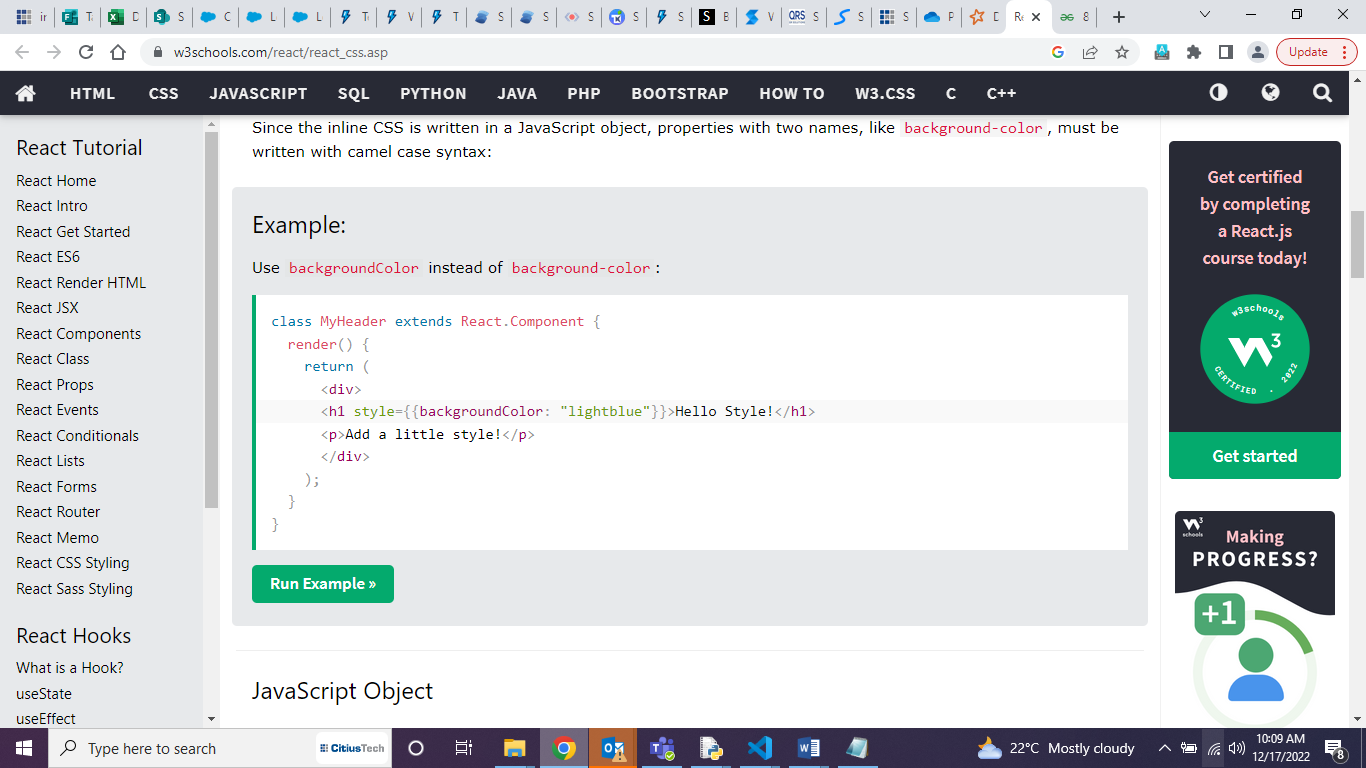
1. 1. In react how do you manage style(css) in component?
   1. Inline CSS

