**What is Cypress?**

The [Cypress framework](https://www.browserstack.com/guide/cypress-framework-tutorial) is a NodeJS-based modern automation tool that supports JavaScript/Typescript as the programing language. There are many [advantages and disadvantages](https://www.browserstack.com/guide/cypress-vs-selenium) of Cypress. However, it is the most popular and easy-to-use tool in the automation world, quickly becomes a favorite of devs and testers despite its recency.

**What is Cypress Test Automation?**

Cypress is a NodeJS-based test automation framework for the modern web. It uses JavaScript as a programming language.

Unlike other tools, the Cypress automation tool can be used for a variety of tests – unit tests, integration tests, [end to end tests](https://www.browserstack.com/guide/end-to-end-testing), API tests, etc. Cypress also comes with multiple in-built functionalities to make developers’ and QA’s jobs easier. Some of these include screen capture, video recording, time travel, easy debugging, etc. Currently, Cypress also supports [cross browser testing](https://www.browserstack.com/cross-browser-testing) on Edge, Firefox, and Chrome.

[Cypress framework](https://www.browserstack.com/guide/cypress-framework-tutorial) uses Mocha and Chai assertions by default, so testers can use assertions from these libraries. The reporting feature is one of the most used features in the automation world. Cypress uses [Mocha reporter](https://mochajs.org/) internally, so testers can configure Mocha reporter or Spec reporters in their specs.

**The Cypress Test Automation CLI**

Cypress comes with Command line Interface aka Cypress CLI which helps users integrate Cypress tests with CI/CD tools such as Jenkins, Azure DevOps, etc. In particular, it also helps testers execute Cypress tests on BrowserStack**.**

**How to set up Cypress for Test Automation**

Bear in mind that all [Cypress testing](https://www.browserstack.com/automate/cypress) must be executed on real browsers for accurate results. Start running tests on 30+ versions of the latest browsers across Windows and macOS with BrowserStack. Use instant, hassle-free parallelization to get faster results without compromising on accuracy. Detect bugs before users do by testing software in [real user conditions](https://www.browserstack.com/real-user-conditions-testing-on-browserstack) with BrowserStack.

[Run Cypress Tests on Real Browsers](https://www.browserstack.com/users/sign_up)

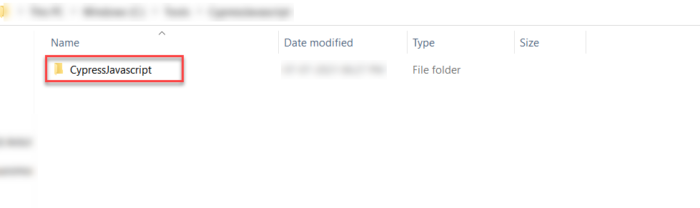
Let’s discuss, step by step, how to perform automation with Cypress. In order to start, users will need the following prerequisites:

1. [Download and Install NodeJS](https://nodejs.org/en/download/)
2. [Download and install Visual Studio Code](https://code.visualstudio.com/download)

Note: Visual Studio Code is the most used IDE. However, testers can use any IDE.

**Step 1: Create Empty Project Folder**

Navigate to the desired location, and create an empty folder inside. In this example,  **CypressJavaScript** is the project folder.



**Step 2: Open the folder in Visual Studio Code or Any IDE**

Go to **Visual Studio Code Menu** > Navigate to **File** > Click on **Open Folder** > Choose the newly created Folder(CypressJavascript) from **Open Folder Wizard**

**Step 3: Create package.json**

The **package.json** helps track all the packages installed for the Cypress automation framework, and it also helps to create shortcut commands to run the tests.

