event.
```

Arguments

- `state` "load"|"domcontentloaded"|"networkidle" *(optional)*

  Optional load state to wait for, defaults to `load`. If the state has been already reached while loading current document, the method resolves immediately. Can be one of:

    - `'load'` - wait for the `load` event to be fired.
    - `'domcontentloaded'` - wait for the `DOMContentLoaded` event to be fired.
    - `'networkidle'` - **DISCOURAGED** wait until there are no network connections for at least `500` ms. Don't use this method for testing, rely on web assertions to assess readiness instead.

- `timeout` float *(optional)*

  Maximum operation time in milliseconds, defaults to 30 seconds, pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_navigation_timeout(), browser_context.set_default_timeout(), page.set_default_navigation_timeout() or page.set_default_timeout() methods.

wait_for_url

Added in: v1.11

Waits for the frame to navigate to the given URL.

Usage

Sync **Async**

```
frame.click("a.delayed-navigation") # clicking the link will indirectly cause a
navigation
frame.wait_for_url("**/target.html")
```

Arguments

- `url` str|Pattern|Callable[URL]:bool

  A glob pattern, regex pattern or predicate receiving URL to match while waiting for the navigation. Note that if the parameter is a string without wildcard characters, the method will wait for navigation to URL that is exactly equal to the string.

- `timeout` float *(optional)*

  Maximum operation time in milliseconds, defaults to 30 seconds, pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_navigation_timeout(), browser_context.set_default_timeout(), page.set_default_navigation_timeout() or page.set_default_timeout() methods.

- `wait_until` "load"|"domcontentloaded"|"networkidle"|"commit" *(optional)*

  When to consider operation succeeded, defaults to `load`. Events can be either:

  - `'domcontentloaded'` - consider operation to be finished when the `DOMContentLoaded` event is fired.
  - `'load'` - consider operation to be finished when the `load` event is fired.
  - `'networkidle'` - **DISCOURAGED** consider operation to be finished when there are no network connections for at least `500` ms. Don't use this method for testing, rely on web assertions to assess readiness instead.
  - `'commit'` - consider operation to be finished when network response is received and the document started loading.

Properties

child_frames

Added in: v1.8

Usage

```
frame.child_frames
```

Returns

- List[Frame]

is_detached

Added in: v1.8

Returns `true` if the frame has been detached, or `false` otherwise.

Usage

```
frame.is_detached()
```

Returns

- bool

name

Added in: v1.8

Returns frame's name attribute as specified in the tag.

If the name is empty, returns the id attribute instead.

> **NOTE**
>
> This value is calculated once when the frame is created, and will not update if the attribute is changed later.

Usage

```
frame.name
```

Returns

- str

page

Added in: v1.8

Returns the page containing this frame.

Usage

```
frame.page
```

Returns

- Page

parent_frame

Added in: v1.8

Parent frame, if any. Detached frames and main frames return `null`.

Usage

```
frame.parent_frame
```

Returns

- NoneType|Frame

url

Added in: v1.8

Returns frame's url.

Usage

```
frame.url
```

Returns

- str

Deprecated

check

Added in: v1.8

⚠ **DISCOURAGED**

Use locator-based locator.check() instead. Read more about locators.

This method checks an element matching `selector` by performing the following steps:

1. Find an element matching `selector`. If there is none, wait until a matching element is attached to the DOM.
2. Ensure that matched element is a checkbox or a radio input. If not, this method throws. If the element is already checked, this method returns immediately.
3. Wait for actionability checks on the matched element, unless `force` option is set. If the element is detached during the checks, the whole action is retried.
4. Scroll the element into view if needed.
5. Use page.mouse to click in the center of the element.
6. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set.
7. Ensure that the element is now checked. If not, this method throws.

When all steps combined have not finished during the specified `timeout`, this method throws a TimeoutError. Passing zero timeout disables this.

Usage

```
frame.check(selector)
frame.check(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `force` bool *(optional)*

  Whether to bypass the actionability checks. Defaults to `false`.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `position` Dict *(optional)* Added in: v1.11

    - `x` float
    - `y` float

  A point to use relative to the top-left corner of element padding box. If not specified, uses some visible point of the element.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

- `trial` bool *(optional)* Added in: v1.11

  When set, this method only performs the actionability checks and skips the action. Defaults to `false`. Useful to wait until the element is ready for the action without performing it.

click

Added in: v1.8

DISCOURAGED

Use locator-based locator.click() instead. Read more about locators.

