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

```
function Profile() {
  return (
    <img
      src="https://i.imgur.com/MK3eW3As.jpg"
```

```
}

export default function Gallery() {
  return (
    <section>
      <h1>Amazing scientists</h1>
      <Profile />
      <Profile />
      <Profile />
```

Show more

These currently live in a **root component file,** named `App.js` in this example. In Create React App, your app lives in `src/App.js`. 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).

`Gallery.js`:

App.js Gallery.js Reset Fork

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


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

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';
```

DEEP DIVE

Default vs Named Exports

Show Details

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!**

> 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. It's a matter of preference. Do what works best for you!

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

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

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

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

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

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

- `Gallery.js`:
  - Exports the `Profile` component as a **named export called** `Profile`.

- 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

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	`export default function Button() {}`	`import Button from './button.js';`

Named	`export function Button() {}`	`import { Button } from './button.js';`

App.js Gallery.js Profile.js Reset Fork

```
// Move me to Profile.js!
export function Profile() {
  return (
    <img
      src="https://i.imgur.com/QIrZWGIs.jpg"
      alt="Alan L. Hart"
    />
  );
}

export default function Gallery() {
  return (
    <section>
      <h1>Amazing scientists</h1>
      <Profile />
      <Profile />
```

Show more

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

Show hint Show solution

PREVIOUS
Your First Component

NEXT

FACEBOOK

Open Source

©2021

Learn React

Quick Start

Installation

Describing the UI

Adding Interactivity

Managing State

Escape Hatches

API Reference

React APIs

React DOM APIs

Community

Code of Conduct

Acknowledgements

Meet the Team

More

React Native

Privacy

Terms