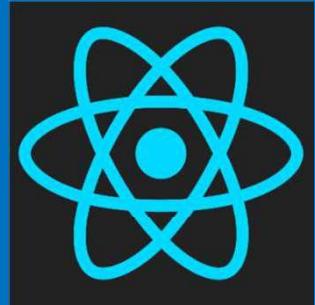


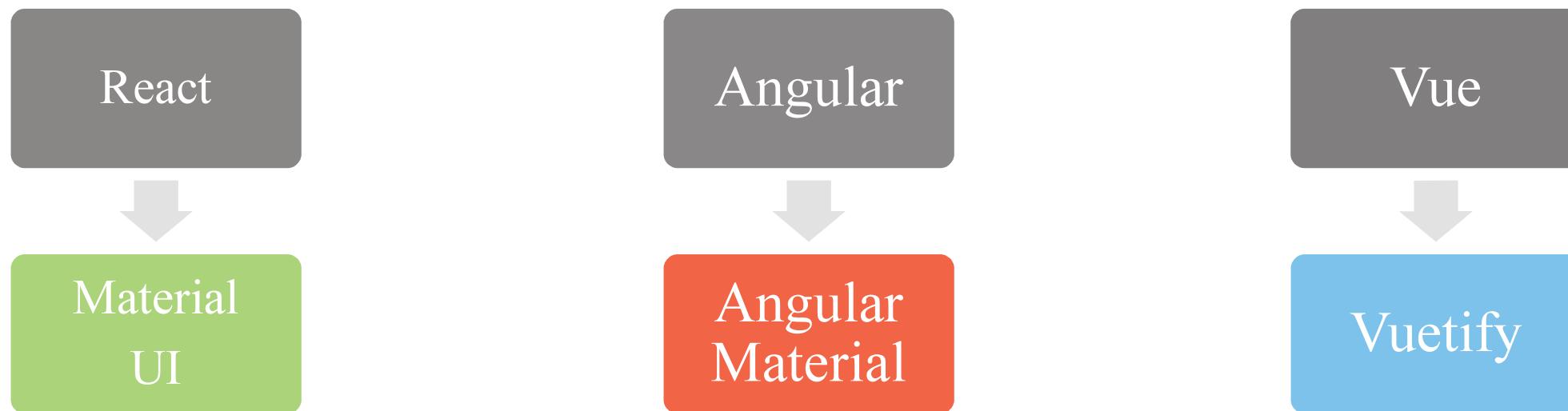
1 Creating Beautiful Apps with Material UI



Material UI Introduction

What is Material UI ?

- UI Component Library
- Provides us with components to build awesome user interfaces in quick time
- Implementation of Google's Material Design Specification
- Using These We Can Build Clean and Elegant UI



Introduction

- Material UI is an open-source React component library that implements Google's [Material Design](#).
- It includes a comprehensive collection of prebuilt components that are ready for use in production right out of the box.
- Material UI is beautiful by design and features a suite of customization options that make it easy to implement your own custom design system on top of our components.

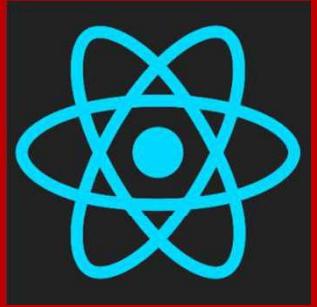
Advantages of Material UI

- **Ship faster:** Over 2,500 open-source [contributors](#) have poured countless hours into these components. Focus on your core business logic instead of reinventing the wheel—we've got your UI covered.
- **Beautiful by default:** we're meticulous about our implementation of [Material Design](#), ensuring that every Material UI component meets the highest standards of form and function, but diverge from the official spec where necessary to provide multiple great options.
- **Customizability:** the library includes an extensive set of intuitive customizability features. [The templates](#) in our store demonstrate how far you can go with customization.

Material UI vs. Base UI

- Material UI and [Base UI](#) feature many of the same UI components, but Base UI comes without any default styles or styling solutions.
- Material UI is *comprehensive* in that it comes packaged with default styles, and is optimized to work with [Emotion](#) (or [styled-components](#)).
- Base UI, by contrast, could be considered the "skeletal" or "headless" counterpart to Material UI—in fact, future versions of Material UI will use Base UI components and hooks for its foundational structure.

- **Cross-team collaboration:** Material UI's intuitive developer experience reduces the barrier to entry for back-end developers and less technical designers, empowering teams to collaborate more effectively. The [design kits](#) streamline your workflow and boost consistency between designers and developers.
- **Trusted by thousands of organizations:** Material UI has the largest UI community in the React ecosystem. It's almost as old as React itself—its history stretches back to 2014—and we're in this for the long haul. You can count on the community's support for years to come (e.g. [Stack Overflow](#)).



Material UI Installation

Default installation

Run one of the following commands to add Material UI to your project:

npm

```
npm install @mui/material @emotion/react @emotion/styled
```

yarn

```
yarn add @mui/material @emotion/react @emotion/styled
```

With Styled - Components

With styled-components

Material UI uses [Emotion](#) as its default styling engine. If you want to use [styled-components](#) instead, run one of the following commands:

npm

```
npm install @mui/material @mui/styled-engine-sc styled-components
```

yarn

```
yarn add @mui/material @mui/styled-engine-sc styled-components
```

Visit the [Styled engine guide](#) for more information about how to configure styled-components.

Peer dependencies



Please note that [react](#) and [react-dom](#) are peer dependencies too:

```
"peerDependencies": {  
  "react": "^17.0.0 || ^18.0.0",  
  "react-dom": "^17.0.0 || ^18.0.0"  
},
```

Robo Fonts

The screenshot shows the Material UI documentation for the Roboto font. The left sidebar includes links for Diamond Sponsors (Octopus Deploy, doIt, ZESTY), Getting started (Overview, Installation, Usage, Example projects, Templates, Learn, Design resources, FAQs, Supported components, Supported platforms, Support), Components, and Component API. The main content area features a header 'Robo font' with a note about its default use in Material UI, installation instructions for npm and yarn, and code snippets for importing Roboto from Fontsource. A callout box explains that Fontsource can load specific subsets, weights, and styles. The right sidebar contains a message of support for Ukraine, a contents list, and links for the Roboto font's installation via npm, yarn, Google Web Fonts, Icons, and CDN.

MUI CORE
Material UI v5.13.6

Diamond Sponsors

- Octopus Deploy
- doIt
- ZESTY

Getting started

- Overview
- Installation**
- Usage
- Example projects
- Templates
- Learn
- Design resources
- FAQs
- Supported components
- Supported platforms
- Support

Components

Component API

Robo font

Material UI is designed to use the [Roboto](#) font by default. You may add it to your project with npm or yarn via [Fontsource](#), or with the Google Fonts CDN.

npm

```
npm install @fontsource/roboto
```

yarn

```
yarn add @fontsource/roboto
```

Then you can import it in your entry point like this:

```
import '@fontsource/roboto/300.css';
import '@fontsource/roboto/400.css';
import '@fontsource/roboto/500.css';
import '@fontsource/roboto/700.css';
```

Fontsource can be configured to load specific subsets, weights and styles. Material UI's default typography configuration relies only on the 300, 400, 500, and 700 font weights.

Google Web Fonts

To install the Roboto font in your project using the Google Web Fonts CDN, add the following

MUI stands in solidarity with the Ukrainian people against the Russian invasion.
Find out how you can help.

CONTENTS

- Default installation
- npm
- yarn
- With styled-components
- npm
- yarn
- Peer dependencies
- Roboto font**
- npm
- yarn
- Google Web Fonts
- Icons
- npm
- yarn
- Google Web Fonts
- CDN

Icons

To use the [font Icon component](#) or the prebuilt SVG Material Icons (such as those found in the [icon demos](#)), you must first install the [Material Icons](#) font. You can do so with npm or yarn, or with the Google Web Fonts CDN.

npm

```
npm install @mui/icons-material
```

yarn

```
yarn add @mui/icons-material
```

CDN



You can start using Material UI right away with minimal front-end infrastructure by installing it via CDN, which is a great option for rapid prototyping. Follow [this CDN example](#) to get started.

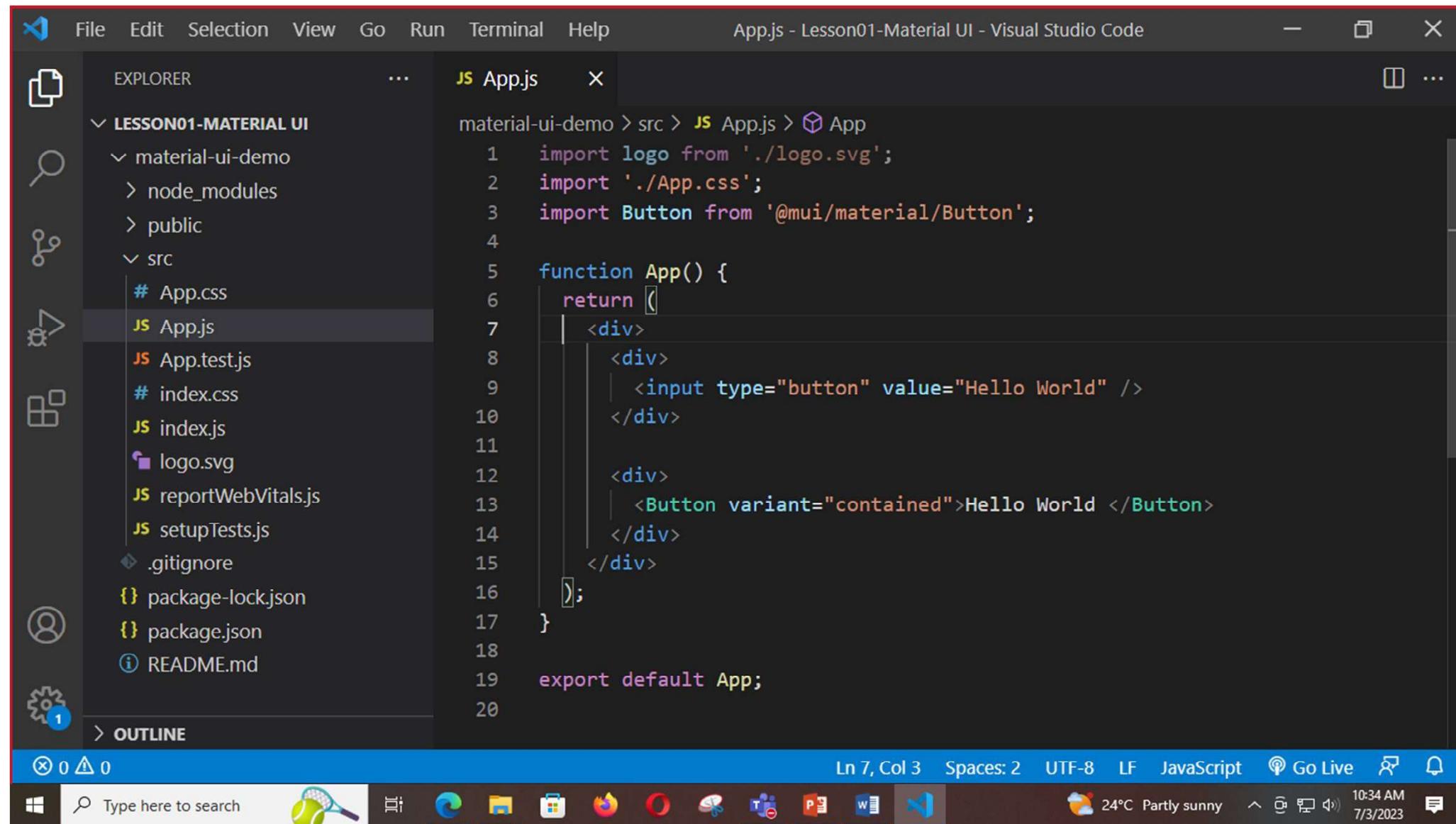
We do *not* recommend using this approach in production. It requires the client to download the entire library—regardless of which components are actually used—which negatively impacts performance and bandwidth utilization.

Two Universal Module Definition (UMD) files are provided:

- one for development: <https://unpkg.com/@mui/material@latest/umd/material-ui.development.js>
- one for production: <https://unpkg.com/@mui/material@latest/umd/material-ui.production.min.js>

The UMD links use the `latest` tag to point to the latest version of the library. This pointer is *unstable* and subject to change as we release new versions. You should consider pointing to a specific version, such as [v5.0.0](#).

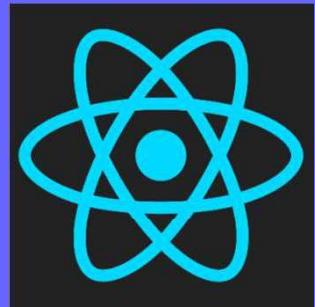
Lets See How to Add Material to React Application



The screenshot shows a Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** App.js - Lesson01-Material UI - Visual Studio Code.
- Explorer Panel:** Shows the project structure under "LESSON01-MATERIAL UI".
 - material-ui-demo
 - node_modules
 - public
 - src
 - # App.css
 - JS App.js** (selected)
 - JS App.test.js
 - # index.css
 - JS index.js
 - logo.svg
 - JS reportWebVitals.js
 - JS setupTests.js
 - .gitignore
 - { package-lock.json
 - { package.json
 - README.md
- Code Editor:** Displays the content of App.js.

```
material-ui-demo > src > JS App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3 import Button from '@mui/material/Button';
4
5 function App() {
6   return (
7     <div>
8       <div>
9         <input type="button" value="Hello World" />
10      </div>
11
12      <div>
13        <Button variant="contained">Hello World </Button>
14      </div>
15    </div>
16  );
17}
18
19 export default App;
```
- Bottom Status Bar:** Shows file statistics (0 changes), current position (Ln 7, Col 3), encoding (UTF-8), language (JavaScript), and a "Go Live" button.
- Taskbar:** Shows the Windows Start button, a search bar, and icons for various applications like Edge, File Explorer, and Task View.
- System Tray:** Shows the date and time (10:34 AM, 7/3/2023), weather (24°C Partly sunny), and other system status indicators.



