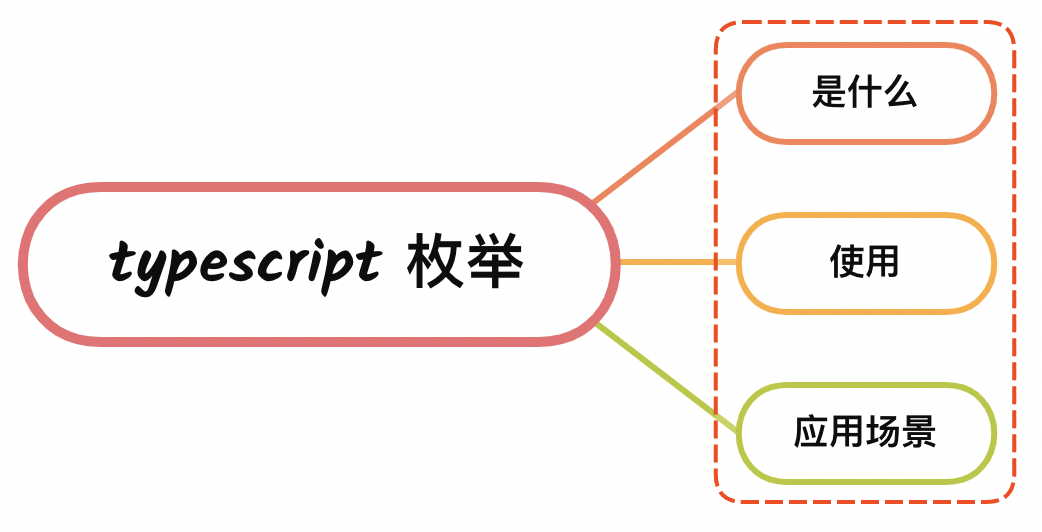
# 面试官：说说你对 TypeScript 中枚举类型的理解？应用场景？



## 一、是什么

枚举是一个被命名的整型常数的集合，用于声明一组命名的常数,当一个变量有几种可能的取值时,可以将它定义为枚举类型

通俗来说，枚举就是一个对象的所有可能取值的集合

在日常生活中也很常见，例如表示星期的SUNDAY、MONDAY、TUESDAY、WEDNESDAY、THURSDAY、FRIDAY、SATURDAY就可以看成是一个枚举

枚举的说明与结构和联合相似，其形式为：

enum 枚举名{  
 标识符①[=整型常数],  
 标识符②[=整型常数],  
 ...  
 标识符N[=整型常数],  
}枚举变量;

## 二、使用

枚举的使用是通过enum关键字进行定义，形式如下：

enum xxx { ... }

声明关键字为枚举类型的方式如下：

// 声明d为枚举类型Direction  
let d: Direction;

类型可以分成：

* 数字枚举
* 字符串枚举
* 异构枚举

### 数字枚举

当我们声明一个枚举类型是,虽然没有给它们赋值,但是它们的值其实是默认的数字类型,而且默认从0开始依次累加:

enum Direction {  
 Up, // 值默认为 0  
 Down, // 值默认为 1  
 Left, // 值默认为 2  
 Right // 值默认为 3  
}  
  
console.log(Direction.Up === 0); // true  
console.log(Direction.Down === 1); // true  
console.log(Direction.Left === 2); // true  
console.log(Direction.Right === 3); // true

如果我们将第一个值进行赋值后，后面的值也会根据前一个值进行累加1：

enum Direction {  
 Up = 10,  
 Down,  
 Left,  
 Right  
}  
  
console.log(Direction.Up, Direction.Down, Direction.Left, Direction.Right); // 10 11 12 13

### 字符串枚举

枚举类型的值其实也可以是字符串类型：  
  
enum Direction {  
 Up = 'Up',  
 Down = 'Down',  
 Left = 'Left',  
 Right = 'Right'  
}  
  
console.log(Direction['Right'], Direction.Up); // Right Up

如果设定了一个变量为字符串之后，后续的字段也需要赋值字符串，否则报错：

enum Direction {  
 Up = 'UP',  
 Down, // error TS1061: Enum member must have initializer  
 Left, // error TS1061: Enum member must have initializer  
 Right // error TS1061: Enum member must have initializer  
}

### 异构枚举

即将数字枚举和字符串枚举结合起来混合起来使用，如下：

enum BooleanLikeHeterogeneousEnum {  
 No = 0,  
 Yes = "YES",  
}

通常情况下我们很少会使用异构枚举

### 本质

现在一个枚举的案例如下：

enum Direction {  
 Up,  
 Down,  
 Left,  
 Right  
}

通过编译后，javascript如下：

var Direction;  
(function (Direction) {  
 Direction[Direction["Up"] = 0] = "Up";  
 Direction[Direction["Down"] = 1] = "Down";  
 Direction[Direction["Left"] = 2] = "Left";  
 Direction[Direction["Right"] = 3] = "Right";  
})(Direction || (Direction = {}));

上述代码可以看到， Direction[Direction["Up"] = 0] = "Up"可以分成

* Direction["Up"] = 0
* Direction[0] = "Up"

所以定义枚举类型后，可以通过正反映射拿到对应的值，如下：

enum Direction {  
 Up,  
 Down,  
 Left,  
 Right  
}  
  
console.log(Direction.Up === 0); // true  
console.log(Direction[0]); // Up

并且多处定义的枚举是可以进行合并操作，如下：

enum Direction {  
 Up = 'Up',  
 Down = 'Down',  
 Left = 'Left',  
 Right = 'Right'  
}  
  
enum Direction {  
 Center = 1  
}

编译后，js代码如下：

var Direction;  
(function (Direction) {  
 Direction["Up"] = "Up";  
 Direction["Down"] = "Down";  
 Direction["Left"] = "Left";  
 Direction["Right"] = "Right";  
})(Direction || (Direction = {}));  
(function (Direction) {  
 Direction[Direction["Center"] = 1] = "Center";  
})(Direction || (Direction = {}));

可以看到，Direction对象属性回叠加

## 三、应用场景

就拿回生活的例子，后端返回的字段使用 0 - 6 标记对应的日期，这时候就可以使用枚举可提高代码可读性，如下：

enum Days {Sun, Mon, Tue, Wed, Thu, Fri, Sat};  
  
console.log(Days["Sun"] === 0); // true  
console.log(Days["Mon"] === 1); // true  
console.log(Days["Tue"] === 2); // true  
console.log(Days["Sat"] === 6); // true

包括后端日常返回0、1 等等状态的时候，我们都可以通过枚举去定义，这样可以提高代码的可读性，便于后续的维护

## 参考文献

* https://zh.wikipedia.org/wiki/%E6%9E%9A%E4%B8%BE
* https://www.jianshu.com/p/b9e1caa4dd98
* https://juejin.cn/post/6844904112669065224#heading-30