#### **CSS Selectors:**

# 1. Using ID CSS Selector

This is the most common way of locating elements, since IDs are supposed to be unique for each element. It can be used with hash (#) sign.



# 2. Using Class CSS Selector

Locating by CSS Selector using a class name is similar to using an ID, but in this case, a dot (.) is used instead of a hash sign.

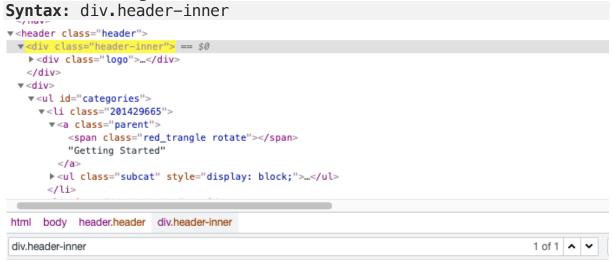


# 3. Using Tag and ID/Class CSS Selector

You can use "tag#ID".



You can use "tag.class" as well.

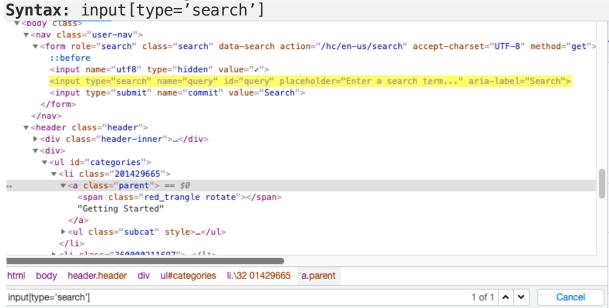


# 4. Using Tag & Attribute & Value

In my opinion, this one is the most used and most practical one. If there is a custom attribute available in the project such as data-test or data-qa-values, you could handle almost all locator issues. If there isn't any, please ask your developers to add those data attributes, which are lifesavers. One good example can be seen below with many data attributes.

```
data-qa-identifier="MainNav-Entries"> (flex)
  ▼<ul class="flex justify-center lg:justify-start leading-4" data-qa-identifier="MainN
  av-GenderSelection"> flex
   ▶...
   ▼
     ▼<a role="button" class href="/herren/"> == $0
       Herren
       e font-bold">Herren
      </a>
    ▶<ul data-qa-identifier="MegaMenuBar" class="jsx-1739437885 flex mt-2 lg:mt-0 pt-2 l
  g:pt-0 lg:ml-6 lg:pl-6 border-t lg:border-t-0 leading-4 list-border-color lg:relativ
  e">... flex
  ▶<div data-qa-identifier="MegaMenuBar-MegaMenu" class="jsx-1739437885">...</div>
 ▶<a role="button" class="order-2 xl:order-1 ml-4 sm:ml-6 xl:m-0 jsx-4090270469 logo-wra
 pper" data-qa-identifier="StoreLogo" data-qa-values="stylebop" href="/damen/">...</a>
 ▶ <nav class="jsx-308443480 order-3 ml-auto xl:m-0">...</nav>
 </div>
</header>
```

It could be used as tag[attribute='value'].



We can also extend this syntax using more than one attribute

Syntax: input[type='search'][name='query']

```
▼<body class>
   ▼<nav class="user-nav">
     ▼<form role="search" class="search" data-search action="/hc/en-us/search" accept-charset="UTF-8" method="ge
        <input name="utf8" type="hidden" value="/">
                             name="query" id="query" placeholder="Enter a search term..." aria-label="Search">
        <input type="submit" name="commit" value="Search">
      </form>
    </nav>
   ▼<header class="header">
     ▼<div class="header-inner">
      ▶ <div class="logo">...</div>
      </div>
     ▼<div>
html body header.header div ul#categories li.\32 01429665 a.parent
                                                                                            1 of 1 🔺 🗸
                                                                                                             Cancel
input[type='search'][name='query']
```

#### 5. Absolute Path CSS Selector

CSS absolute paths refer to the very specific location of the element considering its complete hierarchy in the DOM. Even though it is not a preferred way to use it, in case of need it can be used as follows.



With relative path, we can also locate an element directly, irrespective of its location in the DOM. Assuming that it is the only one, or it is the first element in the DOM. In this case I want to select first<input>.

Syntax: input

# 6. Using Non-Absolute Path CSS Selector

You should use white space between tags to locate the element. Use "." for class and "#" for id.

