Cypress Automated Testing

Javascript Framework for Automated Testing (Day2)

Agenda

- Day1
  - Automated Testing คืออะไร
  - Manual Testing vs Automated Testing
  - Automated Testing Tools ในตลาดที่ได้รับความนิยม
  - Cypress's feature
  - ข้อดี ข้อเสีย Cypress
  - แนะนำเครื่องมือต่าง ๆ และติดตั้ง

Agenda

- Day1
  - แนะนำโครงสร้างของ Project Cypress
  - run Cypress แนะนำ tag scripts ต่าง ๆ ที่จำเป็นต้องรู้
  - browser config
  - command line
  - Selector DOM Elemetns
  - API Commands
  - Catalog of Events

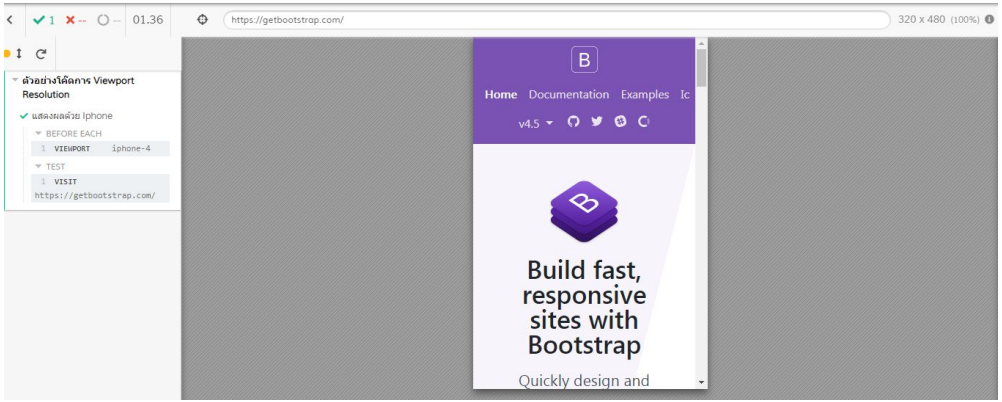
Agenda

- Day2
  - API Commands (ต่อ)
  - Spy ,Stub
  - Assertions (should ,expect)
  - Utilities (lodash ,jQuery ,momentJS , ...)
  - Custom Cypress Commands
  - Cookies

API Commands (examples)

- viewport

```
cy.viewport(width, height)
cy.viewport(preset, orientation)
cy.viewport(width, height, options)
cy.viewport(preset, orientation, options)

cy.viewport(550, 750)     // Set viewport to 550px x 750px
cy.viewport('iphone-6')   // Set viewport to 375px x 667px
```



API Commands (examples)

- should

```
.should(chainers)
.should(chainers, value)
.should(chainers, method, value)
.should(callbackFn)

cy.get('.error').should('be.empty')                    // Assert that '.error' is empty
cy.contains('Login').should('be.visible')               // Assert that el is visible
cy.wrap({ foo: 'bar' }).its('foo').should('eq', 'bar') // Assert the 'foo' property
equals 'bar'

cy.get('option:first').should('be.selected').then(($option) => {
  // $option is yielded
})

cy.get('#btn-focuses-input').click()
cy.get('#input-receives-focus').should('have.focus') // equivalent to
should('be.focused')
```

Assertions (should,expect)

- Cypress bundles the popular Chai assertion library, as well as helpful extensions for Sinon and jQuery, bringing you dozens of powerful assertions for free.
  - BDD Assertions
  - TDD Assertions
  - Chai-jQuery
  - Sinon-Chai
    - spy ,stub
  - Common Assertions
    -

