pa

Contents

[1. Cypress Installation Pre-Requisites 1](#_Toc144829781)

[2. How to Install Cypress 1](#_Toc144829782)

[3. Cypress Settings (In Cypress.config.js) file 1](#_Toc144829783)

[4. To Open Cypress 2](#_Toc144829784)

[5. To Open Cypress Headless: 2](#_Toc144829785)

[6. JS commands: 2](#_Toc144829786)

[7. Cypress Commands 3](#_Toc144829787)

[8. Cypress Commands and Chaining: 6](#_Toc144829788)

[9. JQuery approach 7](#_Toc144829789)

[10. Embedded Commands 7](#_Toc144829790)

[11. Making Local and Global Variables: 7](#_Toc144829791)

[12. Before Function: 8](#_Toc144829792)

[13. Page object Oriented Model: 8](#_Toc144829793)

[14. GIT NOTES: 9](#_Toc144829794)

My Cypress Documentation:

# Cypress Installation Pre-Requisites

* Install npm (<https://nodejs.org/en/download>)
* Install Git Bash (<https://git-scm.com/downloads>)
* Install Visual Studio Code (<https://code.visualstudio.com/download>)
* In Visual Studio> Install Prettier, ES6 Mocha Snippets, Javascipt ES6 Code Snippets, Monokai pro, material icon theme

# How to Install Cypress

* In VS code Terminal > git bash> write “npm init”
* copy paste in vs code terminal using bash as terminal (npm install --save-dev [cypress@12.14.0](mailto:cypress@12.14.0))

# Cypress Settings (In Cypress.config.js) file

* For Cypress Xpath: Go to Support > e2e.js > paste: require("@cypress/xpath");
* In cypress.config.js file: add under e2e

Full code:

*const* { defineConfig } = require("cypress");

*module*.*exports* = defineConfig({

  e2e: {

    setupNodeEvents(*on*, *config*) {

*// implement node event listeners here*

    },

    specPattern: "cypress/e2e/\*\*/\*.{js,jsx,ts,tsx,feature}",

    excludeSpecPattern: "cypress/e2e/1-getting-started",

    excludeSpecPattern: "cypress/e2e/2-advanced-examples",

    chromeWebSecurity: false,

    experimentalSessionAndOrigin: true,

    defaultCommandTimeout: 10000,

    pageLoadTimeout: 120000,

*//baseUrl: "https://automationteststore.com/",*

    baseUrl: "https://www.webdriveruniversity.com/",

    env: {

      first\_name: "Sarah", *//Making a Global Variable and used in contact-us (WebDriver Uni).*

      webDriverUni\_WebPage: "https://www.webdriveruniversity.com/",

      AutomationStore\_WebPage: "https://automationteststore.com/", *//See automation Test Store > Variables and Cy-Commands >     cy.visit(Cypress.env("AutomationStore\_WebPage") + "index.php?rt=content/contact");*

    },

  },

});

1. To Open Cypress :

npx cypress open, ./node\_modules/.bin/cypress run --headed, ./node\_modules/.bin/cypress run --browser chrome  
To run specific test: ./node\_modules/.bin/cypress run -spec cypress (paste relative path and change all backslashes tgeo forward slash)

1. To Open Cypress Headless: ./node\_modules/.bin/cypress open (write in Terminal in bash)

./node\_modules/.bin/cypress run (<https://docs.cypress.io/guides/guides/command-line>)

1. Cypress Best Practices: <https://docs.cypress.io/guides/references/best-practices>
2. describe and it are mocha commands
3. Cypress in Asynchronous, means non cypress commands will run first and cypress commands will run later.

