Microsoft
Learn

STUDENT AMBASSADOR

Microsoft Learn
Student Ambassadors

Typing With TypeScript

Episode 2 : Operators & Data Structures

Shreya Khandelwal
Nirali Sahoo

About the Speakers:

Nirali Sahoo

- Born and brought up in Odisha, India
- CSE Sophomore in IIIT Bhubaneswar
- Beta MLSA

Shreya Khandelwal

§ Born and brought up in Rajasthan, India
§ Beta MLSA

Microsoft Learn
Student Ambassadors

Microsoft Learn Student Ambassadors Program (MLSA)

"Empower every person and every organization on the planet to achieve more."

A community of like-minded people aiming to learn new skills, solve real-world problems, and build communities across the globe.

Summary of Episode 1

What is TypeScript?

- **open-source**, object-oriented programing language, which is developed and maintained by **Microsoft**

- Superset of the JavaScript language

Why TypeScript?

- Used for both, **frontend** and **backend**

- Error-checking at compile time

Difference between JavaScript and TypeScript

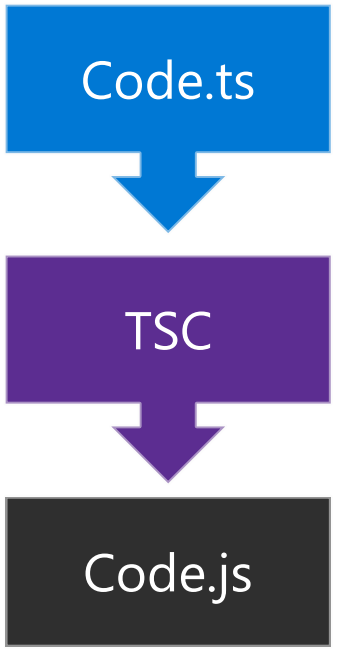
TypeScript	JavaScript
Object oriented programming language	Scripting language
Static typing	Do not support static typing
It compiles the code and highlighted errors during the development time	Interpreted language i.e. errors are highlighted at runtime.

Components of TypeScript

1. Language

- TypeScript language elements
  - Syntax
  - Keywords
  - Type annotations

2. TypeScript Compiler



3. Language Services

- Information to editors for extra features like:
  - IntelliSense
  - Statement Completion
  - Automated refactoring
  - Colorization

ES ECMAScript

- Specification for interpretation of code and languages like JavaScript.
- Browsers and other tools use ECMAScript to interpret JavaScript.
- Need transpilers like Babel to convert modern code to old ECMAScript convention.

Installation and Set-up

Installing TypeScript (via npm)

<u>Requires Node.js</u> (JavaScript Runtime)

Installing Node.js:

https://nodejs.org/en/download/

Installing TS:

npm install -g typescript

Setting up in Visual Studio Code

Compiling:

```
tsc <file_path>
```

Automated compilation:

```
tsc -w <file_path>
```

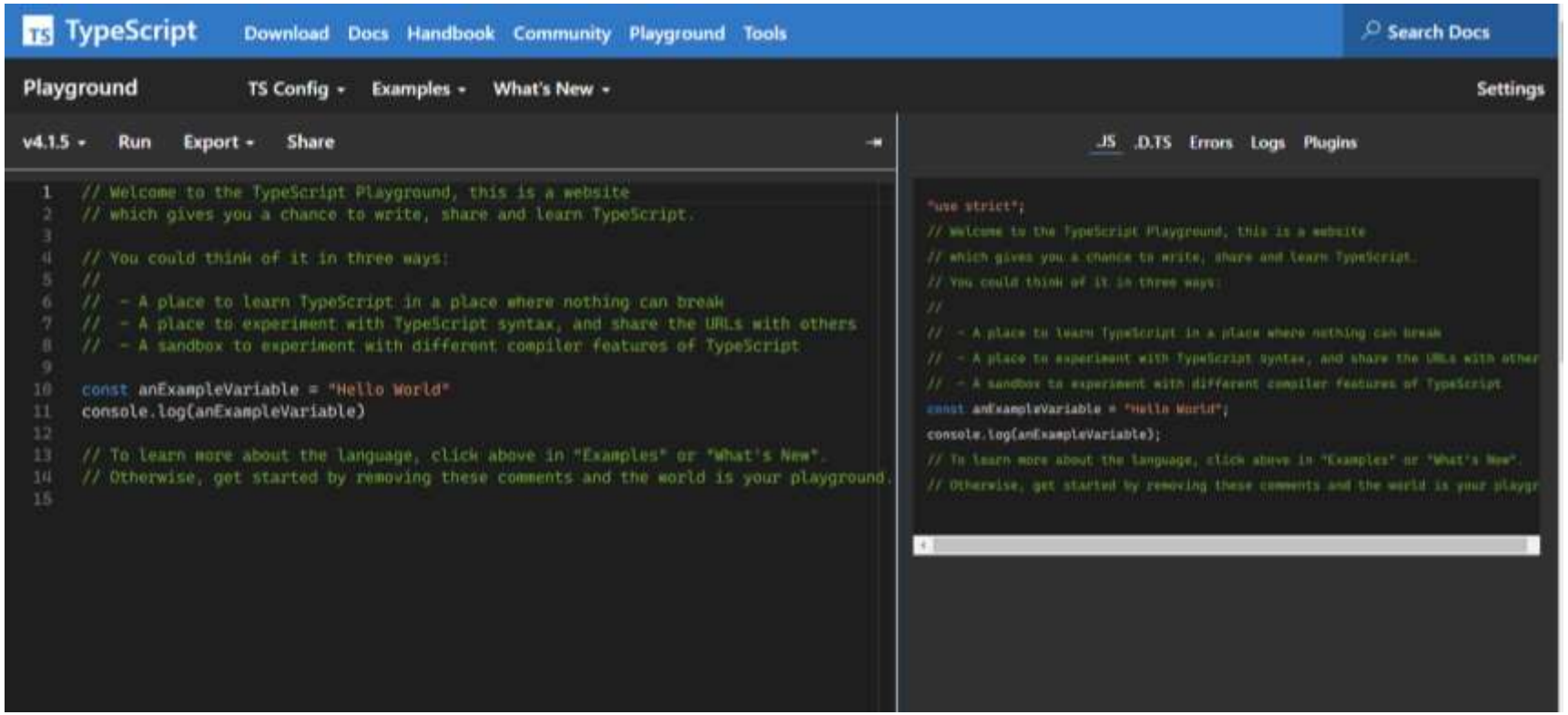
`tsconfig.json` file:

tsconfig.json > ...

```
1  {
2      "compilerOptions": {
3          "target": "ES5"
4      }
5  }
```

Online Compiler

https://www.typescriptlang.org/play



Getting Started

Variables

var

i. Global scope

ii. Variable can be redeclared

```
TS app.ts > ...
for (var i = 0; i <= 3; i++)
{
    var num = i;
}

console.log(num);
```

Inspector | Console

Filter Output

```
3
```

let

i. Limited to block scope

ii. Variable cannot be redeclared

```
TS app.ts > num
for (var i = 0; i <= 3; i++)
{
    let num
}

console.log(num);
```

any

Cannot find name 'num'. ts(2304)

Peek Problem (Alt+F8) No quick fixes available

Variables

const

i. Once declared, its value cannot be changed

```
TS app.ts > ...
1   const num = 5;
2
3           var num: number
4
5   for (var   Cannot redeclare block-scoped variable 'num'. ts(2451)
6   {          Peek Problem (Alt+F8)   No quick fixes available
7       var num = i;
8   }
9
10  console.log(num);
```

static

i. Associated with a class and not with the object

ii. Value can be accessed only when called on a class

```
TS app.ts > ...
1   class class_name {
2       static variable_1 = 1;
3       variable_1 = 2;
4   }
5
6   let object_1 = new class_name();
7
8   console.log(class_name.variable_1);
9   console.log(object_1.variable_1);
```

```
Inspector   Console
Filter Output
1
2
```

Episode 2:
TypeScript Operators & Databases

TypeScript Operators

- Arithmetic operators
- Relational (comparison) operators
- Logical operators
- Bitwise operators
- Assignment operators
- Ternary/Conditional operators
- Concatenation Operator
- Type operator

Arithmetic operator

Addition (+)	Subtraction (-)	Multiplication (*)	Division (/)

`let a = 20;`
`let b = 30;`
`let c = a + b;`
`console.log( c );`
`// Output`
`30`
	`let a = 30;`
	`let b = 20;`
	`let c = a - b;`
	`console.log( c );`
	`// Output`
	`10`
		`let a = 30;`
		`let b = 20;`
		`let c = a * b;`
		`console.log( c );`
		`// Output`
		`600`
			`let a = 100;`
			`let b = 20;`
			`let c = a / b;`
			`console.log( c );`
			`// Output`
			`5`

