Cypress Automation Training

Plural CODEMASH

[Cypress: About 2](#_Toc90971644)

[Cypress: Installation Steps 2](#_Toc90971645)

[Cypress: Run Command 3](#_Toc90971646)

[Cypress: Folder Structure 3](#_Toc90971647)

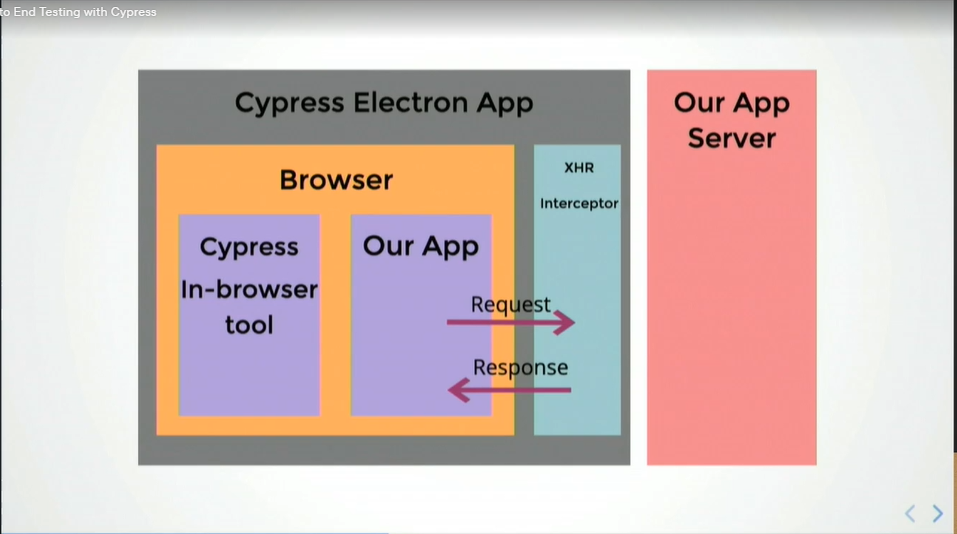
[Cypress: Put in Action [Get Started] 4](#_Toc90971648)

[Application Scenario 4](#_Toc90971649)

[Cypress: Test writing scenarios (Lists) 6](#_Toc90971650)

[Cypress: Steps for test writing (cypress code in action…) 7](#_Toc90971651)

# Cypress: About



# Cypress: Installation Steps

npm install cypress save-dev

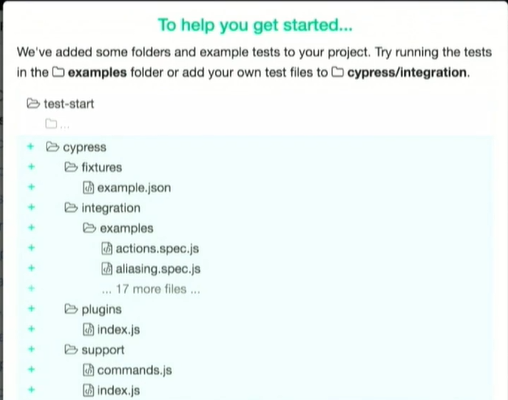
* It will generate cypress.json file
  + All cypress settings
* **cypress / integration** 
  + Contains all the test files named…. myFile.specs.ts
  + Putting all e2e (end to end tests)
  + We can organize accordingly by the creating new files, sub-folders etc…
  + All the tests should be integrated inside only “cypress >> integration” folder
* **cypress/fixtures**
  + use for mock data for certain scenarios in “cypress/fixtures”
* **cypress/plugins**
  + extending cypress functionality
  + use for writing certain tasks task to automate the things
  + can put all that code and keep it organized in the plugin folders, so its can extend Cypress’s features
* **cypress/support**
  + use for shared commands, utilities
  + we have a lot of common code where we can creating/removing/manipulating lots of dat. So, it’s a repeated code
  + we can abstract all that out and then put it in a util folder, which Cypress can also support

