

วิธีติดตั้ง Node.js

และ npm เบื้องต้น

โดย แอดมินโฮ ोन้อยออก

ประวัติการแก้ไข

ครั้งที่	วันที่	รายละเอียดการแก้ไข
1	5 ธ.ค. 2558	เริ่มสร้าง และเผยแพร่ผลงาน
2	1 ม.ค. 2559	เพิ่มเนื้อหา
3	2 ม.ค. 2559	แก้ไขตามคำแนะนำของ Thitipong Suparurkrat และผู้ไม่ประสงค์เอ่ยนาม
4	8 ม.ค. 2559	เพิ่มเนื้อหา
5	10 เม.ย. 2559	แก้ไขเล็กน้อย
6	27 เม.ย. 2559	แก้ไขให้สอดคล้องกับ Node.js เวอร์ชัน 6 ที่ออกมาใหม่

ถ้าท่านดาวน์โหลดทิ้งไว้นาน แล้วเพิ่งมาเปิดอ่าน ก็ขอรบกวนให้โหลดใหม่อีกครั้งที่

http://www.patanasongsivilai.com/itebook_form.html

เพื่อผมอัปเดตแก้ไข pdf ตัวใหม่เข้าไป หรือใครไปดาวน์โหลดมาจากที่อื่น

ก็อาจพลาดเวอร์ชันใหม่ล่าสุดได้ครับ

และรบกวนช่วย**กรอกแบบสอบถาม** ตามลิงค์ข้างบนด้วยนะครับ

แอดมินโฮ ोन้อยอก

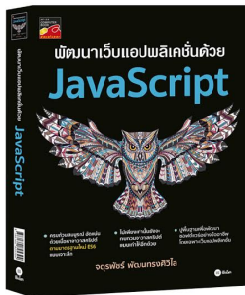
(จตุรพัชร พัฒนทรงศิริไล)

5 ธันวาคม 2558

ถ้าสนใจเกี่ยวกับเพจด้านไอที ก็ติดตามได้ที่ <https://www.facebook.com/programmerthai/>

*EBook เล่มนี้สงวนลิขสิทธิ์ตามกฎหมาย ห้ามมิให้ผู้ใด นำไปเผยแพร่ต่อสาธารณะ เพื่อประโยชน์ในการค้า หรืออื่นๆ โดย
ไม่ได้รับความยินยอมเป็นลายลักษณ์อักษรจากผู้เขียน*

บทนำ

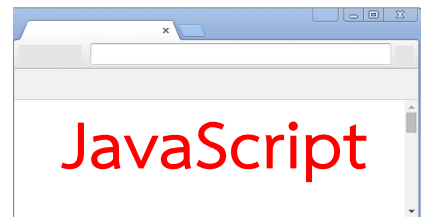


สืบทอดมาจากผมได้แต่งหนังสือ **จาวาสคริปต์ (JavaScript)** มาตรฐานตัวใหม่ **ECMAScript 2015** หรือเรียกสั้น ๆ ว่า **“ES6”** หรือ **“ES6 Harmony”** ซึ่งประกาศออกมาล่าสุด เมื่อกลางเดือนมิถุนายน พ.ศ. 2558 (โดยเล่มนี้ตีพิมพ์และจัดจำหน่ายโดยซีเอ็ด) ส่วนในปี 2559 ก็จะเป็น ES7 (เพิ่มฟีเจอร์ใหม่เข้ามาทีเดียว)

สำหรับคู่มือเล่มนี้ผมแต่งขึ้นมา ก็เพื่อใช้ประกอบหนังสือตามที่กล่าวมาข้างต้นนั่นเองครับ และถ้าเกิดเนื้อหาอะไรผิดพลาดไป เช่น ให้ข้อมูลผิด สะกดอะไรผิดไป อ่านแล้วมันงงไป 7 วัน เป็นต้น ผมก็ขออภัยมา ณ โอกาสนี้ด้วย และท่านก็สามารถชี้แนะผมได้ตลอดเวลา

แนะนำ Node.js ก่อน

ต้องอธิบายอย่างนี้นะครับ จาวาสคริปต์ที่คนส่วนใหญ่รู้จักจะทำงานอยู่ฝั่งเว็บเบราว์เซอร์ ซ้ายไหมละตัวอย่างเช่น ทำงานอยู่บน Internet Explorer (IE), Firefox และ Google Chrome เป็นต้น



ส่วน Node.js ถ้าอธิบายสั้น ๆ ง่าย ๆ มันคือตัวรันไทม์ (Runtime) ภาษาจาวาสคริปต์ โดยไม่ต้องพึ่งพาเว็บเบราว์เซอร์

