1. Cypress – Introduction

Cypress is an open-source and free test automation tool, which can be used extensively in the long run. It is mainly used for front end test automation. This tool is mainly developed to solve the issues that the teams face, while automating an application.
Cypress helps to achieve the following:

- Configure tests.
- Create tests.
- Execute tests.
- Identify errors (if any).

Selenium and Cypress are often compared in terms of their functionalities. However, Cypress is different in architecture and features. Moreover, it resolves some of the issues we face in Selenium.
Cypress is based on Javascript and executes tests within the browser. It helps to develop the tests which include:

- Unit tests.
- End to end tests.
- Integration tests.

Features

The important features of Cypress are listed below:

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

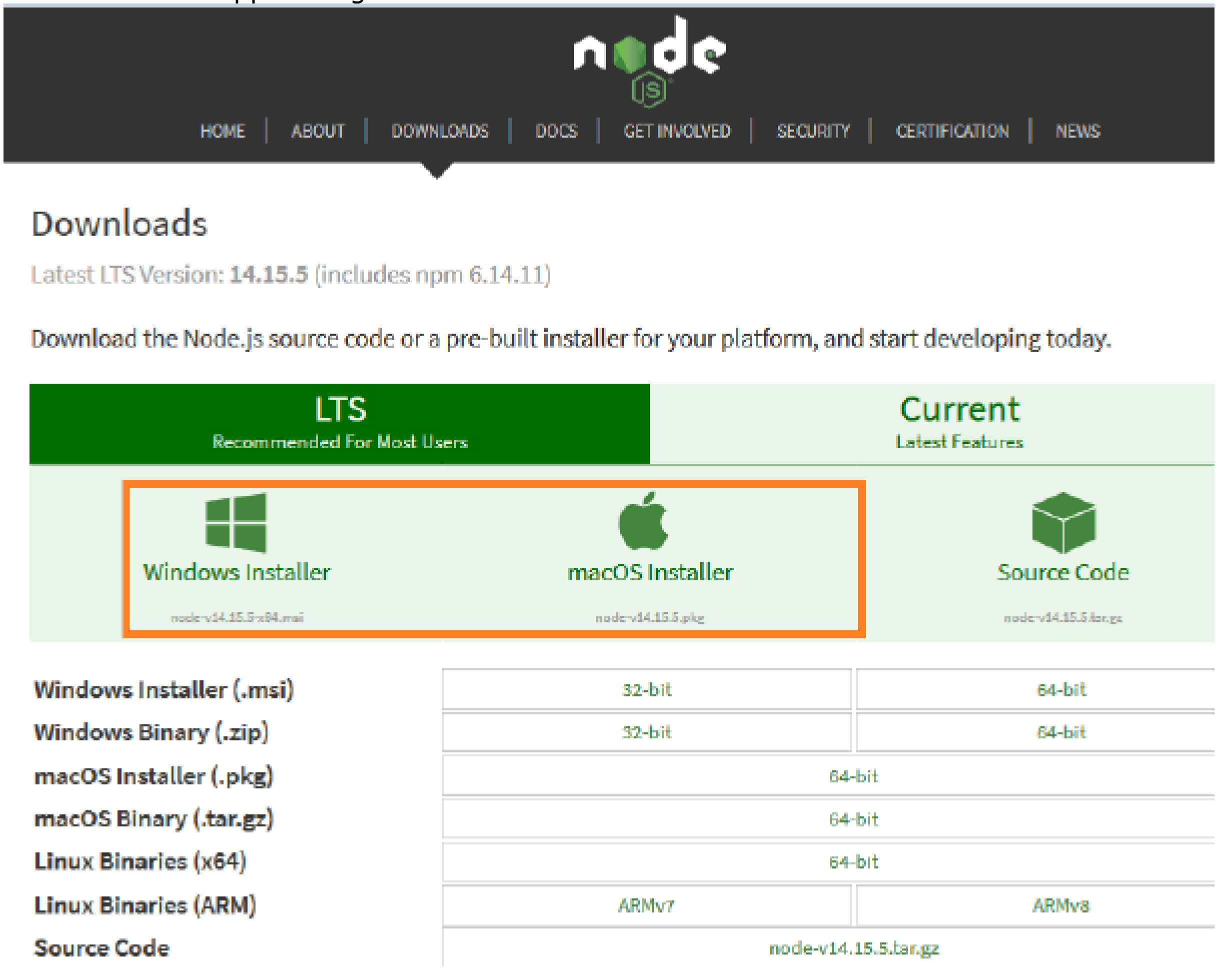
Disadvantages

There are some disadvantages of using Cypress and they are listed below:

