# Cypress commands

01 February 2022 18:14

| TASK | COMMAND | Extention | REMARK |
|---|---|---|---|
| Node version check | node -v | | |
| To generate package.json | npm init | | |
| Cypress installation 1st time | npm install cypress --save-dev | | it will install cypress and save details in package.json |
| Cypress installation alter | npm install cypress | | |
| open Cypress from your projec | .\node_modules\.bin\cypress open | | |
| .\node_modules\.bin\ | | | |
| Run cypress test from the terminal - headless | .\node_module.\.run.\cypress run | | It will run all test file from example folder heedlessly |
| Open Windows Shell Terminal | Set-ExecutionPolicy RemoteSigne | Yes | To run cypress test from the command prompt or VS code terminal |
| Run cypress test from the terminal - visible | ./node_modules/.bin/cypress run --headed | --headed | UI will be visible for user. |
| Run cypress specific test from the terminal | ./node_modules/.bin/cypress run --headed --spec "cypress/integration/examples/MyFirstTest.js | --spec | If user want to run specific test case from the example folder then user need to give file path |
| Run cypress test from the terminal on specific browser | ./node_modules/.bin/cypress run --spec "cypress/integration/examples/MyFirstTest.js" --browser chrome | --browser name | As cypress script runs on electron browser by default but user want to run it on any other browser like chrome, edge. |
| Run cypress test from the terminal and record | ./node_modules/.bin/cypress run --record --spec "cypress/integration/my-spec.js | -- record | run tests from a single spec file and record the results on the Dashboard |
| Run cypress test from the terminal | cypress run --headed --no-exit | --no-exit | To prevent the Cypress Test Runner from exiting after running tests in a spec file |
| Run cypress test from the terminal in parallel | cypress run --record --parallel | --parallel | Run recorded specs in parallel across multiple machines. |
| Verify Cypress | .\node_modules\.bin\cypress verify | | Verify that Cypress is installed correctly and is executable. |
| Verify Cypress version | .\node_modules\.bin\cypress version | | Prints the installed Cypress binary version |
| Clear Cypress Cache | .\node_modules\.bin\cypress clear cache | | Clear the contents of the Cypress cache. |

Reference sites:

https://example.cypress.io/
https://www.nopcommerce.com/en

# Cypress methods

01 February 2022 18:26

| Method Name | Method Signature | Syntax | Remark |
|---|---|---|---|
| Navigate/ launch url | `cy.visit` | `cy.visit('https://www.nopcommerce.com')` | |
| To get title | `cy.title()` | `cy.title().should('eq', 'Fr` | Capture title and comapre with text with equal to. |
| | cy.get(selector) | | |

- https://github.com/cypress-io/cypress-documentation/issues/3075

## Current behavior:

Intellisense/ AutoComplete Not Working in VS Code.

Desired behavior:

Intellisense/ AutoComplete Should Work in VS Code.

- So I dont have to use /// <reference types="Cypress" /> in every cypress file.

Hacky Solution:

1. So I took backup(just in case) and deleted 'Code' folder located in "C:\Users \(UserName)\AppData\Roaming\Code" in Windows.

2. Follow Official Docs
   Or Follow what I did:
   I created 'tsconfig.json'(although I use .js file) under 'cypress' folder and pasted this:

```
{
    "compilerOptions": {
                "allowJs": true,
                "types": ["cypress"]
    },
    "include": ["**/*.*"]
}
```

So I dont have to use /// <reference types="Cypress" /> in every cypress file.

1. Reloaded VS Code and autocomplete started working!
   [I also deleted extension folder and reverted backup - No change - So not necessary to delete extension folders]

I use windows 7 and dont know if it should be done in mac etc. Take backup and dont blame me if you dont know what you are doing. Or just install Webstorm(Jetbrain) and test if autocomplete works and if it works then proceed to the hacky solution I mentioned.

# Cypress 0 - Training from Edgewords

18 January 2022 13:49

## Cypress Training from Edgewords

**Cypress is a fantastic new web testing tool. It is becoming incredibly popular, and not without reason, it provides benefits of:**

- **Fast deployment**, with everything you need – no having to install many other dependencies (includes Mocha, Chai, JQuery, Electron Browser)
- **Very fast test execution** – Cypress executes test within the web browser, unlike other tools that go through a middle layer browser driver
- **Easy to use**, with tools to run tests (Test Runner), to record actions, and also to locate web elements
- Easy **integration into Continuous Integration** Tools, it even includes a GitHub action to execute tests in the cloud through GitHub!
- **Multi-platform, Multi-browser**
- As well as GUI testing, it includes **RESTful API Testing**, and **API Mocking**.
- Cypress provides a hosted **Dashboard** in the cloud to share Test Execution Results!

Cypress is written in JavaScript and runs on NodeJS. It is opensource and free! The Cypress team also provide an amazing dash board service so all of your test run results can be viewed securely in the cloud by anyone that needs access to them.

## Pre-requisites

None, we do have an optional JavaScript primer chapter if you are new to JavaScript. We can also email you the setup instructions before the course so you can setup Cypress in readiness.

**Cypress Setup**

- json & npm
- Installing Cypress
- Cypress VS Code Extensions
- The Test Runner
- Debugging
- Cypress Folder Structure (Scaffolding)

- Cypress is a next generation front end testing tool built for the modern web. We address the key pain points developers and QA engineers face when testing modern applications.
- Cypress is most often compared to Selenium; however Cypress is both fundamentally and architecturally different. Cypress is not constrained by the same restrictions as Selenium.
- This enables you to write faster, easier and more reliable tests.
- Cypress can test anything that runs in a browser.
- Who uses Cypress? - Our users are typically developers or QA engineers building web applications using modern JavaScript frameworks.
- Cypress enables you to write all types of tests:
  a. End-to-end tests
  b. Integration tests
  c. Unit tests
- Cypress consists of a free, open source, locally installed Test Runner and a Dashboard Service for recording your tests.
- **First**: Cypress helps you set up and start writing tests every day while you build your application locally. TDD at its best!
- **Later**: After building up a suite of tests and integrating Cypress with your CI Provider, our Dashboard Service can record your test runs. You'll never have to wonder: Why did this fail?

**Features**

Cypress comes fully baked, batteries included. Here is a list of things it can do that no other testing framework can:

- Time Travel: Cypress takes snapshots as your tests run. Hover over commands in the Command Log to see exactly what happened at each step.
- Debuggability: Stop guessing why your tests are failing. Debug directly from familiar tools like Developer Tools. Our readable errors and stack traces make debugging lightning fast.
- Automatic Waiting: Never add waits or sleeps to your tests. Cypress automatically waits for commands and assertions before moving on. No more async hell.
- Spies, Stubs, and Clocks: Verify and control the behavior of functions, server responses, or timers. The same functionality you love from unit testing is right at your fingertips.
- Network Traffic Control: Easily control, stub, and test edge cases without involving your server. You can stub network traffic however you like.
- Consistent Results: Our architecture doesn't use Selenium or WebDriver. Say hello to fast, consistent and reliable tests that are flake-free.
- Screenshots and Videos: View screenshots taken automatically on failure, or videos of your entire test suite when run from the CLI.
- Cross browser Testing: Run tests within Firefox and Chrome-family browsers (including Edge and Electron) locally and optimally in a Continuous Integration pipeline.

### Setting up tests

There are no servers, drivers, or any other dependencies to install or configure. You can write your first passing test in 60 seconds.

### Writing tests

Tests written in Cypress are meant to be easy to read and understand. Our API comes fully baked, on top of tools you are familiar with already.

### Running tests

Cypress runs as fast as your browser can render content. You can watch tests run in real time as you develop your applications. TDD FTW!

### Debugging tests

Readable error messages help you to debug quickly. You also have access to all the developer tools you know and love.

### Test types

Cypress can be used to write several different types of tests. This can provide even more confidence that your application under test is working as intended.

### End-to-end

Cypress was originally designed to run end-to-end (E2E) tests on anything that runs in a browser. A

Cyprus HTTP call

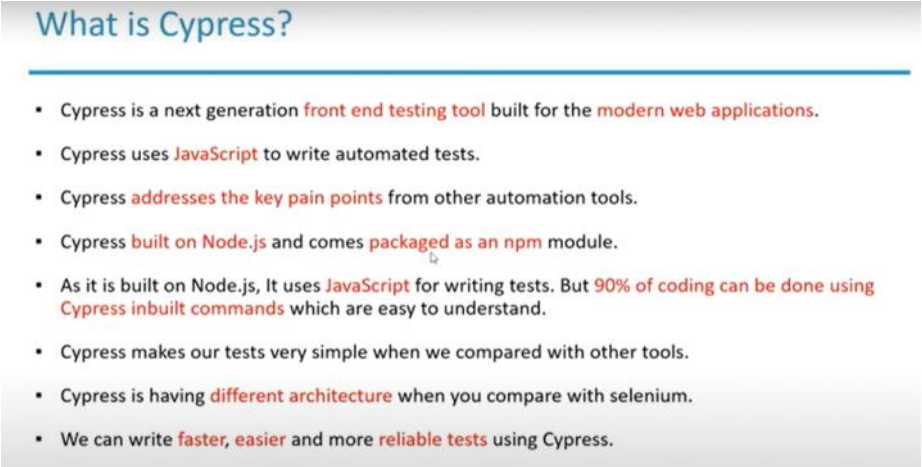
```
it('adds a todo', () => {
 cy.request({
  url: '/todos',
  method: 'POST',
  body: {
   title: 'Write REST API',
  },
 })
  .its('body')
  .should('deep.contain', {
   title: 'Write REST API',
   completed: false,
  })
})
```

typical E2E test visits the application in a browser and performs actions via the UI just like a real user would.

# Cypress 1 - Introduction

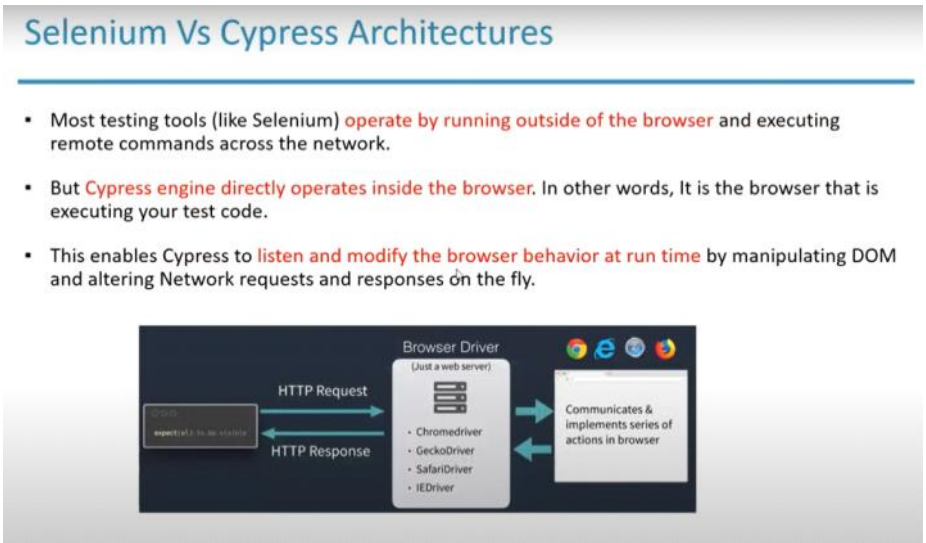
29 January 2022 15:42

**What is Cypress ?**



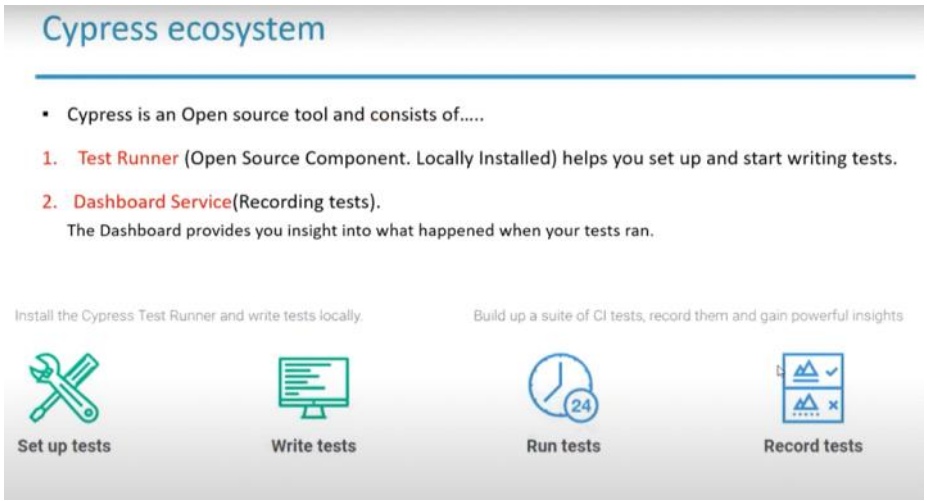

- Cypress is next generation front end automation tool built for the modern web application.
- We can perform web testing using cypress.
- It is especially designed for new generation application like angular.
- Cypress built on NodeJS and comes package as npm module.
- Cypress only used JavaScript to write automated tests.
- It uses JavaScript for writing test cases but 90% of coding can be done using Cypress in-built commands.

**Differences between architecture in Selenium and Cypress?**



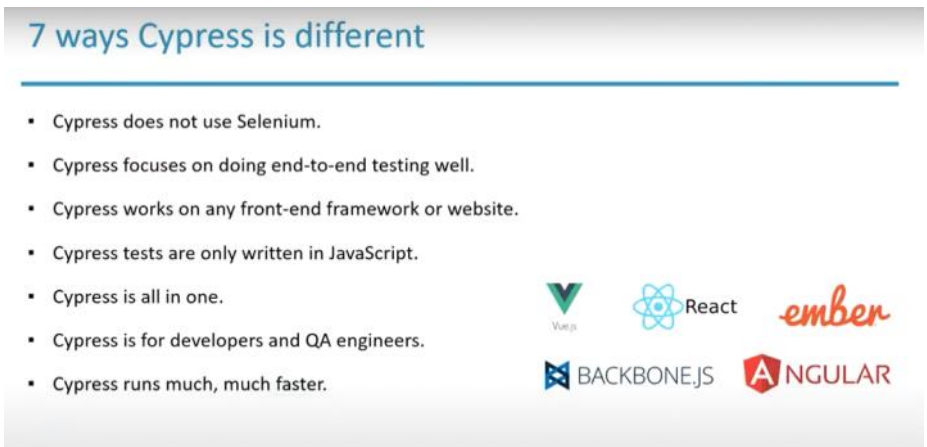

- In a selenium it basically operate by running outside of the browser and executing remote commands across a network.
- Cypress engine directly operates inside the browser. There is no middle layer is required in cypress. It means it is a browser that is executing your code.
- Hence your cypress scripts are very faster than selenium scripts. This is major architecture level difference between selenium and cypress.
- Cypress is not at all related to selenium, it's completely different.
- This feature of cypress particularly enable us to cypress listen and modify the browser behavior at the run time. Run time also we can control certain things in th browser.

**What is the Cypress Ecosystem ?**




- Cypress is an open-source tool which consist of two main components
  a. TestRunner
  b. Dashboard Service.
- Test Runner - we can install this locally and by using this we can set-up and write our tests by using JavaScript.
- Test runner is providing environment to write and maintain our test cases.
- Dashboard Service - dashboard service is basically providing to run our tests and record our tests and have all the information and results of our tests which is maintained in our dashboard.

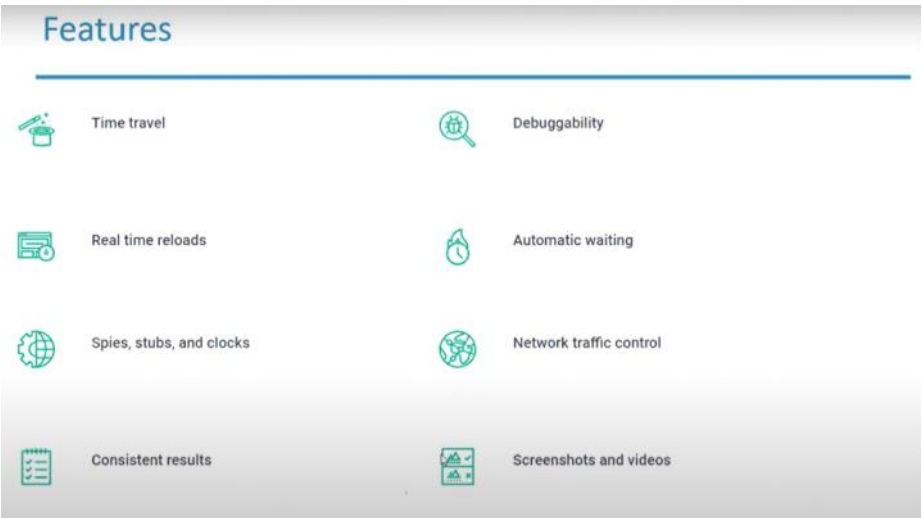
**How Cypress is different ?**




- Cypress does not use selenium. If we go for any other application like appium, protractor so these are the tools are Katalon tool , so these are tool build on top of selenium. Hence selenium is base for them.
- [[[Katalon Studio is a robust automation tool initially released in January 2015 with a Selenium-based engine. Mostly, Katalon is designed to create and reuse automated test scripts for UI without coding. Katalon Studio allows running automated tests of UI elements, including pop-ups, iFrames, and wait-time.]]]
- Cypress is specially designed for handling applications which are developed by different type of JavaScript based frameworks lie VueJS, ReactJS, AngularJS.
- VueJS, ReactJS, AngularJS these are different type of frameworks are available to develop web based applications but selenium does not supports directly these web applications. We would required some add on tools to handle these kind of applications.
- This are new generation technology applications now a days and cypress is especially designed to deal with this applications.
- Cypress works on any front-end framework or website.
- Cypress focus on end-to-end testing.
- Unit testing, Integrated testing, functional testing, API testing, Database testing we can perform using Cypress.
- Cypress is limited only for JavaScript and it is all in one.


