**1)What is the Cypress?**

Cypress is a modern JavaScript-based front-end testing tool written in node js. Cypress runs tests within the browser, which makes testing easier and more reliable. It can run on Windows, Linux and Mac OS.

**2)Explain Cypress architecture.**

Behind Cypress is a Node server process. Tools like Selenium operate outside the web browser while The Cypress engine is coupled within the browser.

Cypress and the Node process constantly interact, synchronise, and conduct tasks on each other’s behalf. Cypress operates at the network layer, reading and modifying web traffic in real time. This allows Cypress to modify not just everything coming in and out of the browser, but also code that may interfere with its ability to automate the browser.

**3)What are the Cypress features?**

1)Cypress can run tests and also execute commands on the browser. so tests are less flaky and much faster.  
2)Automatic wait  
3)Cypress can take snapshots and record videos of tests after executing each step. We don’t need to configure extra plugins like we do in Selenium.  
4)Built-in assertions available.  
5)Network traffic control.

**4)What browsers are supported by Cypress?**

Cypress supports the following browsers:-

Chrome, Chromium, Edge, Firefox, Electron

**5)How Cypress is different then Selenium.**

* Cypress Supports only JavaScript/Typescript languages while Selenium supports all major languages like C#, Java, Python, JavaScript, Ruby, etc.
* Selenium supports all major browsers Chrome, Edge, Internet Explorer, Safari, and Firefox while Cypress supports only Chrome, Firefox, and Edge.
* Configuration is difficult in Selenium however Cypress comes with ready to use framework.
* Cypress is built on top of the mocha framework.

**6)What Testing Framework does Cypress come with?**

We cannot use the Junit or TestNG in Cypress, Cypress comes with Mocha and Chai assertion libraries.

**7)How can I install the Cypress tool?**

1. Use the node package manager

npm init

2. Install Cypress dependency

npm install cypress --save-dev

3. Open Cypress

npx cypress open

**8)How to create a test suit in Cypress?**

A test suit refers to a collection of test cases. We can create a test suit in Cypress using describe() block or context() block. The**describe()** block acts as a suite, and inside that block, each test can be written as a single **it()** block.

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

it('Tc01', () => {

//code

})

})

Using Context() block

describe('My Sample Test Suite', () => {

context('Login Tests', () => {

it('User should log in successfully', () => {

// Your login test code here

cy.visit('/login');

cy.get('#username').type('testuser');

cy.get('#password').type('password123');

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

cy.url().should('include', '/dashboard');

});

it('should display an error message for invalid credentials', () => {

// Your invalid login test code here

cy.visit('/login');

cy.get('#username').type('invaliduser');

cy.get('#password').type('invalidpassword');

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

cy.contains('Invalid credentials').should('be.visible');

});

});

**9)How to interact with DOM elements in Cypress?**

Cypress only supports CSS Selector. However, we can use the **Cypress-Xpath** plugin to work with Xpath.

**10)Can we use XPath in Cypress?**

Cypress by default does not support XPath. However, there is one plugin **Cypress-Xpath** by which we can use XPath to interact with DOM elements.

install this plugin using the npm

npm install -D @cypress/xpath

Go to support->e2e.js and add the following dependency

require('@cypress/xpath');

**11)Can we use BDD with Cypress?**

Cypress does not offer official BDD built-in support, however, the NPM[**Cypress-Cucumber-Preprocessor**](https://www.npmjs.com/package/cypress-cucumber-preprocessor)plugin allows you to write your tests in BDD Cucumber Syntax. This plugin automatically translates Gherkin syntax into Cypress.

**12)How to verify the title of the page in Cypress?**

There are built-in assertions available in Cypress. By using the **“should”** assertion we can verify the title of a webpage in Cypress. cy.title() gets the title of the page and .should() checks if the title is equal to the expected value.

cy.title().should('eq','My Site Title')

**13)What is a Cypress environment variable?**

Environment variables are used to customize the behaviour of test runs based on different environments, configurations, or runtime conditions. There are multiple ways using which environment variables can be configured in Cypress test automation framework.

**1)Using Cypress Configuration File**

Go to the cypress.json file and add configurations.

{

"env": {

"baseUrl": "https://example.com",

"username": "testuser",

"password": "yourPassword"

}

}

These configurations can be accessed in test scripts using Cypress.env()

const baseUrl = Cypress.env('baseUrl');

const username = Cypress.env('username');

const password = Cypress.env('password');

**2. Using Command Line**

Environment variables or configurations can be passed at the execution time from the command line.

npx cypress run --env baseUrl=https://aqh.com,username=testuser,password=yourpassword

**14)What are hooks in Cypress?**

Cypress hooks are used to define or set preconditions that we want to execute before a set of tests or before each test. For example, if we want to read test data from a fixture file or we want to perform some configuration tasks, or cleanup tasks then we can utilize Cypress hooks concepts. The available hooks in Cypress are before(),beforeEach(), after() and afterEach().

