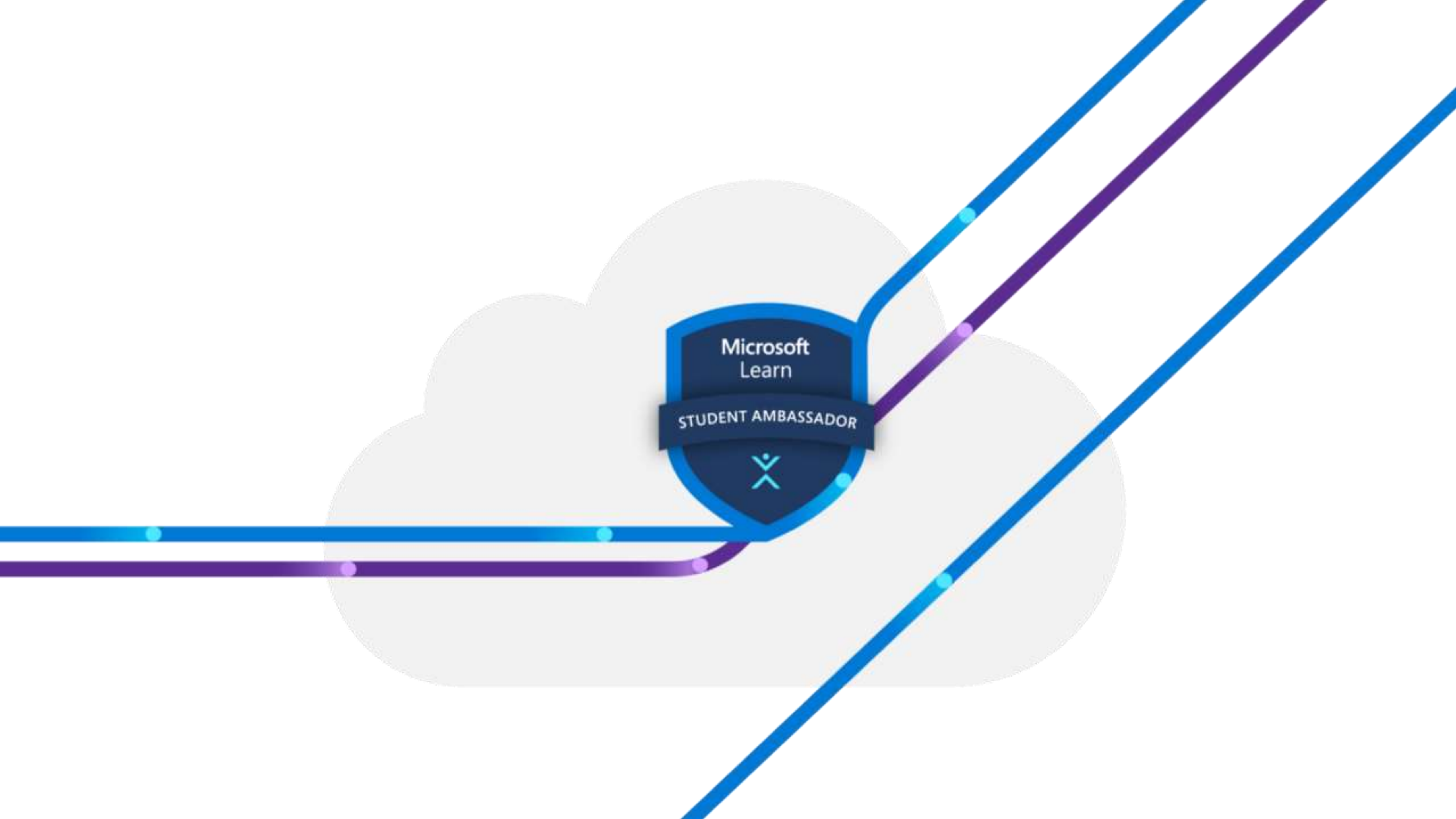

Microsoft
Learn
STUDENT AMBASSADOR

Microsoft Learn
Student Ambassadors

Typing With TypeScript

Episode 3: Decision Making, Loops & Functions

Shreya Khandelwal
Nirali Sahoo

About the Speakers:

Shreya Khandelwal

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

Nirali Sahoo

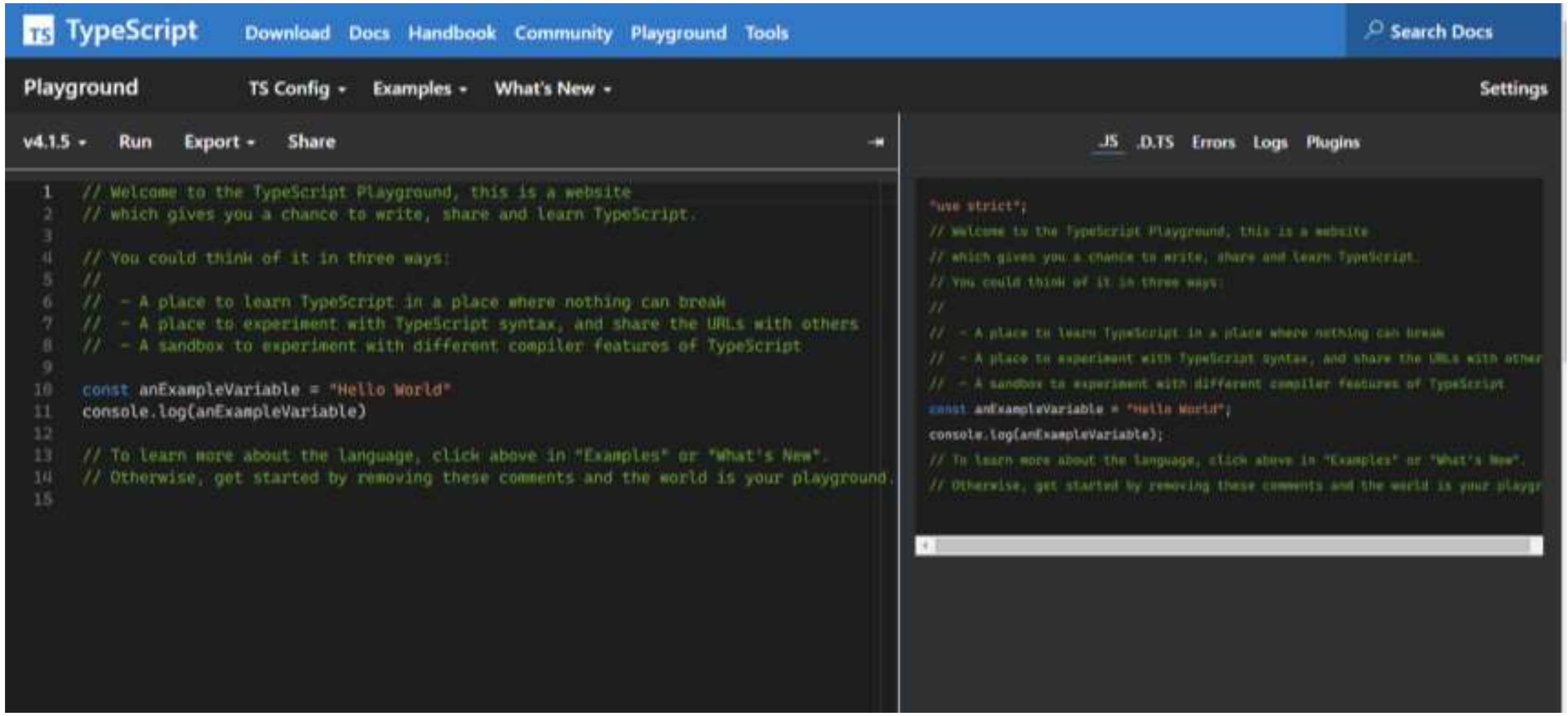
- Born and brought up in Odisha, India
- CSE Sophomore in IIIT Bhubaneswar
- Beta MLSA

