✓ 100 XP

Lab - Export and import module components

20 minutes

In this lab, you'll organize some TypeScript code using modules. As an additional challenge, you'll import an external type library.

Exercise 1

In this exercise, you'll organize the code in a project using modules. The project contains three TypeScript files:

- **module07_main.ts** - Contains the main code of the application.
- **module07_loans.ts** - Contains two interfaces, `Loan` and `ConventionalLoan`.
- **module07_loan-programs.ts** - Contains three functions:
  - `calculateInterestOnlyLoanPayment`, which calculates the payment for an interest only loan.
  - `calculateConventionalLoanPayment`, which calculates the payment for a conventional loan.
  - `calculateInterestRate`, a worker function that calculates the monthly interest rate of the loan.

The `calculateInterestOnlyLoanPayment` and `calculateConventionalLoanPayment` functions accept `principle` and `interestRate` parameters. The difference between them is that the `calculateConventionalLoanPayment` function accepts a third property, `months` that the `calculateInterestOnlyLoanPayment` function does not.

Property	Description
`principle`	The principle amount of the loan.

Property	Description
`interestRate`	The annual interest rate of the loan. For example, 5% is specified as 5.
`months`	The term of the loan specified in months. An interest only loan does not require this property because the number of months is irrelevant (the loan will never be repaid when an interest only payment is made each month.)

Add the required code to define the relationships between the modules.

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

Bash

```bash
git clone https://github.com/MicrosoftDocs/mslearn-typescript
cd mslearn-typescript/code/module-07/m07-start
code .
```

2. Open the file **module07_loans.ts** and add the `export` keyword on the interface declarations.

TypeScript

```typescript
export interface Loan {
    principle: number,
    interestRate: number        //* Interest rate percentage (eg. 14 is 14%)
}
export interface ConventionalLoan extends Loan {
    months: number      //* Total number of months
}
```

3. Open the file **module07_loan-programs.ts**.

4. At the top of the file, add an `import` statement that imports the `Loan` and `ConventionalLoan` interfaces from **module07_loans.ts**. Import both interfaces using one `import` statement and assign them to a variable called `Loans`.

TypeScript

```typescript
import * as Loans from './module07_loans.js';
```

5. Locate `TODO Update the calculateInterestOnlyLoanPayment function`.
6. Add the `export` keyword to the `calculateInterestOnlyLoanPayment` function declaration.
7. Update the type of the function parameter `loanTerms` to the interface `Loans.Loan`.

TypeScript

```
export function calculateInterestOnlyLoanPayment(loanTerms: Loans.Loan): 
string {
    let payment: number;
    payment = loanTerms.principle * 
calculateInterestRate(loanTerms.interestRate);
    return 'The interest only loan payment is ' + payment.toFixed(2);
}
```

8. Locate `TODO Update the calculateConventionalLoanPayment function`.
9. Add the `export` keyword to the `calculateConventionalLoanPayment` function declaration.
10. Update the type of the function parameter `loanTerms` to the interface `Loans.ConventionalLoan`.

TypeScript

```
export function calculateConventionalLoanPayment(loanTerms: 
Loans.ConventionalLoan): string {
    let interest: number = calculateInterestRate(loanTerms.interestRate);
    let payment: number;
    payment = loanTerms.principle * interest / (1 - (Math.pow(1/(1 + inter-
est), loanTerms.months)));
    return 'The conventional loan payment is ' + payment.toFixed(2);
}
```

11. Open the file **module07_main.ts**.
12. Locate `TODO Add the import statement`.
13. Add an `import` statement that imports the `interestOnlyLoan` and `conventionalLoan` functions from **module07_loan-programs.ts**. Assign the functions to a variable called `LoanPrograms`.

TypeScript

```
import * as LoanPrograms from './module07_loan-programs.js';
```

14. Locate `TODO Update the function calls`.

15. In the two variable declarations, update the function calls to reference the `LoanPrograms` variable from the `import` statement.

TypeScript

```
let interestOnlyPayment =
LoanPrograms.calculateInterestOnlyLoanPayment({principle: 30000,
interestRate: 5});
let conventionalLoanPayment =
LoanPrograms.calculateConventionalLoanPayment({principle: 30000,
interestRate: 5, months: 180});
```

16. Save the files.

17. At the command prompt, run the `tsc` command using the `--module commonjs` option to compile **module07_main.ts**.

Bash

```
tsc --module commonjs module07_main.ts
```

18. Test your work in `node` by running the **module07_main.js** file.

Challenge

Select a JavaScript library that you are familiar with and try importing it into a TypeScript file using the `import` statement. After it is imported, it should work exactly the same as it does in JavaScript.

Lab solution

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

Bash

```
cd ../m07-end
code .
```

Open the files **module07_main.ts**, **module07_loans.ts**, and **module07_loan-programs.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? ☆ ☆ ☆ ☆ ☆