100 XP

Lab - Declare a class by using a generic

20 minutes

In this lab, you will extend the functionality of a class by using generics.

Exercise 1: Declare a class by using a generic

The `DataStore` class contains utility functions that can store up to ten `string` items in an array and return the value stored in each item. In this exercise, you will rewrite the `DataStore` class using a generic so it can store items of any type that you specify when it is instantiated.

1. Clone the starting repository by entering the following at the command prompt.

```bash
git clone https://github.com/MicrosoftDocs/mslearn-typescript
cd mslearn-typescript/code/module-06/m06-start
code .
```

2. Open the file **module06.ts**.

3. Locate `TODO: Add and apply a type variable.`

4. In the `DataStore` class declaration, add a type variable called `T`.

```typescript
class DataStore<T> {
    // existing code...
}
```

5. Add the type variable `T` to the `_data` property declaration.

```typescript
private _data: Array<T> = new Array(10);
```

6. In the `AddOrUpdate` function, update the type of the `item` parameter to the type variable `T`.

```
TypeScript

AddOrUpdate(index: number, item: T) {
    if(index >=0 && index <10) {
        this._data[index] = item;
    }
}
```

7. Locate `TODO Test items as numbers`.

8. Test that the type variable can accept numbers. Declare a new variable called `empIDs` and assign a new `DataStore` object to it. Call the `AddOrUpdate` function and assign number type items to it.

```
TypeScript

let empIDs = new DataStore<number>();
empIDs.AddOrUpdate(0, 50);
empIDs.AddOrUpdate(1, 65);
empIDs.AddOrUpdate(2, 89);
console.log(empIDs.GetData(0));          // returns 50
```

9. Locate `TODO Test items as objects`.

10. Test that the type variable can accept a custom object. Declare a `type` called `Pets` that contains three properties: `name` as a `string`, `breed` as a `string`, and `age` as a `number`. Declare a new variable called `pets` and assign a new `DataStore` object to it. Call the `AddOrUpdate` function and assign `Pet` objects to it.

```
TypeScript

type Pets = {
    name: string;
    breed: string;
    age: number
}
let pets = new DataStore<Pets>();
pets.AddOrUpdate(0, { name: 'Rex', breed: 'Golden Retriever', age: 5});
pets.AddOrUpdate(1, { name: 'Sparky', breed: 'Jack Russell Terrier', age:
3});
```

```
console.log(pets.GetData(1));          // returns { name: 'Sparky', breed:
'Jack Russell Terrier', age: 3 }
```

Lab solution

View the final version of the code by entering the following at the command prompt.

Bash

```
cd ../m06-end
code .
```

Open the file **module06.ts** to see the solution to this lab. See the **Lab setup** section above for more information about setting up your development environment to run the solution.

Next unit: Knowledge check

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