

# iOS 雲端服務開發

## 第二週

# 課程資料

📍 <https://github.com/finfin/CloudAppDev>

# Node.js 是什麼

- Javascript 運行平台
- 讓Javascript跳脫瀏覽器的限制
- 輕易建立Web Server
- 事件驅動 (Event driven)
- 非阻塞式 (Nonblocking)

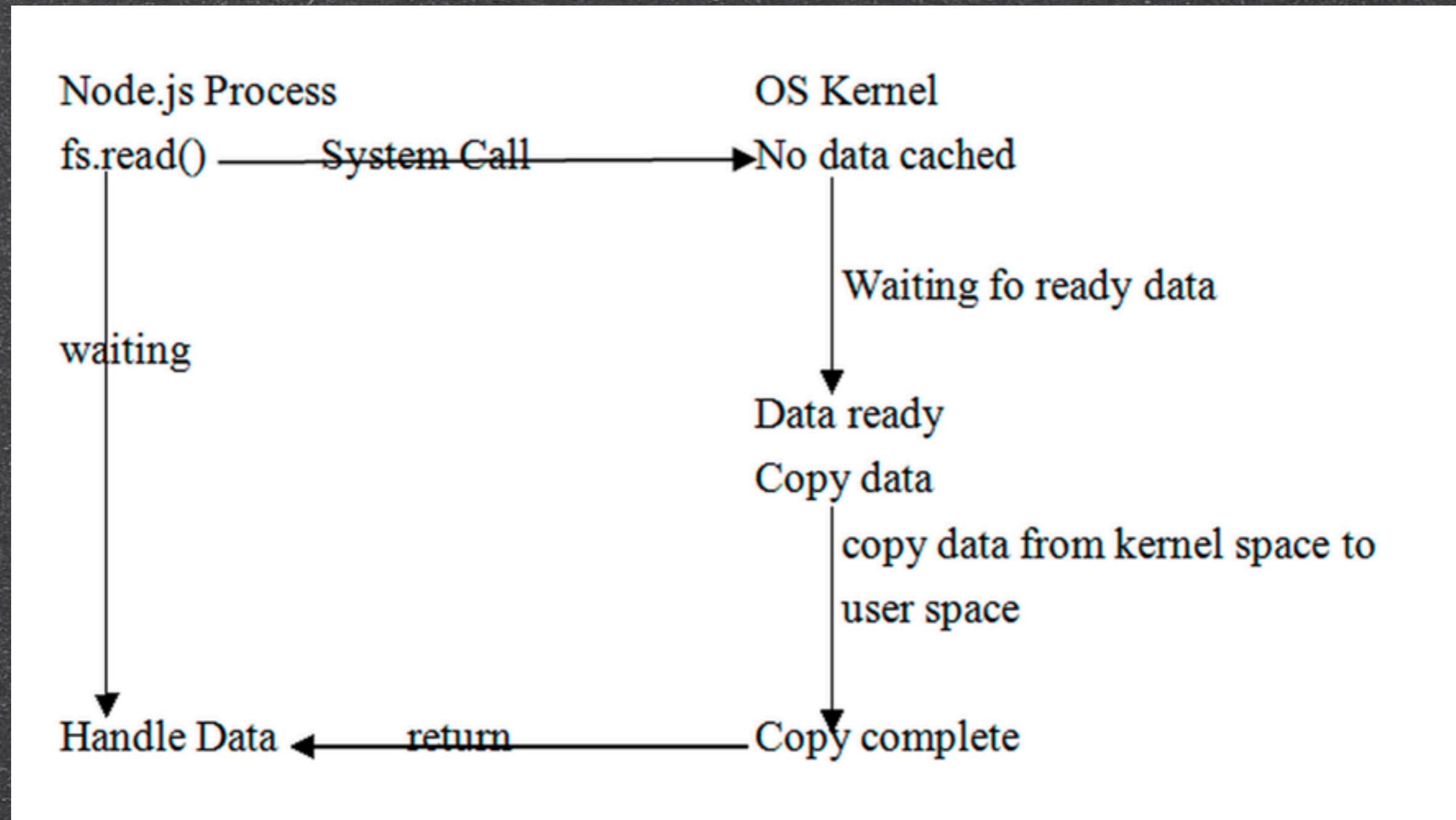
# 為什麼使用 Node.js

- 前後端語言統一，且能共用模組
- 其特性非常適合即時網路服務
- 本身就可以是一個HTTP伺服器
- 強大的套件管理及開源社群

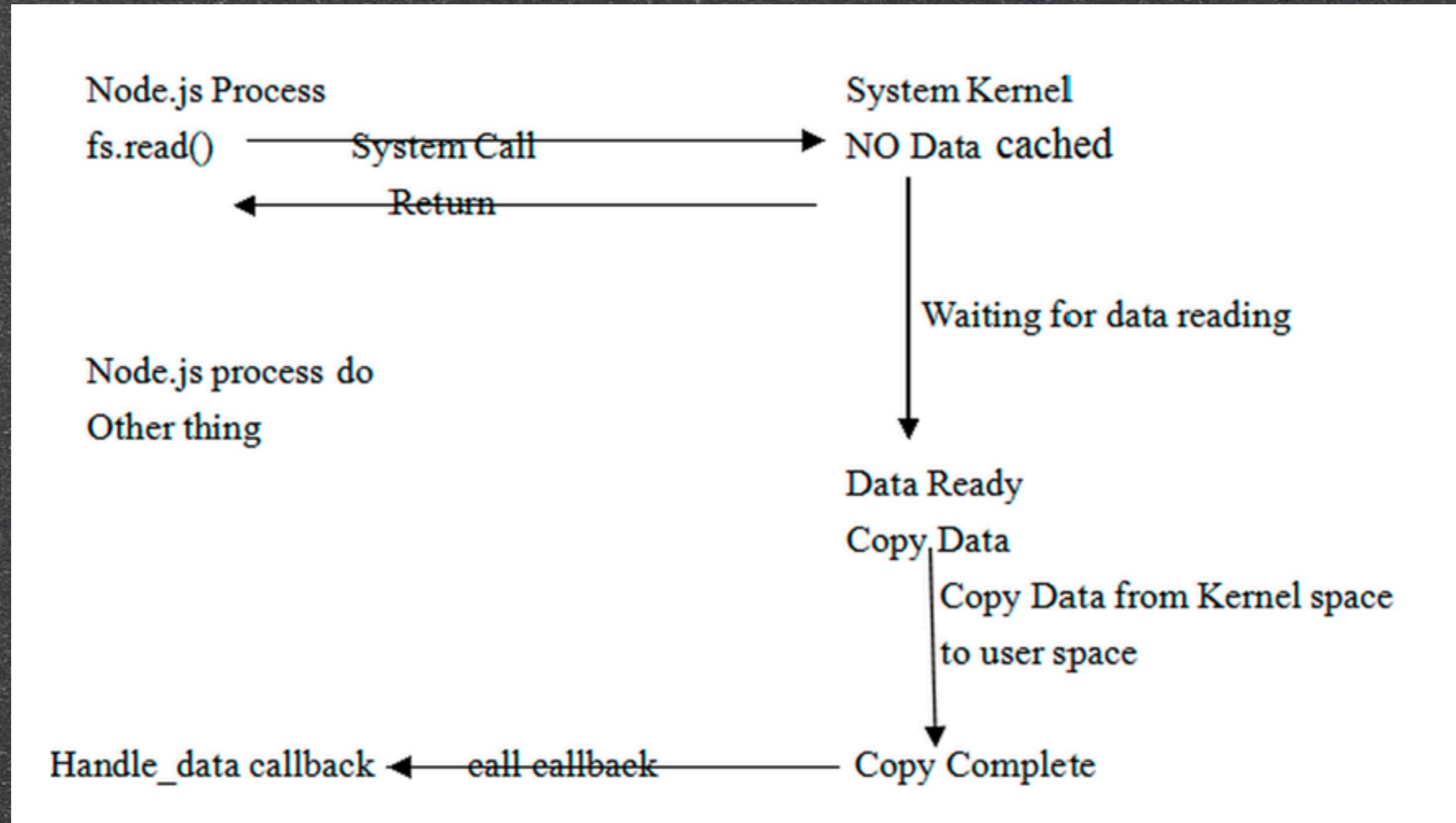
# Node.js 是萬靈丹嗎？

- 非常新的技術，版本甚至還沒1.0
- 對需要大量CPU的工作不拿手 - 直譯式語言
- library不如其他語言來的多樣及完整  
(但在快速發展中)

