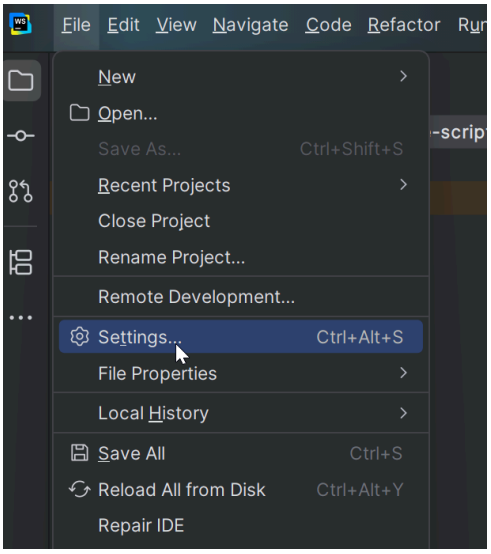
CHIANG MAI UNIVERSITY
Master of Science (Software Engineering)
College of Arts, Media and Technology
2nd Semester / Academic Year 2025
SE713 Backend Development

Introduction

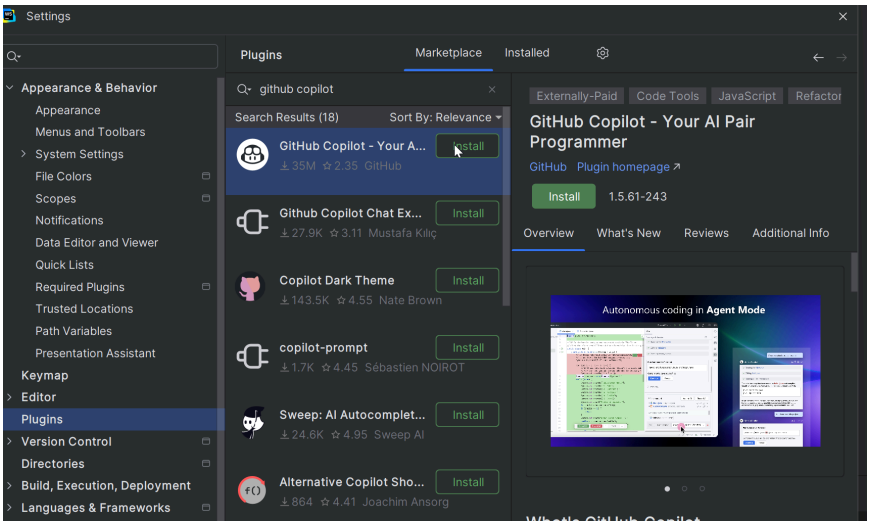
วัตถุประสงค์ เพื่อจัดเตรียมระบบที่ใช้ในการเรียน และเรียนรู้การใช้ node.js เบื้องต้น

Task 1 การจัดเตรียมคอมพิวเตอร์ (20 นาที)

1. สร้าง GitHub Account โดยใช้ email @cmu.ac.th โดยผ่าน url นี้ Sign in to GitHub · GitHub แล้วเลือก Create an Account จากนั้นดำเนินการตามขั้นตอนการสมัครจนกระทั่งเสร็จสิ้น
2. เตรียมบัตรประจำตัวนักศึกษาจากนั้นสมัคร GitHub Student Developer Pack เพื่อขอใช้บริการ GitHub Student Pack เพื่อใช้ GitHub Copilot สำหรับนักศึกษา
3. Download และ Install node.js เพื่อใช้ในการเขียนโปรแกรม โดยเลือกเวอร์ชันที่ลงท้าย LTS (Long-Term Support)
4. Download และ Install Git-scm, Fork เพื่อใช้ในการใช้งาน Git
5. Download และ Install Visual Studio Code WebStrom เพื่อใช้ในกระบวนวิชา
    a. เปิด WebStrom แล้วเปิด เมนู Setting เพื่อเพิ่ม plugins สำหรับ Github

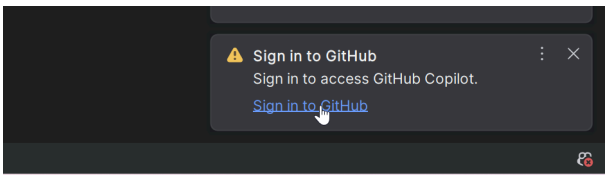


b. เลือก Plugins แล้วเลือก Github Copilot plug in install ลงใน WebStrom IDE

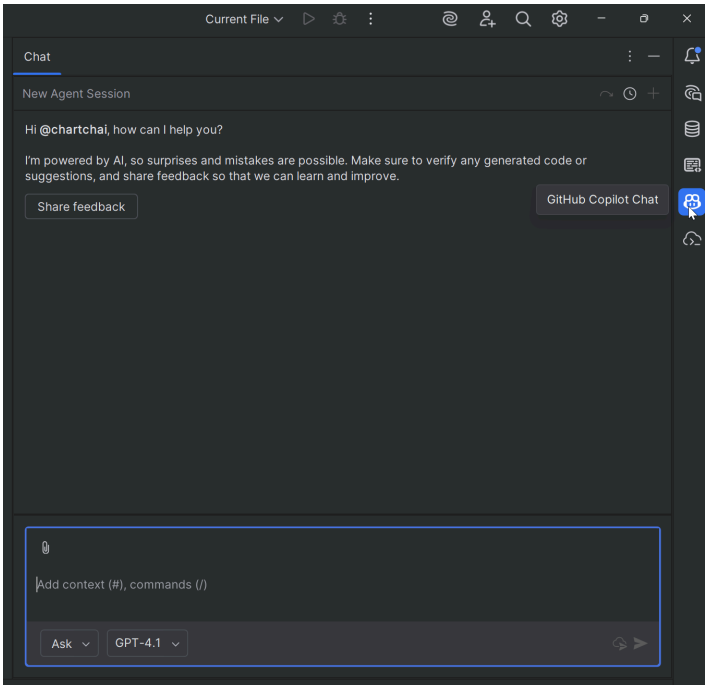


Restart WebStrom IDE เพื่อใช้ Plugins

c. เมื่อติดตั้งเสร็จแล้วให้กดปุ่ม Login เพื่อเชื่อมต่อกับ Github account เพื่อสามารถใช้ Github Copilot ได้

