

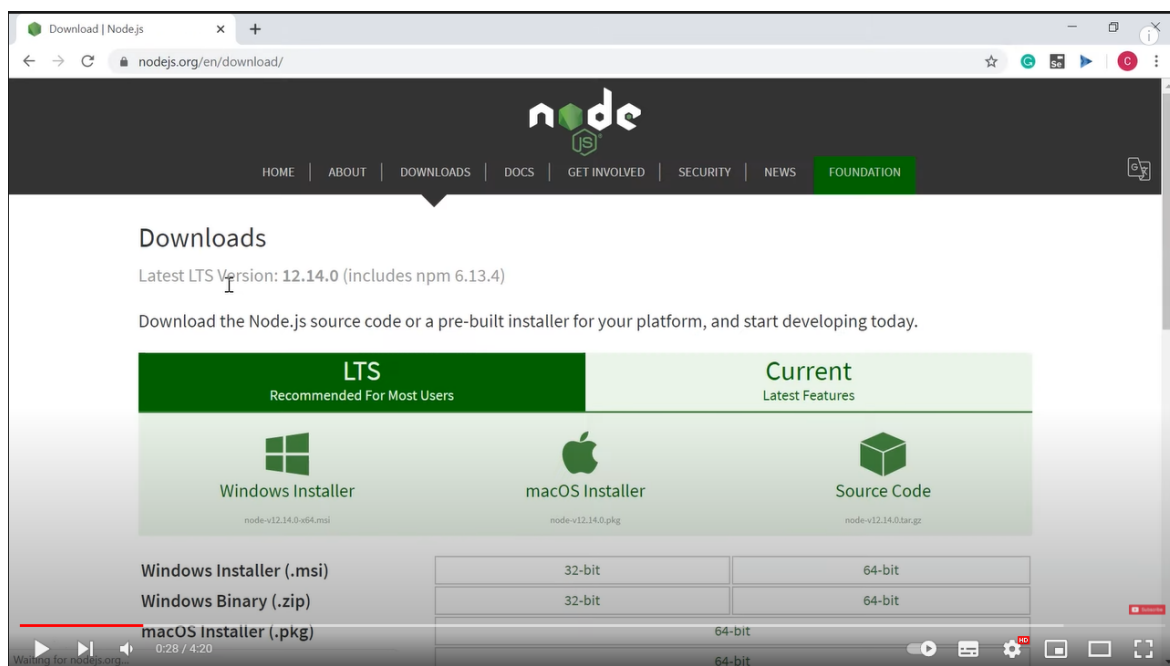
# Workshop #1 GET Method api and API debugging by POSTMAN

## Get Method API

### Topic #1 - การติดตั้ง nodeJS

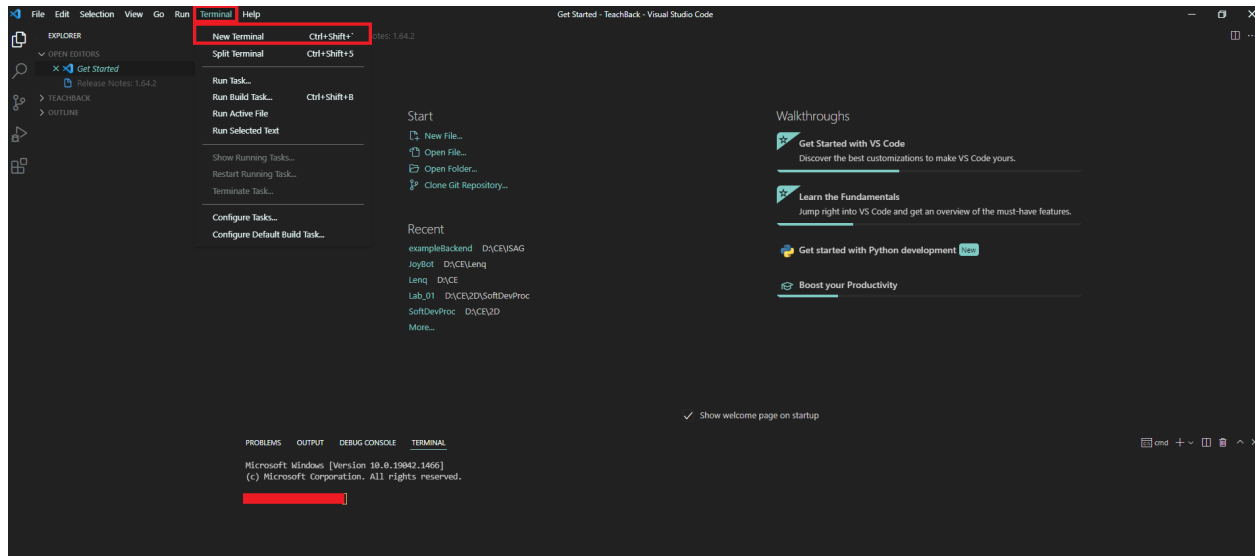
nodeJS เป็น cross-platform runtime environment ของ Javascript ที่ใช้เขียน command-line และ server-side script

เราสามารถติดตั้ง nodeJS ได้จาก [nodejs.dev](https://nodejs.dev) ซึ่งเป็น website ของ nodeJS โดยตรง หรือ ทำตาม video นี้ [youtube.com/watch?v=qYwLOXjAiwM](https://youtube.com/watch?v=qYwLOXjAiwM)