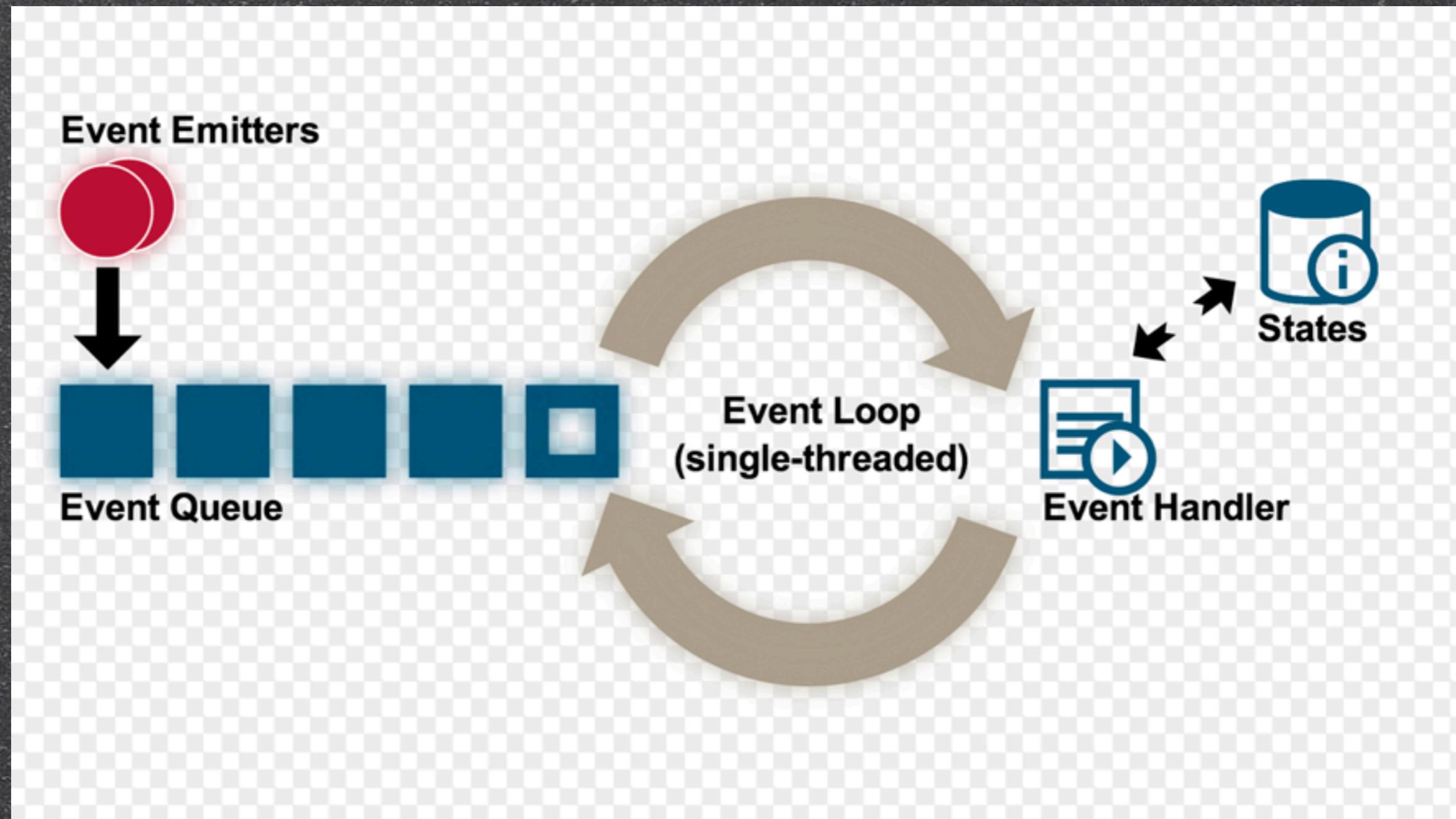
# 阻塞式 vs 非阻塞式



# 阻塞式 VS 非阻塞式



# 事件驅動



# Hello World!

• helloworld.js

```
console.log("Hello World!");
```

• Run:

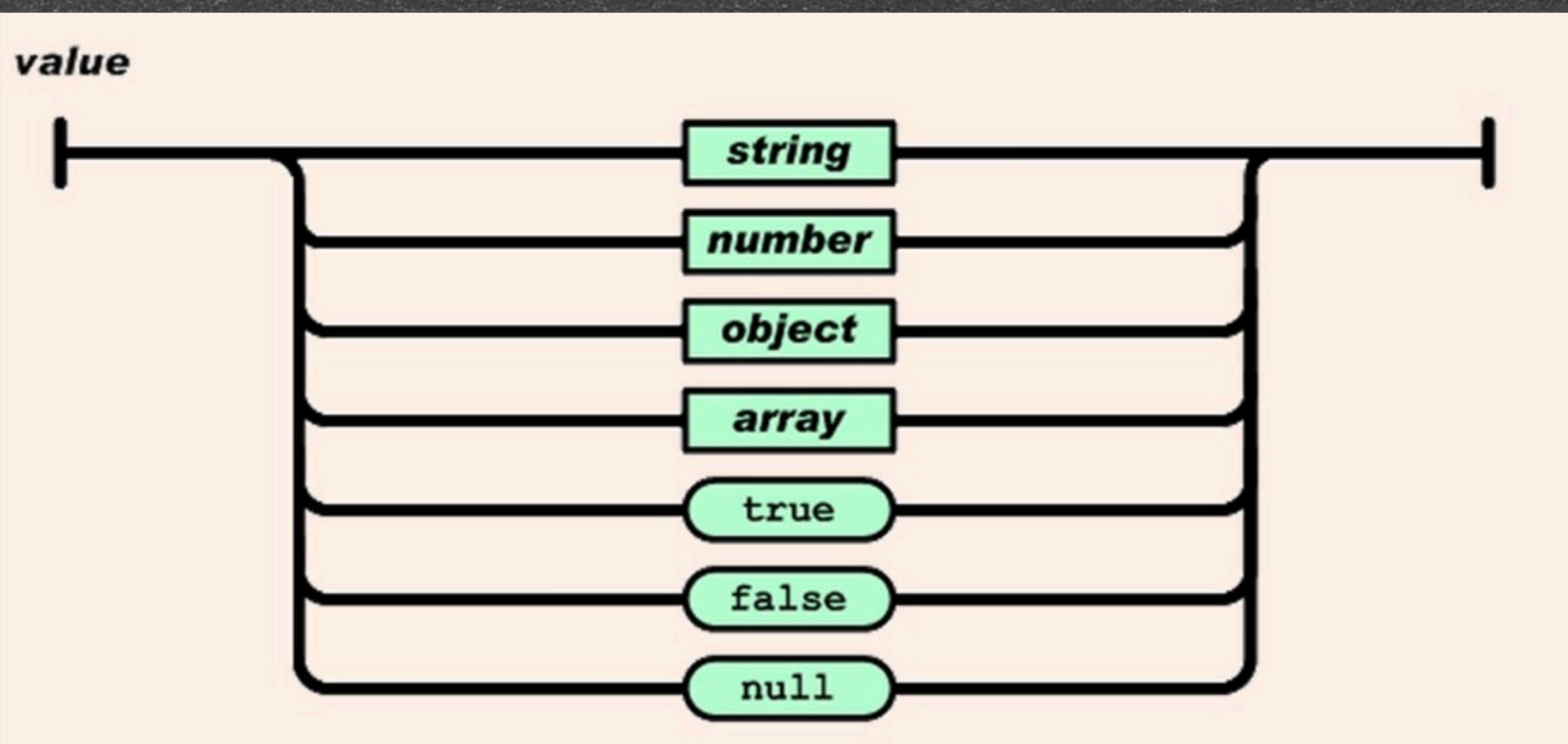
```
$ node helloworld.js  
Hello World!
```