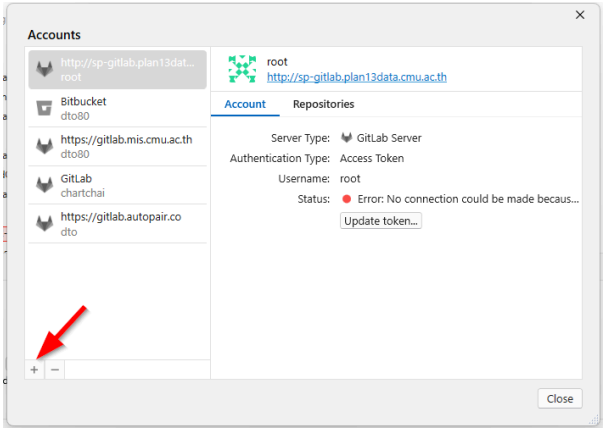


d. เมื่อลงสำเร็จสามารถใช้งานได้โดยการกดปุ่มไอคอน Copilot


Current File
Chat
New Agent Session
Hi @chartchai, how can I help you?
I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.
Share feedback
GitHub Copilot Chat
Add context (#), commands (/)
Ask
GPT-4.1

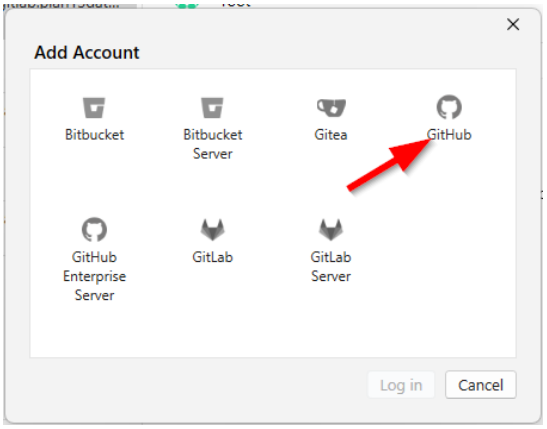
Task 2 ทดลองสร้าง Git Repository (10 นาที)

6. เปิดโปรแกรม Fork เพื่อเชื่อมต่อ Fork กับ GitHub โดย Add Acount ของ GitHub กับระบบ Fork
7. เลือก File -> Accounts… เลือกปุ่ม + เพื่อเพิ่ม Account

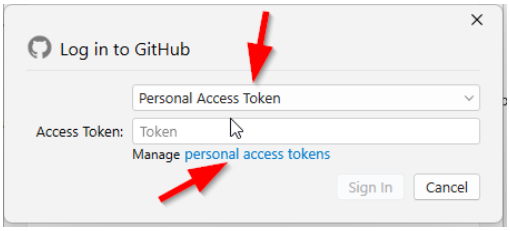


ทั้งนี้หน้าจออาจจะแตกต่างกันไปตามระบบปฏิบัติ

8. เลือกบริการที่ต้องการจะ Add Account เลือก GitHub



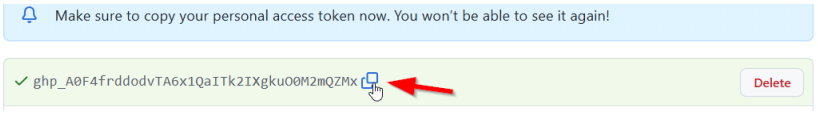
9. เลือก Personal Access Token จากนั้นเลือก personal access tokens เมนู เพื่อสร้าง personal access token ระบบจะเปิดหน้าจอ GitHub เพื่อสร้าง Personal Access Token

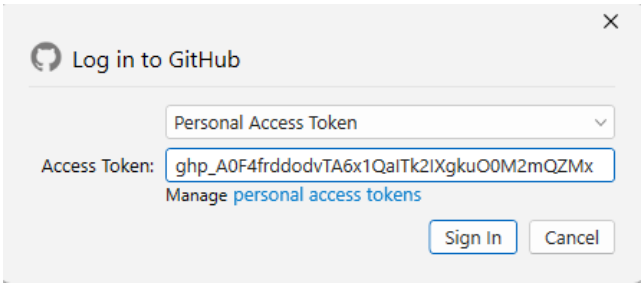


10. ที่หน้าจอการสร้าง Personal Access Token ระบุที่ Note เพื่อสร้างชื่อที่ต้องการอธิบาย จากนั้นเลือก Expiration เพื่อระบุว่าต้องการให้ Token มีอายุเท่าใด ทั้งนี้เพื่อการศึกษา เราสามารถกดเลือก No expiration เพื่อให้ Token ไม่หมดอายุ และสามารถใช้งานได้ตลอด (ทั้งนี้ไม่แนะนำในการทำงานจริงเนื่องจาก token ควรมีอายุจำกัดเพื่อป้องกันปัญหาที่อาจเกิด

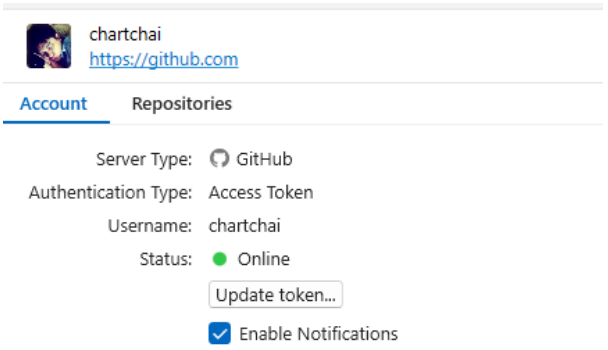
ขึ้นหากเกิดการรั่วไหลของ token) จากนั้นกดปุ่ม Generate Token เพื่อสร้าง Token

