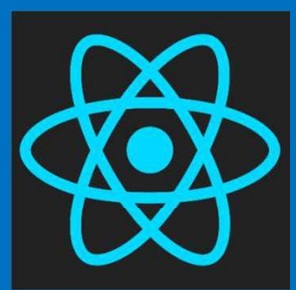
2 Creating Beautiful Apps with Material UI

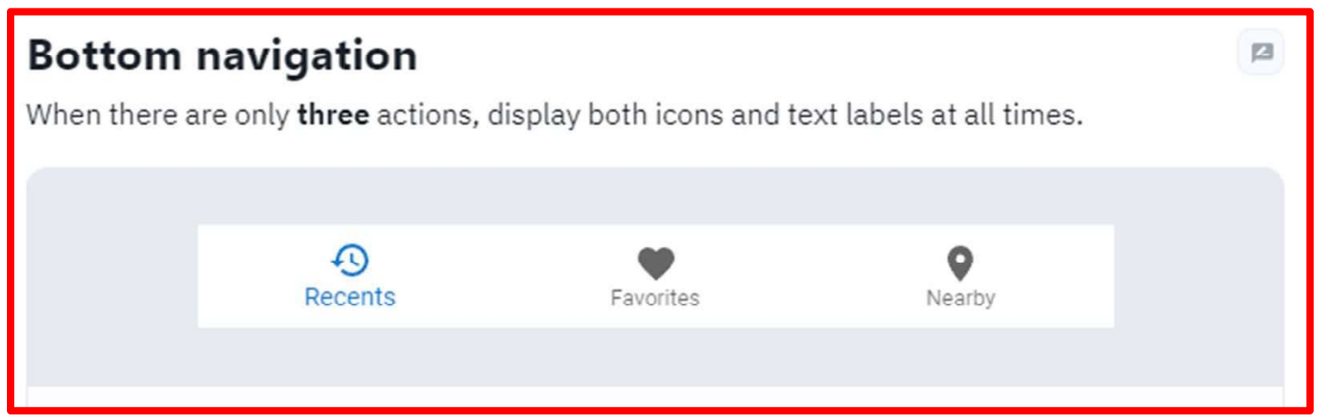


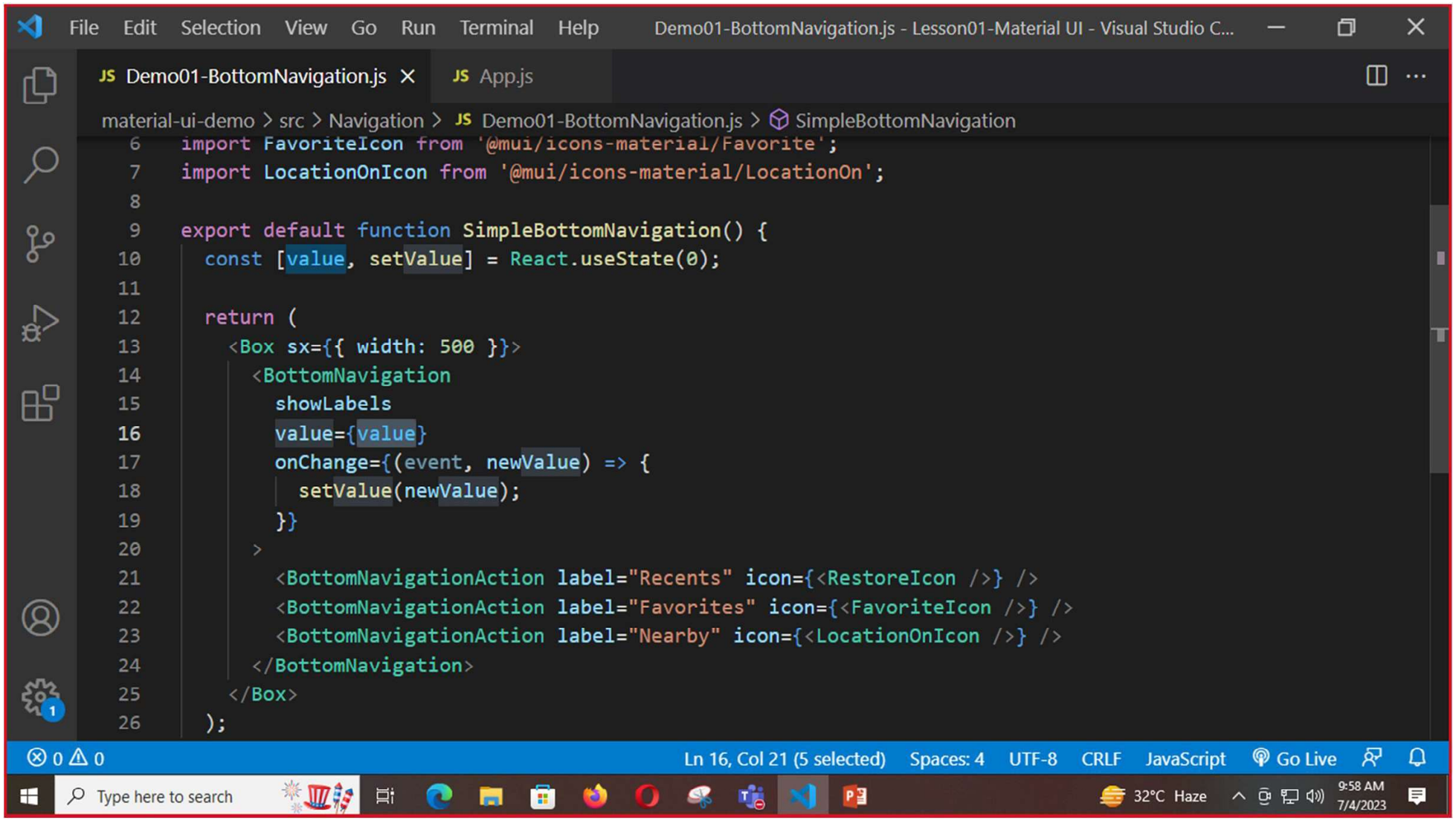
Material UI Navigation

Bottom Navigation

- The Bottom Navigation bar allows movement between primary destinations in an app.

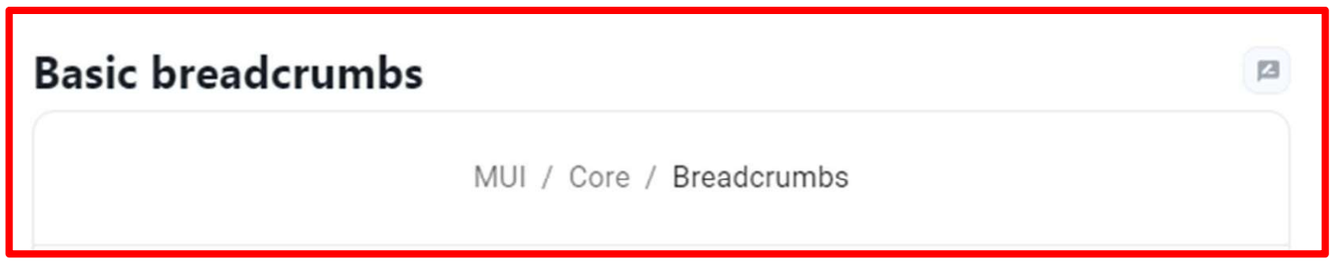
- Bottom navigation bars display three to five destinations at the bottom of a screen. Each destination is represented by an icon and an optional text label. When a bottom navigation icon is tapped, the user is taken to the top-level navigation destination associated with that icon.

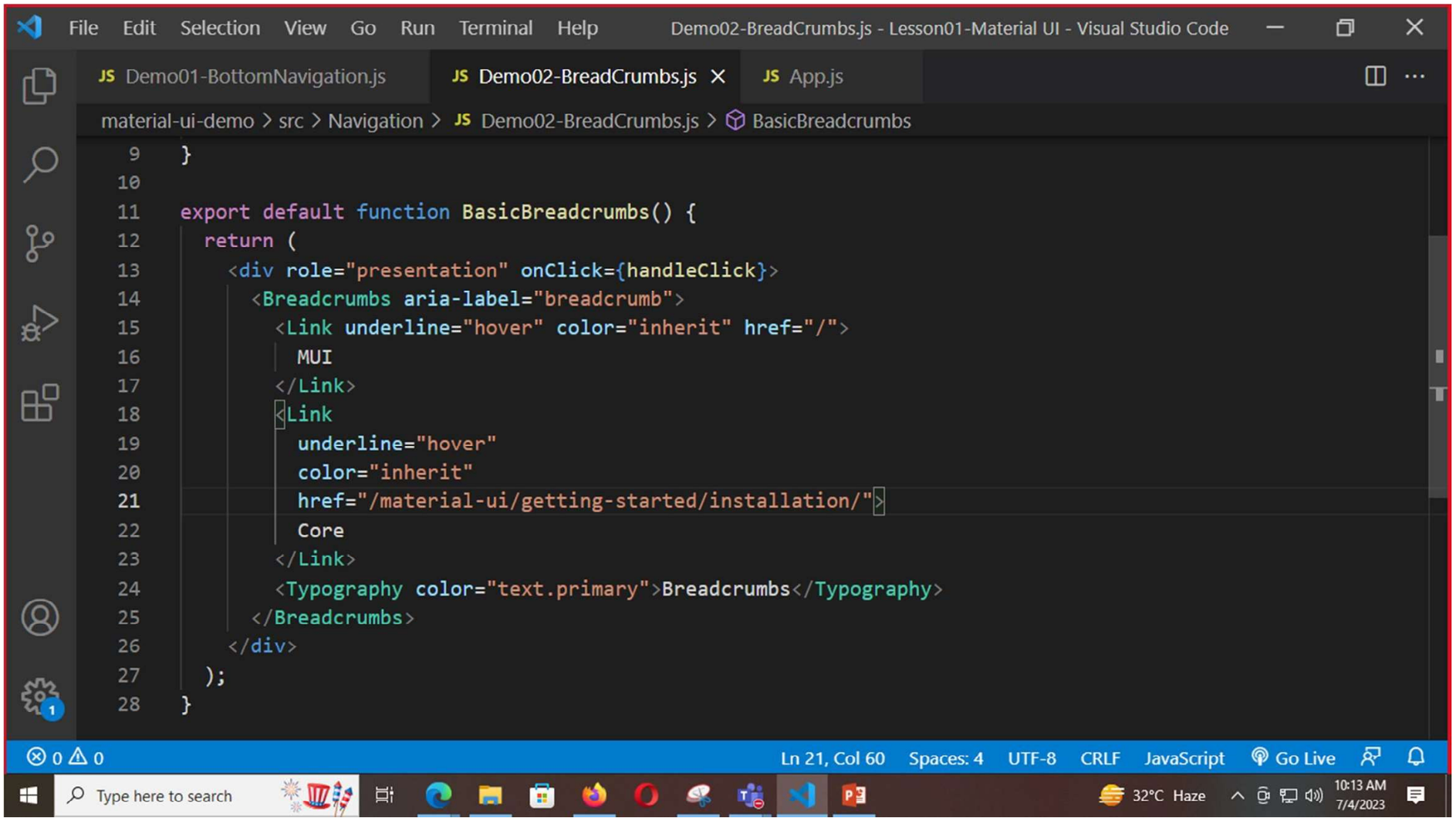



File Edit Selection View Go Run Terminal Help
Demo01-BottomNavigation.js - Lesson01-Material UI - Visual Studio C...
Demo01-BottomNavigation.js
App.js
material-ui-demo > src > Navigation > Demo01-BottomNavigation.js > SimpleBottomNavigation
import FavoriteIcon from '@mui/icons-material/Favorite';
import LocationOnIcon from '@mui/icons-material/LocationOn';
export default function SimpleBottomNavigation() {
const [value, setValue] = React.useState(0);
return (
<Box sx={{ width: 500 }}>
<BottomNavigation
showLabels
value={value}
onChange={(event, newValue) => {
setValue(newValue);
}}
>
<BottomNavigationAction label="Recents" icon={<RestoreIcon />} />
<BottomNavigationAction label="Favorites" icon={<FavoriteIcon />} />
<BottomNavigationAction label="Nearby" icon={<LocationOnIcon />} />
</BottomNavigation>
</Box>
);
Ln 16, Col 21 (5 selected) Spaces: 4 UTF-8 CRLF JavaScript Go Live
Type here to search
32°C Haze
9:58 AM
7/4/2023

Breadcrumbs

- A breadcrumbs is a list of links that help visualize a page's location within a site's hierarchical structure, it allows navigation up to any of the ancestors.




File Edit Selection View Go Run Terminal Help
Demo02-BreadCrumbs.js - Lesson01-Material UI - Visual Studio Code
Demo01-BottomNavigation.js
Demo02-BreadCrumbs.js
App.js
material-ui-demo > src > Navigation > Demo02-BreadCrumbs.js > BasicBreadcrumbs
}
export default function BasicBreadcrumbs() {
return (

<Breadcrumbs aria-label="breadcrumb">
<Link underline="hover" color="inherit" href="/">
MUI
</Link>
<Link
underline="hover"
color="inherit"
href="/material-ui/getting-started/installation/"
Core
</Link>
<Typography color="text.primary">Breadcrumbs</Typography>
</Breadcrumbs>

);
}
0 0
Ln 21, Col 60
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live
Type here to search
32°C Haze
10:13 AM
7/4/2023