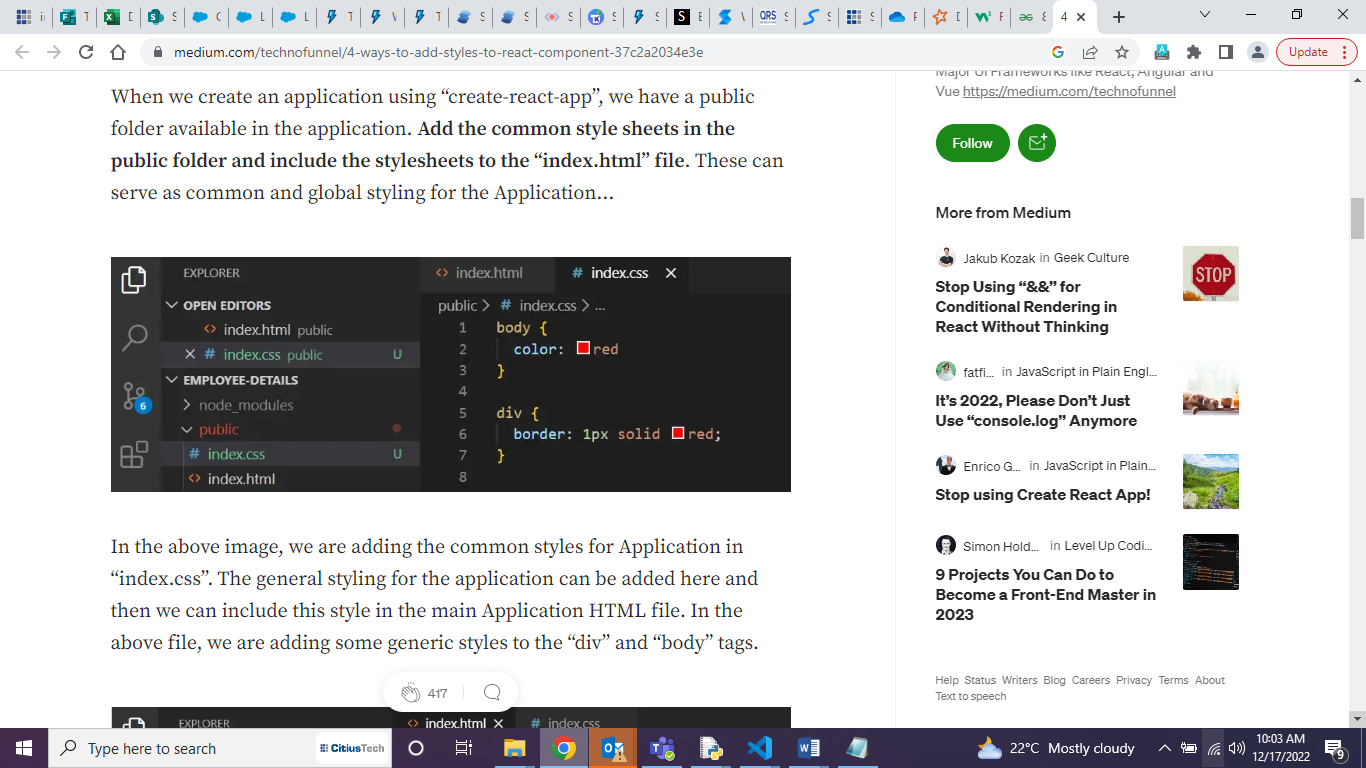


* 1. Normal CSS
  2. CSS in JS
  3. Styled Components
  4. CSS module
  5. Sass & SCSS

