step07a_enum

# Enums

**Enums** (short for "enumerations") in TypeScript are a special data type that allows you to define a set of named constants. Enums provide a way to create a collection of related values that can be referenced by name, enhancing code readability and maintainability.

```
enum Color {Red, Green, Blue};//starts with 0

var c: Color = Color.Green;

enum Color1 {Red = 1, Green, Blue};

var colorName: string = Color1[2];

console.log(colorName); //output green

enum Color2 {Red = 1, Green = 2, Blue = 4};//can assign values to all

var colorIndex = Color2["Blue"];

console.log(colorIndex); //output 4
```

# Types of Enums

TypeScript supports three types of enums:

1. **Numeric Enums**
2. **String Enums**
3. **Heterogeneous Enums**

## 1. Numeric Enums

Numeric enums are the default type. By default, the first value is assigned the numeric value 0, and each subsequent value increments by 1.

```
enum Direction {
  Up,
  Down,
  Left,
  Right}

console.log(Direction.Up);    // Output: 0
console.log(Direction.Down);  // Output: 1
```

## 2. String Enums

String enums allow you to define a set of named constants with string values. This can improve readability by providing meaningful names.

```
enum Color {
   Red = "RED",
   Green = "GREEN",
   Blue = "BLUE"}

console.log(Color.Red); // Output: "RED"
```

## 3. Heterogeneous Enums

```
enum Mixed {
   No = 0,
   Yes = "YES"}

console.log(Mixed.Yes); // Output: "YES"
```

## Using Enums

Enums can be used in various ways, such as in switch statements, comparisons, and more.

*Example with Switch Statement:*

```
enum Status {    Active,     Inactive,     Pending}

function getStatusMessage(status: Status) {
   switch (status) {
      case Status.Active:
         return "The status is active.";
      case Status.Inactive:
         return "The status is inactive.";
      case Status.Pending:
         return "The status is pending.";
      default:
         return "Unknown status.";    }}

console.log(getStatusMessage(Status.Pending)); // Output: "The status is pending."
```

## Reverse Mapping

```
enum Direction {    Up = 1,    Down,    Left,    Right}

console.log(Direction[1]); // Output: "Down"
```

step07b_const_enum

## Const Enums

Const enums are a special type of enums in TypeScript that are defined using the const keyword. They provide a way to define enum values that are inlined during compilation, resulting in better performance and reduced output size

### Defining Const Enums

You define a const enum using the `const` keyword before the `enum` keyword.

```
const enum Direction {    Up = 1,    Down,    Left,    Right}

let move: Direction = Direction.Up;
console.log(move); // Output: 1
```

### Usage of Const Enums

Const enums can be used in the same way as regular enums but with the benefits of inlining

example

```
const enum Color {    Red = "RED",    Green = "GREEN",    Blue = "BLUE"}

function getColorName(color: Color): string {
   return color;}

console.log(getColorName(Color.Green)); // Output: "GREEN"
```

**example**

```
const enum Color {Red, Green, Blue};//starts with 0
var c: Color = Color.Green;

const enum Color1 {Red = 1, Green, Blue};
var colorName: string = Color[2]; //Not allowed with const enums
console.log(colorName);

const enum Color2 {Red = 1, Green = 2, Blue = 4};//can assign values to all
var colorIndex = Color2["Blue"];
console.log(colorIndex);
```