

When id attribute is present.

Use await else error

```
... locator2.spec.ts x locator4.spec.ts locator5.spec.ts locator6.spec.ts locator3.spec.ts
mytest > locator2.spec.ts > test('locator test') callback
1 import {test, expect, Browser, Page, Locator, BrowserContext} from '@playwright/test'
2 import {webkit, chromium, firefox} from '@playwright/test'
3
4 test('locator test', async() => {
5
6     const browser:Browser= await chromium.launch({headless:true, channel:'chrome'})
7     const page:Page=await browser.newPage()
8
9     await page.goto("https://naveenautomationlabs.com/opencart/index.php?route=account/register")
10
11     //three step strategy in pw and cypress.
12     //create webelement using locator.
13     //then perform action.
14     //then assert.
15
16     //using id.
17     //html will contain id attribute for the element.
18     //first name field.
19     const fname:Locator=await page.locator('id=input-firstname')
20     //last name field.
21     const lname:Locator=await page.locator('id=input-lastname')
22
23     //when the await is not used.
24     //it says browser closed before entering anything.
25     fname.fill('tiger')
26     lname.fill('karan labs')
27 });
```

To avoid closing browser after run automatically

```
... locator4.spec.ts locator5.spec.ts locator6.spec.ts locator3.spec.ts X
mytest > locator3.spec.ts > ...
1 import {test, expect, Browser, Page, Locator, BrowserContext} from '@playwright/test'
2 import {webkit, chromium, firefox} from '@playwright/test'
3
4 //see the error when we prevent browser from closing.
5 test('locator test', async() => {
6
7     const browser:Browser= await chromium.launch({headless:true, channel:'chrome'})
8     const page:Page=await browser.newPage()
9
10    await page.goto("https://naveenautomationlabs.com/opencart/index.php?route=account/register")
11
12    //three step strategy in pw and cypress.
13    //create webelement using locator.
14    //then perform action.
15    //then assert.
```

```
15    //then assert.
16
17    //using id.
18    //html will contain id attribute for the element.
19    //first name field.
20    const fname:Locator=await page.locator('id=input-firstname')
21    //last name field.
22    const lname:Locator=await page.locator('id=input-lastname')
23
```

```
23
24    await fname.fill('tiger')
25    await lname.fill('karan labs')
26
27    //to avoid closing browser immediately after running.
28    await new Promise(()=>{})
29    });
```

```
17 //using id

PROBLEMS OUTPUT DEBUG CONSOLE TEST RESULTS TERMINAL PORTS PLAYWRIGHT

Running 1 test using 1 worker
1) [chromium] > mytest\locator3.spec.ts:4:5 > locator test

Test timeout of 30000ms exceeded.

Error Context: test-results\locator3-locator-test-chromium\error-context.md

1 failed
[chromium] > mytest\locator3.spec.ts:4:5 > locator test
```

When class name attribute is present-

```
... locator4.spec.ts x locator5.spec.ts locator6.spec.ts
mytest > locator4.spec.ts > test('locator test') callback
1 import {test, expect, Browser, Page, Locator, BrowserContext} from '@playwright/test'
2 import {webkit, chromium, firefox} from '@playwright/test'
3
4 test('locator test', async() => {
5
6     const browser:Browser= await chromium.launch({headless:true, channel:'chrome'})
7     const page:Page=await browser.newPage()
8
9     //go to the register page.
10    await page.goto("https://naveenautomationlabs.com/opencart/index.php?route=account/register")
11
12    //three step strategy in pw and cypress.
13    //create webelement using locator.
14    //then perform action.
15    //then assert.

15 //then assert.
16
17 //class attribute.
18 //class name attribute is present.
19 //check if the logo is enabled.
20 const logo:Locator=await page.locator(".img-responsive")
21 const logoexist=await logo.isEnabled()
22 console.log(logoexist)
23 });
```