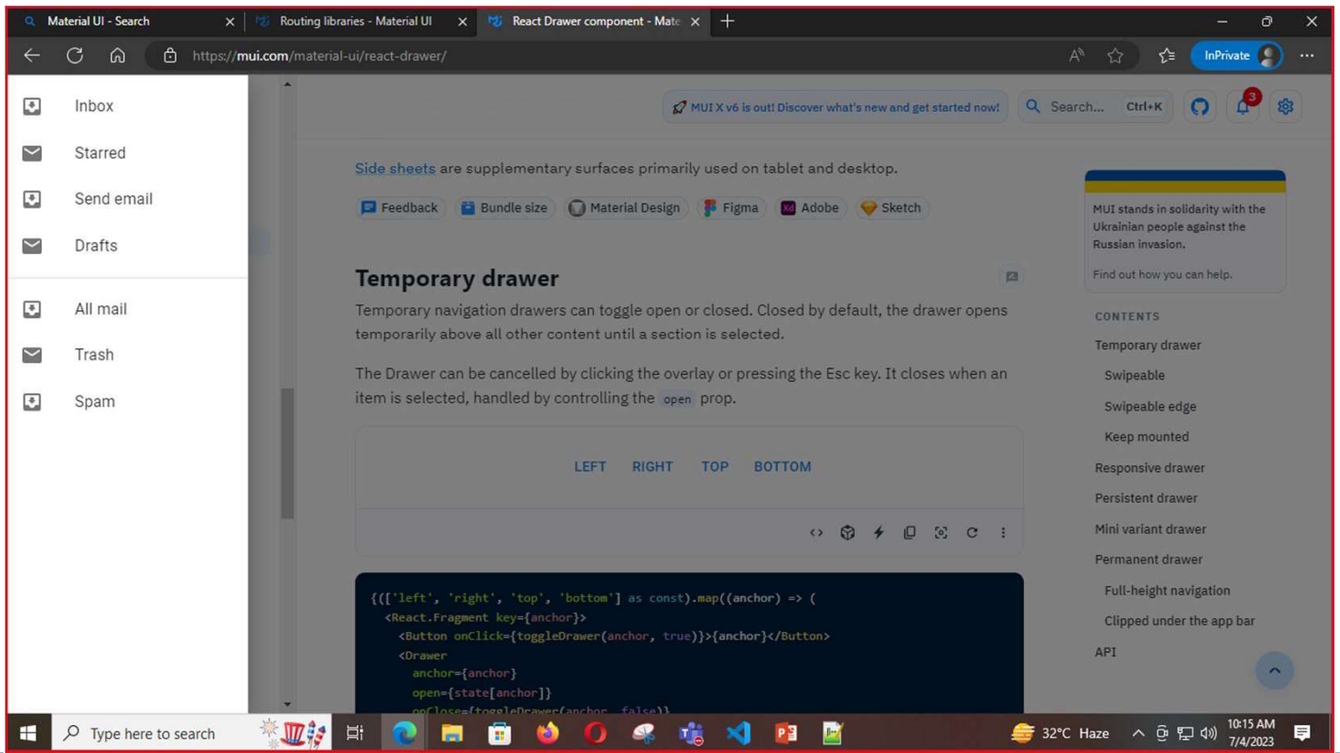
Drawer

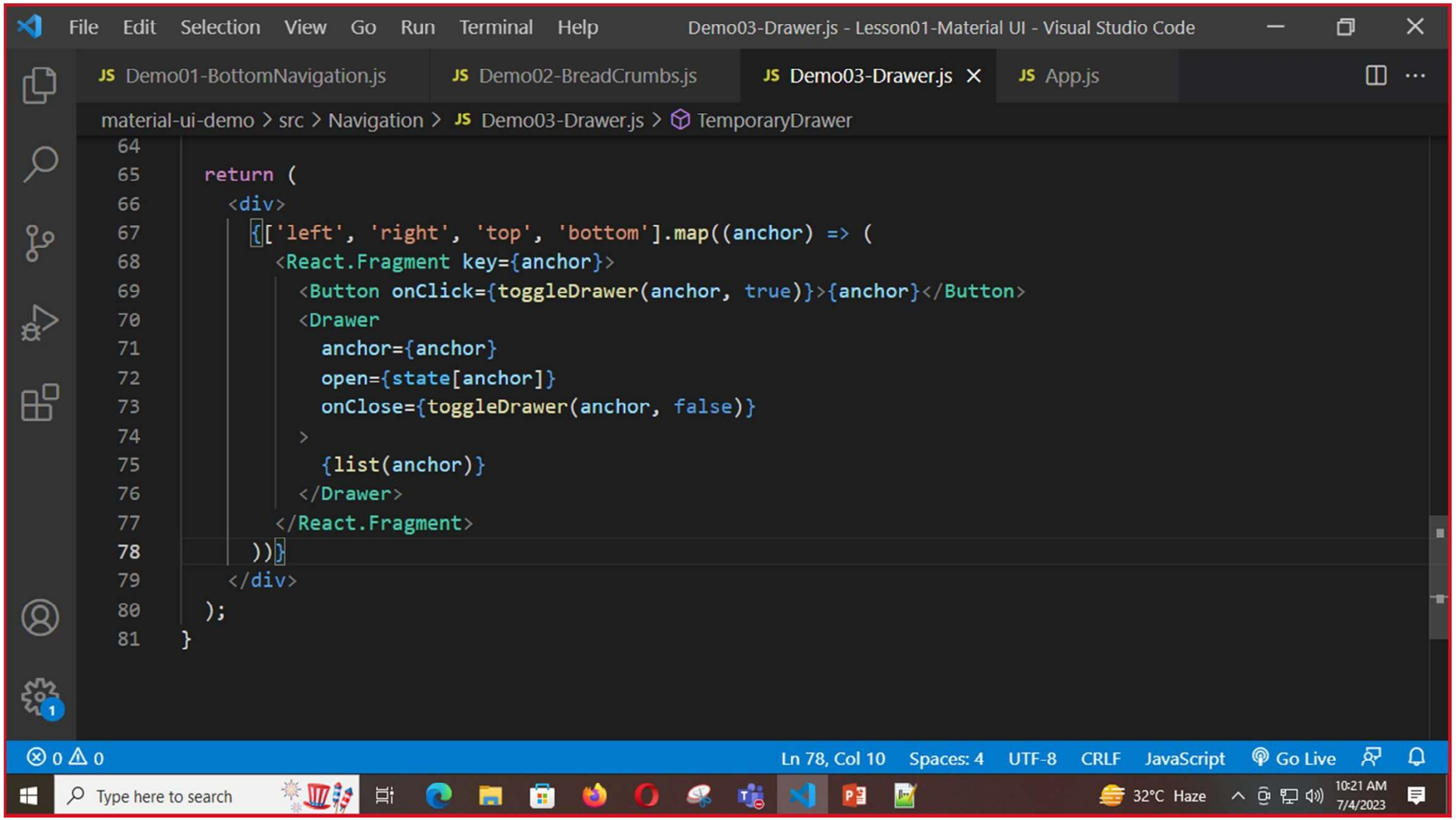
- The navigation drawers (or "sidebars") provide ergonomic access to destinations in a site or app functionality such as switching accounts.

- A navigation drawer can either be permanently on-screen or controlled by a navigation menu icon.

Temporary drawer

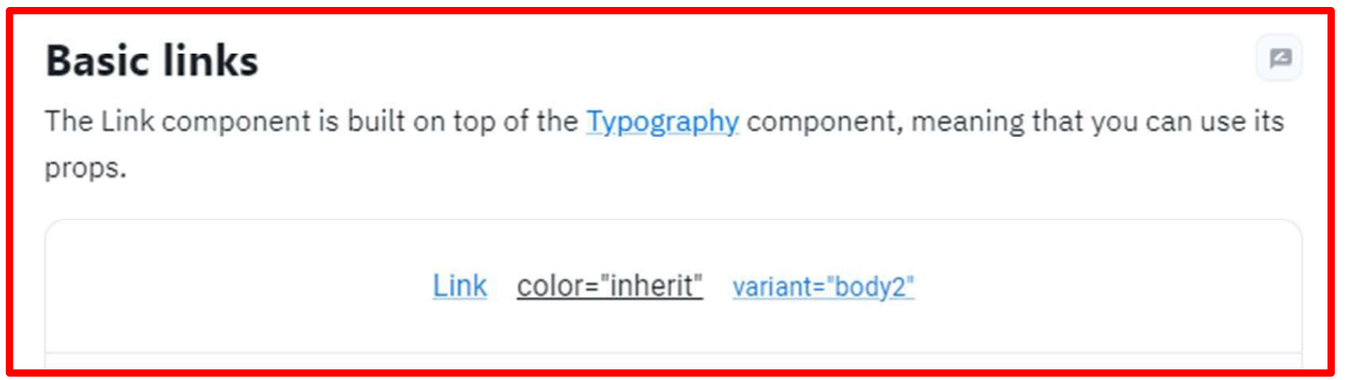
- Temporary navigation drawers can toggle open or closed. Closed by default, the drawer opens temporarily above all other content until a section is selected.
- The Drawer can be cancelled by clicking the overlay or pressing the Esc key. It closes when an item is selected, handled by controlling the open prop.

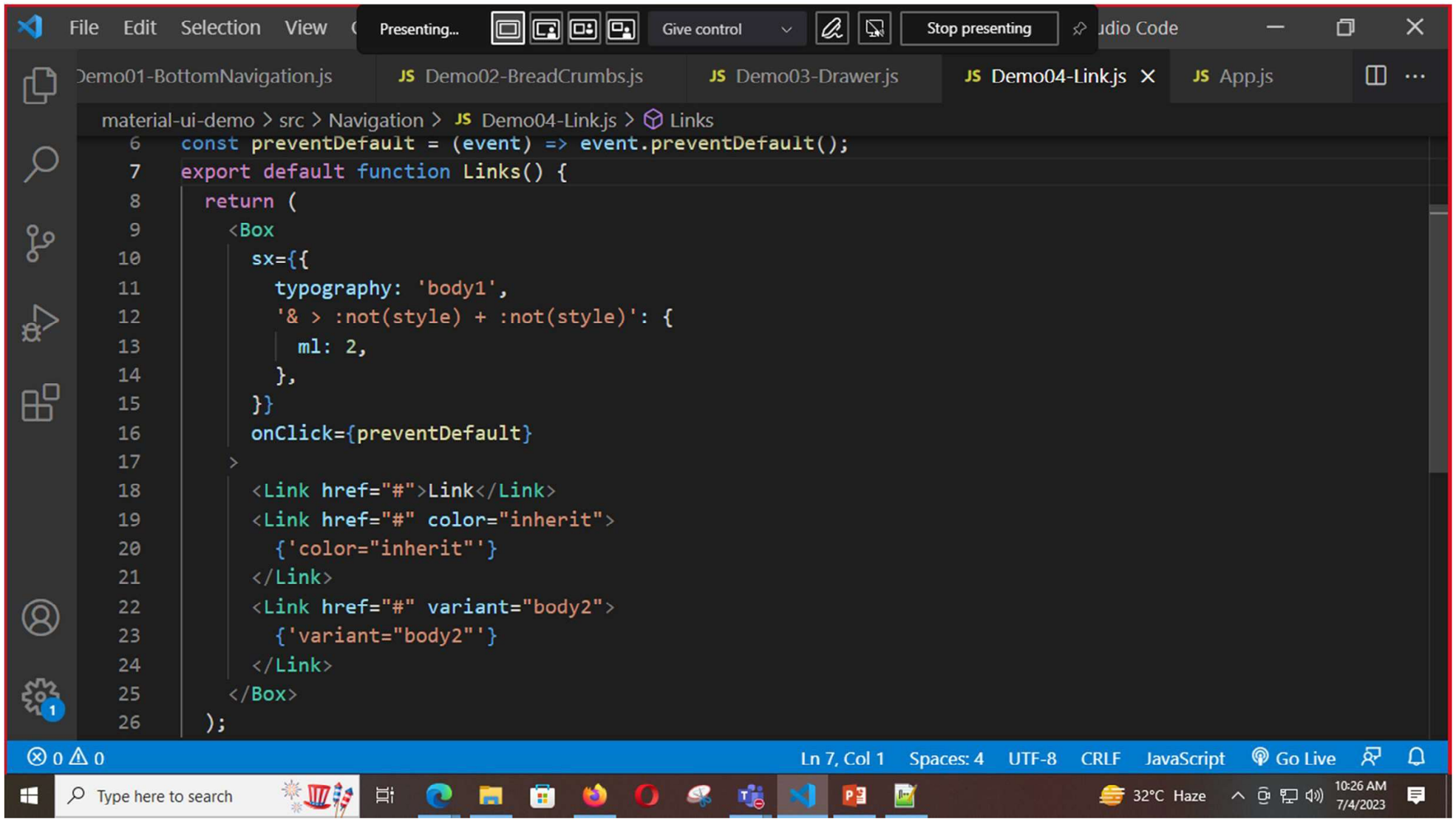



File Edit Selection View Go Run Terminal Help
Demo03-Drawer.js - Lesson01-Material UI - Visual Studio Code
Demo01-BottomNavigation.js
Demo02-BreadCrumbs.js
Demo03-Drawer.js
App.js
material-ui-demo > src > Navigation > Demo03-Drawer.js > TemporaryDrawer
64
65 return (
66
67 {['left', 'right', 'top', 'bottom'].map((anchor) => (
68 <React.Fragment key={anchor}>
69 <Button onClick={toggleDrawer(anchor, true)}>{anchor}</Button>
70 <Drawer
71 anchor={anchor}
72 open={state[anchor]}
73 onClose={toggleDrawer(anchor, false)}
74 >
75 {list(anchor)}
76 </Drawer>
77 </React.Fragment>
78))}
79
80);
81 }
Ln 78, Col 10 Spaces: 4 UTF-8 CRLF JavaScript Go Live
Type here to search
32°C Haze
10:21 AM
7/4/2023

Links

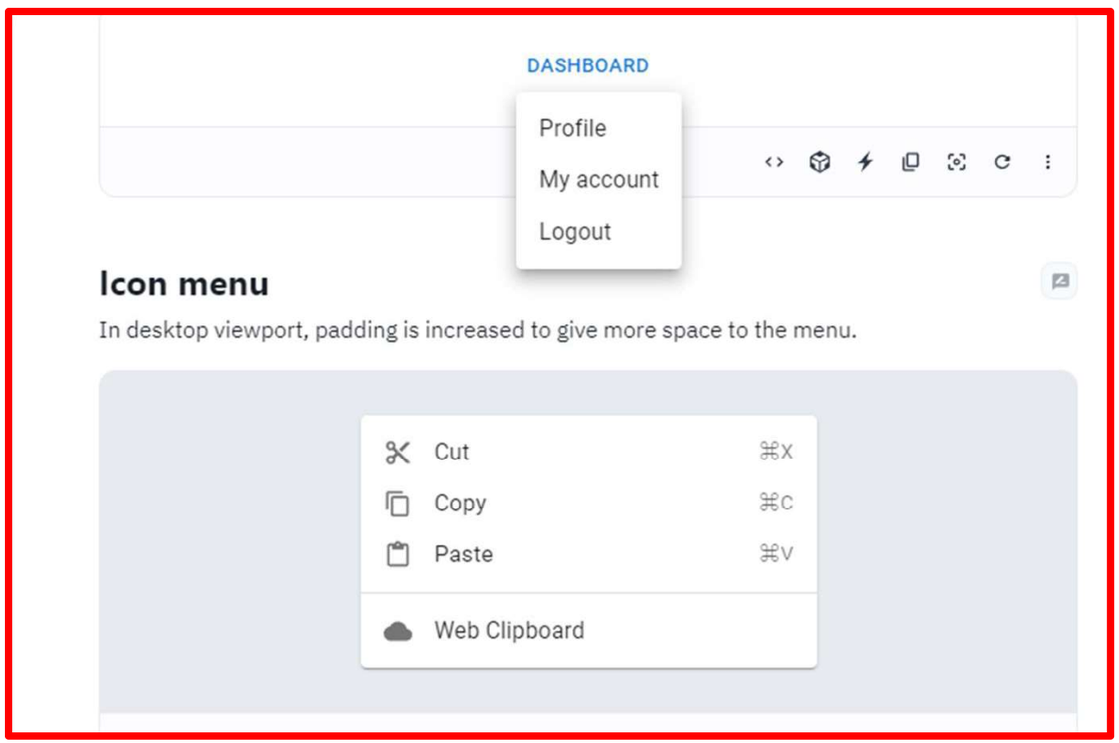
- The Link component allows you to easily customize anchor elements with your theme colors and typography styles.

