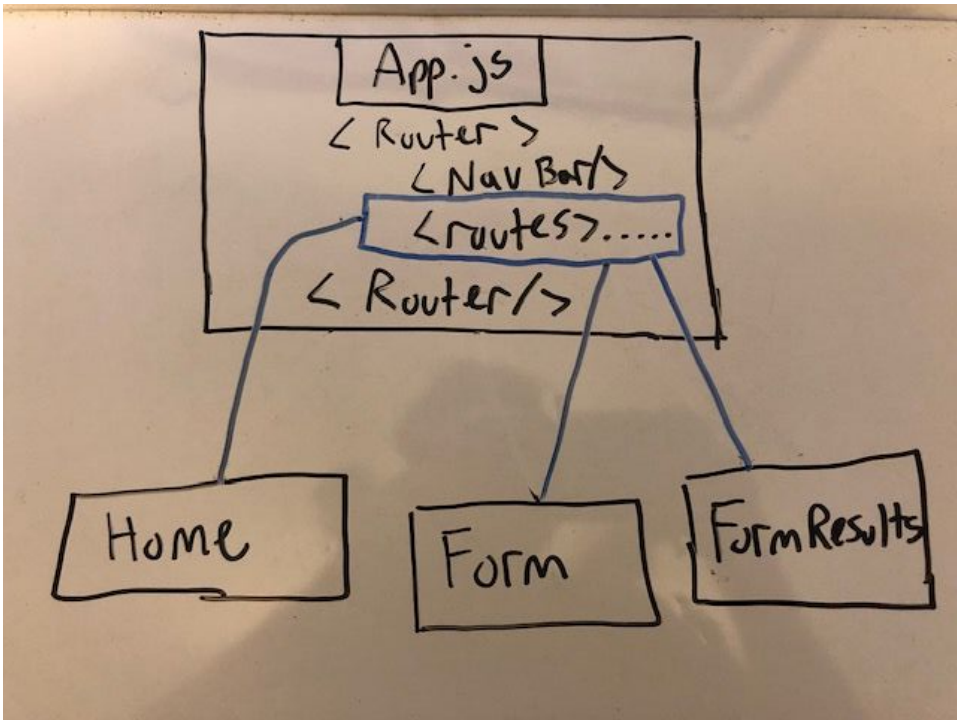
Task 2
Using Props

With the assumption that you guys have research on your own about React, you should know that there are two different types of components in React, stateful and stateless. The reason I bring this up with props, is that you usually pass a components state through props.

Props can only be passed down the hierarchy. Recall our tree diagram of the application.



If you define state in App.js you can pass those values down to any component in the tree.

Task 2a: Pass a function in App.js to a component.

1. Add state to App.js and make a key - value pair with the value of 0. Add an HTML element in App.js that displays this newly added state value above <NavBar>.
2. Create a function in App.js that increments the state value by 1. You should look up how to do this, there is a specific built-in function in React used to manipulate state.
3. Create a button in the NavBar component. Pass the function you just created in App.js to the NavBar component through props. Implement the button so that when it is clicked, it calls the increment function you passed to it.

Your final result should be a button that increments a number displayed on the screen. ***If you need help or hints with any of this, look at the Form.js component.*** It has both state and functions in it. You will have to Google how to pass props to a component. Don't try to mimic the way it is done in the commented out code with Router and Route in App.js.

Task 2b: Pass Props with the Route Component Tutorial

We will be completing the functionality for the Form.js and FormResults.js components for this task. The ultimate goal is to display whatever input text you entered into the form and display it in the FormResults.js component using props.

The Route component has some built-in props that it passes to the component it routes too. We'll be using this built-in functionality for this task.

1. Go to the Form.js file. The button element returned below the two TextField elements has an action listener called onClick which will execute the handleSubmit function when the button is clicked. Look at the handleSubmit function, it calls a very special function that is inherited from the Route props. this.props.history.push()... is a function where you specify a path to redirect to and add props if you wish.

```
handleSubmit(event) {
  this.props.history.push({
    pathname: "/results",
    state: this.state
  });
  event.preventDefault();
}
```

If we analyze the code above, we see that we will redirect to "/results" and add props which hold the values from this.state, which is the firsName and lastName key value pairs found in Form.js's state. So if we click the button, we will get redirected to whatever Route component has the path="/result" in App.js. Additionally, we will have the values stored in state from Form.js in our newly redirected to component.

2. Uncomment the code in App.js. It should be a Route component in the Router and an import for the FormResults component. Now, go to the form on the site, enter in your first name and last name and click submit, it should redirect you to the form results component and display "todo for task2".
3. This is the main task, figure out how to access props in FormResults.js and display the inputs from the form.