Common Assertions (1/2)

```
//length
        cy.get('li.selected').should('have.length', 3)
//class
        cy.get('form').find('input').should('not.have.class', 'disabled')
//value
        cy.get('textarea').should('have.value', 'foo bar baz')
//text content
        cy.get('a').parent('span.help').should('not.contain', 'click me')
//visibled
        cy.get('button').should('be.visible')
//existence
        cy.get('#loading').should('not.exist')
//state
        cy.get(':radio').should('be.checked')
//css
        cy.get('.completed').should('have.css', 'text-decoration', 'line-through')
```

Common Assertions (2/2)

```
// Should callback
<div class="main-abc123 heading-xyz987">Introduction</div>

cy.get('div')
  .should(($div) => {
    expect($div).to.have.length(1)

    const className = $div[0].className

    // className will be a string like "main-abc123 heading-xyz987"
    expect(className).to.match(/heading-/)
  })
```

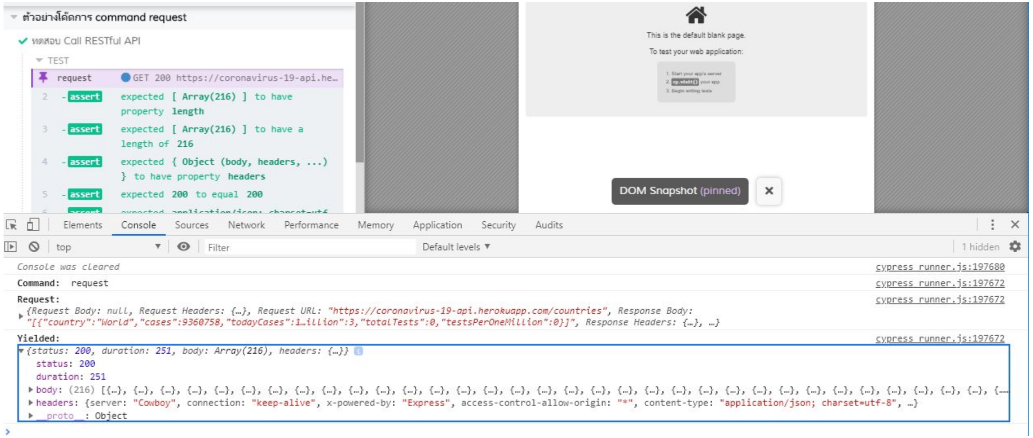
API Commands (examples)

- request

```
cy.request(url)
cy.request(url, body)
cy.request(method, url)
cy.request(method, url, body)
cy.request(options)

cy.request('GET','/table.html').then(res => {
        expect(res.body).contains('Today Cases')
        expect(res).to.have.property('headers')
        expect(res).to.have.property('duration')
        expect(res.status).to.eq(200)
})
cy.request({
  method: 'POST',
  url: '/login_with_form', // baseUrl is prepended to url
  form: true, // indicates the body should be form urlencoded and sets
Content-Type: application/x-www-form-urlencoded headers
  body: { username: 'jane.lane', password: 'password123'}
})
```

workshop#1 (/day2/exercise1.spec.js)

- ใช้ command request call URL https://coronavirus-19-api.herokuapp.com/countries และ assertions cases ต่าง ๆ
    - content-type : application/json
    - header statuscode : 200



API Commands (examples)

- spy

```
cy.spy(object, method)

const obj = {
  foo () {}
}
const spy = cy.spy(obj, 'foo').as('anyArgs')
const withFoo = spy.withArgs('foo').as('withFoo')

obj.foo()
expect(spy).to.be.called
expect(withFoo).to.be.called
// purposefully failing assertion
```

API Commands (examples)

- stub

```
cy.stub()
cy.stub(object, method)
cy.stub(object, method, replacerFn)

const obj = {
  foo () {}
}
const stub = cy.stub(obj, 'foo').as('anyArgs')
const withFoo = stub.withArgs('foo').as('withFoo')

obj.foo()
expect(stub).to.be.called
expect(withFoo).to.be.called // purposefully
failing assertion
```

