

ChatGPT

Share

AB

give me an overview of ES6 import and export functionalities with example

In ES6 (ECMAScript 2015), **import** and **export** are used for **modular JavaScript**, allowing code to be split into reusable pieces across multiple files. This modular structure makes your code easier to maintain and test.

Here's an overview of the functionalities with examples:

1. Named Export and Import

You can export multiple functions, variables, or objects from a file and import only the ones you need.

**Exporting:**

```
javascriptCopy code// mathUtils.jsexport const add = (a, b) => a + b;export const subtract = (a, b) => a - b;export const PI = 3.14159;
```

**Importing:**

```
javascriptCopy code// main.jsimport { add, PI } from './mathUtils.js';console.log(add(5, 3)); // 8console.log(PI); // 3.14159
```

Note: You must use the same names while importing.

2. Default Export and Import

A file can have one **default export**, which can be imported without curly braces and renamed on import.

**Exporting:**

```
javascriptCopy code// greeting.jsconst greet = (name) => `Hello, ${name}!`;export default greet;
```

**Importing:**

```
javascriptCopy code// main.jsimport greet from './greeting.js';console.log(greet('Nahian')); // Hello, Nahian!
```

Default exports are useful when a module exports a single primary functionality.

3. Combining Named and Default Exports

You can mix default and named exports in the same file.

**Exporting:**

```
javascriptCopy code// shapes.jsexport const square = (x) => x * x;export const circleArea = (radius) => Math.PI * radius * radius;const shapesInfo = "Basic shape utilities";export default shapesInfo;
```

**Importing:**

```
javascriptCopy code// main.jsimport shapesInfo, { square, circleArea } from './shapes.js';console.log(shapesInfo); // Basic shape utilitiesconsole.log(square(4)); // 16console.log(circleArea(3)); // 28.274333882308138
```

4. Renaming Exports and Imports

You can rename exports and imports using the **as** keyword.

**Exporting:**

```
javascriptCopy code// data.jsexport const fetchData = () => { /* fetching logic */ };export const saveData = () => { /* saving logic */ };
```

**Importing:**

```
javascriptCopy code// main.jsimport { fetchData as getData, saveData as storeData } from './data.js';getData();storeData();
```

5. Import All (\* as)

You can import all named exports from a module as an object.

**Exporting:**

```
javascriptCopy code// utilities.jsexport const logInfo = (message) => console.log(`Info: ${message}`);export const logError = (error) => console.error(`Error: ${error}`);
```

**Importing:**

```
javascriptCopy code// main.jsimport * as utils from './utilities.js';utils.logInfo('Everything is running smoothly.');
```

6. Re-exporting (Export from Another Module)

You can re-export modules without explicitly importing them into the current file.

**Re-exporting:**

```
javascriptCopy code// allMath.jsexport { add, subtract } from './mathUtils.js';export { square } from './shapes.js';
```

**Importing from re-exported module:**

```
javascriptCopy code// main.jsimport { add, square } from './allMath.js';console.log(add(2, 3)); // 5console.log(square(3)); // 9
```

7. Dynamic Imports (Lazy Loading)

Dynamic `import()` lets you load modules on demand, improving performance by splitting the code into chunks.

**Using Dynamic Import:**

```
javascriptCopy code// main.jsconst loadModule = async () => {const module = await import('./mathUtils.js');console.log(module.add(5, 6)); // 11};loadModule();
```

Useful for code splitting and lazy loading in modern applications.

Summary Table:

Functionality	Export Syntax	Import Syntax
Named Export	<code>export const name = value;</code>	<code>import { name } from './file';</code>
Default Export	<code>export default value;</code>	<code>import name from './file';</code>
Renaming Imports/Exports	<code>export { name as alias };</code>	<code>import { name as alias } ...;</code>
Import All	-	<code>import * as obj from './file';</code>
Re-exporting	<code>export { name } from './file';</code>	-
Dynamic Import	-	<code>import('./file.js')</code>

These features together make ES6 modules extremely flexible for modern web development.

<4/4>

Message ChatGPT

ChatGPT can make mistakes. Check important info.