學了什麼:

NodeJS(fs,http,os,url,require()) module.exports server.listen app.listen

setHeader(Content-Type setchar=utf-8) response.end

json package.json

cd /d xxx

express.js express.Router app.use app.get app.post query(get),body(post)

body-parser

art-template

callback Array.find() findIndex()

MongoDB

mongoose crud

WEB後台服務端

1. java
2. php
3. python
4. ruby
5. .net c#
6. …
7. Node.js 是用Javascript進行編程

JSON（JavaScript Object Notation, /ˈdʒeɪsən/）是由道格拉斯·克羅克福特構想和設計的一種輕量級資料交換格式。

NodeJS不是一門語言,不是庫,不是框架。

NodeJS是一個javascript運行時環境(runtime)。

NodeJS可以解析與執行js代碼。

以前只有瀏覽器可以解析js代碼 , 現在js可以完全脫離瀏覽器來運行。

一切都是由於NodeJS的出現。

瀏覽器中的javascript:

Ecmascript(if,var,function,object,array…)

DOM

BOM

NodeJS中的javascript:

沒有DOM,沒有BOM。(服務端不操作頁面)

有Ecmascript

NodeJS執行環境中,提供了一些服務器級別的操作API。

例如:

文件讀寫

網絡服務構建

網絡通信

http 服務器

等…

NodeJS利用

event-driver 事件驅動,

non-blocking I/O model 非阻塞IO模型(異步)

使NodeJS輕量與高效。

NodeJS package npm ecosystem npm生態系統

npm 包管理工具

npm is the largest ecosystem of open source libraries in the world.

npm世界上最大的開源庫生態系統

大多數javascript相關的包都有放在npm上。

eg. npm install jquery

NodeJS 是構建於chrome 的v8引擎之上

代碼只是具有特定格式的字符串而已

引擎可以幫我們解析與執行代碼。

google chrome 的v8引擎是現在公認 解析執行javascript代碼最快的引擎

node.js的作者把google chrome的v8引擎移植了出來, 開發了一個獨立的javascript運行時環境(runtime)。

1.3 Node.JS能做什麼

web服務器後台

命令行工具(npm(node) git(c語言開發的) hexo(node))

接口服務器

遊戲服務器

對於frontend programmer來說,主要是用它的命令行工具

webpack , gulp, npm

最好知道簡單的命令行操作

cd

cd的全称是Change Directory，直译为改变文件夹，也就是跳转目录、切换路径的意思。它后面可以接驱动器符号、完整路径和相对路径。

1.跳转到当前驱动器的根目录

cd [当前驱动器盘符]:/    例如： cd c:/

要保证当前目录修改成功，必须加/d参数，即cd /d c:/temp

dir

<https://www.howtoing.com/linux-dir-command>

dir （directory，目錄）是一個條列檔案及目錄的命令行界面指令，

ls

**1. 命令格式：**

ls [选项] [目录名]

**2. 命令功能：**

列出目标目录中所有的子目录和文件。

mkdir

mkdir 是 make directory 的意思。在自己系统中，在 mkdir 命令帮助下，可随意创建自己想要的新目录。命令行中键入 mkdir \<dir name> ，其中 \<dir name> 替换成你想创建的目录名。

rm

1.6這門課會學到什麼?

1.B/S 編程模型: (browser-server , backend, 任何服務端技術 這種BS編程模型都是一樣的, 和語言無關, Node只是作為我們學習BS編程模型的一個工具而已)

2.模塊化編程:

RequireJS

SeaJS

`@import(‘文件路徑’)`

以前認知的javascript只能通過script標簽來加載

在Node中可以像@import() 一樣來引用加載Javascript腳本文件

3.NODE常用api

4.異步編程:

回調函數 callback

Promise

async

generator

5.Express Web開發框架

6.EMCA6 (es6)

7.學習node不單單幫我們打開服務端的黑盒子, 同時會幫助我們學習到以後的前端高級內容,會幫助到學習Vue.js React angular

3.1 核心模塊



fs.readFile

fs.writeFile

fs.unlink (刪file)

fs.mkdir make directory

fs.readdir read directory

OS operation system

08-模塊的加載與導出

node中只有模塊作用域,沒有全局作用域。

如果nodeJS想要實現模塊之間的通信, 就要用exports。

a.js文件中:

var b = require('./b');

console.log(b.add(10, 20));

b.js文件中:

exports.add = function(*x*, *y*) {

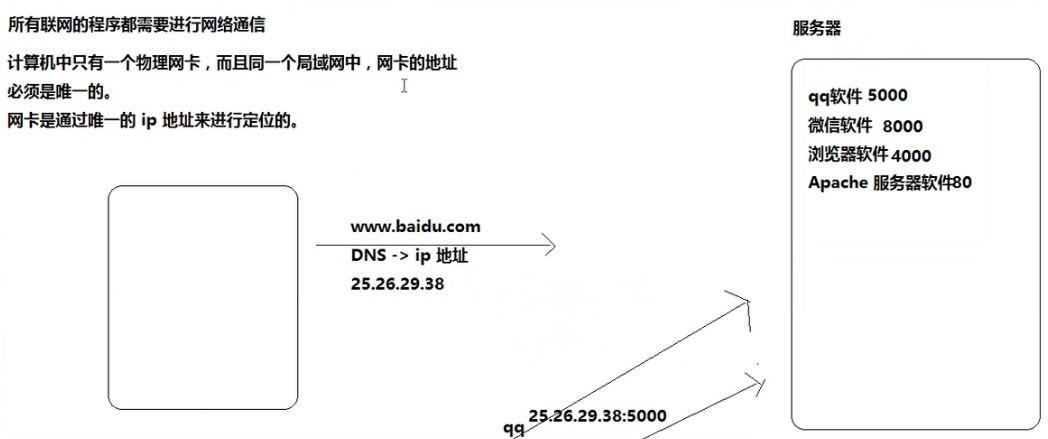
    return *x* + *y*;

