Think Beyond Develop

Think Beyond

TypeScript +Node.js

สำหรับ
Full Stack Developer

- พื้นฐาน TypeScript ครบทุกแง่มุม สำหรับนักเรียน นักศึกษา และโปรแกรมเมอร์
- การใช้งาน TypeScript ขั้นสูง เช่น Generics, Decorators และ Metadata
- สร้างเว็บแอปพลิเคชันในฝั่ง Backend ด้วย Node.js และ Express
- พัฒนาทักษะและก้าวสู่การเป็น Full Stack Developer ในแบบมืออาชีพ
- วิธีใช้งานแพ็คเกจยอดนิยม เช่น axios, cookie-session, lodash, class-validator และอื่น ๆ

พิเศษ !

ดาวน์โหลดโค้ดตัวอย่าง
เพื่อใช้ประกอบการเรียน

จีราวุธ วารินทร์

จัดจำหน่ายโดย IDC PREMIER

คำนำ>

คู่มือ TypeScript + Node.js สำหรับ Full Stack Developer เล่มนี้ถูกออกแบบเพื่อการพัฒนาเว็บแอปพลิเคชันทั้งในฝั่งไคลเอนต์และเซิร์ฟเวอร์ โดยมีเนื้อหาครอบคลุมทุก ๆ เรื่องที่ควรทราบเกี่ยวกับ TypeScript และ Node.js

ส่วนแรกจะเริ่มต้นด้วยการแนะนำพื้นฐานของ TypeScript ตั้งแต่การทำความรู้จักกับตัวแปรและชนิดข้อมูล อาร์เรย์ Tuple และ Enum ฟังก์ชัน ออบเจ็กต์ อินเตอร์เฟส คลาส การสืบทอดคลาส และการใช้งานคลาสร่วมกับอินเตอร์เฟส รวมถึงการทำความเข้าใจชนิดข้อมูลอื่น ๆ ที่ควรทราบใน TypeScript

ส่วนที่สองจะเป็นวิธีใช้งาน TypeScript ในระดับที่ลึกขึ้น ซึ่งประกอบด้วยการใช้งาน Generics, Decorators, Metadata และ Promise ซึ่งเป็นเครื่องมือที่ทำให้การพัฒนาโปรแกรมมีประสิทธิภาพและยืดหยุ่นมากยิ่งขึ้น

ส่วนสุดท้าย เป็นการแนะนำเกี่ยวกับการพัฒนาแอปพลิเคชันในฝั่ง Backend โดยใช้ Node.js เริ่มจากพื้นฐานการใช้ TypeScript กับ Node.js การสร้างและจัดการโมดูล ตัวอย่างการใช้งานไลบรารียอดนิยม การใช้งาน Express การกำหนด Middleware การสร้าง API โดยใช้ Express การจัดการ Router พร้อมทั้งแสดงตัวอย่างการใช้งาน Authentication และ Authorization

เพื่อให้เนื้อหาเข้าใจได้ง่ายขึ้น ผู้เขียนได้ยกตัวอย่างโค้ดจริงจำนวนมากพร้อมคำอธิบายโดยละเอียด หนังสือเล่มนี้จึงเหมาะสำหรับผู้สนใจ นักเรียนนักศึกษา ตลอดจนผู้ที่มีประสบการณ์ในการพัฒนาเว็บแอปพลิเคชันด้วย JavaScript มาก่อน

สุดท้ายนี้ ผู้เขียนหวังเป็นอย่างยิ่งว่า หนังสือเล่มนี้จะช่วยให้ผู้อ่านสามารถนำความรู้เกี่ยวกับ TypeScript และ Node.js ไปประยุกต์ใช้ในงานจริงได้อย่างมีประสิทธิภาพ หากมีข้อผิดพลาดประการใด ผู้เขียนยินดีจักปรับปรุงแก้ไขให้สมบูรณ์ยิ่งขึ้นในโอกาสต่อ ๆ ไป

จีราวุธ วารินทร์

jeerawuth@me.com

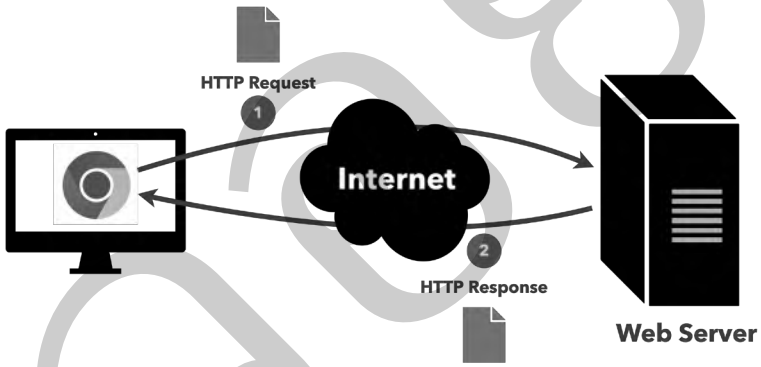
สารบัญ

01
แนะนำ TypeScript

TypeScript (ไทป์สคริปต์) เป็นภาษาคอมพิวเตอร์ที่ถูกพัฒนาโดย Microsoft เปิดตัวครั้งแรกในเดือนตุลาคม 2012 เป็นภาษาคอมพิวเตอร์ที่สร้างขึ้นจาก JavaScript โดยได้เพิ่มคุณสมบัติต่าง ๆ ช่วยให้นักพัฒนาสามารถเขียนโค้ด JavaScript ที่มีโครงสร้างที่ชัดเจนและมีความปลอดภัยมากขึ้น เช่น มีระบบตรวจสอบชนิดข้อมูล (Type Checking) และสนับสนุนการใช้งานของ Object-Oriented Programming (OOP) เต็มรูปแบบ เป็นต้น

เว็บเพจและ JavaScript

เว็บเพจ (Webpage หรือ Web page) คือ เอกสารที่ถูกใช้งานบนอินเทอร์เน็ต และถูกแสดงผลบนเบราว์เซอร์ เมื่อผู้ใช้ต้องการเปิดดูข้อมูลจากอินเทอร์เน็ต จะใช้เว็บเบราว์เซอร์ไปยังที่อยู่ของเว็บเพจ (ใช้ IP Address) เพื่อส่งการร้องขอข้อมูลไปยังเว็บเซิร์ฟเวอร์ (เรียกว่า HTTP Request) เมื่อเว็บเซิร์ฟเวอร์ได้รับการร้องขอ ก็จะส่งผลลัพธ์กลับมายังเบราว์เซอร์ (เรียกว่า HTTP Response) เพื่อนำมาแสดงผลต่อไป



