Node.js 在後端程式的應用

報告者：洪子謙

HOME | ABOUT | DOWNLOADS | DOCS | FOUNDATION | GET INVOLVED | SECURITY | NEWS

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 releases, please update now!

Download for Windows (x64)

8.9.1 LTS
Recommended For Most Users

Other Downloads | Changelog | API Docs

9.2.0 Current
Latest Features

Other Downloads | Changelog | API Docs

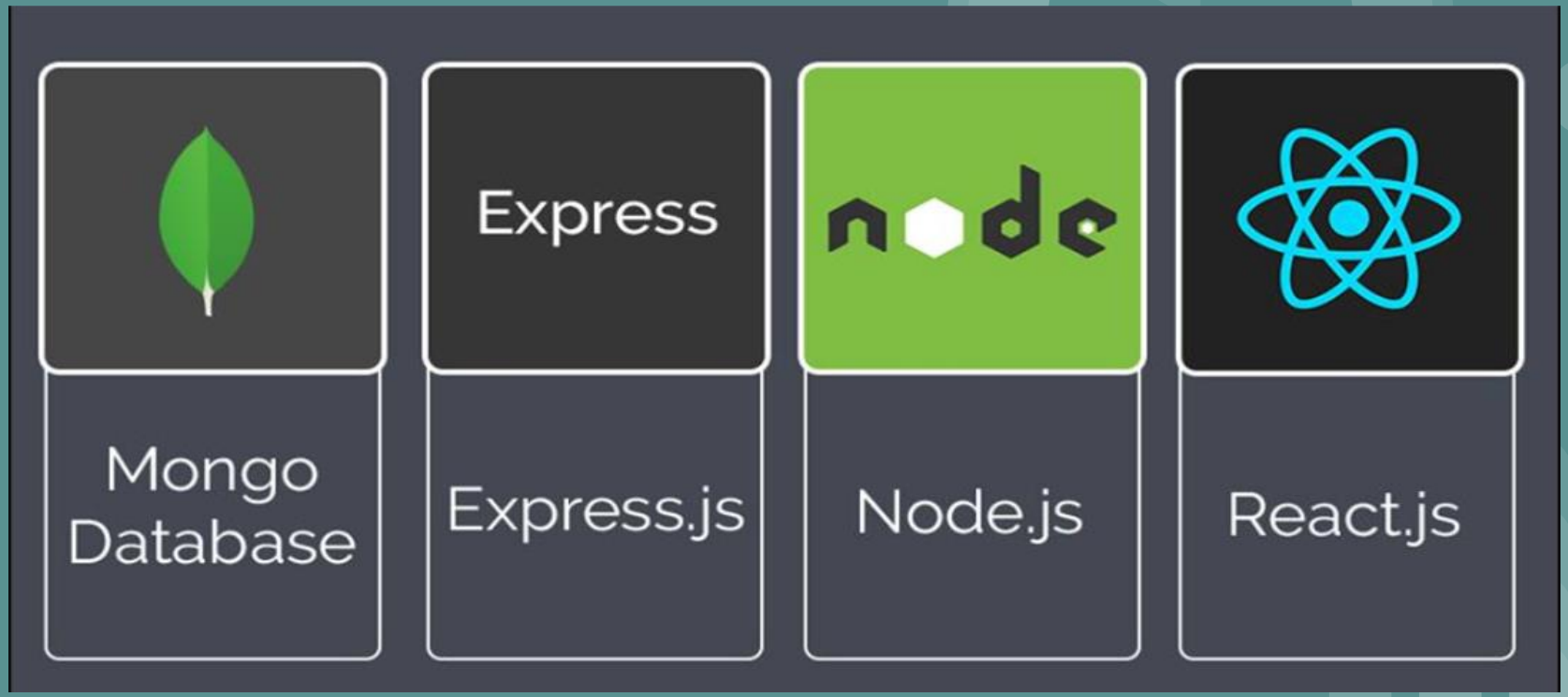
2009年Ryan Dahl建立Node.js。
Node.js使用C++語言編寫。
以V8 JavaScript Engine作為直譯器。
建置一個可執行JavaScript的環境。
事件驅動、非阻塞I/O的特性
可作為web Server（JavaScript）。
擁有可觀的open source 的套件。


JS
REQUIRE·JS
npm
TS
BABEL
JS
A
REQUIRE·JS
npm
TS
REQUIRE·JS

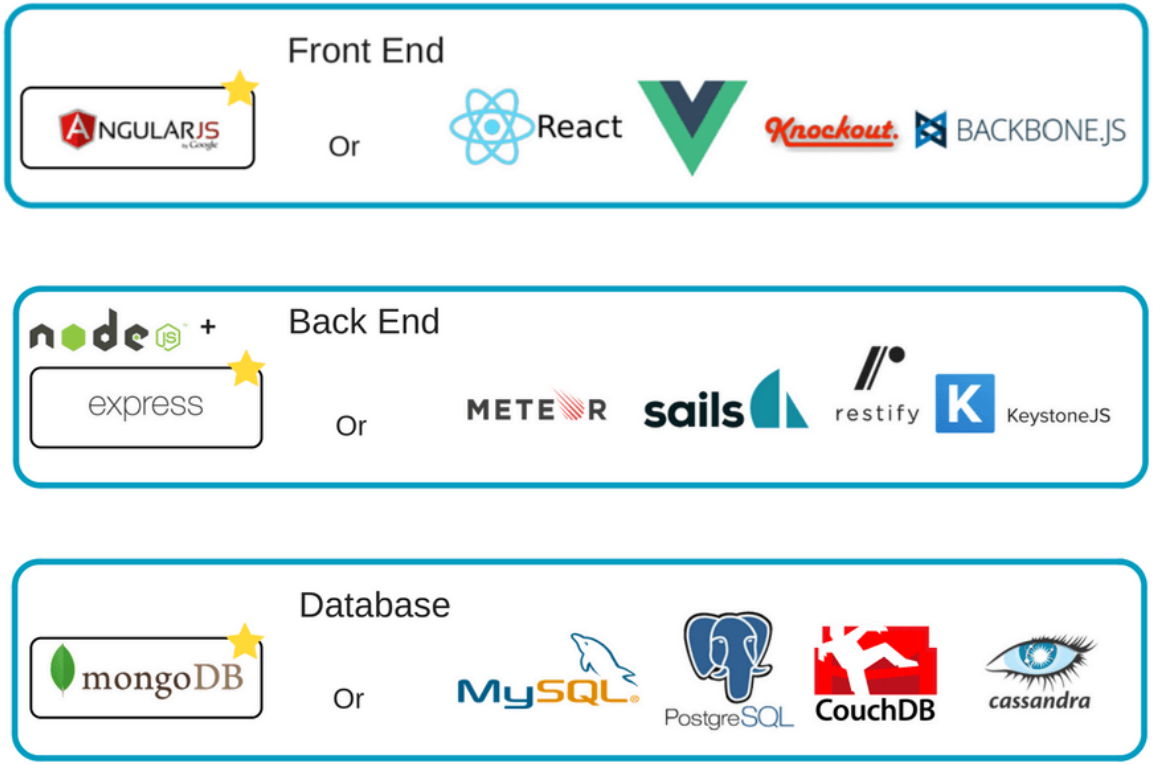
MEAN Stack


Express
Mongo DB
(database system)
Express
(back-end web framework)
Angular.js
(front-end framework)
Node.js
(back-end runtime environment)

