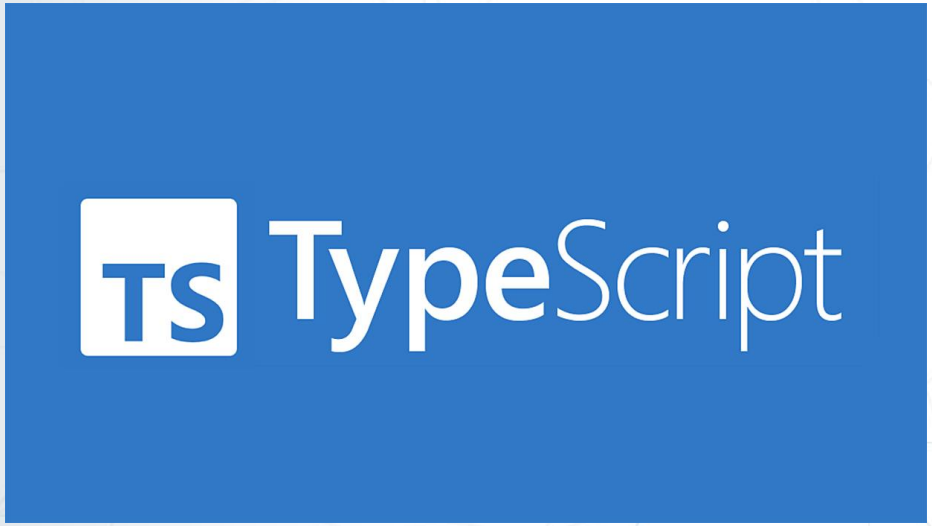
Introduction to TypeScript

Web Beginner Course



TypeScript คืออะไร?

TypeScript คือภาษาโปรแกรมที่ต่อยอดจาก JavaScript เพื่อรองรับการ coding แบบระบุ static type ต่างๆ รวมถึง class และ interface ได้

ข้อดีของ TypeScript

- มี Feature ที่มีประโยชน์อย่าง Code Navigation, Bug Prevention และ Maintainability of Code ไว้ให้แล้ว
- รองรับ Static Type Annotation หรือ Static Typing
- รองรับ Interfaces, Sub-Interfaces, Classes, Subclasses และ ES6
- มีความสามารถด้าน Object-Oriented Programming (OOP) พร้อมทั้งการ Inheritance ของ Private Members และ Interfaces
- มี IDE ให้ใช้อยู่มากมาย

ข้อเปรียบเทียบระหว่าง TypeScript กับ JavaScript

- JavaScript เป็นภาษาที่สามารถเรียนรู้ได้ง่าย ในขณะที่ TypeScript ต้องอาศัยการเรียนรู้ที่มากกว่าและต้องมีความรู้เกี่ยวกับ Script ก่อน
- TypeScript รองรับ Static Typing ซึ่งทำให้คุณสามารถตรวจสอบความถูกต้องของ Type ได้ในระหว่างที่ Compile แต่ในขณะที่ JavaScript เป็น Dynamic Typing ซึ่งตัวแปรแต่ละตัวสามารถเป็น type ใดๆ ก็ได้
- Code ของ TypeScript จำเป็นต้องถูก Compile ก่อน แต่สำหรับ JavaScript ไม่ต้อง Compile ก่อน Run

การติดตั้ง TypeScript

ติดตั้ง TypeScript ผ่าน npm ด้วยคำสั่งต่อไปนี้

```
npm install -g typescript
```

เมื่อติดตั้งแล้ว สามารถตรวจสอบเวอร์ชั่นได้ด้วยคำสั่งต่อไปนี้

```
tsc -v
```

เริ่มใช้งาน TypeScript

สร้างไฟล์ใหม่ชื่อ newbie.ts จากนั้นเพิ่มโค้ดต่อไปนี้ลงไปในไฟล์ที่สร้างขึ้น

```
function setDetail(name: string, age: number) {
    document.getElementById('name').innerHTML = name;
    document.getElementById('age').innerHTML = age.toString();
}

const myName: string = 'Your name here';
const myAge: number = 3;

setDetail(myName, myAge);
```

