Cypress

Friday, March 1, 2024 7:12 PM

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

https://code.visualstudio.com/download

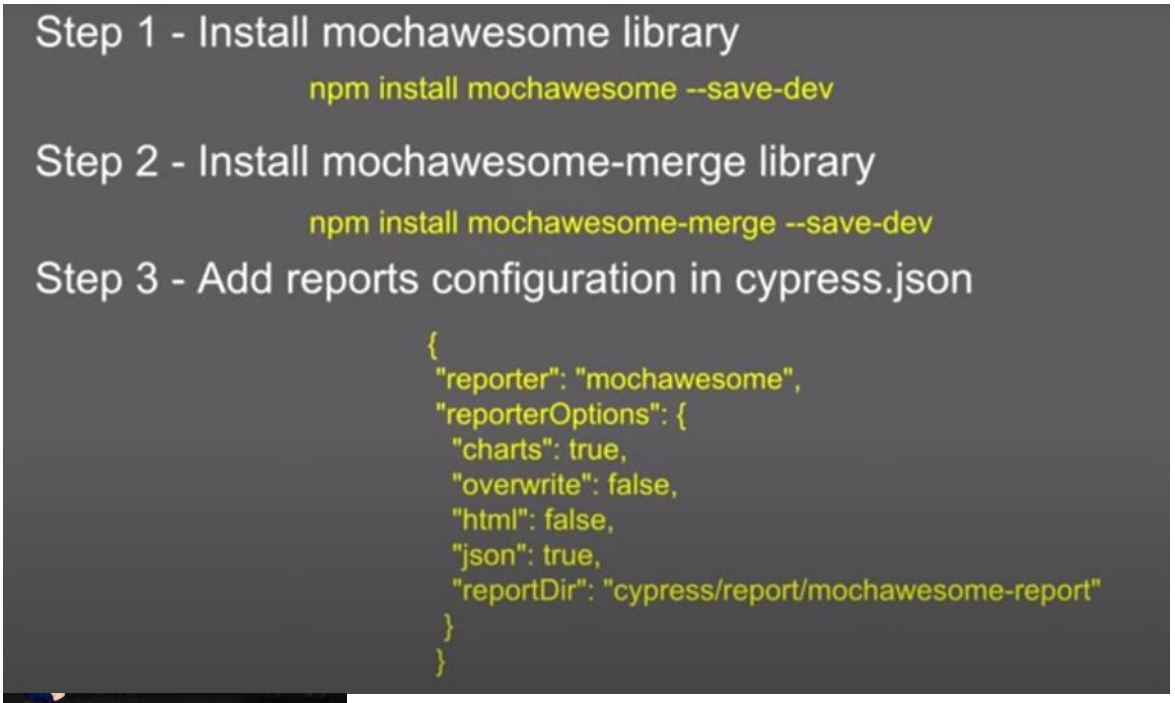
Step 1:
1. Check system requirements
2. Install Node.js
3. Install your preferred IDE - Visual Studio Code, JetBrains Aqua, Eclipse

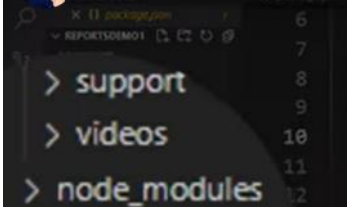
Step 2:
Cypress installation process
a. npm init
b. npm install cypress --save-dev
c. npx cypress open

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

https://learn.cypress.io/

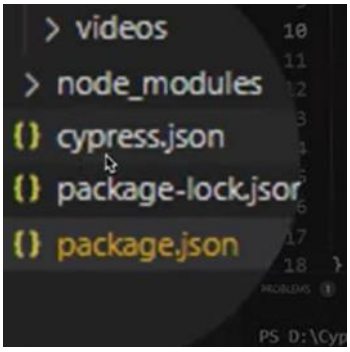
```
npm init
npm install cypress --save-dev
npm i --save-dev @types/mocha
npx cypress open
npm i --save-development axe-core cypress-axe
npx cypress run --spec "cypress/e2e/MyLink/LinkAppAccessibilityCheck.cy.ts" --browser chrome  --headless
npm i -D axe-html-reporter

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





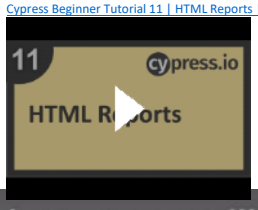
Add these lines in **cypress.config.js**

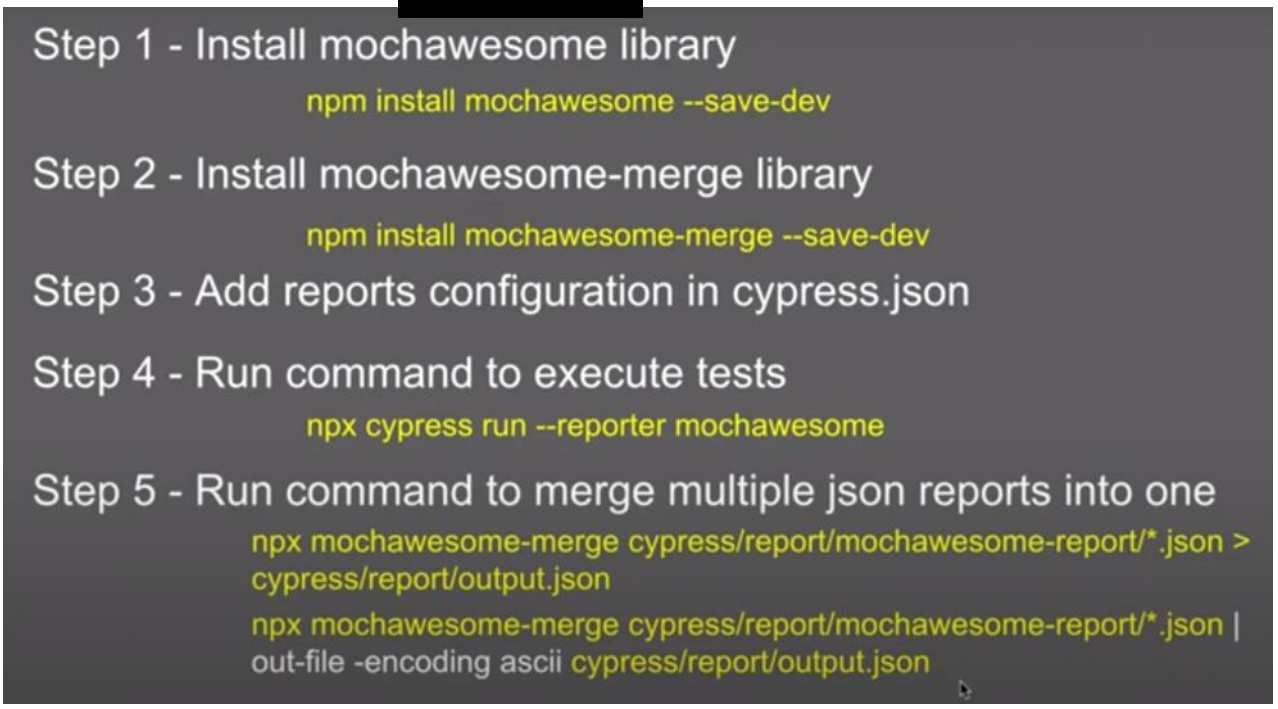
```
{
    "reporter": "mochawesome",
    "reporterOptions": {
        "charts": true,
        "overwrite": false,
        "html": false,
        "json": true,
```




```
    "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



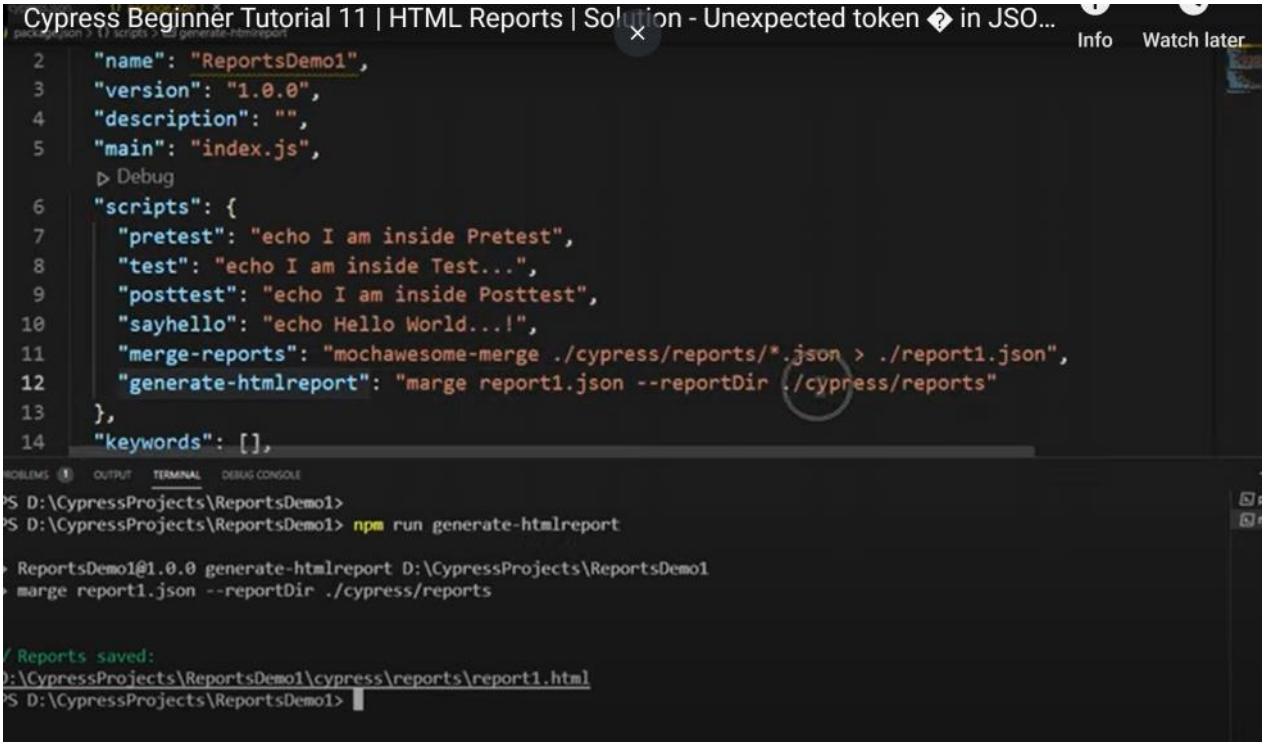



```
npx cypress run --reporter mochawesome
npx mochawesome-merge cypress/reports/*.json -o cypress/reports/output.json
npx mochawesome-merge AccessibilityReports/mochawesome/*.json -o AccessibilityReports/mochawesome/output.json
```

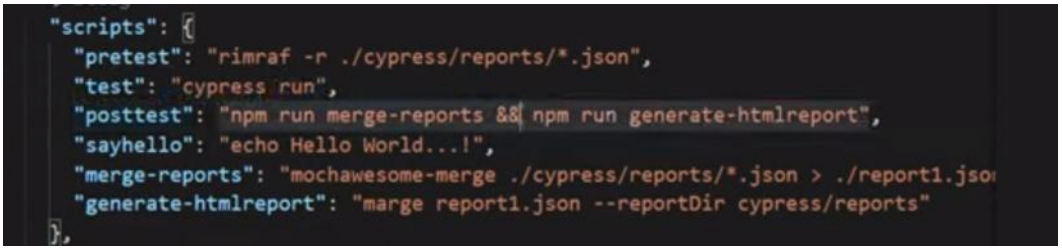
```
npx mochawesome-merge cypress/reports/*.json | out-file -encoding ascii cypress/reports/output.json
```

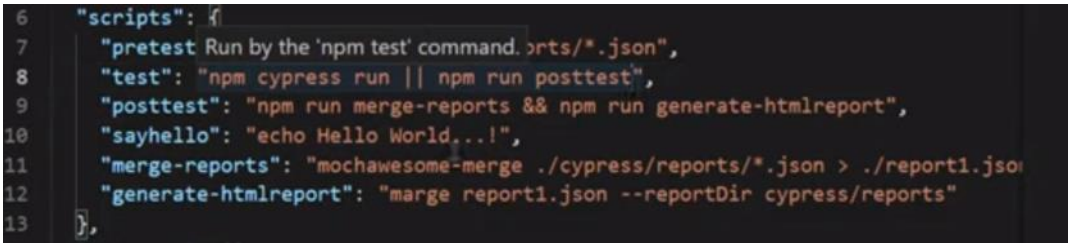

```
npx marge cypress/reports/output.json --reportDir ./ --inline
npx marge AccessibilityReports/mochawesome/output.json --reportDir ./ --inline
```

package.json:



npm i rimraf





npm i -D axe-html-reporter

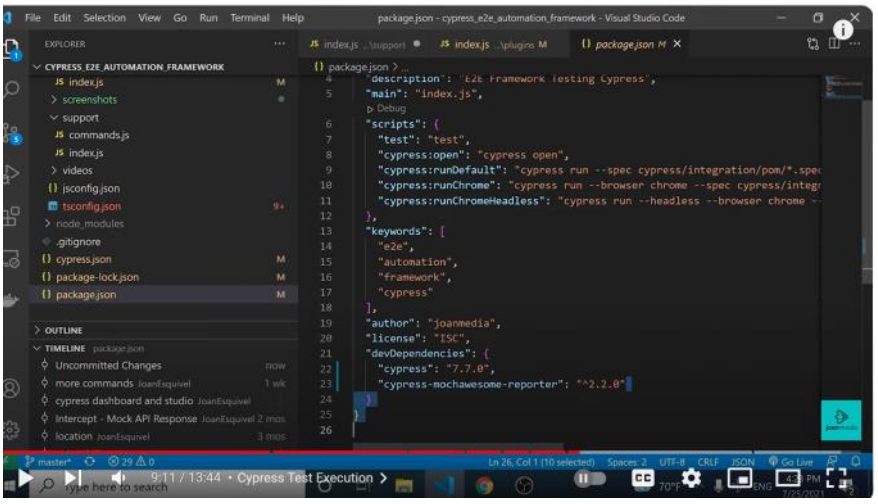
From <https://github.com/lpelypenko/axe-html-reporter?tab=readme-ov-file>

```
    "hello": "echo hello...",
    "script1": "mochawesome-merge ./cypress/report/mochawesome/*.json > report3.json",
    "script2": "marge report3.json --reportDir ./html --inline",
    "script3": "npm run script1 & npm run script2",
    "pretest": "rimraf -r ./cypress/report/mochawesome/*.json",
    "test": "npm run run-cypress-test || npm run posttest",
    "run-cypress-test": "cypsress run",
    "posttest": "npm run merge-reports && npm run generate-htmlreport",
    "merge-reports": "mochawesome-merge ./cypress/report/mochawesome/*.json > -encoding ascii ./cypr
    "generate-htmlreport": "marge ./cypress/report/mochawesome/report.json --reportDir ./cypress/rep
  },



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

