TypeScript

...

Objectives

- Review Resume/Profile Lab
- Set up Cloud9 for TypeScript Development
- What is TypeScript
- Variables - let & const
- Data Types
- Arithmetic Operators
- Comparison Operators
- Logical Operators
- TypeOf

Set Up Cloud9 for TypeScript Development

Step 1 - Create a new BLANK application in cloud9

Step 2 - npm init : we will hit enter all the way until we have to type yes

Step 3 - touch mycode.ts

Step 4 - Inside the package.json file we will change main to the name of or .ts file

Step 5 - Type some typescript code > save > and do tsc mycode.ts to run it

Step 6 - Create a tsconfig file by doing tsc --init in terminal

Step 7 - watch mode tsc mycode.ts --w

What is TypeScript?

TypeScript is a superset of JavaScript which primarily provides optional static typing Classes, and Interfaces. One of the big benefits is to enable IDE's to provide a richer environment for spotting common errors as you type the code. TypeScript compiles to JavaScript.

What is a compiler?
A compiler is a software program that transforms source code written by a developer in a high level programming language into low level object code(binary code) in machine language, which can be understood by the processor

Variables: let & Const

What is a variable?

A variable is a value that can change, depending on conditions or on information passed to the program. It's like a variable like a box to hold something that can change!

There are two ways of declaring variables in TypeScript.

let - used to hold variables that can change.

const - used to hold variables that do not change

Example of using let to declare a variable

```
let myName: string = "Mikaila";

console.log(myName);
```

How we print something to the console?

Example of using const to declare a variable

```
const myAge: number = 19;

console.log(myAge);
```

If we try to assign another age to this variable the compiler will give us an error

We have several data types in TypeScript such as ...

String: used to define text or letters

Number: used to defined variables that must be numbers or decimals

Boolean: used to hold data that will be True or False

Any: used to hold data types that can change to any type such as string or number

Array: used to hold a list of data with indexes starting at 0

Tuples: are like arrays but with mixed types and the order is important

Enums: used to make numbers more expressive

How to declare variables using types

```
let myString: string = “I am a string”;
let myNumber: number = 18;
let myArray: number[ ] = [1,2,4,5,7];
let myArray: string[ ] = [‘1’,‘2’,’3’,’4’];
let myFact: boolean = true;

Let canChange: any = 10;
canChange = ‘changed to a string’;
```

Arithmetic Operators

Addition Plus sign (+)
Subtraction Minus sign (-)
Multiplication Multiplication sign (*)
Division Division sign (/)
Modulus Remainder(%)
PostFix Increment and Decrement (x++) and x(--)
Prefix Increment and Decrement (++x) & (--x)

Comparison Operators - Assuming X = 5

x > 10 = false

x < 10 = true

x >= 5 = true

x <= 100 = true

x == "5" = true // Type coercion

x === "5" = false // No type coercion

x != b = true

x !== "5" = true

Logical Operators

&& AND both sides need to be true

|| OR One side needs to be true

! NOT if something was true it makes it false (vice versa)

Using Operators for calculations

P. E. D. M. A. S rules applies -

The grouping operator ( ) controls the precedence of evaluation of expressions

```
let x : number = 500;
let y : number = 400;
let z : number = 10;
let a : number = 2;
let Answer =  x - y + z  * a ;   // How to fix  - Add brackets before multiplication?
Console.log(Answer);
```

Ternary Condition Operator

It takes three operands such as condition ? val1 : val2

```
let age: number = 20;
let status : number  = ( age >= 21)  ?  'can drink'  :  'cannot drink';

console.log(status);
```

Conditional statements { if statements }

We have two main conditional statements in TypeScript.
The if else statement and a switch statement

IF STATEMENT

```
if (true) {
console.log(true);
} else{
console.log(false);
}
```

