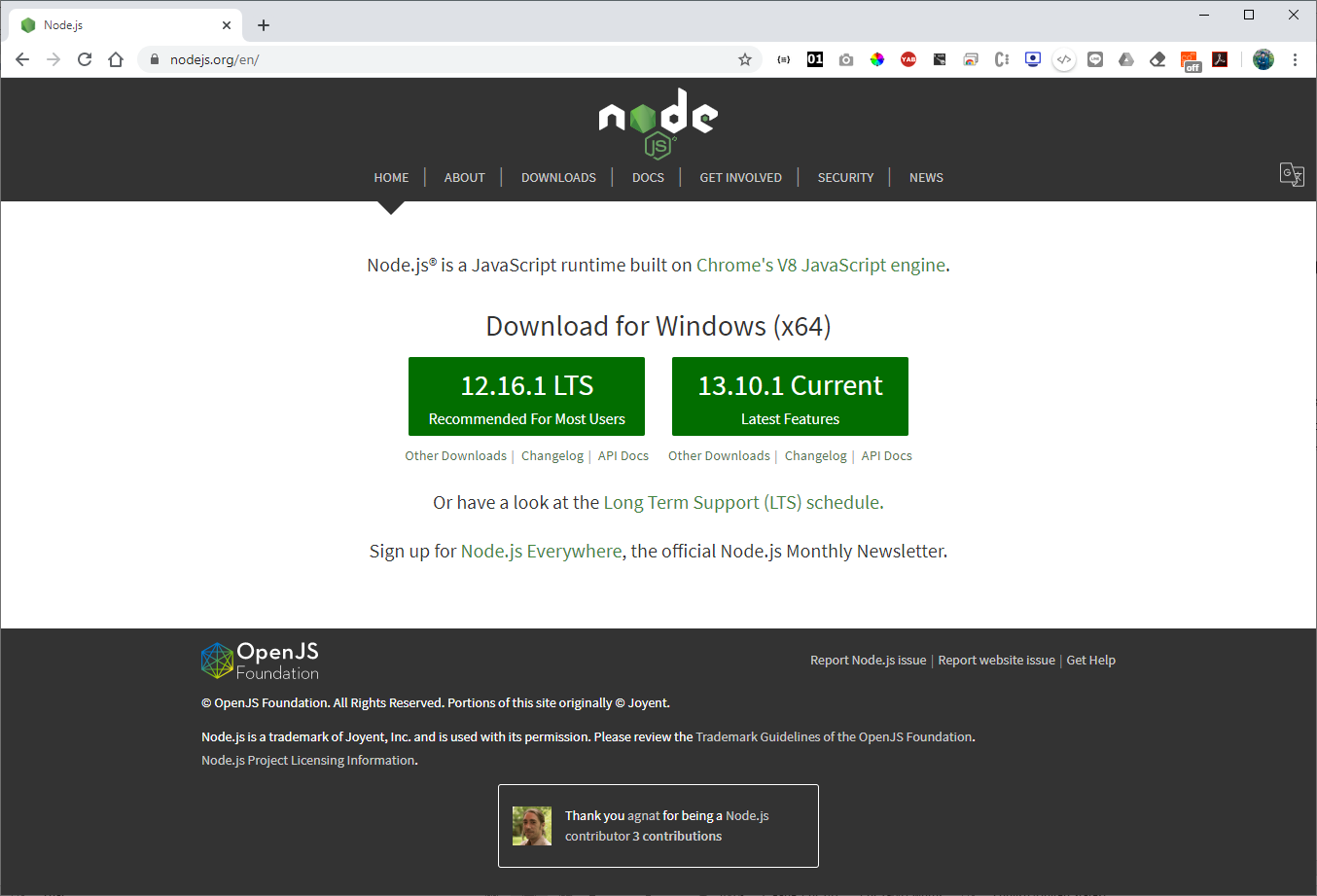
# Chapter 01 ติดตั้งโปรแกรมและจัดเตรียมข้อมูล

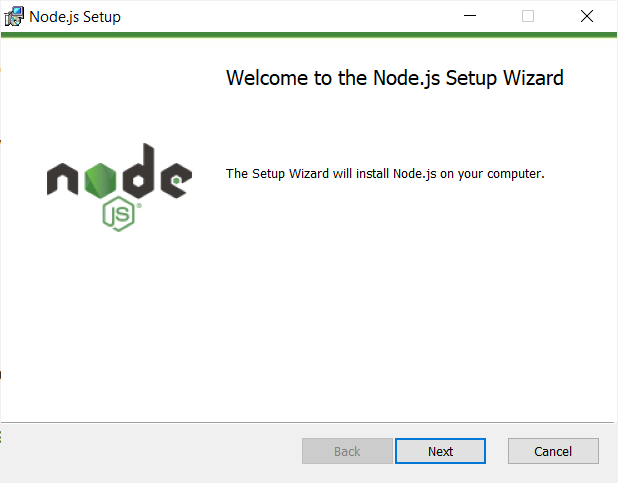
## 1. ติดตั้ง โปรแกรม Node.js

**Node.js** เป็น JavaScript runtime ทำงานบน V8 Engine ของ Google chrome เป็นเทคโนโลยี Server Side ที่เขียนภาษา JavaScript ปัจจุบัน Node.js ถูกนำมาทำเป็น Web Server , Mobile Hybrid , IOT , Webkit , TVOS ,OS อื่น ๆ อีกมาก เรียกได้ว่าเข้าถึงได้หลากหลายเทคโนโลยี

