Cypress

1) What is Cypress

- Cypress is a JavaScript based automation testing tool which is built for modern web applications

2) How to Install Cypress

- First we need to install node.js and visual studio code then create a folder then run a command 'npm init' it creates package.json then run a command 'npm install cypress' it install the latest version of cypress then run a command 'npx cypress open' it open test runner then click on e2e testing then select browser once we configured everything by default we can get a spec files

3) Cypress Folder Structure or Cypress Framework Structure

- Under root folder we have
  1. cypress folder
  2. downloads
  3. e2e
  4. fixtures
  5. support
  6. node modules
  7. cypess.config.js
  8. package.json
  9. apart from this we create Page Object Model folder inside the cypress folder where in :
- **downloads :** Stored all the downloaded files
- **e2e :** Stored all the created spec files
- **fixtures :** Maintain all the test data
- **support :** Here by default we have to 2 files
  1.commands.js : we use custom commands to reduce the codes and reusability purpose
  2. e2e.js : importing different external plugins like xpath, drag and drop, file download, file upload
- **package.json :** Stored all the cypress dependency informations
- **Page Object Model :** Stored all the locators inside the POM to modify the changes and reusable purpose and using class method we create POM and using import method we can read the data from POM

4) Why do we use cypress or cypress Features

- Easy setup, fast execution, timeouts, automatic waitings, retries, screenshots and video recordings for failed test cases, unique test runner, time travel debugging, simple API for testing

5) Latest version of the Cypress And currently using version in your project

- latest version - 13.13.1
- used version - 12.0.1

6) Cypress Advantages

- All cypress commands are synchronized
- Cypress has a test runner concept
- Cypress runs the test cases directly on browser in fast manner
- While running a test cases in cypress it has a features like timeouts, automatic waitings, retries, screenshots and video recordings for failed test cases, time travel debugging and simple API for testing

7) Cypress Disadvantages or Limitations

- Cypress cannot support multi languages it support only the javascript language
- Cypress cannot support multi tabs
- Cypress cannot open multiple domain
- Cypress cannot support multiple browsers it support only the chrome, edge, electron, firefox browser

8) Difference between Cypress and Selenium

- **Cypress :** **Selenium :**

Cypress has a test runner concept	Selenium don't have test runner
Cypress runs the test cases directly on browser	Selenium runs the test cases in browser driver
All cypress commands are synchronized	Synchronization is the major issue in selenium
Cypress support only javascript language	Selenium support multi languages
Cypress cannot support multi tabs	Selenium support multi tabs
Cypress support chrome, edge, electron, firefox browsers	Selenium support multiple browsers

9) Cypress test runner

- run a command npx cypress open so it open the test runner then click on e2e testing and select a browser then it open all the test cases once we click on particular test case it runs the test cases and also if we modify anything in test case it automatically re runs the test case and also we have screenshots video recordings retries for failed test cases

10) Cypress Custom commands

- we use custom commands when we want to make it a method available globally access test cases and to avoid repetitive steps by reducing the codes and for reusability purpose we use custom commands and it is easy to use and easy to maintain
- ex : login here we no need to write again and again in all the time and no need to import and export by using cypress.commands.add method
  directly we can called custom command wherever it required
- Syntax : cypress.commands.add("custom command name login", (u/n, p/w) =>{})

11) Cypress Reusability

- By implementing POM for saving locators, fixtures for test data, support for custom commands to access these to our scripts we can maintain reusability

12) Faced challenges while doing the project and how to overcome this

- Finding the locators is the one of the challenge so we ask developer to add the ID values to overcome this issue
- cypress cannot support multi tabs so we use .invoke method and by removing target attribute we can overcome this issue so tab will open in same window
- In a single test we cannot open multiple domain so we use cy.origin method to overcome

13) How to handle testing in less time

- first we focus on top priority features and critical features and also by optimizing the code we can handle testing in less time

14) How to handle Environment variables in Cypress

- Using cypress.env we can handle environment variables
  Go to Cypress.config.js
  e2e {
  enter the data
  base url, timeouts, retries, viewports
  }
- Below 9.7 version we called it as cypress.json
- Above 10.0 version we called it as cypress.config.js

15) How to read data or values from config file

- first we have to set environment variables then by using cypress.env we can read the data from config file

16) Cypress Configurations

- base url, retries, command timeout, page load timeout, viewport width, viewport height

17) Retires → using retries we can handle failed test cases

- Cypress.config.js
  “retries” : {“ runmode”:3, “openmode”:5}
- runmode → before failing test cases it retries 3 times and it attempts 4 times
- openmode → if open mode is 5 it attempts 6 times