Input Material UI Comps

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** App.js - Lesson01-Material UI - Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Search Bar:** JS App.js X JS Demo06-RadioGroup.js
- Sidebar Icons:** A vertical sidebar on the left contains icons for file operations (New, Open, Save, Find, Replace, Undo, Redo), a search icon, a refresh icon, a gear icon with a '1' notification, and a help icon.
- Code Editor:** The main area displays the following code for `App.js`:

```
material-ui-demo > src > JS App.js > ...
1 import logo from './logo.svg';
2 import './App.css';
3 import Button from '@mui/material/Button';
4 import ComboBox from './Demo01-Autocomplete';
5 import BasicButtons from './Demo02-Button';
6 import BasicButtonGroup from './Demo03-ButtonGroup';
7 import Checkboxes from './Demo04-CheckBox';
8 import FloatingActionButtons from './Demo05-FAB';
9 import RadioButtonsGroup from './Demo06-RadioGroup';
10
11 function App() {
12   return (
13     <div>
14       /* <div>
15         <input type="button" value="Hello World" />
16       </div>
17
18     <div>
19       <Button variant="contained">Hello World </Button>
20     </div> *)
21 }
```
- Bottom Status Bar:** Shows line 37, column 1, spaces: 2, encoding: UTF-8, LF, JavaScript, Go Live, and system status (Windows 10, 25°C, Partly sunny, 12:03 PM, 7/3/2023).

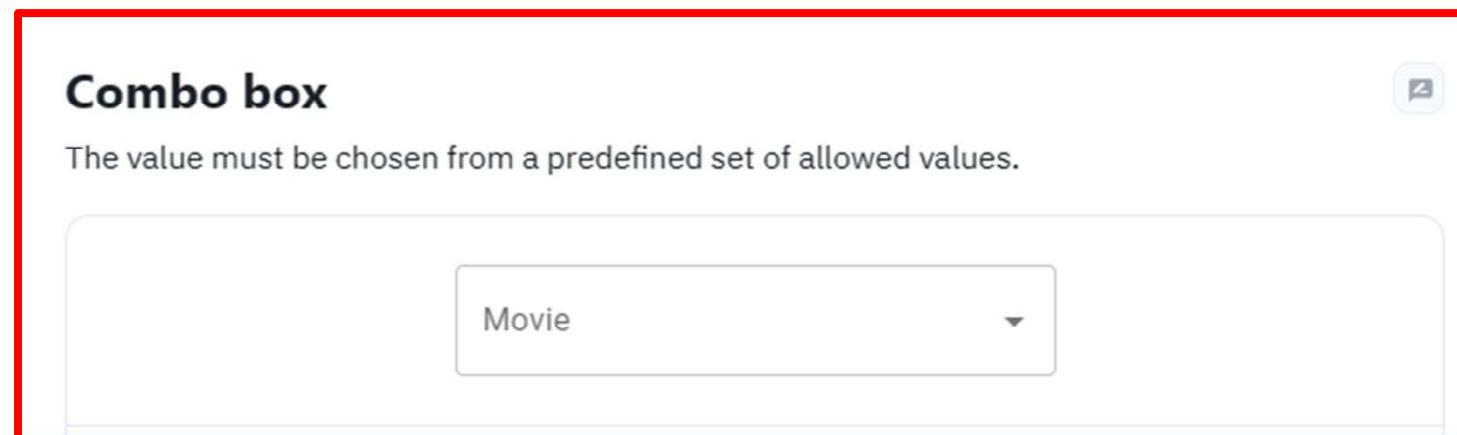
A screenshot of the Visual Studio Code interface. The title bar shows "App.js - Lesson01-Material UI - Visual Studio Code". The left sidebar has icons for file operations, search, navigation, and other tools. The main editor area displays the following code:

```
material-ui-demo > src > JS App.js > ...
22   <ComboBox></ComboBox>
23
24   <BasicButtons></BasicButtons>
25
26   <BasicButtonGroup></BasicButtonGroup>
27
28   <Checkboxes></Checkboxes>
29
30   <FloatingActionButton (alias) function RadioButtonsGroup(): React.JSX.Element
31       import RadioButtonsGroup
32   <RadioButtonsGroup></RadioButtonsGroup>
33
34   </div>
35 )
36 }
37
38 export default App;
39
```

The code uses JSX syntax and imports components from the Material UI library. The status bar at the bottom shows "Ln 37, Col 1", "Spaces: 2", "UTF-8", "LF", "JavaScript", "Go Live" button, and a system tray with weather information (25°C Partly sunny) and date/time (12:03 PM 7/3/2023).

Auto Complete

- The autocomplete is a normal text input enhanced by a panel of suggested options.
- The value for the textbox must be chosen from a predefined set of allowed values, e.g., a location field must contain a valid location name: combo box.
- The textbox may contain any arbitrary value, but it is advantageous to suggest possible values to the user, e.g., a search field may suggest similar or previous searches to save the user time: free solo.

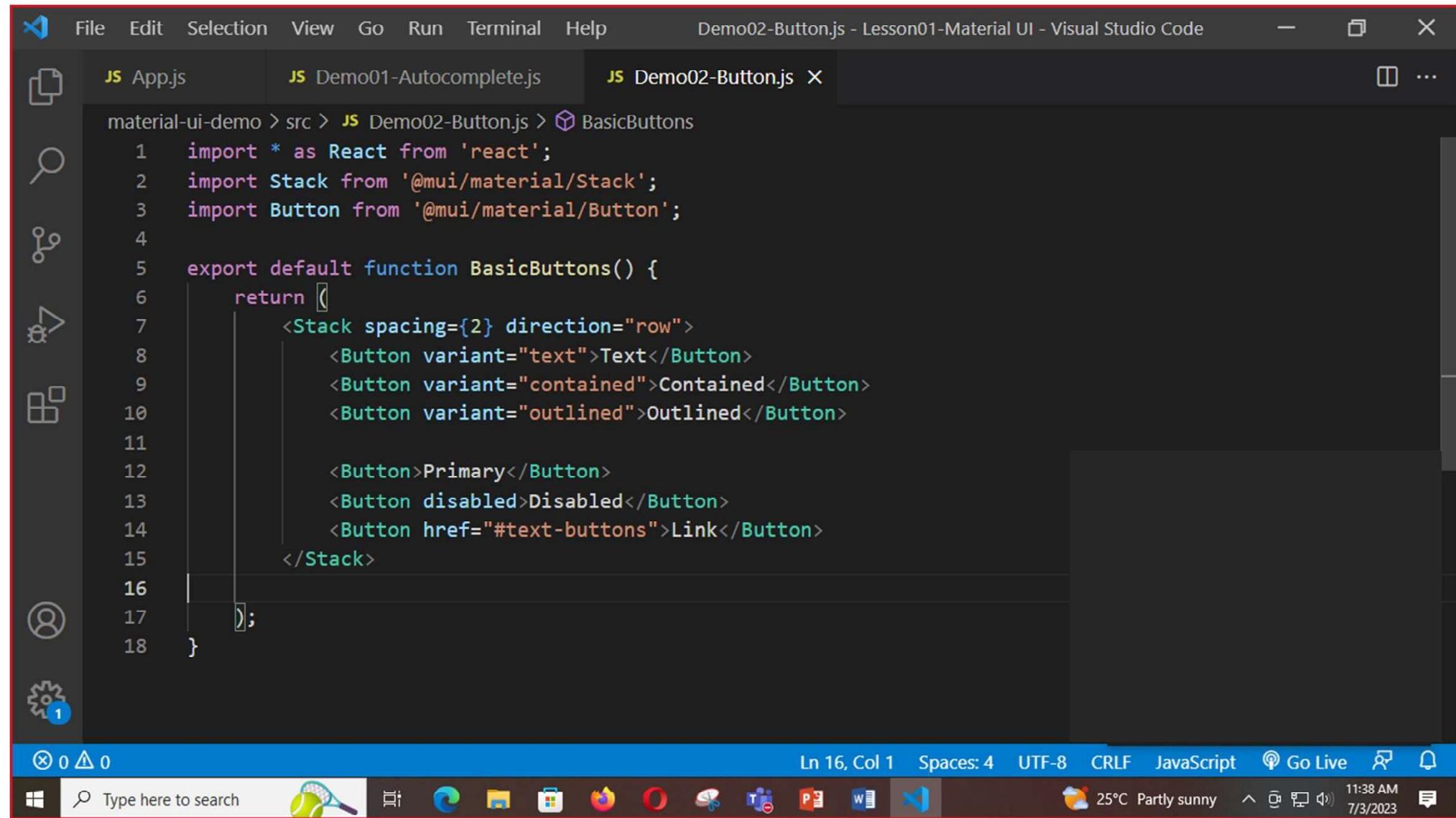


The screenshot shows a Visual Studio Code interface with a dark theme. The top bar includes the title "Demo01-Autocomplete.js - Lesson01-Material UI - Visual Studio Code" and standard menu options: File, Edit, Selection, View, Go, Run, Terminal, Help. The left sidebar features icons for file operations like Open, Save, Find, and others. The main editor area displays a JavaScript file named "Demo01-Autocomplete.js". The code implements a Material UI Autocomplete component for selecting movies from a list of top 100 films. The code uses React hooks and the Material UI Autocomplete API.

```
material-ui-demo > src > Demo01-Autocomplete.js > ComboBox
1 import * as React from 'react';
2 import TextField from '@mui/material/TextField';
3 import Autocomplete from '@mui/material/Autocomplete';
4
5 export default function ComboBox() {
6   return (
7     <Autocomplete
8       disablePortal
9       id="combo-box-demo"
10      options={top100Films}
11      sx={{ width: 300 }}
12      renderInput={(params) => <TextField {...params} label="Movie" />}
13    />
14  );
15}
16
17 // Top 100 films as rated by IMDb users. http://www.imdb.com/chart/top
18 const top100Films = [
19   { label: 'The Shawshank Redemption', year: 1994 },
20   { label: 'The Godfather', year: 1972 },
21   { label: 'The Godfather: Part II', year: 1974 },
```

At the bottom, the status bar shows file statistics (0 changes), code analysis results (0 errors, 0 warnings), and system information (Line 14, Col 5, Spaces: 4, UTF-8, CRLF, JavaScript). It also displays a Go Live button, a taskbar with various application icons, and a system tray showing weather (25°C, Partly sunny), date (7/3/2023), and time (11:08 AM).

- Buttons allow users to take actions, and make choices, with a single tap.
- Buttons communicate actions that users can take. They are typically placed throughout your UI, in places like:
 - Modal windows
 - Forms
 - Cards
 - Toolbars



```
material-ui-demo > src > Demo02-Button.js > BasicButtons
1 import * as React from 'react';
2 import Stack from '@mui/material/Stack';
3 import Button from '@mui/material/Button';
4
5 export default function BasicButtons() {
6     return [
7         <Stack spacing={2} direction="row">
8             <Button variant="text">Text</Button>
9             <Button variant="contained">Contained</Button>
10            <Button variant="outlined">Outlined</Button>
11
12            <Button>Primary</Button>
13            <Button disabled>Disabled</Button>
14            <Button href="#text-buttons">Link</Button>
15        </Stack>
16    ];
17}
18
```

The screenshot shows a Visual Studio Code window with a dark theme. The title bar reads "Demo02-Button.js - Lesson01-Material UI - Visual Studio Code". The left sidebar has icons for file operations like copy, search, and refresh. The main editor area contains the provided code. The status bar at the bottom shows "Ln 16, Col 1" and other settings like "Spaces: 4", "UTF-8", "CRLF", "JavaScript", and "Go Live". The taskbar at the bottom includes a search bar, pinned application icons, and system status indicators.