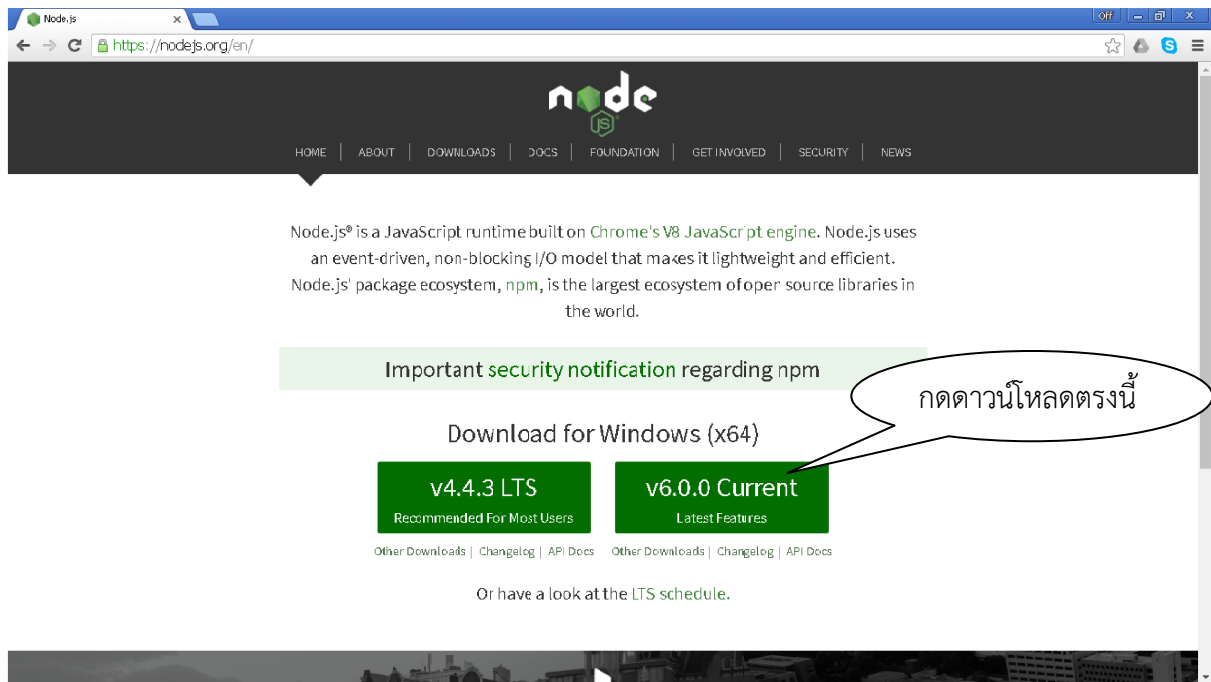
ส่วนใหญ่จะนิยมนำ Node.js ไปทำงานฝั่งเซิร์ฟเวอร์ ด้วยเหตุนี้การพัฒนาเว็บแอปพลิเคชันยุคใหม่จึงสามารถใช้จาวาสคริปต์ล้วน ๆ (เรียกว่า “Full stack JavaScript”) โดยไม่ต้องพึ่งพาภาษาสคริปต์ดัง ๆ เช่น PHP, ASP และ JSP เป็นต้น (หน้าจอบริบทยังต้องพึ่งพา HTML กับ CSS)

ส่วน npm เดี่ยวจะอธิบายทีหลัง เพราะถ้าติดตั้ง Node.js ตัว npm ก็จะถูกติดตั้งมาให้พร้อมกันด้วย

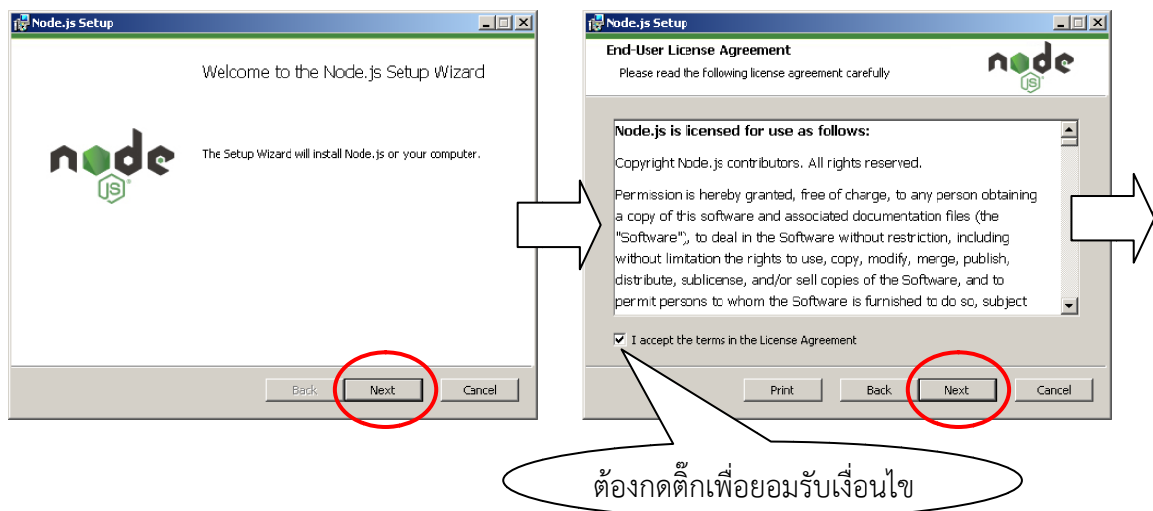
วิธีติดตั้ง Node.js และ npm เบื้องต้น

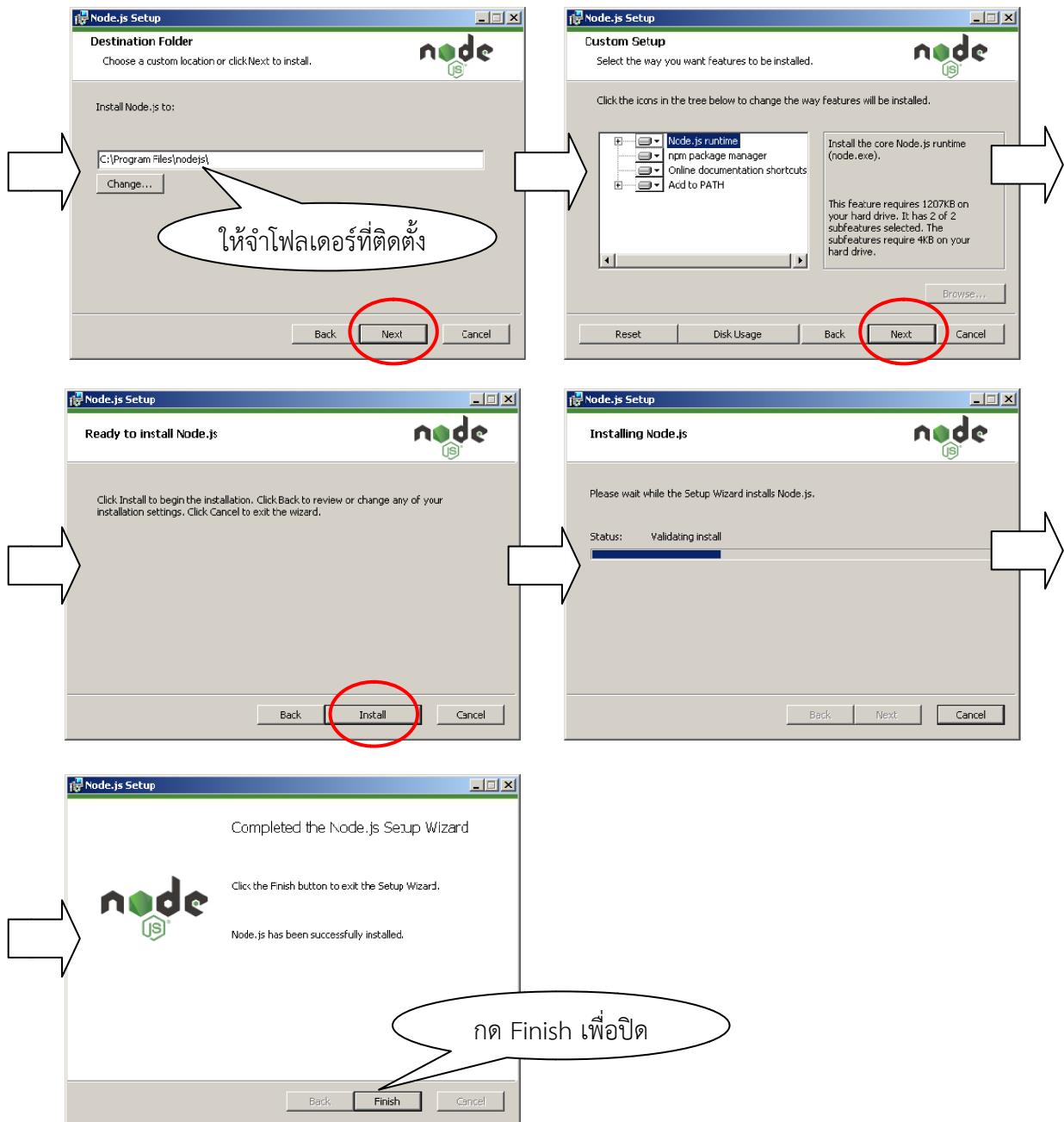
คู่มือเล่มนี้จะแสดงตัวอย่างวิธีติดตั้ง Node.js เวอร์ชัน 6 บนวินโดวส์เท่านั้น และถ้าในอนาคตมีเวอร์ชันใหม่กว่านี้ ก็ให้ติดตั้งทำนองเดียวกัน ตามรายละเอียดดังนี้

