Locator

Locators are the central piece of Playwright's auto-waiting and retry-ability. In a nutshell, locators represent a way to find element(s) on the page at any moment. A locator can be created with the page.locator() method.

Learn more about locators.

Methods

all

Added in: v1.29

When the locator points to a list of elements, this returns an array of locators, pointing to their respective elements.

> **NOTE**
>
> locator.all() does not wait for elements to match the locator, and instead immediately returns whatever is present in the page. When the list of elements changes dynamically, locator.all() will produce unpredictable and flaky results. When the list of elements is stable, but loaded dynamically, wait for the full list to finish loading before calling locator.all().

Usage

Sync Async

```
for li in page.get_by_role('listitem').all():
  li.click();
```

Returns

- List[Locator]

all_inner_texts

Added in: v1.14

Returns an array of `node.innerText` values for all matching nodes.

> **ASSERTING TEXT**
>
> If you need to assert text on the page, prefer expect(locator).to_have_text() with `use_inner_text` option to avoid flakiness. See assertions guide for more details.

Usage

Sync | Async

```
texts = page.get_by_role("link").all_inner_texts()
```

Returns

- List[str]

all_text_contents

Added in: v1.14

Returns an array of `node.textContent` values for all matching nodes.

> **ASSERTING TEXT**
>
> If you need to assert text on the page, prefer expect(locator).to_have_text() to avoid flakiness. See assertions guide for more details.

Usage

Sync | Async

```
texts = page.get_by_role("link").all_text_contents()
```

Returns

- List[str]

and_

Added in: v1.34

Creates a locator that matches both this locator and the argument locator.

Usage

The following example finds a button with a specific title.

Sync | Async

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

Arguments

- `locator` Locator

  Additional locator to match.

Returns

- Locator

blur

Added in: v1.28

Calls blur on the element.

Usage

```
locator.blur()
locator.blur(**kwargs)
```

Arguments

- `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.

bounding_box

Added in: v1.14

This method returns the bounding box of the element matching the locator, or `null` if the element is not visible. The bounding box is calculated relative to the main frame viewport - which is usually the same as the browser window.

Usage

Sync | Async

```
box = page.get_by_role("button").bounding_box()
page.mouse.click(box["x"] + box["width"] / 2, box["y"] + box["height"] / 2)
```

Arguments

- `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|Dict
  - `x` float

    the x coordinate of the element in pixels.

  - `y` float

    the y coordinate of the element in pixels.

- `width` float

  the width of the element in pixels.

- `height` float

  the height of the element in pixels.

Details

Scrolling affects the returned bounding box, similarly to Element.getBoundingClientRect. That means `x` and/or `y` may be negative.

Elements from child frames return the bounding box relative to the main frame, unlike the Element.getBoundingClientRect.

Assuming the page is static, it is safe to use bounding box coordinates to perform input. For example, the following snippet should click the center of the element.

check

Added in: v1.14

Ensure that checkbox or radio element is checked.

Usage

Sync | Async

```
page.get_by_role("checkbox").check()
```

Arguments

- `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.

- `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.

Details

Performs the following steps:

1. Ensure that element is a checkbox or a radio input. If not, this method throws. If the element is already checked, this method returns immediately.
2. Wait for actionability checks on the element, unless `force` option is set.
3. Scroll the element into view if needed.
4. Use page.mouse to click in the center of the element.
5. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set.
6. Ensure that the element is now checked. If not, this method throws.

If the element is detached from the DOM at any moment during the action, this method throws.

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

clear

Added in: v1.28

Clear the input field.

Usage

Sync | Async

```
page.get_by_role("textbox").clear()
```

Arguments

- `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`.

- `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.

Details

This method waits for actionability checks, focuses the element, clears it and triggers an `input` event after clearing.

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 cleared instead.

click

Added in: v1.14

Click an element.

Usage

Click a button:

Sync Async

```
page.get_by_role("button").click()
```

Shift-right-click at a specific position on a canvas:

Sync Async

```
page.locator("canvas").click(
    button="right", modifiers=["Shift"], position={"x": 23, "y": 32}
)
```

Arguments

- `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.

- `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.

Details

This method clicks the element by performing the following steps:

1. Wait for actionability checks on the element, unless `force` option is set.
2. Scroll the element into view if needed.
3. Use page.mouse to click in the center of the element, or the specified `position`.
4. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set.

If the element is detached from the DOM at any moment during the action, this method throws.

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

count

Added in: v1.14

Returns the number of elements matching the locator.

