

---

# Table of Contents

Introduction	1.1
前言	1.2
從測試出發	1.3
誰需要前端測試	1.4
基礎環境安裝	1.5
建立 BDD 前端測試 - mocha	1.6
建立前後端共用測試 - mocha & npm test	1.7
前端整合測試環境安裝	1.8
淺談工具 - Codeception.js	1.9
第一個前端 e2e 整合測試	1.10

# 前端測試實做手冊

獻給所有正在希望持續導入前端測試，或者正在前端測試，整合測試大門徘徊的開發者們。提供一個簡單容易的開端，讓開發者瞭解前端開發的訣竅及入門方法。

從觀念到實際範例程式碼，讓開發者可以直接導入到目前專案，以輕鬆的方式進行前端測試，讓原本測試流程中缺少的那塊拼圖完整吧。

本書籍由 Caesar Chi 本人規劃編寫，特別感謝支持的家人及朋友們，以及上課的學員及設群的朋友給予的寶貴建議，讓本書資源能夠越來越豐富。

不論你之前的經驗是什麼，這邊都會希望接下來的流程，都從‘測試開始’，以‘測試先行’的角度開始進行開發模式，跟著步驟跟著流程，一步一步往前進行，如果遇到任何問題，歡迎與我聯絡。

[clonncd@gmail.com](clonncd@gmail.com)

# 前端測試實做手冊

獻給所有正在希望持續導入前端測試，或者正在前端測試，整合測試大門徘徊的開發者們。提供一個簡單容易的開端，讓開發者瞭解前端開發的訣竅及入門方法。

從觀念到實際範例程式碼，讓開發者可以直接導入到目前專案，以輕鬆的方式進行前端測試，讓原本測試流程中缺少的那塊拼圖完整吧。

本書籍由 Caesar Chi 本人規劃編寫，特別感謝支持的家人及朋友們，以及上課的學員及設群的朋友給予的寶貴建議，讓本書資源能夠越來越豐富。

不論你之前的經驗是什麼，這邊都會希望接下來的流程，都從‘測試開始’，以‘測試先行’的角度開始進行開發模式，跟著步驟跟著流程，一步一步往前進行，如果遇到任何問題，歡迎與我聯絡。

[clonncd@gmail.com](clonncd@gmail.com)

## 從測試出發

會開始讀取本書讀者，大部分都是已經具備基本測試經驗的開發者，或者是比較瞭解前端的開發者。

本書主要以實做為主，順勢導入觀念為輔，希望各位讀者能夠跟著一起進行動手做，從實做中開始瞭解測試的奧妙，天下武功，唯只有經過不斷的實戰與思考才能融會貫通，測試流程也是一樣。

透過本書的實做方式，以及範例程式，提供大家一個明確的路線，瞭解前端測試開發奧妙，讓大家知道如何將前端測試導入，提供更穩定的環境與服務給使用者，這才是身為開發者應該做到的事情。

###測試，就從今天開始。

就讓我們開始進行一系列的實際開發流程，希望帶給各位不一樣的開發視野觀點，同時也希望各位能夠真正動手開始進行實做。

# 誰需要前端測試

在開始進行之前，首先定義一下你有以下的問題

- 每次程式進行更新感覺到不安
- Bug 永遠跟不完，修 A 壞 B 的狀況時常發生
- 追求完美程式架構者
- 苦尋不著前端測試方法者

如果你有以上症狀，或者有相同感受者，請繼續研讀此書，將會令你獲益良多。

希望透過前端測試，讓你測試環境更為完美，到達整體流程的境界

## env install

環境必須要安裝

- git - [git install](#)
- node.js - <https://nodejs.org/en/> (請以 LTS 版本為主)
- bower - <https://bower.io/#install-bower>
- Web server - chrome extension

## mocha run test

```
npm i -g bower
```

## About mocha

- [mocha.js](#)

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. Hosted on GitHub.

## start mocha env

install by script below, and `mocha-basic` is your project name, you can change whatever you want. install mocha and chai

```
mkdir mocha-basic && mocha-basic
bower i mocha
bower i chai
```

create a `index.html` file, content is below,

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <div id="mocha"></div>
    <script src=".bower_components/chai/chai.js"></script>
    <script src=".bower_components/mocha/mocha.js"></script>
    <script src=".js/functions.js"></script>
    <script>mocha.setup('bdd')</script>
    <script src="test.js"></script>
    <script>
        mocha.run();
    </script>
</body>
</html>
```

## test first

create a test folder, and create a new `add.spec.js` file, as a test file for run test add function.

```
mkdir test
touch test/add.spec.js
```

## edit test content

## 建立 BDD 前端測試 - mocha

```
var expect = chai.expect;
var should = chai.should();
describe('Test add function', function() {
    it('test simple add basic', function () {
        var result = add(2, 1);
        result.should.be.equal(3);
    })
});
```

Prepare about `add.js`

```
mkdir js
touch js/add.js
# then edit js/add.js
```

edit `js/add.js`

```
function add(num1, num2) {
    return num1 + num2;
}
```

then run `index.html`, you will see test reslut is correct.

The screenshot shows a browser window with the URL `localhost:1337/index.html`. The page displays the results of a mocha test run. At the top, there is a summary: "passes: 1 failures: 0 duration: 0.07s 100%". Below this, the test title "Test add function" is shown, followed by a green checkmark and the test name "test simple add basic".

## 讓 mocha 同時前後端執行

剛才已經有設定了關於前端執行項目，但是 mocha 實際上也可以在後端進行測試，我們就來改善一下設定，讓我們的程式也可以照常進行吧

