

React Assignment 3

1) What is JSX?

JSX stands for JavaScript XML.

It's a syntax extension for javascript and allows developers to write HTML like code.

Mostly used to create elements and components in react but its not limited to react so other frameworks also use it.

JSX tags have tag name, attributes and children.

React uses babel to transpile the JSX code to javascript code which is understandable by the browsers.

Even though its syntax is closer to HTML, its not exactly same, some attributes are different like

class -> className

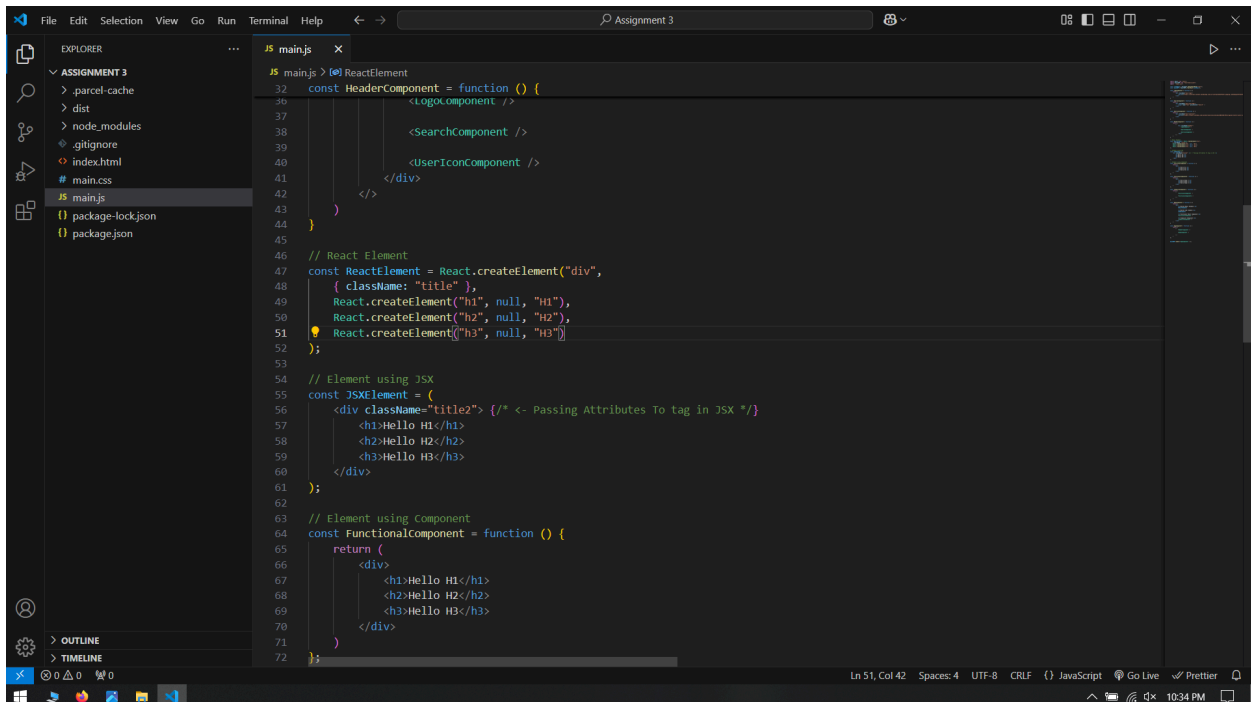
for -> htmlFor

Style -> expects a JavaScript object

on event handlers-> use camel case (onClick, onMouseOver, etc)

innerHTML -> dangerouslySetInnerHTML

Without JSX and With JSX:



```
JS main.js x
JS main.js > [9] ReactElement
32 const HeaderComponent = function () {
33   <LogoComponent />
34   <SearchComponent />
35   <UserIconComponent />
36 </div>
37 }
38
39 // React Element
40 const ReactElement = React.createElement("div",
41 { className: "title" },
42 React.createElement("h1", null, "H1"),
43 React.createElement("h2", null, "H2"),
44 React.createElement("h3", null, "H3"))
45
46 // Element using JSX
47 const JSXElement = (
48   <div className="title2"> { /* <- Passing Attributes To tag in JSX */ }
49   <h1>Hello H1</h1>
50   <h2>Hello H2</h2>
51   <h3>Hello H3</h3>
52 </div>
53 );
54
55 // Element using Component
56 const FunctionalComponent = function () {
57   return (
58     <div>
59       <h1>Hello H1</h1>
60       <h2>Hello H2</h2>
61       <h3>Hello H3</h3>
62     </div>
63   );
64 }
```

2) Superpowers of JSX?

JSX allows you to write HTML-like code in our JavaScript files.

JSX makes it easy to define and use React components.

JSX elements can have attributes, which are used to pass data to the element.