▲ รูปแสดง HTTP Request และ HTTP Response

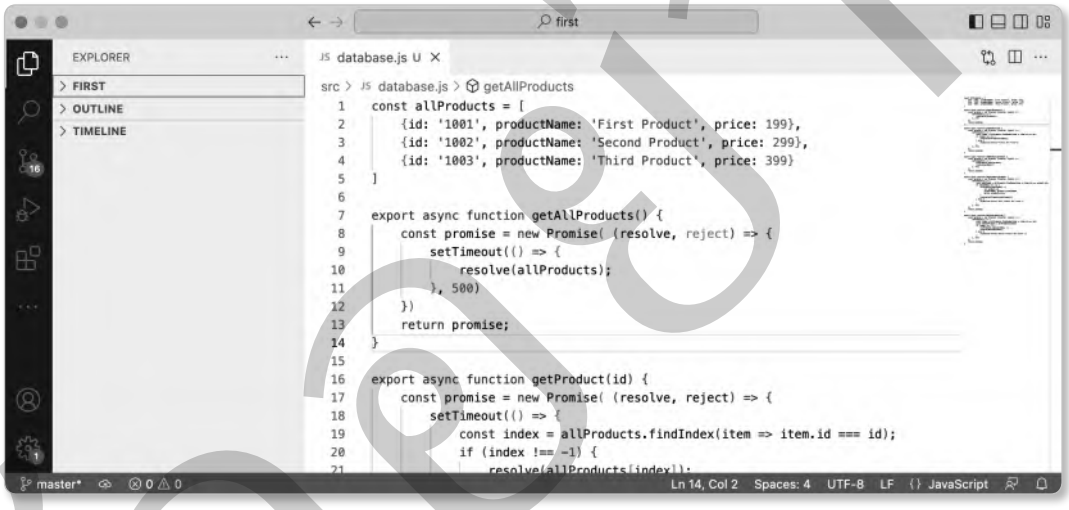
เนื้อหาที่อยู่ในเว็บเพจอาจประกอบด้วยข้อความ ลิงก์ รูปภาพ หรือมีเดียประเภทต่าง ๆ ดังนั้น ข้อมูลเว็บเพจที่ถูกนำมาแสดงผลบนเบราว์เซอร์จึงประกอบไปด้วยไฟล์ประเภทต่าง ๆ มากมาย แต่โดยพื้นฐานแล้ว เว็บเพจจะประกอบด้วยไฟล์หลักอยู่ 3 ประเภท ได้แก่ HTML, CSS และ JavaScript

- **HTML** คือ ไฟล์ที่ใช้กำหนดโครงสร้างของเว็บเพจ บอกให้ทราบว่าภายในเว็บเพจจะมีเนื้อหา และมีจัดเรียงลำดับอย่างไร ไฟล์ HTML จะมีนามสกุลไฟล์เป็น .htm หรือ .html
- **CSS** คือ ไฟล์ที่ใช้ตกแต่งหน้าตาเว็บเพจ เช่น กำหนดขนาดและสีตัวอักษร, กำหนดขนาดรูปภาพ, จัดตำแหน่งการแสดงผลบนอุปกรณ์ที่มีขนาดหน้าจอต่างกัน ฯลฯ ไฟล์ CSS จะมีนามสกุลไฟล์เป็น .css
- **JavaScript** คือ ไฟล์ซึ่งทำให้เว็บเพจสามารถโต้ตอบกับผู้ใช้ได้ หรือทำให้เว็บเพจมีความสามารถบางอย่างเพิ่มขึ้นมา ซึ่งไฟล์ JavaScript จะมีนามสกุลไฟล์เป็น .js

การเขียนโค้ด JavaScript จะมีความสัมพันธ์โดยตรงกับโค้ด HTML และ CSS ที่อยู่ในเว็บเพจ เช่น เขียนโค้ด JavaScript เพื่อแสดงหรือซ่อนเนื้อหาที่อยู่ใน HTML หรือเขียนโค้ด JavaScript เพื่อเลือกใช้งาน CSS สำหรับปรับแต่งหน้าตาเว็บเพจในแบบที่ต้องการ

จุดประสงค์หลักของ JavaScript คือ การทำให้เว็บเพจสามารถโต้ตอบกับผู้ใช้ได้ หรือมีความสามารถบางอย่างเพิ่มขึ้น เช่น ใช้ JavaScript เพื่อขยายรูปเมื่อผู้ใช้เลื่อนเมาส์ไปที่รูปภาพ, ใช้ JavaScript เพื่อสร้างแอนิเมชันเลื่อนข้อความจากซ้ายไปขวา, ใช้ JavaScript เพื่อนำเนื้อหาใหม่มาแทนที่เนื้อหาเดิมโดยไม่ต้องโหลดเว็บเพจใหม่ เป็นต้น

JavaScript เป็นภาษาคอมพิวเตอร์ที่ออกแบบมาสำหรับใช้กับเว็บเพจ ดังนั้น โค้ด JavaScript จึงถูกรันเพื่อใช้งานบนเบราว์เซอร์เป็นหลัก นอกจากนั้น JavaScript ยังสามารถใช้กับสภาพแวดล้อมอื่น ๆ ที่ไม่ใช่เบราว์เซอร์ได้อีกด้วย เช่น Node.js และ Apache CouchDB เป็นต้น



▲ รูปแสดงตัวอย่างโค้ดคำสั่ง JavaScript

จาก JavaScript สู่ TypeScript

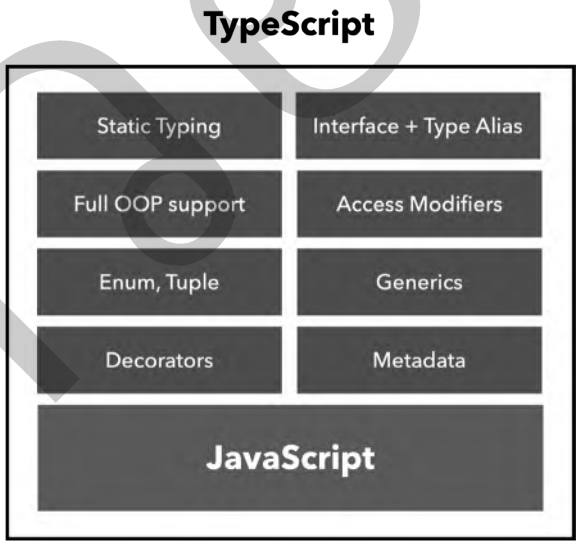
แม้ว่า JavaScript จะเป็นภาษาหลักสำหรับใช้จัดการกับเว็บเพจบนเบราว์เซอร์ แต่การใช้ JavaScript กับแอปพลิเคชันขนาดใหญ่ หรือกับแอปพลิเคชันที่ต้องพัฒนาเป็นทีมนั้นค่อนข้างยุ่งยาก เนื่องจาก JavaScript ไม่มีระบบตรวจสอบชนิดข้อมูล (Type Checking) เมื่อโค้ดมีปัญหามักตรวจสอบข้อผิดพลาดได้ยาก โดยเฉพาะการนำโค้ดจากนักพัฒนาอื่นหรือโค้ดจากภายนอกมาแก้ไขต่อ ก็ยิ่งปวดหัวมากไปกว่าเดิม

เพื่อแก้ปัญหาข้างต้น ภาษาคอมพิวเตอร์ที่ชื่อว่า TypeScript จึงถูกสร้างขึ้น โดยนำ JavaScript มาเพิ่มคุณสมบัติใหม่และทำให้มีประสิทธิภาพยิ่งขึ้น เช่น ข้อมูลจะต้องมีชนิดข้อมูลที่แน่นอน (Type Safety) สามารถตรวจสอบชนิดข้อมูลโดยอัตโนมัติได้ (Type Checking) และรองรับ OOP (Object Oriented Programming) เต็มรูปแบบ เป็นต้น

ต่อไปนี้เป็นตารางเปรียบเทียบความแตกต่างระหว่าง TypeScript และ JavaScript

	TypeScript	JavaScript
กำหนดชนิดข้อมูลที่แน่นอน (Type Safety)	มี	ไม่มี
ระบบตรวจสอบชนิดข้อมูล (Type Checking)	มี	ไม่มี
การใช้งาน OOP	สนับสนุนเต็มรูปแบบ	มีใช้งาน แต่ไม่ได้สนับสนุนเต็มรูปแบบ
สนับสนุน Refactoring	สนับสนุน และปลอดภัย	สนับสนุน แต่ไม่มีระบบตรวจสอบความปลอดภัย
การใช้งาน JavaScript ไลบรารี	จะต้องมี Type Definition Files เพื่อกำหนดชนิดข้อมูลที่อยู่ในไลบรารีนั้น ๆ	รองรับ
นามสกุลไฟล์	.ts	.js

