
Cypress Automated Testing

— Javascript Framework for
Automated Testing (Day1) —

แนะนำ

ทีมวิทยากร

พูลสวัสดิ์ อภิญ (พูล)

วิทยากร

medium.com

poolsawat.com



ภูมิ ชัยศิริ (ภูมิ)

ผู้ช่วยวิทยากร



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

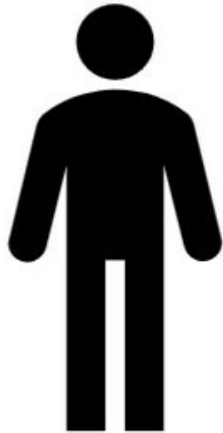
- Day2
 - API Commands (ต่อ)
 - Spy ,Stub
 - Assertions (should ,expect)
 - Utilities (lodash ,jQuery ,momentJS , ...)
 - Custom Cypress Commands
 - Cookies

Automated Testing คืออะไร

- ★ Automated testing or test automation เป็นวิธีการทดสอบ software ที่ใช้ Software tools มาช่วยในการทดสอบ เพื่อควบคุมการดำเนินการทดสอบแล้วเปรียบเทียบผลการทดสอบจริง (actual test results) กับ ผลลัพธ์ที่ที่คาดหวัง (expected results)
- ★ Tester ทำการเขียน code หรือ เขียน test script เพื่อการทดสอบโดยอัตโนมัติ โดยเลือกใช้ Automated testing tools ที่เหมาะสมมาช่วยในการเขียน test script เพื่อตรวจสอบความถูกต้องของ Software
- ★ เป้าหมายคือการดำเนินการทดสอบให้เสร็จสมบูรณ์ในเวลาที่น้อยลง การทำ regression testing ที่ไม่สิ้นเปลืองต้นทุนหรือเวลา

Difference between Manual Testing and Automated Testing

Manual Testing



VS

Automated Testing



Difference between Manual Testing and Automated Testing

Parameter	Automated Testing	Manual Testing
นิยาม(Definition)	เครื่องมืออัตโนมัติ (automation tools.)	ทดสอบด้วยตนเอง (manual testing ,human tester)
ระยะเวลาการดำเนินการ (Processing time)	เร็วกว่าทดสอบด้วยตนเอง (faster than a manual) approach	ใช้เวลานาน ใช้ทรัพยากรคนมาก (time-consuming and takes up human resources.)
การทดสอบเชิงสำรวจ (Exploratory Testing)	ไม่อนุญาตสุ่มทดสอบ (not allow random testing)	ทดสอบ เคสที่เป็นไปได้ (testing is possible)
การลงทุนระยะแรก (Initial investment)	สูง (higher)	ต่ำ (lower)
ความเชื่อถือได้ (Reliability)	น่าเชื่อถือ (reliable) ,เมื่อยล้า (Fatigue)	ความผิดพลาดจากมนุษย์ (human errors)
เปลี่ยนแปลงด้าน UI (UI Change)	มีการเปลี่ยนแปลง สามารถตรวจจับได้ทันทีตามที่คาดหวังไว้	การเปลี่ยนแปลงเล็กน้อยไม่ส่งผลต่อการทดสอบ เช่น แก้ไข id ของ element

Difference between Manual Testing and Automated Testing

Parameter	Automated Testing	Manual Testing
การลงทุน (Investment)	ต้องการเครื่องมือในการทดสอบ (need testing tools)	ต้องการทรัพยากรมนุษย์ (Investment is needed for human resources.)
รายงานการทดสอบ (Test Report Visibility)	มีรายงานรูปแบบตาราง HTML ,PDF	ถูกบันทึกใน Excel , Word ดูย้อนหลังได้ยาก
การสังเกตของคน (Human observation)	ทดสอบจะไม่มี การตรวจสอบ user-friendliness จะไม่คำนึงในเรื่องนี้	ช่วยดูและพิจารณาเรื่อง UI,UX ได้
การทดสอบประสิทธิภาพ (Performance Testing)	สามารถทำ test ได้หลากหลายแบบ Load Testing, Stress Testing, Spike Testing, etc.	ไม่สามารถทดสอบประสิทธิภาพได้จาก manual test
Parallel Execution	สามารถทำแบบทดสอบพร้อมกันได้ หลาย ๆ test พร้อมกัน	เบื้องต้นไม่สามารถทำ test พร้อมกันได้ (ต้องมีการเพิ่มทรัพยากรมนุษย์)
Batch testing	สร้าง Test script ตั้งเวลารันได้	ไม่สามารถทำได้ (เป็นไปได้อย่างยาก)