- It is only based on JavaScript.
- A relatively new tool and hence, the community support is not extensive.
- It cannot perform mobile testing.
- Shadow Document Object Model (DOM) cannot be accessed.

Cypress Environment Setup

For Cypress environment setup, visit the link: https://nodejs.org/en/download/. The screen that will appear is given below:



HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Downloads

Latest LTS Version: **14.15.5** (includes npm 6.14.11)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features

Windows Installer	macOS Installer	Source Code
node-v14.15.5-x64.msi	node-v14.15.5.pkg	node-v14.15.5.tar.gz

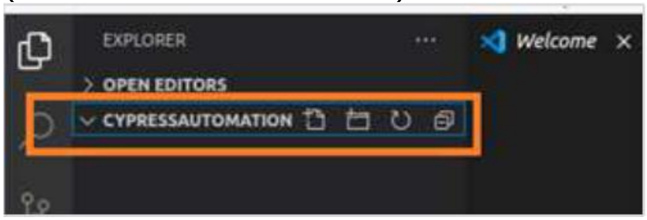
Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v14.15.5.tar.gz	

There shall be both Windows and macOS Installer. We have to get the package as per the local operating system.

For a 64- bit Windows configuration, the following pop-up comes up to save the installer. Once the installation is done, a nodejs file gets created in the Program files.

Next, we need to have a JavaScript editor to write the code for Cypress. For this, we can download Visual Studio Code from the link https://code.visualstudio.com/

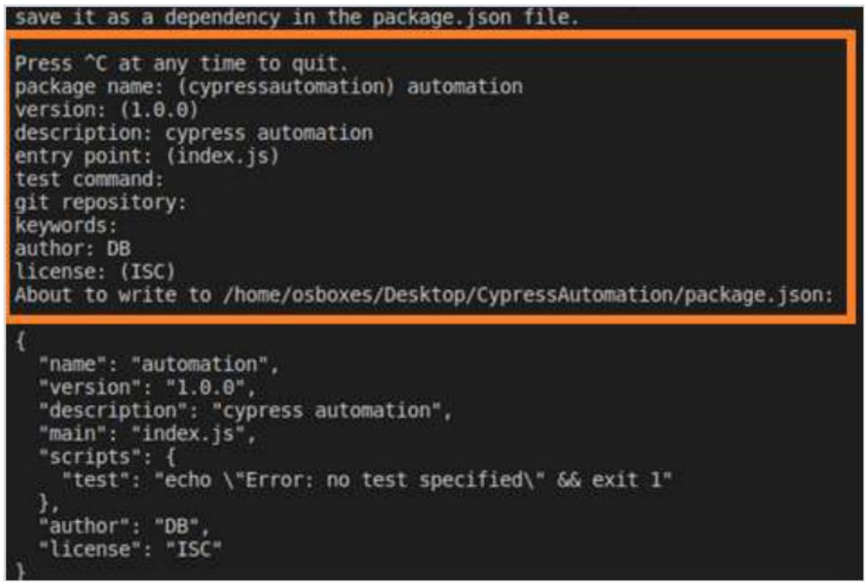
Select the option Open Folder from the File menu. Then, add the CypressAutomation folder (that we have created before) to the Visual Studio Code.