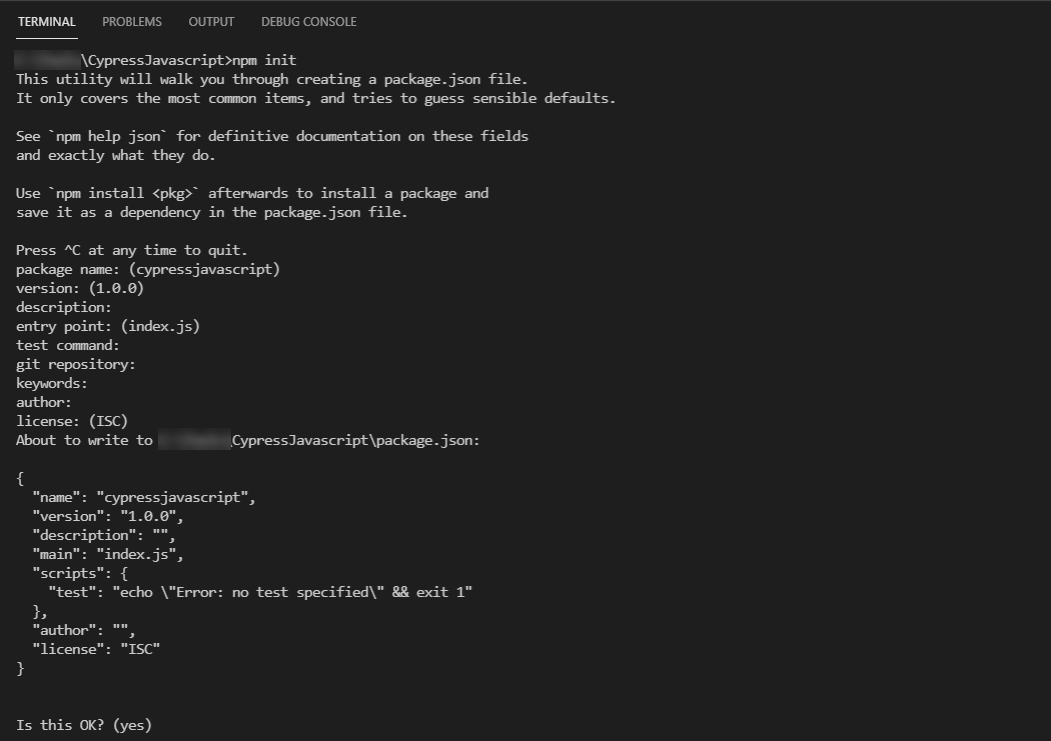
In order to create the **package.json**, open **Terminal** in Visual Studio Code Menu and type the command below:

npm init

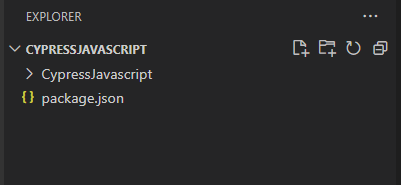
On entering **npm init** in the terminal, it asks for a set of questions. Answer them or hit [Enter] [Enter] until it finishes.

Finally, it asks – Is this OK? (yes). Then, hit [Enter].

Now the file named **package.json** is automatically created in the root folder of the project.



The**package.json** in the root folder looks like this:



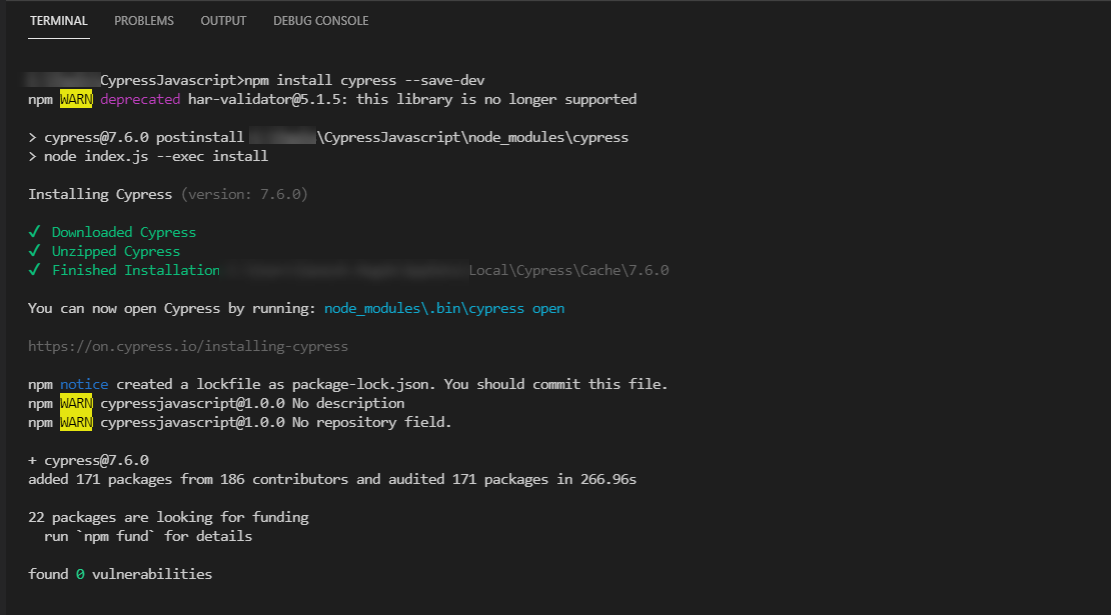
**Step 4: Install Cypress**

Cypress is a NodeJS based automation tool, available as an npm package. Cypress can be also downloaded as an installer, but the recommended way is to install from npm.