11. Copy Token ที่ได้และนำไปใส่ใน Fork (ระวังไม่ควรนำ Token ไปให้ผู้อื่น)

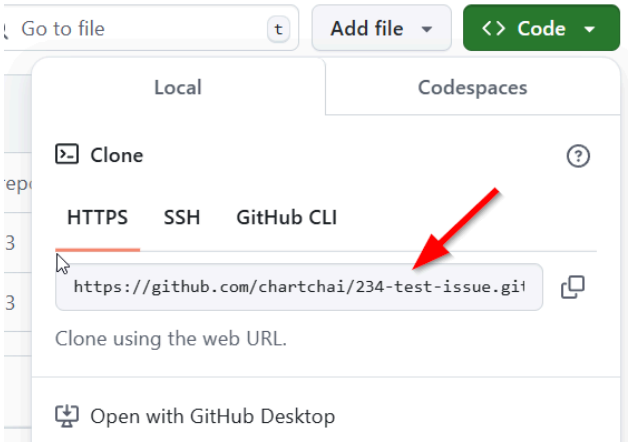




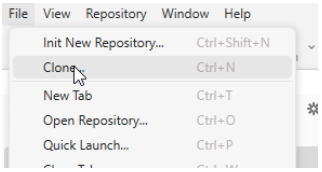
จากนั้นกดปุ่ม Sign in ถ้าการเข้าใช้ระบบถูกต้อง จะสังเกตุว่า มี status online อยู่



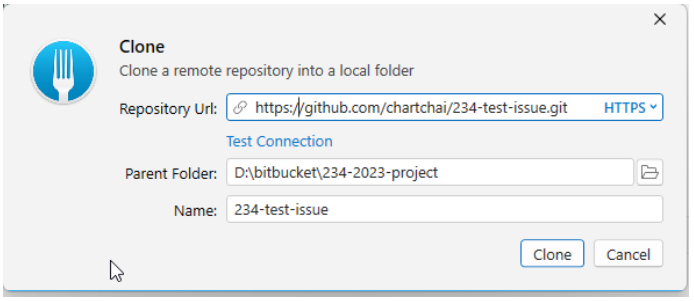
12. เปิดหน้า GitHub WebSite เลือก สร้าง Repository ใหม่
13. เลือกชื่อ repository เป็นอะไรก็ได้จากนั้นกดตกลง
14. ที่หน้า code กดปุ่ม code เพื่อ copy สำหรับการ clone source code



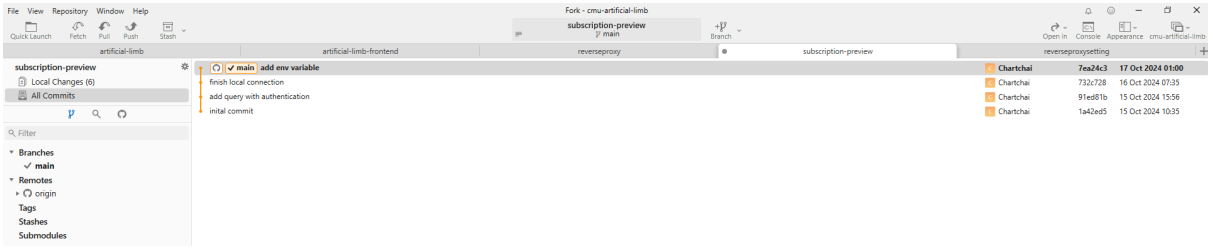
15. เปิด Fork อีกครั้ง แล้วเปิด เมนู File -> Clone



16. Copy url ที่ได้มา ลงใน Repository Url: แล้วเลือกสถานที่เก็บไฟล์บนเครื่องจากนั้นกด Clone



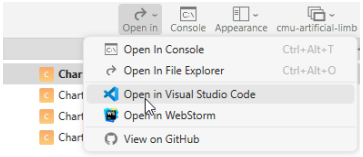
17. Source code และเนื้อหาของไฟล์จะแสดงอยู่ในหน้าจอ Fork



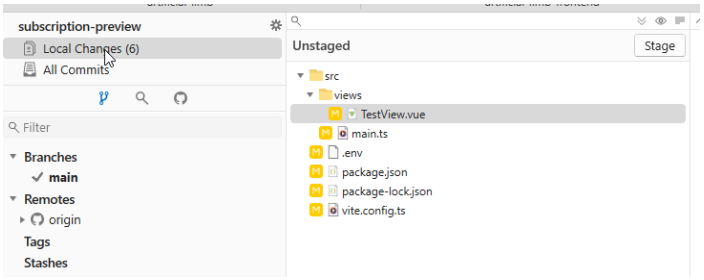
จบ Task

Task 3 การ Commit Code (5 นาที)

