Cypress supports locators:

By default it supports the CSS Selector

We can also use the XPath locator: Possible when once we install cypress XPath plugin.

cy.get() is the method which takes the locators.

cy.get(Locator)

CSS Selector:

tag id

tag class

tag attribute

tag class attribute

ctrl + f -> Open the search box in the browser for finding by string, selector or XPath

Search chrome web store and add the extension selectorhub extension.

then type -> ctrl + shift + i -> beside the Styles, Computed etc we will have SelectorsHub

For ID: syntax: #id

A screenshot of a computer

Description automatically generated

For class: syntax: .className

Note: Make sure dot is present before className

For attribute: syntax: [attribute=’value’]

For tag class attribute: .class[attribute=’value’]

Note: Make sure dot is present before the class

So finally without tag:

#id

.class

[attribute=’value’]

.class[attribute=’value’]

With using tag:

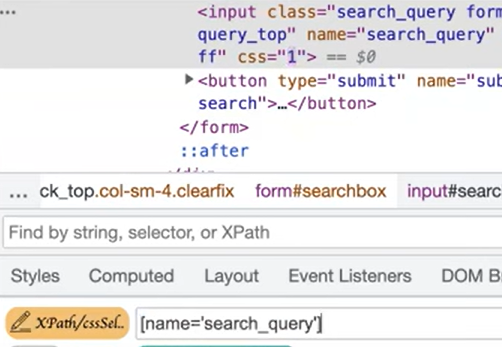
tag#id

tag.class

tag[attribute=’value’]

tag.class[attribute=’value’]

For attribute:



[name=’’search\_query’] or input[name=’’search\_query’]

Example:

1. Create the spec file CSSLocator.cy.js in the e2e folder.
2. Code:

Describe(‘CSSLocators’, () => {

It (“cssLocators”, () => {

cy.visit(“http://automationpractice.com/index.php”)

cy.get(“#search\_query\_top”).type(“T-Shirts”)

**// or with tag->** cy.get(“input#search\_query\_top”).type(“T-Shirts”)

// tag is optional

cy.get(“[name=’submit\_search’]”).click()

// after that we need to search for some element like

A screen shot of a computer

Description automatically generated

// after performing click action whether the t-shir element is displayed or not

cy.get(“.lighter”).contains(“T-Shirts”) // Assertion after clicking on the search

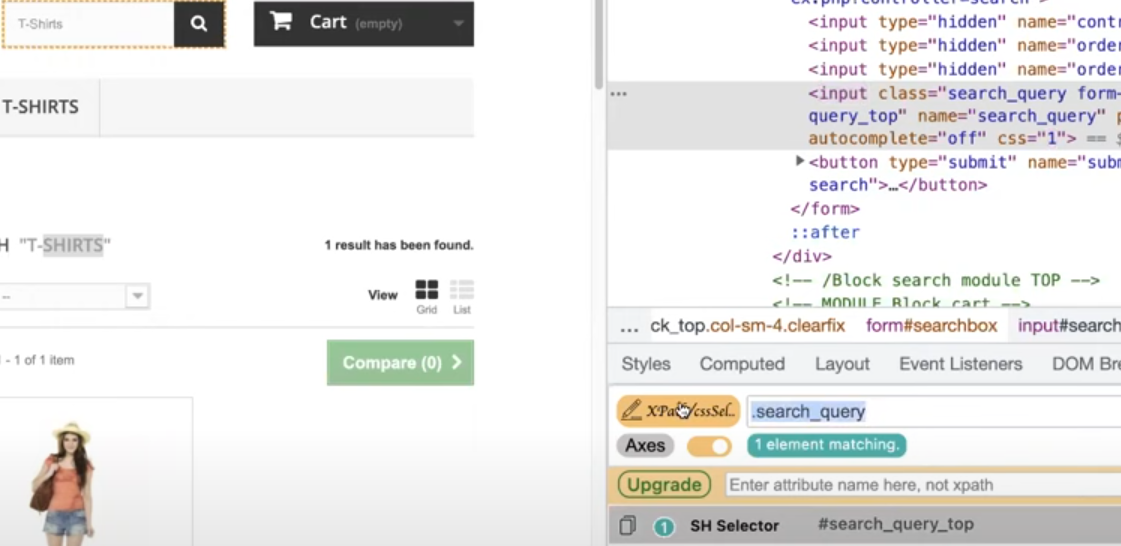
// icon, class named lighter must contains the word T-Shirts

})

})

If you change the script no need to stop the execution and run again it automatically runs.

Change the code -> save it -> test execution starts.



cy.get(“#search\_query\_top”).type(“T-Shirts”) <- instead of id you can use the class as given below

cy.get(“.search\_query”).type(“T-Shirts”)

If id and class is not present then we can use the attributes like type, name, placeholder etc

cy.get(“[name=’search\_query’]”).type(“T-Shirts”) // contain a attribute called name

or we can use the class and attribute

cy.get(“.search\_query[name=’search\_query’]”).type(“T-Shirts”)

With addition tag

Input is tag, search\_query is class, name is the attribute

cy.get(“input.search\_query[name=’search\_query’]”).type(“T-Shirts”)

By using XPath:

We need to install cypress XPath Plugin. Search in google

Type following command in the visual studio terminal:

npm install -D cypress-xpath

In the commands.js present in the support folder must have (for cypress commands)

/// <reference types=”Cypress” />

Otherwise you need to add the above command in every script file present in e2e folder. (For XPath commands): add the following code into the commands.js

/// <reference types=”cypress-xpath” />

In the e2e.js file present in the support add the code below // require(‘./commands’)

require(‘cypress-xpath’)

if the above code is not written in the e2e.js file then we need to write the same thing in every script file you are testing

Now create a new file name XPathLocators.cy.js inside the e2e folder

Describe(‘XPathLocators’, () =>{

It(‘find no of products’, () => {

cy.visit(“http://automationpractice.com/index.php”)

A screenshot of a website

Description automatically generated

// provides all the elements if you use should means assertion

// there are seven product or seven li, get is for css and xpath is for xpath

// You can use the xpath if you install the xpath plugin

cy.xpath(“//ul[@id=’homefeatured’]/li”).should(‘have.length’, 7)

})

It(‘Chained XPath’, () => {

cy.visit(“http://automationpractice.com/index.php”)

//cy.xpath(“//ul[@id=’homefeatured’]/li”).should(‘have.length’, 7)

cy.xpath(“//ul[@id=’homefeatured’]”).xpath(“./li”).should(‘have.length’, 7)

})

})