**What is Cypress?**

Cypress is a next generation front end Automation testing tool built for the modern web applications (Like: Which is developed using Angular or React)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**How Cypress is Unique from other Automation tools?**

Cypress [automatically waits](https://docs.cypress.io/guides/core-concepts/introduction-to-cypress.html#Cypress-is-Not-Like-jQuery) for commands and assertions before moving on (default timeout is 4 sec). No more async hell.

Ability to test Edge Testcases by Mocking the server response

Cypress takes snapshots as your tests run. We can hover over each commands in the [Command Log](https://docs.cypress.io/guides/core-concepts/test-runner.html#Command-Log) to see exactly what happened at each step (means we can time travel to each and every steps, So debugging is awesome and if will get any error, cypress will provide proper details in reports).

Because of its Architectural design, Cypress delivers fast, consistent and reliable test execution compared to other Automation tools (All the commands run in browser itself, so it’s very fast and consistence)

View videos of your entire tests execution when run from the Cypress Dashboard.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Advantage and Disadvantages**

* Cypress supports just one language (JavaScript), one framework (Mocha), and multiple browser (Chrome/ Electron/Firefox/Edge).
* Cypress is used for UI testing, it uses its own unique DOM manipulation and runs directly in the browser with no network communication.
* Free and open source and used for Web browser automation.
* Takes Dom snapshot for each and every command
* Time travel facility in report and very good debugging feature.
* Different domain switching is not allowed while execution due to security issue (it’s architecture is define in such a way and because of this switching to Window/Tab is not handled directly so we need to remove target attribute in DOM and this is intentionally done by Cypress to remove flaky results)
* Faster, easier and more reliable tests.
* There is a rich UI which visually shows you the command execution, assertions, network requests, spies, stubs, page loads, or URL changes.
* Cypress takes snapshots of your application and enables you to time travel back to the state it was in when commands ran.
* Cypress executes the vast majority of its commands inside the browser, so there is no network lag. Commands execute and drive your application as fast as it is capable of rendering.
* Cypress automatically waits for commands and assertions.
* you can pause your test case in middle by simple command.
* you can click hidden elements easily.
* Cypress built on node.js and comes as a npm module and bundled jquery.
* in Cypress browser itself is execute you code.
* It is used to avoid flaky results. and you can mock network request and responses.

**Important Commands**:

"**npm init"** - automatically generates package.json

"**npm install cypress --save-dev"** - it will download and install cypress as a dev dependencies

"**node\_modules\.bin\cypress open"** - open cypress, because all exe file is present in bin folder

Through command line cypress always execute in headless mode and commands are:

**"node\_modules\.bin\cypress run**”: run the suite

**"node\_modules\.bin\cypress run --headed”:** run in headless mode

"**node\_modules\.bin\cypress run --browser chrome**”: run in chrome browser

Cypress built on Node.js and comes packaged as an npm module,

As it is built on Node.js, It uses JavaScript for writing tests. But 90% of coding can be done using Cypress inbuilt commands which are easy to understand (Cypress is asynchronous in nature because JS is asynchronous in nature. Cypress by default resolve promises (they have created wrapper to resolved) but if we will use JS or Jquery function then it's our responsibility to remove/resolve the promises using then() function.

Cypress also [bundles with jQuery](https://docs.cypress.io/guides/references/bundled-tools.html#Other-Library-Utilities) and inherits many of jQuery methods for UI components Identification (but when will use JQuery function it’s our responsibility to remove promises)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Cypress Architecture**

Most testing tools (like Selenium) operate by running outside of the browser and executing remote commands across the network. But Cypress engine directly operates inside the browser. In other words, it is the browser that is executing your test code.

