1. Introduction To MERN stack
2. Introduction.
3. What is MERN stack?
4. MERN Stack flow of project.
5. Course outline.
6. Module Introduction.
7. Understanding the Big Picture
8. Diving Into the Frontend
9. Understanding the Backend
10. REST vs GraphQL.
11. Connecting Node & React
12. MERN – Essentials
13. React JS
14. What Is react.
15. Project Setup.
16. Understanding JSX.
17. Understanding the components.
18. Working with Multiple Components.
19. Using Props to pass Data between Components.
20. Rendering Lists of Data.
21. Handling Events
22. Efficient Child<=>Parent Communication
23. Working with "State"
24. More on State
25. Fetching User Input (Two-way Binding)
26. Designing the frontend
27. Moule introduction.
28. Starting Setup, Pages & Routes.
29. Adding a UsersList Page / Component.
30. Adding a UserItem Component.
31. Styling our App & More Components.
32. Presentational vs Stateful Components -theory.
33. Adding a Main Header.
34. Adding Navlinks.
35. Implementing a Basic Desktop & Mobile Navigation.
36. Understanding Portals.
37. Handling the Drawer State.
38. Animating the Side drawer.
39. Rendering User Places & Using Dynamic Route Segments.
40. Getting Route Params.
41. Adding Custom Buttons.
42. Adding a Modal.
43. Rendering a Map with Google Maps.
44. Continuing without a Credit Card.
45. More on the useEffect() Hook.
46. Adding a Custom Form Input Component.
47. Managing State in the Input Component.
48. Adding Input Validation
49. haring Input Values & Adding Multiple
50. Managing Form-wide State.

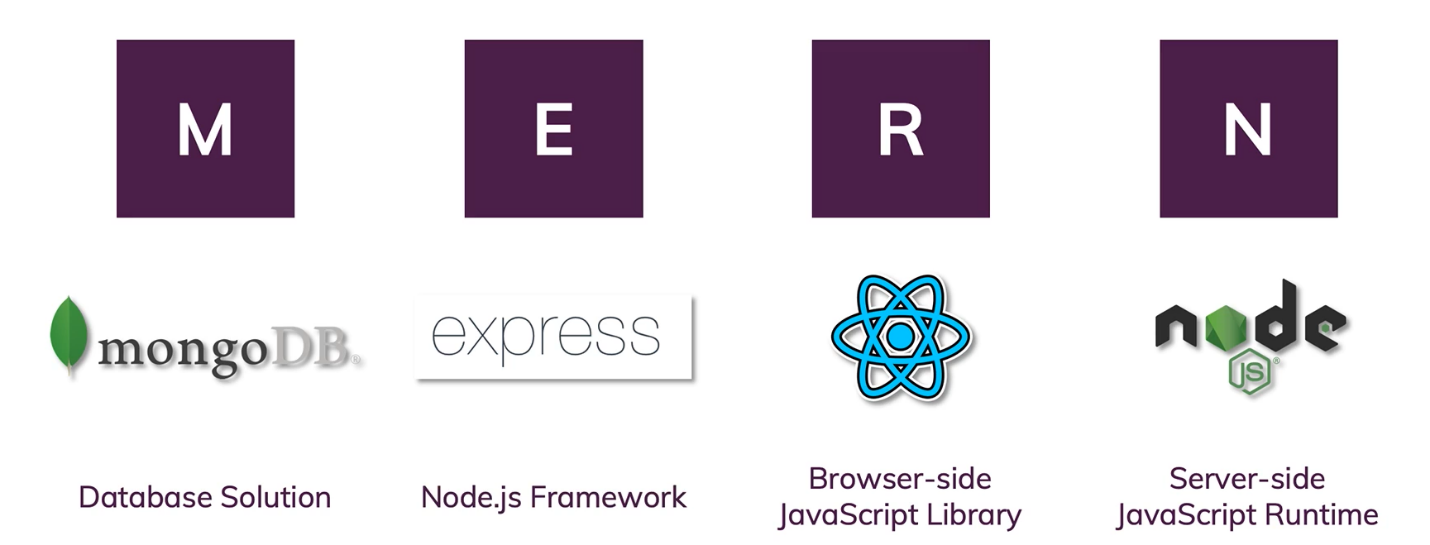
1. Node JS

* What is Node JS.
* Writing our First Node.js Code.

