MEAN Stack實作班01

重要連結

pc密碼：student

無線網路 SSID: csie_guest

帳號: train_html

密碼: 56ACGRX2

講義：https://goo.gl/79HtBW

上課相關檔案：

https://www.dropbox.com/sh/mnxu5fosuhjpug0/AACoP4bMMMPOeEFYKEKPrqXpa?dl=0

事項宣布

1. 出席:40%
2. 作業:60%
   a. 設計Express App並建立routes(20%)
      (5~10分鐘)
   b. 設計Movie Search App(40%)
      (5~10分鐘)

   ＊chrome錄影套件:[sceencastify]

3. email:kuolun@gmail.com

作業說明 https://goo.gl/dl5k8o

1.繳交方式:

用chrome套件錄影demo, 上傳至youtube把連結寄給我

2.Demo內容:

解說程式碼邏輯及功能示範操作

3.DeadLine:

12/16(五) 9:00 am前

4.及格標準:

70分

上課環境

編輯器

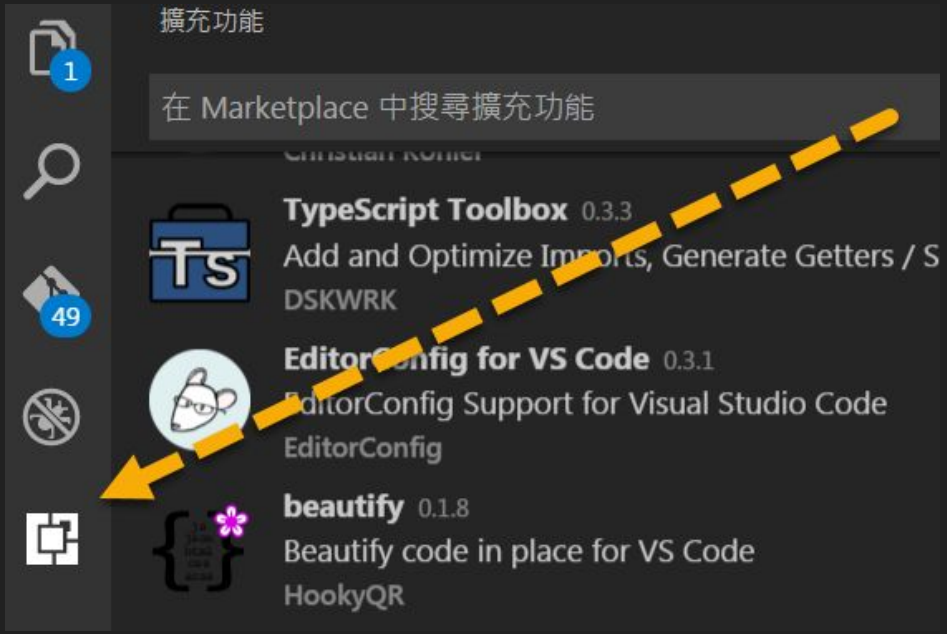
VS Code

https://code.visualstudio.com/

教學

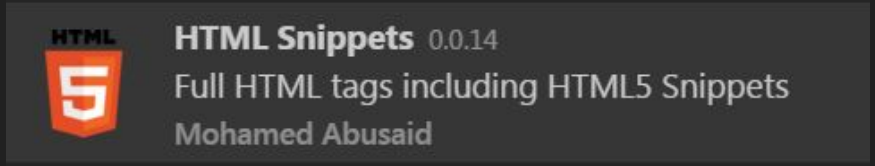
https://channel9.msdn.com/Series/Mastering-Visual-Studio-Code

安裝VS Code套件


擴充功能
在 Marketplace 中搜尋擴充功能
TypeScript Toolbox 0.3.3
Add and Optimize Imports, Generate Getters / S
DSKWRK
EditorConfig for VS Code 0.3.1
EditorConfig Support for Visual Studio Code
EditorConfig
beautify 0.1.8
Beautify code in place for VS Code
HookyQR


JavaScript (ES6) code snippets 1.0.0
Code snippets for JavaScript in ES6 syntax
charalampos karypidis


beautify 0.1.8
Beautify code in place for VS Code
HookyQR


HTML Snippets 0.0.14
Full HTML tags including HTML5 Snippets
Mohamed Abusaid

請先註冊帳號

cloud9

mlab

https://mlab.com/home

Postman

https://www.getpostman.com/

Facebook Developer

https://developers.facebook.com/

Final App DEMO

今日預計內容

- 安裝環境及設定
- Intro to MEAN Stack
- Node.js
- npm
- Express
- 專案架構
- 連接mongoDB
- mongoDB ODM

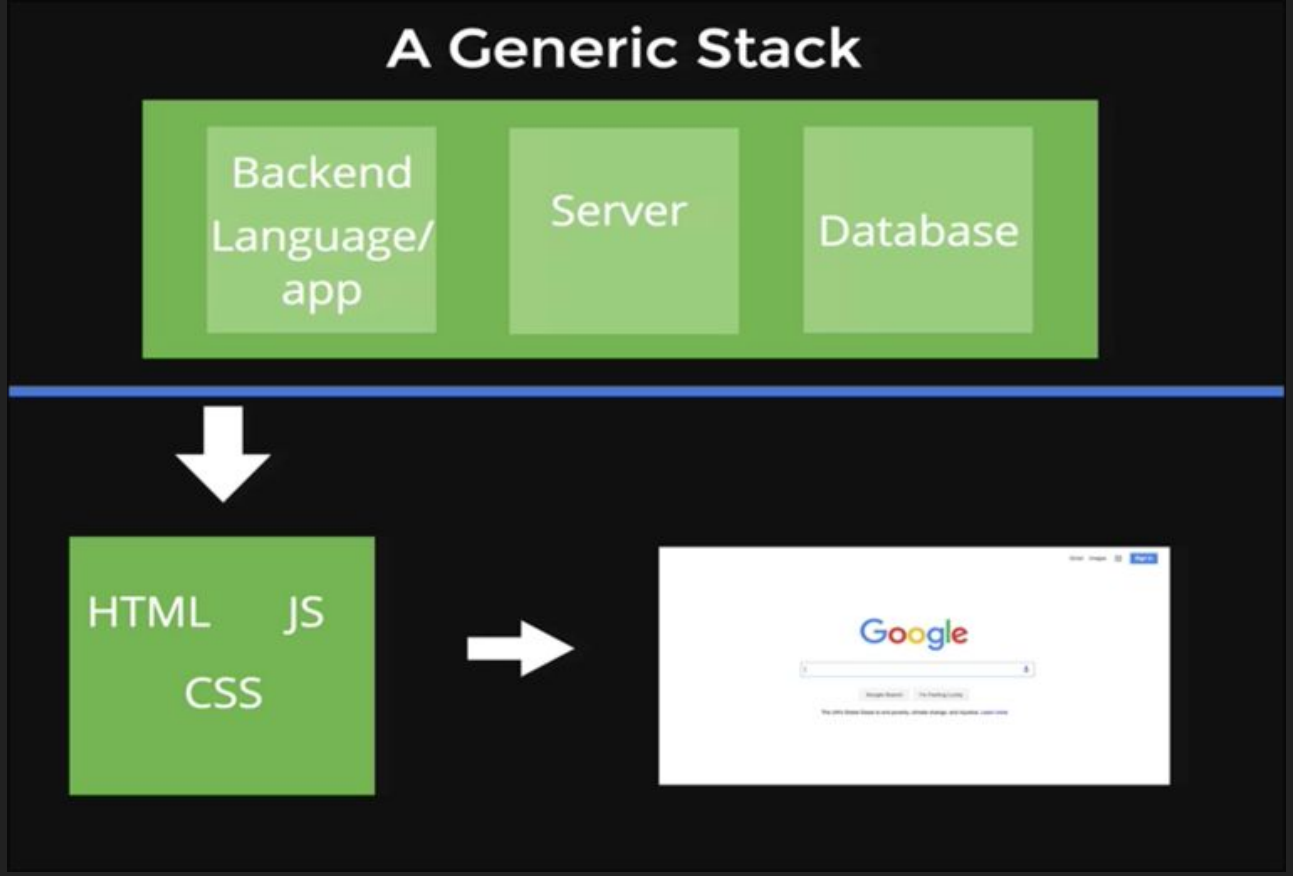
介紹 Full Stack Development

Full Stack Development

開發Web APP

從Database跟Server開始(後端)

User Interface(前端)



http://stackshare.io/airbnb/airbnb

Our Stack

Node JS

Express

MongoDB

HTML JS

CSS

Potential Backend Features

- Check if the user is logged in
- Figure out what HTML, CSS, and JS to send to the User
- Sign Up a User
- Add new post to DB
- Create new comment
- remove post from DB
- Sort/Rank posts
- Create subreddit
- Add to newsletter

