**Namaste React**

**Telegram video notes**.

What does emmet in vs code do?

What is a CDN ?(content delivery n/w)

Interview questions as well

Whether react is a library or a framework? Its a library.Main diff. Is that it takes minimum effort to put a library in our code and can be applied to a very small part of the code also.

Javascript:

createElement function is used to create html tag using JS.

appendChild, to append String to parent.

JS engine gives this functionality so that the browser knows what is a window and what is DOM.

In an array we can store values of different data types and push function can be used to add new values to the array. And POP message to delete the last item of the array.

2. Namaste React Web Series code repository https://github.com/namastedev/namaste-react

3. NetflixGPT Project Github repository - <https://github.com/akshaymarch7/netflix-gpt>

4. YouTube Project GitHub repository - <https://github.com/akshaymarch7/namaste-youtube>

**EPISODE 1: Inception**

Emmet generates code for us.

In vs code: html:5 gives basic skeleton of HTML.

Appendchild can be used to inject some child element inside a root element.

Browser has a JS engine in it, so it is able to understand the code, eg: document, getElementById etc.

What is CDN?

What is crossorigin?

What is the shortest program of JS? : empty file.

**CDN links** are basically websites where react is hosted and we are just injecting it into our website. If we open the src of these CDN links, they are basically Javascript code. React is actually JS files(written by facebook developers).

Ways to inject react in your code:

By CDN links.

<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script> // this is the core of react

<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script // this is the react library which is useful for DOM manipulations.

These are done in 2 different files as react also works in other devices.

**What is crossorigin?**

The crossorigin attribute sets the mode of the request to an HTTP CORS Request(Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism that allows a server to indicate any origins).

In react the createElement takes 3 arguments i.e. the tag, an object(used to give attributes to the tag) and the content of h1 tag(basically the children of the tag that we are creating). I we want more that 1 child i.e. create siblings then pass last argument as Array.

Creating an element is done by react library but creating a root is the job of the ReactDOM.

**First React code :**

<body>

    <div id="root"></div>

    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

    <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

    <script>

        const heading= React.createElement("h1", {}, "Hello world from react!");

        const root = ReactDOM.createRoot(document.getElementById("root"));

        root.render(heading);

    </script>

</body>

Try doing console.log and print the heading variable in above code: it will give an object.

React element is not some HTML tag element, it is actually a React object while rendering it to the DOM it converts itself into the HTML that browser understands.

If Root already has some tag inside it then , the render function will replace it with the content inside the render func.

**Basic React In Javascript not JSX therefore, not that simplified:**

const heading = React.createElement("div", {id: "parent"},

[React.createElement("div", {id: "child1"},

[React.createElement("h1", {}, "I am H1 tag" ),

React.createElement("h2", {}, "I am H2 tag" )]),

React.createElement("div", {id: "child2"},

[React.createElement("h1", {}, "I am H1 tag" ),

React.createElement("h2", {}, "I am H2 tag" )])]);

        const root= ReactDOM.createRoot(document.getElementById("root"));

        root.render(heading);

**EPISODE 2: IGNITING OUR APP**

We also need to know how to make our code production ready.

When we use: create react app , it creates a scaffold for us which is already prod ready.

**Npm**

NPM does not has a full form, but it works as node package manager. All the useful packages are hosted over there.

Installed npm, **Package.json is a configuration for npm,** Basically whatever we entered after doing **npm init** on terminal, comes up on this**.**

**What is the most important package in a project?**

It is a **bundler**. This helps in making the code to be minimized and packed for prod. Eg. Create react app uses webpack and babel bundler.

We will use **parcel**.

Terminal commenad: npm install -D parcel

We have 2 types of dependencies- dev and normal . Dev dep. Is generally required in development phase.

Normal ones are used in Prod also. So, we use –D for dev dependency.

In front of version: eg. 2.4.3, ^2.4.3 (^ is a caret) means it will automatically upgrade itself to a higher version. (even the minor update so this is recommended to avoid breaking of code)

~ this is tilde, it upgrades only to major version updates, not recommended as might break something.

After installing parcel:

Package.lock.json file kind of keep a rack of version of the dependencies and locks the version, it does not contain carat or tilde

Node modules : This contains the actual data or code of the dependencies. But other that just parcel we have so many other files as parcel might have its own dependencies and so on. This is called **transitive dependency**. (kind of a dependency tree)

Since we don’t want to put all these on Github so we put that file in .**gitignore** file. And package.json and package.lock.json can recreate all node modules by **npm install**.

**Command to ignite the app**: npx parcel index.html (root file name) (starting server)

Npx commands are to execute a package.

We can get React using NPM also as CDN links is not a good practice as it is a costly operation, if react version changes then we will have to change the url also.