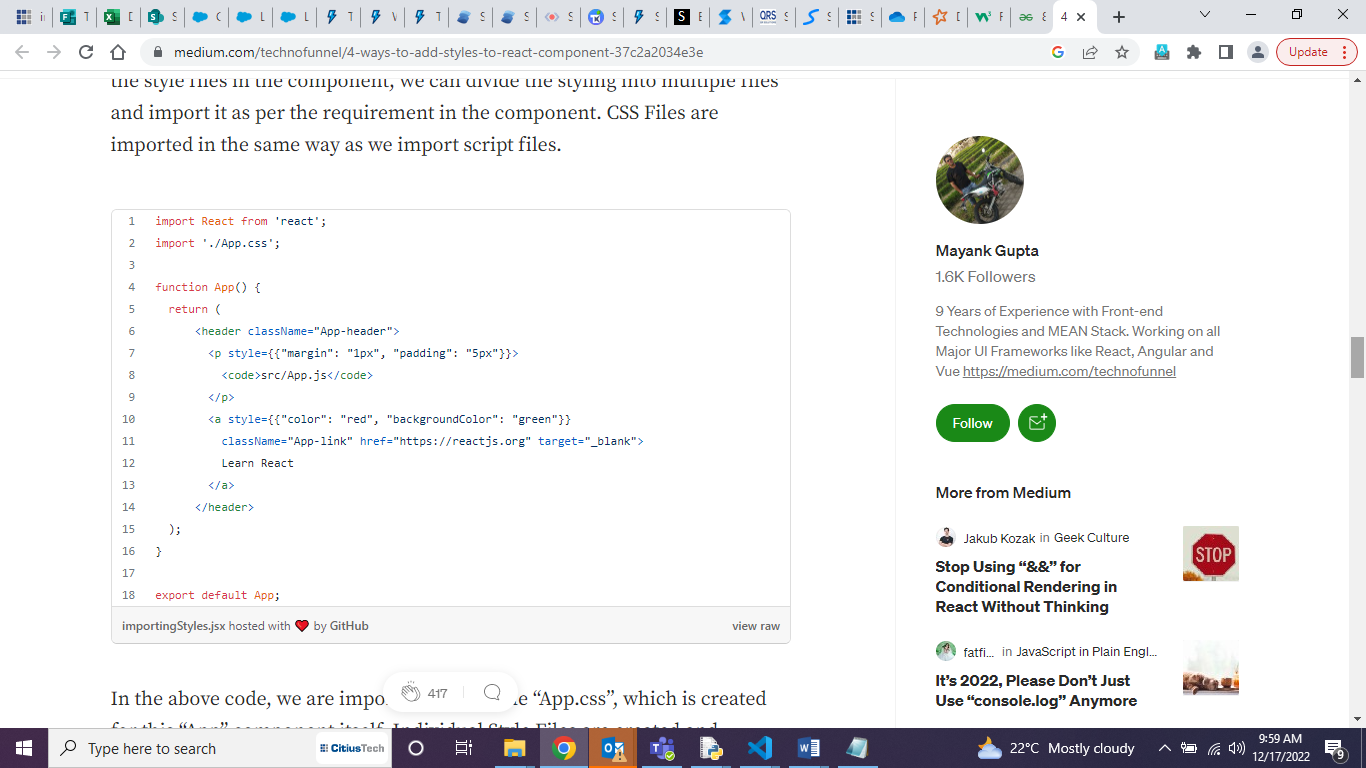
1. 2. How do you manage component style, for example you create a component for button/grid.css?

**React Components can add styling** in the following ways:

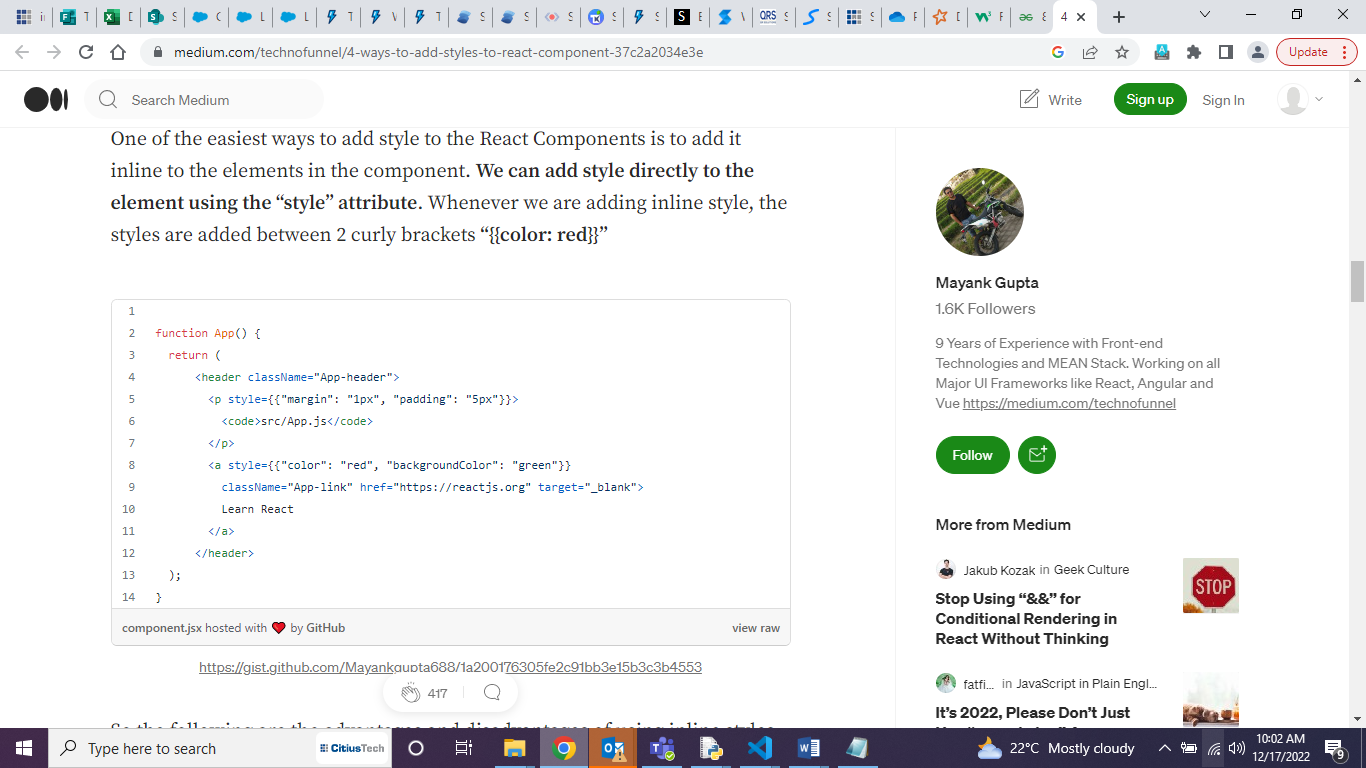
1. **Add the Global Styles to “index.html” File**



