step03a_modules

Modules in TypeScript

Modules in TypeScript are a way to organize and manage code by splitting it into smaller, reusable pieces. They allow you to encapsulate code, making it easier to maintain, understand, and reuse. Modules are a key feature of TypeScript and JavaScript, supporting the concept of modular programming.

Basic Concepts

- **Module**: A file that contains TypeScript code. Any file containing an import or export statement is considered a module.
- **Export**: Makes code (variables, functions, classes, etc.) available for use in other modules.
- **Import**: Brings in code from other modules into the current module.

Exporting

You can export members from a module using the export keyword. There are two primary types of exports: named exports and default exports.

Named Exports

Named exports allow you to export multiple members from a module by their names.

```
// mathUtils.ts
export function add (a: number, b: number): number {
   return a + b;}
export function subtract (a: number, b: number): number {
   return a - b;}
```

You can import named exports using their exact names.

```
// app.ts
import { add, subtract } from './mathUtils';
console.log(add(5, 3)); // Output: 8
console.log(subtract(5, 3)); // Output: 2
```

Default Exports

Default exports allow you to export a single member from a module as the default export.

```
// logger.ts
export default function log(message: string): void {
   console.log(message);}
```

You can import default exports without using curly braces and can assign any name to the imported member.

```
// app.ts
import log from './logger';
log('Hello, world!'); // Output: Hello, world!
```

Importing

You can import members from a module using the `import` keyword.

Importing Named Exports

```
import { add, subtract } from './mathUtils';
```

You can also import all named exports as a single object.

```
import * as MathUtils from './mathUtils';
console.log(MathUtils.add(5, 3)); // Output: 8
console.log(MathUtils.subtract(5, 3)); // Output: 2
```

Importing Default Exports

```
import log from './logger';
```

Re-exports

Modules can re-export members from other modules.

```
// reExport.ts
export { add, subtract } from './mathUtils';
// app.ts
import { add, subtract } from './reExport';
console.log(add(5, 3)); // Output: 8
console.log(subtract(5, 3)); // Output: 2
```

second example

First file name app.ts

```
import a from "./first";
import {b, c as d} from "./second";
console.log(a + b + d);
```

second File name first.ts

```
let a = 5;
export default a;
```

Third file name second.ts

```
export const b = 10;
export const c = 2;
```

step03b_native_ECMAScript_mod

Using Native ECMAScript Modules in Node.js

Native ECMAScript Modules (ESM) are a standardized module system for JavaScript, designed to provide a more efficient, maintainable, and scalable way to structure JavaScript applications. Introduced in ECMAScript 2015 (ES6), ESM allows developers to define modules that can export and import functionalities between files.

Key Features of ECMAScript Modules

Static Structure: ESMs have a static structure, meaning the imports and exports are known at compile-time. This allows for better optimization and more reliable module resolution.

import and export Keywords: These keywords are used to import and export functionalities from modules.

Scope: ESMs have their own scope, so variables and functions declared in a module are not accessible outside the module unless explicitly exported.

Basic Syntax

Exporting

You can export variables, functions, classes, and objects from a module using export.

First file name app.ts

```
import a from "./first.js";
import {b, c} from "./second.js";
console.log(a + b + c);
```

second file name first.ts

```
let a = 5;
export default a;
```

Third file name second.ts

```
const b = 10;
const c = 2;
export {b, c};
```

TS main.ts

```
1   import subtract from "./index.js";
2   console.log(subtract(12,12));
3
4   import { myName, myFood , myAge as newAge, obj as object } from "./i
5   console.log(myName);
6
7   // import {myFood} from "./index.js";
8   console.log(myFood);
9
10  // import { myAge as newAge } from "./index.js";
11  console.log(newAge)
12
```

TS index.ts × TS main.ts

TS index.ts > ...

```
1   export default subtract;
    tabnine: test | explain | document | ask
2   function subtract (numb1 : number , numb2 : nu
3       return numb1 - numb2;
4   };
5
6   let myName = " Areesha Tanoli"
7   let myFood : string []= ["Birayni", "pizza",
8   let myAge = 20;
9   export {myName,myFood,myAge,obj}
10
11  let obj = {
12      name : "Areesha Tanoli",
13      age : 30,
```

https://www.youtube.com/watch?v=rrHxGITHxjI

step03c_import_inquirer_ECMAScrip

Using Inquirer Package

The latest version (9+) of Inquirer has start using Native ECMA Script Packages. In most of our projects and assignment we will use this package.

Give the following command:

```
npm i inquirer

npm i --save-dev @types/inquirer
```

Add .gitignore file and Write your code in app.ts file.