HTML JS

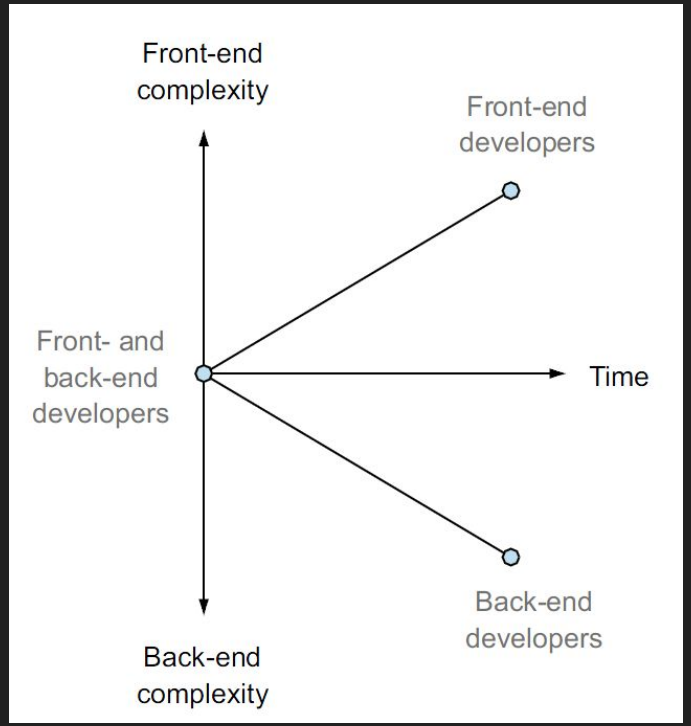
CSS

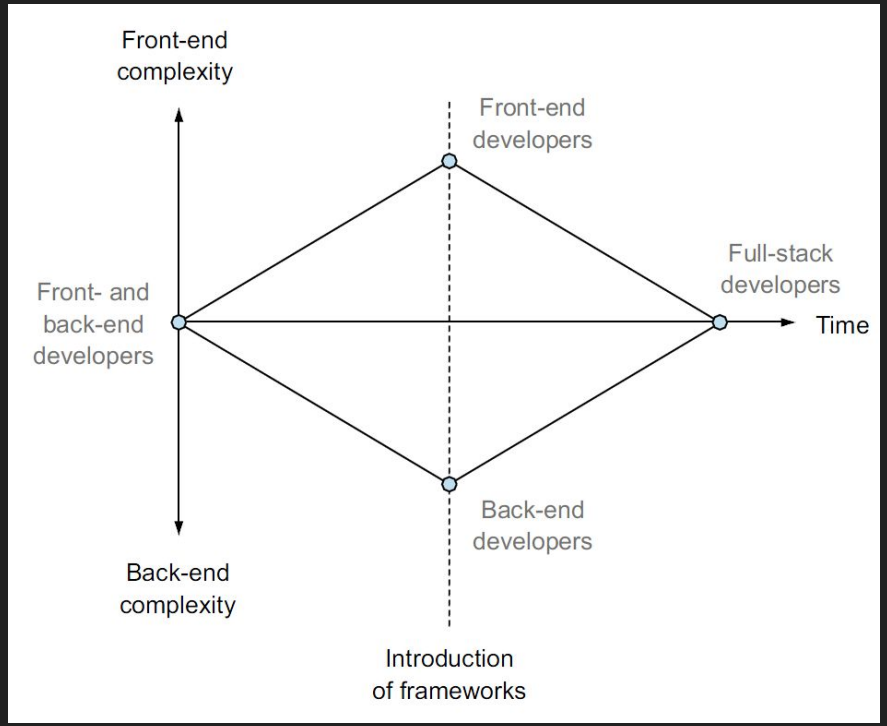
Frontend		Backend
1. Ask for reddit homepage	→	
		2.Get top posts from DB send back home page content
	←	
3. Browser renders home page		
4.User enters "dogs" in search box and submits form	→	
		5.Finds all posts in DB about "dogs"
	←	
7.Browser renders search page		6.Sends back HTML for the search results page

MEAN Stack

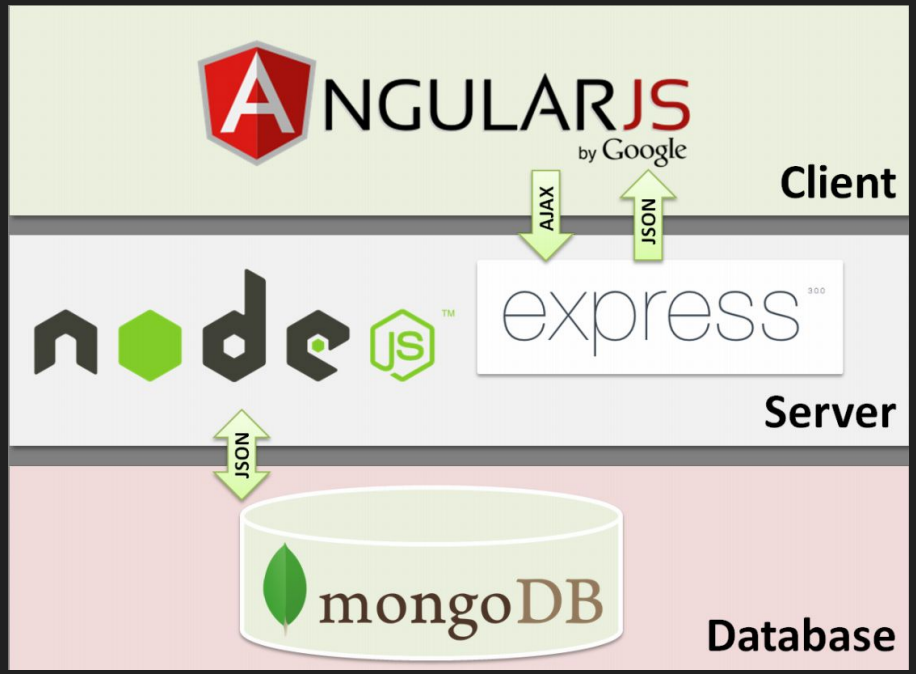
- MongoDB - database
- Express - web framework
- AngularJS - front-end framework
- Node.js - platform

為什麼要學full stack ？





Why MEAN Stack ?



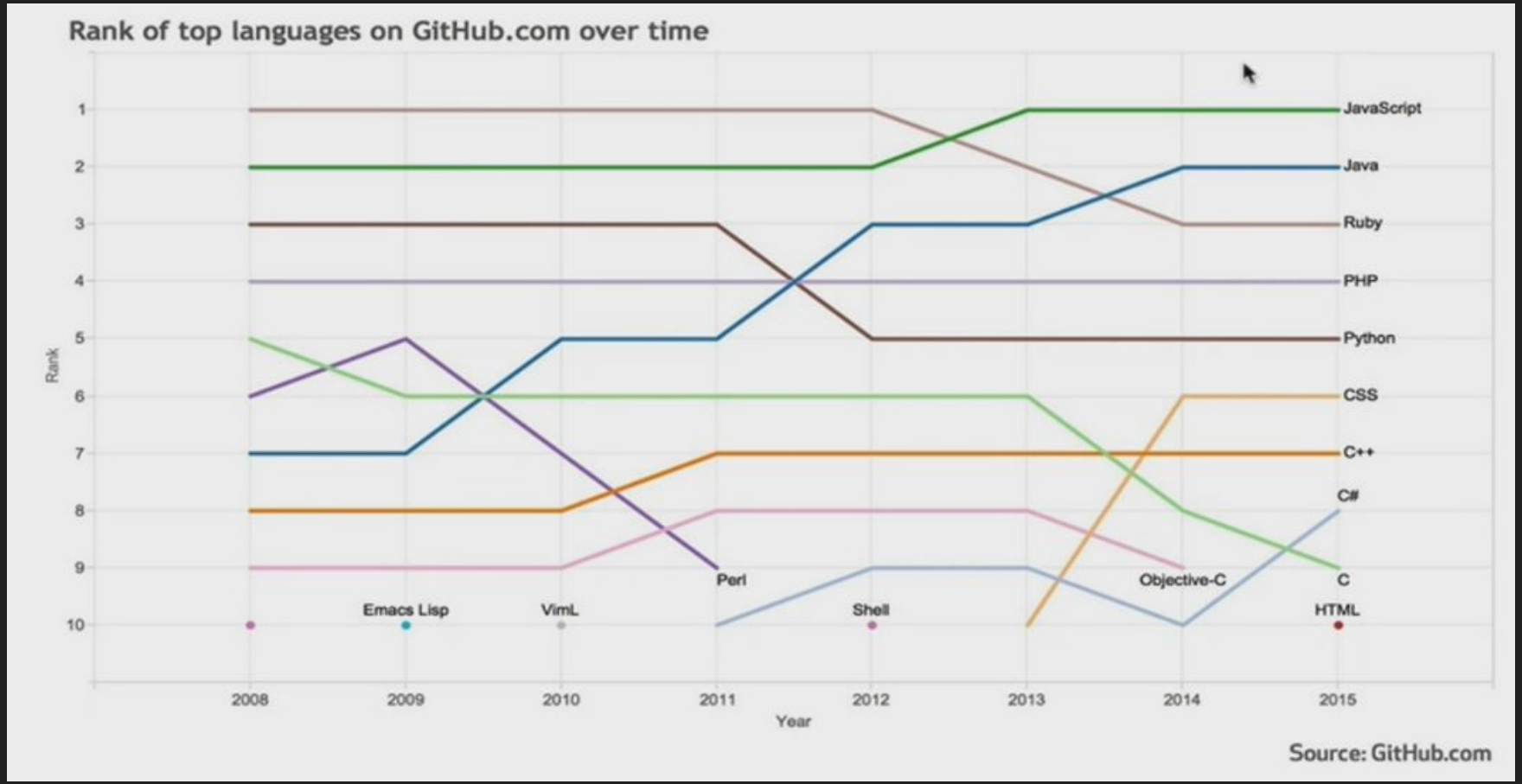
Node.js

Server-Side Javascript Platform

Node是什麼？

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

GitHub - 過去七年趨勢



大家喜歡Node.js的原因

stackshare.io/nodejs

WHY PEOPLE LIKE NODE.JS

- 689 Npm
- 628 Great libraries
- 402 Open source
- 339 Asynchronous
- 300 Great for realtime apps
- 654 Javascript
- 531 High-performance
- 357 Great for apis
- 335 Great community
- 229 Great for command line utilities

Ryan Dahl: Node.js, Event

jsconf.eu/2009/speaker/speakers_selected.html

Ryan Dahl: Node.js, Evented I/O for V8 Javascript

By Malte Ubl | 16.09.09 22:36 | Comme

Ryan Dahl will present on his uber

It is well known that event loops rat
Javascript is a language unencumb
synchronous evented I/O, making i
ties together the V8 Javascript com
system calls, and a carefully desig
fast server-side software. This talk

Ryan Dahl: Original Node.js presentation

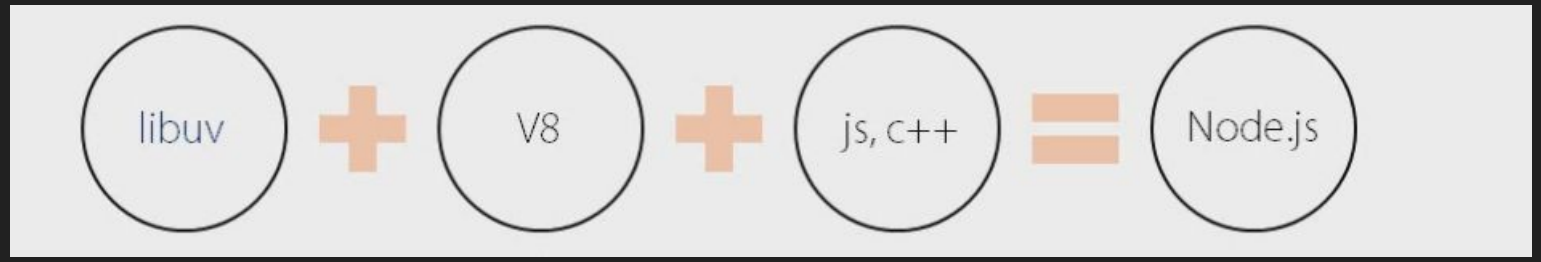
stri8ted | Subscribe | 24 videos

node.js in brief:

- Server-side Javascript
- Built on Google's V8
- Evented, non-blocking I/O. Similar to EventMachine or Twisted.
- CommonJS module system.
- 8000 lines of C/C++, 2000 lines of Javascript, 14 contributors.

