To work with iframes we need to install dependency-

```
npm install -D cypress-iframe
```

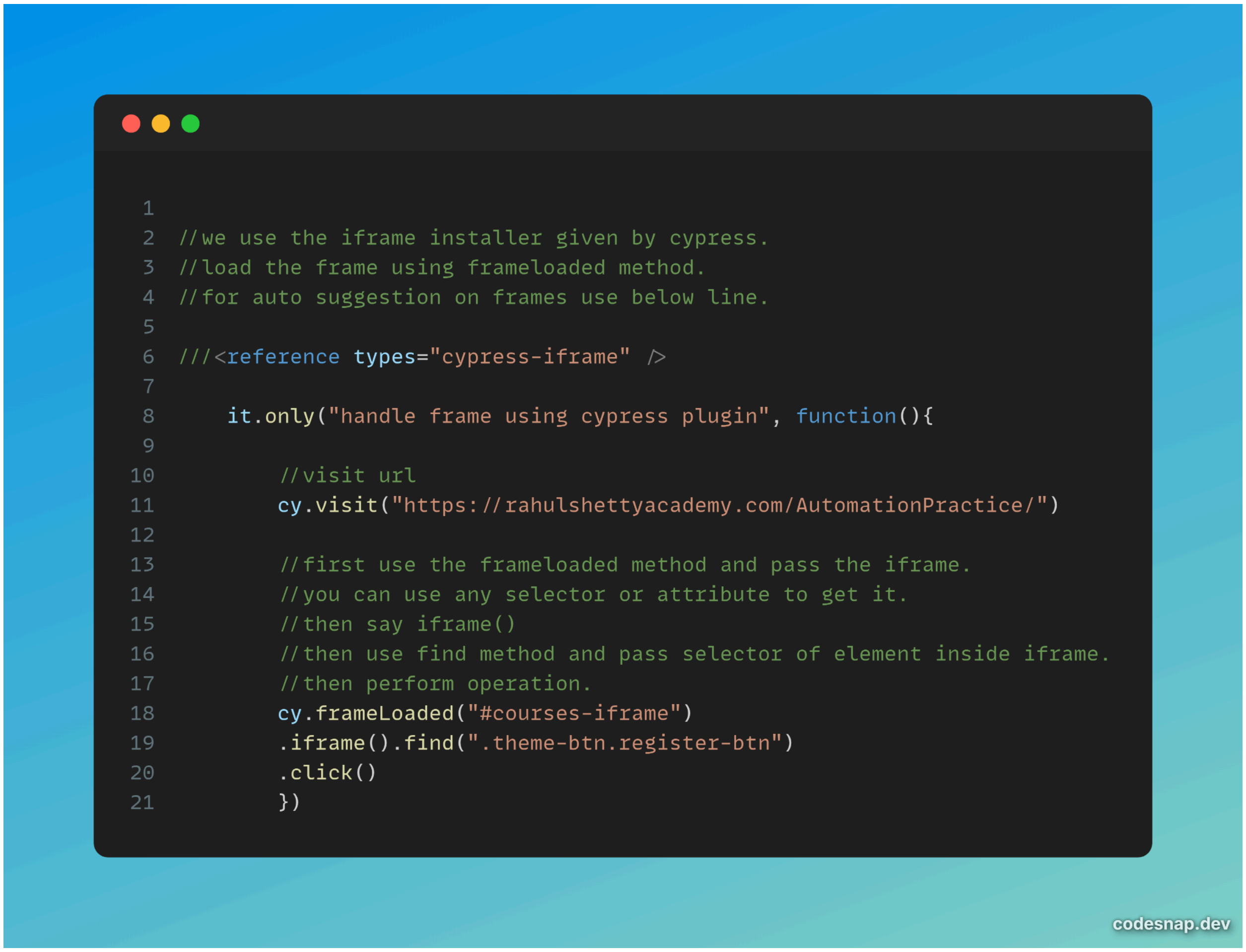
In your `cypress/support/e2e.js` file, add the following:

```
import 'cypress-iframe';

// or

require('cypress-iframe');
```