We need to create the package.json file with the below command from terminal:

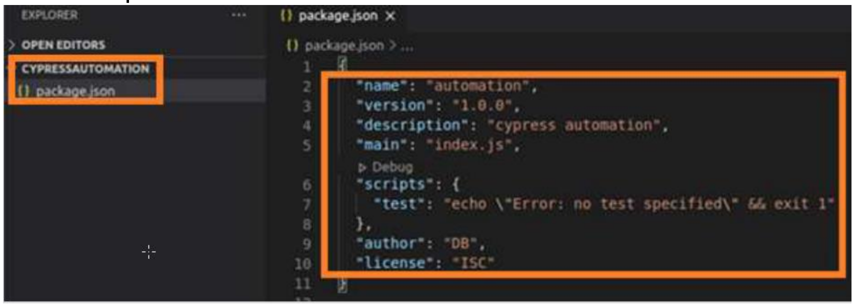
npm init

We have to enter details like the package name, description, and so on, as mentioned in the image given below:



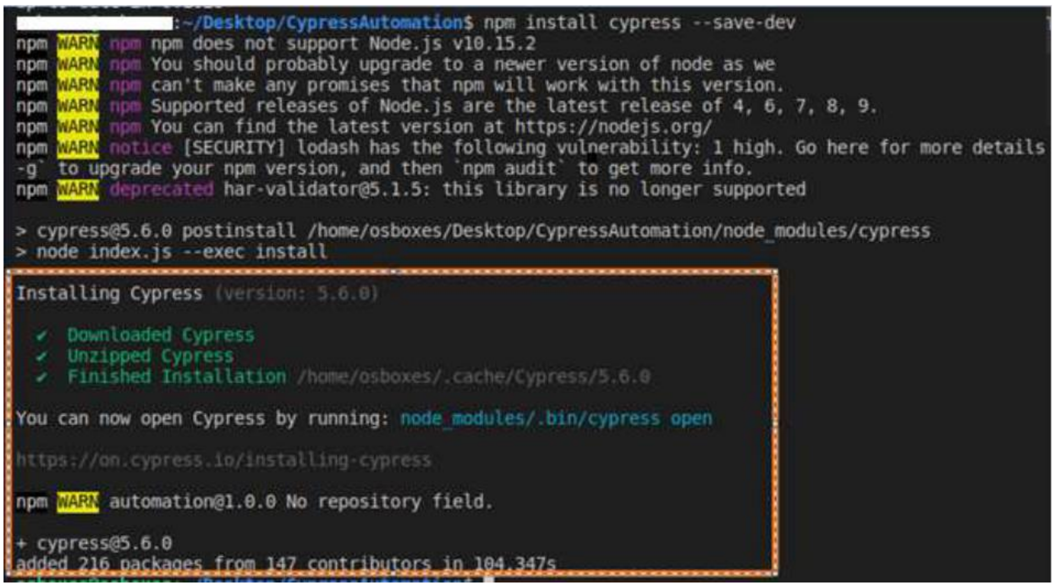
```
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (cypressautomation) automation
version: (1.0.0)
description: cypress automation
entry point: (index.js)
test command:
git repository:
keywords:
author: DB
license: (ISC)
About to write to /home/osboxes/Desktop/CypressAutomation/package.json:

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

Once done, the package.json file gets created within the project folder with the information we have provided.



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

Finally, to install Cypress run the command given below:
`npm install cypress --save-dev`
You will get the following output:



```
[redacted]:~/Desktop/CypressAutomation$ npm install cypress --save-dev
npm WARN npm npm does not support Node.js v10.15.2
npm WARN npm You should probably upgrade to a newer version of node as we
npm WARN npm can't make any promises that npm will work with this version.
npm WARN npm Supported releases of Node.js are the latest release of 4, 6, 7, 8, 9.
npm WARN npm You can find the latest version at https://nodejs.org/
npm WARN notice [SECURITY] lodash has the following vulnerability: 1 high. Go here for more details
-g` to upgrade your npm version, and then `npm audit` to get more info.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported

> cypress@5.6.0 postinstall /home/osboxes/Desktop/CypressAutomation/node_modules/cypress
> node index.js --exec install

Installing Cypress (version: 5.6.0)

  ✔  Downloaded Cypress
  ✔  Unzipped Cypress
  ✔  Finished Installation /home/osboxes/.cache/Cypress/5.6.0

You can now open Cypress by running: node_modules/.bin/cypress open

https://on.cypress.io/installing-cypress

npm WARN automation@1.0.0 No repository field.

