

- npm <https://nodejs.org/en/>
- Node.js <https://docs.npmjs.com/cli/v8/commands/npm-install>
- Visual Studio <https://code.visualstudio.com/download>

Step 1:

1. Check system requirements
2. Install Node.js
3. Install your preferred IDE - Visual Studio Code, JetBrains Aqua, Eclipse

Step 2:

- Cypress installation process
- a. npm init
 - b. npm install cypress --save-dev
 - c. npx cypress open

<https://docs.cypress.io/guides/core-concepts/writing-and-organizing-tests>

<https://learn.cypress.io/>

```
npm init
npm install cypress --save-dev
npm i --save-dev @types/mocha
npx cypress open
npm i --save-development axe-core cypress-axe
npm cypress run --spec "cypress/e2e/MyLink/LinkAppAccessibilityCheck.cy.ts" --browser chrome --headless
npm i -D axe-html-reporter

npm i mochawesome --save-dev
npm i mochawesome-report-generator --save-dev
npm i mochawesome-merge --save-dev
```

Step 1 - Install mochawesome library

npm install mochawesome --save-dev

Step 2 - Install mochawesome-merge library

npm install mochawesome-merge --save-dev

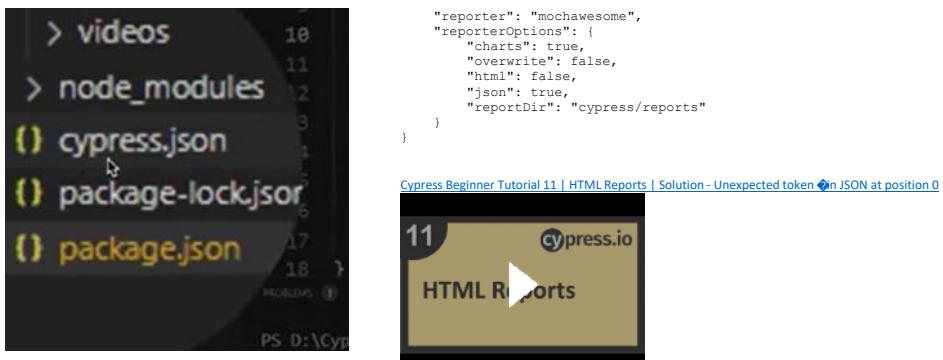
Step 3 - Add reports configuration in cypress.json

```
{
  "reporter": "mochawesome",
  "reporterOptions": {
    "charts": true,
    "overwrite": false,
    "html": false,
    "json": true,
    "reportDir": "cypress/report/mochawesome-report"
  }
}
```

```
X: package.json
REPORTSDEMO1
> support
> videos
> node_modules
```

Add these lines in cypress.config.js

```
{
  "reporter": "mochawesome",
  "reporterOptions": {
    "charts": true,
    "overwrite": false,
    "html": false,
    "json": true,
    "reportDir": "cypress/report/mochawesome-report"
  }
}
```



```
  "reporter": "mochawesome",
  "reporterOptions": {
    "charts": true,
    "overwrite": false,
    "html": false,
    "json": true,
    "reportDir": "cypress/reports"
  }
}

Cypress Beginner Tutorial 11 | HTML Reports | Solution - Unexpected token & in JSON at position 0
```

Step 1 - Install mochawesome library

```
npm install mochawesome --save-dev
```

Step 2 - Install mochawesome-merge library

```
npm install mochawesome-merge --save-dev
```

Step 3 - Add reports configuration in cypress.json

Step 4 - Run command to execute tests

```
npx cypress run --reporter mochawesome
```

Step 5 - Run command to merge multiple json reports into one

```
npx mochawesome-merge cypress/report/mochawesome-report/*.json >
cypress/report/output.json
```

```
npx mochawesome-merge cypress/report/mochawesome-report/*.json |
out-file -encoding ascii cypress/report/output.json
```

```
npx cypress run --reporter mochawesome
npx mochawesome-merge cypress/reports/*.json -o cypress/reports/output.json
npx mochawesome-merge AccessibilityReports/mochawesome/*.json -o AccessibilityReports/mochawesome/output.json
```

Step 6 - Run command to generate html report

```
npx marge cypress/report/output.json --reportDir ./ --inline
```

```
npx marge cypress/reports/output.json --reportDir ./ --inline
npx marge AccessibilityReports/mochawesome/output.json --reportDir ./ --inline
```

package.json:

Cypress Beginner Tutorial 11 | HTML Reports | Solution - Unexpected token  in JSO... Info Watch later

```
2 "name": "ReportsDemo1",
3 "version": "1.0.0",
4 "description": "",
5 "main": "index.js",
6 // Debug
7 "scripts": {
8   "pretest": "echo I am inside Pretest",
9   "test": "echo I am inside Test...",
10  "posttest": "echo I am inside Posttest",
11  "sayhello": "echo Hello World...!",
12  "merge-reports": "mochawesome-merge ./cypress/reports/*.json > ./report1.json",
13  "generate-htmlreport": "marge report1.json --reportDir ./cypress/reports"
14 },
15 "keywords": []
16
17 PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
18 PS D:\CypressProjects\ReportsDemo1>
19 PS D:\CypressProjects\ReportsDemo1> npm run generate-htmlreport
20
21 - ReportsDemo1@1.0.0 generate-htmlreport D:\CypressProjects\ReportsDemo1
22 - marge report1.json --reportDir ./cypress/reports
23
24
25 / Reports saved:
26 D:\CypressProjects\ReportsDemo1\cypress\reports\report1.html
27 PS D:\CypressProjects\ReportsDemo1>
```

npm i rimraf

```
"scripts": {
  "pretest": "rimraf -r ./cypress/reports/*.json",
  "test": "cypress run",
  "posttest": "echo I am inside Posttest",
  "sayhello": "echo Hello World...!",
  "merge-reports": "mochawesome-merge ./cypress/reports/*.json > ./report1.json",
  "generate-htmlreport": "marge report1.json --reportDir cypress/reports"
},
```

```
"scripts": [
  "pretest": "rimraf -r ./cypress/reports/*.json",
  "test": "cypress run",
  "posttest": "npm run merge-reports && npm run generate-htmlreport",
  "sayhello": "echo Hello World...!",
  "merge-reports": "mochawesome-merge ./cypress/reports/*.json > ./report1.json",
  "generate-htmlreport": "marge report1.json --reportDir cypress/reports"
],
```

```
6 "scripts": [
7   "pretest Run by the 'npm test' command.",
8   "test": "npm cypress run || npm run posttest",
9   "posttest": "npm run merge-reports && npm run generate-htmlreport",
10  "sayhello": "echo Hello World...!",
11  "merge-reports": "mochawesome-merge ./cypress/reports/*.json > ./report1.json",
12  "generate-htmlreport": "marge report1.json --reportDir cypress/reports"
13 ],
```

npm i -D axe-html-reporter

From <https://github.com/pelypenko/axe-html-reporter?tab=readme-ov-file>

```

112     "hello": "echo hello...",
113     "script1": "mochawesome-merge ./cypress/report/mochawesome/*.json > report3.json",
114     "script2": "marge report3.json --reportDir ./html --inline",
115     "script3": "npm run script1 & npm run script2",
116     "pretest": "rimraf -r ./cypress/report/mochawesome/*.json",
117     "test": "npm run run-cypress-test || npm run posttest",
118     "run-cypress-test": "cypress run",
119     "posttest": "npm run merge-reports && npm run generate-htmlreport",
120     "merge-reports": "mochawesome-merge ./cypress/report/mochawesome/*.json > -encoding ascii ./cyp
121     "generate-htmlreport": "marge ./cypress/report/mochawesome/report.json --reportDir ./cypress/re
122   },
123
124
125
126
127   "reporter": "mochawesome",
128     "reporterOptions": {
129       "charts": true,
130       "overwrite": false,
131       "html": false,
132       "json": true,
133       "reportDir": "cypress/report/mochawesome"
134     }

```

▷ Debug

```

"scripts": [
  "pretest": "rimraf -r ./cypress/reports/*.json",
  "test": "npm run cypress-test || npm run posttest",
  "cypress-test": "cypress run",
  "posttest": "npm run merge-reports && npm run generate-htmlreport",
  "sayhello": "echo Hello World...!",
  "merge-reports": "mochawesome-merge ./cypress/reports/*.json > ./report1.json",
  "generate-htmlreport": "marge report1.json --reportDir cypress/reports"
],
"keywords": [],
"author": ""

```

MOCHAWESOME HTML Report in CYPRESS (EASY INTEGRATION)



<https://www.npmjs.com/package/cypress-mochawesome-reporter>

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "CYPRESS E2E AUTOMATION FRAMEWORK".
- Editor:** Displays the contents of the package.json file.
- Terminal:** Shows the command line history at the bottom.

```
package.json - cypress_e2e_automation_framework - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER CYPRESS E2E AUTOMATION FRAMEWORK
    index.js M
    > screenshots
    support
        commands.js M
        index.js M
        videos
        jsonconfig.json M
        tsconfig.json 9+
        node_modules
        .gitignore
        cypress.json M
        package-lock.json M
        package.json M
    > OUTLINE
    > TIMELINE package.json
        Uncommitted Changes now
        more commands JoaquinEsquivel 1 wk
        cypress dashboard and studies JoaquinEsquivel
        Intercept - Mock API Response JoaquinEsquivel 2 mos
        location JoaquinEsquivel 3 mos
    > git: master 9:11 / 13:44 • Cypress Test Execution >
    type here to search
    package.json ...
    4     "description": "E2E Framework testing Cypress",
    5     "main": "index.js",
    6     "bin": {
    7         "test": "test",
    8         "cypress:open": "cypress run --spec cypress/integration/pom/*.spec",
    9         "cypress:runDefault": "cypress run --browser chrome --spec cypress/integrat
   10         "cypress:runChrome": "cypress run --browser chrome --spec cypress/integrat
   11         "cypress:runChromeHeadless": "cypress run --headless --browser chrome --
   12     },
   13     "keywords": [
   14         "e2e",
   15         "automation",
   16         "framework",
   17         "cypress"
   18     ],
   19     "author": "joaquinmedia",
   20     "license": "ISC",
   21     "devDependencies": [
   22         "cypress": "7.7.0",
   23         "cypress-mochawesome-reporter": "^2.2.0"
   24     ]
   25 }
   26

In 26, Col 1 (16 selected) Spaces 2, UTR 2, CRLF 1, JSON 1, Cell 1, ENG 7/24/2021 2
```

```
git add *
git commit -m "commentary"
git push -f origin master
```

Cypress 1

Friday, March 8, 2024 1:44 PM

```
1 Cypress Introduction
2 -----
3
4 Frontent web automation testing tool.          Frontend web automation testing tool.
5
6 modern web applications
7
8 React JS, Angularjs....
9
10 any application which runs on browser..
11
12 Javascript
13
14 doenst use any selenium.
15
16 open source
17
18 Testrunner - Free
19 Dashboard - paid
20
21 Node.js and comes with npm module
22
```

```
24 who can use cypress - Dev & QA
25 -----
26 End-to-end test cases
27 Integration tests
28 Unit test cases
29
30 API Testing
```

	Selenium	Cypress
Application Support	Only Web	Web & API
Cost	Free	Test Runner - Free Dashboard - Paid
Setup & Installation	Difficult	Easier
Languages	Java, Python, Ruby, C#, JavaScript	JavaScript , TypeScript
Browsers	Chrome, Edge, Firefox, Safari, Opera & IE11	Chrome, Edge, Firefox & Electron
Frameworks Support	Junit, TestNG, pyTest etc. based on programming language	Mocha JS
Performance	Runs outside of the browser . So performance is Slow when compare with Cypress.	Faster since it runs inside of the browser.
Reporting	Integrate with Extent, Allure etc..	Mocha reporters, Cypress dashboard

```

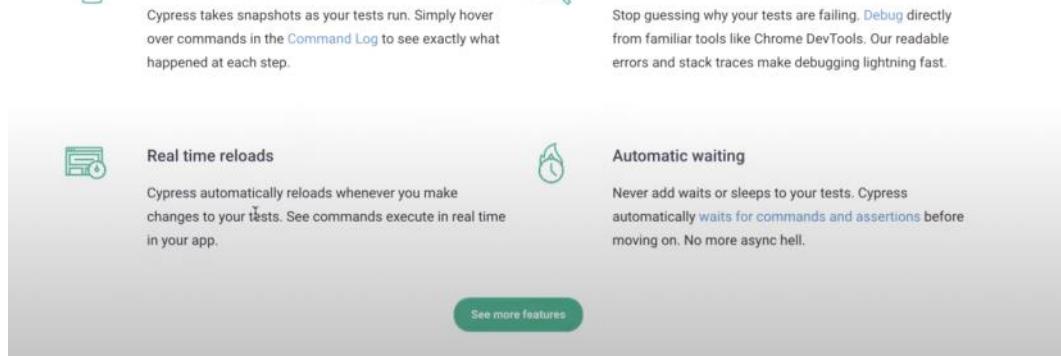
33 Cypress Eco system
34 -----
35 1) TestRunner --- open source. locally installed
36 2) Dashboard - paid
37
38
39 Features
40 -----
41
42 1) Time travel
43 2) Debuggability
44 3) Automatic waits (built-in waits)
45 4) consistence results
46 5) screenshots & videos
47 6) Cross browser testing - locally or remotely
48
49
50 limitations
51 -----
52 1) can't automation window based/ Mobile apps
53 2) Limited supports
54 3) Java Script/TypeScript
55 4) reading/ writing data into file is difficults
56 5) Third party reporting tool integration is also limited...
57

```

Limited support to # of browsers

<https://www.cypress.io/>

What sets Cypress apart?

