**Step-by-Step Notes**

**1. Setup React with Vite**

* Created a new React project using Vite for fast build and modern setup.
* Installed required dependencies, including @mui/material for Material UI.

**2. Created Functional Components for Each Program**

* Created separate React components for each program:
  + EvenOrOdd.jsx
  + Factorial.jsx
  + PrimeComposite.jsx
  + Palindrome.jsx
  + MultiplicationTable.jsx
* Each component contains logic and UI for its respective functionality.

**3. Created Home.jsx as Main Component**

* Home.jsx serves as the main page.
* Imported all program components inside Home.jsx.
* Used React’s useState hook to manage which component is currently selected and displayed.

**4. Display Menu List Using Material UI**

* Used Material UI components such as:
  + Box for layout and container
  + Typography for text headers
  + List and ListItem for clickable menu options
  + ListItemButton for button-like list items with click handling
* Created a list of program names and displayed them using ListItemButton.

**5. Handle Menu Click to Switch Components**

* Added a click event handler (handleListItemClick) on each list item.
* On click, the state (componentName) updates to the selected program.
* Used a switch statement (renderComponent) to render the correct program component based on the selected state.

**6. Styled the UI Using Material UI’s sx Prop**

* Applied external styles to components using the sx prop.
* Added padding, margin, border-radius, shadows, background colors to Box and other components for better UI.
* Example:

jsx

CopyEdit

<Box sx={{ maxWidth: 1080, margin: "0 auto", padding: 2, bgcolor: "#fff", boxShadow: 3 }}>

* Used different colors to differentiate list items and hovered states.

**7. Centered the Main Container**

* Used CSS centering technique with:
  + maxWidth set to match or be less than screen resolution (e.g., 1080px)
  + margin: "0 auto" to center horizontally
* Ensured the main container is centered on the screen and responsive on smaller viewports.

**8. Fixed Import Path Issues**

* Corrected import paths for all program components, matching folder names (programs not programms) and file names.
* This resolved errors like:

pgsql

CopyEdit

Failed to resolve import "./programms/EvenOrOdd" from "src/Home.jsx".

**Summary:**  
You now have a React + Vite project that uses Material UI to display a main menu on the left and loads different program components dynamically on the right inside a centered, styled container.

If you want, I can provide a final version of the full Home.jsx and an example program component to consolidate everything. Would you like that?

4.1-mini

**You said:**

including programs

**ChatGPT said:**

Got it! Here’s a full consolidated step-by-step note **including** a final example of **Home.jsx** and all program components (EvenOrOdd.jsx, Factorial.jsx, PrimeComposite.jsx, Palindrome.jsx, MultiplicationTable.jsx) — with Material UI styling and correct imports.

**Full Step-by-Step with Complete Code**

**1. Project Structure**

css

CopyEdit

src/

components/

EvenOrOdd.jsx

Factorial.jsx

PrimeComposite.jsx

Palindrome.jsx

MultiplicationTable.jsx

Home.jsx

main.jsx

**2. Home.jsx (Main container with menu and render logic)**

jsx

CopyEdit

import { useState } from "react";

import { Box, Typography, List, ListItem, ListItemButton, Paper } from "@mui/material";

import EvenOrOdd from "./components/EvenOrOdd";

import Factorial from "./components/Factorial";

import PrimeComposite from "./components/PrimeComposite";

import Palindrome from "./components/Palindrome";

import MultiplicationTable from "./components/MultiplicationTable";

const Home = () => {

const questions = ["Even or Odd", "Factorial", "Prime or Composite", "Palindrome", "Multiplication Table"];

const [componentName, setComponentName] = useState(null);

const handleListItemClick = (event) => {

const selectedComponent = event.target.innerText;

setComponentName(selectedComponent);