00:26 / 48:32

Node.js platform



Node.js--foundation of the stack

Platform

- 可以讓你建立自己的web server跟web application
  -有內建 HTTP server library
  -不需另外執行Apache或IIS

- 本身非web server 或 language

- 執行速度很快

安裝Node.js

https://nodejs.org/en/

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Important security upgrades for recent V8 vulnerability

Download for OS X (x64)

v4.4.7 LTS
Recommended For Most Users

Other Downloads | Changelog | API Docs

v6.3.1 Current
Latest Features

Other Downloads | Changelog | API Docs

DEMO

$ node -v

$ npm -v

```
kuolundeMacBook-Pro:MEAN_19 kuolun$ node -v
v6.2.0
kuolundeMacBook-Pro:MEAN_19 kuolun$ npm -v
3.8.9
kuolundeMacBook-Pro:MEAN_19 kuolun$
```

安裝時間

Node console

命令列輸入 node

Ctrl + C 跳出

```
系統管理員: 命令提示字元 - node
Microsoft Windows [版本 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\USER>node
> 14+9
23
> "hello " + "world"
'hello world'
>
```

```
> alert("hello");
ReferenceError: alert is not defined
    at repl:1:1
    at sigintHandlersWrap (vm.js:32:31)
    at sigintHandlersWrap (vm.js:96:12)
    at ContextifyScript.Script.runInContext (vm.js:31:12)
    at REPLServer.defaultEval (repl.js:308:29)
    at bound (domain.js:280:14)
    at REPLServer.runBound [as eval] (domain.js:293:12)
    at REPLServer.<anonymous> (repl.js:489:10)
    at emitOne (events.js:101:20)
    at REPLServer.emit (events.js:188:7)
>
```

用node執行javascript檔案：node <filename>

檔案總管

- 開啟編輯器
  - hello.js
- 0813
  - hello.js

hello.js

```
for (var i = 0; i < 10; i++) {
    console.log("hello from js");
}
```

系統管理員: \\

```
Microsoft Windows [版本 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

d:\Dropbox\Dropbox\台大實戰班MEANstack\274期\0813>node hello.js
hello from js
hello from js
hello from js
hello from js
hello from js
hello from js
hello from js
hello from js
hello from js
hello from js
```

Exercise 10分鐘

1. 建立一個echo.js檔

2. 設計一個echo function，接收2個參數
   依據第2個參數去決定要印出幾次
   把下面兩行貼到程式最後面

```
echo( “hello” , 5 );
echo(“ this is awesome”, 10);
```

3. 用node執行此程式

npm (node package manager)

November's Paddleball Marathon

npm Enterprise features pricing documentation support

npm

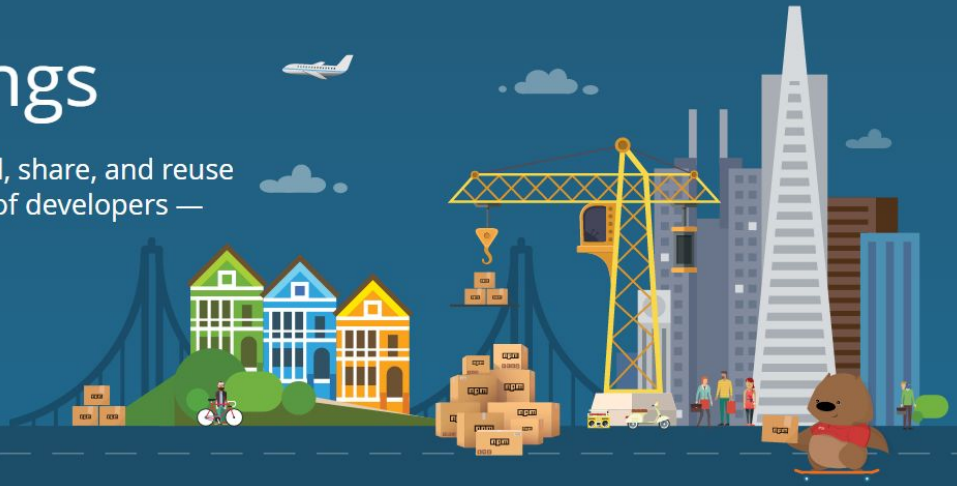
find packages

sign up or log in

Build amazing things

npm is the package manager for JavaScript. Find, share, and reuse packages of code from hundreds of thousands of developers — and assemble them in powerful new ways.

Get started



npm

下載Node.js module或package來擴充app的功能

如:

Mongoose:提供更便利操作mongoDB的API

Morgan:輸出HTTP request至console

body-parser:解析body

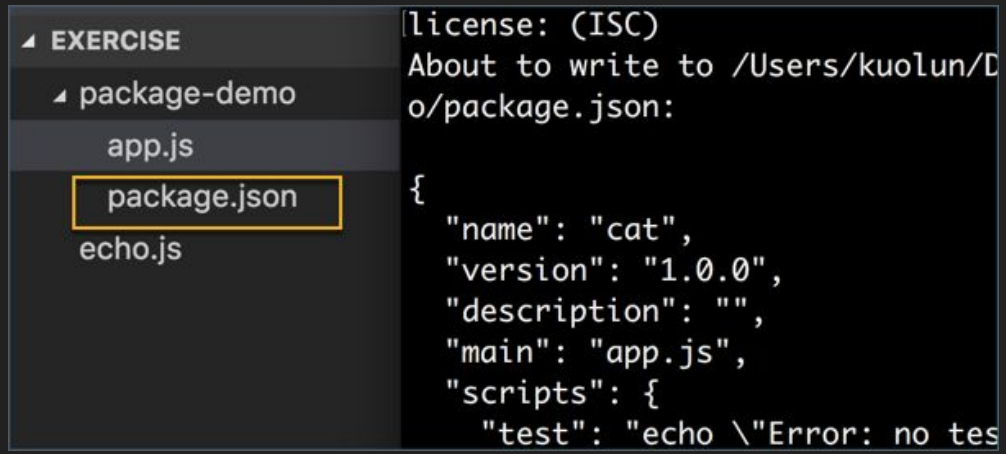
Express:幫忙建立website

練習使用第一個package

npm init

```
kuolundeMacBook-Pro:exercise kuolun$ npm init
This utility will walk you through creating a packa
It only covers the most common items, and tries to

See `npm help json` for definitive documentation on
and exactly what they do.

Use `npm install <pkg> --save` afterwards to instal
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (exercise)
```

```
{
  "name": "cat",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```



安裝cat-me package

https://www.npmjs.com/package/cat-me

EXPLORER

OPEN EDITORS

echo.js

app.js package-demo

EXERCISE

package-demo

node_modules

app.js

package.json

echo.js

```
author:
kuolundeMacBook-Pro:package-demo kuolun$ npm install cat-me
cat@1.0.0 /Users/kuolun/Desktop/台大MEAN/274期/0813/exercise/p
└┬ cat-me@1.0.3
  └┬ yargs@3.32.0
    ├── camelcase@2.1.1
    ├┬ cliui@3.2.0
    │ └┬ strip-ansi@3.0.1
    │ │ └── ansi-regex@2.0.0
    │ └── wrap-ansi@2.0.0
    ├── decamelize@1.2.0
    ├┬ os-locale@1.4.0
    │ └┬ lcid@1.0.0
    │   └── invert-kv@1.0.0
    ├┬ string-width@1.0.1
    │ ├┬ code-point-at@1.0.0
    │ │ └── number-is-nan@1.0.0
    │ └── is-fullwidth-code-point@1.0.0
    ├── window-size@0.1.4
    └── y18n@3.2.1

npm WARN cat@1.0.0 No description
npm WARN cat@1.0.0 No repository field.
```

EXERCISE

- package-demo
  - node_modules
    - .bin
    - ansi-regex
    - camelcase
    - cat-me
      - .npmignore
      - .travis.yml
      - cats.json
      - cmd.js
      - index.js
      - package.json
      - README.md
      - test.js

```
49  "author": {…
53  },
54  "bin": {
55    "catMe": "cmd.js"
56  },
57  "bugs": {
58    "url": "https://github.
59  },
60  "dependencies": {
61    "yargs": "^3.18.0"
62  },
63  "description": "ASCII cat
64  "devDependencies": {
65    "standard": "4.5.4",
66    "tap": "^1.3.2"
67  },
68  "directories": {},
```

使用cat-me package

EXPLORER

OPEN EDITORS

app.js package-demo

EXERCISE

package-demo

node_modules

app.js

package.json

echo.js

app.js

```
1  var cat = require('cat-me');
2
3  console.log(cat());
```

package-demo — -bash — 80×24

```
author:
kuolundeMacBook-Pro:package-demo kuolun$ node app.js
    /\__/\
   /`    '\
 === 0  0 ===
   \  --  /
  /        \
 /          \
|            |
 \  ||  ||  /
  \_oo__oo_/#######o
kuolundeMacBook-Pro:package-demo kuolun$
```

練習時間：5分鐘

建立第一個Server

server.js

```
var http = require('http');


// STEP #1 Basic server:
http.createServer(function(request,response){
    response.writeHead(200);
    response.write('<h1>My Server worked</h1>');
    response.end();
}).listen(3000);
```

Express

http://expressjs.com/zh-tw/

Library v.s Framework

- 都是別人寫的code
- Library-由我們控制何時要使用
- Framework-要按照制訂的規則去使用，可以簡化一些繁瑣的基本工作

Library跟Framework的差異

262

The most important difference, and in fact the *defining* difference between a library and a framework is Inversion of Control.

What does this mean? Well, it means that when you call a library, *you* are in control. But with a framework, the control is inverted: the *framework* calls you. (This is called the Hollywood Principle: Don't call Us, We'll call You.) This is pretty much the definition of a framework. If it doesn't have Inversion of Control, it's not a framework. (I'm looking at you, .NET!)

Basically, all the control flow is already in the framework, and there's just a bunch of predefined white spots that you can fill out with your code.

A library on the other hand is a collection of functionality that *you* can call.

I don't know if the term toolkit is really well defined. Just the word "kit" seems to suggest some kind of modularity, i.e. a set of independent libraries that you can pick and choose from. What, then, makes a toolkit different from just a bunch of independent libraries? Integration: if you just have a bunch of independent libraries, there is no guarantee that they will work well together, whereas the libraries in a toolkit have been designed to work well together – you just don't have to use *all* of them.

But that's really just my interpretation of the term. Unlike library and framework, which *are* well-defined, I don't think that there *is* a widely accepted definition of *toolkit*.

為什麼要用Express？

- popular
- 被廣泛使用
- Github有很多contributors
- Large community
- well-documented and support

Stats

318,504 downloads in the last day

1,729,135 downloads in the last week

7,318,141 downloads in the last month

88 open issues on GitHub

37 open pull requests on GitHub

Express-簡化server setup

- Web application framework
- 處理建立website時的一些重複性tasks
- 設定web server listen request 及回傳 response
- 定義directory structure

Express - Routing

- 根據傳進server的request，去執行對應的code
- 提供簡單介面做routing

  (把URL對應到要處理的code，如去DB讀資料、寫入DB)

DEMO-作業1

建立第一個Express App

First Express App

```
var express = require("express");
var app = express();
```

- 安裝express：npm install express --save
- require(“express”)
  include所有node_modules中express資料夾裡的內容
- node app.js
  沒有錯誤代表執行成功
- callback function包含2個參數
  req：觸發這個route的request物件
  res：回傳response用的物件
- res.send 回傳