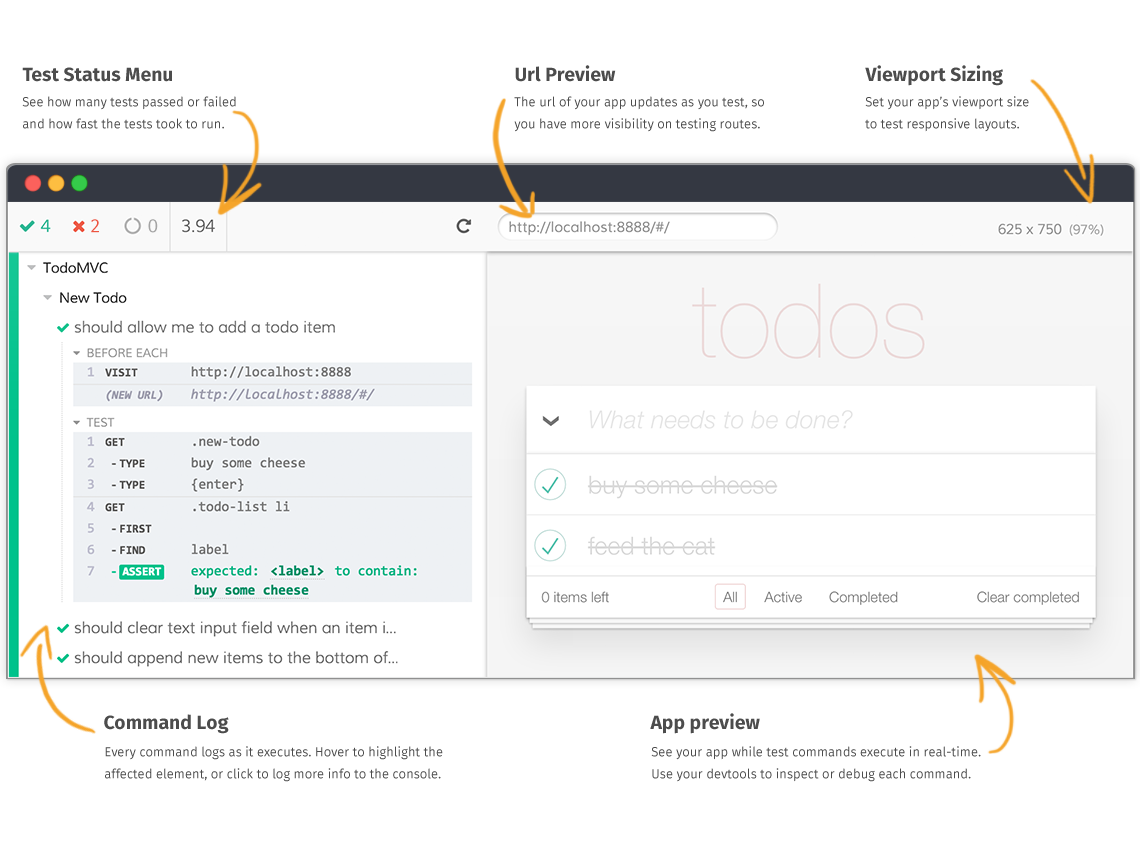
This enables Cypress to listen and modify the browser behavior at run time by manipulating DOM and altering Network requests and responses on the fly

Cypress open doors to New Kind of testing with Having ultimate control over your application (front and back

