### **Handling Multiple Tabs, Multiple Windows, and Alerts in Cypress**

Cypress, by design, doesn't support multiple tabs or windows directly because it aims to provide a consistent and reliable testing environment. However, you can interact with elements in the same window and handle scenarios like alerts, navigation commands, and various mouse events.

### **1. Handling Multiple Tabs and Windows**

While Cypress doesn't handle multiple tabs or windows natively, you can test the content of new tabs or windows by manipulating the href attribute and ensuring it loads within the same tab. Here's how you can handle such scenarios:

// Intercept a link that would open a new tab and force it to open in the same tab

cy.get('a[target="\_blank"]').invoke('removeAttr', 'target').click()

// Alternatively, use the visit command to navigate to the new URL directly

cy.get('a[target="\_blank"]').then(($link) => {

const url = $link.prop('href')

cy.visit(url)

})

### **1. Handling Web Tables**

Web tables are commonly used to display data in a tabular format. You might need to interact with the table, retrieve data from it, or make assertions.

<table id="webTable">

<thead>

<tr>

<th>Name</th>

<th>Age</th>

<th>City</th>

</tr>

</thead>

<tbody>

<tr>

<td>John Doe</td>

<td>30</td>

<td>New York</td>

</tr>

<tr>

<td>Jane Smith</td>

<td>25</td>

<td>Los Angeles</td>

</tr>

</tbody>

</table>

#### **Cypress Commands:**

**Selecting a Cell by Row and Column:**

// Select a specific cell

cy.get('#webTable tbody tr').eq(0).find('td').eq(1).should('have.text', '30')

**Iterating Through Table Rows:**

// Iterate through each row and column

cy.get('#webTable tbody tr').each(($row, index, $rows) => {

cy.wrap($row).within(() => {

cy.get('td').each(($cell, index, $cells) => {

// Do something with each cell

cy.log($cell.text())

})

})

})

### **3. Assertions for Visibility**

Cypress provides built-in assertions to check if elements are visible or hidden.

// Initially, the element should be hidden

cy.get('#myElement').should('not.be.visible')

// Click the show button and assert visibility

cy.get('#showButton').click()

cy.get('#myElement').should('be.visible')

// Click the hide button and assert invisibility

cy.get('#hideButton').click()

cy.get('#myElement').should('not.be.visible')

### **Comprehensive Example**

Here’s a comprehensive example combining web table handling, mocking a web application, and visibility assertions:

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

beforeEach(() => {

// Mock the API response

cy.intercept('GET', '/api/users', { fixture: 'users.json' }).as('getUsers')

})

it('Handles web tables, mocks API responses, and checks visibility', () => {

// Visit the application

cy.visit('/users')

cy.wait('@getUsers')

// Interact with the web table

cy.get('#webTable tbody tr').eq(1).find('td').eq(2).should('have.text', 'Los Angeles')

// Assert visibility of elements

cy.get('#myElement').should('not.be.visible')

cy.get('#showButton').click()

cy.get('#myElement').should('be.visible')

cy.get('#hideButton').click()

cy.get('#myElement').should('not.be.visible')

})

})