จากตารางข้างต้น สามารถสรุปได้ว่า TypeScript คือ การนำ JavaScript มาเพิ่มระบบตรวจสอบชนิดข้อมูล (Type System) รวมถึงคุณสมบัติอื่น ๆ ที่ช่วยให้การเขียน JavaScript มีประสิทธิภาพและปลอดภัยยิ่งขึ้น กล่าวได้ว่า TypeScript ก็คือ Super Set ของ JavaScript นั่นเอง

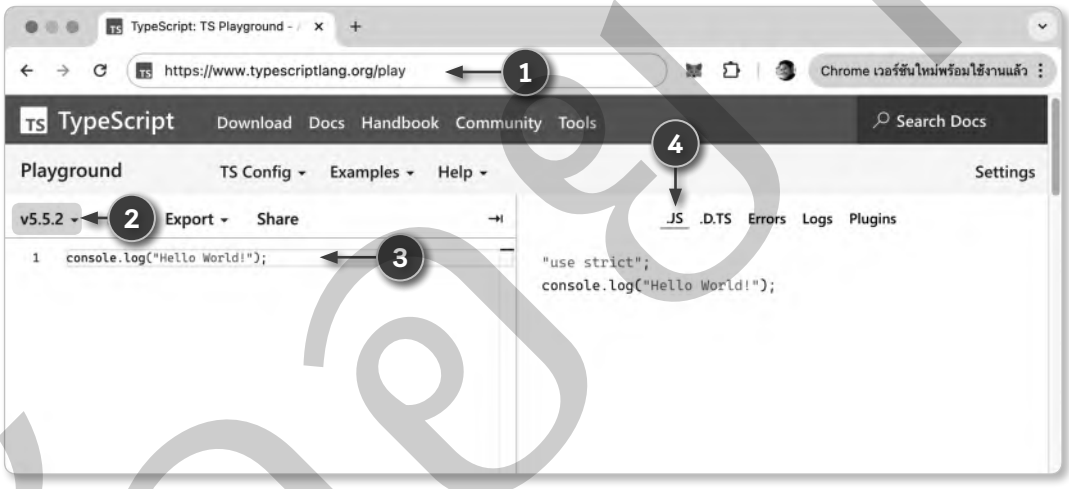


▲ รูปแสดงตัวอย่างคุณสมบัติ TypeScript ที่มากกว่า JavaScript

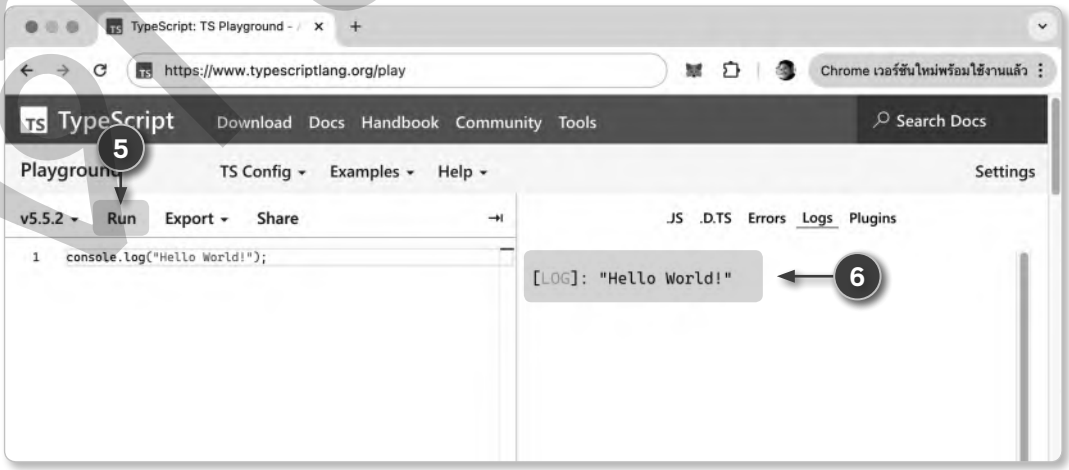
ทดสอบ TypeScript ผ่านเว็บไซต์

การศึกษา TypeScript สามารถทำได้หลายช่องทาง วิธีหนึ่งที่นิยมเพิ่มขึ้นมากในปัจจุบัน คือ การศึกษา TypeScript ผ่านทาง Playground ซึ่งอยู่ที่เว็บเพจ https://typescriptlang.org/play ผู้อ่านสามารถกรอกชุดคำสั่ง TypeScript ลงในเบราว์เซอร์ ผลลัพธ์การรันชุดคำสั่งก็จะปรากฏบนเบราว์เซอร์ ดังตัวอย่าง

1. เปิดเบราว์เซอร์และไปยังแอดเดรส https://typescriptlang.org/play
2. เลือกเวอร์ชันของ TypeScript แนะนำให้เลือกเป็นเวอร์ชันล่าสุด
3. กรอกโค้ดคำสั่ง TypeScript โดยในตัวอย่างเป็นการพิมพ์ข้อความ Hello World! ออกมาที่หน้าต่างคอนโซล
4. คลิกเมนู .JS เพื่อตรวจสอบผลลัพธ์ที่เป็น JavaScript โดยโค้ด TypeScript ที่อยู่ในหน้าต่างด้านซ้ายจะถูกคอมไพลเลอร์แปลงเป็น JavaScript



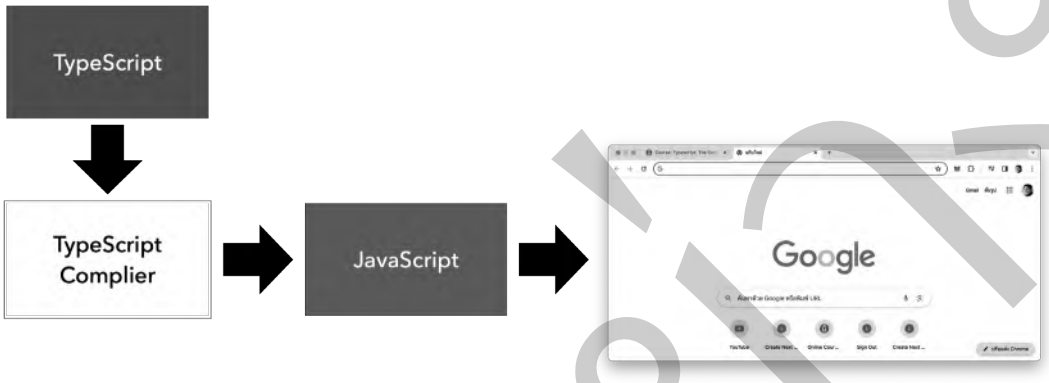
5. คลิกเมนู Run เพื่อรันโค้ด
6. สังเกตผลลัพธ์ข้อความ “Hello World!” ที่ได้ในหน้าต่างด้านขวามือ



เบราว์เซอร์ไม่รู้จัก TypeScript

เนื่องจากเบราว์เซอร์รู้จัก JavaScript แต่ไม่รู้จัก TypeScript ดังนั้น คำสั่ง TypeScript จึงไม่สามารถใช้งานกับเบราว์เซอร์ได้โดยตรง

เพื่อให้เบราว์เซอร์เข้าใจคำสั่งต่าง ๆ ที่เขียนด้วย TypeScript เราจะต้องติดตั้ง TypeScript คอมไพเลอร์ลงในเครื่องคอมพิวเตอร์ เพื่อทำหน้าที่แปลงโค้ด TypeScript ให้กลายเป็น JavaScript เสียก่อน