This method clicks an element matching `selector` by performing the following steps:

1. Find an element matching `selector`. If there is none, wait until a matching element is attached to the DOM.
2. Wait for actionability checks on the matched element, unless `force` option is set. If the element is detached during the checks, the whole action is retried.
3. Scroll the element into view if needed.
4. Use page.mouse to click in the center of the element, or the specified `position`.
5. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set.

When all steps combined have not finished during the specified `timeout`, this method throws a TimeoutError. Passing zero timeout disables this.

Usage

```
frame.click(selector)
frame.click(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `button` "left"|"right"|"middle" *(optional)*

  Defaults to `left`.

- `click_count` int *(optional)*

  defaults to 1. See UIEvent.detail.

- `delay` float *(optional)*

  Time to wait between `mousedown` and `mouseup` in milliseconds. Defaults to 0.

- `force` bool *(optional)*

  Whether to bypass the actionability checks. Defaults to `false`.

- `modifiers` List["Alt"|"Control"|"Meta"|"Shift"] *(optional)*

  Modifier keys to press. Ensures that only these modifiers are pressed during the operation, and then restores current modifiers back. If not specified, currently pressed modifiers are used.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `position` Dict *(optional)*

  - `x` float
  - `y` float

  A point to use relative to the top-left corner of element padding box. If not specified, uses some visible point of the element.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

- `trial` bool *(optional)* Added in: v1.11

  When set, this method only performs the actionability checks and skips the action. Defaults to `false`. Useful to wait until the element is ready for the action without performing it.

dblclick

Added in: v1.8

DISCOURAGED

Use locator-based locator.dblclick() instead. Read more about locators.

This method double clicks an element matching `selector` by performing the following steps:

1. Find an element matching `selector`. If there is none, wait until a matching element is attached to the DOM.
2. Wait for actionability checks on the matched element, unless `force` option is set. If the element is detached during the checks, the whole action is retried.
3. Scroll the element into view if needed.
4. Use page.mouse to double click in the center of the element, or the specified `position`.

5. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set. Note that if the first click of the `dblclick()` triggers a navigation event, this method will throw.

When all steps combined have not finished during the specified `timeout`, this method throws a TimeoutError. Passing zero timeout disables this.

> **NOTE**
>
> `frame.dblclick()` dispatches two `click` events and a single `dblclick` event.

Usage

```
frame.dblclick(selector)
frame.dblclick(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `button` "left"|"right"|"middle" *(optional)*

  Defaults to `left`.

- `delay` float *(optional)*

  Time to wait between `mousedown` and `mouseup` in milliseconds. Defaults to 0.

- `force` bool *(optional)*

  Whether to bypass the actionability checks. Defaults to `false`.

- `modifiers` List["Alt"|"Control"|"Meta"|"Shift"] *(optional)*

  Modifier keys to press. Ensures that only these modifiers are pressed during the operation, and then restores current modifiers back. If not specified, currently pressed modifiers are used.

- `no_wait_after` bool *(optional)*

Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `position` Dict *(optional)*
  - `x` float
  - `y` float

  A point to use relative to the top-left corner of element padding box. If not specified, uses some visible point of the element.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

- `trial` bool *(optional)* Added in: v1.11

  When set, this method only performs the actionability checks and skips the action. Defaults to `false`. Useful to wait until the element is ready for the action without performing it.

dispatch_event

Added in: v1.8

DISCOURAGED

Use locator-based locator.dispatch_event() instead. Read more about locators.

