**CYPRESS**

**Cypress is based on JavaScript:**

1. Unit tests.
2. End to end tests.
3. Integration tests.

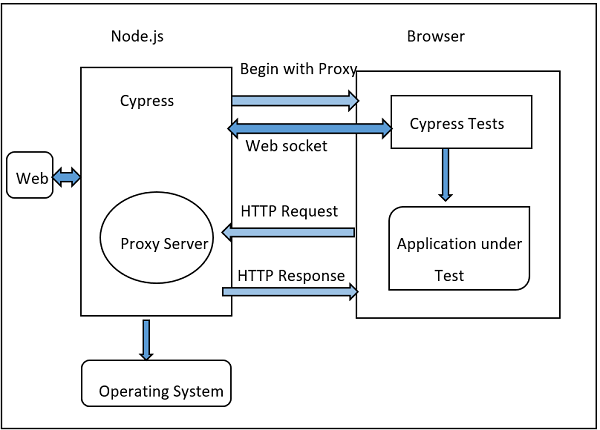
**Features:**

1. Supports Test-Driven development.
2. Provides Dashboard services.
3. Efficient debugging with Developer Tools accompanied with generation of stack trace and errors.
4. Provides the screenshots for failed tests.
5. Not necessary to add waits to stop the execution for some time. By-default, the waits are applied, prior to executing the following step or assertion.
6. Able to monitor and control the characteristics of server response, functions, and timers, which are essentially needed for unit testing.
7. Check and manage network traffic.
8. Allows the multi-browser support.
9. In-built feature to capture videos of execution is available.
10. Can be integrated with continuous integration tools.
11. Page responsiveness with viewport sizing.
12. Reloads changes applied to tests by default.
13. Friendly Application Programming Interfaces (APIs) are available.
14. Test runner available, which allows the test execution straight from the User Interface (UI).

**Disadvantages:**

1. It is only based on JavaScript.
2. A relatively new tool and hence, the community support is not extensive.
3. It cannot perform mobile testing.
4. Shadow Document Object Model (DOM) cannot be accessed.
5. Tabs/child windows are managed by workarounds.

**Architecture:**



.

**Cypress VS Selenium:**

|  |  |  |
| --- | --- | --- |
|  | **Cypress** | **Selenium** |
| **Languages Supported** | Supports JavaScript Only | Supports all popular languages like Java, Python, Ruby, C#, PHP, etc. |
| **Browsers Supported** | Chrome, Edge, Firefox, Electron | Chrome, IE, Safari, Edge, Firefox, Opera |
| **Frameworks Supported** | Supports only Mocha JS | Supports multiple frameworks based on specific programming languages. (For example: JUnit for Java, Cucumber for JavaScript, etc.) |
| **Setup Complexity** | The setup is simple. No dependencies or additional downloads required | Setup is a bit challenging as it requires downloading browser-specific drivers and setting up the test environment |
| **Documentation & Community Support** | Very intuitive documentation along with a rapidly growing community | Well-established documentation and firm community support from users across the globe |

**Installation:**

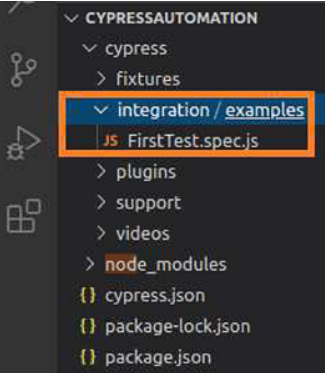
1. npm init
2. npm install cypress --save-dev

**Cypress Test Runner:**

1. Open Cypress by running − node\_modules/.bin/cypress open
2. node\_modules/.bin/cypress open

**Cypress Build First Test:**

**Cypress Folder Structure:**

****

* **fixtures** − Test data in form of key-value pairs for the tests are maintained here.
* **integration** − Test cases for the framework are maintained here.
* **plugins** − Cypress events (prior and post events to be executed for a test) are maintained here.
* **support** − Reusable methods or customized commands, which can be utilised by test cases directly, without object creation are created here.
* **videos** − Executed test steps are recorded in the form of videos and maintained here.
* **node\_modules** − Project dependencies from the npm are maintained in this folder.It is the heart of the Cypress project execution.
* **cypress.json** − Default configurations are set in this folder. The values of the current configurations can be modified here, which overrules the default configurations.
* **package.json** − Dependencies and scripts for the projects are maintained in this folder.

**Basic Test**

// test suite name

describe('Tutorialspoint Test', function () {

// Test case

it('Scenario 1', function (){

// test step for URL launching

cy.visit("https://www.google.com/");

});

});

**Test Execution:**

1. From the command line:

./node\_modules/.bin/cypress run

1. From the Test Runner:

./node\_modules/.bin/cypress open

1. For a specific file from command line:   
   cypress run --spec "<spec file path>"

**Supported Browsers:**

Cypress supports: Chrome, Electron, and Firefox

**Execution from other Browsers:**

./node\_modules/.bin/cypress run -- browser chrome

**Execution in headed mode:**

**./node\_modules/.bin/cypress run – headed**

**Cypress Basic Commands:**

* and

used to create an assertion and is an alias of .should ()

//element is visible & enabled

cy.get('#txt').should('be.visible').and('be.enabled')

//element is checked

cy.contains('Subject').and('be.checked')

* as

provides an alias for later usage

//alias element as parent

cy.get('#txt').find('li').first().as('parent')

* blur

blurs an element in focus.

//blur input

cy.get('#txt'). type('abc').blur()

* check

checks radio buttons or checkboxes and is applied to elements having input tags.

//checks element having class attribute chkbox

cy.get('.chkbox').check()

* children

obtains the sub elements of an element.

//obtains children of element n

cy.get('n').children()

* clear

removes the value from textarea or input.

//removes input abc

cy.get('#txt'). type('abc').clear()

* clearCookie

removes a particular browser cookie.

//clear abc cookie

cy.clearCookie('abc')

* clearCookies

removes the browser cookies from an existing domain and subdomain.

//clear all cookies

cy.clearCookies()

* clearLocalStorage

removes the local Storage data from an existing domain and subdomain.

//clear all local storage

cy. clearLocalStorage ()

* click

clicks an element in Document Object Model (DOM).

//click on element with id txt

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

* contains

obtains an element having a specific text. The element can have more than the text and still match.

//returns element in #txt having Tutor text

cy.get('#txt').contains('Tutor')

* dblclick

double-clicks an element in Document Object Model (DOM).

//double clicks element with id txt

cy.get('#txt').dblclick()

* debug

fixes a debugger and log values are returned by prior command.

//pause to debug at start of command

cy.get('#txt').debug()

* document

obtains window.document on the active page.

cy.document()

* each

iterates through an array having the property length.

//iterate through individual li

cy.get('li').each(() => {...})

* end

ends a command chain.

//obtain null instead of input

cy.contains('input').end()

* eq

 refers to an element at a particular index in an array of elements.

//obtain third td in tr

cy.get('tr>td').eq(2)

* exec
* find
* first
* get
* getCookie
* getCookies
* go
* visit
* next
* parent
* should
* wait
* title
* viewport
* log
* reload