- 1) ให้ไปที่หน้าเว็บ <https://nodejs.org/en/> แล้วดาวน์โหลดไฟล์ติดตั้งตามภาพ



- 2) เมื่อดาวน์โหลดไฟล์ติดตั้งเสร็จเรียบร้อยแล้วเป็น `node-v6.0.0-x64.msi` ก็ให้กด Next ไปเรื่อย ๆ แบบเดียวกับการลงโปรแกรมปกติธรรมดา (โดยใช้ค่าดีฟอลต์ที่กำหนดมาให้หมด) ดังตัวอย่าง





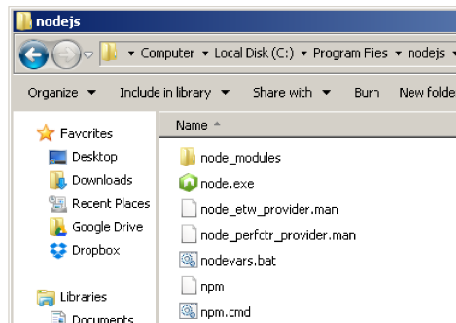
- 3) เมื่อติดตั้ง Node.js เสร็จเรียบร้อยแล้ว ก็ให้เราไปที่ปุ่มเมนู “Start” ของวินโดวส์ มันจะมีโปรแกรม “Node.js” โผล่ขึ้นมา และให้คลิกเพื่อลองรันโปรแกรม ซึ่งจะปรากฏหน้าต่างภาพหน้าถัดไป



ภาพโปรแกรมหน้าจอดำ ๆ ที่เห็น มันคือตัวอินเทอร์พรีเตอร์ (Interpreter) ของ Node.js ซึ่งมีไว้ให้เราพิมพ์คำสั่งในภาษาจาวาสคริปต์ และเราก็สามารถทดสอบการใช้งานง่าย ๆ ด้วยการพิมพ์คำว่า "Hello, world!" ซึ่งควรจะแสดงผลดังภาพ

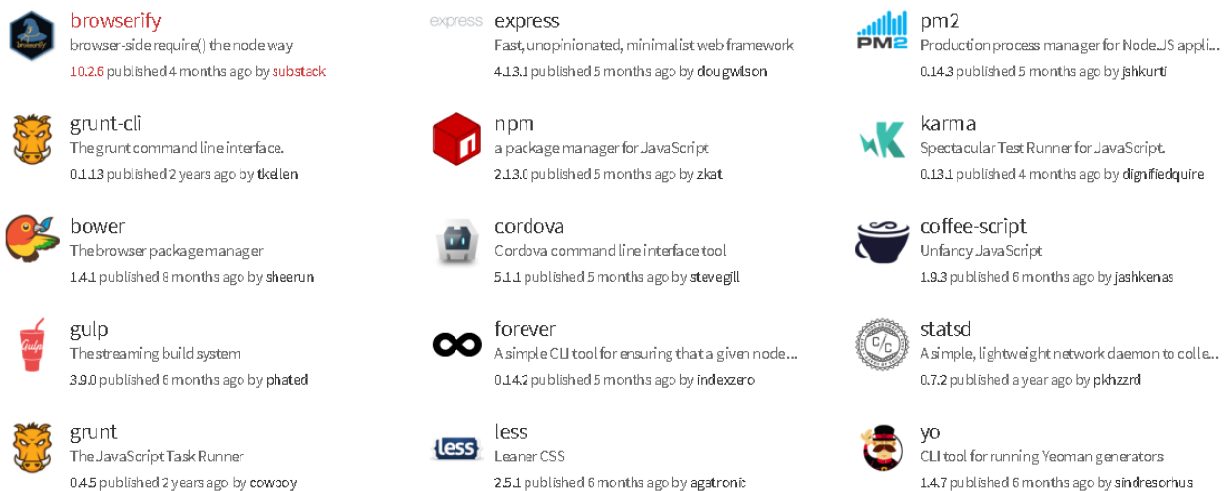
```
Node.js
> "Hello, world!"
'Hello, world!'
>
```