Find command-

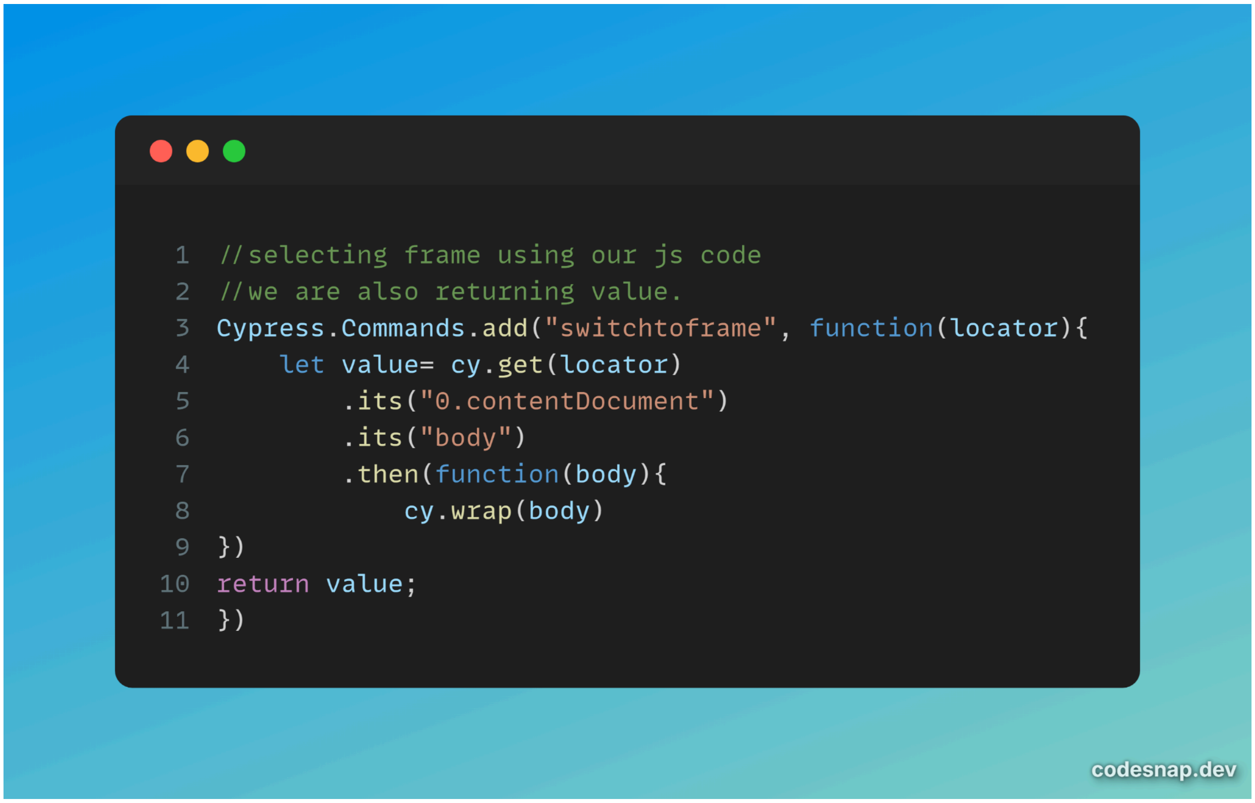
Go inside a frame or area and then use locators to identify elements using find.

Frames with dependency of cypress-



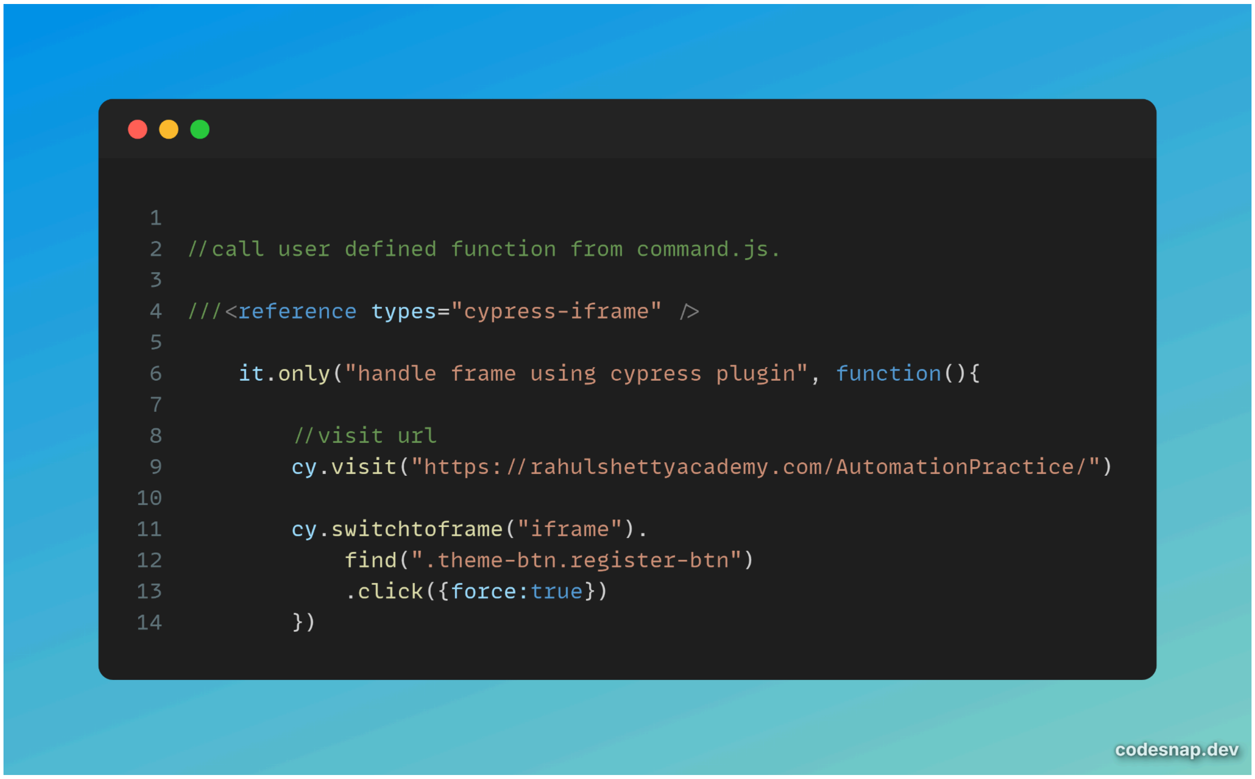
```

//we use the iframe installer given by cypress.
//load the frame using frameloaded method.
//for auto suggestion on frames use below line.

///<reference types="cypress-iframe"  />

    it.only("handle frame using cypress plugin", function(){

        //visit url
        cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

        //first use the frameloaded method and pass the iframe.
        //you can use any selector or attribute to get it.
        //then say iframe()
        //then use find method and pass selector of element inside iframe.
        //then perform operation.
        cy.frameLoaded("#courses-iframe")
        .iframe().find(".theme-btn.register-btn")
        .click()
        })
```

command.js user defined file-



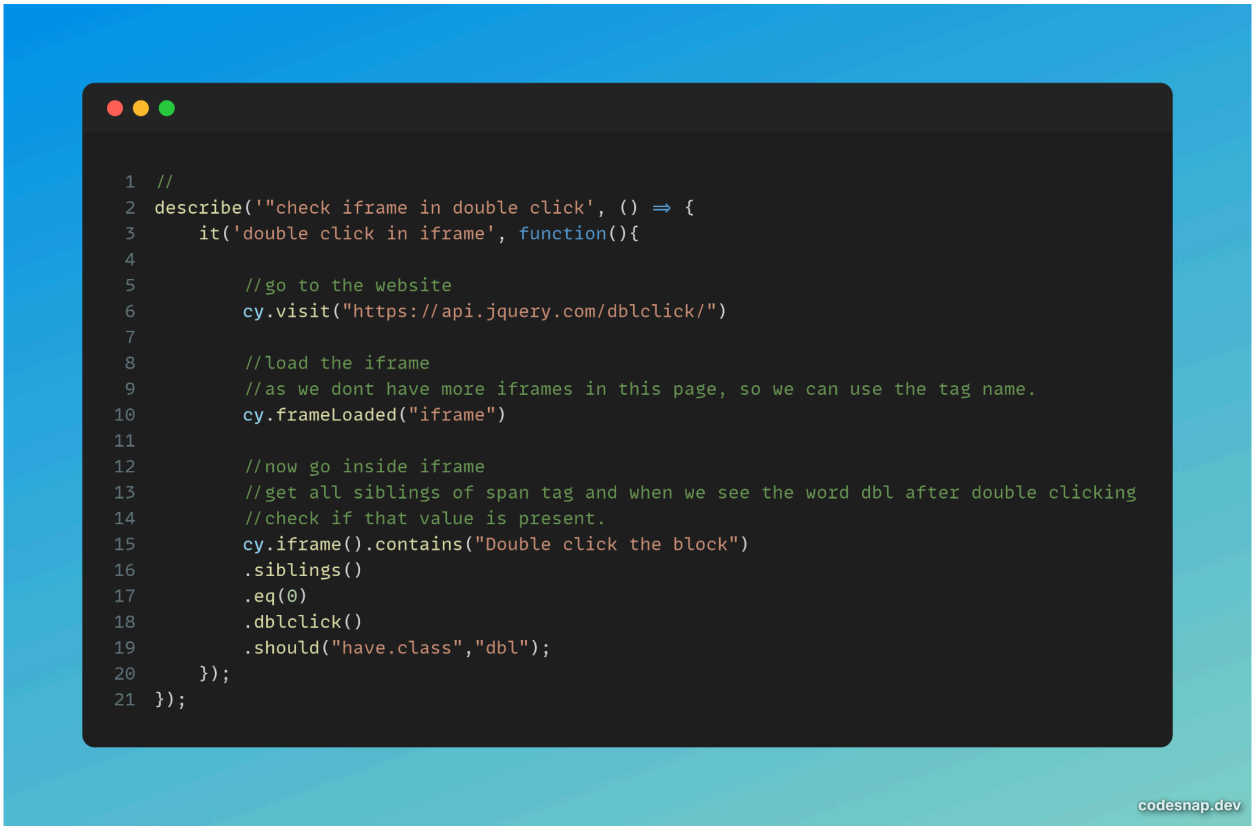
```
//selecting frame using our js code
//we are also returning value.
Cypress.Commands.add("switchtoframe", function(locator){
  let value= cy.get(locator)
    .its("0.contentDocument")
    .its("body")
    .then(function(body){
      cy.wrap(body)
})
return value;
})
```

codesnap.dev

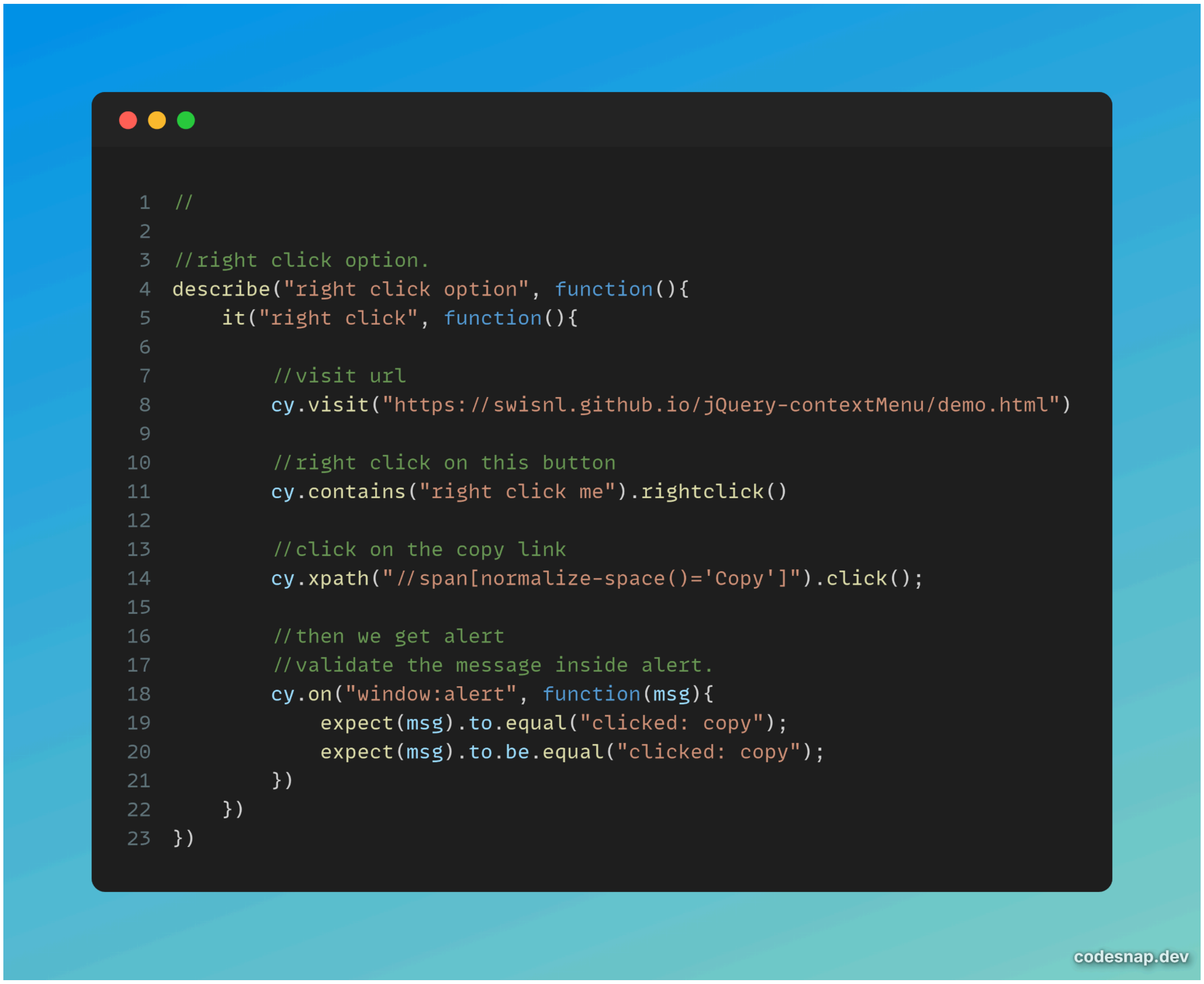


```

//call user defined function from command.js.

///<reference types="cypress-iframe" />

  it.only("handle frame using cypress plugin", function(){

    //visit url
    cy.visit("https://rahulshettyacademy.com/AutomationPractice/")

    cy.switchtoframe("iframe").
      find(".theme-btn.register-btn")
      .click({force:true})
    })
```

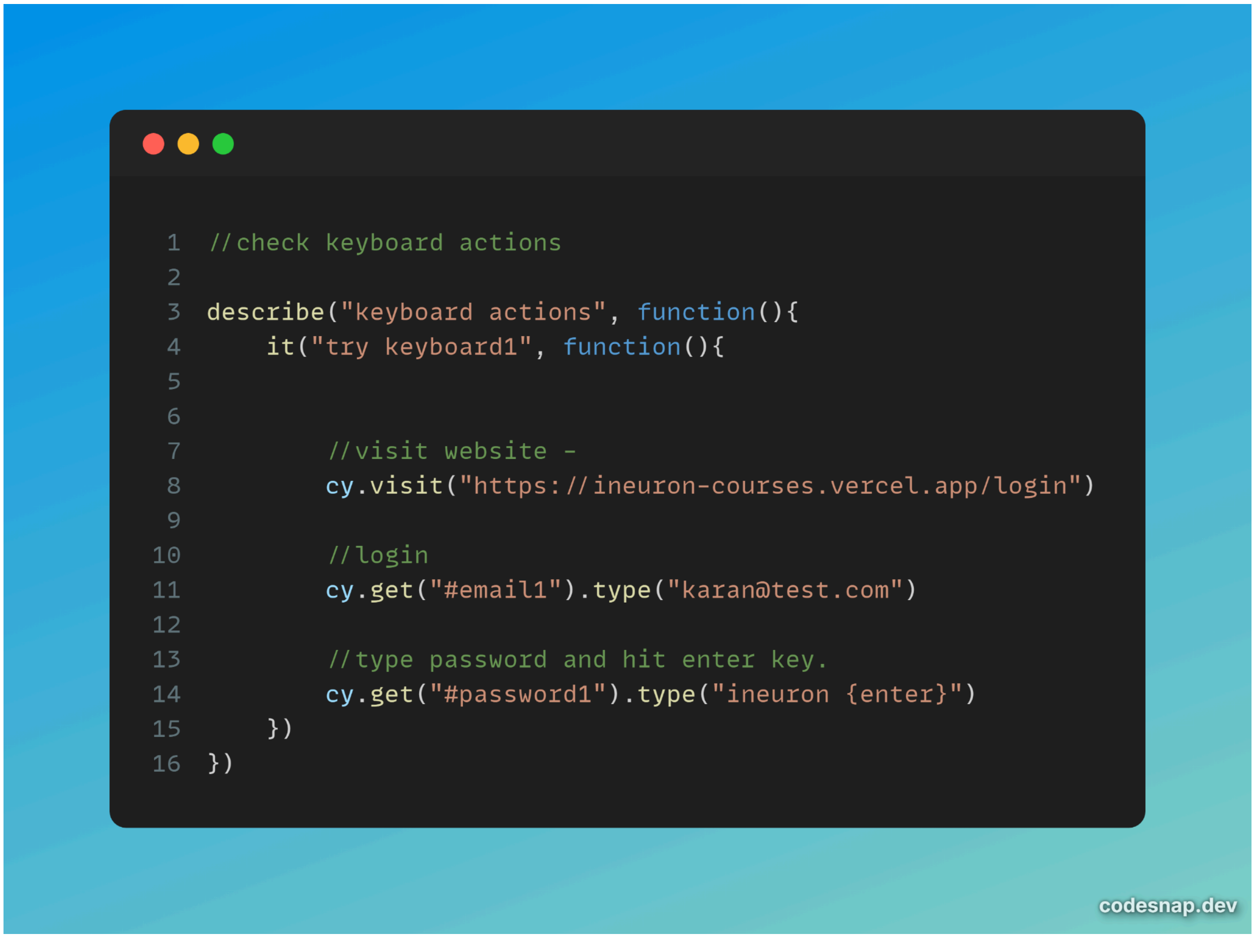
codesnap.dev



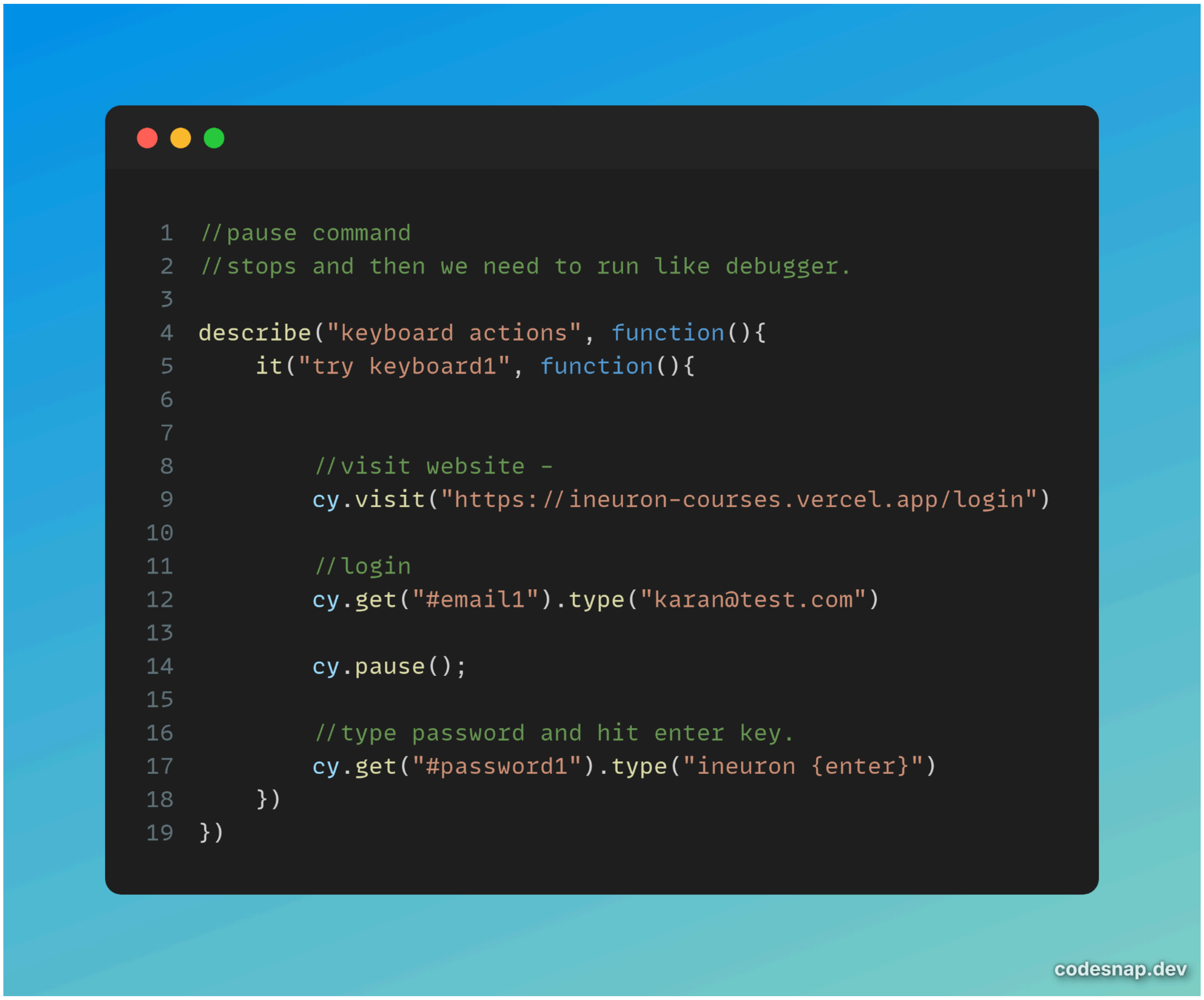
```
//
describe('"check iframe in double click', () => {
    it('double click in iframe', function(){

        //go to the website
        cy.visit("https://api.jquery.com/dblclick/")

        //load the iframe
        //as we dont have more iframes in this page, so we can use the tag name.
        cy.frameLoaded("iframe")

        //now go inside iframe
        //get all siblings of span tag and when we see the word dbl after double clicking
        //check if that value is present.
        cy.iframe().contains("Double click the block")
        .siblings()
        .eq(0)
        .dblclick()
        .should("have.class","dbl");
    });
});
```



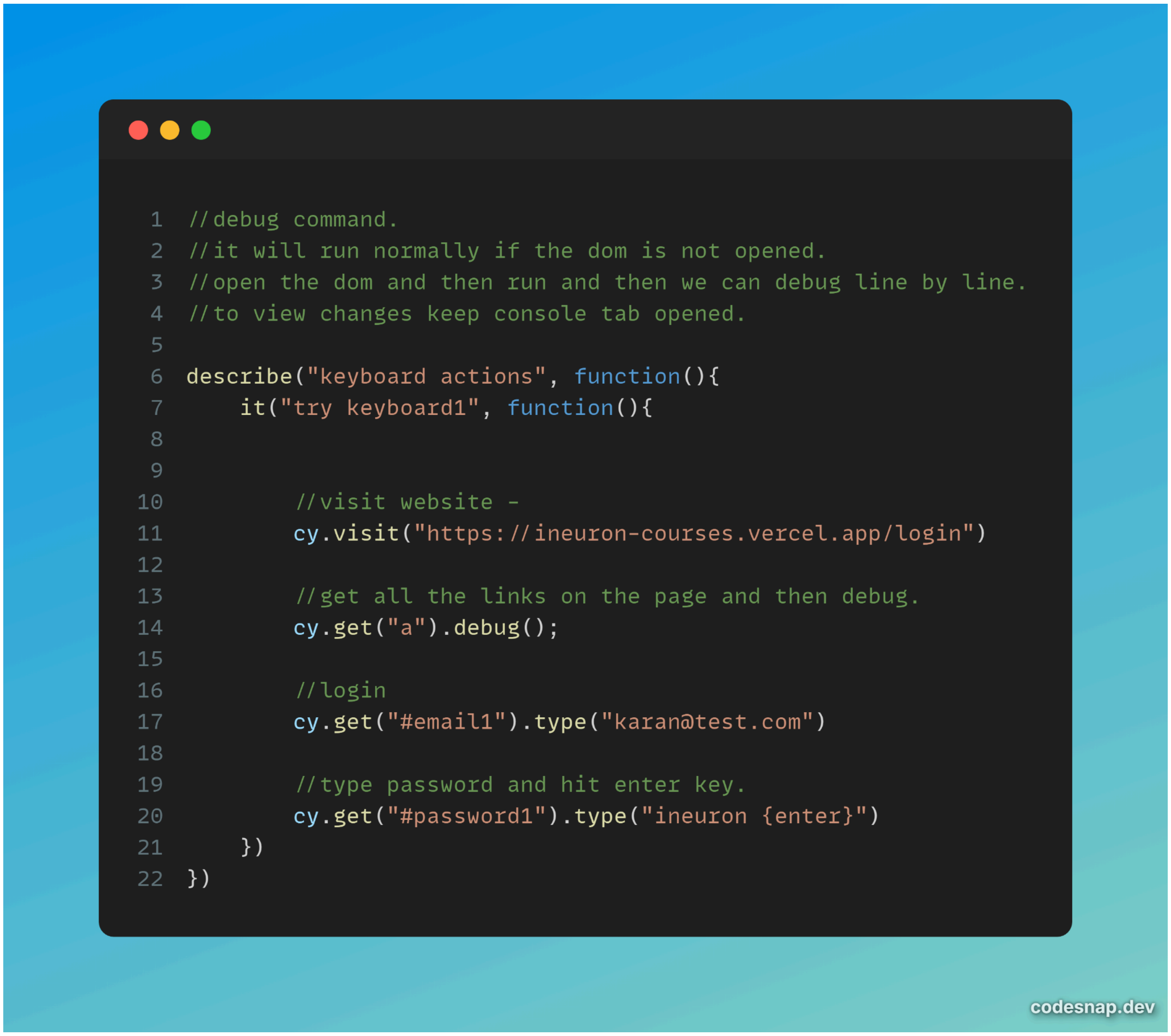
```
//

//right click option.
describe("right click option", function(){
    it("right click", function(){

        //visit url
        cy.visit("https://swisnl.github.io/jQuery-contextMenu/demo.html")

        //right click on this button
        cy.contains("right click me").rightclick()

        //click on the copy link
        cy.xpath("//span[normalize-space()='Copy']").click();

        //then we get alert
        //validate the message inside alert.
        cy.on("window:alert", function(msg){
            expect(msg).to.equal("clicked: copy");
            expect(msg).to.be.equal("clicked: copy");
        })
    })
})
```



```
//check keyboard actions

describe("keyboard actions", function(){
    it("try keyboard1", function(){


        //visit website -
        cy.visit("https://ineuron-courses.vercel.app/login")

        //login
        cy.get("#email1").type("karan@test.com")

        //type password and hit enter key.
        cy.get("#password1").type("ineuron {enter}")
    })
})
```



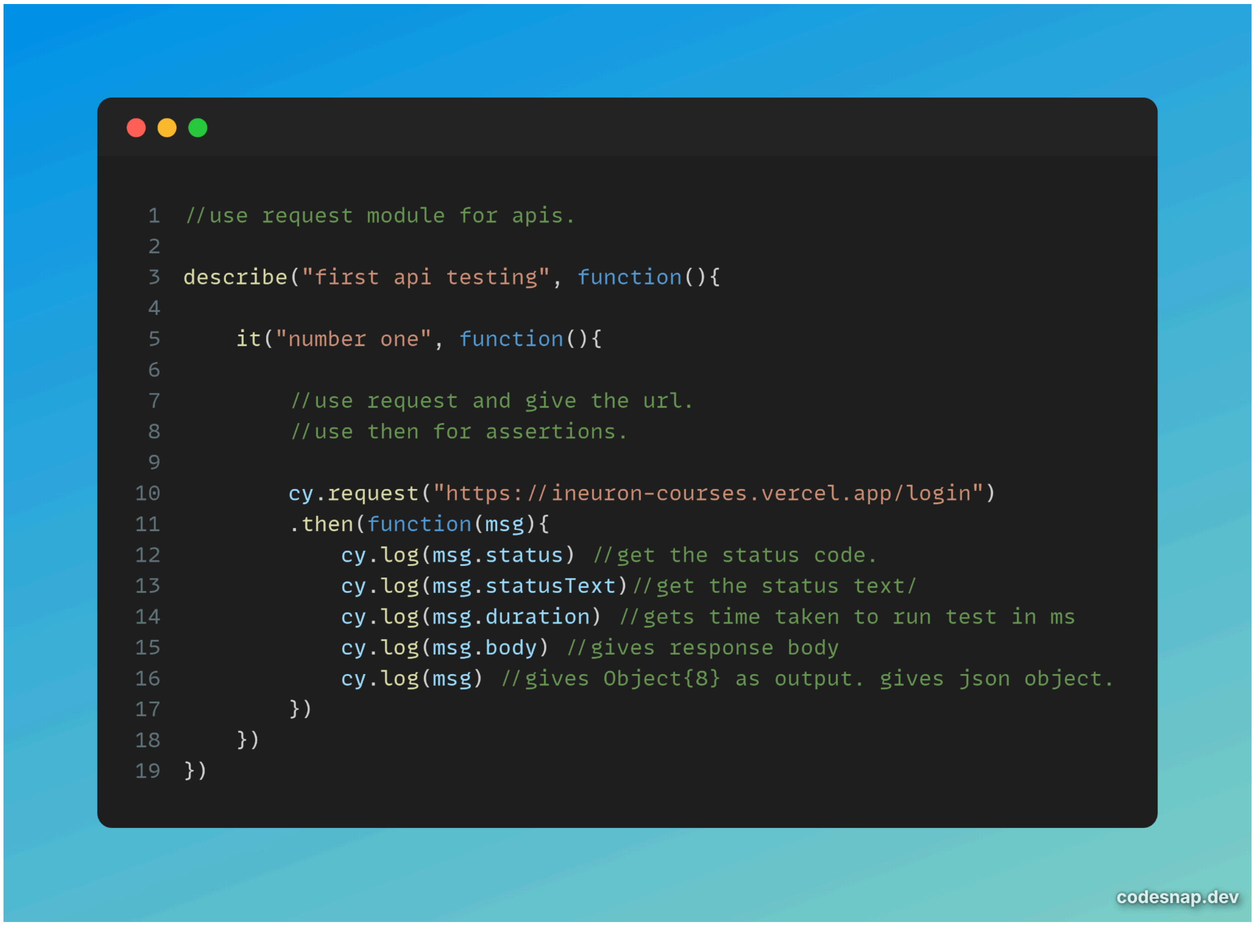
```
//pause command
//stops and then we need to run like debugger.

describe("keyboard actions", function(){
    it("try keyboard1", function(){


        //visit website -
        cy.visit("https://ineuron-courses.vercel.app/login")

        //login
        cy.get("#email1").type("karan@test.com")

        cy.pause();

        //type password and hit enter key.
        cy.get("#password1").type("ineuron {enter}")
    })
})
```