18) Cypress Timeouts

2 Types of Timeouts in cypress

- **Implicit timeouts :** it applicable for all the elements, Ex : page load time out
  Default page load timeout : 60secs/60000ms
- **Explicit timeouts :** it applicable for one particular element, Ex : command time out
  Default command timeout : 4secs/4000ms

  Sleep or cy.wait(5000)
  cy.get("locator",{timeout:5000}).type()

19) How to handle uncaught exception error or console error

- We can handle using cypress.on method
  cypress.on("uncaught : exception", () => {
  return false;
  })

20) Alerts

- We can handle alerts using cy.on method
  There are three types of alerts in cypress
  1. Simple alert
  2. Confirm alert
  3. Prompt alert

- 1. Simple alert -> ok
  cy.on(" window : simple", () => {
  return true;
  })
- 2. Confirm alert -> ok
  cy.on(" window : confirm", () => {
  return true;
  })
- 2. Confirm alert -> cancel
  cy.on(" window : confirm", () => {
  return false;
  })

21) Cypress Hooks

- There are four types of hooks in cypress
  1. before() : It runs only once before all the test cases execute
  2. after() : It runs only once after all the test cases execute
  3. beforeEach() : It runs beforeEach of the test cases execute
  4. afterEach() : It runs afterEach of the test cases execute

22) How to create test suite

- 1st create a file in e2e and using describe we create test suite and inside the describe we create it blocks So describe is for test suite writing and It blocks for test cases writing

23) drag and drop

- For one item we use .drag method and for more than one item we use .trigger method
  cy.get("source item locator").drag("target item locator")
  we cannot drag more than one item so we use .trigger method
  const dataTransfer = new dataTransfer();
  cy.get("source item locator").trigger("dragstart", {
  dataTransfer
  })
  cy.get("target item locator").trigger("drop", {
  dataTransfer
  })

24) Cypress browser

- Default browser is electron browser
- Default mode is headless mode - it runs the test cases in background
- headed mode is open the test runner then run a test cases

25) Run a test cases in command line

- npm install cypress - to install the cypress
- npx cypress open - to open test runner
- npx cypress run - to run all the test cases
- npx cypress run --headed --browser=chrome --spec "relative path"
- npx cypress run --browser=chrome --spec "relative path"
  to run a spec file in particular browser
- npx cypress run --headed -- spec "relativepath"
  to run a spec file in headed mode
- npx cypress run -- spec "relative path"
  to run single spec file in command line
- npx cypress run --config base url=url
  to run a particular base url

26) Difference between npm and npx

- npm - node package manager - npm is used to download the packages
- npx - node package execute - npx is used to run the packages

27) Important Cypress Commands

- cy.visit, cy.contains, cy.type, cy.wait, cy.request, cy.scrollTo, cy.url, cy.reload, cy.title, cy.get, cy.log, cy.click, cy.clear, cy.go

28) How to handle Shadow

- First get the element locator then use .shadow method then use .find inside that we have to give specific locator then use .check
  Ex : cy.get("element locator").shadow().find("specific locator").check()
- .find is to navigate the child element

29) How to handle Windows handling

- using .invoke method and removing target attribute target = -"blank”
  we can handle windows handling so blog will open in same window
  Ex : cy.get("locator").invoke("removeAttr","target").click()

30) How to handle Session

- we can handle sessions using cy.session() method
  so without login every time we can go to main pages so we use sessions
  and also we use session to storing a local storage

31) How to handle iframes

- 1st get the iframe locator and use .then function and inside the function
  what ever the element we want to work on so we saving into variables
  and element = iframe.contents.find (“particular element locator)”
  and using cy.wrap.click or cy.wrap.type we can handle the iframes

32) How to handle tables in cypress

- Get the common locator for all the rows and using .each method we can iterate the rows

33) Asserting title or Verify the title in Cypress

- using cy.title() method we can verify title of the page also we use assertion to verify the title of the page
  Ex : cy.title().should('eq','my page title')

34) How to handle 1st tittle visible and 2nd tittle not visible

- Using conditional testing we can make it pass the test cases in this way we can make it visible

35) Assertions in cypress

- .should and .expect
  Positive assertion - cy.get("locator").should("be.visible")
  Negative assertion - cy.get("locator").should("not.be.visible")
- also using .should assertion we can check visible or not