MERN Stack



Full Stack JavaScript Tools and Technologies



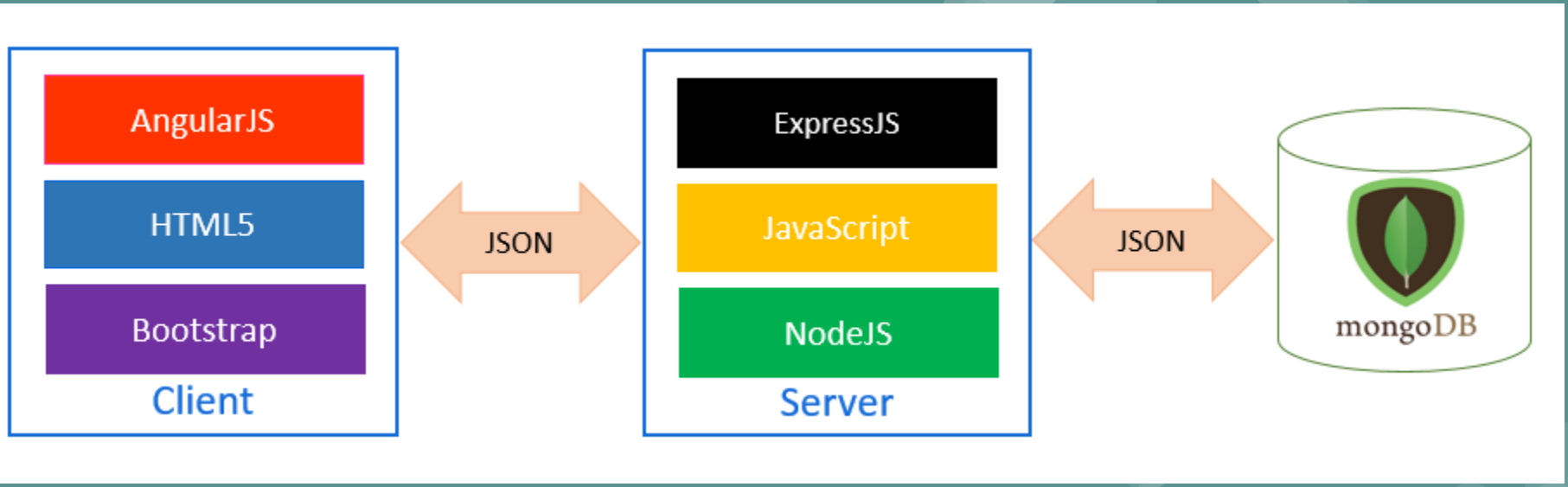

php

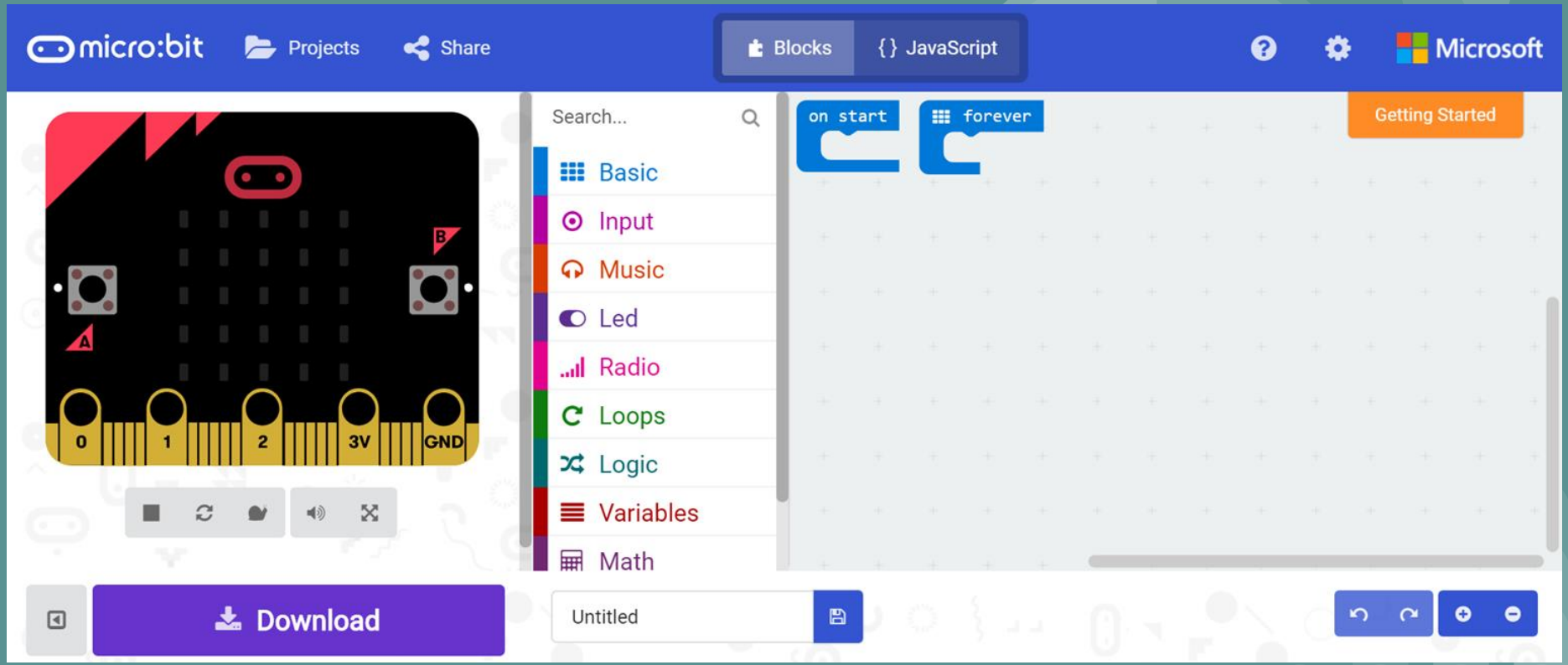

Laravel


MySQL

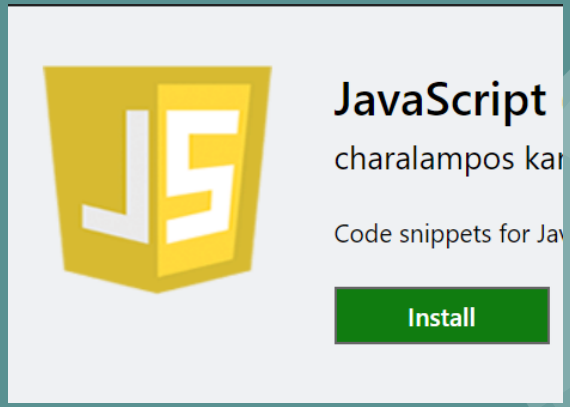

expressjs

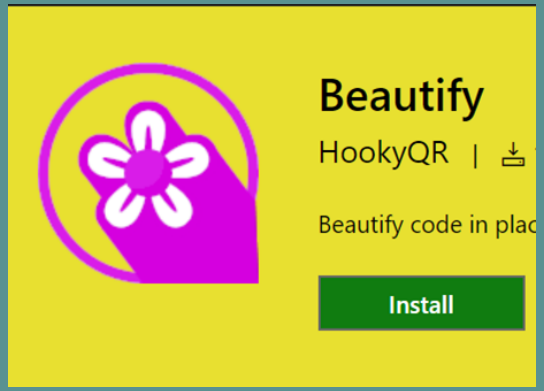

MySQL


AngularJS
HTML5
Bootstrap
Client
JSON
ExpressJS
JavaScript
NodeJS
Server
JSON
mongoDB

