📦 Exports in JavaScript Modules

✅ Named Export

You can export **multiple values** by name. You must use **the same name** when importing.

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

✅ Default Export

You can export **only one default value** per file. You can **rename** it while importing.

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

🔗 Now let's put it all together:

🔧 `utils.js` – Exporting

```
// utils.js

// Named exports
export const add = (a, b) => a + b;
export const subtract = (a, b) => a - b;

// Default export
const multiply = (a, b) => a * b;
export default multiply;
```

💻 `app.js` – Importing

```
// app.js

// Import default export (you can name it anything)
import multiply from './utils.js';

// Import named exports (must use exact names)
import { add, subtract } from './utils.js';
```

```
console.log("Add:", add(5, 3));          // 8
console.log("Subtract:", subtract(5, 3)); // 2
console.log("Multiply:", multiply(5, 3)); // 15
```

🧠 Analogy 🎓

> Think of a file like a **restaurant**:
>
> - 🍝 **Default export** is the **signature dish** – one per restaurant.
> - 🍽 **Named exports** are other **dishes on the menu** – order them by name.

✅ Summary Table:

Feature	Named Export	Default Export
Export Syntax	`export const foo = ...`	`export default foo`
Import Syntax	`import { foo } from ...`	`import foo from ...`
Number per file	Many	Only one
Import Rename?	❌ Use exact name	✅ Can rename on import

🔄 Can We Use Both Named and Default Exports in the Same File?

✅ **Yes, we can!**

You can use **both** `named exports` and a `default export` in the **same file** in JavaScript/React. 💡 It's valid and very commonly used in real-world projects! 🛠

💡 Let's First Understand the Difference

1️⃣ **Named Export**

- Export **multiple things** from a file
- Must use **curly braces** `{}` when importing
- Must use the **exact same name** while importing

```
// file: utils.js
export const add = (a, b) => a + b;
export const multiply = (a, b) => a * b;
```

```
// file: app.js
import { add, multiply } from './utils';
```

2️⃣ Default Export

- Only **one default export per file**
- No curly braces needed
- Can **rename** while importing

```
// file: utils.js
export default function divide(a, b) {
  return a / b;
}
```

```
// file: app.js
import divide from './utils';
```

🔁 Using **Both Together** 🧠

```
// file: utils.js
export const add = (a, b) => a + b;
export const subtract = (a, b) => a - b;
export default function divide(a, b) {
  return a / b;
}
```

✅ This is **valid** — 2 named exports (`add`, `subtract`) + 1 default export (`divide`).

📦 Importing Both in Another File:

```
// file: app.js
import divide, { add, subtract } from './utils';

console.log(add(5, 3));        // ➡️ 8
console.log(subtract(5, 3));   // ➡️ 2
console.log(divide(6, 2));     // ➡️ 3
```

- `divide` comes as **default**
- `{ add, subtract }` come as **named**

⚠️ Gotchas to Remember

🚫 You **cannot have multiple default exports** in a single file.

```
export default function a() {}   // ✅
export default function b() {}   // ❌ Error
```

✅ You **can mix** default and named, but follow **consistent project structure** for clarity.

🧑‍🍳 Real-life Analogy

Imagine a restaurant menu:

- 🍕 **Default dish**: Chef's special (default export — one per day)
- 🍟 **Other dishes**: Side items (named exports — as many as you like)

You order like this:

```
import chefsSpecial, { fries, coke } from './menu';
```

✅ Summary

Feature	Named Export	Default Export
Export multiple?	✅ Yes	❌ Only one
Curly braces needed?	✅ Yes	❌ No
Can rename on import?	❌ No	✅ Yes
Use both in same file?	✅ Yes	✅ Yes

🔥 Pro Tip

In React, we often:

```
// file: Header.js
export const Logo = () => <h1>🍔</h1>;
export const Nav = () => <nav>Home | About</nav>;
export default function Header() {
  return (
    <div>
      <Logo />
      <Nav />
    </div>
  );
}
```

```
// file: App.js
import Header, { Logo, Nav } from './Header';
```

📦 JavaScript Export & Import Patterns

✳️ Named Exports | **★ Default Export** | **🔄 Mix Both**

📄 What is Export & Import in JS?