Using the OR || Comparison Operator

```
let n: string = 'netflix';
let  h: string = 'hulu';
let userInput: string = n;


if( (userInput == n) || (userInput == h) ){
        console.log('I will be streaming movies on netflix OR hulu this weekend' );
}else{
        console.log('I will be studying this weekend');
}
```

Using the AND && Comparison Operator

```
let n: string = 'netflix';
let  h: string = 'hulu';
let userInput1: string = 'netflix';
let userInput2: string = h;

if((userInput1 == n) && (userInput2 == h)){
        console.log(' I will be streaming movies on netflix and hulu this weekend ');
}else{
        console.log('I will be studying this weekend');

}
```

Else If

We use the Else If to test a new condition, if the first condition is false or not met

```
let condition1: number = 1;     let condition2: number = 2;
if(condition1 > condition2){
console.log('One is greater than Two');
}else if(condition1 == condition2){         // change to Not equal
console.log('One is equal to Two');
}else{
console.log('None of the above is true');
}
```

Switch statement

```
switch(value){
        case 0:
        console.log(print something.....);
        break;

        case 1:
        console.log(print something else.....);
        break;

        default:
        console.log(print something else.....);
        break;
}
```

TypeOf

Since static typing is optional in TypeScript you may come across data and need to know the type of data it is. That's when we use TypeOf as a mechanism to check

```
let myName = "Mikaila";

console.log(typeof(myName));
```

will print - String

Practice with the other data types such as boolean, array, number etc...

While Loops

Used to loop through a block of code as long as a specified condition is true.

```
while (condition) {
Code block to be executed
}
```

```
let j : number = 0;    let counter : number = 10;
        while( j < counter ) {
        j++;
        console.log(j);
}
```

DO/While Loop

A variant of the while loop. It will execute the code at least once before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
do{

 Code block to be executed

}
while ( condition);
```

See example on the page next page.

Example of Do while loop

```
let i : number = 0;

do {

    console.log( " The number is " + i );

    i++;

} while(i < 5);
```

For Loops

For loops are used to loop through a block of code a number of times.

```
for(statement 1; statement 2; statement 3){
        Code block to be executed
}
```

For example:

```
for( let x = 0 ;  x < 5 ;  x++ ) {
console.log( ' The value of x is ===> ' + x ) ;
}
```

Array Methods

```
let fruits : string[] = [ "Banana", "Orange", "Apple", "Mango"];
```

toString() converts an array to a string of comma separated array values

```
console.log(fruits.toString());
```

join() behaves like toString but you can specify a separator

```
console.log( fruits.join( ' * ' ) );
```

pop() removes the last element from an array

```
console.log( fruits.pop () );
```

push() adds a new element to the end of the array and also returns the array length

```
console.log( fruits.push (‘ kiwi ‘) );
```

Array Methods continued....

shift() removes the first element from an array and returns the element

```
console.log( fruits.shift () );
```

unshift() adds a new element at the beginning of the array

```
console.log( fruits.unshift ( ' lemon ' ) );
```

.length Used to get the length of the array

```
console.log(' The length of the array is  ' +  fruits.length);
```

For further array methods see w3schools for methods such as slice, splice, sort, reverse

Math Methods

Math.round (x) returns the value of x rounded to the nearest integer //4 . 7
Math.PI returns 3.141592653589793
Math.pow(x, y); returns the value of x to the power of y //8 , 2
Math.sqrt(x) returns the square root of x // 64
Math.abs(-x) returns the absolute positive value // -4 . 7
Math.ceil(x) returns the value of x rounded up to the nearest integer //4 . 4
Math.floor(x) returns the value of x rounded down to its nearest integer //4.7
Math.min(1,2,3)and Math.max(1,2,3) used to find the lowest and highest values
Math.random returns a random number between 0 and 1
For further array methods see w3schools

Functions

A function is a block of code designed to perform a particular task. Functions are executed when called or invoked. For Example: This function multiplies the values of p1 and p2 and returns the answer.

