cypress

Key Differences

Cypress is executed in the same run loop as your application. Behind Cypress is a Node.js Server process. Cypress and the Node.js process constantly communicate, synchronize, and perform tasks on behalf of each other. Having access to both parts (front and back) gives us the ability to respond to your application's events in real-time, while at the same time work outside of the browser for tasks that require a higher privilege.

Cypress also operates at the network layer by reading and altering web traffic on the fly. This enables Cypress to not only modify everything coming in and out of the browser but also to change code that may interfere with its ability to automate the browser.

Cypress ultimately controls the entire automation process from top to bottom, which puts it in the unique position of being able to understand everything happening in and outside of the browser. This means Cypress is capable of delivering more consistent results than any other testing tool.

Because Cypress operates within your application, that means it has native access to every single object. Whether it is the window, the document, a DOM element, your application instance, a function, a timer, a service worker, or anything else - you have access to it in Cypress. There is no object serialization, there is no over-the-wire protocol - you have access to everything at your fingertips.

Cypress Setup

Download Node.js and set environment variables for Node.JS

variable name: NODE_HOME
variable value: C:\Program Files\nodejs

Create a Cypress working folder:
C:\Users\admin\CypressProject

Type and follow instructions:
npm init

Install Cypress:
npm install cypress --save-dev (cypress@4.8.0)

Start Cypress Test Runner (type in CMD or vscode terminal):
node_modules\.bin\cypress open
Full path:
C:\Users\******\CypressAutomation\node_modules\.bin\cypress open

How to run Cypress tests (Terminal, CMD)

Background and UI Run

```
node_modules\.bin\cypress run                (Running in background) (run all test scripts))
node_modules\.bin\cypress run --headed       (Running in UI)
```

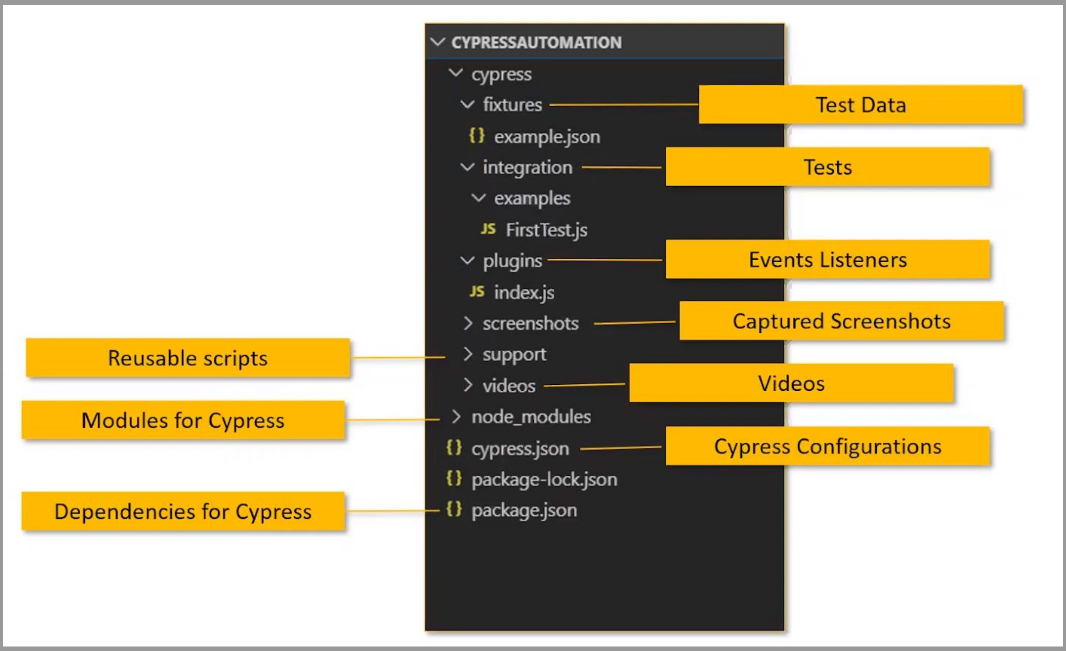
Running single test case under examples directory

```
node_modules\.bin\cypress run --spec "cypress\integration\examples\firsttest.js"
node_modules\.bin\cypress run --record --key **58d11d-****-4c5b-****-e29c4972ae**
```