Example Code Spy & Stub

```
cy.visit('/window.html',{
    onBeforeLoad(win){
     //cy.stub(win, 'prompt').returns('my custom message')
     cy.stub(win,'prompt',()=>{
         return 'my custom message'
     }).as('stubPrompt')
     cy.spy(win,'alert',()=>{

     }).as('stubAlert')
    },
    onLoad(win){
     cy.stub(win,'confirmFunction',()=>{
         cy.log('called custom function confirmFunction')
     })
    }
})
```

ข้อแตกต่างของ cy.spy() และ cy.stub()

Stub ในเชิง Computer Science นั้นคือโค้ดที่เราเขียนเพื่อจำลอง Behavior ของโปรแกรมหรือโค้ดตัวจริง (เช่น Function, Module ต่างๆ) เพื่อใช้สำหรับทำเทสโดยเฉพาะ ให้ผลลัพธ์ด้วยค่าที่กำหนดไว้ (Canned Answers) โดย Stub จะถูกเรียกใช้แทนโค้ดจริงเพื่อควบคุม Behavior ของแอพให้เป็นไปอย่างที่เราต้องการ เพื่อลด Dependencies ต่างๆ

ส่วน **Spy** เพื่อใช้สำหรับทำเทสโดยเฉพาะ ให้ผลลัพธ์ด้วยค่าที่กำหนดไว้ (Canned Answers) โดย Stub จะถูกเรียกใช้แทนโค้ดจริงเพื่อควบคุม Behavior ของแอพให้เป็นไปอย่างที่เราต้องการ เพื่อลด Dependencies ต่างๆSpy คล้ายๆ กับ Stub แต่ว่า Spy นั้นจะไม่มีการเปลี่ยนแปลงหรือควบคุม Behavior การทำงานของโค้ดตัวจริงแต่อย่างใด แต่จะคอยแฝงตัวและ Record&Capture การเรียกใช้งานโค้ดตัวจริงนั้น

workshop#2 (/day2/exercise2.spec.js)

- แก้ไข testscript ให้ทำงานได้ตามที่โจทย์กำหนด
  - เขียน testscript เมื่อกดปุ่ม Login และแสดง alert “Login Successfully." โดยไม่ต้องกรอก username และ password (ไม่ต้องกรอกค่าก็ Login ผ่าน)

Utillities (integrate plugin&library)

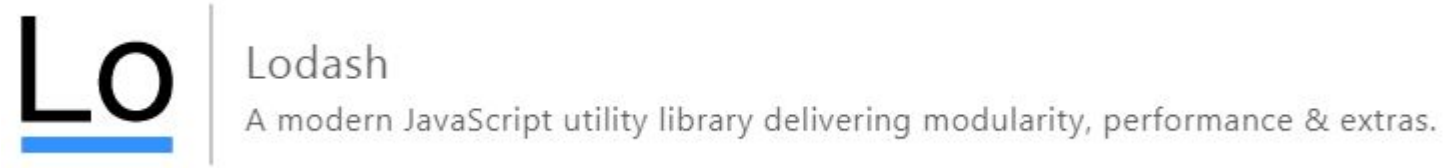
- _
- $
- Blob
- minimatch
- moment
- Promise
- sinon

Utillities (integrate plugin&library)

- _ (lodash)

