



# Web Automation Fundamentals

By Farah Yulianti & Digital Skola  
Desember - 2024



# Table of Content

1. Web UI Automation Intro
2. Selenium Introduction
3. UI Locators
4. Selenium Web UI Test



# Web UI Automation Intro



# Jenis Web UI Automation Test

- **In Process automation** (Cypress)  
Test script dieksekusi didalam proses atau lingkungan yang sama  
berbarengan dengan aplikasi
- **Out process automation** (WebDriver Protocol, Chrome Devtools Protocol)  
Test script dieksekusi dalam lingkungan terpisah, membutuhkan perantara  
dan aturan(protocol) untuk berkomunikasi dengan browser



# **Web UI automation test type**

Beberapa jenis tes yang bisa dilakukan terhadap UI dari sebuah website antara lain:

1. Functionality test
2. Visual test
3. Performance test (rendering)
4. Accessibility testing

# Web UI automation test tools





# Selenium introduction

# Apa itu Selenium



Selenium adalah framework pengujian UI (User Interface) yang juga bersifat Open Source dan telah digunakan secara luas dalam industri pengembangan perangkat lunak. Selenium mendukung pengujian di berbagai browser populer seperti Chrome, Firefox, Internet Explorer, dan Safari.

Berbeda dengan Playwright yang menggunakan WebSocket Protocol, Selenium beroperasi dengan WebDriver, sebuah API standar yang berkomunikasi dengan browser melalui protocol yang disebut JSON Wire Protocol. WebDriver berperan sebagai jembatan antara berbagai bahasa pemrograman dan browser, memungkinkan pengguna untuk menulis skrip tes yang dapat dijalankan lintas browser.

Selenium dapat dikembangkan dengan beberapa opsi bahasa pemrograman: JavaScript, Java, Javascript, Python, Ruby, C#, dan Perl



# Kenapa menggunakan Selenium

1. Selenium mendukung pengujian lintas browser yang luas, termasuk Google Chrome, Mozilla Firefox, Safari, dan Internet Explorer.
2. Sebagai salah satu framework pengujian yang paling populer, Selenium memiliki komunitas pengguna yang besar dan aktif.
3. Selenium dapat diintegrasikan dengan mudah dengan alat-alat lain seperti TestNG, JUnit untuk pengelolaan tes, dan Jenkins atau Bamboo untuk integrasi dan pengiriman berkelanjutan.
4. Selenium Grid memungkinkan pengujian paralel, yang berarti tes dapat dijalankan secara bersamaan di berbagai lingkungan dan browser.

# Selenium Architecture



# Selenium Architecture

Dari diagram pada slide sebelumnya, architecture dari Selenium dapat dijabarkan sebagai berikut:

1. **Selenium Client Library:** Di sini Anda menulis skrip pengujian. Skrip ini menggunakan API Selenium Client untuk berinteraksi dengan berbagai bagian dari arsitektur Selenium. Library untuk klien tersedia dalam berbagai bahasa pemrograman seperti Ruby, Python, Javascript, Java, dan C#, yang memungkinkan untuk memilih bahasa yang paling cocok dengan proyek.

# Selenium Architecture

- 2. JSON Wire Protocol Over HTTP:** Ini adalah protokol komunikasi yang digunakan Selenium untuk berkomunikasi antara skrip yang sudah ditulis dan browser drivers. Pada dasarnya, ini adalah seperangkat aturan yang memungkinkan client library mengirim perintah ke browser dalam bahasa yang dapat dimengerti oleh browser. Perintah ini diformat sebagai objek JSON dan dikirim melalui permintaan HTTP.
- 3. Browser Drivers:** Ini spesifik untuk setiap browser (seperti ChromeDriver untuk Chrome, GeckoDriver untuk Firefox, dll.). Browser drivers menerjemahkan perintah yang diterima dari Selenium Client Library menjadi tindakan yang dapat dieksekusi oleh browser. Mereka bertindak sebagai perantara, menerima perintah dari skrip tes melalui JSON Wire Protocol dan mengeksekusinya di browser.