```
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.jso
  "generate-htmlreport": "marge report1.json --reportDir cypress/reports"
},
"keywords": [],
"author": "",
```

MOCHAWESOME HTML Report in CYPRESS (EASY INTEGRATION)

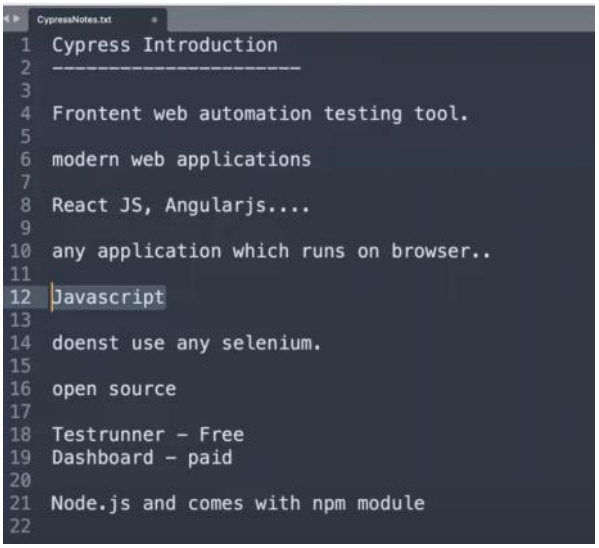
https://www.npmjs.com/package/cypress-mochawesome-reporter




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

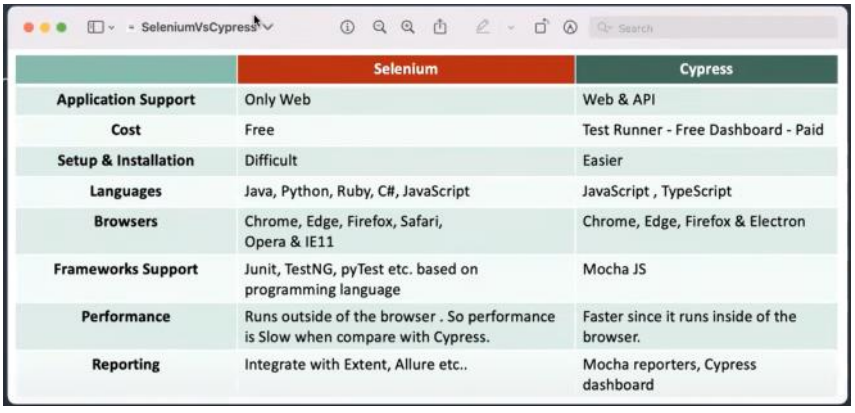
Cypress 1

Friday, March 8, 2024 1:44 PM



Frontend web automation testing tool.

Code is written in JavaScript or TypeScript



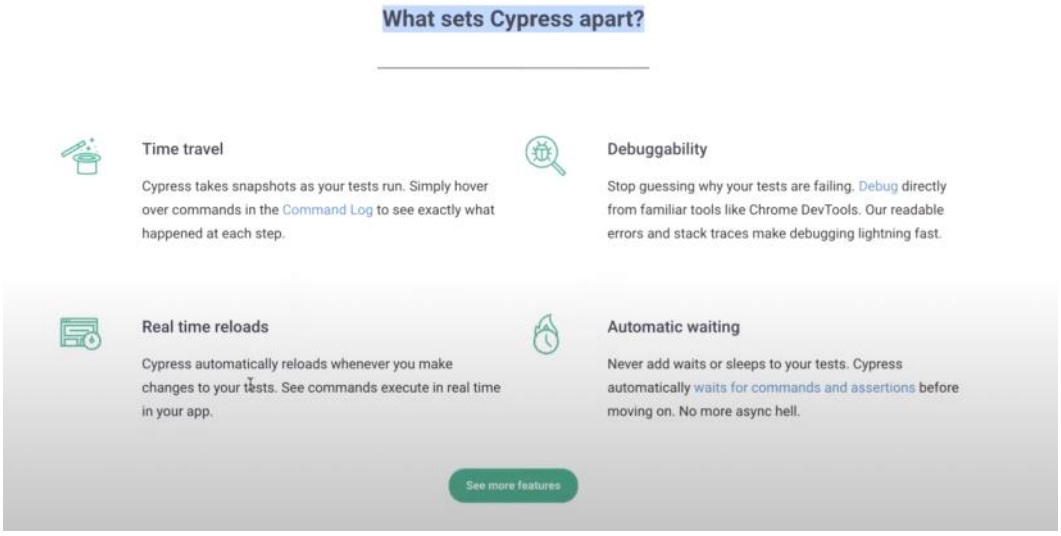
SeleniumVsCypress

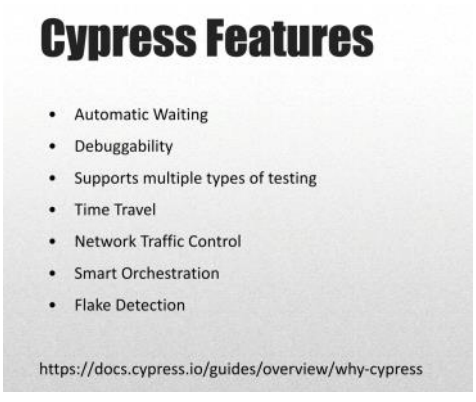
	Selenium	Cypress
Application Support	Only Web	Web & API
Cost	Free	Test Runner - Free Dashboard - Paid
Setup & Installation	Difficult	Easier
Languages	Java, Python, Ruby, C#, JavaScript	JavaScript , TypeScript
Browsers	Chrome, Edge, Firefox, Safari, Opera & IE11	Chrome, Edge, Firefox & Electron
Frameworks 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

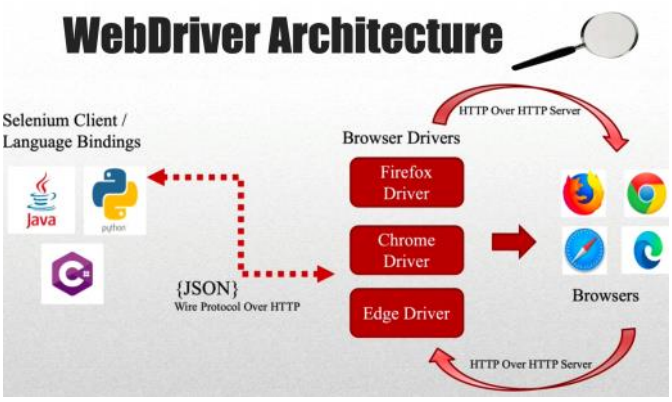
```
Cypress Eco system
------
1) TestRunner  --- open source. locally installed
2) Dashboard - paid


Features
--------

1) Time travel
2) Debuggability
3) Automatic waits (built-in waits)
4) consistence results
5) screenshots & videos
6) Cross browser testing - locally or remotely


limitations
--------
1) can't automation window based/ Mobile apps
2) Limited supports
3) Java Script/Typescript
4) reading/ writing data into file is difficults
5) Third party reporting tool integrtion is alo limited...

```

Limited support to # of browsers

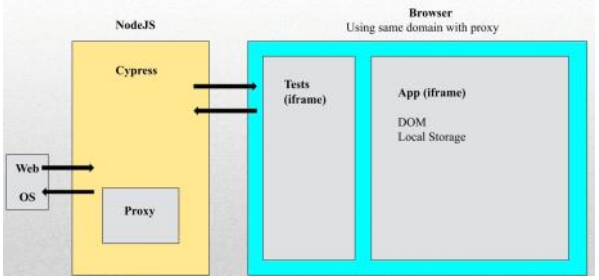
https://www.cypress.io/







Cypress Architecture



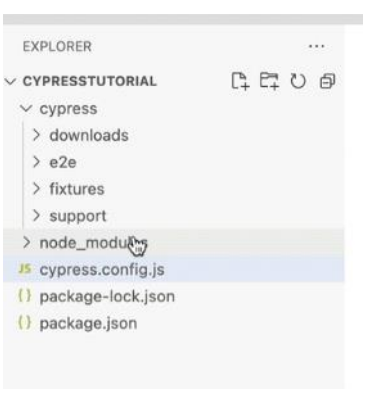
Cypress vs Selenium

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




```
********** Notes **********

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

Java/Python:

Test Class:
- Test Method 1
- Test Method 2
- ...
- ...

cypress > integration > overview > JS demo_spec.js > describe('My First Test') callback > it('Does not do much!') callback

```
describe('My First Test', () => {
    it('Does not do much!', () => {
      expect(true).to.equal(true)
    })
  })
```

Expect comes from Chai Framework

```
******************************** First Test ********************************

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

Mocha and Chai are Testing Trameworks 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

Cypress 2

Saturday, March 9, 2024 11:33 PM

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.json | intellisense_spec.js

cypress > integration > overview > intellisense_spec.js > ...

```
1  /// <reference types="Cypress" />
2
3  describe('Working with intellisense', () => {
4      it('Cypress intellisense should work', () => {
5        cy.visit()
6      })
7    })
```

```
****************************** 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>
    }
  }
```

cypress.json | intellisense_spec.js | index.d.ts

cypress > support > index.d.ts > {} Cypress > Chainable

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

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

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

Cypress 3

Sunday, March 10, 2024 12:21 AM

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
```

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

```
describe('Get Method and CSS Examples', () => {

    it('should learn get() method and CSS examples',  () => {
        cy.visit('https://courses.letskodeit.com/practice')

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

        //Id  #
        cy.get('#name')

        //Class Name . dot
        cy.get('.inputs')

        //Attribute value
        cy.get("[placeholder='Enter your name']")

        //Attribute value
        cy.get("[class='inputs displayed-class']")

        //Tag Name and Attribute value
        cy.get("input[id='name']")


        //Tag Name and Attribute value
        cy.get("input[id='name']:visible")

        //Tag Name and Multiple Attribute values
        cy.get("input[id='name'][placeholder='Enter your name']")

    })

})
```

Use a combination of attriubutes 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