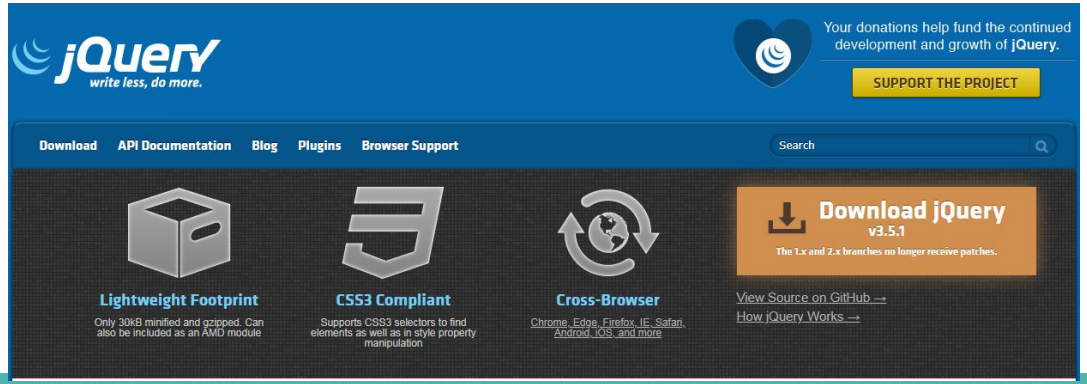
  Cypress._.method()

  Cypress._.keys(obj)



Utillities (integrate plugin&library)

- $ (jQuery)
  - Cypress.$(selector)
  - Cypress.$.Event
  - Cypress.$.Deferred
  - Cypress.$.ajax
  - Cypress.$.get
  - Cypress.$.getJSON
  - Cypress.$.getScript
  - Cypress.$.post



Utillities (integrate plugin&library)

- Blob (Blob)

```
Cypress.Blob.method()

cy.fixture('images/logo.png').as('logo')

cy.get('input[type=file]').then(function($input) {

  return Cypress.Blob.base64StringToBlob(this.logo, 'image/png')

    .then((blob) => {

      $input.fileupload('add', { files: blob })

    })

})
```

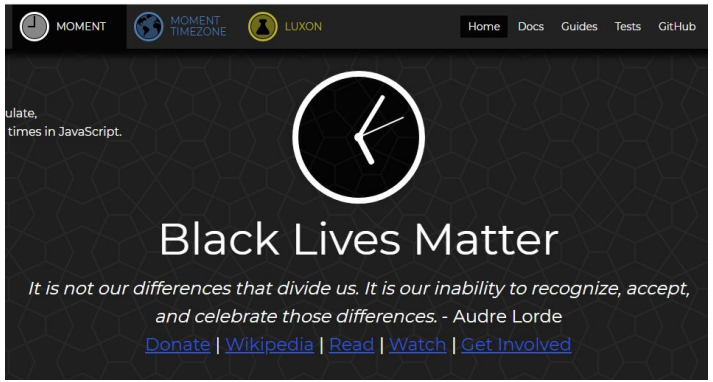
Utillities (integrate plugin&library)

- moment [(Moment.js)]

Cypress.moment()

const todaysDate = Cypress.moment().format('MMM DD, YYYY')

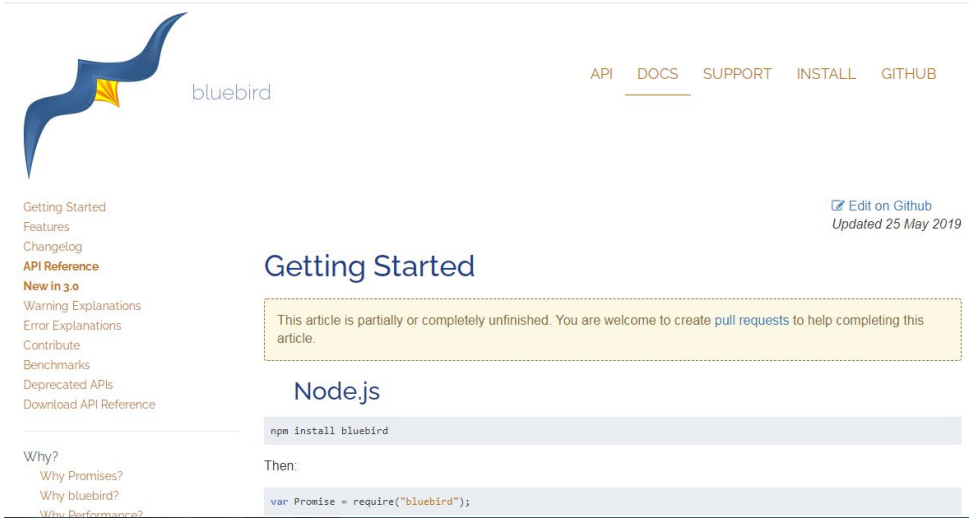
cy.get('span').should('contain', 'Order shipped on: ' + todaysDate)