When we pass class like this then error-

```
... locator5.spec.ts X locator6.spec.ts
mytest > locator5.spec.ts > test('locator test') callback > page
1 import {test, expect, Browser, Page, Locator, BrowserContext} from '@playwright/test'
2 import {webkit, chromium, firefox} from '@playwright/test'
3
4 test('locator test', async() => {
5
6     const browser:Browser= await chromium.launch({headless:true, channel:'chrome'})
7     const page:Page=await browser.newPage()
8
9     //go to the register page.
10    await page.goto("https://naveenautomationlabs.com/opencart/index.php?route=account/register")
11
12    //three step strategy in pw and cypress.
13    //create webelement using locator.
14    //then perform action.
15    //then assert.
16
17    //class attribute.
18    //we cannot pass class attribute in this way.
19    //we get error Unknown engine "class" while parsing selector class=img-responsive
20    const logo:Locator=await page.locator("class=img-responsive")
21    const logoexist=await logo.isEnabled()
22    console.log(logoexist)
23 });
```

Use the visible text on the browser-
Check if text works for button and links also-

```
... locator6.spec.ts X
mytest > locator6.spec.ts > ...
1 import {test, expect, Browser, Page, Locator, BrowserContext} from '@playwright/test'
2 import {webkit, chromium, firefox} from '@playwright/test'
3
4 test('locator test', async() => {
5
6     const browser:Browser= await chromium.launch({headless:true, channel:'chrome'})
7     const page:Page=await browser.newPage()
8
9     //go to the register page.
10    await page.goto("https://naveenautomationlabs.com/opencart/index.php?route=account/register")
11
12    //three step strategy in pw and cypress.
13    //create webelement using locator.
14    //then perform action.
15    //then assert.
16
17    //text attribute.
18    //we capture using the visible text on browser.
19    //check register account seen.
20    const header:Locator=await page.locator('text=Register Account')
21    const headerenabled=await header.isEnabled()
22    console.log(headerenabled)
23
24
25    //text works on any visible element including buttons, links etc.
26
27    //check continue button seen.
28    const continuebutton:Locator=await page.locator('text=Continue')
29    const conitnueisenabled=await continuebutton.isEnabled()
30    console.log(conitnueisenabled)
31
32    //check forgotten password link seen.
33    const forgotpasswordlink:Locator=await page.locator('text=Forgotten Password')
34    const forgotPasswordEnabled=await forgotpasswordlink.isEnabled()
35    console.log(forgotPasswordEnabled)
36 });
```

Css

Use css as the key.

Then use normal way as selenium.

```
... locators7.spec.ts x locators8.spec.ts locators9.spec.ts locators10.spec.ts
mytest > locators7.spec.ts > test('locator test') callback
1 import { test, expect, Browser, Page, Locator, BrowserContext } from '@playwright/test'
2 import { webkit, chromium, firefox } from '@playwright/test'
3
4 test('locator test', async () => {
5
6     const browser: Browser = await chromium.launch({ headless: true, channel: 'chrome' })
7     const page: Page = await browser.newPage()
8
9     //go to the register page.
10    await page.goto("https://naveenautomationlabs.com/opencart/index.php?route=account/register")
11
12    //three step strategy in pw and cypress.
13    //create webelement using locator.
14    //then perform action.
15    //then assert.
16
17    //css selectors.
18    //use css at the start.
19    //then normal like selenium.
20
21    //for email field.
22    const email: Locator = await page.locator('css=input#input-email')
23    //for telephone field.
24    const telephone: Locator = await page.locator('css=input[name="telephone"]')
25    //for privacy checkbox field.
26    const privacycheckbox: Locator = await page.locator('css=input[type="checkbox"]')
27
28    //fill email.
29    await email.fill("karan@test.com")
30    await telephone.fill("3234")
31    await privacycheckbox.click()
32 });
```