Cypress 4

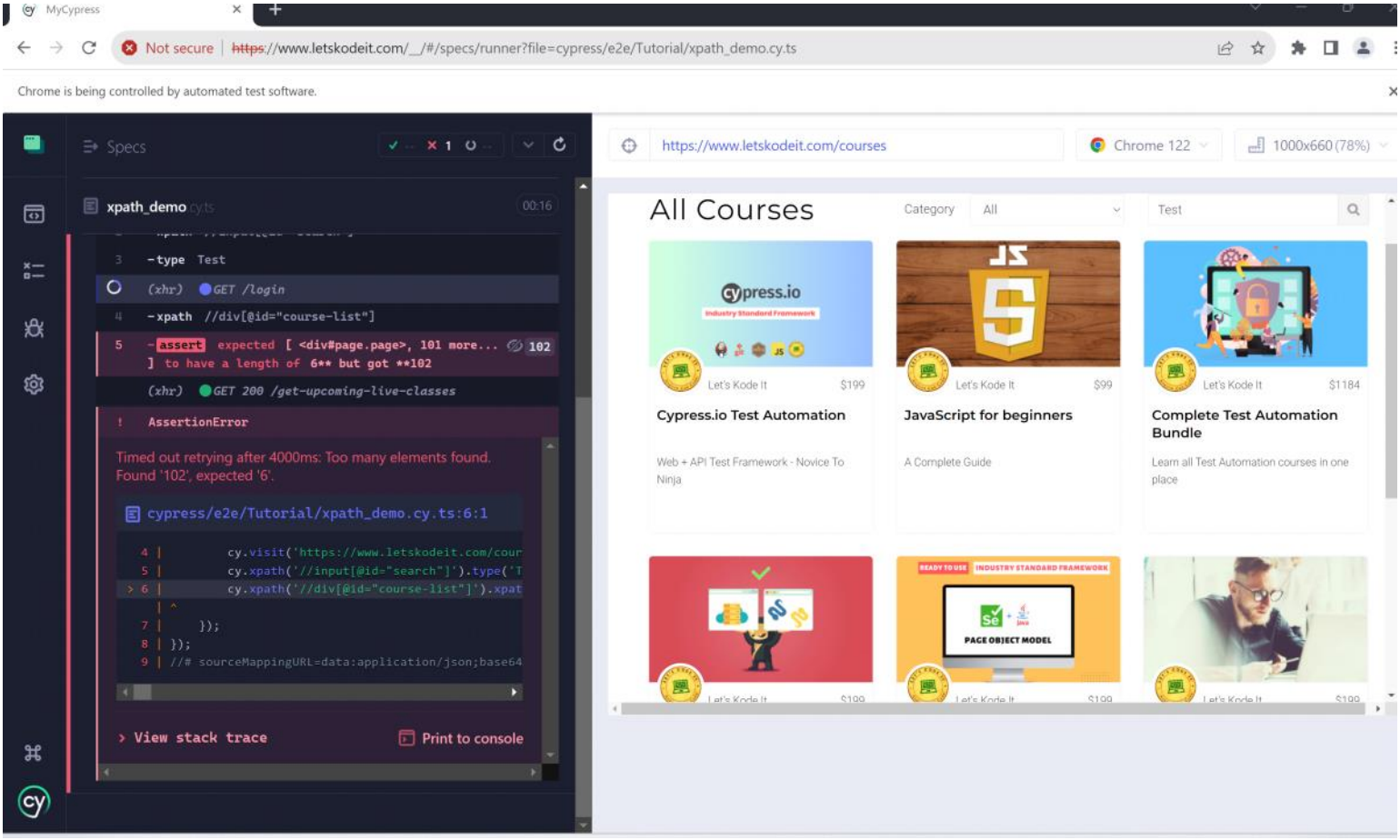
Sunday, March 10, 2024 11:26 PM

```
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

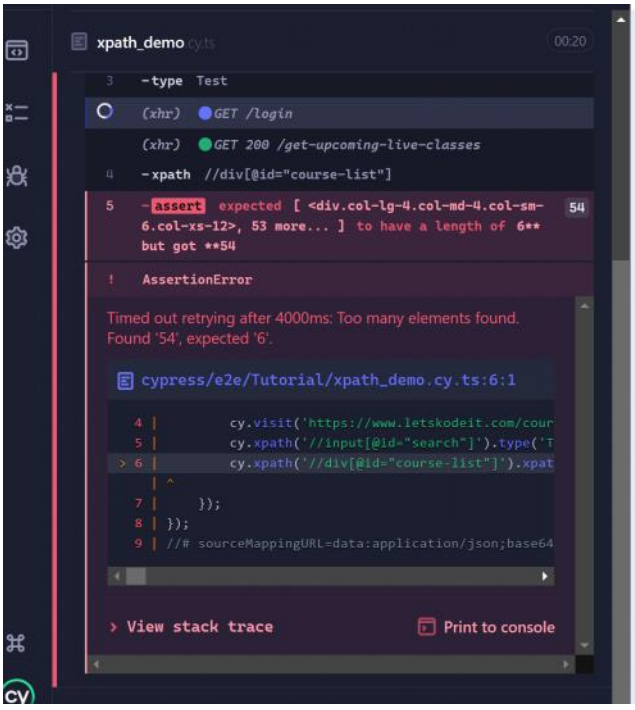


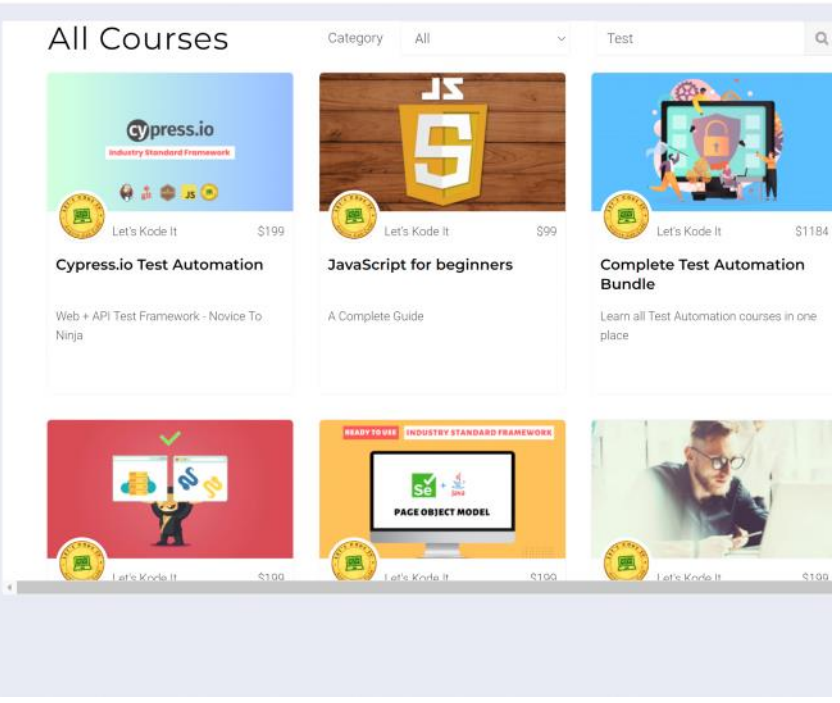
// means anywhere in the document (DOM)
// does not mean anywhere in the current context (node)

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






```
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

Cypress automatically retries to find elements until timeout

Cypress 6

Tuesday, March 12, 2024 3:14 PM

Chaining

cypress > integration > overview > JS chain_commands_demo_spec.js > describe('Chain Commands Demo ') calll

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

      cy.get('button').eq(0).should('contain', 'Open Window')

    })
})
```

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

cypress > e2e > Tutorial > TS chain_demo.cyc.ts > ...

```
describe('Chain command - demo', () => {

    it('should learn chain command details',  () => {
        cy.visit('https://www.letskodeit.com/practice')

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

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

    })
})
```

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.Cypress.Loggable & Cypress.Timeoutable & Cypress.Withinable & Cypress.Shadow>):
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

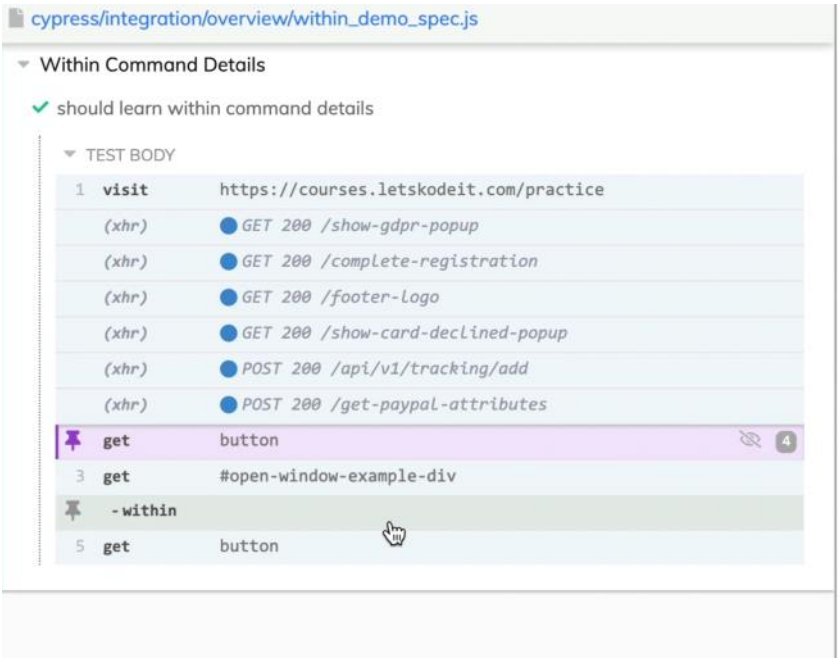
```
cy.get('.list>li')    // Yield the <li>'s in <.list>
cy.get('ul li:first').should('have.class', 'active')
cy.get('.dropdown-menu').click()
```

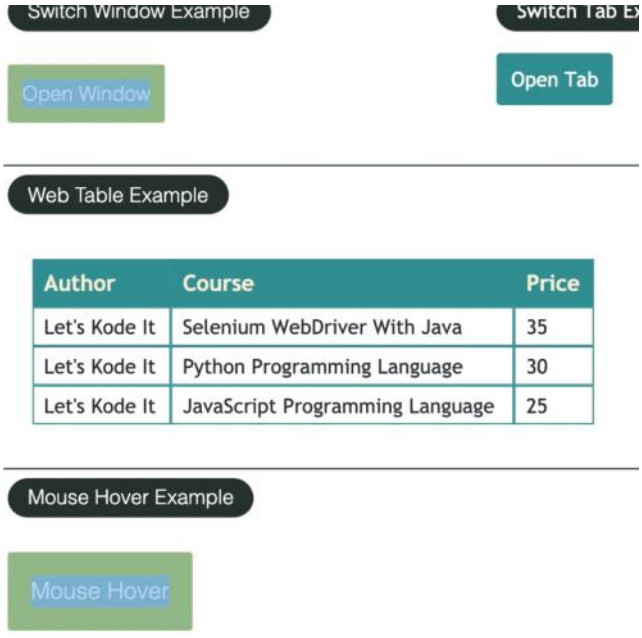
Cypress 7

Tuesday, March 12, 2024 4:05 PM

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

```ts
describe('Within command - demo', () => {

    it('should learn within command details',  () => {
        cy.visit('https://www.letskodeit.com/practice')

        cy.get('button')

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


    })
})
```





********** 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')
})
```

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

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

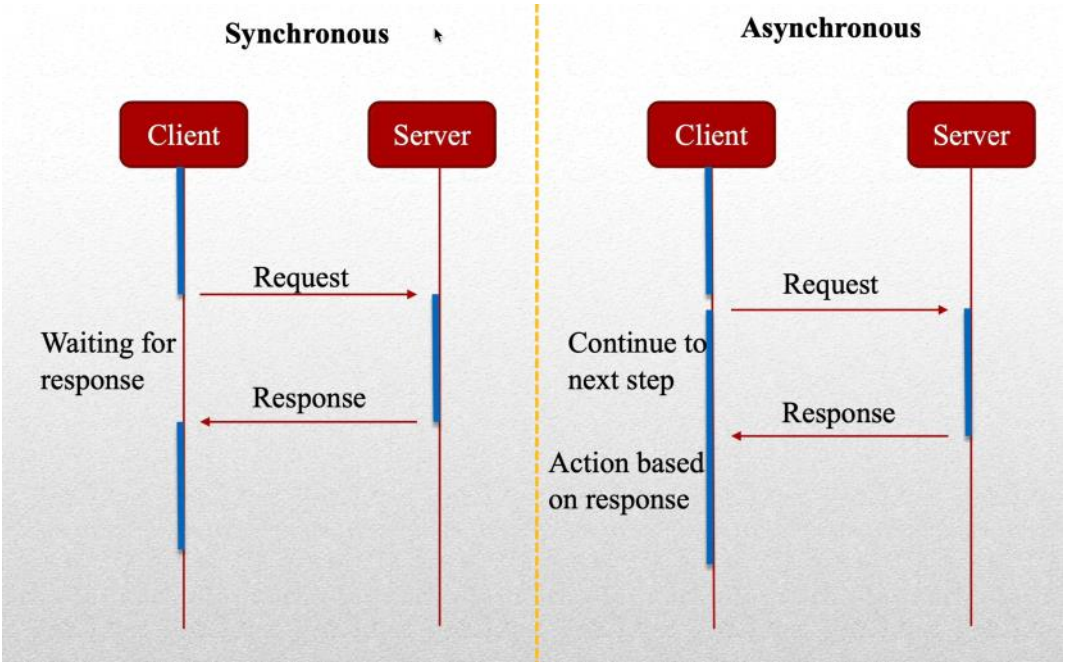
cy.get(locator, options).find(locator, options)

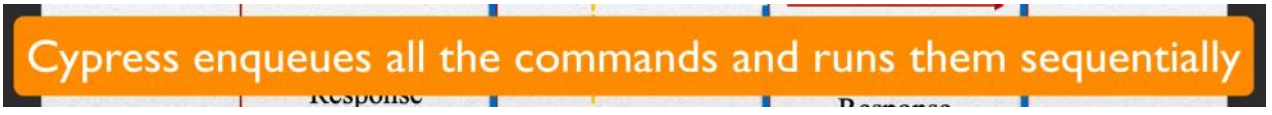
```
Option        -> Default Value  -> Description
log           -> true           -> Enable/Disable command output on the console
timeout       -> defaultTimeout -> Time to wait before throwing an exception
```

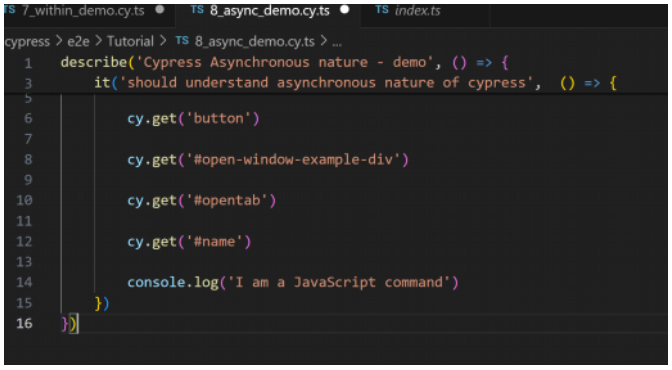
\

Cypress 8

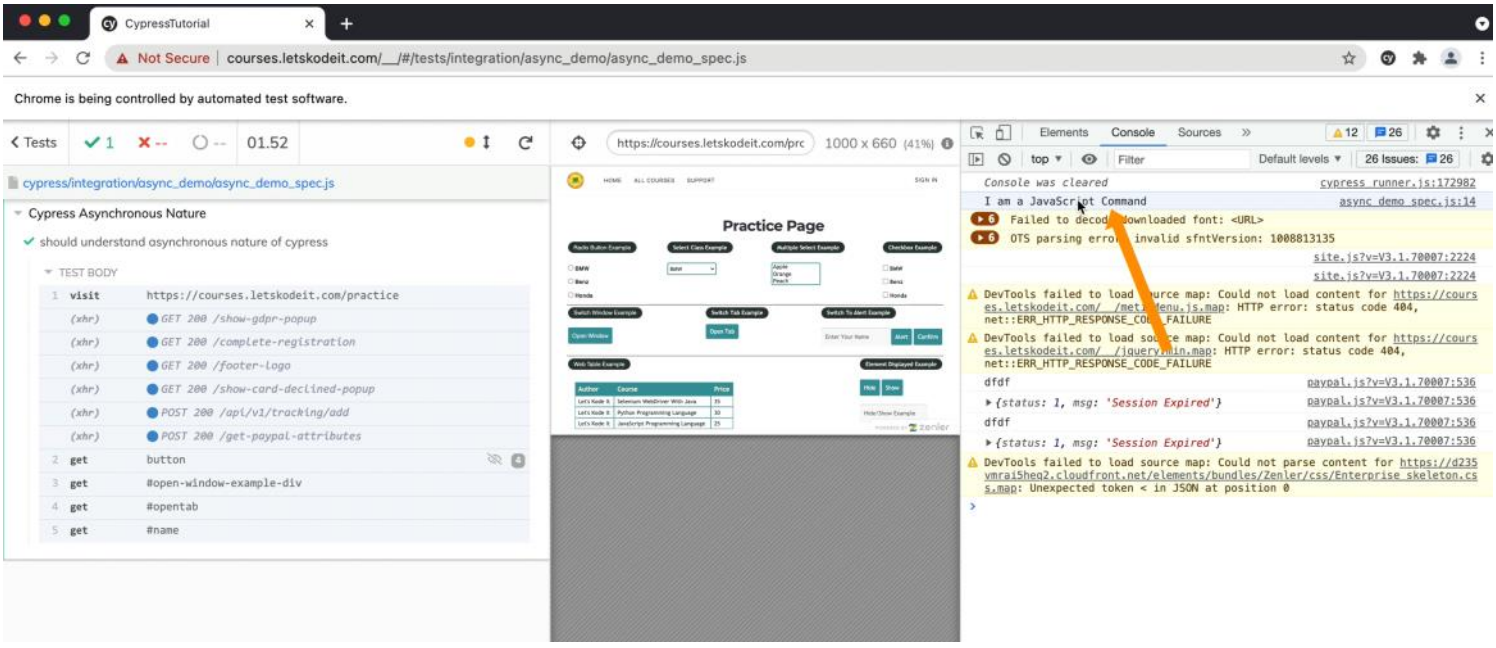
Tuesday, March 12, 2024 11:32 PM







```
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
6           cy.get('button')
7
8           cy.get('#open-window-example-div')
9
10          cy.get('#opentab')
11
12          cy.get('#name')
13
14          console.log('I am a JavaScript command')
15      })
16  })
```