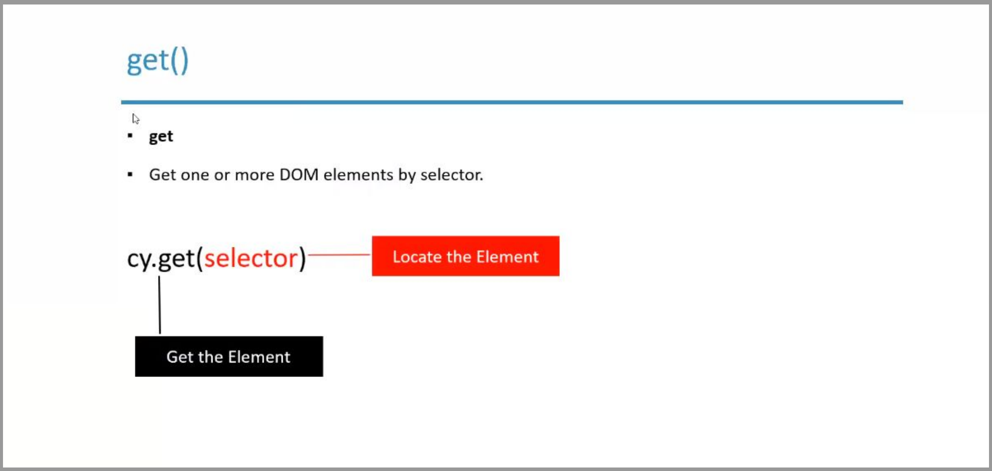
Running in Chrome instead of running Electron:

```
node_modules\.bin\cypress run --browser chrome
```

Folder Structure



Locators and Selectors cy.get(selector)


get()

- **get**
- Get one or more DOM elements by selector.

Login Demo Web Shop:
class locator (.)
Example: .ico-login

Search Box:
id locator (#)
Example: #small-searchterms

Search Button:
[attributes=value]
Example: [type=submit]

Add To Cart:
.class[attrubute=value]
Examples:
.add-to-cart-button[value="Add to cart"]
.input[name=userName]
.input[name=password]

LOGO
Css Selector - Copy Selector
body > div.master-wrapper-page > div.master-wrapper-content > div.header > div.header-logo > a > img

Example: Firsttest.js

```
describe('MyTestSuite', () => {
    it('Verfy Title of The Page - Positive', function()
    {
      cy.visit('http://demowebshop.tricentis.com/')
      cy.title().should('eq', 'Demo Web Shop')
    })
    it('Verfy Title of The Page - Negative', function()
    {
      cy.visit('http://demowebshop.tricentis.com/')
      cy.title().should('eq', 'Web Shop')
    })
  })
```

Example: Locators.spec.js

1. Launch Browser & open URL: https:\\demo.nopcommerce.com
2. Enter text in the Search box “Apple MacBook Pro 13-inch”
3. Click on Search Button
4. Click on Add to cart
5. Provide Quantity 2
6. Click on Add to cart
7. Click on Shopping Cart link at the top of the page
8. Verify the total amount

```
/// <reference types="cypress" />
describe('Locating Elements', function()
{
    it('verify types of locators', function()
    {
        cy.visit("https://demo.nopcommerce.com/") //Opens the URL

        cy.get("#small-searchterms").type("Apple MacBook Pro 13-inch") //Search Box

        cy.get("[type='submit']").click() //Click on Search button

        cy.get(".product-box-add-to-cart-button[value='Add to cart']").click()

        cy.get("#product_enteredQuantity_4").clear().type('6') //Quantity
        cy.wait(3000)

        cy.get('#add-to-cart-button-4').click() //Add to card button after inserting quantity
        cy.wait(3000)

        cy.get("#topcartlink > a > span.cart-label").click() //Shopping card link

        cy.get(".product-unit-price").contains('$1,800.00') //Validation
    })
})
```