## Topic #2 - การสร้างโปรเจค nodeJS

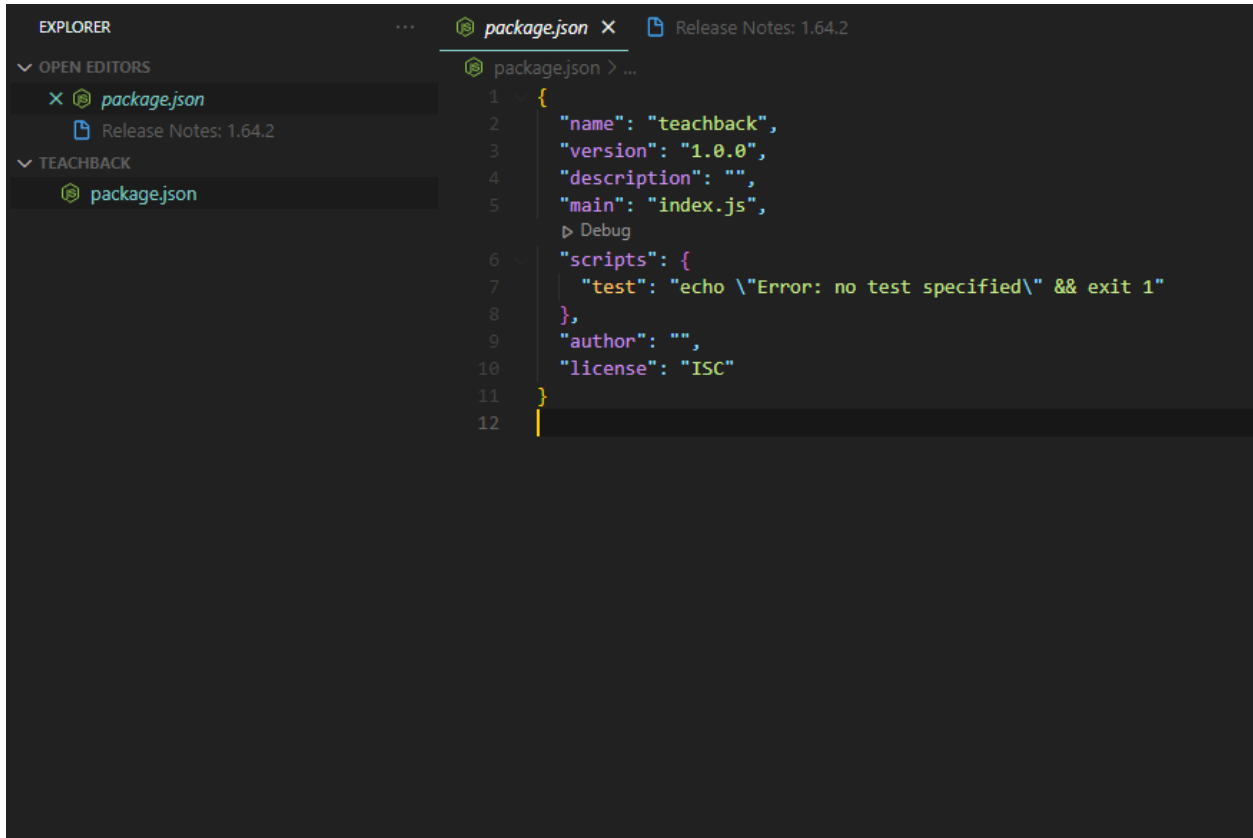
เริ่มจากการสร้าง folder ขึ้นมาเพื่อใช้เก็บ script ต่างๆ ในโปรเจคจากนั้น เปิด cmd ขึ้นมาที่ folder นั้นแล้วพิมพ์ command “code .” เข้าไป หรือ เข้า vscode แล้ว open folder ที่สร้างนั้นขึ้นมา เมื่อเปิดแล้ว ให้เปิด terminal ขึ้นมาใน vscode



เมื่อเปิด Terminal ขึ้นมาแล้วให้พิมพ์คำสั่ง `npm init` เพื่อที่จะได้สร้างโปรเจคของ nodeJS ขึ้นมาใน folder โดยหลังจากพิมพ์แล้วใน terminal จะมีการถามข้อมูลต่างๆของโปรเจคที่จะสร้างขึ้น