The snippet below dispatches the `click` event on the element. Regardless of the visibility state of the element, `click` is dispatched. This is equivalent to calling element.click().

Usage

Sync **Async**

```
frame.dispatch_event("button#submit", "click")
```

Under the hood, it creates an instance of an event based on the given `type`, initializes it with `event_init` properties and dispatches it on the element. Events are `composed`, `cancelable` and bubble by default.

Since `event_init` is event-specific, please refer to the events documentation for the lists of initial properties:

- DragEvent
- FocusEvent
- KeyboardEvent
- MouseEvent
- PointerEvent
- TouchEvent
- Event

You can also specify `JSHandle` as the property value if you want live objects to be passed into the event:

Sync **Async**

```
# note you can only create data_transfer in chromium and firefox
data_transfer = frame.evaluate_handle("new DataTransfer()")
frame.dispatch_event("#source", "dragstart", { "dataTransfer": data_transfer })
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `type` str

DOM event type: `"click"`, `"dragstart"`, etc.

- `event_init` EvaluationArgument *(optional)*

  Optional event-specific initialization properties.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

eval_on_selector

Added in: v1.9

DISCOURAGED

This method does not wait for the element to pass the actionability checks and therefore can lead to the flaky tests. Use locator.evaluate(), other Locator helper methods or web-first assertions instead.

Returns the return value of `expression`.

The method finds an element matching the specified selector within the frame and passes it as a first argument to `expression`. If no elements match the selector, the method throws an error.

If `expression` returns a Promise, then frame.eval_on_selector() would wait for the promise to resolve and return its value.

Usage

Sync Async

```
search_value = frame.eval_on_selector("#search", "el => el.value")
preload_href = frame.eval_on_selector("link[rel=preload]", "el => el.href")
```

```
html = frame.eval_on_selector(".main-container", "(e, suffix) => e.outerHTML +
suffix", "hello")
```

Arguments

- `selector` str

  A selector to query for.

- `expression` str

  JavaScript expression to be evaluated in the browser context. If the expression evaluates to a function, the function is automatically invoked.

- `arg` EvaluationArgument *(optional)*

  Optional argument to pass to `expression`.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

Returns

- Serializable

eval_on_selector_all

Added in: v1.9

DISCOURAGED

In most cases, locator.evaluate_all(), other Locator helper methods and web-first assertions do a better job.

Returns the return value of `expression`.

The method finds all elements matching the specified selector within the frame and passes an array of matched elements as a first argument to `expression`.

If `expression` returns a Promise, then frame.eval_on_selector_all() would wait for the promise to resolve and return its value.

Usage

Sync Async

```
divs_counts = frame.eval_on_selector_all("div", "(divs, min) => divs.length >=
min", 10)
```

Arguments

- `selector` str

  A selector to query for.

- `expression` str

  JavaScript expression to be evaluated in the browser context. If the expression evaluates to a function, the function is automatically invoked.

- `arg` EvaluationArgument *(optional)*

  Optional argument to pass to `expression`.

Returns

- Serializable

expect_navigation

Added in: v1.8

DEPRECATED

This method is inherently racy, please use frame.wait_for_url() instead.

Waits for the frame navigation and returns the main resource response. In case of multiple redirects, the navigation will resolve with the response of the last redirect. In case of navigation to a different anchor or navigation due to History API usage, the navigation will resolve with `null`.

Usage

This method waits for the frame to navigate to a new URL. It is useful for when you run code which will indirectly cause the frame to navigate. Consider this example:

Sync **Async**

