1. For this question consider all of the .js files in the SRC folder and the “imports” (the ones that import another component - FOR example: import Todos from './components/Todos';) that exist at the top of each of these pages:

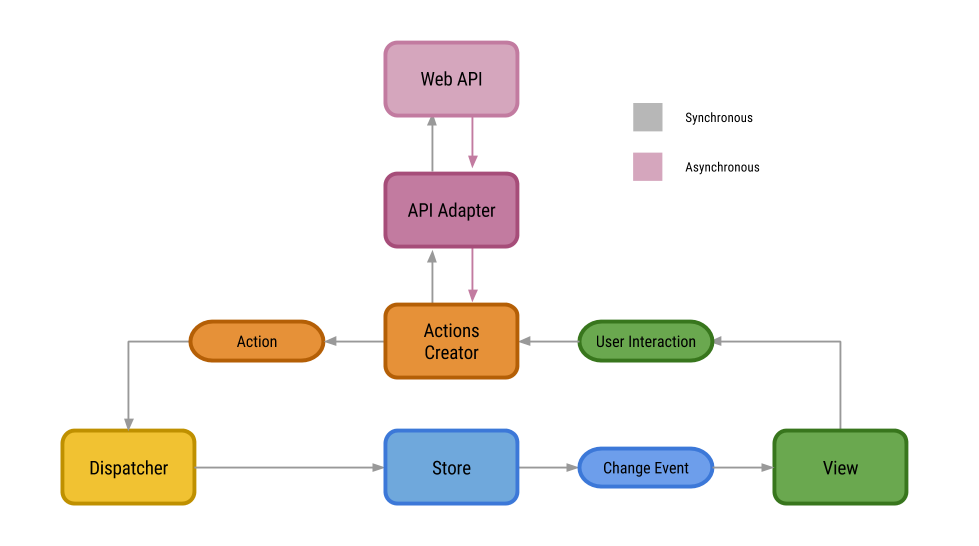
1. What do you notice about the system for importing components into another component in React? (In other words) If App.js is a “Parent” component to “Todos.js” a “child” component - Can you make sense of whether or not there is a rule about whether or not “child” components should be imported into “parent” components *or the opposite*?

Child component should be imported to parent component.

1. Why do components need to be imported at all? Test this by commenting out an import at the top of App.js.

Almost every file containing functional component needs to import react to convert JSX code into vanilla JavaScript. Anywhere we are going to use JSX, we need to import React.

1. Go to this link: <https://drive.google.com/file/d/1Y2sydLwhY51kPyIXWSRcrypCgfQmYU7Y/view?usp=sharing>
   1. Make a copy
   2. Create a diagram that represents all of the components and the way in which they are connected.



TodoApp is a component that use**AddItem and TodoItem components** to make one app that based on few collection on bit.

* 1. Using the diagram that was created, figure out the path that the “delTodo” function followed from when it was first created (in App.js) to when it became functional on the application (on the screen). What was its path? In other words which components was it brought into and how did it finally make its way onto the screen as an “x.”

We are separating our implementation to two files. The first one, **app.js**, is where the code executed by the main thread, including the creation of a new worker, is in. The second file, **worker.js**, includes the code run by the worker we create. It’s where the code for CPU-intensive Fibonacci calculation should be in.

* 1. Do the same thing with the state “todos.” In which components did it originate from? Which components was it brought into and how did it finally make its way to impacting what you see on the screen?

assing an update function allows you to access the current state value inside the updater. Since setState calls are batched, this lets you chain updates and ensure they build on top of each other instead of conflicting:

1. Look inside the AddTodo.js - notice that there is state (“title”) and two functions (onSubmit, onChange).

1. Why is the “title” state inside the AddTodo component but the “todos” state (which is also used in the AddTodo component) is not inside the AddTodo component?

The SDK documentation is as clear as mud for a "non-developer" like myself.  So far I have been able to figure out how to create variables and assign values to variables using JS but Object State Changes is a tough one.