}

用戶自定義模塊:

require , exports。

ip地址和端口號的概念



ip地址用來定位計算機(服務器)

端口號用來定位具體的應用程序

\*所有需要聯網通信的軟件都必須具有端口號



端口號

TCP與UDP段結構中[端口地址](https://baike.baidu.com/item/%E7%AB%AF%E5%8F%A3%E5%9C%B0%E5%9D%80)都是16比特，可以有在0---65535范圍內的端口號。對於這65536個端口號有以下的使用規定[6]  ：

（1）端口號小於256的定義為常用端口，服務器一般都是通過常用端口號來識別的。任何TCP/IP實現所提供的服務都用1---1023之間的端口號，是由ICANN來管理的；端口號從1024---49151是被注冊的端口，也成為“用戶端口”，被IANA指定為特殊服務使用[6]  ；

（2）[客戶端](https://baike.baidu.com/item/%E5%AE%A2%E6%88%B7%E7%AB%AF)只需保證該端口號在本機上是唯一的就可以了。[客戶端](https://baike.baidu.com/item/%E5%AE%A2%E6%88%B7%E7%AB%AF/101081)端口號因存在時間很短暫又稱臨時端口號[6]  ；

（3）大多數TCP/IP實現給臨時端口號分配1024---5000之間的端口號。大於5000的端口號是為其他服務器預留的[6]  。

可以同時開多個服務器, 但不可以用同一個端口號。

10-http.js

response.setHeader()

響應內容類型 Response Headers: Content-Type

Content-Type text/plain; charset=utf-8

server.on('request', function(*req*, *res*) {

*// 在服務端默認發送的數據, 其實是 utf8 編碼的內容*

*// 但是瀏覽器不知道是 utf8 編碼的內容*

*// 瀏覽器在不知道服務器響應內容的編碼的情況下, 會按照當前操作系統的默認編碼去解析*

*// 中文操作系統默認是 gbk編碼*

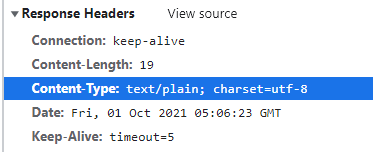
*// 解決方法: 告訴瀏覽器 我們發給你的內容是utf8編碼*

*// 利用respone.setHeader('name','value')*

*res*.setHeader('Content-Type', 'text/plain; charset=utf-8');

*res*.end('here is這裡是: ' + *req*.url);

});