# JSON

## JavaScript Object Notation

- 資料交換語言，為字串格式
- 用於Web開發，近年某些NoSQL資料庫也有採用
- 格式大致很類似Javascript
- 物件： { keys: values }
- 陣列： [ values ]
- 字串以雙引號包覆，反斜線逸出
- 數字以普通十進位 與 指數表示法(e)

# JSON



# JSON

- ECMAScript 5 原生物件
- JSON.stringify(obj)
  - 可以覆寫toJSON()來改變行為
- JSON.parse(string)
- toJSON

```
var foo = {};
foo.bar = "new property";
foo.baz = 3;

var JSONfoo = JSON.stringify(foo);

var backToJS = JSON.parse(JSONfoo);
```

# JSON

- 練習1：讀入下列JSON字串，並以  
{releaseDate}: {title}格式印出每張專輯  
(album)

```
{  
    "name": "Olivia Ong",  
    "gender": "female",  
    "married": false,  
    "age": 27,  
    "albums": [  
        {  
            "title": "A Girl Meets Bossanova",  
            "releaseDate": "2005/3/24",  
        }, {  
            "title": "Romance",  
            "releaseDate": "2011/7/22"  
        }  
    ]  
}
```

# 一個簡單的HTTP伺服器

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

http.createServer(function(request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("Hello World");
  response.end();
}).listen(8888);
```

# 模組

- 一個模組 = 一個檔案
- 可以讀取.js / .node(C/C++) / .json
- 載入自定模組 / 套件or原生模組
- 不會重複載入
- exports & module.exports

# 模組

```
//data.json
[
  {
    "name": "Steve Jobs",
    "age": 58
  },
  {
    "name": "Bill Gates",
    "age": 58
  }
]
```

```
//person.js

function Person(name, age) {
  this.name = name;
  this.age = age;
}

Person.prototype.intro = function() {
  return "Hi there, I am " + this.name + " and I am " + this.age + " years old now.";
}

module.exports = Person;
```

# 模組

```
//util.js

exports.toJson = function(person) {
    return JSON.stringify(person);
}
```

```
//myApp.js

var Person = require('./person');
var data = require('./data');
var util = require('./util')

var bob = new Person("Bob Dylan", 72);

console.log(bob.intro());
console.log(util.toJson(bob));
```

- 練習：util模組新增一個toPerson(data)，能把傳入的json物件轉成Person並回傳

# 模組

- 無論呼叫多少次require，獲得的都是同一個模組

```
var util1 = require('./util')

console.log(util1.getOwner());

var util2 = require('./util')

util2.setOwner("BOB");
console.log("util2 owner: " + util2.getOwner() + ", util1 owner:" +
util1.getOwner());
```

# 套件

- 多個模組的集合、封裝
- 用於發佈、更新、版本控制、相依性管理等
- npm - Node Package Manager

# 套件

- 目錄結構

```
packageName/  
  bin/  
  lib/  
  doc/  
  test/  
  package.json
```

- package.json欄位 : name, description, version, author, contributors, main, keywords, dependencies
- <http://package.json.nodejitsu.com/>

# NPM 套件管理員

- 安裝套件

```
$ npm install express
```

- 移除套件

```
$ npm uninstall express
```

- 全域安裝

```
$ npm install -g nodemon
```

# NPM 套件管理員

- 全域 vs 本地模式
  - 全域會註冊至環境變數PATH
  - 全域無法以require取得
- 如何require全域套件？
  - 本地也安裝一個
  - 使用npm link (不支援windows)

```
//假設我們把express安裝成全域套件  
$ npm link express
```

# NPM 套件管理員

## • package.json

- 每個專案都要有，方便套件管理
- dependencies下面記錄所需套件及版本
- 直接於專案目錄npm install即可安裝所需套件

```
"dependencies" : {  
    "colors"      : "*",  
    "flatiron"    : "0.1.x",  
    "optimist"    : "0.2.x",  
    "union"       : "0.1.x",  
    "ecstatic"   : "0.1.x",  
    "plates"      : "https://github.com/flatiron/plates/tarball/master"  
},
```

# 偵錯

## 指令列偵錯

```
$ node debug #{filename}
```

# 偵錯

## node-inspector

```
//安裝  
$ npm install -g node-inspector
```

```
//先執行程式，debug-brk意思是停在第一行  
$ node --debug-brk #{filename}
```

```
//執行node-inspector  
$ node-inspector
```

● <http://localhost:8080/debug?port=5858>

# 全域物件

- process
- console
- 我們寫的所有變數都是在一個模組裡面，不會成為全域物件

# 全域物件 process

- `process.argv` 指令參數
- `process.stdout` 標準輸出
- `process.stdin` 標準輸入
- `process.nextTick(callback)`

# nextTick