Episode 1 Summary

- **TypeScript**:
  - open-source
  - OOPL
  - developed and maintained by **Microsoft**
  - Superset of the JavaScript language
  - ES6 version of JavaScript.
  - 3 components: Language, TSC, Language services
- **Installation:**
  - Requires npm (node package manager)
  - `npm install -g typescript`
  - Compiling : `tsc <file_path>`
  - Watch mode: `tsc -w <file_path>`
  - For extra compiler options: `tsconfig.json file`

Online Compiler

https://www.typescriptlang.org/play



```
// Welcome to the TypeScript Playground, this is a website
// which gives you a chance to write, share and learn TypeScript.

// You could think of it in three ways:
//
//  - A place to learn TypeScript in a place where nothing can break
//  - A place to experiment with TypeScript syntax, and share the URLs with others
//  - A sandbox to experiment with different compiler features of TypeScript

const anExampleVariable = "Hello World"
console.log(anExampleVariable)

// To learn more about the language, click above in "Examples" or "What's New".
// Otherwise, get started by removing these comments and the world is your playground.
```

```
"use strict";
// Welcome to the TypeScript Playground, this is a website
// which gives you a chance to write, share and learn TypeScript.
// You could think of it in three ways:
//
//  - A place to learn TypeScript in a place where nothing can break
//  - A place to experiment with TypeScript syntax, and share the URLs with other
//  - A sandbox to experiment with different compiler features of TypeScript
const anExampleVariable = "Hello World";
console.log(anExampleVariable);
// To learn more about the language, click above in "Examples" or "What's New".
// Otherwise, get started by removing these comments and the world is your playgr
```

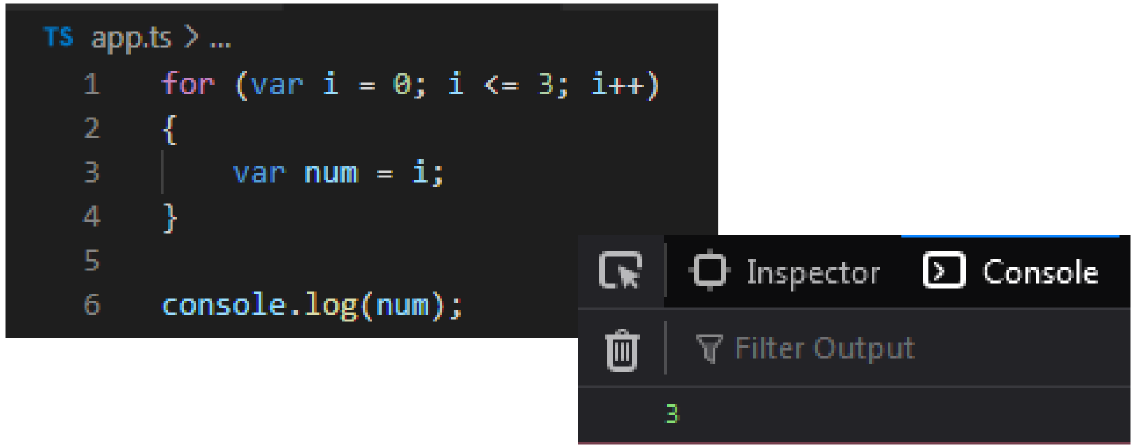
Episode 2 Summary

Variables

`var`

i. Global scope

ii. Variable can be redeclared



TS app.ts > ...

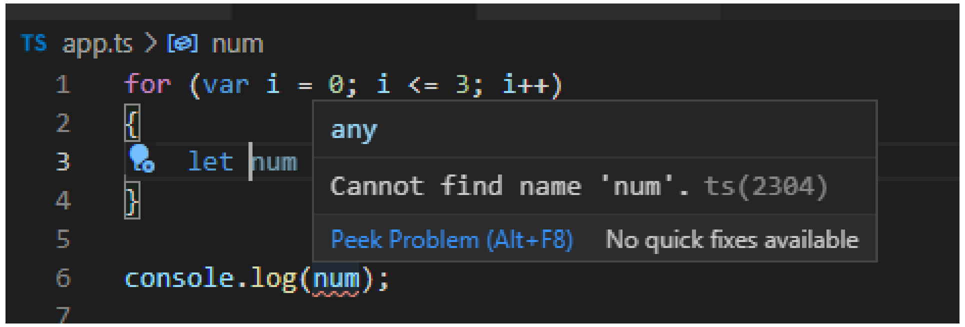
```
for (var i = 0; i <= 3; i++)
{
    var num = i;
}

console.log(num);
```

```
3
```