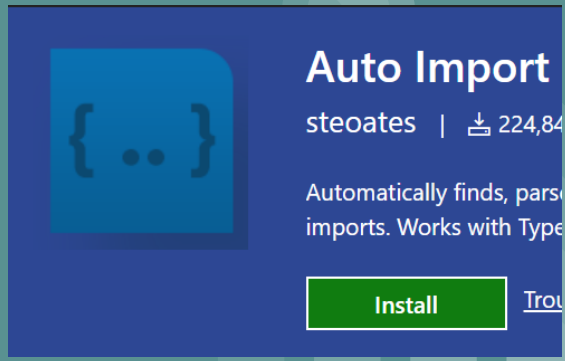

micro:bit
Projects
Share
Blocks
{} JavaScript
Microsoft
B
A
0
1
2
3V
GND
Search...
Basic
Input
Music
Led
Radio
Loops
Logic
Variables
Math
on start
forever
Getting Started
Download
Untitled

環境設定與安裝測試


JavaScript
charalampos kar
Code snippets for Jav
Install


Beautify
HookyQR |
Beautify code in plac
Install


Path Intellisense
Christian Kohler | 159,3
Visual Studio Code plugin that auto
Installation
Launch VS Code Quick Open (⌘+P
ext install path-intellise


Auto Import
steoates | 224,84
Automatically finds, pars
imports. Works with Type
Install
Trou

★ express public

express

mysql public

npm v2.15.0 downloads

★ morgan public

npm v1.9.0 downloads 5M/month

★ nodemon public

cors public

npm v2.8.4 downloads

★ body-parser public

npm v1.18.2 downloads 15M/month

moment public

gitter join chat

建立Node.js網頁伺服器

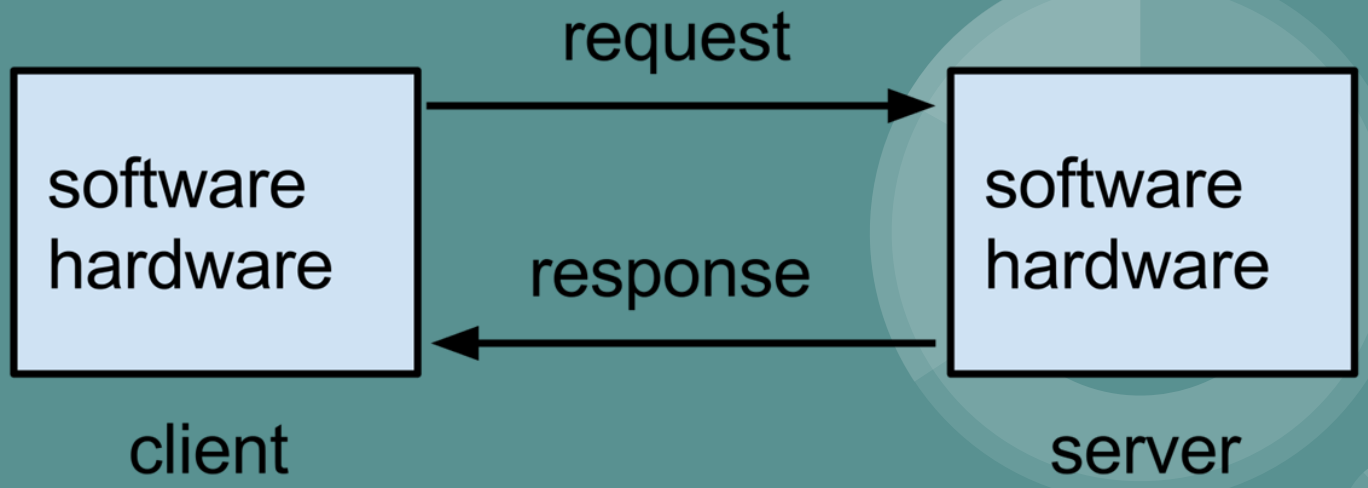
1.建立專案資料夾 mkdir node_server
2.切換至工作路徑 cd node_server
3.初始化專案 npm init or npm init -y
4.建立檔案server.js & coding & save
5.執行程式 node server.js

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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```



Client發出請求，稱之為 Request。
Server收到請求，根據請求處理後回應，稱之為Response。

使用nodemon套件


```
npm install -g nodemon
```

```
npm install --save-dev nodemon
```

Express Web Framework
建立網頁伺服器

1.建立專案資料夾 mkdir express_server
2.切換至工作路徑 cd express_server
3.初始化專案 npm init or npm init -y
4.npm install --save express
5.建立檔案server.js & coding & save
6.執行程式 node server

```
const express = require('express' 4.16.2 )
const app = express()

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(3000, () => console.log('Example app listening on port 3000!'))
```

RESTful

Representational State Transfer，簡稱REST。

它是一種網路架構風格，他並不是一種標準。

將server端提供的內容，視為一項resource。

並呈現在瀏覽器URL上如http://localhost/user/5。

網址為一項資源，針對資源的操作，主要靠HTTP的request method

非RESTful的設計方式

POST /user/add?name=jack Create
GET /user/get?name=jack Read
POST /user/update?name=jack Update
GET /user/remove?name=jack Delete

早期程式設計者或許會設計成如下的URL

RESTful的設計方式

POST /user/jack	Create
GET /user/jack	Read
PUT /user/jack	Update
DELETE /user/jack	Delete

URL設計資源

http request method定義資源的操作

Name	Path	HTTP Verb	Purpose	ActiveRecord CRUD Methods	SQL
Index	/dogs	GET	List all dogs	Dog.all	SELECT * FROM dogs
Show	/dogs/:id	GET	Show information about one dog	Dog.find_by(name: "Boo") OR Dog.find(1)	SELECT * FROM dogs WHERE name = "Boo" OR SELECT * FROM dogs WHERE id = 1
New	/dogs/new	GET	Show new dog form		
Create	/dogs	POST	Create a new dog, then redirect	Dog.create(name: "Boo")	INSERT INTO dogs (name) VALUES ("Boo")
Edit	/dogs/:id/edit	GET	Show edit form for one dog		
Update	/dogs/:id	PATCH	Update one dog, then redirect	Dog.update(id, :age = 10)	UPDATE dogs SET (age = 10) WHERE id = 1
Destroy	/dogs/:id/delete	DELETE	Delete one dog, then redirect	Dog.destroy(id)	DELETE FROM dogs WHERE id = 1

RESTful Routes

A table of all 7 RESTful routes

Name	Path	HTTP Verb	Purpose
Index	/dogs	GET	List all dogs
New	/dogs/new	GET	Show new dog form
Create	/dogs	POST	Create a new dog, then redirect somewhere
Show	/dogs/:id	GET	Show info about one specific dog
Edit	/dogs/:id/edit	GET	Show edit form for one dog
Update	/dogs/:id	PUT	Update a particular dog, then redirect somewhere
Destroy	/dogs/:id	DELETE	Delete a particular dog, then redirect somewhere

Express提供Restful Web API

app.METHOD(PATH, HANDLER)

- app is an instance of express.
- METHOD is an HTTP request method, in lowercase.
- PATH is a path on the server.
- HANDLER is the function executed when the route is matched.

app.METHOD(PATH, HANDLER)