text/plain 普通文本

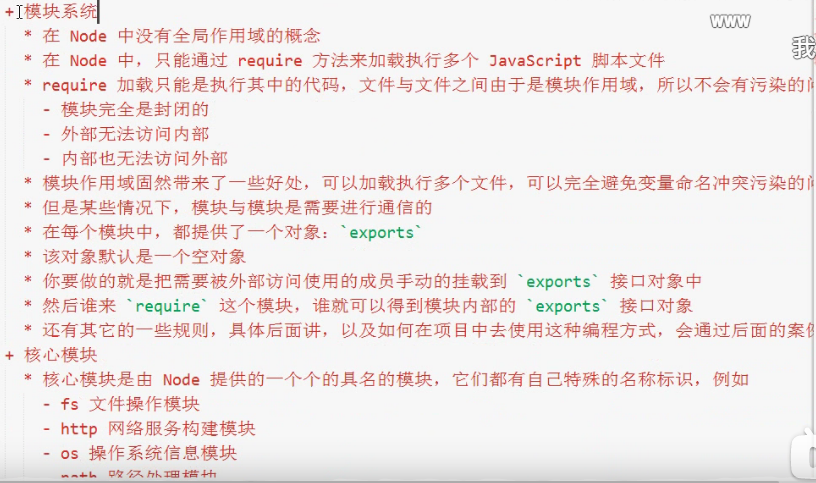
text/html 超文本

不同的資源用不同的Content-Type

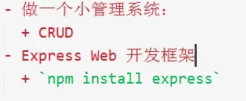
jpg > image/jpeg

圖片不用改編碼,

字符數據就常常要改編碼。

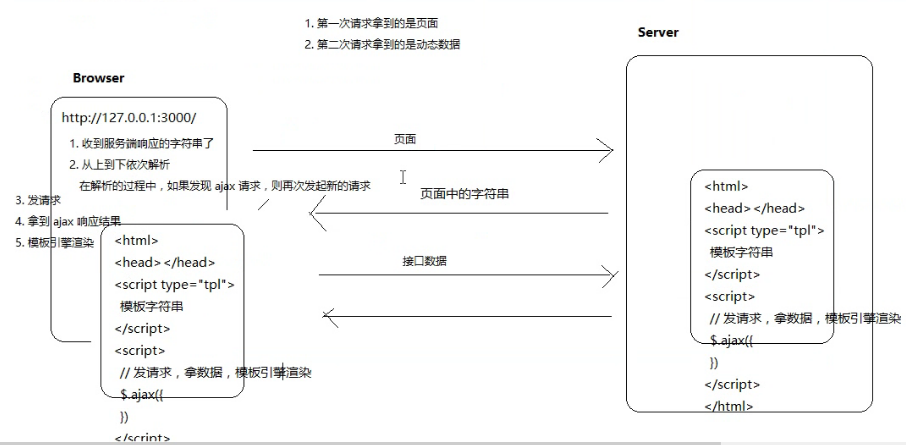


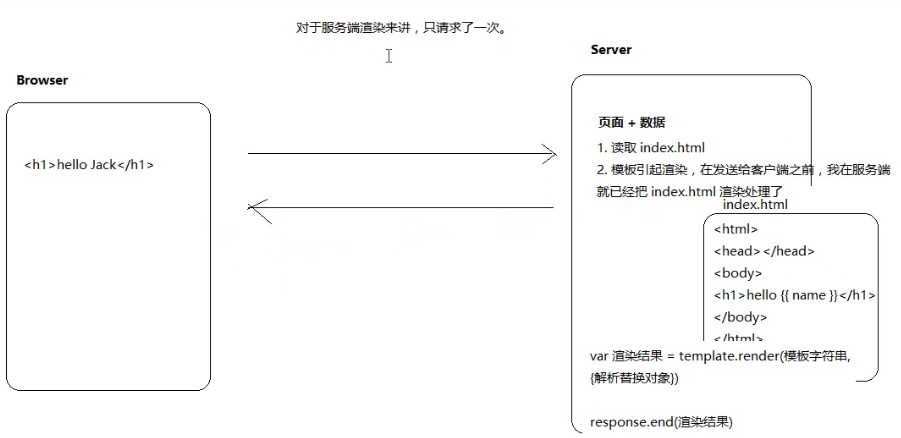




服務端渲染: 在服務端使用模板引擎 (如同: 17-在apache案例中加入模板引擎.js) 我們現在寫的node.js就是服務端,browser是客戶端

1客戶端渲染(第二次請求)



2服務端渲染

第一次請求是,我們輸入網址,向服務端取得html內容,渲染出網站。

在瀏覽器中查看網站的源代碼。

如果能找到,就是服務端渲染。(下一頁需要重新再load一次,向服務端請求)

如果不能找到,就是客戶端渲染(第二次請求取得動態數據)。

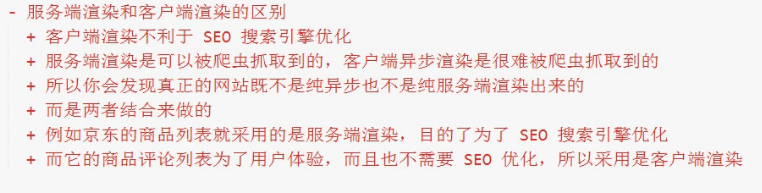
客戶端渲染(異步,ajax第二次請求)

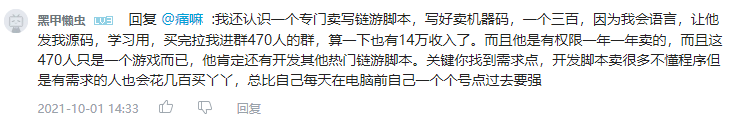
異步(ajax)的數據,SEO是找不到的。 這裡說的異步ajax 是客戶端向服務器請求服務器的API,取得模板引擎數據進行渲染(load得更快,不用等loading)。

客戶端渲染,不利於SEO優化。

https://www.jianshu.com/p/ebdc3ce41c6a

服務端渲染和客戶端渲染的區別





18-留言本

*\*\* 不要再想成文件路徑了, 把所有路徑都想象成 url 路徑 \*\**

*<!-- <link rel="stylesheet" href="../public/css/bootstrap.css"> 雖然也可以 ../在請求時會自動消失-->*

<link rel="stylesheet" href="/public/css/bootstrap.css">

*// \*\*\*\*\*由於訪問靜態資源的lnik會是127.0.0.1:3000/path,會訪問不了,所以我們要統一處理可以訪問的靜態資源\*\*\*\*\**

else if (url.indexOf('/public/') === 0) {

*// 統一處理public為可訪問資源*

*// var.indexOf('x') 是返回字符串x中首次出現位置索引*

*// 統一處理:*

*//  如果請求路徑是以 /public/ 開頭的, 則我認為你要獲取 public中的某個資源*

*//  所以我們就直接可以\*\*\*把請求路徑當作文件路徑來直接進行讀取\*\*\**

            fs.readFile('.' + url, function(*err*, *data*) {

                if (*err*) {

                    return *res*.end('reading public failed');

                }

*res*.end(*data*);

            });

什麼是根目錄？根目錄是指邏輯驅動器的最上一級目錄，也就是儲存子目錄的目錄，本地根目錄例如C盤、D盤等；網站根目錄指的是一個服務器上同時放置多個站點使用，或者一個大規模站點需要放置在幾個服務器上。“/”則表示根目錄，例如D盤下 images 文件夾裡名稱為“aaa.jpg”文件，書寫為：﹤img src="/images/aaa.jpg"﹥

302 Found，原始描述短語為Moved Temporarily，是HTTP協定中的一個狀態碼（Status Code）。 可以簡單的理解為該資源原本確實存在，但已經被臨時改變了位置；換而言之，就是請求的資源暫時駐留在不同的URI下，故而除非特別指定了快取頭部指示，該狀態碼不可快取。

*res*.statusCode = 302;

*res*.setHeader('Location', '/');

The **Location** response header indicates the URL to redirect a page to. It only provides a meaning when served with a 3xx (redirection) or 201 (created) status response.

response.redirect(‘重定向’);

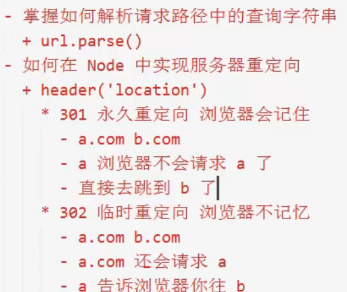
Express 第三方web開發框架

框架里的api 十分好用。





node比php apache更底層,可操控更多東西。



比如[www.sina.com會301](http://www.sina.com會301)永久重定向到[www.sina.com.cn](http://www.sina.com.cn)

301如果不是第一次訪問(會由disk cache)緩存中快取。 (可以經由清cookie來delete cache)。 所以就是(瀏覽器會記住,因為有緩存,下次會load更快)

20-Node中的模塊系統



23-require標識符分析

*\*\*\*第三方模塊加載規則\*\*\*:*

package.json的main

node\_modules 第三方模塊 第三方包 是如何加載的 加載規則

*// 第三方模塊*

*// 第三方模塊也是通過 包名 就可以加載了*

*// require('art-template');*

var template = require('art-template');

console.log(template);

