# Cypress Notes

**Cross browser testing**

Testing the same application in different browsres

**Parlale testing**

Distubuting the test cases to same browser and multiple instant

**What kind of factors to consider to chose the automation tool**

1.platform

2.community support

3.documentation

4.programing language support

5.report generate

**Selenium vs cypress**

1.Language support

Selenium support java,c#,js,ts,python

Cypress support js,ts

2.selenium is for only ui automation

Cypress support ui +api+visual automation

3.cypress run the test cases very fast manercompare to selenium

4.architechral difference

Cypress run the test cases directly in the browser

Selenium it can not interact with the browser it will first interact with the browser driverand browser driver will send the command to the brpwser

**Basic structure of cypress test case**

describe(‘soot name’,()=>{

It(‘test case name’,()=>{

Here Writre the script

})

})

**Ex:** describe(‘verify login functionality’,()=>{

It(‘verify login with valid details’,()=>{

Cy.get()

})

})

**Inbuilt methods in cypress**

Cy.visit()(lunch the url)

Cy.get()(get the locator)

Cy.contains()(when we have a text that time use this)

Cy.reload()(refresh the page)

Cy.go(-1)(backword way(go back one page)) or cy.go(‘back’)

Cy.go(1)(forword way) or cy.go(‘forword’)

.type()(type something)

.clear()(clear the text)

.click()(click the button or link or something)

.only(only particular test case to run)

.skip(to skip the perticuler test case)

**Cypress configurations**

‘BaseUrl’:”give the base url”

‘defaultcommandtimeout’:”100000” (default timeout is 4000 ms)(waiting to find the locator)

‘Pageloadtimeout’:”100000” (default pageloadtimeout is 60000 ms)(waiting to launch the url and load the page)

‘retries’:{open Mode} or {run Mode}(how many times to run failed test cases)

‘viewportwidth’:1000

‘viewportheight’:1000 (different resolutions) (cypress default resolution is 1000\*660)

‘video’:true

**Run the test cases in command**

1.npx cypress open (run the test cases in test runner)

2.npx cypress run (run the test cases in command mode)

**To run the specific folder**

npx cypress open --spec “in that folder any one file relative path and remove the file name and give \*\*”

**to run the specific file**

npx cypress open --spec “relative path of the file”

**headless mode and headed mode**

headless mode headed mode are the two methods in cypress run mode and headless is the default mode

**headless mode**

in this mode we cont identify actuatly which step is running in the background

npx cypress run or npx cypress run --headless

**headed mode**

in this mode it will invoke the browser and we can see actuatly which step is running and we will see everyin thing in the browser

npx cypress run --headed

**specify the browser**

npx cypress run --browser=browser name(npx cypress run --browser=chrome)

**to give the base url to specific file dainamicaly**

we can change the configurations in dainamicaly

npx cypress run --spec “relative path of the file”--config”give the base url”

**how to give the configurations to single spec file**

cy.visit(url,{timeout:100000})(this is the page load time out) (single test case)

cy.get(locator,{timeout:10000})(this is the to find the locator) (single test case)

cypress.config(‘pageloadtimeout’,2000) (single spec file)

cypress.config(‘defaultcommandtimeout’,2000) (single spec file)

it(test case name,{retries:{runmode:2},},()=>{} (single test case)

describe(soot name,{retries:{runmode:2},},()=>{} (single spec file)

**how to give the different config files**

**open mode**

npx cypress open --config-file”relative path of the configfile”

**run mode**

npx cypress run --config-file”relative path of the config file

**how to run the different commands in simple way or where we can store the commands**

we can store the different commads in package.json in side the packge.json in side the script we can store

“variable name”:’write the command’

**Ex**: “command1”:’npx cypress open’

**How to install old version of cypress**

Npm install cypress @version number

**Folder Structure**

Root folder

Cypress

Downloads

e2e

support

fixture

node modules

cypress.config.js

package-lock.json

package.json

**explanation**

one root filder is there in that cypress folder is there

**cypress folder**

in side the cypress folder downloadfolder is there

# download folder

we are downloading anything through the script all saved in to download filder

**e2e folder**

in this folder we can write the test script and we can create the folder and sub folder and files