- Earlier before cypress came in to the market, there are so many frameworks we have to use to handle the web applications which are implemented by using different type of frameworks.
- We can also control the network request HTTP request we can control.
- Cypress is for developers and QA engineers. Developer can use this for unit testing.
- Before cypress - chai assertions

**What are the Features of Cypress ?**



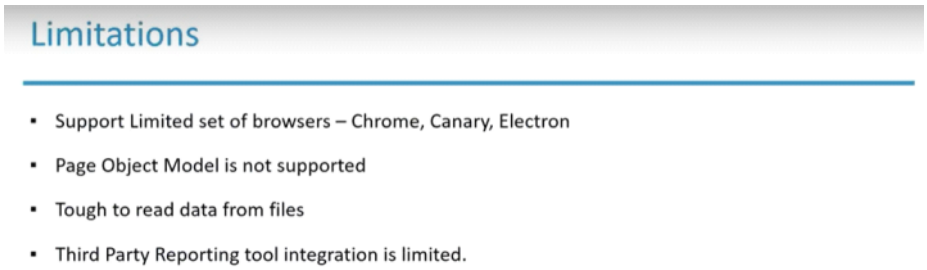

- Official website of Cypress is: https://www.cypress.io

**Features of Cypress:**

1. Time Travel - Cypress takes snapshots as your tests run. Simply hover over commands in the Command Log to see exactly what happened at each step. What tool has done as part of your test, we can just go back and move forward and check each and every step and re-collect and re-play.
2. Real time reloads - Cypress automatically reloads whenever you make changes to your tests. See commands execute in real time in your app. It means while script is running and other side if user modify the test the cypress automatically re-load those changes on the browser.
3. Spies, stubs, and clocks - Verify and control the behavior of functions, server responses, or timers. The same functionality you love from unit testing is right at your fingertips. This features are related to function and client server communication. So we can also verify and control the behavior of functions, server responses, or timers especially used for the API testing.
4. Consistent results - Our architecture doesn't use Selenium or WebDriver. Say hello to fast, consistent and reliable tests that are flake-free.
5. Debuggability - Stop guessing why your tests are failing. Debug directly from familiar tools like Chrome DevTools. Our readable errors and stack traces make debugging lightning fast.
6. Automatic waiting -
    - Never add waits or sleeps to your tests. Cypress automatically waits for commands and assertions before moving on. No more async hell.
    - In selenium we will face a lot of issue in synchronization, (element is not visible, page loading issue etc) in cypress there is no wait at all. Without giving wait our script go smoothly because internally our Cypress itself having some intelligence to wait for the element. As soon as element is available cypress will intercat with an

element. This kind of intelligence is already available in Cypress.
    - Cypress is automatically waits for commands, assertions and elements.
7. Network traffic control - Easily control, stub, and test edge cases without involving your server. You can stub network traffic however you like. Somehow it will also control the network traffic in case of server is down in that case we can also modify the request especially useful in API testing.
8. Screenshots and videos - View screenshots taken automatically on failure, or videos of your entire test suite when run heedlessly.

**What are the limitations in Cypress ?**

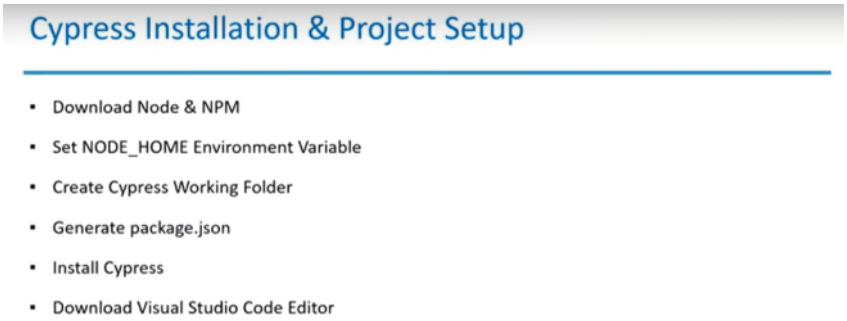



- Currently only supports Chrome browser. [ Cypress team working on supports for IE and Firefox.]
- Other browser also support like Canary, Electron - Electron is another light weight version of chrome.
- Page Object Model [POM] design pattern is not supported in Cypress.
- Tough to read data from the file . Sometimes when you perform any data driven test cases we will prepare data in excel or third party files so here is tough to read that data.
- We have to use in-build report mechanism in cypress so we cannot use/ integrate any third party reporting tool in cypress. We have to use existing reporting which is available in cypress dashboard service.
- Edge is supported in cypress in cypress version `"cypress": "^9.3.1"` >> 01/02/2022.

# Cypress 2 - Environment Setup On Windows

29 January 2022 17:05

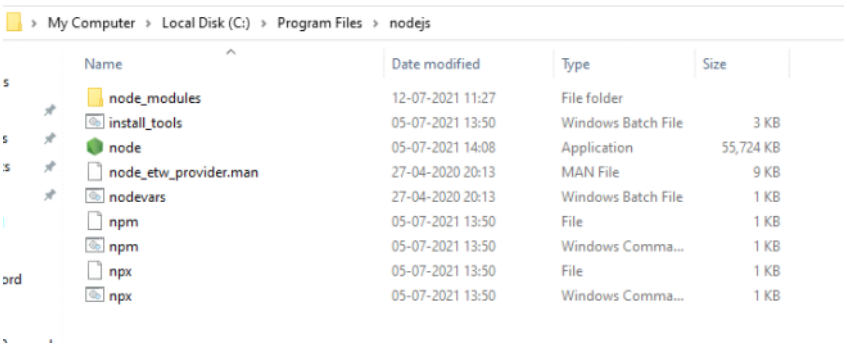
- Download Node and NMP



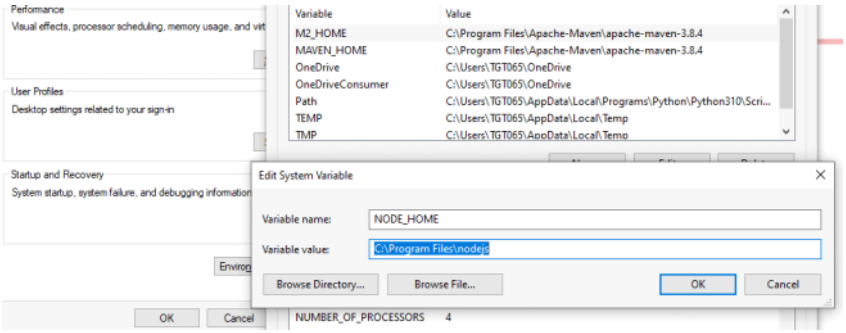

- Node Package Manager - we need to download and install node. npm
- Node will provide an environment where we can download any kind of JavaScript framework
- Navigate to: https://nodejs.org/en/download/
- Download windows installer (.msi) 64 bit


- Node is successfully installed.

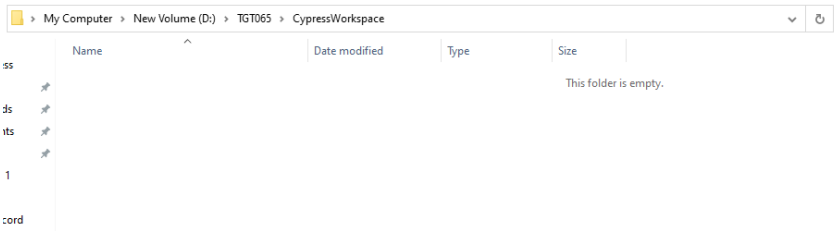
Check node version: node -v


- Set environment variable for nodejs



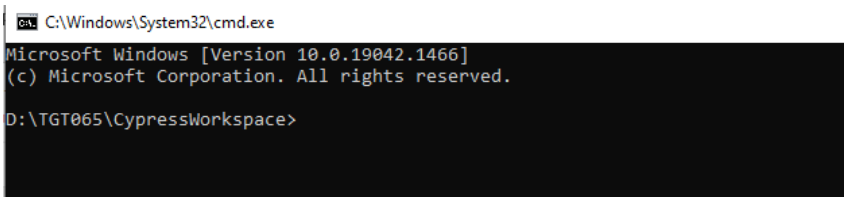

- Create Cypress working folder



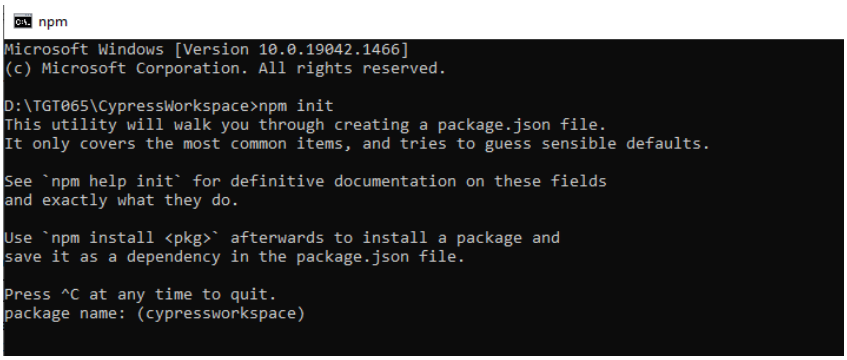

- On node environment we need package.json to download any software.
- We need to specify in package.json file what kind of software we need to install.
- We need to mention cypress dependencies' in package.json file to run npm command.
- Node will read that package.json file and download required software. Anything we can download like protractor, typescript et c.
- package.json is similar file like pom.xml file MAVEN project.

How to generate package.json

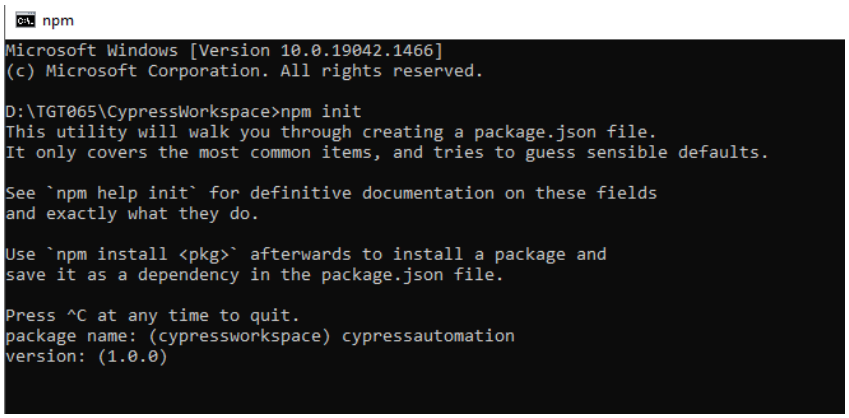
- Navigate to your workspace in command prompt



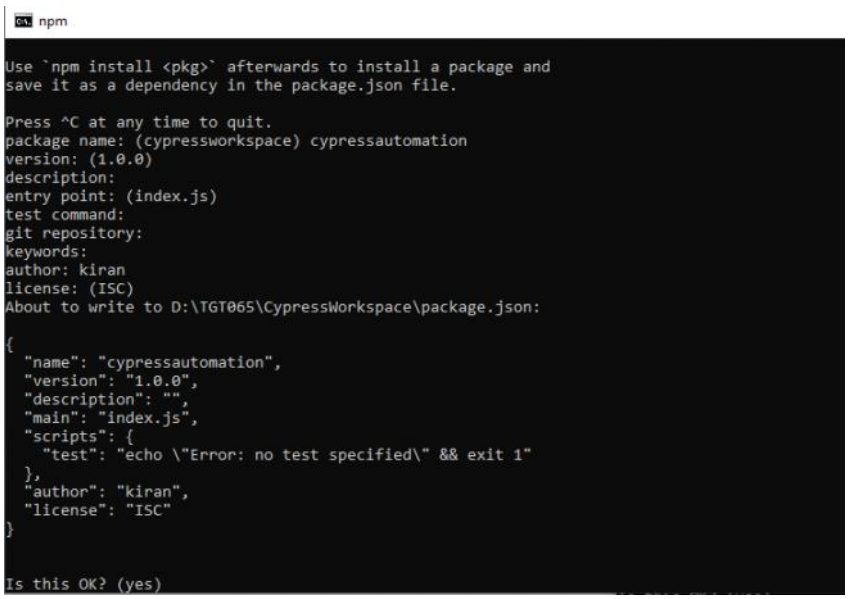

- Type a command: npm init




- Once you run the command [npm init] it will ask for the package name
- Give package name for eg. Cypressautomation




- It will ask for the version, keep it as it is and press enter
- Description
- Entry point
- Test command
- Git repository
- Keyword
- Author - kiran
- License

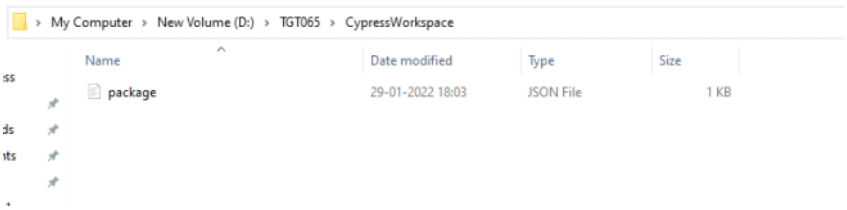



- Here is our json file format has been created

```
Select C:\Windows\System32\cmd.exe
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (cypressworkspace) cypressautomation
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author: kiran
license: (ISC)
About to write to D:\TGT065\CypressWorkspace\package.json:

{
  "name": "cypressautomation",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "kiran",
  "license": "ISC"
}


Is this OK? (yes) yes

D:\TGT065\CypressWorkspace>
```

- Directory - package.json file has been created.

```
D:\TGT065\CypressWorkspace>dir
 Volume in drive D is New Volume
 Volume Serial Number is 82FA-49C7

 Directory of D:\TGT065\CypressWorkspace

29-01-2022  18:03    <DIR>          .
29-01-2022  18:03    <DIR>          ..
29-01-2022  18:03               218 package.json
               1 File(s)            218 bytes
               2 Dir(s)  105,352,634,368 bytes free

D:\TGT065\CypressWorkspace>
```




- Navigate to Cypress installation documentation
- https://docs.cypress.io/guides/getting-started/installing-cypress
  Install Cypress via npm:

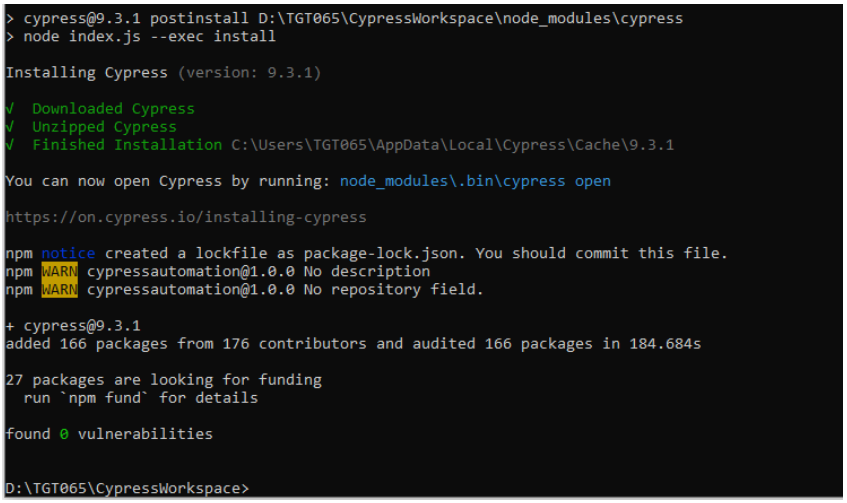
  `cd /your/project/path`

  `npm install cypress --save-dev`

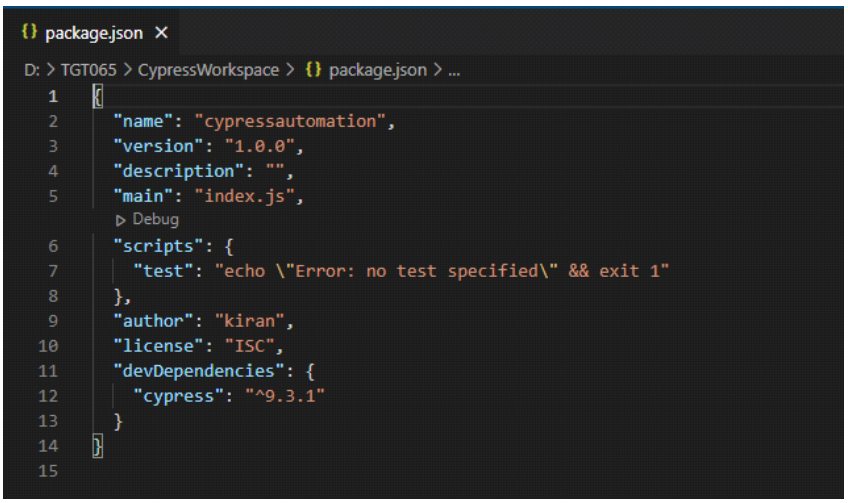
- First time you have to execute full command [`npm install cypress --save-dev`] so it will save details in package.json/
- When you will move your project to another location/pc then you only need to execute command[ `npm install cypress`]

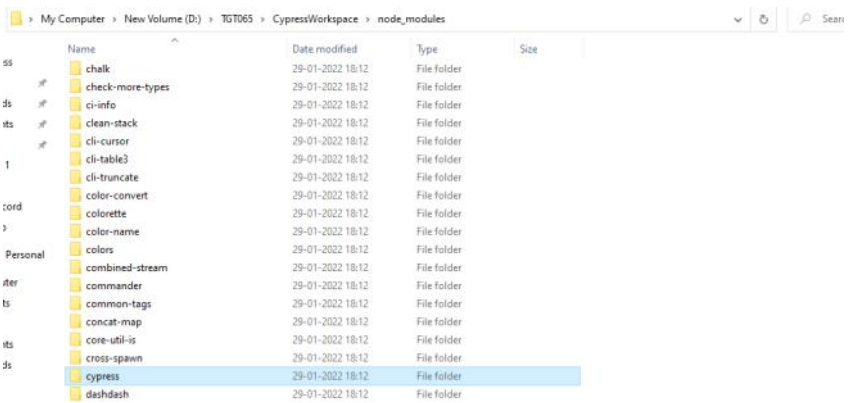
```
D:\TGT065\CypressWorkspace>npm install cypress --save-dev
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the
 API instead.

> cypress@9.3.1 postinstall D:\TGT065\CypressWorkspace\node_modules\cypress
> node index.js --exec install

Installing Cypress (version: 9.3.1)

\  Downloading Cypress      42% 21s
```

- Cypress installation.
- Cypress unzipping.
- Finishing installation.




- Cypress dependencies has been added in package.json file
- Cypress version you can check in package.json file




- Download and install VS Code studio
- Eclipse is best editor for java same like VS Code studio is best editor for the JavaScript.

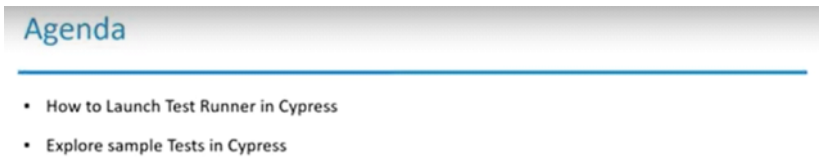
# Cypress Environment Setup On Mac

https://youtu.be/HZ8nhPEKQm8

# Cypress 3 - Test Runner Component

29 January 2022 18:25

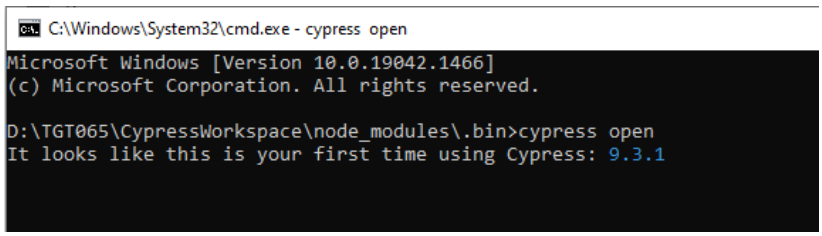
**Test runner component**




- By test runner component we can write and execute our test cases.
- To work with test cases we need test runner component.
- First we need to launch test runner component.
- We need to start cypress within our VS Code studio so for that we need to run a command through terminal.
  1. We can use our windows command prompt terminal.
  2. We can use inbuilt terminal from the VS Code studio.
- Now you can open Cypress from your project root i.e. from inside nodemodule/bin
- Commands are:

| | |
|---|---|
| Ios: | ./node_modules/.bin/cypress open |
| Windows: | .\node_modules\.bin\cypress open |

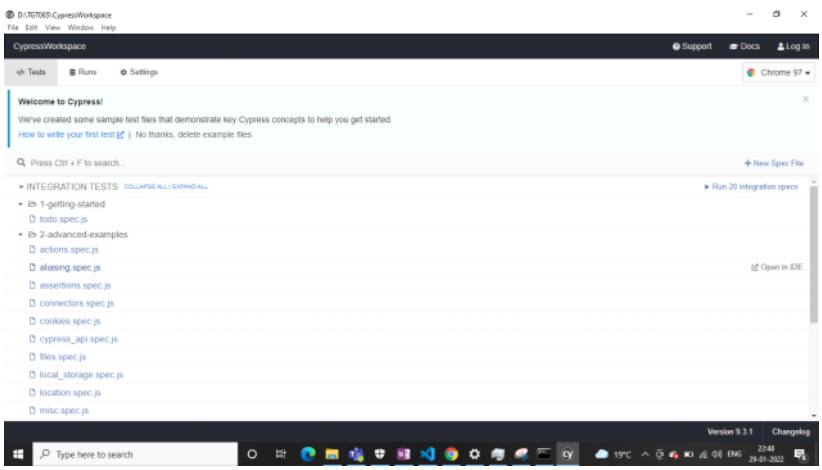
- Navigate to path: D:\TGT065\CypressWorkspace\node_modules\.bin




- First time it will take time open cypress.

