# Interacting with Elements in Cypress

Below items can be used to interact with elements in the webpage,

1. .click() --> clicking
2. .dblclick() -->double clicking
3. .rightclick() -> mouse right clicking
4. .type() -> writing somethinginthe text box
5. .clear() -> clearing the text fields
6. .check() -> checkboxes to check
7. .uncheck() -> uncheck the check box
8. .select() -> dropdown
9. .trigger() -> mouse actions example mouse down , up , right
10. .selectFile() -> to upload the files
11. Download files
12. npm i --save-dev cypress-downloadfile
13. goto e2e.js and mention require('cypress-downloadfile/lib/downloadFileCommand')
14. goto cypress.config.js

const {downfile} = require("cypress-downloadfile/lib/addPlugin")

e2e: {

setupNodeEvents(on, config) {

// implement node event listeners here

on('task',{downfile})

},

}

Mostly all the options in cypress will hold as below,

.dblclick()

.dblclick(options)

.dblclick(position)

.dblclick(position, options)

.dblclick(x, y)

.dblclick(x, y, options)

**Micellaneous Commands:**

cy.go('back')§

cy.wait(500)

cy.intercept()

**Handling windows/Tab:**

Script demo

**Handling of Alert:**

Alerts afre segregated to 2 types ->

* browser based --> the same way we handle our windows/tabs
* Application based --> javascript alert boxes

Script demo

**Handling Frames:**

Installation: this is mandatory if we need to use the iframe/frame in the project

npm install -D cypress-iframe

where ever we need to implement it the cy.js file should hold the packages name

/// <reference types="cypress-iframe" />

import 'cypress-iframe'

**Handling of webtables:**

Script demo

Launch the website -> identify the frame(Css locator) -> navigate to the frame(Frameloaded) -> get the element inside the frame(cy.iframe.find) -> do the action(cy.iframe.find(css).click)

**Fixtures**

Script demo

**Handling Autocomplete**

Script demo

**Stub**

**What is Stub ?**

A stub is a piece of code that is used to stand in for another piece of code that is not yet available or is not possible to give you the desired outcome that you want.

cy. stub() is useful for **stubbing any function or method inside your application** for example: disabling the prompt function on the window object.

**Why stub?**

The stub function in Cypress is useful when you want to test unique scenarios in your application and make sure your application works as expected in every scenario and with various datasets that your application might get.

**Ways to implement**

* cy.stub() : replaces a function, and controls its behavior.
* cy.intercept(): Spy and stub network requests and responses.
* cy.spy(): To wrap a function in a spy, use the cy.spy() command.
* cy.clock(): To control time in the browser.

**Spy:**

* A spy gives you the ability to "spy" on a function, by letting you capture and then assert that the function was called with the right arguments, or that the function was called a certain number of times, or even what the return value was or what context the function was called with.
* spy does not modify the behavior of the function
* Syntax:
  + cy.spy(obj, 'method')

**Clock:**

* Control the date and time of application in order to avoid the application slowness
* **Clock works along with Tick**
  + Syntax:
    - cy.clock()
    - cy.visit('http://localhost:3333')
    - cy.get('#search').type('Acme Company')
    - cy.tick(1000)

Restore the clock back

cy.clock().then((clock) => {

clock.restore()

})

cy.clock().invoke('restore')

**Cypress Framework Concepts**

* Hooks
  + Carry out the pre and post requisites -> setup and tear down features
    - **Before** -> executed before all the “it” is run
    - **After**-> executed after all the “it” is run
    - **BeforeEach** -> executed before each “it” is run
    - **AfterEach** -> executed after each “it” is run

**Scenario1**: if only one it is available

before

beforeeach

it

aftereach

after

**Scenario2**: if two it is available

before

beforeeach

it

aftereach

beforeeach

it

aftereach

after

* Building Custom Commands
  + Re-usability
  + With the help of support -> **command.js**
  + What are the functions available in cypress.command
    - add -> add something new
    - addAll -> consolidated fucntion of two or more methods(multiple add inside the function)
    - Overwrite ->overwrite something
* Parameterise data **-->** fixtures
* POM - Page Object Model
* Config changes in cypress json
* Screenshots and videos, retries
* Environment variables -> command line CLI
* Cucumber implemention
* API - testing with cypress(extension of cy.intercept)
* Github - with cypress
* Jenkins - with cypress
* Docker with cypress

**Framework:**

**Design Pattern**

**POM -> Page Object Model**

**Pages -> locators -->Main page**

* **Locators page itself we can have some actions what the locator should do**
  + **Locator will exists in .js file extension**
* **Pages -> where we invoke the logics available in the locator page**
  + **Cxy.js file extension**

**Example:**

[**https://bstackdemo.com/signin**](https://bstackdemo.com/signin)

[**https://juice-shop.herokuapp.com/#/**](https://juice-shop.herokuapp.com/#/)

**Design Pattern:**

* Code maintainability
* Re-usability-avoid redundancy/codes don’t repeat
* Design principles
  + **SOLID**
    - Single Responsibility
    - Open closed
    - Liskov
    - Inteface seggregation
    - Dependency Inversion

**Configuration:**

**Command line difference:**

npx cypress open -> will ask for the file to be selected

npx cypress run -> all the files(with extension cy.js) unders e2e folder will run

npx cypress run --spec "cypress/e2e/my-spec.cy.js" -> this will run the file named "my-spec.cy.js" under the e2e folder

**other commands:**

npx cypress run --browser chrome 🡪 selecting browser from the commandline

npx cypress run --spec "cypress/e2e/my-spec.cy.js"

npx cypress run --spec "cypress/e2e/login/\*\*/\*" --> match a file name

npx cypress run --config pageLoadTimeout=100000,watchForFileChanges=false

**Run on specific env**

npx cypress run --env host=api.dev.local,port=4222

**Run Parallel** - We need cypress cloud

cypress run --record --parallel

**Run with tags** --> devops

cypress run --record --tag "staging"