Difference between Manual Testing and Automated Testing

Parameter	Automated Testing	Manual Testing
ความรู้เรื่องโปรแกรม (Programming knowledge)	ต้องการความรู้ด้านการเขียนโปรแกรมมิ่ง (need knowledge of programming)	ไม่ต้องการความรู้ด้านโปรแกรมมิ่ง (doesn't need programming knowledge)
การสู้รบ (Engagement)	ทำงานได้ตลอดเวลา ไม่เกิดความเบื่อหน่ายในการทดสอบ (Done by tools. Its accurate and never gets bored!)	การทดสอบซ้ำ ๆ จะทำให้เกิดการเบื่อหน่ายในการทดสอบระยะยาวนาน
แนวทางในอุดมคติ (Ideal approach)	มีประโยชน์เมื่อมีชุดการทดสอบคล้าย ๆ กัน ที่ต้องทำบ่อย ๆ	การทดสอบไม่บ่อย 1 -2 ครั้ง การทดสอบด้วยตนเองก็ตอบโจทย์ในเรื่องนี้
Framework	มี framework ให้เลือกใช้งานได้มากมาย	ไม่มี framework ใช้อย่างมาตรฐานตามกฎการทดสอบที่ตกลงกันในทีม
การออกแบบการทดสอบ (Test Design)	มีแบบอย่างการพัฒนาที่ชัดเจน ตาม framework ที่ใช้งาน	ไม่มีการออกแบบเรื่องการพัฒนาการทดสอบ ทดสอบด้วยตนเอง
Devops	เข้า flow Devops (CI/CD) ทำงานอย่างรวดเร็ว	ไม่สามารถเข้า flow Devops เพราะการทดสอบด้วยตนเอง

Automated Testing Tools ในตลาดที่ได้รับความนิยม



 Katalon Studio



 TestComplete



 TRICENTIS



 cypress.io

Cypress's feature ?

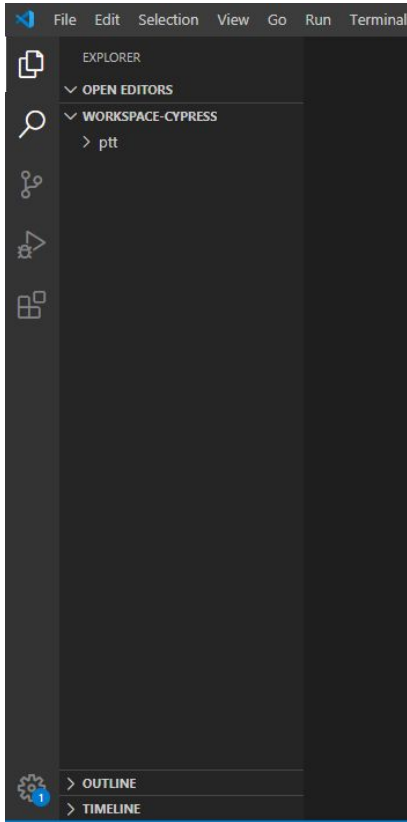
- Time Travel
- Debuggability
- Automatic Waiting
- Spies, Stubs, and Clocks
- Network Traffic Control
- Consistent Results
- Screenshots and Videos
- Cross browser Testing

ข้อดี (Advantages) / ข้อเสีย (disadvantages)

- ★ ไม่ต้องใส่ wait for แบบใน Robot Framework หรือใน selenium ให้เสียเวลา cypress ตัดสินใจให้เอง (ตาม default config ให้ไว้ 60 วินาที)
- ★ เห็น HTTP Status Code ทั้งหมด
- ★ ไม่รองรับ xpath
- ★ เร็ว เพราะไม่ต้องไปรัน webdriver แบบ selenium
- ★ ทำทดสอบได้ทุก Level
- ★ Debug ค่อนข้างง่าย
- ★ Auto Reload เวลามีการแก้ไข Test Code ตัว Cypress จะ Reload เองอัตโนมัติ
- ★ Flake Resistant เท่าที่ลองก็ล้มแทบไม่ล้มกลางทางเลย
- ★ มีฟังก์ชันเก็บรูป เก็บวิดีโอไว้ดูเพื่อตอนทดสอบเฟล

- ยังเป็นของใหม่อยู่ community ให้หาข้อมูลอาจจะยังน้อย
- รองรับเฉพาะ Web base ไม่รองรับการทำ Mobile Automated
- ไม่รองรับภาษาอื่นนอกจาก javascript
- ทำ Parallel test ได้ แต่เสียเงิน

Recommend tools for development



- NodeJs , VS Code , Browser Dev tools (F12)
 - NodeJs V > 8.0 (recommende 12.0)
 - VS Code
 - Explorer (พื้นที่การทำงาน)
 - Search
 - Source Control (git)
 - Run (run ,debug)
 - Extensions (ส่วนเสริม)
- Dev Tools(F12)
 - Insepct elements

Setup project & script run (1/3)

```
$ mkdir test-demo
```

```
$ cd test-demo
```

```
$ npm init -y
```

```
$ npm install cypress@latest --save-dev
```

```
PS D:\poola410\workspace-vs\ptt> npm install cypress@latest --save-dev  
  
> cypress@4.8.0 postinstall D:\poola410\workspace-vs\ptt\node_modules\cypress  
> node index.js --exec install  
  
Installing Cypress (version: 4.8.0)  
  
✓ Downloaded Cypress  
| Unzipping Cypress      74% 72s  
  Finishing Installation
```

Setup project & script run (2/3)

