### **Handling Multiple Tabs, Multiple Windows, and Alerts in Cypress**

Cypress, by design, doesn't support multiple tabs or windows directly because it aims to provide a consistent and reliable testing environment. However, you can interact with elements in the same window and handle scenarios like alerts, navigation commands, and various mouse events.

### **1. Handling Multiple Tabs and Windows**

While Cypress doesn't handle multiple tabs or windows natively, you can test the content of new tabs or windows by manipulating the href attribute and ensuring it loads within the same tab. Here's how you can handle such scenarios:

// Intercept a link that would open a new tab and force it to open in the same tab

cy.get('a[target="\_blank"]').invoke('removeAttr', 'target').click()

// Alternatively, use the visit command to navigate to the new URL directly

cy.get('a[target="\_blank"]').then(($link) => {

const url = $link.prop('href')

cy.visit(url)

})

### 

### **2. Handling Alerts**

Cypress handles the alert by itself, but if we want to Assert the message inside alert, then we have to write some commands.

In Cypress, We don't have to write the code explicitly to just handle the alert.

Cypress automatically handles alerts, confirms, and prompts.

You can also assert their behaviour using cy.on().

// Handling a simple alert

cy.on('window:alert', (text) => {

expect(text).to.contains('This is an alert')

})

cy.get('button#alertButton').click()

// Handling a confirm dialog

cy.on('window:confirm', (text) => {

expect(text).to.contains('Are you sure?')

return true // Simulate clicking 'OK'

})

cy.get('button#confirmButton').click()

### **3. Navigation Commands**

Cypress provides commands to navigate through your application.

#### **Example:**

// Visit a URL

cy.visit('https://example.com')

// Go back to the previous page

cy.go('back')

// Go forward to the next page

cy.go('forward')

// Reload the current page

cy.reload()

### **4. Mouse Event Functions**

Cypress includes several commands for mouse interactions such as click(), dblclick(), rightclick(), and trigger() for more complex events like mouse hover.

#### **Example:**

**Click:**

cy.get('button#clickButton').click()

cy.get('button#doubleClickButton').dblclick()

cy.get('button#rightClickButton').rightclick()

**Mouse Hover:**

To simulate a mouse hover, you can use the trigger() command.

cy.get('button#hoverButton').trigger('mouseover')

Example code combining all :

describe('Comprehensive Test Suite', () => {

it('Handles multiple tabs, alerts, navigation, and mouse events', () => {

cy.visit('https://example.com')

// Handling multiple tabs

cy.get('a[target="\_blank"]').invoke('removeAttr', 'target').click()

cy.url().should('include', '/new-page')

// Handling alerts

cy.on('window:alert', (text) => {

expect(text).to.contains('This is an alert')

})

cy.get('button#alertButton').click()

// Handling confirm dialogs

cy.on('window:confirm', (text) => {

expect(text).to.contains('Are you sure?')

return true // Simulate clicking 'OK'

})

cy.get('button#confirmButton').click()

// Navigation commands

cy.visit('https://example.com/page1')

cy.go('back')

cy.url().should('include', '/previous-page')

cy.go('forward')

cy.url().should('include', '/page1')

cy.reload()

cy.url().should('include', '/page1')

// Mouse event functions

cy.get('button#clickButton').click()

cy.get('button#doubleClickButton').dblclick()

cy.get('button#rightClickButton').rightclick()

cy.get('button#hoverButton').trigger('mouseover')

})

})

Alerts :

///<reference types="cypress" />

describe('alerts practice', () => {

it('normal alert test', () => {

var name = 'Ganesh'

cy.visit('https://rahulshettyacademy.com/AutomationPractice/')

cy.get('#name').type(name)

cy.get('#alertbtn').click(); // clicking in alert button.

// i want to assert the text presnet in alert

cy.on('window:alert',(text)=>{

expect(text).to.contains(name)

})

// conformation alert handling

cy.get('#confirmbtn').click();

cy.on('window:confirm',(alert2)=>{

expect(alert2).to.contains('Hello , Are you sure you want to confirm?')

//return true; // it will select on ok

return false; // will select cancel

})

});

});

describe('drag and drop', () => {

it('test 1 ', () => {

cy.visit('https://v1.training-support.net/selenium/drag-drop')

// locator foe element 1 - ball

// {which:1} -> i want to work with left mouse button

// {which:2} -> i want to work with middle mouse button

// {which:3} -> i want to work with right mouse button

// in trigger() - 2 parameters - mousedown - 1 - left - {which:1} - left button of mouse

cy.get('#draggable').trigger('mousedown',{which:1})

// locator foe element 2- box

// we are telling cypress - to move the mouse from 1 point to the mentioned locator - box

// instructing cypress to release the button forcefully

cy.get('#droppable').trigger('mousemove').trigger('mouseup',{force:true})

cy.get('#draggable').trigger('mousedown',{which:1})

cy.get('#dropzone2').trigger('mousemove').trigger('mouseup',{force:true})

});

});

Hover :

describe('hovering', () => {

it('test1', () => {

cy.visit('https://rahulshettyacademy.com/AutomationPractice/')

// mouse hover - invoke() it will show the elements present inside mpusehover element

cy.get('#mousehover').invoke('show');

// Now it checks the text of top in the UI and clicks on it.

// {force:true} can be used inside click() , if click() action is not working properly

cy.contains('Top').click({force:true});

cy.url().should('include','top') ; // asserting the URL after clicking

});

it('test2 - second approach', () => {

cy.visit('https://rahulshettyacademy.com/AutomationPractice/')

// trigger() is used to perform mouse actions

cy.get('#mousehover').trigger('mouseover')

// Now it checks the text of top in the UI and clicks on it.

// {force:true} can be used inside click() , if click() action is not working properly

cy.contains('Reload').click({force:true});

cy.url().should('include','AutomationPractice') ; // asserting the URL after clicking

});

});

Double click :

///<reference types="cypress" />

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

describe('mouse tests', () => {

it('right click test', () => {

cy.visit('https://demo.guru99.com/test/simple\_context\_menu.html');

cy.xpath("//\*[contains(text(),'right')]").rightclick(); // right clicking

cy.xpath("//li[.='Quit']").click(); // clicking on quit option

cy.get('[ondblclick="myFunction()"]').dblclick(); // double click on element

});

});

### **Handling Web Tables, Mocking the Web Application, and Assertions for Visibility in Cypress**

Cypress is an excellent tool for end-to-end testing of web applications. It provides powerful commands for handling web tables, mocking web applications, and making assertions on the visibility of elements.