FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

THỰC HỌC – THỰC NGHIỆP

LẬP TRÌNH TYPESCRIPT

MODULES & NAMESPACES

http://www.poly.edu.vn

Mục tiêu

- Introduction to modules
- Module exports
- Namespaces
- Using namespaces
- Namespace import

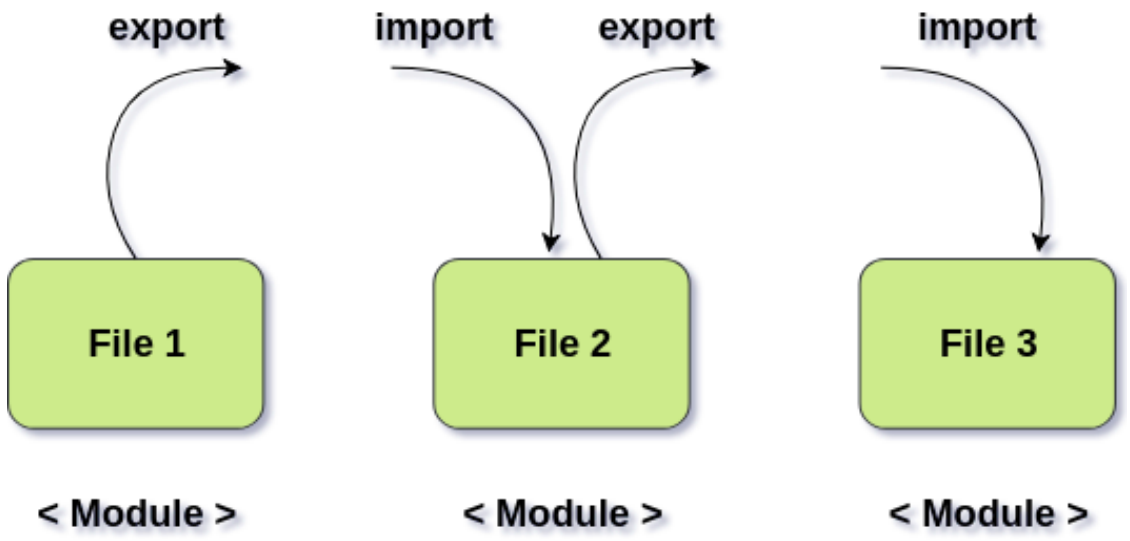
Mục tiêu

- Webpack là gì?
- Cài đặt và sử dụng webpack

PHẦN 1

MODULES

- Module: mục đích để tổ chức lại code được viết bằng typescript



MODULES

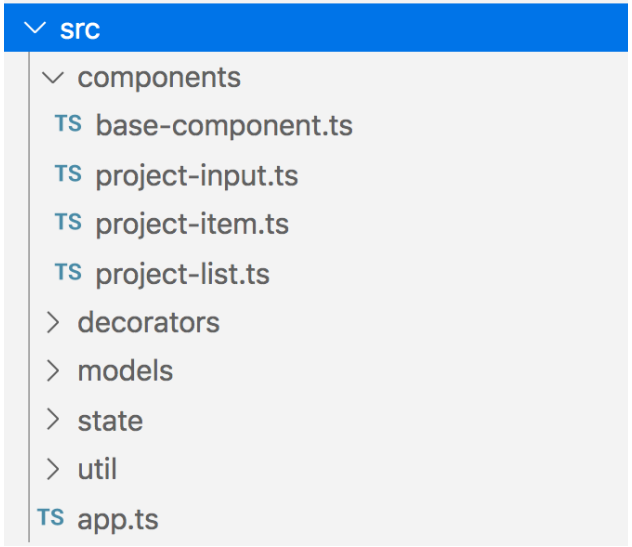
- Cú pháp: sử dụng từ khoá ***export***

```
//FileName : EmployeeInterface.ts
export interface Employee {
    //code declarations
}
```

- Dùng từ khoá ***import*** để khai báo sử dụng module từ file khác. Cú pháp

```
import { class/interface name } from 'module_name';
```

Ví dụ



TÌM ĐIỂM SAI?