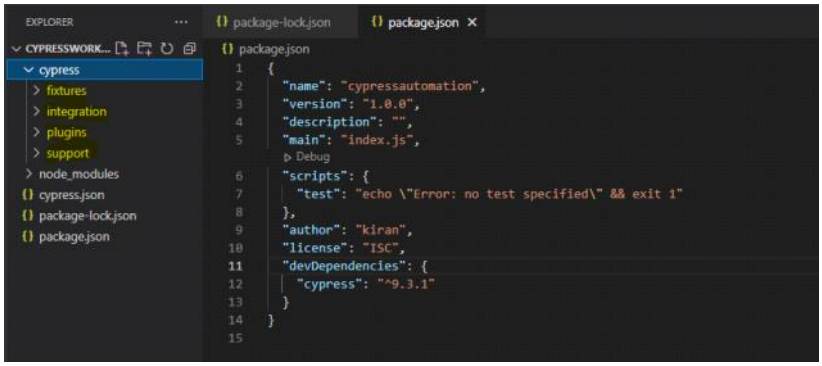
"Navigate to project directory and type the command : **.\node_modules\.bin\cypress open**" it will open the cypress.

Below is our test runner window.

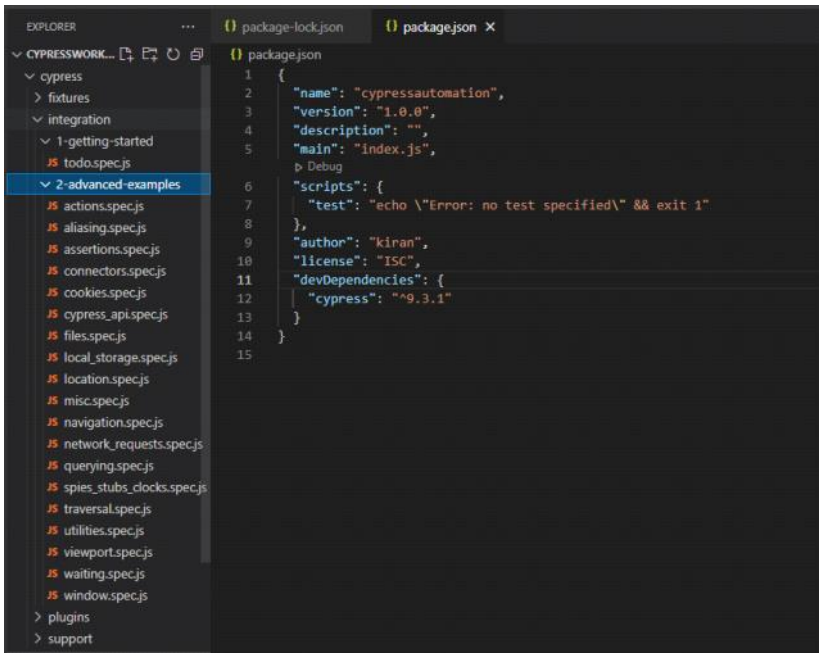



- All the sample test cases provided by Cypress is displayed inside the example.
- Once you open test runner, as soon as you run the command [**.\node_modules\.bin\cypress open]** first time you will get some other addition folders inside your project.

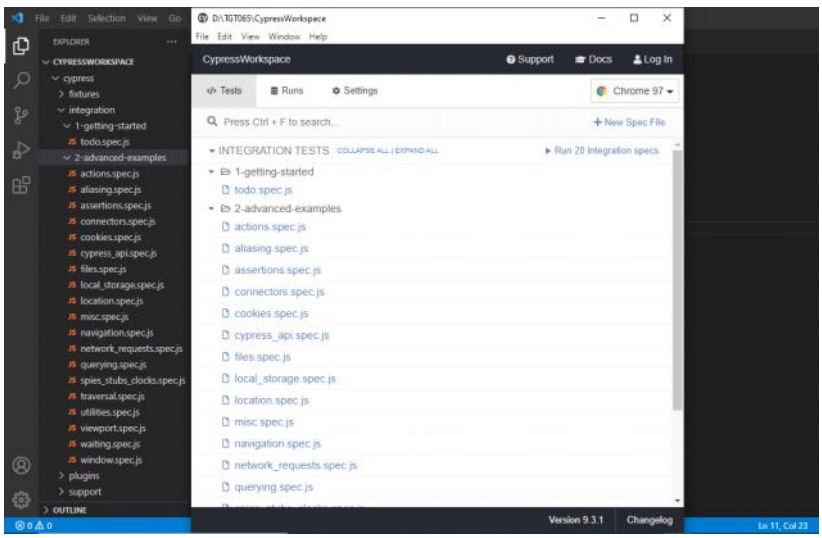
- Once you expand the Cypress you will see some of the important folders, this is basically framework structure which is by default provided by the cypress.




- Inside integration folder cypress has given few demo test cases for the references.
- This will help user to explore how to work with test cases.

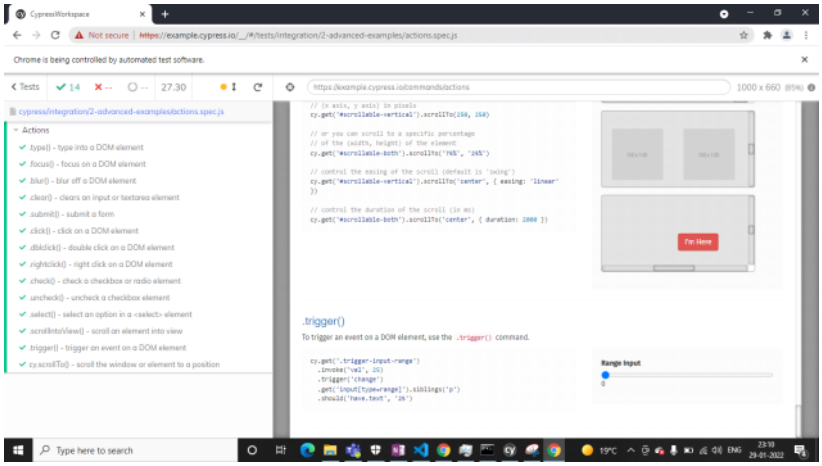



- In the test runner window you can see all the test cases examples is already available in the integration>> example folder
- So once you create test case in VS Code it will automatically pop-up in test runner window.
- But if you want to run your test case that only you have to run it from test runner.
- Create test in VS Code and run it from test runner.
- Test cases will end with .js extension.




**Explore execution of test cases in cypress:**

- To run test cases you just need to click on test case under test runner.



# Cypress 4 - Writing First Test Case

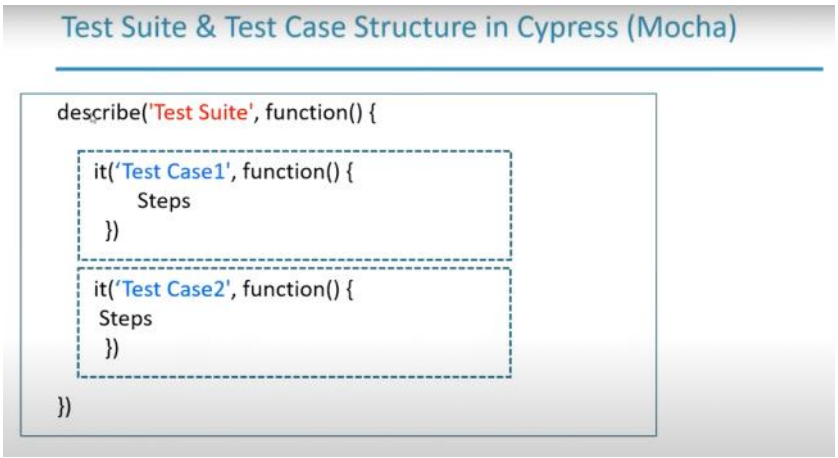
31 January 2022 13:37

**Structure of test cases in Cypress.**

- Inside cypress>>integration vs code we can write out test case file.
- To write our test case we need to follow proper structure and syntax.
- Refer documentation : https://docs.cypress.io/guides/getting-started/writing-your-first-test
- Cypress is web automation toll which ill interact with browser.
- To write test cases and test suite we need to implement some framework.
- In selenium if we want to run/add test suite and test cases and to organize them we need to implement framework like Junit, TestNG same like in Cypress we need to implement framework like Jasmine, Mocha etc.
- Mocha is by-default framework supported by Cypress.
- Below is the syntax coming from mocha framework.

```
describe('My First Test', () => {
  it('Does not do much!', () => {
    expect(true).to.equal(true)
  })
})
```

- Let's understand the syntax and keywords mentioned above code.



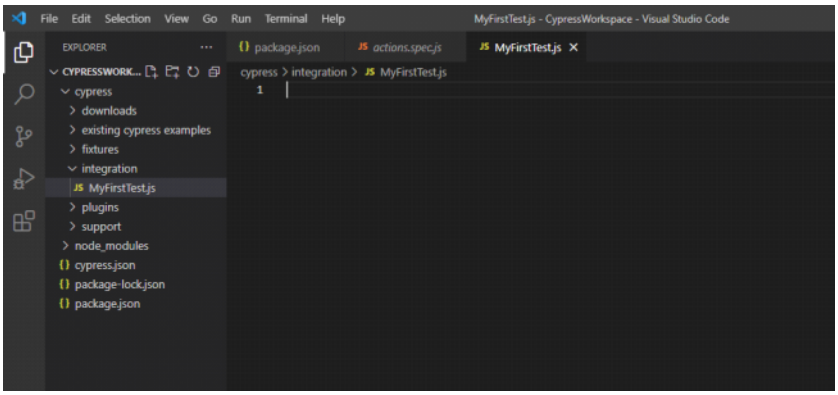

**Describe method:**

- **Describe method** contains two arguments.
- 1st argument will be **string type argument** where we can specify the test suite name.
- 2nd argument is **function()** where we will wrapping our entire test suite inside we created.
- Within this function we can write multiple test cases using with **it keyword**.
- Describe is a keyword will represent a test suite which will have multiple it blocks(test cases)

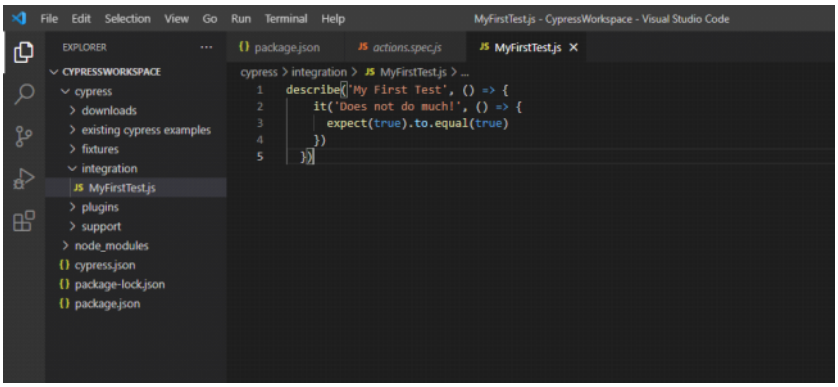
**It keyword**

- It function again it will follow same structure as describe.
- It function will have two arguments.
- 1st argument will be **string type argument** where we can specify the test case name.
- 2nd argument is **function()** where we will wrapping our test cases/ test steps inside we created.
- It is keyword which will represent as a test case.

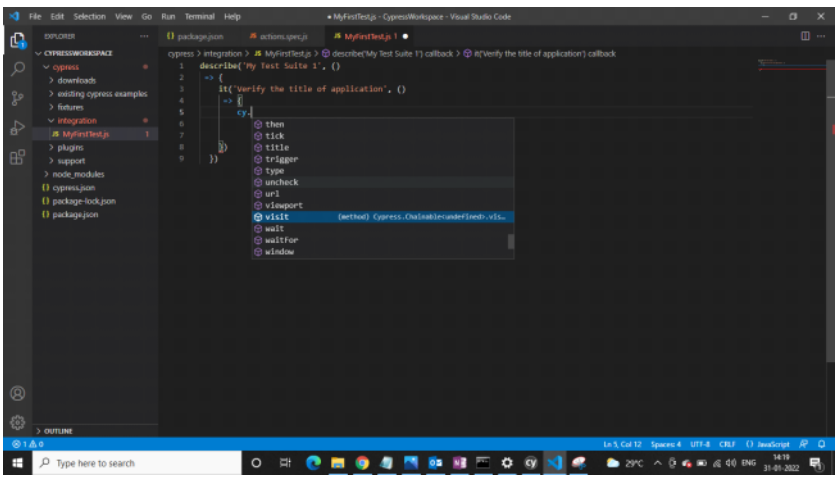
- Cypress is by default using mocha framework, we don't require to install mocha on top of cypress because by default mocha framework automatically installed along with cypress.
- So when you work with testNG in selenium, we will create testNG class inside that we will be creating multiple testNG test methods; similarly describe keyword is representing a parent test suite and function() is wrapping multiple it blocks and every it is considered one test case.
- Moved existing example from integration folder to the cypress inbuilt example folder - for my reference.
- Create js file under integration with name MyFirstTest.js



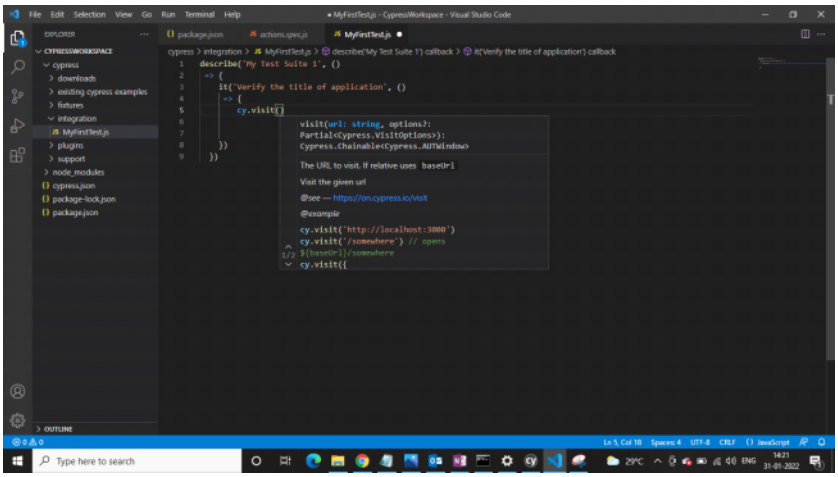
- Copy the code from cypress documentation to js file



- Website: https://www.nopcommerce.com/
- Cy is a parent module, cy is global variable with contains everything ; all the keywords , all the commands we can access only from the cy.cy is kind of predefined object in cypress.
- When user press cy. It will give the all methods and keywords which are available in cypress.



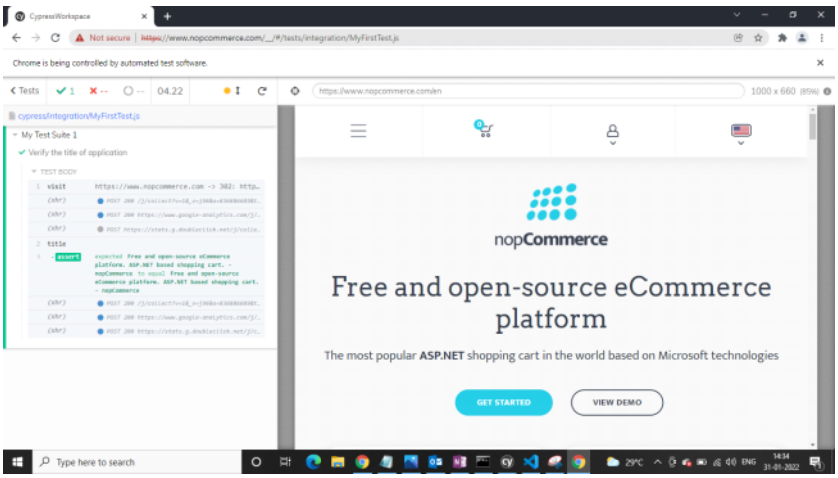
- To navigate an url we need add method called,



```
cy.visit('https://www.nopcommerce.com')
```

- This command will open this url in chrome browser.
- `cy.title().should('eq', 'nopCommerce demo store.')`
- Here we are verify the title of the application with above code.
  - Url: https://www.nopcommerce.com
  - Title: Free and open-source eCommerce platform. ASP.NET based shopping cart. - nopCommerce

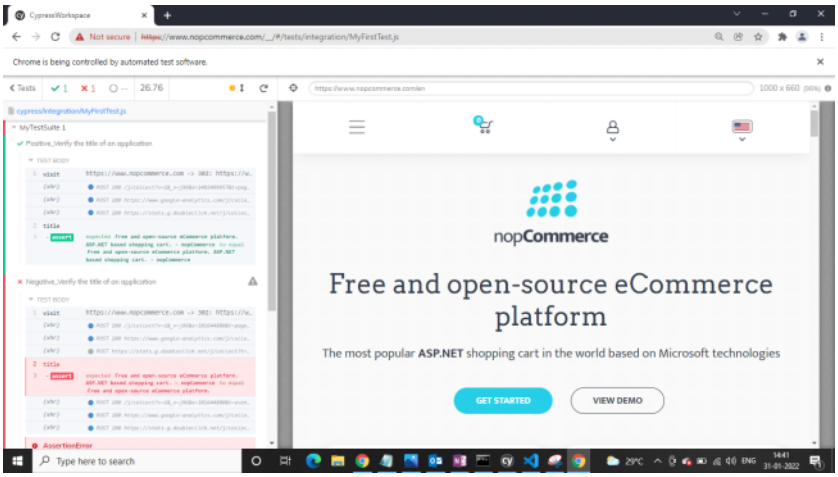
**Results:**




- Add negative test case to check postive expect title will get pass and negative expect title will get fail.
- LOC:

```
describe('MyTestSuite 1', () => {
    it('Positive_Verify the title of an application', () => {
        cy.visit('https://www.nopcommerce.com')
        cy.title().should('eq', 'Free and open-source eCommerce platform. ASP.NET based shopping
cart. - nopCommerce')

    })
    it('Negative_Verify the title of an application', () => {
        cy.visit('https://www.nopcommerce.com')
        cy.title().should('eq', 'Free and open-source eCommerce platform.')

    })
  })
```