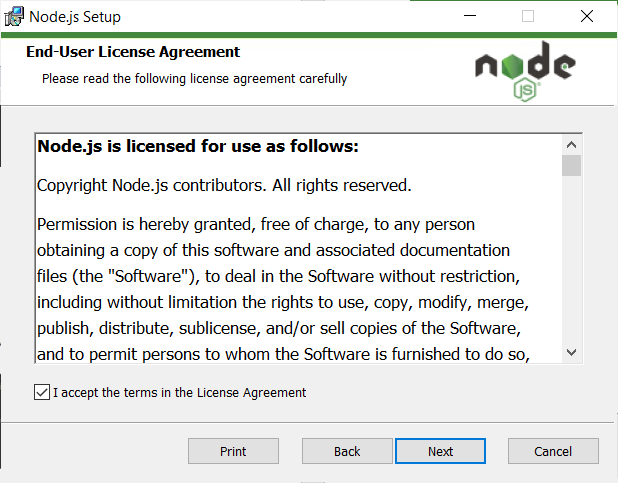
1. ดาวน์โหลดโปรแกรม Node.js จาก https://node.js.org



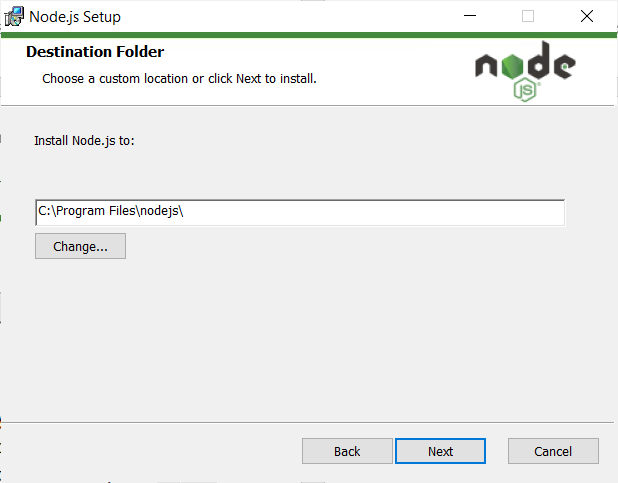
2. คลิกติดตั้งจะปรากฏหน้าต่างติดตั้งของ Node.js แสดงขึ้นมา จากนั้นคลิก ***Next***



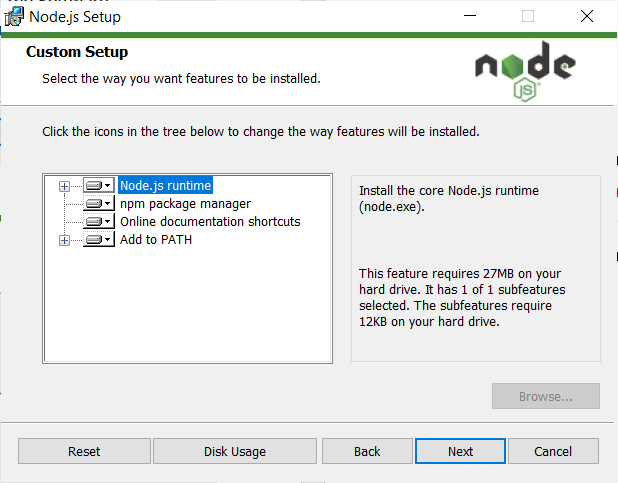
3. ยอมรับ License agreement แล้วคลิก ***Next***



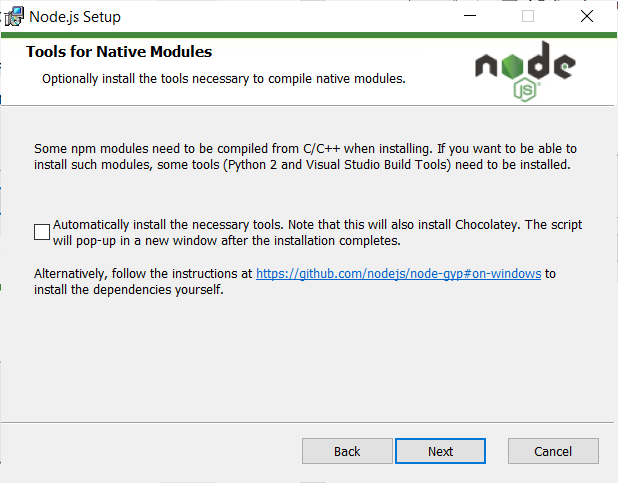
4. ตำแหน่งที่จะติดตั้งบนเครื่อง โดยปกติจะยู่ใน C:\Program Files\nodejs จากนั้นคลิก ***Next***