ButtonGroup

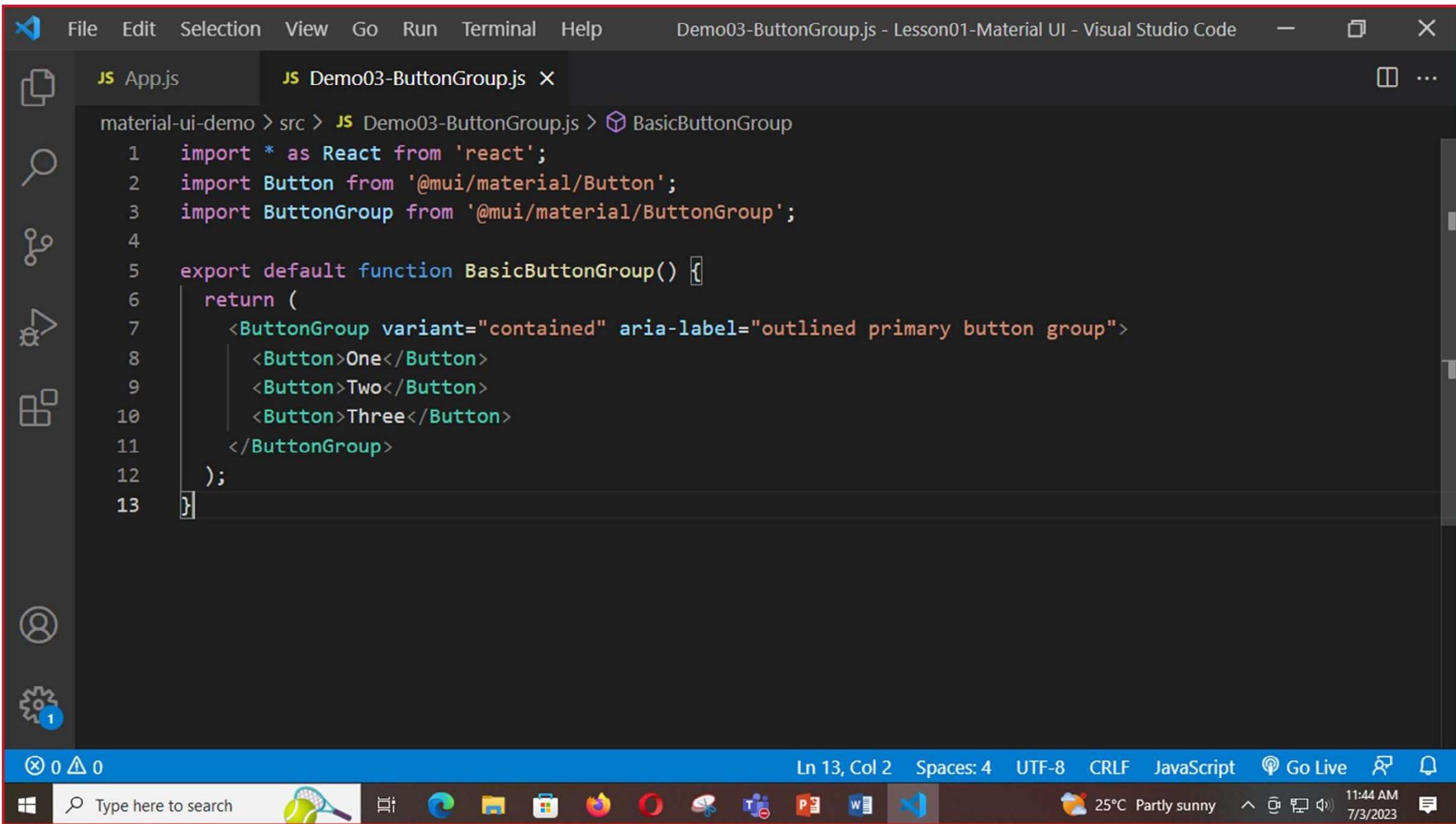
- The ButtonGroup component can be used to group related buttons.

Basic button group

The buttons can be grouped by wrapping them with the `buttonGroup` component. They need to be immediate children.



ONE TWO THREE

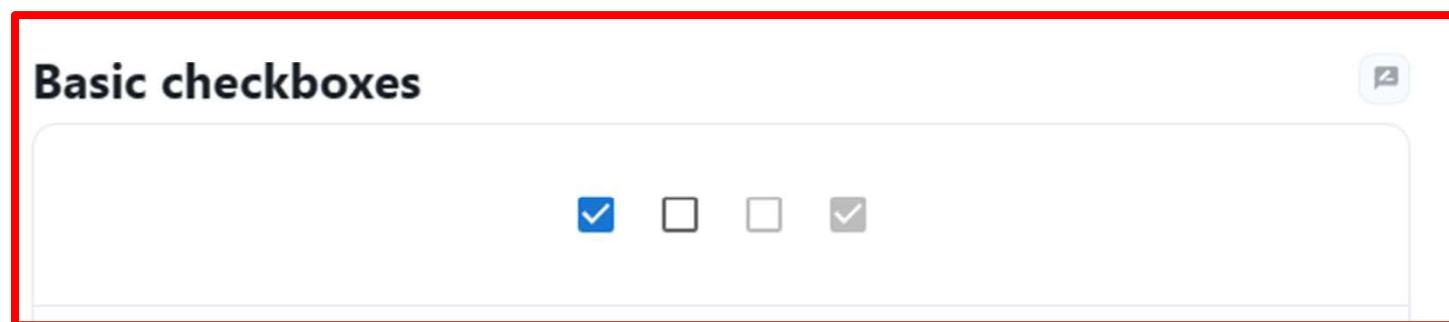


```
material-ui-demo > src > Demo03-ButtonGroup.js > BasicButtonGroup
1 import * as React from 'react';
2 import Button from '@mui/material/Button';
3 import ButtonGroup from '@mui/material/ButtonGroup';
4
5 export default function BasicButtonGroup() {
6   return (
7     <ButtonGroup variant="contained" aria-label="outlined primary button group">
8       <Button>One</Button>
9       <Button>Two</Button>
10      <Button>Three</Button>
11    </ButtonGroup>
12  );
13}
```

The screenshot shows a Visual Studio Code window with a dark theme. The title bar reads "Demo03-ButtonGroup.js - Lesson01-Material UI - Visual Studio Code". The left sidebar has icons for file operations, search, and other tools. The main editor area displays a JavaScript file named "BasicButtonGroup.js" which contains code for creating a button group with three buttons. The status bar at the bottom shows file statistics (0 lines, 0 changes), file location (Ln 13, Col 2), encoding (UTF-8), and file type (JavaScript). It also shows system information like weather (25°C Partly sunny) and system status (11:44 AM 7/3/2023).

Checkbox

- Checkboxes allow the user to select one or more items from a set.
- Checkboxes can be used to turn an option on or off.
- If you have multiple options appearing in a list, you can preserve space by using checkboxes instead of on/off switches. If you have a single option, avoid using a checkbox and use an on/off switch instead.



Demo04-CheckBox.js - Lesson01-Material UI - Visual Studio Code

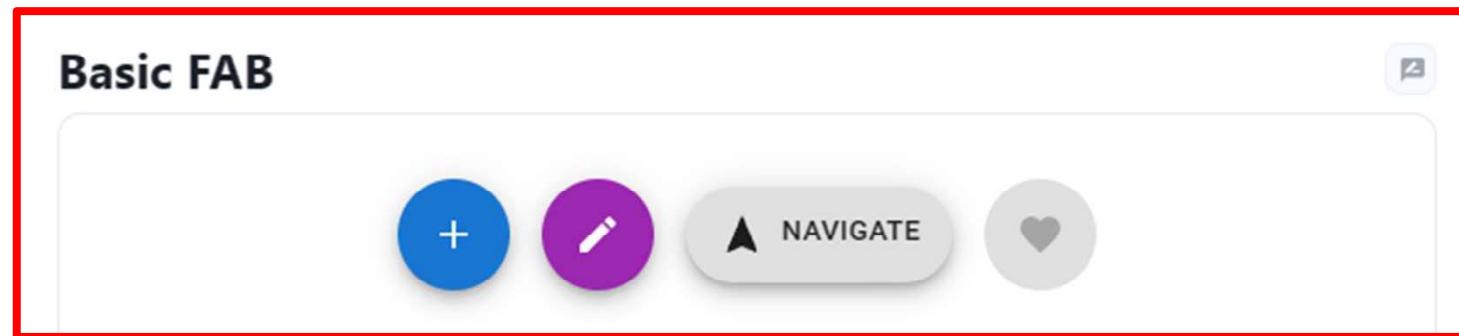
material-ui-demo > src > Demo04-CheckBox.js > Checkboxes

```
1 import * as React from 'react';
2 import Checkbox from '@mui/material/Checkbox';
3
4 const label = { inputProps: { 'aria-label': 'Checkbox demo' } };
5
6 export default function Checkboxes() {
7   return (
8     <div>
9       <Checkbox {...label} defaultChecked />
10      <Checkbox {...label} />
11      <Checkbox {...label} disabled />
12      <Checkbox {...label} disabled checked />
13     </div>
14   );
15 }
```

Ln 12, Col 44 (7 selected) Spaces: 4 UTF-8 CRLF JavaScript Go Live 11:59 AM 7/3/2023

Floating Action Button

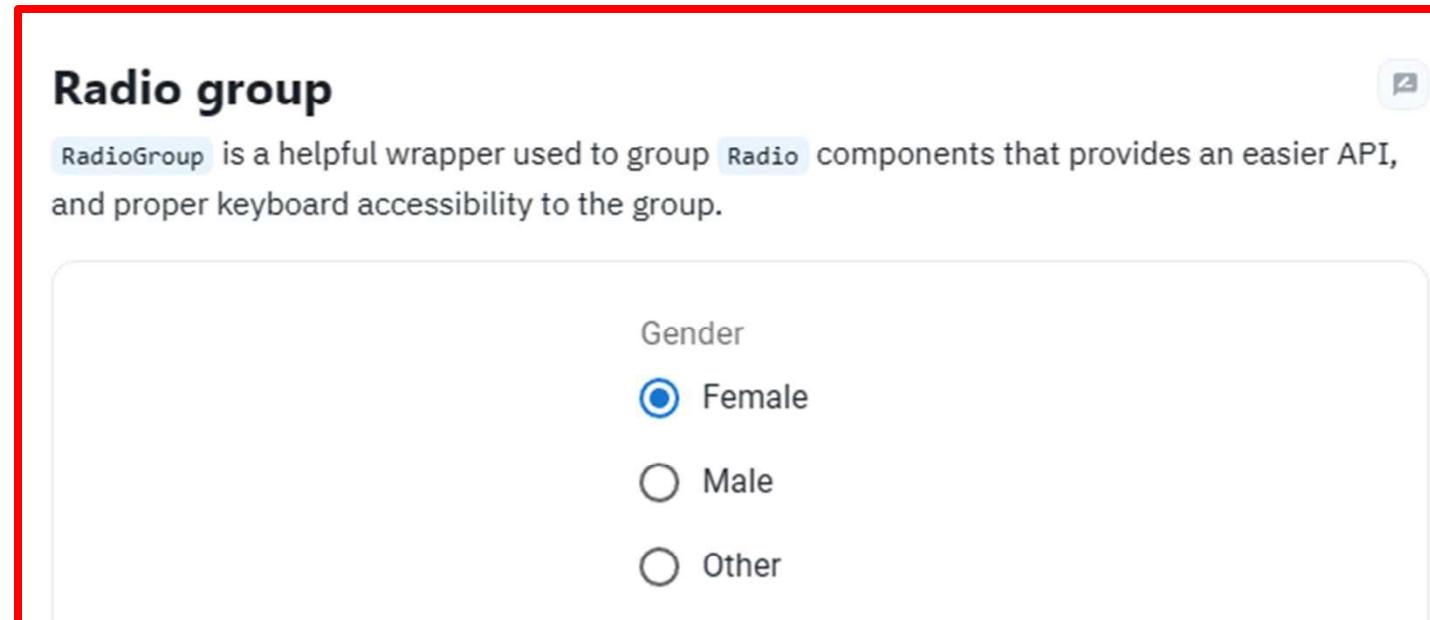
- A Floating Action Button (FAB) performs the primary, or most common, action on a screen.
- A floating action button appears in front of all screen content, typically as a circular shape with an icon in its center. FABs come in two types: regular, and extended.
- Only use a FAB if it is the most suitable way to present a screen's primary action. Only one component is recommended per screen to represent the most common action.

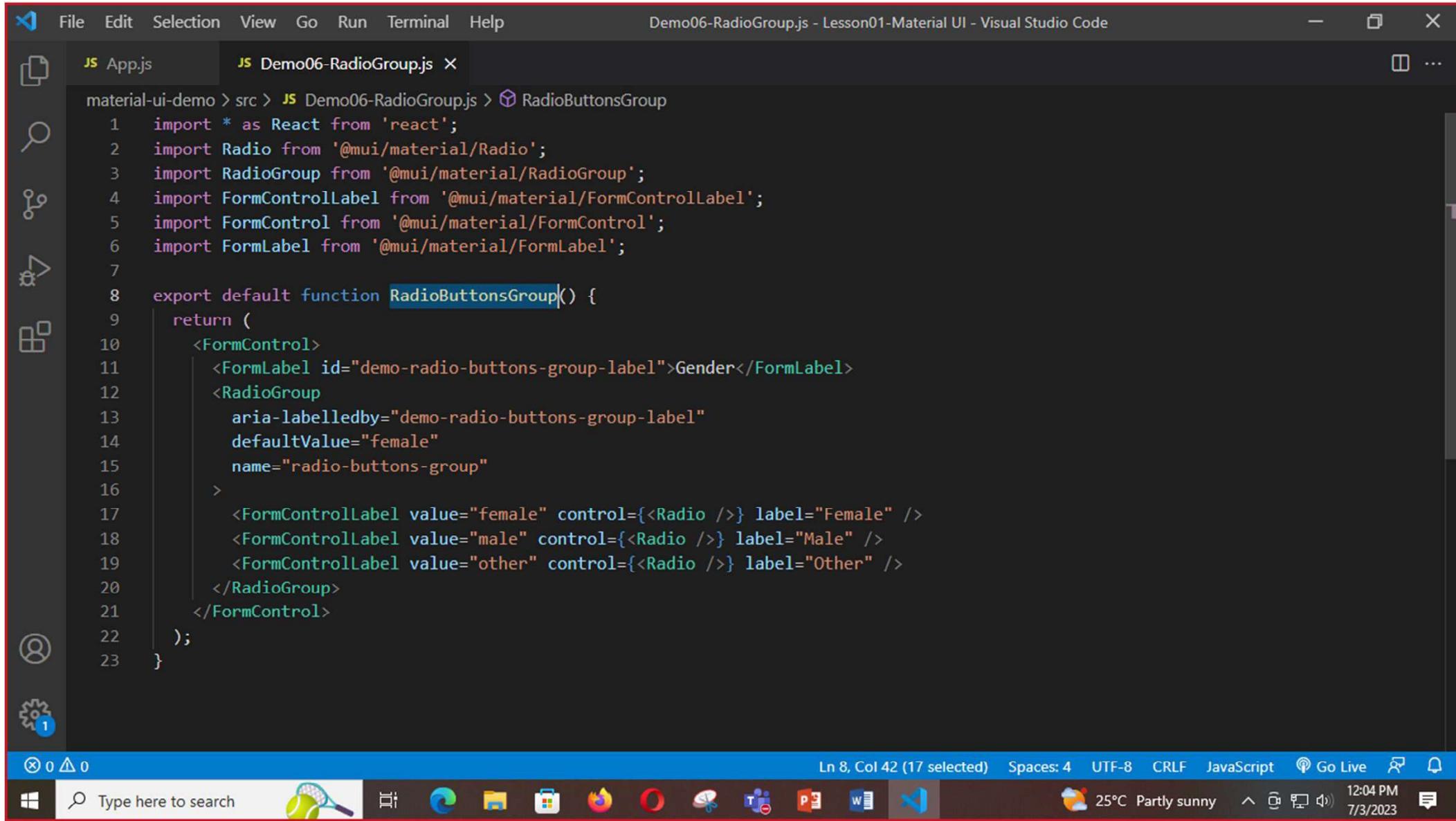


```
material-ui-demo > src > Demo05-FAB.js > FloatingActionButtons
  3 import Fab from '@mui/material/Fab';
  4 import AddIcon from '@mui/icons-material/Add';
  5 import EditIcon from '@mui/icons-material/Edit';
  6 import FavoriteIcon from '@mui/icons-material/Favorite';
  7 import NavigationIcon from '@mui/icons-material/Navigation';
  8
  9 export default function FloatingActionButtons() {
10   return (
11     <Box sx={{ '&': { not(style): { m: 1 } } }}>
12       <Fab color="primary" aria-label="add">
13         <AddIcon />
14       </Fab>
15       <Fab color="secondary" aria-label="edit">
16         <EditIcon />
17       </Fab>
18       <Fab variant="extended">
19         <NavigationIcon sx={{ mr: 1 }} />
20         Navigate
21       </Fab>
22       <Fab disabled aria-label="like">
23         <FavoriteIcon />
24       </Fab>
25     </Box>
26   );
27 }
```