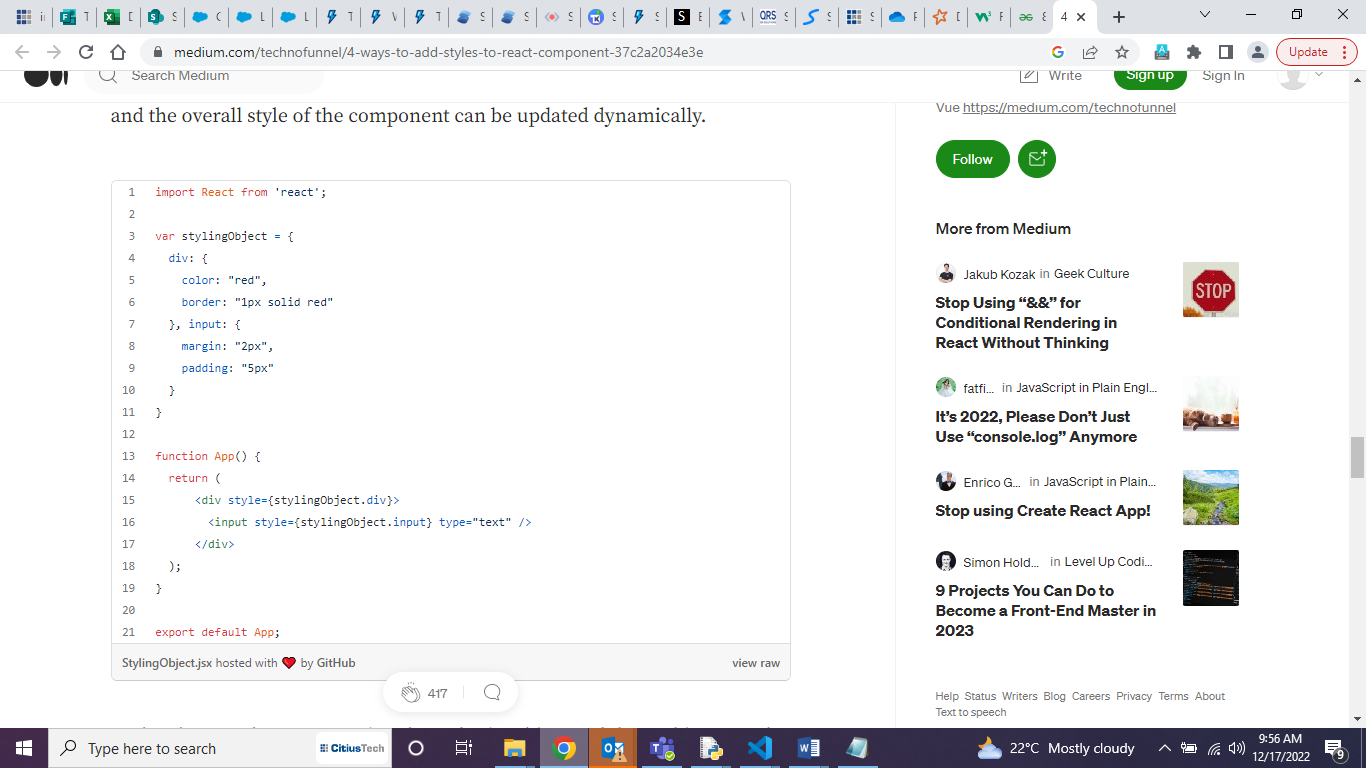
1. **Create a Style for Each Individual Component**



1. **Adding Inline Style to React Component Elements**



1. **Creating Style Objects and Bind to Components**



1. 3. How do you handle API calls in your react application? explain Ajax, Axios and interceptor?