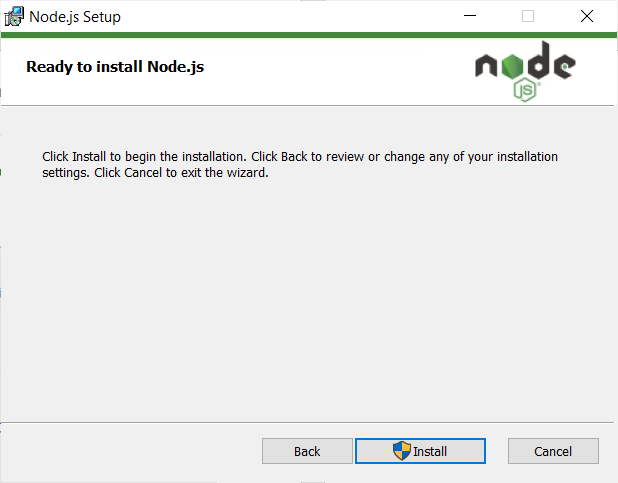
5. รายการให้เลือกติดตั้ง ซึ่งแนะนำให้เลือก Default ที่ระบบมีมาให้ จากนั้นคลิก Next



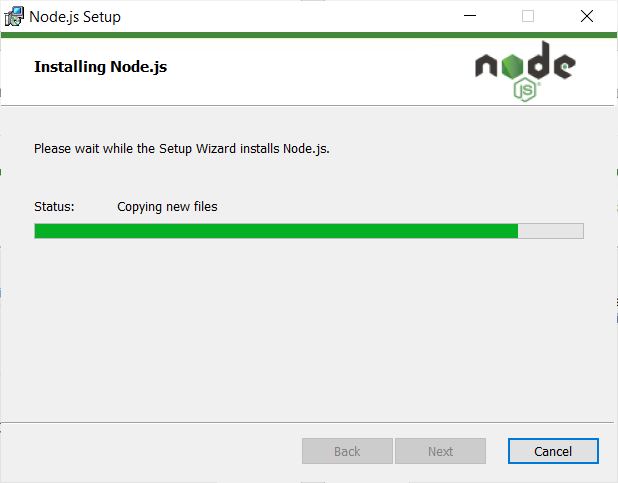
6. เลือกติดตั้ง Module อื่นๆ (ในหนังสือนี้ไม่ติดตั้ง Module อื่น)



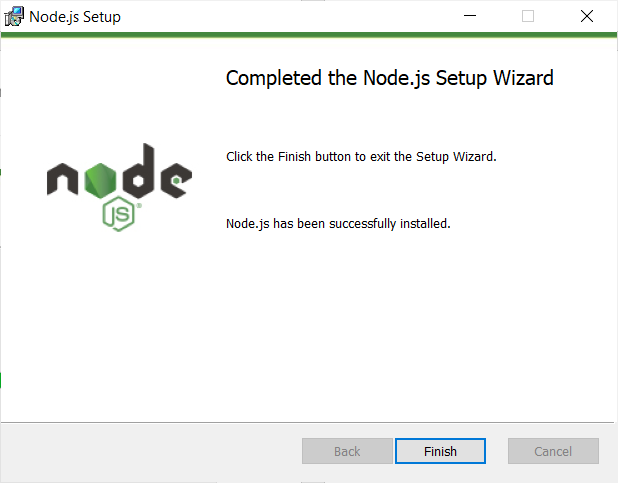
7. คลิก ***Install*** ติดตั้ง Node.js



8. ระบบจะทำการติดตั้ง Node.js ลงในเครี่อง



9. เมื่อติดตั้งสำเร็จจะปรากฏดังภาพด้านล่าง โดยให้คลิกที่ ***Finish*** เพื่อสิ้นสุดการติดตั้งโปรแกรม



10. ตรวจสอบการติดตั้งโดยพิมพ์คำสั่ง node -v และ npm -v ใน Command Prompt หรือ Windows PowerShell เมื่อติดตั้งสำเร็จจะพบเลขเวอร์ชันของ Node.js และ npm ดังนี้