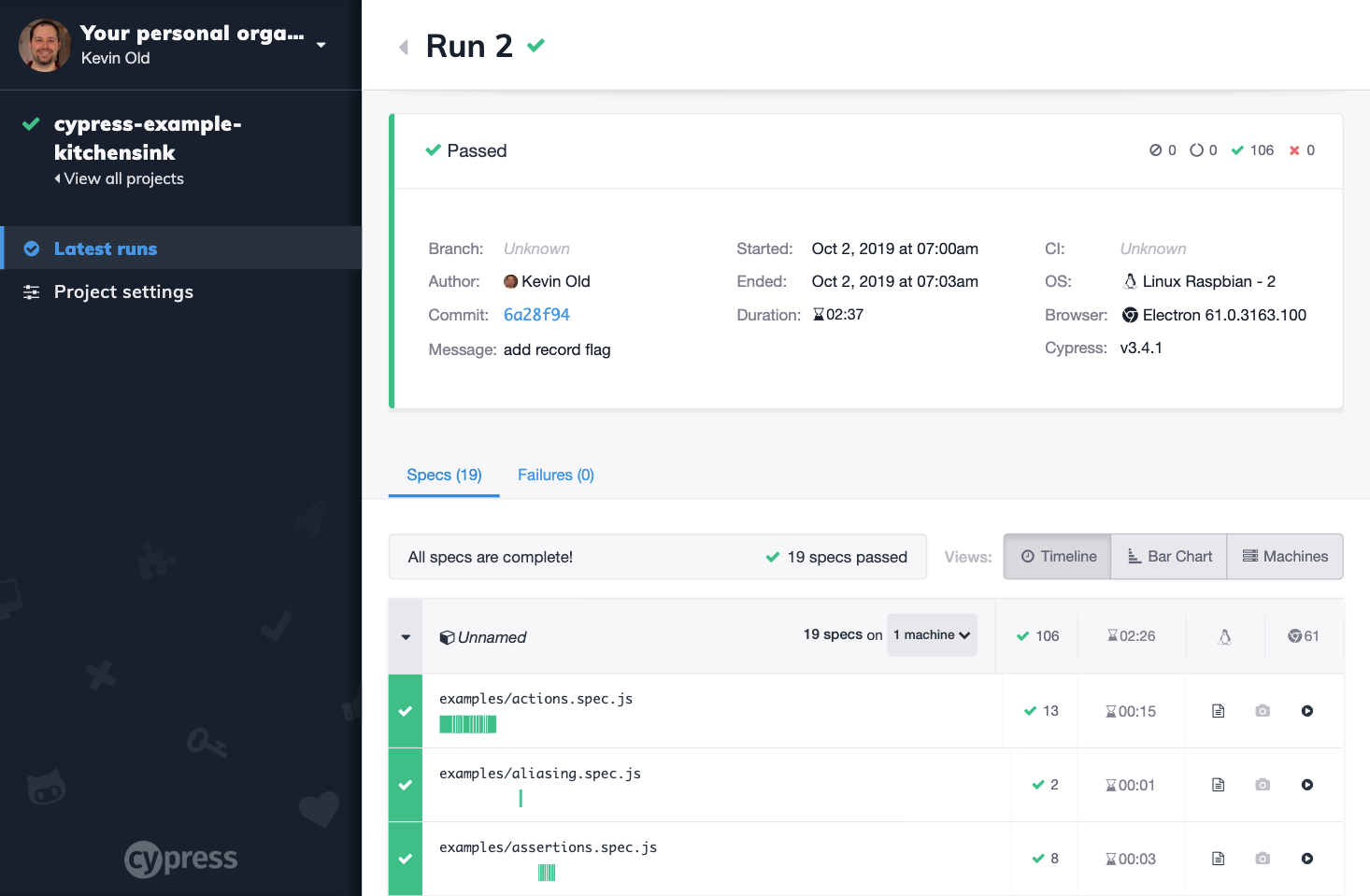
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Cypress Components:

Test Runner (our actual test will execute here and you can see all the logs, debugging, time travel feature , we can select browser also we can see all the configuration is provided by cypress, we can see cypress dashboard etc. we can run single or suite of tests.)



Dashboard Service (video recording, screenshots on failure, passes test cases, failed test cases)



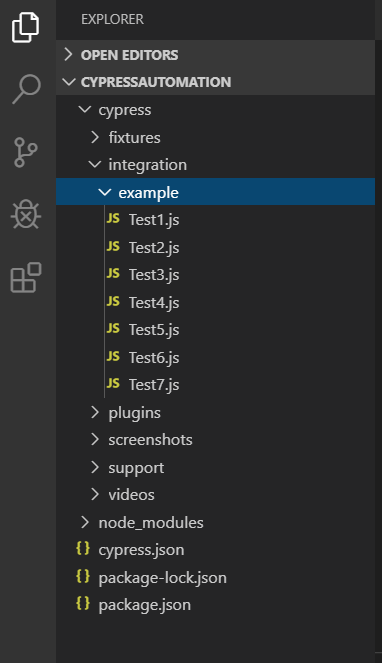
We can also generate reports using mocha and mochawesome report dependency(this is optional)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package.json** : is just like pom.xml, will defined all the dependencies and all.