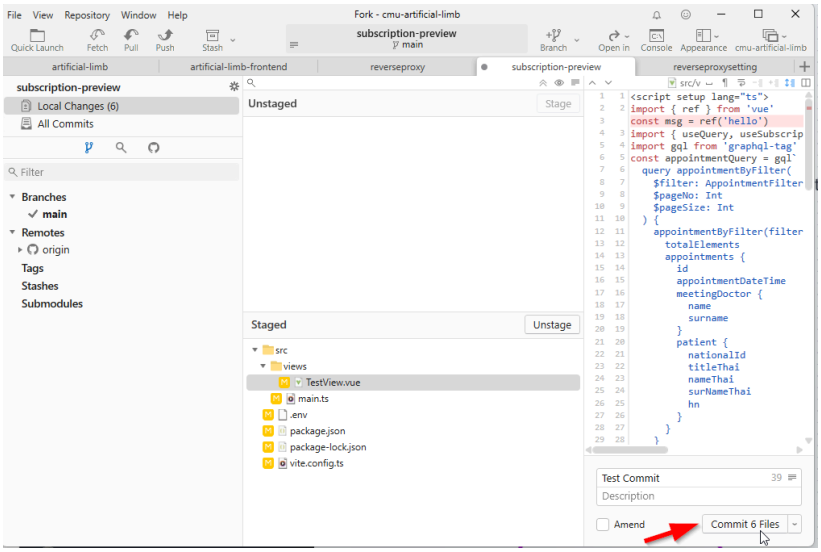
1. จากหน้าจอ Fork เราสามารถแก้ไขข้อมูลใน Folder ได้โดยไม่ต้องจำ Location ของไฟล์ โดยสามารถเปิด VS Code ได้โดยตรง โดยเลือก เมนู Open in



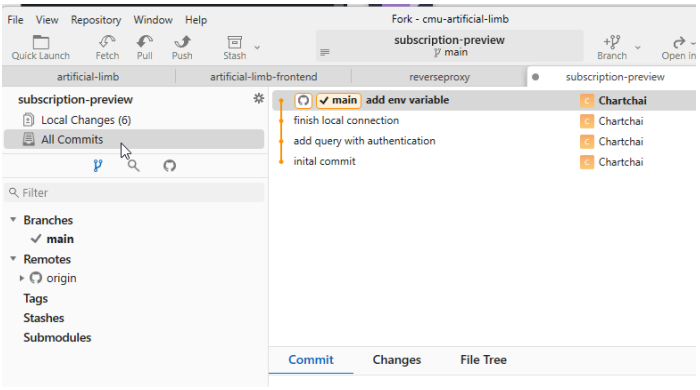
2. VSCode จะถูกเปิดออกมา จากนั้นทดลองสร้าง ไฟล์ใน VSCode แล้วกด Save file
3. จากนั้นกลับมาดูใน Fork จะพบกว่ามีการแจ้งว่ามีการเปลี่ยนแปลงที่ Local Changes



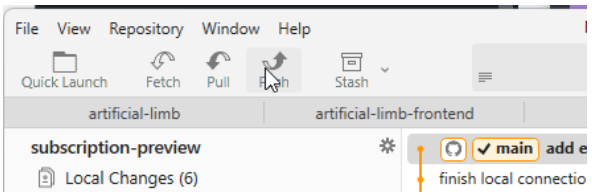
4. เพื่อที่จะ Commit Code กดเลือกไฟล์ที่ต้องการ Commit จากนั้นกดปุ่ม Stage ไฟล์จะย้ายจาก Unstaged เป็น Stage จากนั้นให้พิมพ์ Commit Subject เพื่อระบุเหตุผลในการ commit code จากนั้นกด Commit



5. กดปุ่ม All Commits เพื่อดูประวัติการแก้ไข



6. ทดลองแก้ไขไฟล์เดิม และสร้างไฟล์ใหม่ จากนั้นทดลอง Commit Code เพื่อดู History ที่เกิดขึ้นใหม่
7. กดปุ่ม Push เพื่อส่งไฟล์ที่แก้ไข ไปที่ GitHub



8. เปิดดูหน้า Website เพื่อดู Code ที่เปลี่ยนแปลงไปที่ GitHub

จบ Task

Task 4 การทำงานร่วมกันบน GitHub (20 นาที)

1. เข้า ร่วม repository ที่อาจารย์สร้างให้ที่นี่ (รอให้อาจารย์ add collaborator ให้เสร็จก่อน)
2. Clone Code มาที่เครื่องตนเอง ทำการแก้ไข index.html โดยเพิ่มข้อมูลของตนเอง จากนั้น commit code และ push code กลับไปที่ GitHub
3. สามารถดูผลงานของตนเองได้ที่ url นี้
4. ทั้งนี้นักศึกษา อาจจะ push code ไม่ได้เนื่องจากเกิด conflict ขึ้นให้นักศึกษา pull code ลงมาที่เครื่องตนเองจากนั้นแก้ไข conflict โดยลบส่วนที่มี conflict ออกแล้วแก้ไข code ของตนเอง (โดยลบส่วน >>>, <<<< และแก้ไข code ของตนเอง)
   จบ Task

Task 5 การติดตั้งใช้งาน TypeScript (7 นาที)

1. สร้าง repository ใหม่แล้ว clone ลงมาที่ computer ของผู้เรียนจากนั้นใช้ Web Strom เปิด folder นั้น
2. ในการใช้งาน Type Script จะต้องติดตั้งเครื่องมือสำหรับ Transpile TypeScript ก่อน โดยเครื่องมือจะอยู่ใน npm framework โดยใช้คำสั่งใน Terminal

```
npm init -y
```

เพื่อสร้างระบบของ npm ขึ้นมา

3. จากนั้น ให้เพิ่มส่วนประกอบสำหรับการใช้ Type Script โดยใช้คำสั่ง

```
npm install --save-dev typescript ts-node @types/node
```

ระบบจะ download component ที่ต้องใช้มาใน folder /nodes_modules และมีการแก้ไขไฟล์ package.json