```
with frame.expect_navigation():
    frame.click("a.delayed-navigation") # clicking the link will indirectly cause a navigation
# Resolves after navigation has finished
```

NOTE

Usage of the History API to change the URL is considered a navigation.

Arguments

- `timeout` float *(optional)*

  Maximum operation time in milliseconds, defaults to 30 seconds, pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_navigation_timeout(), browser_context.set_default_timeout(), page.set_default_navigation_timeout() or page.set_default_timeout() methods.

- `url` str|Pattern|Callable[URL]:bool *(optional)*

  A glob pattern, regex pattern or predicate receiving URL to match while waiting for the navigation. Note that if the parameter is a string without wildcard characters, the method will wait for navigation to URL that is exactly equal to the string.

- `wait_until` "load"|"domcontentloaded"|"networkidle"|"commit" *(optional)*

  When to consider operation succeeded, defaults to `load`. Events can be either:

  - `'domcontentloaded'` - consider operation to be finished when the `DOMContentLoaded` event is fired.
  - `'load'` - consider operation to be finished when the `load` event is fired.

  - `'networkidle'` - **DISCOURAGED** consider operation to be finished when there are no network connections for at least `500` ms. Don't use this method for testing, rely on web assertions to assess readiness instead.
  - `'commit'` - consider operation to be finished when network response is received and the document started loading.

Returns

- EventContextManager[Response]

fill

Added in: v1.8

DISCOURAGED

Use locator-based locator.fill() instead. Read more about locators.

This method waits for an element matching `selector`, waits for actionability checks, focuses the element, fills it and triggers an `input` event after filling. Note that you can pass an empty string to clear the input field.

If the target element is not an `<input>`, `<textarea>` or `[contenteditable]` element, this method throws an error. However, if the element is inside the `<label>` element that has an associated control, the control will be filled instead.

To send fine-grained keyboard events, use locator.press_sequentially().

Usage

```
frame.fill(selector, value)
frame.fill(selector, value, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `value` str

Value to fill for the `<input>`, `<textarea>` or `[contenteditable]` element.

- `force` bool *(optional)* Added in: v1.13

  Whether to bypass the actionability checks. Defaults to `false`.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

focus

Added in: v1.8

> **DISCOURAGED**
>
> Use locator-based locator.focus() instead. Read more about locators.

This method fetches an element with `selector` and focuses it. If there's no element matching `selector`, the method waits until a matching element appears in the DOM.

Usage

```
frame.focus(selector)
frame.focus(selector, **kwargs)
```

Arguments

- `selector` str

A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

get_attribute

Added in: v1.8

DISCOURAGED

Use locator-based locator.get_attribute() instead. Read more about locators.

Returns element attribute value.

Usage

```
frame.get_attribute(selector, name)
frame.get_attribute(selector, name, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `name` str

  Attribute name to get the value for.

- `strict` bool *(optional)* Added in: v1.14

When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- NoneType|str

hover

Added in: v1.8

DISCOURAGED

Use locator-based locator.hover() instead. Read more about locators.

This method hovers over an element matching `selector` by performing the following steps:

1. Find an element matching `selector`. If there is none, wait until a matching element is attached to the DOM.
2. Wait for actionability checks on the matched element, unless `force` option is set. If the element is detached during the checks, the whole action is retried.
3. Scroll the element into view if needed.
4. Use page.mouse to hover over the center of the element, or the specified `position`.
5. Wait for initiated navigations to either succeed or fail, unless `noWaitAfter` option is set.

When all steps combined have not finished during the specified `timeout`, this method throws a TimeoutError. Passing zero timeout disables this.

Usage

```
frame.hover(selector)
frame.hover(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `force` bool *(optional)*

  Whether to bypass the actionability checks. Defaults to `false`.

- `modifiers` List["Alt"|"Control"|"Meta"|"Shift"] *(optional)*

  Modifier keys to press. Ensures that only these modifiers are pressed during the operation, and then restores current modifiers back. If not specified, currently pressed modifiers are used.