```
//nexttick.js

var sleep = require('sleep');

function countDown(from) {

    if (!from || parseInt(from) < 0) {
        return;
    }
    sleep.sleep(1);
    console.log("Count down:" + from)
    process.nextTick(function(){countDown(from - 1)});
}

function randomlog() {

    var calls = Math.ceil(Math.random()*10);
    console.log(calls);
    process.nextTick(randomlog);
}
countDown(10);
randomlog();
```

💡 拿掉nexttick會怎樣？

# 全域變數 console

- console.log
- console.dir
- console.error
- console.trace

# 工具函數 Util

- util.inherits
- util.inspect === console.dir
- util.log: console.log + 時間

# 事件 events

- `on(event, listener)`
- `once(event, listener)`
- `emit(event, args...)`
- `removeListener(event, listener)`
- `removeAllListeners([events])`
- `emit('error')`

# 事件 events

```
var events = require('events');
var emitter = new events.EventEmitter();

emitter.on('someEvent', function(arg1, arg2) {
    console.log('listener1', arg1, arg2);
})

emitter.on('someEvent', function(arg1, arg2) {
    console.log('listener2', arg1, arg2);
})

emitter.emit('someEvent', 'A', 123);
```

- ✿ 練習1：把nextTick的countdown改成event呼叫
- ✿ 練習2：試著emit error

# 檔案系統 fs

- 所有操作都有同步與非同步版本
  - ex: `fs.readFile()` / `fs.readFileSync()`
- `fs.readFile` / `writeFile`
- `fs.unlink`, `fs.stat`
- `fs.mkdir` / `fs.rmdir`
- 練習： 寫一函式可讀取目錄下之 `input.txt(utf-8)`，並將其內容字串全部大寫化

# 檔案系統 fs

```
//readfile.js

var fs = require('fs');

fs.readFile('input.txt', 'utf-8', function(err, data) {
  if (err) {
    console.error(err);
  } else {
    console.log(data);
  }
});
```

- 練習： 寫一函式可讀取目錄下之 input.txt(utf-8)，並將其內容字串全部大寫化

# The WEB - HTTP

- 最廣泛使用的網路傳輸協定
  - HTTP request:
    - METHODS: GET, POST, PUT, DELETE, TRACE, CONNECT, OPTIONS, HEAD
  - HTTP response:
    - Response code: 1xx, 2xx, 3xx, 4xx, 5xx
- 用於RESTful

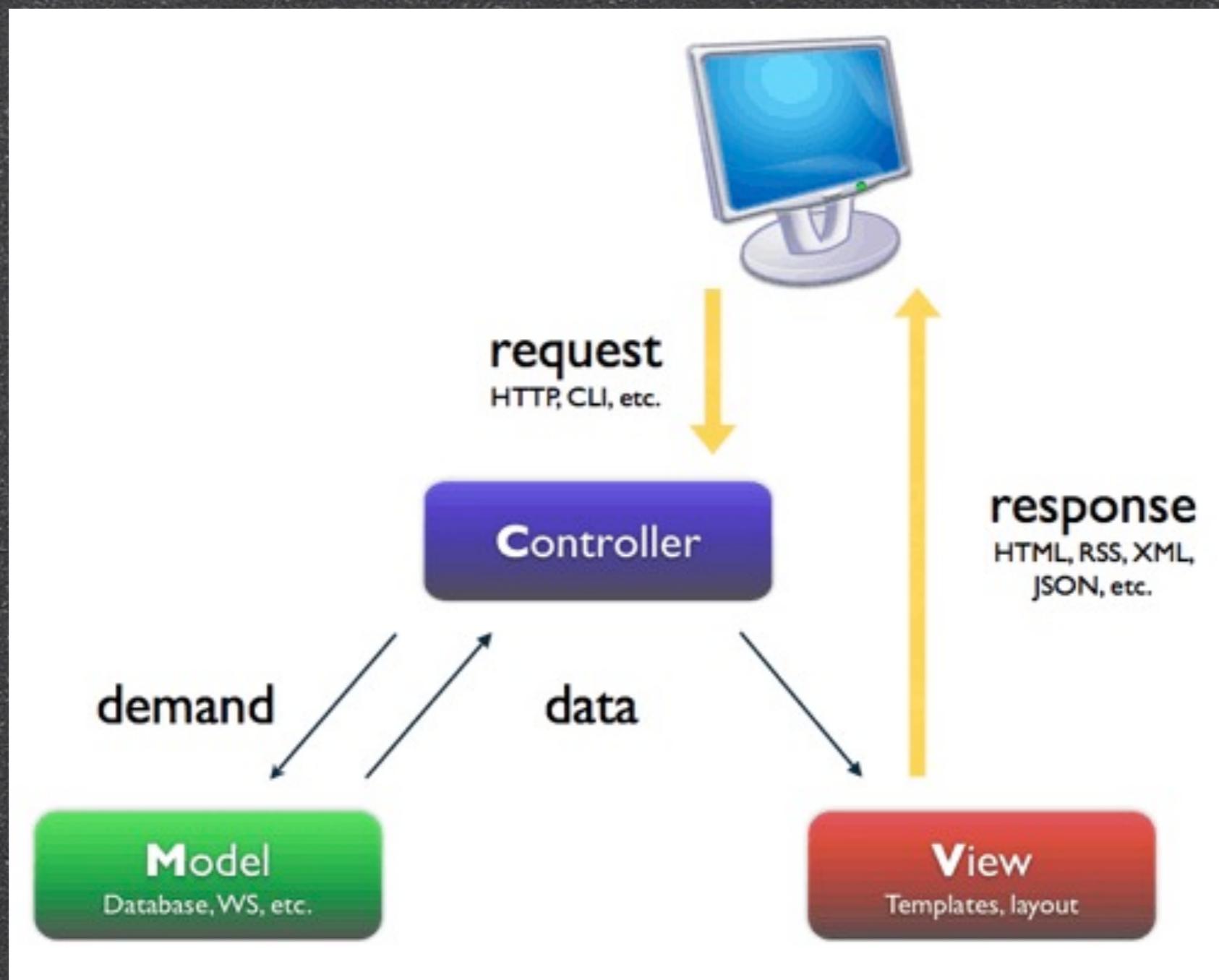
# Express

- Node.js的http核心模組過於底層，express較適合應用服務開發者
- MVC架構
- 提供指令用於快速建立專案