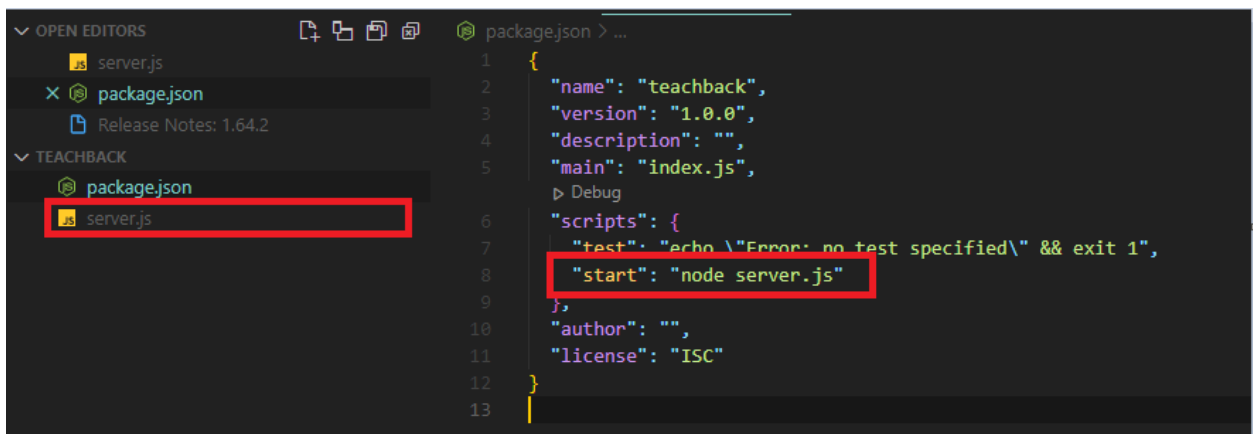


โดยเราก็จะกรอกข้อมูลไปตามที่ต้องการ เมื่อกรอกข้อมูลและกดยืนยันจนครบเราก็จะได้ไฟล์ package.json มาซึ่งสิ่งที่อยู่ในไฟล์ก็จะเป็นข้อมูลที่เราพึงกรอกไป



```
1 {
2   "name": "teachback",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "author": "",
10  "license": "ISC"
11 }
12
```

ซึ่งเราสามารถเพิ่มคำสั่งที่จะรันจากใน terminal ได้ด้วยการเพิ่ม key และ command ที่จะให้ execute ได้ใน object “scripts” เช่น เราจะสร้าง file มาหนึ่ง file เพื่อที่จะเป็นตัวรัน server และเราอยากให้รัน server ผ่าน command “npm start” จะสามารถทำได้โดย



```
1 {
2   "name": "teachback",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1",
8     "start": "node server.js"
9   },
10  "author": "",
11  "license": "ISC"
12 }
13
```

จากในรูปจะเห็นว่ามีการสร้างไฟล์ server.js ขึ้นมาเพื่อเป็นตัวรัน server และใน object “scripts” ใน package.json มีการเพิ่ม key start โดยให้ execute command “node server.js” ซึ่งคือการรันไฟล์ server.js ด้วย command “node” ซึ่งเป็น command ที่ใช้ในการรัน ไฟล์ .js และใน server.js มีโค้ด console.log(“Hi”) อยู่

เมื่อพิมพ์ npm start ลงไปใน terminal จะได้

```
teachback@1.0.0 > npm start

> teachback@1.0.0 start
> node server.js

Hi
```

### Topic #3 - การใช้ express ในการเขียน api

เราจะลง express ด้วยการ ใช้ package manager ของ nodeJS หรือ npm โดยเราจะต้องเข้าไปที่ folder ของ project แล้วพิมพ์ command “npm install express”

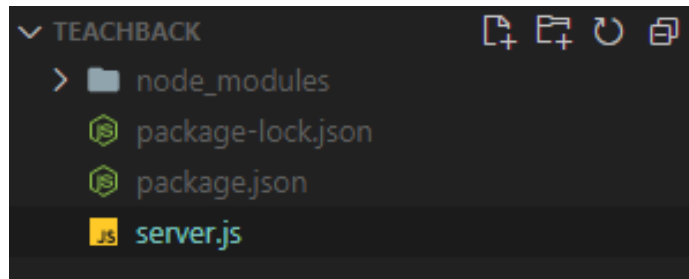
```
teachback@1.0.0 > npm install express
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN teachback@1.0.0 No description
npm WARN teachback@1.0.0 No repository field.

+ express@4.17.2
added 50 packages from 37 contributors and audited 50 packages in 7.048s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

รอนลง express เสร็จแล้วจะมี folder node\_modules และ package-lock.json ขึ้นมาใน folder



ไม่ต้องสนใจบอกเฉยๆ จากนั้นให้เราเข้ามาที่ไฟล์ server.js และเรียกใช้ express ในไฟล์ server.js และ ทำการประกาศ app โดยใช้ express()

```
server.js > [?] app
1  const express = require('express')
2
3  const app = express()
```

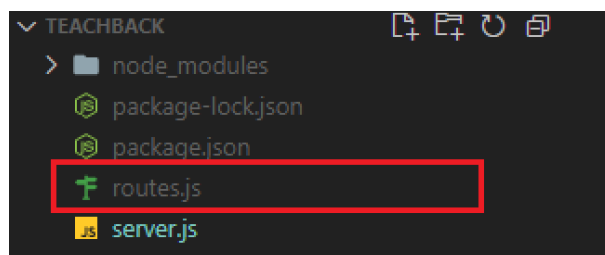
โดย app นี้คือตัวจัดการและรัน server backend ของเรา ใส่ method ต่างๆของ express ลงไปด้วยการใช้ method use ของ app

```
server.js > ...
1  const express = require('express')
2
3  const app = express()
4
5  app.use(express.json())
6  app.use(express.urlencoded({extended:true}))
7  |
```

โดย method ของ express ต่างๆสามารถศึกษาได้จากที่นี่

- [รู้จัก method ของ express](#)
- <https://expressjs.com/>

จากนั้นเราจะสร้าง ไฟล์ routes ขึ้นมาเพื่อที่จะสร้าง route ต่างๆของ api



ในไฟล์ server.js สร้างตัวแปร const เพื่อเก็บไฟล์ routes.js ที่สร้างขึ้นชื่อ routes และเราจะเรียกใช้ route ต่างๆ ผ่าน https://url/api โดยการเติม app.use('/api',routes)