In the root **Project Folder** (CypressJavascript) > **Terminal** >**type**

npm install cypress --save-dev

***Note****:*[*Cypress installation*](https://www.browserstack.com/guide/cypress-installation-for-test-automation)*takes some time to finish, and its speed depends on the user’s internet speed. However, this happens only the first time the tester installs Cypress.*



**Step 5: Open Cypress Window**

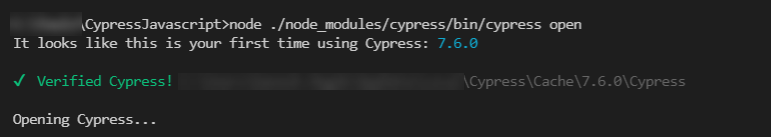
Once Cypress packages have been installed, Cypress, by default, configures some folders. Typically 4 folders will be created inside the Cypress folder namely plugins, support, integration, fixtures – the first time when the tester opens Cypress. In order to open the Cypress window, use one of the following commands:

node ./node\_modules/cypress/bin/cypress open

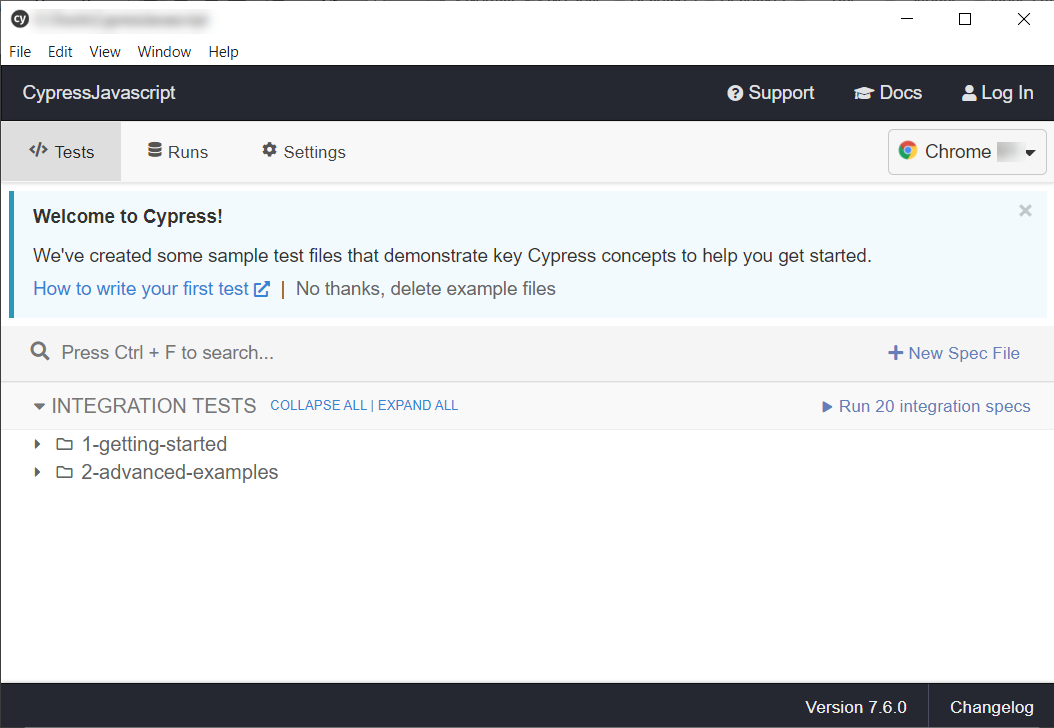
**Or**

npx cypress open

On entering the above command, installation begins:



Upon successful completion of this command, the Cypress window launches, as seen below:



As mentioned earlier, this command also prepares the framework in the background.

It creates a Cypress folder in the project directory, and the Cypress folder will have subfolders like integration, fixtures, plugins, and support.

Now the Project Folder looks like below:

