

How to use async/await

Jing, mqjing@gmail.com

GitHub: [Download](#)

Google doc: [This document](#).

非同步處理是學習使用 node.js 的核心, 現在有新指令 async/await 即將發佈. 將大幅改善程式碼處理非同步的問題. 網路上有一堆範例程式, 但是怎麼執行呢? 要怎麼嘗鮮使用呢? 現在來示範一下, 我完全參考[這篇](#)的做法.

因為 V8 已經把 **async/await** 等新的處理非同步功能 build 進 night build 了. 只要拉 v8 nightbuild code, 然後就可以享受 async/await 的最新功能. 你不需要再像以前一樣, 弄一堆煩死人的 babel 相關的套件.

下面提供兩種方式, 讓你玩新指令.

1. **Docker Version**: 只要三個指令, 完全攻略自動安裝完成. 讓你專注在 js code 上.
2. **手動 Version**: 一步一步安裝必要套件與設定, 讓你知道 Docker 版本的 script 是怎麼做的.

Enjoy & Good Luck.

Jing.

Docker Version

```
# pull the example
git pull https://github.com/jing-tw/lab-nodejs.git
cd ./example/async_new_function

# Step 1: 設定使用 nightbuild 版本 node js
../01_docker_init.sh

# Step 2: 寫 code
vi example.js

# Step 3: Run
../docker_run.sh
```

E.g.

```
jing@jing-Lenovo-G510s:~/test/lab-nodejs/example/async_ex3$ ./docker_run.sh
async_ex3._Debug
timer started
timer finished
jing@jing-Lenovo-G510s:~/test/lab-nodejs/example/async_ex3$
```

Manual Version

Step 1: 安裝 nodejs nightbuild 版本

```
#!/bin/bash

function install_nightbuild(){
  curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.32.1/install.sh | bash

  # load nvm
```

```

export NVM_DIR="/root/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"

NVM_NODEJS_ORG_MIRROR=https://nodejs.org/download/nightly
nvm install 7
nvm use 7
}

install_nightbuild

```

E.g.

```

% Total      % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100 10250  100 10250    0     0  17320      0 --:--:-- --:--:-- --:--:-- 17343
=> Downloading nvm from git to '/root/.nvm'
=> Cloning into '/root/.nvm'...
remote: Counting objects: 6010, done.
remote: Total 6010 (delta 0), reused 0 (delta 0), pack-reused 6010
Receiving objects: 100% (6010/6010), 1.68 MiB | 774.00 KiB/s, done.
Resolving deltas: 100% (3744/3744), done.
Checking connectivity... done.
* (detached from v0.32.1)
  master
=> Compressing and cleaning up git repository
Counting objects: 6010, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5975/5975), done.
Writing objects: 100% (6010/6010), done.
Total 6010 (delta 3970), reused 1828 (delta 0)

=> Appending source string to /root/.bashrc
npm info it worked if it ends with ok
npm info using npm@3.10.9
npm info using node@v7.2.0
npm info ok
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="/root/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh" # This loads nvm
##### 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v7.2.1-nightly201611248cabe28efb (npm v3.10.9)
Creating default alias: default -> 7 (-> v7.2.1-nightly201611248cabe28efb)
Now using node v7.2.1-nightly201611248cabe28efb (npm v3.10.9)
jing@jing-Lenovo-G510s:~/test/lab-nodejs/example/async_ex3$

```

Step 2: coding

```

'use strict';

// [高耗時工作]
// 定義一個名為 timeout 的 function
const timeout = function (delay) {

  // 傳回一個 Promise 物件 [another example]
  // Promise 用法:
  // (a) 在 promise function 裡面做 高耗時非同步工作
  // (b) 完成後, 呼叫 resolve
  return new Promise((resolve, reject) => {
    // 等待 delay million second 後, 才會 resolve
    setTimeout(() => {
      resolve() // timeout 後, 完成
    }, delay)
  });
}

// [使用範例]
async function timer () {
  console.log('timer started')

  // [關鍵片段]
  // 下面的工作, 會按照順序完成
  await Promise.resolve(timeout(100));
  await Promise.resolve(timeout(10));

  console.log('timer finished')
}

timer()

```

Step 3: Run

node **--harmony-async-await** example.js

Verification

```
jing@jing-Lenovo-G510s:~/test/lab-nodejs/example/async_ex3$ . ./docker_run.sh
async_ex3._Debug
timer started
timer finished
jing@jing-Lenovo-G510s:~/test/lab-nodejs/example/async_ex3$
```

Multiple Async Tasks

```
'use strict';

// app.js
const timeout = function (strMessage, delay) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      console.log(strMessage);
      resolve()
    }, delay)
  })
}

// no Promise testing
const task1 = function (strTitle) {
  setTimeout(function() {console.log(strTitle + "task 1");}, 3000);
}

// no Promise testing
const task2 = function (strTitle ) {
  setTimeout(function() {console.log(strTitle + "task 2");}, 100);
}

async function timer (strTitle) {
  console.log(strTitle + " timer started")
  await Promise.resolve(timeout(strTitle + "Async Task 1 ", 100));
  task1(strTitle);
  await Promise.resolve(timeout(strTitle + "Async Task 2", 10));
  task2(strTitle);
  console.log('timer finished')
}

function timer_main () {
```

```
        timer("timer 1"); // timer 1 slot: Async Task 1 -> Async Task 2 and task 1 & task 2
    async
        timer("timer 2"); // timer 2 slot: Asyn Task 1 -> Async and Task 1 & task 2 async
    }

    timer_main()
```

Reference

- <https://blog.risingstack.com/async-await-node-js-7-nightly/>