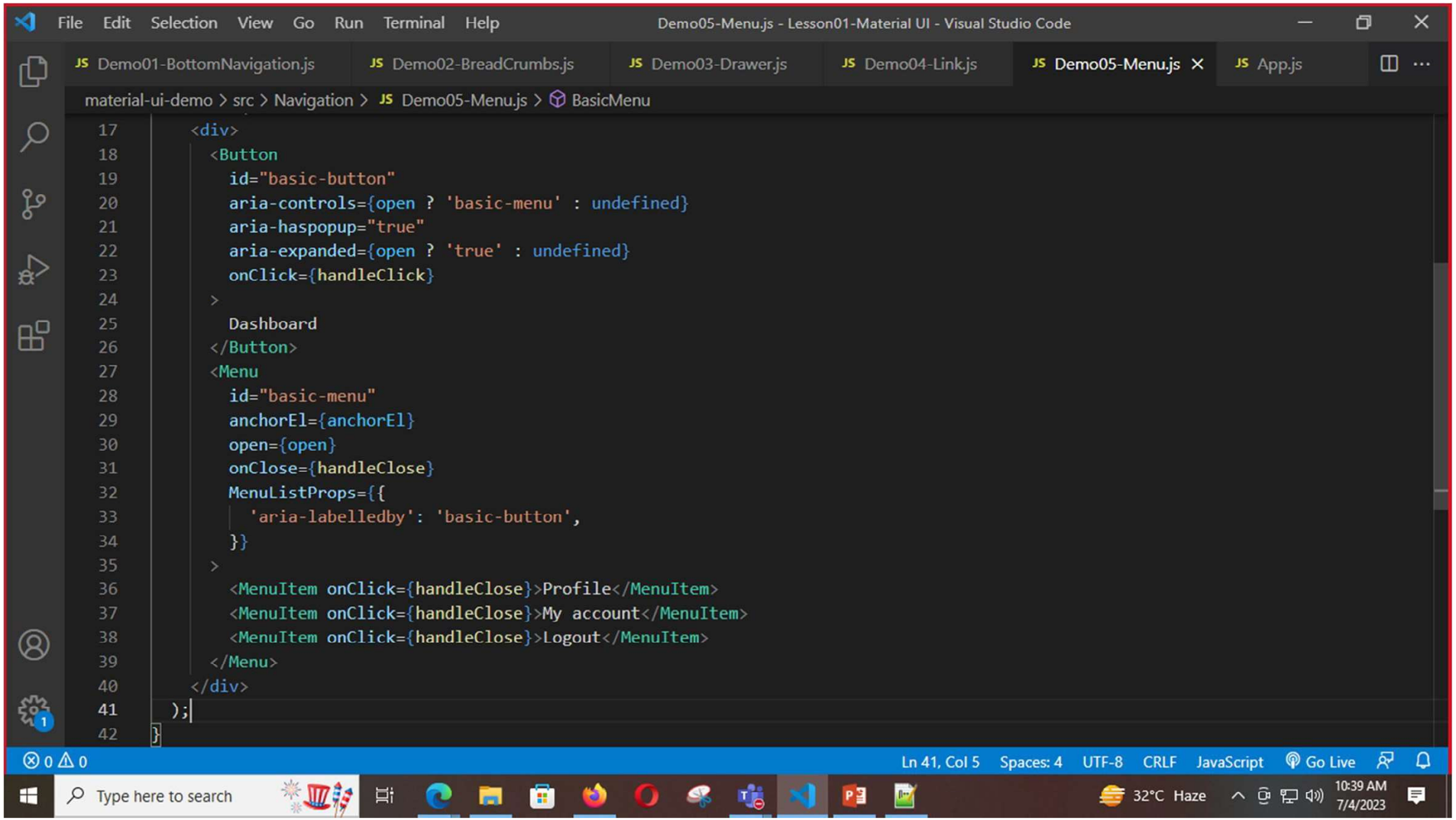



File Edit Selection View
Presenting...
Give control
Stop presenting
Demo01-BottomNavigation.js
Demo02-BreadCrumbs.js
Demo03-Drawer.js
Demo04-Link.js
App.js
material-ui-demo > src > Navigation > Demo04-Link.js > Links
const preventDefault = (event) => event.preventDefault();
export default function Links() {
return (
<Box
sx={{
typography: 'body1',
'& > :not(style) + :not(style)': {
ml: 2,
},
}}
onClick={preventDefault}
>
<Link href="#">Link</Link>
<Link href="#" color="inherit">
{'color="inherit"'}
</Link>
<Link href="#" variant="body2">
{'variant="body2"'}
</Link>
</Box>
);
Ln 7, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live
Type here to search
32°C Haze
10:26 AM
7/4/2023

Menu

- Menus display a list of choices on temporary surfaces.
- A menu displays a list of choices on a temporary surface. It appears when the user interacts with a button, or other control.



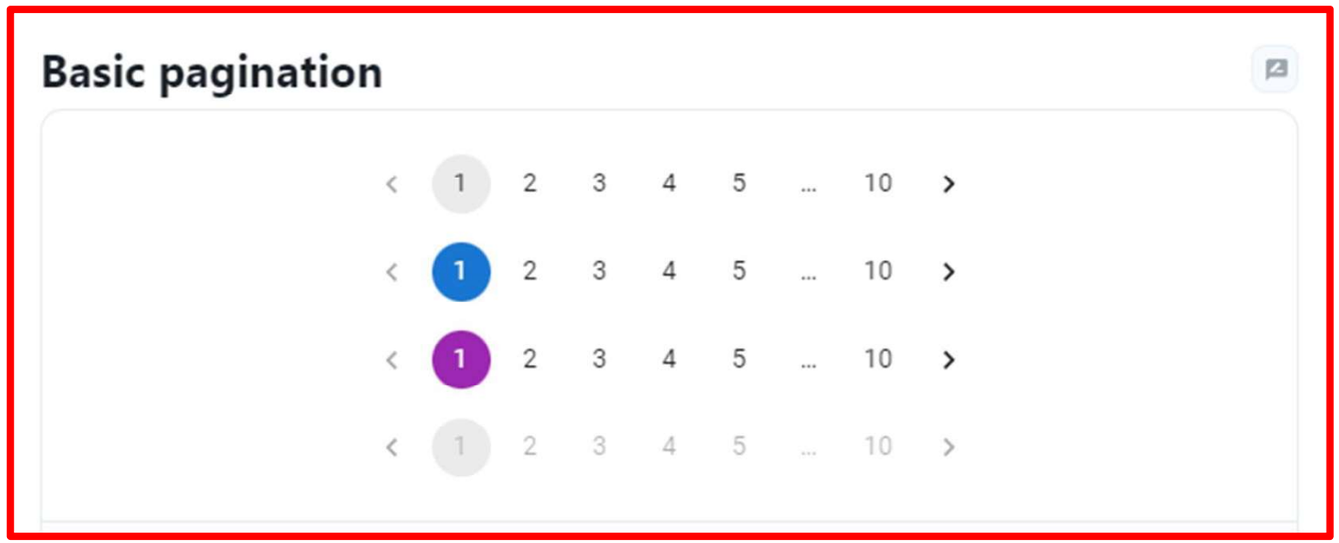

File Edit Selection View Go Run Terminal Help
Demo05-Menu.js - Lesson01-Material UI - Visual Studio Code
Demo01-BottomNavigation.js
Demo02-BreadCrumbs.js
Demo03-Drawer.js
Demo04-Link.js
Demo05-Menu.js
App.js
material-ui-demo > src > Navigation > Demo05-Menu.js > BasicMenu

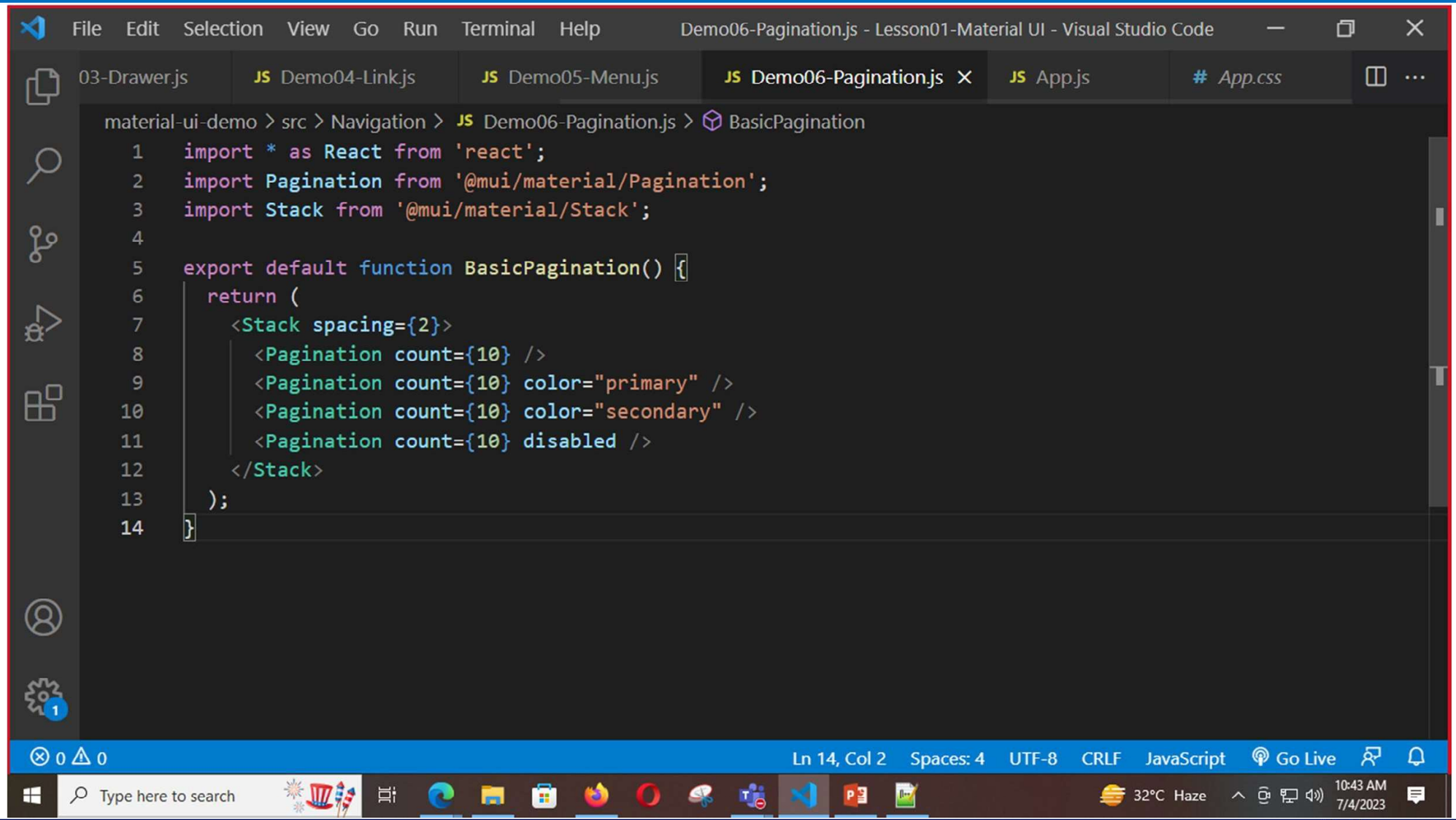
<Button
id="basic-button"
aria-controls={open ? 'basic-menu' : undefined}
aria-haspopup="true"
aria-expanded={open ? 'true' : undefined}
onClick={handleClick}
>
Dashboard
</Button>
<Menu
id="basic-menu"
anchorEl={anchorEl}
open={open}
onClose={handleClose}
MenuListProps={{
'aria-labelledby': 'basic-button',
}}
>
<MenuItem onClick={handleClose}>Profile</MenuItem>
<MenuItem onClick={handleClose}>My account</MenuItem>
<MenuItem onClick={handleClose}>Logout</MenuItem>
</Menu>

);
}
Ln 41, Col 5 Spaces: 4 UTF-8 CRLF JavaScript Go Live
Type here to search
32°C Haze
10:39 AM
7/4/2023

Pagination

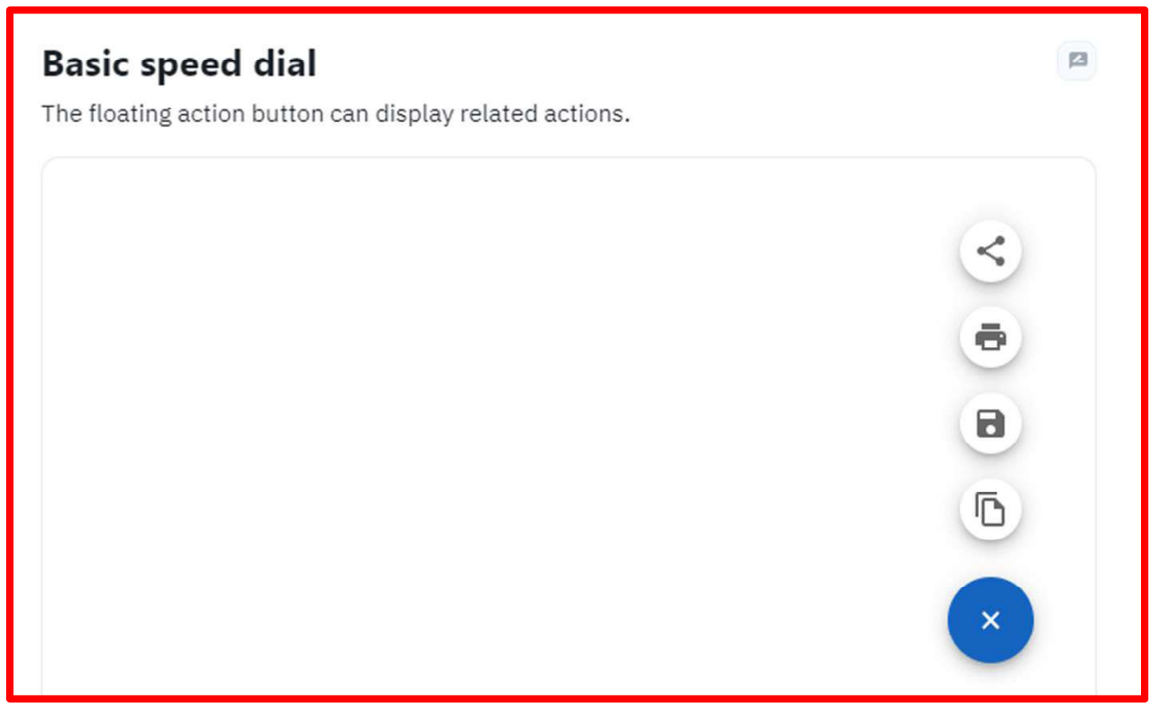
- The Pagination component enables the user to select a specific page from a range of pages.