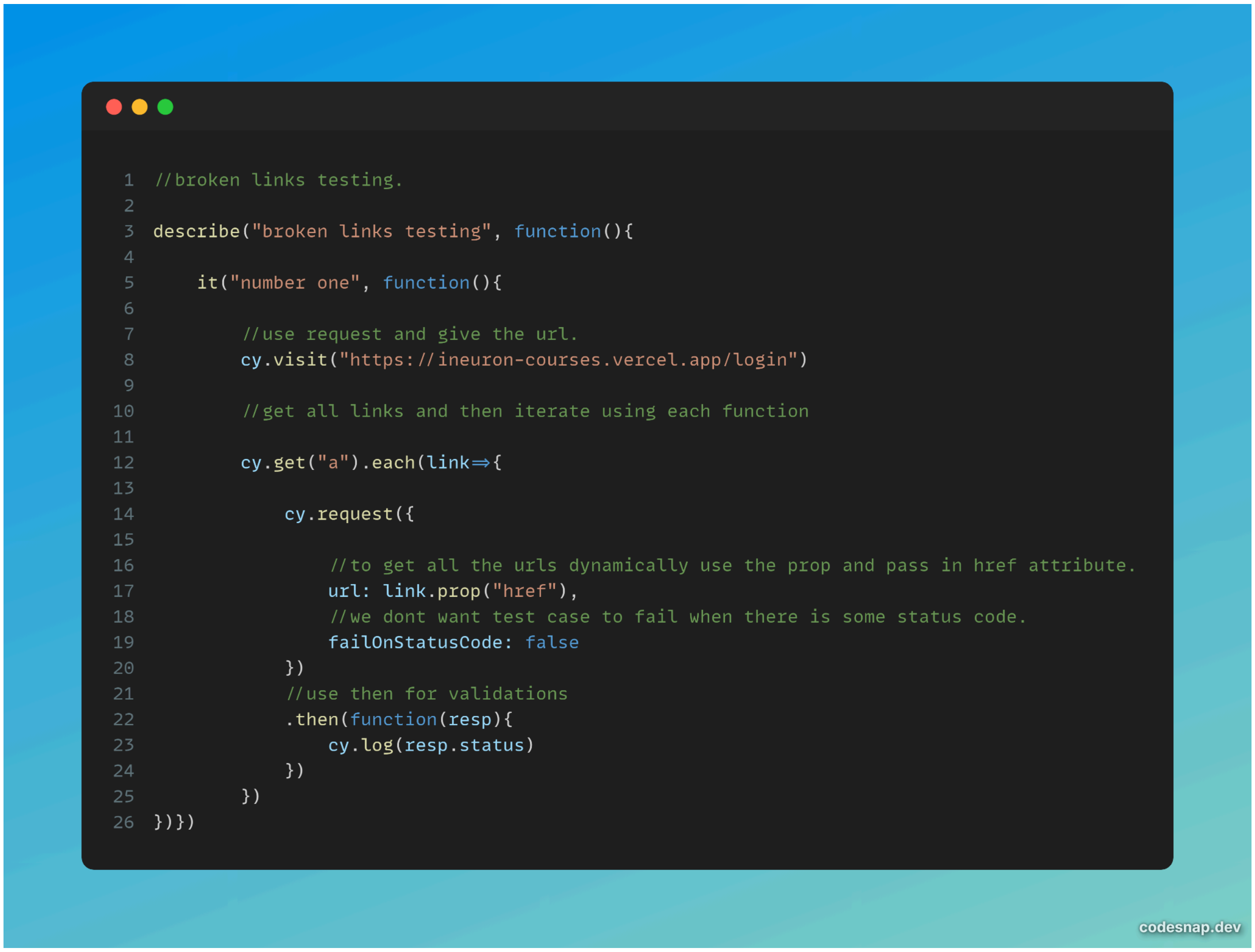
```
//debug command.
//it will run normally if the dom is not opened.
//open the dom and then run and then we can debug line by line.
//to view changes keep console tab opened.

describe("keyboard actions", function(){
    it("try keyboard1", function(){


        //visit website -
        cy.visit("https://ineuron-courses.vercel.app/login")

        //get all the links on the page and then debug.
        cy.get("a").debug();

        //login
        cy.get("#email1").type("karan@test.com")

        //type password and hit enter key.
        cy.get("#password1").type("ineuron {enter}")
    })
})
```

For api testing go to the network tab and you can get all the details.

Use request fixture for api testing.



```javascript
//use request module for apis.

describe("first api testing", function(){

    it("number one", function(){

        //use request and give the url.
        //use then for assertions.

        cy.request("https://ineuron-courses.vercel.app/login")
        .then(function(msg){
            cy.log(msg.status) //get the status code.
            cy.log(msg.statusText)//get the status text/
            cy.log(msg.duration) //gets time taken to run test in ms
            cy.log(msg.body) //gives response body
            cy.log(msg) //gives Object{8} as output. gives json object.
        })
    })
})
```



```
//broken links testing.

describe("broken links testing", function(){

    it("number one", function(){

        //use request and give the url.
        cy.visit("https://ineuron-courses.vercel.app/login")

        //get all links and then iterate using each function

        cy.get("a").each(link=>{

            cy.request({

                //to get all the urls dynamically use the prop and pass in href attribute.
                url: link.prop("href"),
                //we dont want test case to fail when there is some status code.
                failOnStatusCode: false
            })
            //use then for validations
            .then(function(resp){
                cy.log(resp.status)
            })
        })
})})
```