step02_const_let

```
//use const where variable values do not change
const a = 5;
const b : number = 33;
const c ="best";

//I suggest use let instead of var everywhere,
//becuase let has blocked scope
if (true) {
        let z = 4;
        //use z}
else {
        let z = "string";
        //use z
}
console.log("let: " + z);// Error: z is not defined in this scope
```

Variable Declaration

In TypeScript, `const` and `let` are two keywords used for declaring variables, with different rules and use cases that help manage the scope and immutability of variables.

let and const provide block scope, replacing the need for var in modern TypeScript.

let allows reassignment and is useful for variables whose values will change.

const prevents reassignment, promoting immutability for variables that should not change after their initial assignment.

Both keywords enhance the readability and maintainability of code by providing clear intentions about variable usage and scope.

Let

Scope: Block-scoped, meaning the variable is only accessible within the block where it is defined (e.g., within a pair of `{}`).

Reassignment: Can be reassigned to a different value.

```
let age: number = 25;
age = 26; // Valid
if (true) {
   let message: string = "Hello";
   console.log(message); // Valid}
// console.log(message); // Error: 'message' is not defined (because it is out of scope)
```

const

- **Scope**: Block-scoped, similar to `let`.
- **Immutability**: Variables declared with `const` cannot be reassigned after their initial assignment. However, this does not mean the value is immutable. If the value is an object or array, its properties or elements can still be modified.

```
const age: number = 25;
// age = 26; // Error: Cannot assign to 'age' because it is a constant

const person = {
   name: "Alice",
   age: 25
};
person.age = 26; // Valid: The object is still mutable
// person = { name: "Bob", age: 30 }; // Error: Cannot reassign the constant variable
```

Differences Between const and let

1. **Reassignment**:
   - `let` allows reassignment.
   - `const` does not allow reassignment after the initial assignment.
2. **Scope**:
   - Both `let` and `const` are block-scoped, meaning they are only accessible within the block they are defined in.
3. **Temporal Dead Zone**:
   - Both `let` and `const` are subject to the Temporal Dead Zone, meaning they cannot be accessed before their declaration in the block scope.