Guides Dialogs

Dialogs

Introduction

Playwright can interact with the web page dialogs such as `alert`, `confirm`, `prompt` as well as `beforeunload` confirmation.

alert(), confirm(), prompt() dialogs

By default, dialogs are auto-dismissed by Playwright, so you don't have to handle them. However, you can register a dialog handler before the action that triggers the dialog to either dialog.accept() or dialog.dismiss() it.

Sync Async

```
page.on("dialog", lambda dialog: dialog.accept())
page.get_by_role("button").click()
```

> **NOTE**
>
> page.on("dialog") listener **must handle** the dialog. Otherwise your action will stall, be it locator.click() or something else. That's because dialogs in Web are modals and therefore block further page execution until they are handled.

As a result, the following snippet will never resolve:

> **WARNING**
>
> WRONG!

Sync Async

```
page.on("dialog", lambda dialog: print(dialog.message))
page.get_by_role("button").click() # Will hang here
```

> **NOTE**
>
> If there is no listener for page.on("dialog"), all dialogs are automatically dismissed.

beforeunload dialog

When page.close() is invoked with the truthy `run_before_unload` value, the page runs its unload handlers. This is the only case when page.close() does not wait for the page to actually close, because it might be that the page stays open in the end of the operation.

You can register a dialog handler to handle the `beforeunload` dialog yourself:

Sync Async

```
def handle_dialog(dialog):
    assert dialog.type == 'beforeunload'
    dialog.dismiss()

page.on('dialog', lambda: handle_dialog)
page.close(run_before_unload=True)
```