Navigations

Introduction

Playwright can navigate to URLs and handle navigations caused by the page interactions.

Basic navigation

Simplest form of a navigation is opening a URL:

Sync Async

```
# Navigate the page
page.goto("https://example.com")
```

The code above loads the page and waits for the web page to fire the load event. The load event is fired when the whole page has loaded, including all dependent resources such as stylesheets, scripts, iframes, and images.

NOTE

If the page does a client-side redirect before `load`, page.goto() will wait for the redirected page to fire the `load` event.

When is the page loaded?

Modern pages perform numerous activities after the `load` event was fired. They fetch data lazily, populate UI, load expensive resources, scripts and styles after the `load` event was fired. There is no way to tell that the page is `loaded`, it depends on the page, framework, etc. So when can you start interacting with it?

In Playwright you can interact with the page at any moment. It will automatically wait for the target elements to become actionable.

Sync Async

```
# Navigate and click element
# Click will auto-wait for the element
page.goto("https://example.com")
page.get_by_text("example domain").click()
```

For the scenario above, Playwright will wait for the text to become visible, will wait for the rest of the actionability checks to pass for that element, and will click it.

Playwright operates as a very fast user - the moment it sees the button, it clicks it. In the general case, you don't need to worry about whether all the resources loaded, etc.

Hydration

At some point in time, you'll stumble upon a use case where Playwright performs an action, but nothing seemingly happens. Or you enter some text into the input field and it will disappear. The most probable reason behind that is a poor page hydration.

When page is hydrated, first, a static version of the page is sent to the browser. Then the dynamic part is sent and the page becomes "live". As a very fast user, Playwright will start interacting with the page the moment it sees it. And if the button on a page is enabled, but the listeners have not yet been added, Playwright will do its job, but the click won't have any effect.

A simple way to verify if your page suffers from a poor hydration is to open Chrome DevTools, pick "Slow 3G" network emulation in the Network panel and reload the page. Once you see the element of interest, interact with it. You'll see that the button clicks will be ignored and the entered text will be reset by the subsequent page load code. The right fix for this issue is to make sure that all the interactive controls are disabled until after the hydration, when the page is fully functional.

Waiting for navigation

Clicking an element could trigger multiple navigations. In these cases, it is recommended to explicitly page.wait_for_url() to a specific url.

Sync Async

```
page.get_by_text("Click me").click()
page.wait_for_url("**/login")
```

Navigation events

Playwright splits the process of showing a new document in a page into **navigation** and **loading**.

Navigation starts by changing the page URL or by interacting with the page (e.g., clicking a link). The navigation intent may be canceled, for example, on hitting an unresolved DNS address or transformed into a file download.

Navigation is committed when the response headers have been parsed and session history is updated. Only after the navigation succeeds (is committed), the page starts **loading** the document.

Loading covers getting the remaining response body over the network, parsing, executing the scripts and firing load events:

- page.url is set to the new url
- document content is loaded over network and parsed
- page.on("domcontentloaded") event is fired
- page executes some scripts and loads resources like stylesheets and images
- page.on("load") event is fired
- page executes dynamically loaded scripts