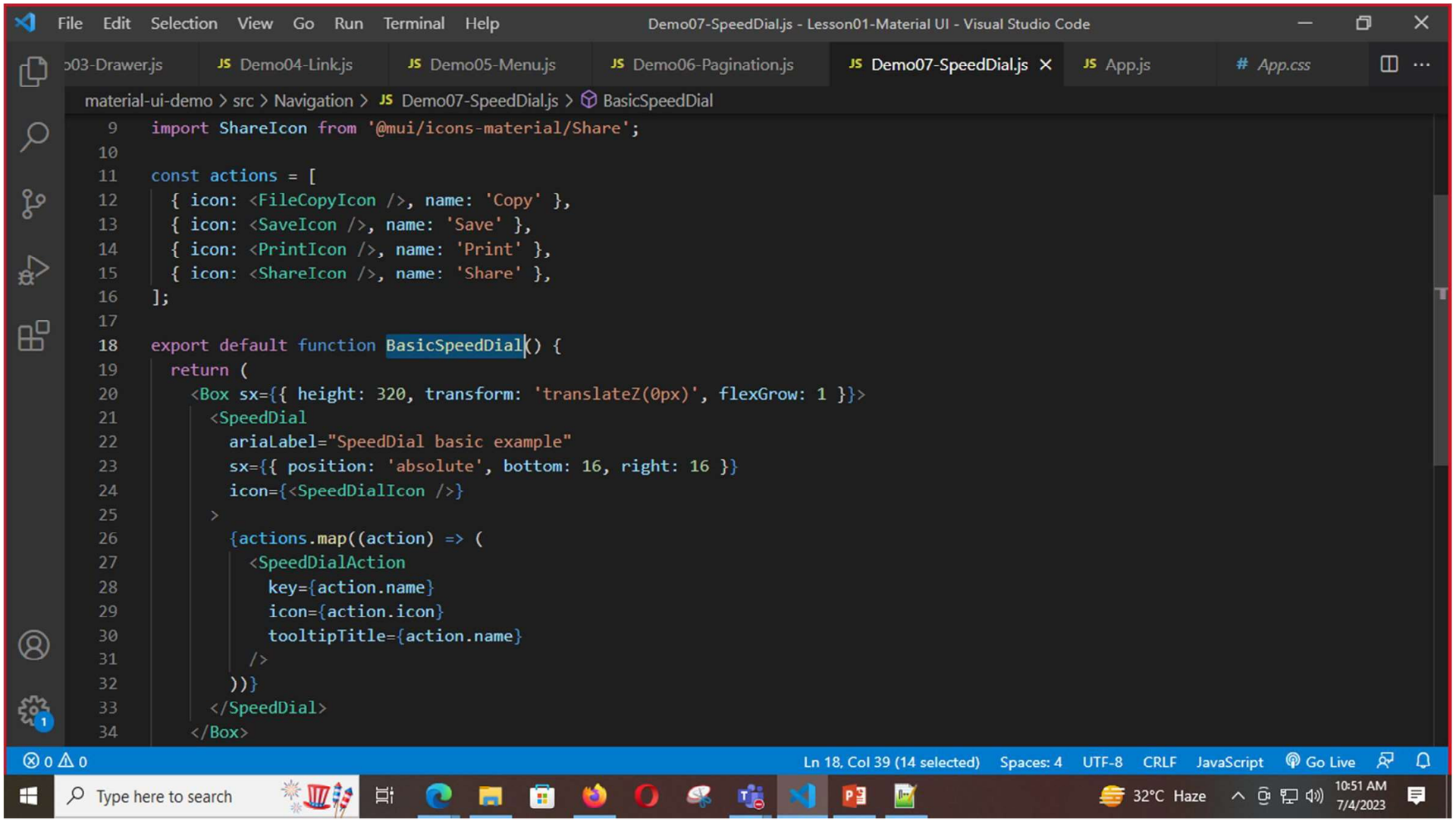



File Edit Selection View Go Run Terminal Help
Demo06-Pagination.js - Lesson01-Material UI - Visual Studio Code
03-Drawer.js
Demo04-Link.js
Demo05-Menu.js
Demo06-Pagination.js
App.js
App.css
material-ui-demo > src > Navigation > Demo06-Pagination.js > BasicPagination
1 import * as React from 'react';
2 import Pagination from '@mui/material/Pagination';
3 import Stack from '@mui/material/Stack';
4
5 export default function BasicPagination() {
6 return (
7 <Stack spacing={2}>
8 <Pagination count={10} />
9 <Pagination count={10} color="primary" />
10 <Pagination count={10} color="secondary" />
11 <Pagination count={10} disabled />
12 </Stack>
13);
14 }
Ln 14, Col 2 Spaces: 4 UTF-8 CRLF JavaScript Go Live
Type here to search
32°C Haze
10:43 AM
7/4/2023

Speed Dial

- When pressed, a floating action button can display three to six related actions in the form of a Speed Dial.
- If more than six actions are needed, something other than a FAB should be used to present them.




File Edit Selection View Go Run Terminal Help
Demo07-SpeedDial.js - Lesson01-Material UI - Visual Studio Code
Demo04-Link.js
Demo05-Menu.js
Demo06-Pagination.js
Demo07-SpeedDial.js
App.js
App.css
material-ui-demo > src > Navigation > Demo07-SpeedDial.js > BasicSpeedDial
import ShareIcon from '@mui/icons-material/Share';
const actions = [
{ icon: <FileCopyIcon />, name: 'Copy' },
{ icon: <SaveIcon />, name: 'Save' },
{ icon: <PrintIcon />, name: 'Print' },
{ icon: <ShareIcon />, name: 'Share' },
];
export default function BasicSpeedDial() {
return (
<Box sx={{ height: 320, transform: 'translateZ(0px)', flexGrow: 1 }}>
<SpeedDial
ariaLabel="SpeedDial basic example"
sx={{ position: 'absolute', bottom: 16, right: 16 }}
icon={<SpeedDialIcon />}
>
{actions.map((action) => (
<SpeedDialAction
key={action.name}
icon={action.icon}
tooltipTitle={action.name}
/>
))}
</SpeedDial>
</Box>
Ln 18, Col 39 (14 selected) Spaces: 4 UTF-8 CRLF JavaScript Go Live
Type here to search
32°C Haze
10:51 AM
7/4/2023

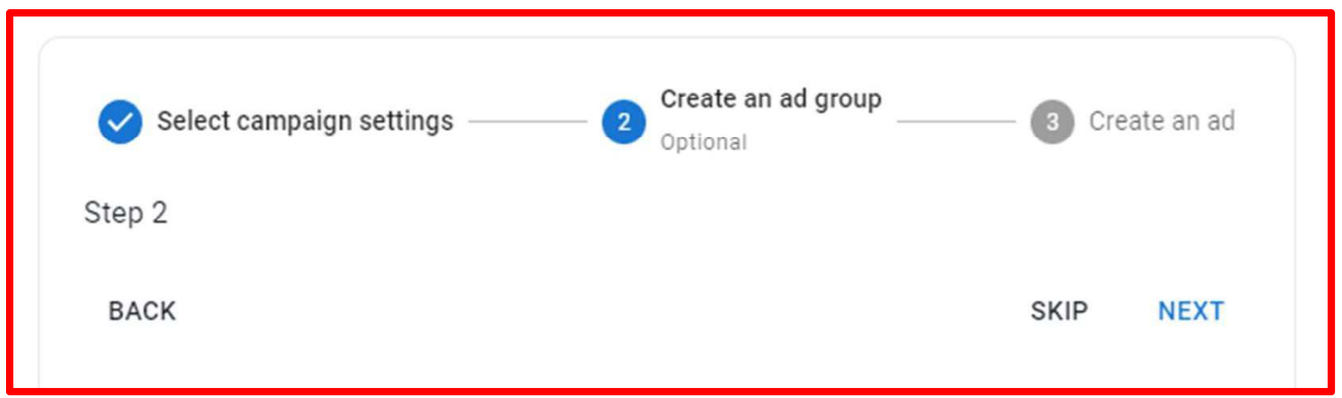
Stepper

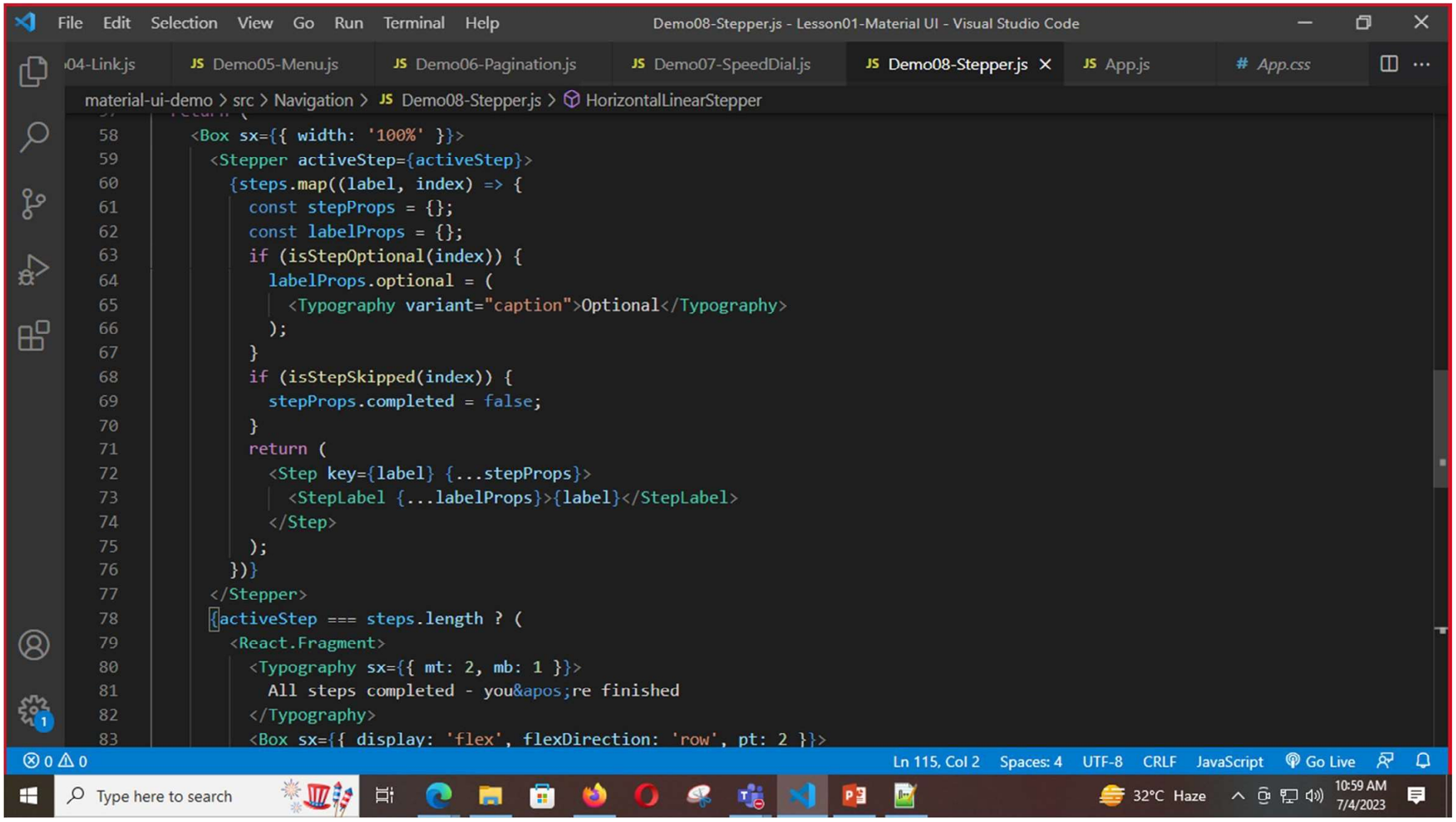
- **Stepper** Shows the indication of Where U [In the sense in which Part of the App] or Like Progress in the Appliction

- Steppers convey progress through numbered steps. It provides a wizard-like workflow.
- Steppers display progress through a sequence of logical and numbered steps. They may also be used for navigation. Steppers may display a transient feedback message after a step is saved.
  - **Types of Steps**: Editable, Non-editable, Mobile, Optional
  - **Types of Steppers**: Horizontal, Vertical, Linear, Non-linear

Horizontal stepper

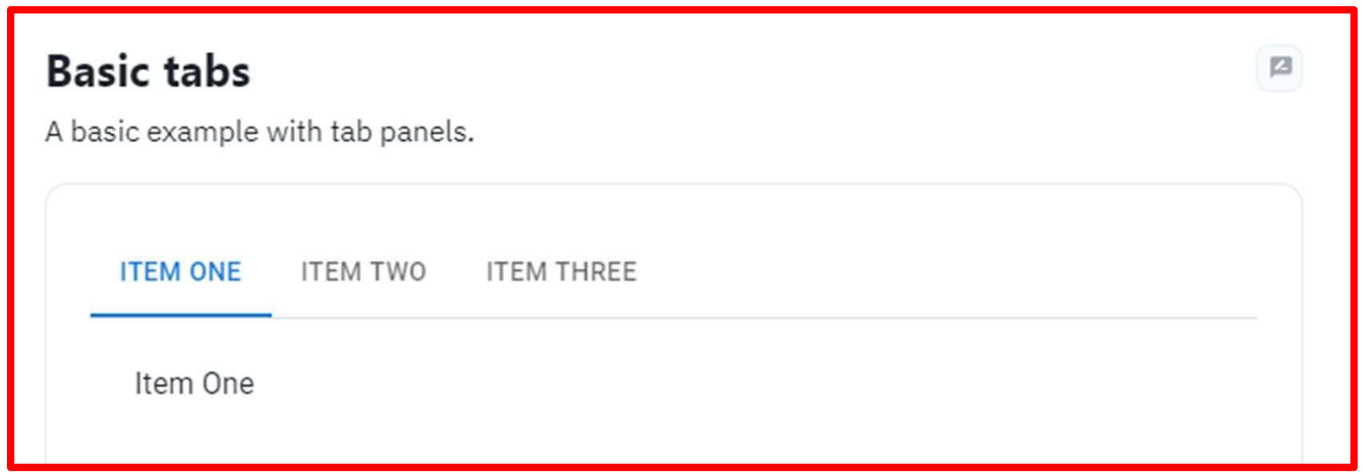
- Horizontal steppers are ideal when the contents of one step depend on an earlier step.
- Avoid using long step names in horizontal steppers.
- Linear
  - A linear stepper allows the user to complete the steps in sequence.
  - The Stepper can be controlled by passing the current step index (zero-based) as the activeStep prop. Stepper orientation is set using the orientation prop.