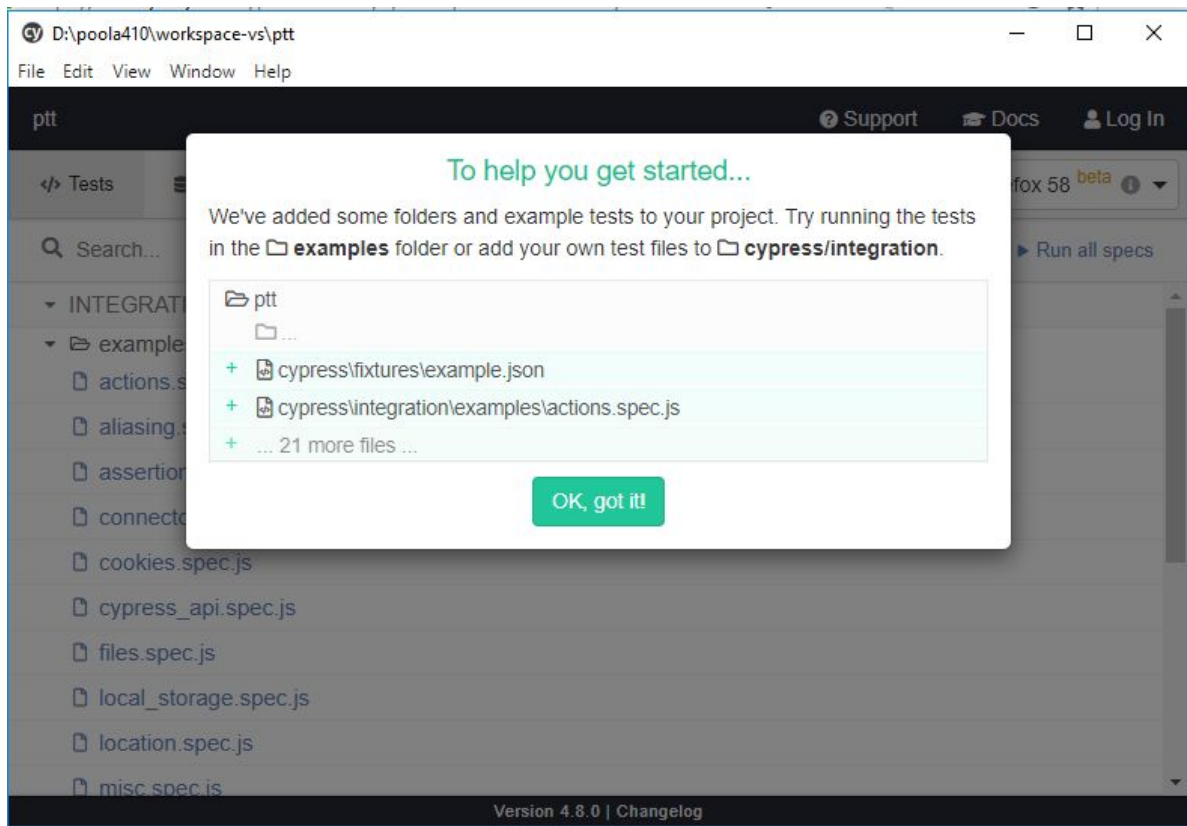
package.json

```
"scripts": {  
  "cp:open": "cypress open",  
  "cp:run" : "cypress run"  
},
```

\$ npm run cp:open

```
PS D:\poola410\workspace-vs\ptt> npm run cp:open  
  
> ptt@1.0.0 cp:open D:\poola410\workspace-vs\ptt  
> cypress open  
  
It looks like this is your first time using Cypress: 4.8.0  
  
✓ Verified Cypress! C:\Users\poola410\AppData\Local\Cypress\Cache\4.8.0\Cypress  
  
Opening Cypress...
```


Setup project & script run (3/3)



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

Introducing the Project Cypress structure.

```

  ▾ ptt
    ▾ cypress
      ▾ fixtures
        {} example.json
      > integration \ examples
      ▾ plugins
        JS index.js
      ▾ support
        JS commands.js
        JS index.js
      > node_modules
      {} cypress.json
      {} package-lock.json
      {} package.json

```

- package.json
- cypress.json
- cypress
 - fixtures (static data config)
 - integration (testscript area)
 - plugins (plugins handle)
 - support
 - commands.js (custom command)
 - index.js (index loader)

Suggesting tag scripts that you need to know

- cy `# cy.<command>`
- describe `# describe('<test-name>', ()=> {})`
- it `# it('<test-unit-name>', ()=> {})`
- expect `# expect('selector', 'assert')`
- context `# context('textgroup-name', ()=> {})`
- before `# before(()=> {})`
- beforeEach `# beforeEach(()=> {})`
- after `# after(()=> {})`
- afterEach `# afterEach(()=> {})`

Browser Config ?