```
// "/" => "Hi there!"
app.get("/", function(req, res){
    res.send("Hi there!");
});
```

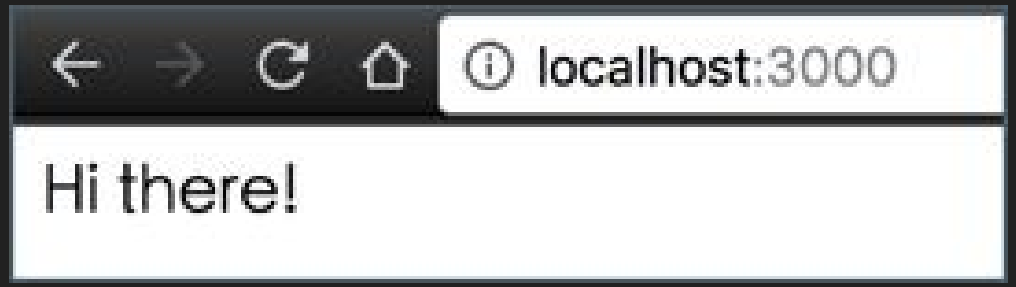
要讓server監聽某個PORT

也有callback function

```
// Tell Express to listen for requests (start server)

app.listen(3000, function(){
    console.log("Server has started!!!");
});
```

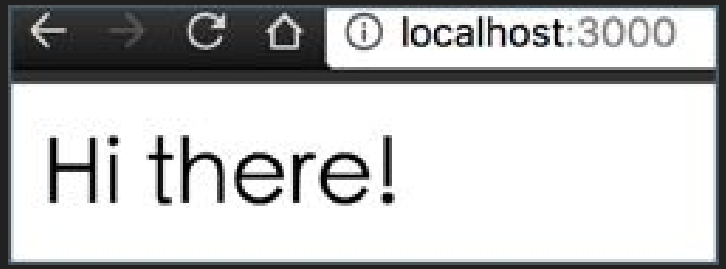
```
// "/bye" => "Goodbye!"
app.get("/bye", function(req, res){
  res.send("Goodbye!!");
});

// "/dog" => "MEOW!"
app.get("/dog", function(req, res){
    console.log("SOMEONE MADE A REQUEST TO /DOG!!!")
   res.send("MEOW!");
});
```

用browser測試



DEMO

```
kuolundeMacBook-Pro:express-demo kuolun$ node app.js
Server has started!!!
```



npm init 會建立package.json

- npm install express --save
  會在dependencies部分儲存該package資料

```
  "scripts": {
    "test": "echo \"Error: no test spe
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.14.0"
  }
}
```

package.json

- 使用--save安裝packages(會在package.json存紀錄)
- package.json的作用
- 使用npm init 建立新的package.json

```
"dependencies": {
  "express": "^4.14.0"
}
```

dependencies

- 這個package需要用到的其他packages

```
"dependencies": {
  "accepts": "~1.3.3",
  "array-flatten": "1.1.1",
  "content-disposition": "0.5.1",
  "content-type": "~1.0.2",
  "cookie": "0.3.1",
  "cookie-signature": "1.0.6",
  "debug": "~2.2.0",
  "depd": "~1.1.0",
  "encodeurl": "~1.0.1",
  "escape-html": "~1.0.3",
  "etag": "~1.7.0",
  "finalhandler": "0.5.0",
  "fresh": "0.3.0",
  "merge-descriptors": "1.0.1",
  "methods": "~1.1.2",
  "on-finished": "~2.3.0",
  "parseurl": "~1.3.1",
  "path-to-regexp": "0.1.7",
  "proxy-addr": "~1.1.2",
  "qs": "6.2.0",
  "range-parser": "~1.2.0",
  "send": "0.14.1",
  "serve-static": "~1.11.1",
  "type-is": "~1.6.13",
  "utils-merge": "1.0.0",
  "vary": "~1.1.0"
},
```

練習時間 - 10分鐘

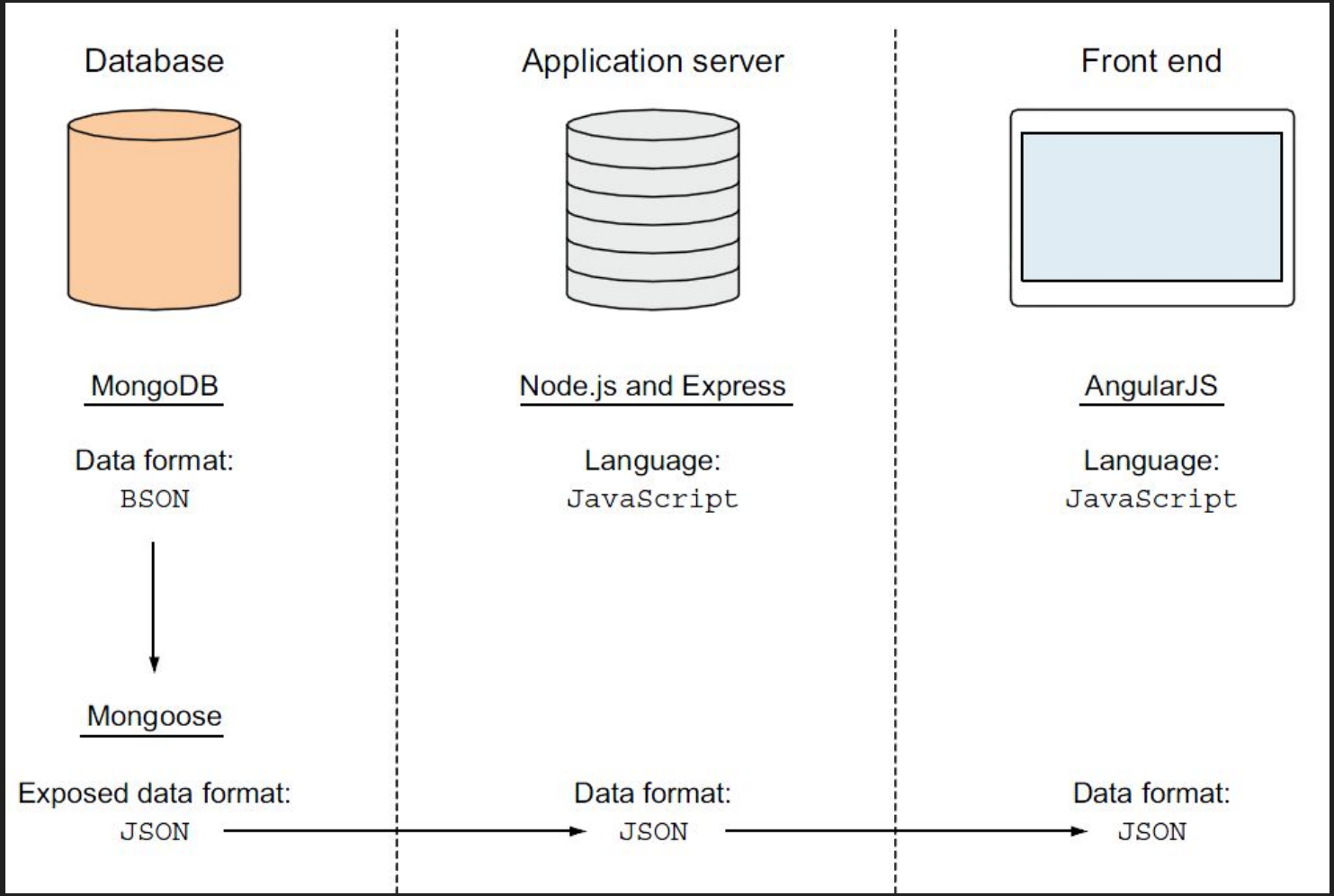
- 切換到expressApp資料夾，建立package.json
- 安裝express package
- 寫出3個route，分別回傳不同的結果
- 用browser測試
- ctrl + c 可跳出node process

Express-記得visitor是誰?

- Nodejs只會看到一連串的HTTP requests
- HTTP是 stateless protocol，沒有儲存session state的概念
- 必須自己來？
- Express也有support ！

  sessions package

MEAN Stack架構圖


Database
Application server
Front end
MongoDB
Node.js and Express
AngularJS
Data format:
BSON
Language:
JavaScript
Language:
JavaScript
Mongoose
Exposed data format:
JSON
Data format:
JSON
Data format:
JSON

其他相關

- Twitter Bootstrap：http://getbootstrap.com/

  建立好的user interface/Respnsive Grid Layout

- Git：source control

  管理code

- Heroku：支援node.js的雲端host

  host app on live URL

建立及設定MEAN Project

建立package.json

- 每個Node application都會存在一個在root folder
- 包含metadata及依賴的package資訊
-

```
{
  "name": "application-name",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "express": "~4.9.0",
    "body-parser": "~1.8.1",
    "cookie-parser": "~1.3.3",
    "morgan": "~1.3.0",
```

Various metadata defining application

Package dependencies needed for application to run

shop 資料夾

-新增shop資料夾

-切換到shop 資料夾, npm init

-建立public 及 styles folder

-建立index.html及 custom.css

Template Engine

- Jade / EJS / JsHtml / Hogan
- 基本流程:
  - 建立HTML template, 包含放data的地方(placeholder)
  - 傳入data
  - engine會compile這兩個成final HTML給browser

加入BootStrap及CSS

```
<!-- BootStrap CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
<!-- font awesome CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css" />
<!--BootStrap social button-->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-social/5.0.0/bootstrap-social.css">
<!--custom css-->
<link rel="stylesheet" href="/css/custom.css">
```

BootStrap需要jQuery

```
<!--BootStrap需jQuery(<3.0)-->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
<!--BootStrap JS-->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-
```

meta for mobile device

```
<title>MEAN Stack ShopApp</title>
<!--meta for mobile device-->
<meta name="viewport" content="width=device-width" initial-scale="1.0">
```

測試bootstrap

index.html

```html
<body>
    <!--導覽列-->
    <div class="container">
        <div class="row">
            <!-- 內容 -->
            <div class="container">
                <button type="submit" class="btn btn-default btn-primary">click</button>
                <hr>
                <footer>
                    <p>&copy; 2016 Company, Inc.</p>
                </footer>
            </div>
        </div>
</body>
```

server.js

