

Week 3 – React.js

Acknowledgement of Country

We respectfully acknowledge the Wurundjeri People of the Kulin Nation, who are the Traditional Owners of the land on which Swinburne's Australian campuses are located in Melbourne's east and outer-east, and pay our respect to their Elders past, present and emerging.

We are honoured to recognise our connection to Wurundjeri Country, history, culture, and spirituality through these locations, and strive to ensure that we operate in a manner that respects and honours the Elders and Ancestors of these lands.

We also respectfully acknowledge Swinburne's Aboriginal and Torres Strait Islander staff, students, alumni, partners and visitors.

We also acknowledge and respect the Traditional Owners of lands across Australia, their Elders, Ancestors, cultures, and heritage, and recognise the continuing sovereignties of all Aboriginal and Torres Strait Islander Nations.



FAQ on Projects

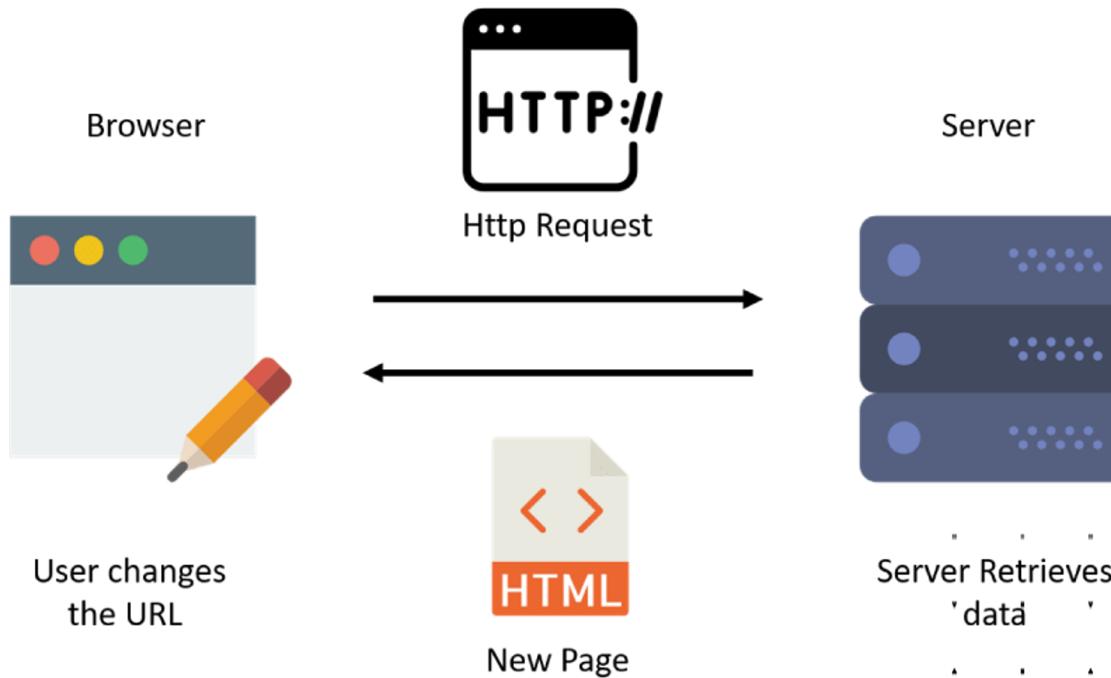
1. About the language you choose to develop the front-end page
2. About the Project Scope & Citation
3. About the Project Prototype
4. About the Project Data

Overall System Design

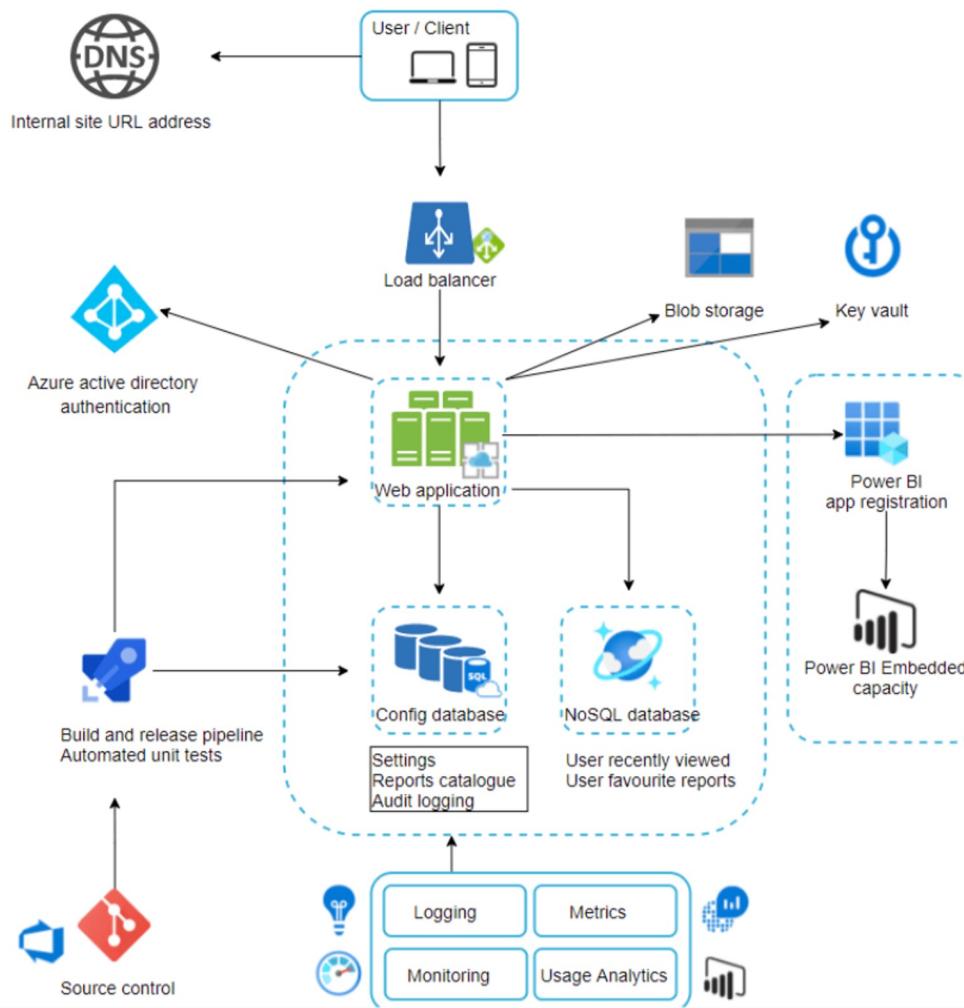
In a web system, the concept of ***Overall System Design*** involves creating a comprehensive blueprint that outlines the collaboration among various elements like components, modules, and services. This blueprint aims to ensure the intended features, performance, scalability, and ease of maintenance are achieved through a well-organized architecture and structure.