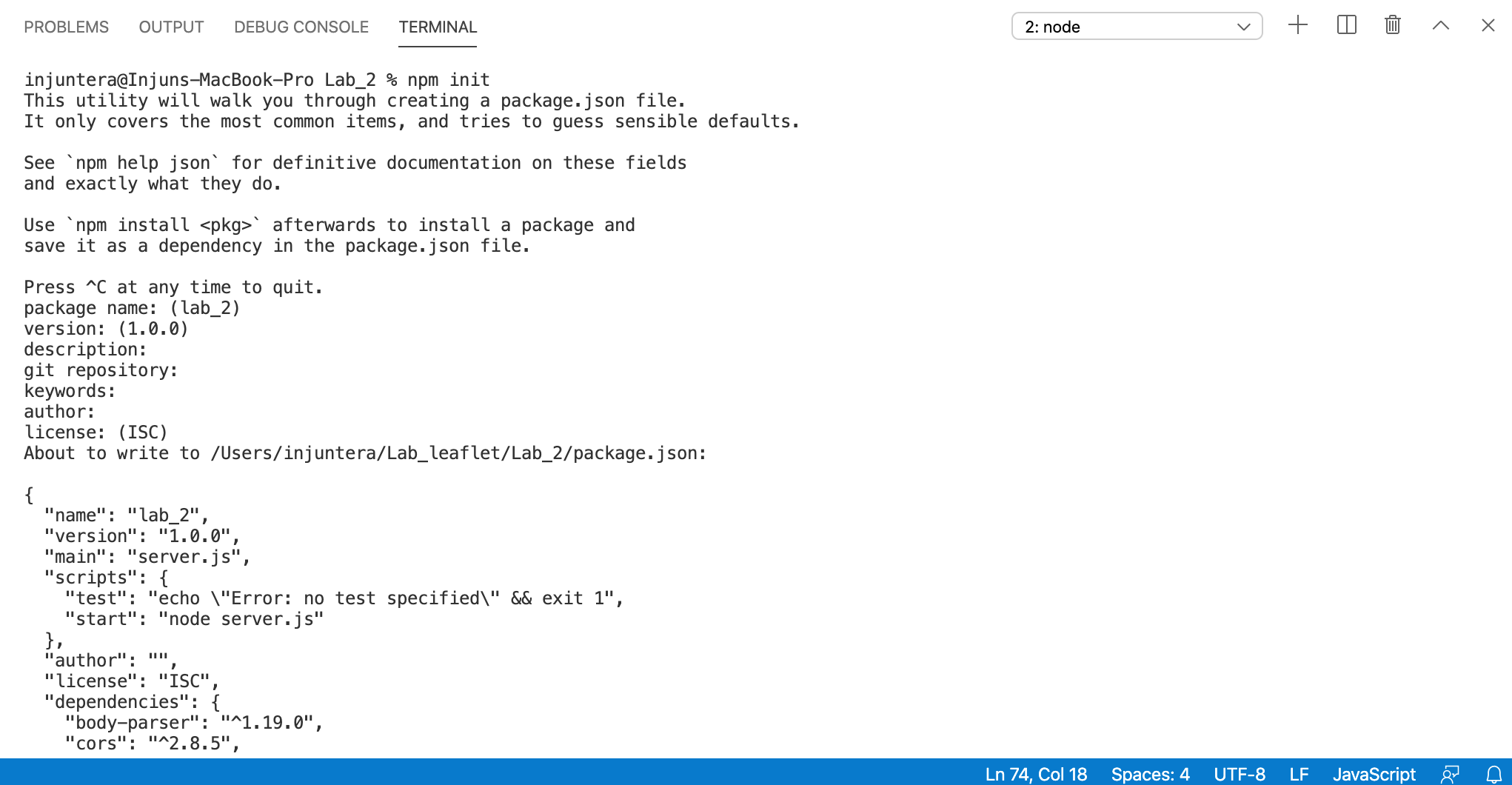


Chapter 04   
Spatial Backend

1. การดึงข้อมูลจาก Database มาใช้งานผ่าน NodeJS และใช้ Postgis ค้นหาข้อมูล

1. เริ่มต้นสร้าง จากนั้นทำการสร้างไฟล์ package.json โดยการพิมพ์คำสั่งใน terminal พร้อมทั้งใส่รายละเอียด package ต่าง ๆ ดังนี้

> npm init



1.1 ติดตั้ง NPM package โดยการพิมพ์คำสั่งใน terminal ลงไปเพื่อเรียก Package ใช้งานภายในเว็บไซต์ ดังนี้

> npm i --save express

> npm i --save cors

> npm i --save pg

> npm i --save body-parser

> npm i --g nodemon

1.2 เปิดไฟล์ server.js จากนั้นพิมพ์คำสั่งเรียกใช้งาน Package ที่ได้ติดตั้งไว้ในตอนแรก ดังนี้

const express = require('express');

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

const cors = require('cors');

const app = express();

1.3 เพิ่มคำสั่งเรียกการใช้งาน cors เพื่อเปิดการใช้งาน cross-Origin Resource Sharing

app.use(cors());

app.options('\*', cors());

1.4 เพิ่มคำสั่งเรียกการใช้งาน bodyparser เพื่อให้อ่านค่า JSON ที่ส่งมาจากเว็บไซต์หลักได้

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({

extended: true

}));

1.5 เพิ่มคำสั่งเรียกใช้งาน pg เพื่อใช้สำหรับเชื่อมต่อกับฐานข้อมูลของ PostgreSQL โดยต้องกำหนดชื่อ Database และ Password เข้าฐานข้อมูลจากตรงนี้ด้วย

const Pool = require('pg').Pool

const db = new Pool({

user: 'postgres',

host: 'localhost',

database: 'dol',

password: '1234',

});

