Variables in TypeScript

A variable is a named storage location that holds a value. Variables are fundamental to programming languages, allowing developers to store and manipulate data in their programs. In TypeScript, variables are declared using the **let**, **const**, or **var** keywords.

Variable Declaration:

```
//using let Keyword

let age: number =25;
let name1: string ="Abdul";
let isStudent: boolean =true;

console.log("Variable Declaration with 'let' Keyword");
console.log(age);
console.log(name1);
console.log(isStudent);


//using const (constant) keyword
// Variables which declared with 'const' keyword cannot be reassigned once they
are initialized

const pi:number = 3.14;
const appName: string = "MyApp";


console.log("Variable Declaration with 'const' Keyword");
console.log(pi);
console.log(appName);
```

```
//Type Inference:
//TypeScript can often infer the type based on the assigned value, so explicit
type annotations are not always necessary

let mrks= 95; // TypeScript infers 'number' type
let greeting ="Hi every one" // TypeScript infers 'string' type


console.log("TypeScript infers type");
console.log(mrks);
console.log(greeting);
```

<u>Variable Scope:</u>

Variable scope refers to the region or context of the code where a variable is accessible. The scope determines where in your code a variable can be used and accessed. TypeScript supports two main types of variable scope: global scope and local scope.

<u>1. Global Scope:</u>

Variables declared outside of any function or block have global scope. They are accessible throughout the entire program.

```
// Gloabal Scope

let globalVar: number = 10;

function myFunc(){
    console.log(globalVar) // Accessible within Functions

}

myFunc();
console.log(globalVar); // Accessible outside functions
```

2. Local Scope:

Variables declared inside a function or block have local scope. They are only accessible within the function or block where they are declared.

```
// Local Scope
console.log("Local Scope of Variable");

function myFunc1(){
    let localVar: number = 20;
    const constVar: boolean = true;
    console.log(localVar) // Accessible within Functions
    console.log(constVar); // Accessible within the function
}

myFunc1();
// console.log(localVar); // Error: localVar is not defined outside the function
// console.log(constVar); // Error: constVar is not defined outside the function
```

Block Scope:

Variables in TypeScript have block-level scope when declared with **let** and **const**, which means they are only accessible within the block (enclosed by curly braces) where they are declared.

Variables declared with **let** and **const** also have block-level scope when declared inside curly braces ({}).

```
// Block Scope
console.log("Block Scope of Variable");

if (true) {
    let blockVar: number = 42;
    const blockConst: string = "Block Scope";

    console.log(blockVar)
    console.log(blockConst)
}

// console.log(blockVar); // Error: blockVar is not defined outside the block
// console.log(blockConst); // Error: blockConst is not defined outside the block
```

Purpose of Variables in TypeScript:

Data Storage:

Variables store data values that can be used and manipulated throughout the program.

Manipulating Data:

Variables allow you to perform operations, calculations, and transformations on the stored data.

Code Readability:

Descriptive variable names enhance code readability, making it easier for developers to understand the purpose of a variable and the data it holds.

Flexibility:

Variables make your code more flexible by allowing you to change values dynamically during the execution of the program.

Type Annotations:

Variables with explicit type annotations provide additional information to the TypeScript compiler, enabling static type checking and catching potential errors early in the development process.

Code Organization:

Variables help organize code by encapsulating data in a named container, improving the structure and maintainability of your programs.