2.10.22

Typescript Programming Language (.ts)

The standard on which typescript was build to ensure the same code execution by typescript engine.

Variable and Datatypes.

Variable - It is a memory location given to a value in a program. It acts like a container of a value.

Declaring a variable-

variablename : datatype = value;
Identifier

Rules for naming a variable.

Contain letter, digit, underscore and dollar sign
Must begin with letter, underscore and dollar sign.
Reserved word cannot be used.
Variable name are case-sensitive and can be overwritten.

var, let and const -

- var is globally declared while let & const are blocked scope
- var can be updated and re-declared.
- let can be updated but non re-declarable.
- const can't neither be updated or re-declarable.
- var are declared with undefined scope,
- const need to be declare and initialized at same time.

Primitive datatypes and Object

Null - does not hold a value let a: null = null;
Number - Integral Number let a: number = 45;
Big Int - Large Integral number let m: bigint = BigInt(70);
String - Collection of character "" let s: string = "shroud";
Boolean - True or False - let b: boolean = true;
Symbol let b: symbol = Symbol("Mom");
Undefined not initialized var let m: undefined = undefined;

typeof - Identifier used to get the type of datatype of a variable.

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

Objects - They are non primitive datatype and stores key, value pair.

```
const obj = {
    name: "Shroud",
    mail : "shroud@dsr.it"
}
```

```
console.log(obj[name]); - Print "Shroud"
console.log(obj[hello]); - Print Undefined
```

Changing values of a object.

```
obj[name] = "Shroud"; - Alternate
obj.name = "Shroud"; // changing existing key.
obj['mailid'] = "shroudx@dsr.it"; // Adding.
```

Operators in Typescript -

1) Arithmetic Operator -

+ , -	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
**	Exponential	a ** b
/	Division	a / b
%	Mod	a % b
++	increment	(PRE) ++a , a++ (POST)
--	decrement	(PRE) --a , a-- (POST)

2) Assignment Operator

=	Equals Assign	a = 90;
+=	Add and Assign	a += 30;
-=	Sub and Assign	a -= 5;
*=	Multi and Assign	a *= 10;
/=	Div and Assign	a /= 5;
**=	Expo and Assign	a **= 5;

Comparision Operator

!=	not equals
==	equals to
===	equals to with equal data type
!==	not equal value or not equal type
>	greater
<	less
>=	greater than equal to
<=	less than equal to
? :	Ternary operator

Logical Operator [operate on boolean]

&&	logical and (return true if all condition are true)
||	logical or (return true if atleast one in true
!	logical not (return the opposite)

Comments - The unexecuted lines of code that is used to make our source code more informative.

Types of comment -

* Single line - // starting and ends with the line
* Multi-line - /* starting and ends with */

Conditional Expression -

Sometimes inorder to execute a program on a condition is called conditional Expression

Types of Conditional Expression -

if - if (condition) {

}

else - else {

}

if-else - if (condition) {

}

else if (condition) {

}

Only one condition from if else if else is executed

Switch case Statement
An alternative to if else that executes code on bases of case.

```
switch (variable){
  case 90:
      // 90's code.
  break;
  case 80:
      // 80's code.
  break;
  case 70:
      // 70's code.
  break;
}
```

break should be there after every case to prevent fall through means break prevent execution of multiple case statement. Fall through can be sometimes be useful.

Ternary Operator

```
condition ? expression-1 : expression-2;
```

Loops in Typescript -
Loops are used to repeat certain statement which are needed to repeat many times in a program
For eg -
Printing 1 to 1000

Types of loop

for loop - Loop a code for certain no of times
for in loop - Loop through key of objects
for of loop - Loop through value of object
while loop - loops a block for a specific condition
do-while loop - while loop variant running atleast one

For-Loop -

```
for (control variable; condition; updation) {
    //code here.
}
```

For in loop - [It also works with array]

```
for (control variable in object) {
    //code.
}
```

For-of loop — data like [Array, String]

```
for (control variable of iterable) {
    //code.
}
```

While-Loop - [entry controlled - runs only if true till true)

```
while (condition) {
    //code.
}
```

Do-While-Loop -

```
do {
    //code.
}
while (condition);
```

Functions in Typescript-

A block of code that need to be repeated many times Function is used to bind that code into one.

```
function nameoffunction(){
    //code.
}

function namefunction(paraone, paratwo, paran){
    //code.
}
```

Parameter is required to run the function if needed Parameter variable name can be a existing variable name but they will remain different.

function naming convention are same as of variables return - After run running a function might return a value. For ex-

```
function main(x,y){
    console.log(x+y);
    return x+y;
}
```

Types are to be included if type should be fixed.

Invoking a function-

```
functionname(); //non-parameterized Function
functionname(val1, val2); //parameterized function
```

Arrow Function-

```
const funcname = () => {
    //code.
}
```