`let`

i. Limited to block scope

ii. Variable cannot be redeclared



TS app.ts > num

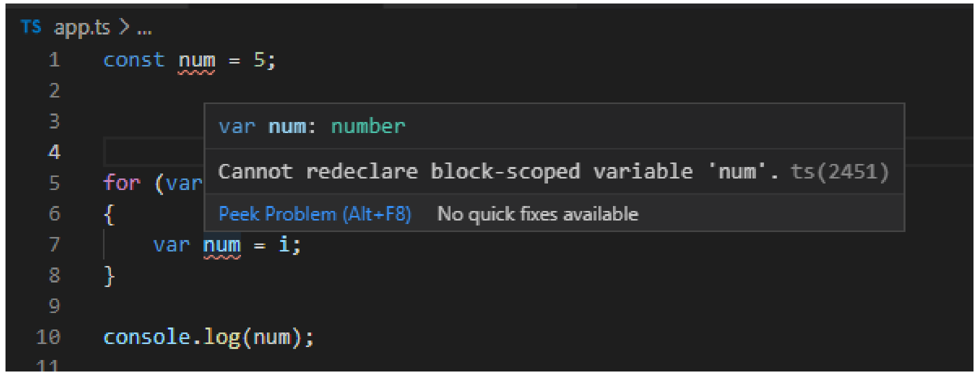
```
for (var i = 0; i <= 3; i++)
{
    let num
}

console.log(num);
```

```
any
Cannot find name 'num'. ts(2304)
```

Variables

const

i. Once declared, its value cannot be changed



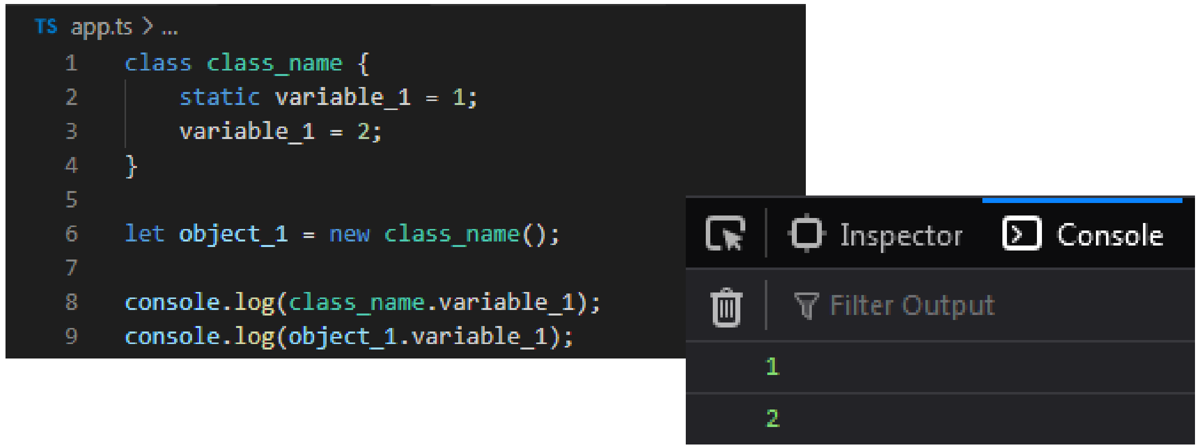
```
TS app.ts > ...
const num = 5;



for (var
{
    var num = i;
}

console.log(num);
```

```
var num: number
Cannot redeclare block-scoped variable 'num'. ts(2451)
Peek Problem (Alt+F8)   No quick fixes available
```

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 > ...
class class_name {
    static variable_1 = 1;
    variable_1 = 2;
}

let object_1 = new class_name();

console.log(class_name.variable_1);
console.log(object_1.variable_1);
```

```
1
2
```

TypeScript Operators

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

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

array methods

- concat() Pop()
- Push()
- indexOf() reverse()

TypeScript Unions

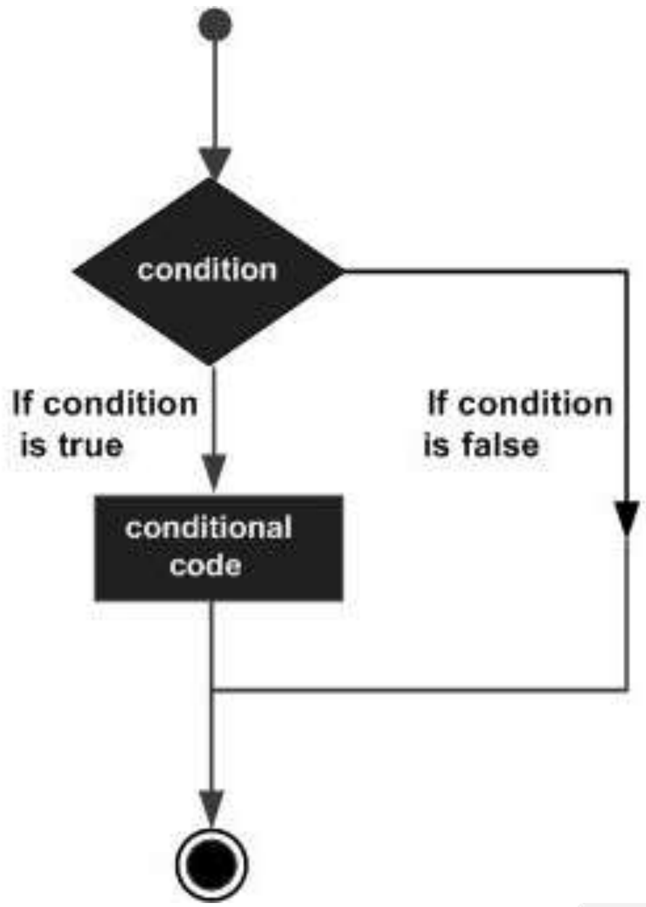
Two or more data types are combined using the pipe symbol (|) to denote a Union Type.