▲ รูปแสดงการคอมไพล์ TypeScript ให้เป็น JavaScript

ติดตั้ง TypeScript

การติดตั้ง TypeScript ลงในคอมพิวเตอร์สามารถแบ่งได้เป็น 3 ขั้นตอน ดังนี้

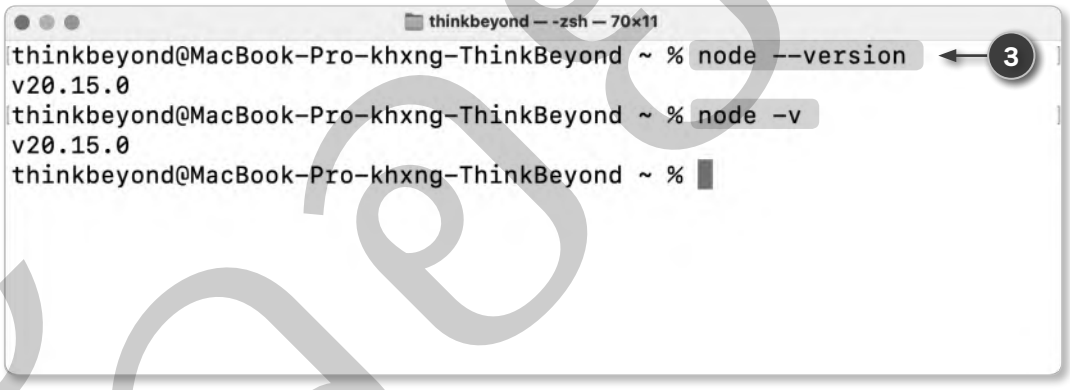
ขั้นตอนที่ 1 ดาวน์โหลดและติดตั้ง Node.js

Node.js เป็นสภาพแวดล้อมที่ช่วยให้นักพัฒนาสามารถใช้ JavaScript พัฒนาแอปพลิเคชันแยกเป็นอิสระจากเบราว์เซอร์ได้ (JavaScript Runtime Environment) โดยหลังจากติดตั้ง Node.js เรียบร้อย จะได้ npm ซึ่งเป็นเครื่องมือสำหรับจัดการกับแพ็คเกจมาใช้งาน

1. ไปยังเว็บไซต์ https://nodejs.org จากนั้นคลิกปุ่มดาวน์โหลด Node.js

2. ดับเบิลคลิกไฟล์ เพื่อติดตั้ง Node.js ไปยังคอมพิวเตอร์

3. หลังจากติดตั้งเรียบร้อย ให้ทดสอบว่าการติดตั้งนั้นถูกต้องหรือไม่ โดยเปิดหน้าต่าง Terminal (หรือหน้าต่าง Command Prompt) จากนั้นพิมพ์คำสั่ง node --version หรือ node -v เพื่อดูหมายเลขเวอร์ชัน หากพบหมายเลขเวอร์ชัน การติดตั้ง Node.js ถือว่าสำเร็จเรียบร้อย



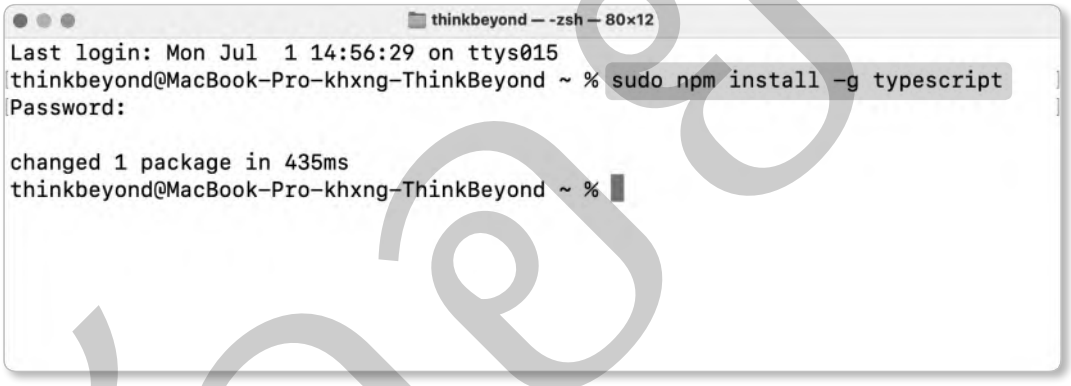
ขั้นตอนที่ 2 ติดตั้ง TypeScript

เนื่องจากเบราว์เซอร์ไม่รู้จัก TypeScript รู้จักแต่ JavaScript ดังนั้น การนำ TypeScript มาใช้งานกับเบราว์เซอร์ เราจะต้องติดตั้ง TypeScript คอมไพลเลอร์ เพื่อทำหน้าที่แปลงจาก TypeScript ให้เป็น JavaScript เสียก่อน

การติดตั้ง TypeScript คอมไพลเลอร์ สามารถติดตั้งผ่าน npm ดังนี้

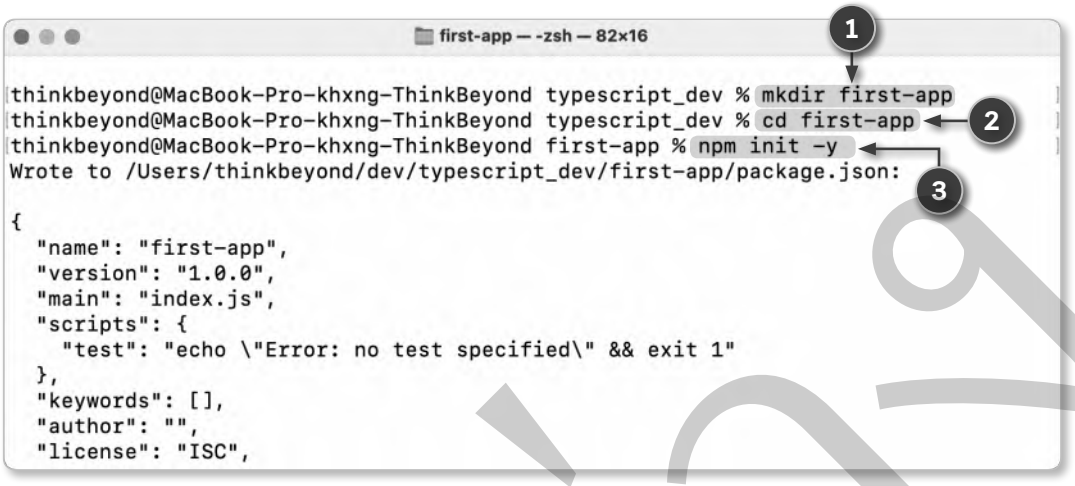
แบบที่ 1 ติดตั้ง TypeScript ในแบบโกลบอล จุดประสงค์คือ ต้องการให้ทุก ๆ แอปพลิเคชันสามารถเข้าใช้งาน TypeScript คอมไพลเลอร์ได้

1. เปิดหน้าต่าง Terminal
2. การติดตั้งในแบบโกลบอลในระบบปฏิบัติการวินโดวส์ ให้พิมพ์คำสั่ง npm install -g typescript และกดปุ่ม <Enter> ส่วนระบบปฏิบัติการแมคโอเอสให้พิมพ์คำสั่ง sudo npm install -g typescript กดปุ่ม <Enter> จากนั้นกรอกรหัสผ่านของผู้ดูแลระบบ