Folder Structure:



**fixtures**: folder is used for Test data (Json or Excel)

**integration**: folder is used to write test scripts and page objects

**plugins**: folder is used to write listener or cypress events

**support**: folder is used to write reusable libraries

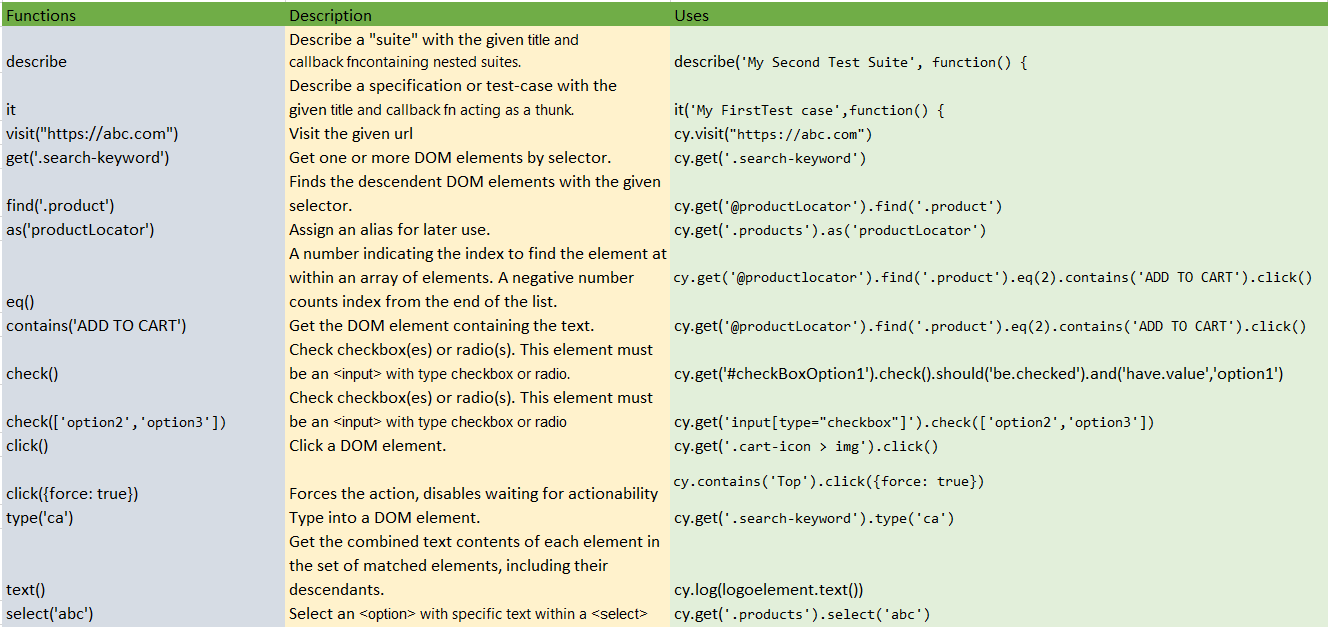
**videos**: folder is used to store videos

