Node.js (1)

ผศ. ดร. เก็จแก้ว ธเนศวร

kejkaew.tha@mail.kmutt.ac.th

ตารางเรียน

สัปดาห์	วันที่	หัวข้อ/รายละเอียด	การบ้าน (ผู้ตรวจ)	ส่ง Assignment
1	8/8/2566	Introduction to Web programming, Basic HTML 1	Assignment 1 (KT,KN)	1
2	15/8/2566	Basic HTML 2 + CSS 1	Assignment 2 (KT,KN)	1,2
3	22/8/2566	CSS 2	Assignment 3 (KT)	1,2,3
4	29/8/2566	Responsive Web design	Assignment 4 (KT,KN)	2,3,4
5	5/9/2566	JavaScript 1 + DOM	Assignment 5 (KT,KN)	3,4,5
6	12/9/2566	JavaScript 2 + GitHub	Assignment 6 (KT)	4,5,6
7	19/9/2566	JavaScript 3	Assignment 7 (KT,KN)	5,6,7
8	26/9/2566	AJAX + JSON	Assignment 8 (KT,KN)	6,7,8
9	3/10/2566	Node.js 1	Assignment 9 (KT)	6,7,8,9
	10/10/2566	No class	ส่ง portfolio web page (KT)	
10	17/10/2566	Node.js 2	Assignment 10 (KT,KN)	8,9,10
11	24/10/2566	Node.js 3	Assignment 11 (KT,KN)	9,10,11
12	31/10/2566	Node.js 4 + MySQL	Assignment 12 (KT)	10,11,12
13	7/11/2566			10,11,12
14	14/11/2566			11,12
15	21/11/2566			
16	28/11/2566	ส่งโปรเจค (9:00-16:00)	KT	
		สอบปลายภาค ในตาราง	KT	

สัปดาห์ที่แล้ว

- AJAX : Asynchronous JavaScript And XML
- JSON: JavaScript Object Notation.
- Assignment 8

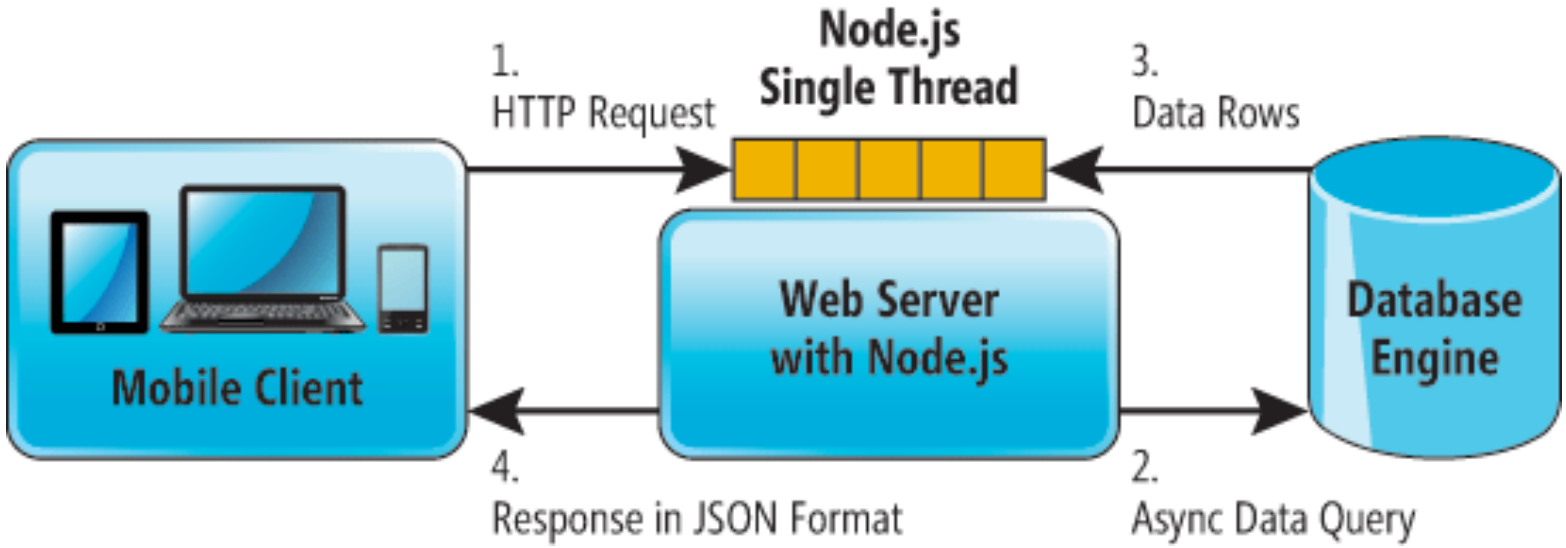
เรื่องที่จะเรียนวันนี้

- Node.js
- http module
- File system module
- Promises
- Assignment 9

kejkaew.tha@mail.kmutt.ac.th

ทบทวน

- Client – Server relationship



kejkaew.tha@mail.kmutt.ac.th

What is Node.js?