FETCH

class MyComponent extends React.Component {

constructor(props) {

super(props);

this.state = {

error: null,

isLoaded: false,

items: []

};

}

componentDidMount() {

fetch("https://api.example.com/items")

.then(res => res.json())

.then(

(result) => {

this.setState({

isLoaded: true,

items: result.items

});

},

// Note: it's important to handle errors here

// instead of a catch() block so that we don't swallow

// exceptions from actual bugs in components.

(error) => {

this.setState({

isLoaded: true,

error

});

}

)

}

render() {

const { error, isLoaded, items } = this.state;

if (error) {

return <div>Error: {error.message}</div>;

} else if (!isLoaded) {

return <div>Loading...</div>;

} else {

return (

<ul>

{items.map(item => (

<li key={item.id}>

{item.name} {item.price}

</li>

))}

</ul>

);

}

}

}

Here is the equivalent with [Hooks](https://reactjs.org/docs/hooks-intro.html):

function MyComponent() {

const [error, setError] = useState(null);

const [isLoaded, setIsLoaded] = useState(false);

const [items, setItems] = useState([]);

// Note: the empty deps array [] means

// this useEffect will run once

// similar to componentDidMount()

useEffect(() => {

fetch("https://api.example.com/items")

.then(res => res.json())

.then(

(result) => {

setIsLoaded(true);

setItems(result);

},

// Note: it's important to handle errors here

// instead of a catch() block so that we don't swallow

// exceptions from actual bugs in components.

(error) => {

setIsLoaded(true);

setError(error);

}

)

}, [])

if (error) {

return <div>Error: {error.message}</div>;

} else if (!isLoaded) {

return <div>Loading...</div>;

} else {

return (

<ul>

{items.map(item => (

<li key={item.id}>

{item.name} {item.price}

</li>

))}

</ul>

);

}

}

1 async getUsers() {

2 // With additional headers

3 const response = await fetch("https://jsonplaceholder.typicode.com/users", {

4 method: "GET", // \*Type of request GET, POST, PUT, DELETE

5 mode: "cors", // Type of mode of the request

6 cache: "no-cache", // options like default, no-cache, reload, force-cache

7 credentials: "same-origin", // options like include, \*same-origin, omit

8 headers: {

9 "Content-Type": "application/json" // request content type

10 },

11 redirect: "follow", // manual, \*follow, error

12 referrerPolicy: "no-referrer", // no-referrer, \*client

13 // body: JSON.stringify(data) // Attach body with the request

14 });

15 this.setState({ users: await response.json() });

16 }

Now let’s try the POST request to post the data to the server using the POST as the request type with Fetch.

1async getUsers() {

2 let body = {

3 userId: 1111,

4 title: "This is POST request with body",

5 completed: true

6 };

7 fetch("https://jsonplaceholder.typicode.com/todos", {

8 method: "POST",

9 body: JSON.stringify(body)

10 })

11 .then(response => {

12 let json = response.json();

13 console.log(json);

14 if (!response.ok) {

15 throw new Error("Network response was not ok");

16 }

17 return response.blob();

18 })

19 .catch(error => {

20 console.error(

21 "There has been a problem with your fetch operation:",

22 error

23 );

24 });

25 }

AXIOS

async getTodos() {

2// With all properties

3 axios

4 .get("https://jsonplaceholder.typicode.com/todos?\_page=1&\_limit=10")

5 .then(response => {

6 console.log(response.data);

7 console.log(response.status);

8 console.log(response.statusText);

9 console.log(response.headers);

10 console.log(response.config);

11 })

12 .catch(function(error) {

13 console.log(error);

14 });

15}

Let’s post the data using the Axios POST request with this example.

import React, { Component } from "react";

2import axios from "axios";

3

4class UsingAxios extends Component {

5 constructor() {

6 super();

7 this.state = {

8 name: "React"

9 };

10 this.getTodos = this.getTodos.bind(this);