**Results:**

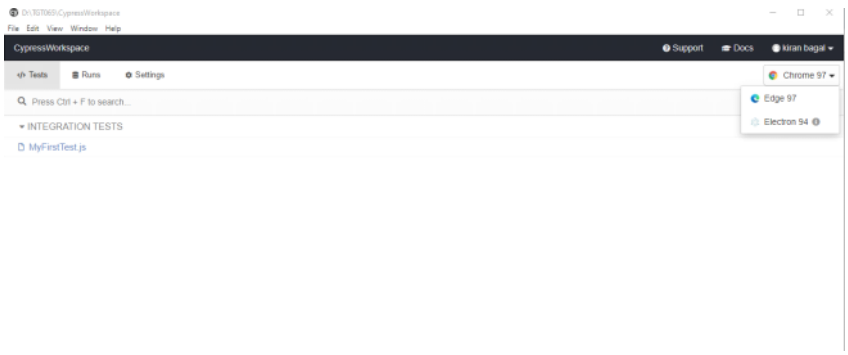

Chrome is being controlled by automated test software.
nopCommerce
Free and open-source eCommerce platform
The most popular ASP.NET shopping cart in the world based on Microsoft technologies
GET STARTED
VIEW DEMO

# Cypress 5 - How To Run Cypress Tests in Test Runner & Terminal

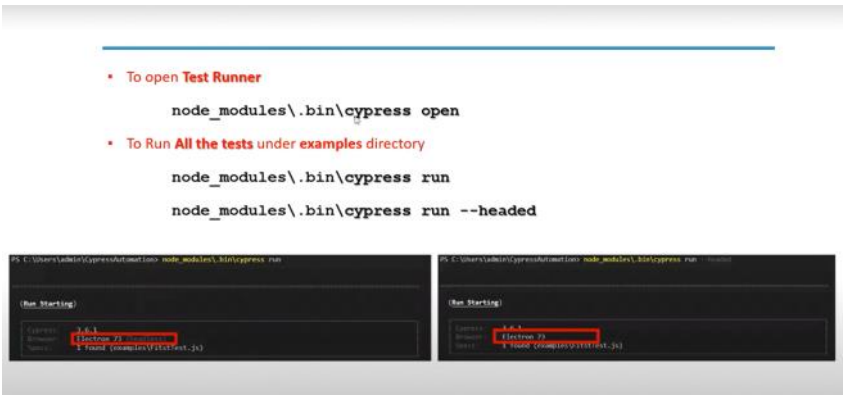
31 January 2022 14:47

**How To Run Cypress Tests in Test Runner & Terminal.**

- Below are the default test browser supported by cypress.




- Cypress `cypress": "^9.3.1"` supports Chrome, Electron and Edge.
- We can run cypress test from ther terminal as well instead of cypress test runner.




**Run cypress test from the cmd/ terminal**

- We can run our cypress test from the terminal with command: .\node_module.\.run.\cypress run
- By default cypress will run the all test from integration>>examples folder.
- By default cypress will use an electron browser.
- This execution will be done in headless browser, UI will not be visible.

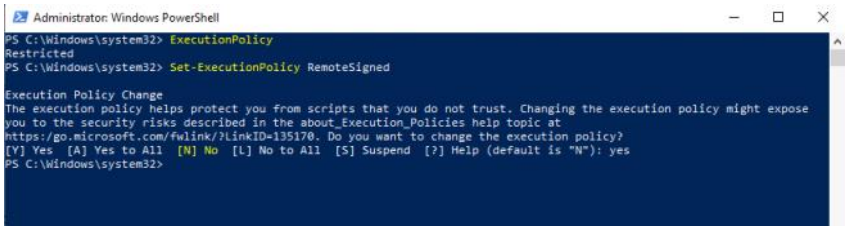
To run cypress test from the command prompt or VS code terminal; follo the following steps.

- Open the windows power shell and run it as administrator.
- Type the command: Set-ExecutionPolicy RemoteSigned
- Press yes

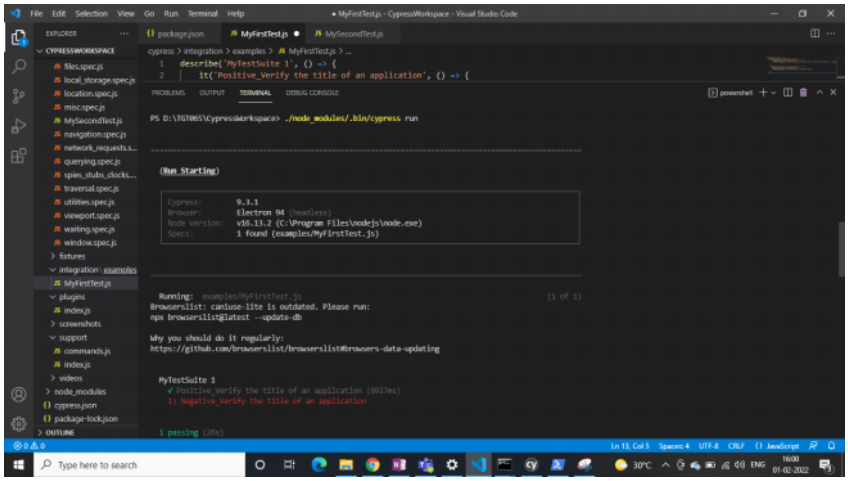
PS C:\Windows\system32> Set-ExecutionPolicy RemoteSigned

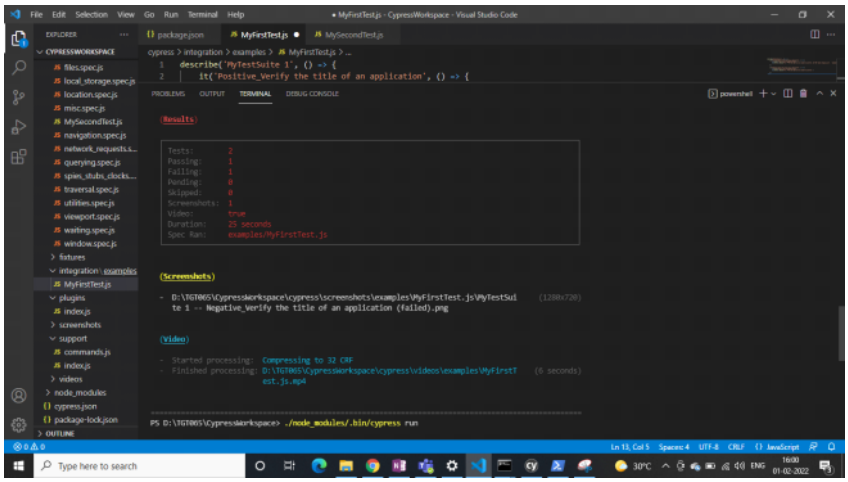
**Execution Policy Change**

The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose you to the security risks described in the about_Execution_Policies help topic at https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): yes
PS C:\Windows\system32>

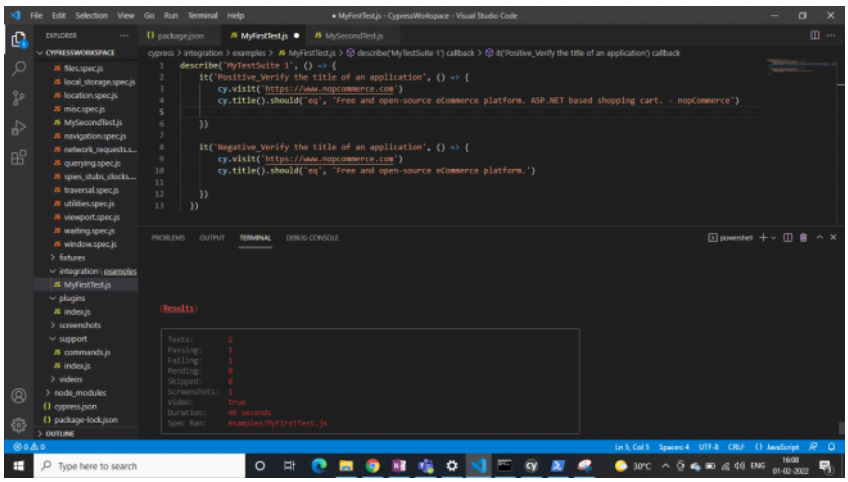



- Now you can run your script from cmd and terminal.



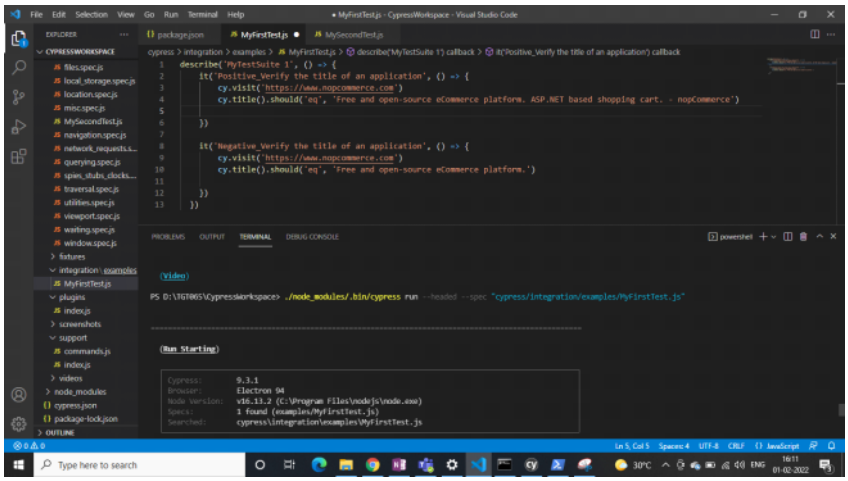


- If user want to execute tests from cmd or terminal but want to see execution/ browser then user can use following command;
- ./node_modules/.bin/cypress run --headed

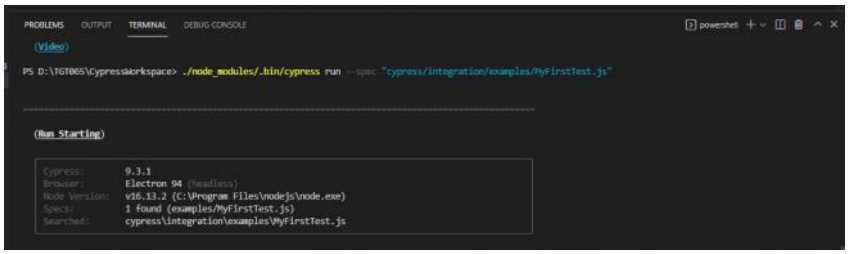


- If user want to run specific test case from the example folder then user need to give file path as below

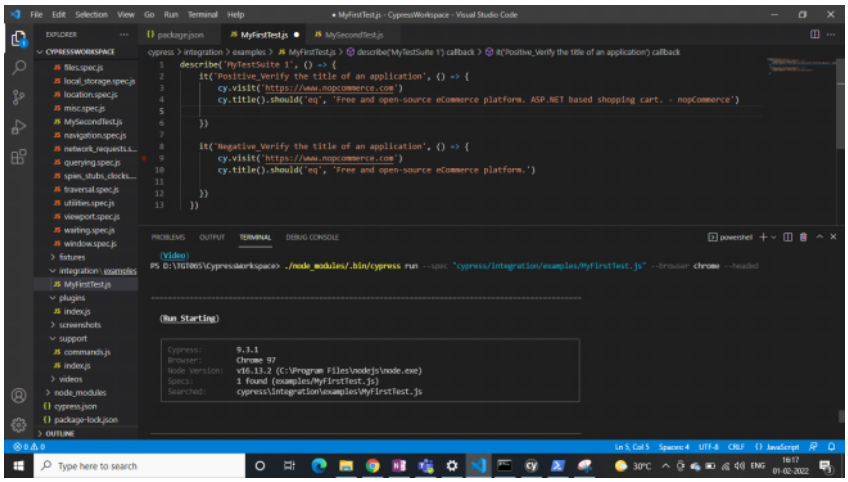
- ./node_modules/.bin/cypress run --headed --spec "cypress/integration/examples/MyFirstTest.js"
- --spec and file path is the synatx to run perticular test case from the example folder.



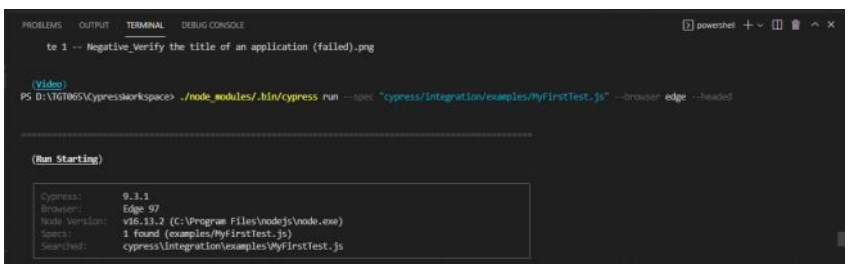
**Headless execution:**



- As cypress script runs on electron browser by default but user want to run it on any other browser like chrome, edge then user can use following command.
- ./node_modules/.bin/cypress run --spec "cypress/integration/examples/MyFirstTest.js" --browser chrome --headed
- -- browser browserName is the synatx for it.



./node_modules/.bin/cypress run --spec "cypress/integration/examples/MyFirstTest.js" --browser edge --headed



- want to run tests from a single spec file and record the results on the Dashboard, the command should be:
- ./node_modules/.bin/cypress run -- --record --spec "cypress/integration/my-spec.js"

**cypress run --no-exit**

To prevent the Cypress Test Runner from exiting after running tests in a spec file, use `--no-exit`.
You can pass `--headed --no-exit` in order to view the command log or have access to developer tools after a `spec` has run.

```
cypress run --headed --no-exit
```

**cypress run --parallel**

Run recorded specs in parallel across multiple machines.

```
cypress run --record --parallel
```

```
cypress run [options]
```

**Options**

| Option | Description |
|---|---|
| --browser, -b | Run Cypress in the browser with the given name. If a filesystem path is supplied, Cypress will attempt to use the browser at that path. |
| --ci-build-id | Specify a unique identifier for a run to enable grouping or parallelization. |
| --config, -c | Specify configuration |
| --config-file, -C | Specify configuration file |
| --env, -e | Specify environment variables |
| --group | Group recorded tests together under a single run |
| --headed | Displays the browser instead of running headlessly |
| --headless | Hide the browser instead of running headed (default during cypress run) |
| --help, -h | Output usage information |
| --key, -k | Specify your secret record key |
| --no-exit | Keep Cypress Test Runner open after tests in a spec file run |
| --parallel | Run recorded specs in parallel across multiple machines |
| --port,-p | Override default port |
| --project, -P | Path to a specific project |
| --quiet, -q | If passed, Cypress output will not be printed to stdout. Only output from the configured Mocha reporter will print. |
| --record | Whether to record the test run |
| --reporter, -r | Specify a Mocha reporter |
| --reporter-options, -o | Specify Mocha reporter options |
| --spec, -s | Specify the spec files to run |
| --tag, -t | Identify a run with a tag or tags |

Opens the Cypress Test Runner.

```
cypress open [options]
```

**Options:**

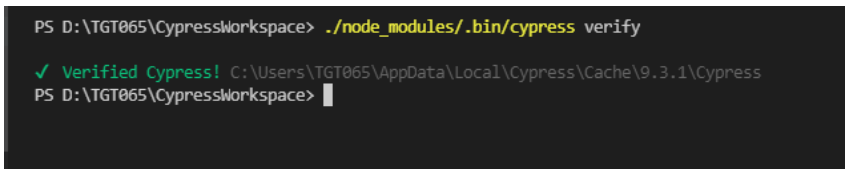
Options passed to `cypress open` will automatically be applied to the project you open. These persist on all projects until you quit the Cypress Test Runner. These options will also override values in your configuration file (`cypress.json` by default).

| Option | Description |
|---|---|
| --browser, -b | Path to a custom browser to be added to the list of available browsers in Cypress |
| --config, -c | Specify configuration |

| | |
|---|---|
| --config-file, -C | Specify configuration file |
| --detached, -d | Open Cypress in detached mode |
| --env, -e | Specify environment variables |
| --global | Run in global mode |
| --help, -h | Output usage information |
| --port, -p | Override default port |
| --project, -P | Path to a specific project |

**cypress verify**

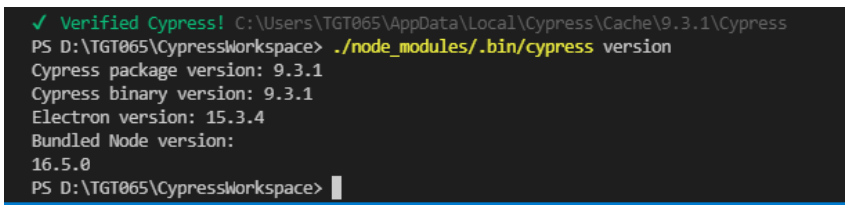
Verify that Cypress is installed correctly and is executable.




**cypress version**

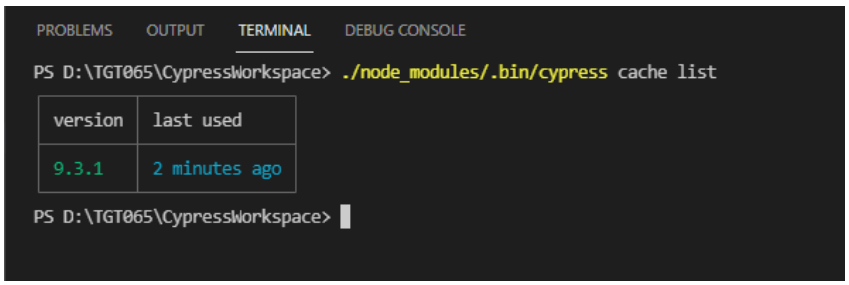
Prints the installed Cypress binary version, the Cypress package version, the version of Electron used to build Cypress, and the bundled Node version.

In most cases the binary and the package versions will be the same, but they could be different if you have installed a diffe rent version of the package and for some reason failed to install the matching binary version.




**cypress cache list**

Print all existing installed versions of Cypress. The output will be a table with cached versions and the last time the binar y was used by the user, determined from the file's access time.



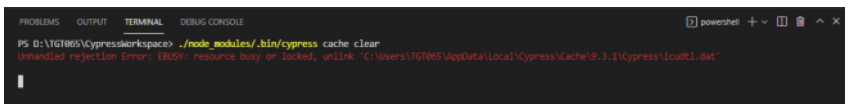

**cypress cache clear**

Clear the contents of the Cypress cache. This is useful when you want Cypress to clear out all installed versions of Cypress that may be cached on your machine. After running this command, you will need to run `cypress install` before running Cypress again.

