Guides Auto-waiting

Auto-waiting

Introduction

Playwright performs a range of actionability checks on the elements before making actions to ensure these actions behave as expected. It auto-waits for all the relevant checks to pass and only then performs the requested action. If the required checks do not pass within the given `timeout`, action fails with the `TimeoutError`.

For example, for locator.click(), Playwright will ensure that:

- locator resolves to an exactly one element
- element is Visible
- element is Stable, as in not animating or completed animation
- element Receives Events, as in not obscured by other elements
- element is Enabled

Here is the complete list of actionability checks performed for each action:

Action	Visible	Stable	Receives Events	Enabled	Editable
locator.check()	Yes	Yes	Yes	Yes	-
locator.click()	Yes	Yes	Yes	Yes	-
locator.dblclick()	Yes	Yes	Yes	Yes	-
locator.set_checked()	Yes	Yes	Yes	Yes	-
locator.tap()	Yes	Yes	Yes	Yes	-

Action	Visible	Stable	Receives Events	Enabled	Editable
locator.uncheck()	Yes	Yes	Yes	Yes	-
locator.hover()	Yes	Yes	Yes	-	-
locator.drag_to()	Yes	Yes	Yes	-	-
locator.screenshot()	Yes	Yes	-	-	-
locator.fill()	Yes	-	-	Yes	Yes
locator.clear()	Yes	-	-	Yes	Yes
locator.select_option()	Yes	-	-	Yes	-
locator.select_text()	Yes	-	-	-	-
locator.scroll_into_view_if_needed()	-	Yes	-	-	-
locator.blur()	-	-	-	-	-
locator.dispatch_event()	-	-	-	-	-
locator.focus()	-	-	-	-	-
locator.press()	-	-	-	-	-
locator.press_sequentially()	-	-	-	-	-
locator.set_input_files()	-	-	-	-	-

Forcing actions

Some actions like locator.click() support `force` option that disables non-essential actionability checks, for example passing truthy `force` to locator.click() method will not check that the target element actually receives click events.

Assertions

Playwright includes auto-retrying assertions that remove flakiness by waiting until the condition is met, similarly to auto-waiting before actions.

Assertion	Description
expect(locator).to_be_attached()	Element is attached
expect(locator).to_be_checked()	Checkbox is checked
expect(locator).to_be_disabled()	Element is disabled
expect(locator).to_be_editable()	Element is editable
expect(locator).to_be_empty()	Container is empty
expect(locator).to_be_enabled()	Element is enabled
expect(locator).to_be_focused()	Element is focused
expect(locator).to_be_hidden()	Element is not visible
expect(locator).to_be_in_viewport()	Element intersects viewport
expect(locator).to_be_visible()	Element is visible
expect(locator).to_contain_text()	Element contains text
expect(locator).to_have_attribute()	Element has a DOM attribute
expect(locator).to_have_class()	Element has a class property

Assertion	Description
expect(locator).to_have_count()	List has exact number of children
expect(locator).to_have_css()	Element has CSS property
expect(locator).to_have_id()	Element has an ID
expect(locator).to_have_js_property()	Element has a JavaScript property
expect(locator).to_have_text()	Element matches text
expect(locator).to_have_value()	Input has a value
expect(locator).to_have_values()	Select has options selected
expect(page).to_have_title()	Page has a title
expect(page).to_have_url()	Page has a URL
expect(response).to_be_ok()	Response has an OK status

Learn more in the assertions guide.

Visible

Element is considered visible when it has non-empty bounding box and does not have `visibility:hidden` computed style.

Note that according to this definition:

- Elements of zero size **are not** considered visible.
- Elements with `display:none` **are not** considered visible.
- Elements with `opacity:0` **are** considered visible.

Stable

Element is considered stable when it has maintained the same bounding box for at least two consecutive animation frames.

Enabled

Element is considered enabled unless it is a `<button>`, `<select>`, `<input>` or `<textarea>` with a `disabled` property.

Editable

Element is considered editable when it is enabled and does not have `readonly` property set.

Receives Events

Element is considered receiving pointer events when it is the hit target of the pointer event at the action point. For example, when clicking at the point `(10;10)`, Playwright checks whether some other element (usually an overlay) will instead capture the click at `(10;10)`.

For example, consider a scenario where Playwright will click `Sign Up` button regardless of when the locator.click() call was made:

- page is checking that user name is unique and `Sign Up` button is disabled;
- after checking with the server, the disabled `Sign Up` button is replaced with another one that is now enabled.