```
var express = require('express');
var app = express();

app.get('/',function(req,res){
    console.log(__dirname);
    res.send('this is main page');
});


app.use(express.static(__dirname+'/public'));

app.listen(3000,function () {
    console.log('Server is running...');
});
```

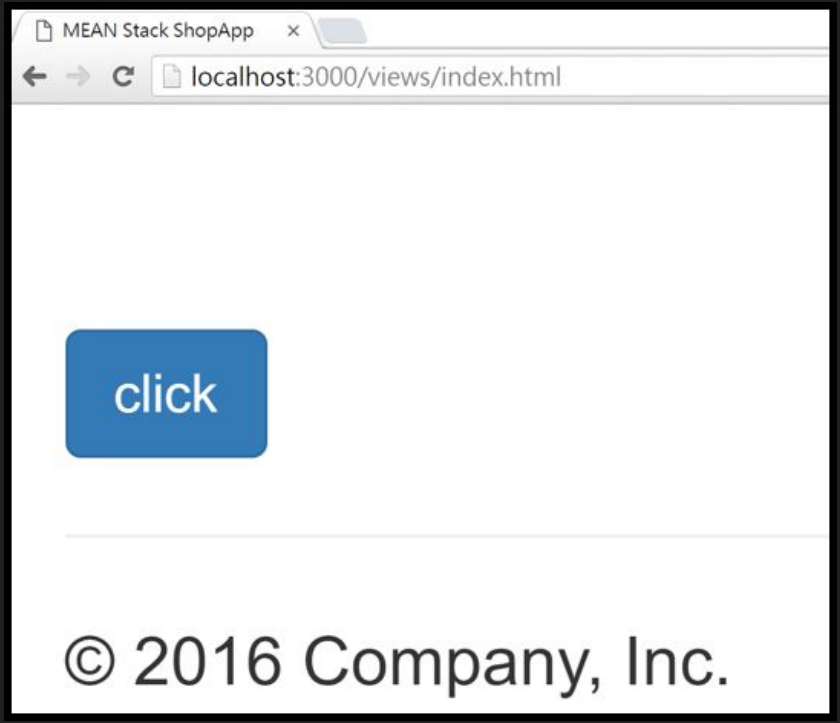
Restart Application

- 若有改application code，需要resart node
- EJS / CSS file / client-side JS 會動態更新
- Ctrl + C => stop
- $ node server.js
- 自動restart
  - 安裝nodemon (globally)
    - $ npm install -g nodemon

Demo + 練習10分鐘

localhost:3000/index.html

看BootStrap有沒有載入



Express middleware

在app.js中可看到一些app.use

當request進到application時，會輪流通過每個middleware

每個middleware可能會對request做些事情

然後會pass給下一個直到application logic部分

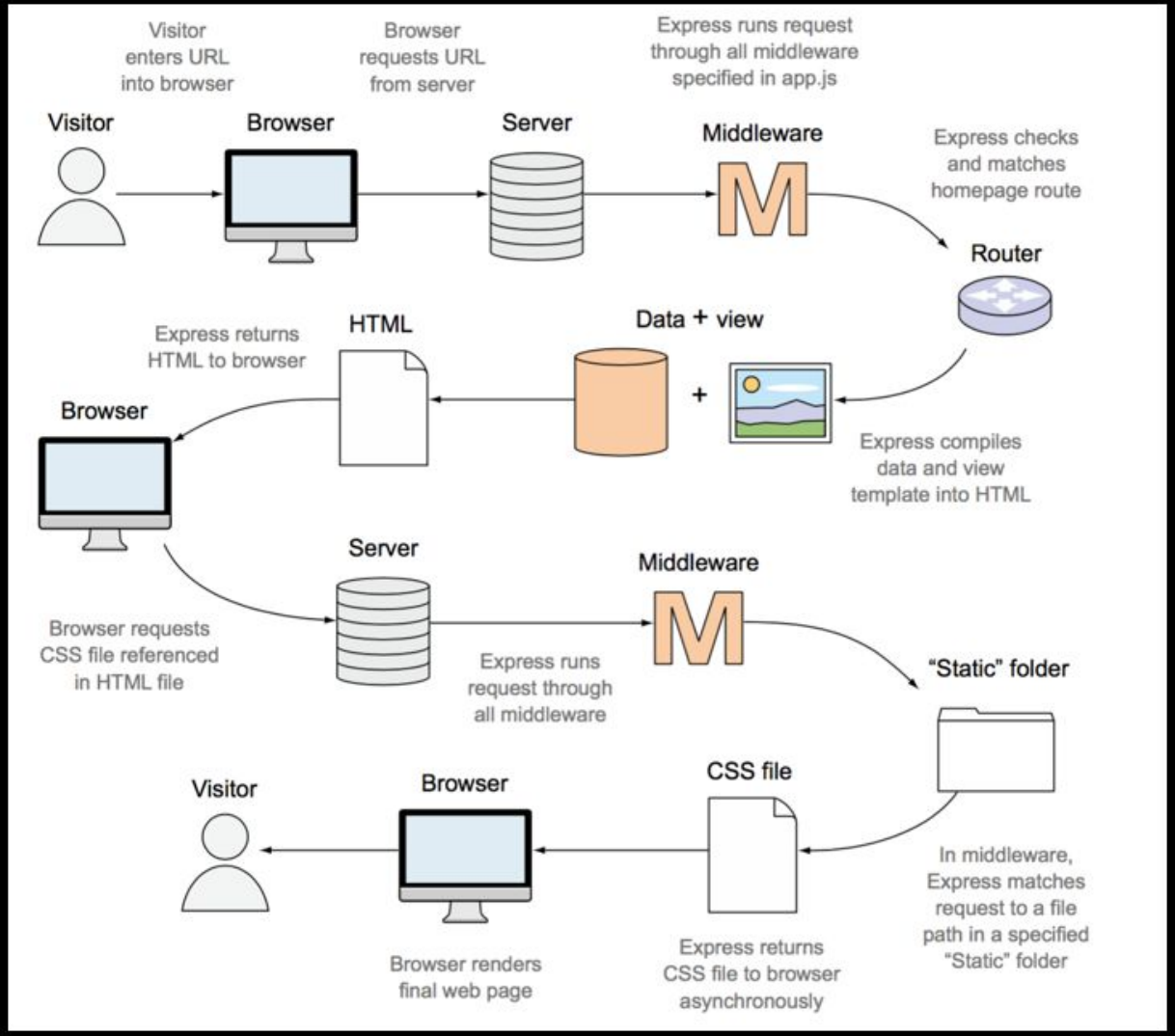
然後回傳response

如：app.use(express.cookieParser( ));

Express middleware

- 有個預設的middleware會去尋找對應到static files的path，當有match時，Express會非同步的return這個file，確保Node.js process不會被這個operation block住

- 當一個request通過所有的middleware後，Express會嘗試去match request的path跟 我們定義的route

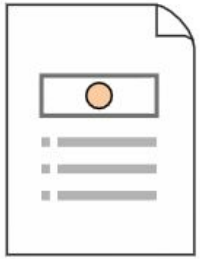

Visitor enters URL into browser
Browser requests URL from server
Express runs request through all middleware specified in app.js
Visitor
Browser
Server
Middleware
Express checks and matches homepage route
Router
Express returns HTML to browser
HTML
Data + view
+
Browser
Express compiles data and view template into HTML
Server
Middleware
Browser requests CSS file referenced in HTML file
Express runs request through all middleware
"Static" folder
Visitor
Browser
CSS file
In middleware, Express matches request to a file path in a specified "Static" folder
Browser renders final web page
Express returns CSS file to browser asynchronously

專案要建立的頁面

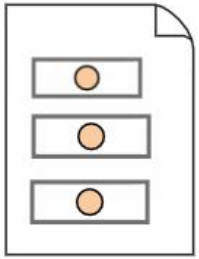
Products

List page

Details page



Cart page



Others

About page

檔案結構

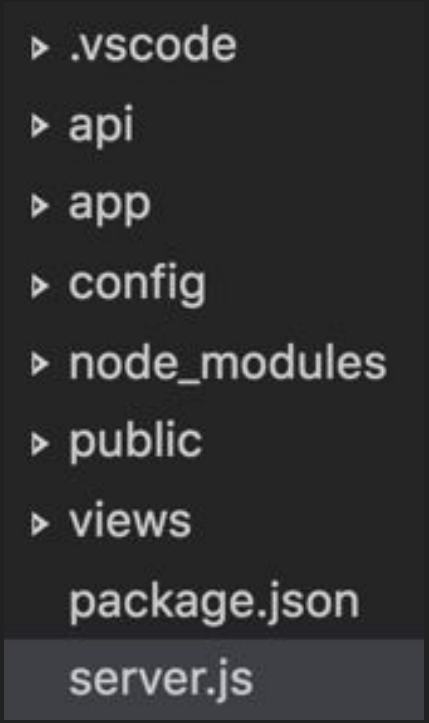
app -controller

views - for EJS

public - for angular+HTML

config -設定

api -建立fake data



DataBase

什麼是Database?

- 儲存資料的地方，免得網頁重整東西就不見
- 有一個跟資料互動的介面(interface)

SQL(relational) vs NoSQL(non-relational)

```
USERS TABLE
id | name  |  age  |  city
-------------------------
1  | Tim   |  57   |  NYC
2  | Ira   |  24   |  Missoula
3  | Sue   |  40   |  Boulder
```

```
SELECT * FROM users
INSERT jflgfdlg  into
```

```
db.dogs.find()
db.dogs.delete({age:14})
```

```
COMMENTS TABLE
id |        text
---------------------------
1  | "lol"
2  | "Come visit Montana!"
3  | "I love puppies!!!"
4  | "Seriously Montana is great!"
```

```
USER/COMMENTS JOIN TABLE
userId  |  commentId
---------------------------
   1         3
   2         2
   2         4
   3         1
```

如果需要新增一個欄位

```
USERS TABLE
id | name  |  age  |  city        favColor
---------------------------
1  | Tim   |  57   |  NYC
2  | Ira   |  24   |  Missoula    purple
3  | Sue   |  40   |  Boulder
```

BSON(binary javascript object notation)

基本上是JS的物件

name/value pairs

```
============================
A NON-RELATIONAL DATABASE:
============================
{
  name: "Ira",
  age: 24,
  city: Missoula,
  comments: [
    {text: "Come visit Montana!"},
    {text: "Seriously Montana is great!"}
  ]
}
```

MongoDB

https://www.mongodb.com/

什麼是MongoDB?

- non-relational database
- 現在跟Node/Express搭配最popular的DB
- 有很多好用的工具

Relational v.s. Document database

firstName	middleName	lastName	maidenName	nickname
Simon	David	Holmes		Si
Sally	June	Panayiotou		
Rebecca		Norman	Holmes	Bec

Relational v.s. Document database

firstName: "Simon"	middleName: "David"	lastName: "Holmes"	nickname: "Si"
lastName: "Panayiotou"	middleName: "June"	firstName: "Sally"	
maidenName: "Holmes"	firstName: "Rebecca"	lastName: "Norman"	nickname: "Bec"

MongoDB document