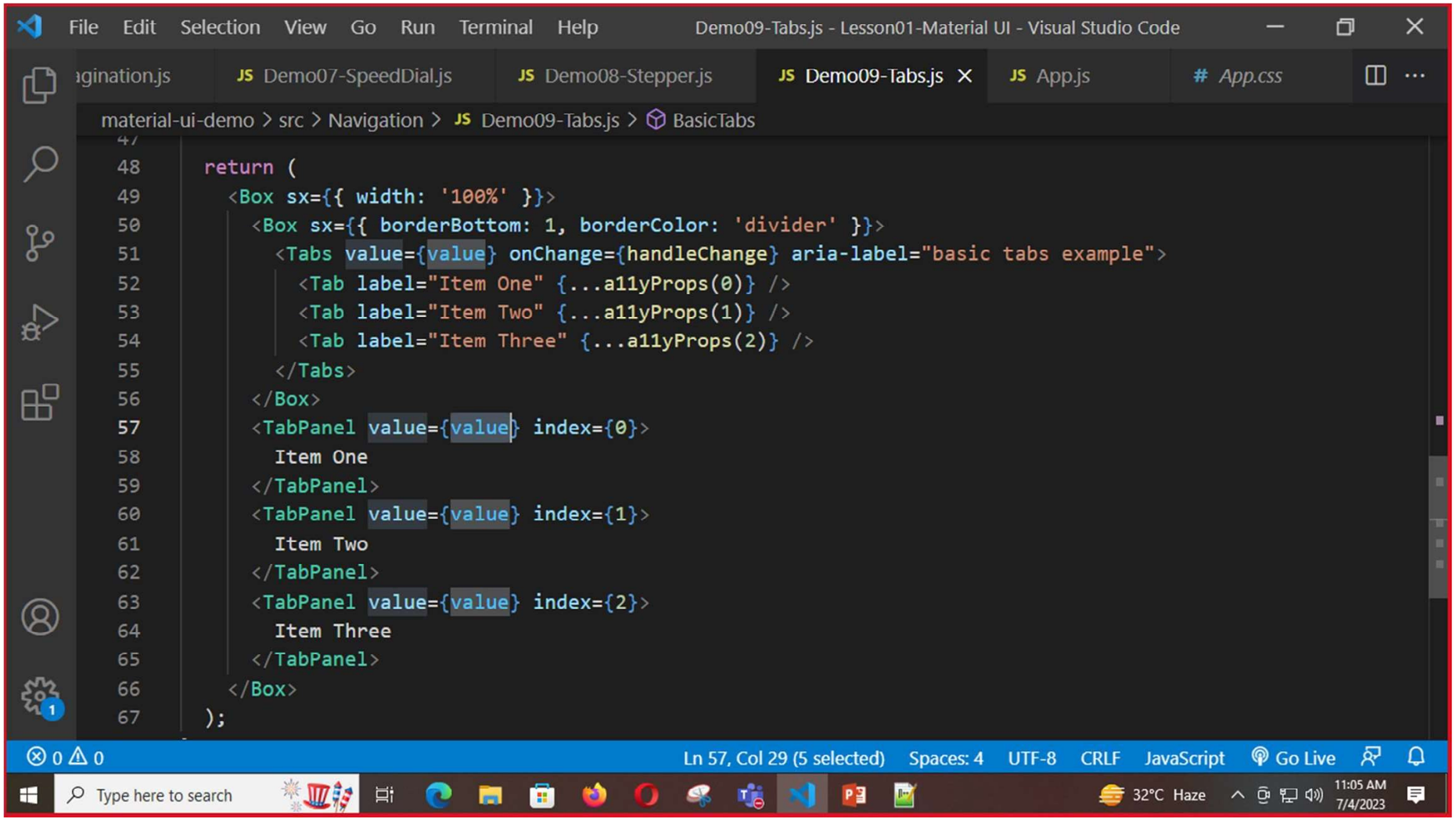


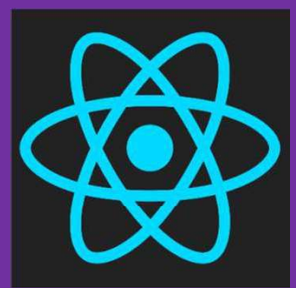

File Edit Selection View Go Run Terminal Help
Demo08-Stepper.js - Lesson01-Material UI - Visual Studio Code
04-Link.js
Demo05-Menu.js
Demo06-Pagination.js
Demo07-SpeedDial.js
Demo08-Stepper.js
App.js
App.css
material-ui-demo > src > Navigation > Demo08-Stepper.js > HorizontalLinearStepper
58 <Box sx={{ width: '100%' }}>
59 <Stepper activeStep={activeStep}>
60 {steps.map((label, index) => {
61 const stepProps = {};
62 const labelProps = {};
63 if (isStepOptional(index)) {
64 labelProps.optional = (
65 <Typography variant="caption">Optional</Typography>
66);
67 }
68 if (isStepSkipped(index)) {
69 stepProps.completed = false;
70 }
71 return (
72 <Step key={label} {...stepProps}>
73 <StepLabel {...labelProps}>{label}</StepLabel>
74 </Step>
75);
76 })}
77 </Stepper>
78 {activeStep === steps.length ? (
79 <React.Fragment>
80 <Typography sx={{ mt: 2, mb: 1 }}>
81 All steps completed - you're finished
82 </Typography>
83 <Box sx={{ display: 'flex', flexDirection: 'row', pt: 2 }}>
Ln 115, Col 2 Spaces: 4 UTF-8 CRLF JavaScript Go Live
Type here to search
32°C Haze
10:59 AM
7/4/2023

Tabs

- Tabs make it easy to explore and switch between different views.
- Tabs organize and allow navigation between groups of content that are related and at the same level of hierarchy.




File Edit Selection View Go Run Terminal Help
Demo09-Tabs.js - Lesson01-Material UI - Visual Studio Code
Demo07-SpeedDial.js
Demo08-Stepper.js
Demo09-Tabs.js
App.js
App.css
material-ui-demo > src > Navigation > Demo09-Tabs.js > BasicTabs
return (
<Box sx={{ width: '100%' }}>
<Box sx={{ borderBottom: 1, borderColor: 'divider' }}>
<Tabs value={value} onChange={handleChange} aria-label="basic tabs example">
<Tab label="Item One" {...a11yProps(0)} />
<Tab label="Item Two" {...a11yProps(1)} />
<Tab label="Item Three" {...a11yProps(2)} />
</Tabs>
</Box>
<TabPanel value={value} index={0}>
Item One
</TabPanel>
<TabPanel value={value} index={1}>
Item Two
</TabPanel>
<TabPanel value={value} index={2}>
Item Three
</TabPanel>
</Box>
);
Ln 57, Col 29 (5 selected) Spaces: 4 UTF-8 CRLF JavaScript Go Live
Type here to search
32°C Haze
11:05 AM
7/4/2023



Material UI Layout

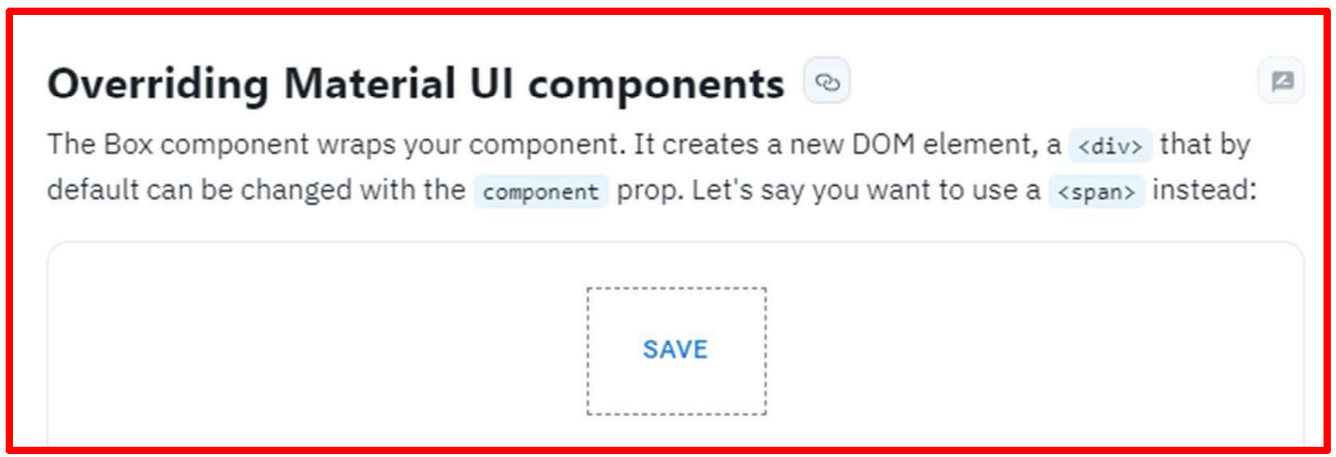
- Layout Management is a Specification which determines the Placement and Size of the Components on the UI.

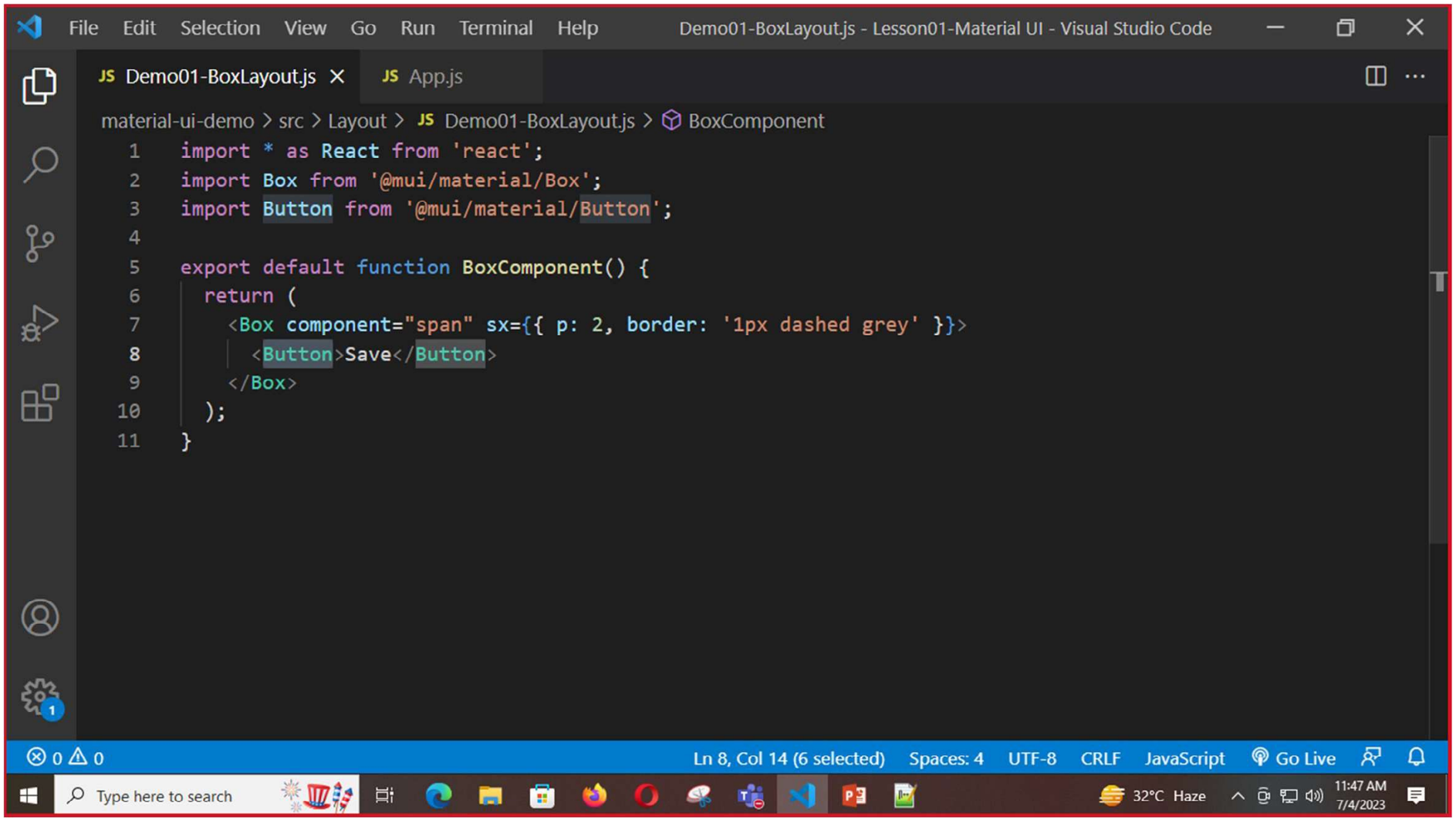
- Eg: Flow Layout, Border Layout, Box Layout, Grid Layout, GridBag Layout, CardLayout, Spring Layout, Mansory Layout, Water Fall Layout …

Box

- The Box component serves as a wrapper component for most of the CSS utility needs.

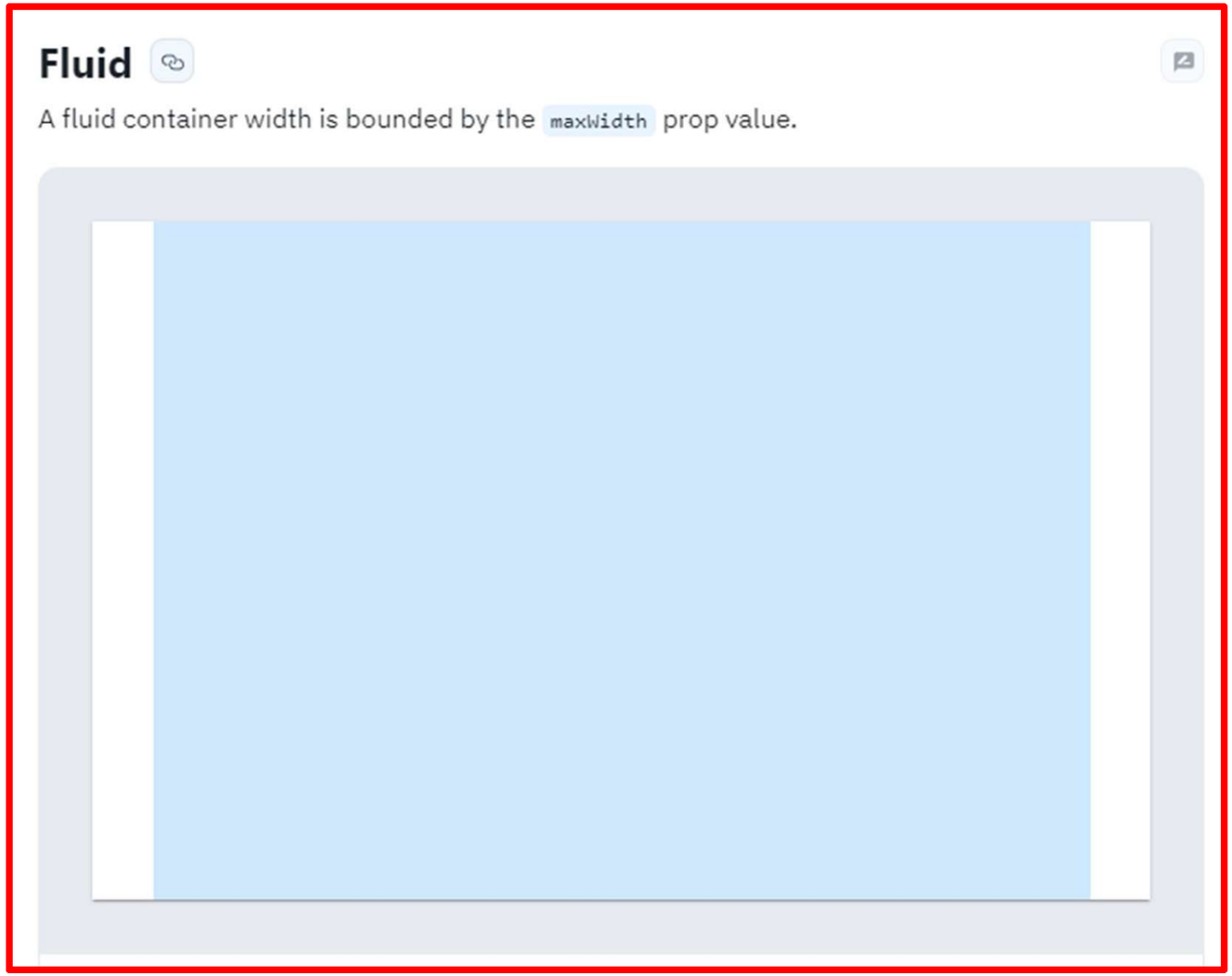
- The Box component packages all the style functions that are exposed in @mui/system