- Node.js is an asynchronous event-driven JavaScript runtime framework
- Designed to build scalable network applications.
- Node.js was developed in 2009
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

Why Node.js?

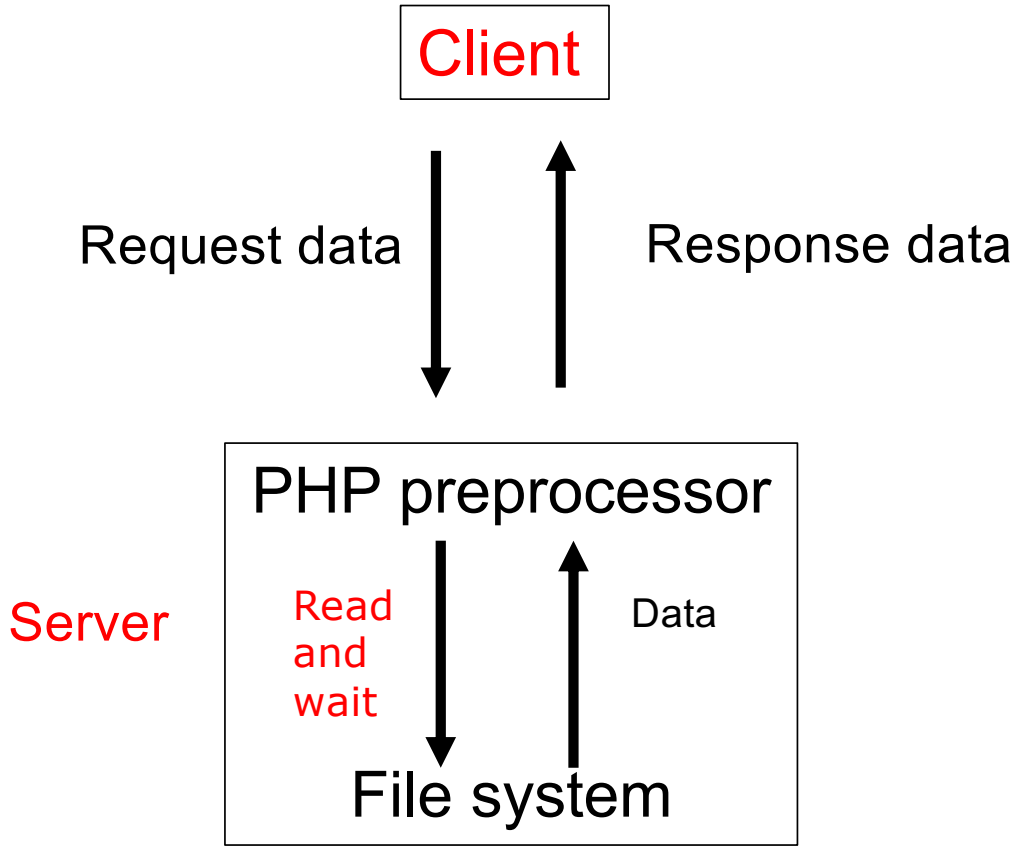
Node.js มีการใช้ asynchronous programming ดังนั้น

- Node.js eliminates the waiting, and simply continues with the next request.
- Node.js runs single-threaded, non-blocking, asynchronously programming, which is very memory efficient.