- cypress open --config <args-configs>
 - cypress open --config
pageLoadTimeout=30000,baseUrl=https://myapp.com
- cypress.json
 - baseUrl (URL ของ Website)
 - port (default randomly generated port)
 - screenshotsFolder (path screenshots)
 - chromeWebSecurity (origin policy and insecure mixed content.)
 - viewportHeight (browser height ,660)
 - viewportWidth (browser width ,1000)

Command Line

- cypress ...
 - run ,open <args>
 - --spec <file-path-name>
 - --port ,-p <port>
 - --config <config-path>
 - --browser <browser-path>

```
$ cypress open
```

```
$ cypress run
```

```
$ cypress open --spec
```

```
"cypress/integration/my-spec.js"
```

```
$ cypress run --spec "cypress/integration/my-spec.js"
```

```
$ cypress open --port 8080
```

```
$ cypress open --config
```

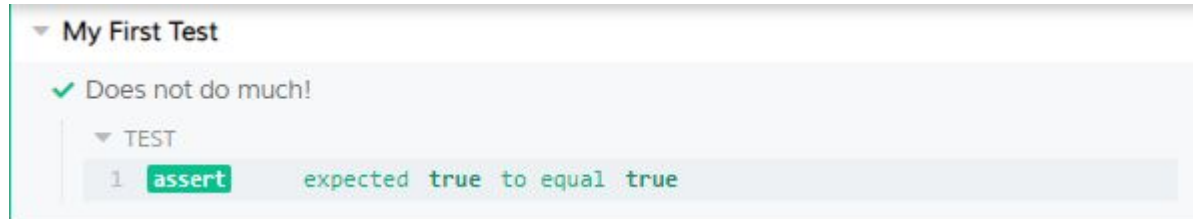
```
pageLoadTimeout=100000,watchForFileChanges=false
```

```
$ cypress open --browser chrome
```

#My First Test (passing test)

- integration
 - sample.spec.js
 - expect **true**
 - actuality **true**

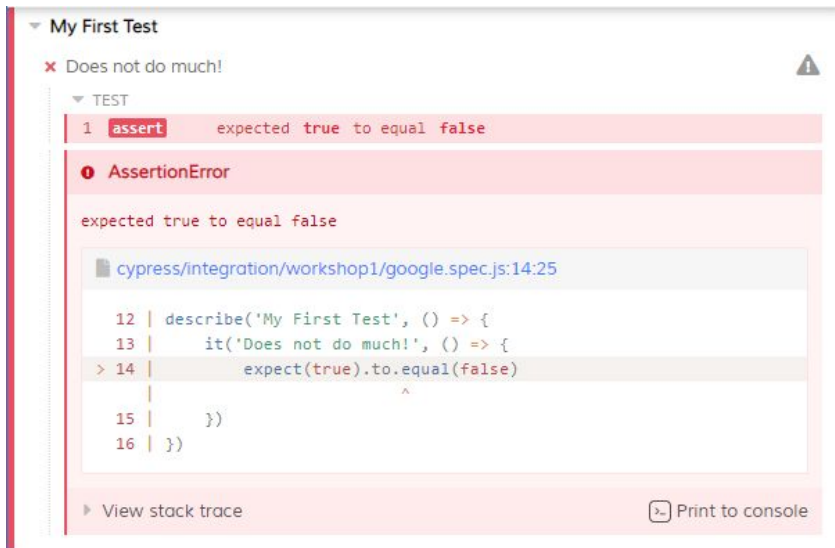
```
describe('My First Test', () => {  
  it('Does not do much!', () => {  
    expect(true).to.equal(true)  
  })  
})
```



#My First Test (failing test)

- integration
 - sample.spec.js
 - expect **true**
 - actuality **false**

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



```
My First Test  
x Does not do much!  
TEST  
1 assert expected true to equal false  
AssertionError  
expected true to equal false  
cypress/integration/workshop1/google.spec.js:14:25  
12 | describe('My First Test', () => {  
13 |   it('Does not do much!', () => {  
> 14 |     expect(true).to.equal(false)  
    |                                     ^  
15 |   })  
16 | })  
View stack trace Print to console
```


Explain the testscript code structure

- What are describe, it, and expect?
 - describe and it come from [Mocha](#)
 - expect comes from [Chai](#)

Cypress builds on these popular tools and frameworks that you hopefully already have some familiarity and knowledge of. If not, that's okay too.

Write a simple test (visit ,get ,type ,click ,contains)

```
describe('simple test script', () => {
  const keyword = 'poolsawat.com'
  it('visit google website & search poolsawat.com', () => {
    cy.visit('https://www.google.com/')
    .get('[jsaction="paste:puy29d"]').type(keyword)
    .get('.FPdoLc > center > .gNO89b').click()
    .get('a').contains(keyword, {timeout : 2000})
  })
})
```

Explain the testscript code structure (visit)

- We can pass the URL we want to visit to cy.visit().