Interacting with UI Elements

- UI Elements
    - Input box
    - Checkbox
    - Radio buttons
    - Dropdown

- Commands
    - visit()
    - url()
    - get()
    - title()

```
/// <reference types="cypress" />
describe('UI Elements', function()
{
    it('Verify Inputbox & Radio buttons', function()
    {
        //Open URL and verify including 'newtours'
        cy.visit("http://newtours.demoaut.com/")
        cy.url().should('include', 'newtours')

        //Verify visibility of fields and type 'mercury' in the fields

cy.get('input[name=userName]').should('be.visible').should('be.enabled').type("mercury")

cy.get('input[name=password]').should('be.visible').should('be.enabled').type("mercury")

        //Click on Sign-in
        cy.get('input[name=login]').should('be.visible').click()

        //Title verification
        cy.title().should('eq', 'Find a Flight: Mercury Tours:')

        //Radio buttons
        //visibility checked status + click
        cy.get('input[value=roundtrip]').should('be.visible').should('be.checked')
        cy.get('input[value=oneway]').should('be.visible').should('not.be.checked').click()

        //Continue button click
        cy.get('input[name=findFlights]').should('be.visible').click()

        //Verify Title
        cy.title().should('eq', 'Select a Flight: Mercury Tours')

    })
       //CheckBox
    it('Hobbies check boxes', function()
    {
        cy.visit("http://demo.automationtesting.in/Register.html") //Opens the URL
        cy.get('#checkbox1').check().should('be.checked').and('have.value', 'Cricket')
        cy.get('#checkbox2').check().should('be.checked').and('have.value', 'Movies')
        cy.get('#checkbox3').check().should('be.checked').and('have.value', 'Hockey')

        cy.get('#checkbox1').uncheck().should('not.be.checked')
        cy.get('#checkbox2').uncheck().should('not.be.checked')
        cy.get('#checkbox3').uncheck().should('not.be.checked')

        cy.get('input[type=checkbox]').check(['Cricket', 'Hockey'])
    })
       //DropDown
    it('Skills Drop Down', function()
    {
        cy.get('#Skills').select('Android').should('have.value', 'Android')
    })

    it('Languages Multi Select', function()
    {
        cy.get('#msdd').click()
        cy.get('.ui-corner-all').contains('English').click()
        cy.get('.ui-corner-all').contains('Japanese').click()
    })

    it('Select countries searchable drop down', function()
    {
        cy.get('[role=combobox]').click({force: true})
        cy.get('.select2-search__field').type('ind')
        cy.get('.select2-search__field').type('{enter}')
    })
})
```

Handling Alerts

```
/// <reference types="cypress" />
describe('Alerts', function()
{
    it('Alerts', function()
    {
        cy.visit("https://testautomationpractice.blogspot.com/")
        cy.get('#HTML9 > div.widget-content > button').click()

        // cy.on('window:alert',(str) => {
        //     expect(str).to.equal('Please enter a valid user name!')       // window:alert
        // })

        cy.on('window:confirm',(str) => {
            expect(str).to.equal('Press a button!')                          // window:confirm
        })
    })
})
```