};

const renderComponent = () => {

switch (componentName) {

case "Even or Odd":

return <EvenOrOdd />;

case "Factorial":

return <Factorial />;

case "Prime or Composite":

return <PrimeComposite />;

case "Palindrome":

return <Palindrome />;

case "Multiplication Table":

return <MultiplicationTable />;

default:

return (

<Typography variant="h6" sx={{ mt: 5, color: "gray" }}>

Select a program from the left menu

</Typography>

);

}

};

return (

<Box

sx={{

maxWidth: 1080,

margin: "50px auto",

display: "flex",

bgcolor: "#f5f5f5",

boxShadow: 3,

borderRadius: 3,

overflow: "hidden",

}}

>

{/\* Left Menu \*/}

<Paper

elevation={4}

sx={{ width: 240, bgcolor: "#1565c0", color: "white" }}

>

<Typography variant="h6" align="center" sx={{ py: 2 }}>

Programs

</Typography>

<List>

{questions.map((question) => (

<ListItem key={question} disablePadding>

<ListItemButton

onClick={handleListItemClick}

sx={{

"&:hover": { bgcolor: "#64b5f6", color: "black" },

backgroundColor: componentName === question ? "#1976d2" : "inherit",

color: componentName === question ? "white" : "inherit",

}}

>

{question}

</ListItemButton>

</ListItem>

))}

</List>

</Paper>

{/\* Right Content \*/}

<Box

sx={{

flexGrow: 1,

bgcolor: "white",

p: 3,

minHeight: 400,

}}

>

{renderComponent()}

</Box>

</Box>

);

};

export default Home;

**3. Example Program Components**

All components have basic input UI and output, styled using Material UI.

**EvenOrOdd.jsx**

jsx

CopyEdit

import { useState } from "react";

import { Box, TextField, Button, Typography } from "@mui/material";

const EvenOrOdd = () => {

const [number, setNumber] = useState("");

const [result, setResult] = useState("");

const handleCheck = () => {

if (number === "") {

setResult("Please enter a number");

return;

}

const num = parseInt(number);

if (isNaN(num)) {

setResult("Invalid number");

return;

}

setResult(num % 2 === 0 ? "Even" : "Odd");

};

return (

<Box sx={{ maxWidth: 400 }}>

<Typography variant="h5" gutterBottom>

Even or Odd Checker

</Typography>

<TextField

label="Enter Number"

value={number}

onChange={(e) => setNumber(e.target.value)}

fullWidth

sx={{ mb: 2 }}

/>

<Button variant="contained" onClick={handleCheck}>

Check

</Button>

{result && (

<Typography sx={{ mt: 2 }} variant="subtitle1">

Result: {result}

</Typography>

)}

</Box>

);

};

export default EvenOrOdd;

**Factorial.jsx**

jsx

CopyEdit

import { useState } from "react";

import { Box, TextField, Button, Typography } from "@mui/material";

const Factorial = () => {

const [number, setNumber] = useState("");

const [result, setResult] = useState(null);

const calculateFactorial = (n) => {

if (n === 0 || n === 1) return 1;

return n \* calculateFactorial(n - 1);

};

const handleCalculate = () => {

const num = parseInt(number);

if (isNaN(num) || num < 0) {

setResult("Please enter a valid non-negative integer");

return;

}

setResult(calculateFactorial(num));

};

return (

<Box sx={{ maxWidth: 400 }}>

<Typography variant="h5" gutterBottom>

Factorial Calculator

</Typography>

<TextField

label="Enter Number"

value={number}

onChange={(e) => setNumber(e.target.value)}

fullWidth

sx={{ mb: 2 }}

/>

<Button variant="contained" onClick={handleCalculate}>

Calculate

</Button>

{result !== null && (

<Typography sx={{ mt: 2 }} variant="subtitle1">

Result: {result}

</Typography>

)}

</Box>

);

};

export default Factorial;

