Merging and Adding Functionality to Enum

In this lesson, you will see two advanced features: merging and adding functions to enum.

We'll cover the following

- Merging values
- Adding functions

Merging values

Like interfaces, an `enum` can be defined in more than one place. You can start defining the `enum` and later define it again. In the end, all values merge into a single `enum`. There is one constraint with multiple definitions of a single `enum`: the first value of every `enum` must have an explicit value. If an explicit value is defined twice, only the last value will be associated with the `enum` when using the reverse value to find an `enum`. Listing the same value twice is not a feature of multiple definitions; a single enumeration definition can have several entries with the same values as well.

```
enum EnumA {
    ChoiceA,
}
enum EnumA {
    ChoiceB = 1,
}

let variable1: EnumA = EnumA.ChoiceA;
console.log(variable1);
```

```
variable1 = EnumA.ChoiceB;
console.log(variable1);
```

Adding functions

Another feature of `enum` is that you can attach functions that will be accessible statically by the `enum`. Using an `enum` with a function means that you can use `Orientation.East` as well as `Orientation.yourFunction`. Defining a function inside an `enum` requires the use of a namespace with an exported function.

```
enum Orientation {
    East,
    West,
    North,
    South,
}
namespace Orientation {
    export function yourFunction() {
        console.log("I am in a Enum");
    }
}
Orientation.yourFunction();
```

The generated JavaScript hooks the function to the enum's function.

```
(function (variableEnumFunctions) {
    let Orientation;
    (function (Orientation) {
        Orientation[Orientation["East"] = 0] = "East";
        Orientation[Orientation["West"] = 1] = "West";
        Orientation[Orientation["North"] = 2] = "North";
        Orientation[Orientation["South"] = 3] = "South";
```

```
    })(Orientation || (Orientation = {}));
    (function (Orientation) {

        function yourFunction() {
            console.log("I am in an Enum");
        }
        Orientation.yourFunction = yourFunction;
    })(Orientation || (Orientation = {}));
    Orientation.yourFunction();
})(variableEnumFunctions || (variableEnumFunctions = {}));
```

As you can see, the final product is that an `enum` is a function that wraps other functions. Hence, it is possible to add functions to an `enum`.

Back

Accessing Enum Values

Next

Definition and Usages

Mark as Completed

Report an Issue