HANDLER(callback)

```
function(req,res) {
        //處理連線要求，並回應
browser
}
```

request -連線要求的資訊和方法
response -回應連線的資訊和方法

```
app.post('/api/todos', function (req, res)
    {
        處理程式碼
    });
```

```
addTodo(todo): Observable<any> {
    let url = this.backendBaseUrl + '/todos';
    let body = JSON.stringify(todo);
    console.log(body);
    return this.http.post(url, body, this.options)
        .map(res => res.json().data);
}
```

```
app.get('/api/todos', function (req, res)
    {
        處理程式碼
    });
```

```
getTodos(delay = 0): Observable<any> {
    return this.http
        .get(this.backendBaseUrl+'/todos')
        .delay(delay)
        .map(res => res.json().data);
}
```

```
app.get('/api/todos/:id', function (req, res)
    {
        處理程式碼
    });
```

```
getTodo(id): Observable<any> {
    let url = this.backendBaseUrl + '/todos/' + id;
    return this.http.get(url)
        .map(res => res.json().data);
}
```

```
app.put('/api/todos/:id', function (req, res)
    {
        處理程式碼
    });
```

```
updateTodo(todo): Observable<any> {
    let url = this.backendBaseUrl + '/todos/' + todo.id;
    console.log(url);
    let body = JSON.stringify(todo);
    return this.http.put(url, body, this.options)
        .map(res => res.json());
}
```

```
app.delete('/api/todos/:id', function (req, res)
    {
        處理程式碼
    });
```

```
deleteTodo(id): Observable<any> {
    let url = this.backendBaseUrl + '/todos/' + id;
    return this.http.delete(url, this.options)
        .map(res => res.json());
}
```

```
app.put('/api/todos/toggledone/:id', function (req, res)
    {
        處理程式碼
    });
```

```
toggleDone(id): Observable<any> {
    let url = this.backendBaseUrl + '/todos/toggleDone/' + id;
    return this.http.put(url, '', this.options)
                .map(res => res.json())
                .do(data => console.log(data));
}
```

todo-serveice.ts

```
//backendBaseUrl: string = environment.url + 'todos.php';
backendBaseUrl: string = 'http://localhost:3000/api';
headers                = new Headers({'Content-Type': 'application/json'});
options                = new RequestOptions({headers: this.headers/*, withCredentials:
```

安装mysql第三方套件

npm install --save express

node.js模組簡介

有三大類的模組：

Core Modules (原生模組)如http、path、fs

Local Modules (自建模組)如db.js、routes.js

Third Party Modules (第三方模組)如express、body-parser

引入模組

撰寫模組db.js

```
module.exports = dbconn
```

```
const dbconn = require('../model/db.js')
```

```
dbconn.query(sql, (err, rows) => ...)
```

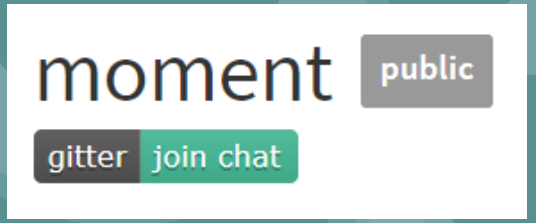
安裝body-parser第三方套件


```
npm install --save body-parser
```

安裝morgan第三方套件


```
npm install --save morgan
```

安裝moment第三方套件




```
npm install --save-dev cors
```

安裝cors第三方套件


```
npm install --save-dev cors
```

Route Parameters路由參數

Route parameters

Route parameters are named URL segments that are used to capture the valu
populated in the `req.params` object, with the name of the route parameter sp

```
Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }
```

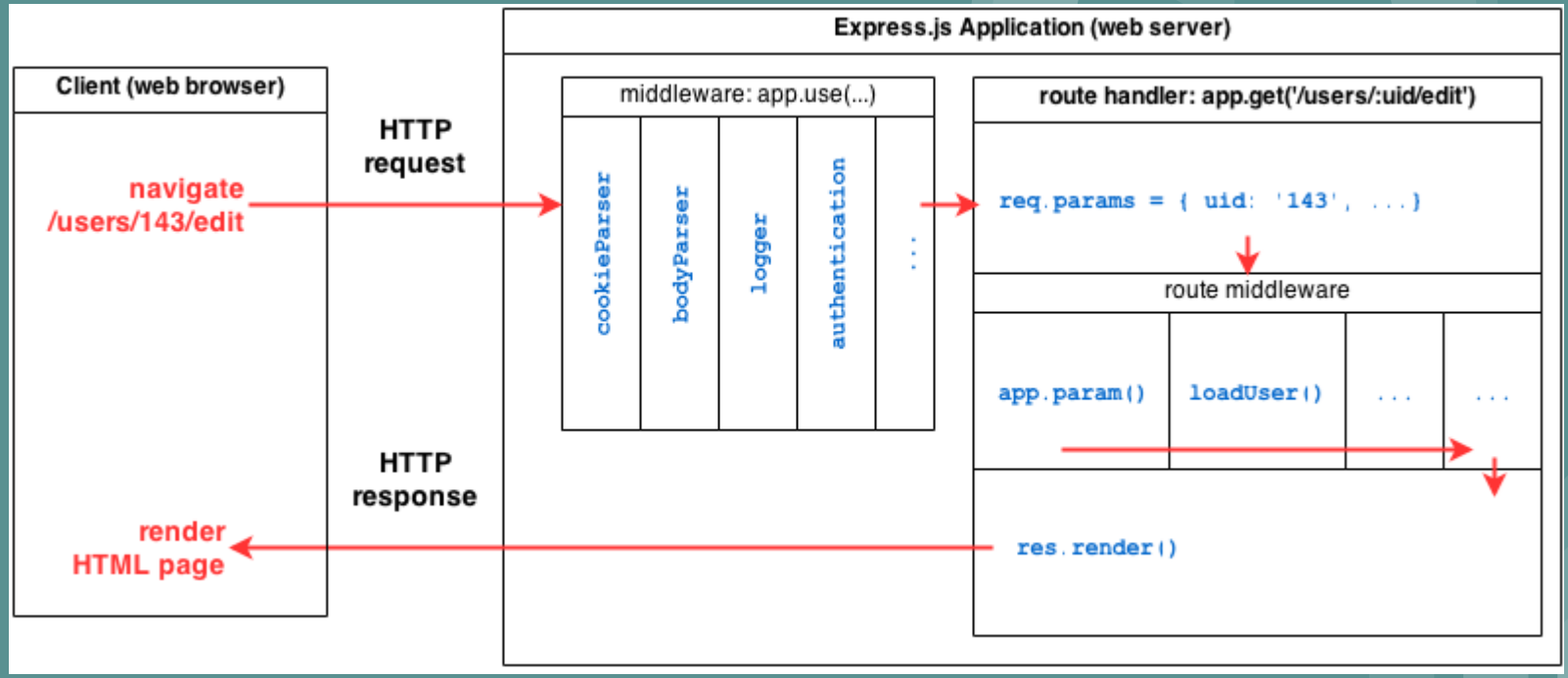
To define routes with route parameters, simply specify the route parameters

```
app.get('/users/:userId/books/:bookId', function (req, res) {
  res.send(req.params)
})
```

