calculaterApp

صورة تحتوي على نص, لقطة شاشة, شاشة عرض, إلكترونيات

تم إنشاء الوصف تلقائياً

From Public --> index.js

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <meta name="theme-color" content="#000000">

    <!--

      manifest.json provides metadata used when your web app is added to the

      homescreen on Android. See https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/

    -->

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json">

    <link rel="shortcut icon" href="%PUBLIC\_URL%/favicon.ico">

    <!--

      Notice the use of %PUBLIC\_URL% in the tags above.

      It will be replaced with the URL of the `public` folder during the build.

      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

      work correctly both with client-side routing and a non-root public URL.

      Learn how to configure a non-root public URL by running `npm run build`.

    -->

    <title>React App</title>

</head>

<body>

    <noscript>

        You need to enable JavaScript to run this app.

    </noscript>

    <div id="root"></div>

    <!--

      This HTML file is a template.

      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.

      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.

      To create a production bundle, use `npm run build` or `yarn build`.

    -->

</body>

</html>

Src --> components-->button.components.jsx

import React from "react";

import "./button.styles.css";

const ButtonComponent = props => {

  return (

    <button

      className="button-style"

      onClick={() => props.handleClick(props.children)}

    >

      {props.children}

    </button>

  );

};

export default ButtonComponent;

src-->components-->button.styles.css

.button-style {

    flex-basis: 25%;

    background: #425062;

    color: #fff;

    padding: 20px;

    font-size: 25px;

    text-align: center;

    border: none;

    border-right: solid 2px #3c4857;

    border-bottom: solid 2px #3c4857;

  }

  .button-style:hover {

    background-color: white;

    color: #425062;

  }

Src-->App.js

import React from "react";

import "./styles.css";

import ButtonComponent from "./components/button.components";

export default function App() {

  const [calculation, setCalculation] = React.useState([0]);

  const [result, setResult] = React.useState(0);

  const [opera, setOpera] = React.useState(null);

  const [prevResult, setPrevResult] = React.useState(null);

  React.useEffect(() => {

    setCalculation([0]);

    setResult(0);

  }, []);

  React.useEffect(() => {

    //console.log(opera);

    if (opera != null) {

      //console.log(calculation);

      //console.log("result --- " + result);

      let itemBeforeOperator = 0;

      let itemAfterOperator = 0;

      let operatorFound = false;

      const length = calculation.length;

      for (let i = length - 1; i >= 0; i--) {

        if (calculation[i] !== "$") {

          if (opera === calculation[i]) {

            operatorFound = true;

            continue;

          }

          if (operatorFound) {

            if (result !== 0) {

              //console.log("Resultwokring")

              itemBeforeOperator = result;

            } else {

              //console.log("Not Resultwokring")

              itemBeforeOperator =

                itemBeforeOperator === 0

                  ? calculation[i]

                  : calculation[i] + itemBeforeOperator;

            }

          } else {

            itemAfterOperator =

              itemAfterOperator === 0

                ? calculation[i]

                : calculation[i] + itemAfterOperator;

          }

        } else {

          if (result !== 0) {

            //console.log("Resultwokring")

            itemBeforeOperator = result;

          }

          break;

        }

      }

      const lastItem = calculation[calculation.length - 1];

      // console.log("lastItem :: " + lastItem);

      // console.log("itemBeforeOperator :: " + itemBeforeOperator);

      // console.log("itemAfterOperator :: " + itemAfterOperator);

      if (lastItem !== opera) {

        setPrevResult(prevResult);

        let updatedResult = 0;

        switch (opera) {

          case "%":

            updatedResult = itemBeforeOperator % itemAfterOperator;

            setResult(updatedResult);

            break;

          case "/":

            updatedResult = itemBeforeOperator / itemAfterOperator;

            setResult(updatedResult);

            break;

          case "\*":

            updatedResult = itemBeforeOperator \* itemAfterOperator;

            setResult(updatedResult);

            break;

          case "+":

            updatedResult =

              parseInt(itemBeforeOperator) + parseInt(itemAfterOperator);

            setResult(updatedResult);

            break;

          case "-":

            updatedResult = itemBeforeOperator - itemAfterOperator;

            setResult(updatedResult);

            break;

          default:

            return;

