https://react.dev/

React public folder: It contains the assets that will be displayed to the user in your app and the robots.txt file, which is used for search engine optimization. Also, there is a manifest.json file, which is used to provide some metadata to a device when you're React powered web app is installed on it. While all these files are necessary, the most important one to know about for now is index.html. A React app gets injected into the specific elements inside the body of the index HTML file. Based on changes happening inside our React app, it injects those updates in that same div of index HTML.

React src folder: App.test.js, setupTests.js, and the reportsWebVitals.js are files related to the app's performance and testing. he most important file in the entire source folder is the index.js file. This file imports everything that this React app needs to render a working React app.

Rest React project files: The .gitignore file is using version control and it's used to specify what files and folders must be excluded from a project. It's important to note that this file is not specific to React. This means that other systems use this file too. The README.md file is a markdown file that gives some basic information on this project. Developers use this when they want to share the project's code on sites like GitHub. The package.json file lists information pertaining to my app, which allows npm to run several scripts and perform various tasks in the app itself. Finally, the package-loc.json file holds the list of all dependencies with a specific versions. The package.json file helps npm rebuild the app on another machine. Or if we delete the node modules folder with all the files that our project needs to run, the package-loc.json file has all the information for npm to be able to rebuild those files reliably. This file is there to ensure the npm tracks all the modules installations properly.

Props.children: To understand the concept of props.children, consider the following real-life situation: you have a couple of apples, and you have a couple of pears. You'd like to carry the apples some distance, so obviously, you'll use a bag.

It's not a "bag for apples". It's not a "bag for pairs". It's just a bag. Nothing about this bag makes it such that it needs to be referred to as a bag in which you'd only and always carry apples, nor a bag in which you'd only and always carry pears.

In a way, the bag "doesn't care" if it is used to carry apples or pears. Nothing about the bag changes. There are no changes in the bag's material, size, shape, or color - because it can handle apples or pears being carried inside of it, without issues.

UseEffect()=>

1st variation : When you put in an empty array as the second argument. That means we want to call our arrow function immediately after the first render and never call it again. So once again, just want to repeat that because it's super critical to understand if we put in an empty array, the error function is going to be called after the first render and it never gets called again.

The second variation of use effect is when we put in no second argument at all.

That means we want to call the arrow function immediately after the first render and then call it immediately after every single re render of our component. And then the third variation that we're going to see rather often is that we have an array with some number of elements inside of it. Once again, I'm showing one element here, but we could very easily have additional elements. Now, this counter thing right here, I'm just kind of imagining that there is some counter piece of state inside of my components.I'm not showing it in this little code snippet, but we can imagine that counter might be a prop or a piece of state. So it is a variable that is defined somewhere else inside of our components. If we put in an array with some number of elements inside of it, that means we want to call the arrow function immediately after the first render. And we also want to call it after re renders if that counter variable changed in between the two renders.