# 7. Class Chaining

It is most common to have more class names for an HTML element. You shouldn't use white space between classes to locate the element. It can also be combined with ID attribute, or with tag name if it is required.

Syntax: .flash.success

# 8. Using Containing Text of an Attribute

You can use tag[attribute\*='containing text'] syntax.

# 9. Using Starting Text of an Attribute

You can use **tag[attribute ^ = 'starting text']** syntax.

# 10. Using Ending Text of an Attribute

You can use tag[attribute\$='ending text'] syntax.



# 11. Using Comma Operator between CSS Locators

You can use "," **operator** between two CSS locator statements to implement OR operation.

# 12. Using first-of-type CSS Selector

You can use "Tag:first-of-type". It will select the first tag element.

Syntax: ul#categories>li:first-of-type

```
| ▼ <div>
| ▼ 
| ▼ 
| ▼ 
| ▼ 
| ■ 
| □ (lass="201429665">...
| ► 
| □ (lass="360000211697">...
| ► 
| □ (lass="115000895445">...
| ► 
| □ (lass="360000217618">...
| ► 
| □ (lass="36000023318">...
| ► 
| □ (lass="36000023318">...
| ► 
| □ (lass="201409709">...
| ► 
| □ (lass="360003863278">...
| ► 
| □ (lass="360003863278">...
| ► 
| □ (lass="36000127518">...
| ► 
| □ (lass="36000127518">...
| ► 
| □ (lass="36000129397">...
| □ (lass="36000129397")
| □ (lass="360001297")
```

# 13. Using last-of-type CSS Selector

You can use "Tag:last-of-type". It will select the last tag element.



# 14. Using tag:nth-of-type(n) CSS Selector

You can use "tag:nth-of-type(n)". It will select the nth tag element of the list.



# 15. Selecting sibling of an element by CSS Selector

There are some cases that we can select one element, then we can reach its siblings. In this case, this approach is very useful.

You can use "+" **operator** to reach sibling element.



# 16. Using NOT operator CSS Selector

This approach is beneficial, when we use one of the selector strategy to select the element/elements, with excluding the others by NOT operator. **Syntax:** div.large-12.columns:not(#content)

```
▼ <div class="row">
       ::before
      ▼<div id="flash-messages" class="large-12 columns">
        ▶<div data-alert id="flash" class="flash success">...</div>
       </div>
       ::after
     </div>
    ▼<div class="row">
       ::before
      ▶ <a href="https://github.com/tourdedave/the-internet">...</a>

... ▼<div id="content" class="large-12 columns" > == $0
       ▶ <div class="example">...</div>
       </div>
       ::after
html.no-js body div.row div#content.large-12.columns
div.large-12.columns:not(#content)
```

Above one is just to show an example, otherwise it is wise to select it with ID attribute.  $\stackrel{\square}{=}$ 

One more example can be seen below. This time I want to reach all the elements having classes "a-section", and "a-spacing-none", which gives 80 results, but I don't want to reach the elements having "fluid-image-container" class. Then using NOT operator would be very useful.

Syntax: div.a-section.a-spacing-none:not(.fluid-image-container)

```
<!-- sp:feature:host-atf -->
 ▼<div id="pageContent" class="a-section a-spacing-none" role="main">
    <a name="top"></a>
   ▶ <style type="text/css">...</style>
   ▶ <div class="off-screen">...</div>
   ▼<div cel_widget_id="desktop-hero-order" id="desktop-banner" class="celwidget" data-</p>
  data-cel-widget="desktop-hero-order">
    ▼<div id="gw-desktop-herotator" data-autorotation-delay="5000" class="a-section a-
    erotator-ready">
      ▶<div data-a-carousel-options="{"set_size":6,"minimum_gutter_width":0,"maintain_
      e":"gateway-desktop-layout.herotator","circular":false,"animation_speed":6000}"
      transition-strategy="slideCircular" data-a-class="desktop" class="a-begin a-carou
      arousel-transition-slideCircular gw-desktop-herotator a-carousel-initialized">...<,
... #gw-layout.a-section.a-spacing-none.aok-relative div#gw-card-layout.a-section.a-spacing-none.gw-c
.a-section.a-spacing-none:not(.fluid-image-container)
                                                                            1 of 60 A
```

# 17. Using tag:nth-child(n) CSS Selector in Selenium

You can use "tag:nth-child(n)". It will select the nth child.