So, we will install react and react-dom using npm.

**Functionalities of parcel:**

-Dev build

-Runs local server

-does HMR – hot module replacement i.e automatically refreshes the browser page when we save any change.

-File watching algo: which is actually written in c++.

-Gives faster build due to caching

-It will also do image optimization

-Does minification of file.

-Do bundling also

-compression of file

-consistent hashing

-code splitting

-differential bundling – support older and other different browsers

-Does diagnostics- gives better error message

-Gives a way to host in HTTPS (SSL)

-Tree shaking- will remove unused code

-different build for dev and prod (command for prod deployment: npx parcel build index.html)

In package.json we can also add the browsersList – you can also give country specific.

**Scripts** In package.json- tells which script to use. Eg for testing, for building it for dev or for building for prod.

For starting in dev mode:

For building for prod:

"start": "parcel index.html",

    "build": "parcel build index.html"

Now we can directly use npm run start or npm run build for running

React elements are kind of equivalent to DOM elements.

**EPISODE-3**

**JSX**

JSX is a JS syntax which is used to make easier to write HTML like structure in JS.

JSX is different from HTML, it is just HTML like syntax.

Const jsxheading = <h1> hello </h1> // this will work exactly like React.

JS engine will not understand this code, as it understands ECMA script (Eg. ES6) but parcel is transpiling(converted to code that browser can understand) it before it reaches JS engine. –This is done by **BABEL in Parcel. Babel is open source JS compiler**

**JSX=>BEBEL transpiles to React.createElement => React JS object => HTML Element (Render)**

**Explore various attributes of JSX-**

Multiline code in JSX should be written in round parenthesis.

**REACT COMPONENTS**

Everything is a comp. in react. There are 2 types of component:

1. Class Based components: this is the old way of writing code
2. Functional components: new way, it is just a JS function which is returning some JSX

We can also write arrow function without the arrow func. Also. And if multiline the write inside round bracket.

import React from "react";

import ReactDOM from "react-dom";

//React element

const jsxheading = (<h1>Namaste react using JSx</h1>);

console.log(jsxheading);

const root= ReactDOM.createRoot(document.getElementById("root"));

//React functional component

const HeadingComponent = ()=> {

        return <h1>Namaste functional component</h1>

}

// can write like this also

const HeadingComponent1 = ()=> <h1>Namaste functional component</h1>

//How to render func. component

root.render(<HeadingComponent/>);

What is component composition?

When we render react component into another component. Composing one component into another.

const Title = () => (<h1>Namaste react title comp</h1>);

const root= ReactDOM.createRoot(document.getElementById("root"));

//React functional component

const HeadingComponent = ()=> {

        return (

                <div>

                <Title/>

        <h1>Namaste functional component</h1>

        </div>)

}

If we use curly braces inside JSX scope then we can write write any JS expression inside it. So by using this we can also call a react element inside react component. Similarly element in an element.

These 3 is basically same as it is just a JS func.

{Title()}

                <Title/>

                <Title></Title>

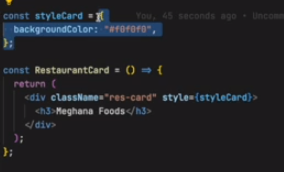
Read how JSX escapes attacks from bad API – Cross site scripting attacks.

**EPISODE-4**

How does a project planning work?

If we plan structure and position of elements then development becomes easy

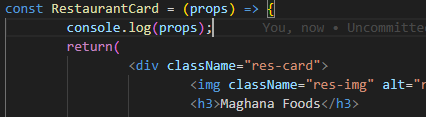
If we want to give inline styles in JSX then we cannot give as we give in HTML, the style attribute takes a JS object.

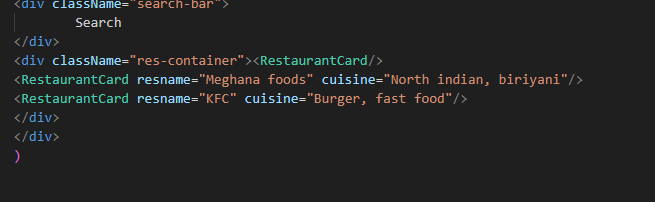


**PROPS**

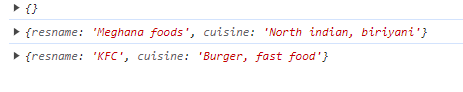
Props are like passing arguments to a function. Since functional components are nothing but JS functions so we can pass arguments to them as well. So props are used to pass dynamic data. This prop is basically passed as an object to the function. Eg:

If we pass like this and console log props in this funct.





Output:



So now the props will be used in the same way as we use a JS object. We can pass any number of props we want to.