*// \*\*\*第三方模塊加載規則\*\*\*:*

*// 加載過程規則: node\_modules/art-template/package.json文件中的main屬性*

*// node\_modules / 包名 / package.json >>  package.json內的 main屬性*

*// package.json文件內: "main": "index.js",*

*// 之後會跑去找index.js*

//

*// \*\*\*如果沒有package.json 或者 package.json 中的main屬性沒有值,*

*// node會自動去 node\_modules/包名/  目錄下找index.js加載*

*// 也就是說 index.js會作為一個默認備選項*

*// 如果以上所有任何一個條件都不成立, 則會進入../ 上一層目錄中找node\_modules查找*

*// 如果上一層也沒有,則會去找上上一層*

*// 直到當前磁碟的根目錄還找不到, 就會報錯:*

*// can not find module xxx*

*// node\_modules通常放在 項目的根目錄中, 這樣項目中所有的子項目都可以加載modules*

*// 不會出現有多個node\_modules*

模塊查找機制



可以看 <<深入淺出Node.js>>中的 模塊系統 章節

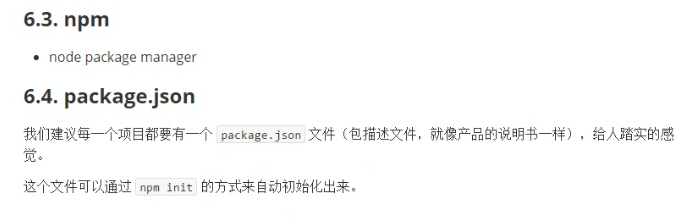
第三方包都是跑它們自己的index.js

在安裝npm時 加上--save 比如npm install --save art-template

這樣package.json就會有一個dependencies 說明項目是依賴什麼模塊。

package.json 包描述文件\*\*\*

npm init 可以生成package.json 描述這個項目的文件



如果有package.json ,可以直接npm install, 會自動安裝package.json下dependencies描述了的包。

node 常用命令

npm int (初始化, 生成package.json,可保存dependencies)

npm install (根據package.json 安裝包)

npm install 包名 (安裝包)

npm install 包名 --save (安裝包 生成package.json(dependencies有對應的包))

npm install 可簡寫 npm i

npm uninstall 包名 (如果有依賴dependencies項會依然保存)

npm uninstall 包名 --save (同時delete dependencies的保存)

npm help 查看使用幫助

npm install 包名 --registry = 網址 (轉到resgistry網址去安裝包)

Express

原生的http在某些方面表現不足以應對我們的開發需求.

所以就需要框架來加快我們的開發效率。

在node中,有很多Web開發框架,我們現在學Express!

npm install express --save

npm init --y 直接默認生成package.json

修改完代碼自動重啟服務器 nodemon

用第三方命令行工具: nodemon來幫我們解決 修改完代碼後 服務器重啟問題。

npm install --global nodemon

使用方法: nodemon 文件名

每次保存文件,就會重啟服務器。

24-Express基本 server

var app = express();

app.get('/', function(*req*, *res*) {

app.use('/public/', express.static('./public/'));

app.listen(3000, function() {

app.get() app.post();

路由router 就是一張表

app.get / 去哪

app/post /login 去哪

app.請求方法(請求路徑,請求處理函數)

Middleware中文翻譯為「中介軟體」，是指從發出HTTP請求後，到應用程式接收回應前，用來處理特定用途的程式。中介軟體可以說是一種封裝的功能。

*// app.use([path,] callback [, callback...])   path是url路徑*

*// express.static(root, [options])             root 是由app.js開始的路徑*

express.static(root, [options]);

The root argument specifies the root directory from which to serve static assets.

第一個root參數指定根目錄由哪裡去提供靜態資源。

The function determines the file to serve by combining req.url with the provided root directory.

結合 app.use的req.url與 提供出的root 根目錄路徑。

When a file is not found, instead of sending a 404 response, it instead calls next() to move on to the next middleware, allowing for stacking and fall-backs.

app.use([path ,] callback , [ callback…])

Mounts the specified [middleware](http://expressjs.com/guide/using-middleware.html) function or functions at the specified path: the middleware function is executed when the base of the requested path matches path.

if there is no [path] argument in app.use() then suited for (suitable for) any path I guess. (neko thinking 2021/10/11)

*// \*\*\*app.use結合 express.static 的意思是*

*// 只要路徑是[path] argument,*

*// 就放開+根據path訪問express.static內的root directory,*

*// 以root directory路徑開始的文件,*

*// 而且root directory不用再在path(reqest.url)上寫了*



推荐

app.use(‘/public/’,express.static(‘./public/’)); 這種寫法

request.query 獲取 get 請求體的數據

request.body 獲取 post請求體的數據 (需要第三方包body-parser)

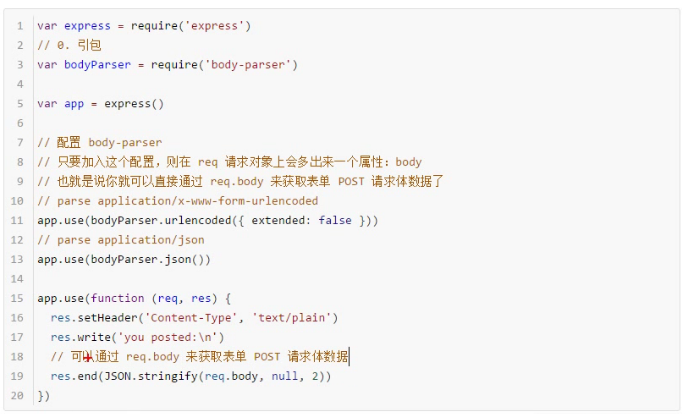
在express獲取表單post請求體數據

在express中沒有內置獲取post 請求體的api,

這里我們需要用第三方包 body-parser。

npm install --save body-parser

body-parser 使用 配置:



注意: 新寫法是以下:

app.use(express.urlencoded({ extended: false }))

app.use(express.json())

JSON: **J**ava**S**cript **O**bject **N**otation(JavaScript 对象表示法)

JSON 是存储和交换文本信息的语法，类似 XML。

JSON 比 XML 更小、更快，更易解析。

**JSON.parse()**

JSON 通常用于与服务端交换数据。

在接收服务器数据时一般是字符串。

我们可以使用 JSON.parse() 方法将数据转换为 JavaScript 对象。

**JSON.stringify()**

JSON 通常用于与服务端交换数据。

在向服务器发送数据时一般是字符串。

我们可以使用 JSON.stringify() 方法将 JavaScript 对象转换为字符串。



路由設計



路由導出與使用

var router = require('./router.js');

1.原理寫法

module.exports = function(*app*) {

*app*.get('/', function(*req*, *res*) {

};

*app*.get('/abc', function(*req*, *res*) {

};

}

2.express.Router() 寫法

*// 1.創建一個express路由容器*

var router = express.Router();

*// 2.把路由都掛載到 router 路由容器中*

router.get('/', function(*req*, *res*) {

};

router.get('/abc', function(*req*, *res*) {

};

*// 3.把router導出  (單一導出)*

module.exports = router;

students.js find()

var dbPath = './db.json'

*// 獲取所有學生列表*

*\*\*\*readFile是異步任務, 如果封裝函數,要用callback回調函數 獲取數據\*\*\**

module.exports.find = function(*callback*) {

    fs.readFile(dbPath, 'utf-8', function(*err*, *data*) {

            if (*err*) {

                return callback(*err*);

            }

*// JSON.parse(data).students;*

            callback(null, JSON.parse(*data*).students);

        }

    );

};

app.js

Student.find(function(*err*, *data*) {

        if (*err*) {

            return *res*.status(500).send('server error.');

        }

*res*.render('index.html', {

            fruits: [

                'apple',

                'banana',

                'orange',

                'melon'

            ],

            students: *data*,

        });

    });

es6 Array.find(callback) 返回條件對應數組成員的值

注意: *es6的find方法得出的結果是之前數據的引用，也就是淺復制，所以後期修改都會修改原來的數據對象*

var objStudent = db\_data.find(*item* => {

            return *item*.id === *studentID*

        });

es6 Array.findIndex()

*// es6 Array.findIndex(); 根據條件 返回索引*

db\_data.findIndex(function(*item*){

            return *item*.id === parseInt(*studentID*);

        });

# Array.prototype.splice()

**splice()** 方法可以藉由刪除既有元素並／或加入新元素來改變一個陣列的內容。

*array*.splice(*start index[*, *deleteCount[*, *item1[*, *item2[*, *...]]]]*)

\*\*\*\*\*

27-express-crud-system 重點 與 總結

重點:

callback 回調函數與 異步api, 異步任務的數據,只能通過回調函數獲取

Array.find(function(item){return condition});

Array.findIndex(function(item){return condition});

Array.splice(start index, [delete count] , [replace item1], [replce item2]);

get 請求數據 req.query 是對象Object類型

post 請求數據 req.body 是對象Object類型

var bodyParser = require('body-parser');

*// bodyParser配置*

*// parse application/x-www-form-urlencoded*

app.use(express.urlencoded({ extended: false }))

*// parse application/json*

app.use(express.json())

JSON.stringify(); 轉為字符串

JSON.parse(); 轉為javascript對象

parseInt();

fs.ReadFile的第二個參數可以轉編碼 相當於我們

data.toString();與res.setHeader(‘Content-Type’, ‘text/plain; charset=utf-8’ );

fs.readFile(Path,’uft-8’,function(err,data){…})

readFile的data是字符串,如果要操作 記得轉Object JSON.parse(data);

fs.writeFile(); 寫上去時要用String字符串,所以記得由object轉string

var new\_data = JSON.stringify(data);

fs.writeFile(Path,new\_data,function(err){});

自己寫的步驟:

npm 安裝 需要用到的第三方包

配置第三方包 代碼 與 加載

處理模板

配置開放靜態資源

配置模板引擎

路由設計 (提取路由模塊 exports一個路由函數,或使用框架Router();)

封裝異步api (業務操作) eg. student.js

具體實現

\*\*\*回調函數的思想\*\*\*



異步api 都會有一個回調函數

fs.readFile

fs.writeFile

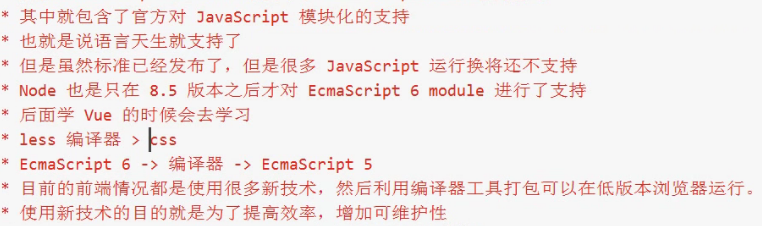
fs.readdir

setTimeout

ajax

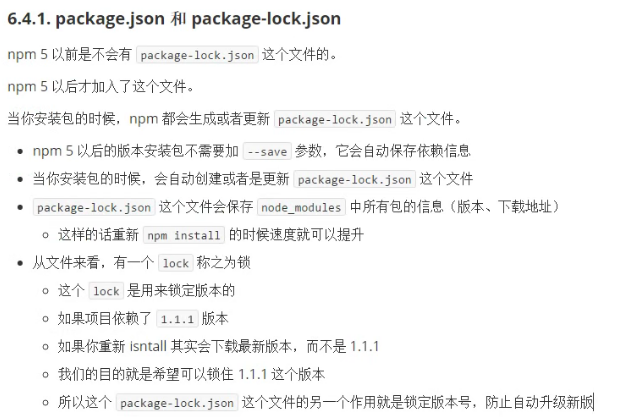
require.js(AMD)與sea.js(CMD)