# 6. JS commands:

1. If Block

if (condition) {

}

1. If-Else Block

if (condition) {

} else {

}

# 7. Cypress Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Command Name | Command Use | Cypress Command | Cypress Documentation Link |
| Cypress General Commands |  |  | <https://docs.cypress.io/api/table-of-contents/> |
| Visit and Click | To visit and click | Cy.visit (“website”) , cy.click (), cy.click ({force:true}) | <https://docs.cypress.io/api/commands/click#Options> |
| Type | To type something | Cy.get (“id”).type (“Name”) | <https://docs.cypress.io/api/commands/type> |
| Chai Assertions (Length, Class, Value, Visibility, Existence, State, Disabled) | To check length, class (enabled/disabled), display value, visibility of class, existence of class, checking state of buttons, enable/disable class | cy.get('li.selected').should(**'have.length'**, 3)  cy.get('form').find('input').should('**not.have.class', 'disabled'**)  cy.get('textarea').should(**'have.value'**, 'foo bar baz')  cy.get('[data-testid="form submit"]').should(**'be.visible'**)  cy.get('[data-testid="loading"]').should(**'not.exist'**)  cy.get(':radio').should(**'be.checked'**)  cy.get('[data-testid="example-input"]')  .should('**be.enabled')**  //or negate the "disabled" assertion  .and('not.be.disabled')  .should(“have.attr”, “name”, “email”), should (“have.text”, “text”) | <https://docs.cypress.io/guides/references/assertions#Adding-New-Assertions>  ,  <https://docs.cypress.io/guides/references/assertions#Chai-jQuery> |
| Contains |  | cy.get('.nav').contains('About'), cy.contains('Hello') , cy.get('form') .contains().submit() | <https://docs.cypress.io/api/commands/contains#Syntax> |
| Document (Page Verification) | Perform assertions in active page | cy.document().should(“have.property”, “charset”).and (“eq”, “UTF-8”), cy.document().its('contentType').should('eq', 'text/html') | <https://docs.cypress.io/api/commands/document#Syntax> |
| Title (Page Verification) | Get and verify title of website and test | Cy.title().should (“include”, “Titleofpage”) | <https://docs.cypress.io/api/commands/title#Syntax> |
| URL (Page Verification) | Get and verify page URL is loaded correctly | Cy.url().should(“include”, “URL name”)  cy.url({ decode: true }).should('contain', '사랑') | <https://docs.cypress.io/api/commands/url#Syntax> |
| Chain Commands | Chain type, get, find, click, contains, check, uncheck, double click, right click | cy.get('textarea.post-body').type('This is an excellent post.')  cy.get(“classname”).contains(“ text)  cy.get (‘class’).find(‘class’).eq(0).click() | <https://docs.cypress.io/guides/core-concepts/introduction-to-cypress#Chains-of-Commands> |
| Log |  | Cy.log() |  |
| Promises (then command) |  | cy.get("a[href$='contact']").click()  .then(function(linkText){  cy.log("Clicked on link using text: " + linkText.text()) })  cy.get (“class”).then(($headerText)=> {  const headerText = $headerText.text()  cy.log (headerText)  expect (headerText).is.a.text()  }) | <https://docs.cypress.io/guides/core-concepts/introduction-to-cypress#Commands-Are-Promises> ,  <https://docs.cypress.io/api/utilities/promise> |
| Each | used to iterate through array like structure, with length property | cy.get(".fixed\_wrapper .prdocutname").each(($el, index, $list) => { cy.log("Index: " + index + " : " + $el.text())  });  $el wrapped element, index (number), list | <https://docs.cypress.io/api/commands/each> |
| Aliases | Alias act as variable, that can be use inside a test to get properties |  | <https://docs.cypress.io/guides/core-concepts/variables-and-aliases#Aliases> |
| origin | Access to different URLS | cy.origin('webdriveruniversity.com', () => {  cy.visit("/");  }) | <https://docs.cypress.io/api/commands/origin#Syntax> |
| Back, forward, reload b |  | cy.go('back')  cy.reload() | <https://docs.cypress.io/api/commands/reload> |
| Handling Multiple Browser Scenarios (Page reificationon) |  | cy.get("#contact-us").invoke("removeAttr", "target"). | <https://docs.cypress.io/guides/references/trade-offs#Multiple-tabs> |
| Alerts |  | cy.on('window:confirm', (str) => {  return true;  }) | <https://docs.cypress.io/api/cypress-api/catalog-of-events#Event-Types> , <https://docs.cypress.io/api/commands/stub#Syntax> |
| Stubs |  | const stub = cy.stub()  cy.on('window:confirm', stub)  cy.get('#button4').click().then(() => {  expect(stub.getCall(0)).to.be.calledWith('Press a button!')  }).then(() => {  return true;  }).then(() => {  cy.get('#confirm-alert-text').contains('You pressed OK!')  }) | <https://docs.cypress.io/api/commands/stub#Syntax> |
| iFrames |  | cy.get('#frame').then($iframe => {  const body = $iframe.contents().find('body')  cy.wrap(body).as('iframe')  }) | <https://docs.cypress.io/guides/guides/web-security#Cross-origin-iframes> |
| Checkboxes, Radio Buttons, |  | cy.get('@option3').uncheck().should('not.be.checked')  cy.get("input[type='checkbox']").check(["option-1", "option-2", "option-3", "option-4"]).should('be.checked') | <https://docs.cypress.io/api/commands/check#Syntax> |
| Dropdown lists |  | cy.get('#dropdowm-menu-2').select('testng').should('have.value', 'testng') |  |
| Mouse Actions (Scroll into view, Drag & Drop,Double click, Click & Hold Assertions |  | cy.get('#actions').scrollIntoView().invoke('removeAttr', 'target').click({force:true})  cy.get('#draggable').trigger('mousedown', {which: 1});  cy.get('#droppable').trigger('mousemove').trigger('mouseup', {force:true})  cy.get('#double-click').dblclick(); |  |
| Children Command | Locates children elements below | cy.get('[class="breadcrumb traversal-breadcrumb"]')  .children(".active")  .should("contain", "Contact Us"); |  |
| Closest |  |  |  |
| Select File |  | cy.get("#myFile").selectFile("cypress/fixtures/laptop.png"); | <https://docs.cypress.io/api/commands/selectfile#Syntax> |
| Hooks |  |  | <https://docs.cypress.io/guides/references/bundled-libraries#Mocha> |
| Fixture | Load data from a file giving its path from fixture folder | cy.fixture('users').as('usersJson') // load data from users.json  cy.fixture('logo.png').then((logo) => {  // load data from logo.png  })  To load data, use before hook:  before(function () {  cy.fixture("example.json").then(function (data) {  //this.data = data;  globalThis.data = data;  });  });  To use: data.last\_name, data.email, | <https://docs.cypress.io/api/commands/fixture#Syntax> |
| Commands /Custom Commands | Make changes in commands.js file | Cypress.Commands.add("selectProduct", (productName) => {  cy.get(".fixed\_wrapper .prdocutname").each(($el, index, $list) => {  if ($el.text().includes(productName)) {  cy.wrap($el).click(); } }); });  save in : support / commands.js (Create file)  To call: cy.selectProduct("Seaweed Conditioner"); | <https://docs.cypress.io/api/cypress-api/custom-commands#Syntax> |
| Changing settings |  |  | <https://docs.cypress.io/guides/references/configuration#Timeouts> |