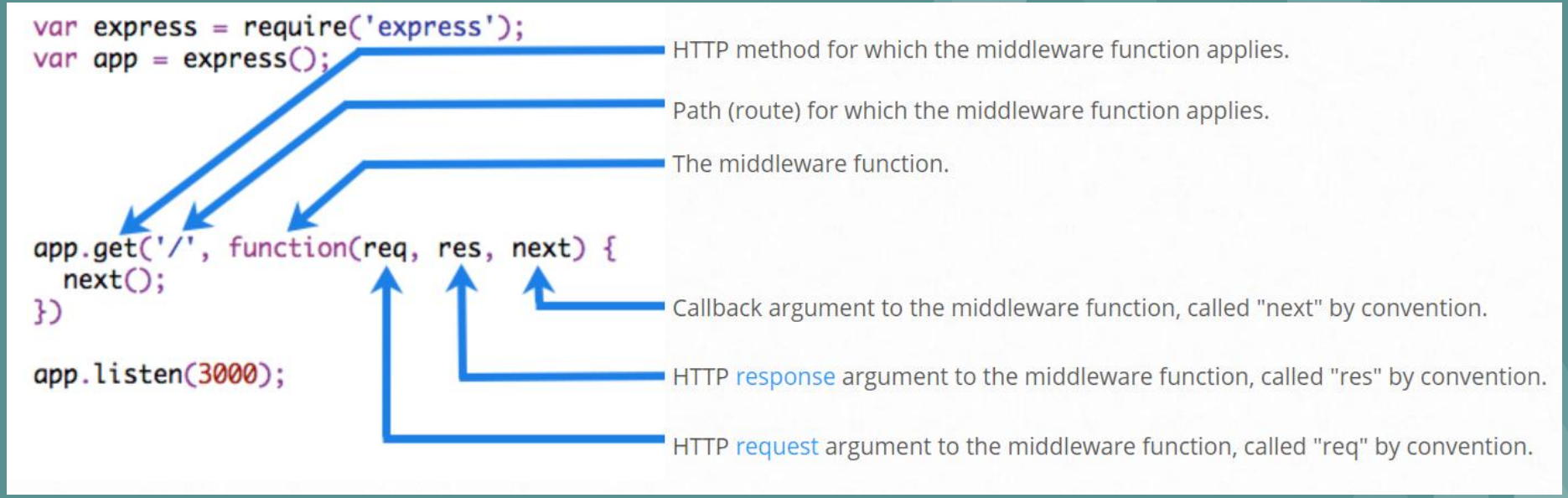
req.body方法來獲取表單資料

```
app.post('/news', function (req, res) {
 let data = {
   title : req.body.title,
   content:req.body.content
   }
   res.send(data);
});
```

middleware中介函式



exprees middleware寫法



exprees middleware

An Express application can use the following types of middleware:

- Application-level middleware
- Router-level middleware
- Error-handling middleware
- Built-in middleware
- Third-party middleware

Router-level middleware

```
app.use(express.static(path.join(__dirname, 'public')));
app.use(express.static(path.join(__dirname, 'dist')));
```

Built-in middleware

```
const api = equire('./routes');
 app.use('/api',api);
```

```
router.get('/todos', (req, res) => {
    let sql = 'SELECT id, title, content, done FROM todos ORDER BY id DESC';
    dbconn.query(sql, (err, rows) => {
• • • •       • • • •     • • • •
    });
});
```

Third-party middleware

```
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
extended: false
}));
```

Third-party middleware

```
app.use(logger('dev'));
app.use(cors());
```

```
SELECT id, title, content, done FROM todos ORDER BY id DESC
```

```
app.get('/api/todos', function (req, res) {
    let sql = 'SELECT id, title, content, done FROM todos ORDER BY id DESC';
    dbconn.query(sql, function (err, rows) {
        if (err) {
            console.log(err);
            res.send(err);
        }
            res.json({
            data: rows
          });

    });
});
```

取得 todo list

```
SELECT id, title, content, done FROM todos WHERE id = ?
```

```
app.get('/api/todos/:id', function (req, res) {
    let sql = `SELECT id, title, content, done FROM todos WHERE id = ?`;
    dbconn.query(sql, [req.params.id], function (err, row) {
        if (err) {
            console.log(err);
            res.send(err);
        }
        console.log(JSON.stringify(row[0]));
        res.json({
            data: row[0]
         });
    });

});
```

取得一筆 todo

```
INSERT INTO todos (title, content, created_at) VALUES
(:title, :content, :created_at)
```

```
app.post('/api/todos', function (req, res) {
    let data = {
        title: req.body.title,
        content: req.body.content,
        created_at: moment().format("YYYY-MM-DD HH-mm-ss")
    }
    let sql = "INSERT INTO todos SET ?";
    dbconn.query(sql, data, function (err, row) {
        if (err) {
            console.log(err);
            res.send(err);
        }
        res.json({
            data: row
          });
    });
});
```

新增 todo

```
UPDATE todos SET title=:title, content=:content, done=:done,
updated_at=:updated_at WHERE id=:id
```

```
app.put('/api/todos/:id', function (req, res) {
    let data = {
        title: req.body.title,
        content: req.body.content,
        done:true,
        updated_at: moment().format("YYYY-MM-DD HH-mm-ss")
    }
    let sql = "UPDATE todos SET ? WHERE id = ?";
    dbconn.query(sql, [data,req.params.id], function (err, row) {
        if (err) {
                        console.log(err);
                }
        res.json({
                        data: row
                });
        });
});
```

更新一筆 todo

```
Update todos SET done= NOT done, updated_at=:updatedAt WHERE id = :id
```

```
app.put('/api/todos/toggledone/:id', function (req, res) {
    let sql = `UPDATE todos SET done= NOT done,
updated_at="${moment().format("YYYY-MM-DD HH-mm-ss")}" WHERE id
=${req.params.id}`;
    dbconn.query(sql, function (err, row) {
        if (err) {
            console.log(err);
            res.send(err);
        }
        console.log(row);
        res.json({
            data: row
          });
    });
});
```

切換一筆 todo 的完成與否

app-routing.module.ts

```
const routes: Routes = [

 {path: 'home', component: HomeComponent},
 {path: 'todos', component: TodoListComponent},
 {path: 'todos/new', component: NewTodoComponent},
 {path: 'todos/:id/edit', component: EditTodoComponent},
...
```


Pricing
Apps
Documentation
Integrations
POSTMAN
Modern software
is built on APIs.
Postman helps you develop APIs faster.
Chrome App
Mac App
Go ahead and download our apps, they're free!