จากนั้น compile โค้ดดังกล่าวให้เป็น JavaScript ที่ Terminal ด้วยคำสั่งต่อไปนี้

```
tsc newbie.ts
```

เริ่มใช้งาน TypeScript (ต่อ)

เมื่อ Compile เสร็จแล้ว จะได้ไฟล์ชื่อ newbie.js ซึ่งเราสามารถทดสอบการทำงานได้โดยการสร้างเพจ HTML ชื่อว่า index.html ซึ่งประกอบด้วยโค้ดต่อไปนี้

```
<html>
  <head>
    I'm just a newbie.
  </head>
  <body>
    <p>
      My name is <span id="name"></span>.<br />I am <span id="age"></span> years
      old. I'm just a newbie.
    </p>
    <script src="newbie.js"></script>
  </body>
</html>
```

จากนั้น double-click ที่ index.html เพื่อเปิดดูบน browser

ชนิดข้อมูลของ TypeScript

ชนิดข้อมูล	คำอธิบาย	ตัวอย่าง
boolean	ชนิดข้อมูลเชิงตรรกะ มีค่าระหว่าง true กับ false เท่านั้น	`let isDone: Boolean = false;`
string	สายอักขระ ใช้เก็บข้อความ	`let color: string = 'red';`

number	ตัวเลขต่างๆ	`let decimal: number = 6;`
		`let hex: number = 0xf00d;`
		`let binary: number = 0b1010;`
		`let octal: number = 0o744;`

Array<type>	แถวลำดับ สำหรับเก็บค่าตาม type ที่ระบุ เกิน 1 ค่า	`// ประกาศด้วย type ตามด้วย []`
		`let list: number[] = [1, 2, 3];`
		`// ประกาศแบบ generic array type`
		`let list: Array<number> = [1, 2, 3];`

Tuple	เหมือน Array แต่ทราบ type ของข้อมูลในบาง index อยู่แล้ว	`let x: [string, number];`
		`// คำสั่งนี้ทำงานได้ปกติ`
		`x = ['hello', 10]; // OK`
		`// ถ้าชนิดข้อมูลไม่ตรงกันจะเกิด error`
		`x = [10, 'hello']; // Error`

ชนิดข้อมูลของ TypeScript (ต่อ)

ชนิดข้อมูล	คำอธิบาย	ตัวอย่าง
enum	ชนิดข้อมูลแบบกำหนดเอง	`enum Color { Red, Green, Blue }` `let c: Color = Color.Green;`

any	เป็นชนิดข้อมูลอะไรก็ได้	`let notSure: any = 4; // number`
		`notSure = 'Maybe a string instead'; // string`
		`notSure = false; // boolean`

void	สำหรับฟังก์ชันที่ไม่ return ค่า	`function warnUser(): void {`
		`  alert('This is my warning message');`
		`}`

null	ค่าว่าง	`let n: null = null;`
undefined	ไม่ระบุชนิดข้อมูล	`let n: undefined = undefined;`

never	ไม่สามารถเกิดขึ้นได้ / ไม่มีจุดจบ	`function infiniteLoop(): never {`
		`  while(true){ }`
		`}`

การประกาศตัวแปรด้วย var

ตัวแปรที่ประกาศด้วย var จะสามารถเข้าถึงได้จากทุกที่ภายใน function, module, namespace หรือ global scope เดียวกัน โดยไม่คำนึงถึง containing block (วงเล็บปีกกา)

```
function f(shouldInitialize: boolean){
    if(shouldInitialize){
        var x = 10;
    }
    return x;
}

f(true); // return ‘10’
f(false); // return ‘undefined’
```

การประกาศตัวแปรด้วย let

ตัวแปรที่ประกาศด้วย let จะสามารถเข้าถึงได้จากภายใน containing block (วงเล็บปีกกา) เท่านั้น

```
function f(input: boolean){
    let a = 100;
    if(input){
        let b = a + 1; // a is still accessible
        return b;
    }
    return b; // Error since b is not declared outside the ‘if’ block
}

f(true); // return ‘100’
f(false); // Error
```