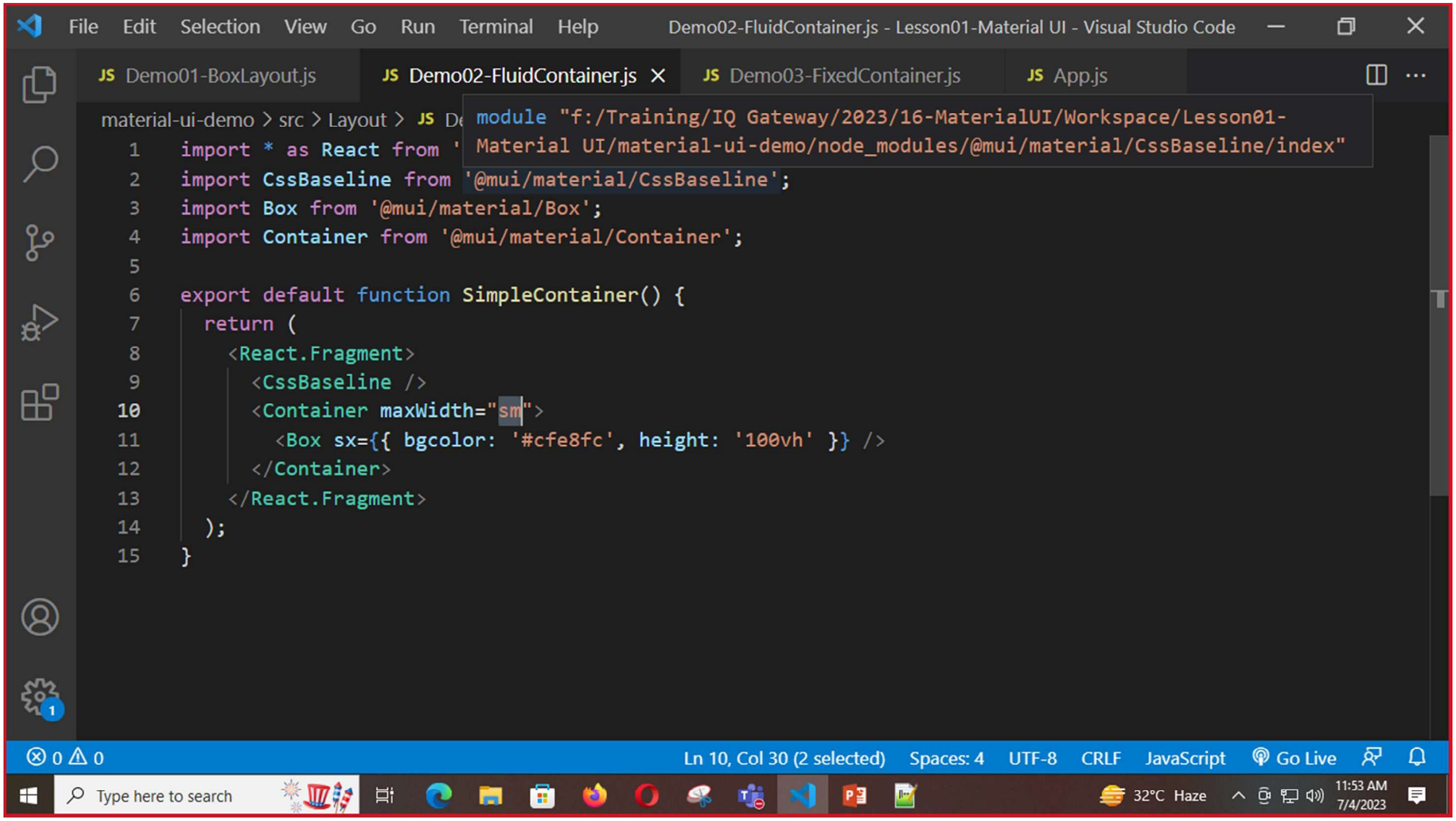



File Edit Selection View Go Run Terminal Help
Demo01-BoxLayout.js - Lesson01-Material UI - Visual Studio Code
Demo01-BoxLayout.js
App.js
material-ui-demo > src > Layout > Demo01-BoxLayout.js > BoxComponent
1 import * as React from 'react';
2 import Box from '@mui/material/Box';
3 import Button from '@mui/material/Button';
4
5 export default function BoxComponent() {
6 return (
7 <Box component="span" sx={{ p: 2, border: '1px dashed grey' }}>
8 <Button>Save</Button>
9 </Box>
10);
11 }
Ln 8, Col 14 (6 selected) Spaces: 4 UTF-8 CRLF JavaScript Go Live
Type here to search
32°C Haze
11:47 AM
7/4/2023

Container

- The container centers your content horizontally. It's the most basic layout element.
- While containers can be nested, most layouts do not require a nested container.

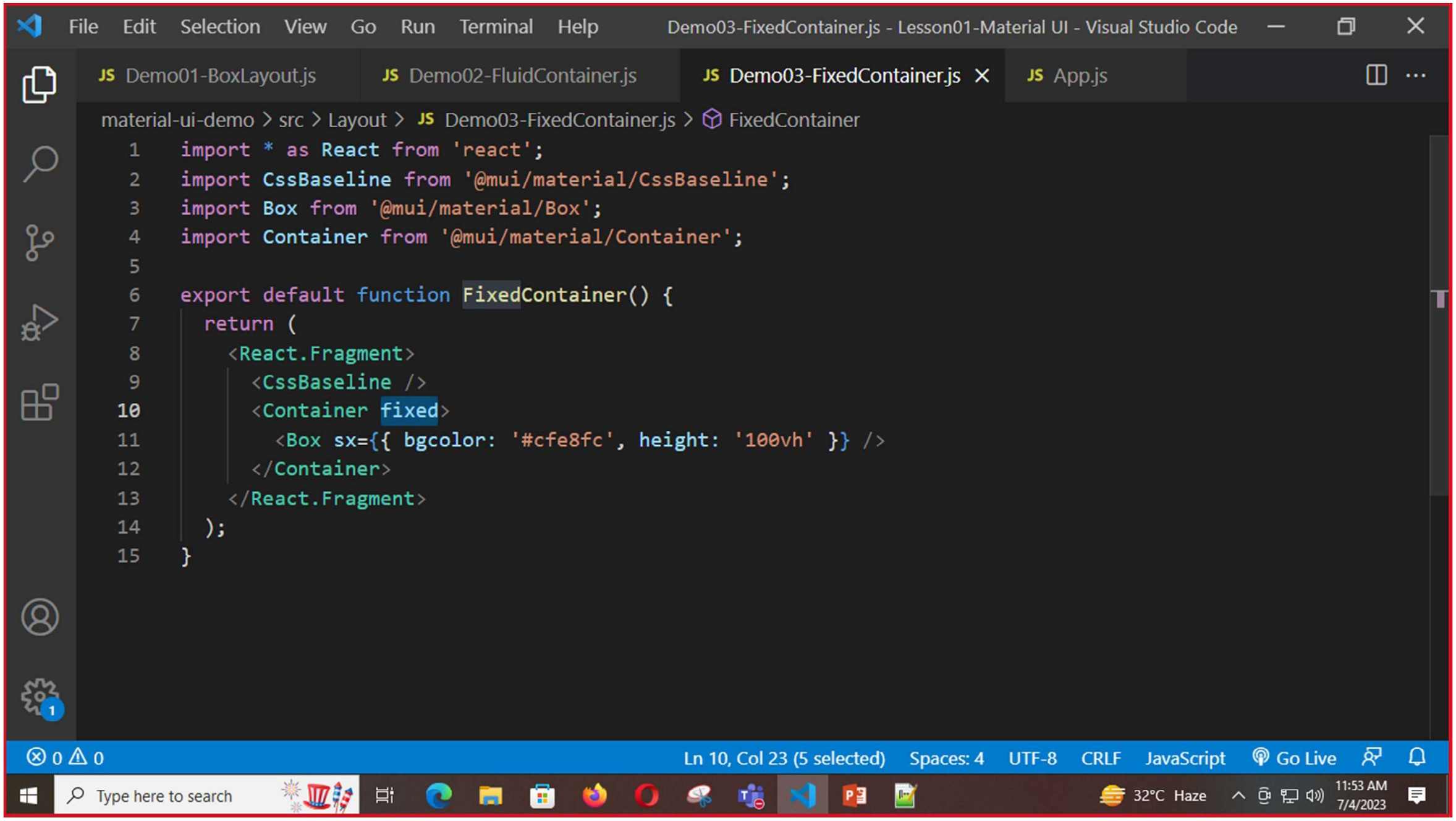

Fluid
A fluid container width is bounded by the maxWidth prop value.


File Edit Selection View Go Run Terminal Help
Demo02-FluidContainer.js - Lesson01-Material UI - Visual Studio Code
Demo01-BoxLayout.js
Demo02-FluidContainer.js
Demo03-FixedContainer.js
App.js
material-ui-demo > src > Layout > JS De
module "f:/Training/IQ Gateway/2023/16-MaterialUI/Workspace/Lesson01-Material UI/material-ui-demo/node_modules/@mui/material/CssBaseline/index"
1 import * as React from '
2 import CssBaseline from '@mui/material/CssBaseline';
3 import Box from '@mui/material/Box';
4 import Container from '@mui/material/Container';
5
6 export default function SimpleContainer() {
7 return (
8 <React.Fragment>
9 <CssBaseline />
10 <Container maxWidth="sm">
11 <Box sx={{ bgcolor: '#cfe8fc', height: '100vh' }} />
12 </Container>
13 </React.Fragment>
14);
15 }
0 0
Ln 10, Col 30 (2 selected)
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live
Type here to search
32°C Haze
11:53 AM
7/4/2023

Fixed

Fixed

If you prefer to design for a fixed set of sizes instead of trying to accommodate a fully fluid viewport, you can set the `fixed` prop. The max-width matches the min-width of the current breakpoint.


File Edit Selection View Go Run Terminal Help
Demo03-FixedContainer.js - Lesson01-Material UI - Visual Studio Code
Demo01-BoxLayout.js
Demo02-FluidContainer.js
Demo03-FixedContainer.js
App.js
material-ui-demo > src > Layout > Demo03-FixedContainer.js > FixedContainer
1 import * as React from 'react';
2 import CssBaseline from '@mui/material/CssBaseline';
3 import Box from '@mui/material/Box';
4 import Container from '@mui/material/Container';
5
6 export default function FixedContainer() {
7 return (
8 <React.Fragment>
9 <CssBaseline />
10 <Container fixed>
11 <Box sx={{ bgcolor: '#cfe8fc', height: '100vh' }} />
12 </Container>
13 </React.Fragment>
14);
15 }
0 0
Ln 10, Col 23 (5 selected)
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live
Type here to search
32°C Haze
11:53 AM
7/4/2023

Grid / Flex

- The Material Design responsive layout grid adapts to screen size and orientation, ensuring consistency across layouts.
- The grid creates visual consistency between layouts while allowing flexibility across a wide variety of designs. Material Design's responsive UI is based on a 12-column grid layout.

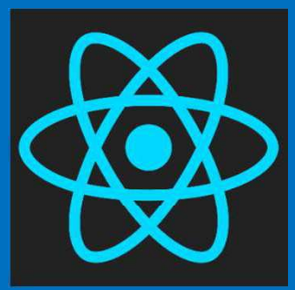
How it works

1. It uses CSS's Flexible Box module for high flexibility.
2. There are two types of layout: *containers* and *items*.
3. Item widths are set in percentages, so they're always fluid and sized relative to their parent element.
4. Items have padding to create the spacing between individual items.
5. There are five grid breakpoints: xs, sm, md, lg, and xl.
6. Integer values can be given to each breakpoint, indicating how many of the 12 available columns are occupied by the component when the viewport width satisfies the breakpoint constraints.

Basic grid

Column widths are integer values between 1 and 12; they apply at any breakpoint and indicate how many columns are occupied by the component.

