Current Skill JS Module

Introduction

One of the most important approaches to software development is **modularity**.

So what does modularity exactly mean? It means that we can split the code of a program into many reusable pieces(usually grouped in files). In software engineering, modularity refers to the extent to which a software/Web application may be divided into smaller modules.

Modularity is successful because developers use prewritten code, which saves resources. Overall, modularity provides greater software development manageability.

That's why ES6 introduced two new keywords: **export** and **import**.

JavaScript types that we can export or import are :

- String
- Number
- Function
- Objects
- Array
- Anything ...

Type

Let's assume we want to use a code that is created in another module, all we have to do is import that module in our work file.

The two types of export are:

- **Named exports:** Zero or more exports per module
- **Default exports:** One per module

Named export

A module can export multiple things by prefixing its declarations with the keyword export. These types of exports are distinguished by their names. In order to import this type, we include the name inside curly brackets. We can also export every module in the file using the * symbol or with an alias using the keyword 'as'.

```
//------ lib.js ------
export function square(x) {
  return x * x;
}
export const sum = (x) => {
  return x + x;
}
//------ main.js ------
import { square, sum } from 'lib' ;
import * as library from 'lib' // this means import every exported module in lib and
```

Default export

Unlike the named export, the default export gives us the right to export only one default module per file.

During the import declaration, we can name the imported module whatever we want as shown in the example below :

```
//------ myFunc.js ------
export default function funcName () { ... };

//------ main.js ------
import myFunc from 'myFunc';
```