app.ts

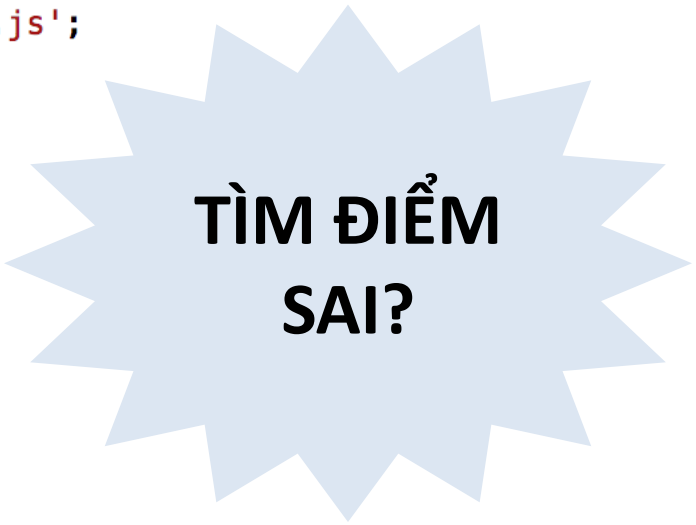
```
import { ProjectInput } from './components/project-input.js';
import { ProjectList } from './components/project-list.js';

new ProjectInput();
new ProjectList('active');
new ProjectList('finished');
```

- Import nhiều module của cùng 1 file. Cú pháp

```
import { export_name1, export_name2 } from 'module_name';
```

- Ví dụ

```
import { DragTarget } from '../models/drag-drop.js';
import { Project, ProjectStatus } from '../models/project.js';
import { Component } from './base-component.js';
import { autobind } from '../decorators/autobind.js';
import { projectState } from '../state/project-state.js';
import { ProjectItem } from './project-item.js';
```



MODULES

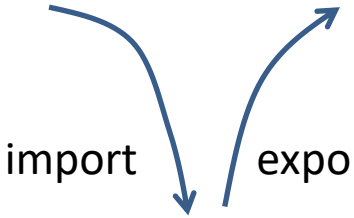
- Biên dịch và thực thi module
- Cú pháp

```
--module <target> <file path>
```

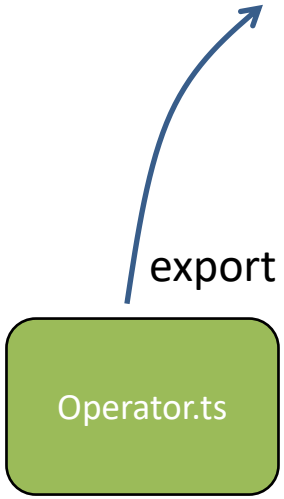
- Ví dụ

```
tsc --module commonjs app.ts
```

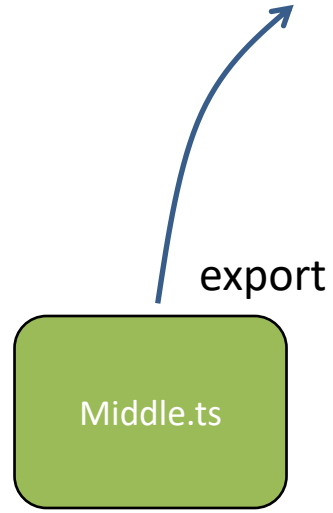
Ví dụ



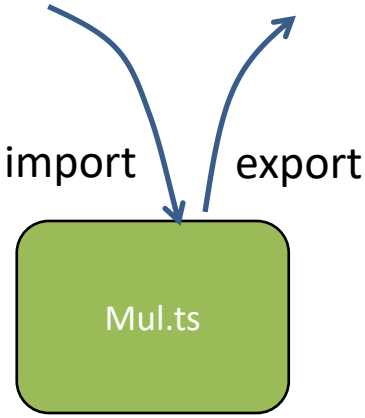
Import Interface Operator
Export Class Add implements Operator



Export default Interface Operator



Export Add, Mul



Import Interface Operator
Export Class Mul implements Operator

- Ví dụ: *Operator.ts*

```ts
interface Operator {
    eval(a: number, b: number): number;
}
export default Operator;
```

```ts
//Add.ts
import Operator from "./Operator";
export class Add implements Operator {
    eval(a: number, b: number): number {
        return a + b;
    }
```

```ts
//Mul.ts
import Operator from "./Operator";
export class Mul implements Operator {
    eval(a: number, b: number): number {
        return a * b;
    }
}
```