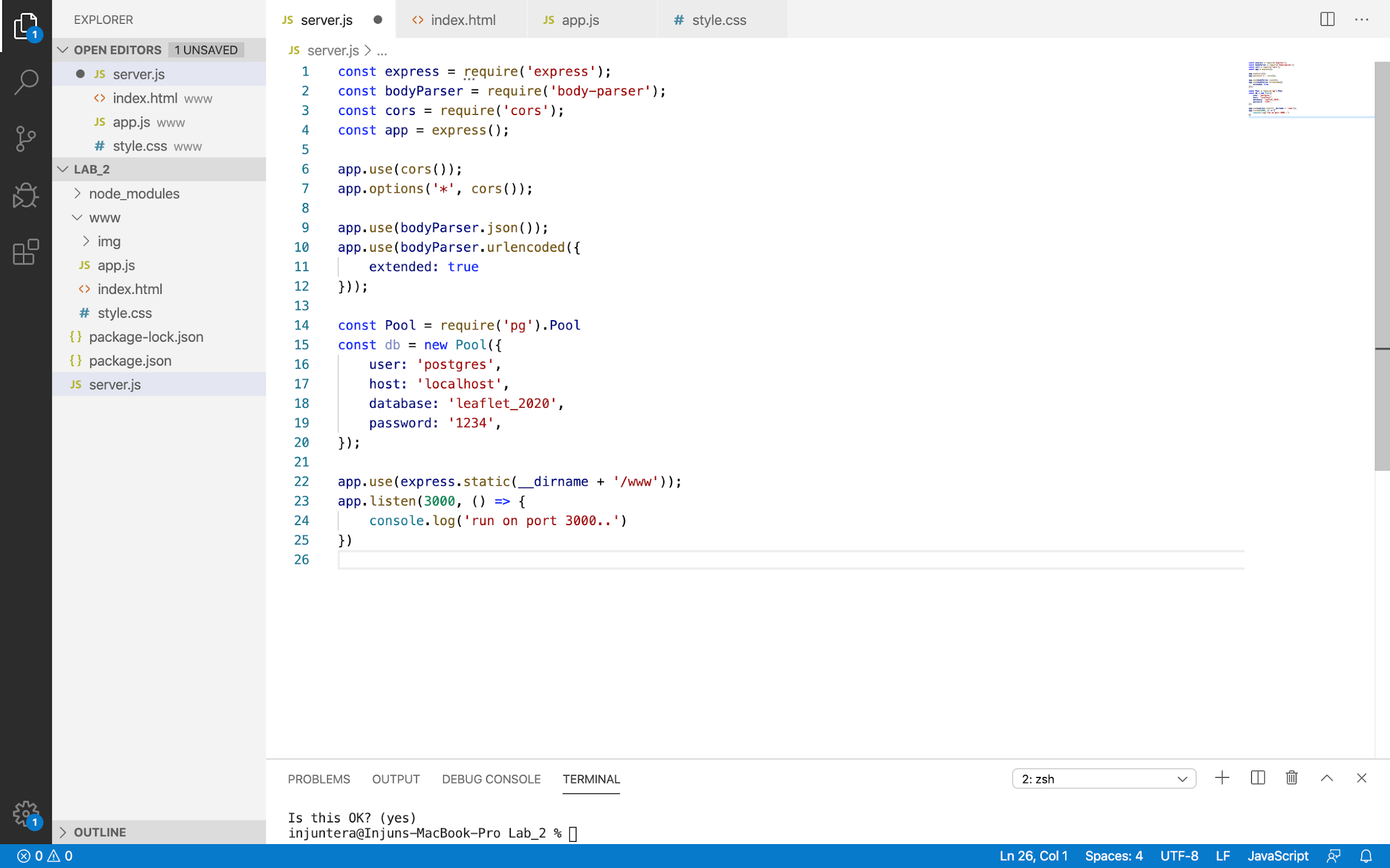
1.6 เพิ่มคำสั่งเรียกใช้งาน express สำหรับใช้สร้าง Webservice ขึ้นมา โดยต้องกำหนด Port ที่ต้องการให้ Run server ในที่นี้จะใช้ Port 3000 ( สามารถเปลี่ยนแปลงได้ ) ซึ่งเราจะกำหนดการแสดงหน้าเว็บไซต์ไปที่ Folder www ดังนี้

app.use(express.static(\_\_dirname + '/www'));

app.listen(3000, () => {

console.log('run on port 3000..')