**cypress cache prune**

Deletes all installed Cypress versions from the cache except for the currently-installed version.

`cypress cache prune`

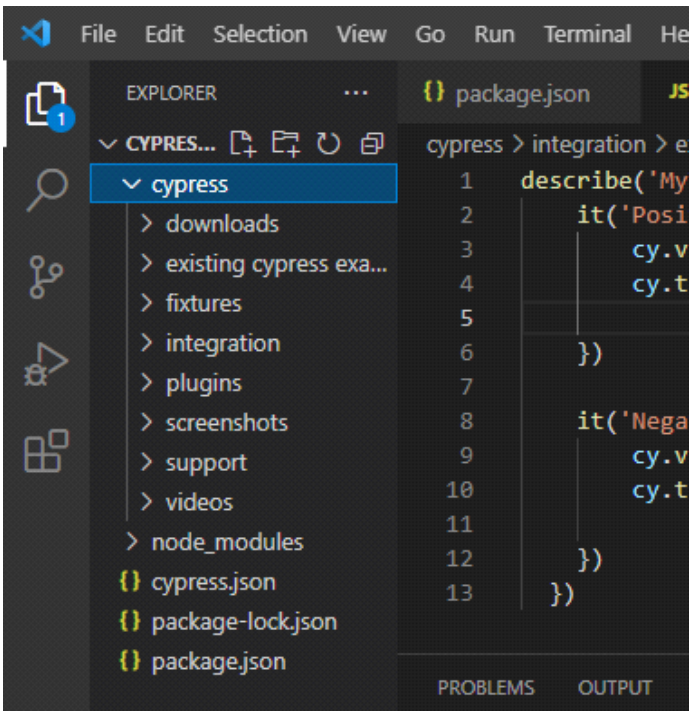


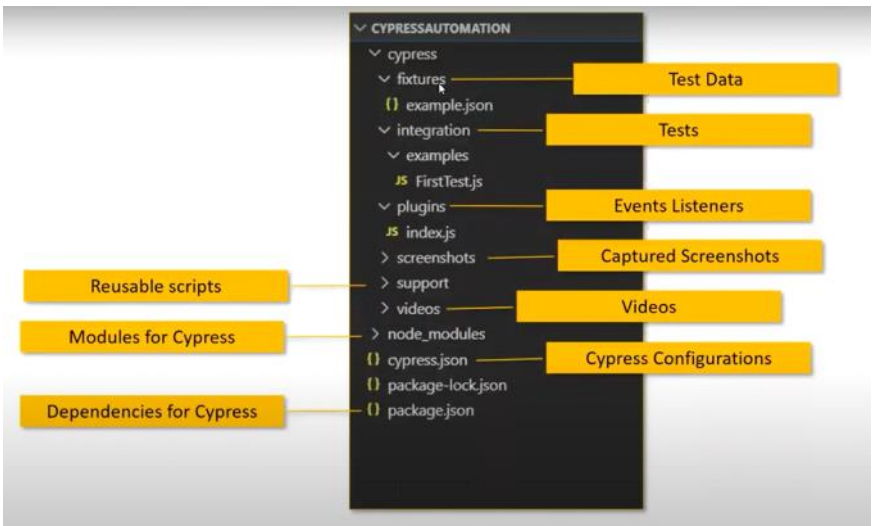
# Cypress 6 - Cypress Project Folder Structure

01 February 2022 16:37

**Cypress Project folder structure:**

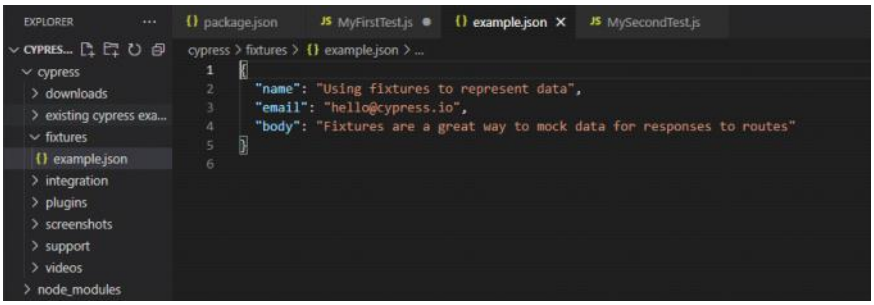
- As soon as we installed cypress, cypress by default will create the folder structure.
- Cypress will be having Mocha framework by default.




**Fixtures:**

- Fixture normally used to maintain test-data.
- Sometimes our test cases need test-data in the form of JSON file, XML file, Excel file so all the test-data file we will maintained in the fixture.
- Purpose of the fixture folder is to maintain test-data.
- Test cases by default refer this location to read the test-data files.
- By default example.json file is available there in the form of JSON file.
- We can also create our own test data file inside fixtures.

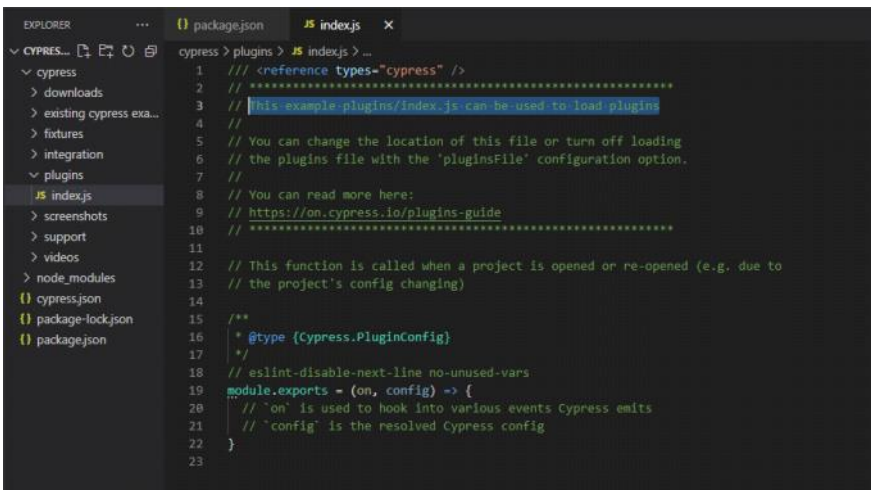



**Integration:**

- Integration folder will have sub-folder as an examples.
- We always create our test cases only inside examples folder.
- By default cypress will look for test cases only integration\examples folder.
- You should not create any test cases outside this example folder.

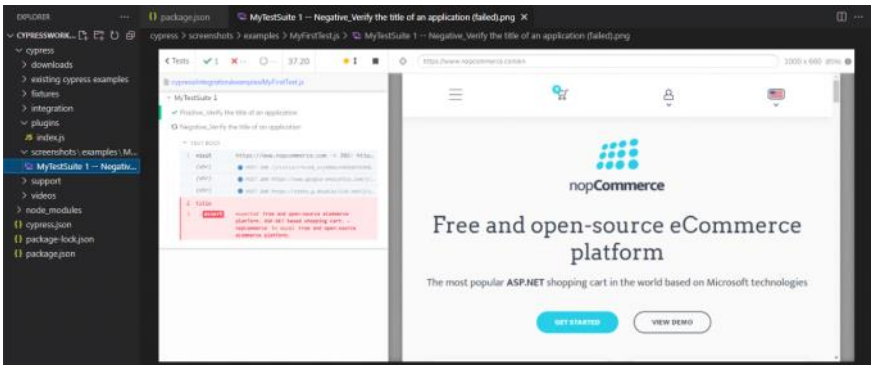

**Plugins:**

- Basically contains an even listeners.
- Suppose out test case got executed and you can blocks some of the events, it means your test case got passed, failed, skipped etc. so based upon an event you need to listen those events and based on those events you need to perform some tasks.
- It is similar concept as Listener TestNG in Selenium.
- Within plugin folder we have to specify the event listeners.
- Based on the events we can perform pre and post tasks.
- This example plugins/index.js can be used to load plugins



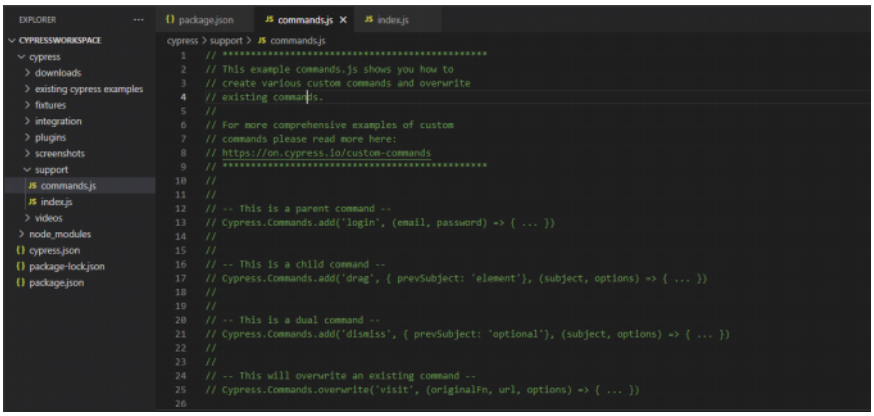

**Screenshots:**

- In this folder by defaults screenshots will be saved as soon as your test got failed.
- Failed screenshots will be available inside this folder.
- Screenshot will be saved here with location hierarchy as below.
- MyTestSuite 1 -- Negative_Verify the title of an application (failed).png



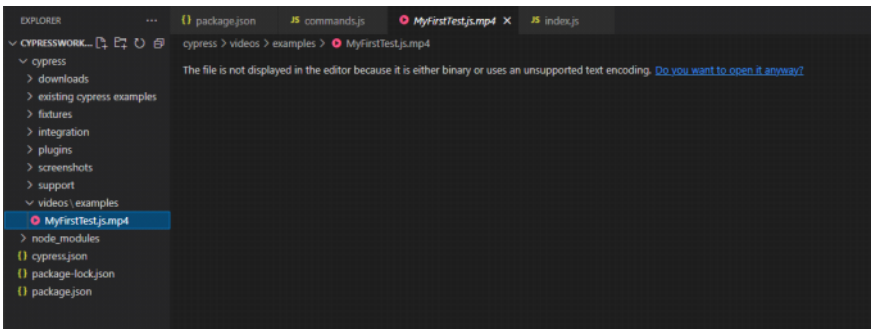

**Support:**

- Support folder contains re-usable scripts.
- Sometime multiple test cases will share the common scipt that we can maintain inside support folder.
- ```
  This example commands.js shows you how to
  // create various custom commands and overwrite
  // existing commands.
  ```
- ```
  This example support/index.js is processed and
  // loaded automatically before your test files.
  ```



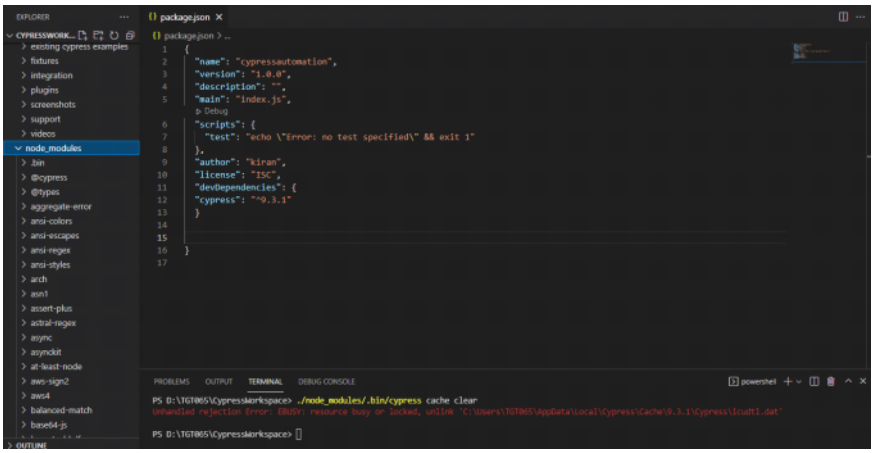

**Videos:**

- Videos folder will maintain our test execution as video and it will be saved here.
- To record video of execution we need to configure settings.



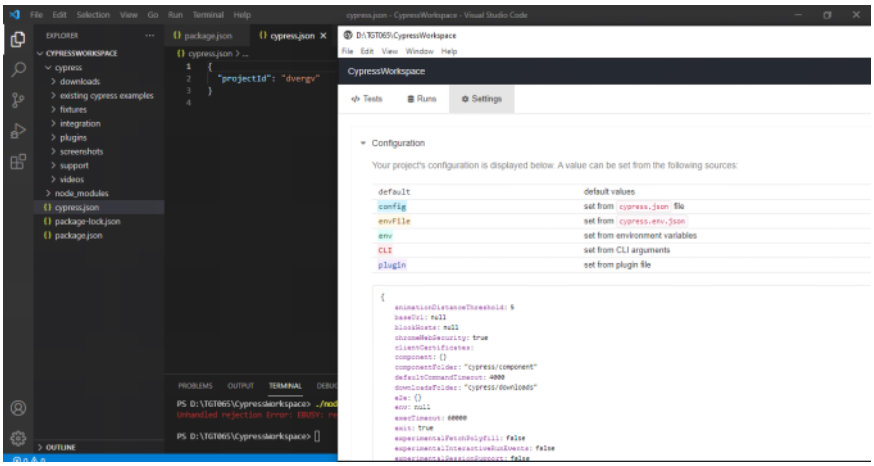

**Noed_modules:**

- As soon as we installed cypress, by default we will get this node_modules folder.
- Node modules is a library.
- Internally it will be having mocha framework.



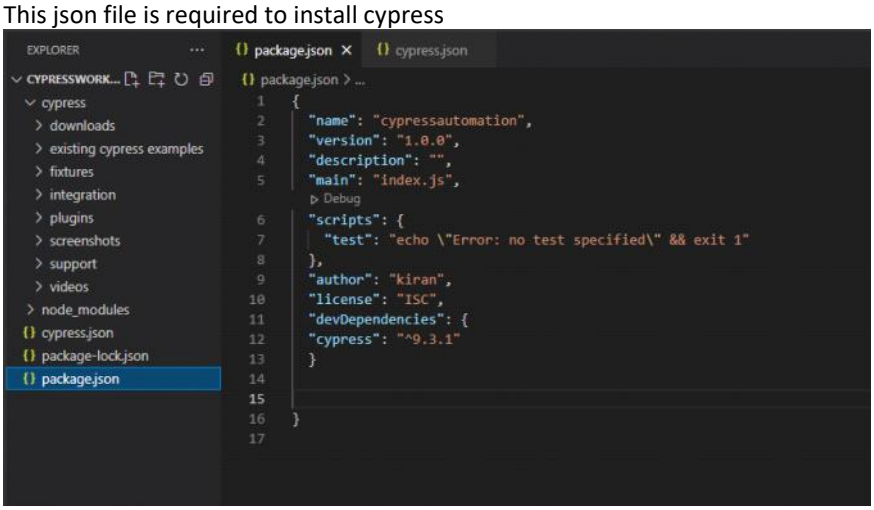
**Cypress.json:**

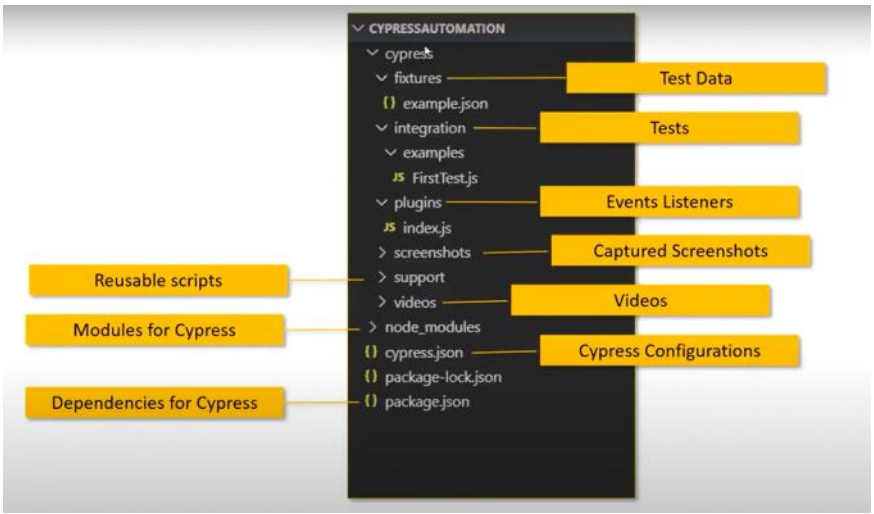
- This file specify the cypress configurations.
- You can check cypress confs as below.
- Open test-runner and navigate to the settings/ configuration
- We can re configure this cypress configuration with help of cypress.json file.
- Example: we can re-confiure the default time
    - Existing time-out is: defaultCommandTimeout:4000\
    - Modified time-out is: defaultCommandTimeout:6000\
- Cypress test-runner level configuration is global configuration and it will applicable for the all test cases.



**Package.json:**

- This json file is required to install cypress



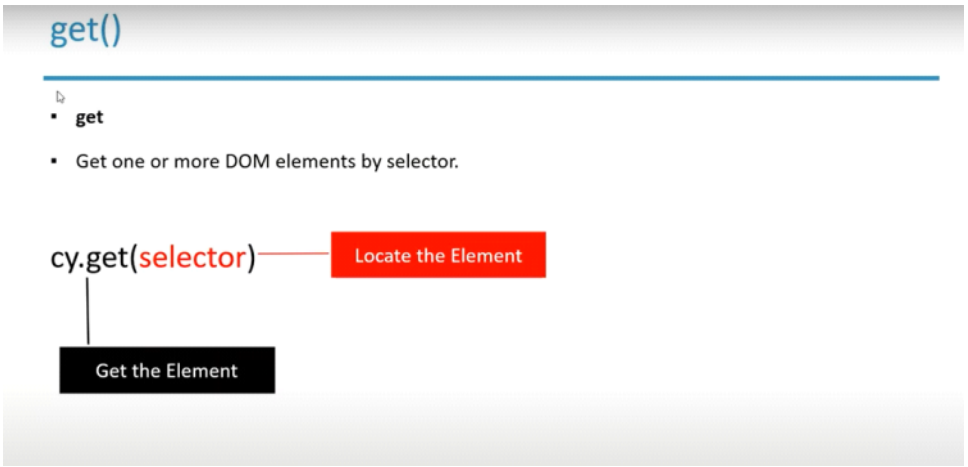

CYPRESSAUTOMATION
cypress
fixtures
example.json
integration
examples
FirstTest.js
plugins
index.js
screenshots
support
videos
node_modules
cypress.json
package-lock.json
package.json
Test Data
Tests
Events Listeners
Captured Screenshots
Videos
Cypress Configurations
Reusable scripts
Modules for Cypress
Dependencies for Cypress

# Cypress 7 - Locators in Cypress

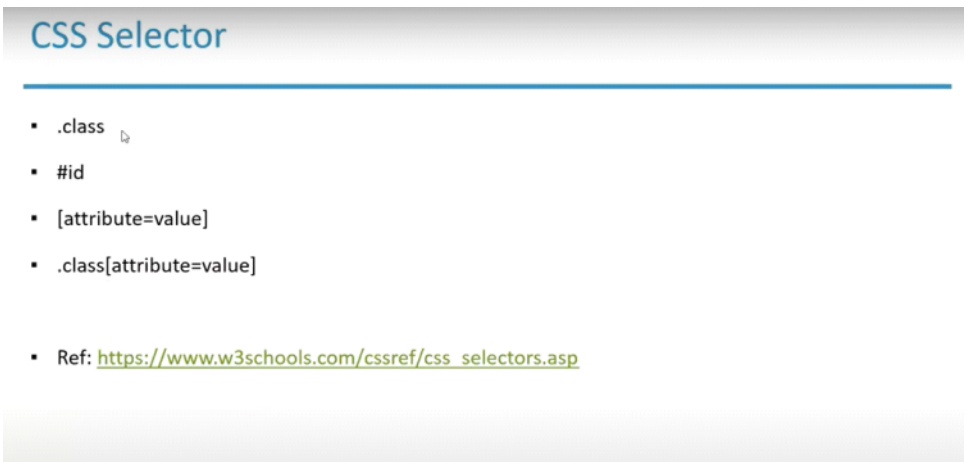
01 February 2022 18:43

**Locators in Cypress:**

- Cypress support CSS Selector as locator.
- CSS Sel we can write in different combination and we can use id, name, class etc.