+ cypress@5.6.0
added 216 packages from 147 contributors in 104.347s
```

Cypress Test Runner

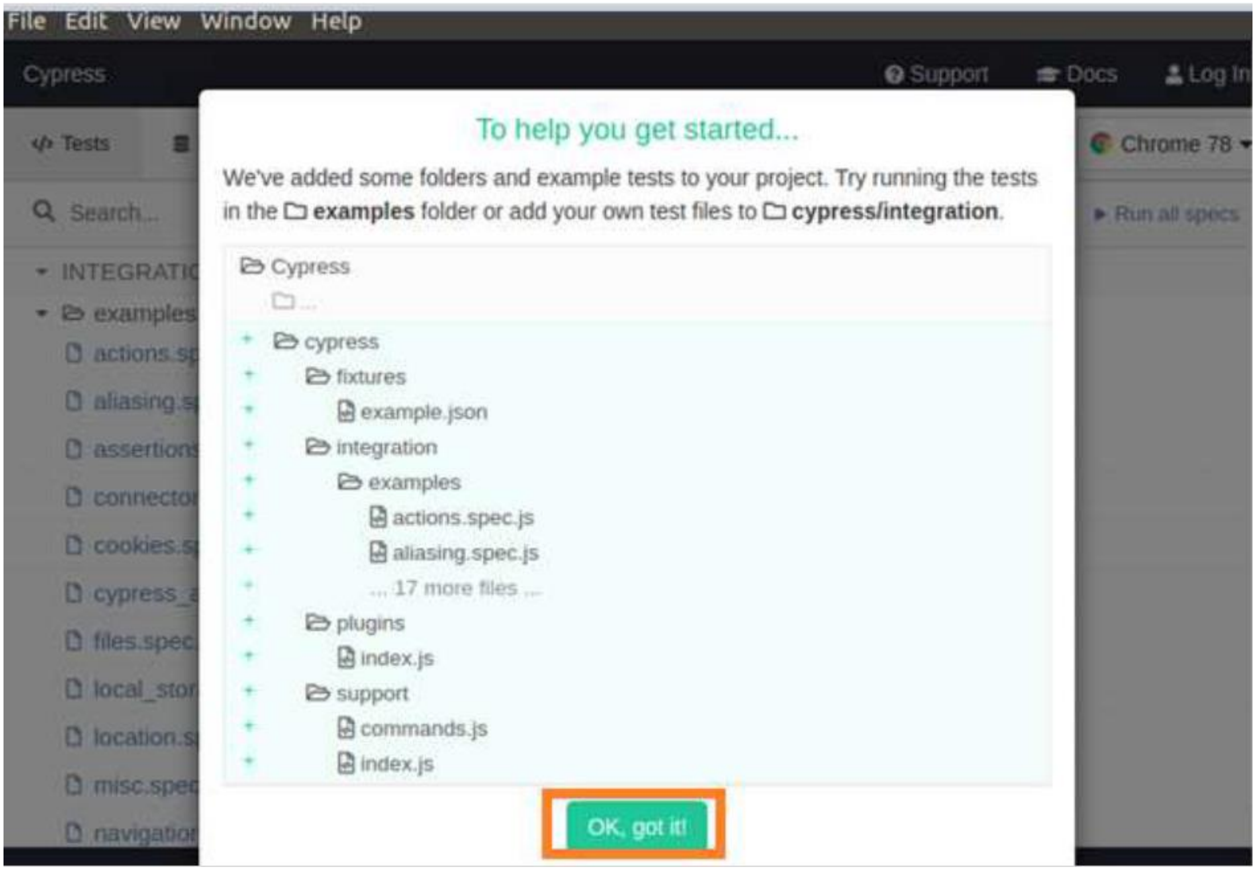
Cypress Test Runner helps to trigger the test execution. As we complete Cypress installation, there comes a suggestion from the tool on the terminal, as mentioned below:

```
You can open Cypress by running: node_modules/.bin/cypress open
```

To open the Test Runner, we have to run the below mentioned command:

```
node_modules/.bin/cypress open
```

The Test Runner window opens up after some time with the message that a sample project folder structure has been provided by Cypress under examples folder.
Click on the OK, got it! button. The screen that will appear on your computer would be as follows:



To help you get started...

We've added some folders and example tests to your project. Try running the tests in the examples folder or add your own test files to cypress/integration.