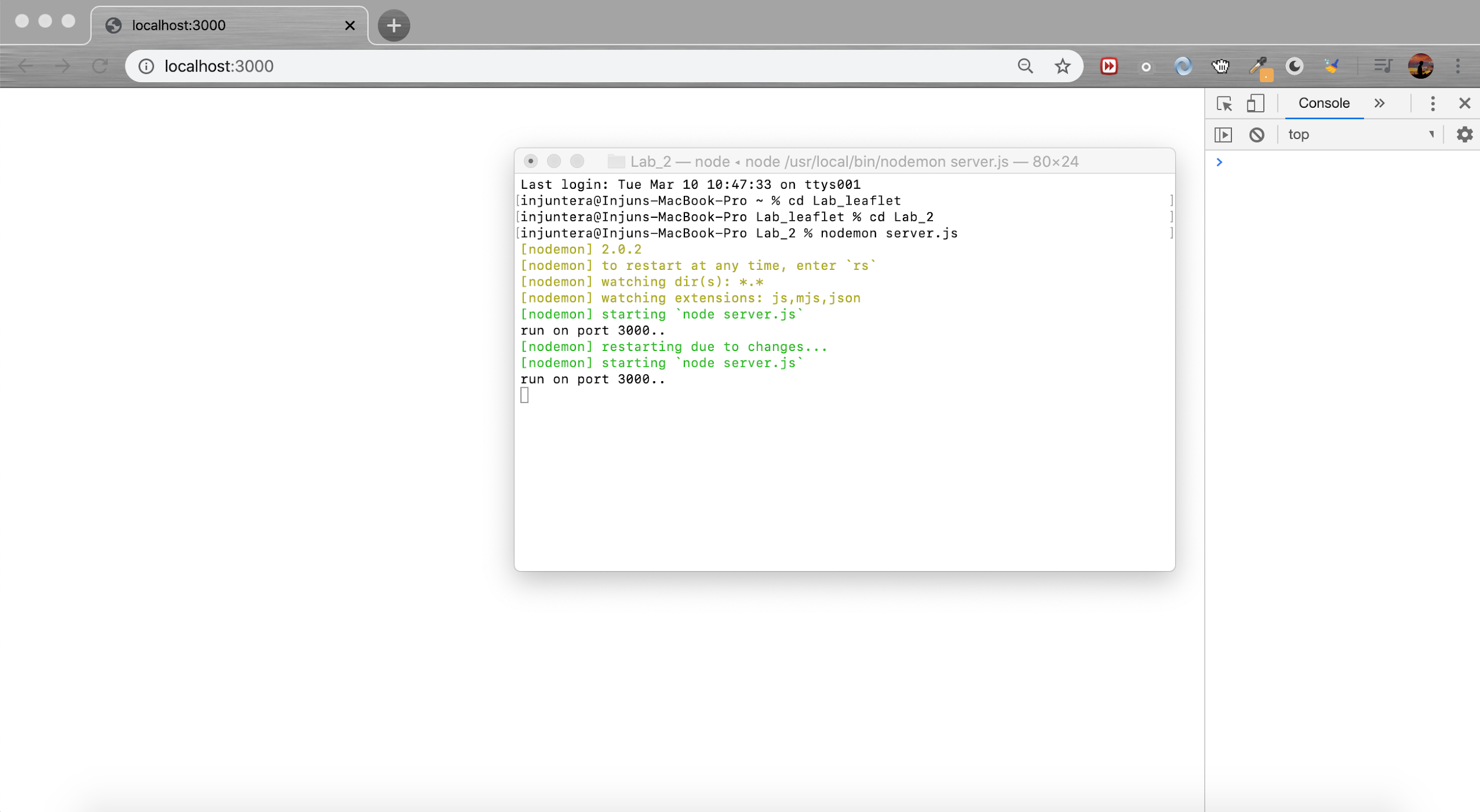
})



1.7 ทำการเขียนคำสั่งใน terminal โดยการใช้ Nodemon ซึ่งจะทำให้การ run ของ nodejs refresh เองโดยอัตโนมัติ ทำให้เราไม่ต้อง start node เองตลอดทุกครั้งที่มีการเปลี่ยนแปลงโค๊ด

> nodemon server.js

เปิด Browser เพื่อแสดงหน้าเว็บโดยการพิมพ์ URL คือ localhost:3000



2. สร้าง Web service สำหรับสร้าง Geojson

2. สร้าง Web service สำหรับสร้าง Geojson ไว้ใช้งานภายในเว็บไซต์ในรูปแบบการ get ข้อมูล ภายในไฟล์ server.js ดังนี้

2.1 เพิ่มคำสั่ง app.get สำหรับสร้างการเรียกข้อมูลแบบ get จากนั้นระบุ path สำหรับใช้เรียกใช้งาน web service ว่า /api\_household ดังนี้

app.get('/amphoe, (req, res) => {

});

2.2 เพิ่มคำสั่ง SQL เพื่อใช้ค้นหาข้อมูลจากฐานข้อมูล โดยการเรียกข้อมูลจากตารางที่ชื่อว่า patient\_households พร้อมทั้งใช้งาน posgis คือ ST\_AsGeoJSON เพื่อสร้าง GeoJSON จากค่า geom ที่ถูกจัดเก็บในฐานข้อมูล ดังนี้

sql = 'SELECT \*,ST\_AsGeoJSON(geom) AS geojson FROM amphoe ';