Radio Group

- The Radio Group allows the user to select one option from a set.
- Use radio buttons when the user needs to see all available options. If available options can be collapsed, consider using a [Select component](#) because it uses less space.
- Radio buttons should have the most commonly used option selected by default.





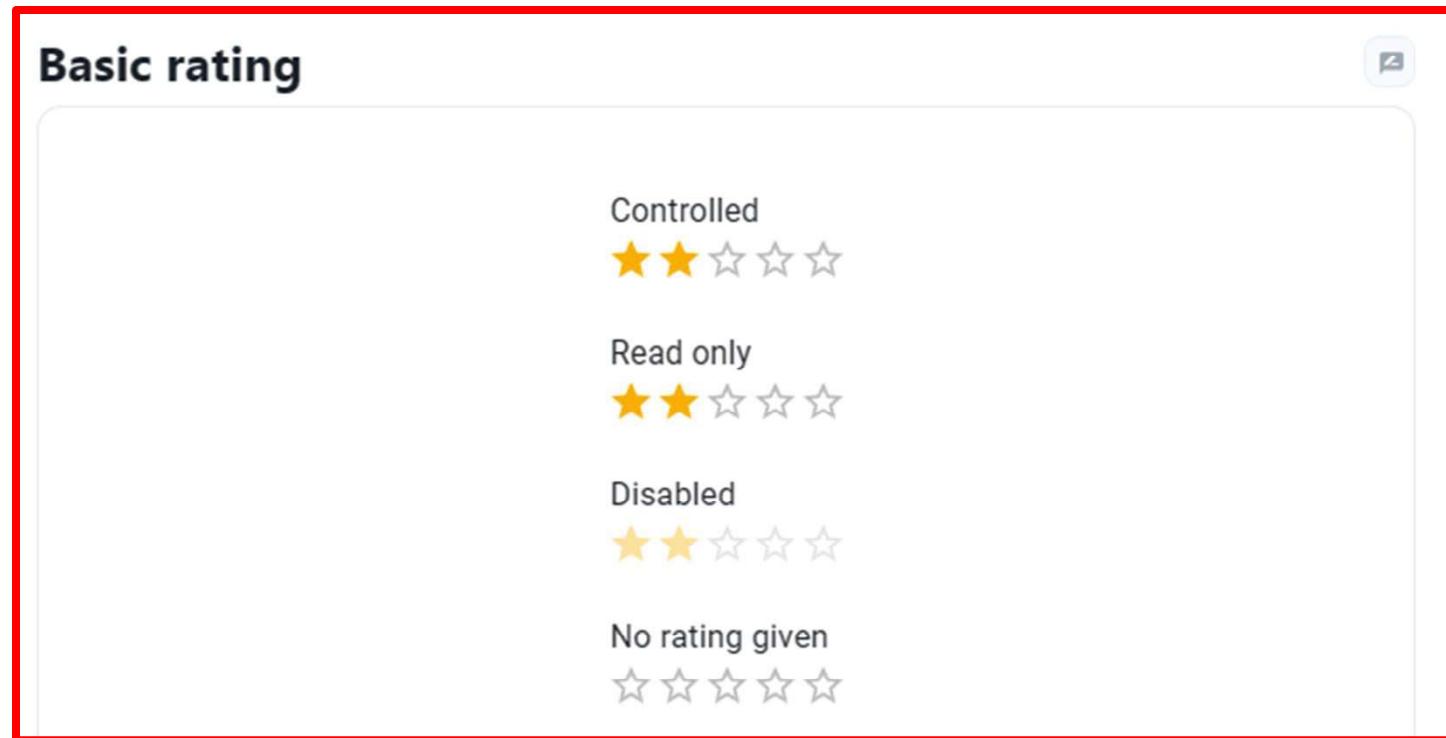
The screenshot shows a Visual Studio Code window with the title "Demo06-RadioGroup.js - Lesson01-Material UI - Visual Studio Code". The left sidebar has icons for file operations, search, and other tools. The main editor pane displays the following code:

```
material-ui-demo > src > Demo06-RadioGroup.js > RadioButtonsGroup
1 import * as React from 'react';
2 import Radio from '@mui/material/Radio';
3 import RadioGroup from '@mui/material/RadioGroup';
4 import FormControlLabel from '@mui/material/FormControlLabel';
5 import FormControl from '@mui/material/FormControl';
6 import FormLabel from '@mui/material/FormLabel';
7
8 export default function RadioButtonsGroup() {
9     return (
10         <FormControl>
11             <FormLabel id="demo-radio-buttons-group-label">Gender</FormLabel>
12             <RadioGroup
13                 aria-labelledby="demo-radio-buttons-group-label"
14                 defaultValue="female"
15                 name="radio-buttons-group"
16             >
17                 <FormControlLabel value="female" control={<Radio />} label="Female" />
18                 <FormControlLabel value="male" control={<Radio />} label="Male" />
19                 <FormControlLabel value="other" control={<Radio />} label="Other" />
20             </RadioGroup>
21         </FormControl>
22     );
23 }
```

The status bar at the bottom shows: 0△0, Ln 8, Col 42 (17 selected), Spaces: 4, UTF-8, CRLF, JavaScript, Go Live, 25°C Partly sunny, 12:04 PM, 7/3/2023.

Rating

- Ratings provide insight regarding others' opinions and experiences, and can allow the user to submit a rating of their own.



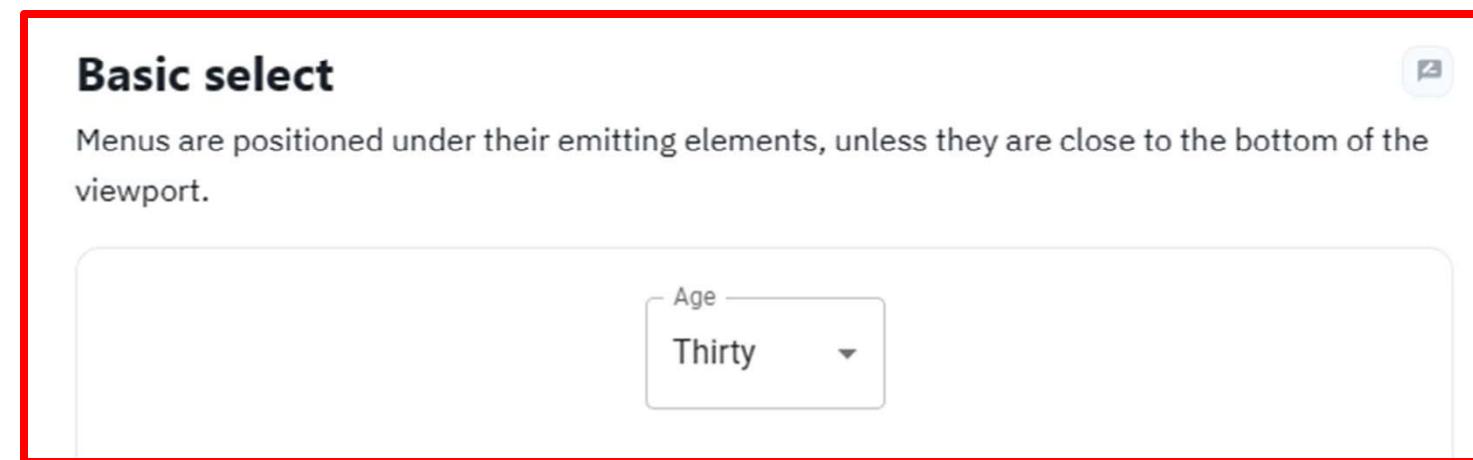
A screenshot of the Visual Studio Code interface. The title bar shows "Demo07-Rating.js - Lesson01-Material UI - Visual Studio Code". The left sidebar has icons for file operations, search, navigation, and settings. The main editor area displays the following code:

```
9  return (
10    <Box
11      sx={{
12        '& > legend': { mt: 2 },
13      }}
14    >
15      <Typography component="legend">Controlled</Typography>
16      <Rating
17        name="simple-controlled"
18        value={value}
19        onChange={(event, newValue) => {
20          setValue(newValue);
21        }}
22      />
23      <Typography component="legend">Read only</Typography>
24      <Rating name="read-only" value={value} readOnly />
25      <Typography component="legend">Disabled</Typography>
26      <Rating name="disabled" value={value} disabled />
27      <Typography component="legend">No rating given</Typography>
28      <Rating name="no-value" value={null} />
29    </Box>
```

The status bar at the bottom shows "Ln 7, Col 57 (4 selected)" and "Spaces: 4".

Select

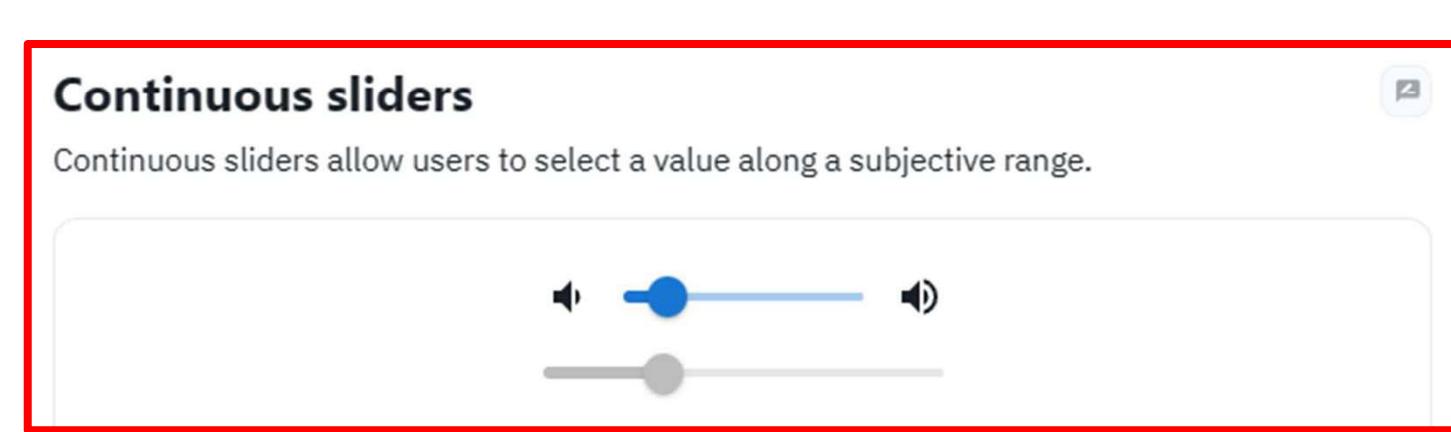
- Select components are used for collecting user provided information from a list of options.