# Cypress: Run Command

npx cypress open

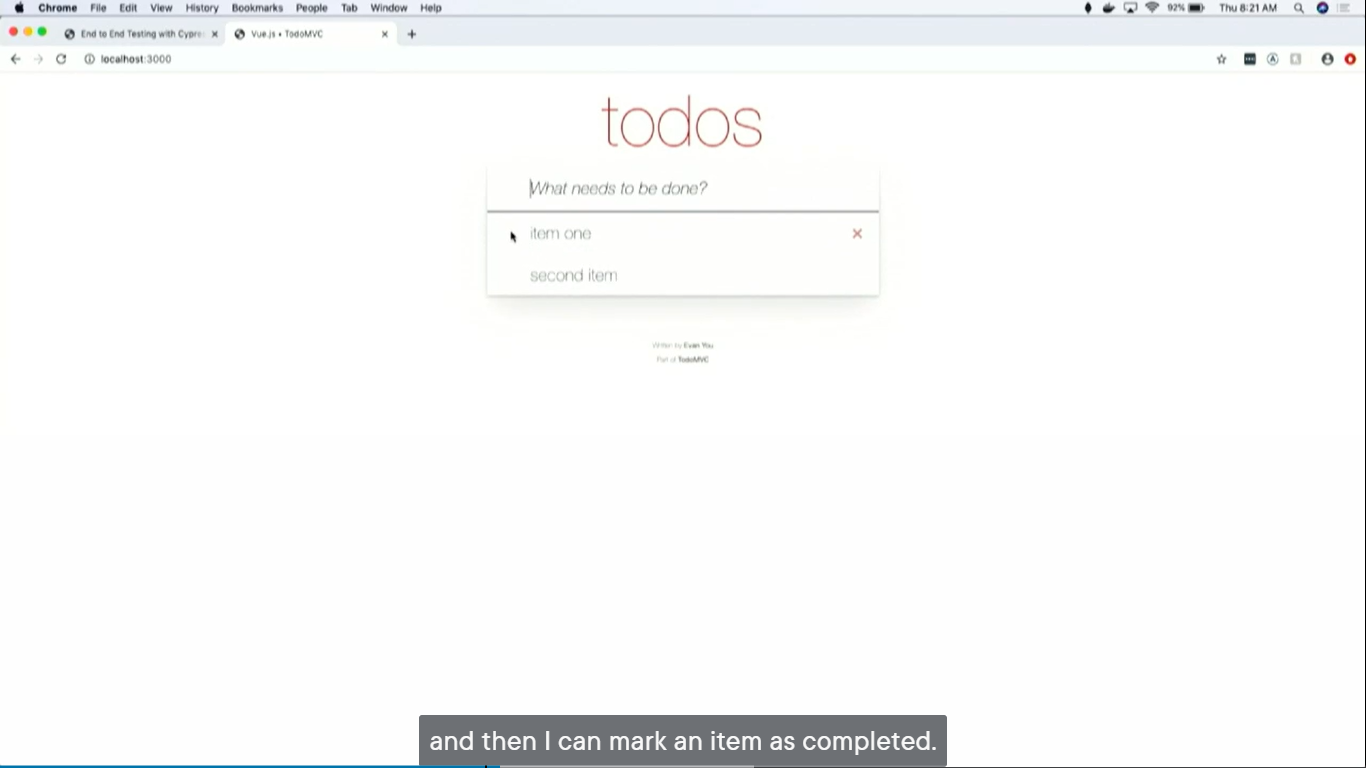
npx cypress run

