JS Modules

Asst.Prof. Dr. Umaporn Supasitthimethee
ผศ.ดร.อุมาพร สุภสิทธิเมธี

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import

https://developer.mozilla.org/en-US/docs/web/javascript/reference/statements/export

JS Modules

- **CommonJS** – **Node.js** supports the CommonJS module format by default. CommonJS modules are characterized by the `require()` for module imports and `module.exports` for module exports
- **ES Modules** – the official standard format to package JavaScript code for reuse and most **modern web browsers** support the ES modules.
- Note that simply enable ES modules in a Node.js package by changing the file extensions from `.js` to `.mjs`

CommonJS in Node.js (1 Function)

```
1 //util.js
2 const displayMessage = (anyMsg) => {
3   console.log(anyMsg)
4 }
5 module.exports = displayMessage
```

```
1 // useModule.js
2 const displayMessage = require('./util.js')
3 displayMessage('Hello, CommonJs Module')
```

CommonJS in Node.js (2 Functions)

```
1 //lib/echo.js
2 function echoMessage(msg) {
3   return msg
4 }
5 function greetingMessage(name) {
6   return `Hi,  ${name}`
7 }
8 module.exports = { echoMessage, greetingMessage }
```

```
1 //useModule.js
2 const { echoMessage, greetingMessage } = require('./lib/echo.js')
3 console.log(echoMessage('Hello, CommonJS'))
4 console.log(greetingMessage('John'))
```

ES Modules Export

- In order to use the export/import declaration in a source file, the file must be interpreted by the runtime as a module. In HTML, this is done by adding `type="module"` to the `<script>` tag, or by being imported by another module.
- The **export** statement is used when creating JavaScript modules to export live bindings to *functions*, *objects*, or *primitive values* from the module so *they can be used by other programs with the **import** statement.*
- There are **two types** of exports:
  - Named Exports (Zero or more exports per module)
  - Default Exports (One per module)

Module Export

```
// Exporting individual features
export let name1, name2, …, nameN // also const
export const name1 = …, name2 = …, …, nameN // also const
export function functionName(){...}
export class ClassName {...}
export const { name1, name2: bar } = o
export const [ name1, name2 ] = array
```

```
// Export list
export { name1, name2, …, nameN }
// Renaming exports
export { variable1 as name1, variable2 as name2, …, nameN }
export { name1 as default /*, … */ }
```

```
// Default exports
export default expression
export default function functionName() {...}
export default function () {...}
export default class {...}
export default class ClassName{...}
```

Named exports are useful when you need to export several values.

When importing this module, named exports must be referred to by the exact same name (optionally renaming it with as),

but the default export can be imported with any name.

https://developer.mozilla.org/en-US/docs/web/javascript/reference/statements/export

Module Import

The import statement cannot be used in embedded scripts unless such script has a type="module".

```
import defaultExport from "module-name"
import { export1 } from "module-name"
import { export1 as alias1 } from "module-name"
import { default as alias } from "module-name"
import { export1 , export2 } from "module-name"
import { export1 , export2 as alias2 , [...] } from "module-name"
import defaultExport, { export1 [ , [...] ] } from "module-name"
```

defaultExportName that will refer to the default export from the module. Must be a valid JavaScript identifier.

module-name The module to import from. This is often a relative or absolute path name to the **.js** file containing the module.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import

ES Module (1 Function)

Way#1

```
1 //util.js
2 export const displayMessage = (anyMsg) => {
3   console.log(anyMsg)
4 }
```

```
1 // useModule.js
2 import { displayMessage } from './util.js'
3 displayMessage('Hello, ES Module')
```

Way#2

```
1 //util.js
2 const displayMessage =(anyMsg)=>{
3   console.log(anyMsg)
4 }
5 export { displayMessage }
```

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7      <title>Document</title>
8    </head>
9    <body>
10     <script type="module" src="useModule.js" />
11   </body>
12 </html>
```

ES Module (2 Functions)

Way#1

```
//lib/echo.js
export function echoMessage(msg) {
  return msg
}
export function greetingMessage(name) {
  return `Hi,  ${name}`
}
```

```
//main.js
import { echoMessage, greetingMessage } from './lib/echo.js'
console.log(echoMessage('Hello, ES Module'))
console.log(greetingMessage('John'))
```

Way#2

```
//lib/echo.js
function echoMessage(msg) {
  return msg
}
function greetingMessage(name) {
  return `Hi,  ${name}`
}
export { echoMessage, greetingMessage }
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script type="module" src="main.js" />
  </body>
</html>
```

ES Module (default export)

Way#1

```
1 //lib/echo.js
2 export default function echoMessage(msg) {
3   return msg
4 }
5 export function greetingMessage(name) {
6   return `Hi,  ${name}`
7 }
```

```
1 //main.js
2 import echoMessage, { greetingMessage } from './lib/echo.js'
3 console.log(echoMessage('Hello, default ES Module'))
4 console.log(greetingMessage('John'))
```

Way#2

```
1 //lib/echo.js
2 function echoMessage(msg) {
3   return msg
4 }
5 function greetingMessage(name) {
6   return `Hi,  ${name}`
7 }
8 export { echoMessage as default, greetingMessage }
```

The default export can be imported with any name

```
1 //main.js
2 import echo, { greetingMessage } from './lib/echo.js'
3 console.log(echo('Hello, default ES Module'))
4 console.log(greetingMessage('John'))
```

Changing `.js` to `.mjs` to enable ES Modules in Node.js

```
1 //util.mjs
2 export const displayMessage = (anyMsg) => {
3   console.log(anyMsg)
4 }
```

```
1 // enable ES modules in a Node.js package
2 // by changing the file extensions from .js to .mjs.
3 // useModule.mjs
4 import { displayMessage } from './util.mjs'
5 displayMessage('Hello, ES Module with .mjs')
```

Another way to enable ES modules in your project by adding a `"type: module"` field inside the nearest `package.json` file (the same folder as the package you're making):

```
{
  "name": "test-module",
  "version": "1.0.0",
  "type": "module",
  "author": "Umaporn Supasitthimethee"
}
```

```
//dataFuncExport.js
//named export
export const frontEndFramework = ['Vuejs', 'React', 'Angular']
//or
const frontEndFramework = ['Vuejs', 'React', 'Angular']
export { frontEndFramework }

export function greeting() {
  return 'Hello, function from another module'
}
//default export
export default function getInstructor() {
  return `Umaporn Supasitthimethee`
}
```

```
<!--index.html -- > //run on browser
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Document</title>

</head>
<body>
  <script src="./main.js" type="module"></script>

</body>
</html>
```

```
//subjectExport.js
const subject = 'INT201'
export {subject}
```

```
//main.js
import defaultExport, {greeting, frontEndFramework as frontEnd} from './dataFuncExport.js'

import {subject} from './subjectExport.js'

console.log(`Frontend Framework: ${frontEnd}`)
console.log(greeting())
console.log(defaultExport)
console.log(subject)
```