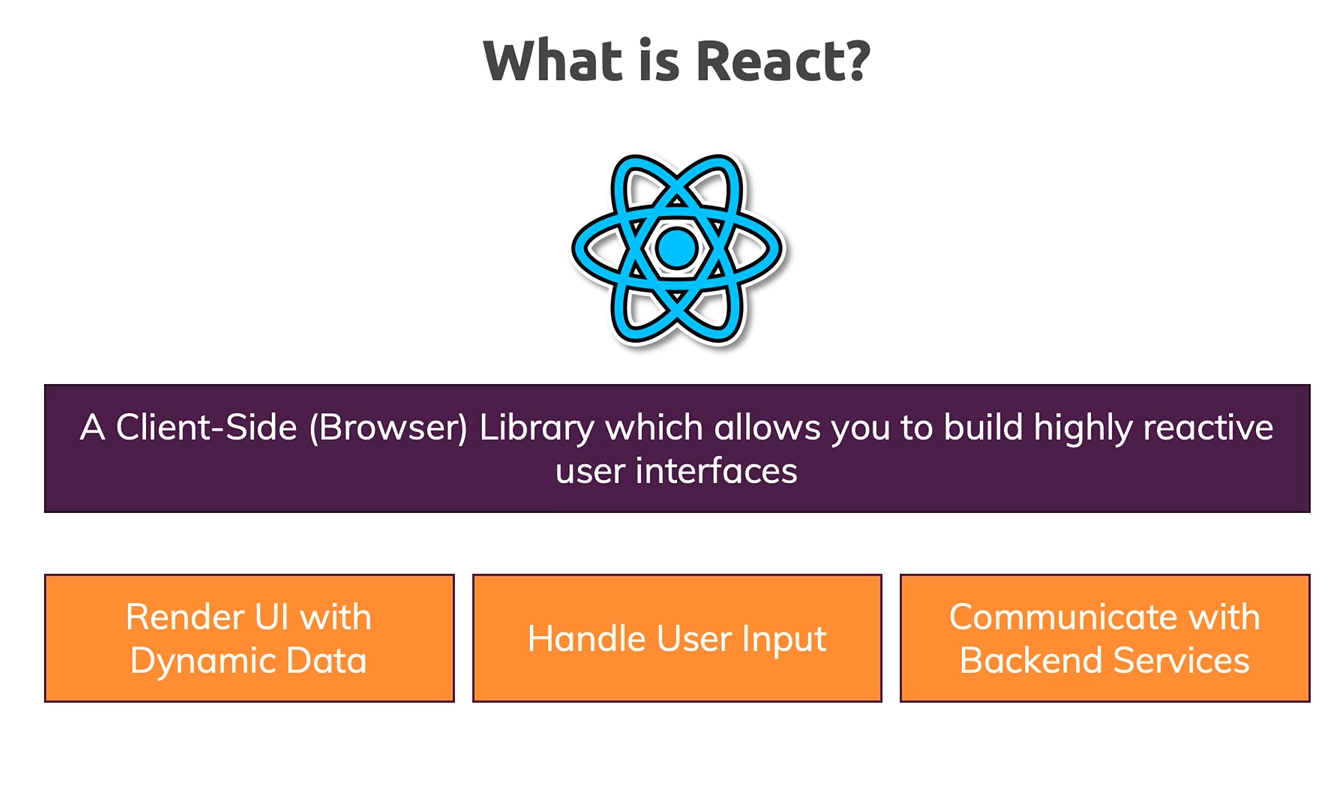
1.intrudtion

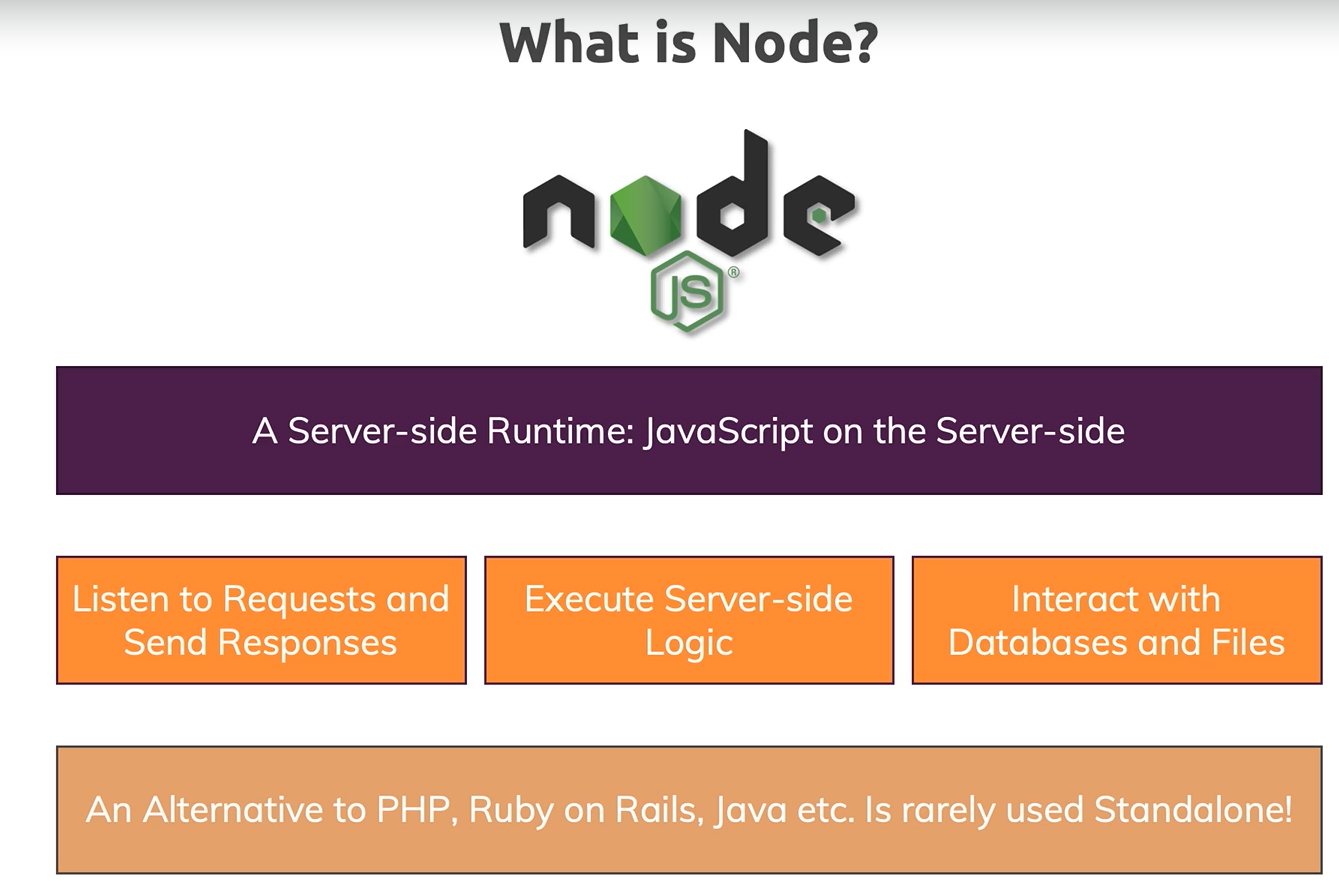
you will learn how to build amazing full stack web applications with MongoDB, Express, React.js. and you will learn how to build amazing full stack web applications with MongoDB, Express, React.js