36) How to handle Mouse events

- We can handle Mouse events by using .trigger, .right click(), .double click()
  cy.get("target").trigger("mouseover/down/up")
  cy.contains("target").doubleclick() and cy.contains("target").rightclick()

37) How to handle Keyboard events

- We can handle Keyboard events by using .type
  cy.get("locator").type('{enter}')

38) Parallel testing in cypress

- In cypress parallel testing is not possible if we use cypress parallel plugins we can do

39) GIT

- GIT is the source code management tool or version control tool to push/pull/clone the code to remote location

40) GIT Hub

- GIT Hub is the remote location place where we are going to save our code

41) How login will work in your framework

- we use cy.session() method for login so automatically it skip the login page every time

42) Cypress framework

- we are using Mocha framework

43) Different Cypress frameworks

- Mocha framework and BDD cucumber framework
  By default we have Mocha framework and in Mocha framework we use describe and it blocks so describe for test suite writing and It blocks for test cases writing

44) Cypress Report generating

- we use mochasam report for report generating in the form of html reports
  and also we use cypress dashboard or cypress cloud reports for report generating

45) Cypress plugins

- Xpath, drag and drop, file download, file upload

46) Cypress components

- 1. Cypress test runner
- 2. Cypress dashboard or Cypress cloud reports

47) Cypress methods or Cypress actions

- . Click, . Type, . Check, . Uncheck, . Select

48) Cypress tags

- . Only, . Skip

49) Cypress browser navigation commands

- cy.visit, cy.go, cy.(+1) for forward, and cy.(-1) for backward

50) Cypress cookies

- cy.clearcookies() so it removes all the cookies

51) Debugging in Cypress

- Using cy.debug() or .debug() method

52) How to click hidden elements in cypress

- using .click({force : true}); and .invoke method we can

53) Commands Executing

- cy.exec("npx cypress open")

54) Can we add jar files in Cypress

- we cannot add jar files in cypress but we can get it in npm packages in package.json by run a command npm install cypress

55) Cypress Viewport

- Cypress default viewport is (1000,660) cy.viewport(1000w,660h)

56) Why do we use node js in cypress

- To run JavaScript codes so we use node js

57) How to run two or more scenarios at a time

- using .only we can run

58) How to run two fixture files at a time

- using data driven testing we can run two fixture files at a time

59) How to display messages or how to print messages in cypress

- using cy.log we can

60) How to manage font colors in Dom

- using css - Right click -> inspect -> go to CSS -> change font

61) How to scroll in cypress

- cy.scrollTo() or cy.scrollTo("center") or cy.get("locator").scrollTo("center")

62) Alias in Cypress

- reuse the elements across the test cases

63) Cypress CLI

- Cypress command line interface to run a test cases in command line

64) How to handle check box in cypress

- using .check() and .uncheck() method we can handle check box

65) How to handle multiple and specific check boxes in cypress

- cy.get("locator").check(["css", "xpath"]) for multiple check boxes
- cy.get("locator").check("css") for specific check boxes

66) How to handle radio buttons in cypress

- using .check() method we can handle radio buttons
  uncheck will not work for radio buttons

67) How to handle drop downs in cypress

- using .select method we can handle drop downs
  cy.get("locator").select("04")

68) How to handle file download in cypress

- using cy.downloadFile we can handle

69) How to handle file upload in cypress

- using .attachFile or.selectFile we can handle

70) Cy.wrap → to yield the resolved values

71) Cy.find → to navigate the child elements

72) Cy.then → to resolved the promises

73) Cy.intercept → to intercept the network calls

74) CSS Locators → Cascading Style Steets

1. **Basic css locator**
   tagname[attributename = ‘attributevalue’]
   cy.get(‘input[name = “username”]’)

2. **Id locator**
   #id value
   cy.get(‘#newpassword’).type()

3. **Class locator**
   . classvalue
   . username

4. **Starts with – prefix locator** → we use ^ cap symbol
   tagname[attrname^ = ‘prefix value of attr’]
   input[name^ = “name”]

5. **Ends with – suffix locator** → we use $ dollar symbol
   tagname[attrname$ = ‘suffix value of attr’]
   input[name$ = “name”]

6. **Multiple attribute locator**
   tagname[attributename = ‘attributevalue’][attr2 = ‘attr2value’]
   input[name = ‘username’][placeholder = ‘username’]

7. **Contains locator**
   tagname[attrname* = ‘prefix value of attr’]
   input[name* = ‘name’]

8. **Indexing locator**
   cy.(‘locator’).eq(3). click()