4) ถ้าคุณลองไปโฟลเดอร์ที่ติดตั้ง Node.js ตามภาพ



อย่างกรณีนี้เครื่องของผู้เขียน โฟลเดอร์ที่ติดตั้ง Node.js จะอยู่ที่ “C:\Program Files\nodejs” ซึ่งจะเห็นไฟล์สำคัญ 2 ตัว ได้แก่ node.exe กับสคริปต์ npm.cmd (สำหรับไฟล์ npm จะเป็นสคริปต์ที่ใช้รันบน OS X หรือ Linux) โดยพวกมันจะมีความหมายดังนี้

- node.exe มันก็คือตัวอินเทอร์พรีเตอร์ในข้อ 3 ที่เอาไว้ใช้รันไฟล์จาวาสคริปต์
- npm.cmd จะเป็นสคริปต์เอาไว้รันโปรแกรม “npm” ซึ่งเป็นตัวบริหารจัดการแพ็คเกจ พร้อมทั้งใช้ดาวน์โหลดและติดตั้งแพ็คเกจต่าง ๆ ซึ่งแพ็คเกจพวกนี้จะเป็นโปรเจกโอเพ่นซอร์ส



ภาพตัวอย่างแพ็คเกจ ที่ต้องใช้ npm ในการบริหารจัดการ

(ที่มา <https://www.npmjs.com/#getting-started> เข้าถึงเมื่อ 22 พ.ย. 2558)

หมายเหตุ ถ้าไปที่เว็บไซต์หลักทางการ <https://nodejs.org/en/> ก็จะเห็นนิยามของ npm ดังนี้

Node.js' package ecosystem, **npm**, is the largest ecosystem of open source libraries in the world.

ถ้าแปลตรงตัว npm (Node Package Manager) คือระบบนิเวศแพ็คเกจของ Node.js และเป็นระบบนิเวศใหญ่ที่สุดของไลบรารีโอเพ่นซอร์สในโลกแห่งนี้ (แปลดูอาจตลกหย่อนนะครับ)

- 5) ให้คุณลองทำการทดสอบ ด้วยการรันโปรแกรม node.exe รวมทั้ง npm.cmd บนคอมพิวเตอร์ของคุณ วินโดวส์ เพื่อดูเลขเวอร์ชันของ Node.js กับ npm ตามลำดับ ดังภาพ

```
C:\>node -v
v6.0.0

C:\>npm -v
3.8.6
```

จากภาพที่เห็น ถ้าคุณพิมพ์คำสั่ง แล้วมันแสดงเลขเวอร์ชันออกมา แสดงว่าเราติดตั้ง Node.js และ npm ได้สำเร็จแล้ว (ถ้าในอนาคตคุณติดตั้งเป็นเวอร์ชันที่ใหม่กว่านี้ ก็จะแสดงเป็นเลขอื่นได้)

หมายเหตุ ภาพที่เห็นเราสามารถพิมพ์คำสั่ง node กับ npm ที่ใดเรคเทอร์ไหนก็ได้ เพราะมันจะถูกกำหนด path ให้ชี้ไปยังโฟลเดอร์ในข้อ 4 โดยอัตโนมัติ

ตัวอย่างการใช้งาน Node.js

มาดูตัวอย่างโค้ดจาวาสคริปต์ที่จะใช้รันบน Node.js ซึ่งผมนำมันมาจาก <https://nodejs.org/en/about/> (เข้าถึงเมื่อ 27 เมษายน 2559)

```
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}/`);
});
```

ในโค้ดที่ผมตีกรอบไฮไลต์สีแดงเอาไว้ มันคือไวยากรณ์ใน ES6 ซึ่งจะมีรายละเอียดดังตาราง

ไวยากรณ์ใน ES6	ความหมาย
const	เป็นการประกาศตัวแปรค่าคงที่ ห้ามเราแก้ไขค่าภายหลังเด็ดขาด และตัวแปรจะมีขอบเขตการมองเห็น ตั้งแต่จุดที่ประกาศใช้งานครั้งแรก
=>	มันคือฟังก์ชันลูกศร (Arrow function) หรือก็คือ Lambda expression ที่มีใช้ในหลาย ๆ ภาษา เช่น <ul style="list-style-type: none">● ถ้าเป็นใน Java 8 ขึ้นไป จะใช้เครื่องหมาย ->● ถ้าเป็น C# จะใช้เครื่องหมาย =>● ถ้าเป็น python จะใช้คีย์เวิร์ด lambda
เทมเพลตสตริง (Template String)	<code>`Server running at http://\${hostname}:\${port}/`</code> // จะเสมือนเขียนเป็น // <code>'Server running at http://' + hostname + ':' + port + '/'</code> // จะได้ผลลัพธ์เป็น <code>'Server running at http://127.0.0.1:3000/'</code> ข้อความในตัวอย่างที่เห็นเรียกว่าเทมเพลตสตริง ซึ่งให้คุณคิดว่ามันคือสตริงแล้วกัน แต่ทว่ามันจะใช้เครื่องหมาย back-tick (``) ที่คล้ายกับเครื่องหมายคำพูดแต่ยาว

	<p>แต่มันจะเอนไปทางซ้ายมือคนอ่านสักเล็กน้อย</p> <p>ให้สังเกตตรงตามตัวอย่าง จะสามารถมีนิพจน์เหล่านี้ได้อยู่ได้</p> <ul style="list-style-type: none"> • <code>{hostname}</code> และ <code>{port}</code> <p>ซึ่งนิพจน์ดังกล่าวก็จะถูกแทนที่ด้วย</p> <ul style="list-style-type: none"> • <code>{hostname}</code> จะถูกแทนที่ด้วยค่าในตัวแปร <code>hostname</code> ซึ่งมีค่าเป็น <code>'127.0.0.1'</code> • <code>{port}</code> จะถูกแทนที่ด้วยค่าในตัวแปร <code>port</code> ซึ่งมีค่าเป็น <code>3000</code>
--	--

ขอพูดถึงฟังก์ชันลูกศรใน ES6 มันก็คือฟังก์ชันไร้ชื่อ (Anonymous function) แต่ทว่ามันจะมีรายละเอียดเชิงลึกต่างจากฟังก์ชันไร้ชื่อทั่ว ๆ ไปในจาวาสคริปต์พอควร (เล่มนี้ยังไม่ขอกล่าวถึงนะครับ)