**fixtures folder**

in this folder we can keep it all the test data and we can create the folders and sub folders and files

**support folder**

in side the support folder two folders are there

1.commands.js

Here we write the all the custum commands and we can maintain reusel script

2.e2e.js

Here we can import all the external libraries

**Videos folder**

If we are running test cases in the command line video folder will be create and it’s capture both failed and passed test cases

**Screen shot folder**

All the failed test cases screen shoots is there

**Node\_modules**

All the libraries are stored in the node modules

**Cypress.config.js**

Here we will add project lavel configurations like timeout,retries,env

**Package.json**

Here dependencies,dev dependencies are there and also information about the packages

**Package.lock.json**

We are not going to tuch in real time

**Cypress hooks**

1.before()

2.after()

3.beforeEach()

4.afterEach()

**Before:** before running any it black in our file it will execute. It will run only one time

**Ex:** before("Before All", ()=>{

cy.log("Before All block")

})

**beforeEach:** before running every it black beforeEach will execute after it black will execute

beforeEach("Before Each it", ()=>{

cy.log("Before each block")

})

**Ex:** beforeEach("launch url", () => {

cy.viewport(1920, 1080)

cy.visit('/web/index.php/auth/login')

})

**After:** after running all the test cases after will execute. And it will also execute one time

after("After all", ()=>{

cy.log("Aftetr All Tests block")

})

**afterEach:** after execution of every it black afterEach will execute

afterEach("After Each it", ()=>{

cy.log("After each block")

})

**order:**

before()

beforeEach()

afterEach()

after()

**POM(Page object moadel)**

Page object model will created by our own

Inside the page object model we have to store the locators page wise

We will explore that page and import that page in cy.js file

Class classname{

Objname(){

return locator

} }

Const objname1 = new classname()

Export default objname1

In cy.js file

Import objname1 from “../../pom name”

**Access is**

Cy.get(objname1.methodname())

**Locators**

Locators are the most important accept in the automation.

By default cypress will support only css locators and xpath locators also support but we can incorporate the xpath plagins

**Xpath locators**

Two types of xpath is there

1.absolute xpath (start from root node and start always ‘/’)

2.relative xpath (start from relative node and always start with ‘//’)

**Basic Syntax:** //tagname[@attributename=’attributevalue’]

**Text:** //tagname[text()=’fulltaxt’]

Partialtext: //tagname[contains(.,’partialtext’)]

**And:** here both the attribute should be present

//tagname[@attr1=’attr1value’][@attr2=’attr2value’] or

//tagname[@attr1=’attr1value’ and @attr2=’attr2value’]

**OR:** here any one attribute should be present

//tagname[@attr1=’attr1value’][@attr2=’attr2value’] or

//tagname[@attr1=’attr1value’ or @attr2=’attr2value’]

**Prefix:** //tagname[start-with(@attributename,’prifixvalue’)]

**Indexing:** (//tagname[@attributename=’attributevalue’])[indexnumber]

**Traversing parent to child node**

//tagname[@attributename=’attributevalue’]/childname/childname

**Traversing child to parent node**

//tagname[@attributename=’attributevalue’]/../.. (.. is parent)

**CSS Locators**

**Basic syntax:** tagname[attributename=’attributevalue’]

**Id selecter:** #id value

**Class selecter:** .class value

When you have a multipul class in same class id that time use ‘.’

