# Solving Locator Instability in Automation Tests

As an automation developer, one of the most common sources of flaky tests is unstable or non-unique selectors. This guide outlines practical solutions to ensure stable element targeting in your automated tests.

## Your Core Problem

- Lack of consistent `id` or `data-\*` attributes.  
- Dynamic class names or DOM structure.  
- Elements that appear/disappear based on app state.  
- Lists, modals, and complex nested structures.

## Battle-Tested Solutions

### 1. Use Custom Data Attributes (`data-testid`, `data-qa`)

If you control the frontend, use custom attributes to mark elements:  
  
<button data-testid="submit-btn">Submit</button>  
  
This provides highly stable and readable selectors.

### 2. Attach Temporary Locators at Runtime

If you don’t control the frontend, inject a temp attribute:  
  
await page.evaluate(() => {  
 const element = document.querySelector('div[placeholder="login"]');  
 if (element) element.setAttribute('data-temp-id', 'login\_input');  
});  
await page.locator('[data-temp-id="login\_input"]').fill('myuser');

### 3. AI or Heuristic-Based Locator Builders

Build selectors using visible text or stable attributes:  
  
const locator = page.getByRole('button', { name: 'Submit' });  
  
Or create a helper function that prioritizes certain attributes.

### 4. Visual/AI-Based Locator Recovery

Use visual tools or Playwright MCP to select elements by intention when the DOM structure is too unstable to rely on static selectors.

## Pro Trick: Hybrid Approach

Combine static, dynamic, and injected selectors into a robust fallback system:  
  
async function getStableLocator(page, hintText, fallbackSelector) {  
 try {  
 return page.locator(`[data-testid="${hintText}"]`);  
 } catch {  
 await page.evaluate(() => {  
 const el = document.querySelector(fallbackSelector);  
 if (el) el.setAttribute('data-testid', hintText);  
 });  
 return page.locator(`[data-testid="${hintText}"]`);  
 }  
}

## Summary Table

|  |  |  |  |
| --- | --- | --- | --- |
| Strategy | Use When | Stability | Dev Effort |
| data-testid | You control frontend | Ultra | Medium |
| Runtime DOM patch | You don’t control frontend | High | Low |
| Smart selector builder | Dynamic apps | Medium | Medium |
| AI fallback (MCP etc.) | Extreme cases | High | High |