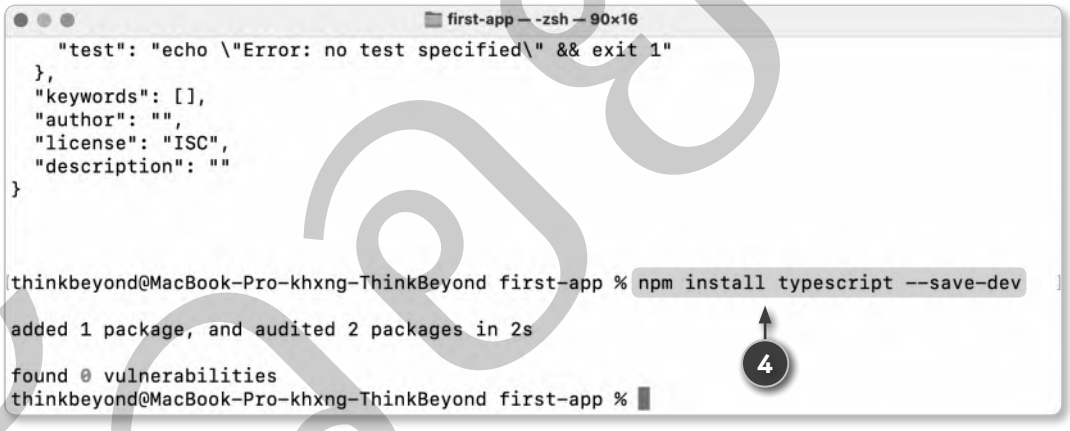


แบบที่ 2 ติดตั้ง TypeScript ในแบบโลคอล จุดประสงค์คือ ต้องการติดตั้ง TypeScript คอมไพลเลอร์ให้ใช้งานได้เฉพาะโฟลเดอร์ปัจจุบันเท่านั้น

1. เปิดหน้าต่าง Terminal แล้วสร้างโฟลเดอร์ใหม่โดยใช้คำสั่ง mkdir เช่น ถ้าต้องการสร้างโฟลเดอร์ first-app ให้พิมพ์คำสั่ง mkdir first-app
2. เข้าไปยังโฟลเดอร์ที่เพิ่งสร้างขึ้น เช่น หากต้องการเข้าไปยังโฟลเดอร์ first-app ก็ให้พิมพ์คำสั่ง cd first-app
3. สร้างไฟล์ package.json โดยการพิมพ์คำสั่ง npm init -y



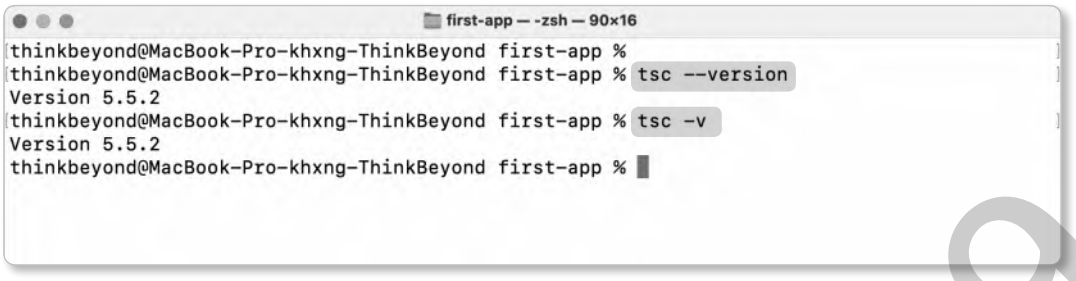
4. ติดตั้ง TypeScript ลงในโฟลเดอร์ปัจจุบัน โดยพิมพ์คำสั่ง npm install typescript --save-dev กดปุ่ม <Enter>



คำสั่ง sudo npm install -g typescript ที่ใช้เพื่อติดตั้ง TypeScript ในระบบปฏิบัติการแมคโอเอสตามข้างต้นนั้น มีความหมายดังนี้

- คำสั่ง sudo จะใช้เมื่อต้องการขออนุญาตจากผู้ดูแลระบบ (sudo ย่อมาจาก Super User Do)
- คำสั่ง -g เป็นการสั่งให้ติดตั้งในแบบ Global ที่ทุก ๆ แอปพลิเคชันสามารถเข้าใช้งาน TypeScript คอมไพลเลอร์ได้
- คำสั่ง typescript เป็นการติดตั้ง TypeScript คอมไพลเลอร์

หลังจากติดตั้ง TypeScript ลงในเครื่องคอมพิวเตอร์เรียบร้อย ให้ตรวจสอบเวอร์ชัน โดยใช้คำสั่ง tsc --version หรือ tsc -v หากพบหมายเลขเวอร์ชันของ TypeScript แสดงว่าการติดตั้งเสร็จสิ้นสมบูรณ์แล้ว



▲ รูปแสดงการตรวจสอบเวอร์ชันของ TypeScript

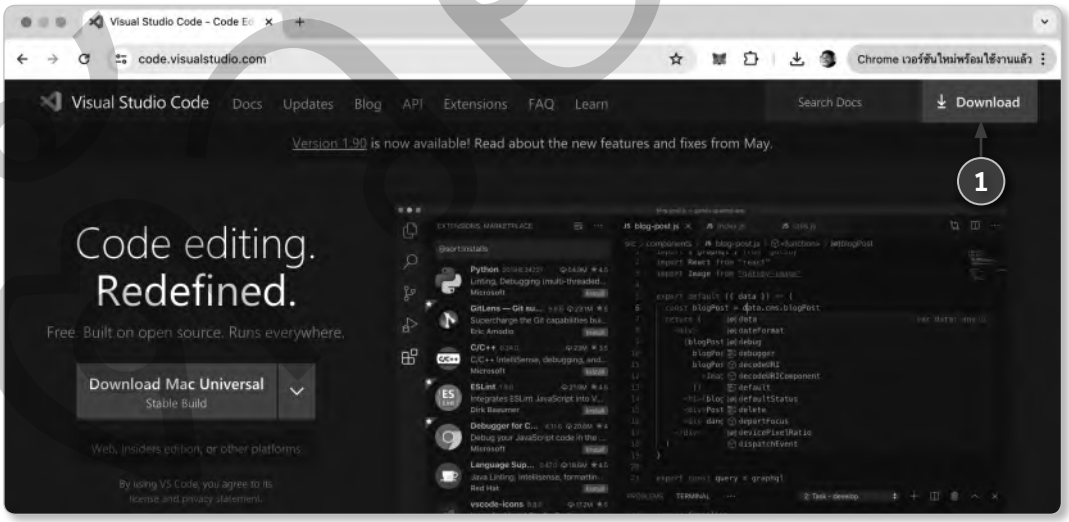
ขั้นตอนที่ 3 ติดตั้ง Code Editor หรือ IDE

Visual Studio Code หรือเรียกแบบย่อว่า VS Code เป็นโปรแกรมสำหรับใช้แก้ไขและทดสอบโค้ด (เรียกว่า Code Editor) จากทางไมโครซอฟท์