```
Type1|Type2|Type3
```

Episode 3 : Decision Making, Loops & Functions

Decision Making

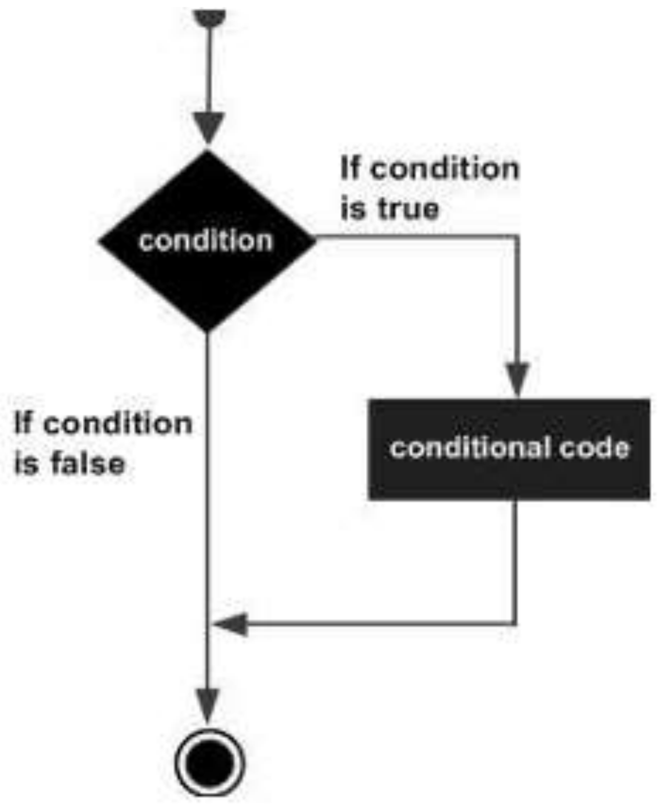
- if statement
- if...else statement
- else…if and nested if statements
- switch statement



If Statement

Syntax:

```
if(boolean_expression) {
   // statement(s) will execute if the
boolean expression is true
}
```



Example

```
var  num:number = 5
if (num > 0) {
   console.log("number is positive")
}
```

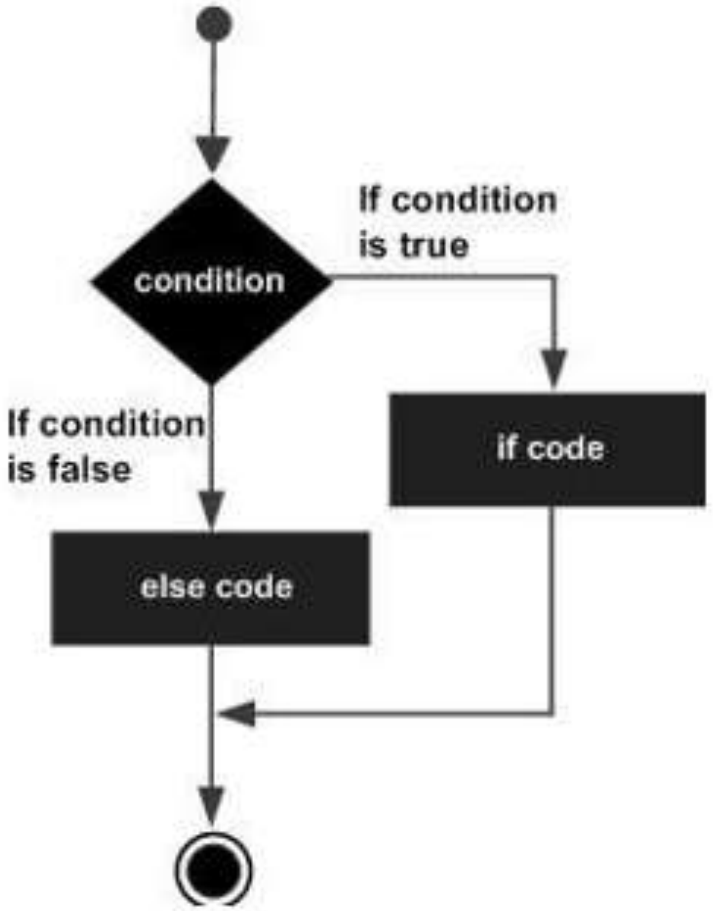
Output:

```
number is positive
```

if…else Statement

Syntax:

```
if(boolean_expression) {
   // statement(s) will execute if the boolean expression is true
} else {
   // statement(s) will execute if the boolean expression is
false
}
```



Example

```
var num:number = 12;
if (num % 2==0) {
   console.log("Even");
} else {
   console.log("Odd");
}
```

Output:

```
Even
```

Nested if statement

Syntax:

```
if (boolean_expression1) {
   //statements if the expression1 evaluates to true
} else if (boolean_expression2) {
   //statements if the expression2 evaluates to true
} else {
   //statements if both expression1 and expression2 result to false
}
```

Example

