**PIAIC**

**Learning type script**

1. **What is web 1.0, 2.0 and 3.0?**

**Web 1.0** was all about **fetching**, and **reading** information.**Web 2.0** is all about **reading**, **writing**, **creating**, and **interact**ing with the end user. It was famously called the participative social web. **Web 3.0** is the third generation of the **World Wide Web**, and is a vision of a **decentralized** web which is currently a work in progress. It is all about **reading**, **writing**, and **owning**.

<https://www.geeksforgeeks.org/web-1-0-web-2-0-and-web-3-0-with-their-difference/>

1. **What is typescript?**

Type Script is JavaScript with syntax for types. Type Script is a strongly typed programming language that builds on JavaScript, giving you better tooling at any scale. Try Type Script Now. Online or via npm. Editor Checks. It’s a superset of java script.

1. **What is node .js?**

Node. Js is an open-source, cross-platform JavaScript runtime environment and library for running web applications outside the client's browser. Ryan Dahl developed it in 2009. It’s a compiler which compile human language in to computer language.

Its latest version is

1. **What is visual studio code?**

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux.

1. Setting up your environment.

There are many ways in which you can set up a coding environment. Such as:

* 1. Integrated Development Environment (IDE). Example: **VS Code,** Sublime Text, Atom, etc.
  2. Web browser. Example: **Chrome**, Firefox, etc.
  3. Online editor (optional). Example: StackBlitz, Replit, etc.

1. Install NodeJs.

<https://nodejs.org/en/download/current/>

Install Version 18.10.0+

node –v

VS Code.

<https://code.visualstudio.com/>

1. Install Typescript

https://www.npmjs.com/package/typescript

npm install -g typescript

tsc —init

Create a new file and name it whatever you want or better name it `index.ts` just for convention.

Open your file in any text editor like vscode, notepad etc.

1. First Typescript Program.

console.log ("hello world")

Run Typescript Program

tsc

It will compile into Java script.

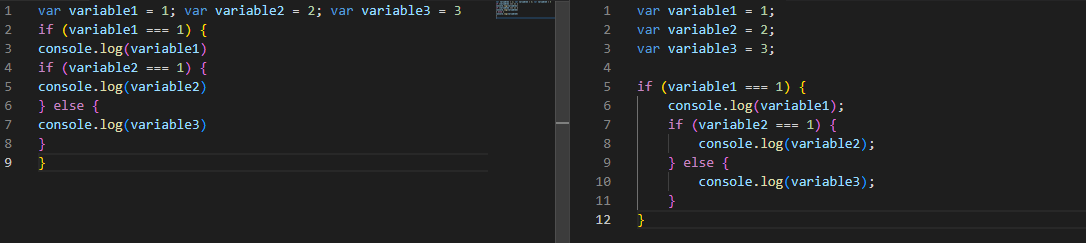
1. Run Typescript Output

node index.js

You will see the output inside the terminal.

1. Writing code

?



1. What is variables?

Variable means anything that can vary. A Type Script variable is simply **a name of storage location**. A variable must have a unique name. Variables are values in your code that can represent different values each time the code runs.

* The first time you create a variable, you declare it. And you need a special word for that: **let** , **var** , or **const** .  
  Example: **let firstname = "Ali";**
* The commonly used naming conventions used for **variables** are camel-case.  
  Example: **let firstName = "Ali";**

1. ﻿ Variable name?

A variable name can't contain any spaces

A variable name can contain only letters, numbers, dollar signs, and underscores.

The first character must be a letter, or an underscore (-), or a dollar sign ($).

Subsequent characters may be letters, digits, underscores, or dollar signs.

Numbers are not allowed as the first character of variable.

1. Comments?

* Single line TypeScript comments **start with two forward slashes (//)**.
* All text after the two forward slashes until the end of a line makes up a comment
* Even when there are forward slashes in the commented text.
* Multi-line Comments
* Multi-line comments start with /\* and end with \*/.
* Any text between /\* and \*/ will be ignored by JavaScript.

1. Statements

A computer program is a list of "instructions" to be "executed" by a computer.

In a programming language, these programming instructions are called statements.

A JavaScript program is a list of programming statements.

TypeScript applications consist of statements with an appropriate syntax. A single statement may span multiple lines. Multiple statements may occur on a single line if each statement is separated by a semicolon.

1. Primitive data types

String

* 1. A string is used to store a text value.  
     Example: **let firstName = "Ali";**

Number

* 1. A number is used to store a numeric value.  
     Example: **let score = 25;**

Boolean

* 1. A Boolean is used to store a value that is either **true** or **false**.  
     Example: **let isMarried = false;**

Undefined

* 1. An undefined type is either when it has not been defined or it has not been assigned a value.  
     Example: **let unassigned;**

Null

* 1. Null is a special value for saying that a variable is empty or has an unknown value.  
     Example: **let empty = null;**

1. **What is Template literals?**

It’s a way to deal with strings and especially dynamic strings; so you don’t need to think more about what’s the next quote to use single or double.