11 }

12

13 componentDidMount() {

14 this.getTodos();

15 }

16

17 async getTodos() {

18 // With all properties

19 let body = {

20 userId: 1111,

21 title: "This is POST request with body",

22 completed: true

23 };

24 axios

25 .post("https://jsonplaceholder.typicode.com/todos", body)

26 .then(function(response) {

27 console.log(response.data);

28 })

29 .catch(function(error) {

30 console.log(error);

31 });

32 }

33

34 render() {

35 const { todos } = this.state;

36 return (

37 <div>

38 <h3>Using Axios in React for API call</h3>

39 <hr />

40 </div>

41 );

42 }

43}

44

45export default UsingAxios;

Axios – delete

handleSubmit = event => {

  event.preventDefault();

  axios.delete(`https://jsonplaceholder.typicode.com/users/${this.state.userName}`)

  .then(res => {

  console.log(res);

  console.log(res.data);

  })

 }

MAKING MULTIPLE REQUESTS WITH AXIOS

axios.all([

  axios.get('https://api.github.com/users/defunkt),

  axios.get('https://api.github.com/users/evanphx)

 ])

 .then(response => {

  console.log('Date created: ', response[0].data.created\_at);

  console.log('Date created: ', response[1].data.created\_at);

 });

1. 4. How to handle API response as consider a example like login page as example user enter credentials and click on the submit button

auth-tutorial/server.js

const express = require('express');

const cors = require('cors')

const app = express();

app.use(cors());

app.use('/login', (req, res) => {

res.send({

token: 'test123'

});

});

app.listen(8080, () => console.log('API is running on http://localhost:8080/login'));

auth-tutorial/src/components/Login/Login.js

import React, { useState } from 'react';

import PropTypes from 'prop-types';

import './Login.css';

async function loginUser(credentials) {

return fetch('http://localhost:8080/login', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(credentials)

})

.then(data => data.json())

}

export default function Login({ setToken }) {

const [username, setUserName] = useState();

const [password, setPassword] = useState();

const handleSubmit = async e => {

e.preventDefault();

const token = await loginUser({

username,

password

});

setToken(token);

}

return(

<div className="login-wrapper">

<h1>Please Log In</h1>

<form onSubmit={handleSubmit}>

<label>

<p>Username</p>

<input type="text" onChange={e => setUserName(e.target.value)} />

</label>

<label>

<p>Password</p>

<input type="password" onChange={e => setPassword(e.target.value)} />

</label>

<div>

<button type="submit">Submit</button>

</div>

</form>

</div>

)

}

Login.propTypes = {

setToken: PropTypes.func.isRequired

};

auth-tutorial/src/components/App/App.js

import React from 'react';

import { BrowserRouter, Route, Switch } from 'react-router-dom';

import './App.css';

import Dashboard from '../Dashboard/Dashboard';

import Login from '../Login/Login';

import Preferences from '../Preferences/Preferences';

import useToken from './useToken';

function App() {

const { token, setToken } = useToken();

if(!token) {

return <Login setToken={setToken} />

}

return (

<div className="wrapper">

<h1>Application</h1>

<BrowserRouter>

<Switch>

<Route path="/dashboard">

<Dashboard />

</Route>

<Route path="/preferences">

<Preferences />

</Route>

</Switch>

</BrowserRouter>

</div>

);

}

export default App;

auth-tutorial/src/components/App/useToken.js

import { useState } from 'react';

export default function useToken() {

const getToken = () => {

const tokenString = localStorage.getItem('token');

const userToken = JSON.parse(tokenString);

return userToken?.token

};

const [token, setToken] = useState(getToken());

const saveToken = userToken => {

localStorage.setItem('token', JSON.stringify(userToken));

setToken(userToken.token);

};

return {

setToken: saveToken,

token

}

}

1. call API and get responses as success or error? In where to handle error in response?