```
var num:number = 2
if(num > 0) {
   console.log(num+" is positive")
} else if(num < 0) {
   console.log(num+" is negative")
} else {
   console.log(num+" is neither positive nor negative")
```

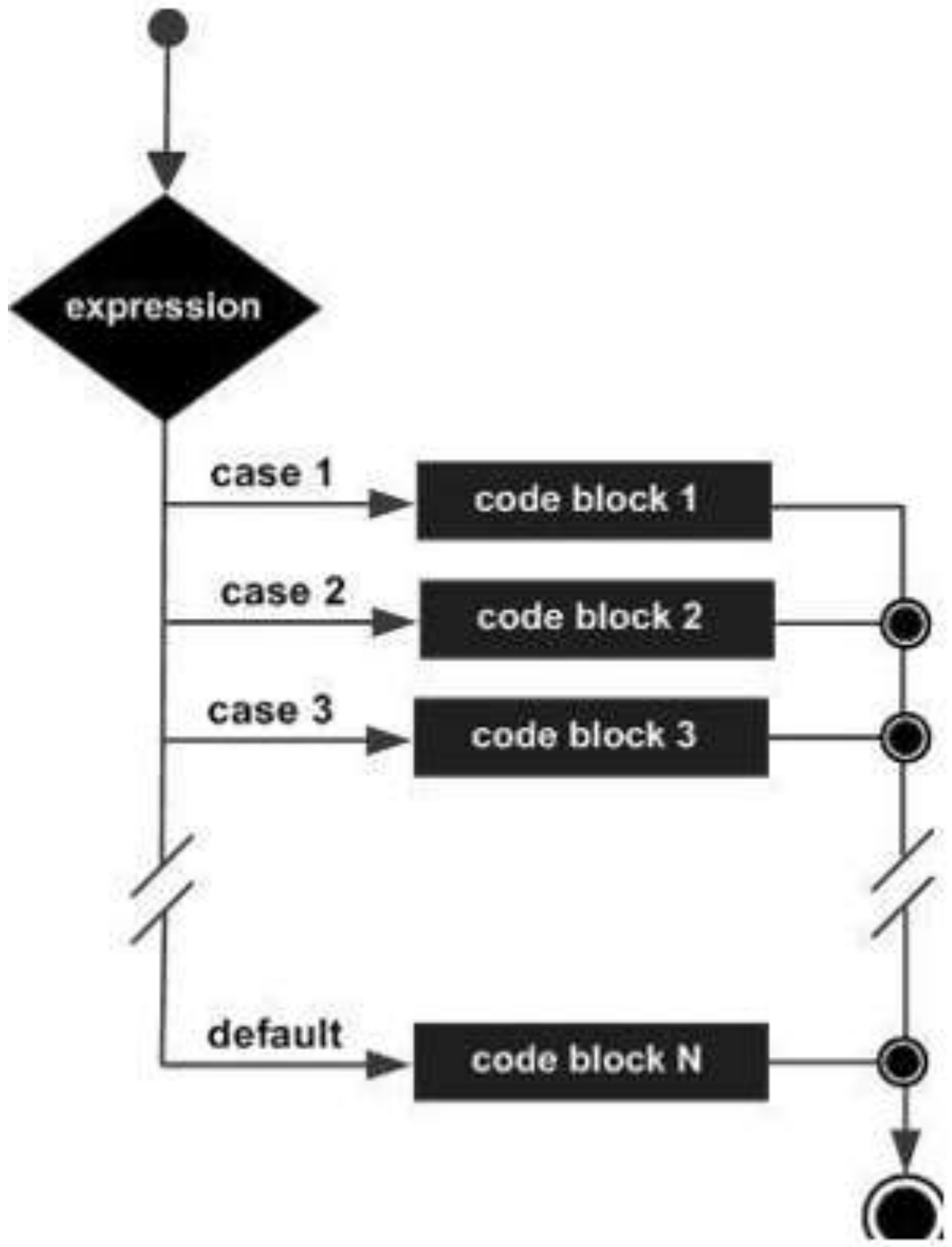
Output:

```
2 is positive
```

Switch...case Statement

Syntax:

```
switch(variable_expression) {
   case constant_expr1: {
      //statements;
      break;
   }
   case constant_expr2: {
      //statements;
      break;
   }
   default: {
      //statements;
      break;
   }
}
```



Example

```
var grade:string = "A";
switch(grade) {
   case "A": {
      console.log("Excellent");
      break;
   }
   case "B": {
      console.log("Good");
      break;
   }
   case "C": {
      console.log("Fair");
      break;
   }
   case "D": {
      console.log("Poor");
      break;
   }
   default: {
      console.log("Invalid choice");
      break;
   }
}
```

Output

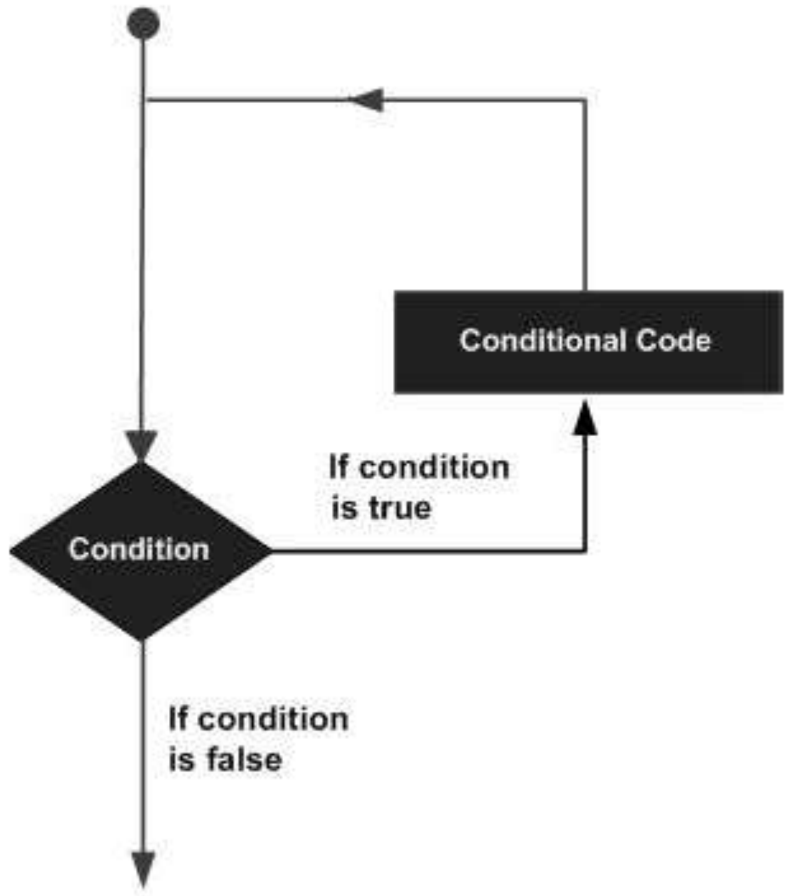
```
Excellent
```

Rules that apply to a switch statement −

- There can be any number of case statements within a switch.
- The case statements can include only constants. It cannot be a variable or an expression.
- The data type of the variable_expression and the constant expression must match.
- Unless you put a break after each block of code, execution flows into the next block.
- The case expression must be unique.
- The default block is optional.

Loops

- Definite loop:
  - for loop
- Indefinite loop:
  - while loop
  - do...while loop



Definite Loop

for loop

Syntax:

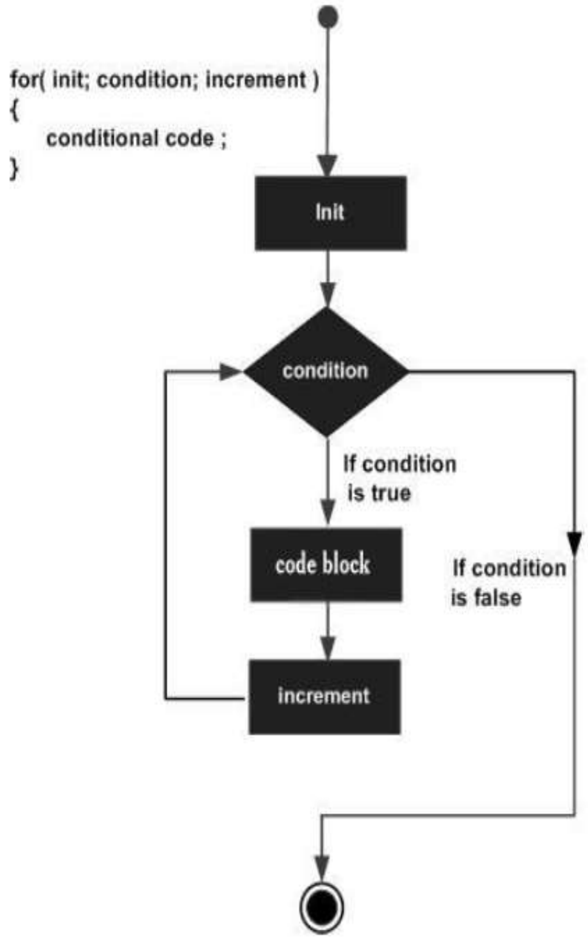
```
for (initial_count_value; termination-condition; step) {
   //statements
}
```

Example