2.3 เพิ่มคำสั่งสร้างชุดข้อมูลต่าง ๆ ที่ได้จากผลการ Query ให้อยู่ในรูปแบบที่ถูกต้องของ GeoJSON ดังนี้

let jsonFeatures = [];

db.query(sql).then((data) => {

var rows = data.rows;

rows.forEach((e) => {

let feature = {

type: 'Feature',

geometry: JSON.parse(e.geojson),

properties: e

};

jsonFeatures.push(feature);

});

let geoJson = {

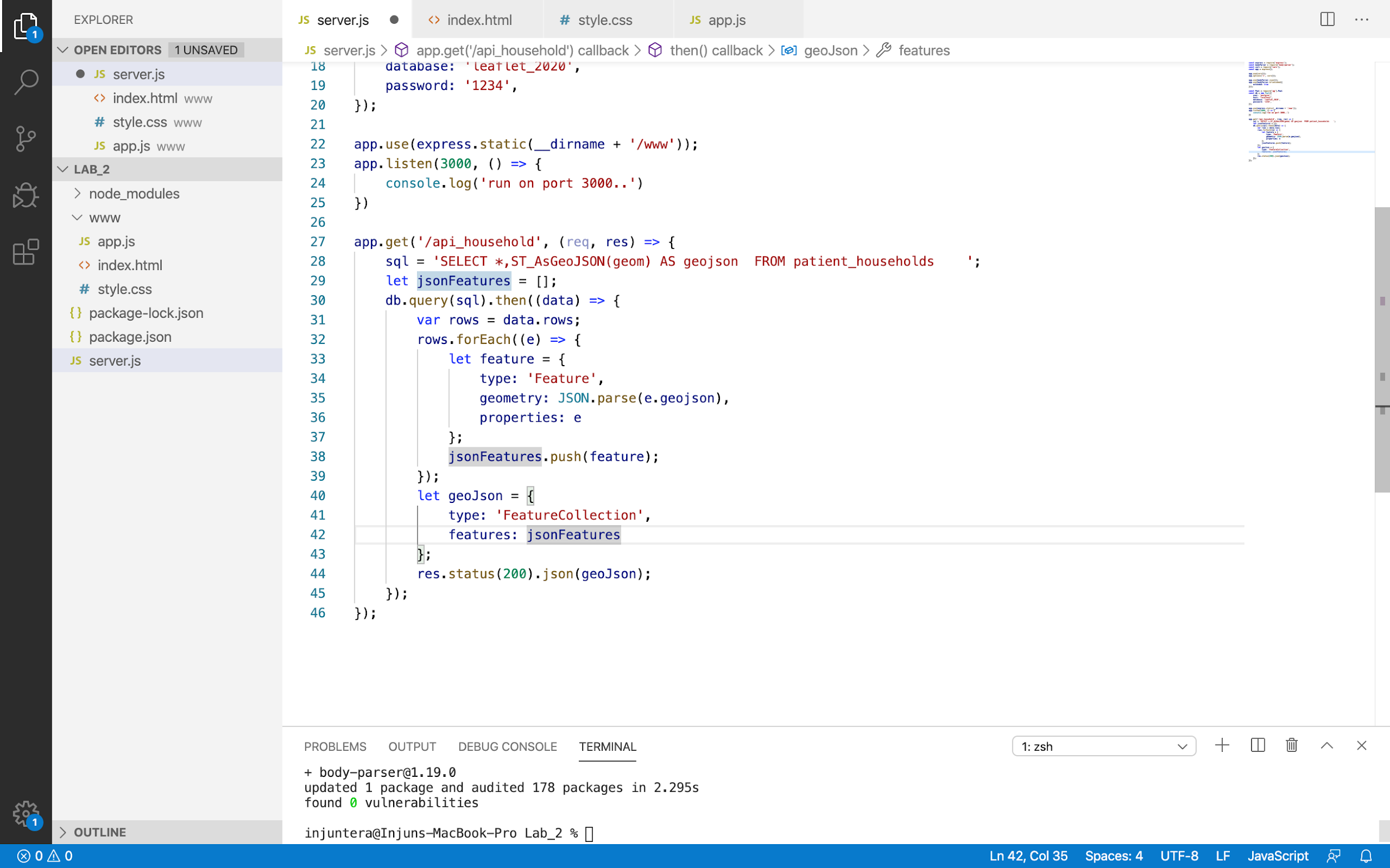
type: 'FeatureCollection',

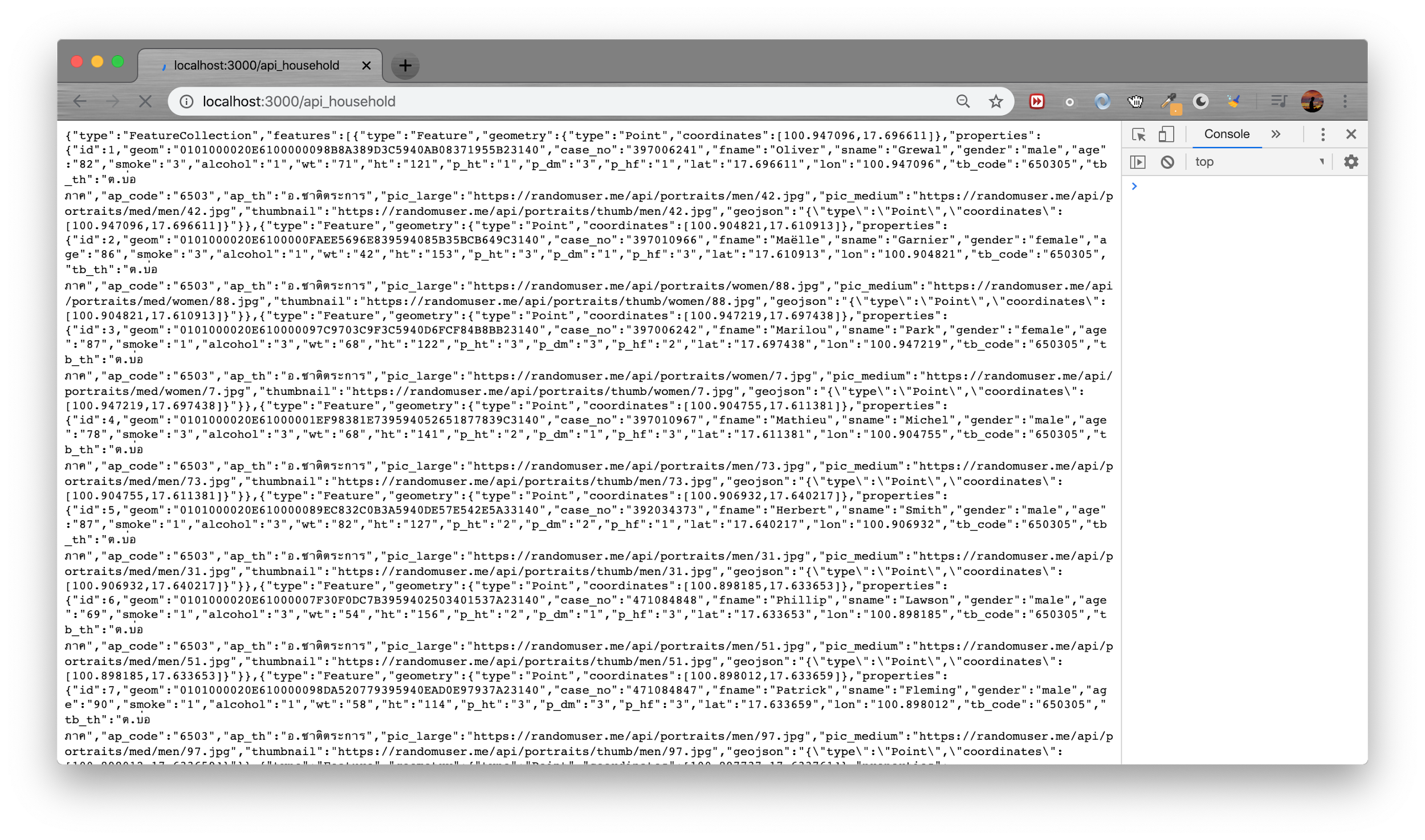
features: jsonFeatures

};

res.status(200).json(geoJson);

});





3. สร้าง Web service และเรียกใช้งาน GeoJSON

6.1 เปิดไฟล์ server.js เพิ่มคำสั่ง app.get สำหรับสร้างการเรียกข้อมูลแบบ get จากนั้นระบุ path สำหรับใช้เรียกใช้งาน web service ว่า /amphoe และเขียน SQL ดังนี้

app.get('/amphoe ', (req, res) => {

sql = 'SELECT \*,ST\_AsGeoJSON(geom) AS geojson FROM amphoe';

let jsonFeatures = [];

db.query(sql).then((data) => {

var rows = data.rows;

rows.forEach((e) => {

let feature = {

type: 'Feature',

geometry: JSON.parse(e.geojson),

properties: e

};

jsonFeatures.push(feature);

});

let geoJson = {

type: 'FeatureCollection',

features: jsonFeatures

};

res.status(200).json(geoJson);

});

});

6.2 เปิดไฟล์ app.js จากนั้นเพิ่มคำสั่งเรียกใช้งาน Web service api\_amphoe ที่ได้สร้างมาแสดงดังนี้

var amphoe = L.layerGroup().addTo(mymap)

var amphoe\_data = []

$.getJSON("/amphoe", function (data) {

L.geoJson(data, {

pointToLayer: function (feature, latlng) {

var hospital\_icon = L.icon({

iconUrl: 'img/amphoe.png',

iconSize: [40, 40]

});

return new L.marker([latlng.lat, latlng.lng], {

opacity: 1,

icon: amphoe\_icon,

});

},

}).addTo(amphoe)

amphoe\_data = data

})