Give the following commands:

```
tsc
node app.js
```

```
import inquirer from "inquirer";

let answers = await inquirer.prompt([{
   name: "age",
   type: "number",
   message: "Enter your Age:"}
]);

console.log("hope you , in " + (60 - answers.age) + " years you will be 60 years old.");
```

Inquirer

[?]

npm package 5.1.2 | license scan passing

A collection of common interactive command line user interfaces.

```
~/Documents/oss/Inquirer.js/examples     master*
❯ node list.js
? What do you want to do? Order a pizza
? What size do you need?
  Jumbo
  Large
❯ Standard
  Medium
  Small
  Micro
```

```
import input from '@inquirer/input';

const answer = await input({ message: 'Enter your name' });
```

Prompts

Input

```
~/Documents/oss/Inquirer.js    master*
❯ node examples/input.js
? What's your first name Simon
? What's your last name Doe
? What's your phone number
>> Please enter a valid phone number
```

```
import select, { Separator } from '@inquirer/select';

const answer = await select({
  message: 'Select a package manager',
  choices: [
    {
      name: 'npm',
      value: 'npm',
      description: 'npm is the most popular package manager',    },
    {
      name: 'yarn',
      value: 'yarn',
      description: 'yarn is an awesome package manager',
    },
    new Separator(),
    {
      name: 'jspm',
      value: 'jspm',
      disabled: true,
    },
    {
      name: 'pnpm',
      value: 'pnpm',
      disabled: '(pnpm is not available)',
    },
  ],
});
```

Select

```
~/Documents/oss/Inquirer.js/examples     master*
❯ node list.js
? What do you want to do? Order a pizza
? What size do you need?
  Jumbo
  Large
❯ Standard
  Medium
  Small
  Micro
```

```
import checkbox, { Separator } from '@inquirer/checkbox';

const answer = await checkbox({
  message: 'Select a package manager',
  choices: [
    { name: 'npm', value: 'npm' },
    { name: 'yarn', value: 'yarn' },
    new Separator(),
    { name: 'pnpm', value: 'pnpm', disabled: true },
    {
      name: 'pnpm',
      value: 'pnpm',
      disabled: '(pnpm is not available)',    },  ],});
```

Checkbox

```
❯ node examples/checkbox.js
? Select toppings
 ◉ Cheddar
 ◯ Parmesan
   = The usual =
❯◉ Mushroom
 ◯ Tomato
   = The extras =
 ◯ Pineapple
(Move up and down to reveal more choices)
```

```
import confirm from '@inquirer/confirm';

const answer = await confirm({ message: 'Continue?' });
```

Confirm

```
~/Documents/oss/Inquirer.js    master*
❯ node examples/pizza.js
Hi, welcome to Node Pizza
? Is this for delivery? (y/N)
```

```
import password from '@inquirer/password';

const answer = await password({ message: 'Enter your name' });
```

Password

```
~/Documents/oss/Inquirer.js    master*
❯ node examples/password.js
? Enter a password [hidden]
? Enter a masked password **********
```

```
import expand from '@inquirer/expand';

const answer = await expand({
  message: 'Conflict on file.js',
  default: 'y',
  choices: [
    {
      key: 'y',
      name: 'Overwrite',
      value: 'overwrite',    },
    {
      key: 'a',
      name: 'Overwrite this one and all next',
      value: 'overwrite_all',    },
    {
      key: 'd',
      name: 'Show diff',
      value: 'diff',    },
    {
      key: 'x',
      name: 'Abort',
      value: 'abort',    },
  ], });
```

Expand

```
~/Documents/oss/Inquirer.js    master*
❯ node examples/expand.js
? Conflict on `file.js`:   (yadxH) y
>> Overwrite
```

```
~/Documents/oss/Inquirer.js    master*
❯ node examples/expand.js
? Conflict on `file.js`:   (yadxH)
  y) Overwrite
  a) Overwrite this one and all next
  d) Show diff
  ──────────────
  x) Abort
  h) Help, list all options
  Answer: d
```

```
import rawlist from '@inquirer/rawlist';

const answer = await rawlist({
  message: 'Select a package manager',
  choices: [
    { name: 'npm', value: 'npm' },
    { name: 'yarn', value: 'yarn' },
    { name: 'pnpm', value: 'pnpm' },
  ],});
```

Raw List

```
~/Documents/oss/Inquirer.js    master*
❯ node examples/rawlist.js
? What do you want to do? Order a pizza
? What size do you need
  1) Jumbo
  2) Large
  3) Standard
  4) Medium
  5) Small
  6) Micro
```

step03d_chalk

Using Inquirer and Chalk Package

The latest version of Inquirer(9+) and Chalk(5+) have started using Native ECMA Script Packages.

In most of our projects and assignment we will use these packages.

Give the following command:

```
npm i inquirer

npm i --save-dev @types/inquirer

npm install chalk
```

Add .gitignore file and Write your code in app.ts file.

Give the following commands:

```
tsc

node app.js
```

```
import inquirer from "inquirer";
import chalk from "chalk";

let answers = await inquirer.prompt([{
   name: "age",
   type: "number",
   message: "Enter your Age:"}
]);

console.log(chalk.blue("yes you, in " + (60 - answers.age) + " years you will be 60 years old."));
```