และถ้าผมใช้ฟังก์ชันลูกศรอธิบายตามเว็บไซต์ของ Node.js คิดว่าคุณน่าจะปวดหัว ตาลายแน่ ๆ โดยเฉพาะใครที่ยังไม่คุ้นเคยกับ ES6 ดีนิก ผมจึงขอแปลงฟังก์ชันลูกศร ให้กลายมาเป็นฟังก์ชันไร้ชื่อธรรมดาดีกว่า ดังนี้

```
const http = require('http');
const hostname = '127.0.0.1';
const port = 3000;
const server = http.createServer(function (req, res) {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});
server.listen(port, hostname, function () {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

ซอร์สโค้ดที่ดัดแปลงมา คุณจะเห็นว่าตำแหน่งที่แต่เดิมเป็นฟังก์ชันลูกศร ผมได้แทนที่ด้วยฟังก์ชันคอลแบ็คไร้ชื่อธรรมดา

สำหรับรายละเอียดของโค้ดในตัวอย่างนี้ทั้งหมด ถ้าคุณยังง ...ผมก็อยากให้คุณลิ้ม ๆ ไปซะ อย่าเพิ่งสนใจอะไรมาก เดี่ยวรายละเอียดเรื่อง Node.js ผมได้แยกไปเขียนอีกเล่มหนึ่งซึ่งแจกฟรีครับ ...ส่วนไวยากรณ์ ES6 ก็อ่านจากหนังสือที่ผมเขียนไว้วางขาย

จากโค้ดดังกล่าวจะสมมติว่า บันทึกลงเป็นไฟล์ชื่อ “server.js” โดยมีโครงสร้างไฟล์เดอร์ดังนี้

```
C:\ES6>
```

```
|-- server.js
```

จะให้รันไฟล์จาวาสคริปต์บนคอมมานไลน์ ด้วยคำสั่ง

```
node server.js
```

ซึ่งในกรณี ผมใช้ Node.js บนวินโดวส์ ก็จะแสดงข้อความออกมาดังภาพ

```
C:\ES6>node server.js  
Server running at http://127.0.0.1:1337/
```

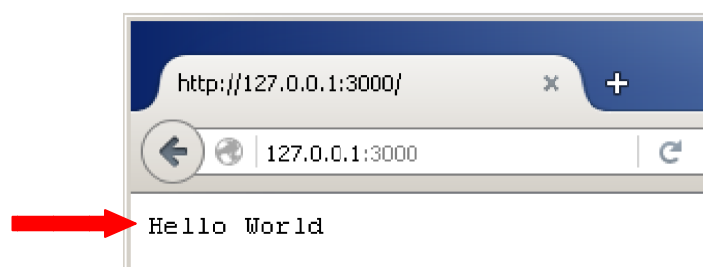
เวลารันคำสั่ง ชื่อไฟล์ server.js ไม่ต้องมีนามสกุล .js ต่อท้ายก็ได้ครับ รันคำสั่งได้เหมือนกัน ดังภาพ

```
C:\ES6>node server  
Server running at http://127.0.0.1:3000/
```

หมายเหตุ ในหนังสือ "พัฒนาเว็บแอปพลิเคชันด้วย JavaScript" ที่ผมเขียน จะใช้แฟล็กเป็น `--use-strict` เพราะเป็นการเขียนจาวาสคริปต์ด้วย ES6 บน Node.js เวอร์ชัน 5 (เวอร์ชันเก่า) จึงต้องระบุแฟล็กตัวนี้ตามภาพข้างล่าง แต่ถ้าเป็น Node.js เวอร์ชัน 6 (ตัวใหม่ล่าสุด) ไม่ต้องระบุแฟล็กตัวนี้

```
C:\ES6>node --use-strict server.js  
Server running at http://127.0.0.1:3000/
```

และถ้าผมลองเปิดเว็บเบราว์เซอร์ขึ้นมา ด้วยการกรอก url เป็น <http://127.0.0.1:3000/> ก็จะมีคำว่า “Hello World” แสดงออกมาดังภาพ ...นั่นหมายความว่า Node.js ทำงานโอเค 😊



*** หมายเหตุ Node.js เวอร์ชัน 6 ในตอนที่ผมแต่งหนังสือ ...มันยังรองรับการใช้งาน ES6 แค่ 93%

จากไฟล์ server.js ของตัวอย่างเดิม ถ้าอยากให้มันรันอยู่ในโหมด debug บน Node.js ก็ให้ใช้คำสั่งดังนี้

```
node debug server.js
```

คำสั่งพื้นฐานที่ควรรู้เมื่อเปิดใช้งานโหมด debug ก็จะมีดังนี้

คำสั่ง	คำอธิบาย
cont หรือ c	จะประมวลผลไปเรื่อย ๆ จนไปหยุดอยู่ที่ตำแหน่ง break point (คำสั่ง debugger)
next หรือ n	เหมือนปุ่ม Step next ใน Developer Tools ทัว ๆ ไป
step หรือ s	เหมือนปุ่ม Step in ใน Developer Tools ทัว ๆ ไป
out หรือ o	เหมือนปุ่ม Step out ใน Developer Tools ทัว ๆ ไป
pause	หยุดรันโค้ด เหมือนปุ่มหยุดใน Developer Tools ทัว ๆ ไป
quit	ออกจากการ debug

คำสั่งมากกว่านี้ลองอ่านเพิ่มเติมได้ที่ (มีเยอะเลย)

- <https://nodejs.org/api/debugger.html>

ผมจะลองเปิดไฟล์ server.js แล้วแก้ไข ด้วยการเพิ่มคำสั่ง **debugger** เข้าไปดังนี้

```
1. const http = require('http');
2. const hostname = '127.0.0.1';
3. const port = 3000;
4. debugger;
5. const server = http.createServer(function (req, res) {
6.   res.statusCode = 200;
7.   res.setHeader('Content-Type', 'text/plain');
8.   res.end('Hello World\n');
9. });
10. server.listen(port, hostname, function () {
11.   debugger;
12.   console.log(`Server running at http://${hostname}:${port}/`);
13. });
```

ตัวอย่างนี้ได้เพิ่มคำสั่ง **debugger** ในบรรทัด 4 และ 11 ก็จะได้เห็นผลการทำงานเวลาอยู่ในโหมด debug ดังภาพข้างล่าง

พิมพ์คำสั่ง