4. เนื่องจาก ไฟล์ใน folder /nodes_module เป็นไฟล์ที่ไม่จำเป็นต้องบรรจุใน repository ดังนั้นก่อนจะ commit file ให้สร้างไฟล์ .gitignore เพื่อป้องกันไม่ให้ไฟล์เหล่านี้ถูก upload ขึ้นสู่ GitHub folder. โดยสามารถถาม GitHub CoPilot ว่า “What is the default .gitignore for npm project” ซึ่งจะได้ผลลัพธ์ที่ copy ออกมาได้ หรืออาจเอาข้อมูลดังนี้

```
 # Node modules
node_modules/
npm-debug.log
yarn-error.log
pnpm-debug.log


# Environment variables
.env
.env.local
.env.*.local


# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*
lerna-debug.log*
.log


# Dependency directories
node_modules/
jspm_packages/


# Build files
dist/
```

```
build/
.cache
.next/
out/
.vite/
.vscode/
.idea/


# OS specific files
.DS_Store
Thumbs.db


# Debugging
*.swp
*.swo


# Typescript
*.tsbuildinfo


# Testing
coverage/
jest.cache/
.nyc_output/


# Editor files
.vscode/
.idea/
*.sublime-project
*.sublime-workspace


# Linting
.eslintcache


# Misc
.npm/
.pnp/
.pnp.js
.parcel-cache/
```

5. เตรียมระบบสำหรับใช้ type script โดยใช้คำสั่ง

   ```
   npx tsc --init
   ```

   ไฟล์ tsconfig.json จะถูกสร้างขึ้น จากนั้น uncomment สองบันทัดนี้ให้มีลักษณะนี้

โดยแก้ไขค่าเป็นดังนี้

```
{
      "compilerOptions": {
       // File Layout
        "rootDir": "./src",
        "outDir": "./dist",

       "target": "ES2020",
       "module": "CommonJS",
       "moduleResolution": "node",
       "esModuleInterop": true,
       "strict": true,
       "skipLibCheck": true
      }
}
```

6. สร้าง folder src โดยใช้คำสั่ง

   ```
   mkdir src
   ```

7. สร้างไฟล์ `src/index.ts`
   แล้ว code ดังนี้

```
console.log("Hello, TypeScript with Node.js!");
```

8. ผู้เรียนสามารถ run program โดยใช้คำสั่ง

```
npx ts-node src/index.ts
```

   เพื่อ transpile และสร้าง executable file ได้ทันที

9. หากผู้เรียนต้องการ transpile TypeScript เป็น JavaScript สามารถทำได้โดยใช้คำสั่ง

   ```
   npx tsc
   ```

   จากนั้นเปิด folder `dist` เพื่อดูผลลัพธ์ที่ได้ออกมา

จากนั้นสามารถ รันโปรแกรม โดยใช้ node เหมือนกับ node js โดยใช้คำสั่ง

```
node dist/index.js
```

10. ทั้งนี้เราสามารถสร้าง script สำหรับ run โปรแกรม โดยไม่ต้องจำคำสั่งทั้งหมด โดยใน package.json เราสามารถแก้ไขส่วนของ script ดังนี้

```
"scripts": {
 "start": "node dist/index.js",
 "build": "tsc",
 "dev": "ts-node src/index.ts"
}
```

จากนั้นเมื่อเราสั่งคำสั่งนี้ใน terminal ระบบจำทำงานดังนี้

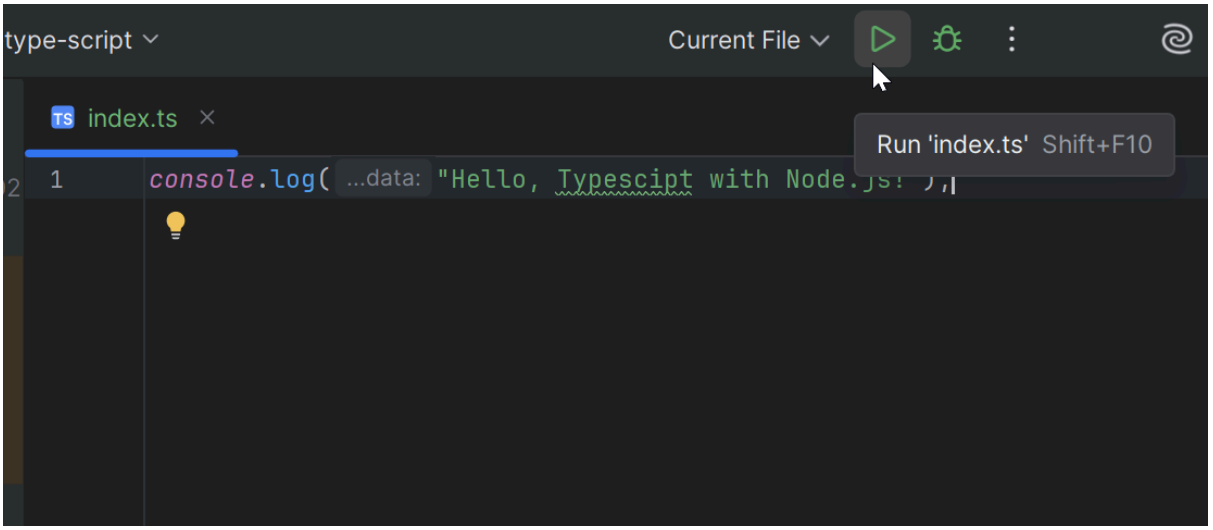
`npm run dev` ระบบจะรัน file โดยตรง

`npm run build` ระบบจะ transpile TypeScript เป็น JavaScript

`npm start` ระบบรัน JavaScript ใน dist folder

ดังนั้นจาก task นี้เราจะรันโปรแกรมเราโดยใช้คำสั่ง npm run dev

หรือสามารถ run application ได้ผ่านการกดปุ่ม run เมื่อเลือกไฟล์ index.ts



Task 6 การใช้งาน data type

1. ทดลอง พิมพ์ code ดังนี้

```
let x:number = 10;
x = 'hello';
```

2. จะเห็นว่า ค่า x มีคำเตือน ทดลอง รันโปรแกรมเพื่อดูผลลัพธ์

จะพบว่ามี TSError นั่นคือโปรแกรม compile ไม่ผ่านยังไม่เกิดการรันโปรแกรมขึ้น

แก้ไข โดยอนุญาติให้ x เป็น string ได้ด้วย โดยแก้ไข code ดังนี้

```
let x:number | string = 10;
x = 'hello';
```

