* ReactJs makes use of JSX instead of HTML

**Components : Functions vs Classes**

React Hooks :

**Public :** folder has single page , index.html file which represent single page application.

**Src**  : react folder holds all components & state used in reactjs.

It has **index.js** which is the entry point for react.

Inside the src folder, it contains **index.js** which contains render() file which holds, ‘root’ & we are inserting App into root div.

**Note :** app is the root component.

App.js contains root component, footer header will its part or, will be inside it.

**Think every thing in terms of the component in react js.**

In reactJs we use JSX as markup than HTML.

In JSX there is some difference than HTML :

1. In HTML we write element as

<element class=”class name” >

But in JSX :

<element className = “class name”>

1. In HTML for label we write **for** as

<label for=”name of element”>

But in JSX :

<label htmlFor = “name of element”>

**App.js**

Function App(){

}

It is function, which acts as component, class can

Also act as component.

**Note :** function App() can return only one element, though inside it another element may exist.

In function app() return div / content , javascript code can be written inside the curly braces.

{ js code}

**Note** : think your app as bunch of containers.

Type **rafce** to create boiler plate for arrow function based components.

**To create class based components :**

import React from 'react'

class App extends React.Component{

render(){

return <h1>Hello from Class</h1>

}

}

export default App;

**Note : props ,** parameters passed to components.

**propTypes :** it specifies what will be the type of props, whether it will

String, integer, boolean etc . It makes the code more robust.

We can use TypeScript too, to check if type of data is correct or , not.

**Note :** while writing boolean or integer in the component make sure to write

Them within {}.

Components in ReactJS is added like XML e.g. :

< **Component />**

**Styling in React js (CSS):**

1. Use stylesheet
2. Style component
3. Direct css in javascript

**Inline styling of components :**

<h1 style={{color: 'red' , backgroundColor:'black'}}>Task Tracker {title}</h1>

**Note :** we use double bracket {{}}

We can also create an object and make **style = {headObject}**

To add onClick function on button you can use :

const Button = ({ color, text }) => {

**const onClick = () => {**

**console.log("Click");**

**};**

return (

<button

onClick={onClick}

style={{ backgroundColor: color }}

className="btn"

>

{text}

</button>

);

};

**Note :** Above code, make button less flexible by fixing the onclick function.

Other way to make onClick button more user defined / reusable is by providing onClick as parameter like a **props.**

**Task components :**

Why are we using tasks array by implementing useState ?

Since useState if defined within the Tasks.js it will have its scope

Within it only. We may need in other component, then, it is better to

Create at app level & pass the task as **props** to the **Tasks component.**

**Doubts in task components :**

What is key in heading tag, <h1> ?

Why it should be unique?

**Home work :**

Fect API and making HTTP request. // Travesty Media has crash course on these.

What is render() ?

Context API, state manager Redux .

When to use {} ? What is its significance ?

What is useState ?

**Resume from : 49:10**