1. Why are the two functions “onSubmit” and “onChange” inside the AddTodo component but the “addTodo” function in App.js which is only used by the AddTodo.js component is not inside the AddTodo.js component?

 computer checks the code from left to right, and the result of the next function will replace the previous one, and so on, until the end. So, even if all the 3 first functions return false, as long as the last one is true

1. Do you think that addTodo function in App.js could be taken out of App.js and placed in the AddTodo.js component without interrupting the Apps functionality?

Cannot be taken.

1. Go to TodoItem.js and notice the following code:

const { id, title, completed } = this.props.todo;

This is an example of “**destructuring**.” It is a shortcut taken by the person writing the code.

1. Comment this line out

This feature is called object destructuring and allows you to take properties of an object and store them conveniently into a variable.

1. Notice that the code now has some issues
2. Review the following link: <https://medium.com/@lcriswell/destructuring-props-in-react-b1c295005ce0>
3. Try to fix the code so that it works without using the destructuring.
4. const obj = {
5. prop1: 1,
6. prop2: 2
7. }
8. const {prop1, prop2} = obj;
9. console.log(prop1, prop2);

1. For this question please do the following
   1. Open the console
   2. Clear the console
   3. Refresh the application
      1. What is the output in the console when you refresh the application?

Console is returning to folder path

* + 1. Where is the code that resulted in this output?
    2. See link: <https://www.w3schools.com/react/react_lifecycle.asp>
       1. Be very specific: When did this happen?

# Program is printing “My Favorite Color is red

” title

1. For this question please take a look at the following link <https://www.w3schools.com/react/react_css.asp>
   1. Find in the code 3 types of styling

Program adding style to header.

class Header extends React.Component {

constructor(props) {

super(props);

this.state = {favoritecolor: "red"};

}

render() {

return (

<h1>My Favorite Color is {this.state.favoritecolor}</h1>

);

}

}

* + 1. Show and Describe those examples.

1. class MyHeader extends React.Component {
2. render() {
3. return (
4. <div>
5. <h1 style={{color: "red"}}>Hello Style!</h1>
6. <p>Add a little style!</p>
7. </div>
8. );
9. }
10. }

1. Take a look at the functions (markComplete, delTodo, addTodo) in App.js component. Each of them contains an “argument” or a “parameter” that was passed through it (id, id, title - respectively).
   1. Inside each of those functions curly brackets input for example **console.log(“testing1”, id);** to see what the “id” is.
   2. What is it?
   3. How exactly did it get there?

1. Notice the following: when you “npm start” from inside the folder the application routes to the home page which has a list of “Todos.” When you click on the word “About” from the heading the application routes you to a different display. When you click on the word “Home” the application routes you back to the original display of “Todos.”
   1. Take a look at the “return” section App.js and Header.js
      1. Try to manipulate the code in such a way that when you do a “npm start” the application loads to the “About” section and only goes to the (original) “Home” section with all of the “todos” if you click on the “Home” link.
   2. Explain what steps were necessary to make the above change.
   3. What is the word “Exact” used for in App.js.

1. Comment out the line “e.preventDefault();” and then press the submit button after entering something in the input box.
   1. How does the behaviour of the application change?
   2. Take a look at this link: <https://www.tutorialspoint.com/stop-making-form-to-reload-a-page-in-javascript>
   3. Explain what happens in an application when you have the “preventDefault()” vs. when you don’t.

1. When we cloned this application it had one ternary operator in it.
   1. If you are unclear on what a ternary operator is look it up.
   2. Do a search of ternary operators in the application folder. What is it doing?
   3. Create your own Ternary Operator in which you change the color of the red buttons to blue whenever the todos have been completed.

1. Explain in regular english what the following function is saying.
   1. Hint: To fully understand what is happening I recommend taking a look at where in the code the markComplete function is invoked
   2. markComplete = (id) => {
   3. this.setState({
   4. todos: this.state.todos.map((todo) => {
   5. if (todo.id === id) {
   6. todo.completed = !todo.completed;
   7. }
   8. return todo;
   9. }),
   10. });
   11. };