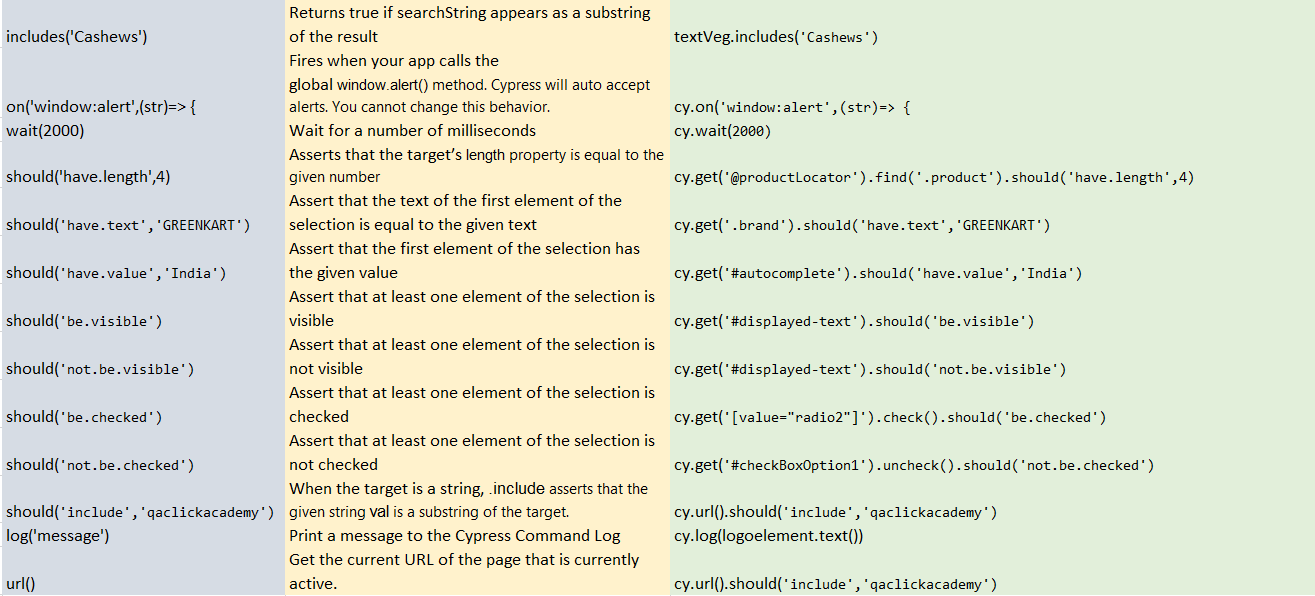
**node\_modules**: when you install cypress, this folder is automatically generates.

**cypress.json**: This file is used to provide configuration, cypress also provide defaults configuration and if we want to change some configuration value, we need to provide those values into this file and cypress will override this values on existing.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Some Important functions and their uses:**





Navigation : cy.visit("https://abc.com")

find('product') : function is used to get descendent elements in DOM.

cy.get('products').find('product').should('have.length',4) : This is called parent child chaining.

eq(2) : Get A DOM element at a specific index in an array of elements.

contains('Add to cart') : Get the DOM element containing the text.

cy.get('products').find('product').eq(2).contains('Add to cart').click()

each(el,index,list) : Iterate through an array like structure (arrays or objects with a length property).

cy.get('.products').find('.product').each((el,index,list) =>

{

const veg= el.find('h4.product-name').text()

if(veg.includes('Cashews'))

{

el.find('button').click()

}

})

JS is asynchronous in nature, means all the steps hit a server at a time, so we need to remove promises. cypress is also asynchronous in nature but they have created wrappers on top of it so steps will execute in sequence but when will use JS functions then it is our responsibility to remove promises. Promises comes in 3 states- resolved/rejected/pending. Whenever will store any step into variable then we need to resolve promises.

text() - jquery function

// const logo=cy.get('.brand')

// cy.log(cy.get('.brand').text())

// cy.log(logo.text())

cy.get('.brand').then(function(logoelement)

{

cy.log(logoelement.text())

})

if we are using same locator again and again, we can alias it :

cy.get('.products').as('productlocator')

cy.get('@productlocator').find('.product').should('have.length',4)

console.log("message"): will print in developer console, but this is not cypress method this is JS method so we need to resolve promise means we have to use then() function in previous step.

cy.log("message"): will print in test runner report.

cypress auto accept Alert but if you want to fetch the text present in the Alert, cypress have a capability by triggering/firing browser events window: alert.

diff b/w behavior (like: checked, visible, disable) and property (like - type, target, id, classname)