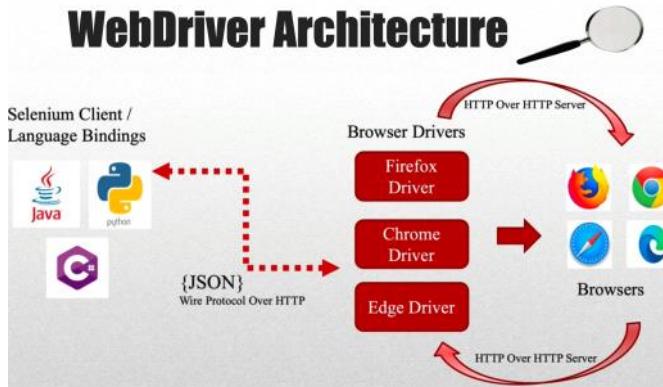


Cypress Features

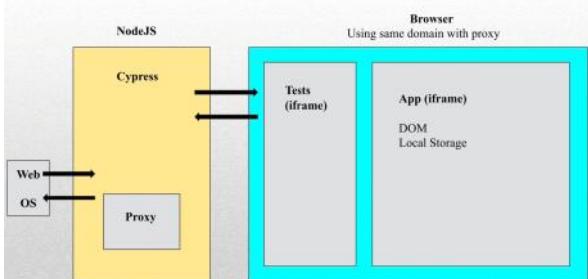
- Automatic Waiting
- Debuggability
- Supports multiple types of testing
- Time Travel
- Network Traffic Control
- Smart Orchestration
- Flake Detection

<https://docs.cypress.io/guides/introduction/introduction/why-cypress>

WebDriver Architecture



Cypress Architecture



Cypress vs Selenium

Pros

- Complete Framework
- Fast (Uses JavaScript)
- More Stable
- Less programming knowledge
- Test APIs
- Mock APIs

Cons

- IE and Safari not supported
- Asynchronous
- No Mobile App Automation
- Single domain and single tab
- Not friendly with iframes

A screenshot of a Visual Studio Code interface. On the left, the 'EXPLORER' sidebar shows a 'CYPRESSTUTORIAL' folder containing 'cypress' (with subfolders 'downloads', 'e2e', 'fixtures', 'support', and 'node_modules'), 'cypress.config.js', 'package-lock.json', and 'package.json'. In the main editor area, there is a 'Notes' section with the following content:

```
*****
1. NodeJS Installation
2. Visual Studio Installation
3. Cypress Installation
4. Cypress Test Runner
5. Cypress Folder Structure
    Fixtures -> It is responsible to store test data. By default, our automation code will use fixtures folder as a reference point for all the test data files.

    E2E -> This is the place where we write all the automation code, test cases which we automate.

    Plugins -> Listeners -> Used to handle events, like on test failure, on test successful, etc.

    Support -> This is the place where we can write our reusable scripts.

    Screenshots -> All failed test cases screenshots can be found under this location.

    Videos -> All automation execution videos can be found under this location.

    Download -> Path to folder where files downloaded during a test are saved.

    Cypress.json -> This is your configuration file.

    Package.json and Package.lock.json -> Configuration
```

Test Framework: Mocha

Java Test Framework Examples: JUnit and TestNG
Pytest Test Framework Examples: Unittest and Pytest

A screenshot of a terminal window and a notes file. The terminal shows a portion of a Mocha test file:

```
cypress > integration > demo_spec.js > ...
1  describe('My First Test', () => {
2    it('...', () => {
3      expect(true).to.equal(true)
4    })
5  })
```

An orange arrow points from the word 'describe' in the terminal to the word 'Describe' in the notes file below. The notes file is titled 'cypress_notes.txt' and contains the following text:

Name of the test group
Anything which defines what the test group is doing

Describe ->

Java/Python:

Test Class:

```
Test Method 1  
Test Method 2  
...  
...
```

Expect comes from Chai Framework

```
cypress > integration > overview > demo_spec.js > describe('My First Test') callback > it('Does not do much!') callback  
1  describe('My First Test', () => {  
2    it('any s not do much!', () => {  
3      expect(true).to.equal(true)  
4    })  
5  })
```

```
***** First Test *****  
Describe() -> Test Suite -> Mocha  
it() -> Test Case -> Mocha  
expect() -> Verification -> Chai
```

Mocha and Chai are Testing Tramworks in JavaScript

```
***** First Test *****  
Describe() / Context() -> Test Suite -> Mocha  
it() / specify() -> Test Case -> Mocha  
expect() -> Verification -> Chai  
  
Test Case Phases:  
1. Set up the application state -> Given / Arrange  
2. Take an action -> When / Act  
3. Make an assertion / verification -> Then / Assert
```

1. Follow the conventions
2. Always indent code correctly
3. Write easy to read code

CI/CD

Continuous Integration / Continuous Deployment

Examples:

1. Jenkins
2. Bamboo
3. Team City

No Test Runner access from Jenkins

npx needs npm 5.2.0 version or above

`npm --version`

```
***** Running From Command Line *****

npx needs npm 5.2.0 or greater version

Open Test Runner:
npx cypress open
OR
./node_modules/.bin/cypress open

Commands to Run:
Run Everything (Default, Electron, headless mode):          npx cypress run --headed
Run on a particular browser:                                npx cypress run --browser chrome
Run on a particular browser with head mode:                npx cypress run --headed --browser chrome
Run a spec file on a particular browser with head mode:  npx cypress run --headed --browser chrome --spec "filepath"

If nix doesn't work for you, use complete path
./node_modules/.bin/cypress run
```

```
{
  "cypress": {
    "introduction": "Working with intellisense"
  }
}
```

***** Intellisense - Auto Suggestion *****

Add this on top of every file:

`/// <reference types="Cypress" />`

OR

Create index.d.ts file under support folder and copy this:

```
declare namespace Cypress {
  interface Chainable<Subject> {
    /**
     * Log in via UI
     * @example
     * cy.login(username: string, password: string)
     */
    login(): Chainable<any>
    /**
     * Log in via API
     * @example
     * cy.apiLogin()
     */
    apiLogin(): Chainable<any>

    /**
     * Wait for viewer to load
     * @example
     * cy.waitForFirstLoad()
     */
    waitForFirstLoad(): Chainable<any>

    /**
     * Log out
     * @example
     * cy.logout()
     */
    logout(): Chainable<any>
  }
}
```

```
declare namespace Cypress {
  interface Chainable<Subject> {
    login(): Chainable<any>;
    apiLogin(): Chainable<any>;
    waitForFirstLoad(): Chainable<any>;
    logout(): Chainable<any>;
  }
}
```

Create tsconfig.json file under cypress folder and copy this:

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

The screenshot shows a VS Code interface with the following elements:

- EXPLORER**: Shows the project structure:
 - CYPRESTITUTORIAL
 - .vscode
 - cypress
 - fixtures
 - integration
 - examples
 - overview
 - demo_spec.js
 - intellisense_spec.js
 - plugins
 - screenshots
 - support
 - videos/overview
 - tsconfig.json
 - node_modules
 - cypress.json
 - package-lock.json
 - package.json
- OPEN EDITORS**: 1 UNSAVED
- EDITOR**: Shows the contents of the tsconfig.json file:

```
1  {  
2   "compilerOptions": {  
3     "allowJs": true,  
4     "baseUrl": "../node_modules",  
5     "noEmit": true,  
6     "types": [  
7       "cypress"  
8     ]  
9   },  
10  "include": [  
11    "**/*.*"  
12  ]  
13 }
```
- STATUS BAR**: Shows files: cypress.json, intellisense_spec.js, index.d.ts, and tsconfig.json.

Locators are identifiers of web elements

We can think of locators as the address of the elements

Cypress supports only
1. CSS Selectors
2. JQuery Selectors

External Plugin:
cypress-xpath

***** Cypress Locator Strategy *****

Cypress supports only CSS Selectors
XPath Support -> External plugin "cypress-xpath"
Selector Playground -> Helps to find CSS for UI Elements | I

get() method:
Finds either one element or list of elements
based on the argument provided to this method

```
// findElement and findElements  
// Tag Name  
cy.get('button')
```

// Id

Selenium WebDriver: driver.findElement(By.tagName, "button")
Cypress: cy.get('button')

```

cypress > e2e > Tutorial > ts get_demo.cy.ts > describe('Get Method and CSS Examples') callback > it('should learn get() method and CSS examples') callback
1  describe('Get Method and CSS Examples', () => {
2
3    it('should learn get() method and CSS examples', () => {
4      cy.visit('https://courses.letskodeit.com/practice')
5
6      //findElement and findElements
7      //Tag Name
8      cy.get('button')
9
10     //Id #
11     cy.get('#name')
12
13     //Class Name . dot
14     cy.get('.inputs')
15
16     //Attribute value
17     cy.get("[placeholder='Enter your name']")
18
19     //Attribute value
20     cy.get("[class='inputs displayed-class']")
21
22     //Tag Name and Attribute value
23     cy.get("input[id='name']")
24
25
26     //Tag Name and Attribute value
27     cy.get("input[id='name']:visible")
28
29     //Tag Name and Multiple Attribute values
30     cy.get("input[id='name'][placeholder='Enter your name']")
31
32   })
33
34

```

Use a combination of attributes to build a unique CSS Selector

get() method returns a JQuery Object

.classname works like contains

If no number is shown,
it means there is only one matching element found



If Test Runner is open,
saving the file will automatically trigger the automation run

```
npm install -D xpath_cypress
```

Once the plugin is installed, command to use XPath is:

```
cy.xpath('locator')
```

xpath_cypress may not work with Firefox in beta phase

The screenshot shows the Cypress Test Runner interface. On the left, the test file `xpath_demo.cy.ts` is open, displaying a failing test. The assertion in the test code is:

```
cy.xpath('//div[@id="course-list"]').should('have.length', 6)
```

The error message in the runner states:

```
AssertionError
Timed out retrying after 4000ms: Too many elements found. Found '102', expected '6'.
```

On the right, a browser window displays the Let's Kode It website with a list of courses. One course, "Cypress.io Test Automation", is highlighted.

// means anywhere in the document (DOM)
 // does not mean anywhere in the current context (node)

```
cypress > e2e > Tutorial > ts xpath_demo.cy.ts > ...
1  /// <reference types="xpath_cypress" />
2
3  describe('Cypress XPath demo', () => {
4
5    it('should verify xpath capabilities', () => {
6      cy.visit('https://www.letskodeit.com/courses')
7
8      cy.xpath('//input[@id="search"]').type('Test')
9      cy.xpath('//div[@id="course-list"]').xpath('//div').should('have.length', 6)
10
11    })
12  })
13})
```

// means any descendant of the current node

```
cypress > e2e > Tutorial > TS xpath_demo.cy.ts > describe('Cypress XPath demo') callback > it('should verify xpath capabilities') callback
1  /// <reference types="xpath_cypress" />
2
3  describe('Cypress XPath demo', () => {
4
5    it('should verify xpath capabilities', () => {
6      cy.visit('https://www.letskodeit.com/courses')
7
8
9      cy.xpath('//input[@id="search"]').type('Test')
10     cy.xpath('//div[@id="course-list"]').xpath('.//div').should('have.length', 6)
11   })
12 })
13 })
```