```
for(var:number i = 0, i <= 5, i++) {
    var sum = sum + i;
}
console.log(sum);
```

Output:

```
15
```



for ...in loop

- For iterating through a set of values like arrays and tuples.

Syntax:

```
for (var val in list_of_values) {
   //statements
}
```

Example

```
let arr = [10, 20, 30, 40];

for (var index in arr) {
  console.log(index); // prints indexes: 0, 1, 2, 3

  console.log(arr[index]); // prints elements: 10, 20, 30, 40
}
```

Indefinite Loop

while loop

Syntax:

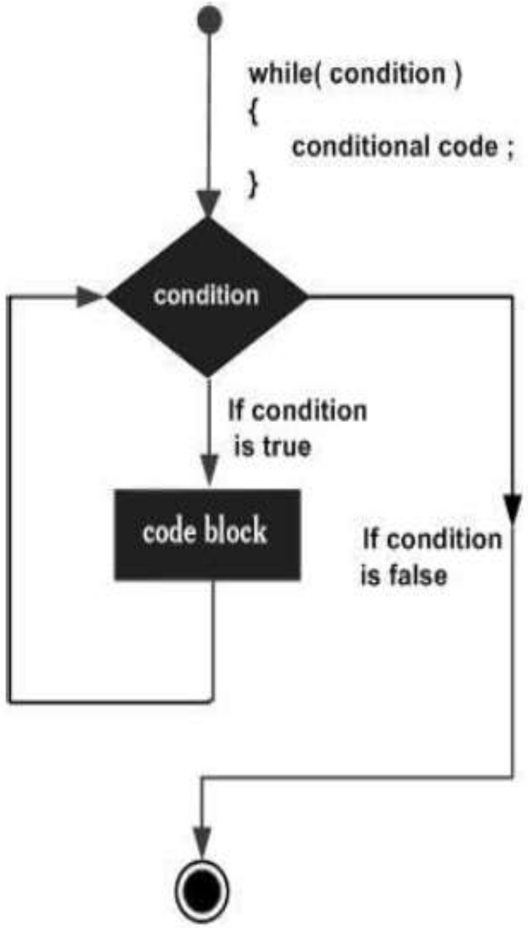
```
while(condition) {
   // statements if the condition is true
}
```

Example

```
var  i:number = 0
while (i <= 5) {
   console.log(i);
    i = i + 1;
}
```

Output:

```
0
1
2
3
4
5
```



do...while loop

Syntax:

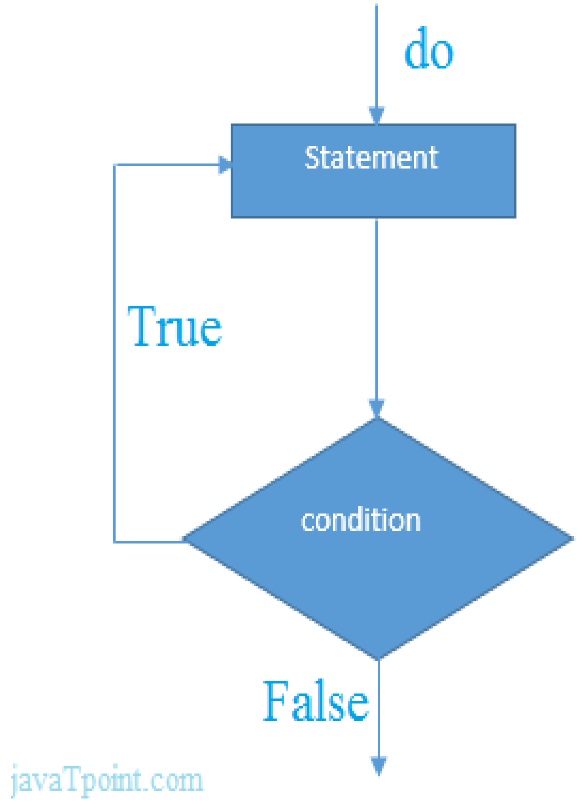
```
do{
    //code to be executed
}while (condition);
```

Example

```
let n = 10;
do {
    console.log(n);
    n++;
 } while(n<=15);
```

Output:

```
10
11
12
13
14
15
```



Break statement

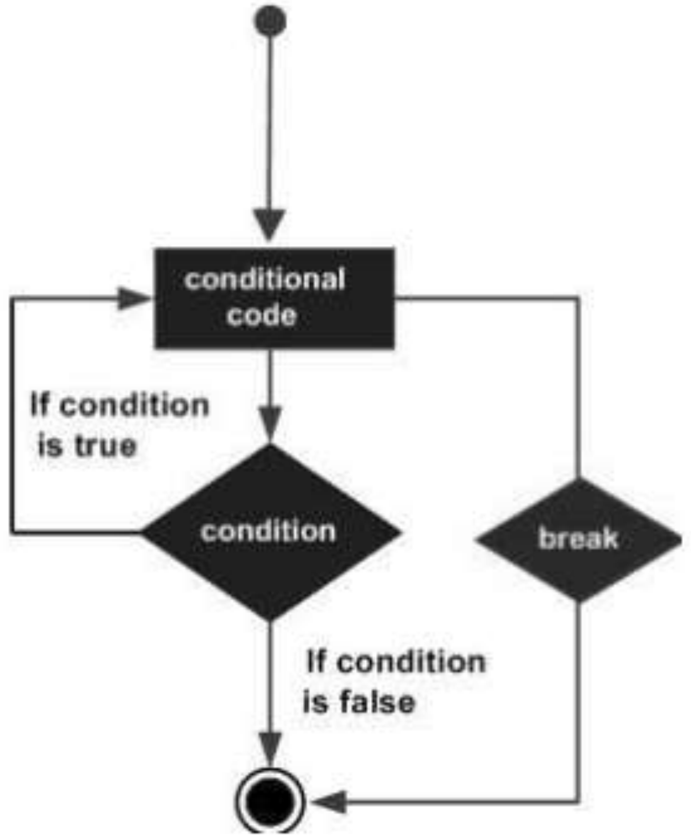
The break statement is used to take the control out of a construct. Using break in a loop causes the program to exit the loop.

Syntax:-