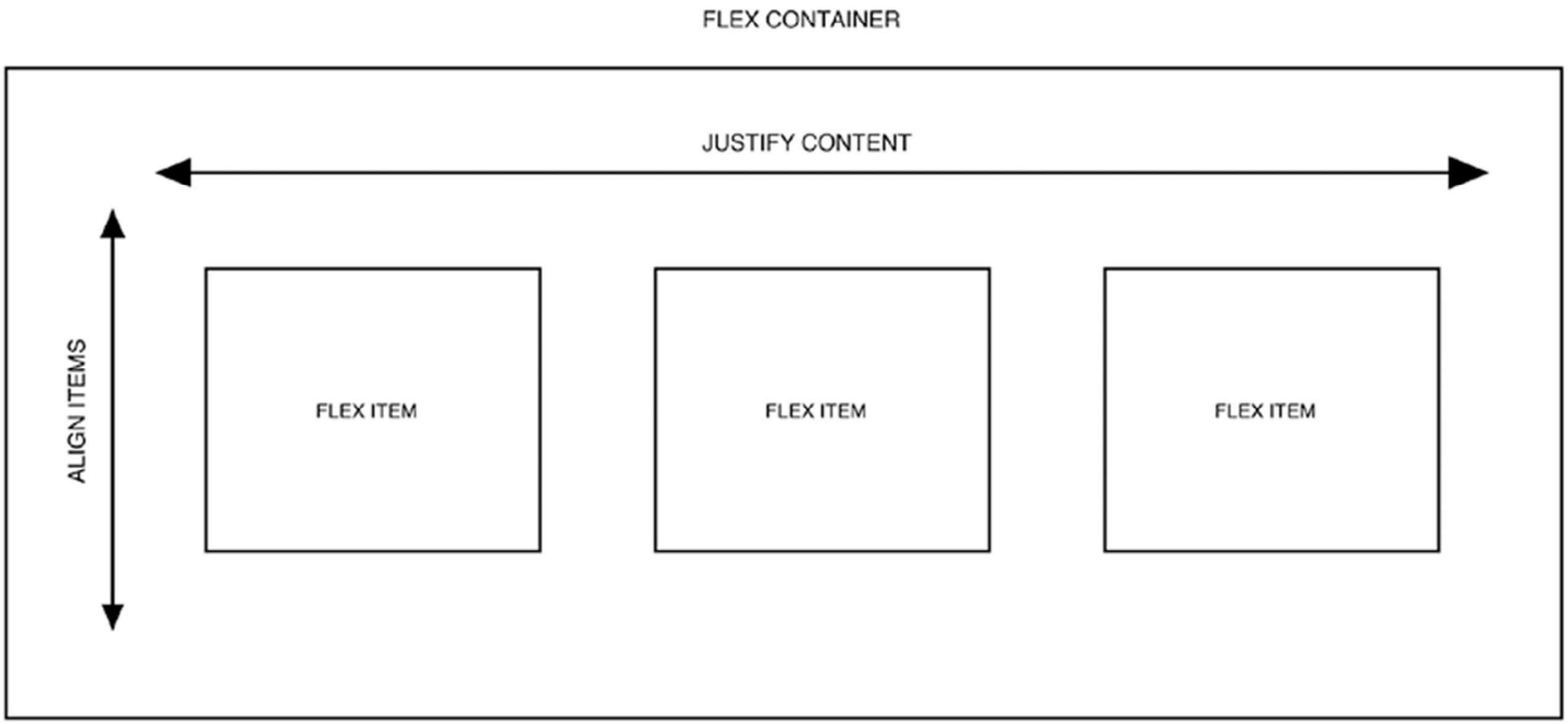
A value given to a breakpoint applies to all the other breakpoints wider than it (unless overridden, as you can read later in this page). For example, `xs={12}` sizes a component to occupy the whole viewport width regardless of its size.



Flex Box

Flexbox

- There was once a time when web development was a lot simpler, as the number of different devices and screen sizes were limited.
- With the introduction of smartphones in 2007, and the wave of different screen sizes that followed, a shift began toward the responsive web.
- CSS3 introduced a new layout mode (an alternative to floats and positioning) called flexbox, which is an easy and responsive method of arranging elements on a page.
- To use flexbox, you simply have to specify the CSS attribute display: flex on the container elements, and any elements within the container (also called flex items) will automatically align into separate columns.


FLEX CONTAINER
JUSTIFY CONTENT
ALIGN ITEMS
FLEX ITEM
FLEX ITEM
FLEX ITEM

The Flex Attribute

- Once the flex container has been set up, the elements inside the container (the flex items) can have a flex property set against them.
- The flex property is a combination (or shorthand) of the flex-grow, flex-shrink, and flex-basis properties.
- These three properties are responsible for determining how much space flex items should be taking up within the container.

Cont …

1. flex-grow: Number value that specifies how much the item will grow relative to the rest of the children within the container
2. flex-shrink: Number value that specifies how much the item will shrink relative to the rest of the children within the container
3. flex-basis: The length of the item; can be auto, inherit or a number followed by a length unit

The default values for the flex attribute are 0, 1, auto. This means that flex-grow is set to 0; flex-shrink is set to 1; and the basis is auto.

align-items

The align-items attribute aligns items vertically within a flexbox.

- `center`: Positions the children in the center of the container
- `flex-start`: Positions the children at the beginning (top) of the container
- `flex-end`: Positions the children at the end (bottom) of the container
- `baseline`: Positions the children so that the baselines align
- `stretch` *(default)*: The children stretch to fill the container.

justify-content

justify-content will position the flex items horizontally.

- If you struggle trying to remember the difference between align and justify, try to remember that justify positions flex items in the same way that the justify positioning works within Microsoft Word.

- `center`: Positions the flex items in the center of the container
- `flex-start` *(default)*: Positions the flex items at the beginning (left) of the container
- `flex-end`: Positions the flex items at the end (right) of the container
- `space-between`: Spreads the flex items evenly across the width of the container
- `space-around`: Spreads the flex items evenly across the width of the container, but with space around the edges of the items

flex-direction

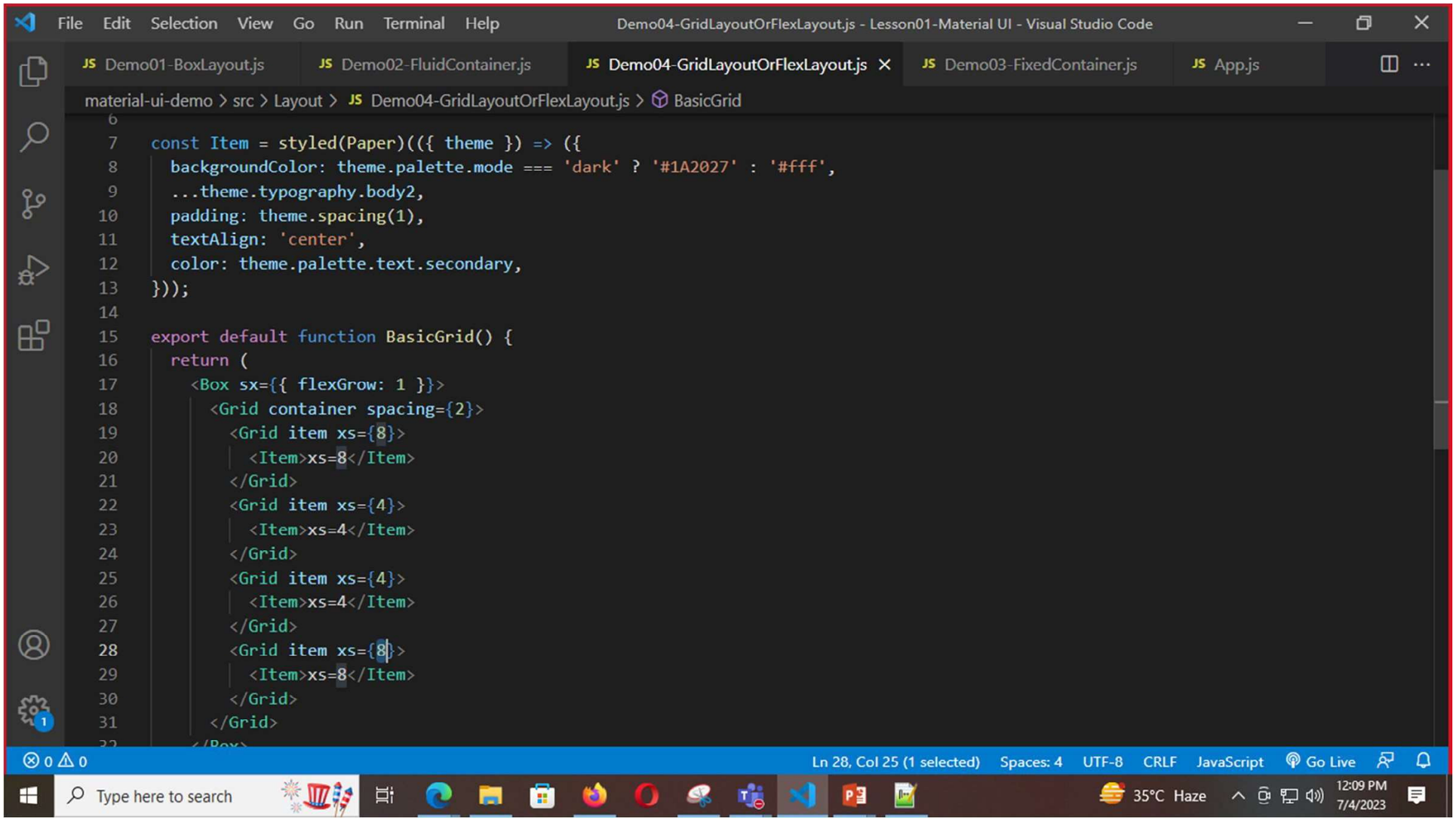
When setting a container to flex, the default direction is row, meaning the children will display side by side horizontally. Using the flex-direction attribute, you can change the default direction of the children to any of the following values:

- `row` *(default)*: Positions the children next to each other horizontally
- `row-reverse`: Positions the children next to each other horizontally, but in reverse order
- `column`: Positions the children under each other vertically, as a column
- `column-reverse`: Positions the children under each other vertically, as a column, but in reverse order

flex-wrap

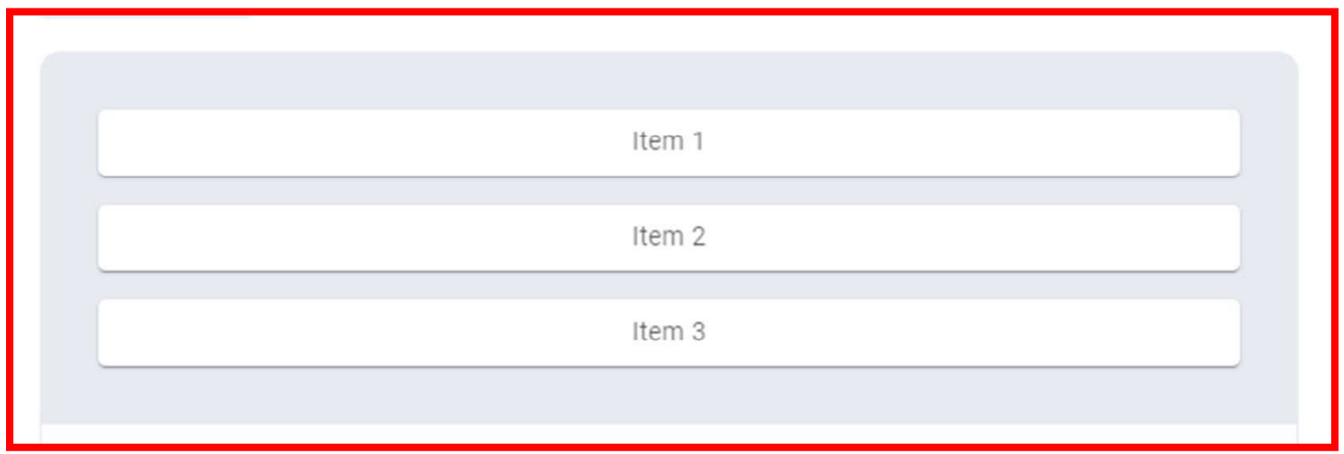
The flex-wrap attribute specifies whether the flex items remain in the same row, and overflow once there are too many, or whether they wrap onto the next line. The attribute has three values:

- `nowrap` *(default)*: Items will not wrap.
- `wrap`: Items will wrap, if needed, in relation to the direction set by the `flex-direction` attribute.
- `wrap-reverse`: Items will wrap, if needed, in reverse order.


File Edit Selection View Go Run Terminal Help
Demo04-GridLayoutOrFlexLayout.js - Lesson01-Material UI - Visual Studio Code
Demo01-BoxLayout.js
Demo02-FluidContainer.js
Demo04-GridLayoutOrFlexLayout.js
Demo03-FixedContainer.js
App.js
material-ui-demo > src > Layout > Demo04-GridLayoutOrFlexLayout.js > BasicGrid
const Item = styled(Paper)(({ theme }) => ({
backgroundColor: theme.palette.mode === 'dark' ? '#1A2027' : '#fff',
...theme.typography.body2,
padding: theme.spacing(1),
textAlign: 'center',
color: theme.palette.text.secondary,
}));
export default function BasicGrid() {
return (
<Box sx={{ flexGrow: 1 }}>
<Grid container spacing={2}>
<Grid item xs={8}>
<Item>xs=8</Item>
</Grid>
<Grid item xs={4}>
<Item>xs=4</Item>
</Grid>
<Grid item xs={4}>
<Item>xs=4</Item>
</Grid>
<Grid item xs={8}>
<Item>xs=8</Item>
</Grid>
</Grid>
Ln 28, Col 25 (1 selected) Spaces: 4 UTF-8 CRLF JavaScript Go Live
Type here to search
35°C Haze
12:09 PM
7/4/2023

Stack

- Stack is a container component for arranging elements vertically or horizontally.
- The Stack component manages the layout of its immediate children along the vertical or horizontal axis, with optional spacing and dividers between each child.