**API Testing Cypress:**

* API (Application Programming Interface) is a set of definitions and protocols for building and integrating applications.
* Methods Supported -> GET,POST,PUT,PATCH,DELETE
  + GET -> fetch the data from API
  + POST -> add the data to api
  + PUT -> modify the data to api
  + DELETE -> delete the entry form API
  + PATCH-> Update the API entry
* How to add custom header to api
* Input-Output -> **Json (Javascript object Notation)**
* Path parameter and Query Parameter
  + Path Parameter 🡪 it will be with in the url
  + Query Parameter 🡪 it will be after a ? mark
* Response codes

**1xx** Informational response – the request was received, resuming the process.

**2xx** Successful – the request was received, comprehended, and accepted. Ex: 201,200

**3xx** Redirection – further action must be taken to complete the request. Ex:

302,312

**4xx** Client error – the request contains the wrong syntax or cannot be fulfilled. Ex:400,404

**5xx** Server error – the server failed to fulfill a valid request. Ex: 500

* Graph QL ->

GraphQL requests are all sent to a single endpoint. We can easily use Cypress to fetch all the uses in our GraphQL endpoint.

**Plugin :**

'npm i cypress-plugin-api'.

Usage:

import 'cypress-plugin-api'

To Print the response body 🡪 JSON.stringify(response)

**Cucumber With Cypress:**

**Cucumber:**

* Gherkins language -> **Given**, **When**, **And**, **Then 🡪 feature file (extension would be .feature)**
  + **Feature file** 
    - **Background**
    - **Scenario**
    - **Scenario outline 🡪 examples**
* Feature file 🡪 Step definitions 🡪 executes the logics for the application
* **Two packages** 
  + cypress-cucumber-preprocessor 🡪 deprecated
  + @badeball/cypress-cucumber-preprocessor
    - https://github.com/badeball/cypress-cucumber-preprocessor/blob/master/docs/quick-start.md

**Installation and Setup:**

Usual commands for new project,

npm init

npm install cypress --save-dev

npm install -g yarn

**Plugin:**

npm install --save-dev cypress-cucumber-preprocessor

npm i -D cypress cypress-cucumber-preprocessor

**Extension:** install a vs code extension

Name: Cucumber (Gherkin) Full Support

Id: alexkrechik.cucumberautocomplete

Description: VSCode Cucumber (Gherkin) Full Language Support + Formatting + Autocomplete

Version: 2.15.2

Publisher: Alexander Krechik

VS Marketplace Link: <https://marketplace.visualstudio.com/items?itemName=alexkrechik.cucumberautocomplete>

Add below item to support file ,

const cucumber = require('cypress-cucumber-preprocessor').default module.exports = (on, config) => {

  on('file:preprocessor', cucumber())

}

Add below item to json file,

"cypress-cucumber-preprocessor": {"nonGlobalStepDefinitions": true,

"stepDefinitions": "cypress/e2e/featurefile/DemoSite"

}

**Points to remember:**

1. nonGlobalStepDefinitions - false

feature file name and folder name is different

-> this scenario will execute

1. nonGlobalStepDefinitions - true

feature file name and folder name is different

-> this will fail because we make cypress to search with respect to the feature file name and folder name as same

**New Version:**

Install the [@badeball/cypress-cucumber-preprocessor](https://github.com/badeball/cypress-cucumber-preprocessor) using the below command

**npm install -D @badeball/cypress-cucumber-preprocessor**

**npm install -D @bahmutov/cypress-esbuild-preprocessor**

**npm install -D @esbuild-plugins/node-modules-polyfill**

**Next step we need to modify the code**

**Running with tags(Applicable for both) 🡪**

**cypress run --env tags=”@smoke”**

**cypress run --env tags=”@smoke or @Regression”**

**cypress run --env tags=”@smoke and @Regression”**

**Reporter:**

**Default cypress reporter ->**

**npm i --save-dev cypress-mochawesome-reporter**

**(or)**

**yarn add -D cypress-mochawesome-reporter**

**we need to update the cypress.config.js file with below items**

**inside the setupNodeEvent 🡪**  require('cypress-mochawesome-reporter/plugin')(on);

**we need to add below items to ,**

reporter: 'cypress-mochawesome-reporter',

reporterOptions: {

charts: true,

reportPageTitle: 'Sample run',

embeddedScreenshots: true,

inlineAssets: true,

saveAllAttempts: false,

},

**Cypress cucumber reporter**

**npm install multiple-cucumber-html-reporter --save-dev 🡪 this on is part of** cypress-cucumber-preprocessor

**Git Hub - Push the code to Git**

**Pre-requisite:**

* **Vscode**
* **Git hub Account creation**
* **Install git for mac/windows**

**Jenkins:**

* **Installation**
  + **Jenkins can be installed via home brew in mac**
  + **Jenkins can be installed via exe in windows**
* **Commands,**
* **Install the latest LTS version: brew install jenkins-lts**
* **Install a specific LTS version: brew install jenkins-lts@YOUR\_VERSION**
* **Start the Jenkins service: brew services start jenkins-lts**
* **Restart the Jenkins service: brew services restart jenkins-lts**
* **Update the Jenkins version: brew upgrade jenkins-lts**

**Mac start**

**sudo brew services start jenkins-lts**

**Search for the process**

**ps -e | grep jenkins**

**Kill the process**

**sudo kill 57696**

**Cypress with docker:**

**Q&A:**

**Prev 🡪 example**

**Alert to click on ok button**