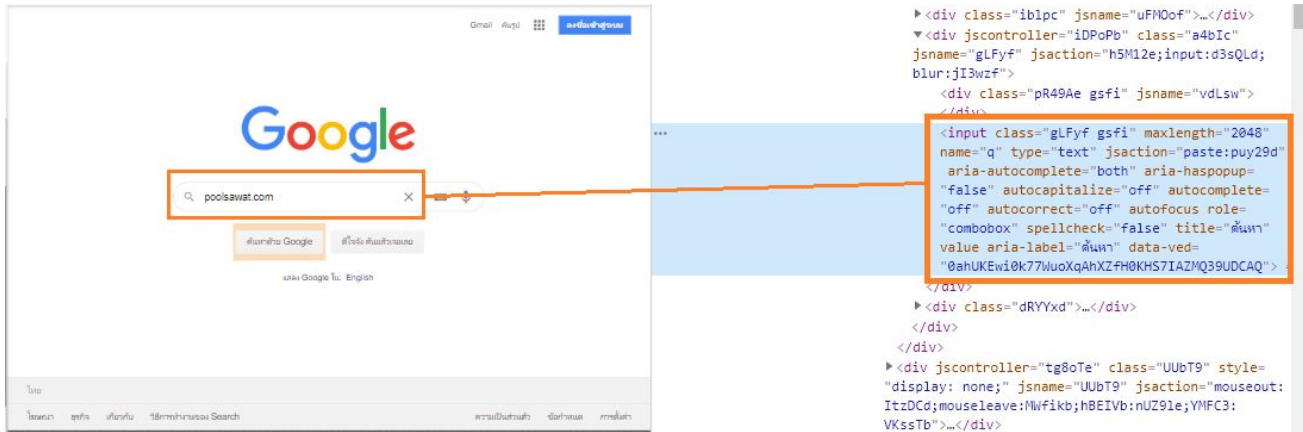
The screenshot displays the Cypress test runner interface. On the left, the test script is visible, showing a `visit` command with the URL `https://www.google.com/`. The browser view on the right shows the Google homepage with the search bar containing the text `poolsawat.com`. The `visit` command in the test script is highlighted with an orange box, and a callout line connects it to the browser's address bar, which also contains the URL `https://www.google.com/`.

```
1 visit https://www.google.com/
2 get [jsaction="paste:puy29d"]
3 -type poolsawat.com
```

Explain the testscript code structure (get,type)

- Search for the target element that requires action with `cy.get()`
- Enter text value in target element with `cy.type()`

```
cy.get('[jsaction="paste:puy29d"]').type('poolsawat.com')
```



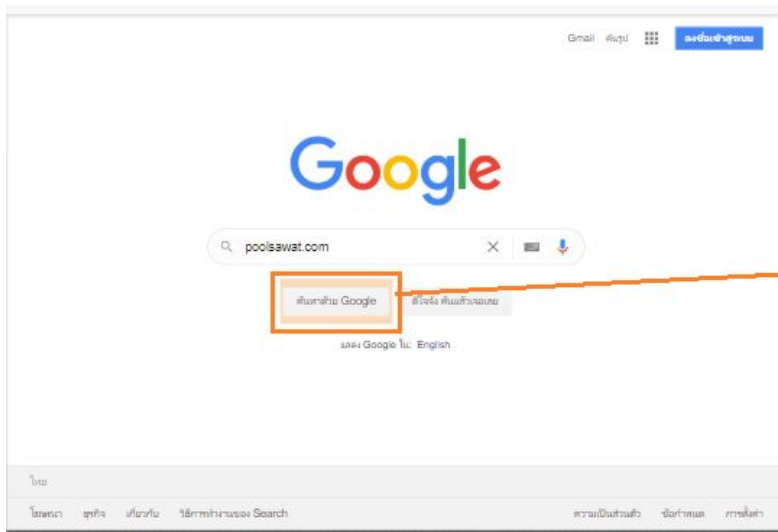
The image shows a browser window displaying the Google search page. The search bar contains the text "poolsawat.com". Below the search bar, there are buttons for "ค้นหา Google" and "ใช้ Google ที่: English". To the right of the browser window, a portion of the DOM tree is visible, showing the HTML structure of the search bar. The following code snippet is highlighted in a blue box:

```
><div class="ib1pc" jsname="UFMOof">...</div>  
▼<div jscontroller="IDPoPb" class="a4bIc"  
  jsname="gLFyf" jsaction="h5M12e;input:d3sQLd;  
  blur:jI3wzf">  
  <div class="pR49Ae gsfi" jsname="vdLsw">  
    </div>  
    <input class="gLFyf gsfi" maxlength="2048"  
      name="q" type="text" jsaction="paste:puy29d"  
      aria-autocomplete="both" aria-haspopup=  
      "false" autocapitalize="off" autocomplete=  
      "off" autocorrect="off" autofocus role=  
      "combobox" spellcheck="false" title="ค้นหา"  
      value aria-label="ค้นหา" data-ved=  
      "0ahUKewi0k77WuoXqAhZfH0KHS7IAZMQ39UDCAQ">  
    </div>  
  <div class="dRYYxd">...</div>  
</div>  
><div jscontroller="tg8oTe" class="UUbT9" style=  
  "display: none;" jsname="UUbT9" jsaction="mouseout:  
  ItzDCd;mouseleave:Mvfkfb;hBEIVb:nUZ91e;YMF3:  
  VKssTb">...</div>
```

Explain the testscript code structure (get,type)

- Search for the target element that requires action with `cy.get()`
- trigger click event on target element with `cy.click()`

```
cy.get('.FPdoLc > center > .gNO89b').click()
```

