

# Playwright Keyboard Methods

## Keyboard actions

Before using keyboard methods, you typically get the keyboard object via the page object:

```
const keyboard = page.keyboard;
```

### 1. keyboard.down()

- Simulates pressing down a key without releasing it.
- Useful for key combinations (e.g., Ctrl + A).

**Example:**

```
await keyboard.down('Control');  
await keyboard.press('A');  
await keyboard.up('Control');
```

### 2. keyboard.up()

- Releases a key that was pressed using down().
- Should be paired with down() for combinations.

**Example:**

```
await keyboard.down('Shift');  
// some other key actions  
await keyboard.up('Shift');
```

### 3. keyboard.press()

- Presses and releases a key instantly.
- Can simulate both single keys and key combinations (e.g., Control+A, Enter, etc.).

**Example:**

```
await keyboard.press('Enter');  
await keyboard.press('Control+A');
```

#### 4. keyboard.type()

- Types full strings one character at a time.

##### Example:

```
await keyboard.type('Hello World!');
```

#### 5. keyboard.insertText(text: string)

- Directly inserts text into a focused input field.
- Does **not** trigger keydown, keyup, or keypress events.
- Faster and more reliable when event simulation is not needed.

##### Example:

```
await keyboard.insertText('Fast text input!');
```

### Uploading Files in Playwright

#### 1. setInputFiles() – Uploading Files in Playwright

##### Purpose:

Used to upload one or more files into an <input type="file"> element on a web page.

##### Syntax:

```
await page.locator('input[type="file"]').setInputFiles('path/to/file');
```

##### Example – Uploading a Single File:

```
await page.goto('https://example.com/upload');  
await page.locator('input[type="file"]').setInputFiles('tests/files/sample1.txt');
```

##### Example – Uploading Multiple Files:

```
await page.goto('https://example.com/upload');  
await page.locator('input[type="file"]').setInputFiles([  
  'tests/files/sample1.txt',  
  'tests/files/sample2.txt'  
]);
```

**Notes:**

- The path should be relative to your project root or absolute.
- It simulates user file uploads and is useful for testing file upload forms.
- Playwright uploads files from the local filesystem into the browser context.

**Downloading Files in Playwright****Steps to Handle File Download:**

1. Start waiting for the download event using `page.waitForEvent('download')`.
2. Trigger the file download (like clicking a download link or button).
3. Save or validate the downloaded file if needed.

**Example:**

```
// Navigate to the download page
await page.goto('https://example.com/download');

// Start waiting for the download before clicking the download link
const [ download ] = await Promise.all([
  page.waitForEvent('download'), // 1. Wait for download to begin
  page.locator('#txtDownloadLink').click() // 2. Click to trigger download
]);

// Optional: Save the file to a specific path
await download.saveAs('downloads/sample.txt');
```

**Notes:**

- Always wrap the `click()` action and `waitForEvent('download')` inside `Promise.all` to avoid race conditions.

**Recap:**

Function	Purpose
<b>setInputFiles()</b>	Upload one or multiple files to the browser
<b>waitForEvent('download')</b>	Wait for file download to begin
<b>saveAs()</b>	Save the downloaded file to a specific location