- **Architecture:** Choose an appropriate architectural style, such as monolithic, microservices, serverless, or event-driven, based on the system's requirements and complexity.
- **Components and Modules:** Identify the major components, modules, and services that make up the system. Define their responsibilities and interactions.
- **Data Management:** Design the data model, including database schemas, data storage mechanisms (relational databases, NoSQL databases, etc.), and data access patterns.
- **User Interface (UI):** Plan the user interface design, user experience (UX), navigation flow, and interaction patterns to ensure an intuitive and user-friendly interface.

Overall System Design



Overall System Design



Structure of React

- node_modules: dependencies
- public folder: store static file (html, img, video)
- src folder: React source code, most of your codes should be here

```
cos30049
├── node_modules
└── public
    ├── favicon.ico
    ├── index.html
    ├── logo192.png
    ├── logo512.png
    ├── manifest.json
    └── robots.txt
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    ├── reportWebVitals.js
    ├── setupTests.js
    └── .gitignore
    └── package-lock.json
    └── package.json
    └── README.md
```

Structure of React

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows the project structure:
 - CODE
 - cos30049
 - > node_modules
 - > public
 - src
 - # App.css
 - JS App.js
 - 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:** The file `App.js` is open, showing the following code:

```
cos30049 > src > JS App.js > ...
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           | Edit <code>src/App.js</code> and save to reload.
11         </p>
12         <a
13           className="App-link"
14           href="https://reactjs.org"
15           target="_blank"
16           rel="noopener noreferrer"
17         >
18           Learn React
19         </a>
20       </header>
21     </div>
22   );
23 }
24
25 export default App;
26
```
- Terminal:** The terminal shows the output of the build process:

```
Compiled successfully!
You can now view cos30049 in the browser.
  Locals: http://localhost:3000
  On Your Network: http://192.168.1.134:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.
babel-preset-react-app is part of the create-react-app project, which
is no longer maintained anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.

^C
(base) xuzhiy@xuzhiy-MBP cos30049 % npm install nodemon -g
added 34 packages in 3s
3 packages are looking for funding
  run 'npm fund' for details
(base) xuzhiy@xuzhiy-MBP cos30049 %
```



App.js: The entry point of the React.js

Run the project: npm start

Structure of React

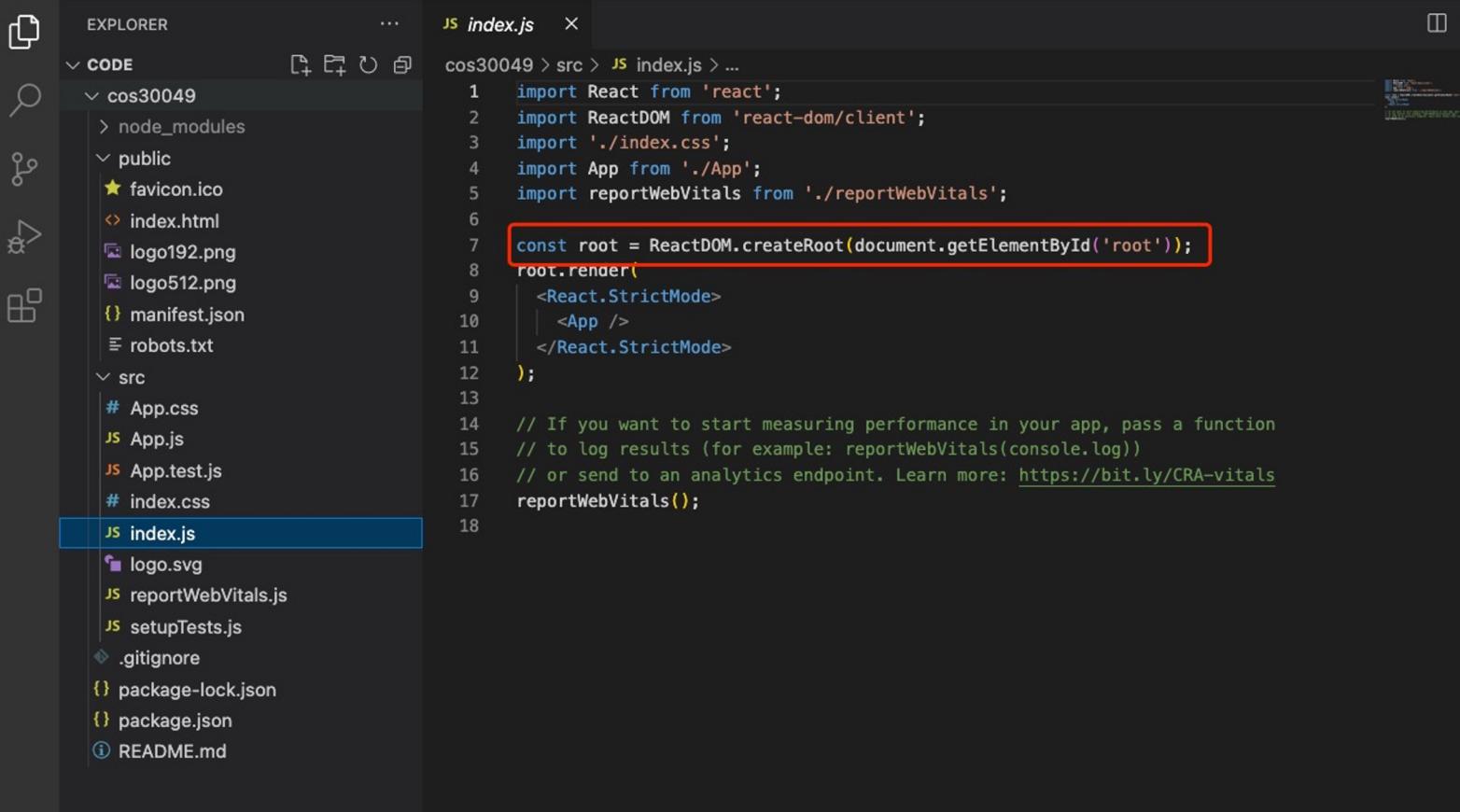
The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows the project structure with files like node_modules, public, src, App.css, App.js, App.test.js, index.css, index.js, logo.svg, reportWebVitals.js, setupTests.js, .gitignore, package-lock.json, package.json, and README.md.
- CODE** tab: The package.json file is open.
- Content of package.json:**

```
1  {
2    "name": "cos30049",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@testing-library/jest-dom": "^5.17.0",
7      "@testing-library/react": "^13.4.0",
8      "@testing-library/user-event": "^13.5.0",
9      "react": "^18.2.0",
10     "react-dom": "^18.2.0",
11     "react-scripts": "5.0.1",
12     "web-vitals": "^2.1.4"
13   },
14   "scripts": {
15     "start": "react-scripts start",
16     "build": "react-scripts build",
17     "test": "react-scripts test",
18     "eject": "react-scripts eject"
19   },
20   "eslintConfig": {
21     "extends": [
22       "react-app",
23       "react-app/jest"
24     ]
25   },
26   "browserslist": {
27     "production": [
28       ">0.2%"
29     ],
30     "development": [
31       "last 1 version"
32     ]
33   }
34 }
```