```
{
  "firstName" : "Simon",
  "lastName" : "Holmes",
  _id : ObjectId("52279effc62ca8b0c1000007")
}
```

Mongoose

- 雖然mongoDB很彈性
- 但大多applications還是需要structure for data (是app需要，非DB)，所以就有了Mongoose
- “elegant MongoDB object modeling for Node.js”
- 讓我們更容易操作DB，如同jQuery讓我們更容易操作DOM 但是不是一定要用Mongoose

Mongoose-什麼是Data Modeling?

- 簡單來說就是:描述data看起來應該是怎麼樣
- 定義document內可以有什麼data跟一定要有什麼data
- 例:

  要儲存User的資料

  user能夠存 firstname/ lastname / email / phone

  但我們只需要 firstname / lasname / email (required)

  且

  email必須為unique

- 這些會定義在 schema

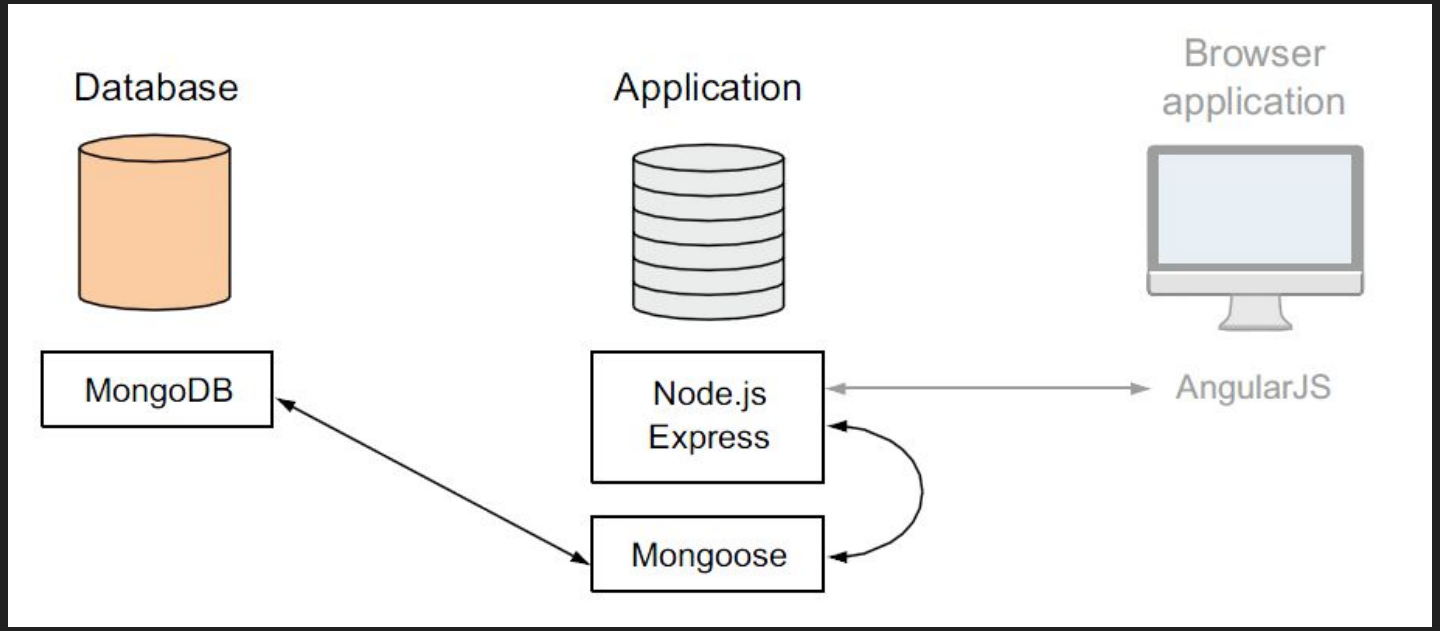
Mongoose-還提供什麼？

- 在MongoDB上建構一層功能
- 更容易連接MongoDB, 讀寫data
- 在schema加上data 驗證

用MongoDB+Mongoose
打造Data model

- Mongoose如何橋接Express跟Node application到MongoDB
- 用Mongoose定義Data model schema
- 連接application到DB
- mlab跟Robomongo

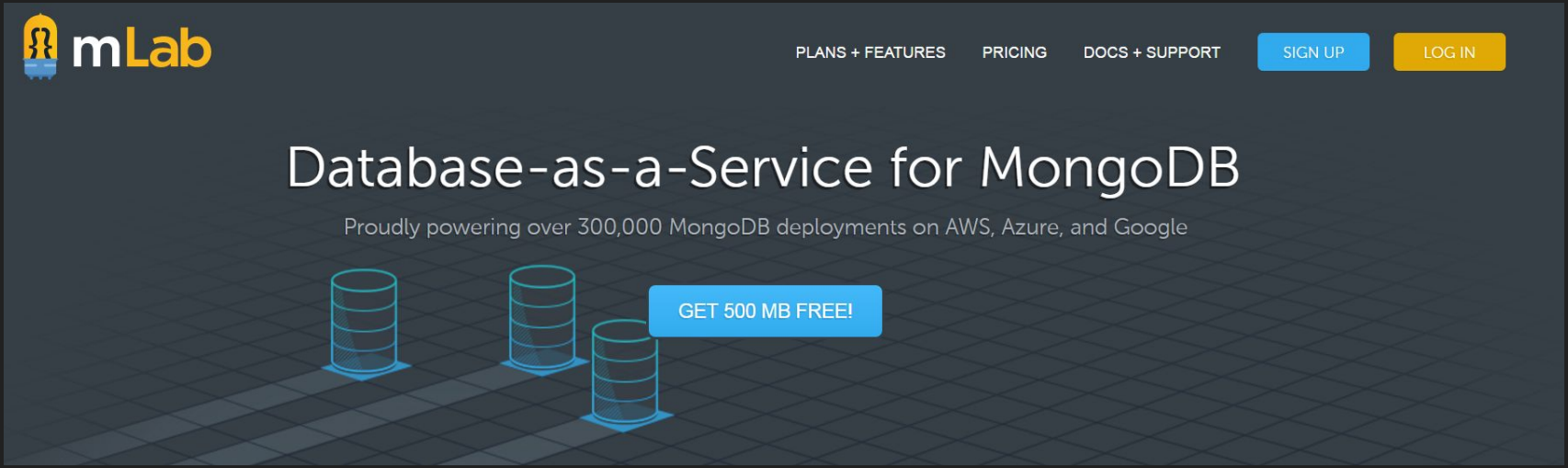
Mongoose 圖示


Database
Application
Browser application
MongoDB
Node.js
Express
AngularJS
Mongoose

註冊及設定mLab

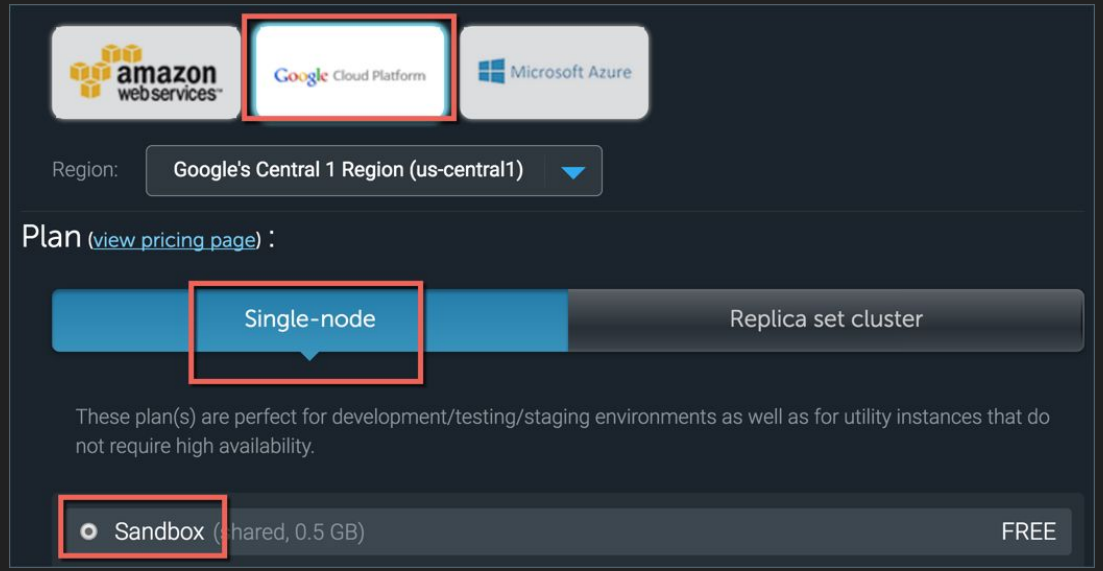
https://mlab.com/

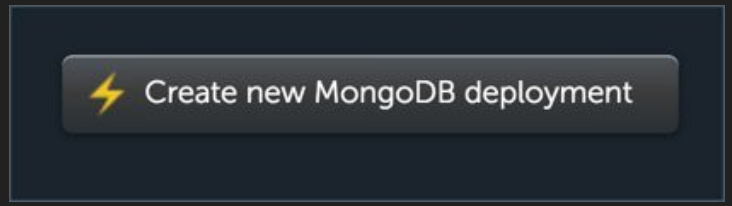
連接mlab （DBaaS)

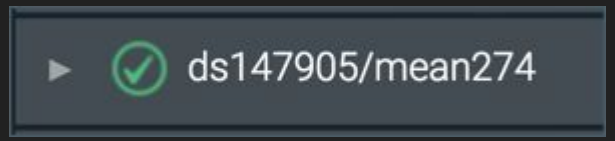


新增DB

輸入DB name

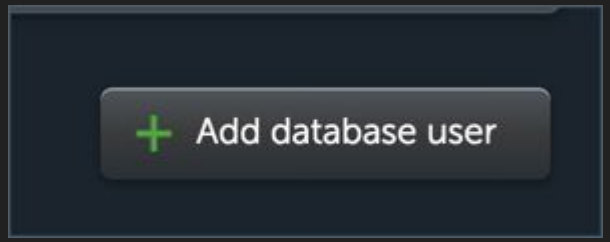


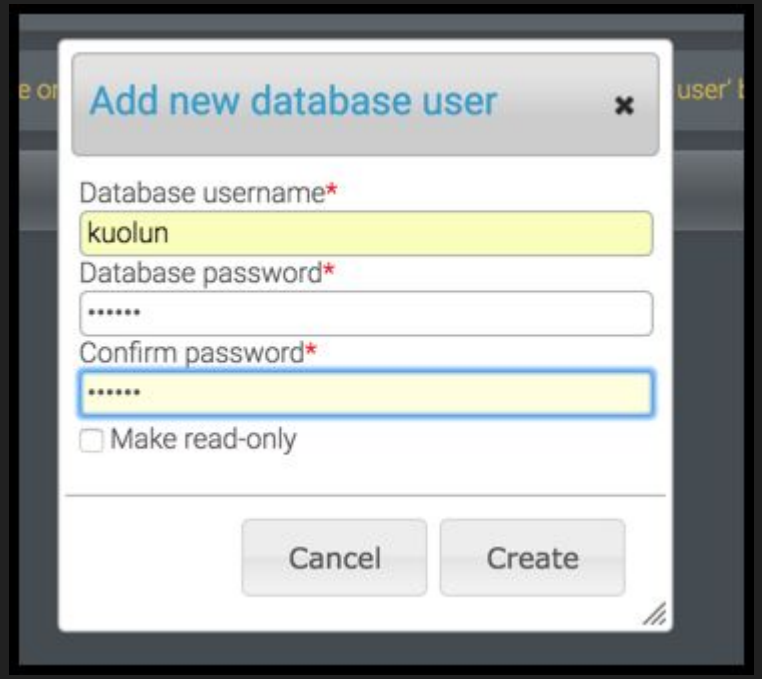


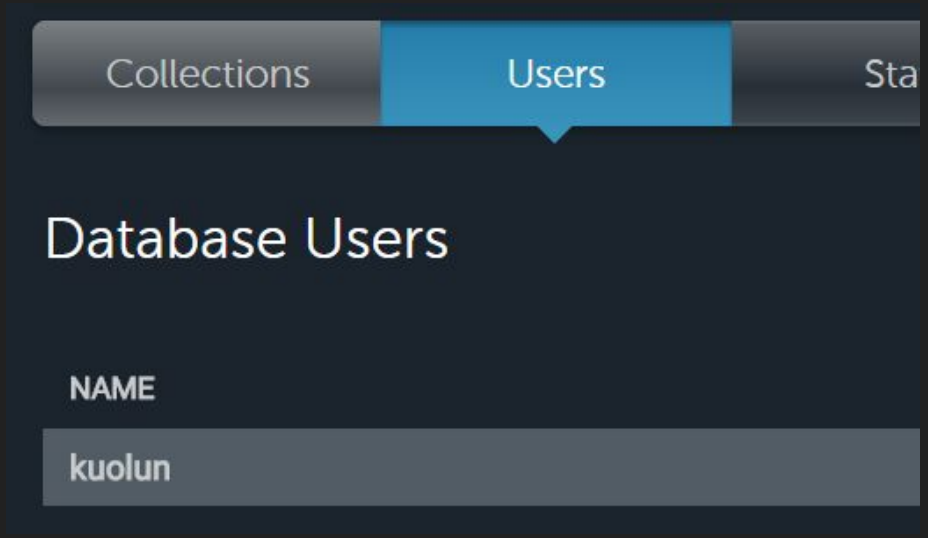


建立db user

點入到剛建立的DB，選Users tab，新增user







MongoDB URI

複製框框處網址，把剛建立的user帳密
填入user/password

To connect using the mongo shell:

```
% mongo ds017584.mlab.com:17584/mean_course -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard MongoDB URI (what's this?):

```
mongodb://<dbuser>:<dbpassword>@ds017584.mlab.com:17584/mean_course
```

Demo - 透過Mongoose新增資料

testDB.js

```
1   var mongoose = require('mongoose');
2   mongoose.connect('');
3
4   var CatSchema = new mongoose.Schema({
5       name: String,
6       age: Number,
7       type: String
8   });
9
10  var Cat = mongoose.model('Cat', CatSchema);
11
```

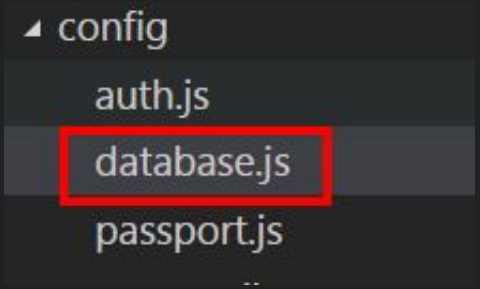
練習:10分鐘

- 建立mlab上的database
- 取得DB 連線
- 建立一筆資料到DB

step1：設定

- 切換到shop資料夾
- npm install mongoose --save

step2: 建立 /config/database.js




```
//引入mongoose
var mongoose = require('mongoose');
```

```
var dbURI = "mongodb://kuolun:kuolun@ds017584.mlab.com:17584/mean_course";
//連到mlab
mongoose.connect(dbURI);
```

step3: Server.js

```
// 連接mlab
require('./config/database.js')
```

Mongoose connection events

- connected
- error
- disconnected

```
Server is running on port 3000..........
Mongoose connected to mongodb://kuolun:kuolun@ds017584.mlab.com:17584/mean_cours
e
```

Demo

練習 10分鐘

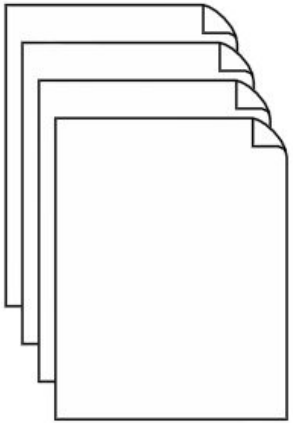
- 建立dbuser
- 修改server.js / database.js
- 測試是否連DB成功

MongoDB
Object-Document Modeler (ODM)

MongoDB Object-Document Modeler (ODM)

- 在application內管理data model
- 每筆資料稱為一個document
- 一群documents會形成一個 collection (可想成table)
- 在Mongoose, document的定義稱為 schema
- 在schema內每個data項目稱為 path

Collection

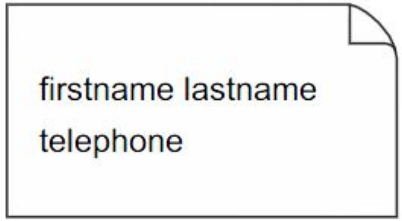


一個collection含
有許多document

Document

每個document
-含有data
-由schema定義結構

Schema



每個schema由一
群path組成

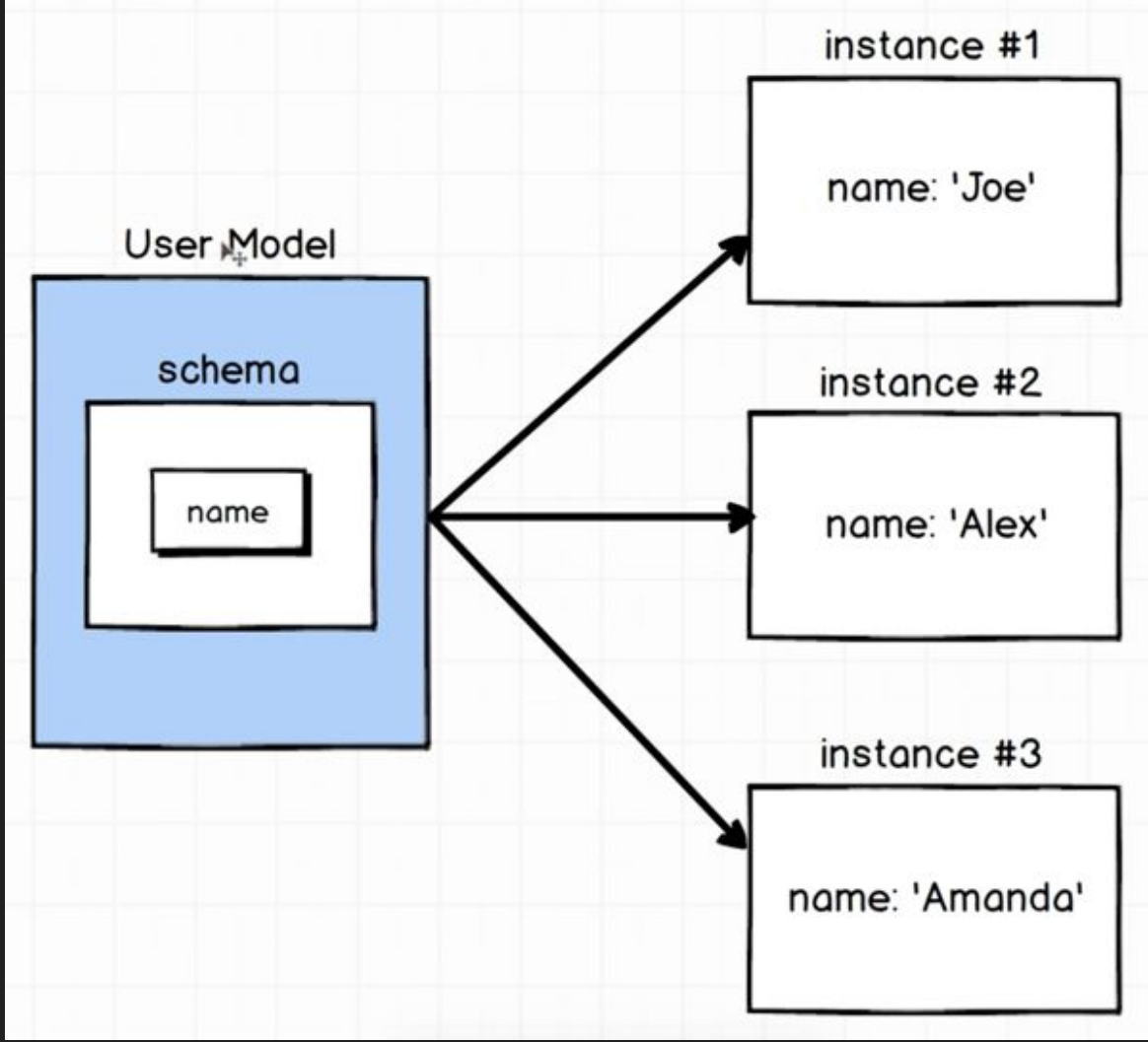
Path

```
firstname:{
    type: String,
    required:true}
```

每個path可有多個
property

Model

- schema compile過後稱為Model
- Mongoose所有的data互動，都是透過Model


User Model
schema
name
instance #1
name: 'Joe'
instance #2
name: 'Alex'
instance #3
name: 'Amanda'

如何在application內定義data?

- JS Object

MongoDB document

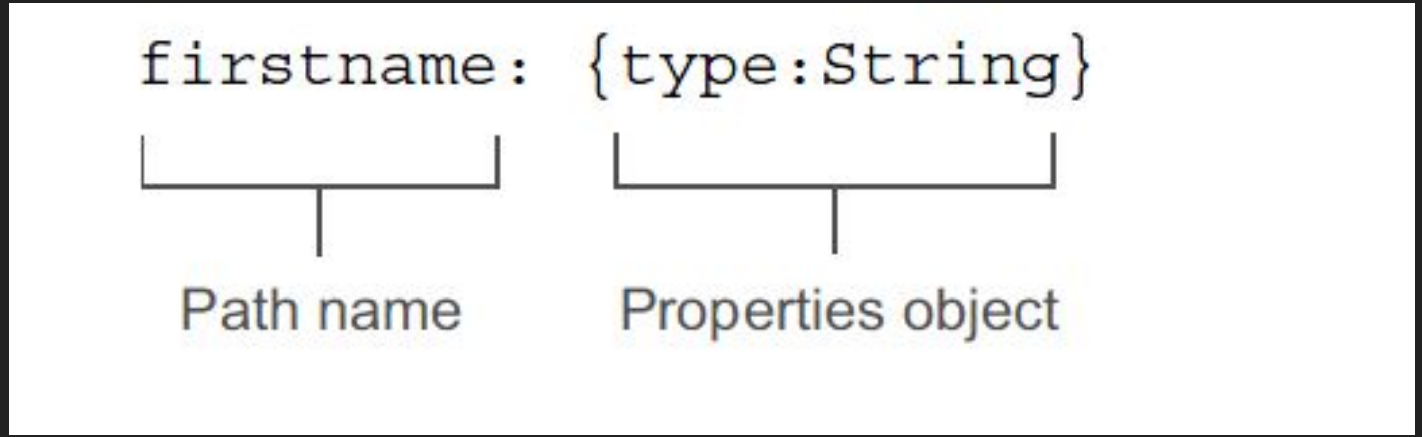
```
{
  "firstname" : "Simon",
  "surname" : "Holmes",
  _id : ObjectId("52279effc62ca8b0c1000007")
}
```