Ex: .oxd input.oxd input value

Text: when you have a text we cont write the locator

In cypress in built method is there cy.contains(‘text’)

**And:** tagname[attributename=’attributevalue’] [attributename=’attributevalue’]

**Prefix value:** always dainamicaly changing the suffix values that time

Tagname[attributename^=’prefix value’]

**Suffix value:** always dainamicaly changing the prifix values that time

Tagname[attributename$=’suffix value’]

Prefix or suffix any values is there we can use tagname[attributename\*=’prifix or suffix value’]

**Traversing child to parent**

Traversing child to parent node is not possible in css but in cypress in built method is there

.parent()

**Traversing parent to child**

Tagname [attributename=’attributevalue’]>child attributename

**Indexing:** many same kind of locators is there that time we are using indexing

.first()

.last()

.eq(index number) (index number always start from ‘0’)

**Following-sibling:** it will jump in the following subling

tagname[attributename=’attibutevalue’]+tagname

**Playground:** playground also will give the locators but it is not give every time proper locators so that’s why we write the own locators

**Data driven testing**

Data driven testing in cypress allows you to run tha same test cases with different set of data

This can help ensure that your application works corrently for various input scenarios

1.create the different set of data in the fixter folder in the form of json

2.in the script using cy.fixture we can do it

**Ex: In the fixture folder**

**File name is username1**{

“username”:’mahesh’

“password”:’mahi#4201’

}

**File name is username2**{

“username”:’nagesh’

“password”:’nagesh#4201’

},

]

**In script before describe black**

const fixturedata = [

{

pathname: "username1",

context: "tittle1"

},

{

pathname: "username2",

context: "tittle2"

}

describe("working with fixtures ", function () {

fixturedata.forEach((data1) => {

describe(data1.context, () => {

before(function () {

cy.fixture(data1.pathname).then(function (data) {

this.data = data;

}); }); }); });

**Access is**

cy.get('input[class="oxd-input oxd-input--active"]').last().type(this.data.username);

cy.get('textarea[placeholder="Type description here"]').type(this.data.password);

**Working with elements and cypress tips and tricks**

**Button:** cy.get(locator).click()

Element is being covered with other elements**(hidden elements)** that time we can use this method **{force:true}**

cy.get(locator).click({force:true})

**click serialy**

cy.get(locator).click({multiple:true})

**type:** type delay default time is 10 ms {delay:800}

if give delay:0 it will past what ever we written in the type()

cy.get(locator).type(‘mahesh’,{delay:0})

**key board press**

cy.get(locator).type(‘cypress.io{enter}’)

**chech boxes:**

**syntax:** .check()

cy.get(locator).check()

assertion check

.should(‘be.checked’)

Ex: cy.get(locator).check().should(‘be.checked’)

**Few check boxes only check**

cy.get(locator).check()([give the values])

ex: cy.get(locator).check()([‘html’,’css’,’js’])

**Un check**

**Syntax:** .uncheck()

cy.get(locator).uncheck()

assertion check

.should(‘not.be.checked’)

**Ex:** cy.get(locator).check().should(‘not.be.checked’)

**cy.wait()** (waiting for some time before executing the next line)

**radio button**

**syntax:** .check()

cy.get(locator).check()

assertion check

.should(‘be.checked’)

Ex: cy.get(locator).check().should(‘be.checked’)

**Un check:** un check is not possible in radio button

**Dropdown**

In dom ‘select’ is the tag name is there that time we can use this

**Syntax:** .select()

Three ways we can select the dropdown

1.text ex: cy.get(locator).select(‘march’)

2.value ex: cy.get(locator).select(‘03’)

3.index ex: cy.get(locator).select(2)

**Any dropdown perticuler value is present or not**

cy.get(locator).should(‘have.attr’,’value’,’give the value’)

**Select the multiple options in dropdown**

cy.get(locator).select([‘value1’,’value2’])

**if in the dropdown ‘select’ tagname is not there that time we can use**

cy.get(locator).click() and

cy.get(value locatoer).click()

**in dropdown selected one value that value is selected or not how will verify is**

cy.get(locator).select(text or value or index number).should(‘have.value’,’here give the value’)

**Alerts**

**Difference between pop up and alert**

**Pop up:** it is web based we can inspect

**Alert:** alert is the windows component we con’t inspect

Three types of alerts is there

**1.js alert or simple alert**

**Ex:** cy.on(‘windows:alert,(name1)=>{

expect(name1).to.equal(‘name2’)

return ture

})

