React Lecture – 2

Importing and Exporting Components and JSX

Importing and Exporting Components

Type of Exports

There are two primary ways to export values with JavaScript:

1. Default exports
2. Named Exports

Default Exports

To do a default export just write **export default** before the function name

To import a default exported component you have follow this syntax
Ex: You have a file Gallery.jsx which have a component named Gallery

import Gallery from ‘./Gallery.jsx’

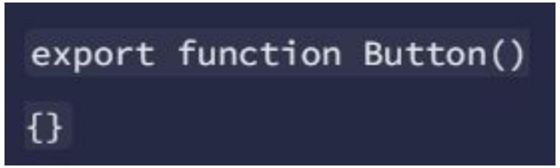
A file can have no more than one default export

When you write a default import, you can put any name you want after import. For example, you could write **import Banana from './Gallery.jsx'** instead and it would still provide you with the same default export

Named Exports

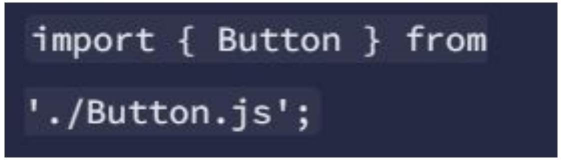
In named exports we don't write **default** after export

Example:



```
export function Button()
{}
```

To import a named export component we need to use the following syntax:



```
import { Button } from
'./Button.js';
```

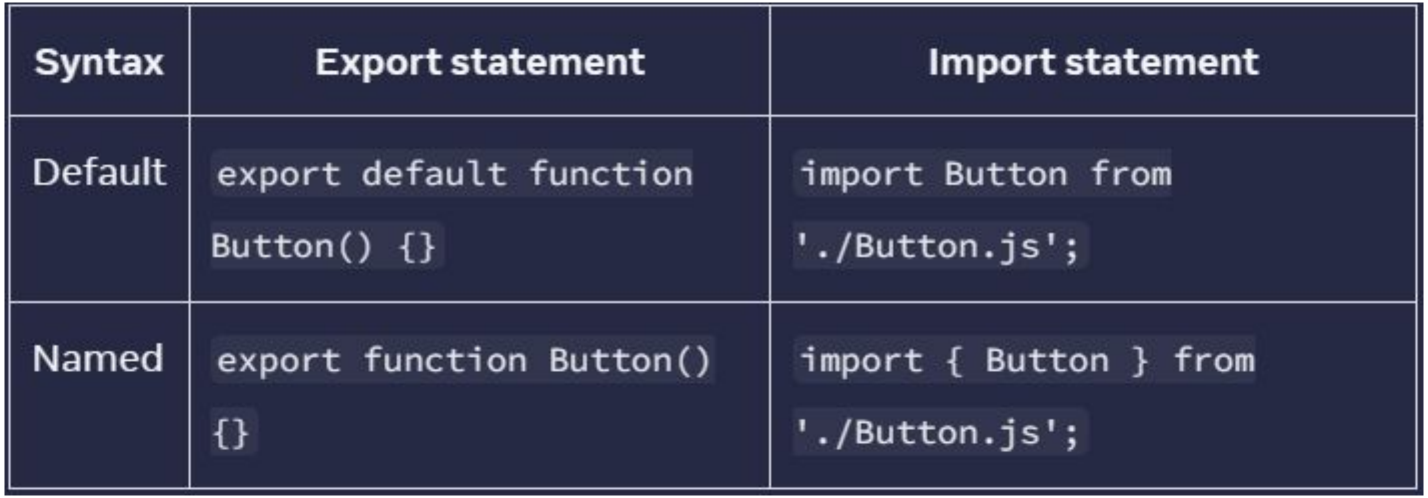
In contrast, with named imports, the name has to match on both sides. That's why they are called named imports!

Mixed Exports

You can mix different exports in a file.
Example you can have multiple named exports and a default export in a single file.