JSX elements can have children, which can be other JSX elements, text, etc

JSX supports event handling, which allows you to respond to user interactions.

JSX supports conditional rendering, which allows you to render different elements based on conditions.

JSX supports list rendering, which allows you to render arrays of data.

JSX supports type checking, which helps catch errors at runtime.

JSX supports tree shaking, which removes unused code from your application.

3) Role of type attribute in script tag ? What options can I use there?

The type attribute tells the browser what type of data/file that we are typing to write in the script or link to the html.

Meaning we can also link/use other scripting language too other than JS or modules etc

In older versions of HTML it was mandatory to use the type="text/javascript".

Options for type attribute:

type = "module" : For modern JS where we use import and export statements, meaning it will tell the browser to treat the script as a module.

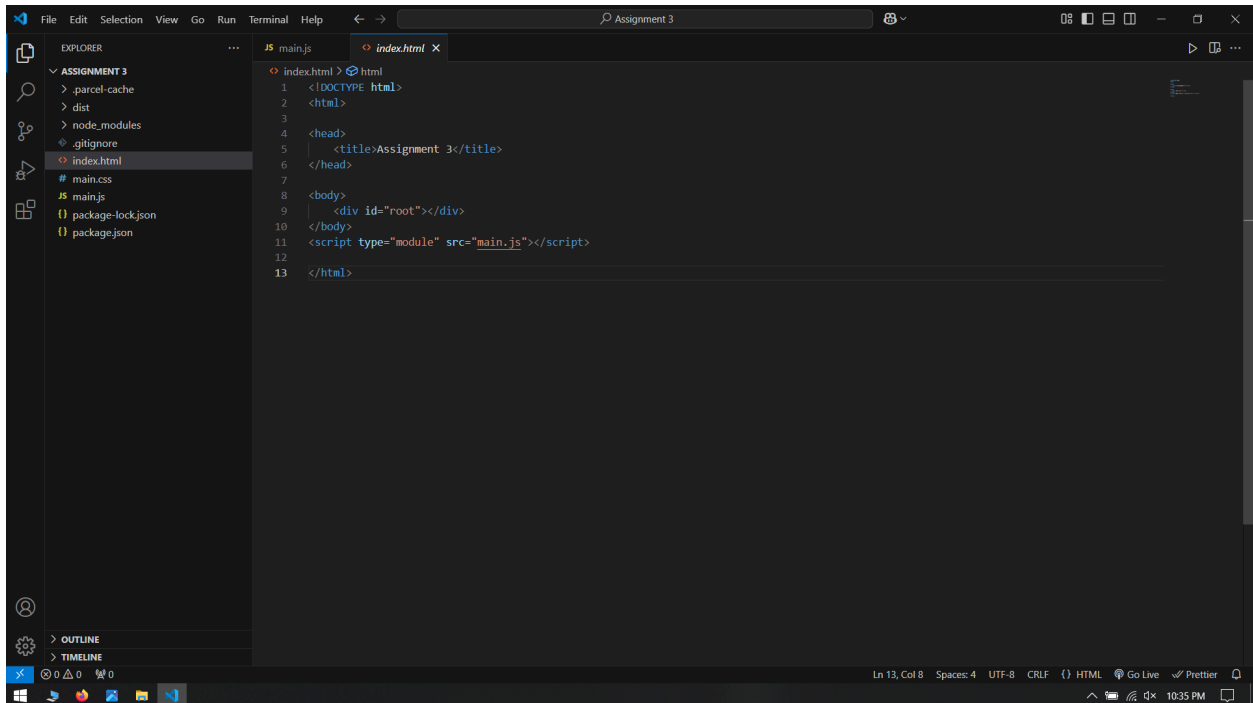
type = "text/javascript" : This attribute explicitly informs the browser that the code is javascript

type = "text/ecmascript" : This value indicates that the script is following the EcmaScript standards.

type = "text/babel" : This value indicates that the script is a babel type and requires babel to transpile it.

type = "text/typescript" : This attribute explicitly informs the browser that the code is TypeScript which is like a super set of JS which adds the type system on top of it.

Example for type = "module"



```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <title>Assignment 3</title>
6 </head>
7
8 <body>
9   <div id="root"></div>
10 </body>
11 <script type="module" src="main.js"></script>
12
13 </html>
```

4) {TitleComponent} vs {<TitleComponent/>} vs {<TitleComponent></TitleComponent>} in JSX ?

{}

 is used in JSX to run/include JavaScript code into the JSX code.

{TitleComponent}

 is used when we want to render the value inside the variable or function called TitleComponent.

<TitleComponent />

 is used when we want to render a React component without any children.

<TitleComponent></TitleComponent>

 is used when we want to render React components with children like the component can have other components, elements, or even text nested inside it.