> **ASSERTING COUNT**
>
> If you need to assert the number of elements on the page, prefer expect(locator).to_have_count() to avoid flakiness. See assertions guide for more details.

Usage

Sync | Async

```
count = page.get_by_role("listitem").count()
```

Returns

- int

dblclick

Added in: v1.14

Double-click an element.

Usage

```
locator.dblclick()
locator.dblclick(**kwargs)
```

Arguments

- `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.

- `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.

Details

This method double clicks the element by performing the following steps:

1. Wait for actionability checks on the element, unless `force` option is set.
2. Scroll the element into view if needed.
3. Use page.mouse to double click in the center of the element, or the specified `position`.
4. 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.

If the element is detached from the DOM at any moment during the action, this method throws.

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

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

dispatch_event

Added in: v1.14

Programmatically dispatch an event on the matching element.

Usage

Sync | Async

```
locator.dispatch_event("click")
```

Arguments

- `type` str

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

- `event_init` EvaluationArgument *(optional)*

  Optional event-specific initialization properties.

- `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.

Details

The snippet above 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().

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 = page.evaluate_handle("new DataTransfer()")
locator.dispatch_event("#source", "dragstart", {"dataTransfer": data_transfer})
```

drag_to

Added in: v1.18

Drag the source element towards the target element and drop it.

Usage

Sync **Async**

```python
source = page.locator("#source")
target = page.locator("#target")

source.drag_to(target)
# or specify exact positions relative to the top-left corners of the elements:
source.drag_to(
  target,
  source_position={"x": 34, "y": 7},
  target_position={"x": 10, "y": 20}
)
```

Arguments

- `target` Locator

  Locator of the element to drag to.

- `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)*

  - `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.

- `target_position` Dict *(optional)*

  - `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.

Details

This method drags the locator to another target locator or target position. It will first move to the source element, perform a `mousedown`, then move to the target element or position and perform a `mouseup`.

evaluate

Added in: v1.14

Execute JavaScript code in the page, taking the matching element as an argument.

Usage

Sync **Async**

```
tweets = page.locator(".tweet .retweets")
assert tweets.evaluate("node => node.innerText") == "10 retweets"
```

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`.

- `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

- Serializable

Details

Returns the return value of `expression`, called with the matching element as a first argument, and `arg` as a second argument.

If `expression` returns a Promise, this method will wait for the promise to resolve and return its value.

If `expression` throws or rejects, this method throws.

evaluate_all

Added in: v1.14

Execute JavaScript code in the page, taking all matching elements as an argument.

Usage

Sync | Async

```
locator = page.locator("div")
more_than_ten = locator.evaluate_all("(divs, min) => divs.length > min", 10)
```

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

Details

Returns the return value of `expression`, called with an array of all matching elements as a first argument, and `arg` as a second argument.

If `expression` returns a Promise, this method will wait for the promise to resolve and return its value.

If `expression` throws or rejects, this method throws.

evaluate_handle

Added in: v1.14

Execute JavaScript code in the page, taking the matching element as an argument, and return a JSHandle with the result.

Usage

```
locator.evaluate_handle(expression)
locator.evaluate_handle(expression, **kwargs)
```

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`.

- `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

- JSHandle

Details

Returns the return value of `expression` as aJSHandle, called with the matching element as a first argument, and `arg` as a second argument.

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

If `expression` returns a Promise, this method will wait for the promise to resolve and return its value.

If `expression` throws or rejects, this method throws.

See page.evaluate_handle() for more details.

fill

Added in: v1.14

Set a value to the input field.

Usage

Sync | Async

```
page.get_by_role("textbox").fill("example value")
```

Arguments

- `value` str

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

- `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`.

- `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.

Details

This method 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().

filter

Added in: v1.22

This method narrows existing locator according to the options, for example filters by text. It can be chained to filter multiple times.

Usage

Sync Async

```
row_locator = page.locator("tr")
# ...
row_locator.filter(has_text="text in column 1").filter(
```

```
    has=page.get_by_role("button", name="column 2 button")
).screenshot()
```

Arguments

- `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

focus

Added in: v1.14

Calls focus on the matching element.

Usage

```
locator.focus()
locator.focus(**kwargs)
```

Arguments

- `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.

frame_locator

Added in: v1.17

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

Usage

Sync **Async**

```
locator = page.frame_locator("iframe").get_by_text("Submit")
locator.click()
```

Arguments

- `selector` str

  A selector to use when resolving DOM element.

Returns

- FrameLocator

get_attribute

Added in: v1.14

Returns the matching element's attribute value.

> **ASSERTING ATTRIBUTES**
>
> If you need to assert an element's attribute, prefer expect(locator).to_have_attribute() to avoid flakiness. See assertions guide for more details.

Usage

```
locator.get_attribute(name)
locator.get_attribute(name, **kwargs)
```

Arguments

- `name` str

  Attribute name to get the value for.

- `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

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.