```
Cypress
  ...
+ cypress
+   fixtures
+     example.json
+   integration
+     examples
+       actions.spec.js
+       aliasing.spec.js
+       ... 17 more files ...
+   plugins
+     index.js
+   support
+     commands.js
+     index.js
```

Then the Test Runner is launched, with the more than one spec files available under the examples folder, as stated below:

To run a specific file, for example, test1.spec.js, we have to click it.

Build First Test

Once Cypress has been configured, a framework gets created within the project which is automatically visible in the Explorer. The new test file (say FirstTest.spec.js) should be created within the integration folder, as mentioned below.

Cypress Folder Structure

Let us understand the folder structure in Cypress. The factors that are included in a Cypress folder are explained below:

- **fixtures**: Test data in form of key-value pairs for the tests are maintained here.
- **integration**: Test cases for the framework are maintained here.
- **plugins:** Cypress events (prior and post events to be executed for a test) are maintained here.
- **support:** Reusable methods or customized commands, which can be utilised by test cases directly, without object creation are created here.

- **videos:** Executed test steps are recorded in the form of videos and maintained here.
- **node_modules**: Project dependencies from the npm are maintained in this folder.

It is the heart of the Cypress project execution.

- **cypress.json:** Default configurations are set in this folder. The values of the current configurations can be modified here, which overrules the default configurations.
- **package.json**: Dependencies and scripts for the projects are maintained in this folder.

Structure of a Basic Test

Cypress follows the JavaScript test frameworks (Mocha, Jasmine, and so on). To create a test in Cypress, we have to adhere to the below mentioned framework guidelines:

- Test suite name has to be provided within the describe function.
- Test case names within a test suite have to be provided within the same or you have to specify the function.
- Test steps within a test case have to be implemented inside the it/specify block.

Basic Test Implementation

The basic test implementation can be done by using the following command:

```
// test suite name
describe('Tutorialspoint Test', function () {
// Test case
it('Scenario 1', function (){
// test step for URL launching
cy.visit("https://www.google.com/");
});
});
```

The cy command used above does not require an object invocation. It becomes available by default on installing the node modules.

Test Execution

For execution **from the command line**, run the command given below:

```
./node_modules/.bin/cypress run
```

Here, all the files within the integration folder get triggered.

For execution **from the Test Runner**, run the command stated below:

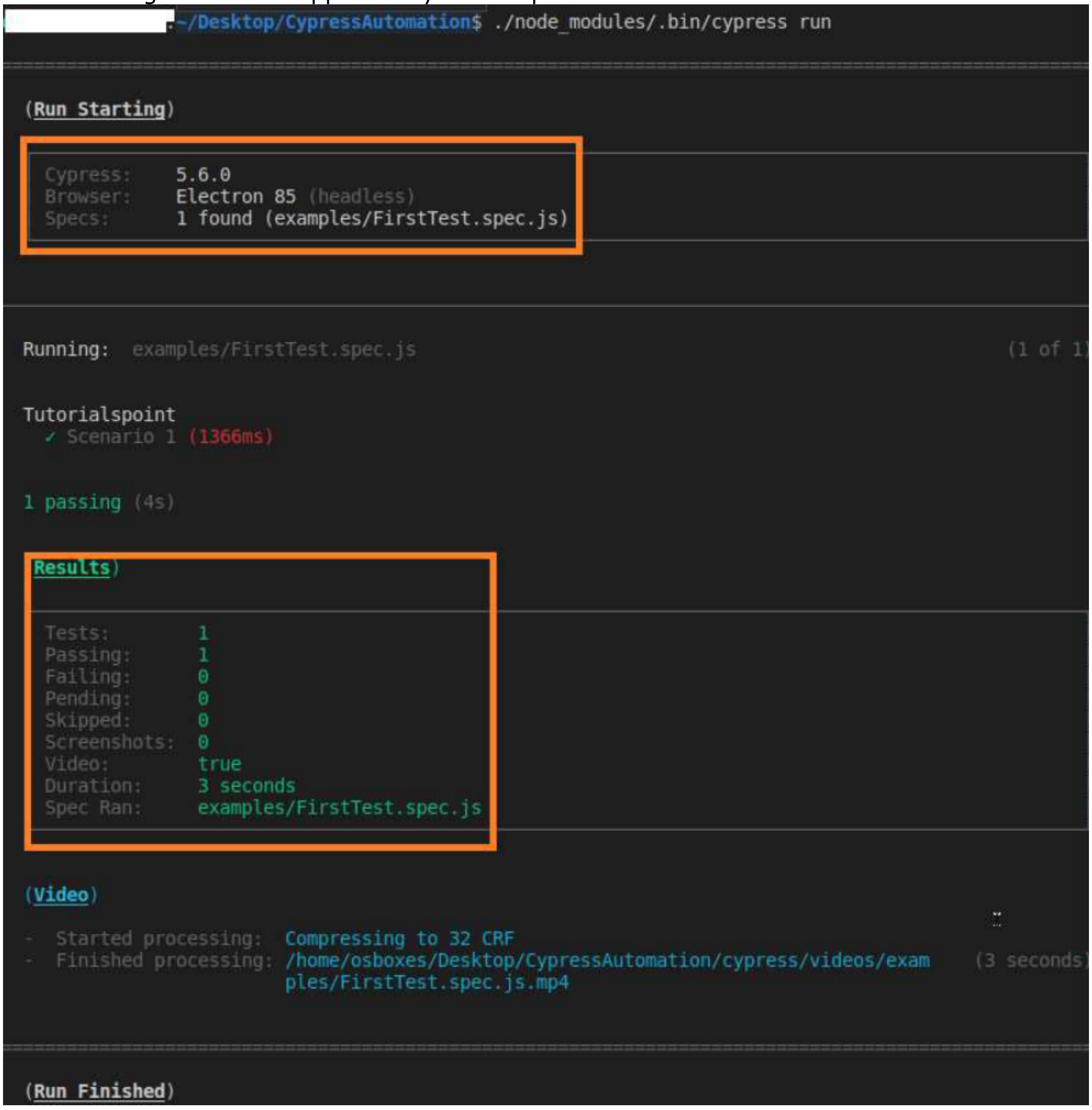
```
./node_modules/.bin/cypress open
```

Then, click on the spec file that we want to trigger for execution.

To trigger execution **for a specific file from command line**, run the command mentioned below:

```
cypress run --spec "<spec file path>"
```

The following screen will appear on your computer:

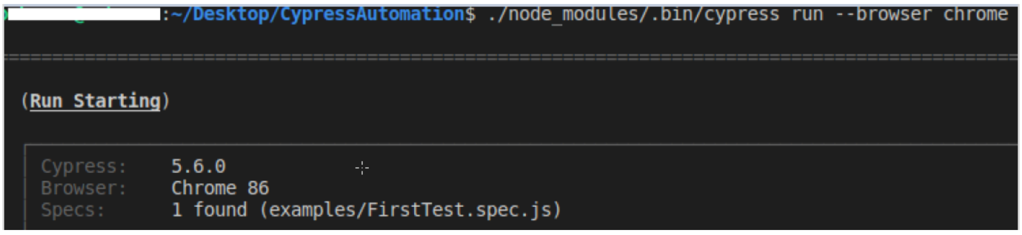


```
[redacted]:~/Desktop/CypressAutomation$ ./node_modules/.bin/cypress run

====================================================================================

  (Run Starting)