More commands can be found in package.json

Structure of React

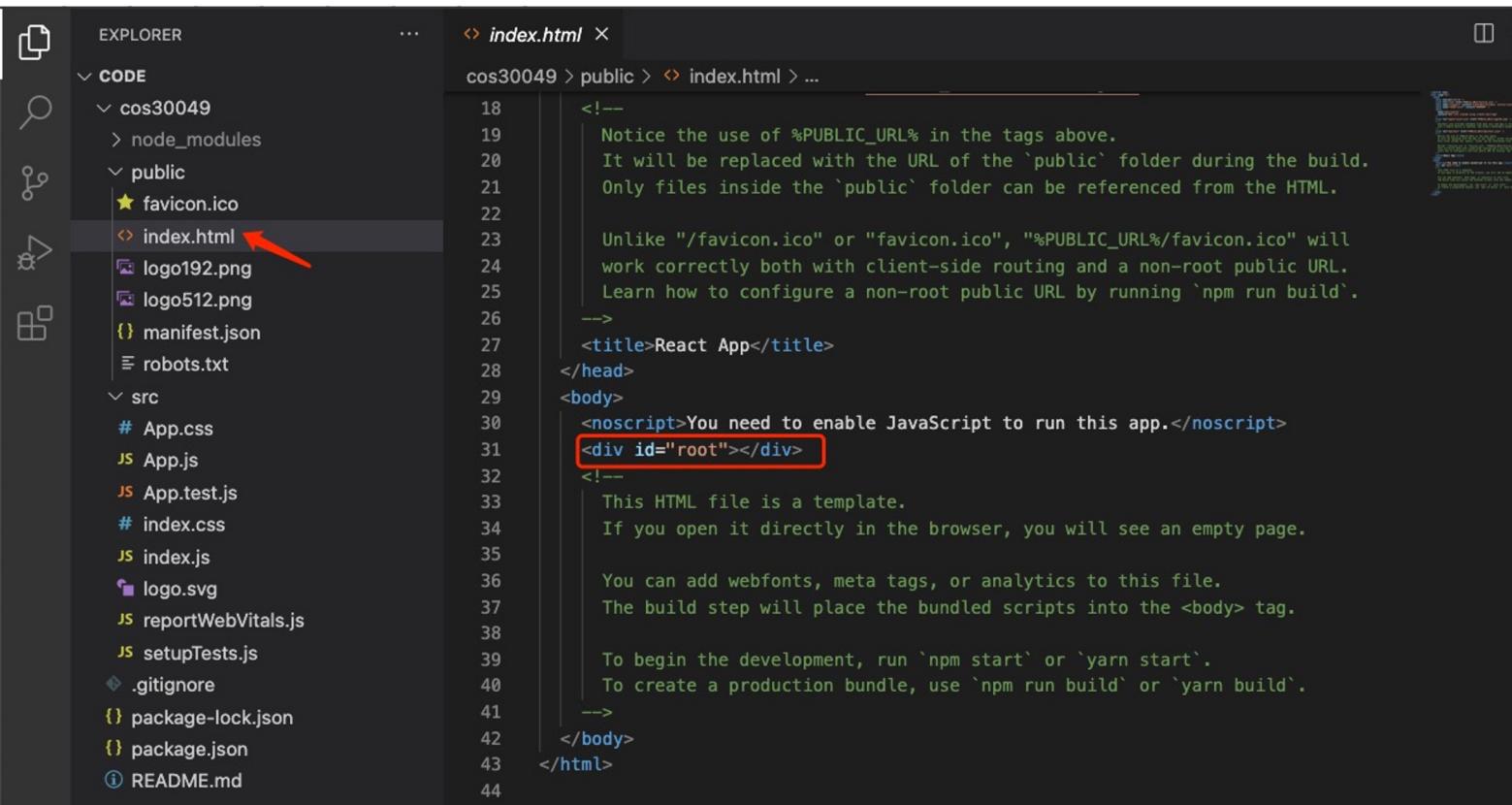


The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists project files: node_modules, public (containing favicon.ico, index.html, logo192.png, logo512.png, manifest.json, robots.txt), and src (containing App.css, App.js, App.test.js, index.css, index.js, logo.svg, reportWebVitals.js, setupTests.js, .gitignore, package-lock.json, package.json, README.md). The file index.js is currently selected and highlighted in blue. The main editor area displays the following code:

```
cos30049 > src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 const root = ReactDOM.createRoot(document.getElementById('root'));
8 root.render(
9   <React.StrictMode>
10   |   <App />
11   </React.StrictMode>
12 );
13
14 // If you want to start measuring performance in your app, pass a function
15 // to log results (for example: reportWebVitals(console.log))
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17 reportWebVitals();
18
```

The line `root.render()` is highlighted with a red rectangular box.

Structure of React



The screenshot shows a code editor interface with the following details:

- EXPLORER** panel on the left showing project files:

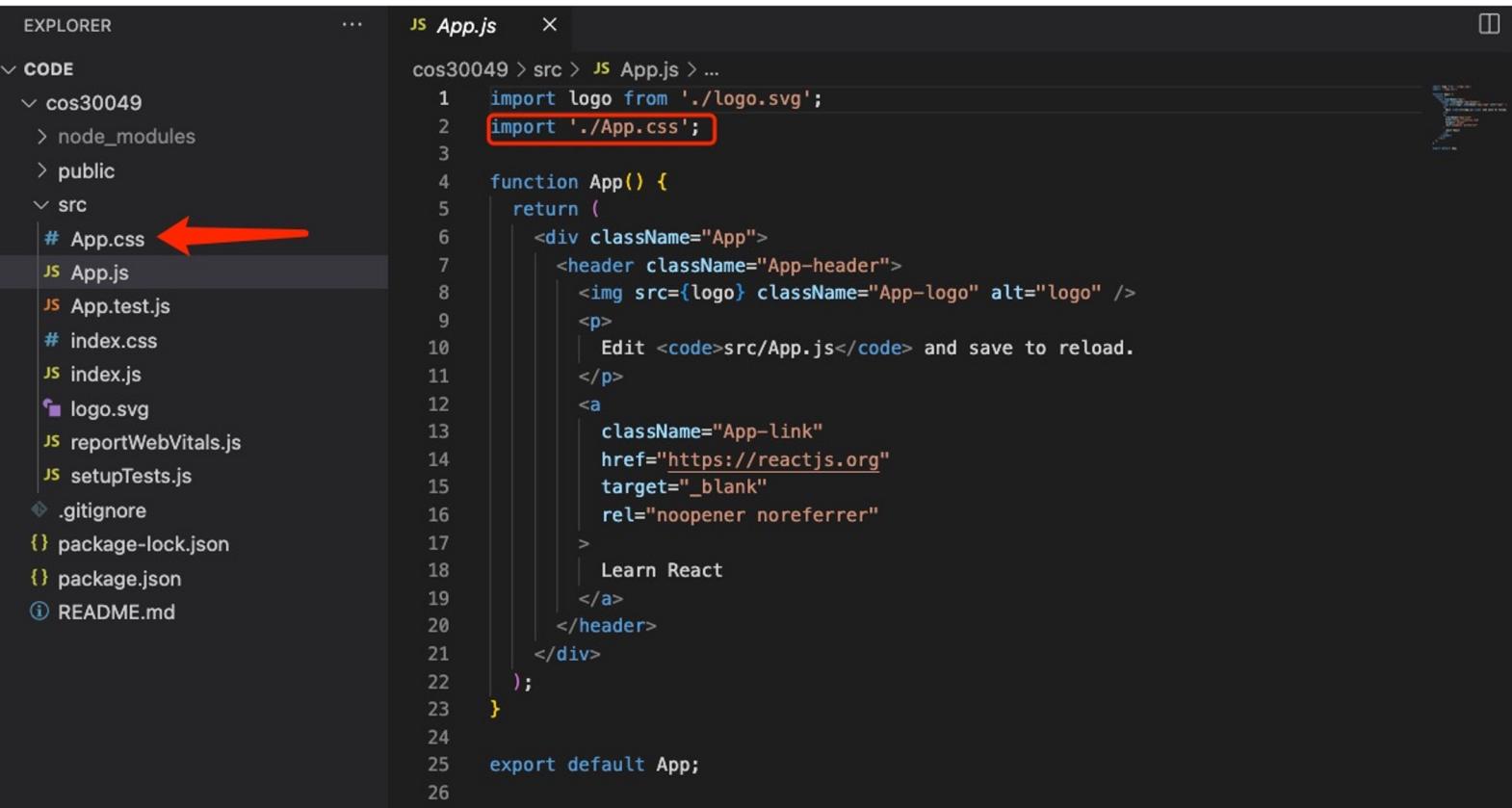
 - cos30049 (selected)
 - node_modules
 - public
 - favicon.ico
 - index.html (highlighted with a red arrow)
 - logo192.png
 - logo512.png
 - manifest.json
 - robots.txt
 - src
 - App.css
 - App.js
 - App.test.js
 - index.css
 - index.js
 - logo.svg
 - reportWebVitals.js
 - setupTests.js
 - .gitignore
 - package-lock.json
 - package.json
 - README.md

- CODE** panel on the right showing the content of `index.html`:

```
18  <!--  
19   Notice the use of %PUBLIC_URL% in the tags above.  
20   It will be replaced with the URL of the `public` folder during the build.  
21   Only files inside the `public` folder can be referenced from the HTML.  
22  
23   Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will  
24   work correctly both with client-side routing and a non-root public URL.  
25   Learn how to configure a non-root public URL by running `npm run build`.  
26  
27-->  
28<title>React App</title>  
29</head>  
30<body>  
31  <noscript>You need to enable JavaScript to run this app.</noscript>  
32  <div id="root"></div>  
33  <!--  
34   This HTML file is a template.  
35   If you open it directly in the browser, you will see an empty page.  
36  
37   You can add webfonts, meta tags, or analytics to this file.  
38   The build step will place the bundled scripts into the <body> tag.  
39  
40   To begin the development, run `npm start` or `yarn start`.  
41   To create a production bundle, use `npm run build` or `yarn build`.  
42-->  
43</body>  
44</html>
```

Styling in React

You can define the elements' CSS in separate files.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree with the following structure:

- CODE
- cos30049
- node_modules
- public
- src
 - # App.css ← (highlighted by a red arrow)
 - 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

The main editor area shows the content of `App.js`:

```
cos30049 > src > JS App.js > ...
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           Edit <code>src/App.js</code> and save to reload.
11         </p>
12         <a
13           className="App-link"
14           href="https://reactjs.org"
15           target="_blank"
16           rel="noopener noreferrer"
17         >
18           Learn React
19         </a>
20       </header>
21     </div>
22   );
23 }
24
25 export default App;
26
```

Basic Concept of React

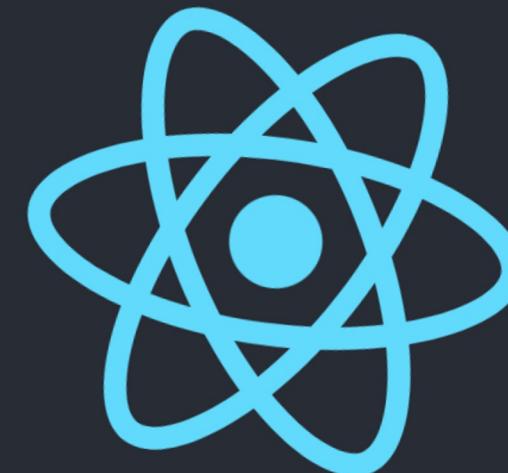
JSX (JavaScript XML) is a JavaScript syntax extension that allows you to write XML or HTML-like structures within your JavaScript code. In the context of React, JSX is widely used to describe the structure and appearance of UI components.