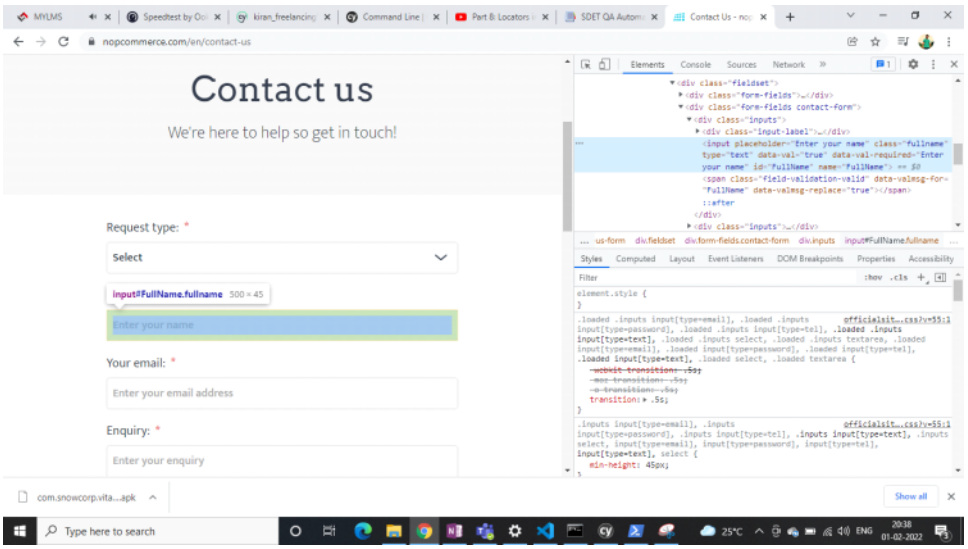

- To locate the element on cypress, method available called cy.get(Selector).
- This method will find that particular element based on the selector we passed as an input.



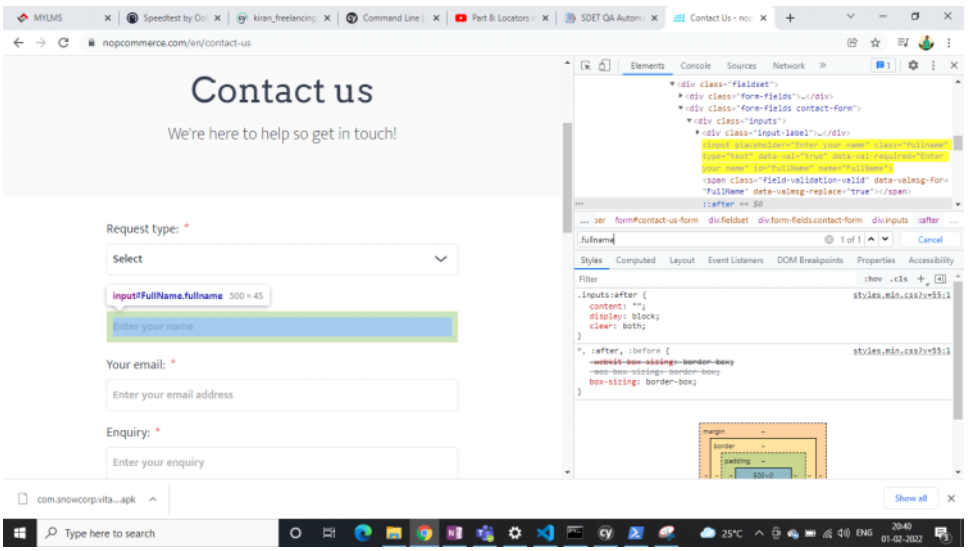

- Whatever attributes are available in page dom we have to use CSS Sel.
- Based on those attribute we can write our own customize selectors.

**Class as a CSS Selector:**

- Syntax is dot[.] followed by classname.
- .ClassName
- Example: `class="fullname"`
- .fullname
- You can locate an element (Enter user name textbox) by using class as CSS Selector.

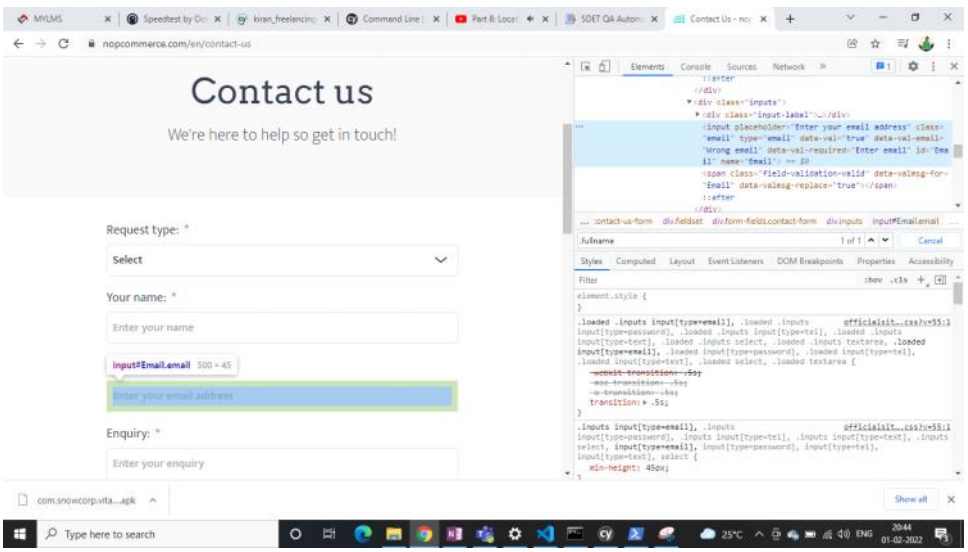



.fullname



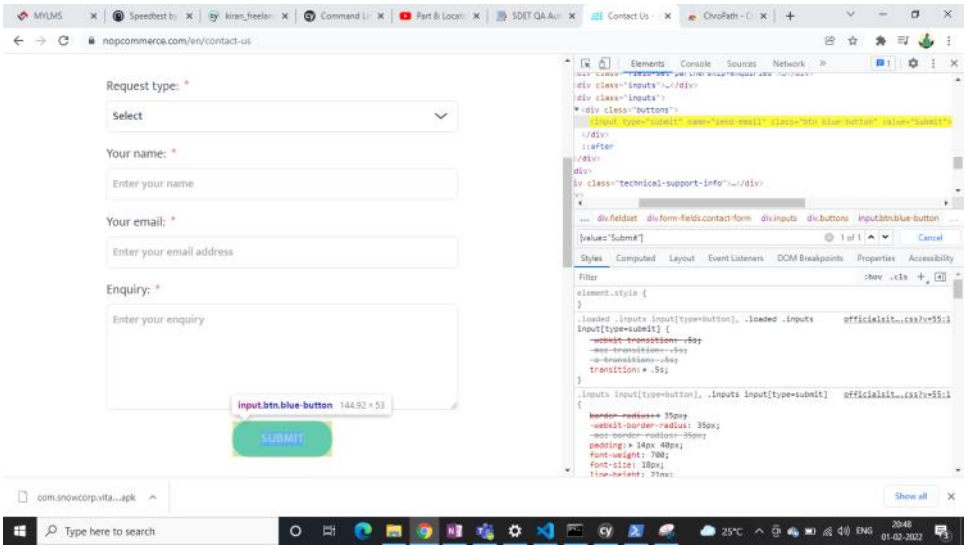

**ID as CSS Selector:**

- Syntax is #id
- `id="Email"`
- #email
- We can locate Email textbox by using ID as CSS Selector.
-



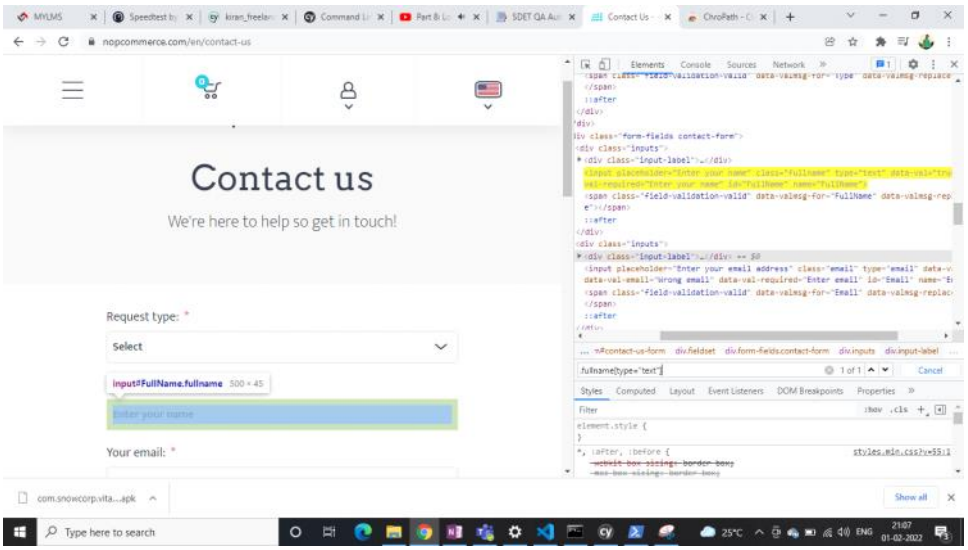
## Attribute as CSS Selector:

- Syntax - [attribute=value]
- In square brackets we can give attribute name = attribute value as CSS Selector.
- Locate submit button with value attribute.




## Class and attribute combination as CSS Selector:

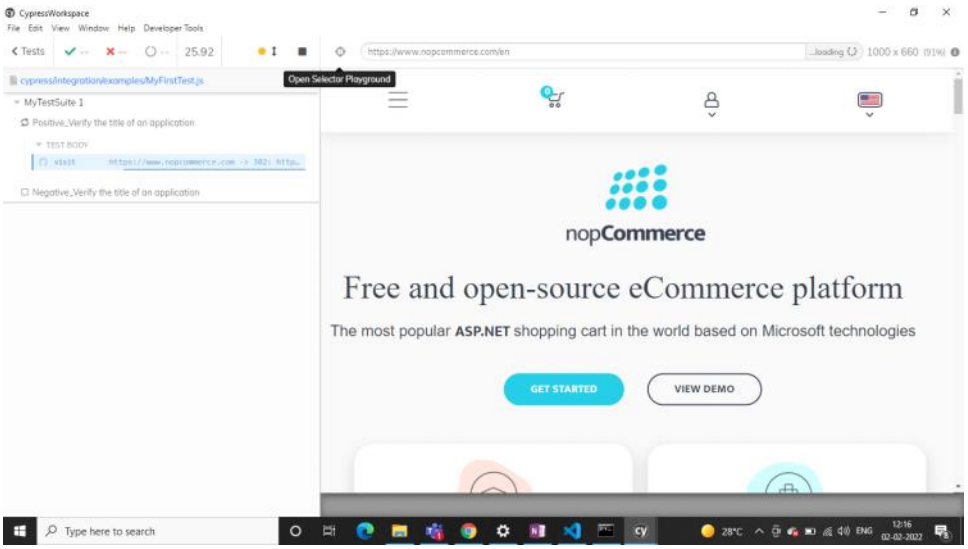
- Syntax: .className[attributeName = attributeValue]
- .form-fields contact-form[type=text]
- .fullname[type="text"]




- Download chropath and use it.
- https://chrome.google.com/webstore/detail/chropath/ljngjbnaijcbncmcnjfhigebomdlkcjo/related
- We can easily capture CSS Selector from chropath.

## CSS Selector from Test Runner:

- You will find the open named "Open Selector Playground"
- This is inbuilt provided by cypress.

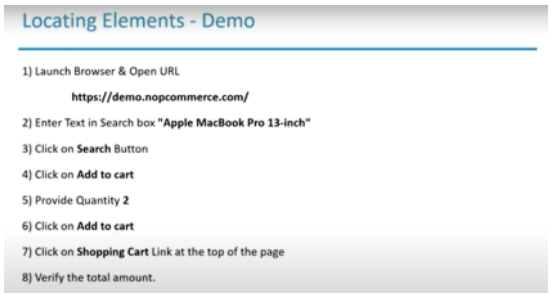



- There are lot of combinations available to **define customize selector.**
- https://www.w3schools.com/cssref/css_selectors.asp

# Cypress 8 - Locators in Cypress - Demo

02 February 2022 11:06

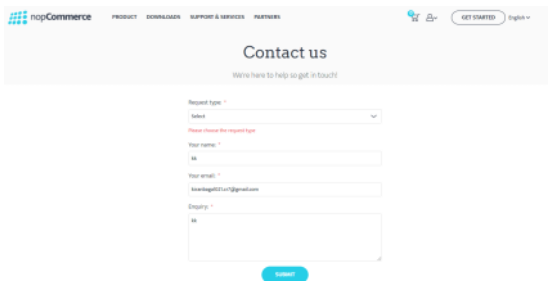
**write your first test case. Locators in Cypress - Demo**




**Below navigation we are going to automate now.**

- Navigate to the application: https://www.nopcommerce.com/en
- Click on Support and Services
- Click on Contact us
- Enter username
- Enter email
- Enter enquiry
- Click on Submit
- Verify the text (field level error)

| Method | Action | Selector |
|---|---|---|
| Click | Support and Services | body > div.master-wrapper-page > header > div > div.header-menu > ul > li:nth-child(3)<br><br>body > div.master-wrapper-page > header > div > div.header-menu > div.menu-toggle<br><br>:nth-child(3) > .default > :nth-child(1) |
| Click | Contact US | body > div.master-wrapper-page > header > div > div.header-menu > ul > li:nth-child(3) > ul > li:nth-child(5) > a<br><br>#mobile-menu > li:nth-child(3) > ul > li:nth-child(5) > a |
| Select | Request Type | |
| Enter | Username | #FullName |
| Enter | Email | [name="Email"] |
| Enter | Enquiry | #Enquiry |
| Click | Submit | #contact-us-form > div > div.form-fields.contact-form > div:nth-child(6) > div > input |
| Verify | Filed level error | #contact-us-form > div > div:nth-child(1) > div > span |
| Click | Hamberger menu | body > div.master-wrapper-page > header > div > div.header-menu > div.menu-toggle > span |

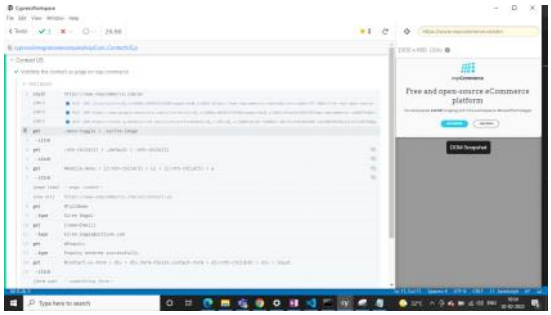


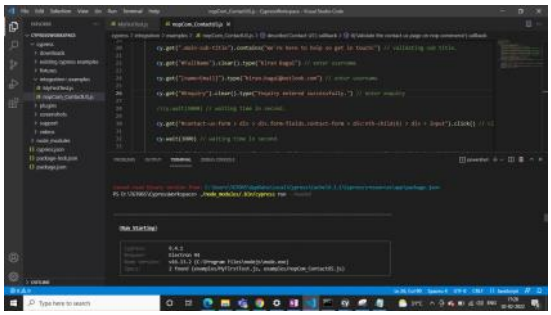

**Created new js file with name: nopCom_ContactUS.js**

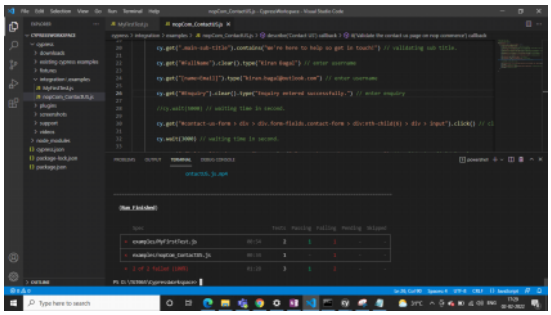
- In the beginning we have added addition statements:
  `<reference types="cypress" />`
- When you will add this at the beginning of we will get auto suggestions from cypress.

```
{
  "projectId": "dvergv"
}
```

- **Execution completed successfully.**







Program:

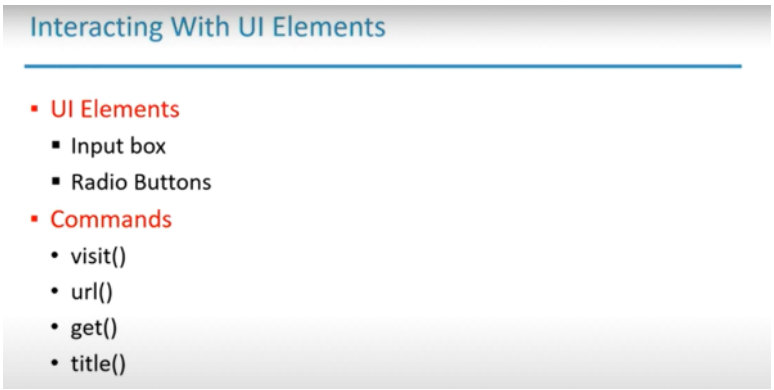
```
describe('Contact US', () => {
    it('Validate the contact us page on nop commerce', () => {
        //cy.visit('https://www.nopcommerce.com')
        cy.visit("https://www.nopcommerce.com/en") // launch the url in browser
        //cy.get("body > div.master-wrapper-page > header > div > div.header-menu > div.menu-toggle > span").click() // click on hamberger menu
        cy.get(".menu-toggle > .sprite-image").click() // click on hamberger menu

        //cy.get("body > div.master-wrapper-page > header > div > div.header-menu > ul > li:nth-child(3)").click() // click on support and services
        cy.get(":nth-child(3) > .default > :nth-child(1)").click() // click on support and services
        //cy.get("body > div.master-wrapper-page > header > div > div.header-menu > ul > li:nth-child(3) > ul > li:nth-child(5) > a").click() // click on contact us
        cy.get("#mobile-menu > li:nth-child(3) > ul > li:nth-child(5) > a").click() // click on contact us
        cy.get(".main-sub-title").contains("We're here to help so get in touch!") // validating sub title.
        cy.get("#FullName").clear().type("Kiran Bagal") // enter username
        cy.get("[name=Email]").type("kiran.bagal@outlook.com") // enter username
        cy.get("#Enquiry").clear().type("Enquiry entered successfully.") // enter enquiry
        //cy.wait(5000) // waiting time in second.
        cy.get("#contact-us-form > div > div.form-fields.contact-form > div:nth-child(6) > div > input").click() // click on submit button
        cy.wait(3000) // waiting time in second.
        cy.get(".field-validation-error").contains("Please choose the request type") // validating field level error
        cy.wait(3000) // waiting time in second.
     })

})
```