# Commands/Custom Commands in commands.js file

1. Create a custom command in commands.js file.
2. Write Cypress.Commands.add (“name”, (callback function) => {

})

Example1 (Adding 1 Call back function):

*// Select Elements on WebPage and Click Shopping Cart Icon*

Cypress.Commands.add("AddProductToBasket", (*productName*) => {

  cy.get(".fixed\_wrapper .prdocutname").each((*$el*, *index*, *$list*) => {

    if ($el.text() === productName) {

      cy.log($el.text());

      cy.get('[class="productcart"]').eq(index).click();

    }

  });

});

Example2 (Adding multiple callback functions):

Cypress.Commands.add(

  "webdriverUni\_ContactForm\_Submission",

  (*firstName*, *lastName*, *email*, *comment*, *$selector*, *textToLocate*) => {

    cy.get('[name="first\_name"]').type(firstName);

    cy.get('[name="last\_name"]').type(lastName);

    cy.get('[name="email"]').type(email);

    cy.get("textarea.feedback-input").type(comment);

    cy.get('[type="submit"]').click();

    cy.get($selector).contains(textToLocate);

  }

);

1. Use it in your test script by writing (example2: cy.get (“webdriverUni\_ContactForm\_Submission”).

# Cypress Commands and Chaining:

*//Uses cypress commands and chaining*

        cy.contains('#ContactUsFrm', 'Contact Us Form').find('#field\_11').should('contain', 'First name')

# JQuery approach

*//JQuery Approach*

        cy.contains('#ContactUsFrm', 'Contact Us Form').then(*text* => {

*const* firstNameText = text.find('#field\_11').text()

            expect(firstNameText).to.contain('First name')

# Embedded Commands

*//Embedded commands (Closure)*

            cy.get('#field\_11').then(*fnText* => {

                cy.log(fnText.text())

                cy.log(fnText)

            })

        })

# Making Local and Global Variables:

1. In cypress.config.js file, add base URL

baseUrl: "https://www.webdriveruniversity.com/",

1. If there are two or more websites, add env variable.

env : {

first\_name:

website:

}

1. To use the base Url: write in test case: cy.visit (‘/) in before each block

  cy.visit("/");

1. To use the Env: write in test case: cy.name inside env as shown below:

    cy.webpage\_WebDriverUniversity();

Cypress.config.js file:

env: {

      first\_name: "Sarah", *//Making a Global Variable and used in contact-us (WebDriver Uni).*

      webDriverUni\_WebPage: "https://www.webdriveruniversity.com/",

      AutomationStore\_WebPage: "https://automationteststore.com/", *//See automation Test Store > Variables and Cy-Commands >     cy.visit(Cypress.env("AutomationStore\_WebPage") + "index.php?rt=content/contact");*