CypressTutorial
Not Secure | courses.letskodeit.com/__/#/tests/integration/async_demo/async_demo_spec.js
Chrome is being controlled by automated test software.
Tests
1
--
--
01.52
cypress/integration/async_demo/async_demo_spec.js
Cypress Asynchronous Nature
should understand asynchronous nature of cypress
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-paypal-attributes
2 get button
3 get #open-window-example-div
4 get #opentab
5 get #name
https://courses.letskodeit.com/prc
1000 x 660 (41%)
Practice Page
Elements
Console
Sources
Filter
Default levels
26 Issues: 26
Console was cleared
I am a JavaScript Command
Failed to decode downloaded font: <URL>
OTS parsing error: invalid sfntVersion: 1008813135
DevTools failed to load source map: Could not load content for https://courses.letskodeit.com/__/metrics-menu.js.map: HTTP error: status code 404, net::ERR_HTTP_RESPONSE_CODE_FAILURE
DevTools failed to load source map: Could not load content for https://courses.letskodeit.com/__/jquery.min.map: HTTP error: status code 404, net::ERR_HTTP_RESPONSE_CODE_FAILURE
dfdf
{status: 1, msg: 'Session Expired'}
DevTools failed to load source map: Could not parse content for https://d235ymra15heq2.cloudfront.net/elements/bundles/Zenler/css/Enterprise_skeleton.css.map: Unexpected token < in JSON at position 0

Cypress 9

Tuesday, March 12, 2024 11:52 PM


```
********** 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')
})
```

Cypress 10

Wednesday, March 13, 2024 12:07 AM

********** 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

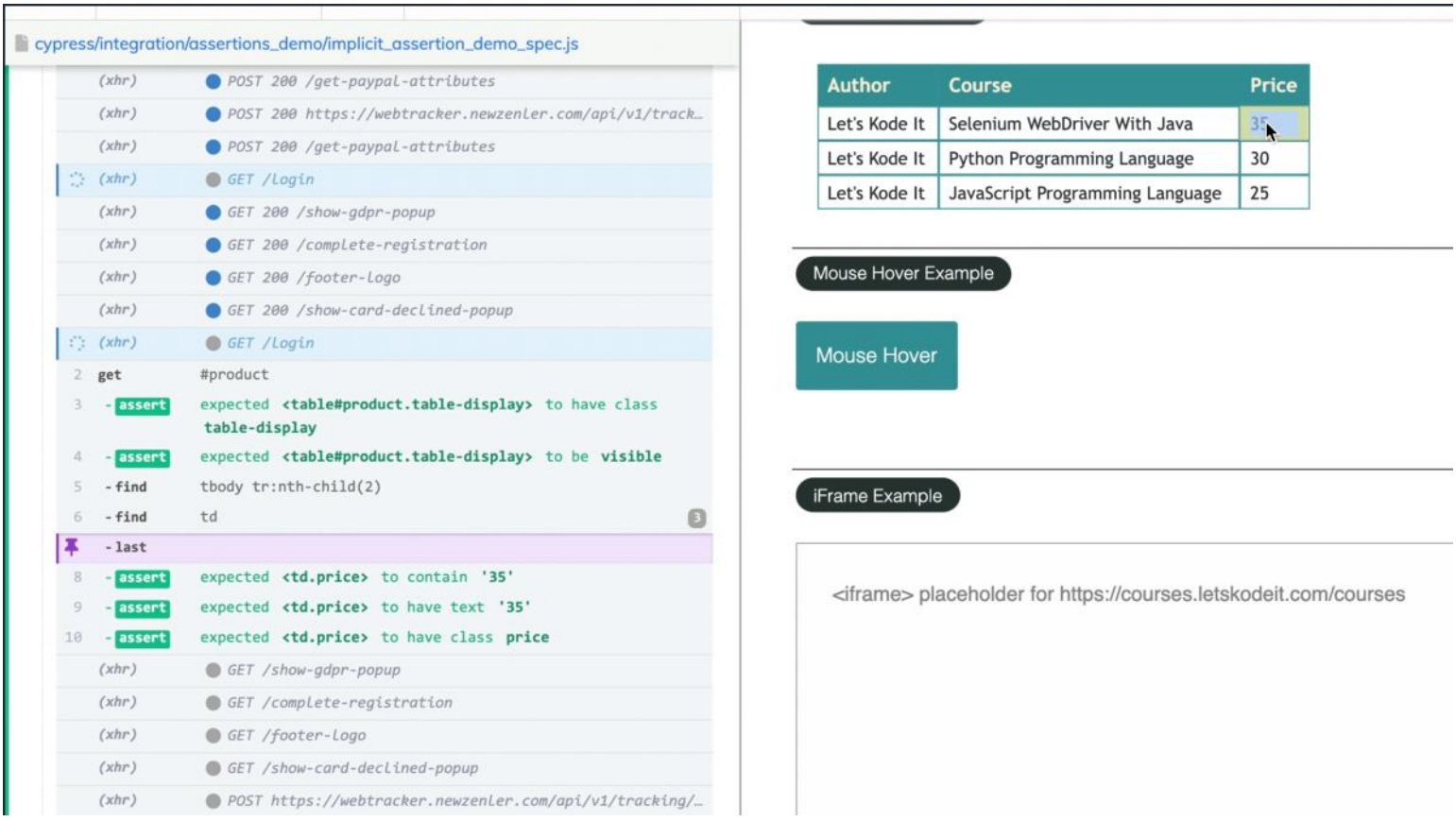
Implicit: Assertions used on the object provided by the parent command.

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

cypress > e2e > Tutorial > TS 10_assertions_demo.cy.ts > ...

```ts
describe('Cypress implicit assertions - demo', () => {

    it('should understand implicit assertions of cypress',  () => {
        cy.visit('https://www.letskodeit.com/practice')

        cy.get('#product')
        .should('have.class', 'table-display')
        .and('be.visible')
        .find('tbody tr:nth-child(2)')
        .find('td')
        .last()
        .should('contain', '35')
        .and('have.text', '35')
        .and('have.class', 'price')


    })
})
```



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

********** 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

Cypress 11

Wednesday, March 13, 2024 5:28 PM

********** 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 **********

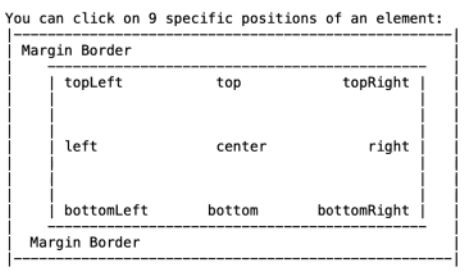
Syntax:

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

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
)le/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)
```

Cypress 13

Thursday, March 14, 2024 9:36 AM

```
****************** 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

Cypress 14

Thursday, March 14, 2024 10:01 AM

```
****************** 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
```

Cypress 15

Thursday, March 14, 2024 2:35 PM

```
****************** 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
```

Cypress 17

Saturday, March 16, 2024 10:47 PM

```
****************** 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
```

Cypress 18

Saturday, March 16, 2024 10:58 PM

```
****************** 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
```

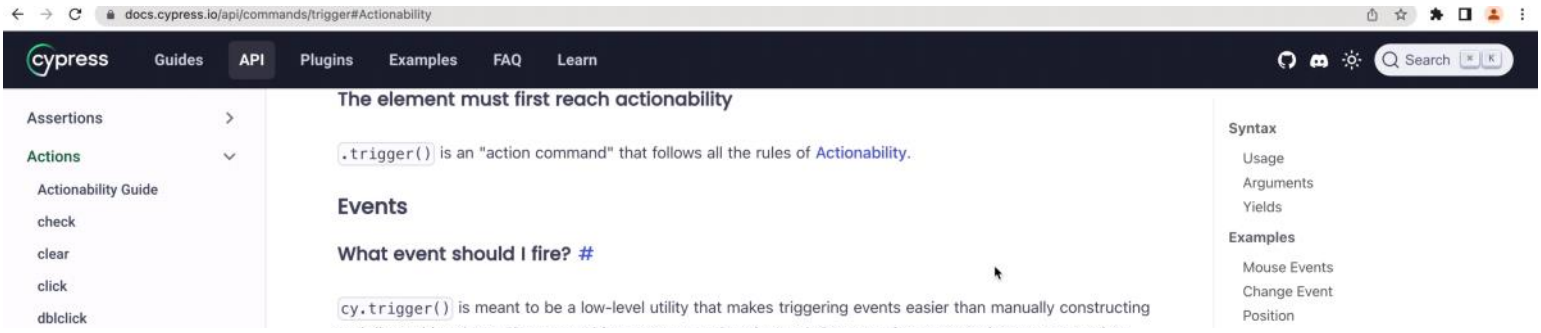
First understand the way your application is implemented, then work on automation code

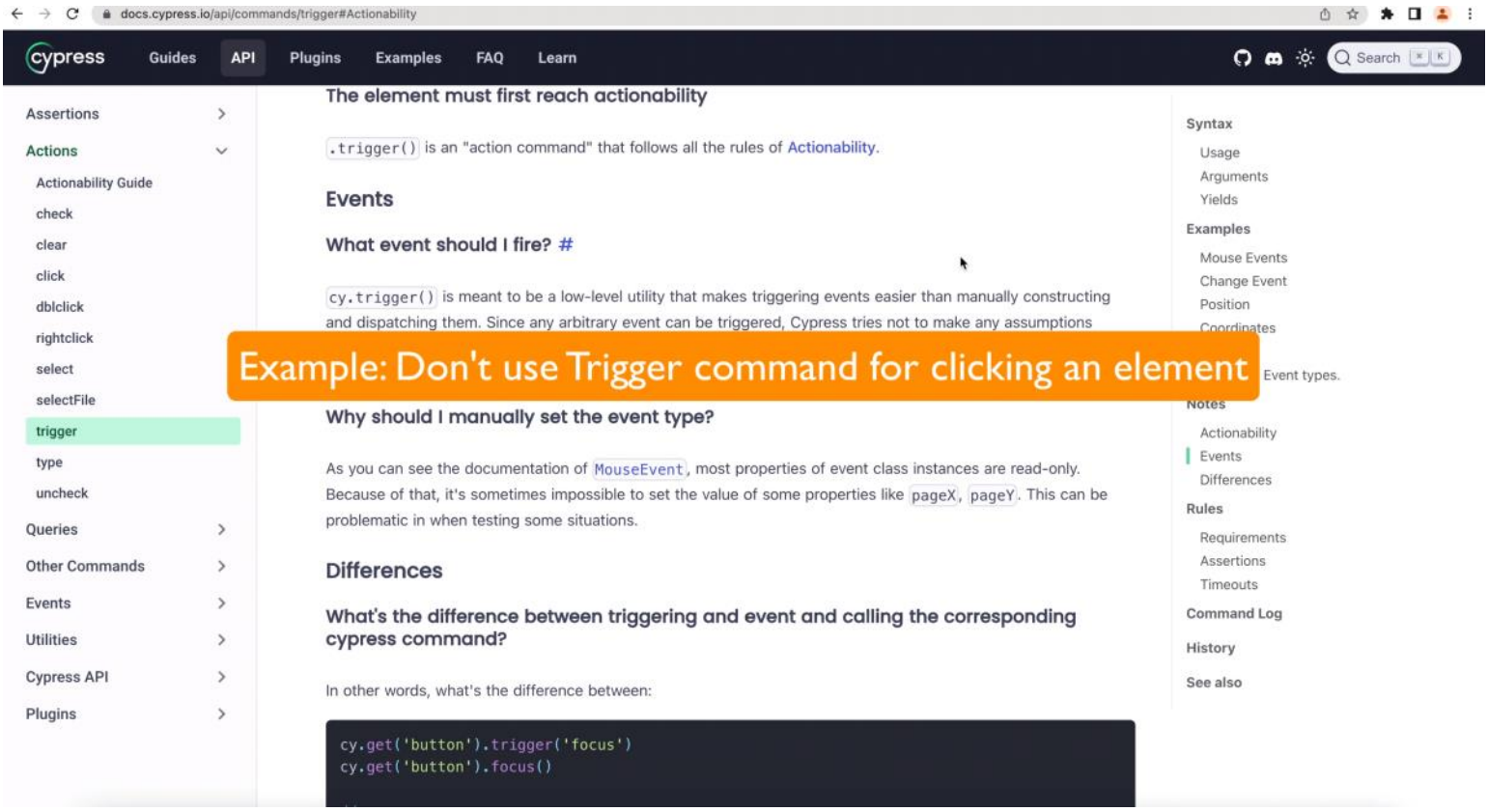
```
Development:
Mouseover can be implemented in two different ways:
1. CSS
2. JavaScript Events
```