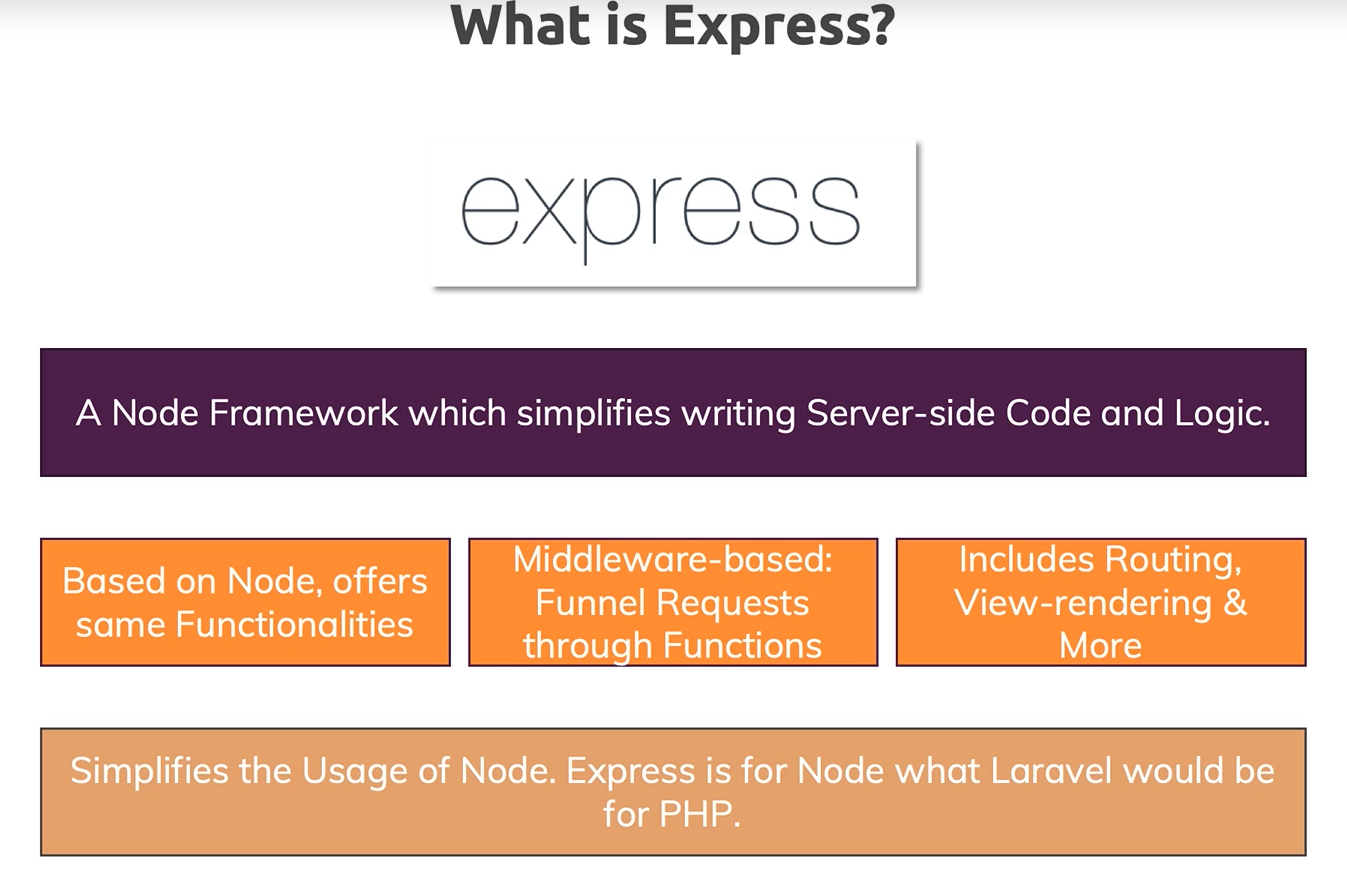
2.what is MERN?