# Cypress: Folder Structure

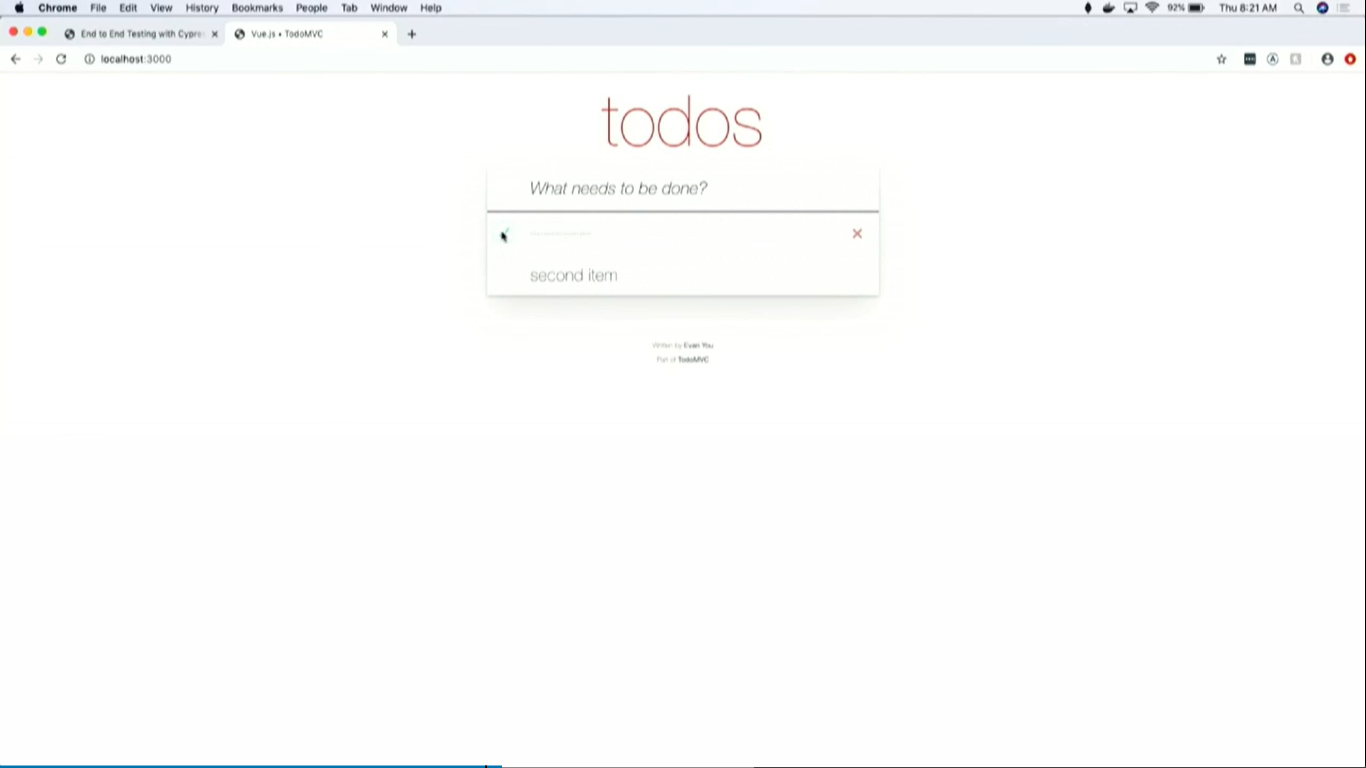


# Cypress: Put in Action [Get Started]