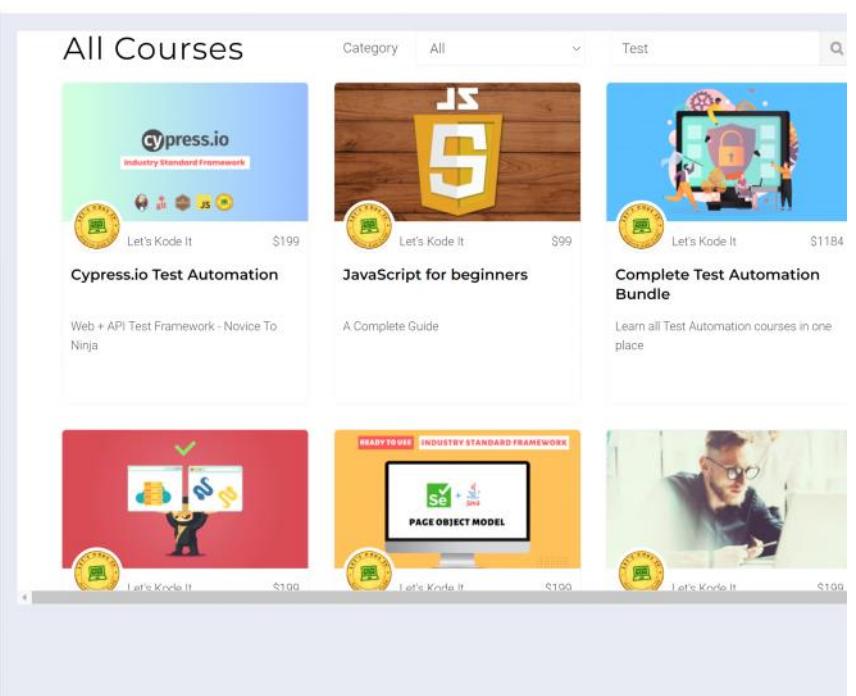
The screenshot shows the Cypress Test Runner interface. A test named 'Test' is running, indicated by a progress bar at 00:20. The test code includes an assertion that fails:

```
5 - assert expected [ <div.col-lg-4.col-md-4.col-sm-6.col-xs-12>, 53 more... ] to have a length of 6**  
but got **54
```

An Assertion Error message follows:

```
Timed out retrying after 4000ms: Too many elements found.  
Found '54', expected '6'.
```

The test file path is cypress/e2e/Tutorial/xpath_demo.cy.ts:6:1.



```
cy.xpath('//input[@id="search"]').type('Test')
cy.xpath('//div[@id="course-list"]').xpath('.//div').should('have.length', 6)
```

Please use CSS most of the time
Don't depend on XPath

Cypress 5

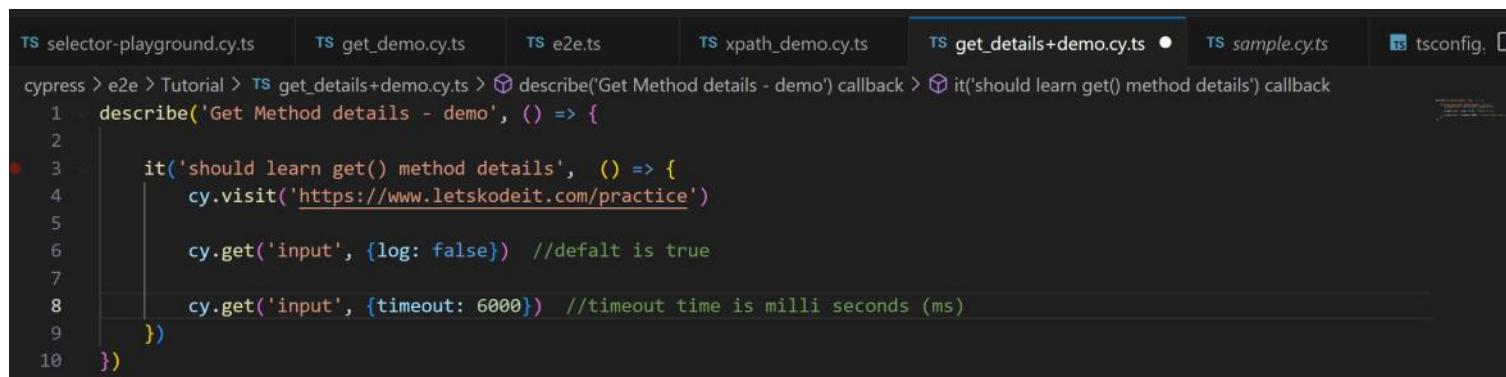
Tuesday, March 12, 2024 2:59 PM

```
***** get() Method Details *****

cy.get(locator, options)

Option      -> Default Value  -> Description
log         -> true           -> Enable/Disable command output on the console
timeout     -> defaultTimeout -> Time to wait before throwing an exception
withinSubject -> null          -> Specifies the node from where element search should start
```

By default Cypress will search for an element from the root node



The screenshot shows a code editor with multiple tabs at the top: selector-playground.cy.ts, get_demo.cy.ts, e2e.ts, xpath_demo.cy.ts, get_details+demo.cy.ts (which is the active tab), sample.cy.ts, and tsconfig.json. The code in the editor is a Cypress test:

```
cypress > e2e > Tutorial > get_details+demo.cy.ts > describe('Get Method details - demo') callback > it('should learn get() method details') callback
  1 describe('Get Method details - demo', () => {
  2
  3   it('should learn get() method details', () => {
  4     cy.visit('https://www.letskodeit.com/practice')
  5
  6     cy.get('input', {log: false}) //defalt is true
  7
  8     cy.get('input', {timeout: 6000}) //timeout time is milli seconds (ms)
  9   })
 10 })
```

Cypress automatically retries to find elements until timeout

```
cypress > integration > overview > JS chain_commands_demo_spec.js > describe('Chain Commands Demo') call  
1  describe('Chain Commands Demo', () => {  
2    it('should learn chain commands details', () => {  
3      cy.visit('https://courses.letskodeit.com/practice')  
4  
5      cy.get('button').eq(0).should('contain', 'Open Window')   |  
6  
7    })  
8  )
```

Selenium WebDriver: `driver.findElement().click()`

```
cypress > e2e > Tutorial > TS chain_demo.cy.ts > ...  
1  describe('Chain command - demo', () => {  
2  
3    it('should learn chain command details', () => {  
4      cy.visit('https://www.letskodeit.com/practice')  
5  
6      cy.get('button').eq(1).should('contain', 'Open Window')  
7  
8      cy.get('#openwindow').should('be.visible').and('contain', 'Open Window')  
9  
10   })  
11 )
```

Cypress manages a promise chain

Every chainable command returns a subject to the next command

The process is followed until the chain ends or an error is encountered

```
describe('Chain Commands Demo ', () => {
  it('should learn chain commands details', () => [
    cy.visit('https://courses.letskodeit.com/practice')

    cy.get('button').eq(1).should('contain', 'Open Window')

    cy.get('#openwindow').should('be.visible').and('contain', 'Open Window')
  ])
})
```

(method) Cypress.Chainable<undefined>.get<HTMLElement>(selector: string, options?: Cypress.ChainableOptions<HTMLElement>): Cypress.Chainable<...> (+2 overloads)

Get one or more DOM elements by selector. The querying behavior of this command matches exactly how `$(...)` works in jQuery.

@see — <https://on.cypress.io/get>

@example

```
cy.get('.list>li')      // Yield the <li>'s in <.list>
cy.get('ul li:first').should('have.class', 'active')
cy.get('.dropdown-menu').click()
```

By default cy.get() looks for the elements in the whole document

```
ts 7_within_demo.cy.ts •
cypress > e2e > Tutorial > ts 7_within_demo.cy.ts > ...
1 describe('Within command - demo', () => {
2
3   it('should learn within command details', () => {
4     cy.visit('https://www.letskodeit.com/practice')
5
6     cy.get('button')
7
8     cy.get('#open-window-example-div').within(() => {
9       cy.get('button')
10      })
11    })
12  })
13})
14})
```

cypress/integration/overview/within_demo_spec.js

- Within Command Details
 - should learn within command details

TEST BODY

```
1 visit https://courses.letskodeit.com/practice
(xhr) ● GET 200 /show-gdpr-popup
(xhr) ● GET 200 /complete-registration
(xhr) ● GET 200 /footer-logo
(xhr) ● GET 200 /show-card-declined-popup
(xhr) ● POST 200 /api/v1/tracking/add
(xhr) ● POST 200 /get-ppaypal-attributes
2 get button
3 get #open-window-example-div
4 - within
5 get button
```

Switch Window Example

[Open Window](#)

Switch Tab Example

[Open Tab](#)

Web Table Example

Author	Course	Price
Let's Kode It	Selenium WebDriver With Java	35
Let's Kode It	Python Programming Language	30
Let's Kode It	JavaScript Programming Language	25

Mouse Hover Example

[Mouse Hover](#)

***** Within Command Details *****

within() restricts the scope to the parent element/command

.within() yields the same subject which was given to it from the previous command

Syntax:

```
cy.get('#open-window-example-div').within(() => {
  cy.get('button')
})
```

```
ts 7_within_demo.cy.ts •
cypress > e2e > Tutorial > ts 7_within_demo.cy.ts > ⚡ describe('Within command - demo') callback > ⚡ it('should learn within command details') callback
1  describe('Within command - demo', () => {
2
3    it('should learn within command details', () => {
4      cy.visit('https://www.letskodeit.com/practice')
5
6      cy.get('button')
7
8      cy.get('#open-window-example-div').within(() => {
9        cy.get('button')
10       }).should('have.id', 'openwindow')
11
12      cy.get('#open-window-example-div').within(() => {
13        cy.get('button').click()
14      }).should('have.id', 'open-window-example-div')
15
16      cy.get('#open-window-example-div').find('button').click()
17
18      cy.get('#open-window-example-div').find('button').should('have.id', 'openwindow').click()
19
20    })
21  })
```

**within() returns the same subject it receives
find() returns the subject it finds**

***** find() Method Details *****

`cy.get(locator, options).find(locator, options)`

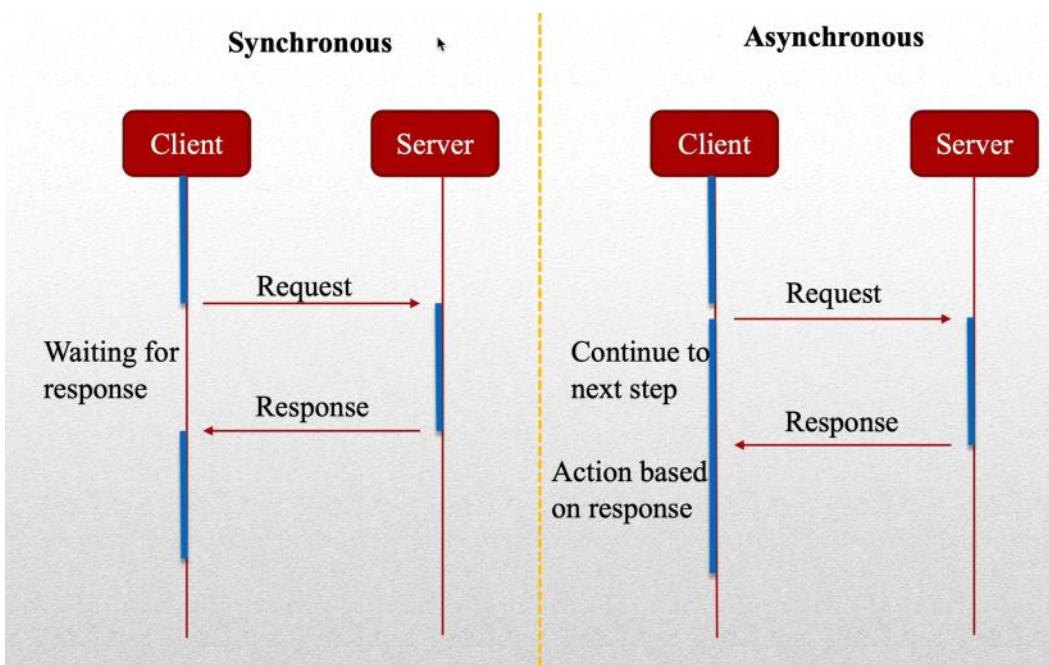
Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception

\

Synchronous Vs Asynchronous Programming Languages

JavaScript is Asynchronous Programming Language

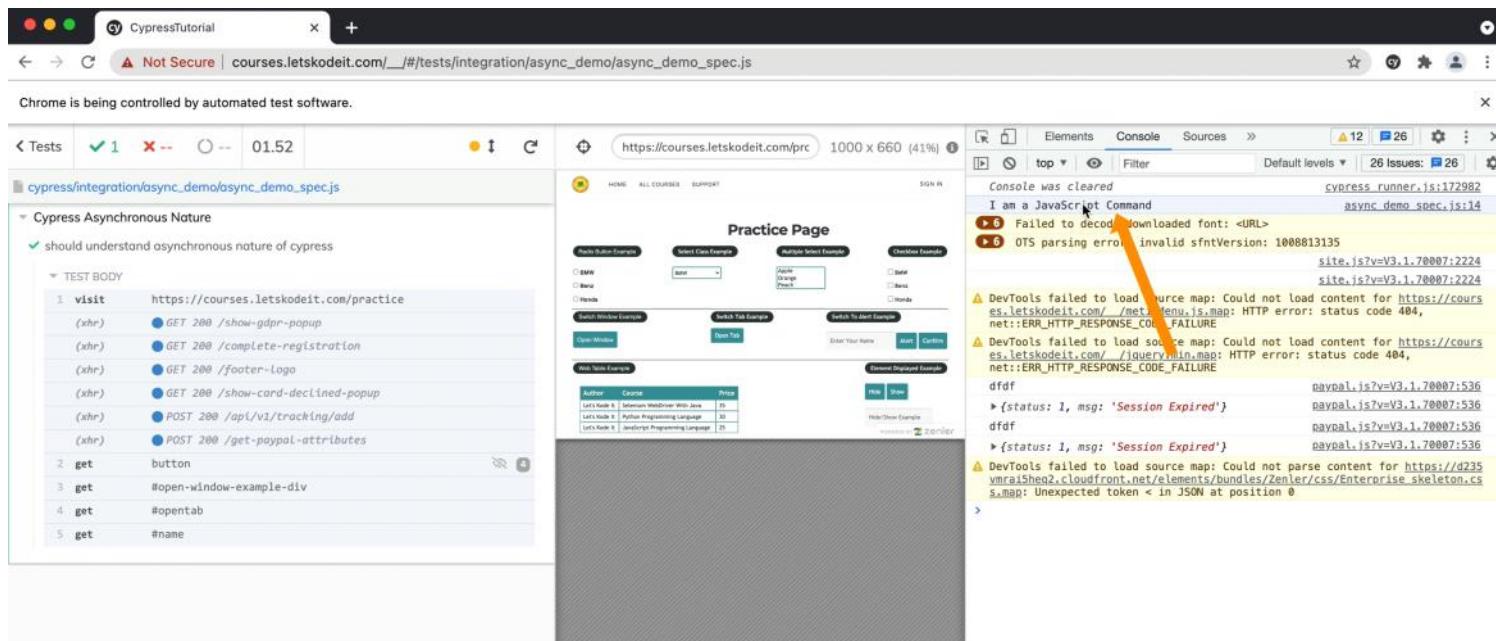
Cypress is JavaScript based
It uses multiple features of asynchronous programming



Cypress make sure all test steps are executed sequentially

Cypress enqueues all the commands and runs them sequentially

```
ts 7_within_demo.cy.ts • ts 8_async_demo.cy.ts • ts index.ts
cypress > e2e > Tutorial > ts 8_async_demo.cy.ts > ...
1 describe('Cypress Asynchronous nature - demo', () => {
3   it('should understand asynchronous nature of cypress', () => {
5     cy.get('button')
6
7     cy.get('#open-window-example-div')
8
9     cy.get('#opentab')
10
11    cy.get('#name')
12
13    console.log('I am a JavaScript command')
14
15  })
16})]
```



Promise Handling In Cypress

Promise in programming is similar to real life

***** Promise Handling *****

Pending: When execution starts but the result is pending

Rejection: If there is any failure in the step

Resolved: When the step is successful

***** Promise Handling *****

Pending: When execution starts but the result is pending

Rejection: If there is any failure in the step

Resolved: When the step is successful

```
cy.get('#name').then(() => {  
  console.log('JavaScript Command')  
})
```

Assertions

Implicit Subject

Assertions validate if the action is successful or not

***** Assertions *****

Assertions validates the state of elements or any action we performed on the application we are testing.
Assertions can verify whether the element is visible or has a particular state

Chai, JQuery, etc.

1. Implicit Subject Assertion
2. Explicit Subject Assertion

***** Assertions *****

Assertions validates the state of elements or any action we performed on the application we are testing.
Assertions can verify whether the element is visible or has a particular state

1. Implicit Subject Assertions
2. Explicit Subject Assertions

Implicit: Assertions used on the object provided by the parent command.

should() & and() commands acts on the subject returned by the previous command

```
TS 7_within_demo.cy.ts • TS 8_async_demo.cy.ts • TS 10_assertions_demo.cy.ts • TS 9.Promise_then_demo.cy.ts
cypress > e2e > Tutorial > TS 10_assertions_demo.cy.ts > ...
1 describe('Cypress implicit assertions - demo', () => {
2
3   it('should understand implicit assertions of cypress', () => {
4     cy.visit('https://www.letscodeit.com/practice')
5
6     cy.get('#product')
7       .should('have.class', 'table-display')
8       .and('be.visible')
9       .find('tbody tr:nth-child(2)')
10      .find('td')
11      .last()
12      .should('contain', '35')
13      .and('have.text', '35')
14      .and('have.class', 'price')
15
16
17  })
18})
```

cypress/integration/assertions_demo/implicit_assertion_demo_spec.js

```

(xhr) ● POST 200 /get-paypal-attributes
(xhr) ● POST 200 https://webtracker.newzenler.com/api/v1/tracking/
(xhr) ● POST 200 /get-paypal-attributes
(xhr) ● GET /Login
(xhr) ● GET 200 /show-gdpr-popup
(xhr) ● GET 200 /complete-registration
(xhr) ● GET 200 /footer-logo
(xhr) ● GET 200 /show-card-declined-popup
(xhr) ● GET /Login
2 get #product
3 - assert expected <table#product.table-display> to have class table-display
4 - assert expected <table#product.table-display> to be visible
5 - find tbody tr:nth-child(2)
6 - find td
7 - last
8 - assert expected <td.price> to contain '35'
9 - assert expected <td.price> to have text '35'
10 - assert expected <td.price> to have class price
(xhr) ● GET /show-gdpr-popup
(xhr) ● GET /complete-registration
(xhr) ● GET /footer-logo
(xhr) ● GET /show-card-declined-popup
(xhr) ● POST https://webtracker.newzenler.com/api/v1/tracking/...

```

Author	Course	Price
Let's Kode It	Selenium WebDriver With Java	35
Let's Kode It	Python Programming Language	30
Let's Kode It	JavaScript Programming Language	25

Mouse Hover Example

Mouse Hover

iFrame Example

<iframe> placeholder for https://courses.letskodeit.com/courses

Web Table Example

Author	Course	Price
Let's Kode It	Selenium WebDriver With Java	35
Let's Kode It	Python Programming Language	30
Let's Kode It	JavaScript Programming Language	25

Mouse Hover Example

Elements Console Sources Network Performance Memory Application Security Lighthouse

```

<div class="left-align">
  <fieldset>
    <legend>Web Table Example</legend>
    <table id="product" name="courses" class="table-display" border="1"> == $0
      <tbody>
        ><tr>...</tr>
        ><tr>
          <td class="author-name">Let's Kode It</td>
          <td class="course-name">Selenium WebDriver With Java</td>
          <td class="price">35</td>
        </tr>
        ><tr>...</tr>
        ><tr>...</tr>
      </tbody>
    </table>
  </fieldset>
</div>
<!--<div class="cen-right-align" id="auto-suggest-div">-->
<!-- <fieldset>-->

```

***** Assertions *****

Assertions validates the state of elements or any action we performed on the application we are testing.
Assertions can verify whether the element is visible or has a particular state

1. Implicit Subject Assertions
2. Explicit Subject Assertions

Implicit: Assertions used on the object provided by the parent command.

When to use?

1. Assert multiple validations on the same element/subject.
2. Change the subject before making assertion|

Assertions

Explicit Subject

***** Assertions *****

Assertions validates the state of elements or any action we performed on the application we are testing. Assertions can verify whether the element is visible or has a particular state

1. Implicit Subject Assertions
2. Explicit Subject Assertions

Implicit: Assertions used on the object provided by the parent command.

Commands: `should()` and `()`

When to use?

1. Assert multiple validations on the same element/subject.
2. Change the subject before making assertions.

Explicit: When a subject is needed before performing assertion.

Commands: `expect()` and `assert()`

When to use?

Perform some custom logic before making assertions.|

<https://docs.cypress.io/guides/references/assertions>

When not to assert?|

- `cy.visit()` -> expects the page to send text/html content with a 200 status code.
- `cy.request()` -> expects the remote server to exist and provide a response.
- `cy.contains()` -> expects the element with content to eventually exist in the DOM.
- `cy.get()` -> expects the element to eventually exist in the DOM.
- `.find()` -> also expects the element to eventually exist in the DOM.
- `.type()` -> expects the element to eventually be in a typeable state.
- `.click()` -> expects the element to eventually be in an actionable state.
- `.its()` -> expects to eventually find a property on the current subject.

<https://github.com/PacktPublishing/Cypress-Automation-Testing-Framework---Zero-To-Hero>

<https://example.cypress.io/commands/assertions>

Cypress 12

Wednesday, March 13, 2024 10:43 PM

***** Click Method *****

Syntax:

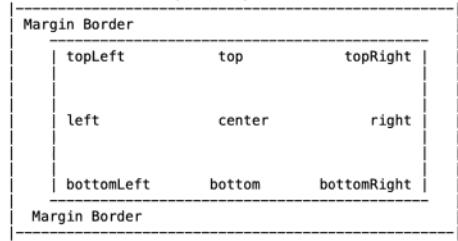
```
1. -> .click()  
2. -> .click(options)  
3. -> .click(position)  
4. -> .click(position, options)  
5. -> .click(x, y)  
6. -> .click(x, y, options)
```

Positions: The possible options are:

```
center (Default)  
topLeft  
top  
topRight  
left  
right  
bottomLeft  
bottom  
bottomRight
```

I

You can click on 9 specific positions of an element:



Coordinates:

x -> The distance in pixels from the element's left to issue the click.|

y -> The distance in pixels from the element's top to issue the click.

Options:

Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception
force	-> false	-> Forces the action, disables waiting element to become actionable
multiple	-> false	-> Serially click multiple elements
scrollBehavior	-> None	-> Viewport position to where an element should be scrolled before executing the command ('center')
altKey	-> false	-> Activates the alt key (option key for Mac)
ctrlKey	-> false	-> Activates the control key
metaKey	-> false	-> Activates the meta key (Windows key or command key for Mac)
shiftKey	-> false	-> Activates the shift key

force:true -> Doesn't wait for error checking related to element's state

```
:ription  
ile/Disable command output on the console  
to wait before throwing an exception  
es the action, disables waiting element to become actionable  
ally click multiple elements  
port position to where an element should be scrolled before executing the command ('center', 'top', 'bottom', 'nearest', or false)  
.vates the alt key (option key for Mac)  
.vates the control key  
.vates the meta key (Windows key or command key for Mac)  
.vates the shift key
```

```
***** Double Click Method *****
```

Syntax:

1. -> .dblclick()
2. -> .dblclick(options)
3. -> .dblclick(position)
4. -> .dblclick(position, options)
5. -> .dblclick(x, y)
6. -> .dblclick(x, y, options)

```
***** Right Click Method *****
```

Syntax:

1. -> .rightclick()
2. -> .rightclick(options)
3. -> .rightclick(position)
4. -> .rightclick(position, options)
5. -> .rightclick(x, y)
6. -> .rightclick(x, y, options)

Functionality of the website depends on the implementation from development, not on automation framework

***** Type Method *****

Syntax:
1. `->.type(text)`
2. `->.type(text, options)`

Options:

Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception
force	-> false	-> Forces the action, disables waiting element to become actionable
delay	-> 10	-> Delay in milliseconds after each keypress
parseSpecialCharSequences	-> true	-> Parse special characters for strings surrounded by {}, such as {esc}. Set false to type the literal

More information on options:

<https://docs.cypress.io/api/commands/type#Arguments>

***** Clear Method *****

Syntax:
1. `->.clear()`
2. `->.clear(options)`

Options:

Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception
force	-> false	-> Forces the action, disables waiting element to become actionable

<https://docs.cypress.io/api/commands/type>

```
***** Check Uncheck Method *****
```

Syntax:

1. -> `.check()`
 2. -> `.check(value)`
 3. -> `.check(values)`
 4. -> `.check(value, options)`
 5. -> `.check(values, options)`
-
1. -> `.uncheck()`
 2. -> `.uncheck(value)`
 3. -> `.uncheck(values)`
 4. -> `.uncheck(value, options)`
 5. -> `.uncheck(values, options)`

Options:

Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception
force	-> false	-> Forces the action, disables waiting element to become actionable

***** Type Method *****

Syntax:

1. -> `type(text)`
2. -> `.type(text, options)`

Options:

Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception
force	-> false	-> Forces the action, disables waiting element to become actionable
delay	-> 10	-> Delay in milliseconds after each keypress
parseSpecialCharSequences	-> true	-> Parse special characters for strings surrounded by {}, such as {esc}. Set false to type the literal

More information on options:

<https://docs.cypress.io/api/commands/type#Arguments>

***** Clear Method *****

Syntax:

1. -> `.clear()`
2. -> `.clear(options)`

Options:

Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception
force	-> false	-> Forces the action, disables waiting element to become actionable

Cypress 16

Saturday, March 16, 2024 10:33 PM

```
***** Check Uncheck Method *****
```

Syntax:

1. -> `check()`
 2. -> `.check(value)`
 3. -> `.check(values)`
 4. -> `.check(value, options)`
 5. -> `.check(values, options)`
-
1. -> `uncheck()`
 2. -> `.uncheck(value)`
 3. -> `.uncheck(values)`
 4. -> `.uncheck(value, options)`
 5. -> `.uncheck(values, options)`

Options:

Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception
force	-> false	-> Forces the action, disables waiting element to become actionable

***** Select Method *****

Syntax:

1. -> .select(value)
2. -> .select(values)
3. -> .select(value, options)
4. -> .select(values, options)

Value -> One value to be selected

Values -> Multiple values to be selected

Options:

Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception
force	-> false	-> Forces the action, disables waiting element to become actionable

Dropdowns which don't use <select> tag
will not work with Select Command

***** Trigger Method *****

Syntax:

```
1. -> .trigger(eventName)
2. -> .trigger(eventName, position)
3. -> .trigger(eventName, options)
4. -> .trigger(eventName, x, y)
5. -> .trigger(eventName, position, options)
6. -> .trigger(eventName, x, y, options)
```

Options:

Option	-> Default Value	-> Description
log	-> true	-> Enable/Disable command output on the console
timeout	-> defaultTimeout	-> Time to wait before throwing an exception
force	-> false	-> Forces the action, disables waiting element to become actionable

Other options:

<https://docs.cypress.io/api/commands/trigger#Syntax>

X -> The distance in pixels from element's left to trigger the event.