What is Node.js files

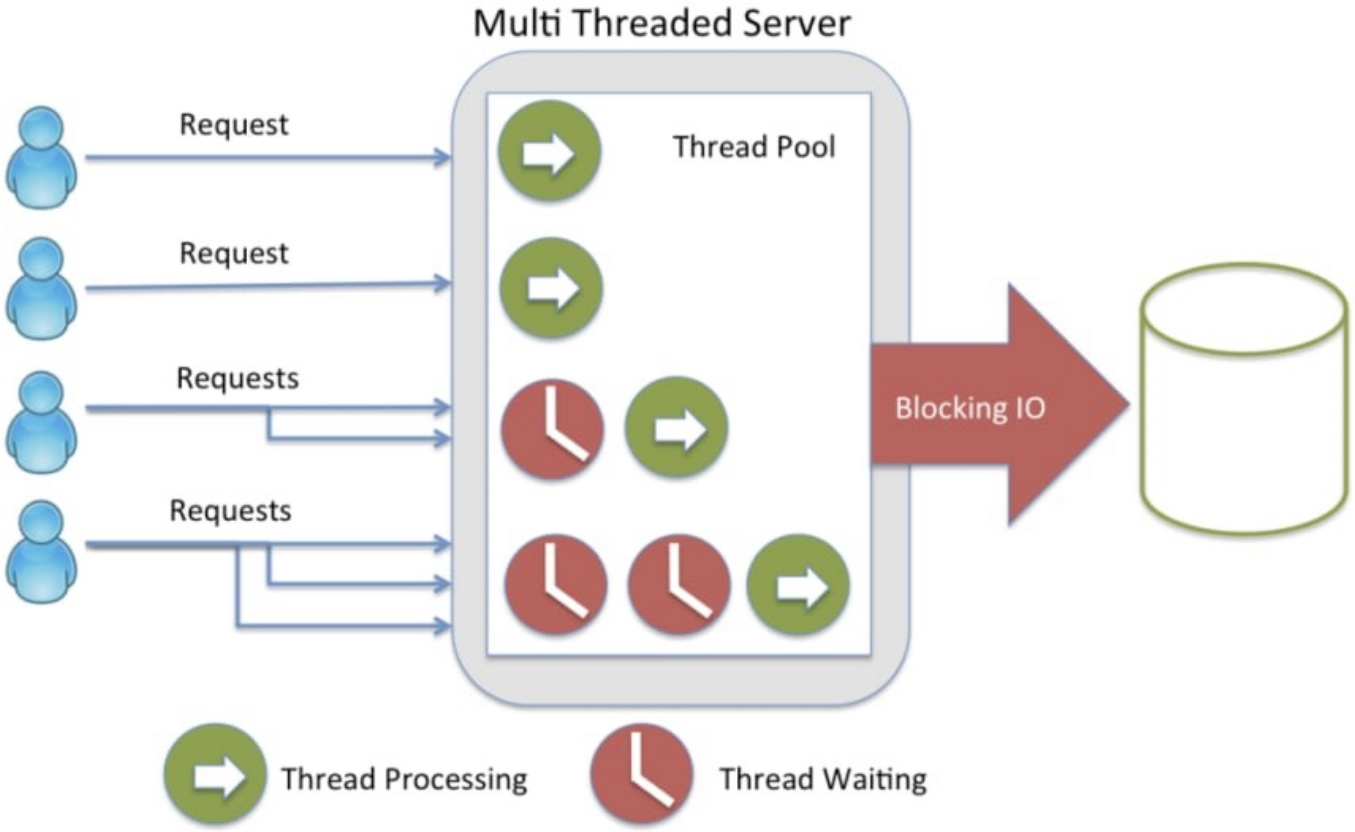
- Node.js files จะประกอบไปด้วยชุดคำสั่งทั้งหมดสำหรับเหตุการณ์ (event) บางอย่าง
- เหตุการณ์ที่ว่าจะเกี่ยวข้องกับงานที่ตัว web application ต้องทำบน server เช่น อ่าน อัพเดต เพิ่ม ข้อมูลใน ฐานข้อมูล
- ดังนั้น Node.js files ต้องรันรอเหตุการณ์ที่จะเกิดขึ้น ใน server เสมอ
- Node.js files have extension ".js"

Here is how PHP or ASP handles a file request:



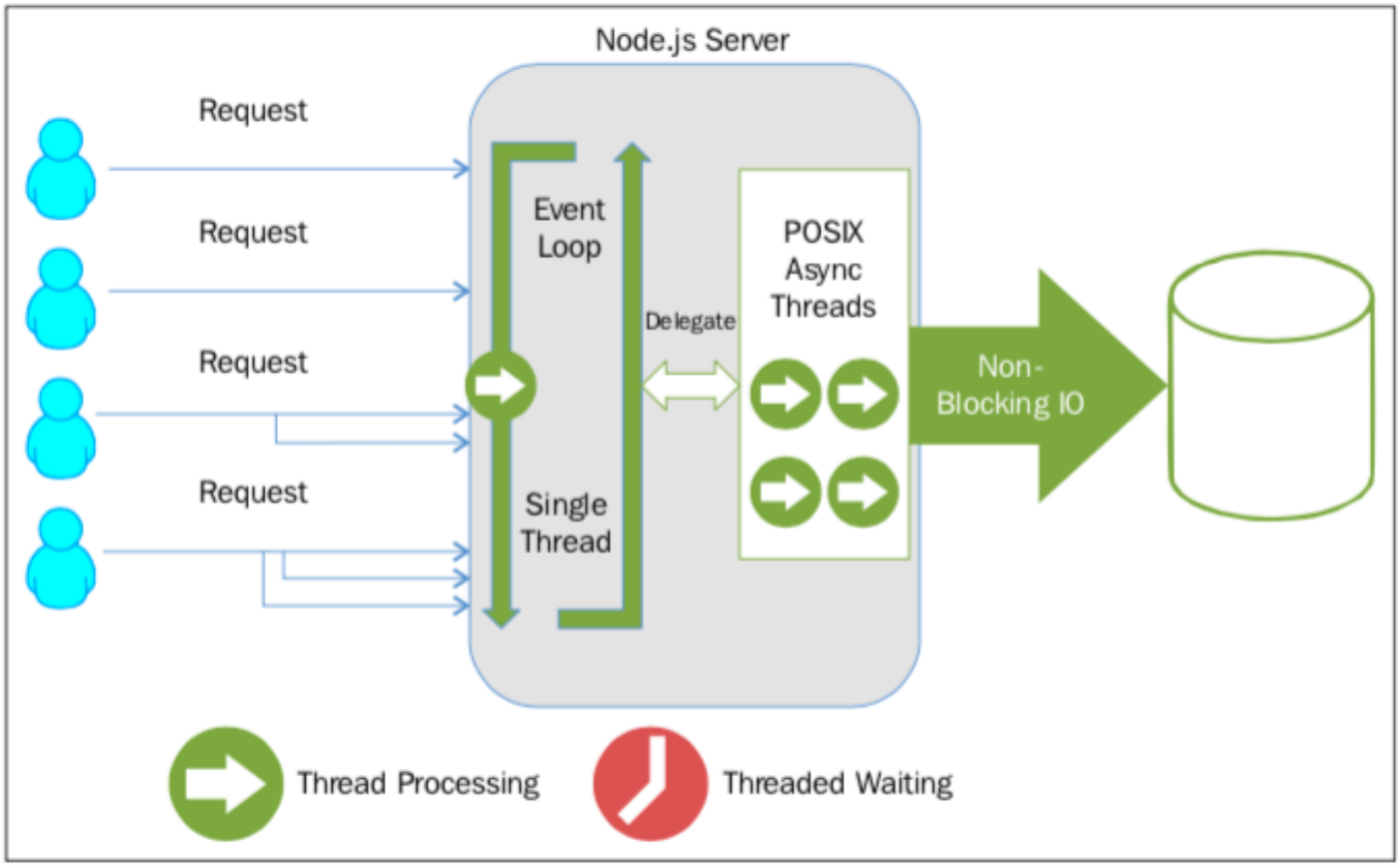
kejkaew.tha@mail.kmutt.ac.th

Multi Threaded Server



kejkaew.tha@mail.kmutt.ac.th

How Node.js works



kejkaew.tha@mail.kmutt.ac.th

สิ่งที่ควรรู้ใน Node.js

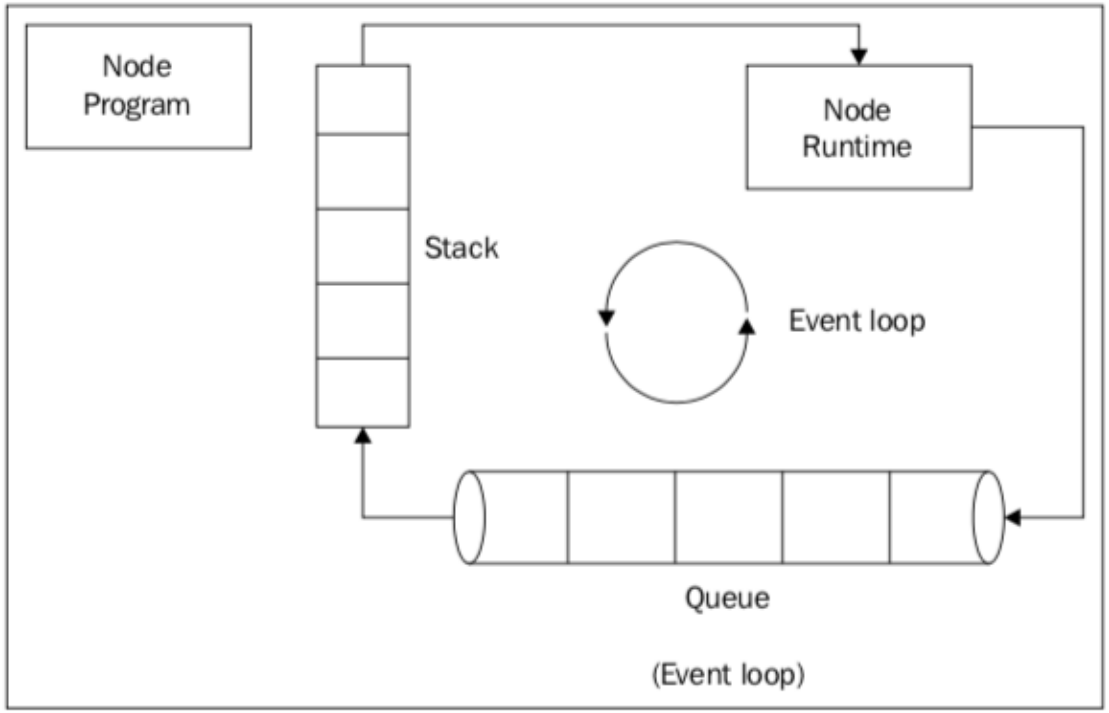
- Callback function = เมื่อทำ task บางอย่างเสร็จแล้ว เช่น กดปุ่ม จะให้ทำ function task อะไรต่อไป
- Blocking or wait = รันแบบ synchronously
- Non-blocking = รันแบบ asynchronously
- Event-driven programming = ลักษณะ flow ของ program ถูกกำหนดโดย user actions เช่น clicks, press