**2.js conform alert**

Here we can use the stub

**Ex:** cy.on(‘windows:conform’,()=>{

Return false or true (true=ok, false=cancel)

})

**3.js promt alert**

**(This is for ok )Ex:** cy.window().then(name1)=>{

cy.stub(name1,’promt).return(‘type some text’)

cy.contains(click for js promt).click()

})

**(This is for cancel)** **Ex:** cy.window().then(name1)=>{

cy.stub(name1,’promt).callFake(()=>null)

cy.contains(click for js promt).click()

})

**How to ingnore any console error**

**Syntax:** cypress.on(‘uncought:expection’,()=>{

Return false

})

**How to verify title of the page**

cy.title (this is the in built method)

cy.title().should(‘eq’,’text of the title’)

**how to get the text from an element**

cy.get(‘locator of the element’).then((text1)=>{

let variablename=text1.text()

cy.log(variablename)

})

**ifraames**

for clicking the iframe elements derectly is not possible

cy.get(‘locator of the iframe’).then(name1)=>{

let variablename =name1.contains().find(locator of the element in iframe)

cy.wrap(variablename).type() or .click() })

**tabs/windows**

switching the one teb to another tab is not possible in cypress

that time we can remove the target attribute that time it will not open new window it will open the same window

**syntax:** cy.get(locator).invoke(‘removeAttr’,’target’).click({force:true})

**how to check the url assertion**

cy.url().should(‘include’,give the assertion url’)

**if we don’t have target attribute that time**

cy.visit(‘url’,{

on BeforeLoad(name1){

cy.stub(name1,’open’)

}

})

**Drag and Drop**

We can multipule ways to do the dreg and drop and I know 2 ways

**1.using plugin:**  first we have to incorporate the drag and drop plugin and we have to perform

**Syntax:** cy.get(‘source iterm locator’).drag(‘target iterm locator’)

Using this plugin we can drag only one value we con’t drag multiple values

**2.datatrensfer:** datatresfer is the in built method in javascript

First we can a object of the datatrasfer class

**Ex:** Const variablename= new DataTransfer()

cy.get(‘source iterm locator’).trigger({‘dragstart’,

varibblename

})

cy.get(‘target iterm locator’).trigger({‘drop’,

variablename

})

**Mouse Evnts or Mouse Actions**

.click()

.dblclick()

.rightClick()

.scrollIntoview() ex: cy.get(locator of the specific position).scrollIntoView()

**Shadow dom element**

For interact with the sadow element we have to use the **.shadow()** method

First get the locator until there no shadow root after we use .shadow() method and find the locator in the shadow

**Syntax:** cy.get(locator).shadow().find(locator)

**How to work with the links**

First we have to go dom and type ‘//a’

Cy.get(‘a’).should(‘have.length’,give the how many links you see in the dom)

**Ex:** cy.get(‘a’).should(‘have.length’,65)

**cy.origin**

when you are click on some one link if it is navigating to the different domine and generate the new link that time we can use cy.origin

**syntax:** cy.origin(‘different domine link’,()=>{

cy.get(locator).type()

})

**How to get the text from the perticuler link**

cy.get(locator).then((name1)=>{

let variablename=name1.text()

cy.log(variablename)

})

**Work with the tables**

To check the how many tables are the in a page **‘//table’**

**In a table how many rows are there count**

//table[locator of the table ]/tbody/tr

**Ex:** cy.xpath(‘//table[locator of the table ]/tbody/tr’).should(‘have.length’,give the value)

**In a table how many colums are there count**

//table[locator of the table ]/tbody/tr[1]/td

**Ex:** cy.xpath(‘//table[locator of the table ]/tbody/tr[2]/td’).should(‘have.length’,give the value)

**In a table 3rd row 4th colum what is the value how to get (static value or constant value)**

cy.xpath(‘//table[locator of the table ]/tbody/tr[2]/td’[2]).then((name1)=>{

let variablename = name1.text()

expect (variablename).to.equal(‘give the value’)

})