Utillities (integrate plugin&library)

- Promise (Bluebird)

```
new Cypress.Promise(fn)

cy.get('button').then(($button) => {

  return new Cypress.Promise((resolve, reject) => {

    // do something custom here

  })

})
```



workshop#3 (/day2/exercise3.spec.js)

- แก้ไข testscript ให้ทำงานได้ตามที่โจทย์กำหนด
  - เขียน assertions ตรวจสอบ /index.html
    - วันที่ในอดีต 15 ปี
    - วันเดือนปี ปัจจุบัน
    - วันเดือนปีในอนาคต 20 ปี

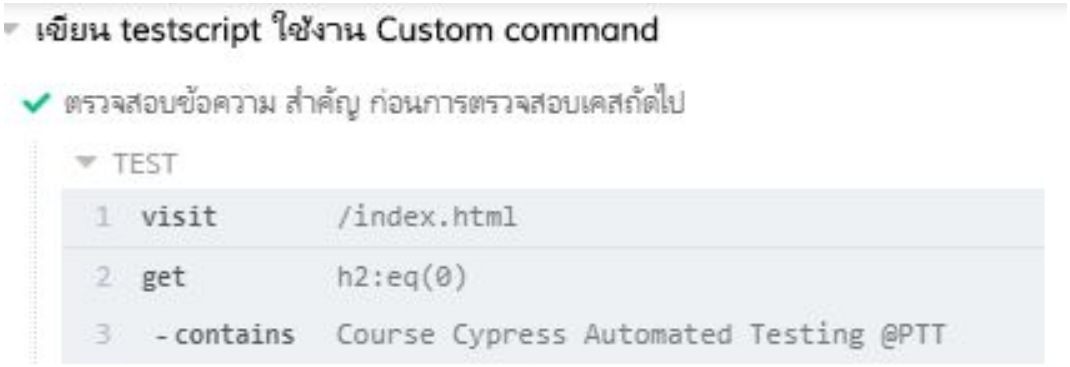
My custom commands

Cypress comes with its own API for creating custom commands and overwriting existing commands. The built in Cypress commands use the very same API that's defined below.

```
Cypress.Commands.add(name, callbackFn)
Cypress.Commands.add(name, options, callbackFn)
Cypress.Commands.overwrite(name, callbackFn)

Cypress.Commands.add('checkToken', (token) => {
  cy.window()
    .its('localStorage.token')
    .should('eq', token)
})
cy.checkToken('abc123')

//You can add your commands to this file. : cypress/support/commands.js
```

workshop#4 (/day2/exercise4.spec.js)

- แก้ไข testscript
  - สร้าง Custom commands ของคุณเอง
  - ตรวจสอบข้อความ “Course Cypress Automated Testing @PTT” ที่ต้องมีการตรวจสอบทุก ๆ ครั้งที่เข้าแต่ละเมนู
  - โดยสร้าง command ชื่อ shouldBeTitle ที่รับ args ที่ชื่อว่า title



Cypress.Cookies

Cookies.preserveOnce() and Cookies.defaults() enable you to control Cypress' cookie behavior. Cypress automatically clears all cookies before each test to prevent state from building up.

```
Cypress.Cookies.debug(enable, options)
Cypress.Cookies.preserveOnce(names...)
Cypress.Cookies.defaults(options)

Cypress.Cookies.debug(true) // now Cypress will log out when it alters cookies

cy.clearCookie('foo')
cy.setCookie('foo', 'bar')
```