# Prerequisites Selenium installation

- Node.js 16+
- Windows 10+, Windows Server 2016+ or Windows Subsystem for Linux (WSL).
- macOS 13 Ventura, or macOS 14 Sonoma.
- Debian 11, Debian 12, Ubuntu 20.04 or Ubuntu 22.04, Ubuntu 24.04, on x86-64 and arm64 architecture.

```
C:\Users\revie>node -v  
v16.20.2
```

# Cara Install Selenium

1. Buat sebuah folder project baru
2. Ketikkan pada terminal:  
**npm install selenium-webdriver**
3. Pastikan instalasi berjalan sukses sampai terlihat tulisan seperti dibawah ini.

```
D:\My Data\Digital Skola\QA Engineer Mini Bootcamp\SeleniumTest>npm install selenium-webdriver
added 17 packages, and audited 18 packages in 9s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 8.19.4 -> 10.9.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.9.2
npm notice Run npm install -g npm@10.9.2 to update!
npm notice
```

# Testing Selenium

1. Buat sebuah file baru dan beri nama **test.js**
2. Ketik seperti file berikut:

```
const { Builder, By, Key, until } = require('selenium-webdriver');

async function exampleTest() {
  let driver = await new Builder().forBrowser('chrome').build();

  try {
    await driver.get('http://www.google.com');

    let searchBox = await driver.findElement(By.name('q'));
    await searchBox.sendKeys('Hello World!', Key.RETURN);

    await driver.wait(until.elementLocated(By.id('result-stats')), 10000);

    let title = await driver.getTitle();
    console.log(`Page title is: ${title}`);
  } finally {
    await driver.quit();
  }
}

exampleTest();
```

3. Ketikkan pada terminal, lalu klik enter untuk menjalankan test script:  
**node test.js**



# UI Locators



# Apa itu UI Element

UI element adalah sebuah individual unit yang di render atau ditampilkan pada halaman aplikasi berdasarkan GUI (Web, Mobile, Desktop, etc).

Apa saja yang user liat didalam sebuah halaman aplikasi adalah elemen. Judul, header, button, input text dan banyak lagi.

UI element bisa jadi memiliki child element, contohnya pada tabel atau pada sebuah list



# Element Locators

UI element dan UI locator adalah suatu hal yang berbeda. UI locator adalah sebuah object atau alamat yang dibutuhkan untuk mencari sebuah atau beberapa UI elemen tertentu pada sebuah halaman.

**Untuk mencari sebuah UI element kita bisa menggunakan locator berupa:**

- **ID**
- **CSS Selector**
- **Class Name**
- **Tag Name**
- **XPath**



# Element Locators

Ketika kita membuat automation test untuk sebuah website, kita akan sangat sering melakukan pencarian locator untuk sebuah elemen.

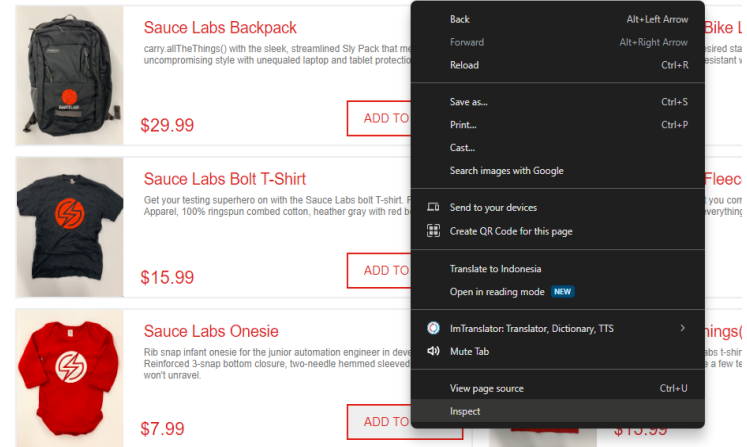
Mencari locator yang tepat bisa jadi sangat *challenging*, terutama ketika tidak ada locator unik seperti id pada sebuah web element

# Bagaimana cara mencari Web Locators

Untuk mencari elemen locator kita bisa menggunakan developer tools yang sudah ter install pada browser.

Kita bisa mengaksesnya dengan cara:

1. klik kanan
2. pilih inspect

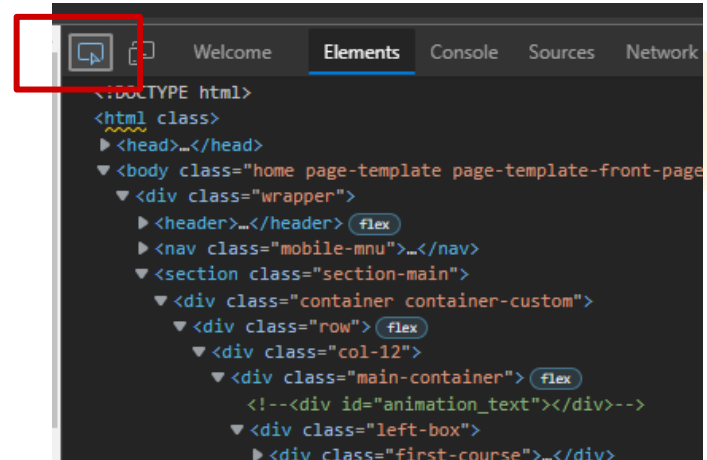


# Bagaimana cara mencari Web Locators

Setelah developer tools muncul, pilih select tools yang ada di pojok kiri atas menu.