**In a table 3rd row 4th colum what is the value how to get (range)**

cy.xpath(‘//table[locator of the table ]/tbody/tr[2]/td’[2]).then((name1)=>{

let variablename = name1.text()

expect (variablename).to.be.within(value1,value2) (value like 2000,3000)

})

1. **.to.equal()**
2. **.to.be.within()**
3. **.to.be.greaterThen()**
4. **.to.be.lessThen()**
5. **.should(‘be.empty’)**
6. **.not.to.be.empty**

**Print the enterie data in a table**

cy .get(‘table[name=”table”]>tbody>tr’).each((name1) =>{

cy.wrap(name1).within((name2) =>{

cy.log(name2.text())

})

})

**In a table in a colum how to get the totel of that colum ( like 1st row ‘30’ 2nd row ‘30’ 3rd row ‘40’ print 100)**

var total;

cy.get('.dataTable> tbody > tr >td:nth-child(4)')

.each((col) => {

cy.wrap(col).within((celldata) => {

var x = celldata.text()

var y = Number(x)

total = total+y

})

})

cy.log(total)

**custom command**

**syntax:** Cypress.Commands.add("addemployee”,()=>{

})

**Ex:**

**In commands.js file**

Cypress.Commands.add("login", (username, password)=>{

cy.visit('https://opensource-demo.orangehrmlive.com/web/index.php/auth/login')

cy.get('input[name="username"]').type(username)

cy.get('input[type="password"]').type(password)

cy.get('button[type="submit"]').click()

})

**Access in the script**

cy.login(‘Admin’,’admin123’) or cy.login(logindata.username,logindata.password)🡺 access in the fixture folder also

**File uplode**

Two ways we can uplode the files

1. **.selectFile()**

cy.get(locator).selectFile(‘relative path of the file’)

1. **Install the file uplode plugin**

cy.get(locator)**.attachFile**(path of the fixture(but remove the fixture name))

**file download**

by using download plugin we can download the files

cy.downloadFile(‘url’,’where you want to download’,’filename’)

**cy.session()**

we are implement one time it will login and it will store the cookies whenever we leanched the different module url of that application or integration of another page url it will restore back the new testcases

beforeEach(function () {

cy.session("Login session", () => {

cy.visit("/web/index.php/auth/login")

cy.get('input[name="username"]').type("Admin")

cy.get('input[type="password"]').type("admin123")

cy.get('button[type="submit"]').click()

})

})

**writeFile**

**writeFile:** this is the in built method in cypress

**syntax:** cy.writeFile() ex: cy.writeFile(‘cypress/fixture/writefiledata/text.json’,’mahesh/n’)

**{flag:’a+’} this write the all values**

**Ex2:** it.only('Cypress Test Case - Write file example', function () {

cy.writeFile('cypress/fixtures/module1/test.txt', "mahesh\n" )

})

**Ex3:** create the json file

it.only('Cypress Test Case - Create Json file ', function () {

cy.writeFile('cypress/fixtures/module1/test6.json', { firstname: 'mahesh', lastname: 'b n'});

})

**readFile:** this is also in built method in cypress

ex1: cy.readFile("cypress/fixtures/logincreds.json").its("username").should("eq", "Admin")

ex2: cy.readFile('cypress/fixtures/logincreds.json').should('exist')

**conditional testing**

it('Check that if you find WikiVoyage on the page, then click on it and validate (Go to If)', () => {

cy.wait(3000)

cy.get('body').then(($body) => {

if ($body.find('[data-jsl10n="wikivoyage.dkrgjoerjg"]').length > 0) {

cy.get('[data-jsl10n="wikivoyage.name"]').click()

cy.wait(3000)

cy.origin('https://www.wikivoyage.org', () => {

cy.title().should('eq', 'Wikivoyage')

})

}

else {

cy.get('[data-jsl10n="wiktionary.name"]').click()

cy.wait(3000)

cy.origin('https://www.wiktionary.org', () => {

cy.title().should('eq', 'Wiktionary')

})

}

})

})