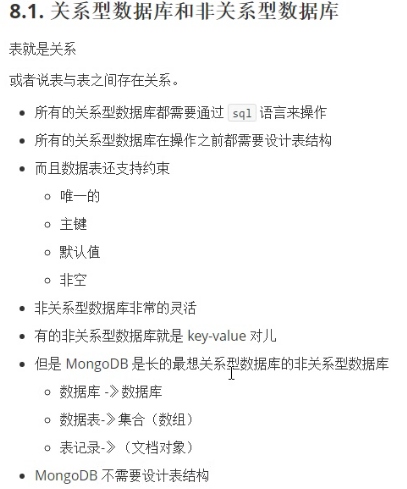
異步模塊定義（AMD）是Asynchronous Module Definition的縮寫，是 RequireJS 在推廣過程中對模塊定義的規范化產出。  
通用模塊定義（CMD）是Common Module Definition的縮寫，是SeaJS 在推廣過程中對模塊定義的規范化產出。

package.json 與 package-lock.json NodeJS 5.x.x version以後出現的



CommonJS  
2009年，美國程序員Ryan Dahl創造了node.js項目，將javascript語言用於服務器端編程。  
這標志"Javascript模塊化編程"正式誕生。因為老實說，在瀏覽器環境下，沒有模塊也不是特別大的問題，畢竟網頁程序的復雜性有限；  
但是在服務器端，一定要有模塊，與操作系統和其他應用程序互動，否則根本沒法編程。  
node.js的模塊系統，就是參照CommonJS規范實現的。在CommonJS中，有一個全局性方法require()，用於加載模塊。

MongoDB



set好 本機的環境變量, 可以使我們在cmd的C:/下也能使用mongod 的指令。

\*\*\*

如何啟動與關閉mongoDB數據庫

data/db

D:/ 路徑下 指令 mongod



用ADMIN開powershell 輸入net start mongod 也可以開啟數據庫

界面更好看。

連接和退出數據庫



1. 開啟數據庫
2. 連接數據庫

\*\*\*

mongoDB基本命令

命令在 連接mongoDB數據連的cmd下輸入()

1.show dbs

查看顯示數據庫

2.use 數據庫名稱

切換到指定想要操作的數據庫(如果沒有就會新建)

3.db

查看當前正在操作的數據庫

4.插入數據 db.集合名.insertOne()

eg: db.students.insertOne({“name”:”Jack”})

5.show collections

查看當前正在操作的數據庫 的集合

6.db.集合名.find()

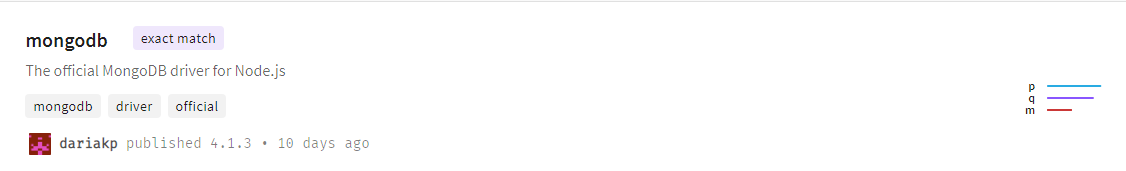
查看當前正在操作的數據庫的集合內所有的數據

**\*\*\*\*\***

**在Node.js中如何操作MongoDB數據**

1.使用官方的mongodb第三方包來操作





<https://github.com/mongodb/node-mongodb-native>

2.使用mongoose第三方包來操作

mongoose基於mongoDB包再一次封裝。

npm i mongoose

<https://mongoosejs.com/>

\*\*\*\*\*

MongoDB數據庫的基本概念:

MySQL的數據表 => 在mongoDB叫集合(集合是數組)

集合內的是文檔(文檔是對象)

數據庫=>集合(數組)=>文檔(對象)



*// 3.創建mongoose.model  (將文檔結構 發布為 模型)*

*// mongoose.model(modelName, schema);*

*// 也可以直接在參數內寫schema, 但現在我們上面寫好了schema*

*// \*\*\*第一個參數modelName*

*// modelName會自動將 大寫開頭的名稱 轉為 小寫複數(s) 的集合名*

*// 例如 Cat 會變做 cats, User 變做 users集合名*

Create 增 module.save();

Retrieve 查詢 .find({conditions})

const User = mongoose.model('User', userSchema);

moduleName.find()

User.find()

moduleName.find({條件}); 返回全部符合條件的文檔

moduleName.findOne({條件}); 返回第一個符合條件的文檔

delete 刪除數據 .remove({conditions}) .findByIdAndRemove({targetID},[callback])

User.remove({ username: 'peter444' }).then(*value* => {

    console.log('successfully removed data!');

    User.find().then(*value* => {

        console.log('刪除後數據如下:');

        console.log(*value*);

    })

}, *reason* => {

    console.log(*reason*);

});

Update 更改,更新數據 .findOneAndUpdate();

.findByIdAndUpdate({targetID},update content, [callback]);

*// 利用 model.findOneAndUpdate(condition,update,options,callback);*

User.findOneAndUpdate({ user: 'peter' }, { password: 'updatedpppppp' }).then(*value* => {

    console.log(*value*);

    console.log('update successed!');

    User.find().then(*value* => {

        console.log(*value*);

    });

}, *reason* => {

    console.log(*reason*);

});