    },

# Before Function:

  before(*function* () {

    cy.fixture("example").then(*function* (*data*) {

      globalThis.data = data;

    });

# Page object Oriented Model:

1. First Create a folder in Support Folder and name it
2. Create .js file with the name. For example, HomePage\_pageObject.js
3. Enter the values like this:
4. Write: class Name {} and export default Name;

*class* HomePage\_PO {}

export default HomePage\_PO;

1. Write visit website code inside the {}

*class* HomePage\_PO {

visitHomepage() {

cy.visit(Cypress.env("webdriveruni\_homepage"));

}

clickOn\_ContactUs\_Button() {

cy.get("#contact-us").invoke("removeAttr", "target").click({ force: true });

}

}

export default HomePage\_PO;

1. Go back to test file where we need to write code (To import page object): before /// <reference types = “Cypress”/>, type: import Name from "../../support/FolderName

import HomePage\_PO from "../../support/pageObjects/WebDriver-Uni/HomePage\_PageObject";

1. In beforeEach function, create a const with name easy to read and store a value with the same name that is used in step 4.
2. Perform the cypress commands on the constant stored.

beforeEach(*function* () {

*//cy.visit("http://www.webdriveruniversity.com");*

*//cy.get("#contact-us").invoke("removeAttr", "target").click({ force: true });*

*const* homePage\_PO = new HomePage\_PO();

homePage\_PO.visitHomepage();

homePage\_PO.clickOn\_ContactUs\_Button();

});

1. GIT NOTES:  
   <https://docs.github.com/en/get-started/quickstart/set-up-git>   
   All notes are as follows
2. [Download and install the latest version of Git](https://git-scm.com/downloads).
3. Set a Git username: git config --global user.name "Mona Lisa"
4. Set your commit email address in Git.
5. Git Configuration Commands :
   1. What's the current directory (present working directory): pwd
   2. Git Config (Global/User-level) Syntax: git config --global setting value
   3. Configure User and Email: General Syntax: git config --global user.name "Your Name", git config --global user.email [you@someplace.com](mailto:you@someplace.com)
   4. Example using course author's information: git config --global user.name "Jason Taylor", git config --global user.email [jason@jasongtaylor.com](mailto:jason@jasongtaylor.com)
   5. Listing All Global Configuration Settings: git config --global –list
   6. Seeing Git's User-based Config file: cat ~/.gitconfig
6. Git Starting Commands:
   1. Fresh Start: pwd , cd projects/, git init git-demo
   2. - Start with Existing Project: pwd , cd projects/, cd website/, ls , git init
7. Command Reference:
   1. Present Workding Directory: pwd
   2. Change Directory: cd folder-name
   3. Git initialization: git init [project-name]
8. project-name parameter is optional. If not supplied, Git will initialize the current directory.
9. Command Listing: pwd, ls, git status, git add . , git status, git commit -m "Initial commit", clear, git status
10. Command Listing - Working Locally, Part One:
    1. Unstage File : git reset HEAD file-name
    2. Adding All Changed Files: git add .
    3. Express Commit for Tracked files: git commit -am "Awesome commit message"
    4. Backout Working Directory Changes: git checkout -- file-name
    5. History: git log, git help log, git log --oneline --graph --decorate –color
    6. Removing file: git rm FileName
    7. Moving Files: mkdir filename
11. Git Remote:
    1. See .ssh folder, if not, then make ssh key folder
    2. Generating an SSH Key: ssh-keygen -t rsa -C "your.name@your-company.com"
    3. Verify SSH authentication: ssh -T [git@github.com](mailto:git@github.com)
    4. Creating a remote repository reference : git remote add remote-name remote-repository-location
    5. git remote -v (The git remote command lists the names of all the remote repositories and the -v parameter (verbose) will display the full URL of the remote repository for each remote name listed)
    6. Send Changes to Remote: git push -u remote-name branch-name , git push remote-name branch-name (The git push sends all your local changes (commits) on branch branch-name to the remote named remote-name. The -u parameter is needed the first time you push a branch to the remote.)
    7. Receive Changes from Remote: git pull remote-name branch-name (The git pull receives all your remote changes (commits) from the remote named remote-name and on branch branch-name.)
12. Create a New Branch: git branch : Checks in which branch you are present currently, git branch -c myNewBranch, to move: git checkout myNewBranch, Git push origin head: pushes the branch into main