```
C:\ES6>node debug server.js
< Debugger listening on port 5858
connecting to 127.0.0.1:5858 ... ok
break in C:\ES6\server.js:1
> 1     const http = require('http');
    2     const hostname = '127.0.0.1';
    3     const port = 3000;
debug> n
break in C:\ES6\server.js:2
    1     const http = require('http');
> 2     const hostname = '127.0.0.1';
    3     const port = 3000;
    4     debugger;
debug> n
break in C:\ES6\server.js:3
    1     const http = require('http');
    2     const hostname = '127.0.0.1';
> 3     const port = 3000;
    4     debugger;
    5     const server = http.createServer(function (req, res) {
debug> c
break in C:\ES6\server.js:4
    2     const hostname = '127.0.0.1';
    3     const port = 3000;
> 4     debugger;
    5     const server = http.createServer(function (req, res) {
    6         res.statusCode = 200;
debug> c
break in C:\ES6\server.js:11
    9     });
   10     server.listen(port, hostname, function () {
> 11     debugger;
   12         console.log('Server running at http://'+hostname+':'+port+'/');
   13     });
debug> quit
C:\ES6>
```

จะเห็นบรรทัดที่กำลังทำงาน

ออกจาก debug

จากรูป ...ผมเดาว่าคุณคงตาลาย ถ้าใช้โหมด debug ของ Node.js

ทางที่ดีลองหา Developer tools ที่มีเครื่องมือช่วย debug ได้ง่ายกว่านี้ และเท่าที่ผมค้นหาในอินเทอร์เน็ต ก็จะมีตัวอย่างเช่น Visual studio Code ซึ่งทำได้นะ ...ตามลิงค์นี้ครับ

<https://code.visualstudio.com/Docs/editor/debugging>

...และถ้าใครมีเครื่องมือดี ๆ ที่ทำ debug ได้ง่าย ๆ เจ๋ง ๆ ดี ๆ ก็ช่วยบอกกันด้วยน้า

ตัวอย่างการใช้งาน npm

ในหัวข้อนี้จะสาธิตการใช้งาน npm เพื่อติดตั้งแพ็คเกจ โดยผมจะยกตัวอย่างการติดตั้ง Cordova ซึ่งมันเป็นแพลตฟอร์มตัวหนึ่ง ที่เราสามารถใช้ภาษาเขียนเว็บ ได้แก่ จาวาสคริปต์ + HTML + CSS สร้างโมบายแอปพลิเคชัน โดย Cordova มันเลือกได้ว่า จะให้เรา Build เป็นแอปที่รันบนระบบปฏิบัติการอะไร ไม่ว่าจะเป็น iOS, Android หรือ Window phones เป็นต้น ...ส่วนรายละเอียดของมัน ถ้าสนใจลองอ่านได้ที่

- <https://cordova.apache.org/>

สำหรับตัวอย่างการใช้งาน npm เบื้องต้น ก็จะเป็นดังนี้

ให้ติดตั้งแพ็คเกจ cordova ลงไป ด้วยคำสั่ง

```
npm install cordova -g
```

ระหว่างที่คำสั่งนี้กำลังทำงาน ก็ขอให้ต่ออินเทอร์เน็ตเอาไว้ด้วย ซึ่งอาจต้องรอเวลาติดตั้งสักครู่หนึ่ง เมื่อคำสั่งทำงานเสร็จแล้ว ก็ควรจะเห็นไฟล์ติดตั้งทั้งหมด ถูกดาวน์โหลดมาเก็บไว้ที่ **รูท** (Root) ดังโฟลเดอร์ข้างล่าง

```
C:\Users\username\AppData\Roaming\npm
```

***เมื่อ username คือชื่อโฟลเดอร์ของผู้ใช้งานบนวินโดวส์

แต่ถ้าเป็น OS X หรือ Linux โฟลเดอร์ที่เป็นรูทจะอยู่ที่

```
/usr/local/share/npm
```

ติดตั้งแพ็คเกจไว้ในไดเรกทอรีปัจจุบัน

ในตัวอย่างนี้จะแสดงวิธีติดตั้งแพ็คเกจไว้ในไดเรกทอรีปัจจุบัน โดยสมมติว่าเป็น “c:\cordova” ดังนั้นผมจึงต้องสร้างโฟลเดอร์ขึ้นมาก่อน แล้วถึงจะ cd เข้าไปด้วยคำสั่งดังนี้

```
C:\> mkdir cordova
```

```
C:\>cd cordova
```

```
C:\cordova>
```

ควรเปิดคอมมานไลน์บนวินโดวส์

ด้วยสิทธิแบบ **Administrator**

เวลาผมจะติดตั้งแพ็คเกจ ก็เลยเปลี่ยนมาพิมพ์คำสั่งดังนี้

```
C:\cordova>npm install cordova
```

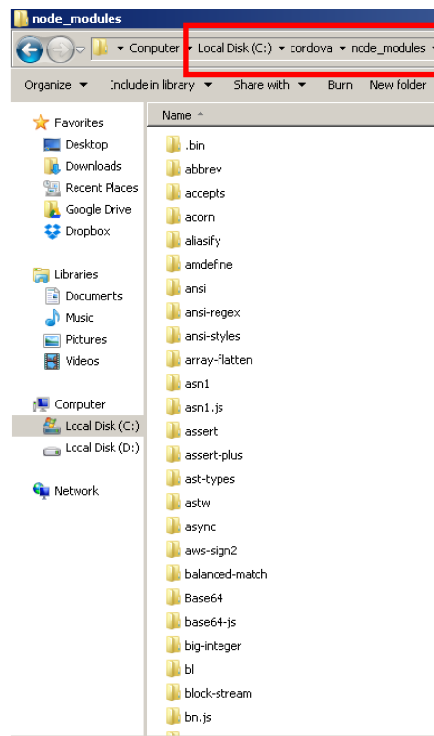
ในตัวอย่างนี้ไม่ได้ใช้แฟล็กเป็น -g ซึ่งจะหมายถึง ให้ดาวน์โหลดไฟล์ติดตั้งมาอยู่ไดเรกทอรีปัจจุบันแทน ไม่ใช่บันทึกเก็บไว้ที่รูท

และเมื่อเปิดโฟลเดอร์ cordova ผมก็จะเห็นโฟลเดอร์ชื่อ node_modules โผล่ขึ้นมาดังนี้ครับ