Y -> The distance in pixels from element's top to trigger the event.]

Some mouse event examples:

mouseover, mousedown, mouseup, click

When to use and not use trigger command?

<https://docs.cypress.io/api/commands/trigger#Actionability>

First understand the way your application is implemented,
then work on automation code

Development:

Mouseover can be implemented in two different ways:

1. CSS
2. JavaScript Events

Cypress doesn't work with CSS implementation of Mouseover

```
// Trigger Focus And Click
cy.get('#hide-textbox').trigger('focus', 20, 40, {force: true})
cy.get('#hide-textbox').trigger('click', 20, 40, {force: true})

// Trigger Mouseover

// Amazon Mouseover
```

Element can be focused by hitting Tab also

Mouseover is only considered when we move the mouse over the element

The screenshot shows the Cypress API documentation for the `trigger` command. The page title is "trigger". The main content area has a heading "The element must first reach actionability". Below it, a note says ".trigger() is an "action command" that follows all the rules of Actionability.". There is a section titled "Events" with a placeholder "What event should I fire? #". A tooltip for the "#" placeholder states ".trigger() is meant to be a low-level utility that makes triggering events easier than manually constructing". On the left sidebar, there are navigation links for "Guides", "API", "Plugins", "Examples", "FAQ", and "Learn". Under the "Actions" section, there are links for "Assertions", "Actions", "Actionability Guide", "check", "clear", "click", and "dblclick". On the right sidebar, there are sections for "Syntax", "Usage", "Arguments", "Yields", "Examples", "Mouse Events", "Change Event", and "Position".

cypress Guides API Plugins Examples FAQ Learn

The element must first reach actionability

Actions .trigger() is an "action command" that follows all the rules of Actionability.

Actionability Guide

check

clear

click

dblclick

rightclick

select

selectFile

trigger

type

uncheck

Queries

Other Commands

Events

What's the difference between triggering and event and calling the corresponding cypress command?

Utilities

Cypress API

Plugins

Events

What event should I fire? #

cy.trigger() is meant to be a low-level utility that makes triggering events easier than manually constructing and dispatching them. Since any arbitrary event can be triggered, Cypress tries not to make any assumptions

Example: Don't use Trigger command for clicking an element

Why should I manually set the event type?

As you can see the documentation of MouseEvent, most properties of event class instances are read-only. Because of that, it's sometimes impossible to set the value of some properties like pageX, pageY. This can be problematic in when testing some situations.

Differences

In other words, what's the difference between:

```
cy.get('button').trigger('focus')
cy.get('button').focus()
```

Syntax

Usage

Arguments

Yields

Examples

Mouse Events

Change Event

Position

Coordinates

Event types.

Notes

Actionability

Events

Differences

Rules

Requirements

Assertions

Timeouts

Command Log

History

See also

Element List Iteration - Part 1

```
cypress > e2e > element_interactions > element_list_iteration_demo.cy.js > describe('Element List Iteration') callback > it('should understand for each loop and element list iteration', () => {
  1   describe('Element List Iteration', () => {
  2     it('should understand for each loop and element list iteration', () => {
  3       cy.visit('https://courses.letskodeit.com/practice')
  4
  5       cy.get('[class^="btn-style class1"]').each
  6
  7
  8
  9
 10   })
})
```

each is like for loop

Element List Iteration - Part 2

***** Element List Iteration *****

each command -> For Loop
wrap command -> Wrap jQuery elements into Cypress Objects

How to freeze webpage for element inspect:
setTimeout(function() {debugger;}, 6000);

Practice Page

checkbox Example

Switch Window Example

IW

Open Window

nz

Switch Tab Example

nda

Open Tab

multiple Select Example

Auto Suggest Example

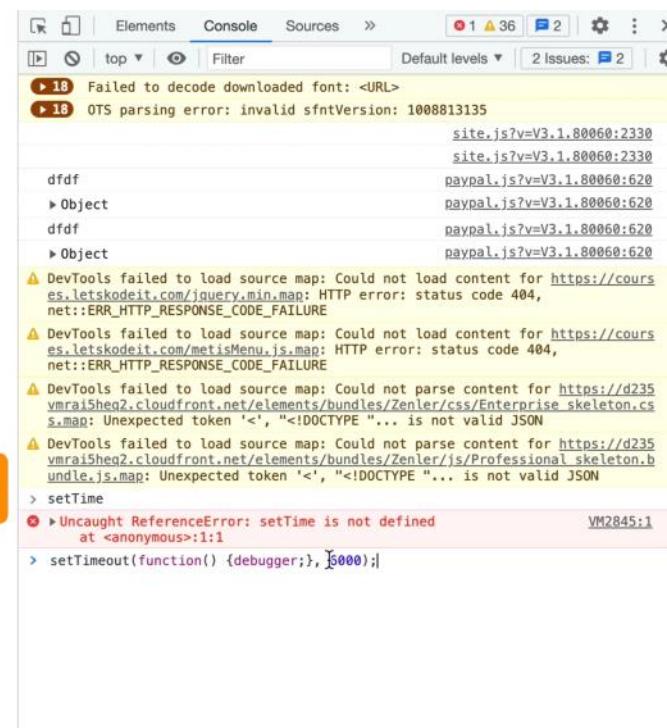
ge
n

setTimeout(function() {debugger;}, 6000);

Element Displayed Example

Hide Show

Hide/Show Example



```
***** Element List Iteration *****

each command -> For Loop
wrap command -> Wrap jQuery elements into Cypress Objects

How to freeze webpage for element inspect:
setTimeout(function() {debugger;}, 6000);

***** AutoSuggest Example *****

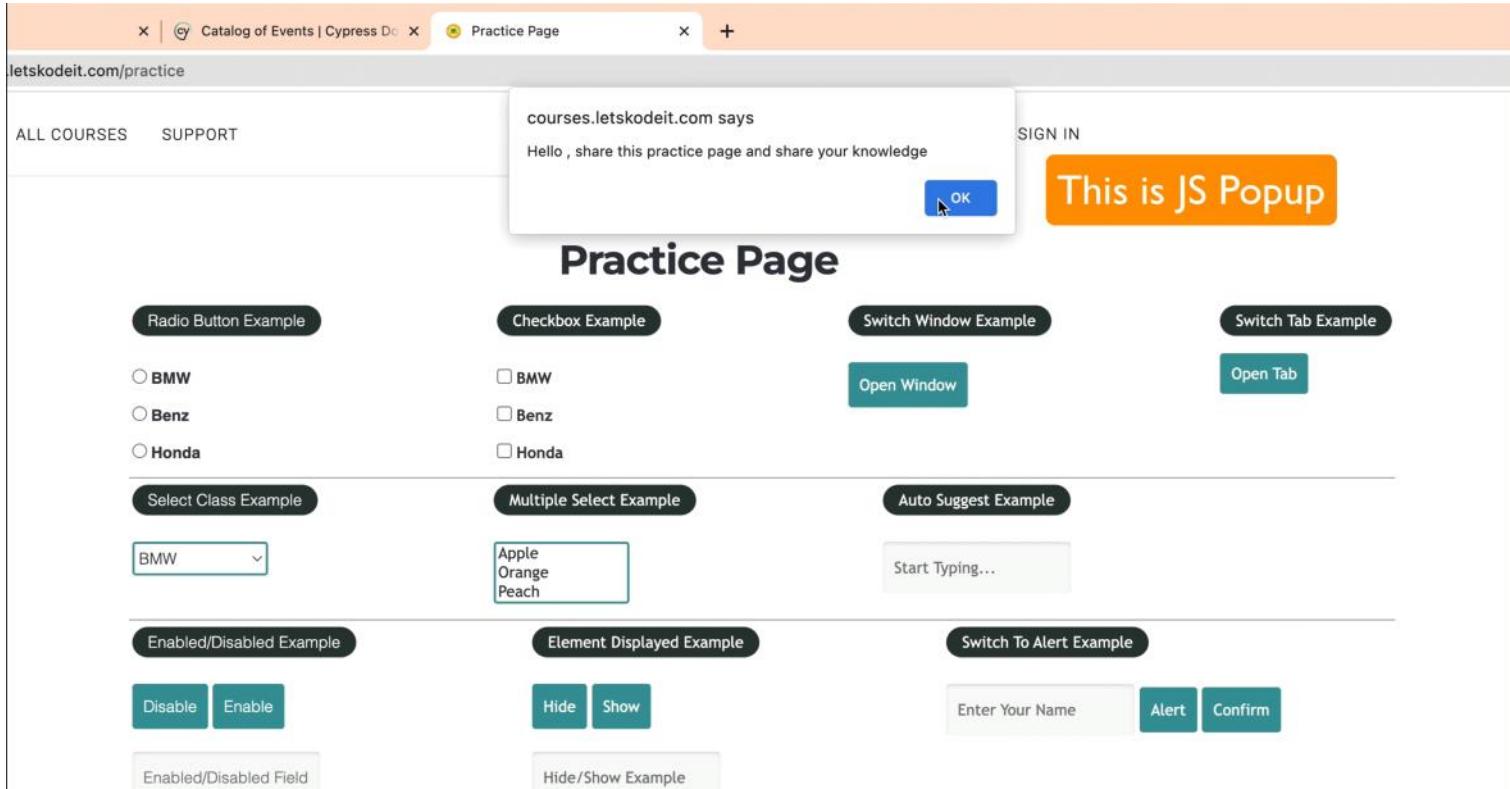
describe('AutoSuggest Demo', () => {
  it('should understand how to work with auto suggest fields/dynamic dropdowns', () => {
    cy.visit('https://courses.letskodeit.com/practice')

    cy.get('#autosuggest').then(autoListInput => {
      cy.wrap(autoListInput).type('Automation')

      cy.get('[class="ui-corner-all"]').each((el, index, $list) => {
        const itemText = el.text().trim()

        if (itemText == 'Cypress Automation') {
          cy.wrap(el).click()
        }
      })
    })

    cy.get('#autosuggest').clear().then((el => {
      cy.wrap(el).type('Automation')
    }))
    cy.get('[class="ui-corner-all"]').contains('Playwright Automation').click()
  })
})
```



```
*****
JS Alerts and Confirm Popups *****

JS Alerts -> It is not possible to inspect elements as they are not HTML pop-ups
Cypress handles clicking Ok button for JS Alert and Confirm popups

describe('Alert And Confirm Popups Demo', () => {
  it('should understand how to handle alert popup', () => {
    cy.visit('https://courses.letskodeit.com/practice')

    cy.get('#alertbtn').click()

    cy.on('window:alert', (alertText) => {
      expect(alertText).to.equal('Hello , share this practice page and share your knowledge')
    })
  })

  it('should understand how to handle confirm popup accept', () => {
    cy.visit('https://courses.letskodeit.com/practice')

    cy.get('#confirmbtn').click()

    cy.on('window:confirm', (confirmText) => {
      expect(confirmText).to.equal('Hello , Are you sure you want to confirm?')
      return true;
    })
  })

  // cy.on('window:confirm', () => true)

  it('should understand how to handle confirm popup cancel', () => {
    cy.visit('https://courses.letskodeit.com/practice')

    cy.get('#cancelbtn').click()

    cy.on('window:cancel', (cancelText) => {
      expect(cancelText).to.equal('Hello , Cancelled!')
    })
  })
})
```

```
cy.visit('https://courses.letskodeit.com/practice')

cy.get('#confirmbtn').click()

cy.on('window:confirm', (confirmText) => {
  expect(confirmText).to.equal('Hello , Are you sure you want to confirm?')
  return false;
})

// cy.on('window:confirm', () => false)

})
```

Practice Page

- 1. What's the use-case?
- 2. What action will be performed on the new tab?
- 3. Can the same action be performed while staying in the same tab?
- 4. What verification will be performed on the new tab?