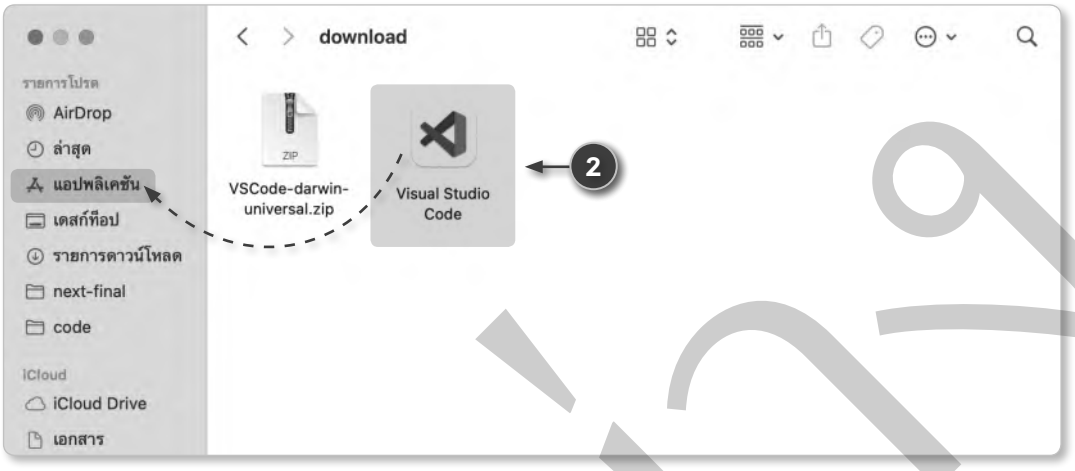
VS Code จัดได้ว่าเป็น Code Editor ยอดนิยมที่มีผู้ใช้งานเป็นจำนวนมากทั่วโลก โดยมีจุดเด่นมากมาย เช่น สามารถดาวน์โหลดไปใช้งานได้ฟรี, สามารถติดตั้งส่วนประกอบเสริมจึงเพิ่มความสามารถได้ไม่จำกัด, มีตัวช่วยกรอกคำสั่ง, มีการใช้สีเพื่อแยกโค้ดคำสั่งแต่ละแบบจึงทำให้แก้ไขโค้ดหรืออ่านโค้ดได้ง่าย และที่สำคัญคือสามารถใช้กับภาษาคอมพิวเตอร์ได้เกือบทุกภาษา ไม่ว่าจะเป็น C, Java, PHP, Python, JavaScript, TypeScript และภาษาอื่น ๆ อีกมากมาย

การติดตั้ง Visual Studio Code เพื่อใช้กับ TypeScript มีขั้นตอน ดังนี้

1. ไปที่เว็บไซต์ https://code.visualstudio.com คลิกปุ่ม Download



2. ติดตั้ง VS Code ในตัวอย่างเป็นการติดตั้งในแมคโอเอสด้วยวิธีลากไฟล์ Visual Studio Code ไปไว้ในโฟลเดอร์แอปพลิเคชัน

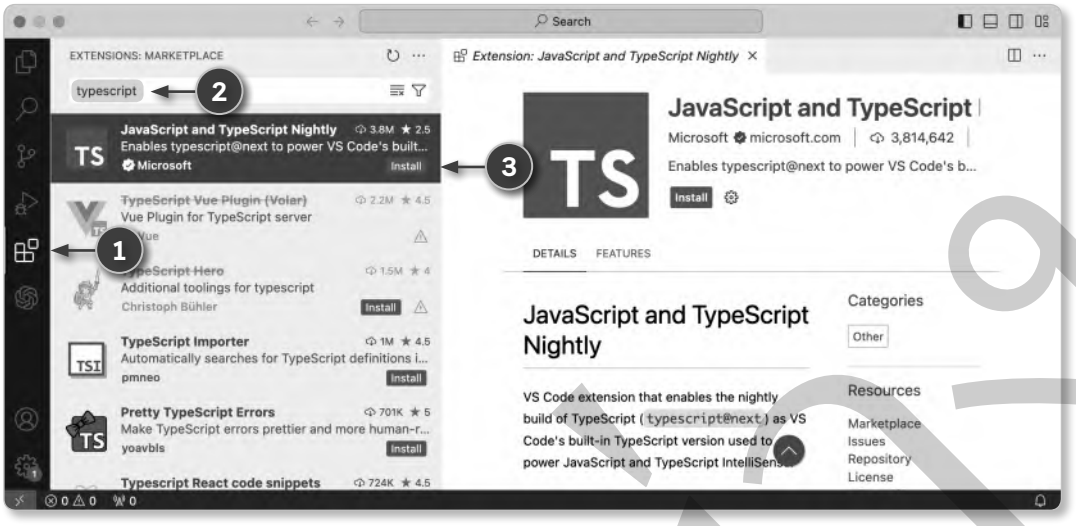


ติดตั้งส่วนประกอบเสริมไปยัง VS Code

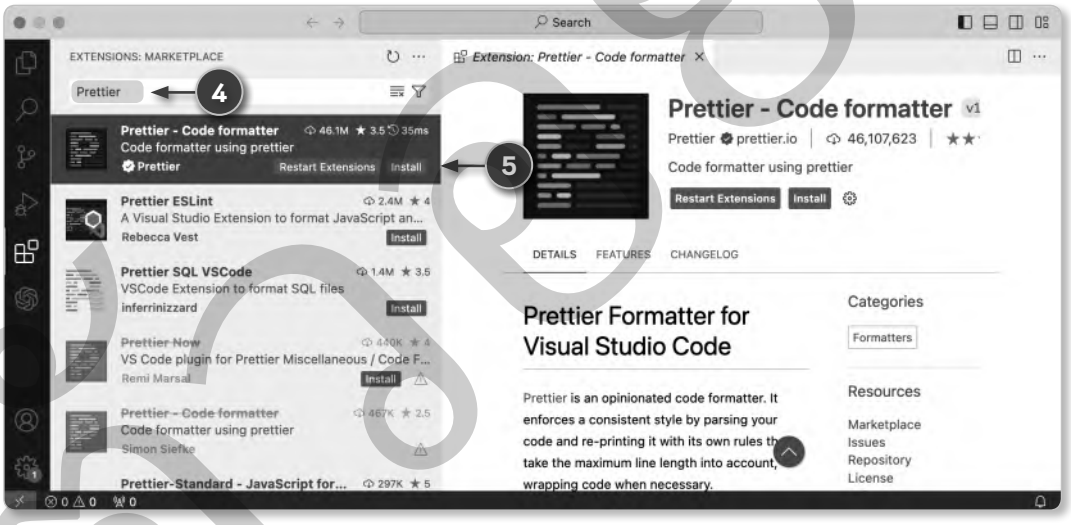
ในตอนแรก Visual Studio Code จะยังไม่มีตัวช่วยเหลือสำหรับการเขียน TypeScript โดยเฉพาะ ดังนั้นเราต้องติดตั้งส่วนประกอบเสริม (Extensions) เข้าไป โดยในหนังสือเล่มนี้ได้ติดตั้งส่วนประกอบเสริม ดังนี้

- JavaScript and TypeScript Nightly สำหรับใช้ทดสอบ TypeScript เวอร์ชันล่าสุด โดยสามารถเลือกเวอร์ชันของ TypeScript ที่ต้องการใช้งานได้
- Prettier - Code formatter ช่วยจัดเรียงโค้ด เมื่อเราสั่งบันทึกโค้ดก็จะทำให้โค้ดสวยงามโดยอัตโนมัติ เช่น จัดการเยื้องของโค้ด ปรับระยะห่างของช่องว่าง จัดระยะของเครื่องหมายปีกกา เป็นต้น

1. เปิดโปรแกรม Visual Studio Code จากนั้นให้คลิกปุ่ม  หรือกดปุ่ม <Ctrl+Shift+X> (ในแมคโอเอสให้กดปุ่ม <Cmd+Shift+X>)
2. ที่ช่องค้นหา ให้กรอก TypeScript เพื่อค้นหาส่วนประกอบเสริมสำหรับ TypeScript
3. ให้คลิกปุ่ม Install ที่ JavaScript and TypeScript Nightly



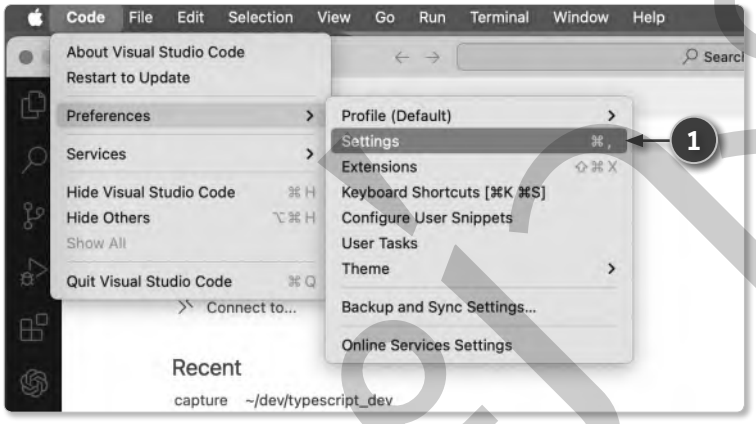
4. ที่ช่องค้นหาให้กรอก Prettier
5. ให้คลิกปุ่ม Install ที่ Prettier - Code formatter