```
function myFunction(p1: number, p2: number) :number{
return   p1 * p2;
}
```

To call or invoke the function we simply do - myFunction(2, 20);

In cloud 9 we log it to the console - console.log(' Ans is ==> ' + myFunction(2,20));

p1 and p2 are known as parameters while 2 and 20 are known as arguments

The return type :number -- means that this function returns a number

Return keyword and function with no parameters

return (expression) ; - The return statement ends function executions and specifies a value to be returned to the caller of the function.
It is affected by the semicolon therefore always end your return statements with a semicolon **;**

```
function myStringFunction(): string{
    return 'Hello world' ;
}

console.log(myStringFunction());
```

Notice that this function has no parameters and takes no arguments

Function with two parameters and a return type of Number

```
function modulous(num1: number, num2: number): number{
    return num1 % num2;
}

console.log(' The answer is  ' , modulous(91 , 2));
```

Notice that this function takes two parameters therefore it needs two arguments

Function with one argument Type and returns a String

```
 function myFavSport(nameOfSport: string): string {
        return ' My favorite sport is  ' + nameOfSport ;
}

console.log(myFavSport('Cheerleading'));
```

Notice that this function takes one argument of type string and returns a string

Function with arguments and no return type & return types

No return type and takes two arguments

```
function add(a: number, b: number):{
 console.log(a + b);
}
add(1, 2);
```

Takes one argument and has a return type

```
function returnMyName(name: string): string{
return 'My name is ' + name;
}
```

Mixing outside variables with parameters

```
let z: number = 100;

function addToZ(x: number, y: number):number {
 return x + y + z;
}

console.log(' The result of adding to z is ' + addToZ(3, 17));
```

Anonymous Function & Function as variables values

An anonymous function has no name. Often used as callback functions. For example You can assign a function with no name to a variable.

```
let xFunc = function (p1 :number, p2 :number): number {
return p1 * p2;
}

console.log(xFunc(20,6));
```

Number of parameters matter

```
function fullName(firstName: string, lastName: string) {
 return firstName + " " + lastName;
}

let result1 = fullName("Bob");                  // error, too few parameters - undefined

let result2 = fullName("Bob", "Adams", "Sr.");  // too many parameters

let result3 = fullName("Bob", "Adams");         // ah, just right


console.log(result1);
console.log(result2);
console.log(result3);
```

Objects

An Object is a collection of named values. The named values are called properties.

Property is the key such as firstName
Value is the value of the key such as “ Mike”

To declare an object using an object literal we simply do something like this -

```
let person: { name: string;  age: number;  sex:string;  likesHipHop: boolean; };
person = {
name: 'Becky',
age: 22,
sex: 'Female',
likesHipHop: true
};
```

Printing out the value of our Object

```
console.log("My name is " + person.name + " " + " I am " + person.age + " years old " + person.sex + " and its " + person.likesHipHop + " That I like HipHop");
```

Try using the person Object to print out something unique. Make it your own!

Object Methods

Methods are actions that can be performed on objects. An object method is a function definition. Let's add a method to our person object and print it out as seen below!

```
person = {
name: 'Becky',
age: 22,
sex: 'Female',
likesHipHop: true
makeNoise: function(){ return "Arhhhhhhhh"; }
};console.log("My name is " + person.name + " " + " I am " + person.age + " years old " + person.sex + " and its " + person.likesHipHop + " That I like HipHop so" + "everybody make some noise " + person.makeNoise());
```

For in - Used to loop through our object

```
for(let x in person){
console.log(x);  //prints the keys
}

for(let y in person){
console.log(person[y]); //print the values
}
```

For in continued....

To print both the keys and values of an Object we combine what we learned so far..

```
for(let z in person){
 console.log( z + " <==> " + person[z]); //print the values
}
```

To just get the value we leverage the key. For example

```
console.log(person["name"]); //get name
console.log(person.name); // get name
```

To set the value of the Object you use the key and assign

```
person.name = "yonce";
console.log(person.name);

console.log(person);
```