Playwright Locators

In Playwright, **locators** are a core concept that power its **auto-waiting** and **retry-ability** features.
Simply put, a locator is a way to **identify element(s)** on the page at any given time, enabling stable and reliable tests.

Quick Reference: Recommended Built-in Locators

Locator Method	Use Case
page.getByRole()	Locate elements by **accessibility roles** like button, checkbox, heading, etc.
page.getByText()	Locate by **visible text** content.
page.getByLabel()	Locate **form controls** using associated label text.
page.getByPlaceholder()	Locate inputs via **placeholder** text.
page.getByAltText()	Locate images by their **alternative text** (alt attribute).
page.getByTitle()	Locate elements by their **title attribute**.
page.getByTestId()	Locate by a **custom data attribute** like data-testid.

Locating Elements

🔹 Locate by Role

page.getByRole() identifies elements based on **how users and assistive technologies perceive them**, using **ARIA roles** and **accessible names**.

📄 Example DOM:

```
<h3>Sign up</h3>
<label><input type="checkbox" /> Subscribe</label><br/>
<button>Submit</button>
```

📌 Test Actions:

```
await expect(page.getByRole('heading', { name: 'Sign up' })).toBeVisible();
await page.getByRole('checkbox', { name: 'Subscribe' }).check();
await page.getByRole('button', { name: /submit/i }).click();
```

✅ **When to use:** Prefer for interactive elements like **buttons, checkboxes, links, lists, headings, tables**, etc.

🔹 Locate by Label

page.getByLabel() allows you to locate a form control using its **associated label text**.

📄 Example DOM:

```
<label>Password <input type="password" /></label>
```

📌 Test Action:

```
await page.getByLabel('Password').fill('secret');
```

✅ **When to use:** Ideal for **form fields** with visible labels.

🔹 Locate by Placeholder

page.getByPlaceholder() finds elements with a given **placeholder** text.

📄 Example DOM:

```
<input type="email" placeholder="name@example.com" />
```

📌 Test Action:

```
await page.getByPlaceholder('name@example.com').fill('playwright@microsoft.com');
```

✅ **When to use:** Best for **inputs** without a label but having a **placeholder**.

🔹 Locate by Text

page.getByText() finds elements based on their **visible text content**.

📄 Example DOM:

```
<span>Welcome, John</span>
```

📌 Test Actions:

```
await expect(page.getByText('Welcome, John')).toBeVisible(); // Partial match
await expect(page.getByText('Welcome, John', { exact: true })).toBeVisible(); // Exact match
```

✅ **When to use:**

- Use for **non-interactive elements** like , , <p>.
- For **interactive elements**, prefer **role-based locators**.

🔹 Locate by Alt Text

page.getByAltText() identifies **images** (and similar elements) based on the **alt attribute**.

📄 Example DOM:

```
<img alt="playwright logo" src="/img/playwright-logo.svg" width="100" />
```

📌 Test Action:

```
await page.getByAltText('playwright logo').click();
```

✅ **When to use:**
When working with **img** and **area** elements that use an **alt** text.

🔹 Locate by Title

page.getByTitle() locates elements based on their **title attribute**.

📄 Example DOM:

```
<span title="Issues count">25 issues</span>
```

📌 Test Action:

```
await expect(page.getByTitle('Issues count')).toHaveText('25 issues');
```

✅ **When to use:**
When your element has a meaningful **title** attribute.

🔹 Locate by Test ID

page.getByTestId() enables locating elements using a **data-testid** attribute (or custom attribute if configured).

📄 Example DOM:

```
<button data-testid="directions">Itinéraire</button>
```

📌 Test Action:

```
await page.getByTestId('directions').click();
```

✅ **When to use:**

- When text or role-based locators are **unstable** or **not suitable**.
- Great for **resilient and stable** automated tests.

Note:
Test IDs are not user-facing. Prefer role or text locators when they are semantically important to users.

Other Locator Options

🔹 Locate by CSS or XPath

Use page.locator() for **CSS** or **XPath** selectors when absolutely necessary.

📌 Test Actions:

```
await page.locator('css=button').click();
await page.locator('xpath=//button').click();

// Auto-detect:
await page.locator('button').click();
await page.locator('//button').click();
```

❌ **Warning:**

- CSS/XPath selectors are **fragile** and **break easily** with DOM changes.
- Avoid using long or deeply nested CSS/XPath chains:

```
await page.locator('#tsf > div:nth-child(2) > div > input').click();
```

✅ **When to use:**
As a last resort when built-in locators are insufficient.

Important Behavior: Strictness

- **Locators are strict by default.**
- If **multiple elements match**, Playwright throws an **exception**.

📌 Example — Throws Error:

```
await page.getByRole('button').click();
```

📌 Example — Works fine with multiple elements:

```
await page.getByRole('button').count();
```

Final Tips

- Favor **role**, **text**, **label**, and **placeholder** locators first.
- Avoid overusing **CSS** or **XPath** locators unless unavoidable.