JS Modules help you split code across files. You can **export** and **import**:

- Specific pieces 🧩 → using **named exports**
- A single default 🪄 → using **default export**
- Or even both 🔄 in one file!

◇ 1. 📦 **Named Exports** – Export Multiple Values

🗁 `main.js`

```
// ☑ NAMED EXPORTS
const add = (a, b) => a + b;
const subtract = (a, b) => a - b;
const multiply = (a, b) => a * b;

// ☑ Export all functions by name (object-style)
export { add, subtract, multiply };
```

> ☑ You can export multiple things using `export {}`

📥 Importing Named Exports

```
// 📥 FROM main.js
import { add, subtract, multiply } from './main.js';

console.log(add(2, 3));         // 5
console.log(subtract(9, 4));    // 5
console.log(multiply(3, 3));    // 9
```

> ☑ Use **curly braces** `{}` 🧠 Names must match exactly (case-sensitive)

⚙ Alternative: Export inline

```
export const divide = (a, b) => a / b;
export const power = (a, b) => a ** b;
```

▨ Real Use Case Example (Your Code)

```
// 🗁 main.js

// ☑ Define
const add = (a, b) => a + b;
const subtract = (a, b) => a - b;
const multiply = (a, b) => a * b;

// ☑ Export
export { add, subtract, multiply };

// 🗁 any other file
import { add, subtract, multiply } from './main.js';

console.log(add(4, 6));         // 10
console.log(subtract(9, 3));    // 6
console.log(multiply(7, 5));    // 35
```

◇ 2. ★ **Default Export** – Only One per File

```
// 🗁 math.js

const divide = (a, b) => a / b;

// ★ Export as default
export default divide;
```

```
// 📥 Import
import divide from './math.js';

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

> ☑ No `{}` needed ☑ You can rename it during import

🔁 3. Using **Named + Default Export Together**

```
// 🗁 utils.js

export const add = (a, b) => a + b;
export const sub = (a, b) => a - b;
export default function multiply(a, b) {
  return a * b;
}
```

```
// 📥 Import mix
import multiply, { add, sub } from './utils.js';

multiply(2, 2); // 4
add(2, 3);      // 5
```

🧠 Summary Table

Feature	**Named Export**	**Default Export**
How many per file?	☑ Multiple	✗ One only
Curly braces on import?	☑ Yes (`{}`)	✗ No
Import name flexibility	⚠ Must match name	☑ Any name allowed
Export Syntax	`export {}` or inline	`export default`
Import Syntax	`import {}`	`import name`

🧠 Analogy Time 🍽

Concept	**Example**	**Analogy**
Named Export	`add`, `sub`	📋 Menu items
Default Export	`multiply`	🍽 Chef's special of the day

📥 Named Import in JavaScript Modules

```
// 🗁 app.js

// ➕ Importing Named Exports
import { add, subtract, multiply } from './mathUtils.js';
```

```
console.log(add(1, 2));        // ✅ Works! Output: 3
console.log(subtract(5, 3));  // ✅ Output: 2
```

🔍 Important Rules:

- ✅ Use **curly braces** `{}` when importing **named exports**
- ⚠️ The import names **must exactly match** the names used during export *(case-sensitive!)*
- 🚫 You **cannot rename** directly like `import sum from` unless you use `as`

✏️ Rename Named Import using `as`

```
// Rename `add` to `sum` during import
import { add as sum } from './mathUtils.js';

console.log(sum(4, 6)); // ✅ Works! Output: 10
```

> 🔄 `as` lets you alias the function name while importing 🧠 Great for avoiding name clashes or improving readability

---