A screenshot of the Visual Studio Code interface displaying a file named `Demo08-Select.js`. The code implements a `BasicSelect` component using the Material UI `Select` component. The component uses the `useState` hook to manage the age state and the `useEffect` hook to handle the change event. The `MenuItem` component is used to list the available age options: Ten, Twenty, and Thirty.

```
material-ui-demo > src > Demo08-Select.js > BasicSelect
  6 import Select from '@mui/material/Select';
  7
  8 export default function BasicSelect() {
  9   const [age, setAge] = React.useState('');
 10
 11   const handleChange = (event) => {
 12     setAge(event.target.value);
 13   };
 14
 15   return (
 16     <Box sx={{ minWidth: 120 }}>
 17       <FormControl fullWidth>
 18         <InputLabel id="demo-simple-select-label">Age</InputLabel>
 19         <Select
 20           labelId="demo-simple-select-label"
 21           id="demo-simple-select"
 22           value={age}
 23           label="Age"
 24           onChange={handleChange}
 25         >
 26           <MenuItem value={10}>Ten</MenuItem>
 27           <MenuItem value={20}>Twenty</MenuItem>
 28           <MenuItem value={30}>Thirty</MenuItem>
 29         </Select>
 30       </FormControl>
 31     </Box>
 32   );
}
```

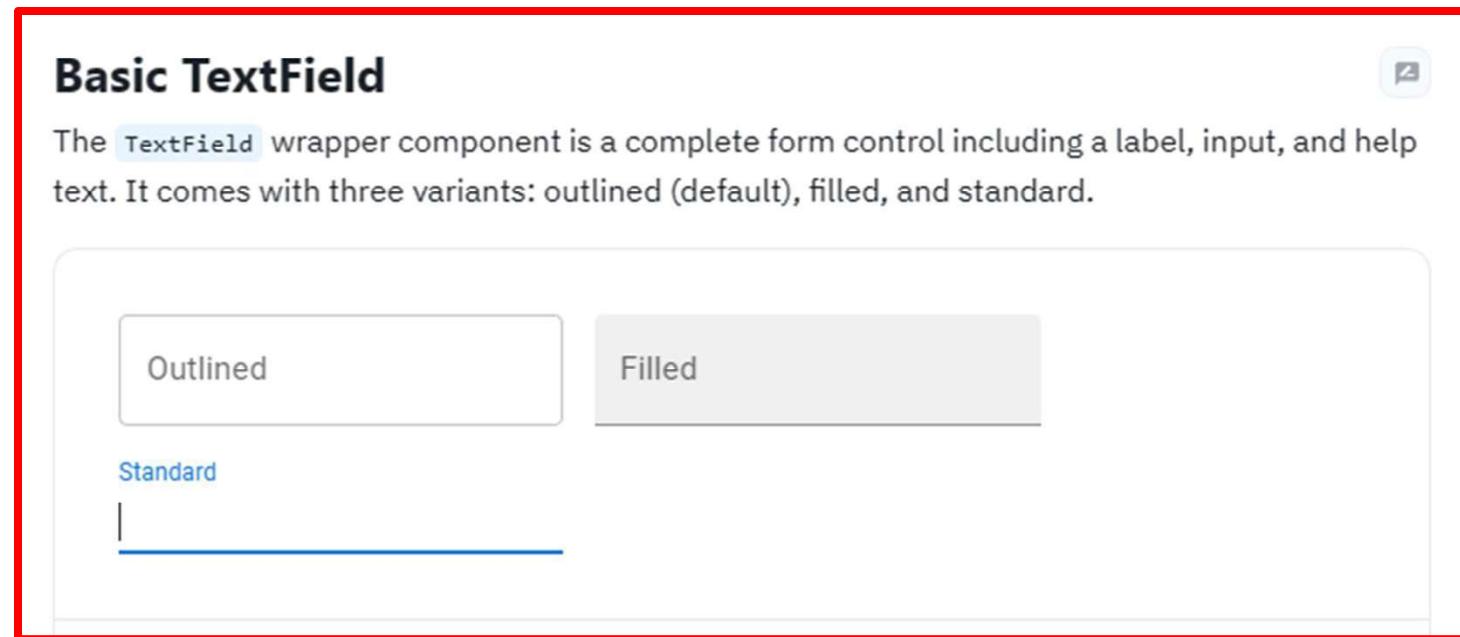
- Sliders allow users to make selections from a range of values.
- Sliders reflect a range of values along a bar, from which users may select a single value. They are ideal for adjusting settings such as volume, brightness, or applying image filters.



```
material-ui-demo > src > Demo09-Slider.js > ContinuousSlider
1 import * as React from 'react';
2 import Box from '@mui/material/Box';
3 import Stack from '@mui/material/Stack';
4 import Slider from '@mui/material/Slider';
5 import VolumeDown from '@mui/icons-material/VolumeDown';
6 import VolumeUp from '@mui/icons-material/VolumeUp';
7
8 export default function ContinuousSlider() {
9   const [value, setValue] = React.useState(30);
10
11   const handleChange = (event, newValue) => {
12     setValue(newValue);
13   };
14
15   return (
16     <Box sx={{ width: 200 }}>
17       <Stack spacing={2} direction="row" sx={{ mb: 1 }} alignItems="center">
18         <VolumeDown />
19         <Slider aria-label="Volume" value={value} onChange={handleChange} />
20         <VolumeUp />
21       </Stack>
22       <Slider disabled defaultValue={30} aria-label="Disabled slider" />
23     </Box>
24   );
25 }
```

TextField

- Text Fields let users enter and edit text.
- Text fields allow users to enter text into a UI. They typically appear in forms and dialogs.



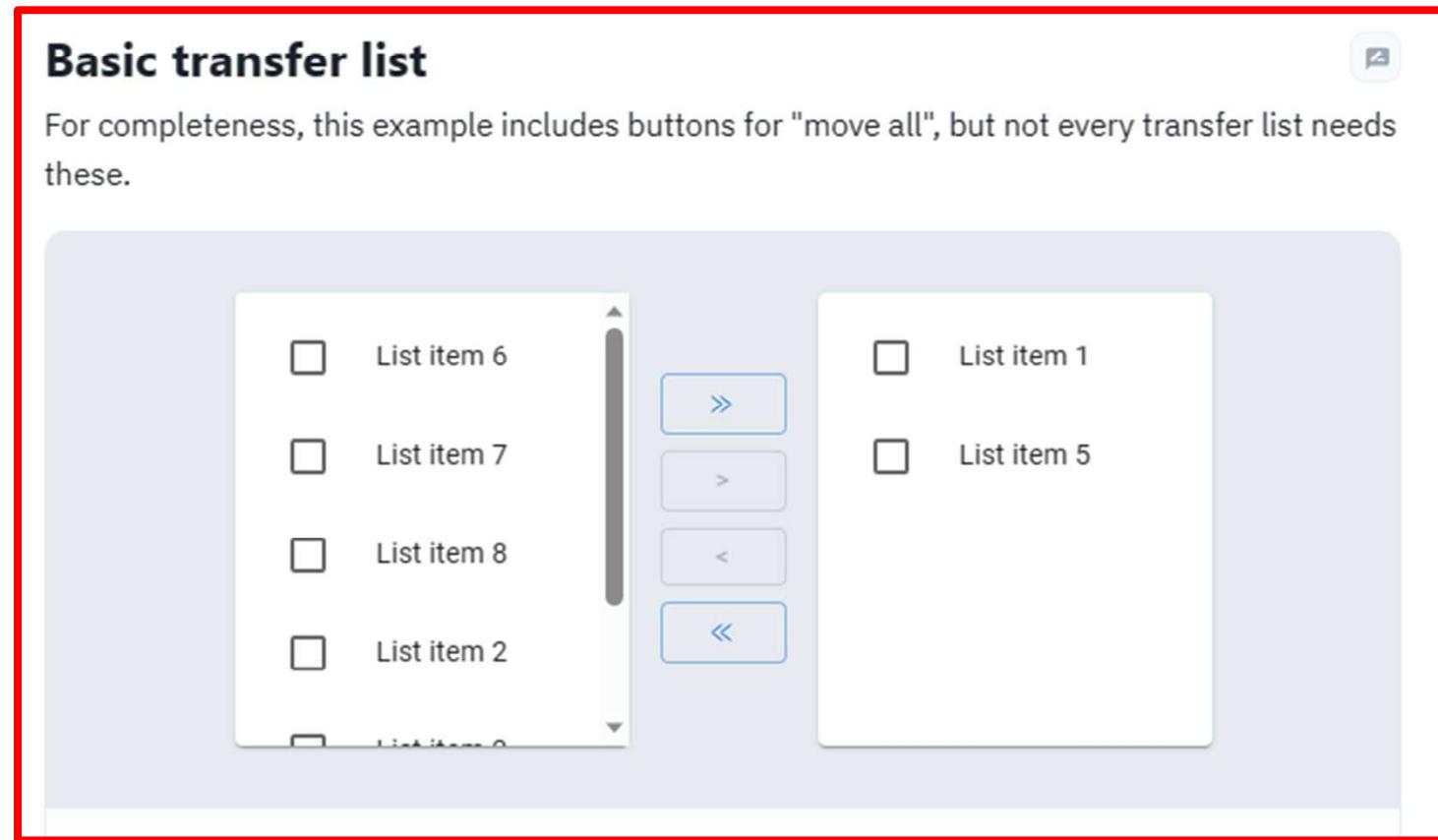
A screenshot of the Visual Studio Code interface displaying a file named `Demo10-TextField.js`. The code implements a `BasicTextFields` component using the `@mui/material` library. The component uses a `Box` component with a `form` component prop and a `sx` style prop to create three different text field variants: outlined, filled, and standard. The code is as follows:

```
material-ui-demo > src > Demo10-TextField.js > BasicTextFields
1 import * as React from 'react';
2 import Box from '@mui/material/Box';
3 import TextField from '@mui/material/TextField';
4
5 export default function BasicTextFields() {
6   return (
7     <Box
8       component="form"
9       sx={{
10         '& > :not(style)': { m: 1, width: '25ch' },
11       }}
12       noValidate
13       autoComplete="off"
14     >
15       <TextField id="outlined-basic" label="Outlined" variant="outlined" />
16       <TextField id="filled-basic" label="Filled" variant="filled" />
17       <TextField id="standard-basic" label="Standard" variant="standard" />
18     </Box>
19   );
20 }
```

The status bar at the bottom shows the file is at line 20, column 2, with 4 spaces, using UTF-8 encoding, and is a JavaScript file. It also displays the current weather as 26°C Partly sunny and the date/time as 12:37 PM 7/3/2023.

Transfer List / Shuttle

- A Transfer List (or "shuttle") enables the user to move one or more list items between lists.



The screenshot shows the Visual Studio Code interface with the title bar "Demo11-TransferList.js - Lesson01-Material UI - Visual Studio Code". The left sidebar is the Explorer view, showing a folder named "LESSON01-MATERIAL UI" containing various JavaScript files and other project files like ".gitignore", "package-lock.json", and "package.json". The file "Demo11-TransferList.js" is currently selected and open in the main editor area. The code implements a TransferList component using React and Material-UI libraries, including functions for calculating differences between two arrays and a functional component definition.

```
material-ui-demo > src > JS Demo11-TransferList.js > TransferList
1 import * as React from 'react';
2 import Grid from '@mui/material/Grid';
3 import List from '@mui/material/List';
4 import ListItem from '@mui/material/ListItem';
5 import ListItemIcon from '@mui/material/ListItemIcon';
6 importListItemText from '@mui/material/ListItemText';
7 import Checkbox from '@mui/material/Checkbox';
8 import Button from '@mui/material/Button';
9 import Paper from '@mui/material/Paper';
10
11 function not(a, b) {
12   return a.filter((value) => b.indexOf(value) === -1);
13 }
14
15 function intersection(a, b) {
16   return a.filter((value) => b.indexOf(value) !== -1);
17 }
18
19 export default function TransferList() {
20   const [checked, setChecked] = React.useState([]);
21   const [left, setLeft] = React.useState([0, 1, 2, 3]);
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Demo11-TransferList.js - Lesson01-Material UI - Visual Studio Code.
- Explorer:** Shows a tree view of files under "LESSON01-MATERIAL UI". The "Demo11-TransferList.js" file is selected.
- Editor:** Displays the code for "Demo11-TransferList.js". The code uses React hooks like useState and useEffect to manage state for a transfer list component.
- Bottom Status Bar:** ShowsLn 143, Col 2, Spaces: 4, UTF-8, CRLF, JavaScript, Go Live, and system icons.
- Taskbar:** Shows the Windows Start button, a search bar, and pinned application icons for Edge, File Explorer, File History, Task View, Microsoft Edge, Microsoft Word, Microsoft Excel, and Microsoft Teams.

```
export default function TransferList() {
  const [checked, setChecked] = React.useState([]);
  const [left, setLeft] = React.useState([0, 1, 2, 3]);
  const [right, setRight] = React.useState([4, 5, 6, 7]);

  const leftChecked = intersection(checked, left);
  const rightChecked = intersection(checked, right);

  const handleToggle = (value) => () => {
    const currentIndex = checked.indexOf(value);
    const newChecked = [...checked];

    if (currentIndex === -1) {
      newChecked.push(value);
    } else {
      newChecked.splice(currentIndex, 1);
    }

    setChecked(newChecked);
  };
}
```

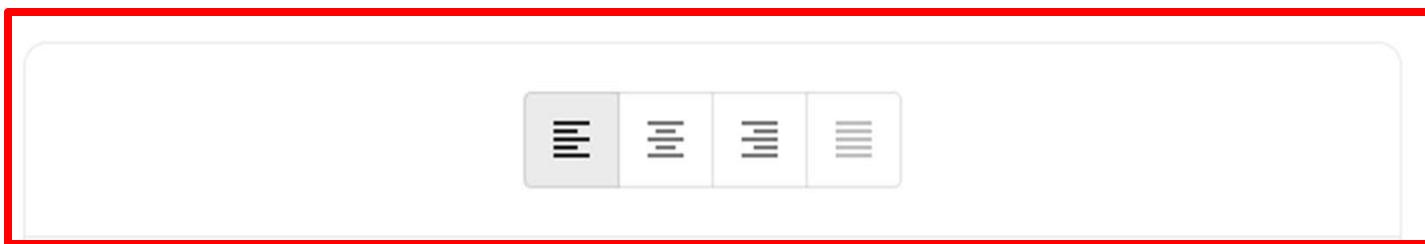
A screenshot of the Visual Studio Code interface. The title bar shows "Demo11-TransferList.js - Lesson01-Material UI - Visual Studio Code". The left sidebar has icons for file operations, search, navigation, and other tools. The main editor area displays the following code:

```
material-ui-demo > src > Demo11-TransferList.js > TransferList
64   <List dense component="div" role="list">
65     {items.map((value) => {
66       const labelId = `transfer-list-item-${value}-label`;
67
68       return (
69         <ListItemIcon
70           key={value}
71           role="listitem"
72           button
73           onClick={handleToggle(value)}
74         >
75           <ListItemIconIcon>
76             <Checkbox
77               checked={checked.indexOf(value) !== -1}
78               tabIndex={-1}
79               disableRipple
80               inputProps={{
81                 'aria-labelledby': labelId,
82               }}
83             />
84           </ListItemIconIcon>
```

The status bar at the bottom shows "Ln 143, Col 2" and "Spaces: 4". It also includes icons for Go Live, a bell, and a gear with a '1'.

Toggle Button

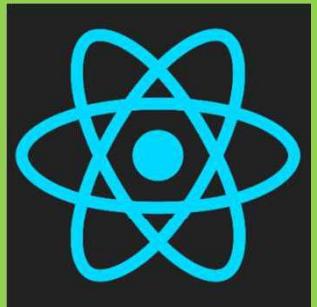
- A Toggle Button can be used to group related options.
- To emphasize groups of related Toggle buttons, a group should share a common container. The ToggleButtonGroup controls the selected state of its child buttons when given its own value prop.



