**📌 What is Playwright?**

**Playwright** is an open-source automation library developed by **Microsoft**. It allows you to **automate modern web applications** with high speed and reliability, supporting **multiple browsers** (Chromium, Firefox, and WebKit) and **multiple languages** (Python, JavaScript/Node.js, Java, .NET).

It is especially useful for:

* Web scraping of **JavaScript-heavy websites**
* **End-to-end (E2E) testing**
* Browser automation

**⚙️ Key Features**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Cross-browser** | Supports Chromium (Chrome), Firefox, and WebKit (Safari engine) |
| **Headless/Headful modes** | Run browsers without UI (headless) or with UI (headful) |
| **Fast and Reliable** | Runs scripts in parallel with auto-wait and retry features |
| **Handles JS-heavy pages** | Works well on SPAs (Single Page Applications) rendered via JavaScript |
| **Multiple tabs/pages** | Automate multi-tab or multi-window scenarios |
| **Emulate devices** | Simulate different screen sizes, geolocation, user agents, etc. |

**🧪 Basic Playwright Script (Python)**

This script launches a browser, navigates to a site, takes a screenshot, and extracts the page title.

from playwright.sync\_api import sync\_playwright

with sync\_playwright() as p:

browser = p.chromium.launch(headless=True)

page = browser.new\_page()

page.goto("https://example.com")

# Take screenshot

page.screenshot(path="screenshot.png")

# Extract title

title = page.title()

print("Page Title:", title)

browser.close()

**📊 Scraping Example with Playwright**

Let’s scrape blog post titles from a JavaScript-rendered page.

from playwright.sync\_api import sync\_playwright

with sync\_playwright() as p:

browser = p.chromium.launch()

page = browser.new\_page()

page.goto("https://quotes.toscrape.com/js") # JS-rendered quotes site

# Wait for the quotes to load

page.wait\_for\_selector(".quote")

# Extract data

quotes = page.query\_selector\_all(".quote")

for quote in quotes:

text = quote.query\_selector(".text").inner\_text()

author = quote.query\_selector(".author").inner\_text()

print(f"{text} - {author}")

browser.close()

**Scraping vs Testing Use**

|  |  |
| --- | --- |
| **Use Case** | **Purpose** |
| Scraping | Extract live content even if rendered via JS |
| Testing | Automate testing for web UI interactions |

**🧠 Why Use Playwright for Scraping?**

| **Feature** | **Benefit** |
| --- | --- |
| JavaScript rendering | Handles dynamic sites (unlike requests/BS4) |
| Fast execution | Faster than Selenium in many use cases |
| Multiple contexts | Simulate multiple users/sessions |
| Screenshot/Recording | Useful for debugging scraping sessions |
| Stealth Mode | Can bypass anti-bot protections (with some tuning) |