# Cypress 9 - Interacting with GUI Elements in Cypress | Input Box & Radio Buttons & Check Box & Drop Downs

02 February 2022 17:39

Interacting with GUI Elements in Cypress | Input Box & Radio Buttons:

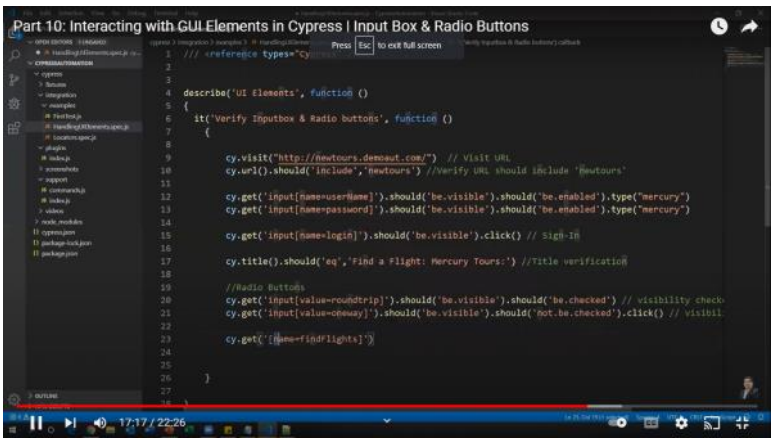
- How to interact with input box and radio button



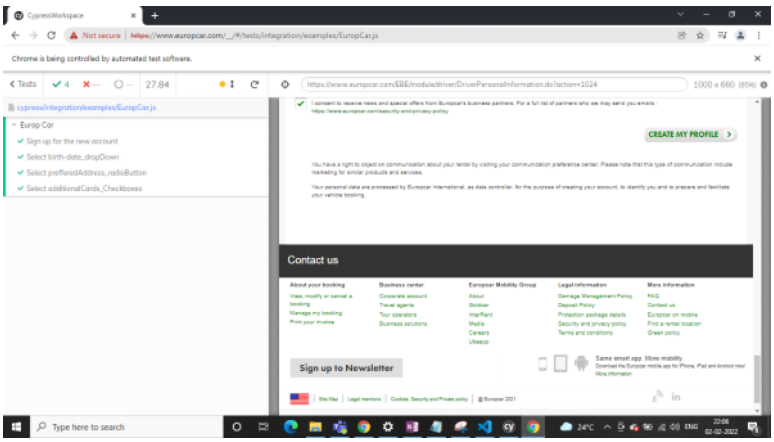

- Demo application: https://www.europcar.com/

| Method | Action | Selector | Testdata |
|---|---|---|---|
| Visit | Navigate | | https://www.europcar.com/<br><br>#didomi-notice-disagree-button > span |
| Click | Hamberger menu | #burgerMenuBtn>.ecw-icon | |
| Click | Access your account | .ecw-burger-menu__list> ul> li:nth-child(4)> a | |
| Click | Create my account | .maincontent_grey> div.login_tab_footer> a | |
| Enter | First name | #firstName | Kiran |
| Enter | Last name | #lastName | Bagal |
| Select | Day | #day[value='06'] | |

- Below screenshots are having the method to check whether radio button is clicked or not
- Visibility check
- Radio button check or not checked.
- Be.check
- Not.be.check
- https://www.youtube.com/watch?v=9W8eyijS0N4



Test result

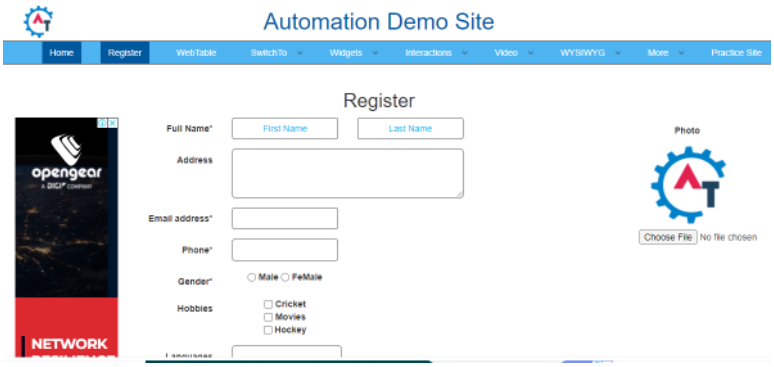



```
/// <reference types="Cypress" />
 describe('Europ Car', () => {
   it('Sign up for the new account', () => {
      cy.visit("https://www.europcar.com/") // launch the url
      cy.wait(3000)
      cy.get("#didomi-notice-disagree-button > span").click() // click on disagree button to close pop-up
      cy.get("#burgerMenuBtn>.ecw-icon").click() // click on hamberger menu
      cy.get(".ecw-burger-menu__list> ul> li:nth-child(4)> a").should('be.visible').click() // click on access your account
      cy.title().should('eq', 'My Account Privilege - Europcar') // verify the title of application
      cy.get(".maincontent_grey> div.login_tab_footer> a").click() //click on create my account
      cy.get("#contents> div> div> div> div> h1").contains("Create My Account") // validate text
      cy.get("#firstNameErrorDiv > label.strong").should('be.visible').contains("First name*") //verify label
      cy.get("#lastNameErrorDiv > label.strong").should('be.visible').contains("Last name*") //verify label
      cy.get("#firstName").should('be.enabled').clear().type("kiran") // enter first name
      cy.get("#lastName").should('be.enabled').clear().type("bagal") // enter last name

    })
    it('Select birth-date_dropDown', () => {
      cy.get("#day").select('18').should('have.value', '18')
      cy.get("#month").select('12').should('have.value', '12')
      cy.get("#year").select('1992').should('have.value', '1992')
      cy.get("#fok> div> a").should('be.visible').click()
      cy.wait(3000)
      cy.get("#didomi-notice-disagree-button > span").click() // click on disagree button to close pop-up
      cy.title().should('eq', 'My Account - Personal Information')

    })
    it('Select prefferedAddress_radioButton', () => {
     //cy.get("#didomi-notice-disagree-button > span").click() // click on disagree button to close pop-up
     cy.get('input[value=H]').should('not.be.checked').click().should('be.checked')
     cy.wait(2000)
     cy.get('input[value=P]').should('not.be.checked').click().should('be.checked')
     //cy.wait(2000)
     cy.get("#firstName").clear().type('kiran')
     cy.get("#lastName").clear().type('bagal')
     cy.get("#labelPreferredAddr").contains('Preferred Address:')

     })
     it('Select additionalCards_Checkboxes', () => {
       cy.get("#phoneNumber").type('9890274568')
       cy.get("#birthCountryCode").select('Argentina').should('have.value', 'AR')
       cy.get("#g2_0 > form > div > div.content > div:nth-child(96) > span").contains('Credit card / debit card information')
       cy.get("#displayAdditionalCardBtn1Div> span> a").should('be.visible').click()
       cy.get("#displayAdditionalCard2Div > div.green_bgd > fieldset:nth-child(2) > div.rfield > div > div > label").click()
       cy.get("#generalConditionsErrorDiv > label").click()
       cy.get("#g2_0 > form > div > div.content > div:nth-child(127) > label").click()
       //cy.get("#g2_0 > form > div > div.content > div:nth-child(127) > label").click().and('have.text', 'I consent to receive news and special
offers from Europcars business partners. For a full list of partners who we may send you emails : ')


       //cy.get("#g2_0 > form > div > div.content > div:nth-child(125)").check().should('be.checked').and('have.id', 'generalConditions')
       //cy.get("##g2_0 > form > div > div.content > div:nth-child(126) > label").check().should('be.checked').and('have.class', 'checkbox-
```

```
label')
        //:nth-child(123) > label



        })

})
```

**Deal with radio button, drop-down and checkboxes in cypress:**

- Demo website: http://demo.automationtesting.in/Register.html




**TO Deal with Radio buttons:**

- Select gender radio buttons for gender
- Check checkboxes for hobbies
- Radio button methods

```
it('Deal with radio buttons', () => {

   cy.get('input[value=Male]').should('be.visible').should('not.be.checked').click().should('be.checked')
   cy.get('input[value=FeMale]').should('be.visible').should('not.be.checked').click().should('be.checked')
   cy.wait(2000)

})
```

- .should('be.visible') : Method will check whether that radio button is visible or not?
- .should('not.be.checked') : this method will make sure that radio button is not to be checked.
- .click() : this method will click/ select radio button on.
- .should('be.checked') : this method will make sure that radio button is to be checked.

**TO Deal with Checkboxes:**

- Check, uncheck, verify and validate checkbox methods:

```
it('Deal with checkboxes', () => {

   cy.get("#checkbox1").should('not.be.checked').check().should('be.checked').and('have.value', 'Cricket')
   cy.get("#checkbox2").should('not.be.checked').check().should('be.checked').and('have.value', 'Movies')
   cy.get("#checkbox3").should('not.be.checked').check().should('be.checked').and('have.value', 'Hockey').uncheck()
   cy.wait(2000)
   cy.get('input[type=checkbox]').check(['Hockey'])
   cy.wait(2000)
   cy.get('input[type=checkbox]').uncheck(['Cricket', 'Movies', 'Hockey'])
   cy.wait(2000)
   cy.get('input[type=checkbox]').check(['Cricket', 'Movies', 'Hockey'])

})
```

- .should('not.be.checked'): this method will make sure that checkbox is not to be checked.
- .check() : this method will check the checkbox.
- .uncheck() : this method will un-check the checkbox.
- .should('be.checked') : this method will make sure that checkbox should be checked.
- .and('have.value', 'Cricket') : this method will make sure that checkbox which is checked will have the value as cricket.

**TO Deal with Drop-downs:**

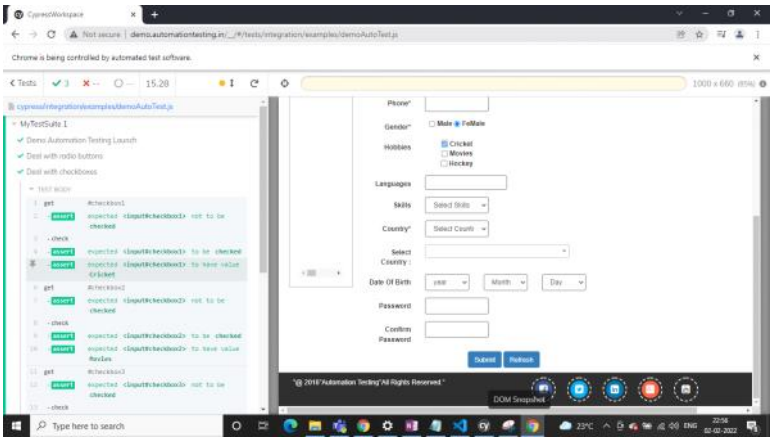
## Drop-down with Select tags:

- Select, un-select checkboxes

```
it('Select birth-date_dropDown', () => {
  cy.get("#day").select('18').should('have.value', '18')
  cy.get("#month").select('12').should('have.value', '12')
  cy.get("#year").select('1992').should('have.value', '1992')
  cy.get("#fok> div> a").should('be.visible').click()
  cy.wait(3000)
  cy.get("#didomi-notice-disagree-button > span").click() // click on disagree button to close pop-up
  cy.title().should('eq', 'My Account - Personal Information')

})
```

- .select('18') : this method will select the value/ option as 18 from the select drop-down
- .should('have.value', '18') : this method will make sure that value/ option as 18 has been selected from the drop-down

**Test Results:**

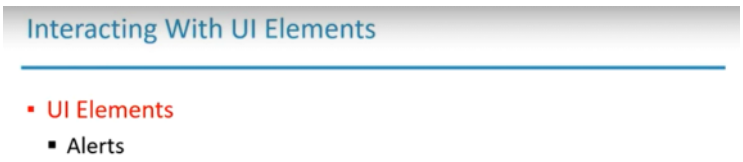


# Cypress 10 - Interacting with GUI Elements in Cypress | Handling Alerts

02 February 2022 22:57

**Interacting with GUI Elements in Cypress | Handling Alerts:**

- Ref link: https://www.youtube.com/watch?v=yzfp85bVUjY




- Alert application site: https://mail.rediff.com/cgi-bin/login.cgi
- Cypress will automatically take care of alert windows.
- We don't require to write script to handle with alerts.
- If we want to verify something on alert window then we have to write the code for the same.
- By default cypress will close alert window automatically.

Error 1:

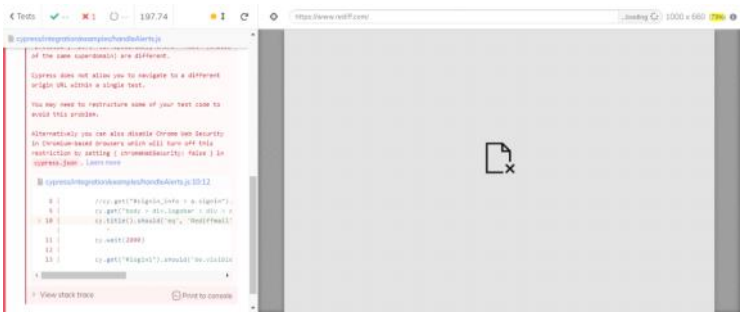
```
cy.visit('https://mail.rediff.com/cgi-bin/login.cgi')
```

**SecurityError**
Blocked a frame with origin "https://www.rediff.com" from accessing a cross-origin frame.

Error 2:

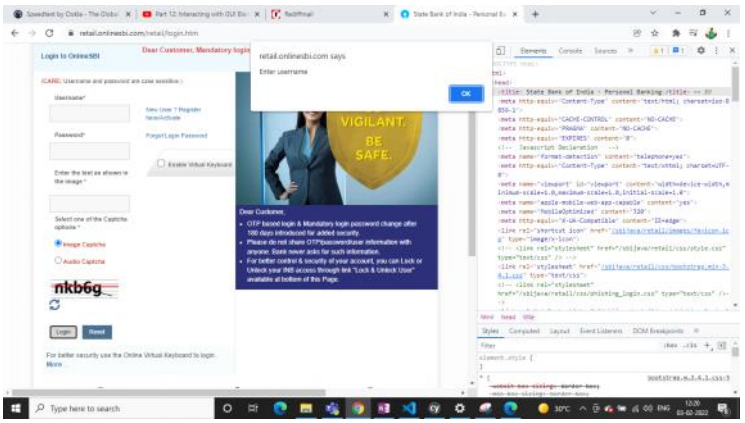
```
cy.visit('https://www.rediff.com/')
```

**CypressError**
Cypress detected a cross origin error happened on page load: > Blocked a frame with origin "https://www.rediff.com" from accessing a cross-origin frame. Before the page load, you were bound to the origin policy: > https://rediff.com A cross origin error happens when your application navigates to a new URL which does not match the origin policy above. A new URL does not match the origin policy if the 'protocol', 'port' (if specified), and/or 'host' (unless of the same superdomain) are different. Cypress does not allow you to navigate to a different origin URL within a single test. You may need to restructure some of your test code to avoid this problem. Alternatively you can also disable Chrome Web Security in Chromium-based browsers which will turn off this restriction by setting { chromeWebSecurity: false } in `cypress.json`.Learn more



**Sbi Login**

| | | |
|---|---|---|
| Url | https://retail.onlinesbi.com/retail/login.htm | |
| Click | #banner > div.continue_btn > a | |
| Click | #Button2 | |




- Cypress will automatically close this alert window.
- We don't require to add code to close the alert. Cypress will automatically take care of that.
- Some cases we need to perform some validation in alert window.
- To perform alert validation we have to write own event.
- As soon as alert pops-up appears internally certain events are triggered.
- We cannot capture alert window directly but whenever we open alert window parallely we need to run an event, need to rise an event.
- That event will capture the alert message.
- Cypress document on event: https://docs.cypress.io/api/events/catalog-of-events#Event-Types

| Event | Details |
|---|---|
| Name: | window:alert |
| Yields: | the alert text (String) |
| Description: | Fires when your app calls the global window.alert() method. Cypress will auto accept alerts. You cannot change this behavior. |

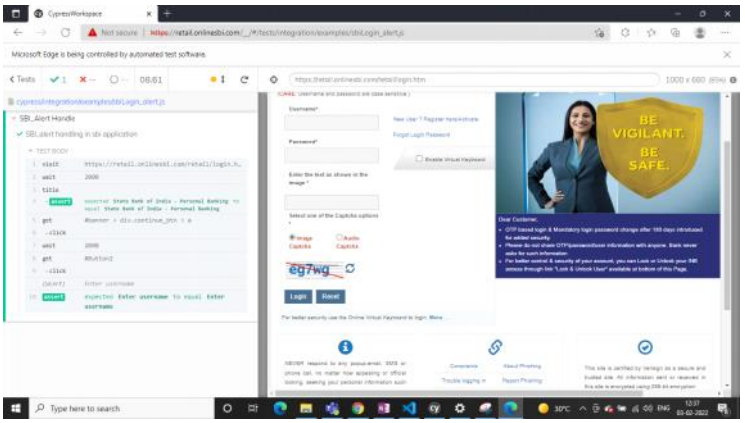
**Syntax to write an event**

**Code:**

```
/// <reference types="Cypress" />
describe('SBI_Alert Handle', () => {
   it('SBI_alert handling in sbi application', () => {
      cy.visit('https://retail.onlinesbi.com/retail/login.htm') // lqunch the url
      cy.wait(2000)
      cy.title().should('eq', 'State Bank of India - Personal Banking') // verify the title

      cy.get("#banner > div.continue_btn > a").click() // click on continue login

      cy.wait(2000)
      cy.get("#Button2").click() // click on login

      cy.on('window:alert', (str) =>
      {
         expect(str).to.equal('Enter username')
      })

   })

 })
```

- window:alert is an event we're raising after login click activity.
- We're capturing alert message inside str variable
- Then after we're validating/ comparing that str message with user data