1async getUsers() {

2 let body = {

3 userId: 1111,

4 title: "This is POST request with body",

5 completed: true

6 };

7 fetch("https://jsonplaceholder.typicode.com/todos", {

8 method: "POST",

9 body: JSON.stringify(body)

10 })

11 .then(response => {

12 let json = response.json();

13 console.log(json);

14 if (!response.ok) {

15 throw new Error("Network response was not ok");

16 }

17 return response.blob();

18 })

19 .catch(error => {

20 console.error(

21 "There has been a problem with your fetch operation:",

22 error

23 );

24 });

25 }

1. 5. Handle 405, 404 and 403 error response

405 Method Not Allowed

403 - Access denied

**Use a wildcard placeholder to handle 404 page not found in React router, e.g. <Route path="\*" element={<PageNotFound />} />. A route that has an asterisk path \* serves as a catch all route. It only matches when no other routes do.**

import React from 'react';

import {Route, Link, Routes} from 'react-router-dom';

export default function App() {

return (

<div>

<div>

<nav>

<ul>

<li>

<Link to="/">Home</Link>

</li>

<li>

<Link to="/about">About</Link>

</li>

<li>

{*/\* 👇️ link to catch all route \*/*}

<Link to="/does-not-exist">Catch all route</Link>

</li>

</ul>

</nav>

<Routes>

<Route path="/about" element={<About />} />

<Route path="/" element={<Home />} />

{*/\* 👇️ only match this when no other routes match \*/*}

<Route path="\*" element={<PageNotFound />} />

</Routes>

</div>

</div>

);

}

function Home() {

return <h2>Home</h2>;

}

function About() {

return <h2>About</h2>;

}

function PageNotFound() {

return (

<div>

<h2>404 Page not found</h2>

</div>

);

}

1. 6. How to handle multiple API call, for example login API will have successfully submitted after getting more consecutive 3 APIs success response. What is your approach to implementing that scenarios.

axios.all([

  axios.get('https://api.github.com/users/defunkt),

  axios.get('https://api.github.com/users/evanphx)

 ])

 .then(response => {

  console.log('Date created: ', response[0].data.created\_at);

  console.log('Date created: ', response[1].data.created\_at);

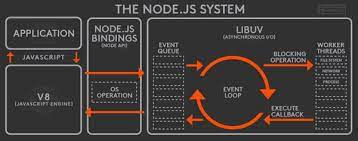
 });

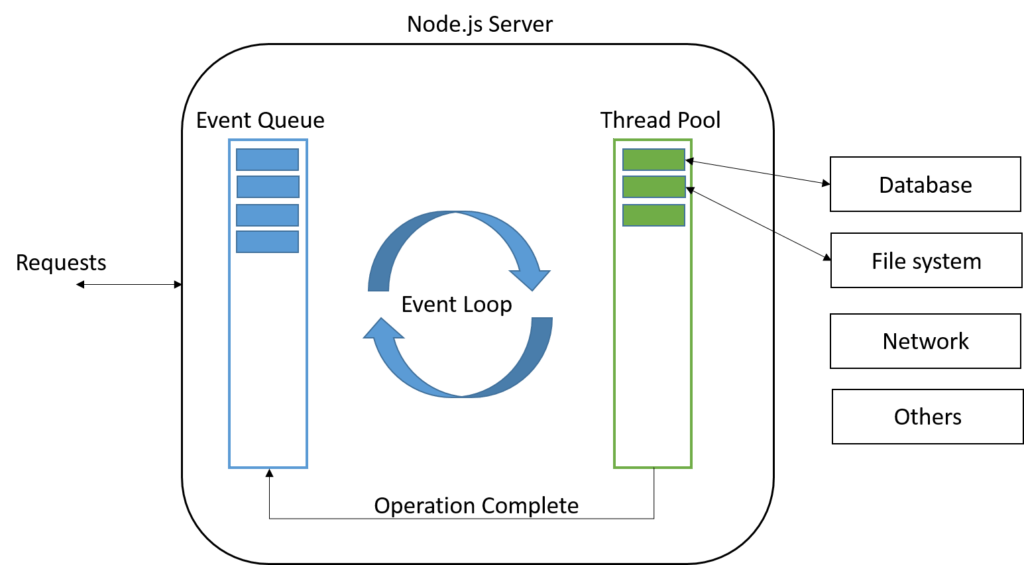
1. (async - await, fetch - .then, .then, .then... and promise.all)
2. 7. In a fetch - .then, .then, .then... API scenarios suppose 1st API call success and get the response then call 2nd API it will failed. After that what will happened about the 3rd API call it will work or not? and which error we will get?
3. 8. In the same scenario what happens in promise.all(), it will reject or calls the APIs?
4. 9. Life cycle of React - Virtual DOM, Performance level and state management
5. 10. In nested component (parent - child component) if changing on the parent component then child component will also re-render or not? If yes How can we prevent child re-rendering and why it's important?
6. (read - higher order component, pure-component, should-component level life cycle, Memo/Strict Mode)
7. 11. How to store data in deep child level scenario for high-level? Explain Redux and use cases?
8. 12. What is Context-API? Difference between Redux and context API?
9. 13. What is middle-ware? how it's work in React-Auth scenario?
10. 14. Do you have Idea about deployment of react-app on google cloud or AWS?
11. 1. What are the advantages of using React?
12. 2. What is JSX?
13. 3. What are the differences between functional and class components?
14. 4. What is the virtual DOM? How does react use the virtual DOM to render the UI?
15. 5. What are the differences between controlled and uncontrolled components?
16. 6. What are the different lifecycle methods in React?
17. 7. Explain Strict Mode in React.
18. 8. How to prevent re-renders in React?
19. 9. Explain React state and props.
20. 10. Explain React Hooks.
21. 11. What are the different ways to style a React component?
22. 12. Name a few techniques to optimize React app performance.
23. 13. What are keys in React? How to pass data between react components?
24. 14. What are Higher Order Components?
25. 15. What is prop drilling in React?
26. 16. What are error boundaries?
27. Can we replace Index.html in react
28. Where we can change or replace the homepage component(landing page)
29. What is Higher Order Component
30. What is the difference between mapStateToProps() and mapDispatchToProps()?
31. Where you will write the api call in the component
32. What is reducer and roles in redux
33. Can we use two reducer?
34. Why do we use redux?
35. how we can use third party library and in which files in will make changes
36. Can we replace Index.html in react?
37. What is the use of connect() Redux?
38. Explain different ways to write mapDispatchToProps()
39. What is the difference between React context and React Redux?
40. Difference btw library and framework
41. What is redux and how we can implement it
42. What is redux saga