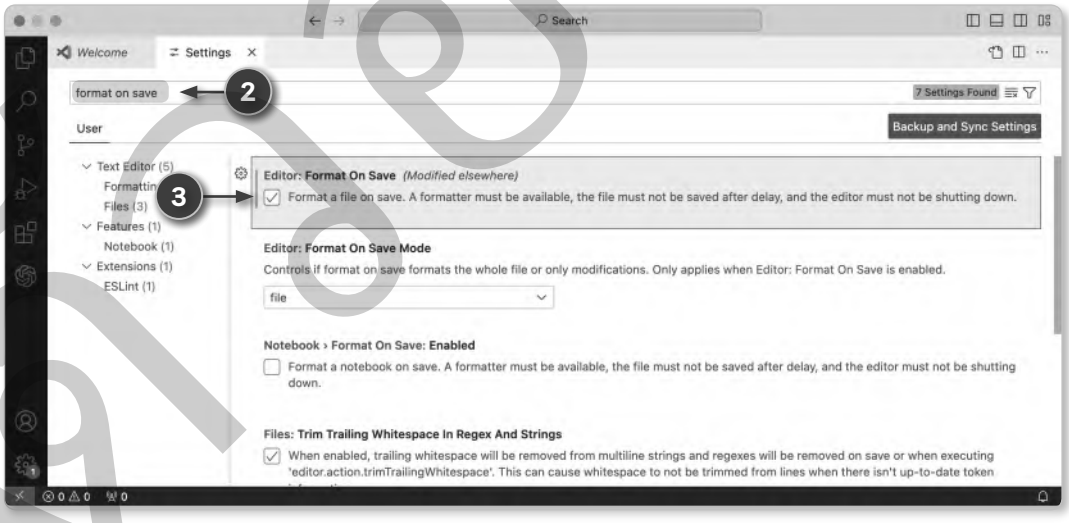
จัดเรียงโค้ดอัตโนมัติทุกครั้งที่บันทึกไฟล์

สามารถปรับแต่ง VS Code ให้เหมาะสมกับสภาพแวดล้อม หรือตามความถนัดที่แตกต่างกันได้ เช่น หลังจากติดตั้งส่วนขยาย Prettier สามารถเลือกได้ว่าจะให้ปรับแต่งโค้ดให้สวยงามเมื่อใด เช่น ถ้าต้องการให้ปรับแต่งทุกครั้งเมื่อมีการบันทึกไฟล์ ก็ให้กำหนดค่าด้วยขั้นตอน ต่อไปนี้

1. เปิดโปรแกรม VS Code แล้วคลิกเมนู Preference > Settings



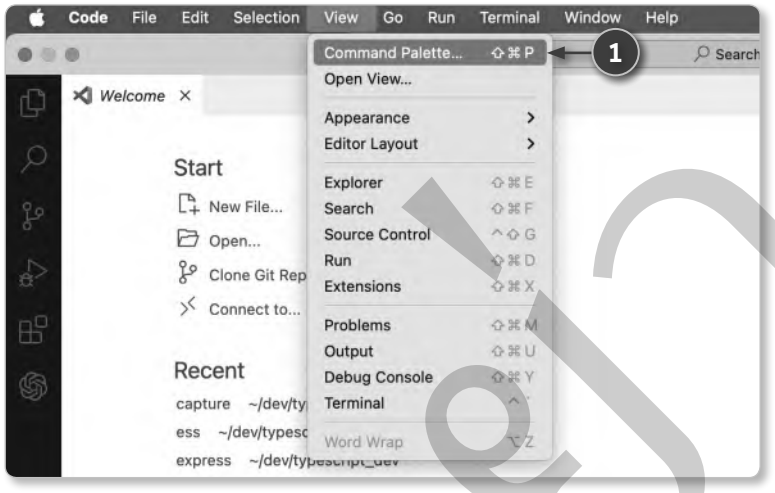
2. เพื่อความรวดเร็ว สามารถค้นหาการตั้งค่าที่ต้องการได้ เช่น พิมพ์ format on save
3. คลิกถูกหน้า Editor: Format a file on save



กำหนด code path ให้กับ VS Code

เพื่อให้สามารถใช้คำสั่ง code . เป็นทางลัดในการเปิด VS Code จากหน้าต่าง Terminal จะต้องกำหนดพาธด้วยขั้นตอน ดังนี้

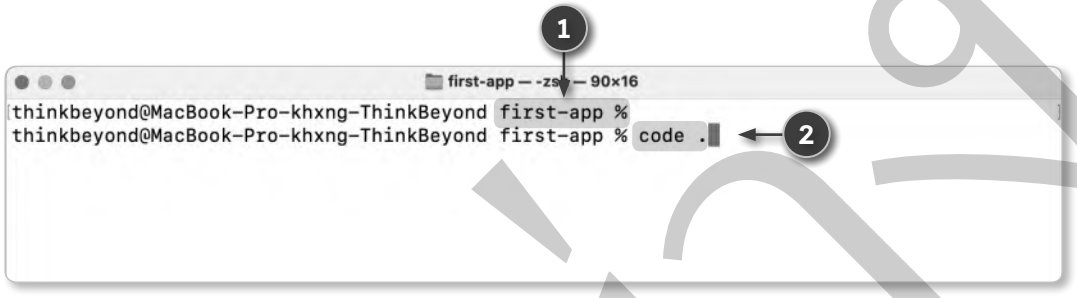
1. เปิดโปรแกรม VS Code แล้วคลิกเมนู View > Command Palette



2. ค้นหาคำสั่งที่เกี่ยวกับการติดตั้งพาธ เช่น พิมพ์ install code path
3. จะปรากฏคำสั่งที่เกี่ยวข้อง ให้คลิกที่ Shell Command: Install ‘code’ command in PATH (ในระบบปฏิบัติการแมคโอเอส จะต้องยืนยันว่าเป็นผู้ดูแลระบบด้วยการกรอกรหัสผ่านอีกครั้ง)

หลังจากทำตามขั้นตอนข้างต้นแล้ว เราสามารถเปิดโฟลเดอร์ หรือโปรเจ็กต์ โดยใช้คำสั่ง code . ได้ ดังตัวอย่าง

1. เปิดหน้าต่าง Terminal (ในแมคโอเอส) หรือหน้าต่าง Command Line (ในวินโดวส์) เข้าไปยังโฟลเดอร์ที่ต้องการ เช่น พิมพ์คำสั่ง cd first-app เพื่อเข้าไปยังโฟลเดอร์ first-app
2. พิมพ์คำสั่ง code . และกดปุ่ม <Enter> เพื่อสั่งให้เปิดโฟลเดอร์ปัจจุบันด้วยโปรแกรม Visual Studio Code (ในที่นี้คือโฟลเดอร์ first-app)

