# 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 page.click(), Playwright will ensure that:

- element is Attached to the DOM
- 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 | Attached | Visible | Stable | Receives Events | Enabled | Editable |
|---|---|---|---|---|---|---|
| check | Yes | Yes | Yes | Yes | Yes | - |
| click | Yes | Yes | Yes | Yes | Yes | - |
| dblclick | Yes | Yes | Yes | Yes | Yes | - |
| setChecked | Yes | Yes | Yes | Yes | Yes | - |
| tap | Yes | Yes | Yes | Yes | Yes | - |
| uncheck | Yes | Yes | Yes | Yes | Yes | - |
| hover | Yes | Yes | Yes | Yes | - | - |

| Action | Attached | Visible | Stable | Receives Events | Enabled | Editable |
|---|---|---|---|---|---|---|
| scrollIntoViewIfNeeded | Yes | - | Yes | - | - | - |
| screenshot | Yes | Yes | Yes | - | - | - |
| fill | Yes | Yes | - | - | Yes | Yes |
| selectText | Yes | Yes | - | - | - | - |
| dispatchEvent | Yes | - | - | - | - | - |
| focus | Yes | - | - | - | - | - |
| getAttribute | Yes | - | - | - | - | - |
| innerText | Yes | - | - | - | - | - |
| innerHTML | Yes | - | - | - | - | - |
| press | Yes | - | - | - | - | - |
| setInputFiles | Yes | - | - | - | - | - |
| selectOption | Yes | Yes | - | - | Yes | - |
| textContent | Yes | - | - | - | - | - |
| type | Yes | - | - | - | - | - |

## Forcing actions

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

## Assertions

You can check the actionability state of the element using one of the following methods as well. This is typically not necessary, but it helps writing assertive tests that ensure that after certain actions, elements reach actionable state:

- element_handle.is_checked()
- element_handle.is_disabled()
- element_handle.is_editable()
- element_handle.is_enabled()
- element_handle.is_hidden()
- element_handle.is_visible()
- page.is_checked()
- page.is_disabled()
- page.is_editable()
- page.is_enabled()
- page.is_hidden()
- page.is_visible()
- locator.is_checked()
- locator.is_disabled()
- locator.is_editable()
- locator.is_enabled()
- locator.is_hidden()
- locator.is_visible()

## Attached

Element is considered attached when it is connected to a Document or a ShadowRoot.

## Visible

Element is considered visible when it has non-empty bounding box and does not have `visibility:hidden` computed style. Note that elements of zero size or with `display:none` are not 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 page.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.