```
C:\cordova
```

```
|-- node_modules
```

***ถ้าคุณไปเปิดโฟลเดอร์ node_modules ก็จะได้เห็นไฟล์ติดตั้งของ cordova เต็มไปหมด ดังภาพ



มาลองสร้างไฟล์ package.json

คราวนี้ผมจะลองสร้างไฟล์ “c:\cordova\package.json” ด้วยคำสั่ง

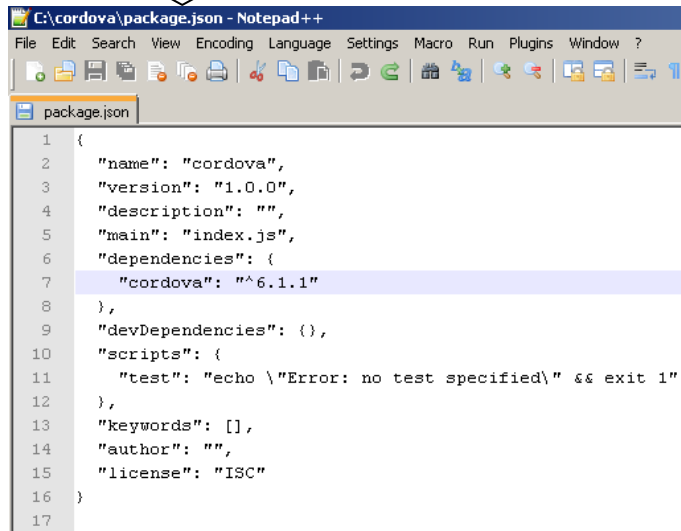
```
C:\cordova>npm init -y
```

หลังจากนั้นก็จะมีไฟล์ package.json ปรากฏขึ้นมาดังนี้ครับ

C:\cordova

|-- node_modules

|-- package.json



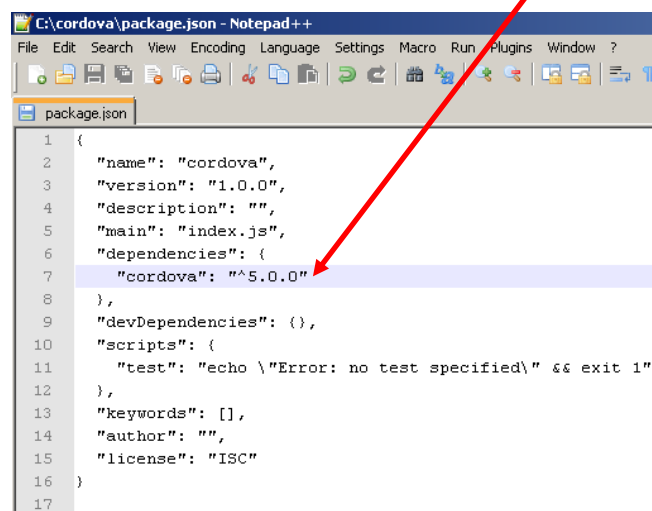
```
1 {
2   "name": "cordova",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "dependencies": {
7     "cordova": "^6.1.1"
8   },
9   "devDependencies": {},
10  "scripts": {
11    "test": "echo \"Error: no test specified\" && exit 1"
12  },
13  "keywords": [],
14  "author": "",
15  "license": "ISC"
16 }
17
```

ตามรูปจะเป็นเนื้อหาของไฟล์ package.json ซึ่งมีโครงสร้างข้อมูลแบบ JSON โดยมันมีไว้ใช้กำหนดค่าเริ่มต้น และช่วยบริหารจัดการโปรเจกได้อย่างดีเยี่ยม

***ถ้าจะเปรียบว่า npm ว่ามันคล้ายกับอะไร เเท่าที่ผมนึกออก ก็คงคล้ายกับ MAVEN ของทาง Java แหะละ

ลองแก้ไฟล์ package.json

ในหัวข้อนี้นผมจะเปิดไฟล์ "package.json" ขึ้นมา แล้วลองแก้ค่า "cordova": "^5.0.0" เข้าไปดังนี้



```
1 {
2   "name": "cordova",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "dependencies": {
7     "cordova": "^5.0.0"
8   },
9   "devDependencies": {},
10  "scripts": {
11    "test": "echo \"Error: no test specified\" && exit 1"
12  },
13  "keywords": [],
14  "author": "",
15  "license": "ISC"
16 }
17
```

ค่าที่ผมแก้เข้าไบนั่น มันเรียกว่า **dependencies** เอาไว้ใช้บอกว่าโปรเจกต์นี้ จะติดตั้งแพ็คเกจอะไรบ้าง และมีเวอร์ชันเป็นอะไร ? ...ซึ่งตัวอย่างนี้ ผมได้ลดเลขเวอร์ชันของ cordova จาก “6.1.1” ให้กลายเป็น “5.0.0”

และถ้าลองลบโฟลเดอร์ “node_modules” ออกไป แล้วพิมพ์คำสั่ง

```
C:\cordova>npm install
```

ซึ่งผลการทำงานก็จะติดตั้งแพ็คเกจ ตามที่เราระบุไว้ใน “package.json” (ประโยชน์ของมันแหละ) โดยผมแค่พิมพ์คำสั่งสั้น ๆ เท่านั้นเอง ...ก็จะได้เห็น node_modules ถูกดาวน์โหลดมาติดตั้งใหม่อีกรอบ

ลองใช้คำสั่ง **npm start**

ก่อนอื่นผมจะก๊อปปี้ไฟล์ “server.js” จากหัวข้อ “ตัวอย่างการใช้งาน Node.js” มาวางไว้ในโปรเจกต์ โดยมีโครงสร้างดังนี้