The purpose of JSX is to simplify the process of creating and rendering React components, making the code more readable and writable. It enables developers to write markup similar to HTML directly within JavaScript, without the need to manually construct virtual DOM elements.

```
JS App.js  ×  
cos30049 > src > JS App.js > ⚡ App  
1 import logo from './logo.svg';  
2 import './App.css';  
3  
4 function App() {  
5   return (  
6     <div className="App">  
7       <header className="App-header">  
8         <img src={logo} className="App-logo" alt="logo" />  
9         <p>  
10          Edit <code>src/App.js</code> and save to reload.  
11        </p>  
12        <a  
13          className="App-link"  
14          href="https://reactjs.org"  
15          target="_blank"  
16          rel="noopener noreferrer"  
17        >  
18          Learn React  
19        </a>  
20      </header>  
21    </div>  
22  );  
23}  
24  
25 export default App;
```

Basic Concept of React

```
cos30049 > src > JS App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5
6     let test_text = "this is test text" ←
7
8     return (
9         <div className="App">
10            <header className="App-header">
11                <img src={logo} className="App-logo" alt="logo" />
12                <p>
13                    Edit <code>src/App.js</code> and save to reload.
14                </p>
15                <a
16                    className="App-link"
17                    href="https://reactjs.org"
18                    target="_blank"
19                    rel="noopener noreferrer"
20                >
21                    {test_text} this is test text
22                </a>
23            </header>
24        </div>
25    );
26}
27
28 export default App;
```



Edit `src/App.js` and save to reload.

this is test text

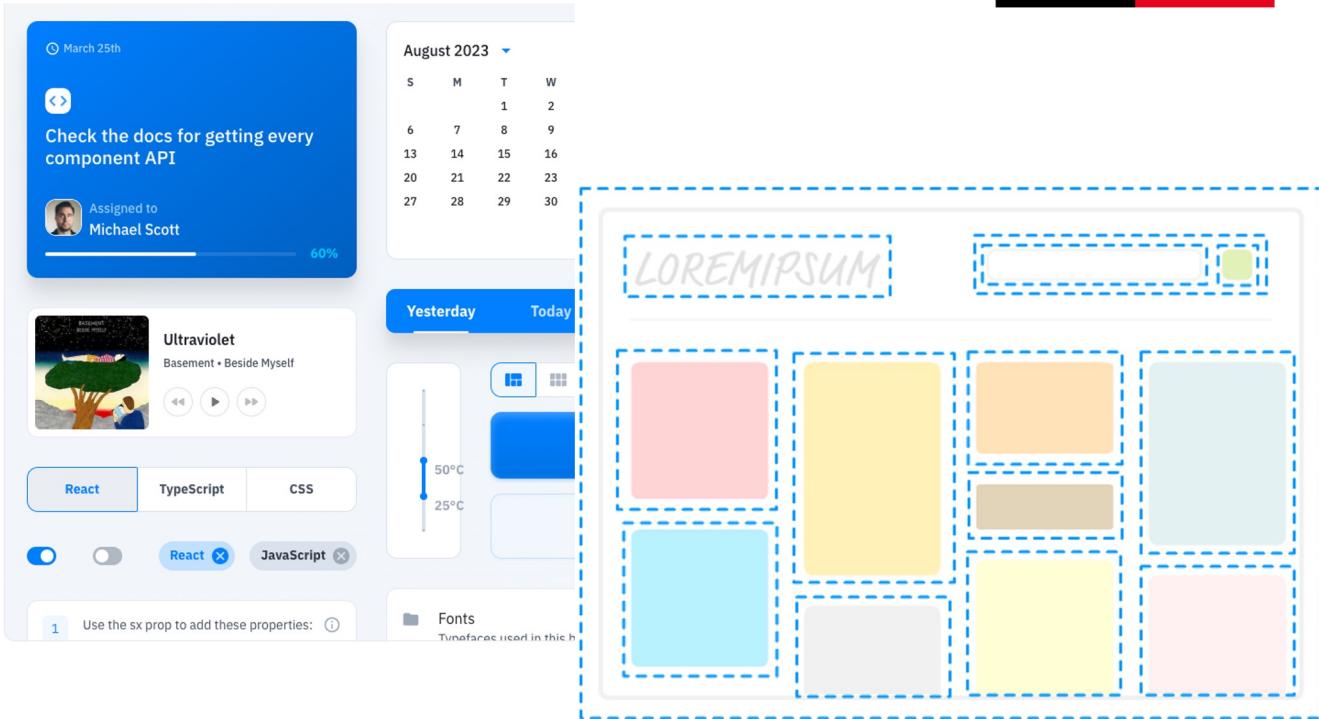
How you will build your front-end website ?

Material-UI

Move faster with intuitive React UI tools

MUI offers a comprehensive suite of free UI tools to help you ship new features faster. Start with Material UI, our fully-loaded component library, or bring your own design system to our production-ready components.

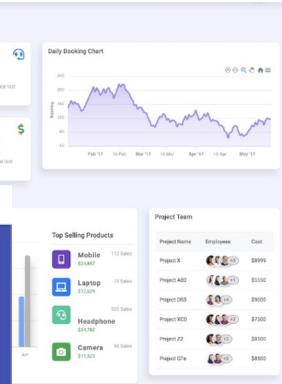
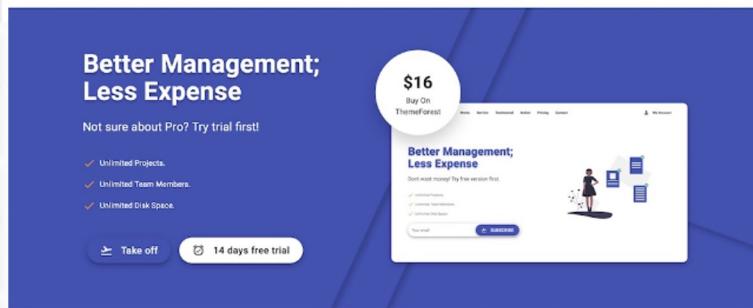
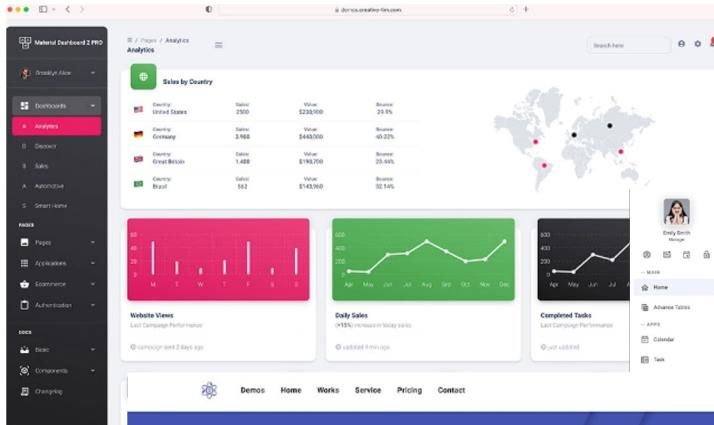
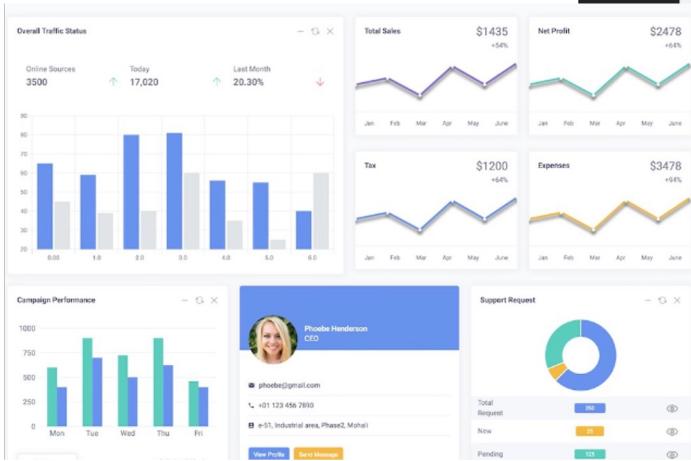
[Discover the Core libraries >](#)



That's a lot of COMPONENTS!