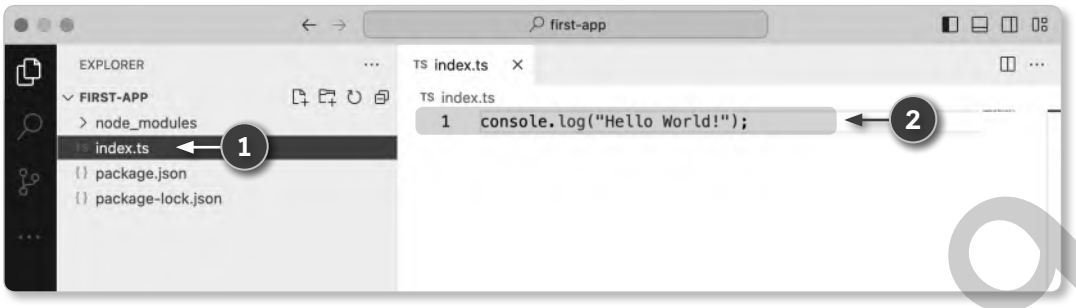


3. ผลลัพธ์ VS Code ถูกเรียกใช้งาน และเปิดโฟลเดอร์ first-app โดยอัตโนมัติ

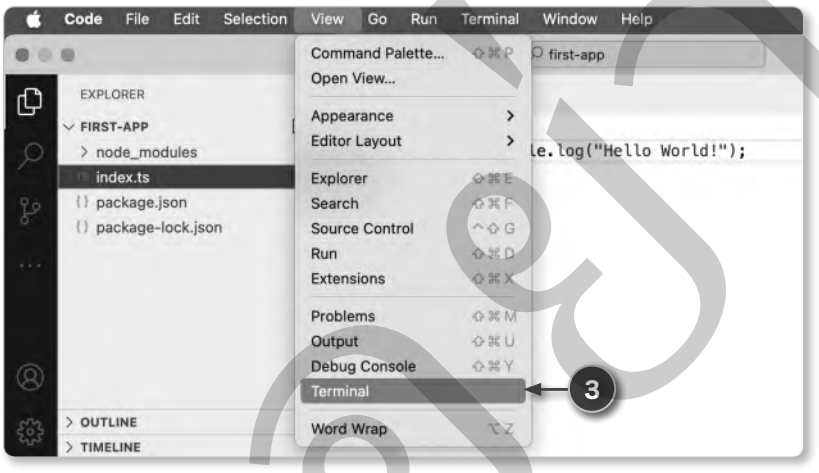
คอมไพล์จาก TypeScript เป็น JavaScript

หลังจากติดตั้ง TypeScript ลงในโปรเจ็กต์ เราสามารถใช้คำสั่ง tsc เพื่อคอมไพล์โค้ดในไฟล์ TypeScript ให้กลายเป็น JavaScript

1. เปิดโปรเจ็กต์ด้วย Visual Studio Code สร้างไฟล์ index.ts ไว้ในโปรเจ็กต์ (ไฟล์นามสกุล .ts หมายถึง TypeScript)
2. กรอกโค้ด TypeScript ลงไปในไฟล์ index.ts เช่น console.log(“Hello World!”);

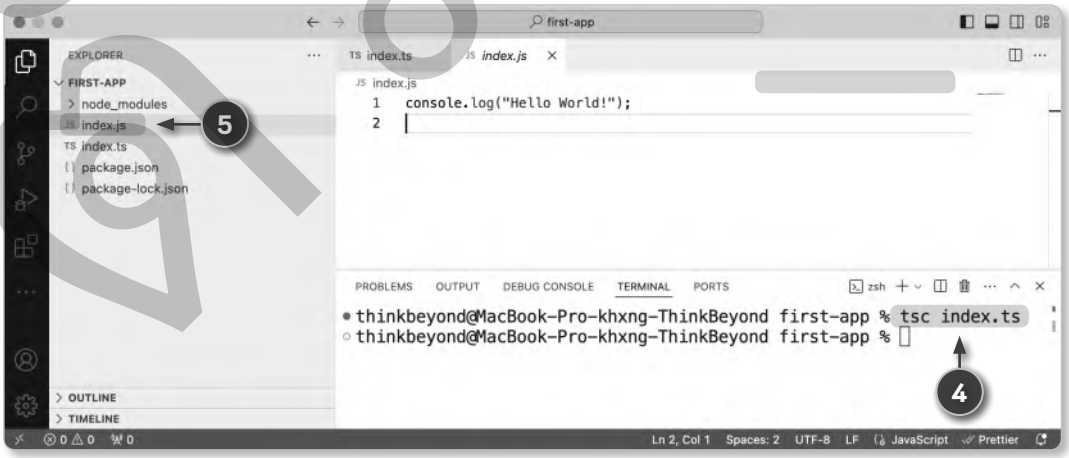


3. เปิดหน้าต่าง Terminal โดยคลิกเมนู View>Terminal



4. กรอกคำสั่ง tsc ตามด้วยชื่อไฟล์ TypeScript ที่ต้องการ เช่น tsc index.ts

5. ผลลัพธ์ไฟล์ index.js จะถูกสร้างโดยอัตโนมัติ ซึ่งเป็นไฟล์ JavaScript ที่ถูกคอมไพล์มาจากไฟล์ index.ts



Think Beyond Develop

ใช้ TypeScript ร่วมกับ Node.js

สร้างได้ทั้ง Frontend และ Backend

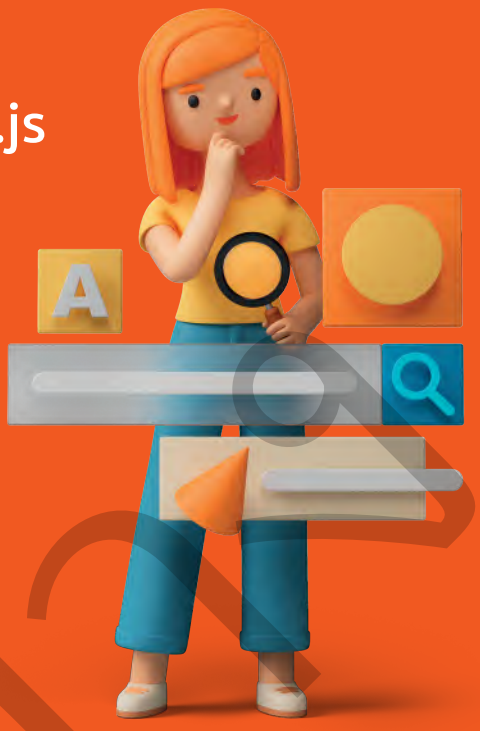
ด้วยเนื้อหาที่แน่นและกระชับ

ปรับแต่งให้เข้าใจได้ง่าย

อ่านและทำตามได้แบบ

Step-By-Step

เก่งได้ แม้ไม่มีพื้นฐานมาก่อน



- สรุปพื้นฐาน TypeScript ทั้งหมดอย่างเป็นระบบ
- อธิบาย TypeScript ด้วยตัวอย่าง พร้อมคำอธิบายในทุกขั้นตอน
- การใช้งาน Enum และ Tuple
- วิธีใช้งานอาร์เรย์และอาร์เรย์เมธอดโดยละเอียด
- กำหนดโครงสร้างข้อมูลโดยใช้ออบเจ็กต์ อินเตอร์เฟส และคลาส
- การใช้งาน Optional Property และ Optional Parameter
- การใช้งาน TypeScript ขั้นสูง เช่น Generics, Decorators และ Metadata

- การทำงานกับ async/await และ Promise อย่างมีประสิทธิภาพ
- วิธีสร้างโปรเจ็กต์ด้วย TypeScript และ Node.js
- ตัวอย่างการอ่านเขียนไฟล์ใน Node.js
- การใช้งานไลบรารีและการติดตั้ง Type Definition Files
- พื้นฐานเกี่ยวกับ API, Middleware และ Router
- การสร้าง API ด้วย Express และ TypeScript
- วิธีจัดการ Authentication และ Authorization ในแอปพลิเคชัน

ISBN (eBook) 885-909-931-024-6

ราคา 480 บาท