

Overview

The root component file

Exporting and importing a component

Exporting and importing multiple components from the same file

Recap

Challenges

Importing and Exporting Components

The magic of components lies in their reusability; you can create components that are composed of other components. But as you nest more and more components, it often makes sense to start splitting them into different files. This lets you keep your files easy to scan and reuse components in more places.

You will learn

- What a root component file is
- How to import and export a component
- When to use default and named imports and exports
- How to import and export multiple components from one file
- How to split components into multiple files

The root component file

In [Your First Component](#), you made a `Profile` component and a `Gallery` component that renders it:

App.js


Download

Reset

Fork

```
1 function Profile() {
2   return (
3     
7   );
8 }
9
10 export default function Gallery() {
```

Amazing scientists



```

9
10 export default function Gallery() {
11   return (
12     <section>
13       <h1>Amazing scientists</h1>
14       <Profile />
15       <Profile />
16       <Profile />
17     </section>
18   );
19 }
20

```

⬆ Show less

Amazing scientists



These currently live in a **root component file**, named `App.js` in this example. Depending on your setup, your root component could be in another file, though. If you use a framework with file-based routing, such as Next.js, your root component will be different for every page.

Exporting and importing a component

What if you want to change the landing screen in the future and put a list of science books there? Or place all the profiles somewhere else? It makes sense to move `Gallery` and `Profile` out of the root component file. This will make them more modular and reusable in other files. You can move a component in three steps:

1. **Make** a new JS file to put the components in.
2. **Export** your function component from that file (using either [default](#) or [named](#) exports).
3. **Import** it in the file where you'll use the component (using the corresponding technique for importing [default](#) or [named](#) exports).

Here both `Profile` and `Gallery` have been moved out of `App.js` into a new file called `Gallery.js`. Now you can change `App.js` to import `Gallery` from `Gallery.js`:

```
1 function Profile() {  
2   return (  
3       
7   );  
8 }  
9  
10 export default function Gallery() {  
11   return (  
12     <section>  
13       <h1>Amazing scientists</h1>  
14       <Profile />  
15       <Profile />  
16       <Profile />  
17     </section>  
18   );  
19 }  
20
```

Amazing scientists



App.js Gallery.js

Reset Fork

```
1 import Gallery from './Gallery.js';
2
3 export default function App() {
4   return (
5     <Gallery />
6   );
7 }
8
```

Amazing scientists



Notice how this example is broken down into two component files now:

1. Gallery.js:

- Defines the `Profile` component which is only used within the same file and is not exported.
- Exports the `Gallery` component as a **default export**.

2. App.js:

- Imports `Gallery` as a **default import** from `Gallery.js`.
- Exports the root `App` component as a **default export**.

Note

You may encounter files that leave off the `.js` file extension like so:

```
import Gallery from './Gallery';
```

Either `./Gallery.js` or `./Gallery` will work with React, though the former is closer to how [native ES Modules](#) work.

You may encounter files that leave off the `.js` file extension like so:

```
import Gallery from './Gallery';
```

Either `./Gallery.js` or `./Gallery` will work with React, though the former is closer to how [native ES Modules](#) work.

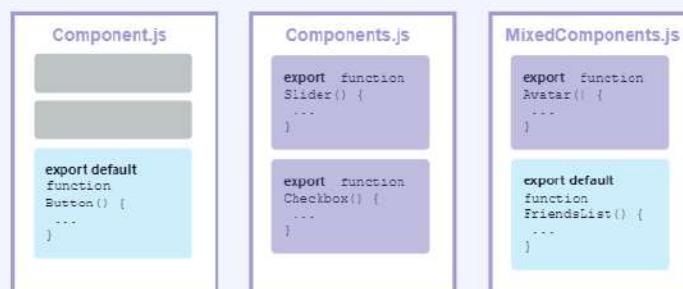
DEEP DIVE

Default vs named exports

^ Hide Details

There are two primary ways to export values with JavaScript: default exports and named exports. So far, our examples have only used default exports. But you can use one or both of them in the same file.

A file can have no more than one *default* export, but it can have as many *named* exports as you like.



one default export

multiple named exports

named export(s)
and one default export

How you export your component dictates how you must import it. You will get an error if you try to import a default export the same way you would a named export! This chart can help you keep track:

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

When you write a *default* import, you can put any name you want after `import`. For example, you could write `import Banana from './Button.js'` instead and it would still provide you with the same default export. In contrast, with named imports, the name has to match on both sides. That's why they are called *named* imports!

People often use default exports if the file exports only one component, and use named exports if it exports multiple components and values. Regardless of which coding style you prefer, always give meaningful names to your component functions and the files that contain them. Components without names, like `export default () => {}`, are discouraged because they make debugging harder.

Exporting and importing multiple components from the same file

What if you want to show just one `Profile` instead of a gallery? You can export the `Profile` component, too. But `Gallery.js` already has a *default* export, and you can't have two default exports. You could create a new file with a default export, or you could add a *named* export for `Profile`. **A file can only have one default export, but it can have numerous named exports!**

 **Note**

To reduce the potential confusion between default and named exports, some teams choose to only stick to one style (default or named), or avoid mixing them in a single file. Do what works best for you!