- `no_wait_after` bool *(optional)* Added in: v1.28

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `position` Dict *(optional)*

  - `x` float
  - `y` float

  A point to use relative to the top-left corner of element padding box. If not specified, uses some visible point of the element.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

- `trial` bool *(optional)* Added in: v1.11

When set, this method only performs the actionability checks and skips the action. Defaults to `false`. Useful to wait until the element is ready for the action without performing it.

inner_html

Added in: v1.8

DISCOURAGED

Use locator-based locator.inner_html() instead. Read more about locators.

Returns `element.innerHTML`.

Usage

```
frame.inner_html(selector)
frame.inner_html(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- str

inner_text

Added in: v1.8

DISCOURAGED

Use locator-based locator.inner_text() instead. Read more about locators.

Returns `element.innerText`.

Usage

```
frame.inner_text(selector)
frame.inner_text(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- str

input_value

Added in: v1.13

DISCOURAGED

Use locator-based locator.input_value() instead. Read more about locators.

Returns `input.value` for the selected `<input>` or `<textarea>` or `<select>` element.

Throws for non-input elements. However, if the element is inside the `<label>` element that has an associated control, returns the value of the control.

Usage

```
frame.input_value(selector)
frame.input_value(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- str

is_checked

Added in: v1.8

DISCOURAGED

Use locator-based locator.is_checked() instead. Read more about locators.

Returns whether the element is checked. Throws if the element is not a checkbox or radio input.

Usage

```
frame.is_checked(selector)
frame.is_checked(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- bool

is_disabled

Added in: v1.8

DISCOURAGED

Use locator-based locator.is_disabled() instead. Read more about locators.

Returns whether the element is disabled, the opposite of enabled.

Usage

```
frame.is_disabled(selector)
frame.is_disabled(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- bool

is_editable

Added in: v1.8

DISCOURAGED

Use locator-based locator.is_editable() instead. Read more about locators.

Returns whether the element is editable.

Usage

```
frame.is_editable(selector)
frame.is_editable(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- bool

is_hidden

Added in: v1.8

DISCOURAGED

Use locator-based locator.is_hidden() instead. Read more about locators.

Returns whether the element is hidden, the opposite of visible. `selector` that does not match any elements is considered hidden.

Usage

```
frame.is_hidden(selector)
frame.is_hidden(selector, **kwargs)
```

Arguments

- `selector` str

A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  > **DEPRECATED**
  >
  > This option is ignored. frame.is_hidden() does not wait for the element to become hidden and returns immediately.

Returns

- bool

is_visible

Added in: v1.8

> **DISCOURAGED**
>
> Use locator-based locator.is_visible() instead. Read more about locators.

Returns whether the element is visible. `selector` that does not match any elements is considered not visible.

Usage

```
frame.is_visible(selector)
frame.is_visible(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  > ⚠ **DEPRECATED**
  >
  > This option is ignored. frame.is_visible() does not wait for the element to become visible and returns immediately.

Returns

- bool

press

Added in: v1.8

> ⚠ **DISCOURAGED**
>
> Use locator-based locator.press() instead. Read more about locators.

`key` can specify the intended keyboardEvent.key value or a single character to generate the text for. A superset of the `key` values can be found here. Examples of the keys are:

`F1` - `F12`, `Digit0`- `Digit9`, `KeyA`- `KeyZ`, `Backquote`, `Minus`, `Equal`, `Backslash`, `Backspace`, `Tab`, `Delete`, `Escape`, `ArrowDown`, `End`, `Enter`, `Home`, `Insert`, `PageDown`, `PageUp`, `ArrowRight`, `ArrowUp`, etc.

Following modification shortcuts are also supported: `Shift`, `Control`, `Alt`, `Meta`, `ShiftLeft`.

Holding down `Shift` will type the text that corresponds to the `key` in the upper case.

If `key` is a single character, it is case-sensitive, so the values `a` and `A` will generate different respective texts.

Shortcuts such as `key: "Control+o"` or `key: "Control+Shift+T"` are supported as well. When specified with the modifier, modifier is pressed and being held while the subsequent key is being pressed.

Usage

```
frame.press(selector, key)
frame.press(selector, key, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `key` str

  Name of the key to press or a character to generate, such as `ArrowLeft` or `a`.