localhost:3000/api/todos/


```
{
    "data": [
        {
            "id": 54,
            "title": "賴揆拍版 3個月加班不得超過138小時",
            "content": "賴清德中午主持行政立法協調會報，會後徐國勇轉述會中共識。其中，對於休息日出勤工資計算，取消「做1給4」、「做5給8」計算方式，改依勞工實際出勤的時間來計算。",
            "done": 1
        },
        {
            "id": 53,
            "title": "政院拍板一例一休修法方向 國民黨團擬提版本因應",
            "content": "對於一例一休修法，立法院國民黨團召開幹部會議研商相關對策，黨團副書記長蔣萬安會後轉述，黨中央與黨團基本上有共識，反對政院版鬆綁七休一、調高加班時數上限及縮短輪班間隔",
            "done": 0
        }
```


```
app.get('/api/todos',...
```

localhost:3000/api/todos/54


```
{
    "data":
        {
            "id": 54,
            "title": "賴揆拍版 3個月加班不得超過138小時",
            "content": "賴清德中午主持行政立法協調會報，會後徐國勇轉述會中共識。其中，對於休息日出勤工資計算，取消「做1給4」、「做5給8」計算方式，改依勞工實際出勤的時間來計算。",
            "done": 1
        }
```


```
app.get('/api/todos/:id',...
```

http://localhost:3000/api/todos/40


```
{
    "message": "刪除資料成功"
}
```


```
app.delete('/api/todos/:id',...
```

http://localhost:3000/api/todos/


```
{
    "message": "新增資料成功"
}
```


```
app.post('/api/todos',...
```

