
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 Elements
 - 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)

- 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')
//visible
  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)
})

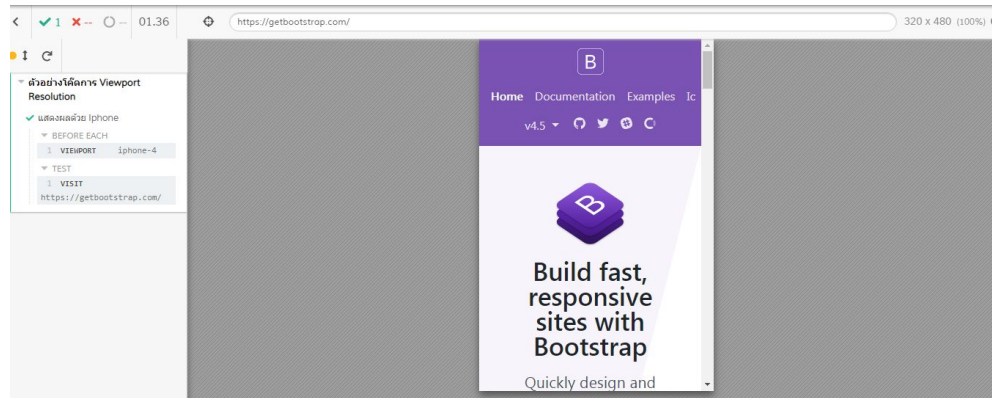
cy.request({
  method: 'POST',
  url: '/login_with_form', // baseUrl is prepended to url
  form: true, // indicates the body should be form urlencoded and sets
  Content-Type: application/x-www-form-urlencoded headers
  body: { username: 'jane.lane', password: 'password123' }
})
```

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)

- fixture

```
cy.fixture(filePath)
cy.fixture(filePath, encoding)
cy.fixture(filePath, options)
cy.fixture(filePath, encoding, options)

cy.fixture('users/admin.json') // Get data from
{fixturesFolder}/users/admin.json
```


Spy & Stub

The Cypress logo, featuring the word "cypress" in a white, lowercase, sans-serif font. The "c" is enclosed within a white circle. The logo is set against a dark blue circular background.

cypress



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
```

The screenshot shows the Cypress test runner interface for a test titled "aliasing spies". It displays a table of spies and stubs, followed by a "BEFORE EACH" block and a "TEST" block. The test block shows two assertions: the first passes, and the second fails with an "AssertionError".

Type	Function	Alias(es)	# Calls
spy-1	foo	anyArgs	1
spy-1.1	foo	withFoo	-

BEFORE EACH

1 VISIT /test.html

TEST

(SPY-1) foo() anyArgs

1 **ASSERT** expected foo to have been called at least once

2 **ASSERT** expected withFoo to have been called at least once, but it was never called

AssertionError: expected withFoo to have been called at least once, but it was never called

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
```



The screenshot shows the Cypress test runner interface. At the top, there is a red 'x' icon and the text 'aliasing stubs'. Below this, there is a section for 'SPIES / STUBS (2)' which contains a table with the following data:

Type	Function	Alias(es)	# Calls
stub-1	foo	anyArgs	1
stub-1.1	foo	withFoo	-

Below the table, there is a section for 'BEFORE EACH' with one test: '1 VISIT /test.html'. Underneath that is a section for 'TEST' with two assertions:

- 1 **ASSERT** (STUB-1) foo() anyArgs: expected foo to have been called at least once
- 2 **ASSERT**: expected withFoo to have been called at least once, but it was never called

The second assertion is highlighted in red, indicating it failed. At the bottom of the screenshot, there is a yellow box containing the error message: 'AssertionError: expected withFoo to have been called at least once, but it was never called'.

ข้อแตกต่างของ `cy.spy()` และ `cy.stub()`

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

ส่วน **Spy** เพื่อใช้สำหรับทำเทสโดยเฉพาะ ให้ผลลัพธ์ด้วยค่าที่กำหนดไว้ (Canned Answers) โดย Stub จะถูกเรียกใช้แทนโค้ดจริงเพื่อควบคุม Behavior ของแอปให้เป็นไปอย่างที่เราต้องการ เพื่อลด Dependencies ต่างๆ Spy คล้ายๆ กับ Stub แต่ว่า Spy นั้นจะไม่มี การเปลี่ยนแปลงหรือควบคุม Behavior การทำงานของโค้ดตัวจริงแต่อย่างใด แต่จะคอยแฝงตัวและ Record&Capture การเรียกใช้งานโค้ดตัวจริงนั้น

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')
    })
  }
})
```