```
C:\cordova
```

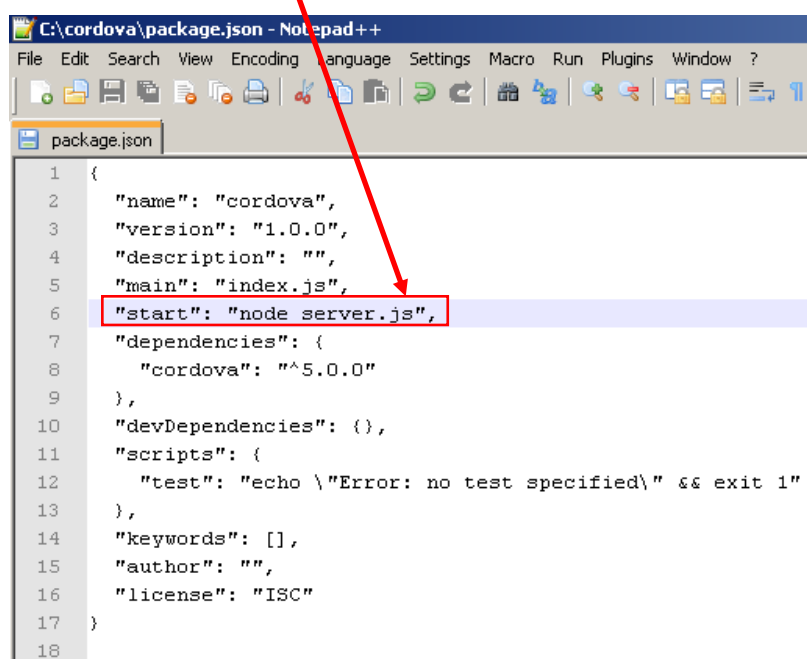
```
-- node_modules
```

```
-- package.json
```

```
-- server.js
```

เพิ่มเข้ามา

จากนั้นก็ให้เพิ่มค่า “start”: “node server.js”, เข้าไปในไฟล์ “package.json” ดังภาพ



```
C:\cordova\package.json - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
package.json
1 {
2   "name": "cordova",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "start": "node server.js",
7   "dependencies": {
8     "cordova": "^5.0.0"
9   },
10  "devDependencies": {},
11  "scripts": {
12    "test": "echo \"Error: no test specified\" && exit 1"
13  },
14  "keywords": [],
15  "author": "",
16  "license": "ISC"
17 }
18
```


ซึ่งค่าที่ผมเพิ่มเข้าไป ก็คือคำสั่งรันไฟล์ server.js บนคอมมานไลน์นั่นเอง

หลังจากนั้น ผมก็จะพิมพ์คำสั่งสั้น ๆ เป็น

```
C:\cordova>npm start
```

ซึ่งมันจะไปรันคำสั่งที่ระบุไว้ใน package.json โดยจะเห็นผลลัพธ์ดังภาพข้างล่าง

```
C:\cordova>npm start
> cordova@1.0.0 start C:\cordova
> node server.js
Server running at http://127.0.0.1:3000/
```

คำสั่งสำคัญที่ควรรู้ของ **npm** (เอาเท่าที่จำเป็นแล้วกันนะ)

รูปแบบคำสั่ง	คำอธิบายโดยย่อ	คำสั่งที่ใช้แทนกันได้
npm -v	แสดงเวอร์ชันของ npm	version (ตัวอย่างการใช้งาน npm -version)
npm init	สร้างไฟล์ package.json เริ่มต้น เพื่อให้ใส่รายละเอียด เพื่อใช้บริหาร โปรเจกของเรา	
npm i package_name	ติดตั้งแพ็คเกจ	install
npm i -g package_name	ติดตั้งแพ็คเกจในระดับ Global (ติดตั้งไว้ที่รูท)	
npm i package_name --save	ติดตั้งแพ็คเกจ พร้อมทั้งอัปเดตไฟล์ package.json (เพิ่มชื่อแพ็คเกจเข้าไป ในพรีออเพอร์ตี้ dependencies)	
npm i package_name --save-dev	ติดตั้งแพ็คเกจ พร้อมทั้งอัปเดตไฟล์ package.json (เพิ่มชื่อแพ็คเกจเข้าไป ใน devDependencies) ประมาณว่าเป็นแพ็คเกจสำหรับให้ นักพัฒนา (Developer ใช้งาน)	

รูปแบบคำสั่ง	คำอธิบายโดยย่อ	คำสั่งที่ใช้แทนกันได้
npm update package_name	อัปเดตแพ็คเกจใหม่	
npm update -g package_name	อัปเดตแพ็คเกจใหม่ในระดับ Global	
npm uninstall package_name	ถอนการติดตั้งแพ็คเกจ	remove, rm, r, un, unlink
npm uninstall -g package_name	ถอนการติดตั้งแพ็คเกจในระดับ Global	
npm uninstall package_name --save	ถอนการติดตั้งแพ็คเกจ (ลบชื่อแพ็คเกจที่อยู่ใน dependencies)	
npm uninstall package_name --save-dev	ถอนการติดตั้งแพ็คเกจ (ลบชื่อแพ็คเกจที่อยู่ใน devDependencies)	
npm ls	แสดงรายการแพ็คเกจที่ได้ติดตั้งทั้งหมด	list, la, ll
npm ls -g	แสดงรายการแพ็คเกจทั้งหมด ซึ่งติดตั้งไว้ที่ระดับ Global	
npm start	ใช้รันสคริปต์ที่ระบุไว้ในพร็อพเพอร์ตี้ start ของไฟล์ package.json ถ้าไม่มีการระบุ ก็จะรันสคริปต์ด้วยคำสั่ง node server.js	

อ่านเพิ่มเติม

สำหรับเรื่อง Node.js และ npm ที่อธิบายเป็นแค่เบื้องต้นเท่านั้น ถ้าสนใจลองอ่านเพิ่มได้ตามลิงค์ข้างล่าง

- <https://nodejs.org/en/about/>
- <https://docs.npmjs.com/>
- <https://booker.codes/how-to-build-and-publish-es6-npm-modules-today-with-babel/>