http://localhost:3000/api/todos/40


```
{
    "message": "修改資料成功"
}
```

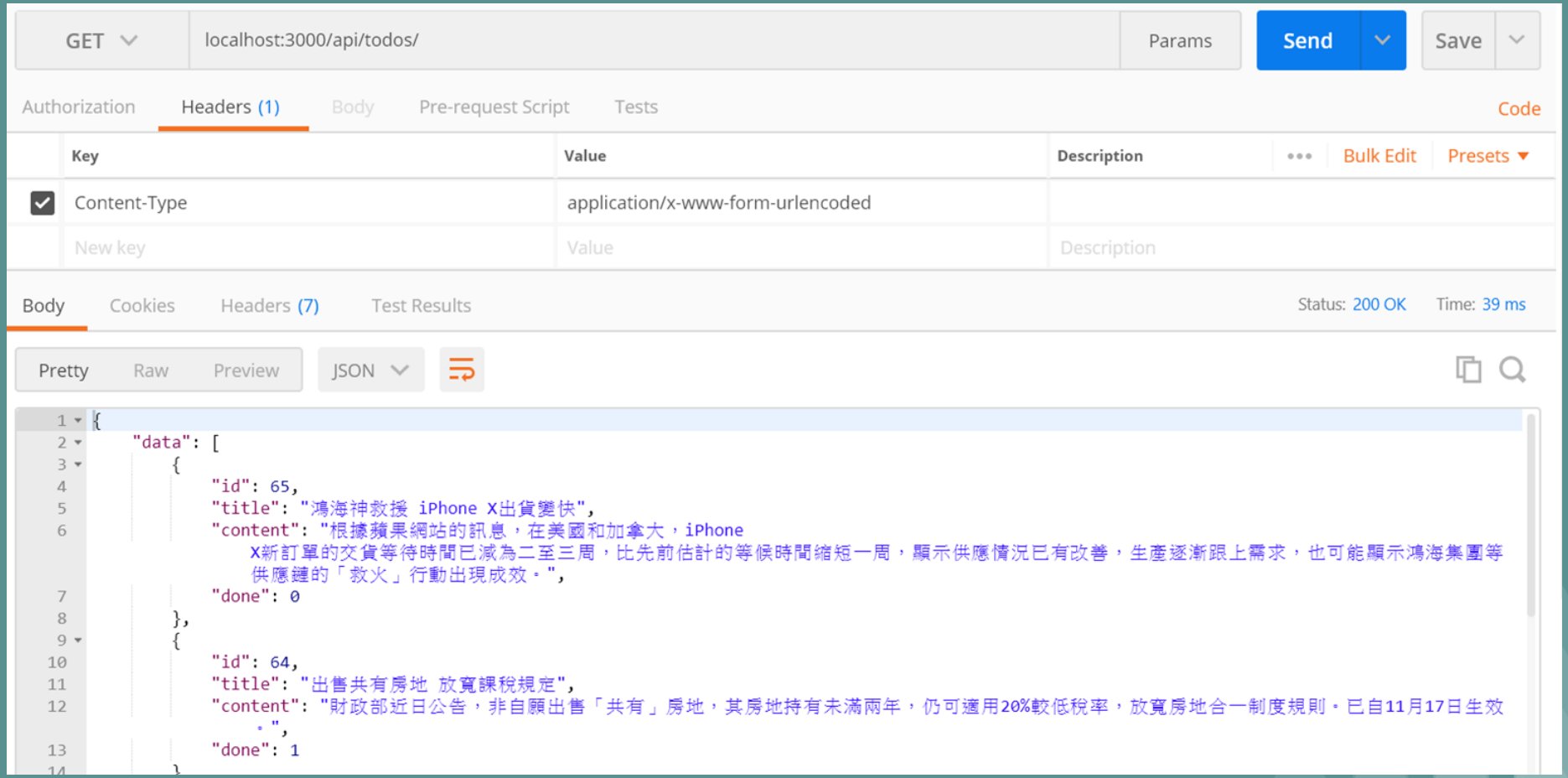

```
app.put('/api/todos/:id', …)
```

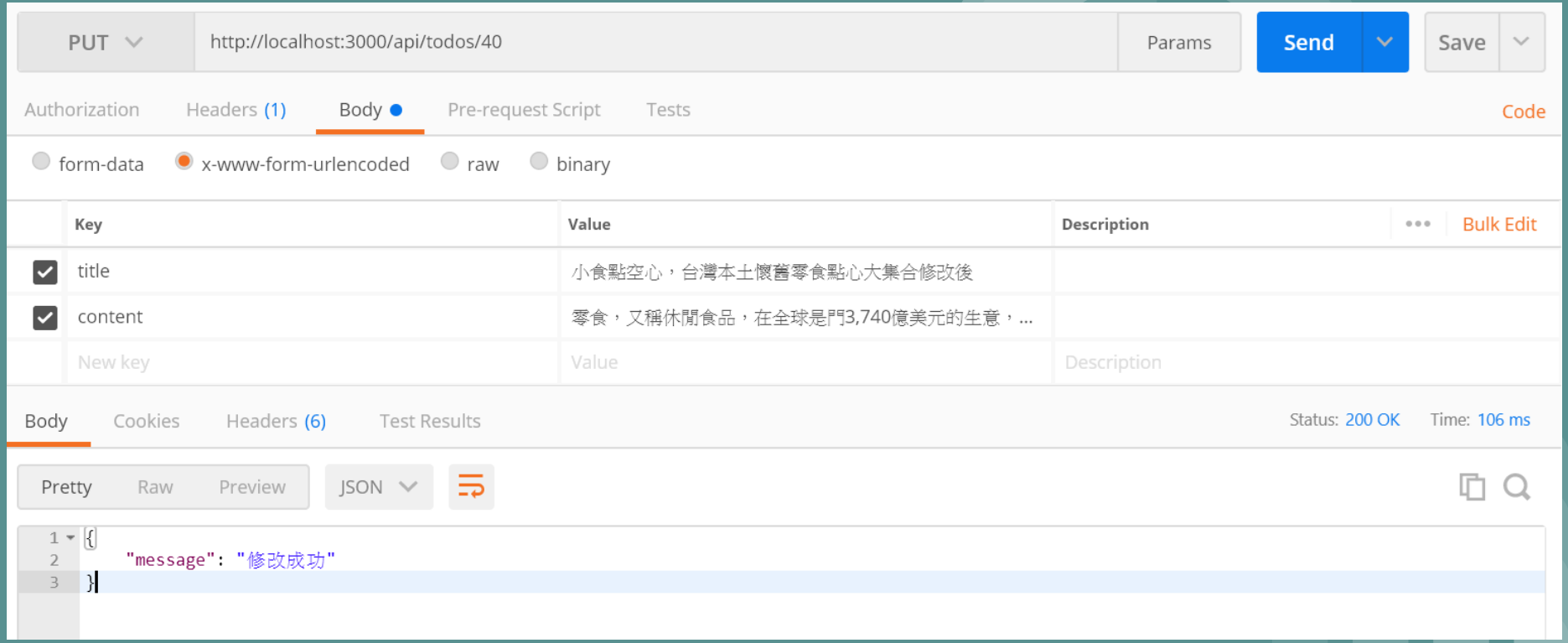
localhost:3000/api/todos/toggledone/40

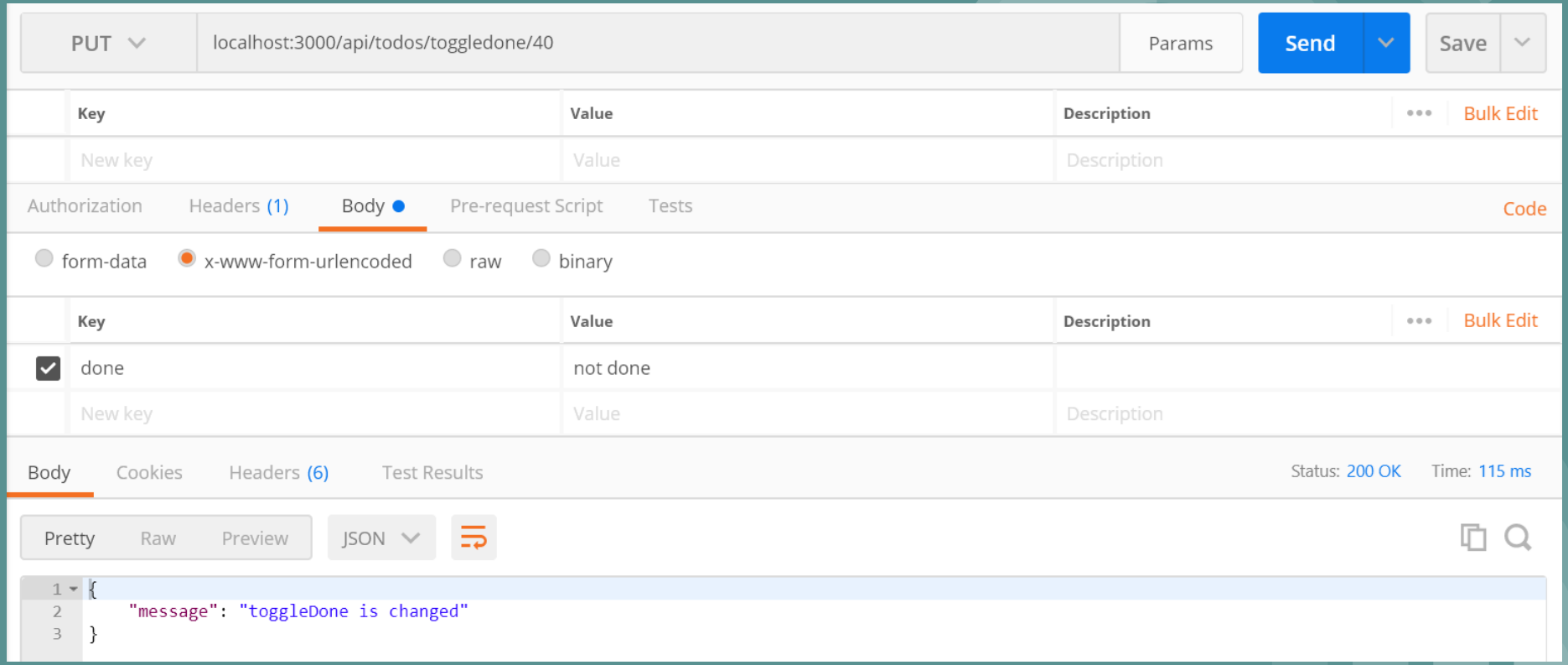

```
{
    "message": "toggleDone is changed"
}
```

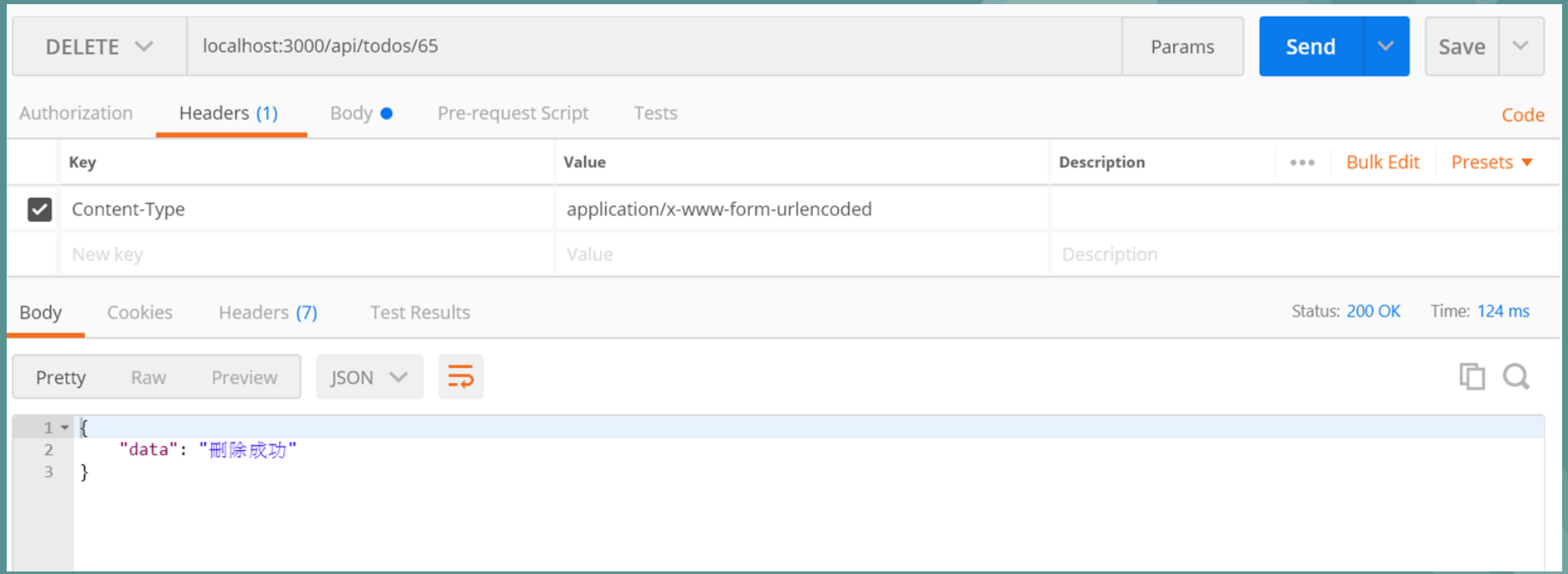

```
app.put('/api/todos/toggledone/:id',... )
```