```
$ npm install -g express  
$ express --help
```

```
$ express messageBoard
```

# MVC



# MVC

- Model: models/
- View: views/
- Controller: routes/

# Express

## • 基本目錄結構

```
project root/  
  routes/  
  public/  
  views/  
  app.js  
  package.json
```

# Express

## ● 常用目錄結構

```
project root/
config/
routes/
doc/
models/
public/
  js/
  css/
  img/
test/
views/
app.js
package.json
README.md
```

# Express

- app.js
  - 程式進入點
  - 設定express: app.set
    - app.get('env')獲取環境變數NODE\_ENV
  - 加載功能: app.use
  - 設定路由: routes
  - 啓動伺服器

# Express

- app.use
  - favicon
  - logger
  - bodyParser
  - methodOverride
- router
- static
- errorHandler

# 路由 Router

- app.all / app.get / app.post /  
app.put / app.[ "delete" ]

```
app.all('*', ensureAuthenticated, loadUser);

ensureAuthenticated = function (req, res, next) {
  if ( req.isAuthenticated() ) {
    return next();
  }
  res.redirect "/logout";
}
```

# Request

- request.params
- request.query

```
//app.js  
app.get('/users/:name', user.show);
```

```
//user.js  
exports.show = function(req, res) {  
    res.send("User:" + req.params.name + "\n" + req.query);  
};
```

# Request

- request.body

```
//app.js  
  
app.post('/users', user.create);
```

```
//user.js  
  
exports.create = function(req, res) {  
  var body = req.body;  
  res.send("Creating user:" + body.name + ", email:" + body.email +  
  ", password:" + body.password);  
};
```

# Response

- res.status
- res.set / res.get
- res.cookie, clearcookie

```
res.cookie('rememberme', '1', { maxAge: 900000, httpOnly: true })
```

- res.redirect([status], url)
- status default: 302

# Response

- `res.send([status,] body)`
  - 會依照輸入body決定content-type
    - 字串 : `text/html`
    - 陣列/物件 : `application/json`
- `res.type(type) = res.set({'Content-type': type})`
- `res.render(view, [model, callback])`

# Views

- Template engine: Jade
  - <http://jade-lang.com/>
- 用以快速建立html頁面
- 優點：最小化要輸入的資料，不需要打一堆< > /
- 缺點：破壞html基本格式

# Views

- res.render

```
//app.js

app.set('views', __dirname + '/views');
app.set('view engine', 'jade');
```

```
//app.js

exports.index = function(req, res){
  res.render('index', { title: 'Express' });
};
```

# Views

• res.render

```
//index.jade
extends layout
block content
h1= title
p Welcome to #{title}
```

