**IS 312 Web Design: TypeScript (TS) for Modern Web Application**

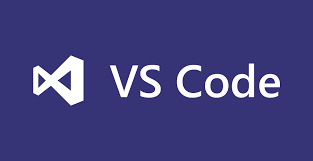
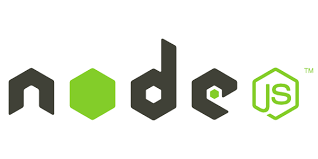
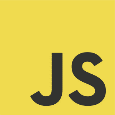
**HOP07: Generic Types**

5/25/2019, Developed by Amrutha Vaidyanathan, Class of 2020

9/23/2020, Revised by Kim Nguyen, Class of 2021

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* Some steps are not explained in the tutorial**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Creating reusable components that can work with variety of types rather than a single one

**Resources**

* [TypeScript documentation](https://www.typescriptlang.org/docs/handbook/generics.html)

**Preparation**

1. Open the VS Code, git clone <URL>(to download this Github repository to your local machine)

1. Change directory to the corresponding folder of each week. For example: Your work for module 1 should be stored under Module 1 folder; your work for module 2 should be stored under Module 2, and so on:

* cd “Module 10”

1. Now, follow the instructions provided in each folder to complete your Hands-on Practice

**Identity function**

The identity function is a function that will return back whatever is passed in. It behaves in the same way as echo command.

1. Create a file **Identity.ts** under **Module10** and type the following code.

A screen shot of a social media post

Description automatically generated

1. In the VS Code terminal compile and run the TS code.

>>> tsc Identity.ts

>>> node Identity.js

A close up of a sign

Description automatically generated

**any** is generic that will cause the function to accept any and all types for the type of **arg**.

**How to capture the type of the argument in such a way that we can also use it to denote what is being returned**

Type variable, a special kind of variable that works on types rather than values.

1. Create a file **Identity\_1.ts** under **Module10** and type the following code.

Screen of a cell phone

Description automatically generated

1. In the VS Code terminal compile and run the TS code.

>>> tsc Identity\_1.ts

>>> node Identity\_1.js



The type variable is T and it is added to the identity function. T will capture the type the user provides.

arg:T is the argument type, the next T mentioned is the return type.

**Generic Type variable**

When using generic functions (eg Identity) we must use generically typed parameters in the body of the function correctly.

1. Create a file **gt\_variable.ts** and type the following code.

A picture containing object, clock, meter

Description automatically generated

1. In the VS Code terminal compile and run the TS code.

>>> tsc gt\_variable.ts

>>> node gt\_variable.js

A picture containing drawing

Description automatically generated

The reason for the above error is because the argument passed might be a number. In that case the number type does not have the length function.

Let look at how arrays are being used

1. Create a file **gt\_array.ts** and type the following code.

A picture containing screen, monitor, black, large

Description automatically generated

1. In the VS Code terminal compile and run the TS code.

>>> tsc gt\_array.ts

>>> node gt\_array.js



The loggingIdentity function takes a type parameter T, and an argument arg which is an array of Ts and returns an array of Ts.

The above code can be written in an alternative way. Look at the highlighted line. You don’t have to type the code. The below code will produce the same output as above code.

A close up of a screen

Description automatically generated

**Multiple parameter passing**

1. Create a file **gt\_multi.ts** and type the following code.

A picture containing monitor, meter, clock, holding

Description automatically generated

1. In the VS Code terminal compile and run the TS code.

>>> tsc gt\_multi.ts

>>> node gt\_multi.js



The T, U and V are the generic type for the argument args1, args2 and args3 respectively. The return type is V and we are returning the args3.

**Generic Classes**

The generic type parameter is specified in angular brackets after the name of the class. A generic class can have generic fields (member variables) or methods.

1. Create a file **generic\_class.ts** and type the following code.

A screenshot of a cell phone

Description automatically generated

1. In the VS Code terminal compile and run the TS code.

>>> tsc generic\_class.ts

>>> node generic\_class.js

A black sign with white text

Description automatically generated

We created a generic class named KeyValuePair with a type variable in the angle brackets <T, U>. The KeyValuePair class includes two private generic member variables and a generic function setKeyValue that takes two input arguments of type T and U. This allows us to create an object of KeyValuePair with any type of key and value.

**Generic Class implements Generic Interface**

1. Create a file **generic\_class1.ts** and type the following code.

A screen shot of a monitor

Description automatically generated

1. In the VS Code terminal compile and run the TS code.

>>> tsc generic\_class1.ts

>>> node generic\_class1.js



The generic class kvProcessor implements the generic interface IKeyValueProcessor. It does not specify the type parameters T and U, instead it allows users to set them themselves. Thus, kvProcessor class can be used with any type of key and value. A variable is defined as generic interface type with underlying types for T and U. So, you don't need to set the generic types for kvProcessor.

**Generic Constraints**

Look at the example under the Generic Type variable (1). In that code we want to access the .length property of arg, but the compiler could not prove that every type had a .length property.

To solve the above problem, we’d like to constrain this function to work with any and all types that also have the .length property.

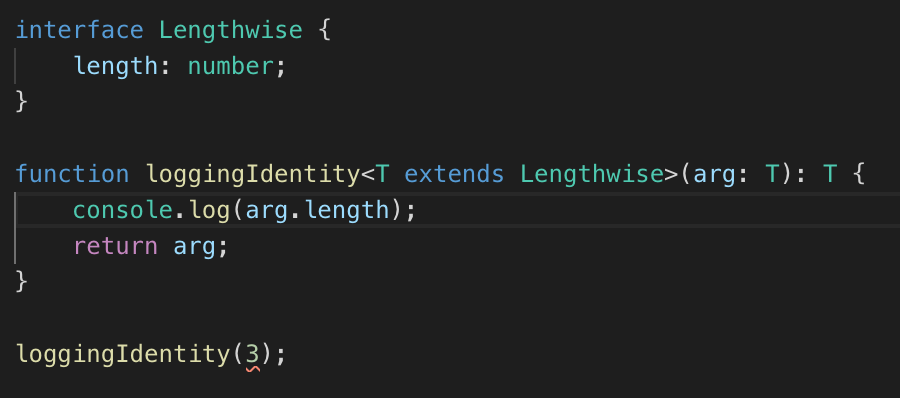
1. Create a file **generic\_constraints.ts** and type the following code.

A screenshot of a cell phone

Description automatically generated

We have created an interface that describes our constraint. Here, we’ll create an interface that has a single .length property and then we’ll use this interface and the extends keyword to denote our constraint.

Let’s pass a value to check if it is working



1. In the VS Code terminal compile and run the TS code.

>>> tsc generic\_constraints.ts

>>> node generic\_constraints.js



A picture containing drawing

Description automatically generated

The error is because the generic function is now constrained, it will no longer work over any and all types. The number doesn't have a .length property.

Let’s pass in values whose type has all the required properties. Replace the above highlighted line with the one below.

A screenshot of a cell phone

Description automatically generated

Compile and run the above code.



**Push your work to GitHub**

Run the following commands to push your work to the GitHub repository:

Open the terminal from the VSCode by hitting the control + ~ key and type the following command:

**>>> git add .**

**>>> git commit -m “Submission for Module 10– YOUR NAME”**

**>>> git push origin master**