**Your First Component**

*Components* are one of the core concepts of React. They are the foundation upon which you build user interfaces (UI), which makes them the perfect place to start your React journey!

! You can even jumpstart your project with the thousands of components shared by the React open source community like [Chakra UI](https://chakra-ui.com/) and [Material UI.](https://material-ui.com/)

**A React component is a JavaScript function that you can sprinkle with markup**

**Step 1: Export the component**

The export default prefix is a [standard JavaScript syntax](https://developer.mozilla.org/docs/web/javascript/reference/statements/export) (not specific to React). It lets you mark the main function in a file so that you can later import it from other files. (More on importing in [Importing and Exporting Components](https://react.dev/learn/importing-and-exporting-components)!)

### Step 2: Define the function

With function Profile() { } you define a JavaScript function with the name Profile.

React components are regular JavaScript functions, but **their names must start with a capital letter** or they won’t work!

### Step 3: Add markup

The component returns an <img /> tag with src and alt attributes. <img /> is written like HTML, but it is actually JavaScript under the hood! This syntax is called [JSX](https://react.dev/learn/writing-markup-with-jsx), and it lets you embed markup inside JavaScript.

But if your markup isn’t all on the same line as the return keyword, you must wrap it in a pair of parentheses:

return <img src="https://i.imgur.com/MK3eW3As.jpg" alt="Katherine Johnson" />;

return (

<div>

<img src="https://i.imgur.com/MK3eW3As.jpg" alt="Katherine Johnson" />

</div>

);

Without parentheses, any code on the lines after return [will be ignored](https://stackoverflow.com/questions/2846283/what-are-the-rules-for-javascripts-automatic-semicolon-insertion-asi)!

## Using a component

Now that you’ve defined your Profile component, you can nest it inside other components. For example, you can export a Gallery component that uses multiple Profile components:

Components can render other components, but **you must never nest their definitions:**

* React lets you create components, **reusable UI elements for your app.**
* In a React app, every piece of UI is a component.
* React components are regular JavaScript functions except:
  1. Their names always begin with a capital letter.
  2. They return JSX markup.

# 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.
* **Make** a new JS file to put the components in.
* **Export** your function component from that file (using either [default](https://developer.mozilla.org/docs/Web/JavaScript/Reference/Statements/export#using_the_default_export) or [named](https://developer.mozilla.org/docs/Web/JavaScript/Reference/Statements/export#using_named_exports) exports).
* **Import** it in the file where you’ll use the component (using the corresponding technique for importing [default](https://developer.mozilla.org/docs/Web/JavaScript/Reference/Statements/import#importing_defaults) or [named](https://developer.mozilla.org/docs/Web/JavaScript/Reference/Statements/import#import_a_single_export_from_a_module) exports).
* **A file can only have one default export, but it can have numerous named exports!**

# Writing Markup with JSX

JSX is a syntax extension for JavaScript that lets you write HTML-like markup inside a JavaScript file. Although there are other ways to write components, most React developers prefer the conciseness of JSX, and most codebases use it.

JSX and React are two separate things. They’re often used together, but you can [use them independently](https://reactjs.org/blog/2020/09/22/introducing-the-new-jsx-transform.html#whats-a-jsx-transform) of each other. JSX is a syntax extension, while React is a JavaScript library.

## The Rules of JSX

1.To return multiple elements from a component, **wrap them with a single parent tag.** This empty tag is called a [*Fragment.*](https://react.dev/reference/react/Fragment) Fragments let you group things without leaving any trace in the browser HTML tree.

**2.** JSX requires tags to be explicitly closed: self-closing tags like <img> must become <img />, and wrapping tags like <li>oranges must be written as <li>oranges</li>.

3. camelCase ~~all~~ most of the things!

### Where to use curly braces

1. **As text** directly inside a JSX tag: <h1>{name}'s To Do List</h1> works, but <{tag}>Gregorio Y. Zara's To Do List</{tag}> will not.
2. **As attributes** immediately following the = sign: src={avatar} will read the avatar variable, but src="{avatar}" will pass the string "{avatar}".

## Using “double curlies”: CSS and other objects in JSX

**<ul style={{ backgroundColor: 'black', color: 'pink' }}> </ul>**

Inline style properties are written in camelCase. For example, HTML <ul style="background-color: black"> would be written as <ul style={{ backgroundColor: 'black' }}> in your component.

React components use props to communicate with each other. Every parent component can pass some information to its child components by giving them props. Props might remind you of HTML attributes, but you can pass any JavaScript value through them, including objects, arrays, and functions.

### Step 1: Pass props to the child component

If double curly braces after person= confuse you, recall [they’re merely an object](https://react.dev/learn/javascript-in-jsx-with-curly-braces#using-double-curlies-css-and-other-objects-in-jsx) inside the JSX curlies

export default function Profile() {

return (

<Avatar person={{ name: 'Lin Lanying', imageId: '1bX5QH6' }} size={100} />

);}

### Step 2: Read props inside the child component

function Avatar({ person, size }) { }

**Don’t miss the pair of { and } curlies** inside of ( and ) when declaring props:

The default value is only used if the size prop is missing or if you pass size={undefined}. But if you pass size={null} or size={0}, the default value will **not** be used.

**Use spread syntax with restraint.** If you’re using it in every other component, something is wrong. Often, it indicates that you should split your components and pass children as JSX