Material-UI

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.



Material-UI

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.



- **Responsive Design:** Material-UI components are designed to be responsive, adapting to different screen sizes from mobile to desktop, providing a seamless user experience.
- **Interactive Experience:** Material-UI components come with built-in animations and transition effects, enhancing the interaction between users and the application.
- **Rapid Development:** By using Material-UI, you can quickly create applications with a modern look without needing to write extensive CSS code from scratch.

Material-UI

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.



- **Consistent Design Language:** Material-UI follows the Material Design guidelines, providing a consistent and visually appealing design language that gives your application a modern and unified look.
- **Rich Component Library:** Material-UI offers an extensive collection of reusable components like buttons, input fields, navigation bars, cards, and more. This enables developers to rapidly build complex UI interfaces.
- **High Customizability:** While Material-UI offers default designs and styles, you can easily customize the appearance and behavior of components to meet specific project requirements.

Material-UI Elements Sample

Small Medium Large

Standard Filled Outlined

Outlined Standard Filled

 ADD TO CART

 Check out this alert!

Username
Ultraviolet

Button

Alert

Text Field

CLICK TO OPEN

Dessert	Calories
Frozen yoghurt	109
Cupcake	305

Menu

Table

Want to see more?

Check out the docs for details of the complete library.

[Learn more >](#)

Material-UI Usage

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

npm yarn pnpm

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



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 with the Google Web Fonts CDN.

npm yarn pnpm

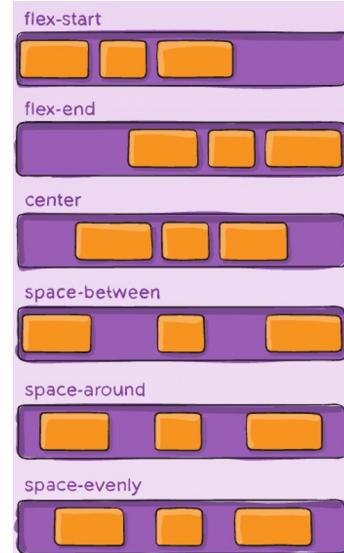
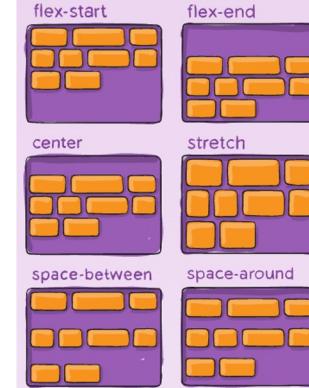
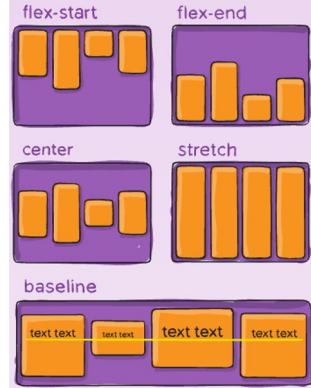
```
npm install @mui/icons-material
```



Material-UI Usage

Material-UI's grid system is employed to manage the arrangement of screens within your application. Instead of creating your own styles to control the layout of Material-UI components, you have the option to utilize the Grid component. Underneath, this component employs CSS flexbox attributes to manage adaptable layouts.

Flexbox, short for Flexible Box Layout, is a CSS module used for designing layouts. It provides a more flexible and efficient approach to arranging, aligning, and distributing elements, especially useful for building complex responsive layouts and arrangements. The goal of Flexbox is to offer developers a more intuitive and powerful layout tool to address many of the challenges posed by traditional CSS layout methods.



Material-UI Grid

In React, "**Grid**" typically refers to CSS Grid layout, which is a CSS technique used to create complex grid-based layouts. CSS Grid allows you to divide an area into rows and columns and place content within the intersections of these rows and columns, creating flexible and multi-dimensional layouts.

- **Grid Container and Grid Items:** You can declare a container element as a grid container, and its child elements become grid items. The grid container defines the overall structure of the grid, while grid items are the content placed within grid cells.
- **Row and Column Definitions:** By setting the `grid-template-rows` and `grid-template-columns` properties on the grid container, you can define the sizes and quantities of rows and columns. You can use fixed values, percentages, auto, etc.

Material-UI Grid

In React, "**Grid**" typically refers to CSS Grid layout, which is a CSS technique used to create complex grid-based layouts. CSS Grid allows you to divide an area into rows and columns and place content within the intersections of these rows and columns, creating flexible and multi-dimensional layouts.

- **Placing Content:** Using the grid-row and grid-column properties, you can determine where grid items are placed in terms of rows and columns and how many rows and columns they span. You can also use the grid-area property to assign a name to a grid area and reference that name within grid items.
- **Automatic Layout:** CSS Grid provides automatic layout capabilities, adjusting the layout dynamically based on the sizes and content of grid items.
- **Grid Gaps:** You can use the grid-row-gap and grid-column-gap properties to set spacing between rows and columns, creating better spacing arrangements.

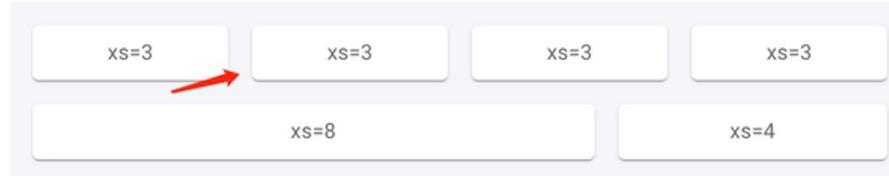
Material-UI Grid

```
<Grid container spacing={2}>
```

```
  <Grid item xs={3}>
    <Item>xs=3</Item>
  </Grid>
```

```
  <Grid item xs={8}>
    <Item>xs=8</Item>
  </Grid>
  <Grid item xs={4}>
    <Item>xs=4</Item>
  </Grid>
```

```
</Grid>
```



the Grid component includes a spacing prop that allows you to control the spacing between its items. This prop defines the spacing between grid items in terms of a CSS spacing scale provided by Material-UI. The available values for the spacing prop are usually integers ranging from 0 to 10.

Material-UI Grid

```
<Grid container spacing={2}>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

</Grid>
```

xs, sm, md, and lg are breakpoint values that represent different screen sizes. They are used to specify how the layout of grid items should adapt and change as the screen size changes.

