FRAMES

INTERACTING WITH FRAMES

Handling frames is crucial for interacting with web pages that contain multiple documents, like iframes.

By Index: Accessing a frame by its index when the order is known.

```
const frameByIndex = page.frame({ index: 0 });
```

By Name: Targeting a frame by its `name` attribute.

```
const frameByName = page.frame({ name: 'frameName' });
```

By URL: Finding a frame by its source URL.

```
const frameByUrl = page.frame({ url: /example.com\/path/ });
```

By ID using frameLocator: Utilizing `frameLocator` to select frames by the ID of the iframe element.

```
const frameById = page.frameLocator('#frameID').frame();
```

Listing All Frames: Retrieving an array of all frames on the page.

```
const allFrames = page.frames();
```

Main Frame: Accessing the main content frame of the page.

```
const mainFrame = page.mainFrame();
```

WINDOWS

HANDLING MULTIPLE WINDOWS (POP-UPS)

Managing multiple windows or tabs is essential for comprehensive testing.

Sequential Approach: Handling new windows as they open, one at a time.

```
const newPage = await Promise.all([
  context.waitForEvent('page'),
  page.click('a[target="_blank"]') // Assuming this opens a new tab
]);
```

Using Promise.all for Multiple Windows: Coordinating the opening of several windows simultaneously.

```
const [window1, window2] = await Promise.all([
  context.waitForEvent('page'),
  context.waitForEvent('page'), // Expecting two new pages
  page.click('#open-first-window'),
  page.click('#open-second-window')
]);
```

waitForEvent('page'): Awaiting the opening of a new page following an action.

```
const newPage = await context.waitForEvent('page');
page.click('#open-new-window'); // Trigger new window
```

FILES UPLOAD

Simulating user file uploads through input fields.

setInputFiles: Specifying files to upload via an input element.

```
await page.setInputFiles('#upload-selector', 'path/to/file.txt');
```

setFiles: Setting files when a file chooser is triggered programmatically by the app.

```
const [fileChooser] = await Promise.all([
  page.waitForEvent('filechooser'),
  page.click('#upload-btn') // Assuming this triggers the file chooser
]);
await fileChooser.setFiles('path/to/file.txt');
```

FILES DOWNLOAD

Handling the download of files and saving them locally.

Download Event and suggestedFilename: Managing the download process and accessing the suggested filename.

```
const [download] = await Promise.all([
  page.waitForEvent('download'),
  page.click('#download-link') // Triggers the download
]);
console.log('Suggested Filename:', download.suggestedFilename());
```

saveAs: Saving the downloaded file to a specified path.

```
const path = await download.path();
await download.saveAs('/path/to/save/file');
```