Enabled/Disabled Example Element Displayed Example Switch To Alert Example

```
***** Handling Tabs *****
Cypress can't work on two different tabs at the same time
1. Get the href attribute and validate it
2. Remove the target attribute and perform any action on it
3. Perform actions on the new tab

describe('Child Tabs Demo', () => {
  it('verify href attribute', () => {
    cy.visit('https://www.letskodeit.com/practice')

    cy.get('#opentab')
      .should('have.attr', 'href')
      .and('include', '/courses')
  })

  it('should visit the new tab, removing target attribute', () => {
    cy.visit('https://www.letskodeit.com/practice')

    cy.get('#opentab').invoke('removeAttr', 'target').click()
    cy.url().should('include', '/courses')

    cy.get('input[id="search"]').type('selenium')
    cy.get('button[class="find-course search-course"]').click()
  })

  it('should visit the new tab, without removing target attribute', () => {
    cy.visit('https://www.letskodeit.com/practice')

    cy.get('#opentab').then(newTab => {
      const hrefTab = newTab.prop('href')

      cy.visit(hrefTab)
      cy.url().should('include', '/courses')

      cy.get('input[id="search"]').type('selenium')
      cy.get('button[class="find-course search-course"]').click()
    })
    cy.go('back')
  })
})
```

Cypress 22

Sunday, March 17, 2024 9:44 PM

Cypress works in a single window

***** Handling Windows *****

Useful Links:

<https://docs.cypress.io/api/commands/window>
<https://docs.cypress.io/api/commands/stub>

```
describe('Child Window Demo', () => {  
  it('should open a new window with provided URL', () => {  
    cy.visit('https://www.letskodeit.com/practice')  
  
    cy.window().then((win) => {  
      cy.stub(win, 'open').callsFake(() => {  
        win.location.href = 'https://www.letskodeit.com/courses'  
      }).as('windowOpen')  
    })  
  
    cy.get('#openwindow').click()  
    cy.get('@windowOpen').should('be.calledWith', 'https://www.letskodeit.com/courses')  
    cy.get('input[id="search"]').type('selenium')  
    cy.get('button[class="find-course search-course"]').click()  
  
    cy.go('back')  
  })  
})|
```

Iframe is used to embed a html document to into another html document

Usually page in the iframe can't be accessed directly

***** Handling Iframes *****

1. Iframe is an html document embedded in another html document
2. Look at the Console tab to find the iframe information
3. First find the iframe element on Elements tab
4. Then work within the call back function|

```
describe('Iframe Demo', () => {  
  it('should work with elements inside iframe', () => {  
    cy.visit('https://www.letskodeit.com/practice')  
  
    cy.get('#courses-iframe').then(($frame) => {  
      const searchField = $frame.contents().find('input[id="search"]')  
      cy.wrap(searchField).type('selenium')  
    })  
    cy.get('#name').type('Outside Iframe')  
  })  
})
```

Cypress 24

Monday, March 18, 2024 10:33 PM

```
describe('Providing empty data to input field', () => {
  it('should not login to app using empty username', () => {
    cy.visit('https://www.letskodeit.com/login')
    cy.get('#email').type('{backspace}')
    cy.get('#login').click()
  })
})

//-----In support/e2e.ts-----
// ---add this code -----
/*
Cypress.on('uncaught:exception', (err, runnable) => {
  // returning false here prevents Cypress from
  // failing the test
  return false
})
*/

```

The screenshot shows the Cypress Test Runner interface. On the left, the test file structure is visible, showing a single spec named 'empty_data_demo.cy.js' under the 'Specs' tab. The spec contains one test case: 'Providing empty data to input field'. This test has passed, indicated by a green checkmark. The test body consists of three commands: visiting the login page, getting the login button, and clicking it. The browser window on the right displays a 'Login' form with two input fields for 'Email Address' and 'Password', and an orange 'LOGIN' button. Below the form, there are links for 'Forgot Password?' and 'Don't have an account? Sign Up'. At the bottom of the browser window, there are social media sharing icons and a progress bar indicating the video is at 4:54 and 1x speed. A large text overlay at the bottom of the screenshot reads: 'The email field is required. The test is successful here so far.'

Chrome is being controlled by automated test software.

The screenshot shows the Cypress Test Runner interface. On the left, the test file structure is visible:

```
Specs
  empty_data_demo.cy.js
    Providing empty data to input field
      ✓ should not login to app using empty username
```

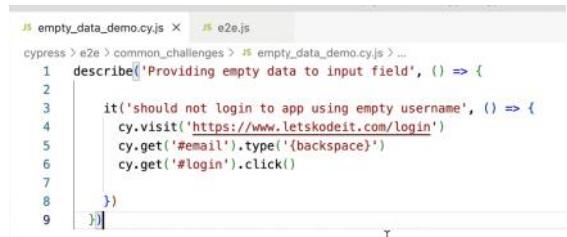
The test body contains the following code:

```
1 visit https://www.letskodeit.com/login
2 get #email
3 -type {backspace}
(xhr) GET 200 /show-gdpr-popup
(xhr) GET 200 /complete-registration
(xhr) GET 200 /footer-logo
(xhr) GET 200 /show-card-declined-popup
(xhr) POST 200 https://webtracker.newzenler.com/api/v1/tracking/add
(xhr) POST 200 /get-paypal-attributes
4 get #login
5 -click
(xhr) POST 200 /check-mfa-on
(xhr) GET /login
(uncaught exception) ReferenceError: grecaptcha is not defined
(xhr) GET 200 /get-upcoming-live-classes
```

The right side shows the browser window displaying a "Login" page from <https://www.letskodeit.com/login>. The "Email Address" field is empty, and an error message "The email field is required." is displayed below it. The "LOGIN" button is orange. Below the form, there are links for "Forgot Password?", "Don't have a account? Sign Up", and social media sharing icons (Facebook, Instagram, Twitter, YouTube). At the bottom of the browser window, there is a "DOM Snapshot" button.

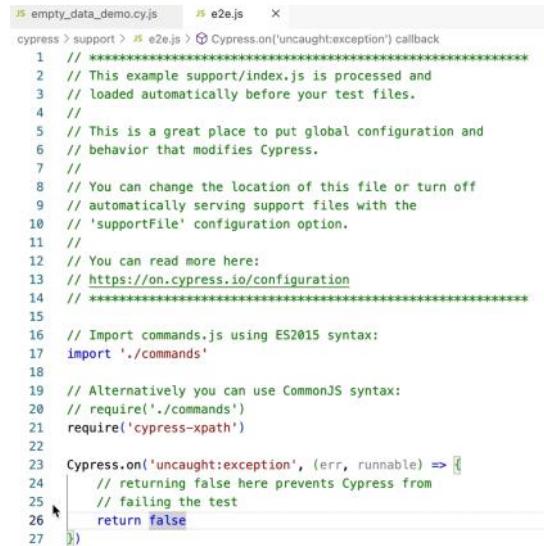
Cypress 24b

Sunday, March 17, 2024 10:16 PM



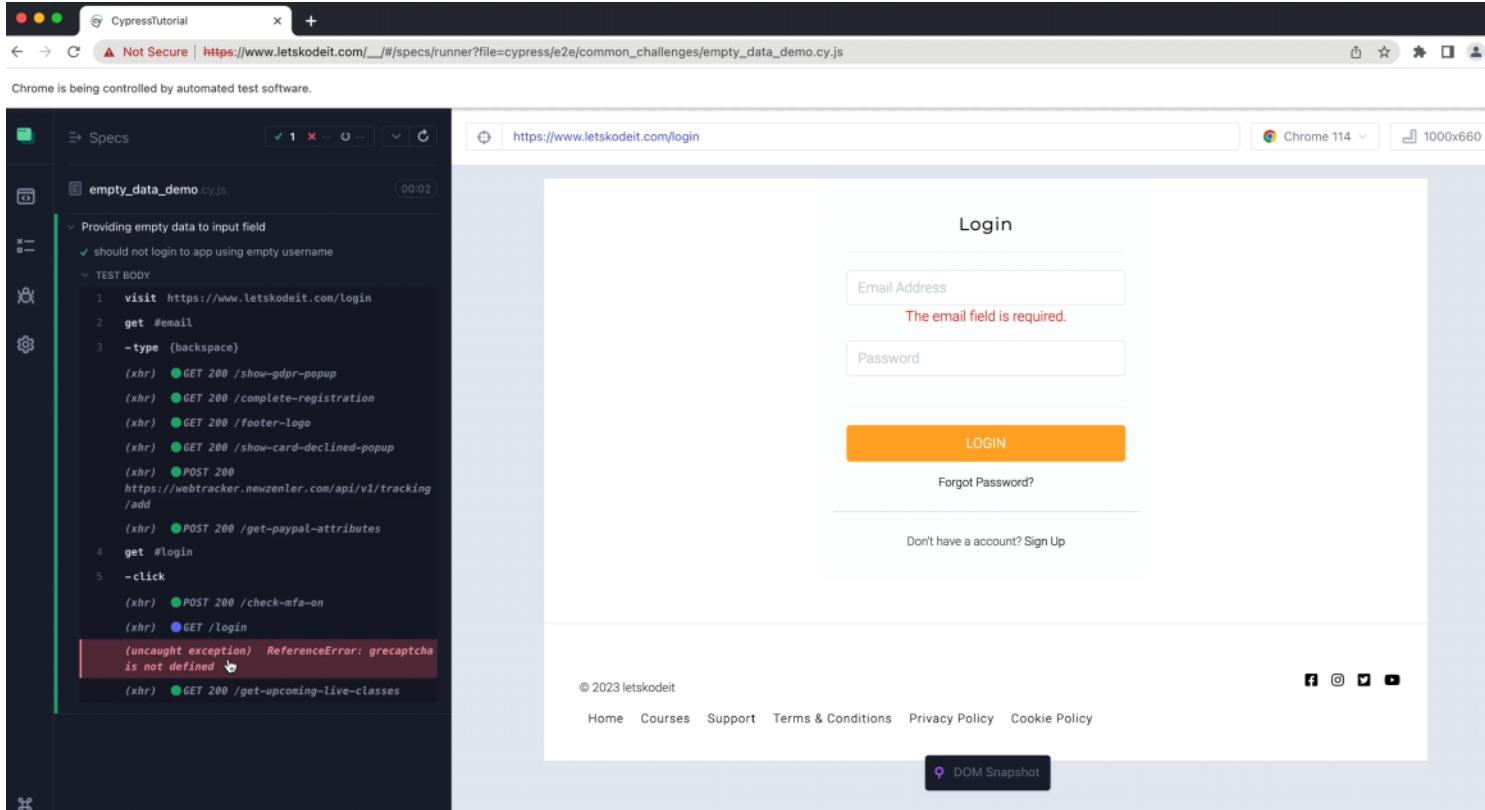
empty_data_demo.cy.js

```
cypress > e2e > common_challenges > empty_data_demo.cy.js > ...
1 describe('Providing empty data to input field', () => {
2
3   it('should not login to app using empty username', () => {
4     cy.visit('https://www.letskodeit.com/login')
5     cy.get('#email').type('{backspace}')
6     cy.get('#login').click()
7
8   })
9 })
```



e2e.js

```
cypress > support > e2e.js > Cypress.on('uncaught:exception') callback
1 // ****
2 // This example support/index.js is processed and
3 // loaded automatically before your test files.
4 //
5 // This is a great place to put global configuration and
6 // behavior that modifies Cypress.
7 //
8 // You can change the location of this file or turn off
9 // automatically serving support files with the
10 // 'supportFile' configuration option.
11 //
12 // You can read more here:
13 // https://on.cypress.io/configuration
14 // ****
15
16 // Import commands.js using ES2015 syntax:
17 import './commands'
18
19 // Alternatively you can use CommonJS syntax:
20 // require('./commands')
21 require('cypress-xpath')
22
23 Cypress.on('uncaught:exception', (err, runnable) => [
24   // returning false here prevents Cypress from
25   // failing the test
26   return false
27 ])
```



CypressTutorial

Not Secure | https://www.letskodeit.com/_/#specs/runner?file=cypress/e2e/common_challenges/empty_data_demo.cy.js

Chrome is being controlled by automated test software.

Specs

empty_data_demo.cy.js

Providing empty data to input field

TEST BODY

```
1 visit https://www.letskodeit.com/login
2 get #email
3 -type {backspace}
(xhr) GET 200 /show-gdpr-popup
(xhr) GET 200 /complete-registration
(xhr) GET 200 /footer-logo
(xhr) GET 200 /show-card-declined-popup
(xhr) POST 200 https://webtracker.newzenler.com/api/v1/tracking/add
(xhr) POST 200 /get-paypal-attributes
4 get #login
5 -click
(xhr) POST 200 /check-mfa-on
(xhr) GET /login
(uncaught exception) ReferenceError: grecaptcha is not defined
(xhr) GET 200 /get-upcoming-live-classes
```

https://www.letskodeit.com/login

Login

Email Address

The email field is required.