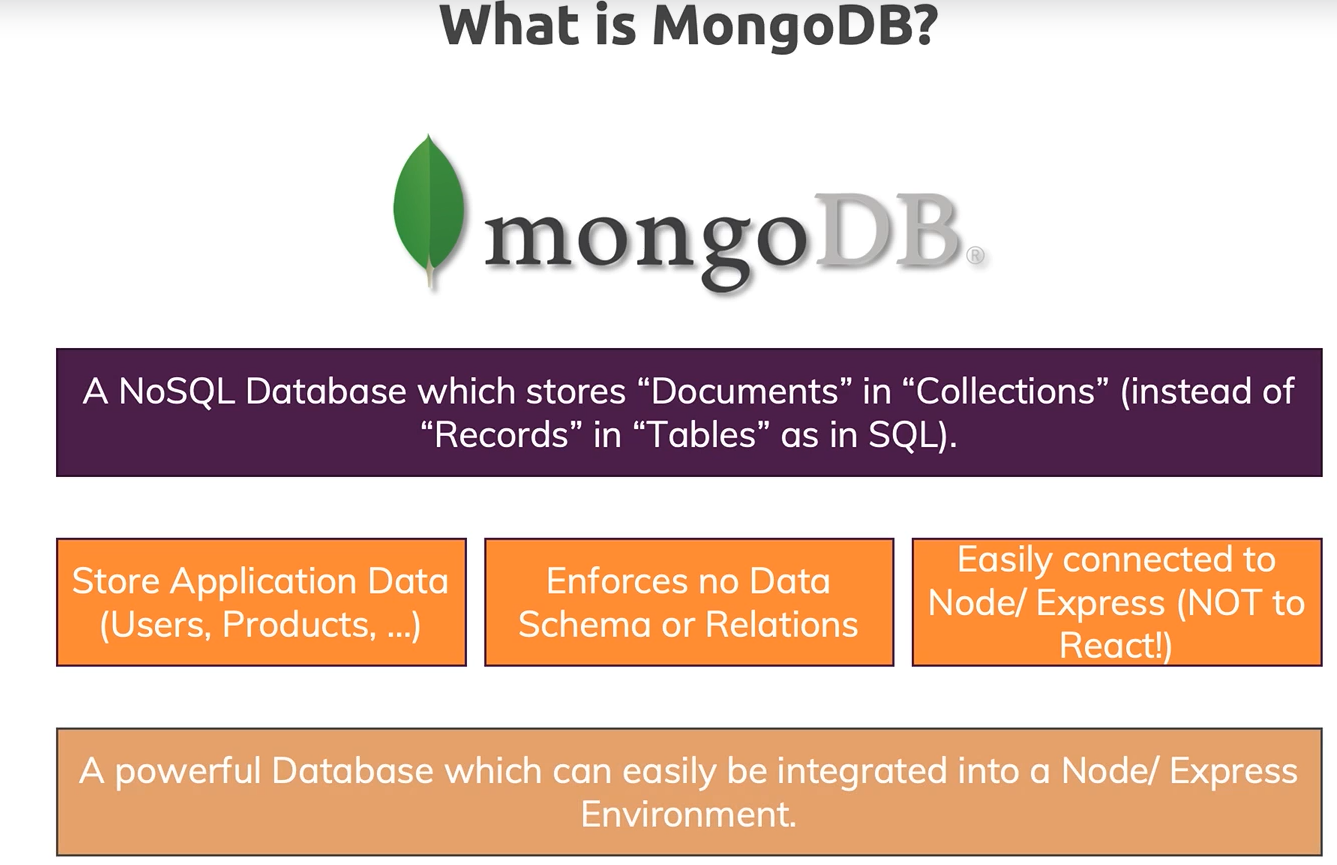


2. What is MERN stack?

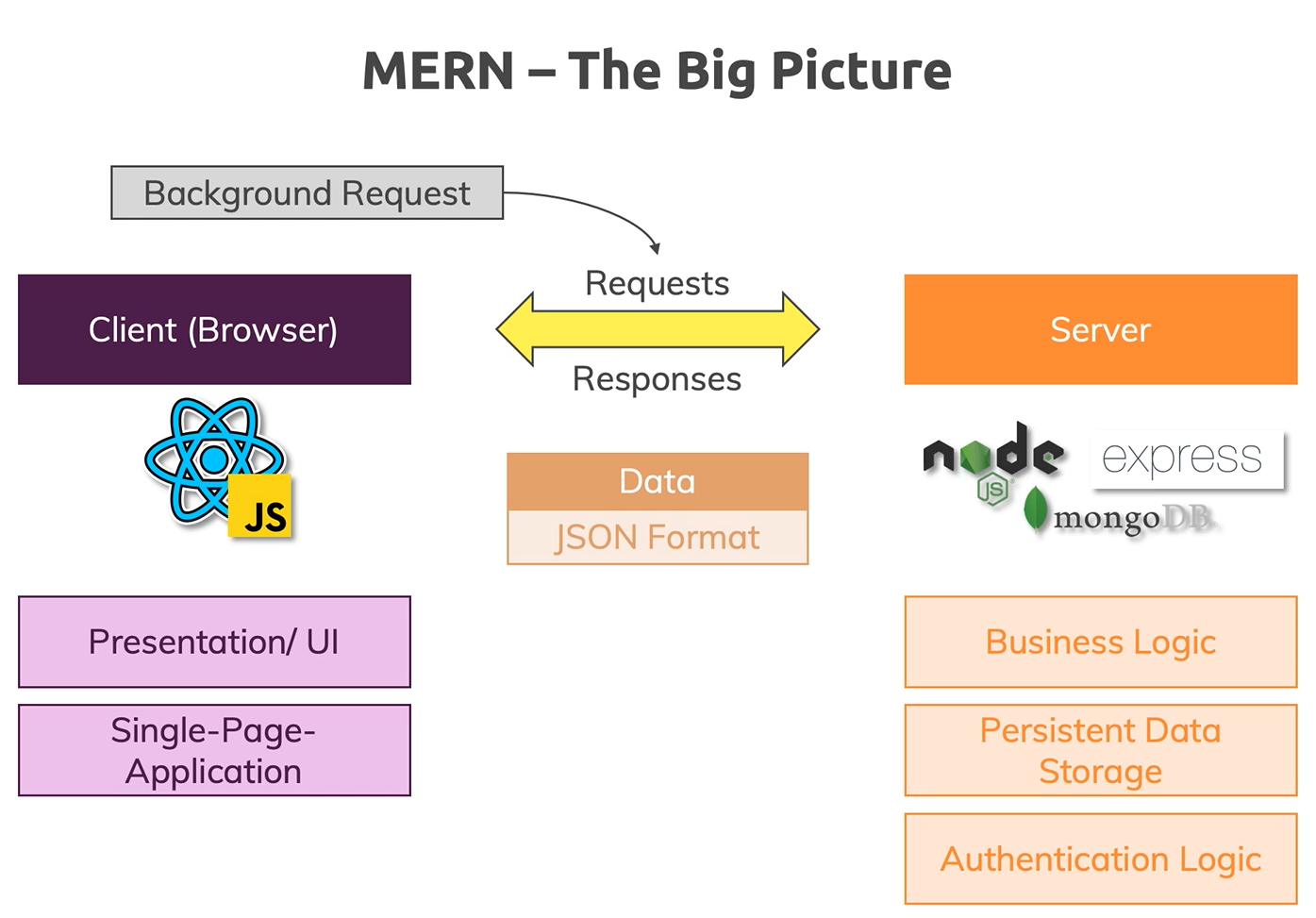




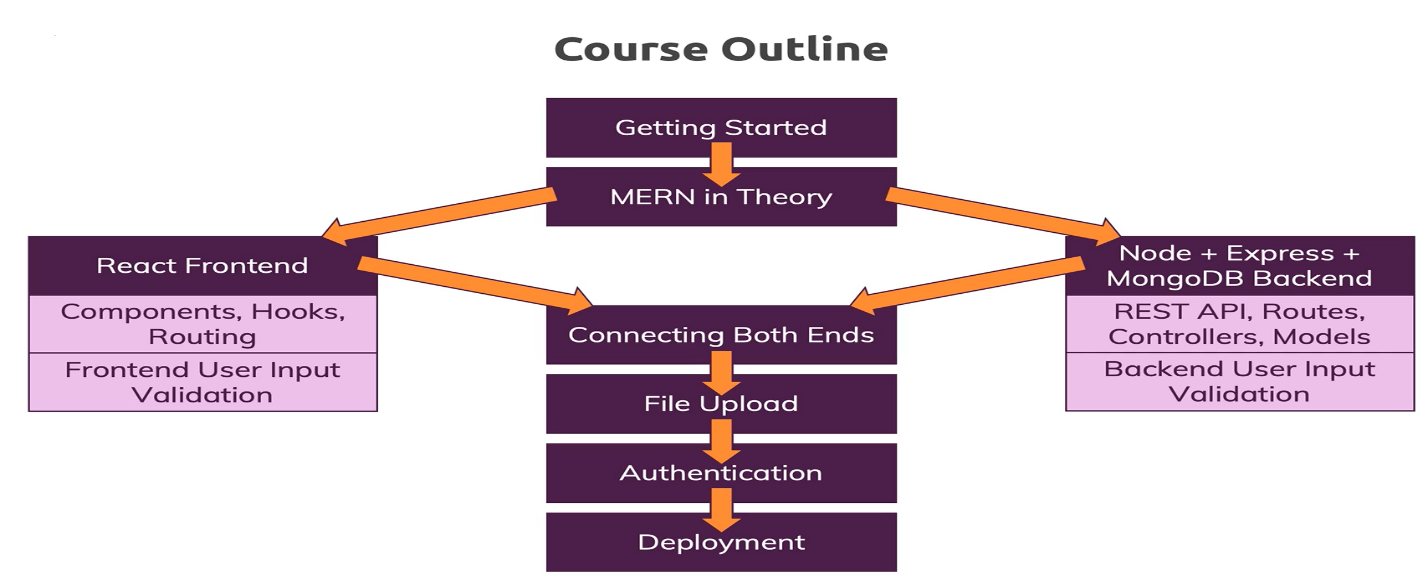




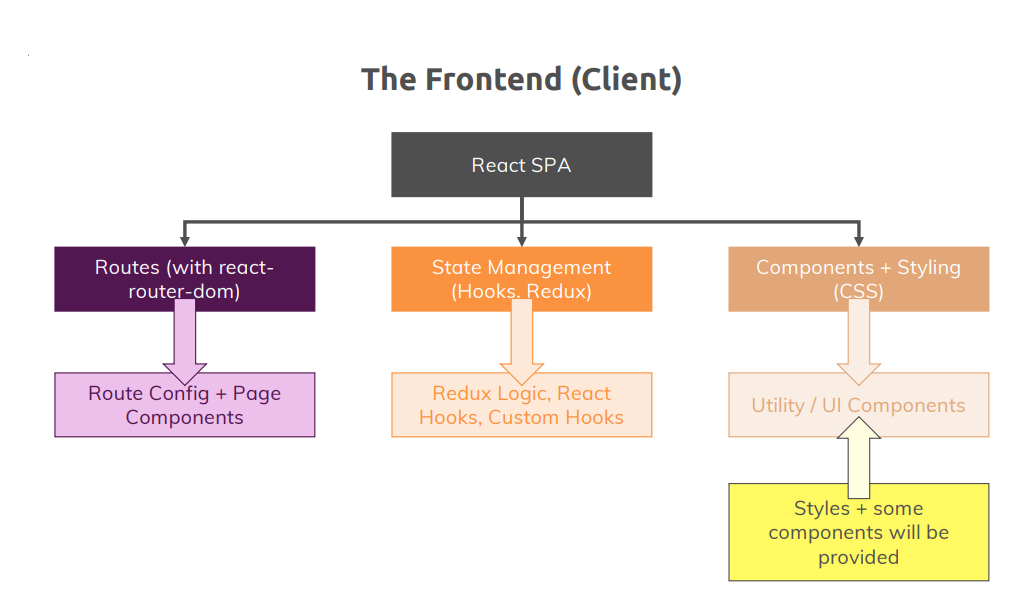