```
File Edit Selection View Go Run Terminal Help Demo12-ToggleButton.js - Lesson01-Material UI - Visual Studio Code
JS App.js JS Demo11-TransferList.js JS Demo12-ToggleButton.js X
material-ui-demo > src > JS Demo12-ToggleButton.js > ⚡ ToggleButtons
9  export default function ToggleButtons() {
10    const [alignment, setAlignment] = React.useState('left');
11
12    const handleAlignment = (event, newAlignment) => {
13      setAlignment(newAlignment);
14    };
15
16    return (
17      <ToggleButtonGroup
18        value={alignment}
19        exclusive
20        onChange={handleAlignment}
21        aria-label="text alignment"
22      >
23        <ToggleButton value="left" aria-label="left aligned">
24          <FormatAlignLeftIcon />
25        </ToggleButton>
26        <ToggleButton value="center" aria-label="centered">
27          <FormatAlignCenterIcon />
28        </ToggleButton>
29        <ToggleButton value="right" aria-label="right aligned">
```

Ln 32, Col 68 (8 selected) Spaces: 4 CRLF JavaScript ⚡ Go Live 🔍 12:52 PM 7/3/2023

Type here to search



Data Display

- Avatars are found throughout material design with uses in everything from tables to dialog menus.

Image avatars

Image avatars can be created by passing standard `img` props `src` or `srcSet` to the component.



Letter avatars

Avatars containing simple characters can be created by passing a string as `children`.



Demo01-Avatar.js - Lesson01-Material UI - Visual Studio Code

```
material-ui-demo > src > Data-Display > Demo01-Avatar.js > ImageAvatars
1 import * as React from 'react';
2 import Avatar from '@mui/material/Avatar';
3 import Stack from '@mui/material/Stack';
4 import Image1 from './images/Image1.jpg'
5
6 export default function ImageAvatars() {
7   return (
8     <Stack direction="row" spacing={2}>
9       {/* <Avatar alt="Remy Sharp" src="Image1.jpg" /> */}
10      <Avatar alt="Remy Sharp" src={Image1} />
11      <Avatar alt="Travis Howard" src="/Data-Display/images/2.jpg" />
12      <Avatar alt="Cindy Baker" src="/Data-Display/images/3.jpg" />
13    </Stack>
14  );
15}
```

Ln 15, Col 2 Spaces: 4 UTF-8 CRLF JavaScript Go Live 1:15 PM 7/3/2023

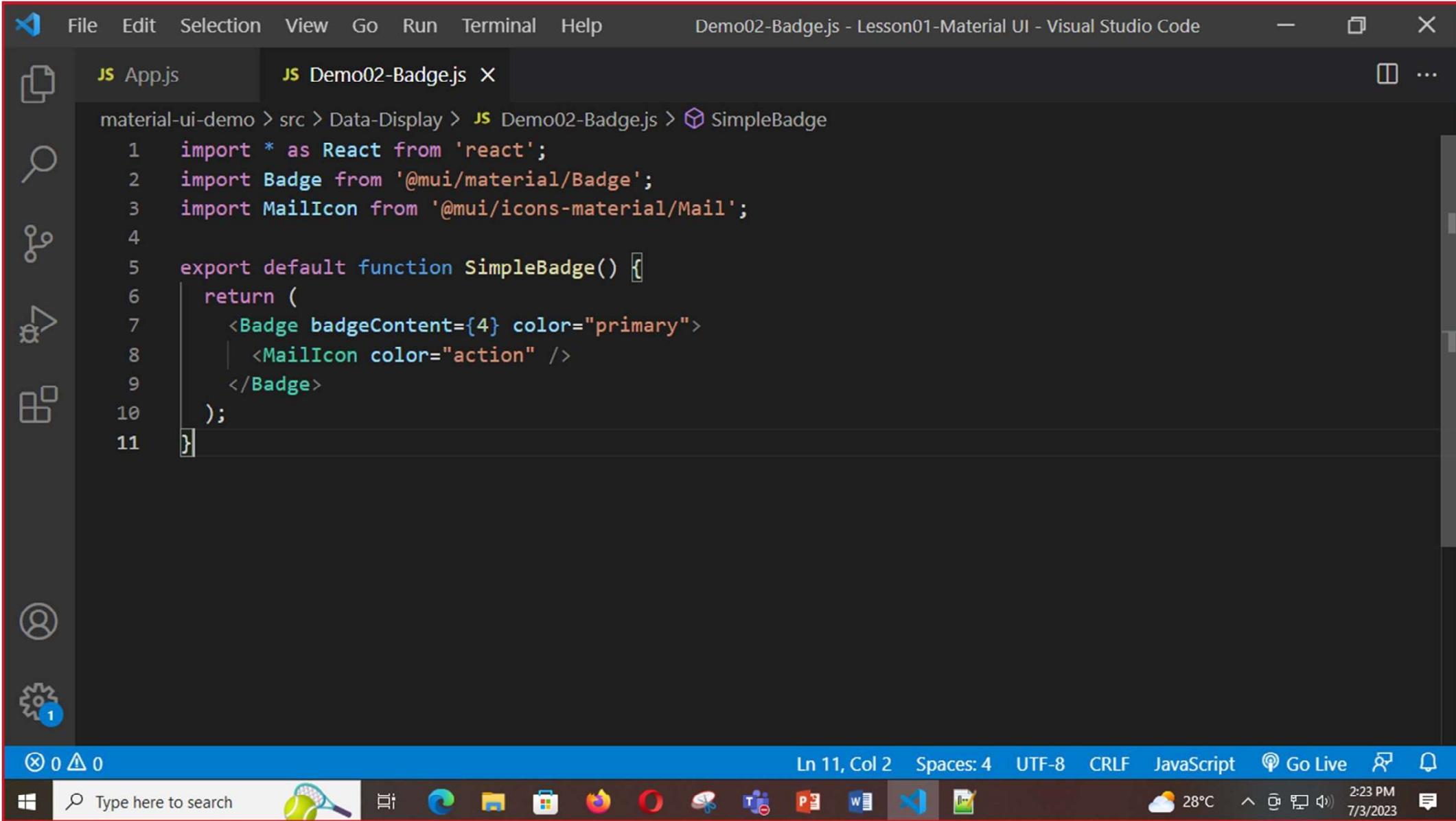
Badge

- Badge generates a small badge to the top-right of its child(ren).

Basic badge



Examples of badges containing text, using primary and secondary colors. The badge is applied to its children.

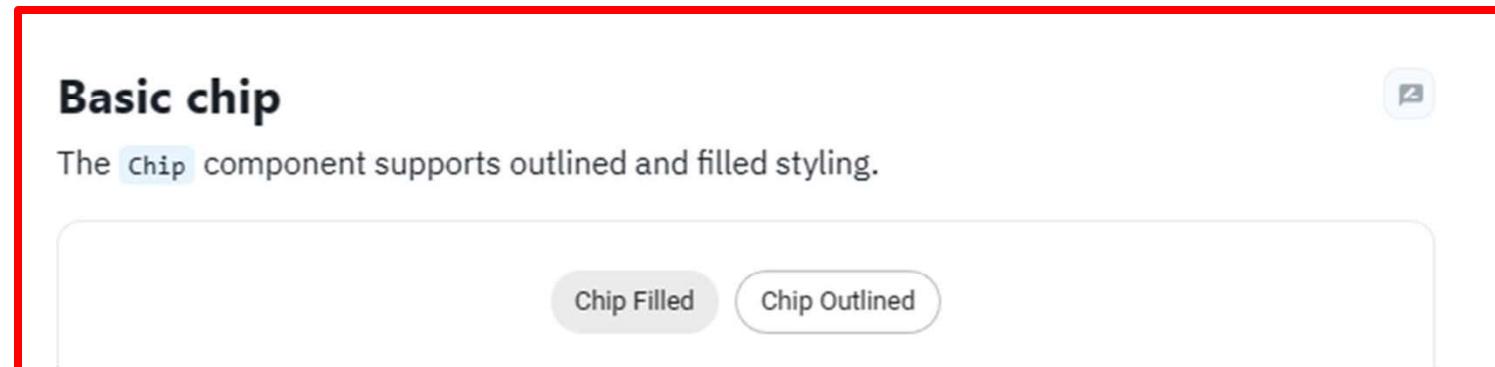


```
material-ui-demo > src > Data-Display > Demo02-Badge.js > SimpleBadge
1 import * as React from 'react';
2 import Badge from '@mui/material/Badge';
3 import MailIcon from '@mui/icons-material/Mail';
4
5 export default function SimpleBadge() {
6   return (
7     <Badge badgeContent={4} color="primary">
8       <MailIcon color="action" />
9     </Badge>
10  );
11}
```

The screenshot shows a Visual Studio Code window with a dark theme. The title bar reads "Demo02-Badge.js - Lesson01-Material UI - Visual Studio Code". The left sidebar has icons for file operations like copy, search, and refresh. The main editor area displays a JavaScript file named "SimpleBadge.js" which contains code for a Material-UI badge component. The code imports React, Badge, and MailIcon from Material-UI. It defines a function that returns a Badge component with a badgeContent of 4 and a primary color, containing a MailIcon with an action color. The bottom status bar shows file statistics (0 errors, 0 warnings), file location (Ln 11, Col 2), encoding (UTF-8), and file type (JavaScript). It also includes a "Go Live" button and system status icons for weather, battery, and date/time.

Chips

- Chips are compact elements that represent an input, attribute, or action.
- Chips allow users to enter information, make selections, filter content, or trigger actions.
- While included here as a standalone component, the most common use will be in some form of input, so some of the behavior demonstrated here is not shown in context.



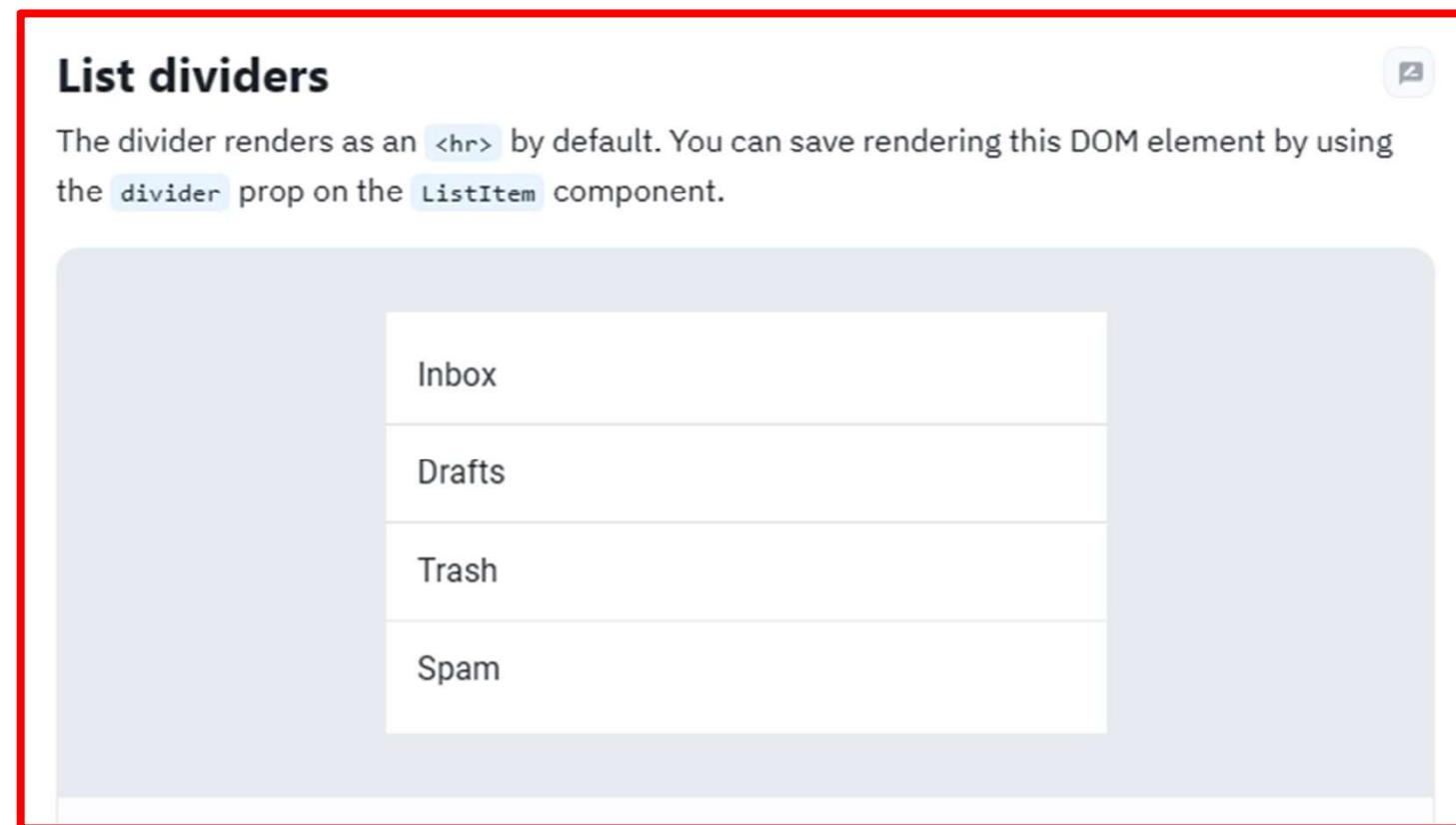
A screenshot of the Visual Studio Code interface. The title bar reads "Demo03-Chips.js - Lesson01-Material UI - Visual Studio Code". The left sidebar shows file icons for "App.js", "Demo02-Badge.js", and "Demo03-Chips.js" (which is currently active). The main editor area displays the following code:

```
material-ui-demo > src > Data-Display > Demo03-Chips.js > BasicChips
1 import * as React from 'react';
2 import Chip from '@mui/material/Chip';
3 import Stack from '@mui/material/Stack';
4
5 export default function BasicChips() {
6   return (
7     <Stack direction="row" spacing={1}>
8       <Chip label="Chip Filled" />
9       <Chip label="Chip Outlined" variant="outlined" />
10    </Stack>
11  );
12}
```

The status bar at the bottom shows "Ln 12, Col 2" and "Spaces: 4". The taskbar at the bottom includes icons for File Explorer, Task List, and several Microsoft Office applications.

Divider

- A divider is a thin line that groups content in lists and layouts.
- Dividers separate content into clear groups.



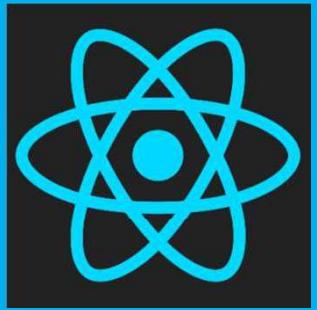
Demo04-Divider.js - Lesson01-Material UI - Visual Studio Code

material-ui-demo > src > Data-Display > Demo04-Divider.js > ListDividers

```
11  };
12
13 export default function ListDividers() {
14   return (
15     <List sx={style} component="nav" aria-label="mailbox folders">
16       <ListItem button>
17         <ListItemText primary="Inbox" />
18       </ListItem>
19       <Divider />
20       <ListItem button divider>
21         <ListItemText primary="Drafts" />
22       </ListItem>
23       <ListItem button>
24         <ListItemText primary="Trash" />
25       </ListItem>
26       <Divider light />
27       <ListItem button>
28         <ListItemText primary="Spam" />
29       </ListItem>
30     </List>
```

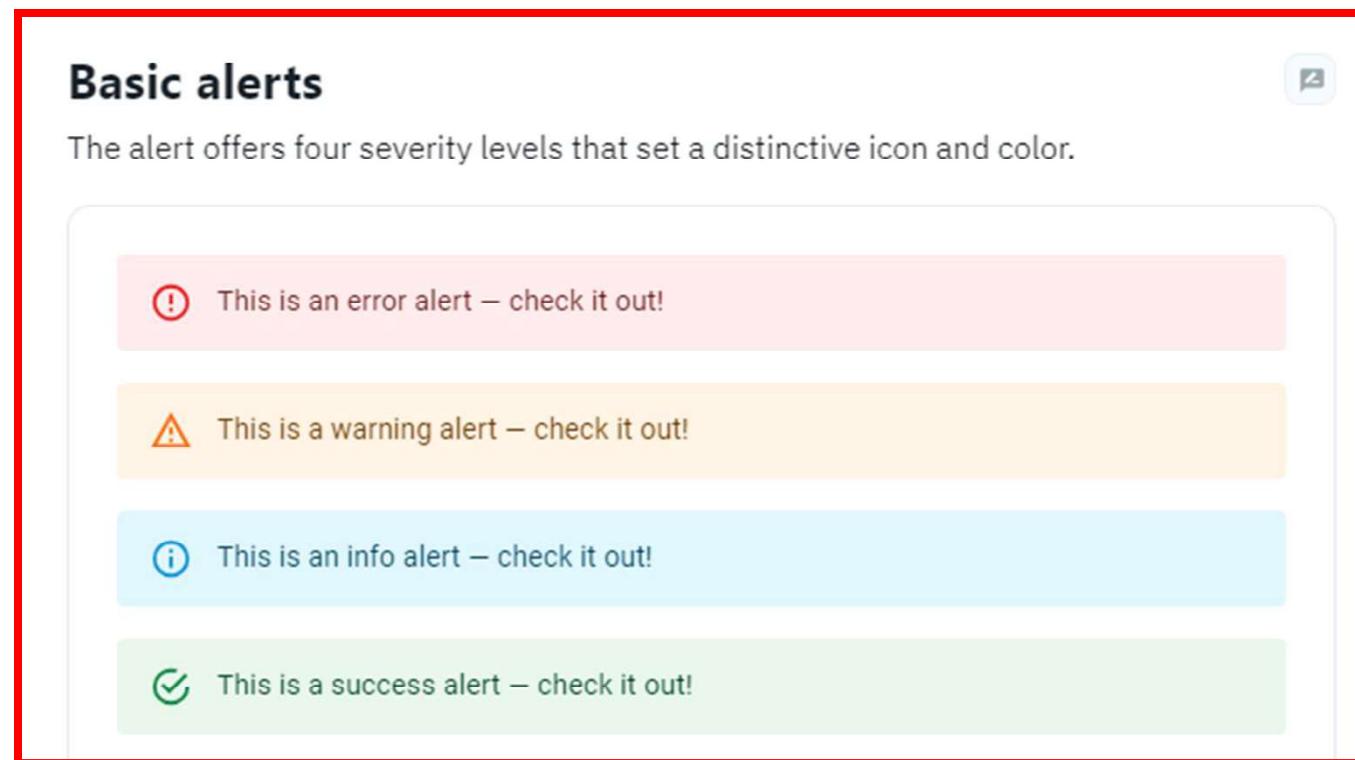
Ln 32, Col 2 Spaces: 4 UTF-8 CRLF JavaScript ⚡ Go Live 🔍 📲 2:32 PM 7/3/2023

- Guidance and suggestions for using icons with Material UI.
- Material UI provides icon support in three ways:
 1. Standardized [Material Icons](#) exported as React components (SVG icons).
 2. With the [SvgIcon](#) component, a React wrapper for custom SVG icons.
 3. With the [Icon](#) component, a React wrapper for custom font icons.



Feedback

- An alert displays a short, important message in a way that attracts the user's attention without interrupting the user's task.





The screenshot shows a Microsoft Edge browser window displaying a React application. The title bar says "test-react-app". The left sidebar has icons for file, search, and other developer tools. The main content area shows two tabs: "Demo01-Alert.js" (selected) and "App.js". The code in "Demo01-Alert.js" is as follows:

```
src > Feedback > Demo01-Alert.js > BasicAlerts
1 import * as React from 'react';
2 import Alert from '@mui/material/Alert';
3 import Stack from '@mui/material/Stack';
4
5 export default function BasicAlerts() {
6   return (
7     <Stack sx={{ width: '100%' }} spacing={2}>
8       <Alert severity="error">This is an error alert - check it out!</Alert>
9       <Alert severity="warning">This is a warning alert - check it out!</Alert>
10      <Alert severity="info">This is an info alert - check it out!</Alert>
11      <Alert severity="success">This is a success alert - check it out!</Alert>
12    </Stack>
13  );
14}
```

The browser status bar at the bottom shows: Ln 5, Col 31, Spaces: 4, UTF-8, CRLF, {}, JavaScript, Go Live, 4:12 PM, 7/3/2023.

Backdrop

- The Backdrop component narrows the user's focus to a particular element on the screen.
- The Backdrop signals a state change within the application and can be used for creating loaders, dialogs, and more. In its simplest form, the Backdrop component will add a dimmed layer over your application.

The screenshot shows a Microsoft Edge browser window with a red border. The address bar displays the URL <https://mui.com/material-ui/react-backdrop/>. The page content is from the Material-UI website, specifically the React Backdrop documentation. On the left, a sidebar lists various UI components: Chip, Divider, Icons, Material Icons, List, Table, Tooltip, Typography, FEEDBACK, Alert, Backdrop (which is highlighted), Dialog, Progress, Skeleton, Snackbar, SURFACES, Accordion, App Bar, Card, and Paper. The main content area features a heading "Example" and a paragraph explaining that the demo shows a basic Backdrop with a Circular Progress component. Below this is a code editor showing the React component code for the example. A sidebar on the right contains a message about MUI's stance on the Ukrainian invasion and links to help resources. The bottom of the screen shows the Windows taskbar with the Edge icon, a search bar, and system status icons.

```
<Button onClick={handleOpen}>Show backdrop</Button>
<Backdrop
  sx={{ color: '#fff', zIndex: (theme) => theme.zIndex.drawer + 1 }}
  open={open}
  onClick={ handleClose }
>
  <CircularProgress color="inherit" />
</Backdrop>
```

Dialog

- Dialogs inform users about a task and can contain critical information, require decisions, or involve multiple tasks.
- A Dialog is a type of **moda** window that appears in front of app content to provide critical information or ask for a decision. Dialogs disable all app functionality when they appear, and remain on screen until confirmed, dismissed, or a required action has been taken.
- Dialogs are purposefully interruptive, so they should be used sparingly.

React D X | React A X | Amazon X | Auto H X | CARMA X | CARMA X | AUTO A X | Autoist X | Autoist X | New tab X | React A X | +

Sign in  ... 

Chip
Divider
Icons
Material Icons
List
Table
Tooltip
Typography

FEEDBACK

Alert
Backdrop
Dialog
Progress
Skeleton
Snackbar

SURFACES

Accordion
App Bar
Card
Paper

<https://mui.com/material-ui/react-dialog/>

MUIX v6 is out! Discover what's new and get started now!

Search... Ctrl+K

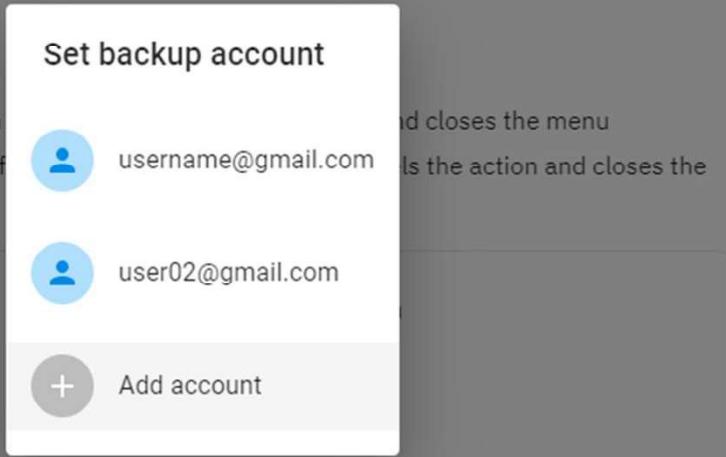
3

Basic dialog

Simple dialogs can provide additional details or actions about a list item. For example, they can display avatars, icons, clarifying subtext, or orthogonal actions (such as adding an account).

Touch mechanics:

- Choosing an option
- Touching outside of dialog



username@gmail.com

user02@gmail.com

Add account

Selected: {selectedValue}

<Typography variant="subtitle1" component="div">

</Typography>

<Button variant="outlined" onClick={handleClickOpen}>

28°C Mostly cloudy 4:17 PM 7/3/2023

MUI stands in solidarity with the Ukrainian people against the Russian invasion.

Find out how you can help.

CONTENTS

Basic dialog
Alerts
Transitions
Form dialogs
Customization
Full-screen dialogs
Optional sizes
Responsive full-screen
Confirmation dialogs
Draggable dialog
Scrolling long content
Performance
Limitations

MENTORLABS™

The screenshot shows a Microsoft Edge browser window with a dark theme. The title bar says "test-react-app". The address bar also shows "test-react-app". The page content displays a list of emails under the heading "Set backup account". The code for "Demo03-Dialog.js" is visible in the editor, showing the implementation of a dialog component using Material UI's List and ListItem components.