RE-EXPORT

- *Middle.ts*

```
export {Add} from "./Add";
export {Mul} from "./Mul";
```

NAMESPACES

- Namespace được sử dụng để gom nhóm logic các chức năng
- Namespace có thể gồm: interfaces, class, function và variable để hỗ trợ một hoặc một nhóm các chức năng liên quan

NAMESPACE

Cú pháp

```
namespace namespace_name
{

}
```

Ví dụ

```
namespace studentCalc{
  export function AnualFeeCalc(feeAmount: number, term: number){
  return feeAmount * term;
  }
}
```

NAMESPACES

- Để truy xuất đến interfaces, classes, functions và variables trong các namespace khác, sử dụng cú pháp sau:

```
namespace_name.className;
namespace_name.functionName;
```

USING NAMESPACES

- Tham chiếu ***namespace*** từ 1 file typescript
- Cú pháp

```
/// <reference path="Namespace_FileName.ts" />
```

- Ví dụ

```
/// <reference path="./studentCalc.ts" />

let TotalFee = studentCalc.AnualFeeCalc(1500, 4);

console.log("Output: " +TotalFee);
```

NAMESPACES

- Biên dịch và thực thi namespace
- Cú pháp

```
--outFile <file js> <file ts>
```

- Ví dụ

```
tsc --outFile sample.js Test.ts
```

- Ví dụ: thực thi nhiều file module typescript vào 1 file js

```
tsc --outFile combine.js filename1.ts filename2.ts
```

NAMESPACE & MODULES

- Module
  - Có thể chứa cả code và khai báo
  - Sử dụng từ khoá export để tạo hiển thị các chức năng module
- Namespace
  - Cách tổ chức mã dành riêng cho typescript
  - Sử dụng từ khoá namespace và từ khoá keyword để hiển thị các thành phần namespace

NAMESPACE & MODULES

- Module
  - Phải import trước và sử dụng ở nơi khác
  - Dùng lệnh ***--module*** để biên dịch
  - ***Export*** tất cả trong 1 module là có thể truy xuất ngoài module
- Namespace
  - Dùng cú pháp tham chiếu để sử dụng
  - Dùng lệnh ***--outFile*** để biên dịch
  - Phải ***export*** functions và classes để có thể truy xuất ngoài namespace

NAMESPACE & MODULES

Khai báo module

- FileName: **addition.ts**

```
export class Addition{
  constructor(private x?: number, private y?: number){
  }
  Sum(){
      console.log("SUM: " +(this.x + this.y));
  }
}
```

Khai báo namespace

- FileName: **StoreCalc.ts**

```
namespace invoiceCalc {
  export namespace invoiceAccount {
     export class Invoice {
        public calculateDiscount(price: number) {
           return price * .60;
        }
     }
  }
}
```

Truy xuất module

```
import {Addition} from './addition';
let addObject = new Addition(10, 20);
addObject.Sum();
```