By default all locators in pw is css.
Even if we remove css it works.

```
... locators8.spec.ts × locators9.spec.ts × locators10.spec.ts
mytest > locators8.spec.ts > test('locator test') callback
1 import { test, expect, Browser, Page, Locator, BrowserContext } from '@playwright/test'
2 import { webkit, chromium, firefox } from '@playwright/test'
3
4 test('locator test', async () => {
5
6     const browser: Browser = await chromium.launch({ headless: true, channel: 'chrome' })
7     const page: Page = await browser.newPage()
8
9     //go to the register page.
10    await page.goto("https://naveenautomationlabs.com/opencart/index.php?route=account/register")
11
12    //three step strategy in pw and cypress.
13    //create webelement using locator.
14    //then perform action.
15    //then assert.
16
17    //css selectors.
18    //use css at the start.
19    //then normal like selenium.
20
21    //for email field.
22    //remove css word and it will work.
23    //not mandatory for css word.
24    const email: Locator = await page.locator('input#input-email')
25    //for telephone field.
26    const telephone: Locator = await page.locator('input[name="telephone"]')
27    //for privacy checkbox field.
28
29
30    const telephone: Locator = await page.locator('input[name="telephone"]')
31    //for privacy checkbox field.
32    const privacycheckbox: Locator = await page.locator('input[type="checkbox"]')
33
34    //fill email.
35    await email.fill("karan@test.com")
36    await telephone.fill("3234")
37    await privacycheckbox.click()
38
39 });
```

Xpath-

Initial write xpath.

Rest same as selenium.


```
locators9.spec.ts X locators10.spec.ts
mytest > locators9.spec.ts > ...
1 import { test, expect, Browser, Page, Locator, BrowserContext } from '@playwright/test'
2 import { webkit, chromium, firefox } from '@playwright/test'
3
4 test('locator test', async () => {
5
6     const browser: Browser = await chromium.launch({ headless: true, channel: 'chrome' })
7     const page: Page = await browser.newPage()
8
9     //go to the register page.
10    await page.goto("https://naveenautomationlabs.com/opencart/index.php?route=account/register")
11
12    //three step strategy in pw and cypress.
13    //create webelement using locator.
14    //then perform action.
15    //then assert.
16
17    //xpath.
18    //write xpath word.
19    //then normal like selenium.
20
21    //for password field.
22    const password:Locator=await page.locator('xpath=//input[@id="input-password"]')
23
24    //for search text box.
25    const search:Locator=await page.locator('xpath=//input[@name="search" and @type="text"]')
26
27    //fill password and search something.
28    await password.fill("23423@#$$#@")
29    await search.fill("macbook")
30 });
```

Without xpath word also it works fine-

```
lcoators10.spec.ts X
mytest > lcoators10.spec.ts > test('locator test') callback
1 import { test, expect, Browser, Page, Locator, BrowserContext } from '@playwright/test'
2 import { webkit, chromium, firefox } from '@playwright/test'
3
4 test('locator test', async () => {
5
6     const browser: Browser = await chromium.launch({ headless: true, channel: 'chrome' })
7     const page: Page = await browser.newPage()
8
9     //go to the register page.
10    await page.goto("https://naveenautomationlabs.com/opencart/index.php?route=account/register")
11
12    //three step strategy in pw and cypress.
13    //create webelement using locator.
14    //then perform action.
15    //then assert.
```

```
15    //then assert.
16
17    //xpath.
18    //write xpath word.
19    //then normal like selenium.
20
21    //for password field.
22    //without xpath word also it works.
23    const password:Locator=await page.locator('//input[@id="input-password"]')
24
```

```
23    const password:Locator=await page.locator('//input[@id="input-password"]')
24
25    //for search text box.
26    const search:Locator=await page.locator('//input[@name="search" and @type="text"]')
27
28    //fill password and search something.
29    await password.fill("23423@#$#@")
30    await search.fill("macbook")
31    });
```