1. MERN Stack flow of project.



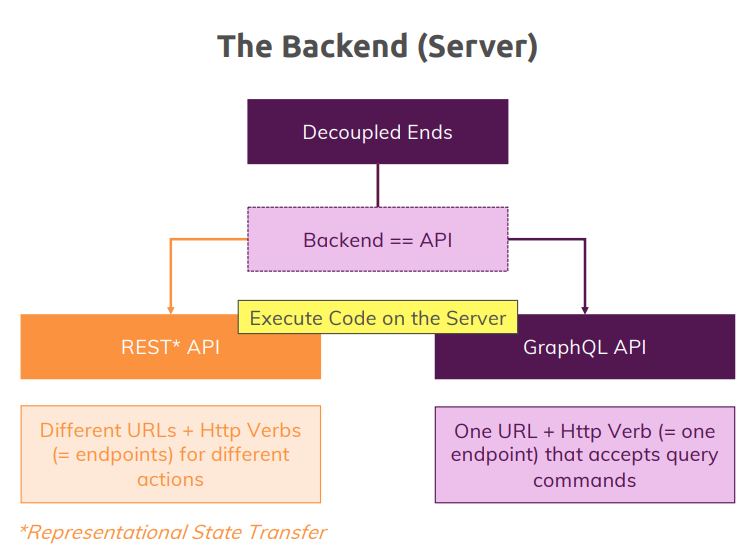
1. Course outline.



1. Module Introduction.
2. Diving Into the Frontend



1. Diving Into the Backend.



1. Diving Into the Frontend

1)What is React.

