# Exercise 4 Working with React Components

You will build a React application with *functional components*. You can use the previous development environment.

1. You will create a main App component that can hold other components.
   1. Open a command prompt and navigate to react-dev-env, and start the development server: npm start
   2. Open src/App.js from the react-dev-env project in an editor. You should see the following:

function App() { return (

<div id='main' >

<h1>My App</h1>

</div>

);

}

* 1. Add a functional component named Header after the App() function. The component should return an h3 element displaying the app's title.

function Header(){

return <h3>My App Title</h3>;

}

* 1. Modify the App function to display the Header component you just created and save the file.

function App() { return (

<div id='main' >

<Header />

</div>

);

}

* 1. Create a variable named title with the value My React App
  2. Replace the contents of the <h3> in the Header component with a JSX expression so that it displays the title variable you just created.
  3. The changes made to src/App.js should look like this:

const title = "My React App" function Header(){

return <h3>{title}</h3>;

}

* 1. Save the App.js file. Next we will add a few more functional components to App.js
  2. Add a functional component named Body that displays a div containing a paragraph with some random text.
  3. Add the Body component to the JSX code on the App component.
  4. Add a functional component named Footer that displays a div containing an <h4> tag with the text "App Footer"
  5. Add the Footer component to the JSX code on the App component. When you are done, the component functions should look something like this. Save and run the file:

function App() { return (

<div id='main' >

<Header />

<Body />

<Footer />

</div>

);

}

const title = "My React App" function Header(){

return <h3>{title}</h3>;

}

function Body(){

return ( <div><p>some random text</p></div> );

}

function Footer(){

return ( <div><h4>App Footer</h4></div> );

}

export default App;

1. Next you will create variables to hold property values and pass them to components.
   1. Open App.js in an editor. Add the two variables shown here (add them right after the 'title' variable):

var footerText = "footer text"

var author = {

name: "John Doe",

phone: "800-123-1212",

email: "jdoe@gmail.com"

}

* 1. Modify the component to pass footerText a property named "text" to the Footer component where it appears inside the App component. The resulting App component should look like this:

function App(){ return (

<div id='main'>

<Header />

<Body />

<Footer text={footerText} />

</div>

);

}

* 1. Modify the Footer functional component to take "props" as a parameter and then use a JSX expression to replace the text in the <h4> tag with "props.text"
  2. The Footer component should end up looking like this:

function Footer(props){

return (<div><h4>{props.text}</h4></div>);

}

* 1. Save changes. Switch to the browser window and make sure the footer is showing the property value that was passed to it
  2. Do the same with the Body component
     1. Pass "author" as a property named "author" to the Body component.
     2. Modify the Body component to take "props" as a parameter.
     3. Add a paragraph before the existing paragraph in the Body component that displays the author's information.
     4. The resulting App component should look like this:

function App(){ return (

<div>

<Header />

<Body author={author} />

<Footer text={footerText}/>

</div> );

}

* + 1. The resulting Body component might look like the following. Save changes. Refresh the browser make sure the page is showing the property value that was passed to it.

function Body(props){

return ( <div><p>Author:{props.author.name}</p>

<p>some random text</p></div> );

}

1. Add CSS styles in src/App.css
   1. Add a section for a class named "boxed" with the following rules:
      1. Width = 200px ( width: 200px; )
      2. Solid single line black border ( border: 1px solid black; )
   2. The code you add should look like this. Save the file.

.boxed{

width: 200px;

border: 1px black solid;

}

* 1. Edit src/App.js. Modify the App component.
     1. Remove the 'id=main' attribute from the outer div of the component
     2. Add the "boxed" class to the outer div of the component.
     3. Make sure to use the proper JSX syntax for setting the class
     4. The code should look like below:

function App(){ return (

<div className={'boxed'}>

<Header />

* + 1. Add the following style directly to the <h3> and <h4> elements of the Header and Footer components respectively.

backgroundColor: lightgrey

* + 1. The code to do this in JSX would look like this:

<h3 style={{backgroundColor: 'lightgrey'}} >

<h4 style={{backgroundColor: 'lightgrey'}} >

* + 1. Save changes. Switch to the browser and make sure the page is using the styles you set up. This could look better if we apply the following styles:

margin: '0px', padding: '5px', textAlign: 'center',

* + 1. We probably don't want to place these all inline in the component. Instead let's create a variable to hold them.
    2. Add the following variable to App.js (add it above the 'title' variable):

const divStyle = { backgroundColor: 'lightgrey', margin: '0px',

padding: '5px', textAlign: 'center',

};

* + 1. Modify the 'style' attribute of the <h3> and <h4> elements of the header and footer components to use 'divStyle' like this:

<h3 style={divStyle} >

<h4 style={divStyle} >

* + 1. The final Header and Footer components would look like this:

function Header(){

return <h3 style={divStyle} >{title}</h3>;

}

function Footer(props){

return ( <div><h4 style={divStyle} >{props.text}</h4></div> );

}

* + 1. Save changes. Switch to the browser and make sure the page is showing the styles we just added.

1. Fragments: When creating React component functions with JSX one of the requirements is that the JSX code return a single element. In real life, there are situations where you need to have it return multiple adjacent elements. In this lab part, we will review the issue and how the situation is solved by using React Fragments.
   1. Create a new component function in App.js called "FragTest".

function FragTest(props){ return <ul><ColorList /></ul>

}

* 1. Create a new component function named "ColorList" that returns a list of colors:

function ColorList(props){ return "Red, Yellow, Blue"

}

* 1. Add "Fragtest" to your App component after the Body component and before the Footer component. Save changes.

function App(){ return (

<div>

<Header />

<Body author={author} />

<FragTest />

<Footer text={footerText} />

</div> );

}

* 1. Now let's change the ColorList component so that it returns multiple adjacent elements. Change the return statement to match the following:

function ColorList(props){

return <li>Red</li>

<li>Yellow</li>

<li>Blue</li>

}

* 1. Save changes. Switch to the command prompt that is showing the development server. You should see an error.
  2. One way we could fix this is by wrapping the <li> elements with a <div>

function ColorList(props){ return <div>

<li>Red</li>

<li>Yellow</li>

<li>Blue</li>

</div>

}

* 1. Save changes. Switch to the browser. The error in the development server command prompt should go away.
  2. Open Chrome developer tools (F12) and click on the "Elements" tab. Expand the elements so you can see the ones added by ColorList. Let’s create the ColorList component so that there is no extra <div>, with React Fragments.
  3. Replace the <div> in ColorList with <> [short version of React Fragments]. When React creates the DOM it skips over and leaves out instances of the React.Fragment.

function ColorList(props){ return <>

<li>Red</li>

<li>Yellow</li>

<li>Blue</li>

</>

}

* 1. Save changes. Switch to the browser. In the browser, you should see the bulleted list like before. Open the Elements tab in the Chrome dev tools and check out the list. You should see that there is no longer a <div> or any other element in the hierarchy between <ul> and the <li> elements. Comment out the <FragTest /> element in the App component using the JSX comment syntax:

function App() { return (

<div className={'boxed'} >

<Header/>

<Body author={author} />

{/\* <FragTest /> \*/}

<Footer text={footerText} />

</div>

);

}