**🧪 Cypress Interview Questions & Answers**

**✅ Beginner Level**

**1. What is Cypress?**  
**Answer:**  
Cypress is a front-end testing tool built for the modern web. It allows developers to write end-to-end, integration, and unit tests in JavaScript.

**2. How is Cypress different from Selenium?**  
**Answer:**

* Cypress runs directly in the browser and executes in the same run loop as the application.
* Selenium uses WebDriver and runs outside the browser.
* Cypress is faster and easier to debug due to its architecture and automatic wait.

**3. What types of tests can you write in Cypress?**  
**Answer:**

* End-to-end tests
* Integration tests
* Unit tests

**4. What is the default testing framework in Cypress?**  
**Answer:**  
Cypress uses **Mocha** as the test framework and **Chai** for assertions.

**✅ Intermediate Level**

**5. How do you handle API calls in Cypress?**  
**Answer:**  
Using cy.request() to make API calls or intercepting network requests using cy.intercept() to stub and assert on API interactions.

**6. How do you handle test data or fixtures in Cypress?**  
**Answer:**  
Cypress supports fixtures (static test data) via the fixtures folder. You can load them using cy.fixture('file.json').

**7. How does Cypress handle waits?**  
**Answer:**  
Cypress automatically waits for DOM elements to appear and animations to complete. You can also use cy.wait() for manual waits.

**8. How to run Cypress tests headlessly?**  
**Answer:**  
Use the CLI command:

bash

CopyEdit

npx cypress run

**✅ Advanced Level**

**9. How can you do cross-browser testing in Cypress?**  
**Answer:**  
Cypress supports Chrome, Edge, Firefox, and Electron. You can specify the browser using:

bash

CopyEdit

npx cypress run --browser firefox

**10. How do you manage test retries in Cypress?**  
**Answer:**  
You can configure test retries using retries in the config file or per test:

js

CopyEdit

it('retries test', { retries: 2 }, () => {

// test logic

});

**11. Can Cypress be used for mobile testing?**  
**Answer:**  
Indirectly. You can simulate mobile viewports with cy.viewport() but cannot test native mobile apps.

**12. How do you maintain custom commands in Cypress?**  
**Answer:**  
Custom commands are defined in cypress/support/commands.js.

**✅ Code-Oriented & Practical Questions**

**1. How do you assert the value of an input field in Cypress?**

js

CopyEdit

cy.get('input[name="email"]').should('have.value', 'test@example.com');

**2. Write a test to log into a web app using Cypress.**

js

CopyEdit

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

it('logs in successfully', () => {

cy.visit('/login');

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

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

cy.get('form').submit();

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

});

});

**3. How would you check that a button is disabled/enabled?**

js

CopyEdit

cy.get('button#submit').should('be.disabled');

// or

cy.get('button#submit').should('not.be.disabled');

**4. Write a custom Cypress command for login.**

js

CopyEdit

// In commands.js

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

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

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

cy.get('form').submit();

});

**5. How do you retry an element in Cypress?**  
Cypress automatically retries until a timeout, but you can increase timeout manually:

js

CopyEdit

cy.get('.loading-spinner', { timeout: 10000 }).should('not.exist');

**6. How would you check if a table contains a specific value?**

js

CopyEdit

cy.get('table').contains('td', 'Expected Value');

**🎭 Playwright Interview Questions & Answers**

**✅ Beginner Level**

**1. What is Playwright?**  
**Answer:**  
Playwright is an open-source Node.js library by Microsoft for browser automation, supporting Chromium, Firefox, and WebKit with a single API.

**2. What languages does Playwright support?**  
**Answer:**  
JavaScript, TypeScript, Python, Java, and C#.

**3. How do you launch a browser in Playwright?**  
**Answer:**

js

CopyEdit

const { chromium } = require('playwright');

const browser = await chromium.launch();

**✅ Intermediate Level**

**4. How do you handle waits in Playwright?**  
**Answer:**  
Playwright uses auto-waiting and you can also use explicit waits like page.waitForSelector().

**5. How do you perform API testing in Playwright?**  
**Answer:**  
Using the request context:

js

CopyEdit

const request = await playwright.request.newContext();

const response = await request.get('https://api.example.com/data');

**6. How can you handle authentication in Playwright?**  
**Answer:**  
By either storing session cookies or using storageState to persist login sessions.

**7. How to run Playwright tests in parallel?**  
**Answer:**  
Playwright Test Runner supports parallel execution out of the box. It runs files and tests in parallel by default.

**✅ Advanced Level**

**8. How do you perform cross-browser testing in Playwright?**  
**Answer:**  
Playwright allows running tests across Chromium, Firefox, and WebKit:

js

CopyEdit

for (const browserType of ['chromium', 'firefox', 'webkit']) {

const browser = await playwright[browserType].launch();

}

**9. How do you mock network responses in Playwright?**  
**Answer:**  
Using page.route():

js

CopyEdit

await page.route('\*\*/api/\*', route => {

route.fulfill({

status: 200,

body: JSON.stringify({ data: 'mocked' })

});

});

**10. What is Playwright Test Runner?**  
**Answer:**  
An integrated test runner that comes with Playwright. It provides fixtures, test retries, parallel execution, and rich reporting.

**11. How do you take screenshots or videos in Playwright?**  
**Answer:**  
Screenshots:

js

CopyEdit

await page.screenshot({ path: 'screenshot.png' });

Videos (in config):

js

CopyEdit

use: {

video: 'on',

}

**1. Write a Playwright test to validate a successful login.**

js

CopyEdit

test('Login test', async ({ page }) => {

await page.goto('https://example.com/login');

await page.fill('#username', 'admin');

await page.fill('#password', 'password123');

await page.click('button[type="submit"]');

await expect(page).toHaveURL(/.\*dashboard/);

});

**2. How do you handle file uploads in Playwright?**

js

CopyEdit

await page.setInputFiles('input[type="file"]', 'path/to/file.pdf');

**3. Write a function to check if an element is visible or hidden.**

js

CopyEdit

const isVisible = async (page, selector) => {

return await page.isVisible(selector);

};

**4. How do you create a reusable login function?**

js

CopyEdit

async function login(page, user, pass) {

await page.goto('/login');

await page.fill('#user', user);

await page.fill('#pass', pass);

await page.click('#submit');

}

**5. How would you test a dynamic dropdown using Playwright?**

js

CopyEdit

await page.click('#dropdown');

await page.click('text=Option 2');

await expect(page.locator('#dropdown')).toHaveText('Option 2');

**6. How do you capture console logs during test execution?**

js

CopyEdit

page.on('console', msg => console.log(msg.text()));

**7. Write a test to validate a REST API response using Playwright.**

js

CopyEdit

const request = await request.newContext();

const response = await request.get('https://api.example.com/data');

expect(response.status()).toBe(200);

const json = await response.json();

expect(json.name).toBe('John');

**🔄 Scenario-Based Logic Questions**

1. **How would you test a shopping cart that updates totals dynamically?**
   * Add multiple items
   * Assert cart count
   * Check price calculations
   * Verify final checkout price
2. **What if a CAPTCHA appears after 3 failed logins—how do you test it?**
   * Simulate 3 failed attempts using the login form
   * Assert CAPTCHA appears using visibility check
   * Mock/disable CAPTCHA in testing if needed
3. **How would you test a table that loads via pagination/AJAX?**
   * Scroll or click to next page
   * Wait for new rows with page.waitForSelector() or cy.intercept()
   * Validate row contents
4. **Write a Cypress/Playwright script to test a modal popup.**
   * Open modal
   * Validate content
   * Click outside or close button
   * Confirm modal is closed