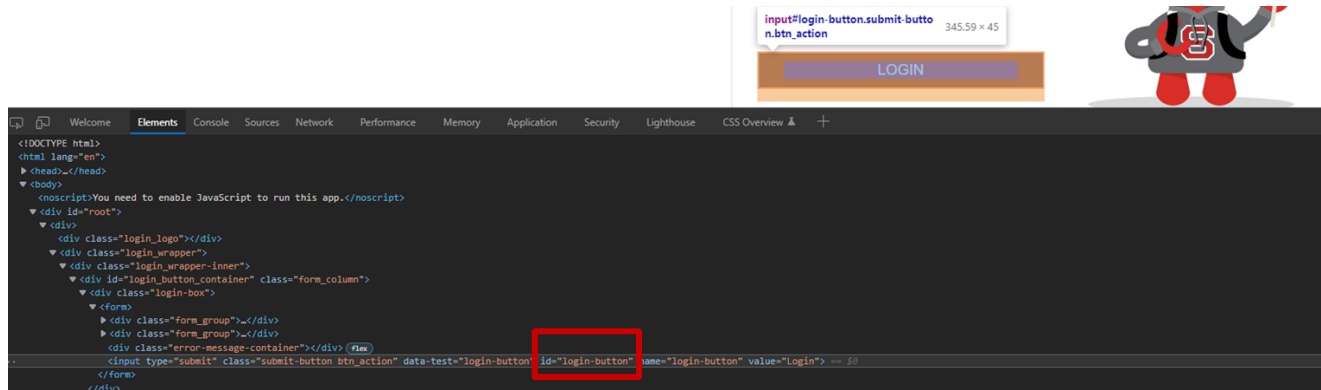
Setelah itu kita cukup melakukan hover ke elemen yang kita mau untuk mencari info mengenai locator dari elemen tersebut

Select Tools



# Locator ID

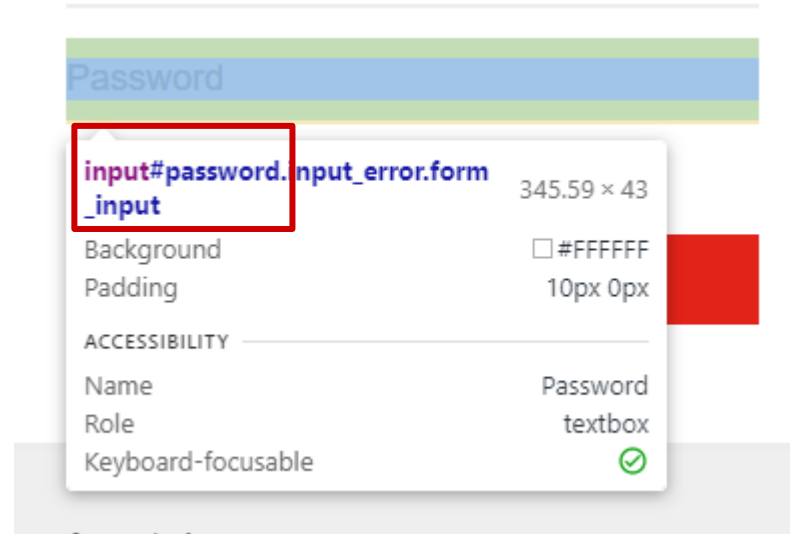
Menggunakan locator ID merupakan cara yang paling direkomendasikan karena value nya yg konsisten, unik dan simple. ID umumnya ditambahkan oleh developer, apabila belum ada seorang QA bisa untuk request pada developer untuk menambahkan Locator ID. Pada contoh dibawah, ID dari element button LOGIN adalah **“login-button”**



# Locator CSS Selector

Apabila dalam sebuah elemen tidak memiliki ID locator maka salah satu alternatif nya adalah dengan menggunakan CSS Selector.

Pada contoh disamping, element input Password memiliki CSS Selector value **"input#password"**



# Locator Xpath

Xpath merupakan bahasa query, untuk memilih dan mendapatkan informasi elemen dari file XML atau HTML.

