# Create the alerts.js file in the advanced folder.

Which has the following code:

///<reference types=”cypress” />

describe(‘Alert practice’, () => {

it(‘normal alert test’, () => {

var name = “Ganesh”;

cy.visit(“https://rahulshettyacademy.com/AutomationPractice/”)

cy.get(“#name”).type(name)

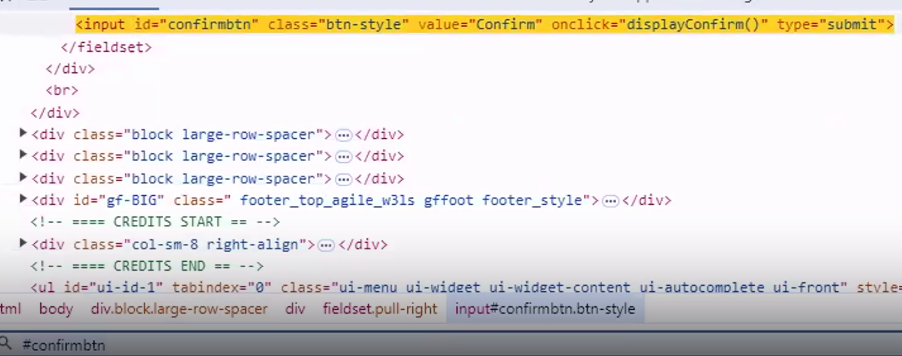
cy.get(“#alertbtn”).click(); // clicking in alert button

// I want to assert the text present 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?’)

})

});

});

## In the integration folder create the mouseaction folder having right.js file

Code in the right.js file is:

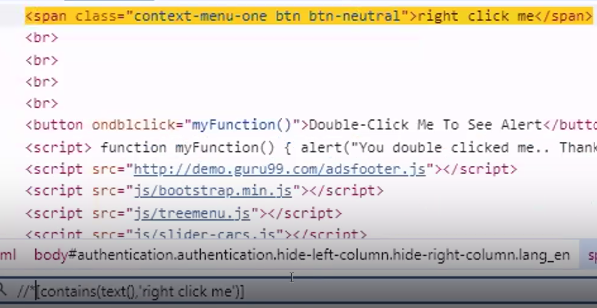
///<reference types=”cypress” />

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

describe(‘mouse tests’, () => {

it(‘right click test’, () => {

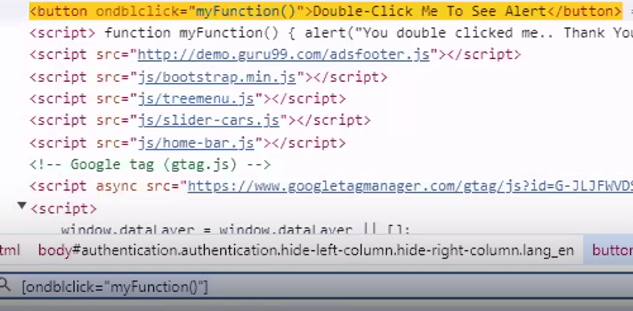
cy.visit(“https://demo.guru99.com/test/simple\_context\_menu.html”);



cy.xpath(“//\*[contains(text(), ‘right’)]”).rightclick()



cy.xpath(“//li[.=’Quit’]”).click();



//double Click on element

cy.get(‘[ondblclick=”myFunction()”]’).dblclick();

});

});

## Create the mhover.js file in the mouseactions folder.

Code present in the mhover.js:

describe(‘hovering’, () => {

it(‘test1’, () => {

cy.visit(‘https://rahulshettyacademy.com/AutomationPractice/’)

// Mouse hover – invoke() it will show the elements present inside mousehover 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});

// asserting the url after clicking

cy.url().should(“inlcude”, “top”)

})

it(‘test2 – Second approach’, () => {

cy.visit(‘https://rahulshettyacademy.com/AutomationPractice/’)

// trigger() is used to perform mouse actions where mouseover is one of the option

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});

// asserting the url after clicking

cy.url().should(“include”, “AutomationPractice”)

})

})

## Create the dragdropaction.js file in the mouseactions folder

describe(‘drag and drop’, () => {

it(‘test 1’ () => {

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

//locator for 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 for element 2 – box

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

// instructing the 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})

})

})