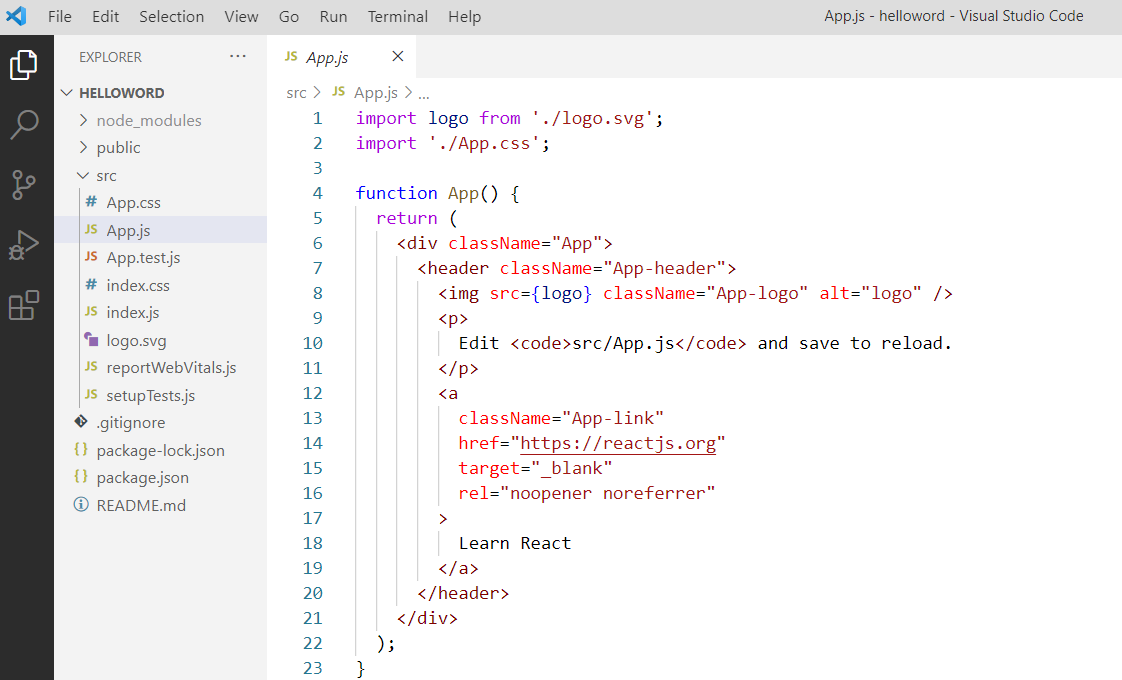
React is front JS Library.

2)Project Setup.

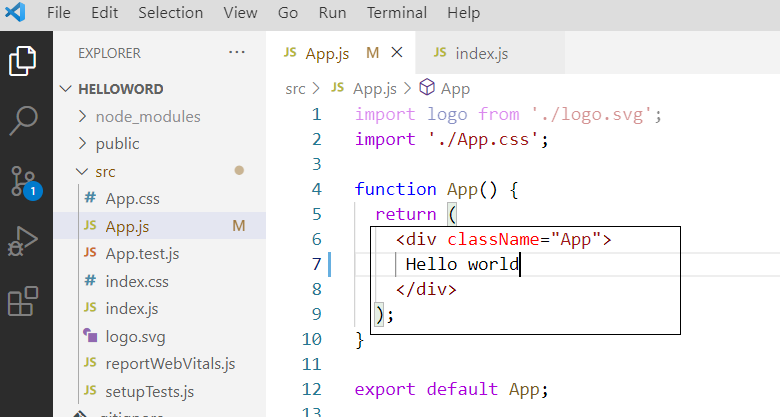
Create a file using a command ***npx-create-react-app helloworld***.

cd helloword

npm start

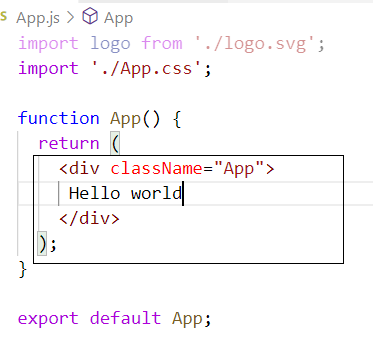


3. Understanding JSX.



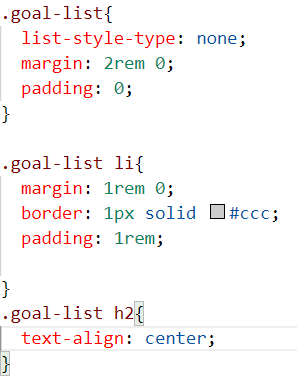
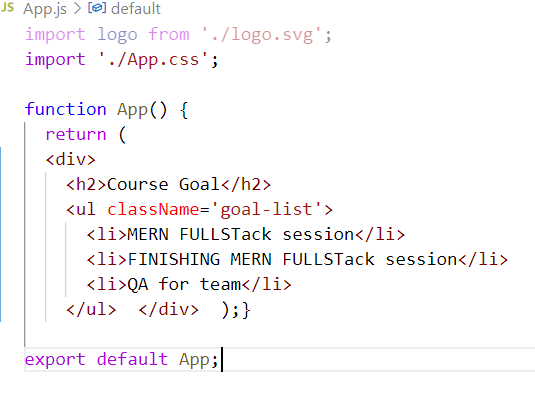
1. Understanding the components.

this function here is of course a regular JavaScript function but it's also a React component.

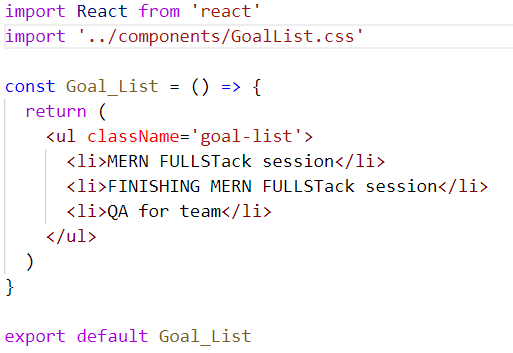
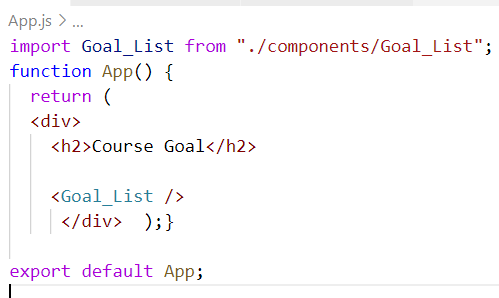


it can be a function which returns jsx or returns React create element calls or it can be a Javascript, class that has a render method.

