# BONUS: React Fragments

(https://github.com/learn-co-curriculum/react-hooks-fragments) (https://github.com/learn-co-curriculum/react-hooks-fragments/issues/new)

## Learning Goals

- Use fragments in React components to return top-level elements

## Why Fragments

It is required that every React component must return a **single** JSX element. Because of this, we often use elements such as `div` to wrap other elements within our JSX. When rendered, this creates a DOM element for that outer `div`, which is sometimes unnecessary. For example:

```
function ChildComponent() {
  return (
    <div>
      <p>Hey, I am a child</p>
      <p>My name is child component</p>
    </div>
  );
}

function ParentComponent() {
  return (
    <div className="parent">
      <ChildComponent />
      <ChildComponent />
    </div>
  );
}
```

This setup creates a DOM structure that looks like this:

```
<div class="parent">
  <div>
    <p>Hey, I am a child</p>
    <p>My name is child component</p>
  </div>
  <div>
    <p>Hey, I am a child</p>
    <p>My name is child component</p>
  </div>
</div>
```

Those nested `div`s don't have any purpose here and don't have any styling besides their default properties. Without them though, we would have an error as there are *two* `p` tags being returned in the `ChildComponent`. Instead, we could use React Fragments, preventing the extra `div`s from being added to the DOM:

```
function ChildComponent() {
  //The wrapping 'div' here has been replaced with a React fragment
  return (
    <React.Fragment>
      <p>Hey, I am a child</p>
      <p>My name is child component</p>
    </React.Fragment>
  );
}

function ParentComponent() {
  return (
    <div>
      <ChildComponent />
      <ChildComponent />
```

```
      </div>
    );
  }
```

With the fragment in place, the DOM will now look like this:

```
<div>
  <p>Hey, I am a child</p>
  <p>My name is child component</p>
  <p>Hey, I am a child</p>
  <p>My name is child component</p>
</div>
```

`<React.Fragment>` allows a component to return multiple elements **without adding a wrapper element that adds to the DOM.**

You can also use the shorthand syntax for fragments to make your JSX cleaner:

```
function ChildComponent() {
  // <> === <React.Fragment>
  return (
    <>
      <p>Hey, I am a child</p>
      <p>My name is child component</p>
    </>
  );
}
```

Fragments are not restricted to the outermost element being returned in JSX. Imagine you had an array of book objects in your props that you want rendered to the DOM. Each book has multiple attributes you want to display, but you don't need an element that wraps around these attributes. A fragment can be used here, and can still take a key attribute:

```
function Bookshelf(props) {
  return (
```

```
      <section>
        {props.books.map((book) => (
          <React.Fragment key={book.id}>
            <h1>{book.title}</h1>
            <h2>{book.author}</h2>
          </React.Fragment>
        ))}
      </section>
    );
  }
```

# Conclusion

Fragments are a small addition to React overall, but when used properly, can reduce a lot of unnecessary DOM bloat. They allow us a bit more flexibility in how we write our components, eliminating the need for wrapper elements.

# Resources

- **React Fragment Support (https://reactjs.org/blog/2017/11/28/react-v16.2.0-fragment-support.html)**