```

server.js > ...
1  const express = require('express')
2  const app = express()
3  const routes = require('./routes')
4
5  app.use(express.json())
6  app.use(express.urlencoded({extended:true}))
7
8  app.use(['/api', routes])

```

ในไฟล์ routes.js เราจะใช้ express และ สร้าง router ด้วย method Router ของ express

```

routes.js > ...
1  const express = require('express')
2  const router = express.Router()
3
4

```

โดยการวางแผนการเขียน script ของ backend นั้นจะแล้วแต่คนที่เป็นคนเขียนแต่ว่าการเขียนที่ตรงนั้นจะต้องอ่านได้ง่าย และสามารถเข้าใจได้ง่าย โดยการเขียนในที่นี้จะเป็นการแยก script ที่จะเก็บ function ไว้ตามการใช้งาน แต่ใน workshop นี้จะไม่ลงรายละเอียดไปตรงนั้น

โดยการเขียน function ที่ router จะทำหลังมีการ request เข้ามานั้นจะสามารถเขียนได้ทั้งใน routes.js เลยหรือว่าจะ import มาจากไฟล์อื่นก็ได้

การเขียน function ใน routes.js โดยเราจะให้ส่ง Hello world กลับไปเป็น response ใน path url/api/ หลังได้ method get ได้โดย

```

routes.js > ...
1  const express = require('express')
2  const router = express.Router()
3
4  router.get('/', (req, res) => {
5    res.send('Hello world')
6  })

```

โดยหลัง router นั้นจะเป็น . แล้วตามด้วย method ที่ api นั้นจะรับและใน arguments จะเป็น path แล้วตามด้วย handler

หลังจากนั้นเราจะมาลองรัน server เป็น localhost บนเครื่องคอมพิวเตอร์ของเรา

เริ่มจากกลับไป server.js การที่จะรัน server ได้นั้นเราจะต้องมี port สำหรับติดต่อมาที่ server ของเรา

```

const port = 5000
app.listen(port, () => console.log(`server up and running at port ${port}`))

```

โดยเราจะกำหนด port ไว้ที่ 5000 และให้ app server ของเราไปรอ request ที่ port นั้น จากนั้นเปิด terminal และพิมพ์ npm start

```

C:\CE\ISAG\TeachBack> npm start

> teachback@1.0.0 start D:\CE\ISAG\TeachBack
> node server.js

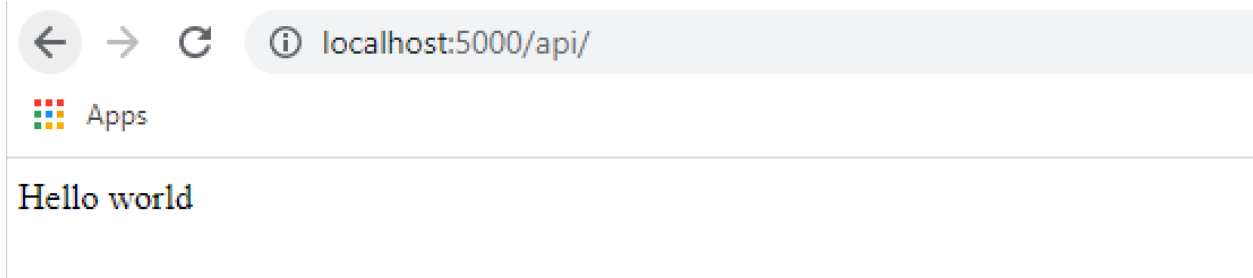
server up and running at port 5000

```

จะเห็นได้ว่ามี log ออกมาตามที่เราเขียนไว้ที่ app.listen

เมื่อเราลองเปิด browser แล้วพิมพ์ localhost:{portที่ตั้งไว้}/api/ ลงไปที่ url bar

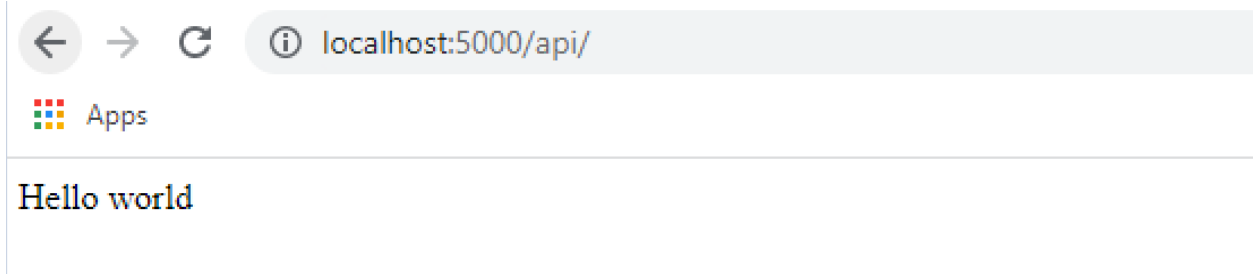
แล้วก็จะเห็นว่า มี Hello world ขึ้นมาที่ web browser



เราสามารถใช้ function ที่เขียนอยู่ใน script อื่นได้ด้วยการใช้ require และ export method ของ javascript เช่น