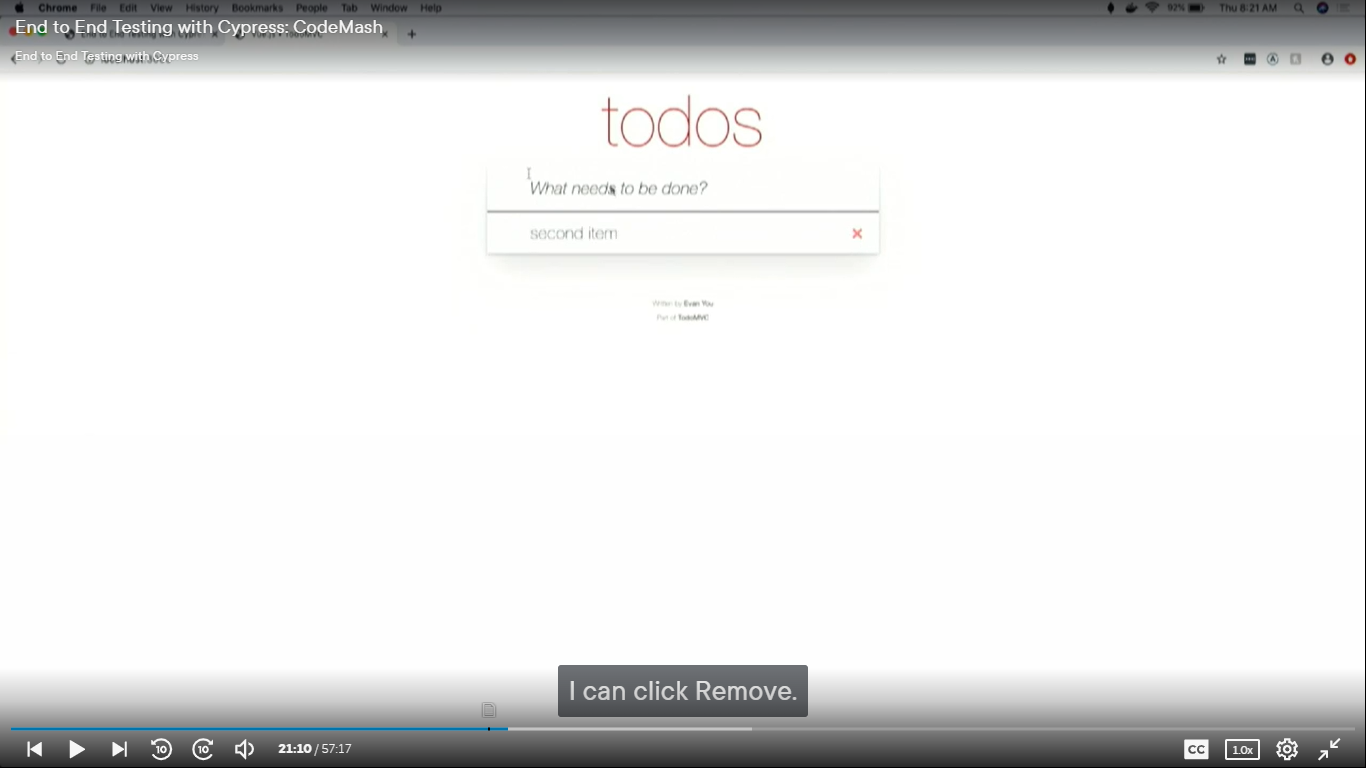
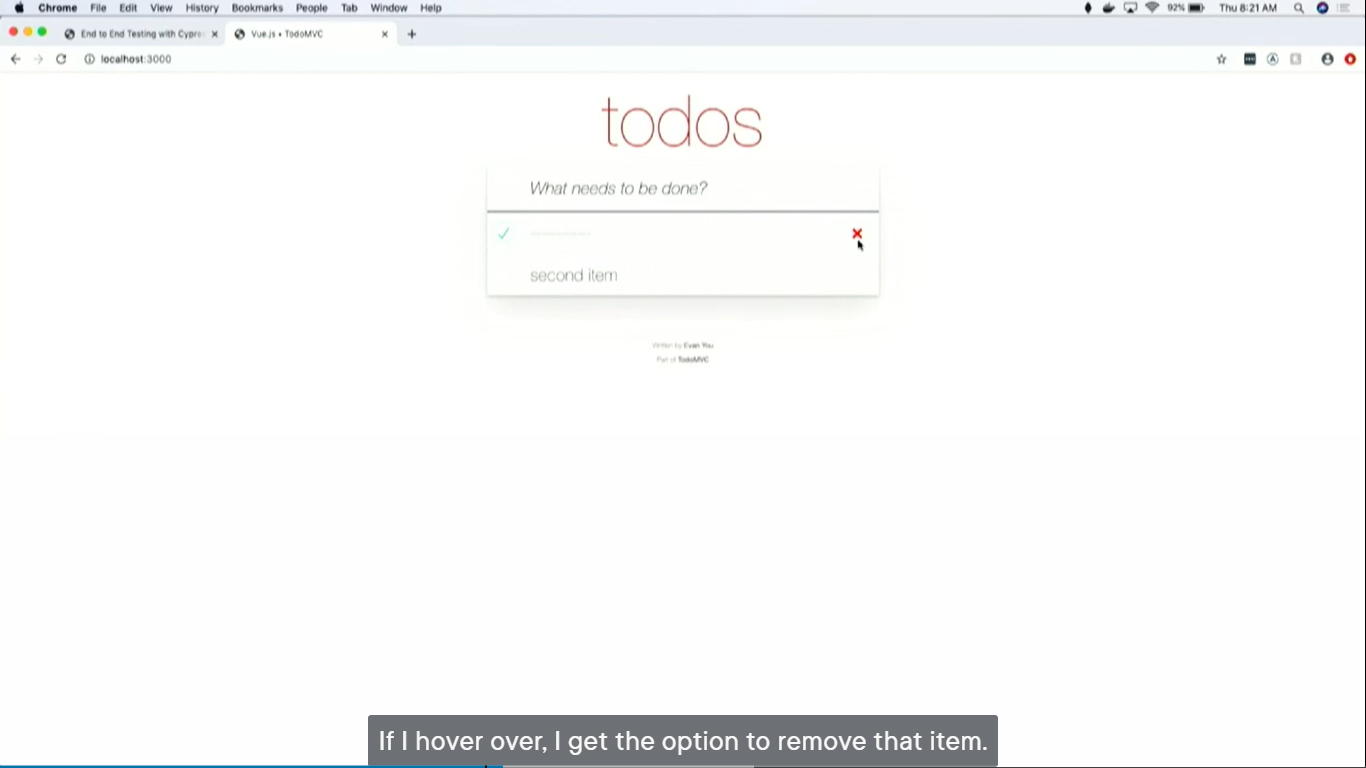
## Application Scenario



