# CSS Locators

## What are CSS Selectors?

- **CSS (Cascading Style Sheets)** Selectors in Selenium are used to identify and locate web elements based on their id, class, name, attributes and other attributes.
- **CSS** is a preferred locator strategy as it is simpler to write and faster as compared to XPath.

There are **two main types of selectors**:

- **Absolute CSS Selectors** – Follow the full path to find the element.
- **Relative CSS Selectors** – Find the element using a shortcut path.

Practice on this sample website:
https://testpages.eviltester.com/styled/basic-web-page-test.html

## 1. Absolute CSS Selectors – Full Path

These selectors go step-by-step from the root (html) to the target element, like following a complete map.

**Examples:**

| What it Selects | Selector |
|---|---|
| **Whole HTML page** | html |
| **<head> section** | html > head |
| **<title> tag inside head** | html > head > title |
| **<body> section** | html > body |
| **<h1> inside a div in body** | html > body > div > h1 |
| **All <p> tags inside nested divs** | html > body > div > div > p |
| **<p> with id "para1"** | html > body > div > div > p[id='para1'] |
| **<p> with class "sub"** | html > body > div > div > p[class='sub'] |
| **Alternate ways:** | |
| **p#para1 = <p> with id="para1"** | html > body > div > div > p#para1 |
| **p.sub = <p> with class="sub"** | html > body > div > div > p.sub |
| **<p> with both id and class** | html > body > div > div > p[id='para1'][class='main'] |

**Note:** Tools like **SelectorsHub** often avoid absolute paths because they're long and fragile.

## 2. Relative CSS Selectors – Shortcuts

These selectors find elements directly without tracing the full path.

**Examples:**

| What you want | Selector |
|---|---|
| **<p> inside <body>** | body p |
| **<p> anywhere in the HTML** | html p |
| **<p> with id="para2"** | p[id='para2'] or p#para2 |
| **<p> with class="main"** | p[class='main'] or p.main |
| **Any element with id="para1"** | [id='para1'] or #para1 |
| **Any element with class="sub"** | [class='sub'] or .sub |

### Selecting Specific Children

You can select elements based on their **position** among siblings.

| Task | Selector |
|---|---|
| **First child inside a div** | body > div > *:first-child |
| **Last child inside a div** | body > div > *:last-child |
| **3rd child inside a div** | body > div > *:nth-child(3) |

### Attribute Selectors (Pattern Matching)

Use these when you want to match part of an attribute's value.

| Task | Selector |
|---|---|
| **Class starts with "ma"** | p[class^='ma'] |
| **Class ends with "ub"** | p[class$='ub'] |
| **Class contains "ai"** | p[class*='ai'] |

## Combining Selectors

You can combine selectors to filter even more specifically.

| Task | Selector |
|---|---|
| **<p> with both id and class** | p[id='para1'][class='main'] |
| **<p> not having id="para1"** | p:not([id='para1']) |
| **<p> with class="sub" but not id="para1"** | p:not([id='para1'])[class='sub'] |
| **<p> that does NOT have id="para1" AND NOT class="main"** | p:not([id='para1']):not([class='main']) |

## Following Sibling Selectors

Used to select elements that come right after another element.

| Task | Selector |
|---|---|
| **<p> right after <p id='para1'>** | p[id='para1'] + p |
| **Any tag right after <head>** | head + * |

## Generic way to use CSS in automation:

| CSS Selector Type | CSS Format |
|---|---|
| **Tag with ID** | tag#id |
| **Tag with Class** | tag.classname |
| **Tag with Attribute** | tag[attribute="value"] |
| **Tag with Class and Attribute** | tag.classname[attribute="value"] |