```
routes.js > ...
1  const express = require('express')
2  const router = express.Router()
3  const homeController = require('./controller/homeController')
4
5  router.get('/', (req,res)=>homeController.helloHome(req,res))
6
7  module.exports = router
8  |
```

```
controller > js homeController.js > [?] <unknown>
1  ✓ async function helloHome(req,res){
2    |   res.send('Hello World')
3    | }
4  module.exports = {helloHome}
```



ก็จะได้น้ำเว็บเหมือนเดิมมา

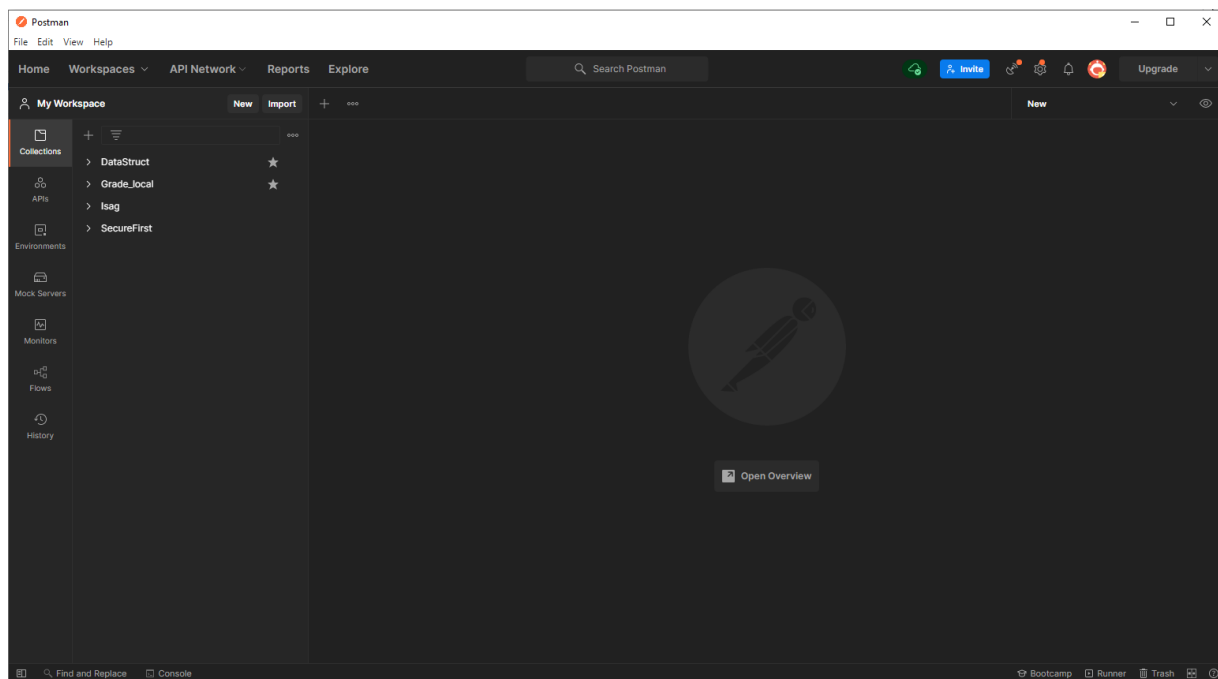


ซึ่งในส่วนที่น้องๆจะต้องเขียนในงาน website ของ ISAG นั่นก็คือ script ที่จะเก็บ function ที่จะให้ router ดึงไปใช้นั่นเอง