```
var i:number = 1
while(i<=10) {
   if (i % 5 == 0) {
      console.log ("The first multiple of 5  between 1 and 10 is : "+i)
      break     //exit the loop if the first multiple is found
   }
   i++
}
```

Output:-

```
5
```



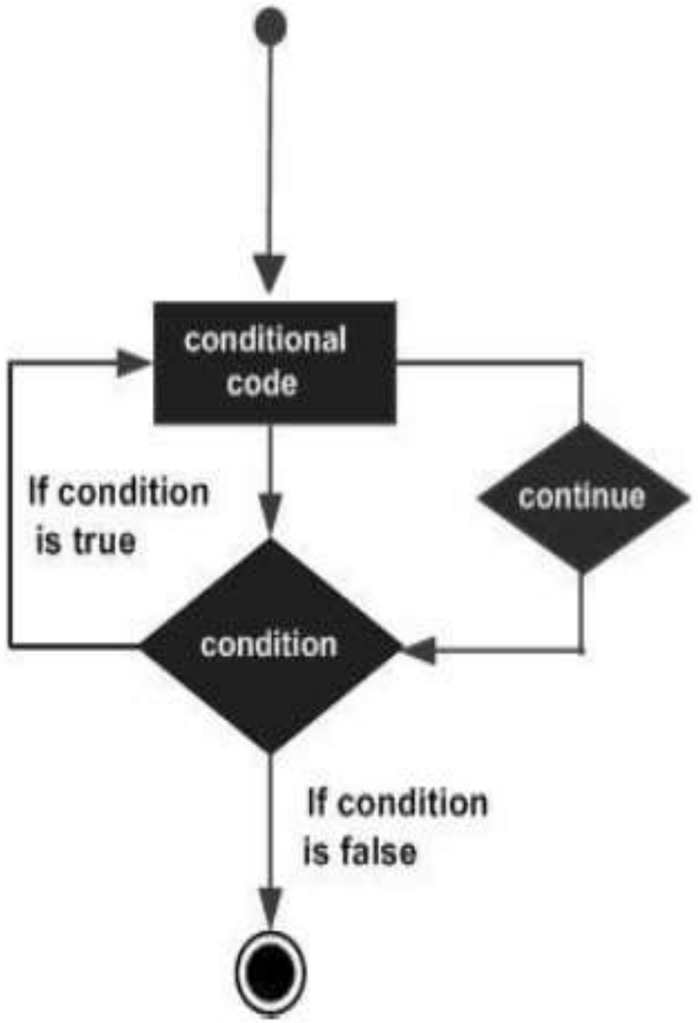
Continue statement

The **continue** statement skips the subsequent statements in the current iteration and takes the control back to the beginning of the loop. Unlike the break statement, the continue doesn't exit the loop.

Syntax:-

```
var num:number = 0
var count:number = 0;

for(num=0;num<=20;num++) {
  if (num % 2==0) {
    continue
  }
  count++
}
console.log (" The count of odd values between 0 and 20 is: "+count)
```

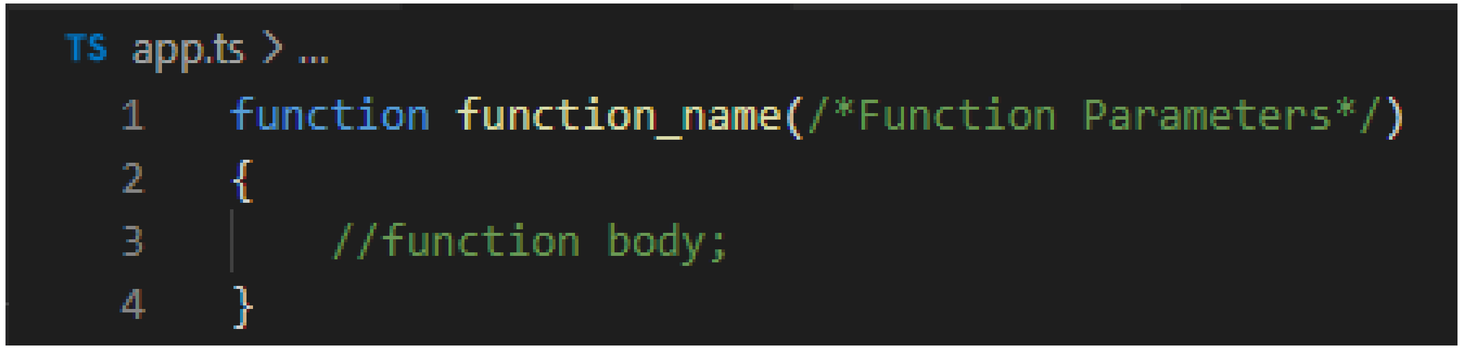
Output:-



Functions

Functions

- Specific parts of programs used to accomplish specific tasks.
- Generally, contain of three parts:
    1. Function name
    2. Function parameters
    3. Function body



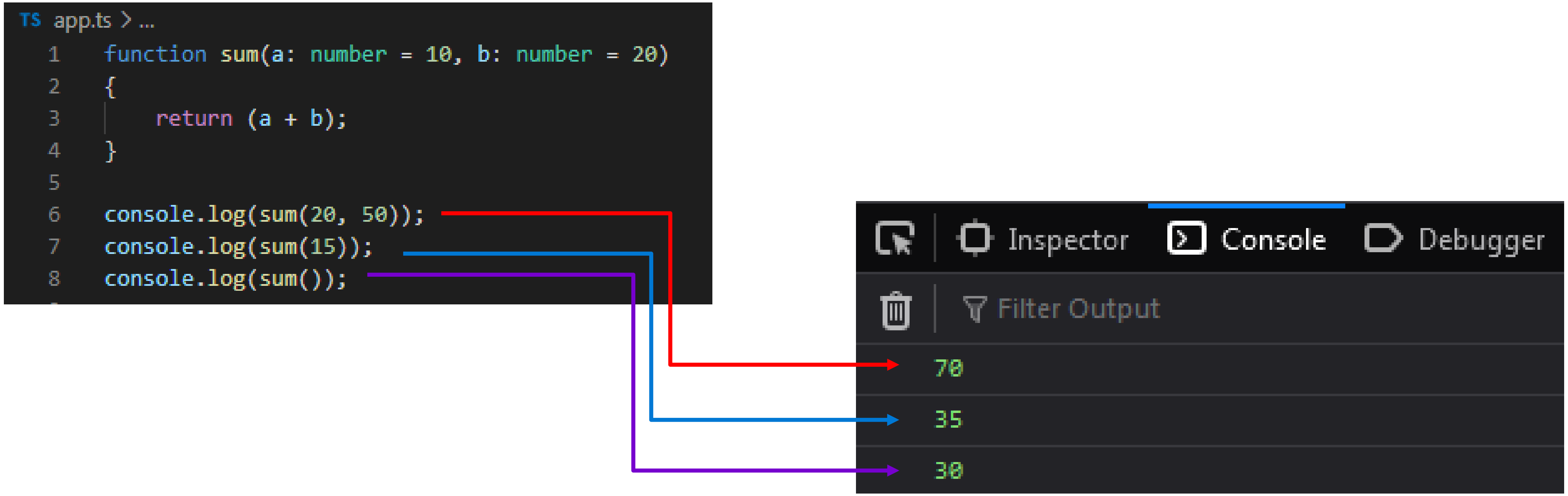
app.ts > ...