Password

LOGIN

Forgot Password?

Don't have a account? Sign Up

© 2023 letskodeit

Home Courses Support Terms & Conditions Privacy Policy Cookie Policy

DOM Snapshot

The screenshot shows the Cypress Test Runner interface. On the left, the test file `empty_data_demo.cy.js` is open, displaying a single test case: `should not login to app using empty username`. The test body contains the following code:

```
1 visit 'https://www.letskodeit.com/login'
2 get '#email'
3 -type {backspace}
(xhr) GET 200 /show-gdpr-popup
(xhr) GET 200 /complete-registration
(xhr) GET 200 /footer-logo
(xhr) GET 200 /show-card-declined-popup
(xhr) POST 200 https://webtracker.newzenler.com/api/v1/tracking/add
(xhr) POST 200 /get-paypal-attributes
4 get '#login'
5 -click
(xhr) POST 200 /check-mfa-on
(xhr) GET /login
(uncaught exception) ReferenceError: grecaptcha is not defined
(xhr) GET 200 /get-upcoming-live-classes
```

The right side of the screen shows a screenshot of a web browser window for `https://www.letskodeit.com/login`. The page has a "Login" header. It features two input fields: "Email Address" and "Password". Below the fields is a large orange "LOGIN" button. Underneath the button, there are links for "Forgot Password?", "Don't have a account? Sign Up", and social media sharing icons (Facebook, Twitter, LinkedIn, YouTube).

Cypress 25

Monday, March 18, 2024 10:39 PM

```
describe('Hooks Demo', () => {
  // before() Hook
  // It runs before starting the first test, only once before any test starts
  before('This is the before Hook', () => {
    cy.log("Before All Tests")
  })

  // after() Hook
  // It runs after completing all tests, only once after completing all tests
  after('This is the after Hook', () => {
    cy.log("After All Tests")
  })

  // beforeEach() Hook
  // It runs before every test
  beforeEach('This is beforeEach Hook', () => {
    cy.log('Before Every Test')
  })

  // afterEach() Hook
  // It runs after every test
  afterEach('This is afterEach Hook', () => {
    cy.log('After Every Test')
  })

  it('Test 1', () => {
    cy.log('Test 1')
  })

  it('Test 2', () => {
    cy.log('Test 2')
  })
})
```

hooks_demo.cy.js

95ms

Hooks Demo

Test 1

BEFORE ALL

1 log Before All Tests

BEFORE EACH

1 log Before Every Test

TEST BODY

1 log Test 1

AFTER EACH

1 log After Every Test

Test 2

BEFORE EACH

1 log Before Every Test

TEST BODY

1 log Test 2

AFTER EACH

1 log After Every Test

AFTER ALL

1 log After All Tests

Default blank page

This page was cleared by navigating to about:blank.

All active session data (cookies, localStorage and sessionStorage) across all domains was cleared.

DOM Snapshot

Cypress 26

Monday, March 18, 2024 10:51 PM

```
describe('Include/Exclude Demo', () => {
  it.only('Test 1', () => {
    cy.log('Test 1')
  })

  it.skip('Test 2', () => {
    cy.log('Test 2')
  })

  it('Test 3', () => {
    cy.log('Test 3')
  })

  it('Test 4', () => {
    cy.log('Test 4')
  })
})
```

***** Including / Excluding Tests *****
Only -> Runs it() block where it.only() is used
Skip -> Skips it() block where it.skip() is used
If both are used, preference is given to it.only()

Cypress 27

Monday, March 18, 2024 11:01 PM

```
*****
** Fixtures ****
Fixtures provide an easy way to store and read external data for test automation code

describe('Fixtures Demo', () => {
  let globalData;
  before('Before Hook', () => {
    cy.fixture("example").then((data) => {
      globalData = data
    })
  })

  it('should understand how to use fixtures for reading data', () => {
    cy.visit("https://www.letskodeit.com/login");
    cy.get('#email').type(globalData.testid1.username);
    cy.get('#login-password').type(globalData.testid1.password);
  })
}
/*
Create a file, call it example.json, create the file in fixtures folder
-----
{
  "testid1": {
    "username": "Test Username",
    "password": "Test Password"
  },
  "testid2": {
    "course_name": "Cypress"
  }
-----
*/

```

When key is provided in the code, it fetches the values from the JSON file

Cypress 28

Monday, March 18, 2024 11:19 PM

```
describe('Fixtures Multiple Data Demo', () => {
  let globalData;

  before('Before Hook', () => {
    cy.fixture("search_course").then((data) => {
      globalData = data
    })
  })

  it('should run same test with multiple data', () => {
    globalData.testid1.forEach(testData => {
      cy.visit('https://www.letskodeit.com/courses')
      cy.get('input[placeholder="Search Course"]').type(testData.search_course)
      cy.get('button[class="find-course search-course"]').click()
      cy.xpath('//h4[normalize-space()="{course_name}"]'.replace("{course_name}", testData.click_course)).click()
    })
  })
  /*
Create a file, call it search_course.json, create the file in fixtures folder
-----
{
  "testid1": [
    {"search_course": "java", "click_course": "Selenium WebDriver With Java"},  

    {"search_course": "java", "click_course": "Java Step By Step For Testers"}
  ]
}
-----
*/
}
```

Cypress 29

Monday, March 18, 2024 11:35 PM

```
describe('Custom Command Get Text Demo', () => {
  it('should use custom command to get text', () => {
    cy.visit('https://www.letskodeit.com/practice')
    cy.get('#openwindow').then(($element) => {
      const itemText = $element.text()
      expect(itemText).eq('Open Window')
    })
    cy.get('#openwindow').getText().then((elementText) => {
      expect(elementText).eq('Open Window')
      cy.log(elementText.text())
    })
  })
})
//-----In support/commands.ts-----
// --add this code -----
/*
declare namespace Cypress {
  interface Chainable<Subject> {
    getText(elementText: string): Chainable<Subject>;
  }
}
Cypress.Commands.add('getText', {prevSubject: 'element'}, ($element) => {
  cy.wrap($element).scrollIntoView()
  return cy.wrap($element).invoke('text')
})
*/
// *****
// 
// -- This is a parent command --
// Cypress.Commands.add("login", (email, password) => { ... })
// 
// 
// 
// --- This is a child command ---
// Cypress.Commands.add("drag", { prevSubject: 'element' }, (subject, options) => { ... })
// 
// 
```

Parent Command Examples:
`cy.get()`
`cy.visit()`

Child Command Example:
`.click()`

Usage:
`cy.get('#selector').click()`

```
describe('Custom Command Get Text Demo', () => {
  it('should use custom command to get text', () => {
    cy.visit('https://www.letskodeit.com/practice')

    cy.get('#openwindow').then(($element) => {
      const itemText = $element.text()
      expect(itemText).eq('Open Window')
    })
  })
})

Cypress.Commands.add(['getText', {prevSubject: 'element'}, ($element) => {
  cy.wrap($element).scrollIntoView()
  return cy.wrap($element).invoke('text')
}])
```

<---- In support/commands.js

```

1 describe('Custom Command Get Text Demo', () => {
2
3     it('should use custom command to get text', () => {
4         cy.visit('https://www.letskodeit.com/practice')
5
6         cy.get('#openwindow').then(($element) => {
7             const itemText = $element.text()
8             expect(itemText).eq('Open Window')
9         })
10
11     cy.get('#openwindow').getText().then((elementText) => {
12         expect(elementText).eq('Open Window')
13     })
14
15 })
16

```

***** Custom Commands *****

1. Don't make everything a custom command
2. Don't implement multiple functionality in a single custom command
3. Skip test UI steps

The screenshot shows a code editor with a sidebar displaying a project structure and a main editor area showing a TypeScript file.

Project Structure:

- CYPRESTITUTORIAL
 - .vscode
 - cypress
 - downloads
 - e2e
 - assertions_demo
 - async_demo
 - element_interactions
 - elements_list
 - examples
 - framework_preparation
 - custom_command_practical_demo.cy.js
 - custom_get_text_demo.cy.js
 - fixtures_demo.cy.js
 - fixtures_multiple_data_demo.cy.js
 - hooks_demo.cy.js
 - include_exclude_demo.cy.js
 - overview
 - windows_popups_frames
 - fixtures
 - plugins
 - screenshots
 - support
 - commands.js
 - e2e.js
 - index.d.ts
 - videos
 - tsconfig.json
 - node_modules
 - cypress.config.js
 - package-lock.json
 - package.json

Code Snippet (index.d.ts):

```

cypress > support > TS index.d.ts > {} Cypress > • Chainable > ⓘ searchCourse
1 declare namespace Cypress {
2     interface Chainable<Subject> {
3         /**
4          * Log in via UI
5          * @example
6          * cy.login(email: string, password: string)
7          */
8         login(email: string, password: string): Chainable<any>
9
10        /**
11         * Log in via UI
12         * @example
13         * cy.login(email: string, password: string)
14         */
15         searchCourse(category: string, : string): Chainable<any>
16
17        /**
18         * Log in via API
19         * @example
20         * cy.apiLogin()
21         */
22         apiLogin(): Chainable<any>
23
24        /**
25         * Wait for viewer to load
26         * @example
27         * cy.waitForFirstLoad()
28         */
29         waitForFirstLoad(): Chainable<any>
30

```

Below the code editor, there are tabs for PROBLEMS (2), OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab shows the output: GET /login 200 452.525 ms - -

What is a Framework?



A framework defines the teams way of doing things – a ‘Common Standard’.

- Easy to maintain and read
- Data Driven tests
- Meaningful Reporting
- Standard and Consistent Coding
- Encapsulation of UI interactions
- Maximize Code Re-Usability

“Duplication is the primary enemy of a well-designed system.”