How Event Loop Works

- เนื่องจาก Node.js มีการรัน แบบ asynchronous ดังนั้น เราต้องเข้าใจการทำงานของมัน
- ใน event loop, Node.js มีการจัด queue เพื่อเก็บตัว callback function และจะทำการ execute เมื่อ stack ว่าง
- ดังนั้นสิ่งที่เป็น callback function จะรันที่หลังตัว task ปกติ ดังรูป

kejkaew.tha@mail.kmutt.ac.th

How Event Loop Works



kejkaew.tha@mail.kmutt.ac.th

ตัวอย่างเช่น

JS server.js > ...

```
1  console.log('i am first');
2
3  setTimeout(() => {
4    console.log('i am second');
5  }, 0);
6
7  console.log('i am third');
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
i am first
i am third
i am second
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$
```


kejkaew.tha@mail.kmutt.ac.th

Arrow Function

- Arrow functions allow us to write shorter function syntax:
- ตัวอย่างเช่น

```
10  function hello(){
11      console.log("Hello");
12  }
13  hello()
```

```
15  hello = () => {
16      console.log("Hello");
17  }
18  hello()
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
Hello
```


kejkaew.tha@mail.kmutt.ac.th

Arrow function with parameters

```
20  hello = (message) => {
21      console.log("Hello "+message);
22  }
23  hello("Kejkaew")
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
Hello Kejkaew
```

Create server

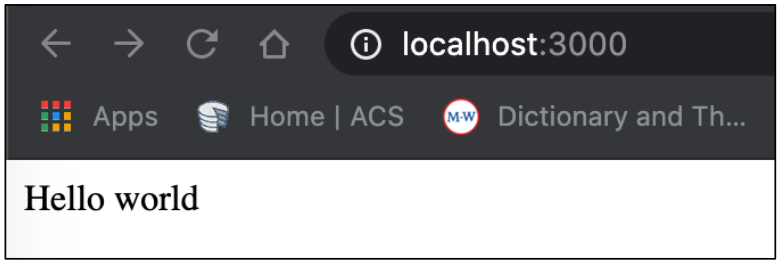
- ลองรัน server

```
25  const http = require('http');
26  const hostname = 'localhost';
27  const port = 3000;
28  const server = http.createServer((req, res) => {
29      res.writeHead(200, {'Content-Type': 'text/html'});
30      res.write("Hello world");
31      res.end();
32  });
33
34  server.listen(port, hostname, () => {
35    console.log(`Server running at   http://${hostname}:${port}/`);
36  });
```

Run node

```
kejkaew@Kejkaews-MacBook-Pro-2 node-1 % node server.js
Server running at   http://localhost:3000/
```

Output



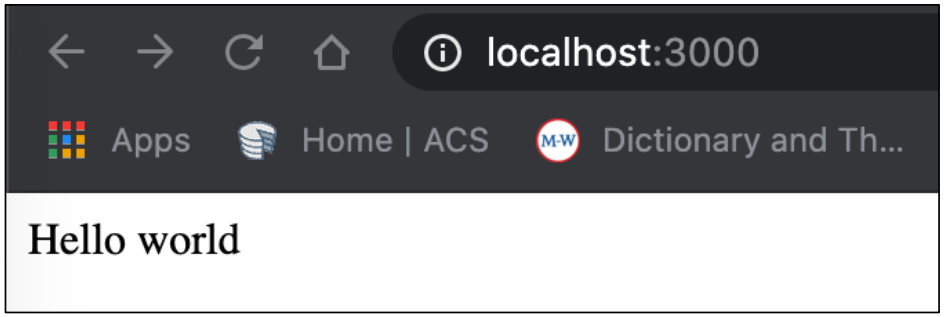
The Built-in HTTP Module

- Node.js มี built-in module ที่ชื่อว่า HTTP ซึ่งใช้สำหรับส่งข้อมูลผ่าน HTTP protocol
- สำหรับการใช้ module นี้ ต้องทำการ import ก่อนโดยใช้ require() method:
- HTTP module จะทำการสร้าง HTTP server ที่รอการร้องขอจาก client และส่งข้อมูลกลับไปหา client ตามที่ขอ
- ใช้ createServer() method สำหรับ สร้าง HTTP server