```
<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

```
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

highlight

Added in: v1.20

Highlight the corresponding element(s) on the screen. Useful for debugging, don't commit the code that uses locator.highlight().

Usage

```
locator.highlight()
```

hover

Added in: v1.14

Hover over the matching element.

Usage

Sync | Async

```
page.get_by_role("link").hover()
```

Arguments

- `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.

- `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.

Details

This method hovers over the element by performing the following steps:

1. Wait for actionability checks on the element, unless `force` option is set.
2. Scroll the element into view if needed.
3. Use page.mouse to hover over the center of the element, or the specified `position`.
4. Wait for initiated navigations to either succeed or fail, unless `noWaitAfter` option is set.

If the element is detached from the DOM at any moment during the action, this method throws.

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

inner_html

Added in: v1.14

Returns the `element.innerHTML`.

Usage

```
locator.inner_html()
locator.inner_html(**kwargs)
```

Arguments

- `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: v1.14

Returns the `element.innerText`.

> **ASSERTING TEXT**
>
> If you need to assert text on the page, prefer expect(locator).to_have_text() with `use_inner_text` option to avoid flakiness. See assertions guide for more details.

Usage

```
locator.inner_text()
locator.inner_text(**kwargs)
```

Arguments

- `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: v1.14

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

> **ASSERTING VALUE**
>
> If you need to assert input value, prefer expect(locator).to_have_value() to avoid flakiness. See assertions guide for more details.

Usage

Sync | Async

```
value = page.get_by_role("textbox").input_value()
```

Arguments

- `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

Details

Throws elements that are not an input, textarea or a select. However, if the element is inside the `<label>` element that has an associated control, returns the value of the control.

is_checked

Added in: v1.14

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

> **ASSERTING CHECKED STATE**
>
> If you need to assert that checkbox is checked, prefer expect(locator).to_be_checked() to avoid flakiness. See assertions guide for more details.

Usage

Sync | Async

```
checked = page.get_by_role("checkbox").is_checked()
```

Arguments

- `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: v1.14

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

> **ASSERTING DISABLED STATE**
>
> If you need to assert that an element is disabled, prefer expect(locator).to_be_disabled() to avoid flakiness. See assertions guide for more details.

Usage

Sync Async

```
disabled = page.get_by_role("button").is_disabled()
```

Arguments

- `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: v1.14

Returns whether the element is editable.

> **ASSERTING EDITABLE STATE**
>
> If you need to assert that an element is editable, prefer expect(locator).to_be_editable() to avoid flakiness. See assertions guide for more details.

Usage

Sync Async

```
editable = page.get_by_role("textbox").is_editable()
```

Arguments

- `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_enabled

Added in: v1.14

Returns whether the element is enabled.

> **ASSERTING ENABLED STATE**
>
> If you need to assert that an element is enabled, prefer expect(locator).to_be_enabled() to avoid flakiness. See assertions guide for more details.

Usage

Sync | Async

```
enabled = page.get_by_role("button").is_enabled()
```

Arguments

- `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: v1.14

Returns whether the element is hidden, the opposite of visible.

> **ASSERTING VISIBILITY**
>
> If you need to assert that element is hidden, prefer expect(locator).to_be_hidden() to avoid flakiness. See assertions guide for more details.

Usage

Sync | Async

```
hidden = page.get_by_role("button").is_hidden()
```

Arguments

- `timeout` float *(optional)*

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

Returns

- bool

is_visible

Added in: v1.14

Returns whether the element is visible.

> **ASSERTING VISIBILITY**
>
> If you need to assert that element is visible, prefer expect(locator).to_be_visible() to avoid flakiness. See assertions guide for more details.

Usage

Sync Async

```
visible = page.get_by_role("button").is_visible()
```

Arguments

- `timeout` float *(optional)*

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

Returns

- bool

locator

Added in: v1.14

The method finds an element matching the specified selector in the locator's subtree. It also accepts filter options, similar to locator.filter() method.

Learn more about locators.

Usage

```
locator.locator(selector_or_locator)
locator.locator(selector_or_locator, **kwargs)
```

Arguments

- `selector_or_locator` str|Locator

  A selector or locator 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

nth

Added in: v1.14

Returns locator to the n-th matching element. It's zero based, `nth(0)` selects the first element.

Usage

Sync Async