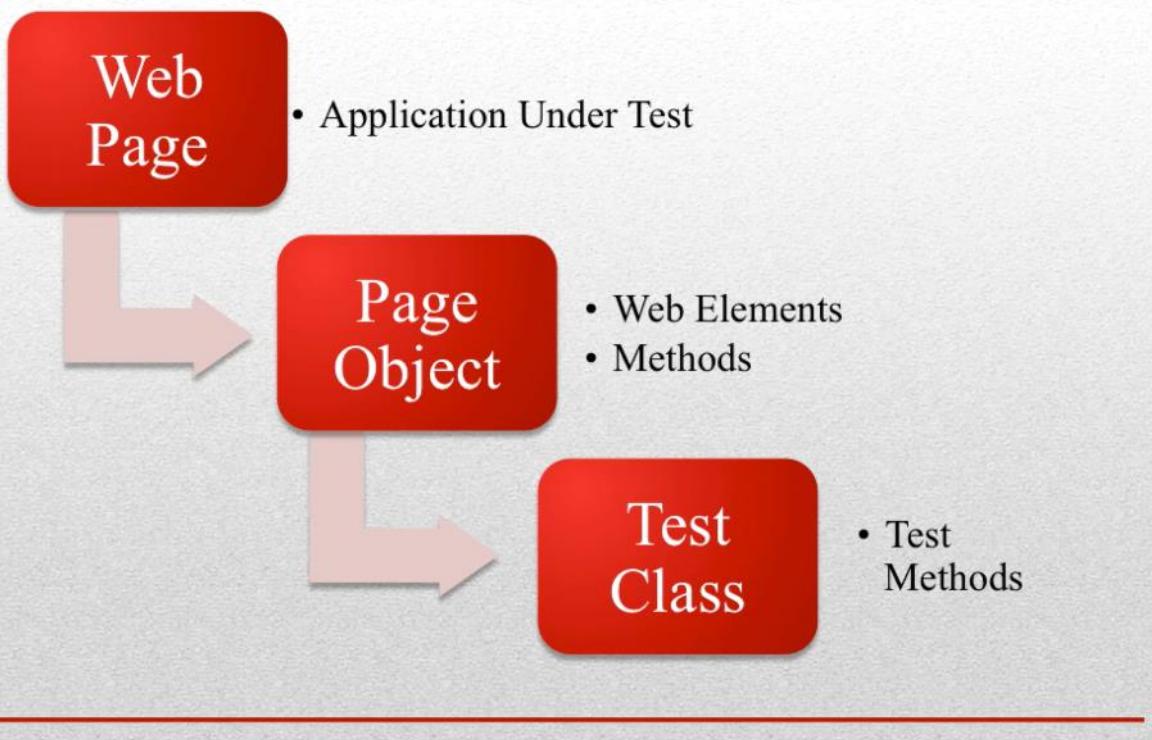
Robert Martin – Clean Code Collection

DRY Principle - Don't Repeat Yourself

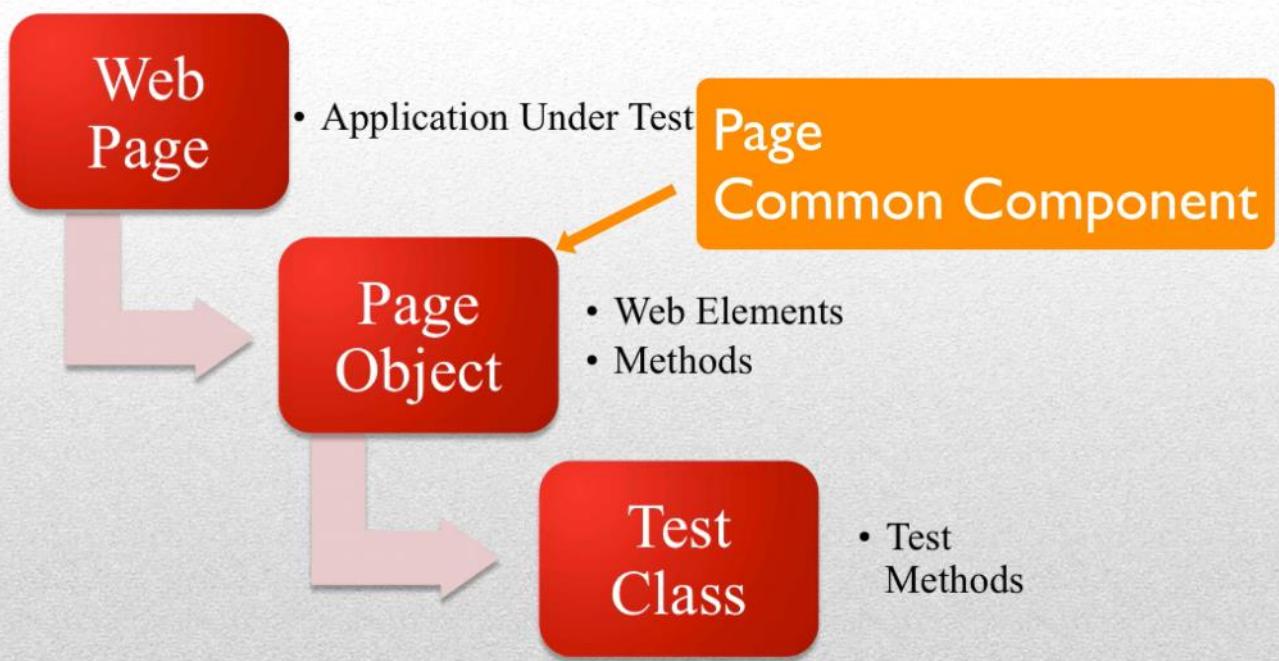
What is a Page Object Model?

- A page object wraps an HTML page with APIs, that allows us to work with page elements.
- A page object should also provide an interface which is easy to program.
- The page object should encapsulate the code used to find and manipulate the elements/data in the page itself.
 - A good rule of thumb is to imagine changing the concrete page - in which case the Test Class shouldn't change.

What is a Page Object Model?



What is a Page Object Model?



Advantage

- **Code reusability** – Write code once and use it in different tests.
 - **Code maintainability** – There is a clean separation between test code and page specific code.
 - **Efficient** – Shorten the learning curve for testers and help QA teams meet timelines.
 - **Readability** – Improves readability due to clean separation between test code and page specific code
 - Very beneficial for beginners and even experienced people joining a new team.
-

What we will use?

Page Object Model + Data Driven

- Hybrid Framework

Advantage:

All advantages of POM

+

Data Driven Testing

Single Test Method
with multiple data inputs

Cypress 32

Tuesday, March 19, 2024 11:25 PM

***** No Framework Issues *****

1. It is too long
2. It has redundant code
3. It is not readable
4. Makes maintenance very difficult, it is not a robust framework

Cypress Excel

Sunday, April 7, 2024 8:46 PM

```
npm install node-xlsx --save-dev
npm install jsonpath
```

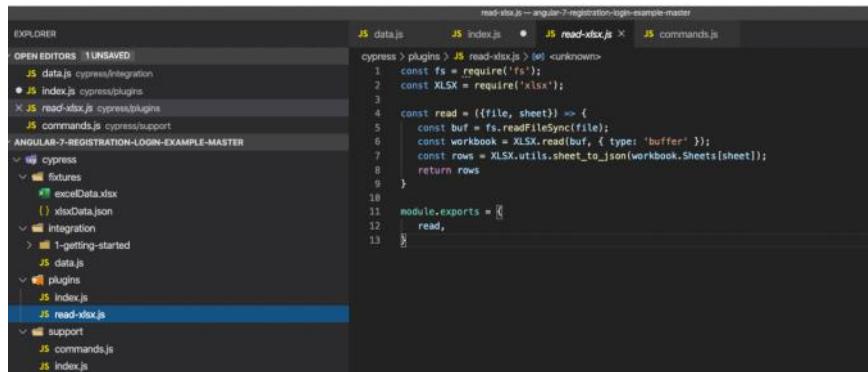
From <<https://kishorsharma69.wordpress.com/2021/07/06/cypress-data-driven-testing-with-excel-file/>>



```
JS data.js      JS index.js  ●  JS commands.js
cypress > plugins > JS index.js > ...
1  /// <reference types="cypress" />
2
3
4  /**
5   * @type {Cypress.PluginConfig}
6   */
7 // eslint-disable-next-line no-unused-vars
8 module.exports = (on, config) => {
9   // 'on' is used to hook into various events Cypress emits
10  // 'config' is the resolved Cypress config
11 }
12
13 const xlsx = require('node-xlsx').default;
14 const fs = require('fs'); // for file
15 const path = require('path'); // for file path
16 module.exports = (on, config) => {
17   on('task', { parseXlsx({ filePath }) =>
18     { return new Promise((resolve, reject) =>
19       { try
20         {
21           const jsonData = xlsx.parse(fs.readFileSync(filePath));
22           resolve(jsonData);
23         } catch (e)
24         {
25           reject(e);
26         }
27       });
28     });
29   });
30   module.exports = (on, config) => {
31     on('task', {
32       'readXlsx': readXlsx.read
33     })
34   }
35 }
```

cypress/plugins/index.js:

```
const xlsx = require("node-xlsx").default;
const fs = require("fs");
const path = require("path");
module.exports = (on, config) => {
  on('task', { parseXlsx({filePath}) =>
    { return new Promise((resolve, reject) =>
      { try
        {
          const jsonData = xlsx.parse(fs.readFileSync(filePath));
          resolve(jsonData);
        } catch(e) {
          reject(e);
        }
      })
    })
  });
  const readXlsx = require("./read-xlsx");
  module.exports = (on, config) => {
    on('task', {
      'readXlsx': readXlsx.read
    })
  }
}
```



```
JS data.js      JS index.js  ●  JS read-xlsx.js X  JS commands.js
cypress > plugins > JS read-xlsx.js > (6) <unknown>
1  const fs = require('fs');
2  const XLSX = require('xlsx');
3
4  const read = ({file, sheet}) => {
5    const buf = fs.readFileSync(file);
6    const workbook = XLSX.read(buf, { type: 'buffer' });
7    const rows = XLSX.utils.sheet_to_json(workbook.Sheets[sheet]);
8    return rows
9  }
10
11 module.exports = [
12   read,
13 ]
```

cypress/plugins/read-xlsx.js:

```
const xlsx = require("node-xlsx").default;
const fs = require("fs");
const read = ({file, sheet}) => {
  const buf = fs.readFileSync(file);
  const workbook = xlsx.read(buf, { type: 'buffer' });
  const rows = xlsx.utils.sheet_to_json(workbook, Sheets[sheet]);
  return rows
}
module.exports = {
  read,
}
```

The screenshot shows a code editor interface with two main panes. The left pane is the 'EXPLORER' view, displaying the file structure of an Angular application named 'ANGULAR-7-REGISTRATION-LOGIN-EXAMPLE-MASTER'. The right pane is the 'commands.js' file, which is currently open.

EXPLORER View:

- OPEN EDITORS: 1 UNSAVED
- JS data.js cypress/integration
- JS index.js
- JS read-xlsx.js
- XJS commands.js cypress/support
- ANGULAR-7-REGISTRATION-LOGIN-EXAMPLE-MASTER
- cypress
 - fixtures
 - excelData.xlsx
 - xlsxData.json
 - Integration
 - 1-getting-started
 - JS data.js
 - plugins
 - JS index.js
 - JS read-xlsx.js
 - support
 - JS commands.js
 - JS index.js
- node_modules
- src
 - .gitignore
 - cypress.json
 - LICENSE
 - package-lock.json
 - package.json
 - README.md
 - tsconfig.json
 - webpack.config.js

```
commands.js — angular-7-registration-login-example-master

JS data.js      JS index.js  ● JS read-xlsx.js  JS commands.js ×

cypress > support > JS commands.js > ...
1 // ****
2 // This example commands.js shows you how to
3 // create various custom commands and overwrite
4 // existing commands.
5 //
6 // For more comprehensive examples of custom
7 // commands please read more here:
8 // https://on.cypress.io/custom-commands
9 // ****
10 //
11 //
12 // -- This is a parent command --
13 // Cypress.Commands.add('login', (email, password) => { ... })
14 //
15 //
16 // -- This is a child command --
17 // Cypress.Commands.add('drag', { prevSubject: 'element'}, {subject, options} =
18 //
19 //
20 // -- This is a dual command --
21 // Cypress.Commands.add('dismiss', { prevSubject: 'optional'}, {subject, option
22 //
23 //
24 // -- This will overwrite an existing command --
25 // Cypress.Commands.overwrite('visit', (originalFn, url, options) => { ... })
26 //
27 Cypress.Commands.add("parseXlsx", (inputFile) => {
28   return cy.task('parseXlsx', { filePath: inputFile });
29 });
30 
```

Control table

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer (Left):** Shows a tree view of a project structure under "CSR7-CYPRESS". The "arrangementWizard.cy.js" file is selected.
- Code Editor (Center):** Displays a portion of the "arrangementWizard.cy.js" file containing Cypress commands for interacting with an arrangement wizard. The code includes assertions for accessibility and navigation steps.
- Bottom Status Bar:** Shows the file path as "cypress/e2e/arrangementWizard.cy.js", the current line as "Ln 43, Col 39 (30 selected)", and the status message "Share this window watch failed".

Accessibility Testing

Monday, March 11, 2024 2:57 PM

The screenshot displays a web browser window with two main panels. The left panel shows a 'Testing for Accessibility' section with a list of items:

- Screen Readers – visual impairments. NVDA free screen reader tool
- Keyboard Navigation
 - o Tab navigation
 - o Focus styles
 - o Perform actions with keyboard
- Color Contrast – low vision or color blindness
- Forms and Labels – Proper labelling and structure. Associate labels to form elements

The right panel shows 'Tools available to use:' with a list of tools:

- Chrome Lighthouse (Chrome extension)
- WAVE Evaluation Tool (Chrome extension)
- Axe DevTools (Chrome extension)
- NVDA (Screen Reader)
- JAWS ("Job Access With Speech")

Below these panels is a screenshot of a bill summary page from a utility company. The page includes sections for 'Your Next Bill' and 'Your Last Bill', showing usage details like 'Electric Use So Far' (547 kWh) and 'Gas Use So Far' (1,469.00 Therms). A note states: 'As of 33 days into your billing cycle.' The page also includes a 'View My Usage >' link and an 'Important' section with usage notes.

Insights | What is the difference between | WebAIM Articles | Files for accessibility website - | +

The following apply to the entire page.

powered by WebAIM

WAVE web accessibility evaluation tool

Styles: OFF ON

Contrast

Summary Details Reference Order Structure Contrast

Click a Contrast icon below or within the web page to view details.

Foreground Color: #474747 Background Color: #FFFFFF Lightness: 100% Lightness: 100%

Contrast Ratio: 9.29:1

Text Size: Normal Sample WCAG AA: Pass WCAG AAA: Pass

Desaturate page

WCAG does not detect contrast of background gradients, transparency, etc. For background images, WCAG requires a fallback background color in case the image does not display.

triangle image for button

CDS Demo

ABOUT US PRODUCTS ABOUT CONTACT

Dashboard My Usage Ways to Save Subscriptions

Account # 300ENetG JAN DOE 2169 Green St SAN ANTONIO, TX 78250 Electric + Gas

Very low contrast (2.35:1)
Very low contrast between text and background colors.
REFERENCE CODE

Usage Threshold Exceeded
You have exceeded the threshold of 100 CCF that you established for your Gas meter 300ENetG_G
View Usage

Usage Threshold Exceeded
You have exceeded the threshold of 100 kWh that you established for your Electric meter 300ENetG_E
View Usage

Your Next Bill Your Last Bill As of 33 days into your billing cycle

Code Notifications >

7:59 AM 2023-12-06 ENG US