```
//layout.jade
doctype 5
html
head
  title= title
  link(rel='stylesheet', href='/stylesheets/style.css')
body
  block content
```

# 留言板開發

- 開發一個留言板
- 功能：
  - 使用者輸入 id 登入
  - 所有人都可以看到全部留言
  - 登入後能看到自己的留言

# 留言板 - 功能規劃

- 首頁
- 登入id
- 所有留言
- 自己的留言
- 登出

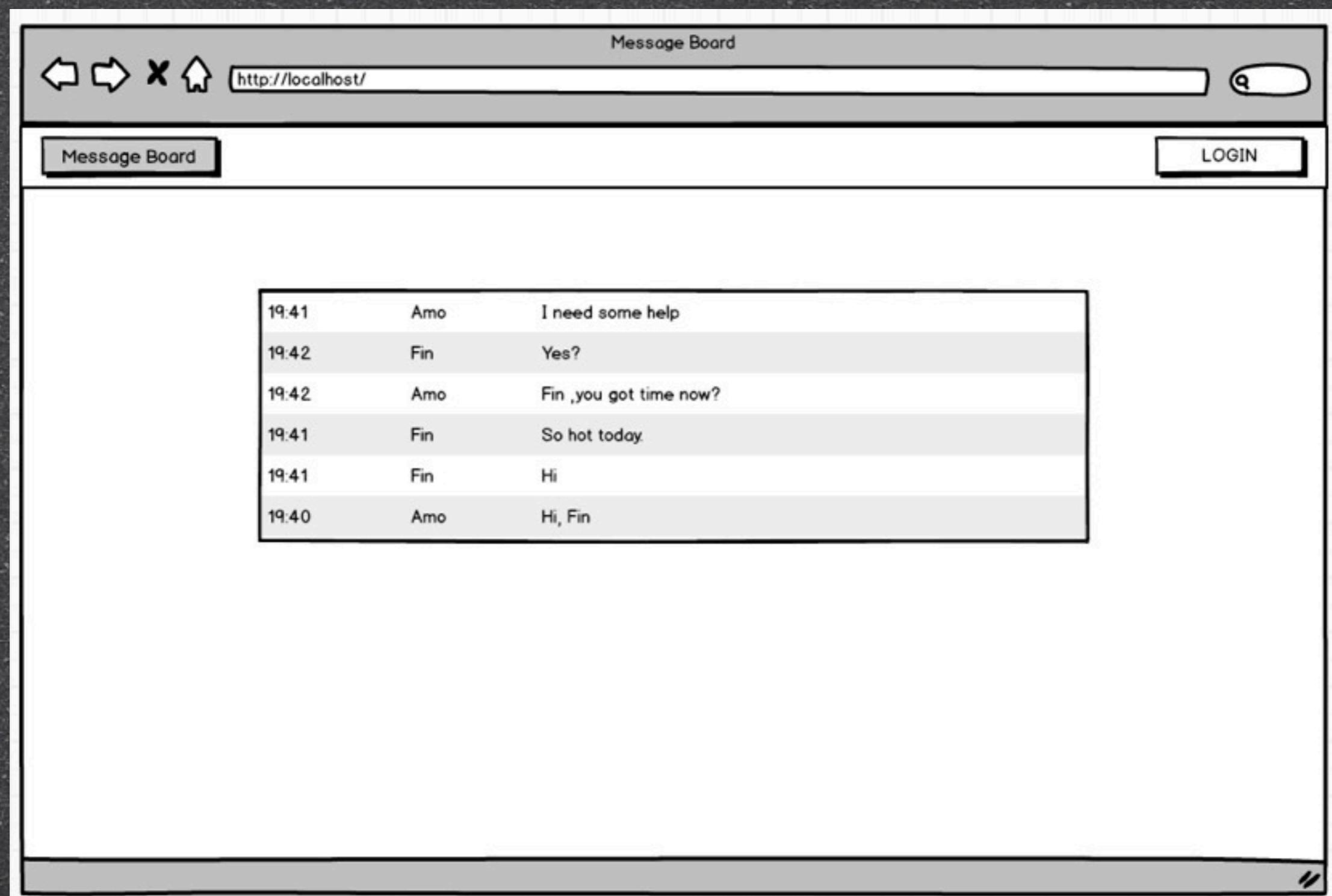
# 留言板 - 模型

- 留言
- 使用者
- 留言文字
- 時間
- 使用者不需模型，只需把id存在session

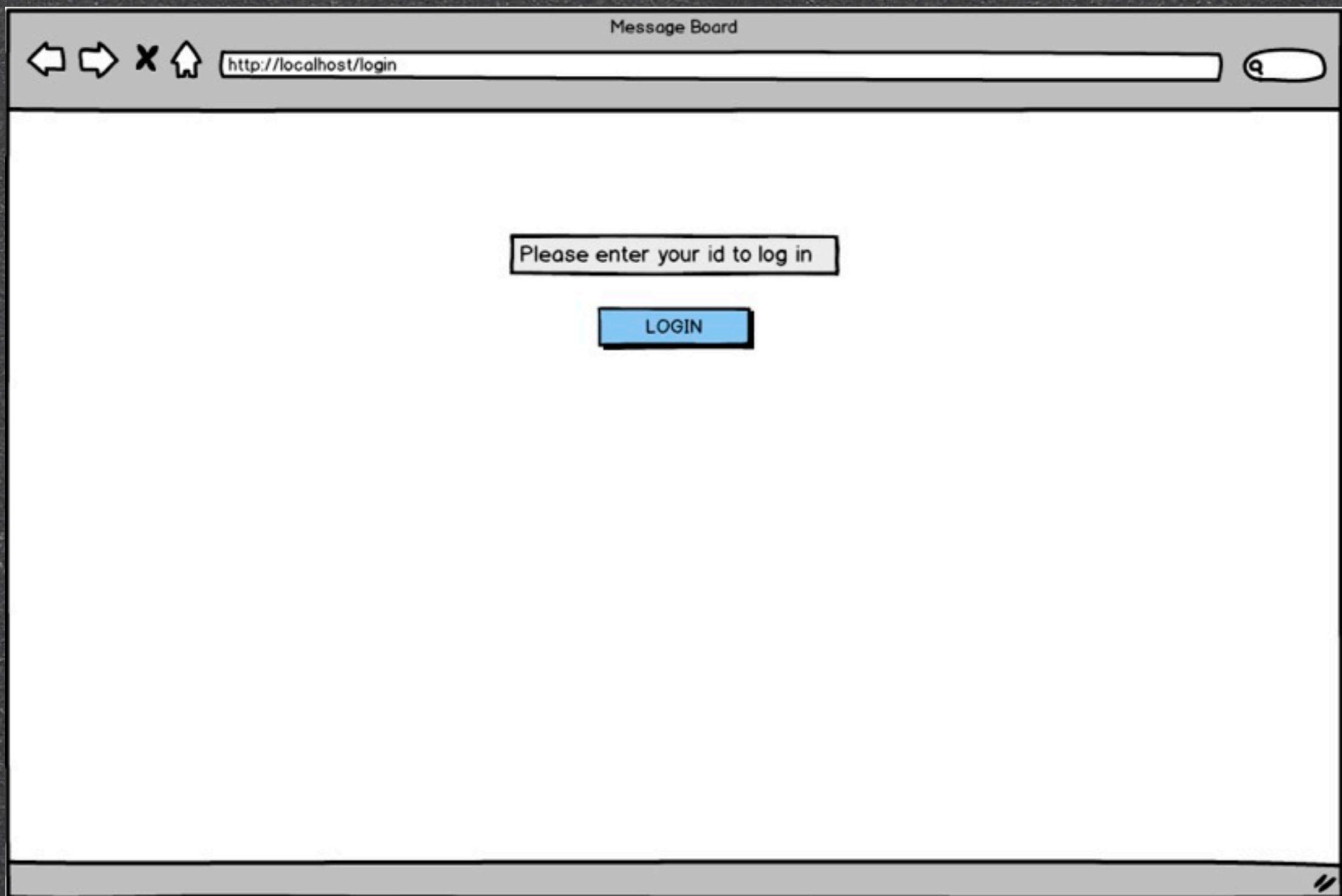
# 留言板 - 路由規劃

- POST /login
- GET /logout
- GET /
- GET /mymessage
- POST /mymessage
- DELETE /message/:id

# 留言板 - 登入前首頁



# 留言板 - 登入



# 留言板 - 登入後首頁

The screenshot shows a web browser window with the title "Message Board". The address bar displays "http://localhost/" and includes standard navigation icons (back, forward, search, etc.). The main content area has a header with "Message Board" on the left and "My Messages" and "LOGOUT" on the right. Below this is a form with a text input field containing "Enter your message here..." and a blue "Post" button. A scrollable list of messages follows:

19:41	Amo	I need some help
19:42	Fin	Yes?
19:42	Amo	Fin ,you got time now?
19:41	Fin	So hot today.
19:41	Fin	Hi
19:40	Amo	Hi, Fin

# 留言板 - 登入後首頁

Message Board

http://localhost/mymessage

Message Board      My Messages      LOGOUT

Enter your message here...

19:42	Fin	Yes?
19:41	Fin	So hot today.
19:41	Fin	Hi

# 留言板 - 留言

- instance method/variable:

- save: 新增留言
- name: 名字
- text: 留言
- date: 日期

- class method/variable:

- remove: 刪除留言
- myMessage: 找出自己的留言
- messages: 所有留言

# 留言板

- week2/messageBoard
- 完成所需的功能：
  - 我的留言 / 刪除留言
  - 登出
  - 其他未完成的方法

# Unit Testing

- mocha + chai

