Pass Data from One Component to Another Component in ReactJS:

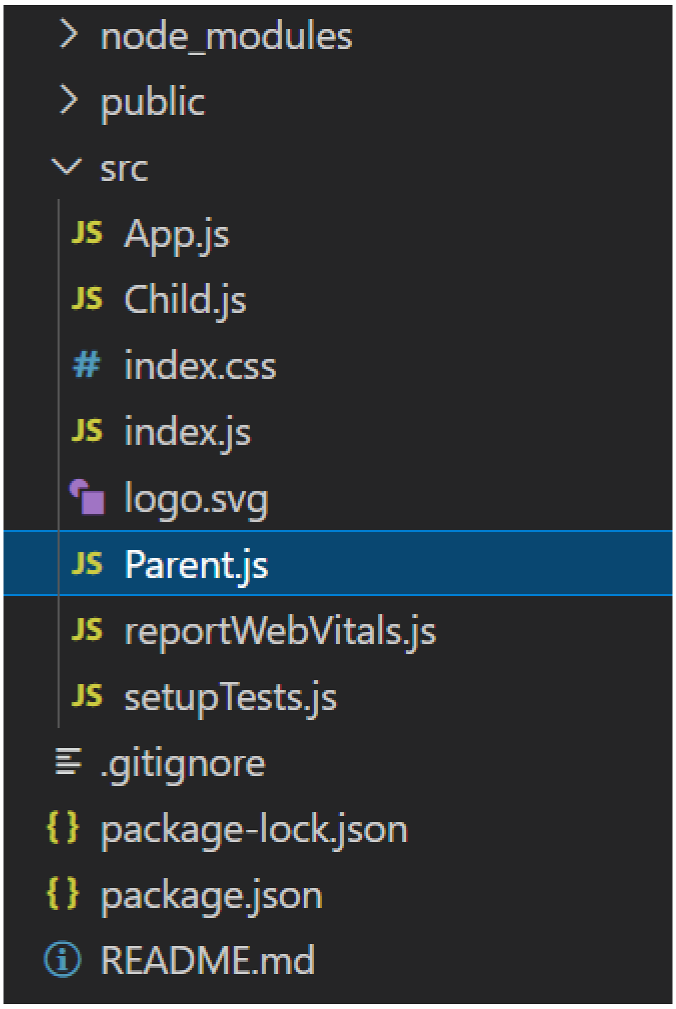
In React, passing data from one component to another component is a common task. It helps build a dynamic and interactive web application. We can pass data in components from parent to child, child to parent, and between siblings as shown below.

The 3 approaches to share or pass the data between React components.

- Passing data from Parent to Child in React
- Passing data from Child to Parent Component
- Passing data between Siblings

Project Structure

We have created two Components named **Child.js** and **Parent.js** as shown below.



```
> node_modules
> public
v src
    App.js
    Child.js
    index.css
    index.js
    logo.svg
    Parent.js
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

We have created two Components named **Child.js** and **Parent.js** as shown in the above structure

Approach 1: Passing data from Parent to Child in React

For passing data from parent to child component, we use props. Props data is sent by the parent component and cannot be changed by the child component as they are read-only.

Example: The following example covers **how to pass data from Parent to Child Component in ReactJS**.

CSSJavaScriptJavaScriptJavaScript

```
// Filename - Child.js

import React from "react";

const Child = (props) => {
    return <h3> {props.data} </h3>;
};

export default Child;
```

```
// Filename - App.js

import React from "react";
import "./index.css";
import Parent from "./Parent";
import "./App.css";

const App = () => {
    return (
        <div className="App">
            <h1 className="car">I Love Car</h1>
            <h3>This is App.js Component</h3>
            <Parent />
        </div>
    );
};

export default App;
```

```
// Filename - Parent.js

import React from "react";
import Child from "./Child";

const Parent = () => {
    const data = "Data from Parent to Child";
    return (
        <div>
```

```
            <h4>This is Parent component</h4>
            <Child data={data} />
        </div>
    );
};

export default Parent;
```

```
// Filename - Child.js

import React from "react";

const Child = (props) => {
    return <h3> {props.data} </h3>;
};

export default Child;
```

Step to Run Application: Run the application using the following command from the root directory of the project

```
npm start
```

Output: This output will be visible on http://localhost:3000/

```
I Love Car

This us App.js Component

This is Parent component

Data from Parent to child
```

Approach 2: Passing data from Child to Parent Component

For passing the data from the child component to the parent component, we have to create a callback function in the parent component and then pass the callback function to the child component as a prop. This callback function will retrieve the data from the child component. The child component calls the parent callback function using props and passes the data to the parent component.

Example: The following example covers how to pass data from Child to Parent Component in ReactJS.