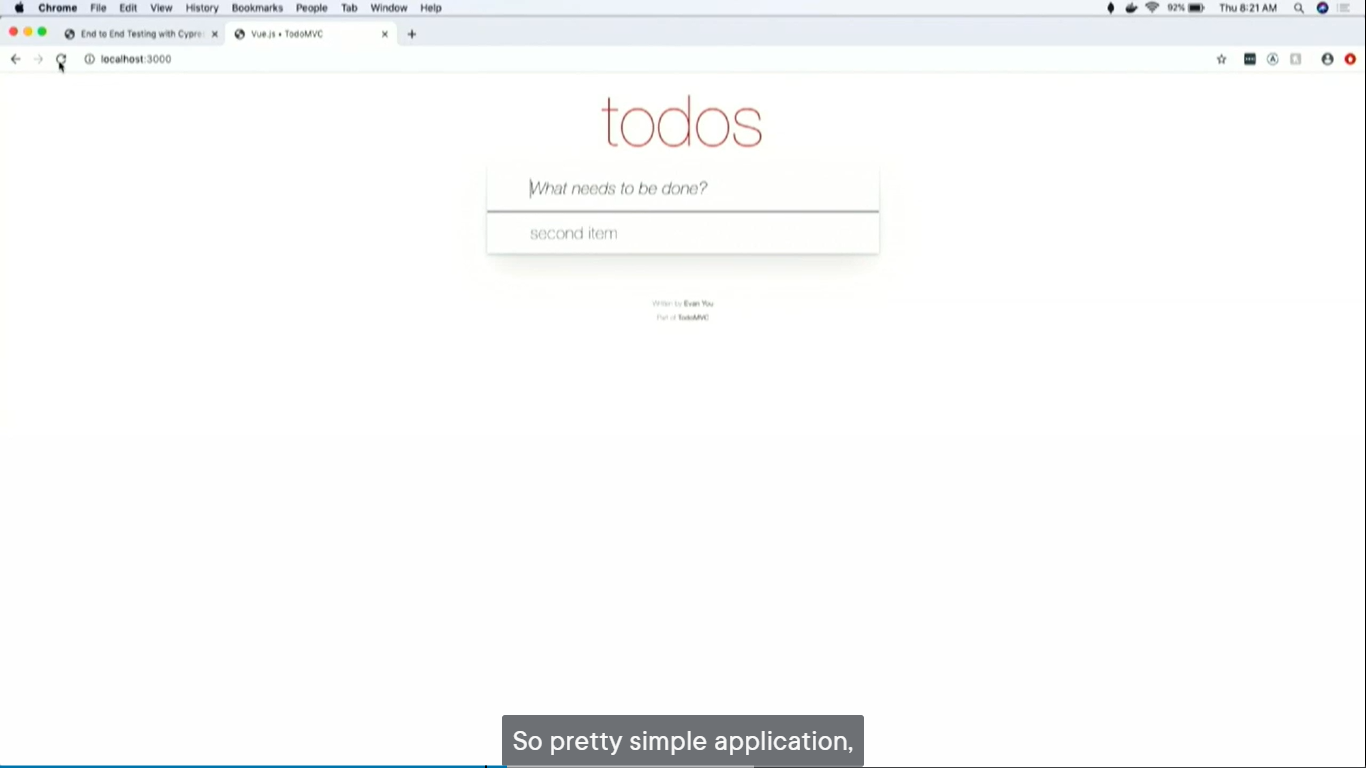
* can add items by write something in input text



* can mark an as item has completed



* on hoven there is option to remove that marked item
* can click to remove that item



* when refresh, can get the item back

## Cypress: Test writing scenarios (Lists)

1. able to… add items
2. able to… mark things as a completed
3. able to… delete items
4. able to test… if certain requests are made
5. able to test… if loading elements are displayed
6. able to test… can I reset my state before each test runs?

## Cypress: Steps for test writing (cypress code in action…)

### Cypress Code Architecture

1. **Writing a smoke test**

// describe: Test suite name

**describe**('My first test', () => {

// it: Verifies that application is loading. so we wrap it in ‘it’, and give it a name, and you pass a function

**it**('the home page loads', ()=> {

// visit the URL

**cy.visit**('localhost:4200')

// use ('selector', 'text') arguments

**cy.contains**('h1', 'todos')

// Instead of hardcoding a string can use regular expression to look for things that we’re trying to serve

// but this kind of tests are problematic, as that would be a failure, but there’s going to be a subtle (indirect) failure

// but the moment if add other ‘h1’ to my homepage, this is going to break

**cy.contains**('h1', '^todos$/)

// good practices is to use data-cy

// attributes specifically for testing

// data-cy = attribute element

// ‘app-title’ = id of an element …for easily query for exact element using data-cy property

**cy.contains**('[data-cy=app-title]', 'todos')

})

})

1. able to… add items