First, **export** `Profile` from `Gallery.js` using a named export (no `default` keyword):

```
export function Profile() {  
  // ...  
}
```

Then, **import** `Profile` from `Gallery.js` to `App.js` using a named import (with the curly braces):

```
import { Profile } from './Gallery.js';
```

Finally, **render** `<Profile />` from the `App` component:

```
export default function App() {  
  return <Profile />;  
}
```

Now `Gallery.js` contains two exports: a default `Gallery` export, and a named `Profile` export. `App.js` imports both of them. Try editing `<Profile />` to `<Gallery />` and back in this example:

App.js Gallery.js


Reset Fork

```
1 import Gallery from './Gallery.js';  
2 import { Profile } from './Gallery.js';  
3  
4 export default function App() {  
5   return (  
    <Gallery />  
  );  
}
```



App.js Gallery.js Reset Fork

```
1 import Gallery from './Gallery.js';
2 import { Profile } from './Gallery.js';
3
4 export default function App() {
5   return (
6     <Profile />
7   );
8 }
9
```



Now you're using a mix of default and named exports:

- `Gallery.js`:
 - Exports the `Profile` component as a **named export** called `Profile`.
 - Exports the `Gallery` component as a **default export**.
- `App.js`:
 - Imports `Profile` as a **named import** called `Profile` from `Gallery.js`.
 - Imports `Gallery` as a **default import** from `Gallery.js`.
 - Exports the root `App` component as a **default export**.


Recap

On this page you learned:

- What a root component file is
- How to import and export a component
- When and how to use default and named imports and exports
- How to export multiple components from the same file

App.js Gallery.js ↺ Reset 🔗 Fork

```
1 export function Profile() {
2   return (
3     
7   );
8 }
9
10 export default function Gallery() {
11   return (
12     <section>
13       <h1>Amazing scientists</h1>
14       <Profile />
15       <Profile />
16       <Profile />
17     </section>
18   );
19 }
20
```



^ Show less

Now you're using a mix of default and named exports:

- `Gallery.js`:
 - Exports the `Profile` component as a **named export called `Profile`**.
 - Exports the `Gallery` component as a **default export**.
- `App.js`:
 - Imports `Profile` as a **named import called `Profile`** from `Gallery.js`.
 - Imports `Gallery` as a **default import** from `Gallery.js`.
 - Exports the root `App` component as a **default export**.

Try out some challenges

Challenge 1 of 1: Split the components further

Currently, `Gallery.js` exports both `Profile` and `Gallery`, which is a bit confusing.

Move the `Profile` component to its own `Profile.js`, and then change the `App` component to render both `<Profile />` and `<Gallery />` one after another.

You may use either a default or a named export for `Profile`, but make sure that you use the corresponding import syntax in both `App.js` and `Gallery.js`! You can refer to the table from the deep dive above:

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

App.js Gallery.js Profile.js Reset Fork

```
1 // Move me to Profile.js!
2 export function Profile() {
3   return (
4     
8   );
9 }
10
11 export default function Gallery() {
12   return (
13     <section>
14       <h1>Amazing scientists</h1>
15       <Profile />
16       <Profile />
```

App.js Gallery.js Profile.js

↺ Reset ↗ Fork

```
1 // Move me to Profile.js!
2 export function Profile() {
3   return (
4     
8   );
9 }
10
11 export default function Gallery() {
12   return (
13     <section>
14       <h1>Amazing scientists</h1>
15       <Profile />
16       <Profile />
17       <Profile />
18     </section>
19   );
20 }
21
```



^ Show less

After you get it working with one kind of exports, make it work with the other kind.

💡 Show hint

🔍 Show solution

App.js Gallery.js Profile.js

Reset Fork

```
1 import Gallery from './Gallery.js';
2 import { Profile } from './Gallery.js';
3
4 export default function App() {
5   return (
6     <div>
7       <Profile />
8     </div>
9   );
10 }
11
```



^ Show less

[App.js](#) [Gallery.js](#) [Profile.js](#)

[↺ Reset](#) [↗ Fork](#)

1



⤴ Show less

Solution

This is the solution with named exports:

App.js Gallery.js Profile.js

Reset Fork

```
1 import Gallery from './Gallery.js';
2 import { Profile } from './Profile.js';
3
4 export default function App() {
5   return (
6     <div>
7       <Profile />
8       <Gallery />
9     </div>
10  );
11 }
12
```



Amazing scientists



This is the solution with default exports:

App.js Gallery.js Profile.js

Reset Fork

```
1 import Gallery from './Gallery.js';
2 import Profile from './Profile.js';
3
4 export default function App() {
5   return (
6     <div>
7       <Profile />
8       <Gallery />
9     </div>
10  );
11 }
12
```



Amazing scientists



Solution

This is the solution with named exports:

App.js Gallery.js Profile.js

↺ Reset ↗ Fork

```
1 import { Profile } from './Profile.js';
2
3 export default function Gallery() {
4   return (
5     <section>
6       <h1>Amazing scientists</h1>
7       <Profile />
8       <Profile />
9       <Profile />
10    </section>
11  );
12 }
13
```



Amazing scientists



This is the solution with default exports:

App.js Gallery.js Profile.js

↺ Reset ↗ Fork

```
1 import Profile from './Profile.js';
2
3 export default function Gallery() {
4   return (
5     <section>
6       <h1>Amazing scientists</h1>
7       <Profile />
8       <Profile />
9       <Profile />
10    </section>
11  );
12 }
13
```



Amazing scientists



Solution

This is the solution with named exports:

App.js Gallery.js Profile.js

Reset Fork

```
1 export function Profile() {  
2   return (  
3       
7   );  
8 }  
9
```



Amazing scientists



This is the solution with default exports:

App.js Gallery.js Profile.js

Reset Fork

```
1 export default function Profile() {  
2   return (  
3       
7   );  
8 }  
9
```



Amazing scientists