```typescript
function function_name(/*Function Parameters*/)
{
    //function body;
}
```

Function Parameters

Default values

- For parameters that use a particular value frequently *(not always)*

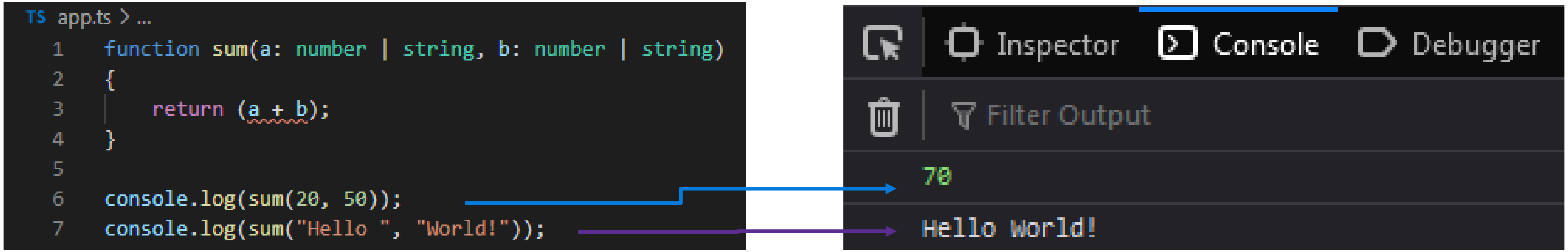


```
function sum(a: number = 10, b: number = 20)
{
    return (a + b);
}

console.log(sum(20, 50));
console.log(sum(15));
console.log(sum());
```

```
70
35
30
```

Union types

- Assigning multiple types to a parameter using the pipe operator (|)

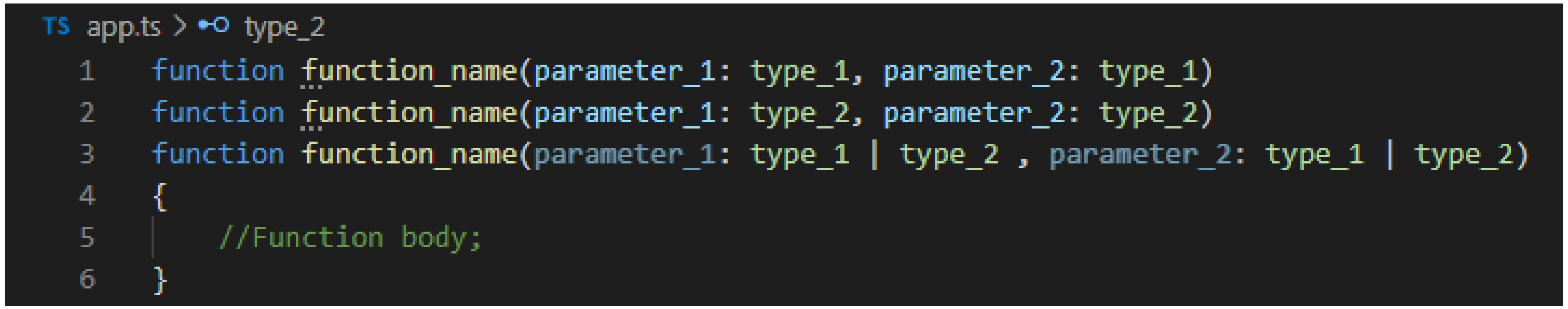

app.ts > ...

```
function sum(a: number | string, b: number | string)
{
    return (a + b);
}

console.log(sum(20, 50));
console.log(sum("Hello ", "World!"));
```

```
70
Hello World!
```

Function Overloading

- Creating multiple functions with the *same name* but *different implementations.*



app.ts > type_2

```
function function_name(parameter_1: type_1, parameter_2: type_1)
function function_name(parameter_1: type_2, parameter_2: type_2)
function function_name(parameter_1: type_1 | type_2 , parameter_2: type_1 | type_2)
{
    //Function body;
}
```

QnA

Topics discussed:

- Decision Making:
  - if…else
  - switch case
- Loops:
  - for
  - while
  - Do…while
- Functions:
  - Default values
  - Union types
  - Function Overloading

Linkedin/shreyakhandelwal99

Linkedin/niralisahoo