**Handling Dialogs:**

**page.on(event, handler)**

* *Accept an alert:*

await page.on("dialog", (dialog) => dialog.accept());

* *Dismiss an alert:*

*await page.on("dialog", (dialog) => dialog.dismiss());*

**page.once(event, handler):**

page.once('dialog', async (dialog) => {

if (dialog.type() === 'prompt') {

await dialog.accept('My answer');

}

});

|  |  |
| --- | --- |
| *Page.on* | *Page.once* |
| **listen for an event continuously** | **listens for an event only once**. |

**Handling iFrames: (inlineframes)**

* **page.frames():** Returns an array of all frames on the page.
* **page.frameLocator**(nameOrSelector): Returns a **frame locator** based on its name or CSS selector.
* **frame.locator(selector):** Creates a **locator** within the frame to find elements.

**Handling Multipage:**

**Using event listener:**

Const windowPromise=page.**waitForEvent(‘page’)**;

await page.getByText('open the popup').click();

const popup = await popupPromise;

**Using Promise.all:**

const [window] = await Promise.all([

context.waitForEvent('page'),

page.click(selector),

])

**Note**: page[index].**bringTofront()**-> to focus on the specific page when multiple page context

**FileUpload:**

**Method:1 (Direct)**

page.**setInputFiles**(selectors,’/path/to/your/file.txt');

**Method:2 (Using event listener)**

const uploadFile = page.**waitForEvent**('filechooser');

await page.click(selector)

const file = await uploadFile;

await file.**setFiles**("filepath");;

**FileDownload:**

**Using Event listener:**

**Method:1**

const fileDownload=page.**waitForEvent**('download');

await page.click(selectors);

const filePath = await fileToDownload;

await filePath.saveAs("downloaded files/"+fileDownload.suggestedFilename());

**Method:2**

page.on('download', async (download) => {

await download.**saveAs**("./downloads/" + download.suggestedFilename());

});

await page.click(selectors);

**Reading Test data:**

**Using CSV file :**

**Install csv parser: npm install csv-parser**

const testData= **parse**(fs.readFileSync(path.join(\_\_dirname,"loginData.csv")),{

columns:true, //to ignore the headers

skip\_empty\_lines:true

})

for(const data of testData){

test(`Reading data from CSV file ${data.Testcase\_id}`, async({page})=>{

})

**Using json data:**

const testData=**JSON.parse**(fs.readFileSync(path.join(\_\_dirname,"credentials.json"),'utf-8'));

for(const data of testData){

test(`Reading data from json file ${data.testcase\_id}`, async({page})=>{})

**Using env data:**

**Install dotenv: npm install dotenv**

dotenv.config();

test('Read data from env file',async({page})=>{

const url = **process.env.QA\_URL** as string

await page.goto(url)})

**SessionStorage:**

await page.context().

**storageState**({path:"filenpath"})

**Using test function:**

test.use({storageState:"filePath"});

test("Using sessionStorage",async({page})=>{

await page.goto("homePage")

})