```
require("chai").should();

describe('Array', function(){
  beforeEach(function(done){
    //test setup
  });

  describe('#indexOf()', function(){
    it('should return -1 when the value is not present', function(){
      (-1).should.equal([1,2,3].indexOf(5));
      (-1).should.equal([1,2,3].indexOf(0));
    });

    //more tests
  })
})
})
```

# Unit Testing

- mocha + chai

```
$ npm install -g mocha  
$ mocha --recursive ${testDir}
```

- 練習：幫messageBoard加上unit test

# 控制流 flow control

- 多層callback有可能會造成這樣的程式碼

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

fs.mkdir('./hello', 0777, function(err){
  if (err) throw err;

  fs.writeFile('./hello/world.txt', 'Hello!', function(err){
    if (err) throw err;
    console.log('File created with contents: ');

    fs.readFile('./hello/world.txt', 'UTF-8', function(err, data){
      if (err) throw err;
      console.log(data);
    });
  });
});
```

- 常見發生情況： 需要多個非同步結果(ex: I/O)  
才能執行下一步時

# 控制流 flow control

## 解決方式1：拆分為不同function

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

fs.mkdir('./hello', 0777, makeDirectory);

function makeDirectory(err){
  if (err) throw err;
  fs.writeFile('./hello/world.txt', 'Hello!', writeFile);
}

function writeFile(err){
  if (err) throw err;
  console.log('File created with contents: ');
  fs.readFile('./hello/world.txt', 'UTF-8', readFile);
}

function readFile(err, data){
  if (err) throw err;
  console.log(data);
}
```

## 解決方式2：async套件

```
var fs = require('fs');
var async = require('async');

async.series([
    function(callback) {
        fs.mkdir('./hello', 0777, function(err) {
            if (err) {
                callback(err);
            }
            callback(null, true);
        });
    },
    function(callback) {
        fs.writeFile('./hello/world.txt', 'Hello!', function(err) {
            if (err) {
                callback(err);
            }
            console.log('File created with contents: ');
            callback(null, true);
        });
    },
    function(callback) {
        fs.readFile('./hello/world.txt', 'UTF-8', function(err, data) {
            if (err) {
                callback(err);
            }
            console.log(data);
            callback(null, true);
        });
    }
], function(err, results) {
    if(err) {
        console.log("Error occurred:", err);
    }
    console.log("operation completed!");
});
```

# Async

- `async.each(arr, iterator, callback)`
- `async.series(tasks, [callback])`
- `async.parallel(tasks, [callback])`

# Async

- 練習：寫一函式，會先判斷檔案是否存在 `fs.exists`，之後讀取檔案大小 `fs.stat`，再讀取檔案內容一併印出 `fs.readFile`
- 不用 `async` 怎麼寫
- 使用 `async` 怎麼寫

# 參考資料

- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>