# Componentizing Our React App

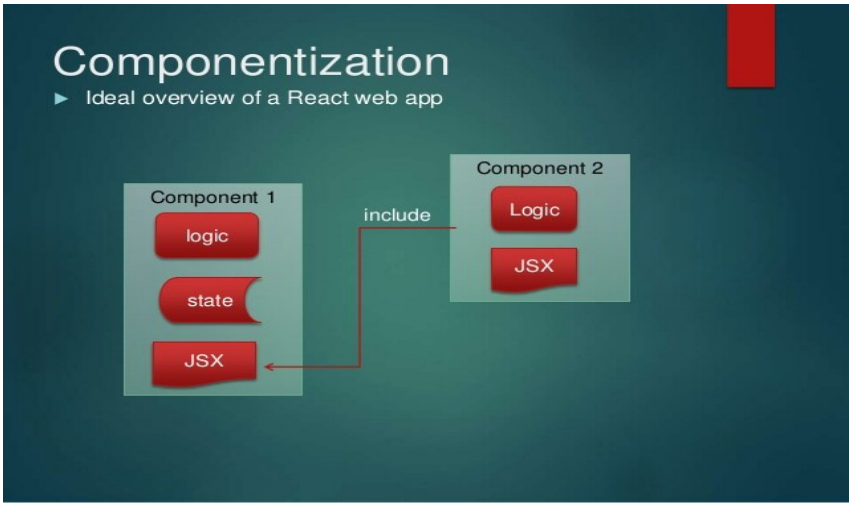
At this point, our app is a **monolith**. Before we can make it do things, we need to break it apart into **manageable**, **descriptive components**. React doesn't have any hard rules for what is and isn't a component – that's up to you! In this Note, we will show you a sensible way to break our **app** up into **components**.

## Defining our first component

Defining a component can seem tricky until you have some practice, but the gist is:

- If it represents an obvious "**chunk**" of your app, it's probably a component
- If it gets **reused** often, it's probably a **component**.

That **second bullet** is especially valuable: making a component out of **common UI element**s allows you to change your code in one place and see those changes everywhere that component is used. You don't have to break everything out into components right away, either. Let's take the second bullet point as inspiration and make a component out of the most **reused**, most important piece of the **UI**: a todo list item.



## Make a <Todo />

Before we can make a component, we should **create** a **new file** for it. In fact, we should make a **directory** just for our components. The following commands make a components directory and then, within that, a file called **Todo.js.** Make sure you're at the **root** of your app before you run these!

*mkdir src/components*
*touch src/components/Todo.js*

Our new Todo.js file is currently empty! Open it up and give it its first line:

```
import React from "react";
```

Since we're going to make a component called Todo, you can start adding the code for that to Todo.js too, as follows. In this code, we define the function and export it on the same line:

```
export default function Todo() {
  return (

  );
}
```

This is OK so far, but our component has to return something! Go back to src/App.js, copy the first <li> from inside the unordered list, and paste it into Todo.js so that it reads like this:

```
export default function Todo() {
  return (
    <li className="todo stack-small">
      <div className="c-cb">
        <input id="todo-0" type="checkbox" defaultChecked={true} />
        <label className="todo-label" htmlFor="todo-0">
          Eat
        </label>
      </div>
      <div className="btn-group">
        <button type="button" className="btn">
          Edit <span className="visually-hidden">Eat</span>
        </button>
        <button type="button" className="btn btn__danger">
          Delete <span className="visually-hidden">Eat</span>
        </button>
      </div>
    </li>
  );
}
```

**Note**: **Components must always return something. If at any point in the future you try to render a component that does not return anything, React will display an error in your browser**.

