

ES6 Modules: Import and Export

In this lesson, we cover all the different ways to import and export functionalities from modules in ES6.

In JavaScript ES6, you can import and export functionalities from modules. These can be functions, classes, components, constants, essentially anything you can assign to a variable. Modules can be single files or whole folders with one index file as entry point.

After we bootstrapped our application with *create-react-app* at the beginning, we already used several `import` and `export` statements in the initial files. The `import` and `export` statements help you to share code across multiple files. Historically there were already several solutions for this in the JavaScript environment, but it was a mess because there wasn't a standardized method of performing this task. JavaScript ES6 added it as a native behavior eventually.

These statements embrace code splitting, where we distribute code across multiple files to keep it reusable and maintainable. The former is true because we can import a piece of code into multiple files. The latter is true because there is only one source where you maintain the piece of code.

We also want to think about code encapsulation, since not every functionality needs to be exported from a file. Some of these functionalities should only be used in files where they have been defined. File exports are basically a public API to a file, where only the exported functionalities are available to be reused elsewhere. This follows the best practice of encapsulation.

The following examples showcase the statements by sharing one or multiple variables across two files. In the end, the approach can scale to multiple files and could share more than simple variables.

The act of exporting one or multiple variables is called a named export:

```
// file1
```

```
const firstname = 'Robin';  
const lastname = 'Wieruch';  
  
export { firstname, lastname };
```

And import them in another file with a relative path to the first file.

index.js
file1.js

// file2.js
import { firstname, lastname } from './file1.js';

console.log(firstname);
// output: robin







You can also import all exported variables from another file as one object.

index.js
file1.js

// file2.js
import * as person from './file1.js';

console.log(person.firstname);
// output: robin



Imports can have aliases, which are necessary when we import functionalities from multiple files that have the same named export.

index.js
file1.js

// file2.js
import { firstname as username } from './file1.js';

console.log(username);
// output: Robin



There is also the **default** statement, which can be used for a few cases:

- to export and import a single functionality
- to highlight the main functionality of the exported API of a module

- to have a fallback import functionality

```
// file1.js
const robin = {
  firstname: 'robin',
  lastname: 'wieruch',
};

export default robin;
```

You have to leave out the curly braces to import the default export.

index.js

file1.js

```
// file2.js
import developer from './file1.js';

console.log(developer);
// output: { firstname: 'robin', lastname: 'wieruch' }
```



The import name can differ from the exported default name, and it can be used with the named export and import statements:

```
// file1.js
const firstname = 'robin';
const lastname = 'wieruch';

const person = {
  firstname,
  lastname,
};

export {
  firstname,
  lastname,
};

export default person;
```

Import the default or the named exports in another file:

index.js

file1.js

```
// file2.js
import developer, { firstname, lastname } from './file1.js';
```

```
import developer, { firstname, lastname } from './file1.js';

console.log(developer);

// output: { firstname: 'robin', lastname: 'wieruch' }
console.log(firstname, lastname);
// output: robin wieruch
```



You can spare the extra lines, and export the variables directly for named exports as in `file1.js`.

index.js

file1.js

```
import { firstname, lastname } from './file1.js';

console.log(firstname);
console.log(lastname);
```



These are the main functionalities for ES6 modules. They help you to organize your code, to maintain it, and to design reusable module APIs. You can also export and import functionalities to test them which you will do in a later chapter.

Further Reading:

- Read about [ES6 import](#)
- Read about [ES6 export](#)