Type Script

Ramesh Maharaddi

History

- 1995: JavaScript is born as LiveScript
- 1997: ECMAScript standard is established
- 2000–2005: XMLHttpRequest, a.k.a. AJAX, gains popularity in app such as Outlook Web Access (2000) and Gmail (2004) and Google Maps (2005).
- 2009: ES5 comes out (this is what most of us use now) with forEach, Object.keys, Object.create
- 2015: ES6/ECMAScript2015 comes out; it has mostly syntactic sugar

Little bit about Typescript

- Typescript is an open-source programming language developed and maintained by Microsoft.

- It is a strict syntactical superset of JavaScript.

- How to install Typescript
  - **npm install -g typescript**

Compiling, Running and Watching

- File extension

  .ts

- Compile Single File

  tsc filename.ts

- Multiple File

  Initialize configuration file using **tsc –init**

  Run **tsc** command this will compile all the ts files in the directory

- Running

  Node filename.js

- Watching

  Tsc --watch

Sample Program console.ts

Create new file with .ts extension

console.ts

```
console.log('Hello from Type Script');
```

Compile ts file

```
tsc console.ts
```

Run it

```
node ts.js
```

Basic Types

Data Type	Description
Boolean	let isDone: boolean = false;
Number	let decimal: number = 6;
String	let color: string = "blue";
Array	let list: number[] = [1, 2, 3]; let list: Array<number> = [1, 2, 3];
Tuple	let x: [string, number]; x = ["hello", 10];

Any	let notSure: any = 4;
	notSure = "maybe a string instead";
	notSure = false; // okay, definitely a boolean

Destructuring

- The destructuring assignment syntax in a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

- Array
- Object

Array destructuring

```
let input = [1, 2];
let [first, second] = input;
console.log(first); // outputs 1
console.log(second); // outputs 2
```

With parameters to a function:

```
function f([first, second]: [number, number]) {
console.log(first); console.log(second);
}

f([1, 2]);
```

Object destructuring

```
let o = { a: "foo", b: 12, c: "bar" };
let { a, b } = o;
```

Property renaming

```
let { a: newName1, b: newName2 } = o;
```

```
let { a, b }: { a: string, b: number } = o;
```

Spread

- The spread operator is the opposite of destructuring. It allows you to spread an array into another array, or an object into another object.

- Spread operator … (three dots)

Spread Example

```
let first = [1, 2];
let second = [3, 4];
let bothPlus = [0, ...first, ...second, 5];
//bothPlus the value [0, 1, 2, 3, 4, 5].
```

You can also spread objects:

```
let defaults = { food: "spicy", price: "$$", ambiance: "noisy" };
let search = { ...defaults, food: "rich" }; //search is { food: "rich", price: "$$", ambiance: "noisy" }
```

Classes

```
class Greeter {
        greeting: string;
        constructor(message: string) {
                this.greeting = message;
        }
        greet() {
                return "Hello, " + this.greeting;
        }
}

let greeter = new Greeter("world");
console.log(greeter.greet());
```

- Public, private, and protected modifiers

- Public by default

Interfaces

```
interface LabelledValue {
        label: string;
}
```

Optional Properties

```
interface SquareConfig {
        color?: string;
        width?: number;
 }
```

ES6 Features

- Default Parameters in ES6
  - var link = function(height = 50, color = 'red', url = 'http://azat.co') { ... }
- Template Literals in ES6
  - var name = `Your name is ${first} ${last}.`
  - var url = `http://localhost:3000/api/messages/${id}`
- Multi-line Strings in ES6
  - var lines = `line 1 \n
    line 2
    line 3`

Arrow Functions in ES6

```
var ids = ['5632953c4e345e145fdf2df8','563295464e345e145fdf2df9']
var messages = ids.map(value => `ID is ${value}`) // implicit return
```

Block-Scoped Constructs Let and Const

```
function varvslet() {
  console.log(i);
  for( var i = 0; i < 3; i++ ) {
    console.log(i); // 0, 1, 2
  };
  console.log(i);
  for( let j = 0; j < 3; j++ ) {
    console.log(j);
  };
  console.log(j);
 }
varvslet();
```

Loops

Data Type	Description

For	var num = 5
	var factorial = 1;
	for(let i = num ; i >= 1; i--) {
	factorial *= i ;
	}
	console.log(factorial);

For…in	var obj = {a:1, b:2, c:3};
	for (var prop in obj) {
	console.log(obj[prop]);
	}

For…of	for (let val of[12 , 13 , 123])
	{
	console.log(val)
	}

Union Types

- A union type describes a value that can be one of several types. We use the vertical bar (|) to separate each type, so number | string | boolean is the type of a value that can be a number, a string, or a Boolean

```
function addHtmlPadding(value: string, padding: string | number){

}


let paddedString = addHtmlPadding('Hello', 10);
let paddedString1 = addHtmlPadding('Hello', '10');
let paddedString2 = addHtmlPadding('Hello', true);
```