Student.findByIdAndUpdate(*req*.body.id, *req*.body, function(*err*) {

es6 解析賦值

*// const { Schema } = mongoose;*

*// 相當於 var Schema = mongoose.Schema;*

*// 是es6 解析賦值  ES6 destructuring assignment*

*// 把mongoose對象內的mongoose.Schema解析成  變量名為 Schema*

*// 原寫法(沒有解析賦值下)*

*// const userSchema = new mongoose.Schema({*

mongoDB簡單總結



throw 拋出異常 1.可以中止程序執行, 2.error打印到控制台

多個異步同時執行, 無法保證代碼的順序

eg:文件比較大的可能會讀取比較慢(讀取完成比執行順序較後的異步任務慢), 即使在順序上它是比較早。

回調地獄可以解決這個順序問題。(callback內放另一個callback)

在回調函數callback內放下一個順序的回調函數。 就可以根據順序來出現讀取結果。



32-Promise (.then() 內 return)

*then內的return另一個Promise (p2)結果,*

*就可以在下一個then中接收到(p2)的resolve(value),reject(error)*

p1.then(*value* => {

    console.log(*value*);

    return p2;

}).catch(*err* => {

    console.log(*err*);

}).then(*value* => {

    console.log(*value*);

    return p3;

}).catch(*err* => {

    console.log(*err*);

}).then(*value* => {

    console.log(*value*);

}).catch(*err* => {

    console.log(*err*);

})

33-封裝異步api \*\*\*\*\*

*// 封裝異步function的話,記得是return new Promise() 是(function)*

*// 如果只是寫一個promise,就不用return     是(variable)*

function myFS(*path*) {

    return new Promise((*resolve*, *reject*) => {

        fs.readFile(*path*, 'utf-8', (*err*, *data*) => {

            if (*err*) {

                reject(*err*);

            }

            resolve(*data*);

        });

    });

};

34-Promise-usage-scenario

let readData = function(*path*) {

    return new Promise((*resolve*, *reject*) => {

        fs.readFile(*path*, 'utf-8', function(*err*, *data*) {

            if (*err*) {

                reject(*err*)

            } else {

                resolve(*data*)

            }

        });

    })

}

*// 這個readData(path) 在實用上可能是兩個不同path的file 所以會用到then內return (promise嵌套)*

Ajax

# 什麼是Ajax

**Ajax(Asynchronous JavaScript and XML) 異步JavaScript和XML**

Ajax實際上是下面這幾種技術的融合：

* (1)XHTML和CSS的基於標准的表示技術
* (2)DOM進行動態顯示和交互
* (3)XML和XSLT進行數據交換和處理
* (4)XMLHttpRequest進行異步數據檢索
* (5)Javascript將以上技術融合在一起

**客戶端與服務器，可以在【不必刷新整個瀏覽器】的情況下，與服務器進行異步通訊的技術**

# 為什麼我們需要Ajax？

在我們之前的開發，每當用戶向服務器發送請求，**哪怕只是需要更新一點點的局部內容，服務器都會將整個頁面進行刷新。**

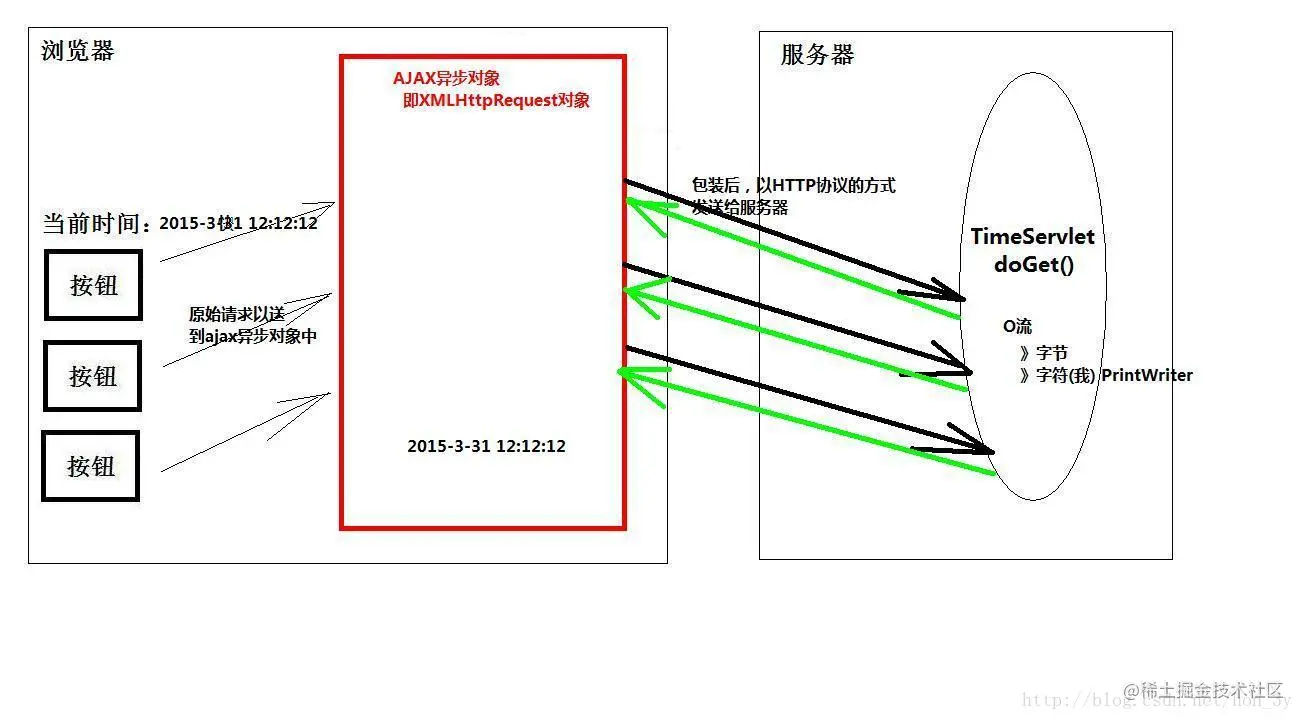
* **性能會有所降低(一點內容，刷新整個頁面！)**
* **用戶的操作頁面會中斷(整個頁面被刷新了)**