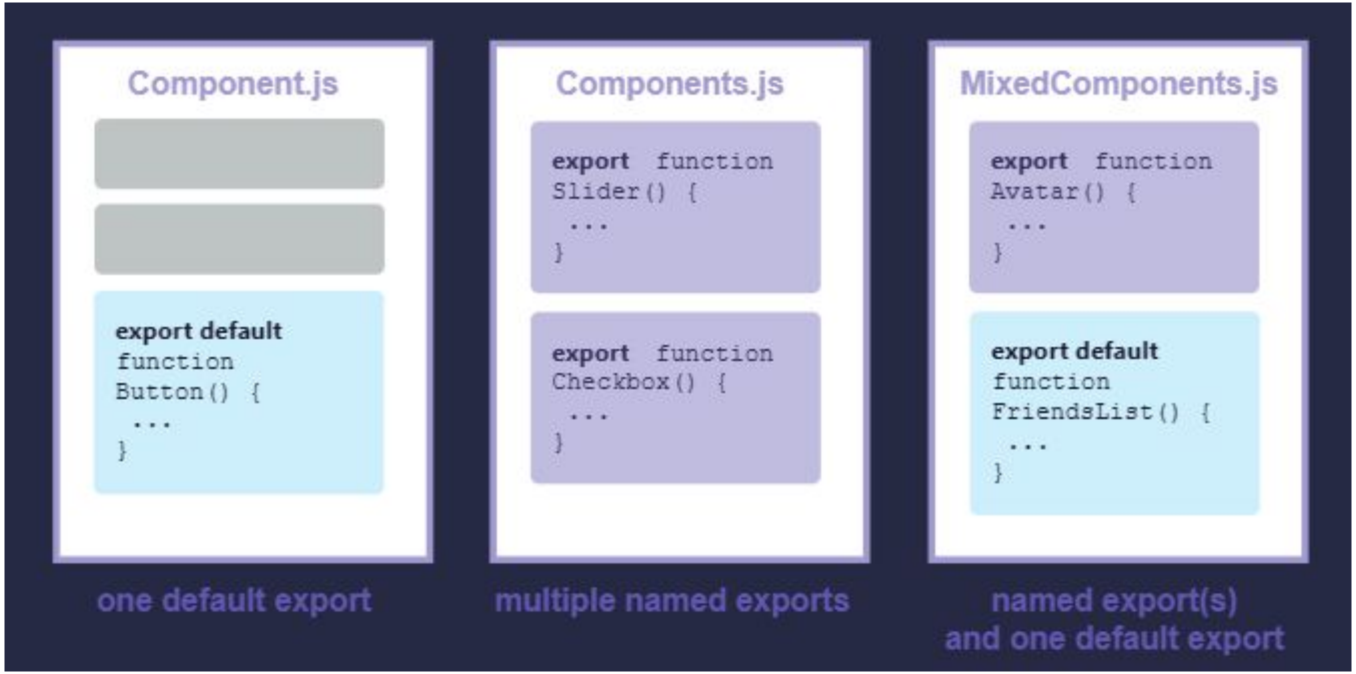
To import you have to use the respective syntax.

Exports Summary



Syntax	Export statement	Import statement
Default	`export default function Button() {}`	`import Button from './Button.js';`
Named	`export function Button() {}`	`import { Button } from './Button.js';`

Exports Summary



```
export default
function
Button() {
  ...
}
```

```
export function
Slider() {
  ...
}
```

```
export function
Checkbox() {
  ...
}
```

```
export function
Avatar() {
  ...
}
```

```
export default
function
FriendsList() {
  ...
}
```

JSX

What is JSX

JSX is a syntax extension for JavaScript that lets you write HTML-like markup inside a JavaScript file. It’s not required to use JSX when writing React components, but it does make writing them more concise.

Rules for JSX

If you were to take some valid HTML and copy it straight into your React component, it would not work. This is due to some of the rules JSX implements, that aren't present in HTML.

1. **Return a single root element**

   If you wish to return multiple elements in a component, you can do so by wrapping them in a parent tag. This can be a <div>, or, if you don't want the elements to have a container, you could use a React fragment

Rules for JSX

2. **Close all tags**

In HTML, many tags are self-closing and self-wrapping. In JSX however, we must explicitly close and wrap these tags.

<input> would become <input />, and <li> would become <li></li>

Rules for JSX

3. **CamelCase** most of the things

JSX turns into JavaScript, and attributes of elements become keys of JavaScript objects, so you can’t use dashes or reserved words such as class. Because of this, many HTML attributes are written in camelCase. Instead of stroke-width, you’d use strokeWidth, and instead of class you’d use className