DESTRUCTURING:

Instead of passing props as a whole object we can also destructure it. Eg:

(props) 🡪({resname, cuisine}) // these both will function similarly. OR

Const {resname, cuisine} = props

So pass as props.resname or directly resname after destructuring.

Config driven ui: This means that our website is driven by data or configs. Eg, offers and restaurants , everything will be different for different cities and areas. Basically UI is controlled by data. The api or data is the config.

**How to print array with comma as separater?**

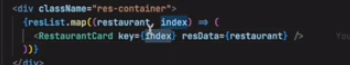
Use join function. Eg: arrayName.join(“,”);

What is optional chaining?

Instead of for loop we should use map(), filter(), reduce() in React JS. <https://www.youtube.com/watch?v=zdp0zrpKzIE>

API that I am using : <https://www.swiggy.com/dapi/restaurants/list/v5?lat=12.9715987&lng=77.5945627&is-seo-homepage-enabled=true&page_type=DESKTOP_WEB_LISTING>

**IMPORTANT:** We should pass a unique key for every element in the map function to avoid the warning. For example if there are 20 components at the same level or many number of restaurants, now if we add a new res card then React will rerender all the cards again as it will treat all the cards same . Some people also use indexes as key but we should not, it is a bad practice according to React docs only.



**EPISODE- 5**

Instead of all the code in root file we keep all the code in one source code folder or src folder. We need to move all the components to different different files.

Not important but good practice. Similarly, all the hardcoded data or strings should be kept in different file, some utils, constants or config files. Utils are like utilities that can be used across the app.

Normal way to export is:

export default filename;

but we cannot do multiple default exports. So then we use name export i.e. directly writing export in front of the const.

So now while importing we mention named imports inside curly braces.

Can we use a default export along with named export?

Here we will create a button which when clicked gives only top rated restaurant.

**Why is react faster?**

As it can do faster dom manipulation. Data layer and ui layer gets updates very quickly together.

**State variable in react**

We use react hooks to create state variables. Here we use the hook useState. It is called state variable as it maintains the state of the application. We pass the default value of the variable in the useState func. And we modify this default value by a set function.

So whenever a state variable changes, react re-renders the UI.

React hooks are normal JS utility functions. We need to import these functions.

useState(): used to create super powerful variables in react

useEffect()

**Whenever a state variable updates, react rerenders the component.**

**Reconciliation Algorithm:**

Also known as **react fiber.** The “reconciliation” algorithm in React is how the decision to re-render the component is made. In the browser, DOM manipulation is expensive and time consuming, both in mounting and unmounting. Part of what makes React very performant is its reconciliation algorithm.

If there is update in UI then actually virtual DOM is created first which is a representation of actual DOM.

**Diff Algorithm** : This finds out the difference b/w this virtual DOM and the previous one. So react fiber is an algo which finds this difference and updates virtual DOM and it uses diff algo for it. This is how react becomes fast. All this is a part of efficient DOM manipulation which makes react faster.

Learn more about react fiber here: <https://github.com/acdlite/react-fiber-architecture>.

Read **what is implemental rendering**.

const [listOfResState, setListOfResState] = useState(resList)

this is basically destructuring of array which means useState returns an array.

**EPISODE-6**

**Monolith architecture:**

Earlier only one project was created with all the things in one project eg. API, UI, Authentication, notification etc etc. So even for single change whole project was supposed to be deployed.

**Microservice Architecture:**

In this all the services are in different projects and all together make a big app. This is known as **separations of concerns** and follows a **single responsibility principle** where each n every service has its own job, they don’t interfere with each other.

How do these services communicate with each other?

All the servies run on their own specific port and all can be mapped to domain name.

There are 2 ways to make API calls to fetch the data:

1.Load the page 🡪 make api calls 🡪 after 500ms 🡪render UI

2.Load page 🡪render UI 🡪make API call 🡪after getting data re-render UI.

In react we always use the second approach as in that atleast the skeleton is visible first. And react has a very fast rendering mechanism so 2 renders is okay.

Useeffect callback function is called after the component renders. Eg. If it is written inside body component then. So this becomes helpful in the second approach of calling the API.

Fetch() function Is given to us by the browser. It is a JS function. Fetch returns a promise , so aync await is used to resolve the promise.

Cross-origin resource sharing (CORS) is a standard mechanism that allows JavaScript XMLHttpRequest (XHR) calls executed in a web page to interact with resources from non-origin domains. CORS is a commonly implemented solution to the same-origin policy that is enforced by all browsers.

Our browsers block us from calling one origin to another. When there is an origin mismatch then this error comes up.

CORS video of akshay

**38mins.**