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

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

```
▼ เขียน testscript จำลองการใช้งาน Spy & Stub
✓ เพิ่ม stub เพื่อให้การทำงาน ผ่าน แสดง "Login Successfully."
  ▶ SPIES / STUBS (1)
    ▼ TEST
      1 visit      /login.html
      2 get        #login
      3 - submit
      (stub-1)    isRequired("username")
      (stub-1)    isRequired("password")
      (alert)     Login Successfully.
      4 assert   expected Login Successfully. to equal
                    Login Successfully.
```

Utilities (integrate plugin&library)

- `—`
- `$`
- `Blob`
- `minimatch`
- `moment`
- `Promise`
- `sinon`

Utilities (integrate plugin&library)

- [\(lodash\)](#)

Cypress.__.method()

Cypress.__.keys(obj)



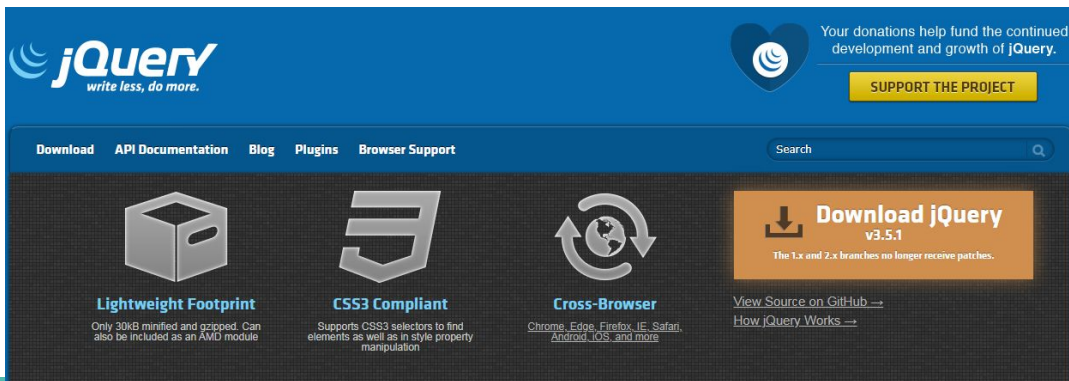
Lodash

A modern JavaScript utility library delivering modularity, performance & extras.

Utilities (integrate plugin&library)

- \$ (jQuery)

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



The screenshot shows the jQuery website homepage. At the top left is the jQuery logo with the tagline "write less, do more.". To the right is a heart icon and the text "Your donations help fund the continued development and growth of jQuery." with a yellow "SUPPORT THE PROJECT" button. Below this is a navigation bar with links for "Download", "API Documentation", "Blog", "Plugins", and "Browser Support", along with a search box. The main content area features three icons: a box for "Lightweight Footprint", a stylized 'E' for "CSS3 Compliant", and a globe with arrows for "Cross-Browser". A large orange button says "Download jQuery v3.5.1" with the note "The 1.x and 2.x branches no longer receive patches." Below the button are links for "View Source on GitHub" and "How jQuery Works".

Utilities (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 })  
  
  })  
  
})
```

Playground for blob-util

Welcome to the playground for [blob-util](#)

Below is a little Kirby GIF you can play around with.



Here's some code to get you started. Copy-paste this into your console:

```
var img = document.getElementById('kirby');  
blobUtil.imgSrcToBlob(img.src).then(function(blob) {  
  var blobURL = blobUtil.createObjectURL(blob);  
  
  var newImg = document.createElement('img');  
  newImg.src = blobURL;  
  
  img.parentNode.appendChild(newImg);  
});
```

If you see two Kirbys, you're on your way!

Utilities (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)
```



Utilities (integrate plugin&library)

- Promise (Bluebird)

```
new Cypress.Promise(fn)
```

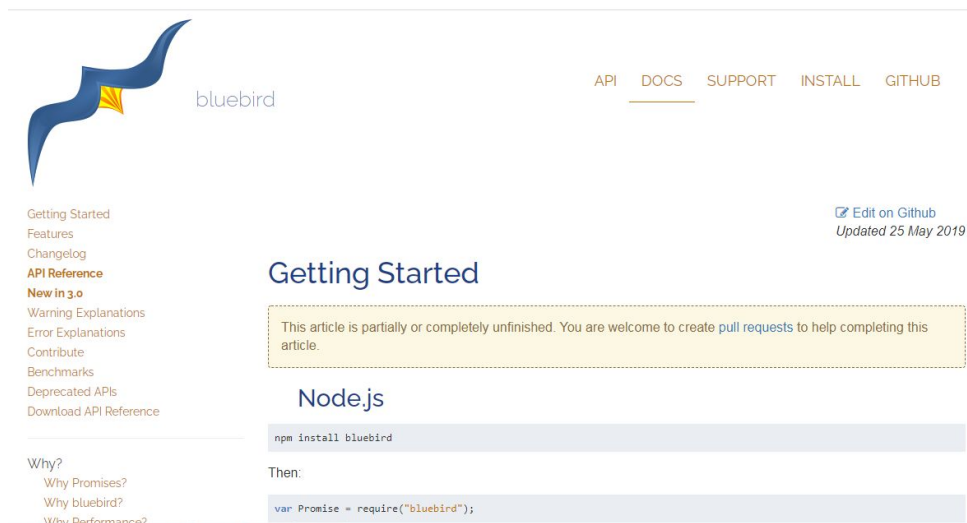
```
cy.get('button').then(($button) => {
```