Modulus (%)	**Increment (++)**	**Decrement (--)**	

`let a = 95;`
`let b = 20;`
`let c = a % b;`
`console.log( c );`
`// Output`
`15`
	`let a = 55;`
	`a++;`
	`console.log( a );`
	`// Output`
	`56`
		`let a = 55;`
		`a--;`
		`console.log( a );`
		`// Output`
		`54`	

Relational Operator

Is equal to (==)	Identical (===)	Not equal to (!=)	Not identical (!==)

`let a = 10;`
`let b = 20;`
`console.log(a==b);`
`//false`
`console.log(a==10);`
`//true`
	`let a = 10;`
	`let b = 20;`
	`console.log(a===b);`
	`//false`
	`console.log(a===10);`
	`//true`
		`let a = 10;`
		`let b = 20;`
		`console.log(a!=b);`
		`//true`
		`console.log(a!=10);`
		`//false`
			`let a = 10;`
			`let b = 20;`
			`console.log(a!==b);`
			`//true`
			`console.log(a!==10);`
			`/false`

Greater than (>)	**Greater than equal to**	**Less than (<)**	**Less than or equal to**

`let a = 30;`
`let b = 20;`
`console.log(a>b);`
`//true`
`console.log(a>30);`
`//false`
	`let a = 20;`
	`let b = 20;`
	`console.log(a>=b);`
	`//true`
	`console.log(a>=30);`
	`//false`
		`let a = 10;`
		`let b = 20;`
		`console.log(a<b);`
		`//true`
		`console.log(a<10);`
		`//false`
			`let a = 10;`
			`let b = 20;`
			`console.log(a<=b);`
			`//true`
			`console.log(a<=10);`
			`//true`

Logical Operator

Logical AND (&&)	Logical OR (||)	Logical NOT (!)

`let a = false;`
`let b = true;`
`console.log(a&&b);  //false`
`console.log(b&&true);  //true`
`console.log(b&&10);  //10 which is also 'true'`
	`let a = false;`
	`let b = true;`
	`console.log(a||b);  //true`
	`console.log(b||true);  //true`
	`console.log(b||10);  //true`
		`let a = 20;`
		`let b = 30;`
		`console.log(!true);  //false`
		`console.log(!false);  //true`
		`console.log(!a);  //false`
		`console.log(!b);  /false`
		`console.log(!null);  //true`

Bitwise Operators

Bitwise AND (&)	Bitwise OR (|)	Bitwise XOR (^)

`let a = 2;`
`let b = 3;`
`let c = a & b;`
`console.log(c);  //`
Output
2
	`let a = 2;`
	`let b = 3;`
	`let c = a | b;`
	`console.log(c);  //`
	Output
	3
		`let a = 2;`
		`let b = 3;`
		`let c = a ^ b;`
		`console.log(c);  //`
		Output
		1

Bitwise NOT (~)	**Bitwise Right Shift (>>)**	**Bitwise Left Shift (<<)**

`let a = 2;`
`let c = ~ a;`
`console.log(c);  //`
Output
-3
	`let a = 2;`
	`let b = 3;`
	`let c = a >> b;`
	`console.log(c);  //`
	Output
	0
		`let a = 2;`
		`let b = 3;`
		`let c = a << b;`
		`console.log(c);  //`
		Output
		16

Assignment Operators

Assign	Add and assign	Subtract and assign

`let a = 10;`
`let b = 5;`
`console.log("a=b:"`
`+a);  //`
Output
10
	`let a = 10;`
	`let b = 5;`
	`let c = a += b;`
	`console.log(c);  //`
	Output
	15
		`let a = 10;`
		`let b = 5;`
		`let c = a -= b;`
		`console.log(c);  //`
		Output
		5

Multiply and assign	**Divide and assign**	**Modulus and assign**

`let a = 10;`
`let b = 5;`
`let c = a *= b;`
`console.log(c);  //`
Output
50
	`let a = 10;`
	`let b = 5;`
	`let c = a /= b;`
	`console.log(c);  //`
	Output
	2
		`let a = 16;`
		`let b = 5;`
		`let c = a %= b;`
		`console.log(c);  //`
		Output
		1

Ternary/Conditional Operator

expression ? expression-1 : expression-2;

- **expression:** It refers to the conditional expression.
- **expression-1:** If the condition is true, expression-1 will be returned.
- **expression-2:** If the condition is false, expression-2 will be returned.