```
npm init
```

輸入後過程中會有一些對話，內容可以全部留白，內容大致上會是如下描述，

```
name: (mocha-basic)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /Users/chicaesar/workspace/mokayo/frontend-test-exercise/mocha-basic/package.json:

{
  "name": "mocha-basic",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "directories": {
    "test": "test"
  },
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Is this ok? (yes)

接著環境中就會多出一個 `package.json` 針對於 node.js 後端專案的設定檔。

接下來就開始繼續安裝後端測試套件，並且設定到環境中吧

```
npm i bower chai --save-dev
```



## 開始進行 codeceptjs

在環境完成安裝設定之後，我們就可以開始編寫第一個 e2e 測試腳本。

透過 codeceptjs 指令進行新增腳本，

```
codeceptjs gt
```

透過 gt 參數，代表 generator test 的意思，開始進行測試的編輯，並且同時設定檔案名稱為 `first_test.js`，因為我們前面已經設定 test 執行路徑將會讀取 `*_test.js`，所以命名規則必須要符合才有辦法進如執行環境中。

```
Test root is assumed to be $PATH_ENV
Creating a new test...
-----
? Filename of a test first_test.js
? Feature which is being tested First test.js
Test for first_test.js was created in $PATH_ENV/first_test.js
```

如上述設定完成後，就會看到新增一個基本的測試環境可以提供使用，

## 第一個測試腳本

完成上述描述後，我們就可以開始進行第一個測試腳本，

```
Feature('First test.js');

Scenario('test something', (I) => {
    I.amOnPage('https://www.google.com.tw/');
    I.see('Google');
});
```

`I.amOnPage` 表示瀏覽器即將前往的網址。

I.see 為最後確認 assertion，確定驗證資訊是否與預測相同，目前會直接找到 page title 進行驗證。

我們可以透過底下指令，就會看到立即開啟一個 google page，並且進行驗證的動作。

```
codeceptjs run --steps
```

# 介紹 CodeceptJS

MODERN ERA ACCEPTANCE TESTING FOR NODEJS

CodeceptJS 是一個基於 WebDriver 全新的 E2E 測試框架。以使用者角度進行使用者行為測試，進行簡單使用者操作步驟來編寫測試

<http://codecept.io/>

## 安裝方式

```
npm install -g codeceptjs
```

透過 npm 指令就可以快速將 codeceptjs 安裝到環境中，除了安裝 codeceptjs 之外，還要安裝底層 e2e 執行環境，

例如需要優先安裝 `webdriverio`，

```
npm install webdriverio
```

## 執行方法

安裝好 coeceptjs 之後，可以開始進行設定環境，透過 `codeceptjs` 指令

```
codeceptjs init
```

開始進行設定專案，目前我們僅需要進行處理 `webdriverio` 就選擇 `webdriverio` 項目即可，以及確定資料夾路徑。

設定流程大約會如下描述，

```
Test root is assumed to be $PATH_ENV

Welcome to CodeceptJS initialization tool
It will prepare and configure a test environment for you

Installing to $PATH_ENV/
? Where are your tests located? ./*_test.js
? What helpers do you want to use? WebDriverIO
? Where should logs, screenshots, and reports to be stored? .
/output
? Would you like to extend I object with custom steps? No
Configure helpers...
? [WebDriverIO] Base url of site to be tested http://localhost
? [WebDriverIO] Browser in which testing will be performed chrome
Config created at $PATH_ENV/codecept.json
Directory for temporary output files created at `_output`
Almost done! Create your first test by executing `codeceptjs
gt` (generate test) command
```

設定好之後，就會看到資料夾中多出了一個 `codecept.json` 檔案，這也就是 `codecept.json` 的執行環境。

大概描述內容如下，

```
{  
    "tests": "./*_test.js",  
    "timeout": 10000,  
    "output": "./output",  
    "helpers": {  
        "WebDriverIO": {  
            "url": "http://localhost",  
            "browser": "chrome"  
        }  
    },  
    "include": {},  
    "bootstrap": false,  
    "mocha": {},  
    "name": "06_e2e_codeceptjs"  
}
```

如此一來就完成 codeceptjs 環境設定，記得開始測試之前一定要先安裝 selenium 以及模擬環境。

# github 程式體驗

透過底下描述，我們可以快速瞭解 github 頁面流程，以及透過快速進行頁面測試，檔名設定為 `github_test.js`，

```
Scenario('search', (I) => {
  I.amOnPage('https://github.com/search');
  I.fillField('Search GitHub', 'mokayo');
  I.pressKey('Enter');
  I.see('miiixr/mokayo', 'a');
});

Scenario('signin', (I) => {
  I.click('Sign in');
  I.see('Sign in to GitHub');
  I.fillField('Username or email address', 'user@user.com');
  I.fillField('Password', '123456');
  I.click('Sign in');
  I.see('Incorrect username or password.', '.flash-error');
});

Scenario('register', (I) => {
  within('.js-signup-form', function () {
    I.fillField('user[login]', 'User');
    I.fillField('user[email]', 'user@user.com');
    I.fillField('user[password]', 'user@user.com');
    I.fillField('q', 'aaa');
    I.click('button');
  });
  I.see('There were problems creating your account.');
  I.click('Explore');
  I.seeInCurrentUrl('/explore');
});
```

檔案名稱設定完後，一樣進行執行，將會看到瀏覽器成果。

```
codeceptjs run --steps
```