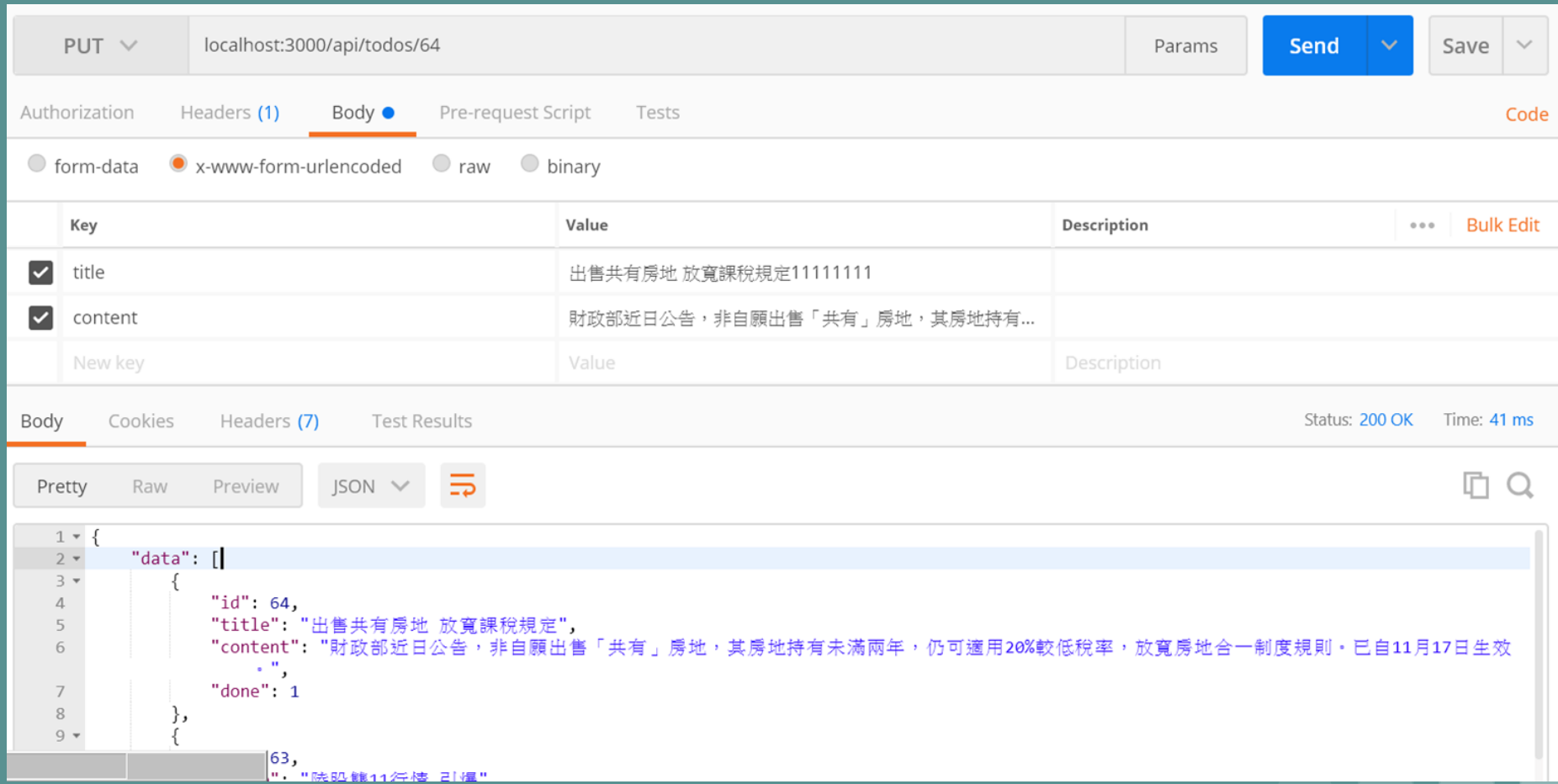

POST
localhost:3000/api/todos/
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
form-data
x-www-form-urlencoded
raw
binary
Key
Value
Description
Bulk Edit
title
國門陸橋大火 清潔隊員被法辦
content
警方調查結果，排除人為縱火之嫌，已將負責管理的環...
New key
Value
Description


GET
localhost:3000/api/todos/
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
Key
Value
Description
Bulk Edit
Presets
Content-Type
application/x-www-form-urlencoded
New key
Value
Description
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 39 ms
Pretty
Raw
Preview
JSON
{
"data": [
{
"id": 65,
"title": "鴻海神救援 iPhone X出貨變快",
"content": "根據蘋果網站的訊息，在美國和加拿大，iPhone X新訂單的交貨等待時間已減為二至三周，比先前估計的等候時間縮短一周，顯示供應情況已有改善，生產逐漸跟上需求，也可能顯示鴻海集團等供應鏈的「救火」行動出現成效。",
"done": 0
},
{
"id": 64,
"title": "出售共有房地 放寬課稅規定",
"content": "財政部近日公告，非自願出售「共有」房地，其房地持有未滿兩年，仍可適用20%較低稅率，放寬房地合一制度規則。已自11月17日生效。",
"done": 1


PUT
http://localhost:3000/api/todos/40
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
form-data
x-www-form-urlencoded
raw
binary
Key
Value
Description
Bulk Edit
title
小食點空心，台灣本土懷舊零食點心大集合修改後
content
零食，又稱休閒食品，在全球是門3,740億美元的生意，...
New key
Value
Description
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 106 ms
Pretty
Raw
Preview
JSON
{
"message": "修改成功"
}


PUT
localhost:3000/api/todos/toggledone/40
Params
Send
Save
Key
Value
Description
Bulk Edit
New key
Value
Description
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
form-data
x-www-form-urlencoded
raw
binary
Key
Value
Description
Bulk Edit
done
not done
New key
Value
Description
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 115 ms
Pretty
Raw
Preview
JSON
{
"message": "toggleDone is changed"
}


DELETE
localhost:3000/api/todos/65
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
Key
Value
Description
Bulk Edit
Presets
Content-Type
application/x-www-form-urlencoded
New key
Value
Description
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 124 ms
Pretty
Raw
Preview
JSON
{
"data": "删除成功"
}


PUT
localhost:3000/api/todos/64
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
form-data
x-www-form-urlencoded
raw
binary
Key
Value
Description
Bulk Edit
title
出售共有房地 放寬課稅規定11111111
content
財政部近日公告，非自願出售「共有」房地，其房地持有...
New key
Value
Description
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 41 ms
Pretty
Raw
Preview
JSON
{
"data": [
{
"id": 64,
"title": "出售共有房地 放寬課稅規定",
"content": "財政部近日公告，非自願出售「共有」房地，其房地持有未滿兩年，仍可適用20%較低稅率，放寬房地合一制度規則，已自11月17日生效
。",
"done": 1
},
{
63,

localhost:3000/api/todos

```
// 20171122110108
// http://localhost:3000/api/todos

{
  "data": Array[9][
    {
      "id": 64,
      "title": "出售共有房地 放寬課稅規定11111111",
      "content": "財政部近日公告，非自願出售「共有」房地，其房地持有未滿兩年，仍可適用20%較低稅率，放寬房地合一制度規則。已自11月17日生效。11111111111",
      "done": 1
    },
    {
      "id": 63,
      "title": "陸股雙11行情 引爆",
      "content": "由阿里巴巴發起的「1111光棍節」大購物日將於本周六登場。法人分析，今年淘寶天貓雙11光棍節銷售額上看人民幣1,500億元（約新台幣6,825億元），建議投資人以主動式大中華股票型基金，掌握陸股中的相關題材",
      "done": 0
    },
    {
      "id": 58,
      "title": "國門陸橋大火 清潔隊員被法辦",
      "content": "警方調查結果，排除人為縱火之嫌，已將負責管理的環保局蘆竹清潔隊邱姓隊員，依公共危險罪嫌函送桃園地檢署偵辦。",
      "done": 1
```

課程完成品
程式Demo