**How to use Template literals?**

It uses a `backticks` to write string within it.

Let myName = "daniyal”;

Let hello = "Hello "+ myName;

console.log (hello); //Hello daniyal

1. Type of data?

let my\_name = "khalid"

console.log(typeof my\_name);

By this method we can find the type of any data.

1. Operators

// ARITHMECTIC OPERATORS A

// 01. adition

let n1= 20

let n2= 10

console.log(n1+n2)

//02. substraction

let n1= 10

let n2= 10

console.log(n2-n1);

//03 multiplication

let n1=30

let n2= 5

console.log(n1\*n2);

//04 division

let n1 = 30

let n2 = 10

console.log(n1 / n2);

//05 exponentiation

let n1 = 4

let n2 = 2

console.log(n1 \*\* n2);

//06 modulus = remaning

let n1 = 10

let n2 = 3

console.log(n1 % n2);

let n1 =15

let n2 =3

console.log(n1 % n2);

//ASSIGNMENT OPERATORS B

/\*let n=5

console.log(n);

let n1 = n+5

console.log(n1);

let n2 = n1+10

console.log(n2);

let n3 = n2-n1

console.log(n3);

// COMPARISION OPERATORS C

let n = 5

console.log(n == 5); //equal to value -true

console.log(n === 5); // equal to value and types -t

console.log(n != 5); // is not equal to value -f

console.log(n > 8); //  n is greatrer then 8 -f

console.log(n < 8); // n is less then 8 –t

console.log(n >=8);// n is greater then equal then 8 -f

console.log(n <=8);// n is less then equal then 8 -t

// LOGICAL OPERATORS D

// symbles

// && [AND OP], || [OR OP], ! [NOT]

// AND Opr. &&

/\* T, T = T

   T, F = F

   F, T = F

   F, F = F\*/

// OR opr. ||

 /\* T, T = T

    T, F = T

    F, T = T

    F, F = F\*/

 //NOT opr. !

/\*T = F

    F = T \*/

//exmples

let n = 5;

console.log(n >=5 && n < 10); // t

console.log(n > 5 && n< 10); // f

console.log(n >= 5 || n< 10); // t

console.log(n > 5 || n< 10); // t

console.log( !(n< 10) ); // f

console.log( !(n> 10) ); // t

console.log(n >= 5 && n< 10); t

console.log(n >= 5 && n> 10); f

1. Functions

To begin, just as in JavaScript, TypeScript functions can be created both as a named function or as an anonymous function. This allows you to choose the most appropriate approach for your application, whether you’re building a list of functions in an API or a one-off function to hand off to another function.

To quickly recap what these two approaches look like in JavaScript:

// Named function

01 method

function addition(x:number, y:number) {

    return x+y

}

let x = 5

let y = 5

console.log(x+y);

function subtraction(x:number, y:number) {

    return x-y

}

let x1 = 20

let y1 = 10

console.log(x1-y1);

function multiplication(x:number, y:number) {

    return x\*y

}

let x2 = 5

let y2 = 2

console.log(x2\*y2);

function division(x:number, y:number) {

    return x/y

}

let x3 = 20

let y3 = 2

console.log(x3/y3);

// Anonymous function

02 method

let num1 = function addition (x:number, y:number) {

    return x+y;

}

let x =5

let y =5

console.log(x+y);

03 method

let z = 10

let num1 = function addition (x:number, y:number) {

    return x+y+z;

}

let x =5

let y =5

console.log(x+y+z);

03 method

function addition(x:number, y:number): number{

    return x+y

}

let x1:number  =5

let y1:number = 5

console.log(x1+y1);

04 method

let myAdd: (x: number, y: number) => number = function (

    x: number,

    y: number

  ): number {

    return x + y;

  };

  let x = 5

  let y = 5

  console.log(x,y);

05 method

  let name\_of\_collage = function name(first\_name:string, last\_name:string) {

    return first\_name+""+last\_name

  }

let first\_name = "Punjab "

let last\_name = "group of collages okara"

console.log(first\_name + last\_name);

06 method

let second\_name = "group of collage "

  let name\_of\_collage = function name(first\_name:string, last\_name:string) {

    return first\_name+""+second\_name +""+last\_name;

  }

let first\_name = "Punjab "

let last\_name = "okara"

console.log(first\_name + second\_name + last\_name);

Function for array?

1. If, Else and Else If Statements

Use **if** to specify a block of code to be executed, if a specified condition is true.

Use **else** to specify a block of code to be executed, if the same condition is false.

