香港中文大學
The Chinese University of Hong Kong

FROM JAVASCRIPT TO TYPESCRIPT

ESTR2106 2022–23 Term 1

Building Web Applications

Dr. Chuck-jee Chau
chuckjee@cse.cuhk.edu.hk

OUTLINE

- JavaScript vs. TypeScript
- Static typing
- Arrays
- Functions
- Objects and Interface
- Self-defined types

JAVASCRIPT VS TYPESCRIPT

- “TypeScript is JavaScript’s runtime with a compile-time type checker” *–from TypeScript Handbook*
- TypeScript (TS) is a superset of JavaScript (JS) which allows optional data type definitions in the code
- TypeScript first appeared in 2012 (v0.8), developed by Microsoft
  - Current version: TypeScript 4.8.2

JAVASCRIPT VS TYPESCRIPT

- Only JavaScript can be executed by web browsers…
- A compiler (transpiler) converts TS code to JS code
  - `tsc` compiler can be installed using **`npm`** (to be introduced later)
  - Online compilers are also available
    - *e.g., https://www.typescriptlang.org/play*
- More efforts is done in editors to ensure correct type when coding, e.g., VS Code
- Basically, all JavaScript behaviours are maintained, yet with data type support

STATIC TYPING

- JavaScript allows dynamic typing: type is determined at code execution
- In TypeScript, the variable declaration allows a ***type annotation***

```
let x:number = 5;
let y:string = "hello";

y = x;
console.log(y);
// errors will be given, yet still executed!
```

- Note: the annotation is ***optional***, so even skipping it the code is still valid TypeScript

STATIC TYPING

- If flexibility in type is needed, use the type **any**

```
let x:number = 5;
let y:any = "hello";

console.log(typeof y); // string
y = x;
console.log(typeof y); // number
console.log(y);
```

- If the type is not specified, and TypeScript fails to infer it from context, the type becomes **any**

ARRAYS

- A TypeScript array can have the same type for all elements

```
let a1:number[] = [5,4,3,2,1];
let a2:Array<Number> = [6,7,8,9,0];
```

- If mixed datatype is necessary, a ***union type*** can be used

```
let a3:(string|number)[] = [1, "two", 3, "four"];
for (i of a3)
  console.log(typeof i);
```

FUNCTIONS

- Functions involve the type annotation of the ***argument***, and the ***return value***

```
function checkTrue(input: number): boolean {
  return input?true:false; //ternary condition
}
console.log(checkTrue(5));
console.log(checkTrue(0));
```

- Contextual typing is done to infer the types of anonymous functions and arrow functions

OBJECTS AND INTERFACE

- Objects can usually contain elements of different type
- An interface can help to provide a shape of expected types

```
interface Student {
  name: string;
  gpa: number;
}

let s1:Student = {name: "chuckjee", gpa: 2.9};
console.log(s1.name);
console.log(s1.gpa);
```

SELF-DEFINED TYPES

- One very useful way to enforce value checking is to use ***Union Types***

```
type odd = 1|3|5|7|9;
let x:odd;

x = 4; // Type '4' is not assignable to type 'odd'.
console.log(typeof(x)); // number
```

- Direct union of common types is also possible

READ FURTHER…

TypeScript in 5 minutes

- https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

TypeScript in VS Code

- https://code.visualstudio.com/docs/languages/typescript

TypeScript Handbook

- https://www.typescriptlang.org/docs/handbook/intro.html