**PrimeComposite.jsx**

jsx

CopyEdit

import { useState } from "react";

import { Box, TextField, Button, Typography } from "@mui/material";

const PrimeComposite = () => {

const [number, setNumber] = useState("");

const [result, setResult] = useState("");

const isPrime = (num) => {

if (num <= 1) return false;

if (num <= 3) return true;

if (num % 2 === 0 || num % 3 === 0) return false;

for (let i = 5; i \* i <= num; i += 6) {

if (num % i === 0 || num % (i + 2) === 0) return false;

}

return true;

};

const handleCheck = () => {

const num = parseInt(number);

if (isNaN(num) || num < 0) {

setResult("Please enter a valid non-negative integer");

return;

}

if (num === 0 || num === 1) {

setResult(`${num} is neither prime nor composite.`);

return;

}

setResult(isPrime(num) ? "Prime" : "Composite");

};

return (

<Box sx={{ maxWidth: 400 }}>

<Typography variant="h5" gutterBottom>

Prime or Composite Checker

</Typography>

<TextField

label="Enter Number"

value={number}

onChange={(e) => setNumber(e.target.value)}

fullWidth

sx={{ mb: 2 }}

/>

<Button variant="contained" onClick={handleCheck}>

Check

</Button>

{result && (

<Typography sx={{ mt: 2 }} variant="subtitle1">

Result: {result}

</Typography>

)}

</Box>

);

};

export default PrimeComposite;

**Palindrome.jsx**

jsx

CopyEdit

import { useState } from "react";

import { Box, TextField, Button, Typography } from "@mui/material";

const Palindrome = () => {

const [text, setText] = useState("");

const [result, setResult] = useState("");

const handleCheck = () => {

const normalized = text.toLowerCase().replace(/[^a-z0-9]/g, "");

const reversed = normalized.split("").reverse().join("");

setResult(normalized === reversed ? "Palindrome" : "Not a palindrome");

};

return (

<Box sx={{ maxWidth: 400 }}>

<Typography variant="h5" gutterBottom>

Palindrome Checker

</Typography>

<TextField

label="Enter Text"

value={text}

onChange={(e) => setText(e.target.value)}

fullWidth

sx={{ mb: 2 }}

/>

<Button variant="contained" onClick={handleCheck}>

Check

</Button>

{result && (

<Typography sx={{ mt: 2 }} variant="subtitle1">

Result: {result}

</Typography>

)}

</Box>

);

};

export default Palindrome;

**MultiplicationTable.jsx**

jsx

CopyEdit

import { useState } from "react";

import { Box, TextField, Button, Typography, Table, TableBody, TableCell, TableRow } from "@mui/material";

const MultiplicationTable = () => {

const [number, setNumber] = useState("");

const [table, setTable] = useState([]);

const handleGenerate = () => {

const num = parseInt(number);

if (isNaN(num)) {

setTable([]);

return;

}

const tempTable = [];

for (let i = 1; i <= 10; i++) {

tempTable.push({ multiplier: i, result: num \* i });

}

setTable(tempTable);

};

return (

<Box sx={{ maxWidth: 400 }}>

<Typography variant="h5" gutterBottom>

Multiplication Table Generator

</Typography>

<TextField

label="Enter Number"

value={number}

onChange={(e) => setNumber(e.target.value)}

fullWidth

sx={{ mb: 2 }}

/>

<Button variant="contained" onClick={handleGenerate}>

Generate

</Button>

{table.length > 0 && (

<Table sx={{ mt: 2 }}>

<TableBody>

{table.map((row) => (

<TableRow key={row.multiplier}>

<TableCell>{row.multiplier} × {number} =</TableCell>

<TableCell>{row.result}</TableCell>

</TableRow>

))}

</TableBody>

</Table>

)}

</Box>

);

};

export default MultiplicationTable;

**Summary**

* **Home.jsx**: Main container with side menu and dynamic rendering using state.
* **Material UI**: Used for UI components and styling (Box, Typography, List, Button, etc.).
* **State management**: Used React useState for menu selection and form input.
* **Component modularization**: Each program is a separate React functional component.
* **UI/UX**: Clear layout, interactive menu, feedback on user input.