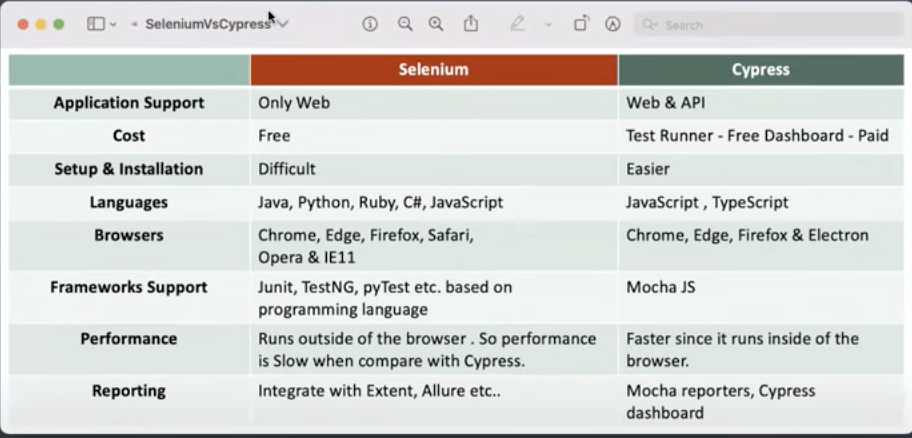
**CYPRESS Automation**

**Introduction:** Web automation tool for modern web apps (built on react js or angular js etc.). It uses by default javascript as the programming language. It does not use Selenium library. It is open source. Widely used components are Test Runner and Dashboard. Test runner is used to write and execute test cases, design framework etc.. Dashboard is a paid version and gives advanced features in it. Cypress is built on Node JS environment.

**Typical Users:** Developers and Testers. Used to automate end-to-end test cases, integration test cases, API testing, and unit test cases.

**Pros/Cons**

**Features of Cypress:**

Time Travel: Snapshot of every line run that you can refer to debug.

Debuggability: It has access to browser developer tools as it runs on the same browser internally.

Built-in waits: No need to implement Wait mechanism as the default wait mechanism is in cypress to handle waits

Consistent: it is consistent as compared to Selenium as it runs inside the browser. Whereas selenium employs drivers to interact with browsers which causes inconsistent results.

Screenshots/videos: will always be captured.

Cross-Browser Testing: Locally or remotely

CI/CD integration

**Limitations of Cypress**

Can’t automate window-based apps.

Limited browser support as compared to selenium.

Language limitations as it only supports JS and TS.

Reading and writing data on external files is a problem.

Limited reporting capabilities.

Third-party integration is limited.

**Pre-requisites:**

It requires nods.js /and VS code to write your code

**Installation and verification Commands**

npm -i init - created package.json

npx cypress open

**Application flow**

Select E2E for testing and Unit for development

Select the browser that you want to work with. Default is electron

Specs are a set of test cases that you will be writing

**Write a Test case**

1. **By using Cypress UI-** you can go directly and create a new spec with a spec name. name can be anything but it should be .spec.js.
2. **By using VS code-** You can directly write your test case with the below syntax. You can use the arrow function as well as the “function” keyword to define.

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

it('test 1', () => {

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

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

})

}

)

describe('Test Suite function', function() {

it('test 1', function() {

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

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

})

}

)

Describe block encapsulates it block. Describe block is used to define suite name whereas it blocks contain actual Test implementation

**Run a Test Case**

1. **By using Cypress runner portal- Select any spec and it shall run the spec and its test cases.**
2. **By using visual studio commands which are**

npx cypress run

npx cypress run —headed

npx cypress run —browser Chrome —headed

npx cypress run --spec cypress/e2e/mytest.cy.js --headed --browser chrome

**Locators in Cypress**

1. CSS Selector (you can use selector hub to make this easier)

To get CSS (tagname is optional)

Tagname id (tagnema#id)

Tagname class (tagname.class)

Tagname attribute (tagname[attribute =‘value’])

Tagname class attribute (tagname.class[attribute=‘value’])

cy.get() command is used to locate the element.

Syntax to locate the locator is

cy.get(locator).get()

2. Xpath (only when you have installed Cypress Xpath plugin)

**Important keywords**

.visit(“url”)

.title()

.should (‘eq’,’expected’)