จะเห็นว่า error หายไป

3. ทดลอง run code จะเห็นว่า code สามารถ รันได้

การตรวจสอบว่า ตัวแปรมีชนิดอะไร สามารถตรวจสอบได้ด้วยคำสั่ง typeof

```
let x:number | string = 10;
x = 'hello';
if (typeof x === 'string') {
   console.log('x is a string');
}else if (typeof x === 'number') {
   console.log('x is a number');
}else {
   console.log('x is neither a string nor a number');
}
```

ลองดูส่วนผลลัพธ์ที่ออกมา

แล้วลองเปลี่ยน x เป็น ค่าตัวเลข ผลลัพธ์ที่ได้คืออะไร

4. ทดสอบการเขียนโปรแกรมเพื่อเปลี่ยนชนิดข้อมูล

```
let i = 0
i = 'hello'
console.log(i)
```

จะพบว่ามีการฟ้อง error ขึ้นมาเนื่องจาก ‘hello’ เป็น String ไม่ใช่ number แต่เมื่อกดรัน โปรแกรมจะทำงานได้

5. เพื่อนำคำเตือนออกแก้ไข code เป็น แล้วตรวจสอบชนิดของข้อมูล

```
let i = 0
i = 'hello' as any
console.log(i)
```

โปรแกรมจะเปลี่ยน hello i ให้เป็น type any จึงสามารถ assign ไปใน i ได้ และ i จะถูกเปลี่ยยนชนิดของตนเองโดยอัตโนมัติ

Task 7 การประกาศตัวแปรด้วย const let

1. **สร้างไฟล์ใหม่ ทดสอบสร้างตัวแปรด้วยคำสั่งนี้**

```
let x = 10;
x = 15;
console.log(x);
```

ทดลองรันโปรแกรม

2. ทดลองปรับ source code ดังนี้

```
const x = 10;
x = 15;
console.log(x);
```

ทดลองดูผลลัพธ์ที่ได้

3. ทดลองสร้าง Object type

```
interface Person {
   name: string;
   age: number;
}

let a: Person = {
   name: 'Alice',
   age: 30
}

a = {
   name: 'Bob',
   age: 25
}
console.log(a);
```

แล้วทดลองรันโปรแกรม

4. ปรับ source code เป็นดังนี้

```
interface Person {
   name: string;
   age: number;
}

const a: Person = {
   name: 'Alice',
   age: 30
}

a = {
   name: 'Bob',
   age: 25
}
console.log(a)
```

แล้วทดลองรันโปรแกรม

5. ลองทดลองเขียนโปรแกรมดังนี้

```
interface Person {
   name: string;
   age: number;
```

```
}

const a: Person = {
   name: 'Alice',
   age: 30
}

a.name = 'Bob'
a.age = 25
console.log(a)
```

แล้วทดลองรันโปรแกรม

6. ลองทดลองเขียน List

```
const list = [1, 2, 3, 4, 5];
console.log(list);
```

ทดลองรันเพื่อดูผลลัพธ์

Task 8 การเขียนโปรแกรม if else (15 นาที)

1. ลบข้อมูลเดิมในไฟล์ index.ts
2. ทดลองเขียน code if แล้วทดลองรันโปรแกรม

```
let x:number = 10
if (x > 10) {
  console.log('x is more than 10')
}
```

3. ลองเปลี่ยนค่า x แล้วดูผลลัพธ์ที่เปลี่ยนแปลงไป
4. ทดลองเขียน if-else statement ดังตัวอย่าง

```
let x:number = 9
if (x > 10) {
  console.log('x is more than 10')
}else {
  console.log('x is less than 10')
}
```

5. ทดลองเปลี่ยนค่า x เป็นค่าอื่น
6. ทดลองเขียนโปรแกรม ให้กำหนดค่าสองค่าคือ ค่า x และ ค่า y ถ้าค่า x มากกว่า 5 และ y > 12 ให้พิมพ์ข้อความ “ค่ามากจัง” มิเช่นนั้นให้แสดงค่า x*y + x/y
7. เมื่อทำเสร็จอย่าลืม save ไฟล์และ commit code
8. ทดลองเขียน if -elseif -else statement ดังตัวอย่าง

```
let x =7
if (x > 10) {
  console.log('x is more than 10')
} else if (x < 5){
    console.log('x is less than 5')
}
else {
  console.log('x is between 5 and 10')
}
```

ทดลอง run program และลองเปลี่ยนค่า x เพื่อดูผลลัพธ์

9. เขียนโปรแกรมเพื่อแสดงผลการแสดงเกรด ของนักศึกษา โดยถ้าคะแนนมากกว่า 80 ให้แสดงเกรด A ถ้าคะแนนมากกว่า 60 ได้เกรด B ถ้าได้คะแนนมากกว่า 50 ได้ C ต่ำกว่านั้นได้ F

ทดสอบรันแล้ว commit source code

Task 9 การเขียนโปรแกรมโดยใช้ For Loop (10 นาที)

1. **สร้าไฟล์ for.ts ทดลองเขียนโปรแกรมดังนี้**
2. **เขียนโปรแกรมดังนี้**

```
let i = 0;
for (i = 0; i < 10; i++) {
  console.log(i);
}
```

3. **รันโปรแกรมเพื่อดูผลลัพธ์**
4. **เราสามารถอ่านค่าใน list ได้ดังนี้**

```
let color = ['red', 'blue', 'green', 'yellow', 'orange', 'purple']
for (let i = 0; i < color.length; i++) {
  console.log(color[i])
}
```