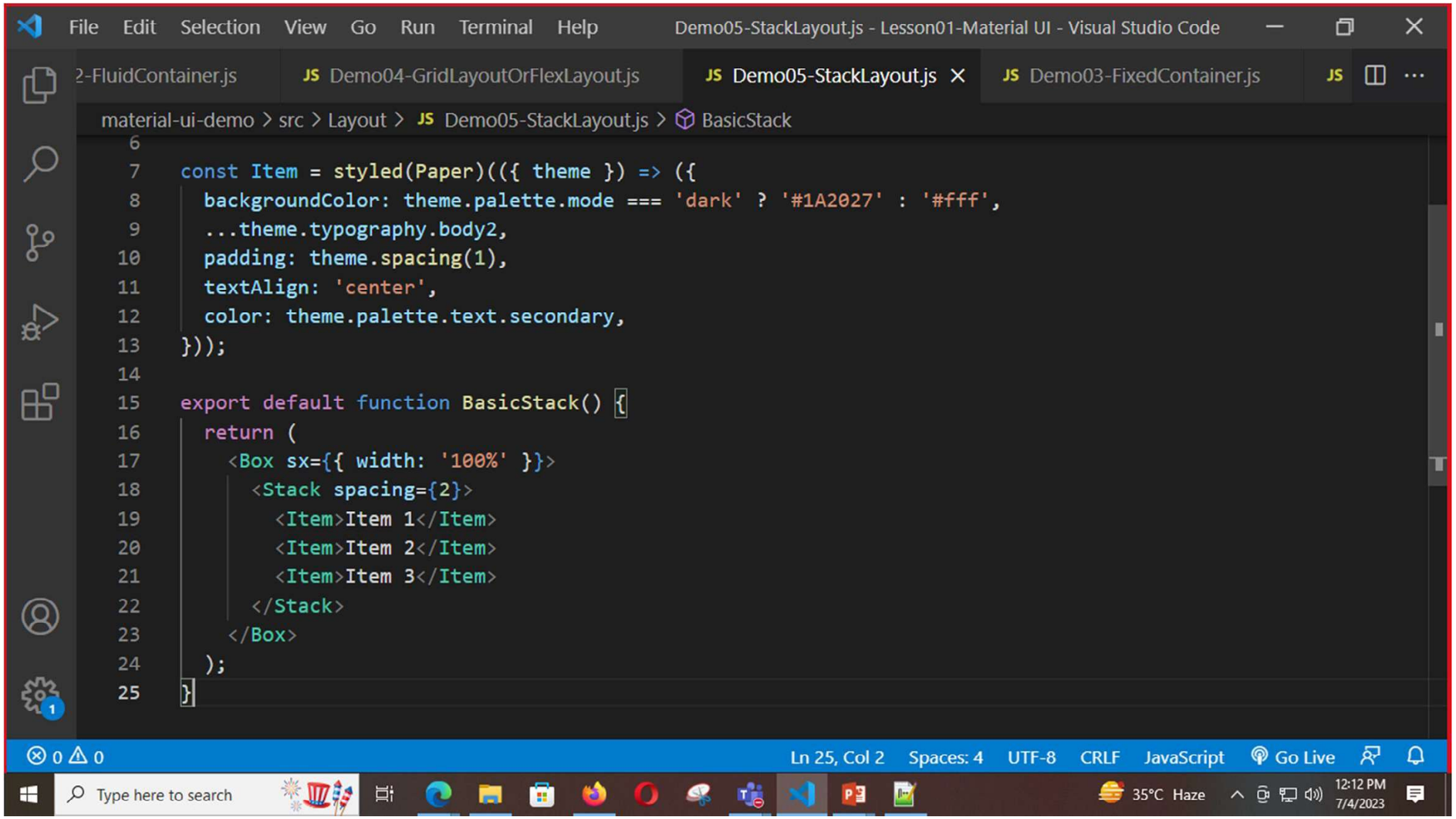
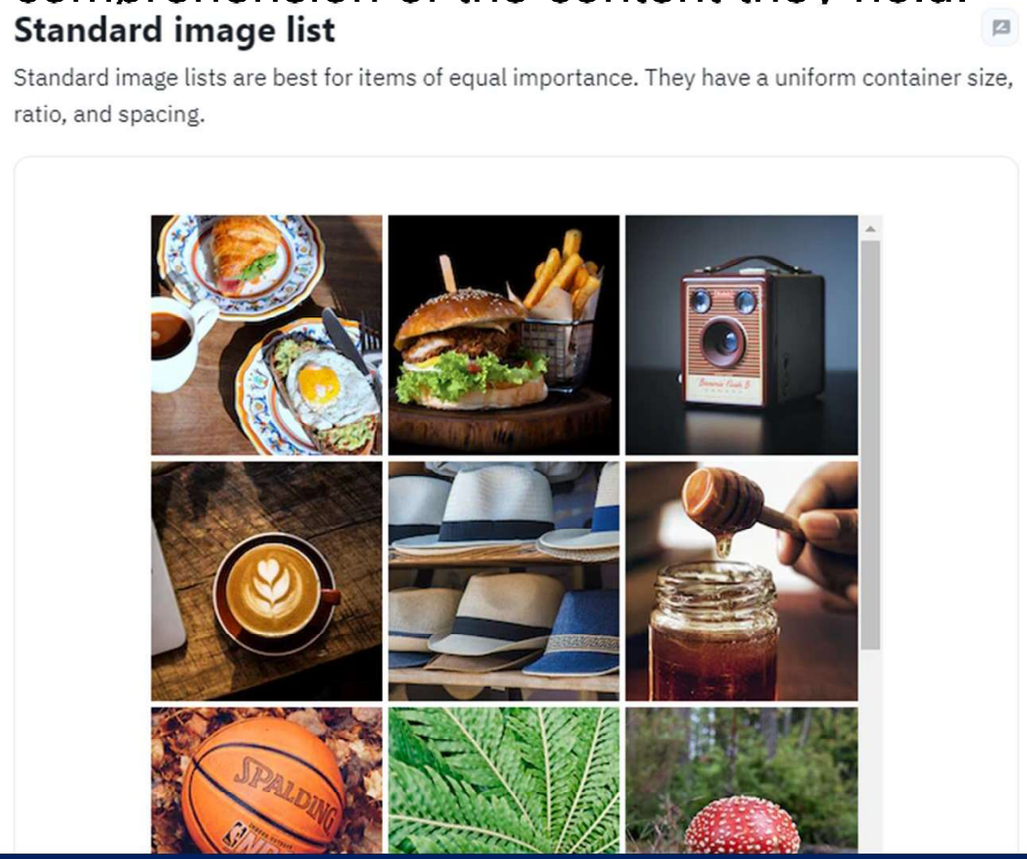
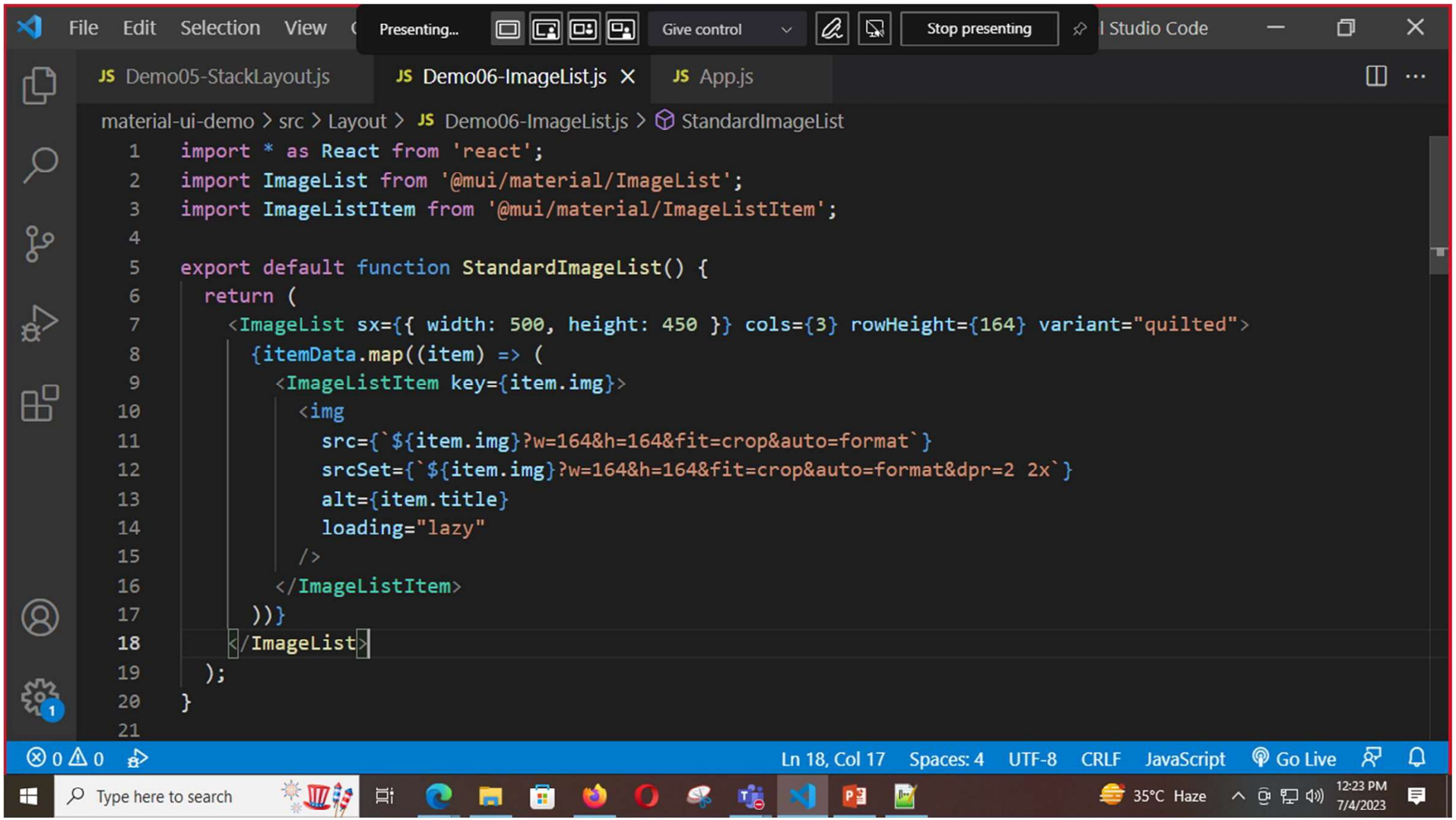

File Edit Selection View Go Run Terminal Help
Demo05-StackLayout.js - Lesson01-Material UI - Visual Studio Code
2-FluidContainer.js
Demo04-GridLayoutOrFlexLayout.js
Demo05-StackLayout.js
Demo03-FixedContainer.js
material-ui-demo > src > Layout > Demo05-StackLayout.js > BasicStack
6
7 const Item = styled(Paper)(({ theme }) => ({
8 backgroundColor: theme.palette.mode === 'dark' ? '#1A2027' : '#fff',
9 ...theme.typography.body2,
10 padding: theme.spacing(1),
11 textAlign: 'center',
12 color: theme.palette.text.secondary,
13 }));
14
15 export default function BasicStack() {
16 return (
17 <Box sx={{ width: '100%' }}>
18 <Stack spacing={2}>
19 <Item>Item 1</Item>
20 <Item>Item 2</Item>
21 <Item>Item 3</Item>
22 </Stack>
23 </Box>
24);
25 }
0 0
Ln 25, Col 2
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live
Type here to search
35°C Haze
12:12 PM
7/4/2023

Image List

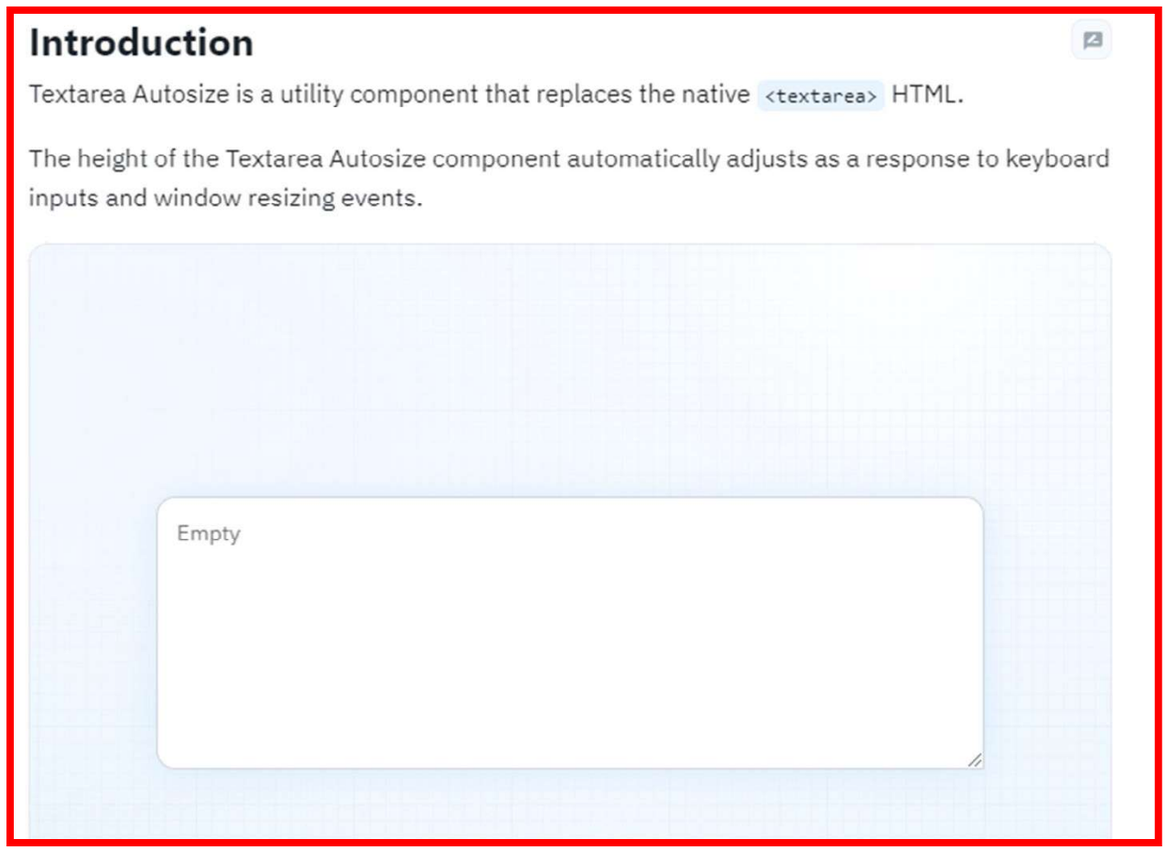
- The Image List displays a collection of images in an organized grid.
- Image lists represent a collection of items in a repeated pattern. They help improve the visual comprehension of the content they hold.




File Edit Selection View
Presenting...
Give control
Stop presenting
Studio Code
Demo05-StackLayout.js
Demo06-ImageList.js
App.js
material-ui-demo > src > Layout > Demo06-ImageList.js > StandardImageList
import * as React from 'react';
import ImageList from '@mui/material/ImageList';
import ImageListItem from '@mui/material/ImageListItem';
export default function StandardImageList() {
return (
<ImageList sx={{ width: 500, height: 450 }} cols={3} rowHeight={164} variant="quilted">
{itemData.map((item) => (
<ImageListItem key={item.img}>
<img
src={`${item.img}?w=164&h=164&fit=crop&auto=format`}
srcSet={`${item.img}?w=164&h=164&fit=crop&auto=format&dpr=2 2x`}
alt={item.title}
loading="lazy"
/>
</ImageListItem>
))}
</ImageList>
);
}
Ln 18, Col 17
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live
Type here to search
35°C Haze
12:23 PM
7/4/2023

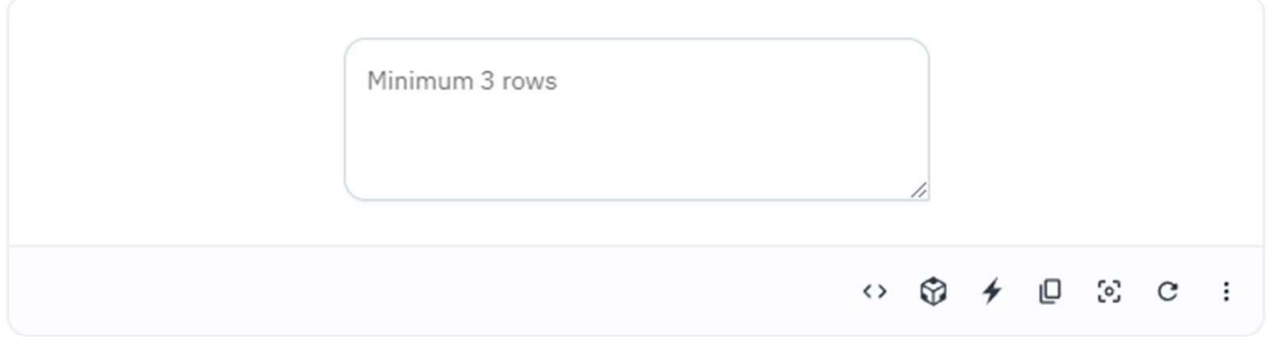
Textarea Autosize

- The Textarea Autosize component gives you a textarea HTML element that automatically adjusts its height to match the length of the content within.



Minimum height

Use the `minRows` prop to define the minimum height of the component:




```
<StyledTextarea
  aria-label="minimum height"
  minRows={3}
  placeholder="Minimum 3 rows"
/>
```