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




docs.cypress.io/api/commands/trigger#Actionability
cypress
Guides
API
Plugins
Examples
FAQ
Learn
Search
Assertions
Actions
Actionability Guide
check
clear
click
dblclick
rightclick
select
selectFile
trigger
type
uncheck
Queries
Other Commands
Events
Utilities
Cypress API
Plugins
The element must first reach actionability
.trigger() is an "action command" that follows all the rules of Actionability.
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
Event types.
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
What's the difference between triggering and event and calling the corresponding cypress command?
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
Notes
Actionability
Events
Differences
Rules
Requirements
Assertions
Timeouts
Command Log
History
See also

Cypress 19

Saturday, March 16, 2024 11:39 PM

Element List Iteration - Part 1

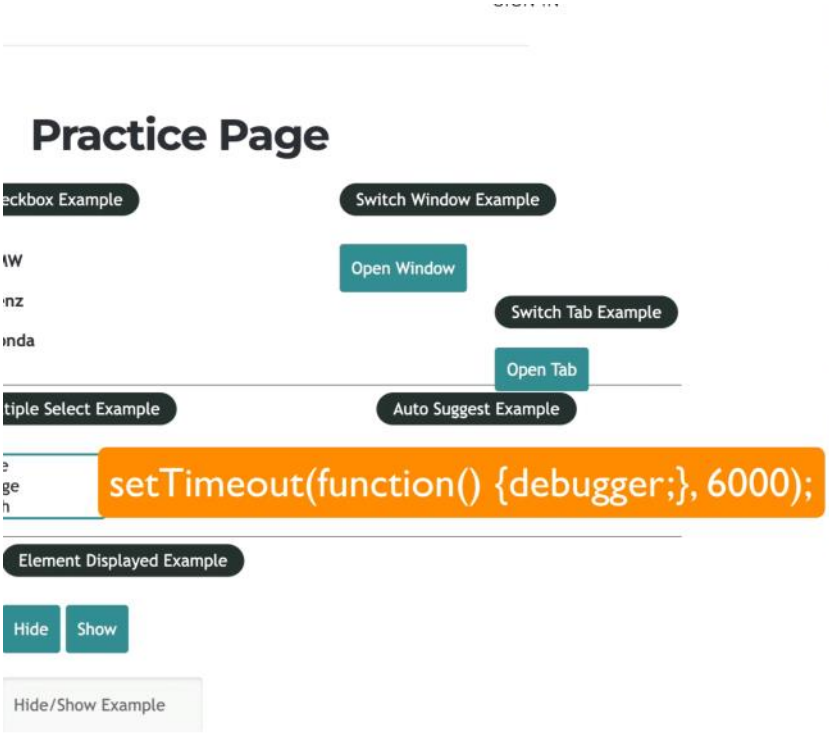
cypress > e2e > element_interactions > element_list_iteration_demo.cy.js > describe('Element List Iteration') callback > it('should understand f

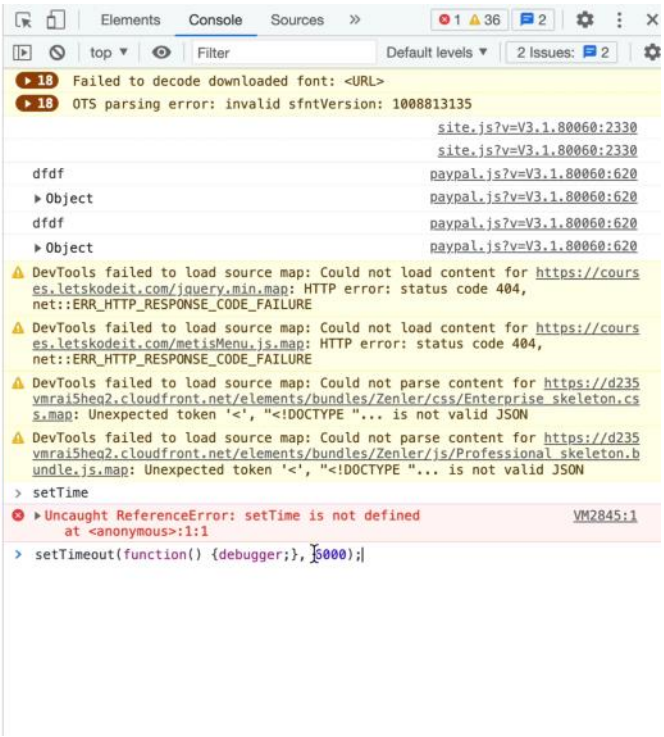
```
describe('Element List Iteration', () => {
    it('should understand for each loop and element list iteration', () => {
        cy.visit('https://courses.letskodeit.com/practice')

        cy.get('[class^="btn-style class1"]').each


    })
})
```

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);
```

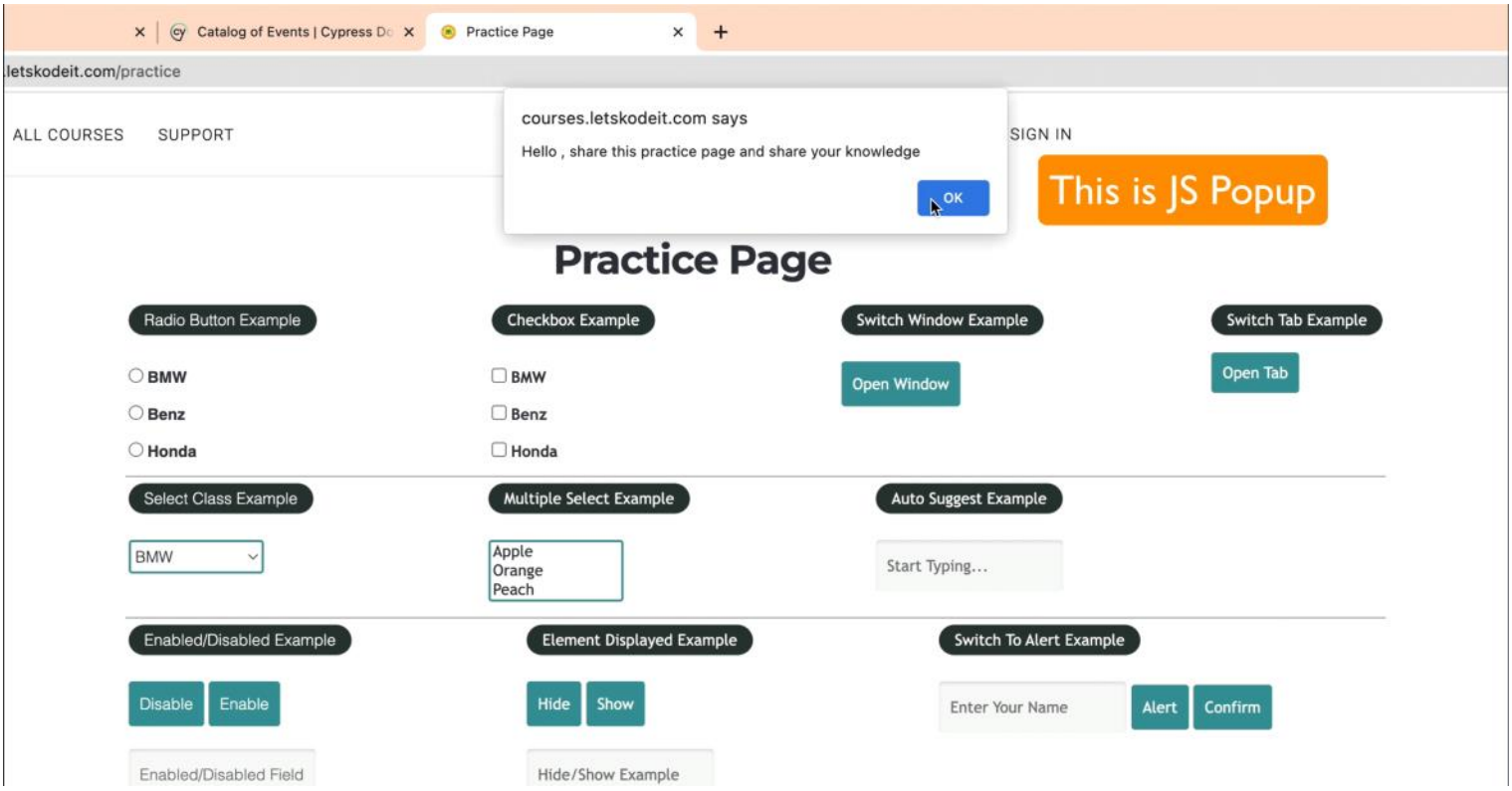




```
****************** Element List Iteration ******************

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()
          }
        })
    })

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

  })
})
```

Cypress 20

Sunday, March 17, 2024 5:56 PM



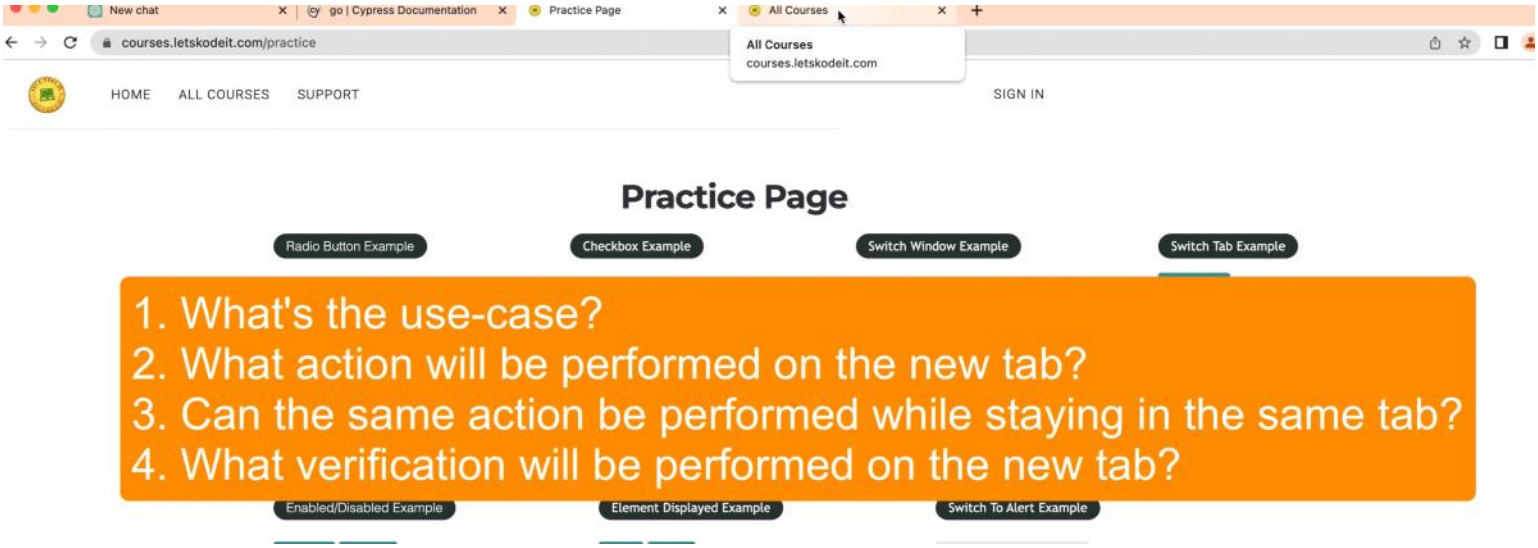

```
****************** JS Alerts and Confirm Popups ******************

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('#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)

  })
})
```

Cypress 21

Sunday, March 17, 2024 6:16 PM




```
****************** 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


```
****************** 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')
  })
})
```

