Regular expressions (regex) are powerful tools for **matching patterns** within strings, and they are highly useful in testing frameworks like **Playwright** for **validating titles, URLs, text content**, and many other elements on a webpage. Let’s dive into how **regular expressions** are used in Playwright, specifically for **validating titles** and other applications.

**What is a Regular Expression?**

* A **regular expression** (regex) is a **sequence of characters** that define a **search pattern**.
* It is typically used for **string matching** or **searching**, allowing you to look for a particular pattern instead of a specific exact value.
* Regex can be used to **match dynamic content** and **reduce test fragility** when the content changes slightly but follows a general pattern.

**Using Regular Expressions in Playwright for Title Validation**

In Playwright, you can use **regular expressions** to validate a page's title using methods like page.title() or expect(page).toHaveTitle(). When using regex, the validation is more **flexible**, allowing you to match the title even if parts of it are **dynamic** or **variable**.

Here is an example:



**Explanation**

* **expect(page).toHaveTitle()**: This method is used to **assert** that the page’s title matches a given value or **pattern**.
* **/.\*LoginPage Practice/**: This is a **regular expression** used to match the title.
  + The **slashes (/)** indicate the **start and end** of the regex.
  + **.\*** is a **wildcard** that matches **any character** (except a newline) **zero or more times**.
  + **LoginPage Practice** is the **literal string** you want to find.

This regex will **match any title** that contains the words **"LoginPage Practice"**, no matter what comes before it.

**Why Use Regular Expressions for Title Validation?**

* **Dynamic Titles**: Sometimes page titles include **dynamic values**, such as usernames, counters, or other variables that change. Using a regex allows you to match **only the constant parts** of the title.
* **Partial Matches**: You may only be interested in a **portion of the title**. Regex helps you validate just that part without needing an exact match.
* **Flexible Validation**: It makes your tests more **robust** and **less fragile**, especially if the format or prefix/suffix of a title can change but the core part remains the same.

**Common Regular Expressions in Playwright**

Here are some common **regex patterns** you can use to validate titles and other elements:

1. **Match a Title that Starts with a Specific Word**

await expect(page).toHaveTitle(/^LoginPage/);

* + **^** matches the **start** of the string.
  + **LoginPage** must be at the beginning of the title.

1. **Match a Title that Ends with a Specific Word**

await expect(page).toHaveTitle(/Practice$/);

* + **Practice$** indicates that the title should **end** with "Practice".
  + **$** matches the **end** of the string.

1. **Case-Insensitive Matching**

await expect(page).toHaveTitle(/loginpage practice/i);

* + The **i** flag makes the regex **case-insensitive**, meaning it will match regardless of whether the letters are uppercase or lowercase.

1. **Optional Characters or Words**

await expect(page).toHaveTitle(/LoginPage (Practice|Demo)/);

* + The **(Practice|Demo)** allows for either "Practice" **or** "Demo" to match. This is called **alternation**.
  + This is useful when the title could have **variations**.

1. **Match Any Character (Wildcard)**

await expect(page).toHaveTitle(/.\*LoginPage Practice/);

* + **.\*** matches **any number** of characters.
  + This is useful when you **don’t care** what comes before or after a specific word.

**Examples of Title Validation with Regex**

Consider a website that has a **dynamic title**. For example:

* **Title of a webpage**: "Welcome to LoginPage Practice | Rahul Shetty Academy"

**Exact Title Matching (Static)**

If the title is **always the same**, you could use an exact match:

javascript

Copy code

await expect(page).toHaveTitle('Welcome to LoginPage Practice | Rahul Shetty Academy');

However, this would fail if **anything changes** (e.g., a minor update adds a word or punctuation).

**Flexible Title Matching (Using Regex)**

For **flexibility**, you can use a **regex** to match only the part that you expect to be **constant**:

javascript

Copy code

await expect(page).toHaveTitle(/.\*LoginPage Practice/);

This matches **any title** that contains "LoginPage Practice", regardless of what comes before or after.

**Benefits of Using Regular Expressions in Testing**

1. **Reduced Flakiness**: Regex helps to create more **reliable tests** by accounting for minor variations, reducing the chance of false failures.
2. **Ease of Maintenance**: With dynamic pages, you don’t have to keep updating your tests for every small change in the title.
3. **Flexible Assertions**: Validating **partial matches** helps when only a specific part of the page’s title or content matters for a test.

**Other Uses of Regular Expressions in Playwright**

* **URL Matching**: You can also use regex to **validate URLs** with expect(page).toHaveURL().

await expect(page).toHaveURL(/.\*dashboard/);

This ensures you are on a page that contains "dashboard" in its URL.

* **Text Matching**: Use regex to **validate text** content on a page or within an element.

const element = await page.locator('.message');

await expect(element).toHaveText(/Success.\*/);

This will match any text that **starts with "Success"**.