```
30 return (
31   <Dialog onClose={handleClose} open={open}>
32     <DialogTitle>Set backup account</DialogTitle>
33     <List sx={{ pt: 0 }}>
34       {emails.map((email) => (
35         <ListItem disableGutters>
36           <ListItemIcon onClick={() => handleListItemClick(email)} key={email}>
37             <ListItemIconAvatar>
38               <Avatar sx={{ bgcolor: blue[100], color: blue[600] }}>
39                 <PersonIcon />
40               </Avatar>
41             </ListItemIconAvatar>
42             <ListItemText primary={email} />
43           </ListItemIcon>
44         </ListItem>
45       )));
46       <ListItem disableGutters>
47         <ListItemIcon
48           autoFocus
49           onClick={() => handleListItemClick('addAccount')}>
```

- Progress indicators commonly known as spinners, express an unspecified wait time or display the length of a process.
- Progress indicators inform users about the status of ongoing processes, such as loading an app, submitting a form, or saving updates.
 - **Determinate** indicators display how long an operation will take.
 - **Indeterminate** indicators visualize an unspecified wait time.

Circular

Circular indeterminate



↔ ⚙ ⚡ 📱 ☰ C :

```
<CircularProgress />
```

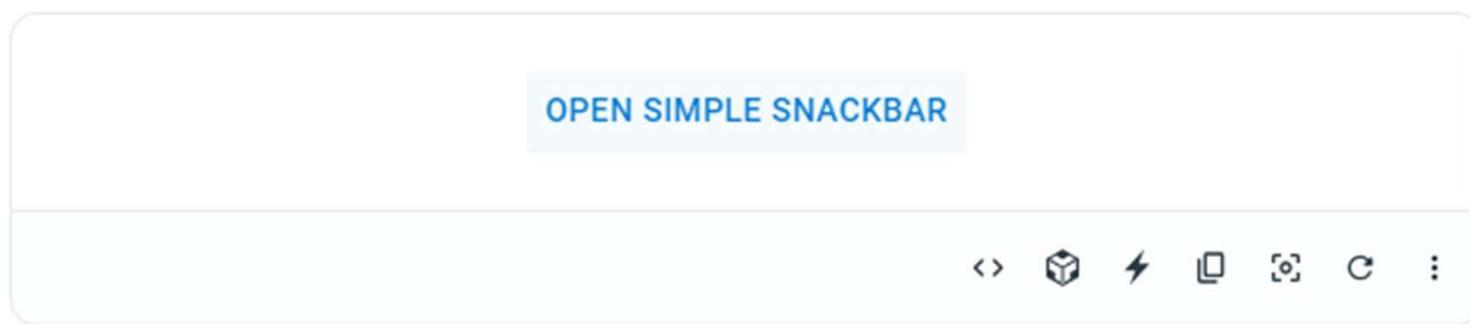
Circular color



- Snackbars provide brief notifications. The component is also known as a toast.
- Snackbars inform users of a process that an app has performed or will perform. They appear temporarily, towards the bottom of the screen. They shouldn't interrupt the user experience, and they don't require user input to disappear.
- Snackbars contain a single line of text directly related to the operation performed. They may contain a text action, but no icons. You can use them to display notifications.

Simple snackbars

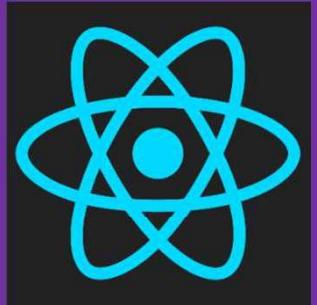
A basic Snackbar that aims to reproduce Google Keep's Snackbar behavior.



```
<Button onClick={handleClick}>Open simple Snackbar</Button>
<Snackbar
  open={open}
  autoHideDuration={6000}
  onClose={handleClose}
  message="Note archived"
  action={action}
/>
```

```
src > Feedback > Demo05-SnackBar.js > SimpleSnackbar
  30   color="inherit"
  31   onClick={handleClose}
  32   >
  33     <CloseIcon fontSize="small" />
  34   </IconButton>
  35   </React.Fragment>
  36 );
  37
  38 return (
  39   <div>
  40     <Button onClick={handleClick}>Open simple snackbar</Button>
  41     <SnackBar
  42       open={open}
  43       autoHideDuration={6000}
  44       onClose={handleClose}
  45       message="Note archived"
  46       action={action}
  47     />
  48   </div>
  49 );
```

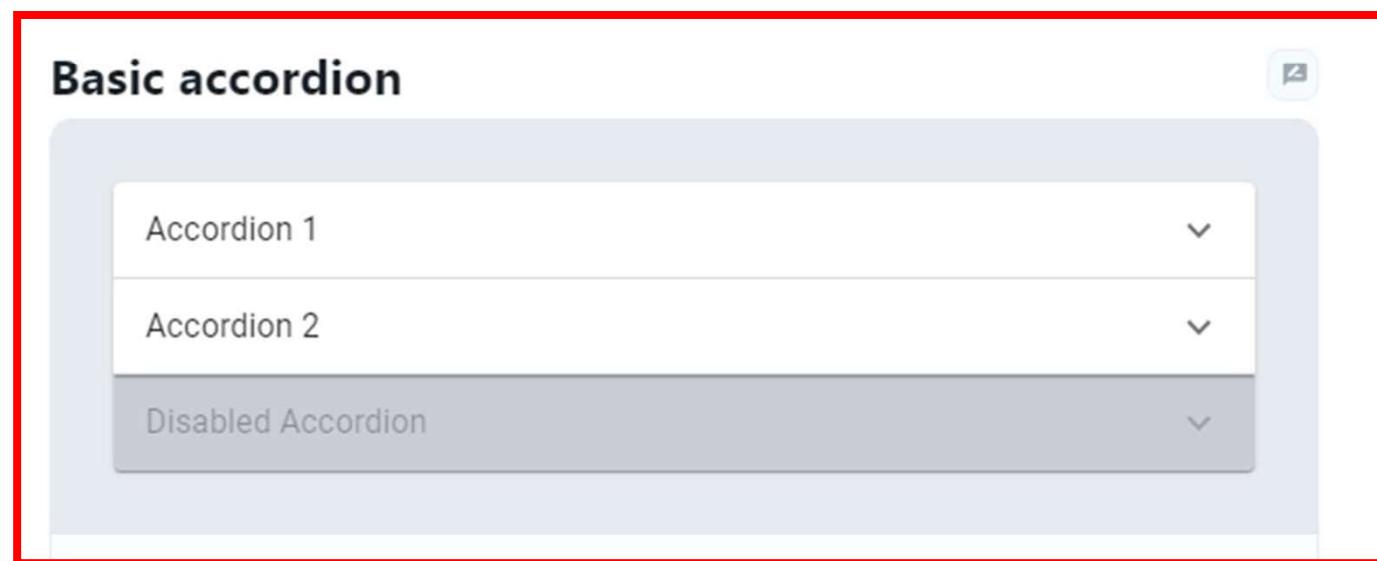
Ln 41, Col 16 (8 selected) Spaces: 4 UTF-8 CRLF {} JavaScript Go Live 4:32 PM 7/3/2023



Surfaces

Accordion

- The accordion component allows the user to show and hide sections of related content on a page.
- An accordion is a lightweight container that may either be used standalone, or be connected to a larger surface, such as a card.



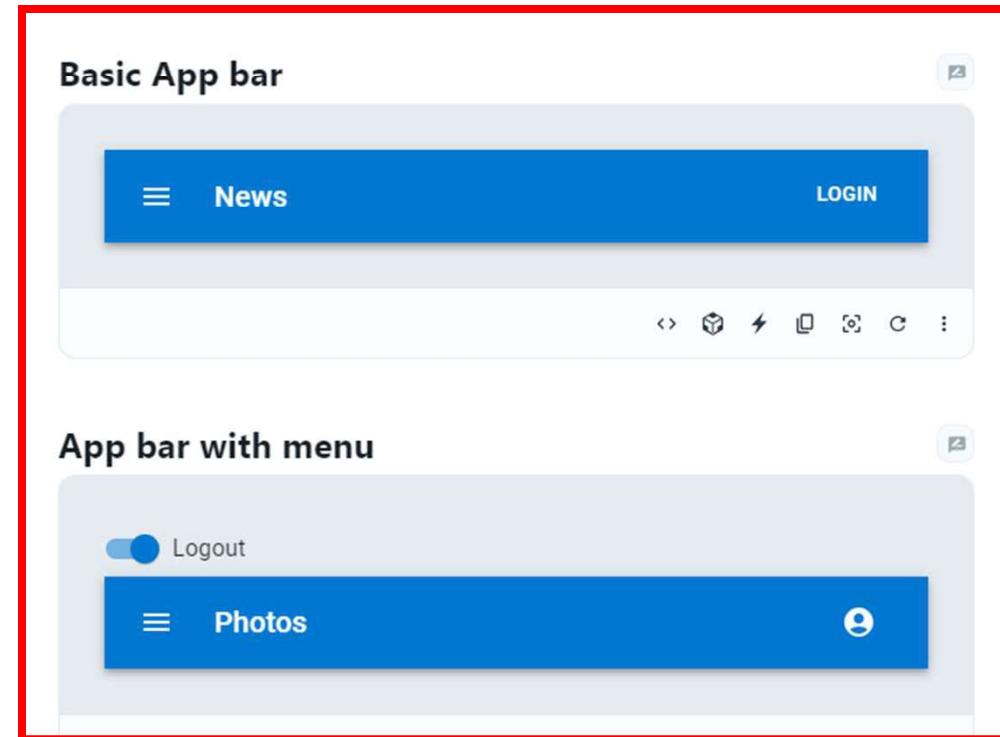
A screenshot of a Windows desktop environment. At the top, a blue header bar contains the text "test-react-app". Below it is a dark-themed code editor window titled "JS Demo01-Accordion.js X". The code editor shows a React component structure:

```
src > Surfaces > JS Demo01-Accordion.js > SimpleAccordion
return (
  <div>
    <Accordion>
      <AccordionSummary
        expandIcon={<ExpandMoreIcon />}
        aria-controls="panel1a-content"
        id="panel1a-header"
      >
        <Typography>Accordion 1</Typography>
      </AccordionSummary>
      <AccordionDetails>
        <Typography>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse
          malesuada lacus ex, sit amet blandit leo lobortis eget.
        </Typography>
      </AccordionDetails>
    </Accordion>
    <Accordion>
      <AccordionSummary
        expandIcon={<ExpandMoreIcon />}
        aria-controls="panel2a-content"
        id="panel2a-header"
      >
        <Typography>Accordion 2</Typography>
      </AccordionSummary>
      <AccordionDetails>
```

The code editor interface includes a left sidebar with icons for file operations like copy, paste, search, and refresh, and a status bar at the bottom showing file information, encoding, and system status.

App Bar

- The App Bar displays information and actions relating to the current screen.
- The top App bar provides content and actions related to the current screen. It's used for branding, screen titles, navigation, and actions.



A screenshot of the Visual Studio Code (VS Code) interface, showing a React application named "test-react-app".

The Explorer sidebar on the left shows the project structure:

- > OPEN EDITORS
- < TEST-REACT-APP
 - > public
 - src
 - Feedback
 - JS Demo01-Alert.js
 - JS Demo02-Backdrop.js
 - JS Demo03-Dialog.js
 - JS Demo04-Progress.js
 - JS Demo05-SnackBar.js
 - Surfaces
 - JS Demo01-Accordion.js
 - JS Demo02-AppBar.js** (selected)
 - # App.css
 - JS App.js
 - JS App.test.js
 - # index.css
 - JS index.js
 - logo.svg
 - JS reportWebVitals.js
 - JS setupTests.js
- > OUTLINE
- > TIMELINE

The current file, "Demo02-AppBar.js", is open in the main editor. The code defines a functional component "ButtonAppBar" that returns a component containing an with a and an with a . It also includes and components.

```
JS Demo01-Alert.js JS App.js JS Demo01-Accordion.js JS Demo02-AppBar.js X
src > Surfaces > JS Demo02-AppBar.js > ButtonAppBar
8 import MenuIcon from '@mui/icons-material/Menu';
9
10 export default function ButtonAppBar() {
11   return (
12     <Box sx={{ flexGrow: 1 }}>
13       <AppBar position="static">
14         <Toolbar>
15           <IconButton
16             size="large"
17             edge="start"
18             color="inherit"
19             aria-label="menu"
20             sx={{ mr: 2 }}
21           >
22             <MenuIcon />
23           </IconButton>
24           <Typography variant="h6" component="div" sx={{ flexGrow: 1 }}>
25             News
26           </Typography>
27           <Button color="inherit">Login</Button>
28         </Toolbar>
29       </AppBar>
30     </Box>
31   );
32 }
```

The status bar at the bottom shows the following information:

- Ln 32, Col 2
- Spaces: 4
- UTF-8
- CRLF
- { } JavaScript
- Go Live
- 28°C Mostly cloudy
- 4:45 PM
- 7/3/2023

- Cards contain content and actions about a single subject.
- Cards are surfaces that display content and actions on a single topic.
- They should be easy to scan for relevant and actionable information. Elements, like text and images, should be placed on them in a way that clearly indicates hierarchy.

Complex Interaction

On desktop, card content can expand. (Click the downward chevron to view the recipe.)



A screenshot of a Windows desktop environment. In the center is a Microsoft Edge browser window with a blue header bar containing the text "test-react-app". Below the header is a dark-themed code editor showing a React component named "Demo03-Card.js". The code uses the Material UI library and includes a card with an avatar, actions, and a media component displaying a paella image. The code editor's status bar at the bottom shows "Ln 29, Col 41 (16 selected)" and other file-related information. To the left of the browser is a dark-themed file explorer window titled "TEST-REACT-APP" which lists various project files and folders. The taskbar at the bottom of the screen contains icons for the Start button, search, File Explorer, Edge, Mail, OneDrive, Teams, and Power BI, along with system status icons like battery level, signal strength, and weather.

```
src > Surfaces > Demo03-Card.js > RecipeReviewCard
      return (
        <Card sx={{ maxWidth: 345 }}>
          <CardHeader
            avatar={

              <Avatar sx={{ bgcolor: red[500] }} aria-label="recipe">
                R
              </Avatar>
            }
            action={
              <IconButton aria-label="settings">
                <MoreVertIcon />
              </IconButton>
            }
            title="Shrimp and Chorizo Paella"
            subheader="September 14, 2016"
          />
          <CardMedia
            component="img"
            height="194"
            image="/static/images/cards/paella.jpg"
            alt="Paella dish"
          />
          <CardContent>
            <Typography variant="body2" color="text.secondary">
              This impressive paella is a perfect party dish and a fun meal to cook
            </Typography>
          </CardContent>
        </Card>
      )
    # App.css
    JS App.js
    JS App.test.js
    # index.css
    JS index.js
    logo.svg
  
```