**Execution:**




**Confirmation Alert:**

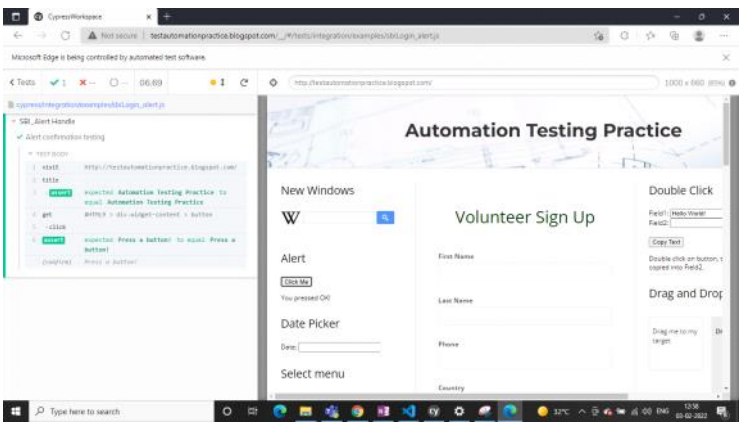
- Ref website: http://testautomationpractice.blogspot.com/
- Doc ref: https://docs.cypress.io/api/events/catalog-of-events#App-Events

| Event | Details |
|---|---|
| Name: | window:confirm |
| Yields: | the confirmation text (String) |
| Description: | Fires when your app calls the global window.confirm() method. Cypress will auto accept confirmations. Return false from this event and the confirmation will be canceled. |

**Code:**

```
it('Alert confirmation testing', () => {
   cy.visit('http://testautomationpractice.blogspot.com/') // lqunch the url
   cy.title().should('eq', 'Automation Testing Practice') // verify the title

   cy.get("#HTML9 > div.widget-content > button").click() // click on click me
   cy.on('window:confirm', (str) =>
   {
      expect(str).to.equal('Press a button!')
   })
})
```

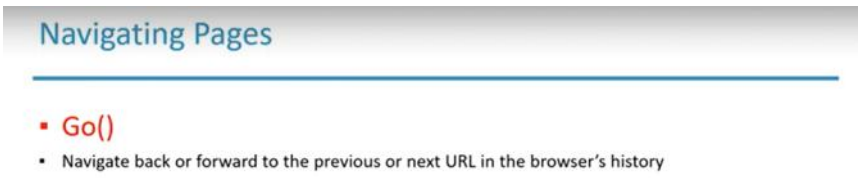
**Execution:**



# Cypress 11 - Navigate back or forward in the browser's history | GO Command

03 February 2022 13:00

**Navigate back or forward in the browser's history | GO Command:**




- https://www.youtube.com/watch?v=V6fBraff5u0
- By using this command we can navigate back or forward to the previous or next url in the browser history.
- Refer: https://docs.cypress.io/api/commands/go

  **Syntax**

  cy.go(direction)

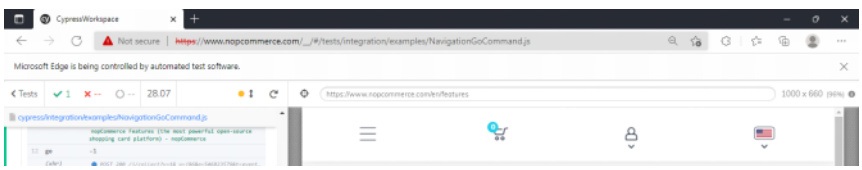
  cy.go(direction, options)

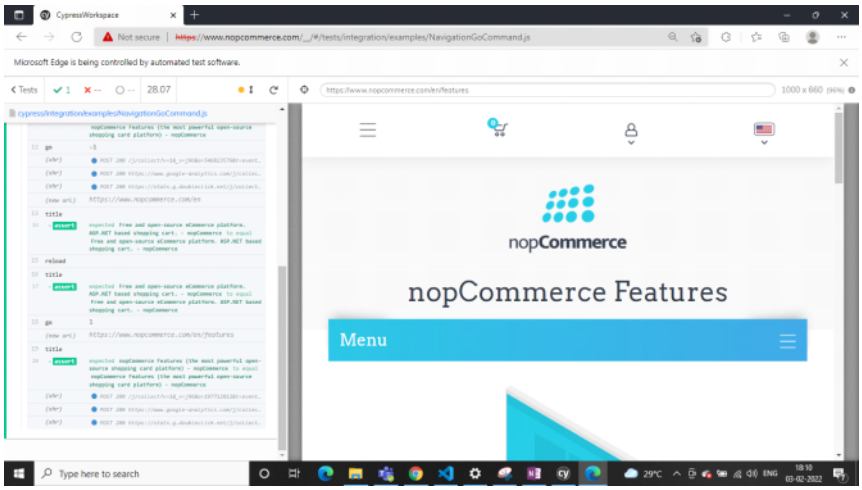
- cy.go('back') // equivalent to clicking back button
- cy.go('forward') // equivalent to clicking forward button
- cy.go(-1) // equivalent to clicking back button
- cy.go(1) // equivalent to clicking forward button
- cy.visit('http://localhost:8000/folders').go('back')
- `cy.reload() // reload current page.`

**Code:**

```
/// <reference types="Cypress" />
describe('Go command_Navigation in cypress', () => {
   it('navigation in browser history', () => {
      cy.visit("https://www.nopcommerce.com/en") // launch url
      cy.title().should('eq', 'Free and open-source eCommerce platform. ASP.NET based shopping cart. - nopCommerce')
      cy.get(".menu-toggle > .sprite-image").click() // click on hamberger menu
      cy.get(":nth-child(1) > .default > :nth-child(1)").click() // click on product
      cy.get("#mobile-menu > :nth-child(1) > .sublist > :nth-child(3) > a").click() // click on features
      cy.title().should('eq', 'nopCommerce Features (the most powerful open-source shopping card platform) - nopCommerce')
      //cy.go('back')
      cy.go(-1)
      cy.title().should('eq', 'Free and open-source eCommerce platform. ASP.NET based shopping cart. - nopCommerce')
      cy.reload()
      cy.title().should('eq', 'Free and open-source eCommerce platform. ASP.NET based shopping cart. - nopCommerce')
      //cy.go('forward')
      cy.go(1)
      cy.title().should('eq', 'nopCommerce Features (the most powerful open-source shopping card platform) - nopCommerce')

   })

 })
```

**Execution:**




Microsoft Edge is being controlled by automated test software.
nopCommerce
nopCommerce Features
Menu

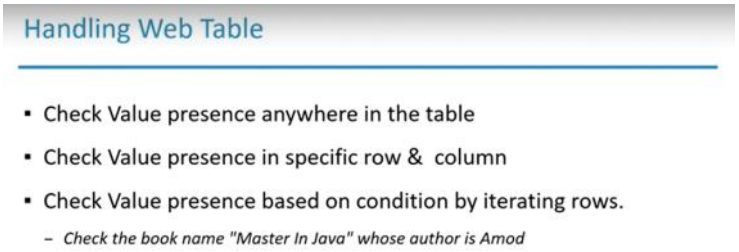
➢

# Cypress 12 - Handling Web/ HTML Table in Cypress

03 February 2022 18:13

**Handling Web/ HTML Table in Cypress:**

- We will chech how we can work with web table
- Ref: https://www.youtube.com/watch?v=XrpvzUr8esY




- Ref: http://testautomationpractice.blogspot.com/
- Below is the table we're going to verify


HTML Table

| BookName | Author | Subject | Price |
|---|---|---|---|
| Learn Selenium | Amit | Selenium | 300 |
| Learn Java | Mukesh | Java | 500 |
| Learn JS | Animesh | Javascript | 300 |
| Master In Selenium | Mukesh | Selenium | 3000 |
| Master In Java | Amod | JAVA | 2000 |
| Master In JS | Amit | Javascript | 1000 |

- Check the value presence anywhere in the table.
- We will verify the JavaScript value present in the table or not?

**Code 1:**

```
/// <reference types="Cypress" />
describe('WebTable in cypress', () => {
  it('web table verification in cypress', () => {
        cy.visit("http://testautomationpractice.blogspot.com/") //launch url
        cy.title().should('eq', 'Automation Testing Practice') // verify title of the page.
        // check perticular value is present in the entire table
        cy.get('table[name=BookTable]').contains('td', 'Javascript').should('be.visible')
        cy.get('table[name=BookTable]').contains('th', 'Price').should('be.visible')
        cy.get('table[name=BookTable]').contains('td', '3000').should('be.visible')
        //cy.get('table[name=BookTable]').contains('td', 'kirankita').should('not.be.visible')
        // check perticular value is present in the specific location - row and colomn
        cy.get("table[name=BookTable] > tbody > tr:nth-child(7) > td:nth-child(2)").contains('Amit').should('be.visible')
        cy.get("table[name=BookTable] > tbody > tr:nth-child(5) > td:nth-child(3)").contains('Selenium').should('be.visible')
        //check value presence based on condition by iterating rows.
        // verify the
        cy.get("table[name=BookTable]> tbody> tr td:nth-child(2)").each(($name,index,$list) => {
          const authorName = $name.text()
          if(authorName.includes("Amod")){
            cy.get("table[name=BookTable]> tbody> tr td:nth-child(1)").eq(index).then(function(bname)
            {
              const bookName = bname.text()
              expect(bookName).to.equal('Master In Java')
            })
          }
        })
  })

})
```

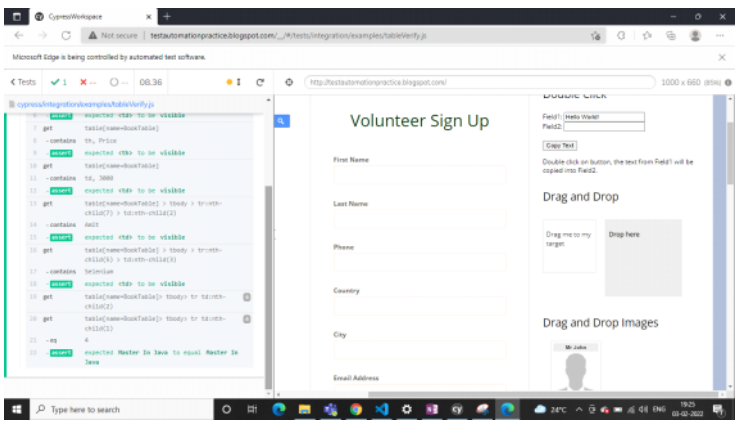
| Method Name | Method Description | Comment |
|---|---|---|

| | | |
|---|---|---|
| cy.visit() | Launch the url | |
| cy.title() | To get the title of the page | |
| cy.get() | Get the locator value for the element | |
| .Contains() | Get the DOM element containing the content. | content (String, Number, RegExp) |
| .should() | BDD Assertions<br><br>https://docs.cypress.io/guides/references/assertions#Chai | .should('not.have.class', 'disabled') |
| :nth-child(n) | The :nth-child(n) selector matches every element that is the nth child, regardless of type, of its parent.<br><br>https://www.w3schools.com/cssref/sel_nth-child.asp | p:nth-child(3n+0) {<br>background: red;<br>} |
| .each() | Iterate through an array like structure (arrays or objects with a length property).<br><br>https://docs.cypress.io/api/commands/each#Syntax | .each(callbackFn)<br><br>cy.get('ul>li').each(() => {...}) // Iterate through each 'li'<br><br>cy.getCookies().each(() => {...}) // Iterate through each cookie |
| .text() | Get text | |
| .eq() | Get A DOM element at a specific index in an array of elements.<br><br>https://docs.cypress.io/api/commands/eq | .eq(index)<br>.eq(indexFromEnd)<br>.eq(index, options)<br>.eq(indexFromEnd, options) |
| .then(callbackFn)<br>.then(options, callbackFn) | Enables you to work with the subject yielded from the previous command.<br><br>https://docs.cypress.io/api/commands/then#Syntax | |

**Explanation:**

- With help of selector(`table[name=BookTable]> tbody> tr td:nth-child(2))` we will point/ match all author name from the table colomn author.
- As soon as author name match with "Amod" immidetly we're shifting to same row first column to check the book name as "Master in Java"
- Here index is representing as row.

**Execution:**




**Code 2:**

```
it('check value presence based on condition by iterating rows.', () => {
  cy.visit("http://testautomationpractice.blogspot.com/") //launch url
  cy.title().should('eq', 'Automation Testing Practice') // verify title of the page.
  //check value presence based on condition by iterating rows.
  cy.get("table[name=BookTable]> tbody> tr td:nth-child(2)").each(($name,index,$list) => {
    const authorName = $name.text()
    if(authorName.includes("Amit")){
      cy.get("table[name=BookTable]> tbody> tr td:nth-child(1)").eq(index).then(function(bname)
      {
        const bookName = bname.text()
        //expect(bookName).to.equal('Master In JS')
          if(bookName.includes("Master in JS")) {
            cy.get("table[name=BookTable]> tbody> tr td:nth-child(1)").eq(index),then(function(subject)
```

```
                {
                    const subjectName = subject.text()
                    expect(subjectName).to.equal('Javascript')
                    console.log("Assertion successful")
                })
            }
        })
    }
  })
})
```

**Verify the "Amit as author" against "bookname is Master in JS" and "Subject as Javascript"**

HTML Table

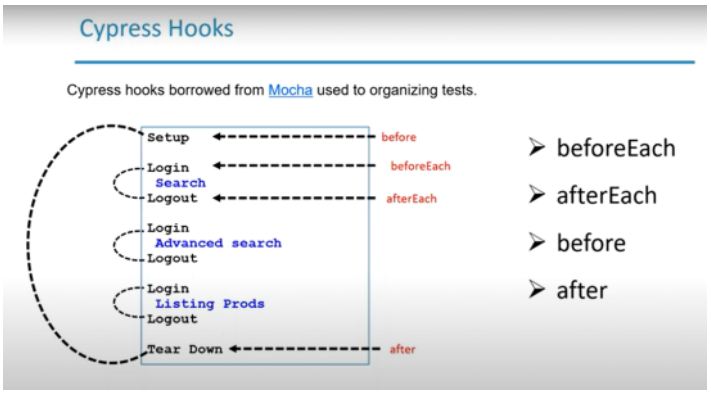
| BookName | Author | Subject | Price |
|---|---|---|---|
| Learn Selenium | Amit | Selenium | 300 |
| Learn Java | Mukesh | Java | 500 |
| Learn JS | Animesh | Javascript | 300 |
| Master In Selenium | Mukesh | Selenium | 3000 |
| Master In Java | Amod | JAVA | 2000 |
| Master In JS | Amit | Javascript | 1000 |

# Cypress 13 - Working with Cypress Hooks | beforeEach | afterEach | before | after

03 February 2022 22:01

**Working with Cypress Hooks | beforeEach | afterEach | before | after**

- Ref: https://www.youtube.com/watch?v=9a7klIPllXY




- Cypress hooks borrowed from Mocha framework.
- This concept is used to organize your tests in cypress.
- There are 4 types of hooks provided.
  - beforeEach
  - afterEach
  - Before
  - After
- As per above screenshot
  - We do have 3 tests: search, advance search and listing products
  - we need to login beforeEach test
  - We need to logout afterEach test
- In this case we don't require to write multiple block of code before and after tests.
- Before executing all three tests [search, advance search and listing products] if we need to do certain settings/ pre-requisite then that block of the code we can write under before.
- Before will execute only once before executing all blocks.
- After executing all tests after block will get execute at the end.

| | |
|---|---|
| **Before** | This block will execute only once before executing all the tests. |
| **beforeEach** | This block will execute multiple time before each tests. |
| **afterEach** | This block will execute multiple times after each tests. |
| **After** | This block will execute only once after executing all the tests. |

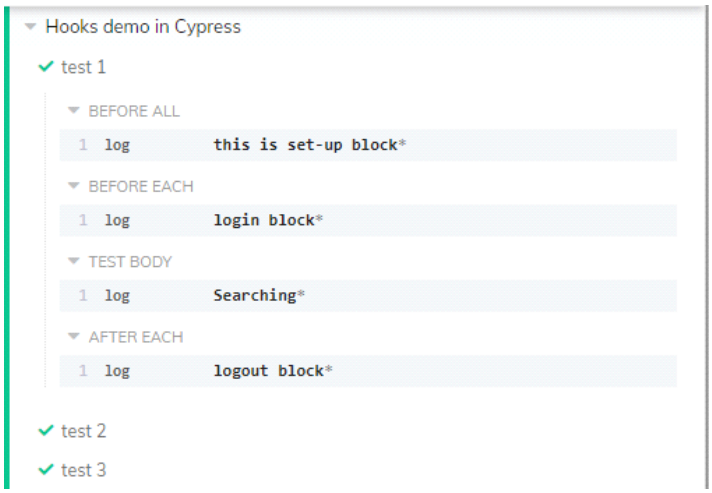
- All the hooks statement should be the part of describe block.

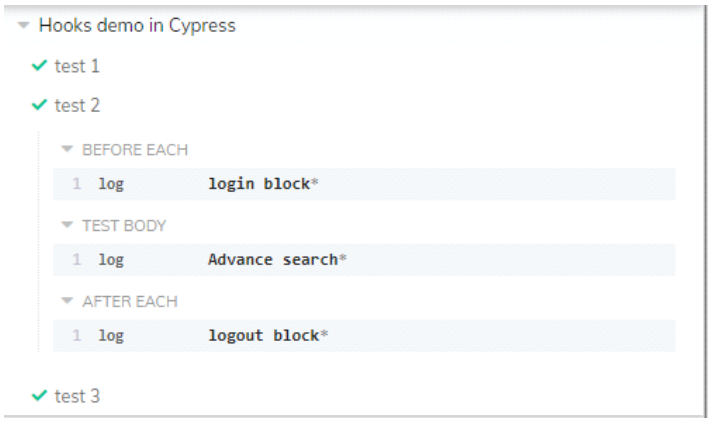
**Code:**

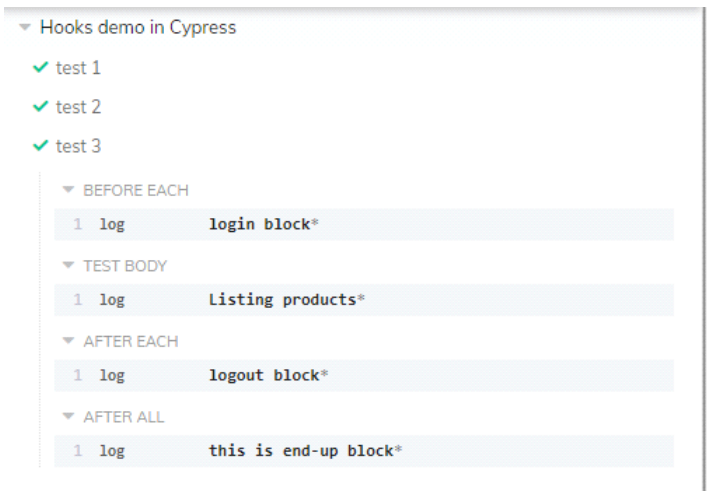
```
/// <reference types="Cypress" />
describe('Hooks demo in Cypress', () => {
    before(function() {
      cy.log('********this is set-up block*********')
    })
      after(function(){
        cy.log('********this is end-up block*********')
      })
        beforeEach(function(){
          cy.log('********login block*********')
        })
          afterEach(function(){
            cy.log('********logout block*********')
          })
        it('test 1', () => {
          cy.log('********Searching*********')
        })
      it('test 2', () => {
```

```
        cy.log('********Advance search*********')
      })
    it('test 3', () => {
      cy.log('********Listing products*********')
    })
})
```

**Result:**



# Cypress 14 - Working with Fixtures in Cypress | Data Driven Testing | Parameterization

03 February 2022 22:31

Working with Fixtures in Cypress | Data Driven Testing | Parameterization:

- Ref: https://www.youtube.com/watch?v=N8ZRqh3idgQ

# Cypress screenshots of the erros

29 January 2022 22:26