Cypress.browser

Cypress.browser returns you properties of the browser.

```
Cypress.browser // returns browser object

it('log browser info', () => {
  console.log(Cypress.browser)
  // {
  //   channel: 'stable',
  //   displayName: 'Chrome',
  //   family: 'chromium',
  //   isChosen: true,
  //   majorVersion: 80,
  //   name: 'chrome',
  //   path: '/Applications/Google Chrome.app/Contents/MacOS/Google Chrome',
  //   version: '80.0.3987.87',
  //   isHeaded: true,
  //   isHeadless: false
  // }
})
```

Cypress.config

get and set configuration options in your tests.

```
{
  "defaultCommandTimeout": 10000
}
Cypress.config() // => {defaultCommandTimeout: 10000, pageLoadTimeout: 30000, ...}
```

Cypress.dom

Cypress.dom.method() is a collection of DOM related helper methods.

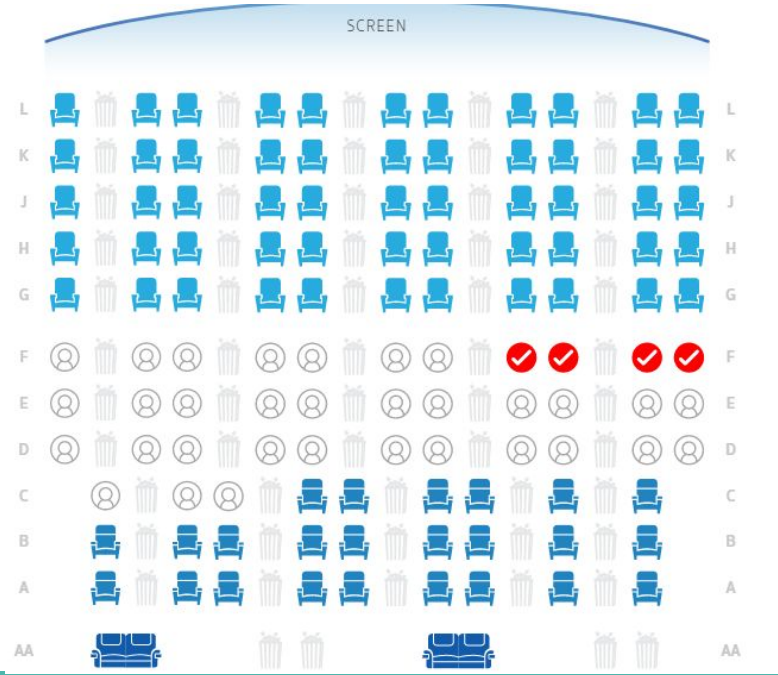
```
Cypress.dom.isHidden(element)

cy.get('body').then(($el) => {
  Cypress.dom.isDom($el) // true
})
cy.get('input').then(($el) => {
  Cypress.dom.isFocusable($el) // true
})
```

Cypress other APIs

- Cypress.isBrowser
- Cypress.isCy
- Cypress.log
- Cypress.platform
- Cypress.spec
- Cypress.version
-

workshop#5 (/day2/exercise5.spec.js)

- workshop สุดท้าย ใส่พลังอย่างเต็มที่
  - สร้าง testscript จองตั๋วภาพยนต์ ตั้งแต่เริ่มแรก จนถึงหน้ายืนยันการจ่ายเงิน
    - เลือกภาพยนต์เรื่อง “Trolls World Tour”
    - เลือกจอง วันพรุ่งนี้
    - เลือกเวลา (กำหนดเอง ตามใจคุณ)
    - เลือกโซน กทม.
    - เลือกที่นั่ง 4 ที่นั่ง (Zone D,E,F,G โซนละที่นั่ง)
    -



Resources link website for Cypress

- cypress.io
- medium.com cypress-io-thailand
- facebook cypress-io-thailand
- github cypress official