**Ajax就是能夠做到局部刷新**！

**XMLHttpRequest對像是Ajax中最重要的一個對象**。**使用Ajax更多的是編寫客戶端代碼**，而不是服務端的代碼。

\*\*\*\*\*

當我們使用AJAX之後，**瀏覽器是先把請求發送到XMLHttpRequest異步對象之中，異步對象對請求進行封裝，然後再與發送給服務器。服務器並不是以轉發的方式響應，而是以流的方式把數據返回給瀏覽器**

XMLHttpRequest異步對象會**不停監聽服務器狀態的變化，得到服務器返回的數據，就寫到瀏覽器上【因為不是轉發的方式，所以是無刷新就能夠獲取服務器端的數據】**  


## 創建XMLHttpRequest對象

var xhr = new XMLHttpRequest();

## XMLHttpRequest對象的屬性和方法

### ****方法****

* **open()**(**String method,String url,boolean asynch**,String username,String password)
* **send(content)**
* **setRequestHeader(String header,String value)**
* getAllResponseHeaders()
* getResponseHeader(String header)
* abort()

**常用的方法就是黑色粗體的前三個**

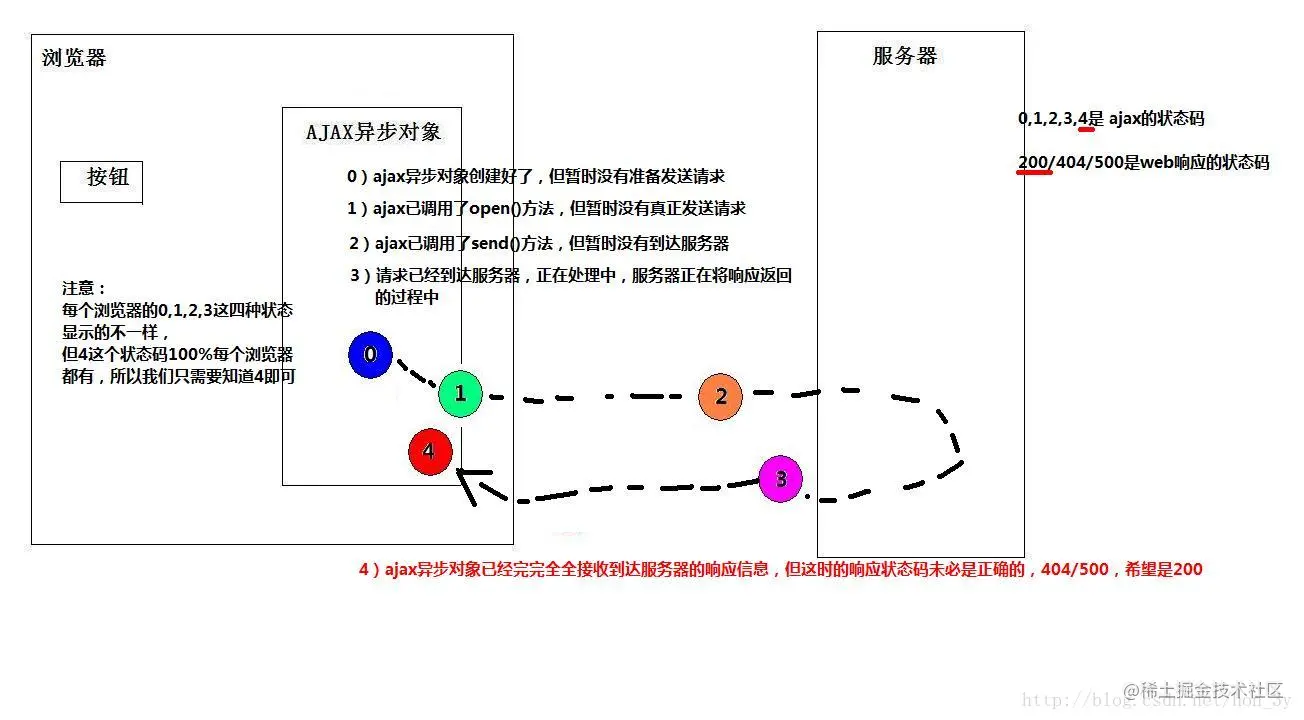
* **open()：該方法創建http請求**
  + **第一個參數是指定提交方式(post、get)**
  + **第二個參數是指定要提交的地址是哪**
  + **第三個參數是指定是異步還是同步(true表示異步，false表示同步)**
  + 第四和第五參數在http認證的時候會用到。是可選的
* **setRequestHeader(String header,String value)：設置消息頭（使用post方式才會使用到，get方法並不需要調用該方法）**
  + **xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");**
* **send(content)：發送請求給服務器**
  + **如果是get方式，並不需要填寫參數，或填寫null**
  + **如果是post方式，把要提交的參數寫上去**

### ****屬性****

 **onreadystatechange：請求狀態改變的事件觸發器（readyState變化時會調用此方法），一般用於指定回調函數**

 **readyState：請求狀態readyState一改變，回調函數被調用，它有5個狀態**

* 0：未初始化
* 1：open方法成功調用以後
* 2：服務器已經應答客戶端的請求
* 3：交互中。Http頭信息已經接收，響應數據尚未接收。
* **4：完成。數據接收完成**



* **responseText：服務器返回的文本內容**
* **responseXML：服務器返回的兼容DOM的XML內容**
* **status：服務器返回的狀態碼**
* statusText：服務器返回狀態碼的文本信息
* 上面有兩個地方都提及了回調函數，回調函數是什麼？？
* 回調函數就是**接收服務器返回的內容！**



# 編寫第一個Ajax程序

**檢測用戶輸入的用戶名是否為"zhongfucheng"，只要不是zhongfucheng，就可以使用！**

application/x-www-form-urlencoded:

<form encType=””>中默認的encType，form表單數據被編碼為key/value格式發送到服務器（表單默認的提交數據的格式）