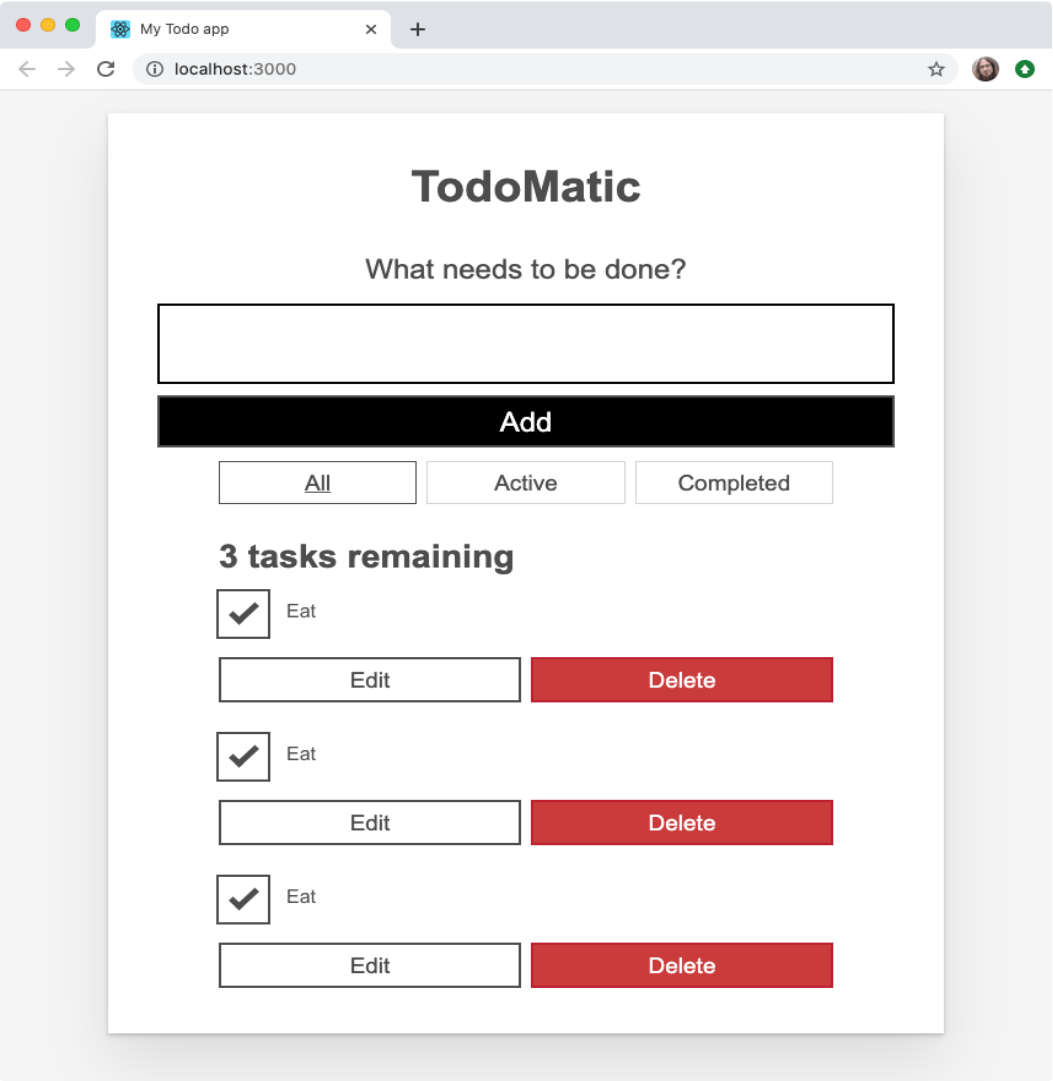
Our Todo component is complete, at least for now; now we can use it. In App.js, add the following line near the top of the file to import Todo:

*import Todo from "./components/Todo";*

With this component imported, you can replace all of the <li> elements in App.js with <Todo /> component calls. Your <ul> should read like this:

```
<ul
  role="list"
  className="todo-list stack-large stack-exception"
  aria-labelledby="list-heading"
>
  <Todo />
  <Todo />
  <Todo />
</ul>
```

When you look back at your browser, you'll notice something unfortunate: your list now repeats the first task three times!



In Similar Manner, We can componentize other things as well like each todo item,todo lists, and many more things.