## POSTMAN API debugging

### Topic #1 Postman installation

การติดตั้ง Postman นั้นสามารถทำได้โดยการโหลดตัวติดตั้ง postman desktop จาก <https://www.postman.com/downloads/> เมื่อเปิด Postman ขึ้นมาจะมีหน้าต่างประมาณนี้

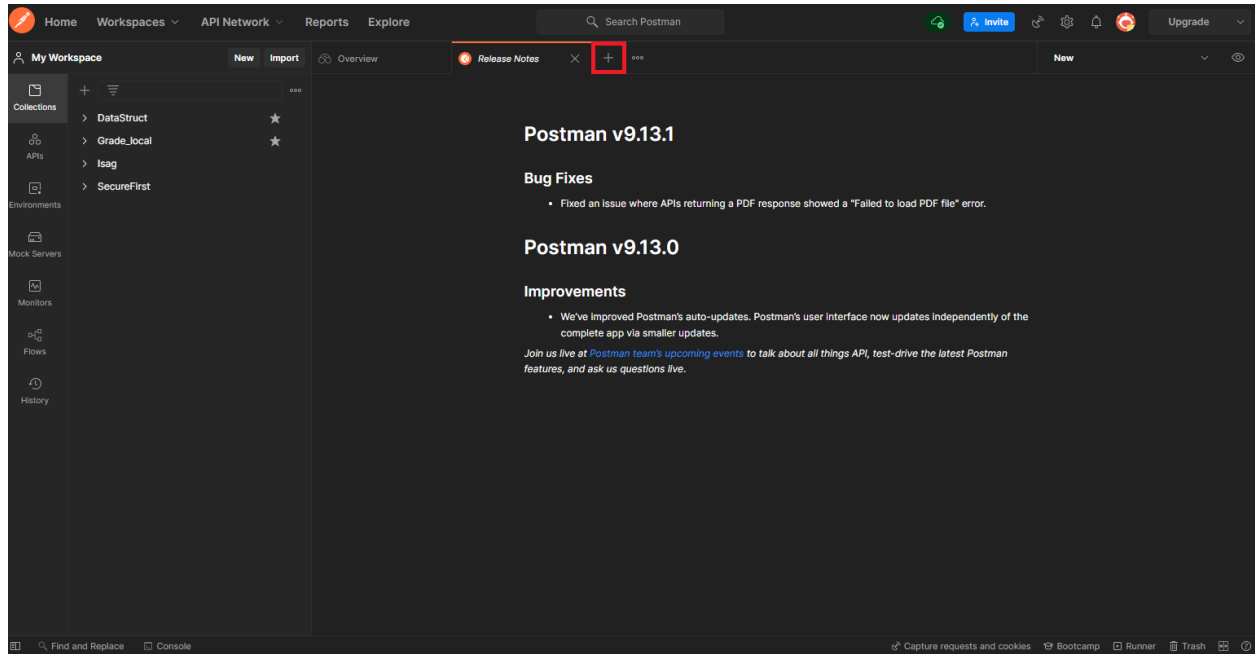


## Topic #2 Basic Postman Usage

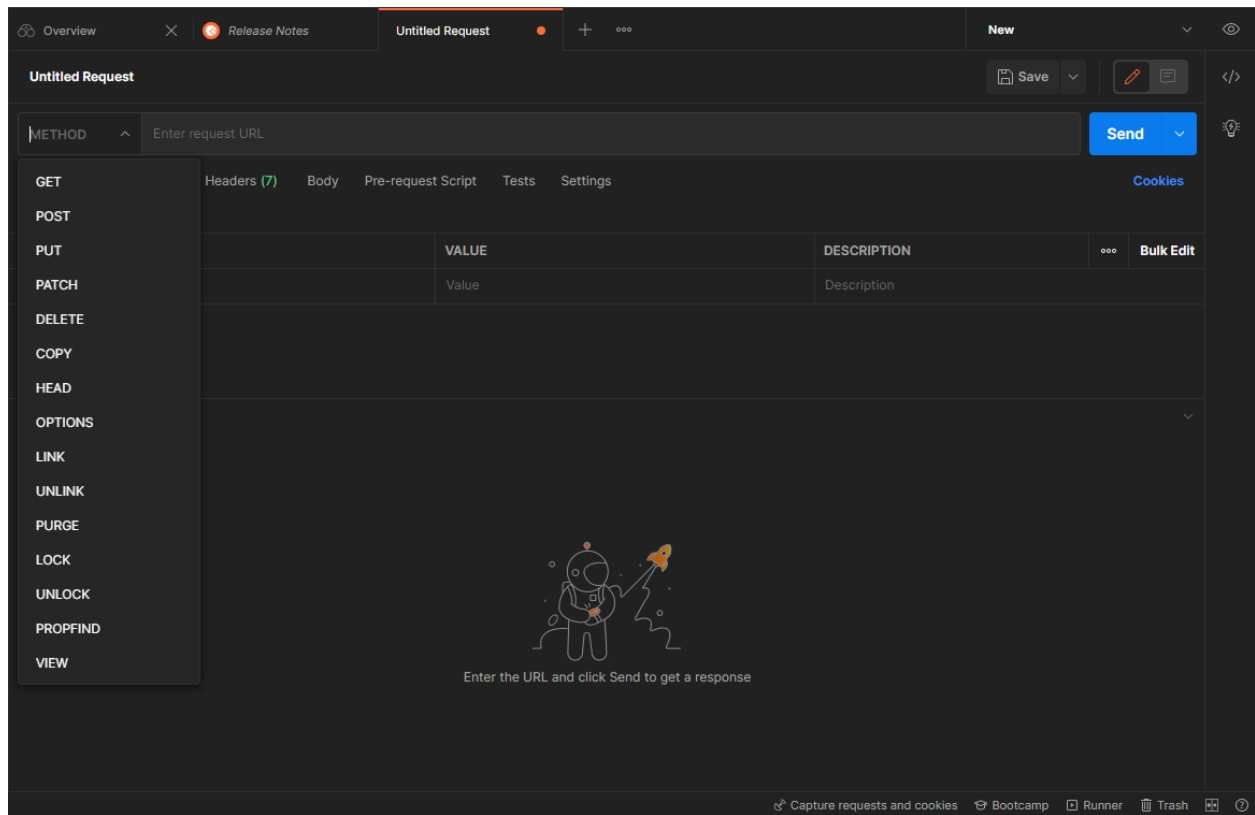
การใช้ postman นั้นจะสามารถใช้ในการส่ง request ด้วย Method ต่างๆยกตัวอย่างเช่นการส่ง req ไปที่

<https://jsonplaceholder.typicode.com/todos>

เริ่มด้วยการสร้าง req ด้วยการกด + ที่อยู่ด้านบน



เมื่อกดแล้วจะได้หน้าต่าง ที่ไว้ตั้งคำว่า req ของเราจะมีอะไรบ้างโดยเราจะเลือก Method ของ request ของเราเป็น GET



โดยหลังจากนั้นเราจะใส่ URL ที่เราต้องการส่ง req ไปในช่อง request URL