Cypress 23

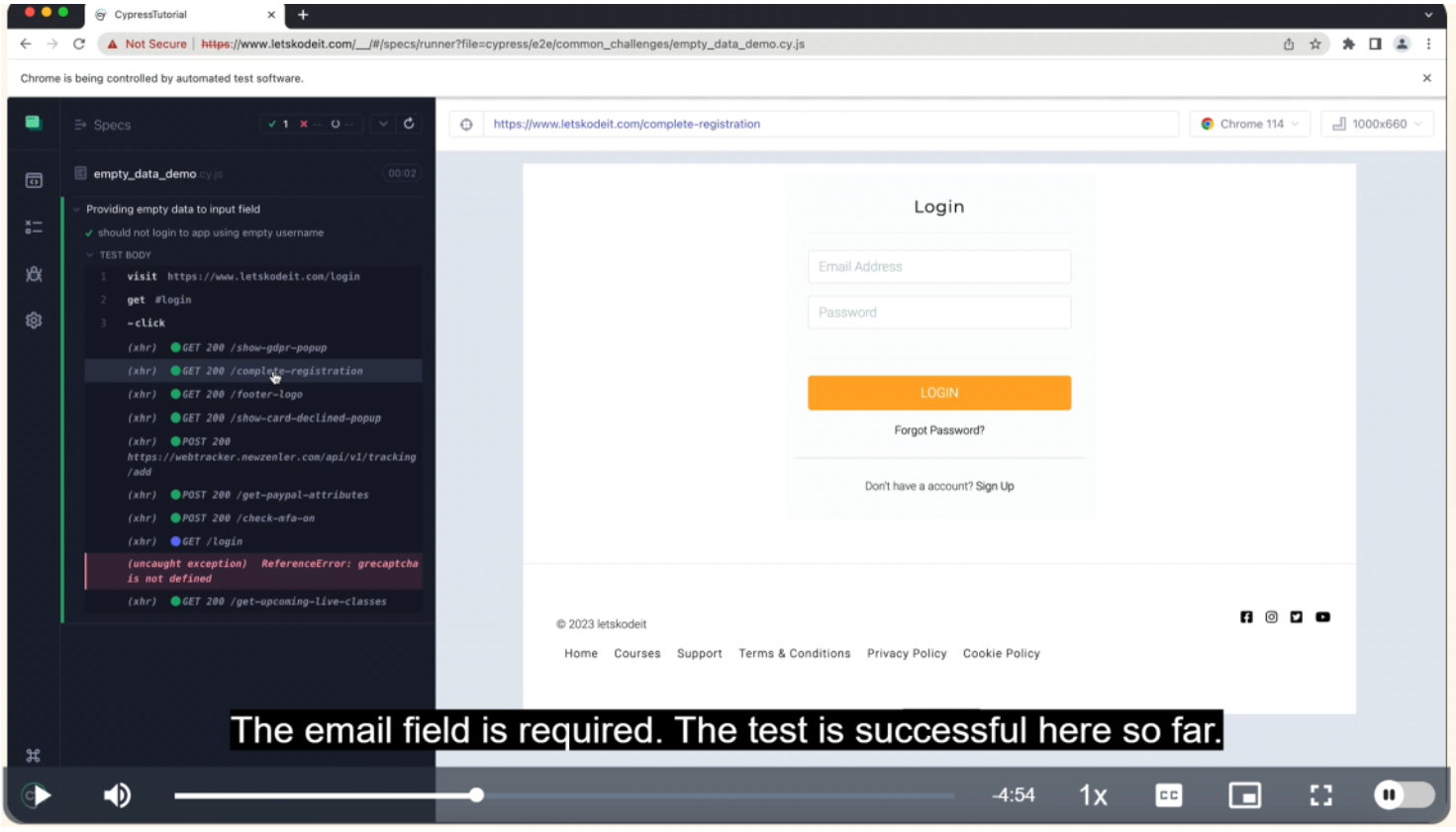
Sunday, March 17, 2024 9:59 PM

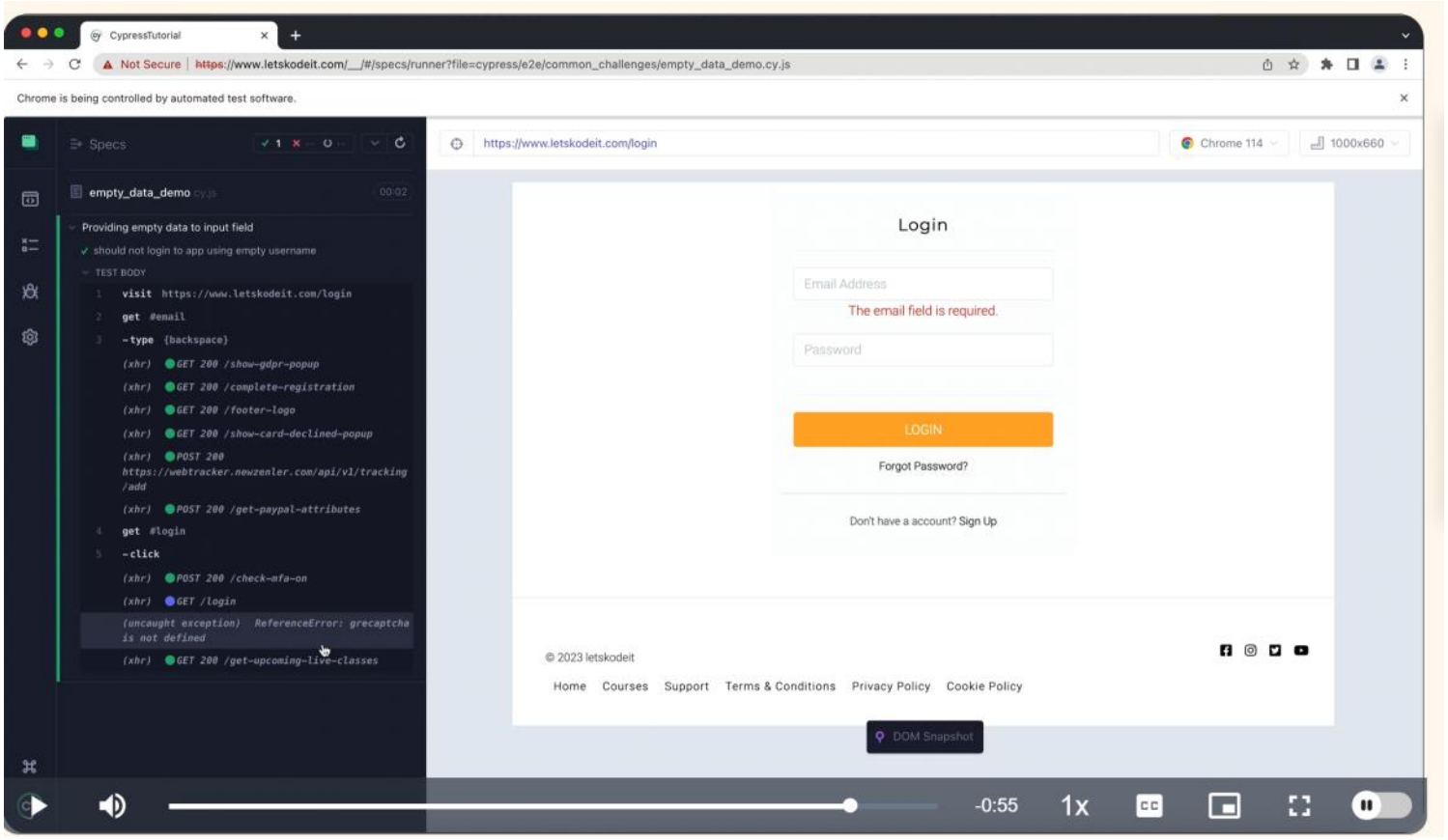

```
****************** Handling Iframes ******************

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
  })

  */
```




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
Providing empty data to input field
should not login to app using empty username
TEST BODY
visit https://www.letskodeit.com/login
get #email
-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
get #login
-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
Chrome 114
1000x660
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
-0:55
1x

Cypress 24b

Sunday, March 17, 2024 10:16 PM

empty_data_demo.cy.js e2e.js

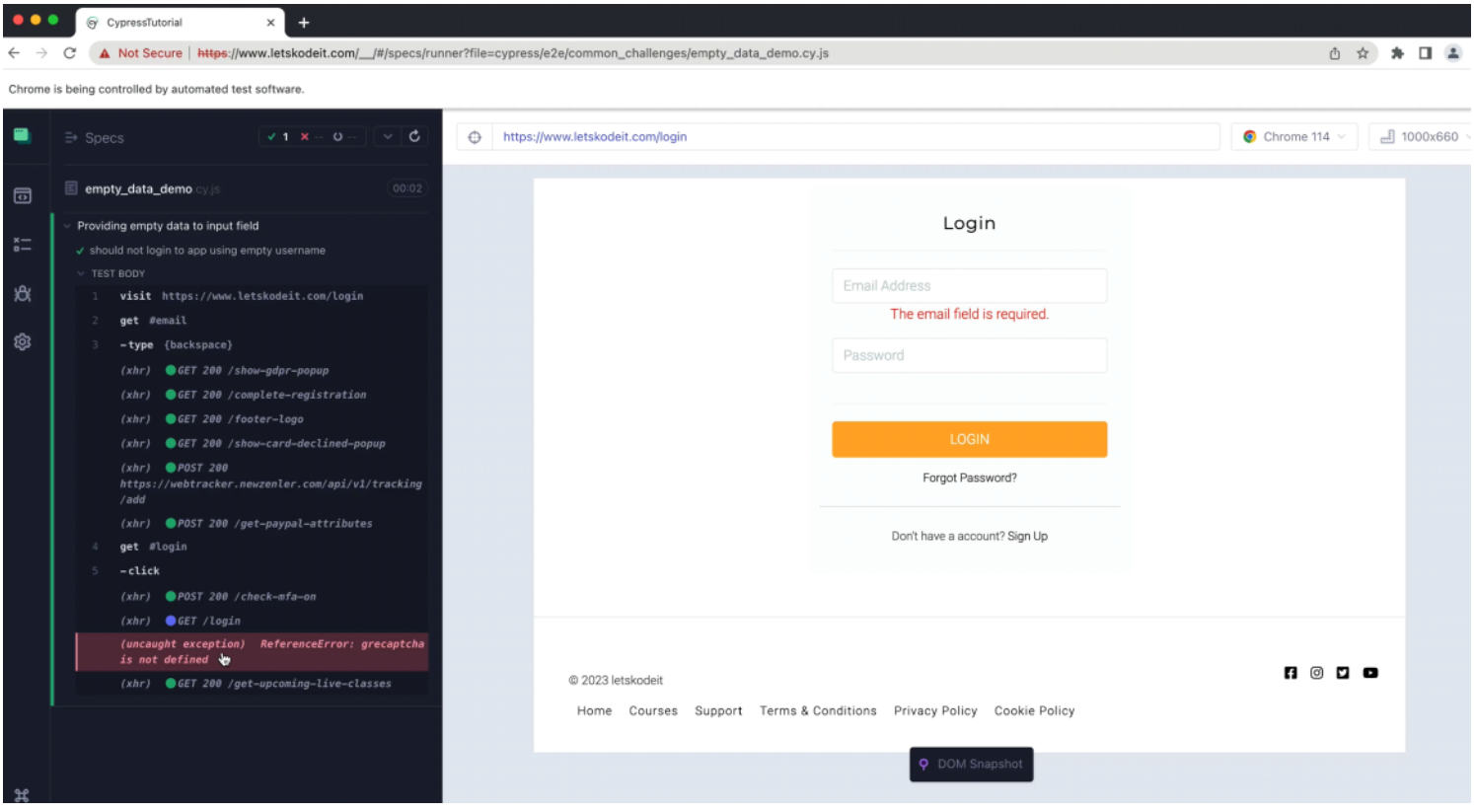
cypress > e2e > common_challenges > empty_data_demo.cy.js > ...

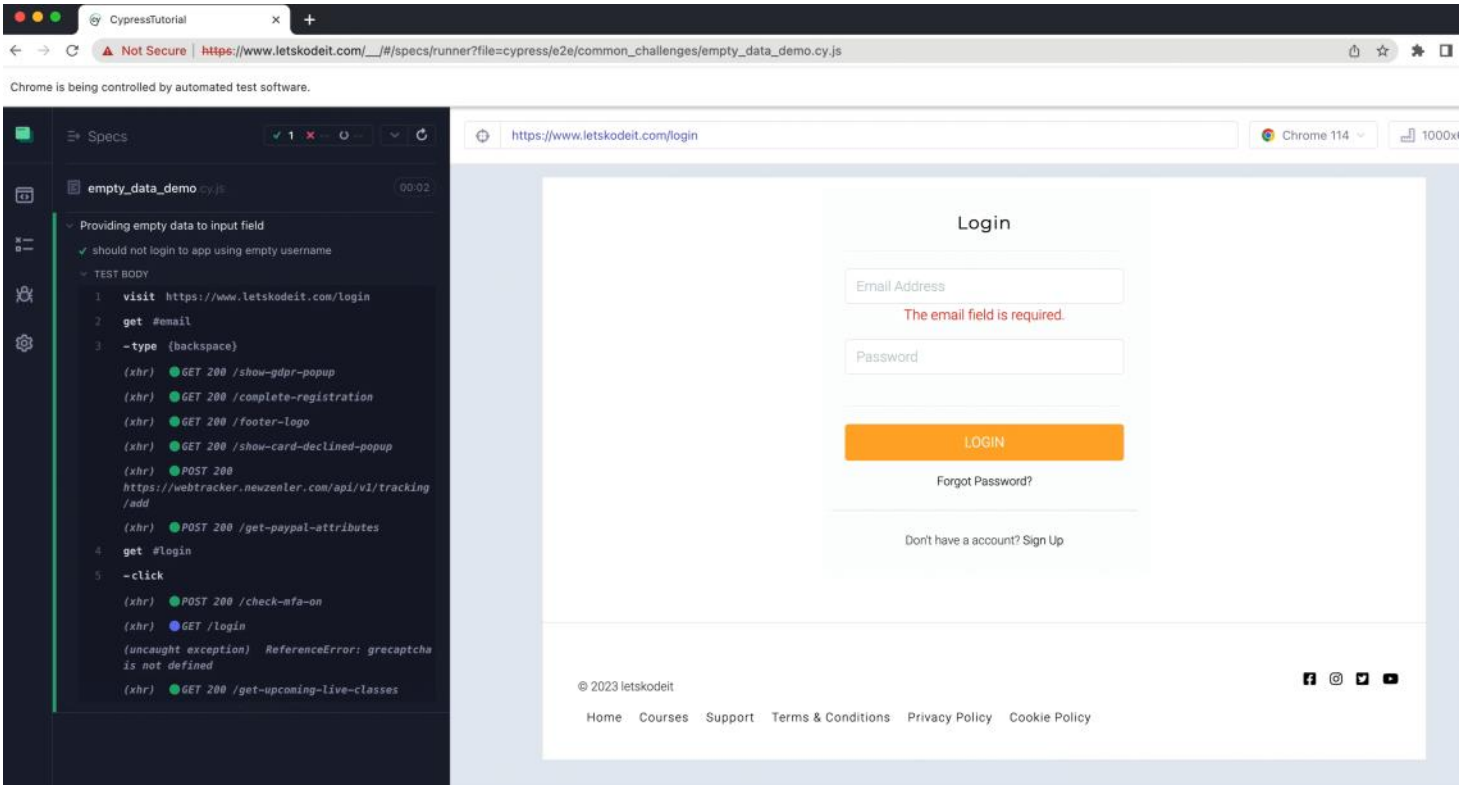
```
describe('Providing empty data to input field', () => {

    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()

    })
})
```

empty_data_demo.cy.js e2e.js

cypress > support > e2e.js > Cypress.on('uncaught:exception') callback

```
// ***********************************************************
// This example support/index.js is processed and
// loaded automatically before your test files.
//
// This is a great place to put global configuration and
// behavior that modifies Cypress.
//
// You can change the location of this file or turn off
// automatically serving support files with the
// 'supportFile' configuration option.
//
// You can read more here:
// https://on.cypress.io/configuration
// ***********************************************************

// Import commands.js using ES2015 syntax:
import './commands'

// Alternatively you can use CommonJS syntax:
// require('./commands')
require('cypress-xpath')

Cypress.on('uncaught:exception', (err, runnable) => {
    // returning false here prevents Cypress from
    // failing the test
    return false
})
```