75) Push the code before PR

- 1st we need to create a file then add the script then run a command git add. then we need to add the files so it add the files or click on + button it add all the files then once we committing the files we have to push the files

76) How to rise PR

- After automating one test case we need to rise a PR and once PR is reviewed by TL then we resolved comments and once PR is approved we merged the particular script to master branch

77) How to resolved PR

- 1st we need to push the origin code then we accepting the incoming changes then we modify once we push it so it's resolved the PR

78) Functional testing

- Testing each and every component of the application and also we test as per the requirement feature is working or not

79) Non functional testing

- Here we test performance testing, usability testing, compatibility testing

80) Smoke testing(verification testing)

- Testing the critical functionality of the application and also we test build is stable or not

81) Sanity testing

- Testing verified bugs are resolved or not and also we test the sanity of the application

82) Regression Testing

- Testing the unchanged features

83) Re testing

- Testing only the fixed bugs

84) Integration Testing

- Testing the data flow between 2 modules

85) Compatibility Testing

- Testing the application in different software and hardware environment

86) SDLC - Software development life cycle

- It is a procedure to develop the software application

Requirement Analysis
|
Design
|
Coding
|
Testing
|
Deploy

87) STLC - Software testing life cycle

- It is a procedure to test the software application

Requirement Analysis
|
Test plan writing
|
Test case writing
|
Test environment setup
|
Test case execution
|
Defect tracking
|
Test closure reports
|
Retrospect meeting

88) Defect life cycle

- To track the bug and check the status of the bug we use defect life cycle

Open —--------- Deffered/postponed
| Duplicate
Assign Reject
| Cannot be fixed
Fixed Not reproduceble
|
Retest
|
Close or Reopen

89) Different types of bugs

- **Blocker -** it block the module so we cannot test
- **Critical -** application is working but not getting result
- **Major -** we can continue the testing
- **Medium -** 1st name is taking but last name is not taking still we can save
- **Small or Trivial or Cosmetic -** small issues like spelling, alignment, colors, shape

90) Bug Severity

- Severity is how it will affect to functionality
  S1 - Critical - block the testing completely
  S2 - High - it's working but not getting result
  S3 - Medium - application doesn't meet certain criteria or unnatural behavior
  S4 - Low - it's a valid defect but not impact on functionality

91) Bug Priority

- Priority is how soon developer should fix the issue
  P1 - High - must be fixed immediately
  P2 - Medium - unnatural behavior
  P3 - Low - it's not impact on functionality

92) Defect

- Expected result is not equal to actual result or feature is not working according to the requirement

93) Bug

- Bug is a informal name of defect

94) Error

- It is a mistake done in the programming

95) Project members

- Total 8 members
  Product owner or client
  Scrum master
- Scrum members
  2 front end developers
  2 back end developers
  2 testers

96) Scrum artifacts

- Modules → Features → Epic 1 → User story 1 → User story 2 → 5 Story points → Epic 2 → User story 1 → User story 2 → User story 3 → 7 Story points
- Sprint duration is usually 2 weeks

97) Scrum events

- Scrum meeting or stand up meeting
- Sprint planning meeting
- Sprint review meeting
- Sprint retrospect meeting
- Estimation meeting
- Grooming meeting

98) Sprint velocity

- How many story points we completed in a day

99) When we are going to release the build to end user

- Usually after completing of 2 or 3 sprints
- Some case after completing 1 sprint also we can release

100) Jenkins

- Jenkins is a CI CD tool - continuous integration continuous development tool using this tool all our release activities will be automated
- In Jenkins we can create a jobs and pipeline based on our requirements
- Dev ops team will setup the Jenkins activities

101) NAN and IOT

- not a number and Internet of things

102) Why you choice cypress automation

- I haven't choice this its my client reference

103) Why did you choose this job

- Because of carrier growth

1) ES6

- It is a version of javascript language
  Let, Const, Arrows, Classes, Arrays, Strings, Operators, Parameters, Loops

2) Difference between Var, Let, Const

- **Var : Let : Const :**

1.Var is global scope	1.Let is block scope	1.Const is block scope
2.Redeclare a variable is possible	2.Redeclare a variable is not possible	2.Redeclare a variable is not possible
3.Reassign the value is possible	3.Reassign the value is possible	3.Reassign the value is not possible
4.Hosting is possible	4.Hosting is possible	4.Hosting is not possible
5.Var is available from initial version of javascript	5.let is available from ES 6 version	5.Const is available from ES 6 version

- Hoisting define a variable before declaration

3) Variables

- Variables is a memory location we can stored data and can be declared in var, let, const

