003 TypeScript – enums and more VS Config

TypeScript Kata List – Blog Page

[TypeScript Kata List on GitHub](https://github.com/robertdunaway/katas-typescript)

# Duration

10 minutes

# Brief

Using enums and setting up Visual Studio to play nicely with TypeScript.

### For More Information

BING/GOOGLE: “TypeScript enums”

# Instructions

Get tutorial folder or the entire katas-typescript repo.

Open the [before/\*.sln] file and execute the kata.

Feel free to execute this kata multiple times because repetition creates motor memory.

# Github

* Before (start kata with this)
  + <https://github.com/robertdunaway/katas-typescript/tree/master/003%20TypeScript%20-%20enums%20and%20more%20VS%20Config/before>
* After
  + <https://github.com/robertdunaway/katas-typescript/tree/master/003%20TypeScript%20-%20enums%20and%20more%20VS%20Config/after>

# Kata

Create an Enum using “Red”, “Green”, and “Blue”. This is a commonly used to demonstrate Enums.

enum Color { Red, Green, Blue };

TIP: Immediately you will notice red lines under your code with a message that indicates these are identifiers are duplicated. This is because the gulp tasks created a copy of the ts file in the wwwroot directory. To resolve this we need to create a new file in the root of our project named “tsconfig.json” and add the following content to it.

{

"exclude": [

"bower\_components",

"node\_modules",

"wwwroot"

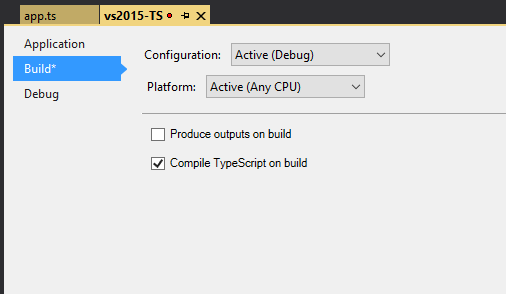
]

}

Voila! Problem solved!

Well, the problem is almost solved. Now when you build the TS file is used to properly generate a minified JS and map file. Everything should work except as you run your project you might notice that Visual Studio does a little file manipulation of its own which completely breaks everything.

To stop Visual Studio from interfering go to the Project 🡪 Properties and select the Build tab. Here you can disable Visual Studio’s compile for TypeScript on build.



Problem solved… for reals this time.

Now, back to work…

Create a variable and set it equal to the “Green” enum value.

var myColor: number = Color.Green;

console.log('myColor = ' + myColor);

Create another variable and set it equal to the enum in 0 (zero) position. By default, enums are zero based.

var myColor2: string = Color[0];

console.log('myColor2 = ' + myColor2);

Create another “color” type enum but this time name the enum start at 5.

enum Color2 { Red = 5, Green, Blue };

Create a variable and set it to one of the new enum values.

var myColor3: number = Color2.Green;

console.log('myColor3 = ' + myColor3);

# Next

Take a few minutes and imagine more examples.