- `delay` float *(optional)*

  Time to wait between `keydown` and `keyup` in milliseconds. Defaults to 0.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

query_selector

Added in: v1.9

DISCOURAGED

Use locator-based frame.locator() instead. Read more about locators.

Returns the ElementHandle pointing to the frame element.

CAUTION

The use of ElementHandle is discouraged, use Locator objects and web-first assertions instead.

The method finds an element matching the specified selector within the frame. If no elements match the selector, returns `null`.

Usage

```
frame.query_selector(selector)
frame.query_selector(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to query for.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

Returns

- NoneType|ElementHandle

query_selector_all

Added in: v1.9

DISCOURAGED

Use locator-based frame.locator() instead. Read more about locators.

Returns the ElementHandles pointing to the frame elements.

> **CAUTION**
>
> The use of ElementHandle is discouraged, use Locator objects instead.

The method finds all elements matching the specified selector within the frame. If no elements match the selector, returns empty array.

Usage

```
frame.query_selector_all(selector)
```

Arguments

- `selector` str

  A selector to query for.

Returns

- List[ElementHandle]

select_option

Added in: v1.8

> **DISCOURAGED**
>
> Use locator-based locator.select_option() instead. Read more about locators.

This method waits for an element matching `selector`, waits for actionability checks, waits until all specified options are present in the `<select>` element and selects these options.

If the target element is not a `<select>` element, this method throws an error. However, if the element is inside the `<label>` element that has an associated control, the control will be used instead.

Returns the array of option values that have been successfully selected.

Triggers a `change` and `input` event once all the provided options have been selected.

Usage

Sync Async

```
# Single selection matching the value or label
frame.select_option("select#colors", "blue")
# single selection matching both the label
frame.select_option("select#colors", label="blue")
# multiple selection
frame.select_option("select#colors", value=["red", "green", "blue"])
```

Arguments

- `selector` str

  A selector to query for.

- `force` bool *(optional)* Added in: v1.13

  Whether to bypass the actionability checks. Defaults to `false`.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

- `element` ElementHandle|List[ElementHandle] *(optional)*

  Option elements to select. Optional.

- `index` int|List[int] *(optional)*

  Options to select by index. Optional.

- `value` str|List[str] *(optional)*

  Options to select by value. If the `<select>` has the `multiple` attribute, all given options are selected, otherwise only the first option matching one of the passed options is selected. Optional.

- `label` str|List[str] *(optional)*

  Options to select by label. If the `<select>` has the `multiple` attribute, all given options are selected, otherwise only the first option matching one of the passed options is selected. Optional.

Returns

- List[str]

set_checked

Added in: v1.15

⚠ **DISCOURAGED**

Use locator-based locator.set_checked() instead. Read more about locators.

This method checks or unchecks an element matching `selector` by performing the following steps:

1. Find an element matching `selector`. If there is none, wait until a matching element is attached to the DOM.
2. Ensure that matched element is a checkbox or a radio input. If not, this method throws.
3. If the element already has the right checked state, this method returns immediately.
4. Wait for actionability checks on the matched element, unless `force` option is set. If the element is detached during the checks, the whole action is retried.
5. Scroll the element into view if needed.
6. Use page.mouse to click in the center of the element.
7. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set.

8. Ensure that the element is now checked or unchecked. If not, this method throws.

When all steps combined have not finished during the specified `timeout`, this method throws a TimeoutError. Passing zero timeout disables this.

Usage

```
frame.set_checked(selector, checked)
frame.set_checked(selector, checked, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `checked` bool

  Whether to check or uncheck the checkbox.