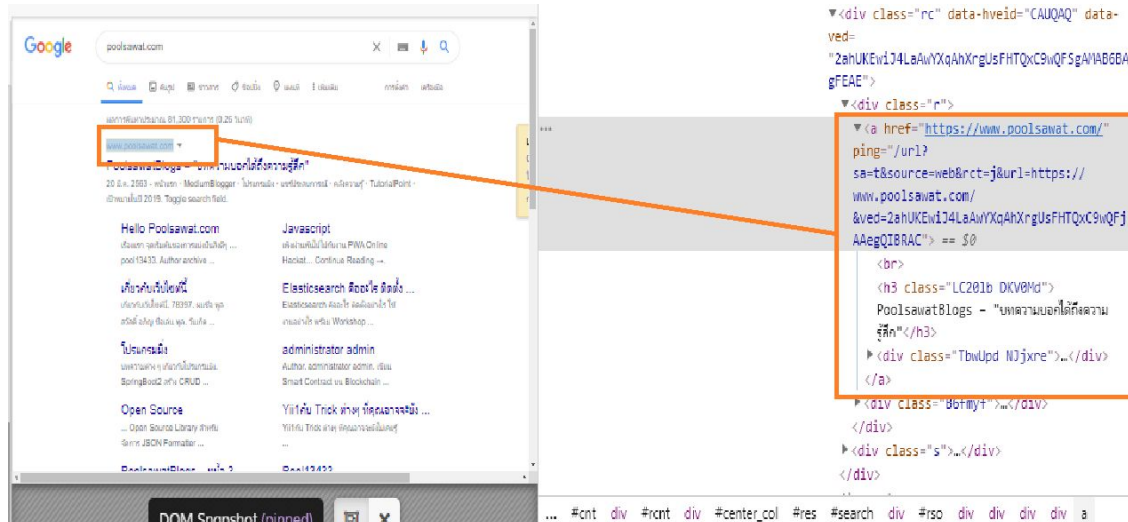


```
    <div class="gKYYX0">...</div>
  </div>
</div>
<div jscontroller="tg8oTe" class="UUbT9" style="display: none;" jsname="UUbT9" jsaction="mouseout: ItzDCd;mouseleave:MMf1kb;hBEIVb:nUZ91e;YMFC3: VKssTb">...</div>
<div class="FPdoLc tfB0Bf">
  <center>
    <input class="gNO89b" value="ค้นหาด้วย Google" aria-label="ค้นหาด้วย Google" name="btnK" type="submit" data-ved="0ahUKEwi0k77WuoXqAhXZfH0KHS7IAZMQ4dUDCA" data-cypress-el="true" == $0
    <input class="RNmpXc" value="ดูใจจ้ด ค้นหาด้วย Google" aria-label="ดูใจจ้ด ค้นหาด้วย Google" name="btnI" type="submit" jsaction="sf.lck" data-ved="0ahUKEwi0k77WuoXqAhXZfH0KHS7IAZMQ19QECAw">
  </center>
</div>
</div>
<div style="background:url(/images/searchbox/desktop_searchbox_sprites302_hr.png)"> </div>
```

Explain the testscript code structure (contains)

- Get Dom element and check text with `cy.contains()` ,pass in an options object }

```
cy.get('a').contains('poolsawat.com',{timeout : 2000,matchCase : true})
```



Get one or more DOM elements by selector

- The querying behavior of this command matches exactly how `$(...)` works in jQuery.
- Get the tag element
 - `cy.get('input').should('be.disabled')`
- Find the first li descendent within a ul
 - `cy.get('ul li:first').should('have.class', 'active')`
- Find the dropdown-menu and click it
 - `cy.get('.dropdown-menu').click()`
- Find 5 elements with the given data attribute
 - `cy.get('[data-test-id="test-example"]').should('have.length', 5)`
- Find the link with an href attribute containing the word "questions" and click it
 - `cy.get('a[href*="questions"]').click()`
-

Clone Project

- git clone https://github.com/pool13433/cypress-demo-testsuite.git
- cd cypress-demo-testsuite
- npm install
- npm run cp:open
-

**** Run VSCode with Administrator privileges.**

```
> cypress@4.9.0 postinstall D:\poola410\workspace-vs\workspace-cypress\cypress-demo-testsuite\node_modules\cypress
> node index.js --exec install
```

```
Installing Cypress (version: 4.9.0)
```

- Downloading Cypress 83% 10s
- Unzipping Cypress
- Finishing Installation

Get one or more DOM elements by selector

- Get the tag element
 - `cy.get('input').should('be.disabled')`
- Find the first li descendent within a ul
 - `cy.get('ul li:first').should('have.class', 'active')`
- Find the dropdown-menu and click it
 - `cy.get('.dropdown-menu').click()`
- Find 5 elements with the given data attribute
 - `cy.get('[data-test-id="test-example"]').should('have.length', 5)`
- Find the link with an href attribute containing the word "questions" and click it
 - `cy.get('a[href*="questions"]').click()`
-