Example

```
let num = 16;
let result = (num > 0) ? "True":"False"
console.log(result);
```

Concatenation Operator

Append the two string

Cannot add space

Example

```
let message = "Welcome to " + "Microsoft";
console.log("Result of String Operator: " +message);
```

Type Operators

In - used to check for the existence of a property on an object.

```
let Bike = {make: 'Honda', model: 'CLIQ',
year: 2018};
console.log('make' in Bike);   //
```

Output:

```
true
```

Delete - It is used to delete the properties from the objects.

```
let Bike = { Company1: 'Honda',
             Company2: 'Hero'};
delete Bike.Company1;
console.log(Bike);   //
```

Output:

```
{ Company2: 'Hero'}
```

Typeof - It returns the data type of the operand.

```
let message = "Welcome to " + "Event";
console.log(typeof message);  //
```

Output:

```
String
```

Instanceof - It is used to check if the object is of a specified type or not.

```
let arr = [1, 2, 3];
console.log( arr instanceof Array ); //
true
console.log( arr instanceof String ); //
false
```

TypeScript Type Annotation

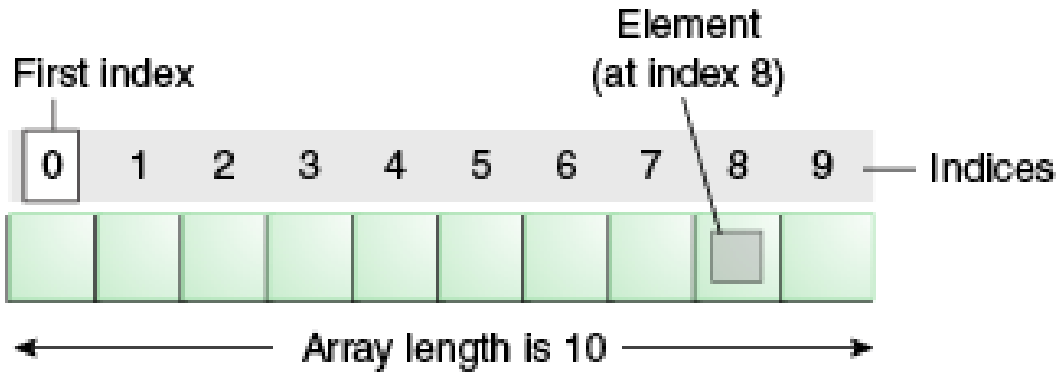
Type Annotations are annotations which can be placed anywhere when we use a type. It helps the compiler in checking the types of variable and avoid errors when dealing with the data types.
We can specify the type by using a **colon(: Type)** after a variable name, parameter, or property.

Syntax:

```
var variableName: TypeAnnotation = value;
```

var age: number = 44; // number variable
var name: string = "Rahul"; // string variable
var isUpdated: boolean = true; // Boolean variable

TypeScript Arrays




```
let array_name:datatype[] = [val1,val2,valn..]
```

There are two types of an array:

- Single-Dimensional Array - let array_name:datatype[]=[a1, a2, a3];
- Multi-Dimensional Array - let arr_name:datatype[][] = [[a1,a2,a3], [b1,b2,b3]];

array methods

- concat()
- Push()
- indexOf()

Pop()

reverse()

TypeScript Tuples

store a collection of values of varied types.

Syntax:

```
var tuple_name = [value1,value2,value3,…value n]
```

Example:

var mytuple = [10,"Hello"];

Tuple Operations

Length

Push

Pop

TypeScript Unions

TypeScript gives programs the ability to combine one or two types. Union types are a powerful way to express a value that can be one of the several types. Two or more data types are combined using the pipe symbol (|) to denote a Union Type.

Syntax: Union literal

```
Type1|Type2|Type3
```

Example

```
var val:string|number
val = 12
console.log("numeric value of val "+val)
val = "This is a string"
console.log("string value of val "+val)
```

Output:

```
numeric value of val  12
string value of val this is a string
```

QnA

Topics discussed:

- Operators:
  - Arithmetic
  - Relational
  - Logical
  - Bitwise
  - Assignment
  - Ternary/Conditional
  - Concatenation
  - Type
- Type Annotation
- Arrays
- Tuples

Microsoft Learn
STUDENT AMBASSADOR

LinkedIn/shreyakhandelwal99

LinkedIn/niralisahoo