Truy xuất namespace

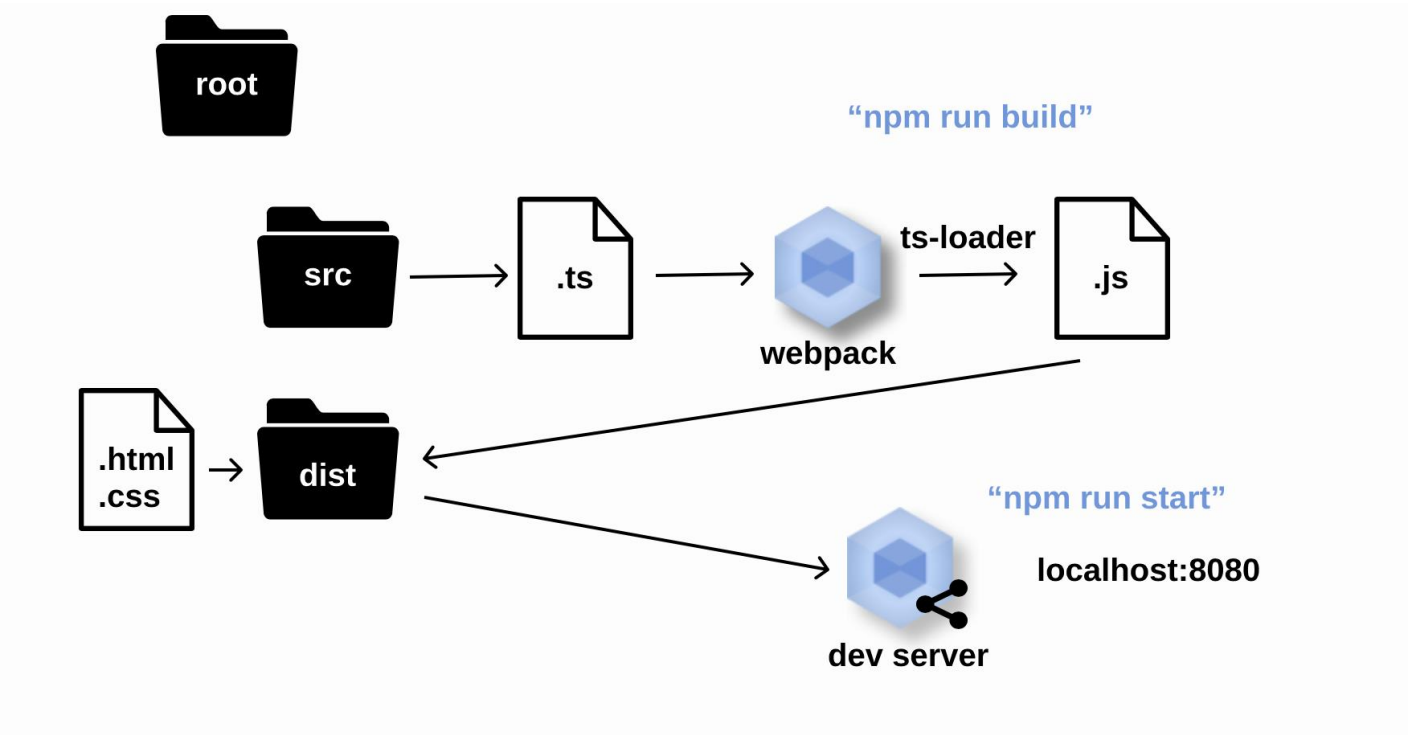
```
///<reference path="./StoreCalc.ts"/>
let invoice = new invoiceCalc.invoiceAccount.Invoice();
console.log("Output: "" +invoice.calculateDiscount(400));
```


demo

PHẦN 2

WEBPACK LÀ GÌ

- Webpack là một công cụ đóng gói giúp compile các module javascript (module bundler) theo cấu trúc project



CÀI ĐẶT WEBPACK

- Bước 1: Cài đặt webpack

```
npm install --save-dev webpack
# or specific version
npm install --save-dev webpack@<version>
```

  - --save
  - --save-dev

- Bước 2: Cài đặt webpack-cli

```
npm install --save-dev webpack-cli
```

CÀI ĐẶT WEBPACK

Bước 2: Cài đặt ts-loader

```
npm install --save-dev typescript ts-loader
```

- ts-loader: trình tải typescript

Bước 3: Chỉnh sửa tsconfig.json

```
{
  "compilerOptions": {
    "outDir": "./dist/",
    "noImplicitAny": true,
    "module": "es6",
    "target": "es5",
    "jsx": "react",
    "allowJs": true
  }
}
```

CÀI ĐẶT WEBPACK

Bước 4: Tạo webpack.config.json

```
const path = require('path');

module.exports = {
  entry: './src/index.ts',
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        use: 'ts-loader',
        exclude: /node_modules/,
      },
    ],
  },
  resolve: {
    extensions: [ '.tsx', '.ts', '.js' ],
  },
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```

Input files

Webpack.config.json

- Entry: cấu hình đầu vào
  - Nhận 3 loại giá trị: string, array, object

- Ví dụ:

```
module.exports = {
  entry: './src/app.ts',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist')
  }
};
```

Ví dụ

```
module.exports = {
    entry: {
        index:'./src/index.ts',
        app:'./src/app.ts',
    },
    output: {
        filename: '[name].bundle.js',
        path: path.resolve(__dirname, 'dist'),
    }
};
```

Webpack.config.json

- Mode: development, **production**, none
- Entry: cấu hình đầu vào
  - Nhận 3 loại giá trị: string, array, object
- Output: nơi lưu kết quả bundle file
  - Phân biệt path và publicPath
- Resolve: nếu tập tin không có thành phần mở rộng (extension), webpack sẽ import tập tin mà không cần khai báo extension


demo

SUMARRY

- ☑ Namespace
- ☑ Module
- ☑ Webpack


thank
you!