**Summary for Interviews**

* **Regular expressions** allow you to **validate dynamic content** by matching patterns rather than fixed values.
* They help you create **robust and flexible tests**, especially when titles, URLs, or page content are subject to **slight variations**.
* In Playwright:
  + Use expect(page).toHaveTitle() with regex to **validate titles** dynamically.
  + Common regex symbols:
    - **^** for start of string, **$** for end of string.
    - **.\*** for matching **any number** of characters.
    - **|** for **alternation** (matching multiple possibilities).
    - **i** flag for **case-insensitive** matching.

Regular expressions are an essential part of making your **Playwright tests** both **flexible** and **powerful**, especially in **UI testing** where minor changes can often cause fixed assertions to break.

**.\* ka matlab kya hai?**

* **.** ka matlab hai "koi bhi ek character" - matlab **koi bhi akshar**, **number**, ya **special character** ho sakta hai (bas **newline** ko chhodkar).
* **\*** ka matlab hai "zero ya zyada baar" - iska matlab hai ki jo bhi character **.** se match ho raha hai, wo **zero se kitni bhi baar** aa sakta hai.

Toh jab hum **.\*** likhte hain, iska matlab hota hai:

* Koi bhi **character**, jo **kitni bhi baar** aa sakta hai. Yeh **khaali** bhi ho sakta hai, ya phir ismein **bahut saare characters** bhi ho sakte hain.

**Example se samjhein**

Maan lo humein **page ka title** check karna hai ki usmein **"LoginPage Practice"** shabd aa raha hai ya nahi.

Hum yeh check kar sakte hain:

await expect(page).toHaveTitle(/.\*LoginPage Practice/);

Yahan **.\*** ka matlab hai ki:

* **"LoginPage Practice"** se pehle koi bhi character ho sakta hai, aur wo **kitni bhi baar** aa sakta hai.
* Example ke liye:
  + **"Welcome to LoginPage Practice"** - yeh match ho jayega.
  + **"Something LoginPage Practice"** - yeh bhi match ho jayega.
  + **"LoginPage Practice"** - yeh bhi match ho jayega.

Matlab **.\*** use karke hum ensure kar rahe hain ki **"LoginPage Practice"** se pehle kuch bhi ho sakta hai, aur yeh sab match ho jayega.

Bilkul, agar hum **LoginPage Practice** ke baad bhi kuch content ko match karna chahte hain, toh **.\*** ka use aage bhi kiya ja sakta hai.

**.\* Before and After LoginPage Practice**

* Agar hum chahte hain ki **"LoginPage Practice"** se **pehle** aur **baad** mein kuch bhi ho sakta hai, toh hum regex ko is tarah likh sakte hain:

await expect(page).toHaveTitle(/.\*LoginPage Practice.\*/);

**Explanation:**

* **.\*LoginPage Practice.\***:
  + **Pehle wala .\***: Iska matlab hai ki **LoginPage Practice** se pehle koi bhi character (jo zero ya zyada baar aa sakte hain) ho sakte hain.
  + **LoginPage Practice**: Yeh woh specific text hai jisko hum match karna chahte hain.
  + **Baad wala .\***: Yeh ensure karta hai ki **LoginPage Practice** ke **baad** bhi kuch bhi ho sakta hai (koi bhi character zero ya zyada baar).

**Example Se Samjhein:**

* **Title**: "Welcome to LoginPage Practice | Rahul Shetty Academy"
  + Yeh title **match ho jayega** kyunki:
    - **LoginPage Practice** se pehle kuch bhi ho sakta hai (.\*) - yahaan "Welcome to" hai.
    - **LoginPage Practice** match ho raha hai.
    - **LoginPage Practice** ke baad kuch bhi ho sakta hai (.\*) - yahaan " | Rahul Shetty Academy" hai.
* **Title**: "Something Else - LoginPage Practice - More Content"
  + Yeh bhi **match ho jayega** kyunki:
    - **.\*** se pehle kuch bhi ho sakta hai ("Something Else - ").
    - **LoginPage Practice** match ho raha hai.
    - Baad mein **.\*** ke saath kuch bhi content ho sakta hai (" - More Content").

**Summary**

* **.\*LoginPage Practice.\***:
  + **Pehle .\***: Pehle kuch bhi character ho sakta hai.
  + **LoginPage Practice**: Specific text jo match hona chahiye.
  + **Baad ka .\***: Baad mein bhi kuch bhi character ho sakta hai.

Is tarah se hum ensure kar sakte hain ki humara title **LoginPage Practice** ko contain karta ho, chahe uske pehle ya baad kuch bhi ho. Yeh Playwright mein use karne se **dynamic content** ko handle karne mein madad milti hai, aur aapke tests **zyada robust** bante hain.