5. **ทดลองเขียนโปรแกรม โดยสร้าง list ของชื่อคน จำนวน 5 คนโดยตรวจสอบชื่อแต่ละคน ว่าถ้าตัวอักษรในชื่อยาวเกิน 6 ตัวอักษรให้แสดงผลว่า “สวัสดีคุณ ชื่อคน” แต่ถ้าตัวอักษรในชื่อยาวน้อยกว่า 6 ตัวอักษรให้แสดงผลว่า “Hello ชื่อคน”**

Task 10 การเขียนโปรแกรมโดยใช้ For Loop 2 (10 นาที)

1. ทดลองใช้ **for...of** เพื่อวนลูปอ่านค่า (value) ใน List/Array

```
let color = ['red', 'blue', 'green', 'yellow', 'orange', 'purple']
for (const c of color) {
  console.log(c);
}
```

   รันโปรแกรมเพื่อดูผลลัพธ์: จะแสดงค่าของแต่ละ element ใน List

2. ทดลองใช้ **for...in** เพื่อวนลูปอ่าน index ใน Array หรือ property ของ Object

```
let color = ['red', 'blue', 'green', 'yellow', 'orange', 'purple']
for (const index in color) {
  console.log(`Index: ${index}, Value: ${color[index]}`);
}
```

   รันโปรแกรมเพื่อดูผลลัพธ์: จะแสดง index และค่าของแต่ละ element

Task 11 การเขียนโปรแกรมโดยเรียกใช้ Function 1 (10 นาที)

1. ในภาษา JavaScript สามารถเขียน function ได้หลายรูปแบบ โดยมีการเรียกใช้งานที่เหมือนกัน ทดสอบเขียน function ในรูปแบบ Function Declaration ดังนี้

```
function greet(name){
    console.log('Hello ' + name);
}
greet('John');
```

ทดลอง run program เพื่อดูผลลัพธ์

2. ในการใช้งาน Function เราสามารถให้ Function คืนค่าจากการคำนวนเพื่อให้สามารถนำไปใช้งานที่หลากหลายได้มากขึ้น ดังนี้

```
function greet(name){
    return 'Hello ' + name
}
greet('John');
```

เมื่อทดลองรันโปรแกรมดู เห็นผลลัพธ์หรือไม่

3. เราไม่สามารถเห็นผลลัพธ์ได้เนื่องจาก ไม่ได้มีการสั่ง พิมพ์ค่าใน source code ดังนั้นเราจึงต้องนำค่าที่ได้คืนมาจาก function มาแสดงผล โดยคำสั่ง console.log ดังนี้

```
function greet(name){
    return 'Hello ' + name
}

console.log(greet('John'))
```

4. จะเห็นว่าผลลัพธ์ที่ได้ ได้มาจากการเขียนโปรแกรมข้างต้น จากนั้นทดลองเขียนโปรแกรม โดยสร้าง function ที่รับ list เป็นคะแนนของนักศึกษา (เช่น [10,20,30] จากนั้นคำนวนผลรวมของข้อมูลใน list แล้วคืนค่าผลรวมออกมาเป็นผลลัพธ์ของ function

Task 12 การเขียนโปรแกรมด้วย Function ในรูปแบบอื่นๆ (10 นาที)

1. นอกจากรูปแบบขั้นต้นแล้ว เรายังเขียน function ในรูปแบบอื่นๆ ที่สามารถทำงานได้เหมือนกัน แต่การเรียกใช้งานยังเหมือนเดิม เช่น การสร้าง Function Expression

```
const greet = function greet(name){
    return 'Hello ' + name
}

console.log(greet('John'))
```

ทดลอง run program ผลลัพธ์ที่ได้ยังเหมือนเดิม

2. นอกจากนี้ยังสามารถเขียนรูปแบบ Arrow Function Description ดังนี้

```
const greet = (name) => 'Hello ' + name

console.log(greet('John'))
```

3. และหากต้องการให้ Arrow Functions ทำงานได้มากกว่า หนึ่งบันทัด สามารถ ใช้ { } คร่อม และใช้หลายๆ statement ได้ แต่อย่างไรก็ดี จำเป็นต้องมีคำสั่ง return เพื่อคืนค่าอีกด้วย อีกทั้งยังสามารถรับได้หลายๆ function อีกด้วย

```
const multiply = (x,y) => { return x * y }

console.log(multiply(5, 10))
```

4. สร้าง function ในรูปแบบ Arrow Functions โดยให้รับค่าตัวแปร สองค่า แล้วคืนค่าที่มีค่ามากกว่าออกมา

Task 13 การเขียนโปรแกรมด้วย functions โดยกำหนดชนิดข้อมูล (Explicit Type) (10 นาที)

1. การสร้าง function ผู้เรียนควรสร้าง function โดยมีการระบุ data type ของ input parameter และ ค่าที่คืนให้ชัดเจน เพื่อให้ ts compiler สามารถเจาะจงชนิดข้อมูลและได้ผลลัพธ์ที่ดีได้
   โดยทดลองสร้าง function ที่ไม่ระบุชนิดตัวแปร ดังนี้

```
const add = (a,b) => {
   return a+b;
}
console.log(add(1,2));
```

2. ทดลองเพิ่มตัวแปร ลงไปใน function

```
const add = (a:number,b:number) => {
   return a+b;
}
const result = add(1,2) + 0;
console.log(result, 'type of result:', typeof result);
```

   แล้วดูผลลัพธ์

3. บังคับให้ function add คืนค่าเป็น String แล้วดูคำเตือนที่พบ

```
const add = (a:number,b:number):string => {
   return a+b;
}
const result = add(1,2) + 0;
console.log(result, 'type of result:', typeof result);
```

4. ดังนั้นแก้ไข code ให้คืนค่าเป็น string โดยแก้ไข code ดังนี้

```
const add = (a:number,b:number):string => {
   const result =  a+b;
   return result.toString();
}
```