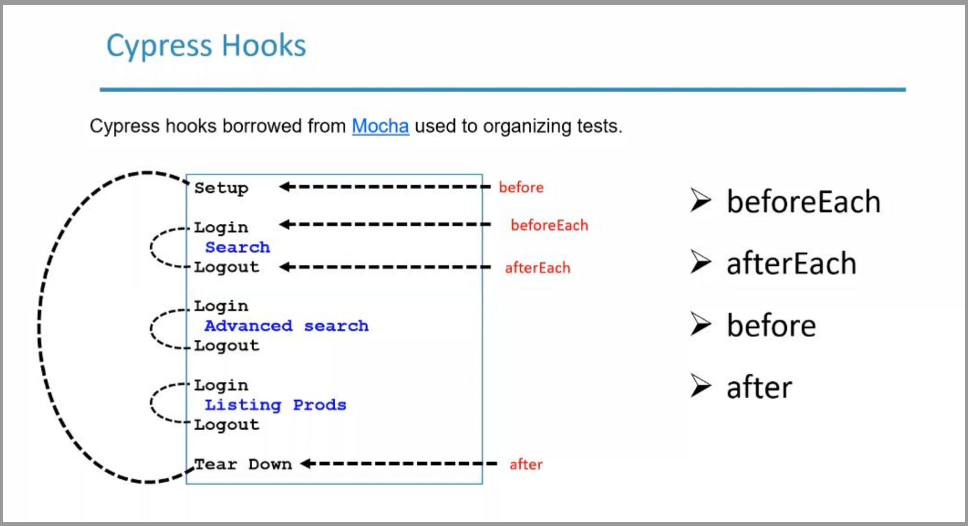
Handling Web/ HTML Table

- Check Value presence anywhere in the table
- Check Value presence in specific row & column
- Check Value presence based on condition by iterating rows

```
/// <reference types="cypress" />
describe('Table', function()
{
    it('Table test', function()
    {
        cy.visit('https://testautomationpractice.blogspot.com/')

        //(1) Check value presene anywhere in the table
        cy.get('table[name=BookTable]').contains('td', 'Learn Selenium').should('be.visible')

        //(2) Check Value presence in a specific row & column
        cy.get('table[name=BookTable] > tbody > tr:nth-child(2) > td:nth-child(3)').contains('Selenium').should('be.visible')

        //(3) Verify the book name "Master in Java" whose author is Amod
        cy.get('table[name=BookTable] > tbody > tr:nth-child(2)').each(($el, index, $list) => {

            const text=$el.text()
            if(text.includes("Amod"))
            {
                cy.get('table[name=BookTable] > tbody > tr:nth-child(1)').eq(index).then(function(bname)
                {
                    const bookName=bname.text()
                    expect(bookName).to.equal("Master in Java")
                })
            }

        })
    })
})
```

Working with Cypress Hooks | beforeEach | afterEach | before | after

Cypress hooks borrowed from **Mocha** used to organize tests


Cypress Hooks

Cypress hooks borrowed from Mocha used to organizing tests.

```
describe('Hooks', () => {
    before(() => {
      // runs once before all tests in the block
      cy.log('*** Setup Block ***')
    })

    beforeEach(() => {
      // runs before each test in the block
      cy.log('*** LOGIN Block ***')
    })

    afterEach(() => {
      // runs after each test in the block
      cy.log('*** LOGOUT Block ***')
    })

    after(() => {
      // runs once after all tests in the block
      cy.log('*** Tear Down Block ***')
    })
```

Cypress Fixture (Test Data)

Load fixed set of data located in a file

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

describe('MyTestSuite', function() {

    before(function() {
        cy.fixture('example').then(function(data){
            this.data=data
        })
    })
        //Takes data from example.json
        it('FixturesTest-example.jsom', function(){
            cy.visit('https://admin-demo.nopcommerce.com/login')
            cy.get('input[name=Email]').clear().type(this.data.email) //email
            cy.get('input[name=Password]').type(this.data.password) //password
            cy.get('input[type=submit]').click() //signin
        })
        //HardCoded
        it('FixturesTest-hardcoded', function(){
            cy.visit('https://admin-demo.nopcommerce.com/login')
            cy.get('input[name=Email]').clear().type('admin@yourstore.com') //email
            cy.get('input[name=Password]').clear().type('admin') //password
            cy.get('input[type=submit]').click() //signin
        })

})
```

How to create Custom commands in Cypress (support - > commands.js)

Example: Login steps put into command.js (create reusable steps for login) - Custom command

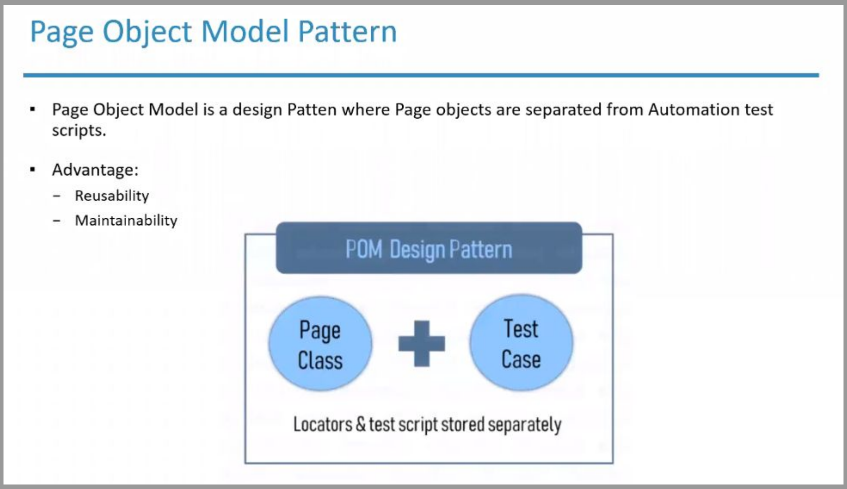
Inside support\command.js type:

```
Cypress.Commands.add("login", (email, password) => {

    cy.visit('https://admin-demo.nopcommerce.com/login')
    cy.get('input[name=Email]').clear().type(email) //email
    cy.get('input[name=Password]').clear().type(password) //password
    cy.get('input[type=submit]').click() //signin
})
```

Inside example\customcommands.js type:

```
/// <reference types="cypress" />
describe('CustomSuite', function()
{
    it('LoginTest', function()
    {
        cy.login('admin@yourstore.com', 'admin')
        cy.title().should('be.equal', 'Dashboard / nopCommerce administration')

        cy.login('admin@yourstore.com', 'admin123')
        cy.title().should('be.equal', 'Your store. Login')

        cy.login('admin@yourstore123.com', 'admin')
        cy.title().should('be.equal', 'Your store. Login')
    })

    it('Add customer', function()
    {
        cy.login('admin@yourstore.com', 'admin')
        cy.log('Adding customer')
    })

    it('Edit customer', function()
    {
        cy.login('admin@yourstore.com', 'admin')
        cy.log('Editing customer')
    })
})
```

Page Object Model (POM) - using for reusability and easy maintenance


Page Object Model Pattern

- Page Object Model is a design Patten where Page objects are separated from Automation test scripts.
- Advantage:
  - Reusability
  - Maintainability

POM Design Pattern
Page Class
Test Case
Locators & test script stored separately

In examples folder create folder PageObjects
In PageObjetect folder create LoginPage.js

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

class LoginPage
{
    visit()
    {
        cy.visit("https://admin-demo.nopcommerce.com/login")
    }

    fillEmail(value)
    {
        const field=cy.get('[id=Email]')
        field.clear()
        field.type(value)
        return this
    }

    fillPassword(value)
    {
        const field=cy.get('[id=Password]')
        field.clear()
        field.type(value)
        return this
    }

    submit()
    {
        const button=cy.get('[type=submit]')
        button.click()
    }
}
export default LoginPage
```
In examples folder create file: pageobjectpatterndemo.spec.js

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

import LoginPage from '../PageObjects/LoginPage'

describe('Test Suite', function()
{
    it('Valid Login Test', function()
    {
        const lp = new LoginPage()
        lp.visit()
        lp.fillEmail('admin@yourstore.com')
        lp.fillPassword('admin')
        lp.submit()
        cy.title().should('be.equal', 'Dashboard / nopCommerce administration')
    })
})
```

Cypress Command-Line & Dashboard Services

- How to run Cypress from the command-line
- How to specify which spec files to run
- Working with Dashboard features
    - Capture screenshot
    - Recording

Go to Spec-Runner
Go to Runs Tab
Click: Set up project to record
Login to the Dashboard using GitHub or Google
After login, Set up project window is open
Click: Set up project button after selecting public/private radio buttons
Automatically get projectId, and run command with key
Go to cypress.json file and check if there is projectId

```
{
  "projectId": "um8ttp"
}
```

```
cypress run --record --key **58d11d-****-4c5b-****-e29c4972ae****
```

Run from CMD locally:
```
node_modules\.bin\cypress run --record --key **58d11d-****-4c5b-****-e29c4972ae****
```

Note: “projectId” and “key” are imortant for CI

Run from CMD, and Integration with Jenkins

Go to package.json file

```
{
  "name": "cypressautomation",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "node_modules\\.bin\\cypress run --config pageLoadTimeout=100000",
    "runtest":"npm run test --"
  },
  "author": "furisha",
  "license": "ISC",
  "devDependencies": {
    "cypress": "^4.8.0"
  }
}
```

Note: or execute only one spec.js:

```
"test": "node_modules\\.bin\\cypress run --spec cypress\\integration\\examples\\firsttest.js
--config pageLoadTimeout=100000",
```

C:\Users\******\CypressAutomation\npm run runtest

Start Jenkins:
C:\Jenkins>java -jar jenkins.war
In Browser open loachost: http://localhost:8080/
Go to the New Item
Enter an item name: CypressProject
OK
In Configuration Check: “Use custom workspace”
Inside Directory insert path to your project: “C:\Users\******\CypressAutomation”
Scroll down to Build section
Click on add build step and in dropdown menu select: “Execute windows batch command”
Inside Command TexBox insert: npm run runtest
Apply/Save
Build Now

Note: if we have code already on GitHub then srcoll to Source Code Management section
Select Git

Shortcuts

```
C:\Users\******\CypressAutomation\
node_modules\.bin\cypress run
node_modules\.bin\cypress open
node_modules\.bin\cypress run --headed
node_modules\.bin\cypress run --spec "cypress\integration\examples\firsttest.js"
node_modules\.bin\cypress run --record --key **58d11d-****-4c5b-****-e29c4972ae**
node_modules\.bin\cypress run --browser chrome
```