**15)How to read values from the Configuration file in Cypress.**

We can read the configuration values from Cypress using Cypress.config();

For example, if we have defined the env in the config.json file like this:

{"env": stage}

same can be accessed in scripts like

let timeout = Cypress.config('env')

**16)How is the test data maintained in Cypress?**

The fixtures folder keeps all the necessary data in the Cypress project. It helps us to get the input data from external files. This directory can store multiple JSON files and these JSON files are read by multiple tests. All fixture data has to be declared within the before hook block.

Refer to the sample fixture file.

{

"Username": "AQH",

"City" : "DELHI"

}

cy.fixture('sampleData.json').then((*data*) => {

cy.log(`Name: ${data.name}, Age: ${data.age}`);

});

path: Location of the JSON file.

**17)Can Cypress tests be run in a headless mode?**

Cypress by default executes tests in headless mode. However, we can explicitly define the headless flag as mentioned below:

npx cypress run --headless

In headless mode, Cypress executes the tests in the Electron browser without displaying the graphical user interface.

**18)How do you verify that a button is visible or not?**

To verify whether the button is visible or not on the webpage use inbuilt Cypress assertion.

cy.get('button#form-submit').should('be.visible')

**19)How many types of assertions are available in Cypress?**

Assertions are the validations that verify that this behaviour or functionality is intended or not. There are 2 types of assertions available in Cypress.

**1)Implicit Assertions**: These are inbuilt assertions. We can have positive as well as negative implicit assertions. Perfect examples of implicit assertions are should() or and() type assertions.

1.1)Positive Assertion

cy.get('li.todo').should('have.length', 3)

1.2)Negative Assertion

cy.get('li.todo').should('not.have.length', 2)

**2)Explicit Assertions**: Explicit assertions are used to perform some custom logic before applying assertion. An example of explicit assertion is the “**expect()”** assertion.

const employee =

{ name: 'QA', age: 30, }

assert.isObject(employee, 'value is object')

**20)How to perform API testing in Cypress?**

To perform API testing in Cypress use cy. request().This command is used to request a specific URL.

describe('API Testing in cypress', function () {

it('Hit Get Request validate its response status code and body', () => {

cy.request({

method: 'GET',

url: 'https://randomuser.me/api/',

qs: 'results=1'

}).then((*response*) => {

expect(response.status).to.eq(200)

expect(response.body).to.have.property('info')

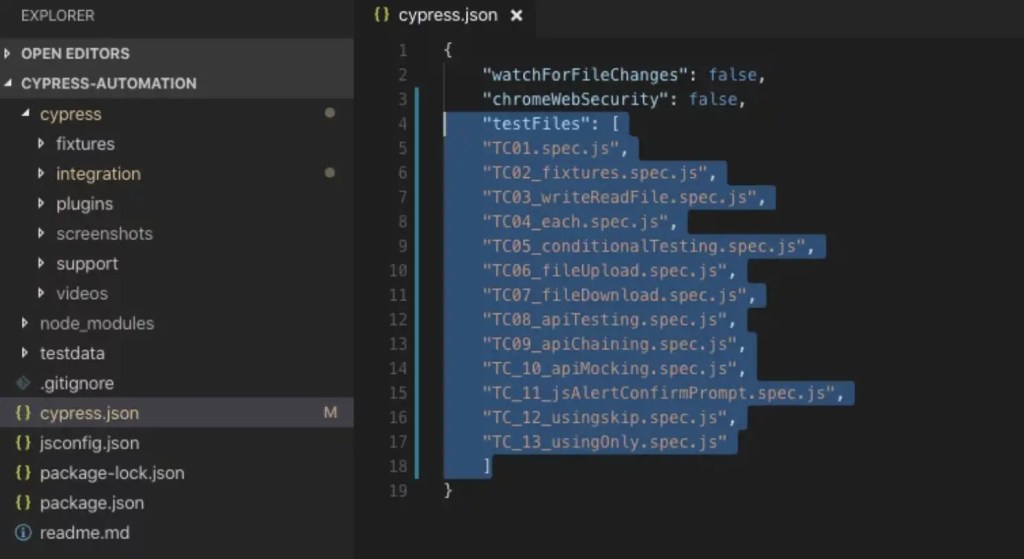
})

})

})

**21)How to execute tests in a particular order in Cypress?**

To execute Cypress tests, put all the file names in the desired order in the Cypress.JSON file under the**test files** property section.



**22)How to handle reusability in the Cypress framework?**

By default Cypress includes the index.js and commands.js in the Support folder. Index.js executes before every test file. We can put reusable behaviour such as custom commands and global overrides in this folder.

**23)What are the drawbacks of the Cypress testing tool?**

* Supports only JavaScript.
* No support for multiple tabs.
* Iframes are not supported.
* Only available for Web Testing.

**24)How to perform browser navigation in Cypress?**

Cypress has a go() command to navigate forward and backward in the browser.