5. ทดลองสร้าง function findMax ที่รับ list ของ number แล้วไปหาค่าที่มากที่สุด แล้วคืนค่าออกมา เป็น string ของค่านั้น (สามารถทดลองถาม GitHub copilot) ผ่าน Chat feature ของ Copilot ได้

Task 14 การเรียกใช้ Function จากไฟล์อื่น (10 นาที)

1. จากไฟล์เดิม สร้างไฟล์ใหม่ชื่อว่า funtions.ts
2. ย้าย function add จาก index.ts มาที่ function.ts โดยไม่ต้องนำการใช้งานตามมา

ดังนั้นใน function.ts จะประกอบไปด้วย

```
const add = (a:number,b:number):string => {
   const result =  a+b;
   return result.toString();
}
```

และ index.ts จะเหลือ code

```
const result = add(1,2) + 0;
console.log(result, 'type of result:', typeof result);
```

3. เมื่อรัน index.ts จะพบว่าไม่สามารถรันได้เนื่องจากไม่พบ add function
4. แก้ไขไฟล์ทั้งสองไฟล์ดังนี้

function.ts

```
export const  add = (a:number,b:number):string => {
   const result =  a+b;
   return result.toString();
}
```

index.ts

```
import { add } from './function';
const result = add(1,2) + 0;

console.log(result, 'type of result:', typeof result);
```

5. ทดลองเพิ่ม function ใน function.ts

```
export const  subtract = (a:number,b:number):string => {
   const result =  a-b;
   return result.toString();
}
```

หากต้องการใช้งานสามารถใช้ได้ดังนี้

```
import { add,subtract } from './function';
const result = add(1,2) + 0;
```

```
const result2 = subtract(1,2) + 0;
console.log(result, 'type of result:', typeof result);
console.log(result2, 'type of result2:', typeof result2);
```

6. สามารถใช้ default export หากไม่ต้องการระบุ {} ของ function

แก้ไข function.ts ดังนี้

```
const  add = (a:number,b:number):string => {
   const result =  a+b;
   return result.toString();
}
export default add;
```

แก้ไขส่วน import ของ index.ts ดังนี้

```
import add, { subtract } from './function';
```

ทดลองย้าย function ที่สร้างจาก task ที่แล้วไปยัง ไฟล์ function2 แล้ว import มาใช้ในไฟล์ index.ts

Task 15 การเรียกใช้ JavaScript Object Notation (JSON)

1. ในการเขียนโปรแกรมเราสามารถสร้าง Object ของข้อมูลได้ ให้ทดลองสร้างไฟล์ชื่อ json.ts และเพิ่ม code ดังนี้

```
const object = {
    name: "John",
    age: 30,
    city: "New York"
}


console.log(object)
```

2. เราสามารถดึงค่าจาก field ที่อยู่ใน json object ได้จากการใช้ . และชื่อของ field ดังนี้

```
const object = {
    name: "John",
    age: 30,
    city: "New York"
}


console.log(object.name)
```

3. ในบางครั้งเราอาจต้องการสร้าง String จาก json ก็สามารถทำได้ โดยใช้คำสั่งดังนี้

```
const object = {
    name: "John",
    age: 30,
    city: "New York"
}


console.log(object.name)
const jsonStr = JSON.stringify(object);
console.log("Object: ", object)
console.log("json String: ", jsonStr)
```

โปรดสังเกตุความแตกต่างระหว่างผลลัพธ์ที่ได้จากการ พิมพ์ object โดยตรงและพิมพ์ object หลังจากที่ผ่าน funciton JSON.stringify(object)

4. ให้สร้าง function ที่รับ object ของนักศึกษา ที่ประกอบไปด้วยข้อมูลชื่อ นามสกุล อายุ เกรดเฉลี่ย จากนั้น ตรวจสอบว่าเกรดเฉลี่ยมีค่ามากกว่า 2 หรือไม่ ถ้าเกินให้แปลง object เป็น String และคืนค่ามา ถ้าไม่เกิน ให้คืนคำว่า “คุณ (ชื่อ) พ้นสภาพ”

Task 16 การเรียกใช้งาน Higher Order Function (7 นาที)

1. **สร้างไฟล์ HigerOrder.ts จากนั้นเพิ่มข้อมูลดังนี้ เพื่อใช้ในการสร้าง list ใหม่ที่มีค่าเท่ากับ ค่าใน list แต่ละค่า ยกกำลัง 2**

```
const n = [1, 2, 3, 4, 5];
const square = n.map((num) => num * num);
console.log(square);
```

2. **อีกตัวอย่างหนึ่งคือการใช้ Hihger Order Function เพื่อใช้เลือกเฉพาะข้อมูลที่มี ตรงกับข้อกำหนด ซึ่งในที่นี้คือมีค่า num %2 เท่ากับ 0**

```
const n = [1, 2, 3, 4, 5];
const even = n.filter((num) => num % 2 === 0);
console.log(even);
```

3. **เขียนโปรแกรมเพื่อ รับ list ของตัวเลข โดยเมื่อรับค่ามาแล้วให้เลือกเฉพาะเลขที่หารด้วย 3 ลงตัว แล้วนำค่าที่ได้มายกกำลังสาม แล้วคืนค่า list ที่ได้ออกมา**

Task 17 ตัวอย่างการเรียกใช้ High Order Function อื่นๆ (10 นาที)

1. **สร้าง timer.js แล้วพิมพ์ code นี้ลงไป**

```
waitAndPrint = (message, delay) => {
    setTimeout(() => {
        console.log(message);
    }, delay);
}


// Example usage:
waitAndPrint("Hello, world!", 2000);
```

2. **Update function นี้ให้แสดงผลคำว่า hello world ทุกๆ 3 วินาที และให้คำว่า world เพิ่มขึ้นทุกครั้งที่พิมพ์ออกมา**