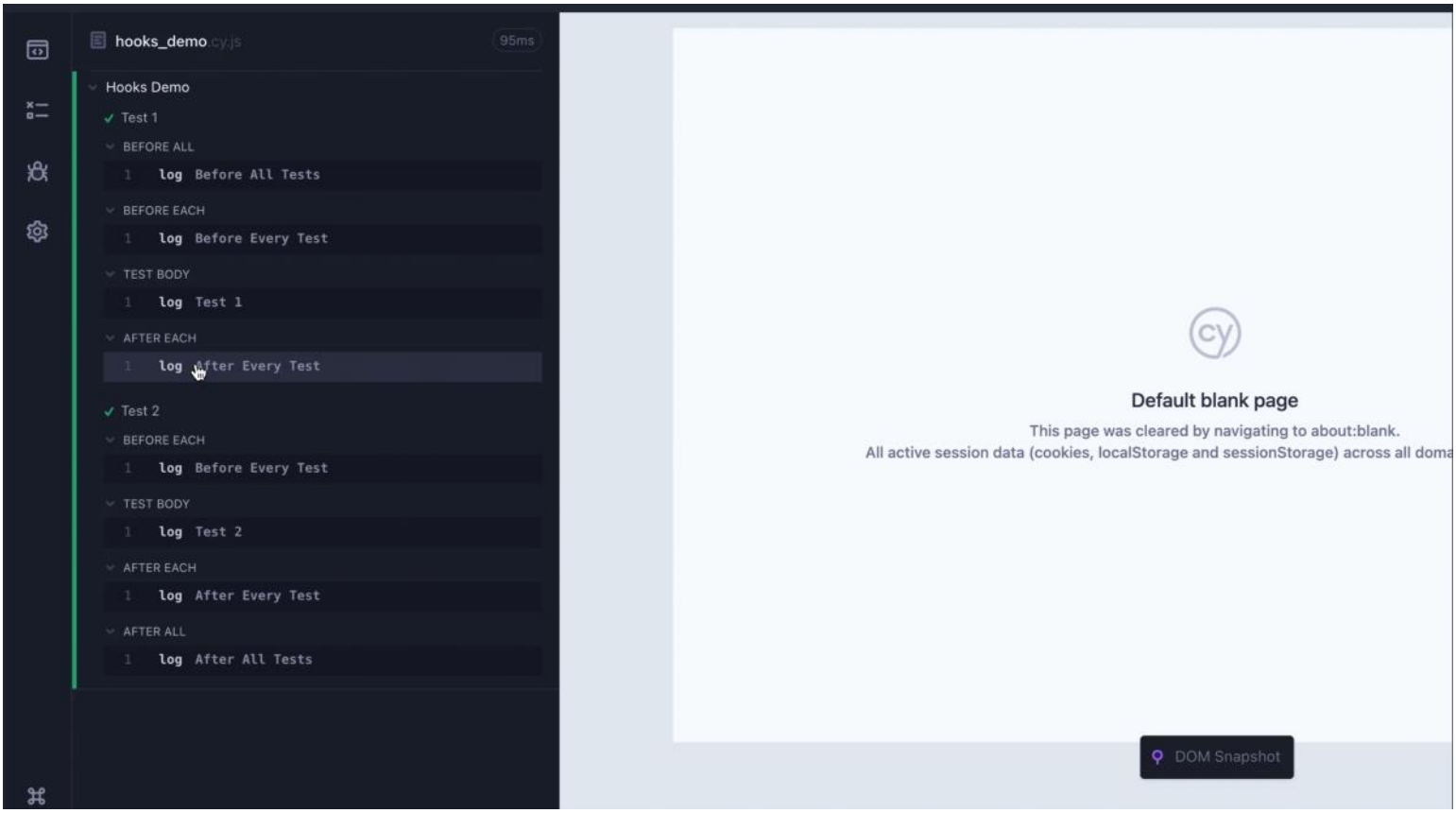



CypressTutorial
Not Secure https://www.letskodeit.com/__/#/specs/runner?file=cypress/e2e/common_challenges/empty_data_demo.cy.js
Chrome is being controlled by automated test software.
Specs
empty_data_demo
Providing empty data to input field
should not login to app using empty username
TEST BODY
visit https://www.letskodeit.com/login
get #email
-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
get #login
-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
Chrome 114
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

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')
    })
  })
```



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 golbalData;
    before('Before Hook', () => {
        cy.fixture("example").then((data) => {
            golbalData = data
        })
    })

    it('should understand how to use fixtures for reading data', () => {
      cy.visit("https://www.letskodeit.com/login");
      cy.get('#email').type(golbalData.testid1.username);
      cy.get('#login-password').type(golbalData.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) => { ... })
//
//
```

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

```
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  //
```

Child Command Example:
.click()

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

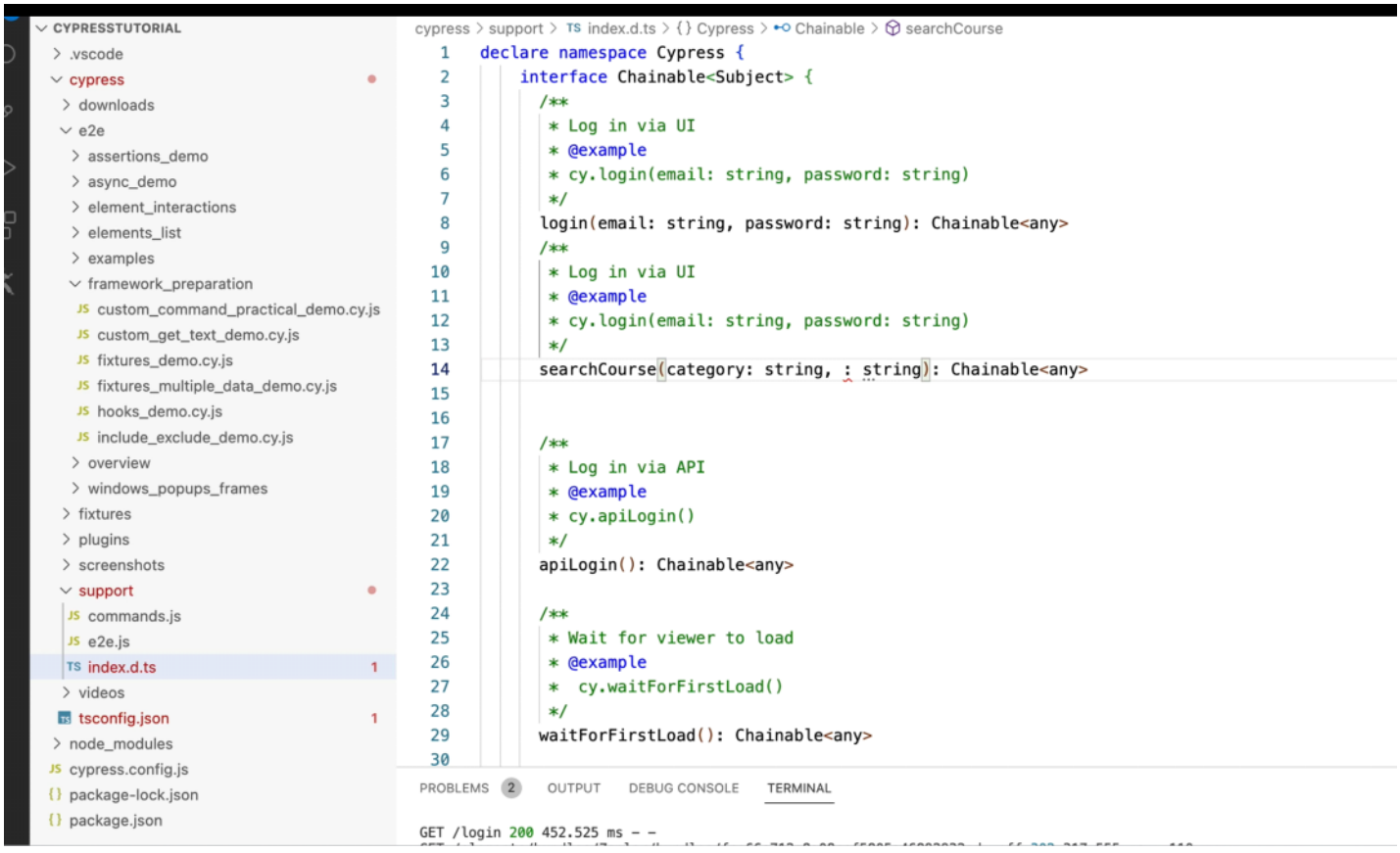
```
describe('Custom Command Get Text Demo', () => {

    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')
      })

    })
  })
```

<---- In support/command.js

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

```javascript
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')
    })

  })
})
```

```
****************** 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
```



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.”

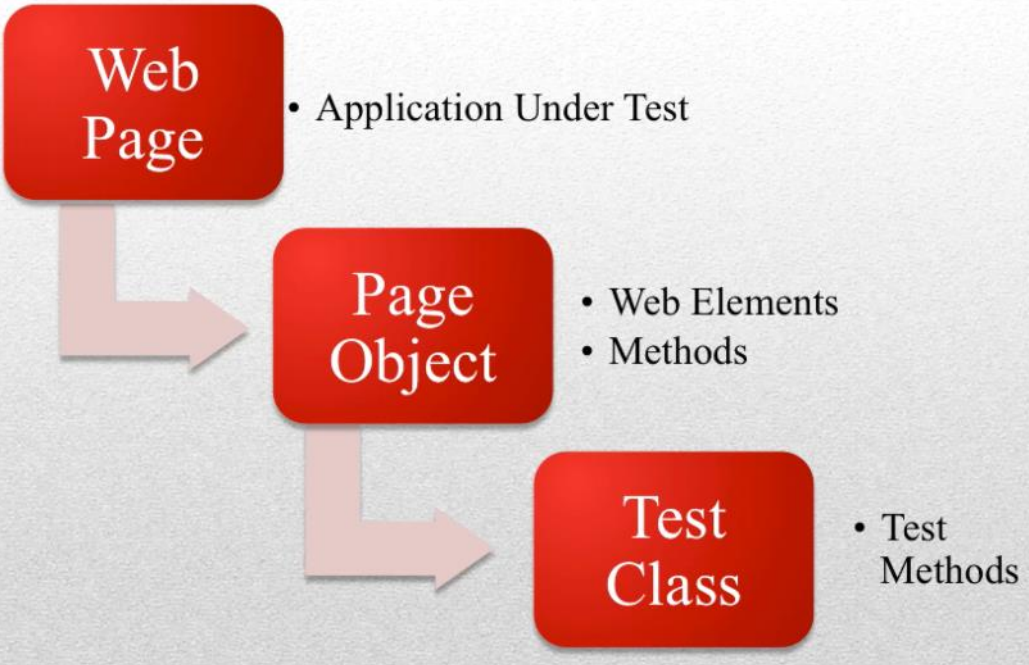
Robert Martin – Clean Code Collection

DRY Principle - Don’t Repeat Yourself

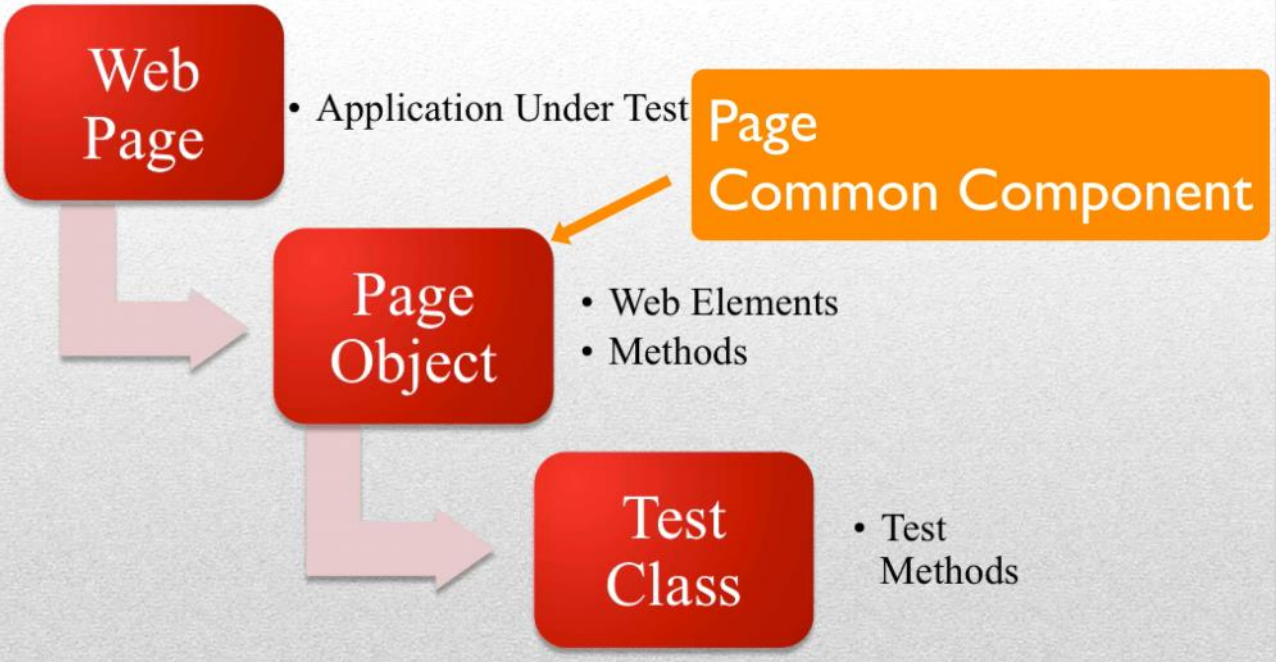
What is a Page Object Model?