ตัวอย่าง

```
25  const http = require('http');
26  const hostname = 'localhost';
27  const port = 3000;
28  const server = http.createServer((req, res) => {
29      res.writeHead(200, {'Content-Type': 'text/html'});
30      res.write("Hello world");
31      res.end();
32  });
33
34  server.listen(port, hostname, () => {
35    console.log(`Server running at   http://${hostname}:${port}/`);
36  });
```

Output



kejkaew.tha@mail.kmutt.ac.th

- Function ที่อยู่ใน http.createServer() method จะทำงานรอจนกว่าจะมีผู้ใช้ติดต่อเข้ามาที่ port ตามที่เรากำหนด เช่น 3000
  - มี parameters 2 ตัว คือ request (req) และ response (res)
  - req argument คือ request จาก client, as an object
  - res argument คือ response จาก server, as an object

kejkaew.tha@mail.kmutt.ac.th

- ถ้าจะแสดงผล response จาก HTTP server ผ่านทาง web browser ในรูปแบบ HTML
  - เราต้องใส่ HTTP header และ content type:
- argument ตัวแรกของ res.writeHead() method คือ status code ซึ่ง 200 = OK
- argument ตัวที่สอง คือ object ของ response headers.

kejkaew.tha@mail.kmutt.ac.th

- res.write(message) = ส่ง response ไปที่ client ตามที่ header กำหนด
- res.end() = เป็นการบอกว่า server ส่งข้อมูลต้องการส่ง หมดแล้ว

kejkaew.tha@mail.kmutt.ac.th

File system in Node.js

- Read
- Write
- Append
- Delete
- Check if files exists

kejkaew.tha@mail.kmutt.ac.th

Read a File in Node.js

- ในการ import ตัว File System module สามารถใช้ require('fs’)
- The fs.readFile(filename,callback) method is used to read files on computer.

ตัวอย่าง

```
40  const fs=require('fs');
41  fs.readFile('test.txt', (err, data) => {
42      if (err)
43          throw err;
44
45      console.log("Content :  " + data);
46  });
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
Content :  Node.js is an open-source, cross-platform, back-end JavaScript runtime environment t
eb browser. Node.js lets developers use JavaScript to write command line tools and for server-s
age content before the page is sent to the user's web browser. Consequently, Node.js represents
evelopment around a single programming language, rather than different languages for server-sic
from https://en.wikipedia.org/wiki/Node.js
```

Write a File in Node.js

- fs.writeFile(filename, data, callback) method ใช้สำหรับทำการเขียน file
- ถ้ามี filename นั้น อยู่แล้วจะทำการเขียนทับ
- ถ้าไม่มี filename นั้น จะทำการสร้างใหม่

kejkaew.tha@mail.kmutt.ac.th

ตัวอย่าง

```
48  var fs =  require('fs');
49  var content= "this is the content in the file";
50  fs.writeFile('message.txt', content , (err) => {
51      if (err)
52          throw err;
53      console.log('saved!');
54  });
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
saved!
```


kejkaew.tha@mail.kmutt.ac.th

Append a File using Node.js

- fs.appendFile(filename,data,callback) method เป็นการเขียน data ใหม่เข้าไปใส่ใน filename ที่มีอยู่แล้ว
- ถ้าไม่มี filename นั้น จะทำการสร้างใหม่

ตัวอย่าง

```
56  const fs = require('fs');
57  new_data = "This data will be appended at the end of the file.";
58  fs.appendFile('message.txt', new_data , (err) => {
59      if(err)
60          throw err;
61      console.log('The new_content was appended successfully');
62  });
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
The new_content was appended successfully
```


kejkaew.tha@mail.kmutt.ac.th

Delete a File in Node.js

- ถ้าต้องการลบ filename ให้ใช้ fs.unlink(pathของ filename, callback) method.

```
64  const fs = require('fs');
65  const filename = 'message.txt';
66  fs.unlink(filename, (err) => {
67      if (err)
68          throw err;
69      console.log('File deleted successfully');
70  });
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
File deleted successfully
```


kejkaew.tha@mail.kmutt.ac.th

Check if files exists

- fs.access(path ของ filename, mode, callback)
- เป็น method ที่ใช้สำหรับตรวจสอบว่าการเข้าถึง filename นั้น
- mode argument ใช้เพื่อกำหนดว่าจะเป็นการเข้าถึง filename นั้นแบบไหน

ตัวอย่าง

```
72  const fs = require('fs');
73  const path = './test.txt';
74  fs.access(path, fs.F_OK, (err) => {
75    if (err) {
76      console.log("File not found");
77      return;
78    }else{
79        console.log("Exist!");
80    }
81  });
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
Exist!
```


kejkaew.tha@mail.kmutt.ac.th

Callback hell

```
1  // Callback Hell
2
3
4  a(function (resultsFromA) {
5      b(resultsFromA, function (resultsFromB) {
6          c(resultsFromB, function (resultsFromC) {
7              d(resultsFromC, function (resultsFromD) {
8                  e(resultsFromD, function (resultsFromE) {
9                      f(resultsFromE, function (resultsFromF) {
10                         console.log(resultsFromF);
11                     })
12                 })
13             })
14         })
15     })
16 });
17
```

Nested callbacks stacked below one another forming a pyramid structure

Promises

- Promises เป็นวิธีที่ใช้จัดการกับ asynchronous code เพื่อไม่ให้เกิด callback hell
- เมื่อ promise ถูกเรียกใช้ มันจะอยู่ในสถานะ pending state ซึ่งหมายความว่า callback ทุกตัวที่มี promise ยังทำงานอยู่ และเมื่อตัว callback ก่อนหน้าทำงานสำเร็จ (resolve) แล้ว callback ตัวนั้นก็จะเริ่มทำงานต่อ

```
// Callback Hell


a(function (resultsFromA) {
    b(resultsFromA, function (resultsFromB) {
        c(resultsFromB, function (resultsFromC) {
            d(resultsFromC, function (resultsFromD) {
                e(resultsFromD, function (resultsFromE) {
                    f(resultsFromE, function (resultsFromF) {
                        console.log(resultsFromF);
                    })
                })
            })
        })
    })
});

```

- ดังนั้นในการสร้าง promise สุดท้ายแล้วจะเกิดได้สองสถานะ คือ resolved state หรือ rejected state ไปเรื่อยๆ จนจบ chain ของ promise
- Promise API เริ่มต้นโดยการเรียกใช้ Promise constructor ซึ่งมี syntax คือ new Promise(callback):
- Promise.then(onFulfilled, onRejected) สามารถใส่ argument ได้ 2 แบบ คือ callback สำหรับ resolved state (onFulfilled) และ callback สำหรับ rejected state (onRejected) แต่สามารถใส่แค่ตัวเดียว คือ onFulfilled ก็ได้

ตัวอย่าง ที่ไม่ใช้ promises

```
84  const fs=require('fs');
85  fs.readFile('test.txt', (err, data) => {
86      if (err)
87          throw err;
88      else{
89          fs.writeFile('message.txt', data , (err) => {
90              if (err)
91                  throw err;
92              console.log('saved!');
93          });
94      }
95  });
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
saved!
```

ตัวอย่าง Promises

```
97  const fs = require('fs');
98  const readData = () => {
99      return new Promise((resolve,reject) => {
100         fs.readFile('test.txt', (err, data) => {
101             if (err)
102                 reject(err);
103             else
104                 resolve(data);
105         });
106     })
107 }
```

```
109 const writeData = (data) => {
110     return new Promise((resolve, reject) => {
111         fs.writeFile('message.txt', data , (err) => {
112             if (err)
113                 reject(err);
114             else
115                 resolve("saved!")
116         });
117     })
118 }
119
120 readData().then(writeData).then((out) => console.log(out));
```

Output

```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
saved!
```


kejkaew.tha@mail.kmutt.ac.th

Read and Write JSON File

- เหมือนกับที่อ่านและเขียน text file

```
123  const fs = require('fs');
124  const readData = () => {
125      return new Promise((resolve,reject) => {
126          fs.readFile('jfile.json','utf8', (err, data) => {
127              if (err)
128                  reject(err);
129              else
130              {
131                  console.log(data);
132                  resolve(data);
133              }
134
135          });
136      })
137  }
```


kejkaew.tha@mail.kmutt.ac.th

```
136  const writeData = (data) => {
137      return new Promise((resolve, reject) => {
138          fs.writeFile('new_jfile.json', data , (err) => {
139              if (err)
140                  reject(err);
141              else
142                  resolve("saved!")
143          });
144      })
145  }
146
147  readData().then(writeData).then((out) => console.log(out));
```