การประกาศตัวแปรด้วย const

ตัวแปรที่ประกาศด้วย const จะสามารถเข้าถึงได้จากภายใน containing block (วงเล็บปีกกา) เช่นเดียวกันกับ let แต่มีข้อแตกต่างตรงที่เมื่อ initialize ค่าไปแล้วจะไม่สามารถเปลี่ยนค่าของตัวแปรได้อีก (เป็นค่าคงที่)

```
const numLivesForCat = 9;

const kitty = {
    name: ‘Aurora’,
    numLives: numLivesForCat
}

// Error since kitty is const
kitty = {
    name: ‘Danielle’,
    numLives: numLivesForCat
}

// However, the value of each key still changeable
kitty.name = ‘Rory’;
kitty.numLives--;
```

Interfaces

เนื่องจากหัวใจสำคัญของ TypeScript คือชนิดของข้อมูล การประกาศ interface เพื่อระบุรูปร่างหน้าตาของ object ว่าอยากให้มี field ใดบ้าง และแต่ละ field เป็น type ใดบ้างจึงมีความสำคัญมากในการควบคุมข้อมูลที่ส่งผ่านระหว่างกันไม่ให้ผิดเพี้ยนไปจากที่ต้องการ

```
interface LabelledValue {
    label: string; // Required key
    size?: number; // Optional key
}

// OK
const label1: LabelledValue = {
    label: 'Label 1'
}

// OK
const label2: LabelledValue = {
    label: ‘Label 2’,
    size: 10
}
```

Classes and Inheritance

TypeScript รองรับการเขียนโปรแกรมแบบ OOP โดยการประกาศ class และสามารถสร้าง subclass ได้ด้วย

```
class Animal {
  private name: string;
  constructor(_name: string) {
    this.name = _name;
  }
  move(distanceInMeter: number = 0) {
    console.log(`${this.name} moves ${distanceInMeter}m.`);
  }
}

class Dog extends Animal {
  bark() {
    console.log("Woof! Woof!");
  }
}

const dog = new Dog('Lucky');
dog.move(10); // ‘Lucky moves 10m.’
dog.bark(); // ‘Woof! Woof!’
```

Functions

TypeScript รองรับการประกาศฟังก์ชันทั้งแบบตั้งชื่อ และแบบไม่ตั้งชื่อ

```
// Named function
function add(a: number, b: number) {
    return a + b;
}

// Anonymous function
let myAdd = function (x, y) { return x + y; };
```

Arrow Function

TypeScript รองรับการประกาศฟังก์ชันแบบ Arrow Function ได้ด้วย โดยข้อดีอีกอย่างหนึ่งของการประกาศแบบดังกล่าวคือสามารถใช้ this เพื่ออ้างอิงถึง field อื่นๆ ที่อยู่ภายใน object scope เดียวกัน

```
let deck = {
    suits: [‘spades’, ‘hearts’, ‘clubs’, ‘diamonds’],
    cards: Array(52),
    createCardPicker: function() {
        // ‘this’ inside the arrow function refers to ‘deck’ object
        return () => {
            let pickedCard = Math.floor(Math.random() * 52);
            let pickedSuit = Math.floor(pickedCard / 13);
            return { suit: this.suits[pickedSuit], card: pickedCard % 13 };
        }
    }
}

let cardPicker = deck.createCardPicker();
let pickedCard = cardPicker();
alert(‘card: ’ + pickedCard.card + ‘ of ’ + pickedCard.suit);
```

Iterators

Built-in type ต่างๆ เช่น Array, Map, String, etc. สามารถใช้งาน for..of และ for..in เพื่อวนลูปค่าออกมาใช้งานได้ โดยข้อแตกต่างในการใช้งานมีดังนี้

```
let list = [4, 5, 6];

// for..in returns an index
for (let i in list) {
    console.log(i); // ‘0’, ‘1’, ‘2’
}

// for..of returns a value
for (let i of list) {
    console.log(i); // ‘4’, ‘5’, ‘6’
}
```

References

- https://www.typescriptlang.org/docs/

End of Chapter