```
banana = page.get_by_role("listitem").nth(2)
```

Arguments

- `index` int

Returns

- Locator

or_

Added in: v1.33

Creates a locator that matches either of the two locators.

Usage

Consider a scenario where you'd like to click on a "New email" button, but sometimes a security settings dialog shows up instead. In this case, you can wait for either a "New email" button, or a dialog and act accordingly.

Sync Async

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

Arguments

- `locator` Locator

  Alternative locator to match.

Returns

- Locator

press

Added in: v1.14

Focuses the matching element and presses a combination of the keys.

Usage

Sync | Async

```
page.get_by_role("textbox").press("Backspace")
```

Arguments

- `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`.

- `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.

Details

Focuses the element, and then uses keyboard.down() and keyboard.up().

`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.

press_sequentially

Added in: v1.38

> **TIP**
>
> 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.

Focuses the element, and then sends a `keydown`, `keypress`/`input`, and `keyup` event for each character in the text.

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

Usage

Sync | Async

```
locator.press_sequentially("hello") # types instantly
locator.press_sequentially("world", delay=100) # types slower, like a user
```

An example of typing into a text field and then submitting the form:

Sync Async

```
locator = page.get_by_label("Password")
locator.press_sequentially("my password")
locator.press("Enter")
```

Arguments

- `text` str

  String of characters to sequentially press 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`.

- `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.

screenshot

Added in: v1.14

Take a screenshot of the element matching the locator.

Usage

Sync Async

```
page.get_by_role("link").screenshot()
```

Disable animations and save screenshot to a file:

Sync Async

```
page.get_by_role("link").screenshot(animations="disabled", path="link.png")
```

Arguments

- `animations` "disabled"|"allow" *(optional)*

  When set to `"disabled"`, stops CSS animations, CSS transitions and Web Animations. Animations get different treatment depending on their duration:

  - finite animations are fast-forwarded to completion, so they'll fire `transitionend` event.
  - infinite animations are canceled to initial state, and then played over after the screenshot.

  Defaults to `"allow"` that leaves animations untouched.

- `caret` "hide"|"initial" *(optional)*

  When set to `"hide"`, screenshot will hide text caret. When set to `"initial"`, text caret behavior will not be changed. Defaults to `"hide"`.

- `mask` List[Locator] *(optional)*

  Specify locators that should be masked when the screenshot is taken. Masked elements will be overlaid with a pink box `#FF00FF` (customized by `mask_color`) that completely covers its bounding box.

- `mask_color` str *(optional)* Added in: v1.35

  Specify the color of the overlay box for masked elements, in CSS color format. Default color is pink `#FF00FF`.

- `omit_background` bool *(optional)*

  Hides default white background and allows capturing screenshots with transparency. Not applicable to `jpeg` images. Defaults to `false`.

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

The file path to save the image to. The screenshot type will be inferred from file extension. If `path` is a relative path, then it is resolved relative to the current working directory. If no path is provided, the image won't be saved to the disk.

- `quality` int *(optional)*

  The quality of the image, between 0-100. Not applicable to `png` images.

- `scale` "css"|"device" *(optional)*

  When set to `"css"`, screenshot will have a single pixel per each css pixel on the page. For high-dpi devices, this will keep screenshots small. Using `"device"` option will produce a single pixel per each device pixel, so screenshots of high-dpi devices will be twice as large or even larger.

  Defaults to `"device"`.

- `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.

- `type` "png"|"jpeg" *(optional)*

  Specify screenshot type, defaults to `png`.

Returns

- bytes

Details

This method captures a screenshot of the page, clipped to the size and position of a particular element matching the locator. If the element is covered by other elements, it will not be actually visible on the screenshot. If the element is a scrollable container, only the currently scrolled content will be visible on the screenshot.

This method waits for the actionability checks, then scrolls element into view before taking a screenshot. If the element is detached from DOM, the method throws an error.

Returns the buffer with the captured screenshot.

scroll_into_view_if_needed

Added in: v1.14

This method waits for actionability checks, then tries to scroll element into view, unless it is completely visible as defined by IntersectionObserver's `ratio`.

Usage

```
locator.scroll_into_view_if_needed()
locator.scroll_into_view_if_needed(**kwargs)
```

Arguments

- `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.

select_option

Added in: v1.14

Selects option or options in `<select>`.

Usage

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

Sync **Async**

```
# single selection matching the value or label
element.select_option("blue")
# single selection matching the label
element.select_option(label="blue")
```