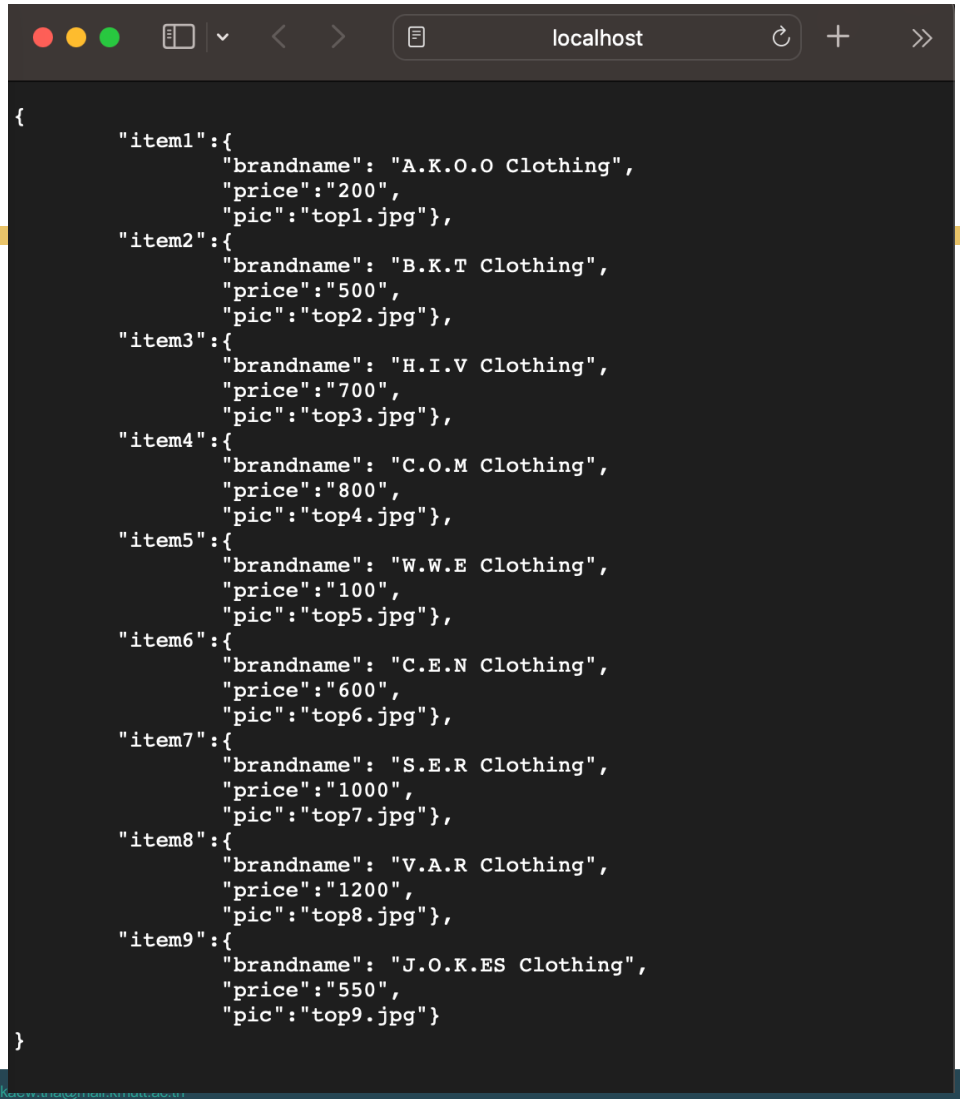
```
(base) Kejkaews-MacBook-Pro:eventloop Kejkaew$ node server.js
{
    "name": "Joe",
    "lastname": "Doe",
    "age": 55
}
saved!
```


kejkaew.tha@mail.kmutt.ac.th

Assignment 9

- assignment9.zip file
- ให้นักศึกษาทำการแก้ไข json file โดยใช้ node.js และ Promise มีขั้นตอนดังนี้
  - อ่านไฟล์ cloth1.json
  - แก้ไขข้อมูล โดยทำการเพิ่ม จำนวนเสื้อผ้าที่เหลืออยู่ในแต่ละ item ตามข้อมูลที่กำหนดให้
  - ทำการ write ข้อมูลที่เป็น object ที่แก้ไขแล้วลงใน file ใหม่ที่ชื่อว่า new_cloth.json และแสดงผลบน browser ตามตัวอย่าง
- ส่งด้วยนะคะ

ผลที่แสดงบน browser



Week 9: Classroom game

- ทำ classroom game ใน LEB2 ก่อนสัปดาห์หน้า

kejkaew.tha@mail.kmutt.ac.th