Xpath menggunakan path notation untuk menemukan elemen nya. Cara penulisan Xpath notation ada 2 tipe:

- Absolute XPath
- Relative XPath

XPath:

```
/daftartoko/toko/daftarbarang/barang[@harga>400000]  
/daftartoko/toko/daftarbarang/barang[3]
```

Dokumen XML:

```
<?xml version="1.0" encoding="utf-8"?>  
<daftartoko>  
  <toko>  
    <daftarbarang>  
      <barang jenis="logam" harga="10000">besi 1 kg</barang>  
      <barang jenis="logam" harga="450000">emas 1 gram</barang>  
      <barang jenis="material" harga="53000">semen putih 50 kg</barang>  
      <barang jenis="material" harga="20000">batu kerikil</barang>  
      <barang jenis="elektronik" harga="1000000">telepon selular</barang>  
    </daftarbarang>  
  </toko>  
</daftartoko>
```



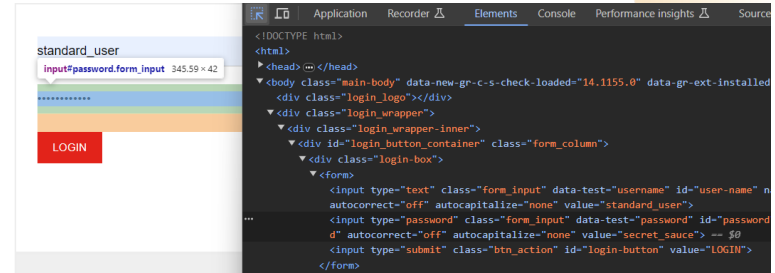
# Absolute Xpath

Model absolute Xpath menampilkan path dari root element hingga element tujuan secara eksplisit.

Contoh untuk Password:

**`"/html/body/div/div/div[2]/div[1]/div[1]/div/form/input"`**

**Menggunakan absolute Xpath sangatlah tidak direkomendasikan karena struktur HTML sangatlah mudah berubah.**

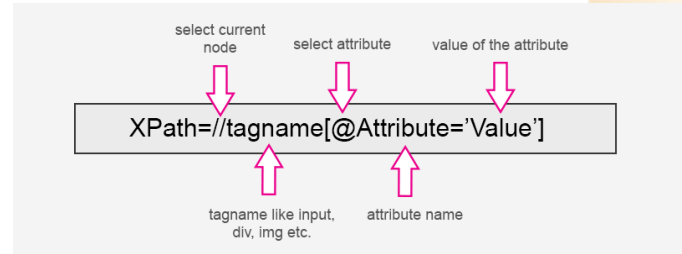


# Relative Xpath

Adalah versi lebih ringkas dari Absolute Xpath, dari path yang sama pada absolute xpath sebelumnya, berikut adalah model Relative XPath nya:

**`“//input[@id="password"]”`**

Cara penulisan relative Xpath terlampir pada gambar disamping



# Tools untuk Mencari Locator

## POM Builder Extension



### POM Builder – Auto-generate CSS/XPath Locator

Remove from Chrome

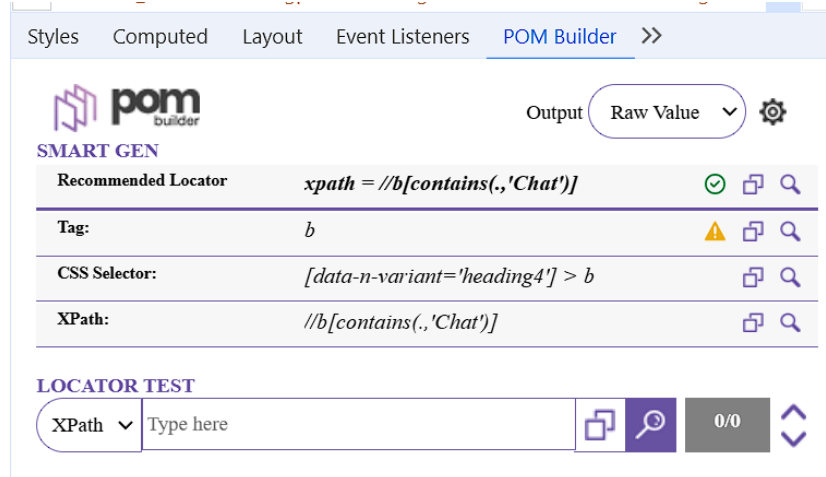
 [www.pombuilder.com](https://www.pombuilder.com) 3.9 ★ (28 ratings)

Extension

Developer Tools

10,000 users

Bisa dibuka di inspect element



Recommended Locator	Tag	CSS Selector	XPath
<code>xpath = //b[contains(.,'Chat')]</code>	<code>b</code>	<code>[data-n-variant='heading4'] &gt; b</code>	<code>//b[contains(.,'Chat')]</code>