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

```
    // do something custom here
```

```
  })
```

```
})
```



The screenshot shows the Bluebird.js website. At the top left is the Bluebird logo, a blue bird-like shape with a yellow beak. To its right is the word "bluebird". In the top right corner, there are navigation links: "API", "DOCS", "SUPPORT", "INSTALL", and "GITHUB". Below the logo is a list of navigation items: "Getting Started", "Features", "Changelog", "API Reference", "New in 3.0", "Warning Explanations", "Error Explanations", "Contribute", "Benchmarks", "Deprecated APIs", and "Download API Reference". On the right side, there is a link to "Edit on Github" with the text "Updated 25 May 2019". The main content area is titled "Getting Started" and contains a yellow warning box that says "This article is partially or completely unfinished. You are welcome to create pull requests to help completing this article." Below this is a section titled "Node.js" with a code block showing the installation command: `npm install bluebird`. Underneath, it says "Then:" followed by another code block: `var Promise = require("bluebird");`

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

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

ปี



วันและเวลา ปัจจุบัน 24/06/2020 16:45:51

วันและเวลา ย้อนหลัง 10 ปี คือ 24/06/2010 16:45:12

วันและเวลา อนาคต 10 ปี คือ 24/06/2030 16:45:12

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

เขียน testscript ใช้งาน Custom command

✓ ตรวจสอบข้อความ สำคัญ ก่อนการตรวจสอบเคสถัดไป

TEST

```
1 visit /index.html
2 get h2:eq(0)
3 -contains Course Cypress Automated Testing @PTT
```

Cypress.Cookies

`Cypress.Cookies.preserveOnce()` and `Cypress.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)

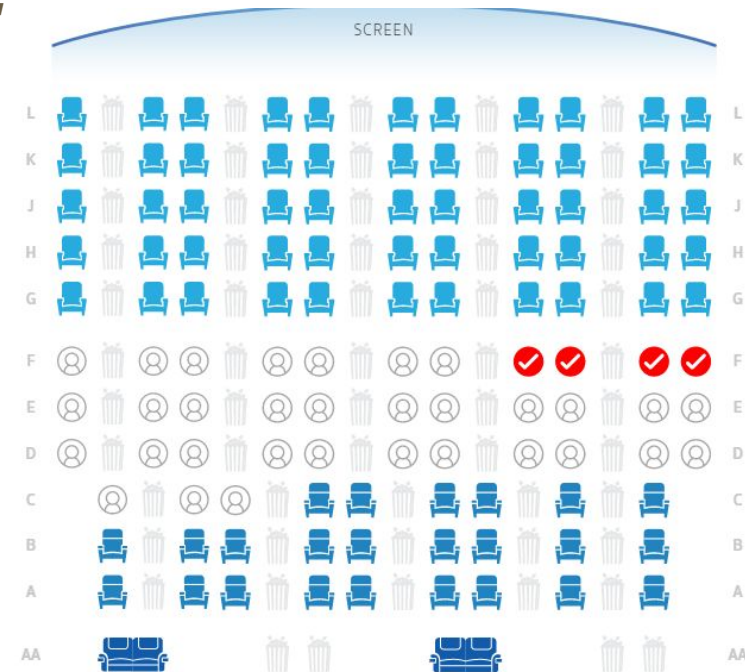
cy.get('body').then(($el) => {
  Cypress.dom.isDom($el) // true
})
cy.get('input').then(($el) => {
  Cypress.dom.isFocusable($el) // true
})
```

Cypress other APIs

- [Cypress.isBrowser](#)
- [Cypress.isCy](#)
- [Cypress.log](#)
- [Cypress.platform](#)
- [Cypress.spec](#)
- [Cypress.version](#)
-

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

- workshop สดท้าย ใส่พลังอย่างเต็มที่
 - สร้าง testscript จองตัวภาพยนตร์ ตั้งแต่เริ่มแรก จนถึงหน้ายืนยันการจ่ายเงิน
 - เลือกภาพยนตร์เรื่อง "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](https://medium.com/cypress-io-thailand)
- facebook [cypress-io-thailand](https://facebook.com/cypress-io-thailand)
- github [cypress official](https://github.com/cypress/cypress)