對應的Mongoose schema

```
{
  firstname : String,
  surname : String
}
```

Schema path

firstname:String 是簡單寫法 (如果只有要定義data type)

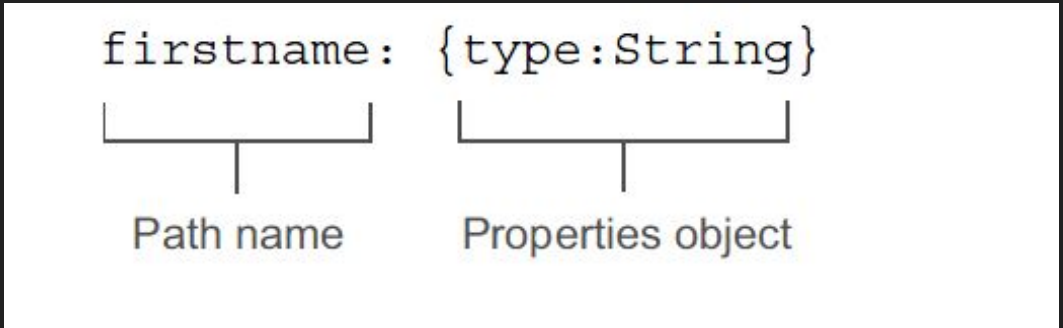


data types - 8種 (每個path都要指定其中1種)

1. String
2. Number
3. Date -- ISODate object
4. Boolean
5. Buffer -- for binary information，如images
6. Mixed -- 任意data type
7. Array -- 可以是相同data type的array/ array of nested sub-documents
8. ObjectId -- 通常用來參考別的documents中的_id path

properties object

- javascript obejct
- 至少要包含data type
- validation / 範圍 / 預設值



定義 Mongoose schemas

Mongoose提供constructor function：

```
var mongoose = require('mongoose');

var  review = new mongoose.schema({ });
```

Example schema

```
var locationSchema = new mongoose.Schema({
  name: String,
  address: String,
  rating: Number,
  facilities: [String]
});
```

Example

```
locations: [{
  name: 'Starcups',
  address: '125 High Street, Reading, RG6 1PS',
  rating: 3,
  facilities: ['Hot drinks', 'Food', 'Premium wifi'],
  distance: '100m'
}]
```

name is a string

address is another string

rating is a number

facilities is an array of strings

Default value

```
rating : { type : Number , “default” : 0 }
```

Required Field

```
name : { type : String , required: true }
```

Number Boundaries

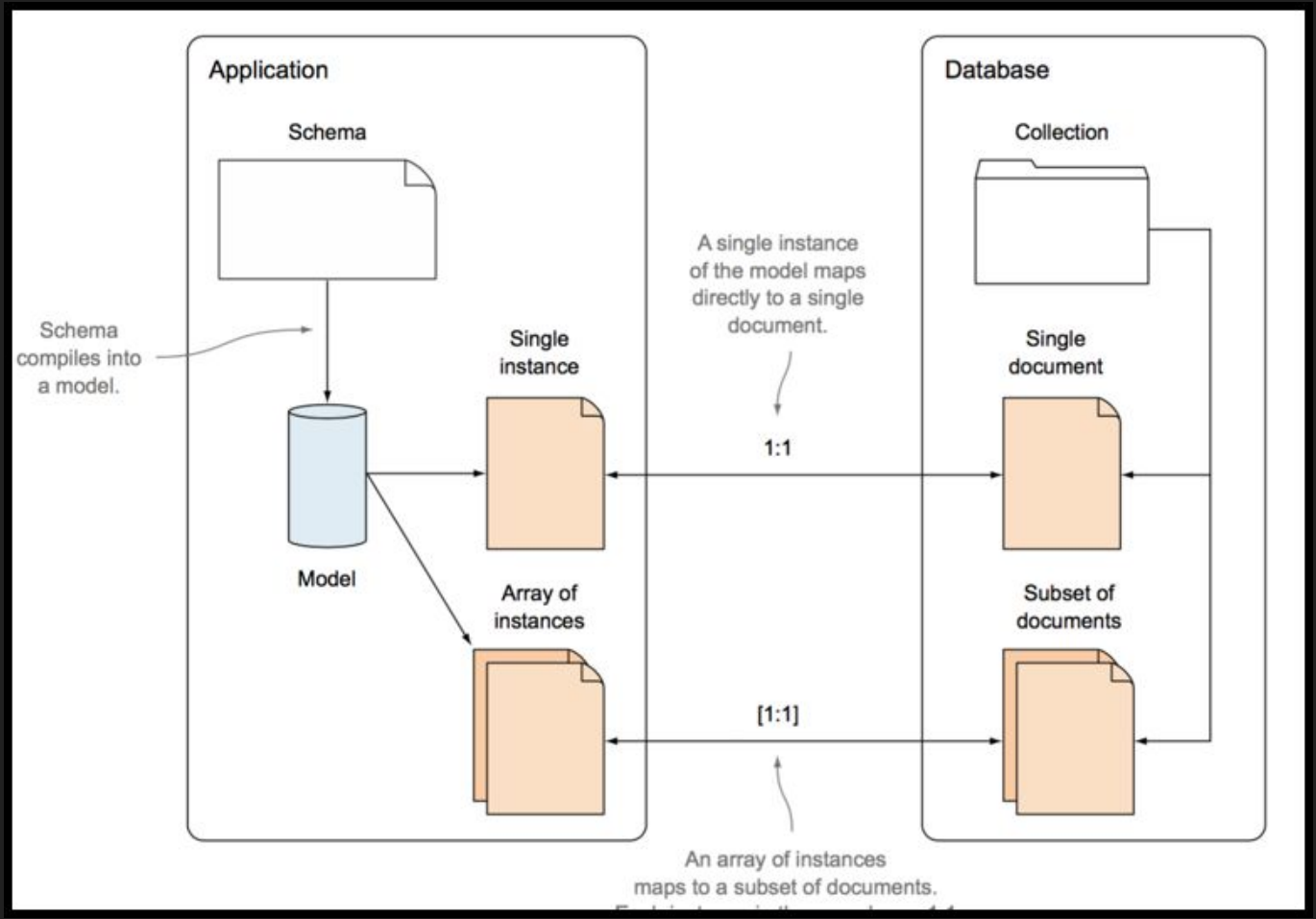
```
rating : {type: Number, “default “ :0, min:0 , max:5 }
```

Indexes

- 更快更有效率的query
- 不用掃描整個collection內的documents
- 每個document可以有多個index
- 舉例:
  - 家裡的文件是否有分類, 假設要全部擺在一起, 如何找出某年某月的信用卡賬單？

Schema to model

- Application不直接跟schema互動，而是跟model
- model是compile過的schema
- 一個model的實體(instance)會對應到一個document
- model可以 create/read/save/delete data


Application
Schema
Schema compiles into a model.
Model
Single instance
Array of instances
A single instance of the model maps directly to a single document.
1:1
[1:1]
An array of instances maps to a subset of documents.
Database
Collection
Single document
Subset of documents

compile

```
mongoose.model('Location', locationSchema, 'Locations');
```

Connection name

The name of the model

The schema to use

MongoDB collection name (optional)

```
mongoose.model('Location', locationSchema);
```

來建立schema

- product category
- product
- user

建立app/models/category.js

category名稱

類別:String

唯一

小寫

category.js

```
1   //include mongoose
2   var mongoose = require('mongoose');
3   var Schema = mongoose.Schema;
4
5   //create Category schema
6   var Categoryschema = Schema({
7       name: {
8           type: String,
9           unique: true,
10          lowercase: true
11      }
12  });
13
14
15  //export Category model
16  module.exports = mongoose.model('Category', Categoryschema);
```

建立app/models/product.js

產品的類別 對應到Category Model

產品名稱

產品價格

產品圖片

product.js

```
1   //include mongoose
2   var mongoose = require('mongoose');
3   var Schema = mongoose.Schema;
4
5   //create product schema
6   var schema = Schema({
7       category: {
8           type: Schema.Types.ObjectId,
9           ref: 'Category'
10      },
11      name: String,
12      price: Number,
13      image: String
14  });
15
16  //export product model
17  module.exports = mongoose.model('Product', schema);
```

一筆product在DB裡的樣子

```
1  {
2      "_id": {
3          "$oid": "578b461223dd391e9032422c"
4      },
5      "image": "http://goo.gl/p95H3Y",
6      "price": 215,
7      "name": "Kindle fire",
8      "category": {
9          "$oid": "5785f1c5a18925d41b2842a8"
10     },
11     "__v": 0
12 }
```

建立app/models/user.js

- email
- token-facebook認證後提供
- facebook--profile id
- profile
  - picture
  - username
- data
  - cart: item陣列
  - totalValue

```
//include mongoose
var mongoose = require('mongoose');
var Schema = mongoose.Schema;


//create user Schema
var userSchema = Schema({
    email: {…
    },
    token: String,
    facebook: String,
    //基本資料
    profile: {…
    },
    //傳輸資料
    data: {…
    }

});

//export user model
module.exports = mongoose.model('User', userSchema);
```

⚓ Profile Page

f Facebook

id: 10157090854765075
email: kuolun@gmail.com
name: Kuo-Lun Huang

```
//create user Schema
var userSchema = Schema({
    email: {
        type: String,
        unique: true,
        lowercase: true
    },
    token: String,
    facebook: String,
    //基本資料
    profile: {
        username: {
            type: String,
            default: ''
        },
        picture: {
            type: String,
            default: ''
        }
    },
```

```
//傳輸資料
data: {
    totalValue: {
        type: Number,
        default: 0
    },
    //購物車array 每個element包含產品 數量
    //reference到product id
    cart: [{
        product: {
            type: mongoose.Schema.Types.ObjectId,
            ref: 'Product'
        },
        quantity: {
            type: Number,
            default: 1,
            min: 1
        },
        subtotal: {
            type: Number,
            default: 0,
            min: 0
        }
    }]
}
```

```
1  {
2      "_id": {
3          "$oid": "578c9cc8cc7bb9342873fac9"
4      },
5      "email": "kuolun@gmail.com",
6      "token":
"EAAOtYNHiRcsBAJ7HKsf7VC5VAGBaIX0hhRc1olMnb4iUguolRq7DMM0Z
5I0GrLzi80Vah61I2Ho6rSvZCZCfRzSMx6cwAZDZD",
7      "facebook": "10157090854765075",
8      "data": {
9          "cart": [
10             {
11                 "product": {
12                     "$oid": "578b46ad04a4ce6490e74b99"
13                 },
14                 "_id": {
15                     "$oid": "5792208bb72c061100e5e31a"
16                 },
17                 "subtotal": 2892,
18                 "quantity": 3
19             }
20         ],
21         "totalValue": 2892
22     },
23     "profile": {
24         "picture": "https://graph.facebook.com/10157090854
25         "username": "Kuo-Lun Huang"
26     },
27     "__v": 70
28 }
```

練習:10分鐘

- 建立app/models資料夾
- 建立category.js
- 建立product.js
- 建立user.js