Get one or more DOM elements by selector

```
describe('แสดง Testscript การ Selector DOM Element ด้วยวิธีการต่าง ๆ', () => {
  it('selector by ID=>#', () => {
    cy.visit('/register.html').get('#firstName').type('firstName').get('#lastName').type('lastName')
  });
  it('selector by ClassName=>.cssClassName', () => {
    cy.visit('/register.html').get('.firstName').type('firstName').get('.lastName').type('lastName')
  });
  it('selector by Attribute=>[data-attr=""]', () => {
    cy.visit('/register.html')
      .get('[data-attr="firstName"]').type('firstName').get('[data-attr="lastName"]').type('lastName')
      .log('[data-attr="firstName"]')
      .get('[name="firstName"]').clear().type('firstName').get('[name="lastName"]').clear().type('lastName')
      .log('[data-attr="firstName"]')
  });
  it('selector by tag=>h1 ,p ,span', () => {
    cy.visit('/register.html').get('input:eq(0)').type('firstName').get('input:eq(1)').type('lastName')
  });
});
```

workshop#1 (/day1/exercise1.spec.js)

- ทำการ inspect element ช่วยหาวิธีการ selector ของหน้า login นี้
/login.html
 - เขียน testscript select input UserName ,Password ทำได้ก็แบบ ต้องเขียนอย่างไร
 - ใส่ value UserName = "Cypress" ,Password = "Automated" ด้วย command .type()

API Commands

- and
- as
- blur
- check
- children
- clear
- click
- closest
- contains
- dbclick
- debug
- document
- end
- eq
- exec
- filter
- find
- first
- fixture
- focus
- focused
- get
- getCookie
- getCookies
- hash
- hover
- invoke
- its
- last
- location
- log
- next
- nextAll
- nextUntil
- not
- parent
- parents
- parentsUntil
- pause
- prev
- prevAll
- prevUntil
- readFile
- reload
- request
- rightclick
- root
- route
- screenshot
- scrollIntoView
- select
- server
- setCookie
- should
- siblings
- spread
- spy
- stub
- submit
- task
- then
- tick
- title
- trigger
- type
- uncheck
- url
- viewport
- visit
- ...

API Commands (normally)

- as
- check
- click
- contains
- get
- log
- not
- request
- screenshot
- select
- should
- spy
- stub
- submit
- then
- title
- type
- uncheck
- url
- wait
- window
- wrap

API Commands (examples)

- as

```
cy.get('.main-nav').find('li').first().as('firstNav') // Alias first 'li' as @firstNav
cy.route('PUT', 'users', 'fx:user').as('putUser') // Alias that route as @putUser
cy.stub(api, 'onUnauth').as('unauth') // Alias that stub as @unauth
cy.spy(win, 'fetch').as('winFetch')
```

```
cy.visit('/register.html').get('[name="gender"]').as('asGenderRadios')
```

- check

```
.check()
.check(value)
.check(options)
.check(value, options)
```

```
cy.visit('/register.html').get('[name="gender"]').as('asGenderRadios')
.get('@asGenderRadios').eq(0).check()
.get('@asGenderRadios').eq(1).check()
```

API Commands (examples)

- click

```
cy.visit('/')  
.get('[data-tab="third"]').click()  
.get('a[data-tab="first"]').click()  
.get('[data-tab="second"]').click()  
.get('[data-tab="five"]').click()
```

- contains

```
.contains(content)  
.contains(content, options)  
.contains(selector, content)  
.contains(selector, content, options)  
  
cy.visit('/change-password.html')  
.get('[data-tab="first"]').contains('New Password')
```

API Commands (examples)

- get

```
cy.get(selector)
cy.get(alias)
cy.get(selector, options)
cy.get(alias, options)

cy.get('input').should('be.disabled')
cy.get('form').within() => {
  cy.get('input').type('Pamela') // Only yield inputs within form
  cy.get('textarea').type('is a developer') // Only yield textareas within form
}
cy.get('ul#todos').as('todos')
```


API Commands (examples)

- log

```
cy.log(message)  
cy.log(message, args...)
```

```
cy.log('command 0001')  
cy.log('command 0002')  
cy.log('command args', ['one', 'two', 'three'])
```

- not

```
.not(selector)  
.not(selector, options)
```

```
cy.visit('/register.html')  
  .get('[type="text"]')  
  .not('#firstName')  
  .should('have.length',3)
```

API Commands (examples)

- select

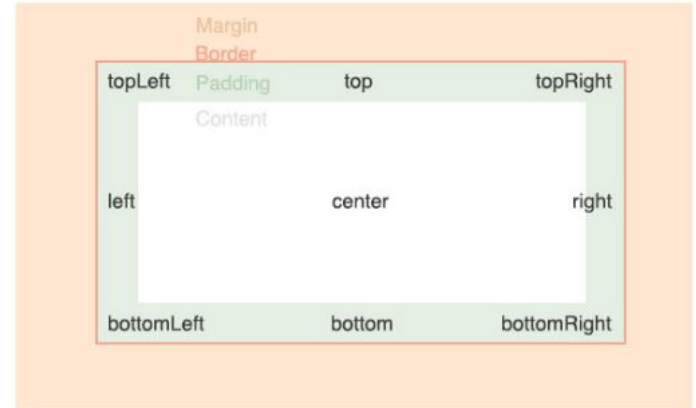
```
.select(value)  
.select(values)  
.select(value, options)  
.select(values, options)
```

```
cy.visit('/register.html')  
.get('select[name="occupation"]').select('Driver')
```