Overview

Release Notes

GET https://jsonplaceh...

New

https://jsonplaceholder.typicode.com/todos

GET

https://jsonplaceholder.typicode.com/todos

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests


Settings

Cookies

Query Params

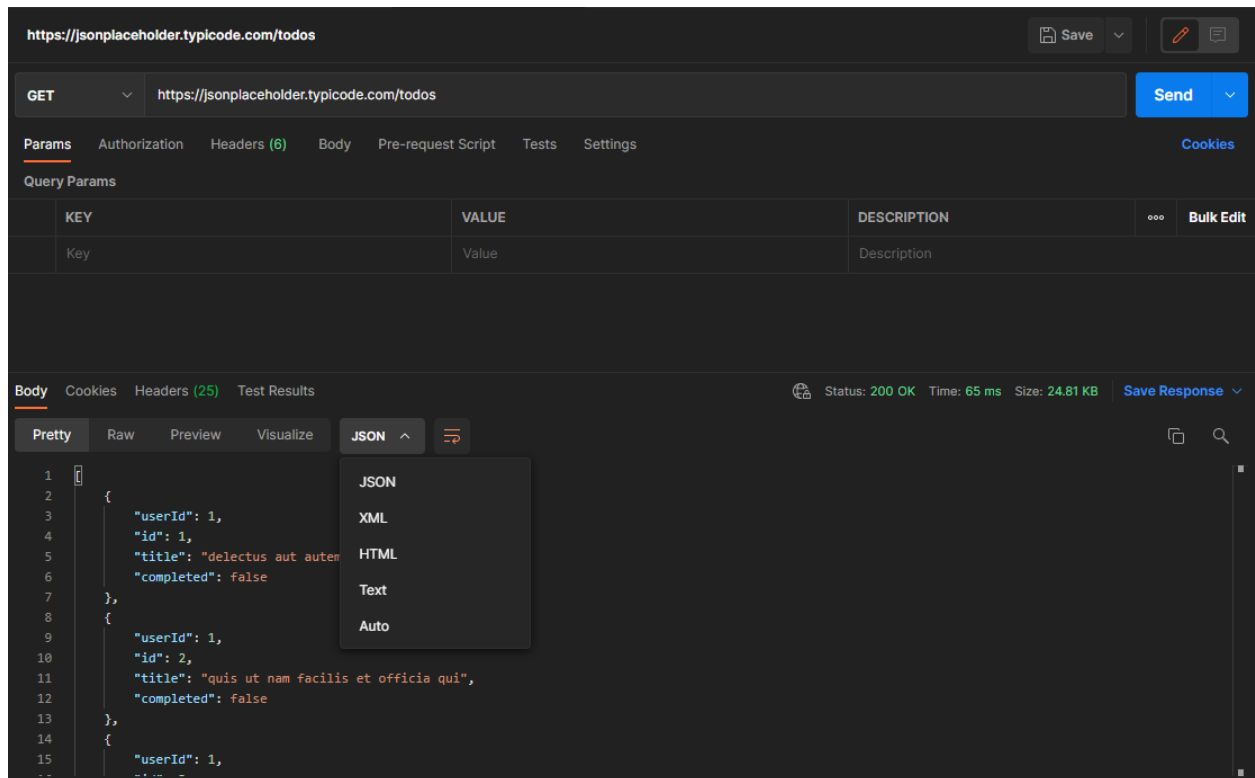
KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Response



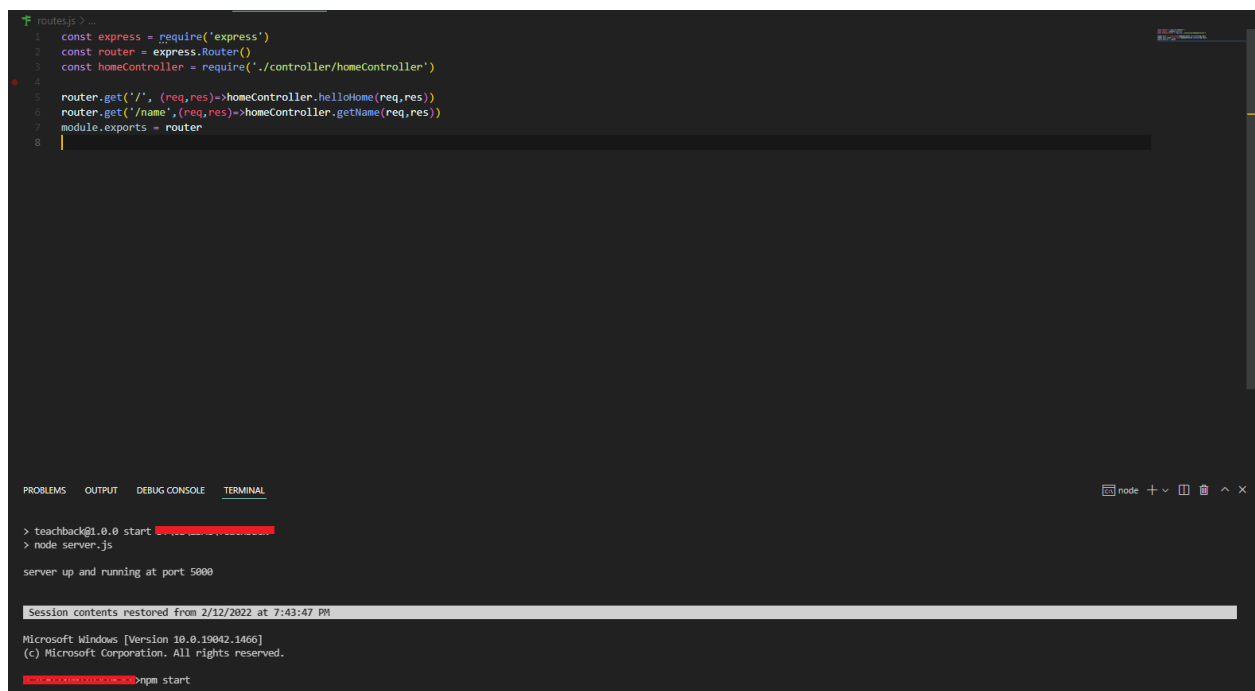
Click Send to get a response

เมื่อก็ send ไปเราก็จะสามารถดู response ที่ส่งกลับมาได้ในหน้าต่าง Response และเลือกรูปแบบของ Response เป็น JSON

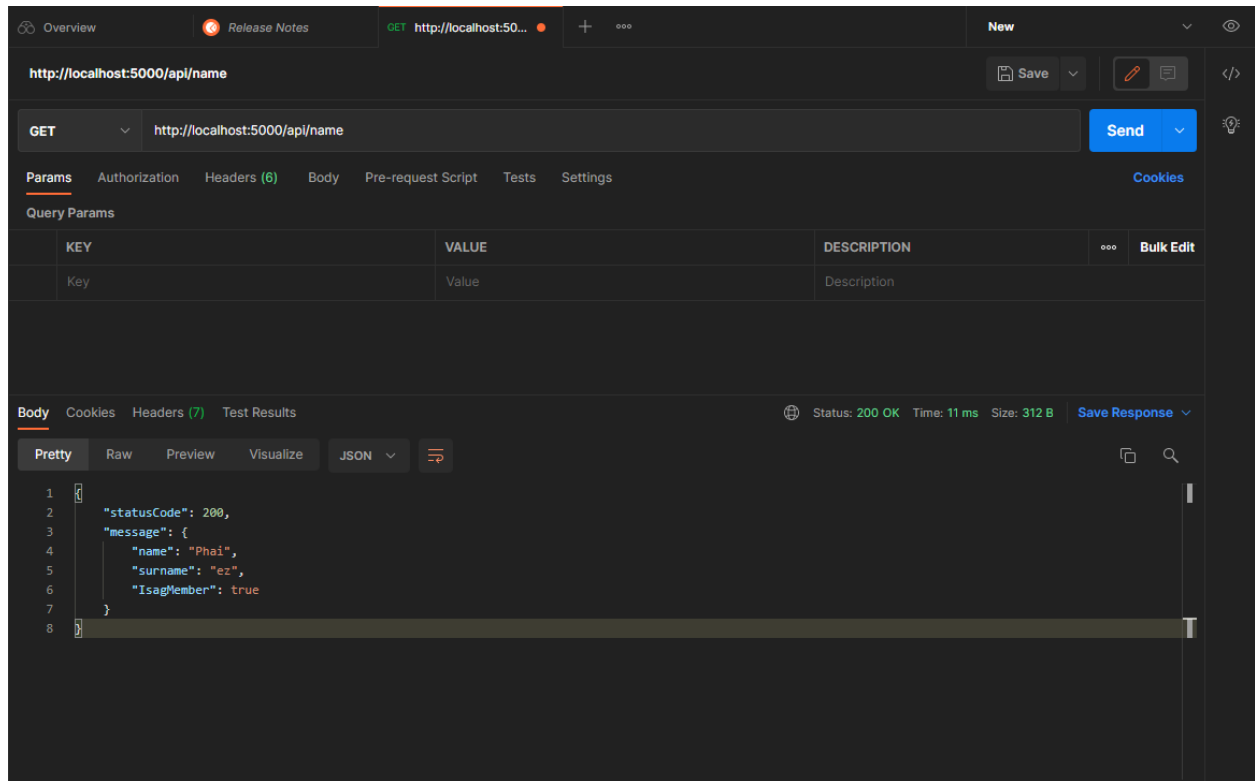


จะเห็นว่าเราได้ response กลับมาซึ่ง response ตรงนี้คือ response ที่เราจะต้องส่งไปให้กับ frontend ดังนั้นเราจึงสามารถนำ postman เพื่อมา debug ดู response ที่เราเขียนขึ้นก่อนที่จะนำไปเชื่อมกับ frontend ได้

เช่น การเพิ่มเติมจาก code ด้านบนและรัน server เป็น localhost ได้



จะเห็นว่าบรรทัดที่ 6 นั้นจะมี path url ใหม่ขึ้นมาเป็น /name ซึ่งจะส่งข้อมูลต่างๆกลับมา เราก็จะสามารถดูหน้าตาของ response ที่ส่งกลับมาได้ด้วย postman



## Work

เขียน API ที่ใช้ method GET และ path /api/name โดยจะส่ง response กลับมาเป็น statusCode เป็น int และ message เป็น obj ที่จะมี name และ surname เป็น string และ IsagMember เป็น Boolean

โดยอัปไฟล์ขึ้น github และส่ง link มาที่  
ส่งงานมาที่

URL : <https://forms.gle/koGhD8TZjadtqVYs8>