1. To go forward

cy.go('forward')

or

cy.go(1)

2. To go backwards

cy.go('back')

or

cy.go(-1)

**25)How to click a hidden element in Cypress?**

cy. get(.checkbox) is trying to interact with a web element having the ‘.checkbox’ class, then the .check() method is called to check a checkbox or radio button. Since the element is hidden so { force: true } option is used to force the action, which means it will check the checkbox even if it’s not visible or is disabled.

cy.get('.checkbox').check({ force: true })

**26)How to skip a test in Cypress?**

We can use **it.skip()** or **describe.skip()**. That it/describe block will be skipped during test execution.

**27)How to upload a file in Cypress?**

By using .selectFile() command, a file can be uploaded easily.

cy.get('#upload')

.selectFile('cypress/images/logo.png')

**28)How to scroll into view in Cypress?**

Cypress has the **scrollIntoView()** command which scrolls an element into view.

cy.get('footer').scrollIntoView()

or

cy.get('#nav').scrollIntoView({ offset: { top: 150, left: 0 } }) //scroll using offset

**29)How to select the child element in Cypress?**

Use the cy. children() command to select the child element of a given element.

cy.get('div.icon-nav > ul').children('.mobile-search').click()

**30)Explain the .contains() in Cypress?**

cy. contains() command is used to search for a specific text on the webpage. It can be used in two ways:

1) cy.contains('testing')

2) cy.get('#footer').contains('Terms and conditions')

**31)What are the other alternatives to the Cypress tool?**

Several alternatives are available in the market of Cypress. A few of them are as follows.

1.WebdriverIO

2. CasperJS

3. Playwright

4. Jest

**32)How to generate a test report in Cypress?**

Cypress does not provide built-in support to generate test reports but we can integrate third-party plugins like Mochawesome, and Allure to generate HTML reports. Please visit this [article](https://automationqahub.com/how-to-generate-html-report-in-cypress-framework/) to learn how to generate reports in the Cypress framework.

**33)How do logging in to Cypress?**

We don’t need to add any plugin to perform logging in Cypress. Cypress provides the **“cy.log()”** command. This command prints the message on the console.

**34)What is the role of the cy.visit() command in Cypress?**

The cy. visit() command is used to navigate to a specific URL or route in Cypress tests. It allows you to start your test at a specific page or state.

{

"baseUrl" : "https://www.programsbuzz.com/"

}

describe('Automating The Signin',()=> {

it('visit the site ',()=>{

cy.visit('/')

})

})

**35)How to check your application’s responsiveness?**

To check the application’s responsiveness we can use “cy. viewport()” command. It manages the orientation and dimension of the application.

describe('Test to get window size', function() {

it('Get the window size', function() {

cy.viewport()

cy.viewport(320, 480)

})

})

**36)How to capture screenshots in the Cypress framework?**

In Cypress, we don’t need to add any dependency or plugin to capture screenshots like we used to do in Selenium. Cypress has the inbuilt capability to capture screenshots of the fail tests. It also allows us to take explicit screenshots during test execution. For example, if for debugging purposes later I want to take a screenshot of a step where the user provided some input then we can do this by using the following command.

cy.screenshot()

This post covers a wide range of topics related to **Cypress interview questions**. Gain Practical experience in Cypress tool to understand the topics.

**37)How do you scroll in Cypress?**

Cypress provides several commands to perform the scroll operation. Some of them are as follows:

**1). scrollTo() Command:**

This command scrolls the entire viewport or a specific element into view. We can provide the specific coordinates or positions like top, bottom etc.

1).cy.scrollTo(0, 500);

2).cy.scrollTo("topRight");

3).cy.scrollTo("bottom");

4).cy.scrollTo(0, 100);

**2. scrollIntoView() command**

This command helps in scrolling till the specific element is visible.

cy.get('#elementId').scrollIntoView();

**3.scrollBy() Command:**

This command lets us scroll by a specified number of pixels.

cy.get('body').scrollBy(0, 200);

**38)Can Cypress be used for testing mobile applications?**

As per Cypress’s Official documentation, Cypress will never support native mobile app automation but it can be used to test the mobile applications that are developed in a browser, such as with the Ionic Framework.

39**)****What is cy.task() functions in Cypress?**

**cy.task()** is the powerful command in cypress which helps to create some set of actions outside the scope of cypress. You can execute any node commands using the task. There are lot of things we can do with the **cy.task()** below is just one example.

**Example:**

**Problem: cy.log()** doesn’t work if you are running cypress **CLI**, we can’t use **console.log()** directly on our cypress framework to fix this issue we can take a help of task.

### 40) How can I run the tests in headless mode?

Cypress support headless mode, we can pass it as command line parameter

cypress run --headless --browser chrome

### 41) How can I debug in Cypress?

Cypress provides two way to debug the tests.

Using cy.pause()

Using .debug

**Option 1:** Using cy.pause

When we use the cy.pause()  to pause the execute, we can do intended action  manually and we can continue next

cy.visit('https://www.google.com')

cy.pause() //pauses the execution here

**Option 2:** Using .debug

To use this command we need to open the developer tools, This will set at debug point and it will pause the execution in that point

cy.visit('https://www.google.com')

cy.debug() //pauses the execution here