Each breakpoint (a key) matches with a fixed screen width (a value):

- **xs**, extra-small: 0px
- **sm**, small: 600px
- **md**, medium: 900px
- **lg**, large: 1200px
- **xl**, extra-large: 1536px

Material-UI Grid

```
<Grid container spacing={2}>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xS=3</Item>
  </Grid>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xS=3</Item>
  </Grid>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xS=3</Item>
  </Grid>

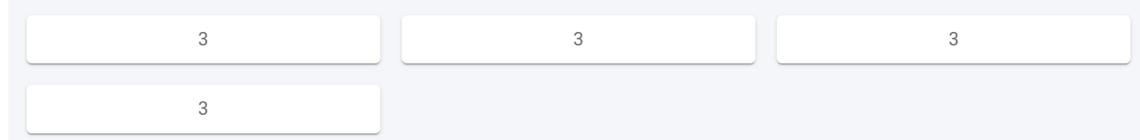
  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xS=3</Item>
  </Grid>

</Grid>
```

lg sample



md sample



Material-UI Grid

```
<Grid container spacing={2}>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

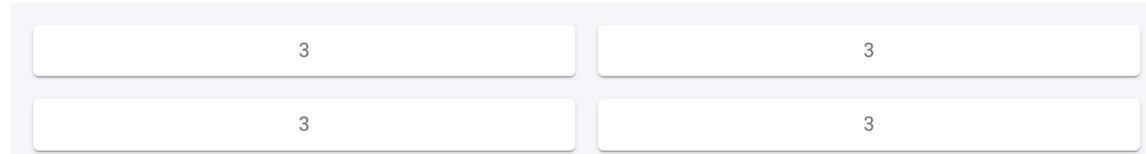
  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

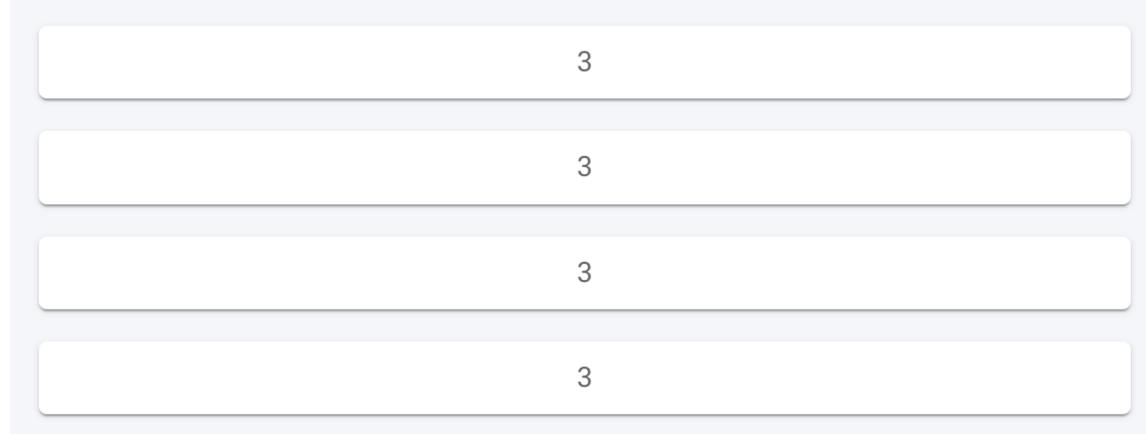
  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

</Grid>
```

sm sample



xs sample



Material-UI Grid

```
<Grid container spacing={2}>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

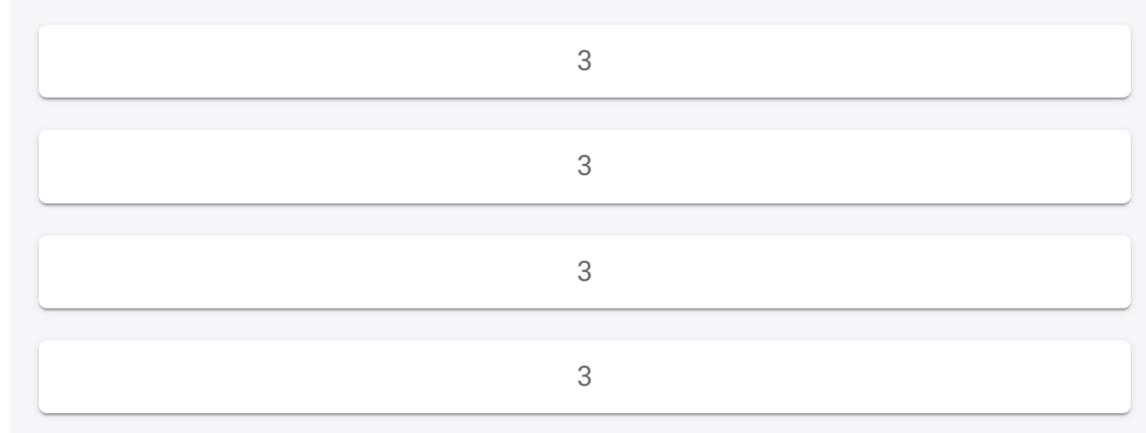
  <Grid item xs={12} sm={6} md={4} lg={3}>
    <Item>xs=3</Item>
  </Grid>

</Grid>
```