- 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

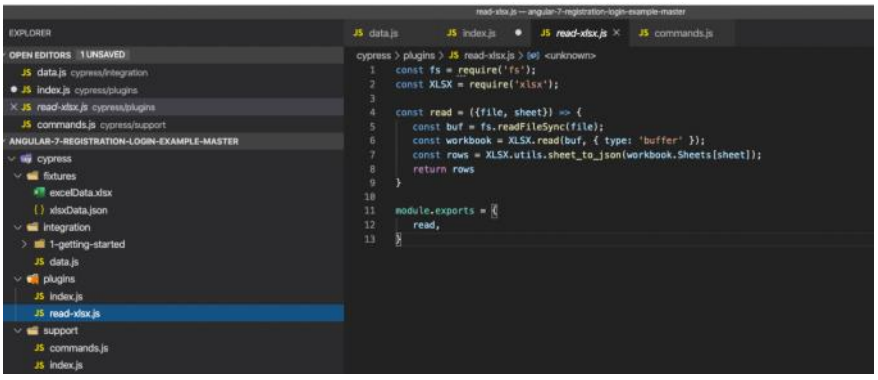
Sunday, April 7, 2024 8:46 PM

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

    npm install jsonpath
```

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

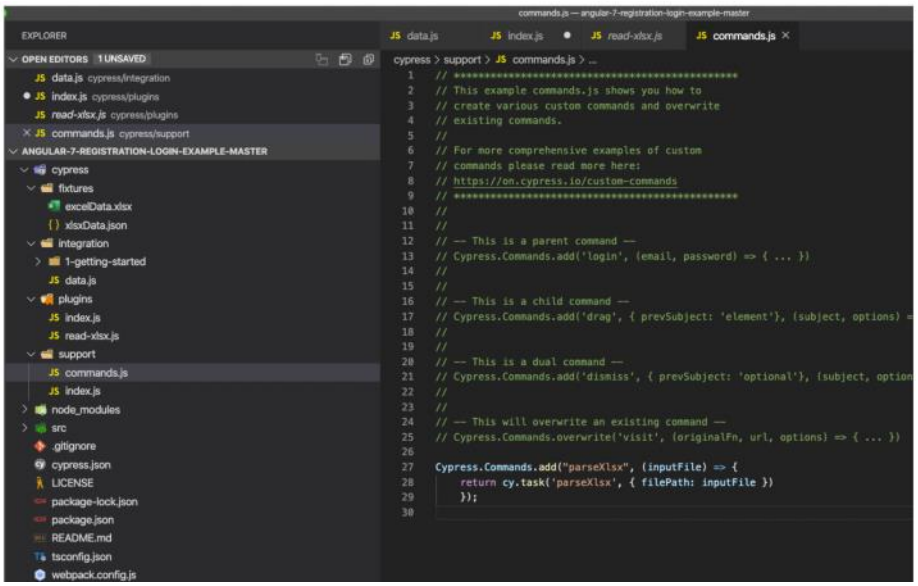
cypress/plugins/index.js:

```
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
    })
}
```



cypress/plugins/read-xlsx.js:

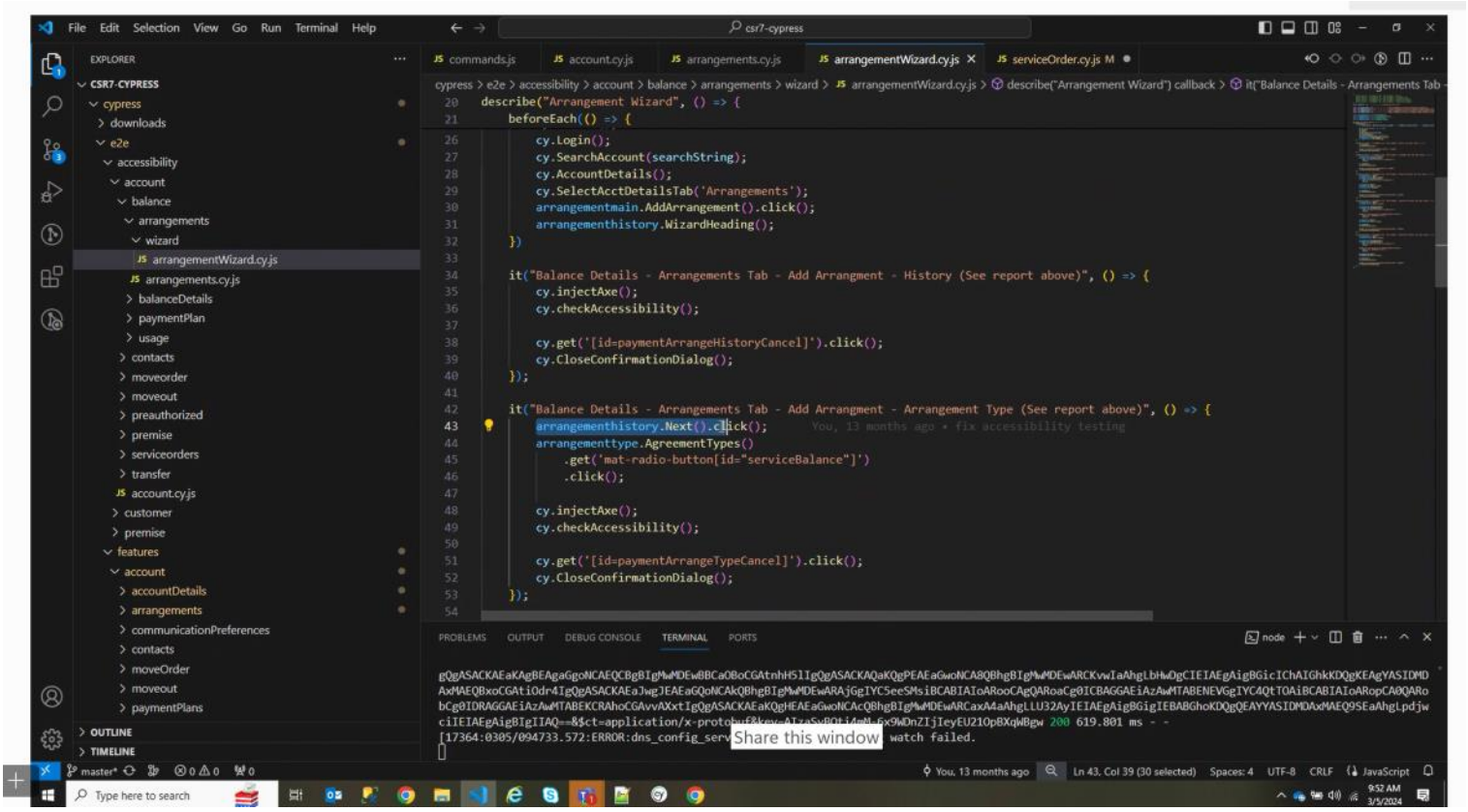
```
const xlsx = require("node-xlsx").default;
const fs = require("fs");
const read = ({file, sheet}) => {
    const buf = fs.readFileSync(file);
    const workbook = xlsx.read(buf, {type: 'buffer'});
    const rows = xlsx.utils.sheet_to_json(workbook, Sheets[sheet]);
    return rows
}
module.exports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



cypress/support/commands.js:

```
Cypress.Commands.add('parseXlsx', (inputfile) => {
    return cy.task('parseXlsx', { filepath: inputfile})
})
```

Control table
Custom Database Structure Control



Accessibility Testing

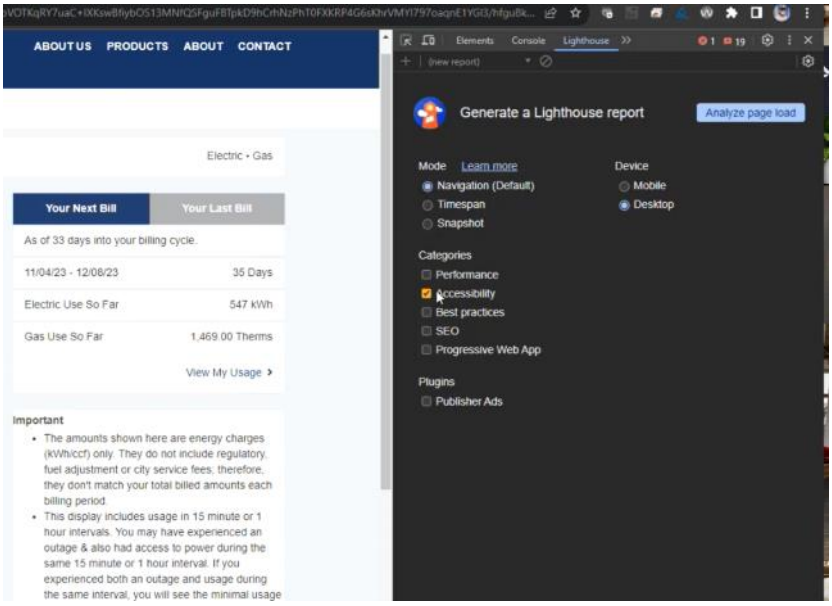
Monday, March 11, 2024 2:57 PM

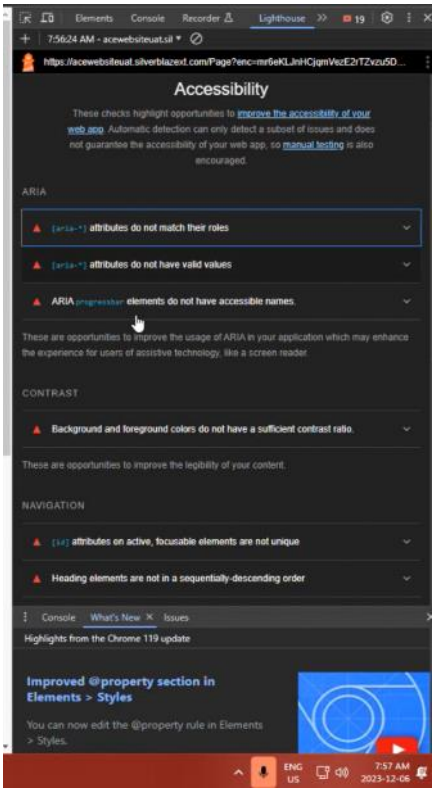
Testing for Accessibility

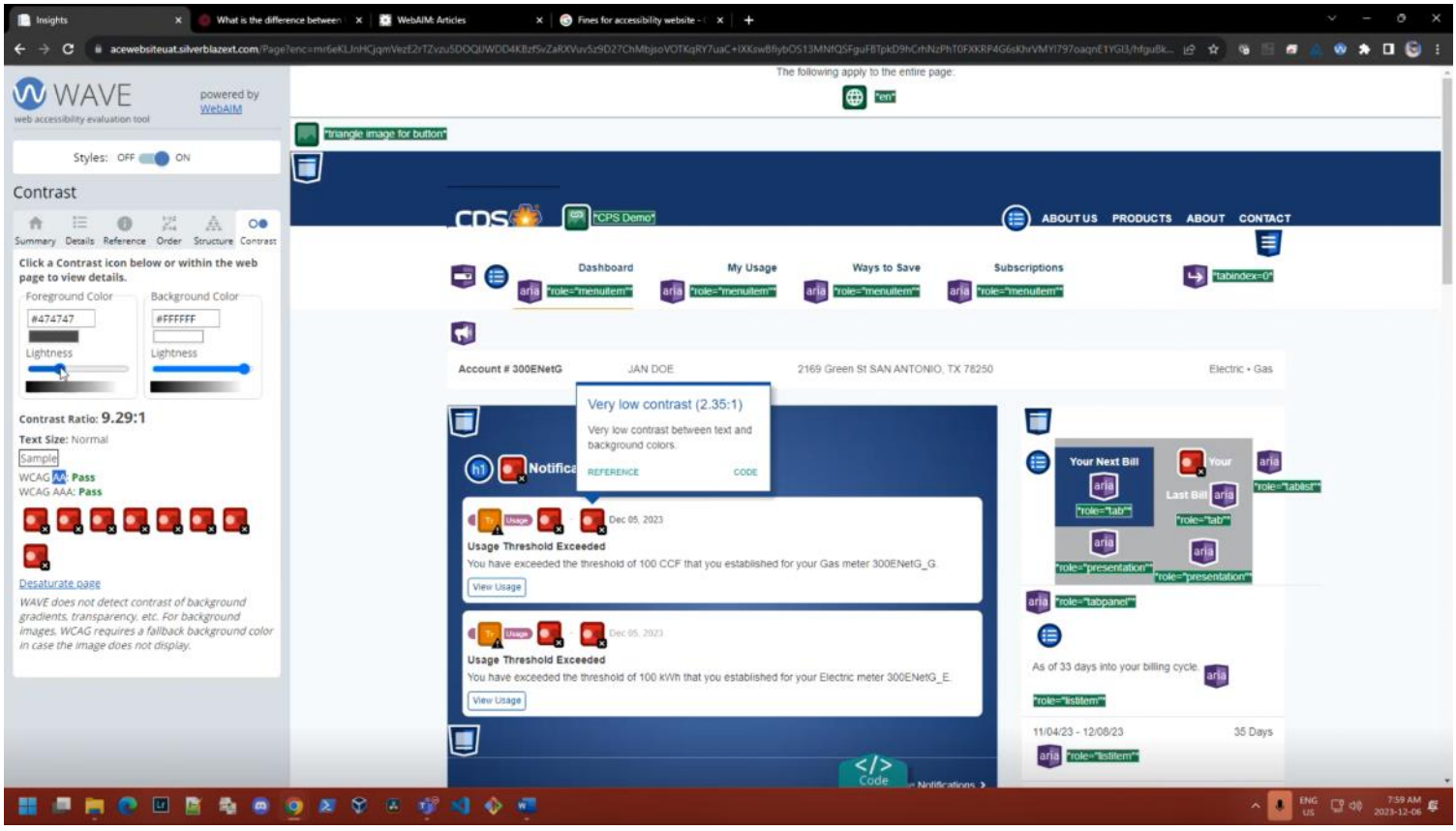
- Screen Readers – visual impairments. NVDA free screen reader tool
- Keyboard Navigation
  - Tab navigation
  - Focus styles
  - Perform actions with keyboard
- Color Contrast – low vision or color blindness
- Forms and Labels – Proper labelling and structure. Associate labels to form elements

Tools available to use:

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






WAVE
web accessibility evaluation tool
powered by WebAIM
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
Lightness
Background Color
#FFFFFF
Lightness
Contrast Ratio: 9.29:1
Text Size: Normal
Sample
WCAG AA: Pass
WCAG AAA: Pass
Desaturate page
WAVE 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.
The following apply to the entire page:
"en"
"triangle image for button"
"CPS Demo"
ABOUT US PRODUCTS ABOUT CONTACT
Dashboard
My Usage
Ways to Save
Subscriptions
"tabindex='0'"
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
Last Bill
"role='tablist'"
"role='tab'"
"role='presentation'"
"role='tabpanel'"
As of 33 days into your billing cycle.
"role='listitem'"
11/04/23 - 12/08/23
35 Days
Code