  ┌──────────────────────────────────────────────────────────────────────────────┐
  │ Cypress:    5.6.0                                                            │
  │ Browser:    Electron 85 (headless)                                           │
  │ Specs:      1 found (examples/FirstTest.spec.js)                             │
  └──────────────────────────────────────────────────────────────────────────────┘


────────────────────────────────────────────────────────────────────────────────────

  Running:  examples/FirstTest.spec.js                                      (1 of 1)


  Tutorialspoint
    ✓ Scenario 1 (1366ms)


  1 passing (4s)


  (Results)

  ┌──────────────────────────────────────────────────────────────────────────────┐
  │ Tests:        1                                                              │
  │ Passing:      1                                                              │
  │ Failing:      0                                                              │
  │ Pending:      0                                                              │
  │ Skipped:      0                                                              │
  │ Screenshots:  0                                                              │
  │ Video:        true                                                           │
  │ Duration:     3 seconds                                                      │
  │ Spec Ran:     examples/FirstTest.spec.js                                     │
  └──────────────────────────────────────────────────────────────────────────────┘


  (Video)

  -  Started processing:  Compressing to 32 CRF
  -  Finished processing: /home/osboxes/Desktop/CypressAutomation/cypress/videos/exam    (3 seconds)
                          ples/FirstTest.spec.js.mp4


====================================================================================

  (Run Finished)
```

Cypress can run tests in browsers like Chrome, Electron, and Firefox. In the Test Runner, we have the option to choose the browser from the right upper corner.
Also, it must be noted that if a browser option is not available, it means we do not have the latest version of that browser in our system.

Execution from Other Browsers

The execution from other browsers from Command Line is explained below:
To run the **execution in Chrome**, you need to run the below mentioned command:
./node_modules/.bin/cypress run -- browser chrome
You can see the following screen:

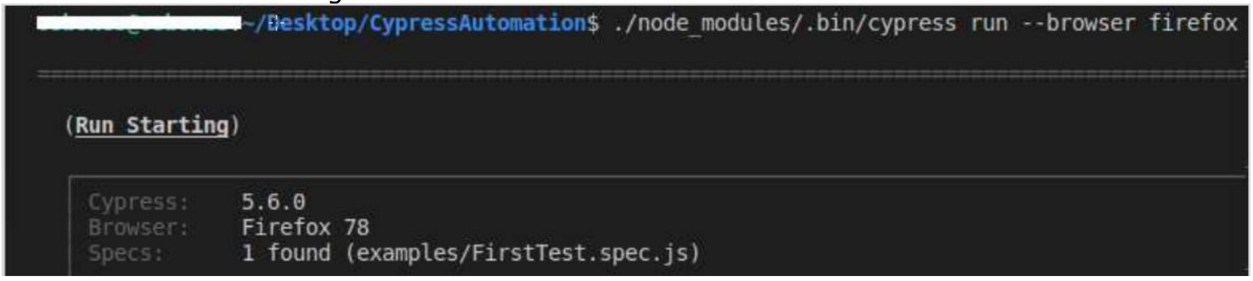


```
[redacted]:~/Desktop/CypressAutomation$ ./node_modules/.bin/cypress run --browser chrome

====================================================================================================

  (Run Starting)

  ┌──────────────────────────────────────────────────────────────────────
  │ Cypress:    5.6.0
  │ Browser:    Chrome 86
  │ Specs:      1 found (examples/FirstTest.spec.js)
```

To run the **execution in Firefox**, run the command given below:

./node_modules/.bin/cypress run -- browser firefox

You can see the following screen:



```
[redacted]~/Desktop/CypressAutomation$ ./node_modules/.bin/cypress run --browser firefox

====================================================================================================

  (Run Starting)

  ┌──────────────────────────────────────────────────────────────────────
  │ Cypress:    5.6.0
  │ Browser:    Firefox 78
  │ Specs:      1 found (examples/FirstTest.spec.js)
```

To run the **execution in headed mode**, run the command given below:

`./node_modules/.bin/cypress run -- headed`

From the command line, Cypress executes tests in headless mode, if no option is specified.