4) Coersion

- it is a dynamically typed language and automatic conversion of values from one data type to another data type

5) Constructors

- Constructor is a special type of method where name is same as class name

6) Arrows =>

- arrow functions are concise syntax for defining functions

7) Functions ()

- Set of statements
- to avoid repetitive coding lines we can write a functions

8) Commenting And Uncommenting

- Commenting - ctrl + k + c and Uncommenting - ctrl + k + u

9) Concatenation

- Joining two strings

10) Data Types

- Different types of datas can be stored in variables
  There are 2 types of data types
  - Primitive data types
  - Non Primitive data types

- **Primitive data types -** primitive data types are the predefined data types provided by javascript language also called as in built data types
  - string - x = “kiran”
  - number - x = 10
  - boolean- x = true or x = false
  - null - empty or no values
  - undefined - undefined or not assigning the values

- **Non Primitive data types -** the data types are derived from primitive data types of javascript language also called as derived data types or reference data types
  - arrays -
  - objects -
  - regular expressions -

11) Difference between Primitive data types and Non Primitive data types

- **Primitive data types**. **Non Primitive data types**

cannot be modified	can be modified
store one type of data	store more than one type of data
we can call methods	we cannot call methods
starts with lowercase	starts with uppercase
size depends on type of data structure	size is not fixed

12) Operators

- operator is a symbol used to perform operations on variables and values
  - arithmetic operators
  - relational operators
  - logical operators
  - ternary operators
  - assignment operators
  - bitwise operators
  - increment and decrement operators

13) Difference between = = and = = = operators

- = = comparing values, it indicates condition is true, and values are equal
- = = = comparing data types, it indicates condition is false and values or not equal

14) Strings

- Sequence of characters
  written in double quote or single quote
  let num = "10" or let num = '10'
  let str = "kiran" or let str = 'kiran'

- Convert string to number
  let num = "10".
  num1 = number(num)
  console.log(num1)

  Convert number to string
  let num = 20
  num1 = string(num)
  console.log(num1)

- **trim -** remove the space at beginning and end of the string
- **Split -** convert string to array
- **Join -** convert array to string

15) Arrays

- Collections of same data types
  ex : let arry = {1, kiran, true}

  sort and reverse
  let arr1 = [1,2,3,4,5]
  arr2 = arr1.sort()
  console.log(arr2)
  arr3 = arr1.reverse()
  console.log(arr3)

  o/p : [1,2,3,4,5]
  [5,4,3,2,1]

16) Arrays methods

- **Push :** adding element at the end of the array
- **Unshift :** adding element at the beginning of the array
- **Pop:** delete the element at the end of the array
- **Shift :** delete the element at the beginning of the array
- **Length :** to get the array length
- **Concatenation :** joining two arrays
- **Slice :** select part of the array and returns new array
- **Splice :** remove existing elements and add new elements

17) Oops

- Oops is a object oriented programming language and it's deals with real world entity using programming language
- **Class**
- **Objects**
- **Inheritance**
- **Encapsulation**
- **Polymorphism**

18) Class

- Class is collections of objects

19) Objects

- Objects is collections of properties
    - add properties
    - delete properties
    - edit properties

20) Inheritance

- Acquiring properties from parent class

21) Encapsulation

- Data binding between two methods in a class

22) Polymorphism

- Polymorphism is the ability to create many forms
  2 types in polymorphism
- **method overloading -** multiple methods can have the same name with different parameters in a class
- **method overriding -** override the parent class and display the child class

23) Async And Await

- **async** is used to define a function that returns a promise
- **await** is used to stop the function until resolved the promise

24) Synchronous and Asynchronous

- synchronous executes the code line by line and must finish each line before moving to next one
- synchronous executes the code sequentially

- Asynchronous not executes the code line by line and without finishing each line can move to next one
- asynchronous executes the code independently

Self Introduction

- Hi Iam Kiran G, Iam from Bangalore
- total i have 5 years of experience in software field
  3 years of experience in Accend systems pvt ltd as a automation tester and
  2 years of experience in Accenture as a project coordinator
- and in my previous company i was a part of AIEP health care project
- and I have a strong knowledge on automation testing like cypress with javaScript using mocha framework and source code management tool like Git and Git hub and Jenkins for cicd purpose and also i worked on agile methodology and i used JIRA for bug reporting purpose

Roles and Responsibilities :

- Involved in writing test plan and test cases and writing automation test scripts
- and bug reporting and bug tracking and also I involved in web application UI automation testing, functional testing, regression testing, smoke testing