```
/* Filename - App.css */

.App {
    text-align: center;
}
.container {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: center;
}

.item {
    min-width: 33rem;
    text-align: left;
}

.car {
    color: green;
}
```

```
// Filename - App.js

import React from "react";
import "./index.css";
import Parent from "./Parent";
import "./App.css";

const App = () => {
    return (
        <div className="App">
            <h1 className="car">I love car</h1>
            <Parent />
        </div>
    );
};

export default App;
```

```
// Filename - Parent.js

import React from "react";
import Child from "./Child";

class Parent extends React.Component {
    state = {
        msg: "",
    };
    handleCallback = (childData) => {
        this.setState({ msg: childData });
    };
    render() {
        const { msg } = this.state;
        return (
            <div>
                <Child
                    parentCallback={this.handleCallback}
                />
                <h1> {msg}</h1>
            </div>
        );
    }
}

export default Parent;
```

```
// Filename - Child.js

import React from "react";

class Child extends React.Component {
    onTrigger = () => {
        this.props.parentCallback("Welcome to car world");
    };
    render() {
        return (
            <div>
                <br></br> <br></br>
                <button onClick={this.onTrigger}>
                    Click me
                </button>
            </div>
        );
    }
}

export default Child;
```

Approach 3: Passing data between Siblings

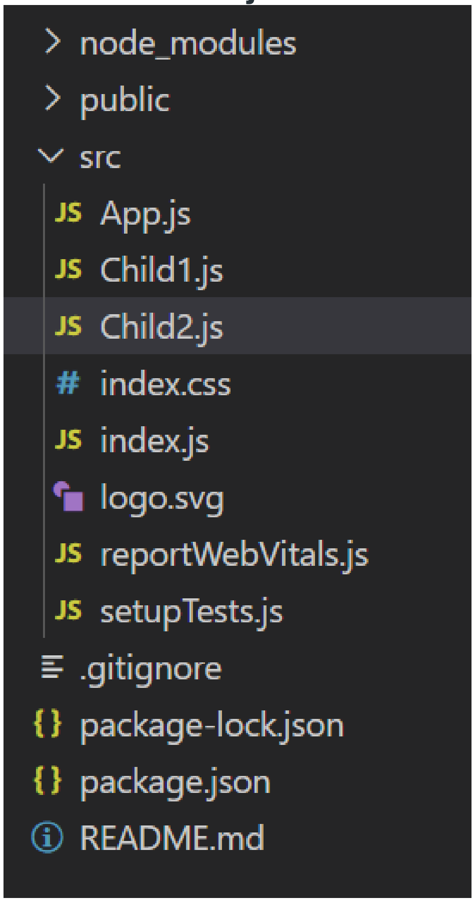
For passing data among siblings, there are multiple methods we can choose from as shown below:

- Combination of the above two methods (callback and use of props).
- Using Redux.
- ContextAPI

Here we will use context API to pass the data. When the button in child1 is clicked it will update the data in the Child2 component.

Folder Structure:

It will look like the following. We have created two Components named **Child1.js** and **Child2.js** as shown below.



```
> node_modules
> public
v src
    App.js
    Child1.js
    Child2.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

Example: In this example we will use context API to pass the data from one sibling Child1 to other sibling Child2.

```
/* Filename - App.css */

.App {
    text-align: center;
}

.car {
    color: green;
}



// Filename - App.js

import { React, useState, createContext } from "react";
import "./index.css";
import Child1 from "./Child1";
import "./App.css";
import Child2 from "./Child2";

// Create a new context and export
```

```
export const NameContext = createContext();

// Create a Context Provider
const NameContextProvider = ({ children }) => {
    const [name, setName] = useState(undefined);

    return (
        <NameContext.Provider value={{ name, setName }}>
            {children}
        </NameContext.Provider>
    );
};

const App = () => {
    return (
        <div className="App">
            <h1 className="car">I love car</h1>
            <NameContextProvider>
                <Child1 />
                <Child2 />
            </NameContextProvider>
        </div>
    );
};

export default App;


// Filename - Child1.js

import React, { useContext } from "react";
import { NameContext } from "./App";

const Child1 = () => {
    const { setName } = useContext(NameContext);
    function handle() {
        setName("car");
    }
    return (
        <div>
            <h3>This is Child1 Component</h3>
            <button onClick={() => handle()}>Click </button>
        </div>
    );
};

export default Child1;


// Filename - Child2.js

import React, { useContext } from "react";
import { NameContext } from "./App";

const Child2 = () => {
    const { name } = useContext(NameContext);

    return (
        <div>
            <br />
            <h4>This is Child2 Component</h4>
            <h4>hello: {name}</h4>
        </div>
    );
};

export default Child2;
```

Step to Run Application: Run the application using the following command from the root directory of the project
npm start