sm sample



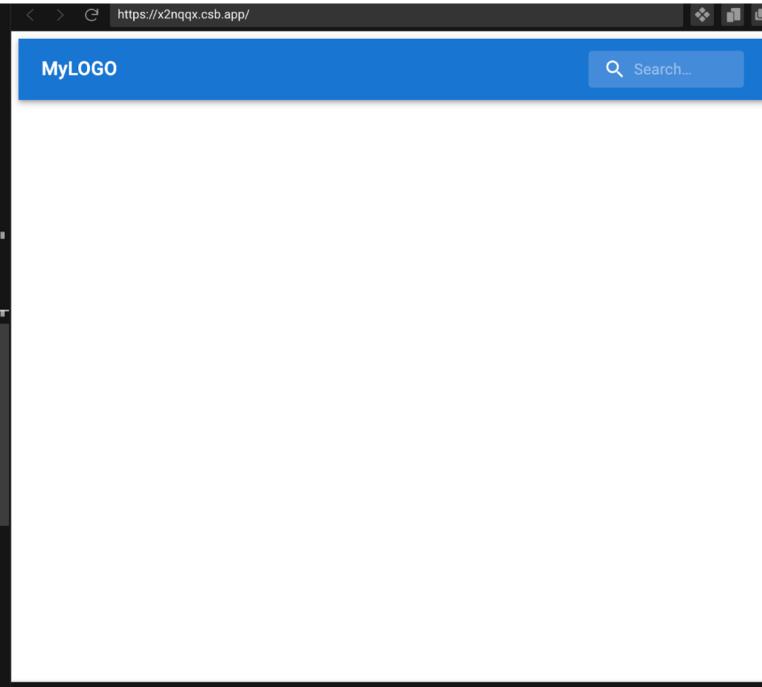
xs sample



Material-UI App Bar

App Bars serve as the foundational element in any Material-UI application. They establish context and typically remain visible to users as they navigate through the app.

```
export default function SearchAppBar() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <AppBar position="static">
        <Toolbar>
          <Typography variant="h6"
            nowrap
            component="div"
            sx={{ flexGrow: 1, display: { xs: "none", sm: "block" } }}
          >
            MyLOGO
          </Typography>
          <Search>
            <SearchIconWrapper>
              <SearchIcon />
            </SearchIconWrapper>
            <StyledInputBase
              placeholder="Search..."
              inputProps={{ "aria-label": "search" }}
            />
          </Search>
        </Toolbar>
      </AppBar>
    </Box>
  );
}
```



Material-UI AppBar

flex-grow is a property used in CSS Flexbox layout that specifies the distribution ratio of flexible items within the available space. When the container doesn't have enough space to accommodate all flexible items, flex-grow determines how each item receives additional space.

This property takes a number as a value, indicating the relative proportion of allocation. By default, the flex-grow value of a flexible item is 0, which means it won't receive any additional space.

For example, if a container has two flexible items, one with `flex-grow: 1` and another with `flex-grow: 2`, when there is available extra space, the second item will receive twice as much space as the first item.

```
<Box sx={{ flexGrow: 1 }}>
  <AppBar position="static">
    <Toolbar>
      <Typography variant="h6" noWrap component="div" sx={{ flexGrow: 1, display: { xs: "none", sm: "block" } }}>
        MyLOGO
      </Typography>
      <Search>
        <SearchIconWrapper>
          <SearchIcon />
        </SearchIconWrapper>
        <StyledInputBase placeholder="Search..." inputProps={{ "aria-label": "search" }} />
      </Search>
    </Toolbar>
  </AppBar>
</Box>
```

Material-UI AppBar

<Typography> : Use typography to present your design and content as clearly and efficiently as possible.

sx={{ flexGrow: 1, display: { xs: "none", sm: "block" } }} is a style attribute used in Material-UI's Grid component. This syntax is employed to control the visibility of elements at different screen sizes (breakpoints).

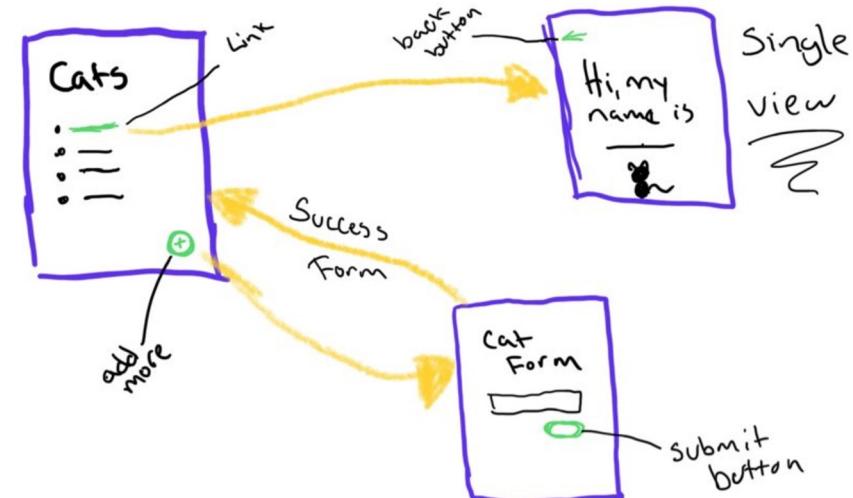
- "**none**" and "**block**" are values for the CSS display property. "none" indicates that the element is not displayed and does not occupy space, while "block" means the element is displayed and occupies layout space.

```
<Box sx={{ flexGrow: 1 }}>
  <AppBar position="static">
    <Toolbar>
      <Typography
        variant="h6"
        noWrap
        component="div"
        sx={{ flexGrow: 1, display: { xs: "none", sm: "block" } }}
      >
        MyLOGO
      </Typography>
      <Search>
        <SearchIconWrapper>
          <SearchIcon />
        </SearchIconWrapper>
        <StyledInputBase
          placeholder="Search..."
          inputProps={{ "aria-label": "search" }}
        />
      </Search>
    </Toolbar>
  </AppBar>
</Box>
```

Material-UI Page Navigation

When building modern Single-Page Applications (SPAs), navigation and transitions between pages are crucial functionalities. React **Router** is a library designed for managing routing and navigation within React applications, making it easier and more flexible to implement page transitions and navigation in SPAs.

```
npm install react-router-dom
```



Key concepts and features of React Router

- Route Management: React Router allows you to define different routes in your application, with each route corresponding to a specific page or component. By using the Route component, you can associate paths with the components to render. This enables React Router to automatically load the appropriate component based on changes in the URL.
- Nested Routes: You can create complex page structures and nested layouts within a page by using nested Route components. This allows you to build hierarchically organized and interactive user interfaces.
- Navigation Links: Use the Link component to create navigation links, enabling users to click on links to transition from one page to another. The Link component handles URL and route matching, ensuring users are directed to the correct path.

Key concepts and features of React Router

- Parameter Passing: Route parameters can be used to pass data, such as passing a product ID in the URL and loading corresponding data in a component.
- Programmatic Navigation: In addition to user-initiated navigation through links, you can programmatically navigate, for instance, automatically redirecting to another page after processing a form submission.
- Route Guards: React Router provides mechanisms for performing actions before switching pages, such as validating user identity or checking permissions.

Material-UI Page Navigation

```
import React from 'react';
import { BrowserRouter as Router, Route } from 'react-router-dom';
import Home from './Home';
import About from './About';

function App() {
  return (
    <Router>
      <Route path="/" exact component={Home} />
      <Route path="/about" component={About} />
    </Router>
  );
}

export default App;
```

Material-UI Page Navigation

The Link component is built on top of the Typography component, meaning that you can use its props.

```
import React from 'react';
import { Link } from 'react-router-dom';

function Navigation() {
  return (
    <div>
      <Link to="/">Home</Link>
      <Link to="/about">About</Link>
    </div>
  );
}

export default Navigation;
```

Learning Resources



MUI Documentation

<https://mui.com/material-ui/getting-started/>

What is Flexbox ?

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

What is the Grid system ?

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

Breakpoints in MUI

<https://mui.com/material-ui/customization/breakpoints/>

Link in MUI

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