# Selenium Web UI Test

# Basic Structure

```
const { Builder, By, until, Key } = require('selenium-webdriver');
const assert = require('assert');

async function saucedemoLoginTest() {
  let driver = await new Builder().forBrowser('chrome').build();

  try {
    await driver.get('https://www.saucedemo.com/');

  } finally {
    await driver.quit();
  }
}

saucedemoLoginTest();
```

- Kode ini mengimpor modul **Builder**, **By**, **until**, dan **Key** dari paket **selenium-webdriver**, yang digunakan untuk membangun dan mengelola sesi browser. Modul **assert** juga diimpor untuk melakukan asersi dalam tes.
- Setelah menjalankan kode yang diperlukan dalam blok try, blok finally memastikan bahwa sesi browser ditutup dengan memanggil driver.quit()

# Get Element Method

Untuk mendefinisikan UI element, Selenium support beberapa method seperti yang tercantum pada tabel dibawah ([referensi](#))

Method	Description
<code>driver.find_element(By.XPATH, xpath)</code>	Untuk mencari elemen menggunakan locator atribut dengan XPath
<code>driver.find_element(By.XPATH, "//*[@text()='text']")</code>	Untuk mencari elemen menggunakan konten text
<code>driver.find_element(By.CSS_SELECTOR, "[data-testid='id']")</code>	Untuk mencari elemen menggunakan atribut 'data-testid'
<code>driver.find_element(By.CSS_SELECTOR, css)</code> <b>atau</b> <code>driver.find_element(By.XPATH, xpath)</code>	Untuk mencari elemen menggunakan CSS atau XPath

# Action Method

Untuk melakukan interaksi tertentu dengan element, banyak action method yang sudah disediakan oleh Selenium ([ref](#)), diantaranya

Method	Description
<code>element.sendKeys("text")</code>	Untuk mengisi input atau textbox
<code>element.click()</code>	Untuk meng click element
<code>actions.move({origin: hoverable}).perform();</code>	Untuk melakukan hover diatas element
<code>actions.dragAndDrop(draggable, droppable).perform();</code>	Untuk melakukan drag and drop

# Expect/Assertion Method

Setelah melakukan action, tentu kita harus melakukan validasi apakah action yang dilakukan menghasilkan resul yang sesuai dengan ekspektasi. Selenium menyediakan metode assertion yang sudah disesuaikan dengan kebutuhan Web UI Automation seperti:

Method	Description
<code>assert element.is_displayed()</code>	Untuk melakukan pengecekan apakah elemen terlihat di dalam halaman web
<code>assert "text" in element.text</code>	Untuk melakukan pengecekan apakah sebuah elemen mengandung text tertentu
<code>assert element.is_enabled()</code>	Untuk melakukan pengecekan apakah sebuah elemen dalam keadaan aktif
<code>assert.strictEqual()</code>	Untuk memastikan bahwa fungsi mengembalikan hasil yang diharapkan dalam kondisi tertentu



# Contoh keseluruhan kode

```
const { Builder, By, until, Key } = require('selenium-webdriver');
const assert = require('assert');

async function saucedemoLoginTest() {
  let driver = await new Builder().forBrowser('chrome').build();

  try {
    await driver.get('https://www.saucedemo.com/');

    await driver.findElement(By.id('user-name')).sendKeys('standard_user');
    await driver.findElement(By.css('input[placeholder="Password"]')).sendKeys('secret_sauce');

    await driver.findElement(By.css('input[value="Login"]')).click();

    let titleText = await driver.findElement(By.xpath("//div[@class='app_logo']")).getText();
    assert.strictEqual(titleText.includes('Swag Labs'), true, 'Title does not include "Swag Labs"');

    let menuButton = await driver.findElement(By.id('react-burger-menu-btn'));
    assert.strictEqual(await menuButton.isDisplayed(), true, 'Menu button is not visible');

  } finally {
    await driver.quit();
  }
}

sauceDemoLoginTest();
```



# Thank You

#BertalentaDigital | [digitalskola.com](https://digitalskola.com)

© Copyright by Digital Skola 2024



# Tugas

1. Buatlah sebuah repo dengan nama **selenium-ui-test-digitalskola**
2. Setup Selenium project pada folder tersebut
3. Buatlah sample test dengan menggunakan website [www.saucedemo.com](http://www.saucedemo.com), dengan skenario seperti berikut:
  - a. User success login
  - b. Validate user berada di dashboard setelah login
  - c. Add item to cart
  - d. Validate item sukses ditambahkan ke cart
4. Push ke github dan submit link repository ke LMS