```
# multiple selection for blue, red and second option
element.select_option(value=["red", "green", "blue"])
```

Arguments

- `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`.

- `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]

Details

This method 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.

select_text

Added in: v1.14

This method waits for actionability checks, then focuses the element and selects all its text content.

If the element is inside the `<label>` element that has an associated control, focuses and selects text in the control instead.

Usage

```
locator.select_text()
locator.select_text(**kwargs)
```

Arguments

- `force` bool *(optional)*

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

- `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.

set_checked

Added in: v1.15

Set the state of a checkbox or a radio element.

Usage

Sync | Async

```
page.get_by_role("checkbox").set_checked(True)
```

Arguments

- `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.

- `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.

Details

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

1. Ensure that matched element is a checkbox or a radio input. If not, this method throws.
2. If the element already has the right checked state, 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 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.

set_input_files

Added in: v1.14

Upload file or multiple files into `<input type=file>`.

Usage

Sync | Async

```
# Select one file
page.get_by_label("Upload file").set_input_files('myfile.pdf')

# Select multiple files
page.get_by_label("Upload files").set_input_files(['file1.txt', 'file2.txt'])

# Remove all the selected files
page.get_by_label("Upload file").set_input_files([])
```

```
# Upload buffer from memory
page.get_by_label("Upload file").set_input_files(
    files=[
        {"name": "test.txt", "mimeType": "text/plain", "buffer": b"this is a
test"}
    ],
)
```

Arguments

- `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`.
- `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.

Details

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

tap

Added in: v1.14

Perform a tap gesture on the element matching the locator.

Usage

```
locator.tap()
locator.tap(**kwargs)
```

Arguments

- `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.

- `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.

Details

This method taps the element by performing the following steps:

1. Wait for actionability checks on the element, unless `force` option is set.
2. Scroll the element into view if needed.
3. Use page.touchscreen to tap the center of the element, or the specified `position`.
4. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set.

If the element is detached from the DOM at any moment during the action, this method throws.

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

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

text_content

Added in: v1.14

Returns the `node.textContent`.

> **ASSERTING TEXT**
> If you need to assert text on the page, prefer expect(locator).to_have_text() to avoid flakiness. See assertions guide for more details.

Usage

```
locator.text_content()
locator.text_content(**kwargs)
```

Arguments

- `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

uncheck

Added in: v1.14

Ensure that checkbox or radio element is unchecked.

Usage

Sync | Async

```
page.get_by_role("checkbox").uncheck()
```

Arguments

- `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.

- `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.

Details

This method unchecks the element by performing the following steps:

1. Ensure that element is a checkbox or a radio input. If not, this method throws. If the element is already unchecked, this method returns immediately.
2. Wait for actionability checks on the element, unless `force` option is set.
3. Scroll the element into view if needed.
4. Use page.mouse to click in the center of the element.
5. Wait for initiated navigations to either succeed or fail, unless `no_wait_after` option is set.
6. Ensure that the element is now unchecked. If not, this method throws.

If the element is detached from the DOM at any moment during the action, this method throws.

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

wait_for

Added in: v1.16

Returns when element specified by locator satisfies the `state` option.

If target element already satisfies the condition, the method returns immediately. Otherwise, waits for up to `timeout` milliseconds until the condition is met.

Usage

Sync Async

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

Arguments

- `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.

- `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.

Properties

first

Added in: v1.14

Returns locator to the first matching element.

Usage

```
locator.first
```

Returns

- Locator

last

Added in: v1.14

Returns locator to the last matching element.

Usage

Sync | Async

```
banana = page.get_by_role("listitem").last
```

Returns

- Locator

page

Added in: v1.19

A page this locator belongs to.

Usage

```
locator.page
```

Returns

- Page

Deprecated

element_handle

Added in: v1.14

> ⚠ **DISCOURAGED**
>
> Always prefer using Locators and web assertions over ElementHandles because latter are inherently racy.

Resolves given locator to the first matching DOM element. If there are no matching elements, waits for one. If multiple elements match the locator, throws.

Usage

```
locator.element_handle()
locator.element_handle(**kwargs)
```

Arguments

- `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

- ElementHandle

element_handles

Added in: v1.14

> ⚠ **DISCOURAGED**
>
> Always prefer using Locators and web assertions over ElementHandles because latter are inherently racy.

Resolves given locator to all matching DOM elements. If there are no matching elements, returns an empty list.

Usage

```
locator.element_handles()
```

Returns

- List[ElementHandle]

type

Added in: v1.14

> **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().

Focuses the element, and then sends a `keydown`, `keypress`/`input`, and `keyup` event for each character in the text.

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

Usage

Arguments

- `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`.

- `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.