Use **else if** to specify a new condition to test, if the first condition is false.

let user\_name = "waleed"

let code =123456

if (user\_name==="waleed") {

    console.log("kindly inter your code");

    if (code ===12345) {

        console.log("welcome to home ");

    }

    else{

        console.log("invalid code");

    }

}else{

        console.log("invalid user");

    }

   let age = 1

if (age < 2) {

console.log("the person is a baby.");

}else if (age>=2 && age<4) {

console.log("the person is a toddler.");

}else if (age>=4 && age<13) {

    console.log("the person is a kid.");

}else if (age>=13 && age<20) {

    console.log("the person is a teenager.");

} else if (age>=20 && age<60) {

    console.log("the person is an adult.");

}else if (age=60 || age>60) {

    console.log("the person is an elder.");

}

1. Arrays

let array = [1, 2, 3, 4, 5]

console.log(array);

let fruits\_array = ["Apple","Banana","Mango","Gava","Pineapple"];

console.log(fruits\_array);

let call\_name\_of\_fruit1 = fruits\_array[0]

console.log(call\_name\_of\_fruit1);

let call\_name\_of\_fruit2 = fruits\_array[1]

let call\_name\_of\_fruit3 = fruits\_array[2]

let call\_name\_of\_fruit4 = fruits\_array[3]

let call\_name\_of\_fruit5 = fruits\_array[4]

console.log(call\_name\_of\_fruit2,call\_name\_of\_fruit3,call\_name\_of\_fruit3,call\_name\_of\_fruit4,call\_name\_of\_fruit5);

Arrays: Adding and removing elements.

When you work with arrays, it is easy to **remove elements** and **add new elements**. This is what **popping** and **pushing** is.

The **pop()** method removes the last element from an array:

The **pop()** method returns the value that was **"popped out"**

let use\_pop = fruits\_array.pop();

console.log(use\_pop);//remove pineapple which was at last

console.log(fruits\_array);

The **push()** method adds a new element to an array (at the end).

The **push()** method returns the new array length.

let fruits\_array = ["Apple","Banana","Mango","Gava","Pineapple"];

let use\_push = fruits\_array.push("Graaps");

console.log(use\_push);//add graaps at the end of array

console.log(fruits\_array);

Arrays: Removing, inserting, and extracting elements

* Shifting is equivalent to popping, but working on the **first element** instead of the **last**.
* The **shift()** method removes the first array element and "**shifts**" all other elements to a lower index.
* The **shift()** method returns the value that was "**shifted out**".
* let fruits\_array = ["Apple","Banana","Mango","Gava","Pineapple"];
* let use\_shift = fruits\_array.shift();
* console.log(use\_shift);//remove apple at the start of array
* console.log(fruits\_array);
* The **unshift()** method adds a new element to an array   
  (at the beginning), and "**unshifts**" older elements:
* The **unshift()** method returns the new array length.
* let fruits\_array = ["Apple","Banana","Mango","Gava","Pineapple"];
* let use\_unshift = fruits\_array.unshift("Orange");
* console.log(use\_unshift);//add orange at the start of array
* console.log(fruits\_array);

**Splicing and Slicing Arrays**

* The **splice()** method adds new items to an array.
* let fruits\_array = ["Apple","Banana","Mango","Gava","Pineapple"];
* //                 [   0        1       2       3         4    ]
* let use\_splice = fruits\_array.splice(4,0,"lemon","kiwi");
* console.log(use\_splice);//add lemon and kiwi after gava and before pineapple at the start of array
* console.log(fruits\_array);
* The **slice()** method slices out a piece of an array.
  + Example:  
    const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];  
    const citrus = fruits.slice(1); // [Orange,Lemon,Apple,Mango]
  + **Notes:**The **slice()** method creates a new array.
* let fruits\_array = ["Apple","Banana","Mango","Gava","Pineapple"];
* //                 [   0        1       2       3         4    ]
* let use\_slice = fruits\_array.slice(1,3);
* console.log(use\_slice);//remove single or set of elements
* //console.log(fruits\_array);

1. For Loops

* Loops are handy, if you want to run the same code over and over again, each time with a different value.
* // Syntax:
* for (expression 1; expression 2; expression 3) {

// code block to be executed

* }
* From the example above, you can read
* Expression 1 sets a variable before the loop starts (let i = 0).
* Expression 2 defines the condition for the loop to run (i must be less than 5).
* Expression 3 increases a value (i++) each time the code block in the loop has been executed.
* for (let i = 0; i <3; i++) {
* console.log("hello world");// output run 3 times
* for (let i = 0; i <=3; i++) {
* console.log("hello world");// output run 4 times
* for (let i = 0; i <3; i++) {
* console.log("hello world" + i);// output run 3 times with index number like hello world0,hello world1,hello world2,
* var cleanestCities = ["Karachi", "Lahore", "Islamabad", "Peshawar"];
* for (var i = 0; i <= 4; i++) {
* if ("Islamabad" === cleanestCities[i]) {
* console.log("It's one of the cleanest cities");
* break;
* }
* }
* let beautycities = ["lahore","okara", "islamabad","sahiwal"];
* for (let i = 0; i <= 4; i++) {
* if ("islamabad" === beautycities[i]) {
* console.log("fine");
* break;
* }
* }