Redux Saga is a middleware library used to allow a Redux store to interact with resources outside of itself asynchronously. This includes making HTTP requests to external services, accessing browser storage, and executing I/O operations. These operations are also known as side effects. Redux Saga helps to organize these side effects in a way that is easier to manage.

1. How we can use middleware in our application in node and react js
2. What is event loop in node?
3. Why node is single threaded?

According to Node.js documentation, a Node.js application runs using the event loop. The event loop is what allows Node.js to perform non-blocking I/O operations and explains how Node.js can be asynchronous. The event loop, aka the main thread, allows running one thing at a time. Having said that, Node.js JavaScript code runs on a single thread.





1. Explain the flow of login page UI to backend?
2. How we can implement security in our application ex. token
3. How will you do password encryption decryption?
4. Explain Reactjs MVC flow
5. What is node (overview)
6. Explain Redux workflow
7. What is state management
8. Difference between old react version and new react version
9. How can we handle component life cycle using react hooks
10. What is clouser?
11. What is the difference between map and set function?
12. What is hoisting?
13. What are the other libraries we can use with react for enhancing UI?
14. Mysql queries - on join

Hi Yesterday undergone React Interview for Health edge client

1.var addWith5 = addFunction(5);

console.log(addWith5(10)); // 15

console.log(addWith5(100)); // 10

var addWith10 = addFunction(10);

console.log(addWith10(10)); // 20

console.log(addWith10(100)); // 110

2. currying function

sumCurringFn() // 0

sumCurringFn(2)(3)() // 5

sumCurringFn(2)(3)(5)() // 10

sumCurringFn(2)(3)(5)(10)(100)() // 120

3.css tasks given one anchor tag and asked to hide text and load the url using only css

4.we have to align div horizontally n vertically

5.promises

6.anonymous functions

7.position

8.box model

9.react

10.components

11.y those components

12.life cycle methods

13.closure

14.anonymous function vs normal function

15.arrow functions

16.scoping

lexical scoping

18.y react is library? Hope it would be helpful

What are the different ways to pass data between parent and child component

(function() { console.log(1); setTimeout(function(){console.log(2)}, 1000); setTimeout(function(){console.log(3)}, 0); console.log(4); })();

2. for (let i = 0; i < 5; i++) {

setTimeout(function() { console.log(i); }, i \* 1000 );

}

3. let arrayOfLetters = ['a','b','c','d','e','f'];

const anotherArrayOfLetters = arrayOfLetters;

arrayOfLetters = [];

console.log(anotherArrayOfLetters);

is armstrong number or not