- scrollTo

```
cy.scrollTo(0, 500) //  
Scroll the window 500px down  
cy.get('.sidebar').scrollTo('bottom')  
// Scroll 'sidebar' to its bottom
```

```
cy.get('#infinite-scroll-list').scrollTo(  
0, 500)
```



API Commands (examples)

- title

```
cy.title()
cy.title(options)

cy.title().should('eq', 'My Awesome Application')
```

- type

```
.type(text)
.type(text, options)

cy.get('input').type('Hello, World') // Type 'Hello, World' into the 'input'
```

API Commands (examples)

- visit

```
cy.visit(url)
cy.visit(url, options)
cy.visit(options)
```

```
cy.visit('http://localhost:3000') // Yields the window of the remote page
cy.visit('./pages/hello.html')
```

- wait

```
cy.wait(time)
cy.wait(alias)
cy.wait(aliases)
cy.wait(time, options)
cy.wait(alias, options)
cy.wait(aliases, options)
```

```
cy.wait(2000) // wait for 2 seconds
```

workshop#2 (/day1/exercise2.spec.js)

- แก้ไขไฟล์ exercise2.spec.js
 - เขียน testscript ใช้คำสั่ง `.get()` `.type()` เพิ่มค่าลงใน form ให้ครบถ้วน

Course Cypress Automate Testing @PTT

Index Register Register List Window Login Change Password Table Coronavirus-19

First Name * Last Name * Gender * Male FeMale Occupation *

Phone * Email * Birth Date * Age *

Address

API Commands (examples)

- submit

```
.submit()
.submit(options)

<form id="contact">
  <input type="text" name="message">
  <button type="submit">Send</button>
</form>

cy.get('#contact').submit()
```

- then

```
.then(callbackFn)
.then(options, callbackFn)

cy.get('button').then(($btn) => {
  const cls = $btn.attr('class')

  cy.wrap($btn).click().should('not.have.class', cls)
})
```

API Commands (examples)

- uncheck

```
.uncheck()
.uncheck(value)
.uncheck(values)
.uncheck(options)
.uncheck(value, options)
.uncheck(values, options)

cy.get('[type="checkbox"]').uncheck()
```

- url

```
cy.url()
cy.url(options)

cy.get('#user-edit a').click()
cy.url().should('include', '/users/1/edit') // => true
cy.url().should('eq', 'http://localhost:8000/users/1/edit') // => true
```

API Commands (examples)

- window

```
cy.window()
cy.window(options)

cy.visit('http://localhost:8080/app')
cy.window().then((win) => {
  // win is the remote window
  // of the page at: http://localhost:8080/app
})
```

- wrap

```
cy.wrap(subject)
cy.wrap(subject, options)

const getName = () => {
  return 'Jane Lane'
}

cy.wrap({ name: getName }).invoke('name').should('eq', 'Jane Lane') // true
```


API Commands (examples)

- screenshot

```
cy.screenshot()  
cy.screenshot(fileName)  
cy.screenshot(options)  
cy.screenshot(fileName, options)
```

```
cy  
.visit('/register.html').screenshot('screenshot-register')  
.visit('/window.html').screenshot('screenshot-window')  
.visit('/login.html').screenshot('screenshot-login')  
.visit('/change-password.html').screenshot('screenshot-change-password')  
.visit('/table.html').screenshot('screenshot-table')
```

```
check screenshot // cypress/screenshots/workshop1/commands.spec.js
```

workshop#3 (/day1/exercise3.spec.js)

- แก้ไขไฟล์ exercise3.spec.js
 - เขียน testscript capture screen หน้าจอ ตาม tabs ทั้งหมด ประกอบไปด้วย index ,register ,register ,Rregister list ,window ,...
 - หน้า /window.html ต้องการให้ capture screen ส่วนของปุ่ม "Window Alert และ Window Confirm" เท่านั้น ตามภาพ



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')
```

workshop#4 (/day1/exercise4.spec.js)

- แก้ไขไฟล์ exercise4.spec.js
 - เขียน testscript assertions 10 เงื่อนไข เช่น contains('ข้อความ') ,should('have.value','ข้อความ') ,...
 - แก้โจทย์อื่น ๆ

Catalog of Events

- uncaught:exception
- window:confirm
- window.alert
- window:before:load
- window:load
- window:before:unload
- window:unload
- url:changed
- fail
- viewport:changed
- scrolled
- command:enqueued
- command:start
- command:end
- command:retry
- log:added
- log:changed
- test:before:run
- test:after:run

Catalog of Events (examples)

```
cy.on('uncaught:exception', (err, runnable) => {  
  expect(err.message).to.include('something about the error')  
  return true  
})
```

```
cy.on('log:added', (msg) => {  
  //console.log('msg ::==' + JSON.stringify(msg))  
})
```

```
cy.on('window:confirm', val => {  
  console.log('val ::==' + val)  
})
```

```
cy.on('window:alert', val => {  
  console.log('val ::==' + val)  
})
```

```
cy.on('window:before:load', object=>{  
  console.log('object ::==' + JSON.stringify(object))  
})
```

workshop#5 (/day1/exercise5.spec.js)

- แก้ไขไฟล์ exercise5.spec.js
 - เขียน testscript เงื่อนไขตรวจจัดการ click window.alert 5 ครั้ง