1. Working with Multiple Components.

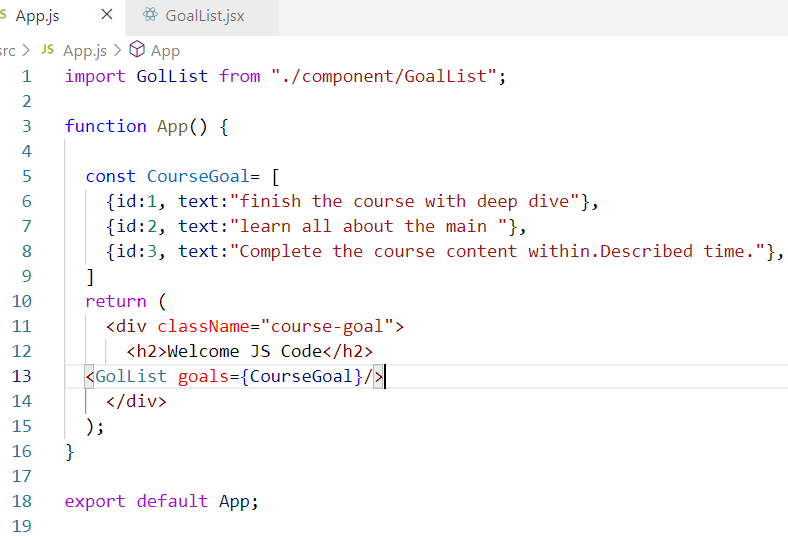


So we are going to create a one more component. So that is the goal list Goal\_List component.



1. Using Props to pass Data between Components.

So really splitting it into components has the idea of separating concerns, of keeping your files small, focused and manageable which, if you're working in bigger projects, is a huge win because it makes it



1. Rendering List Data



1. Designing the frontend
2. Starting Setup, Pages & Routes.

npx create-react-app FrontEnd

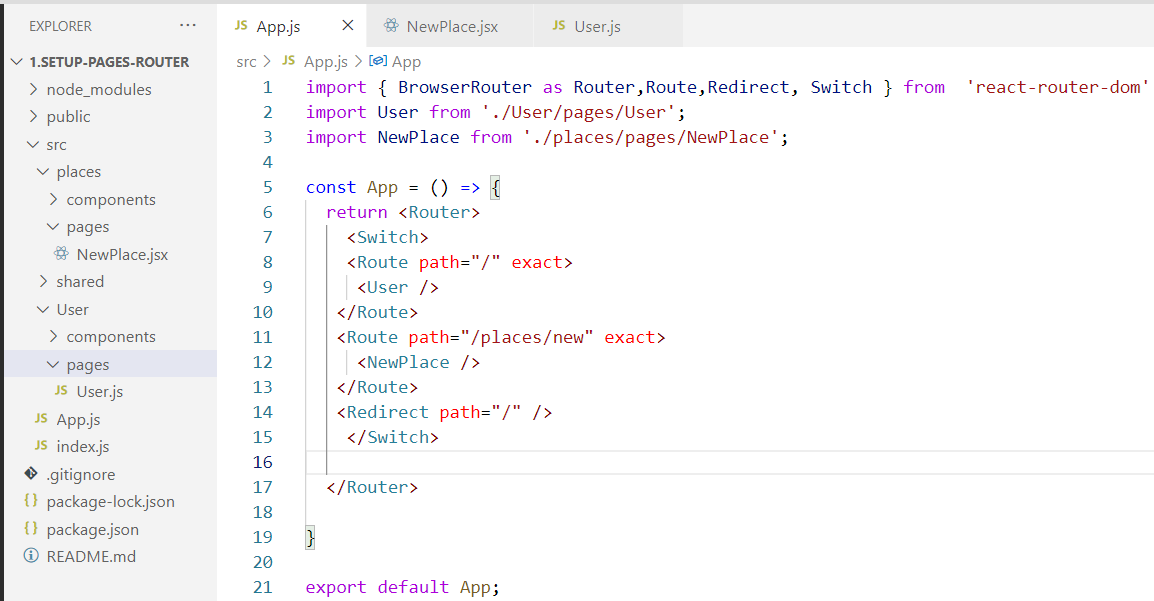
cd FrontEnd

npm start

creating a routing for React application using React router Dom 5.