- `force` bool *(optional)*

  Whether to bypass the actionability checks. Defaults to `false`.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `position` Dict *(optional)*

  - `x` float
  - `y` float

  A point to use relative to the top-left corner of element padding box. If not specified, uses some visible point of the element.

- `strict` bool *(optional)*

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

- `trial` bool *(optional)*

  When set, this method only performs the actionability checks and skips the action. Defaults to `false`. Useful to wait until the element is ready for the action without performing it.

set_input_files

Added in: v1.8

⚠ **DISCOURAGED**

Use locator-based locator.set_input_files() instead. Read more about locators.

Sets the value of the file input to these file paths or files. If some of the `filePaths` are relative paths, then they are resolved relative to the current working directory. For empty array, clears the selected files.

This method expects `selector` to point to an input element. However, if the element is inside the `<label>` element that has an associated control, targets the control instead.

Usage

```
frame.set_input_files(selector, files)
frame.set_input_files(selector, files, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `files` Union[str, pathlib.Path]|List[Union[str, pathlib.Path]]|Dict|List[Dict]

  - `name` str

    File name

  - `mimeType` str

    File type

  - `buffer` bytes

    File content

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

tap

Added in: v1.8

⚠ DISCOURAGED

Use locator-based locator.tap() instead. Read more about locators.

This method taps an element matching `selector` by performing the following steps:

1. Find an element matching `selector`. If there is none, wait until a matching element is attached to the DOM.
2. Wait for actionability checks on the matched element, unless `force` option is set. If the element is detached during the checks, the whole action is retried.
3. Scroll the element into view if needed.

4. Use page.touchscreen to tap the center of the element, or the specified `position`.
5. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set.

When all steps combined have not finished during the specified `timeout`, this method throws a TimeoutError. Passing zero timeout disables this.

> **NOTE**
>
> `frame.tap()` requires that the `hasTouch` option of the browser context be set to true.

Usage

```
frame.tap(selector)
frame.tap(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `force` bool *(optional)*

  Whether to bypass the actionability checks. Defaults to `false`.

- `modifiers` List["Alt"|"Control"|"Meta"|"Shift"] *(optional)*

  Modifier keys to press. Ensures that only these modifiers are pressed during the operation, and then restores current modifiers back. If not specified, currently pressed modifiers are used.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `position` Dict *(optional)*

  - `x` float
  - `y` float

A point to use relative to the top-left corner of element padding box. If not specified, uses some visible point of the element.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

- `trial` bool *(optional)* Added in: v1.11

  When set, this method only performs the actionability checks and skips the action. Defaults to `false`. Useful to wait until the element is ready for the action without performing it.

text_content

Added in: v1.8

DISCOURAGED

Use locator-based locator.text_content() instead. Read more about locators.

Returns `element.textContent`.

Usage

```
frame.text_content(selector)
frame.text_content(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `strict` bool *(optional)* Added in: v1.14

When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- NoneType|str

type

Added in: v1.8

DEPRECATED

In most cases, you should use locator.fill() instead. You only need to press keys one by one if there is special keyboard handling on the page - in this case use locator.press_sequentially().

Sends a `keydown`, `keypress`/`input`, and `keyup` event for each character in the text. `frame.type` can be used to send fine-grained keyboard events. To fill values in form fields, use frame.fill().

To press a special key, like `Control` or `ArrowDown`, use keyboard.press().

Usage

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `text` str

  A text to type into a focused element.

- `delay` float *(optional)*

  Time to wait between key presses in milliseconds. Defaults to 0.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

uncheck

Added in: v1.8

DISCOURAGED

Use locator-based locator.uncheck() instead. Read more about locators.

This method checks an element matching `selector` by performing the following steps:

1. Find an element matching `selector`. If there is none, wait until a matching element is attached to the DOM.
2. Ensure that matched element is a checkbox or a radio input. If not, this method throws. If the element is already unchecked, this method returns immediately.
3. Wait for actionability checks on the matched element, unless `force` option is set. If the element is detached during the checks, the whole action is retried.
4. Scroll the element into view if needed.
5. Use page.mouse to click in the center of the element.
6. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set.
7. Ensure that the element is now unchecked. If not, this method throws.

When all steps combined have not finished during the specified `timeout`, this method throws a TimeoutError. Passing zero timeout disables this.

Usage

