

React - ES6 Modules

Topics : [React JS](#)

Written on [January 02, 2024](#)

In React, as well as in modern JavaScript development in general, ES6 Modules are extensively used to organize and structure code. ES6 Modules provide a way to encapsulate and export functionality for use in other parts of your application. Here's a basic overview of how ES6 Modules are used in React:

Exporting from a Module:

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

1. Named Exports:

```
// utils.js
export const add = (a, b) => a + b;
export const subtract = (a, b) => a - b;
```

You can then import these named exports in another file:

```
// App.js
import { add, subtract } from './utils';

console.log(add(5, 3)); // 8
console.log(subtract(10, 4)); // 6
```

2. Default Exports:

```
// utils.js
const multiply = (a, b) => a * b;
export default multiply;
```

You can then import the default export without using curly braces:

```
// App.js
import multiply from './utils';

console.log(multiply(2, 3)); // 6
```

Importing into a Module:

You can import functionality from other modules using the `import` statement.

```
// App.js
import React from 'react'; // Importing a library or module
import MyComponent from './MyComponent'; // Importing a local module
```

Combining Named and Default Exports:

You can also combine named and default exports in a single module.

```
// math.js
export const add = (a, b) => a + b;
export const subtract = (a, b) => a - b;
export default (a, b) => a * b;

// App.js
import multiply, { add, subtract } from './math';

console.log(add(5, 3)); // 8
console.log(subtract(10, 4)); // 6
console.log(multiply(2, 3)); // 6
```

Using Aliases:

You can also use aliases when importing to make your code more readable.

```
// App.js
import { add as addition } from './utils';

console.log(addition(5, 3)); // 8
```

Folder Structure:

For larger React applications, it's common to organize your files into folders and use `index.js` files to create a clear module structure. For example:

```
src/
|-- components/
| |-- Button/
| |-- index.js
| |-- Button.js
|-- utils/
| |-- math.js
|-- App.js
|-- index.js
```

In this structure, you can import a module using its folder path:

```
// App.js
import Button from './components/Button';
import { add } from './utils/math';
```

ARYATECHNO