        }

      }

    } else {

      return;

    }

  }, [opera, calculation]);

  const num\_handle = value => {

    if (calculation.length === 1 && calculation.indexOf(0) === 0) {

      setCalculation([value]);

    } else {

      setCalculation([...calculation, value]);

    }

    //console.log("num\_handle :: " + value);

  };

  const spcl\_handle = value => {

    if (calculation.length === 1 && calculation[calculation.length - 1] === 0) {

      return;

    }

    switch (value) {

      case "C":

        setCalculation([0]);

        setResult(0);

        setOpera(null);

        return;

      case "%":

        updateCalculation("%");

        setOpera("%");

        break;

      case "/":

        updateCalculation("/");

        setOpera("/");

        break;

      case "+":

        updateCalculation("+");

        setOpera("+");

        break;

      case "-":

        updateCalculation("-");

        setOpera("-");

        break;

      case "\*":

        updateCalculation("\*");

        setOpera("\*");

        break;

      case "del":

        if (calculation.length === 1) {

          if (result === calculation[calculation.length - 1]) {

            setCalculation([0]);

            setResult(0);

          }

        }

        const newCalculation = calculation.pop();

        if (newCalculation.length) {

          setResult(prevResult);

          setCalculation([...calculation]);

        } else {

          setCalculation([0]);

          setResult(0);

        }

        return;

      case ".":

        setCalculation([...calculation, "."]);

        return;

      case "=":

        setCalculation([result]);

        setOpera(null);

        return;

      default:

        return;

    }

  };

  const updateCalculation = value => {

    if (result !== 0) {

      setPrevResult(result);

      setCalculation([...calculation, "$", value]);

    } else {

      setCalculation([...calculation, value]);

    }

  };

  const showCalculation = () => {

    return calculation.filter(item => item !== "$");

  };

  return (

    <div className="App">

      <div className="show-calculation">

        <span className="result-calculation">{showCalculation()}</span>

        <span className="dashed-line" />

        <span className="final-result">

          {result === 0 ? "Start Calculating" : result}

        </span>

      </div>

      <div className="button-layout">

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          C

        </ButtonComponent>

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          ≠

        </ButtonComponent>

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          %

        </ButtonComponent>

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          /

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          7

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          8

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          9

        </ButtonComponent>

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          \*

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          4

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          5

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          6

        </ButtonComponent>

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          -

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          1

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          2

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          3

        </ButtonComponent>

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          +

        </ButtonComponent>

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          .

        </ButtonComponent>

        <ButtonComponent handleClick={value => num\_handle(value)}>

          0

        </ButtonComponent>

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          del

        </ButtonComponent>

        <ButtonComponent handleClick={value => spcl\_handle(value)}>

          =

        </ButtonComponent>

      </div>

    </div>

  );

}

Src-->index.js

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

const rootElement = document.getElementById("root");

ReactDOM.render(<App />, rootElement);

src--> Styles

\*{

    box-sizing: border-box;

  }

  .App {

    font-family: sans-serif;

    text-align: center;

    display: flex;

    flex-direction: column;

    margin: 20px auto 20px auto;

    width:400px;

    background: #3A4655;

    min-height: 400px;

    box-shadow: 0 8px 50px -7px black;

  }

  .show-calculation{

    display:flex;

    margin: 20px 25px;

    color:grey;

    flex-direction: column;

  }

  .result-calculation{

    font-size:24px;

    flex:1;

    display: flex;

    justify-content: right;

  }

  .dashed-line{

    flex:1;

    margin-top:10px;

    border-bottom: 1px dashed grey;

  }

  .final-result{

    font-size:30px;

    flex:1;

    margin: 15px 25px 10px 0px;

    display: flex;

    color:white;

    justify-content: right;

  }

  .button-layout{

    display:flex;

    justify-content: space-around;

    flex-wrap: wrap;

  }