```
frame.uncheck(selector)
frame.uncheck(selector, **kwargs)
```

Arguments

- `selector` str

  A selector to search for an element. If there are multiple elements satisfying the selector, the first will be used.

- `force` bool *(optional)*

  Whether to bypass the actionability checks. Defaults to `false`.

- `no_wait_after` bool *(optional)*

  Actions that initiate navigations are waiting for these navigations to happen and for pages to start loading. You can opt out of waiting via setting this flag. You would only need this option in the exceptional cases such as navigating to inaccessible pages. Defaults to `false`.

- `position` Dict *(optional)* Added in: v1.11

  - `x` float
  - `y` float

  A point to use relative to the top-left corner of element padding box. If not specified, uses some visible point of the element.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

- `trial` bool *(optional)* Added in: v1.11

When set, this method only performs the actionability checks and skips the action. Defaults to `false`. Useful to wait until the element is ready for the action without performing it.

wait_for_selector

Added in: v1.8

DISCOURAGED

Use web assertions that assert visibility or a locator-based locator.wait_for() instead. Read more about locators.

Returns when element specified by selector satisfies `state` option. Returns `null` if waiting for `hidden` or `detached`.

NOTE

Playwright automatically waits for element to be ready before performing an action. Using Locator objects and web-first assertions make the code wait-for-selector-free.

Wait for the `selector` to satisfy `state` option (either appear/disappear from dom, or become visible/hidden). If at the moment of calling the method `selector` already satisfies the condition, the method will return immediately. If the selector doesn't satisfy the condition for the `timeout` milliseconds, the function will throw.

Usage

This method works across navigations:

Sync **Async**

```
from playwright.sync_api import sync_playwright, Playwright

def run(playwright: Playwright):
    chromium = playwright.chromium
    browser = chromium.launch()
    page = browser.new_page()
    for current_url in ["https://google.com", "https://bbc.com"]:
        page.goto(current_url, wait_until="domcontentloaded")
        element = page.main_frame.wait_for_selector("img")
        print("Loaded image: " + str(element.get_attribute("src")))
    browser.close()

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

Arguments

- `selector` str

  A selector to query for.

- `state` "attached"|"detached"|"visible"|"hidden" *(optional)*

  Defaults to `'visible'`. Can be either:

  - `'attached'` - wait for element to be present in DOM.
  - `'detached'` - wait for element to not be present in DOM.
  - `'visible'` - wait for element to have non-empty bounding box and no `visibility:hidden`. Note that element without any content or with `display:none` has an empty bounding box and is not considered visible.
  - `'hidden'` - wait for element to be either detached from DOM, or have an empty bounding box or `visibility:hidden`. This is opposite to the `'visible'` option.

- `strict` bool *(optional)* Added in: v1.14

  When true, the call requires selector to resolve to a single element. If given selector resolves to more than one element, the call throws an exception.

- `timeout` float *(optional)*

  Maximum time in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout. The default value can be changed by using the browser_context.set_default_timeout() or page.set_default_timeout() methods.

Returns

- NoneType|ElementHandle

wait_for_timeout

Added in: v1.8

⚠ **DISCOURAGED**

Never wait for timeout in production. Tests that wait for time are inherently flaky. Use Locator actions and web assertions that wait automatically.

Waits for the given `timeout` in milliseconds.

Note that `frame.waitForTimeout()` should only be used for debugging. Tests using the timer in production are going to be flaky. Use signals such as network events, selectors becoming visible and others instead.

Usage

```
frame.wait_for_timeout(timeout)
```

Arguments

- `timeout` float

  A timeout to wait for