***COMPONENTS:*** components is folder can be reusable, so that is we have displayed yet.

***PAGES:*** pages are the folder. For keeping the reactor pages.



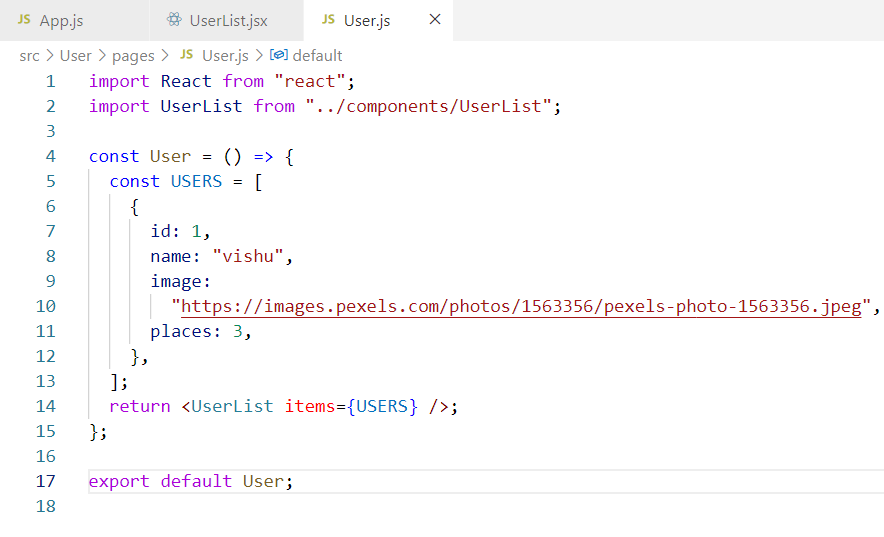
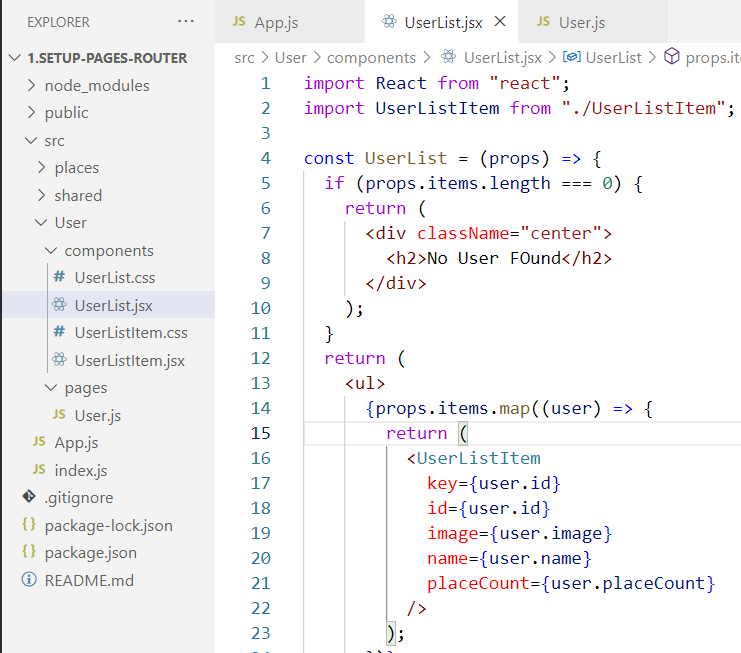
***Router***: all the pages are included within. Router. All the route’ s are kept in router.

***Route***: Route is the component which will redirect to specific page.

***Redirect:*** redirect component is used to redirect the page for a specific condition.

***Switch :*** even if a successful redirection of a Route component, it will go for the redirect component. To prevent that one. So we are using a switch. Component.

1. Adding a UsersList Page / Component



1. Styling our App & More Components.

Just we are adding some stiles here.

6 . Presentational vs Stateful Components -theory

we have presentational components and we have stateful components you could say or put in other words, we have some components which are relatively dumb, which just outputs some content,

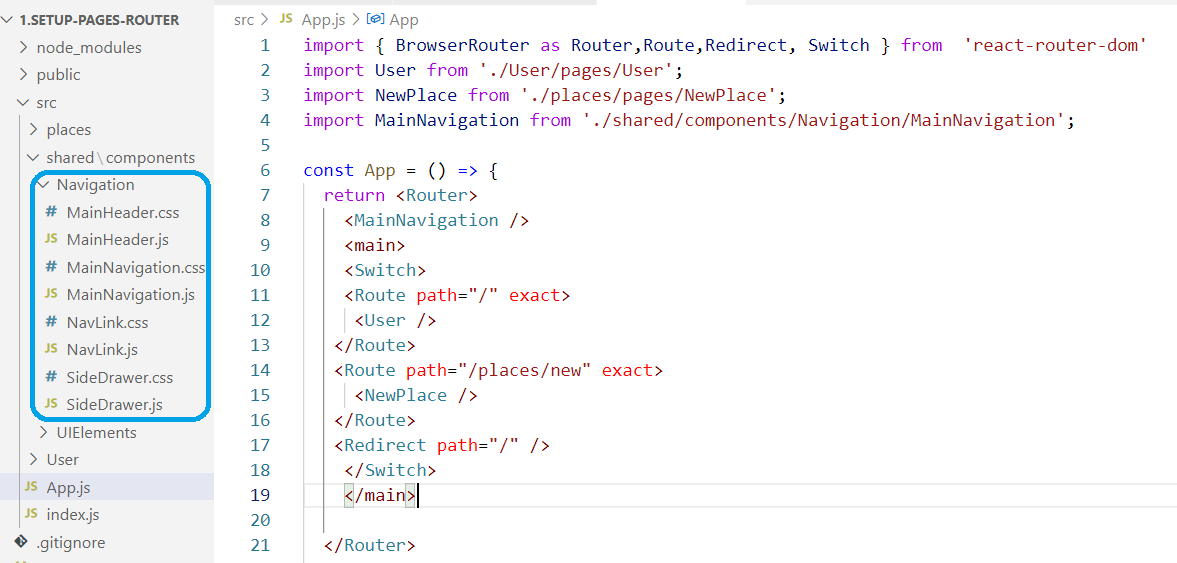
apply some styling, some structure and then we have components which also are smarter and manage some

their child components receive new data and so on. The card component which I've provided to you for example, dumb component  
ex user,card,avtar

stateful component will contain actual logic and state.

7. Adding a Main Header.

Writing a code for the *MainHeader.js* and *MainNavigation.Js*.



8. Adding Navlinks

We have added. Links to. NavLink.js component.

Added and NavLink.css file also.



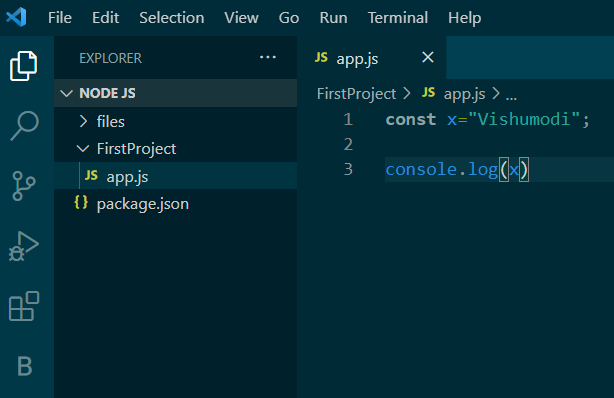
2.NODE JS

* What is Node JS

So Node.js is in the end a host environment for Javascript, a runtime for Javascript you could say.

most of us started with Javascript in the browser,it is the only programming language we can use in the browser to write some logic there, to execute some

* Writing our First Node.js Code



Writing File

