Before going to dive at first we should know the difference between Library & Framework.

Library	Framework
1. Library is a set of reusable function used by computer program.	1. It is a set of piece of code that dictates the architecture of our project.
2. They are important in program linking & binding process.	2. They provide a proper standard way to build and deploy programs.
3. Code calls to library	3. Framework calls the code.
4. JQury is the Java script library.	4. Angular.js is the Javascript based frame-mework.

Why choose React JS over other frameworks libraries::

Component - Based Architecture :- Make it easy to manage and reuse components.

Virtual Dom :- Enhance performance by updating only parts of the DOM that need changes.

Strong Community & Ecosystem :- Large community support, extensive documentation.

React JS Intro

1. What is React JS?

React JS is a popular Javascript library for building user interfaces, especially single page applications. It allows developers to create reusable UI components, manage the state efficiently, and render updates quickly using a virtual DOM. Developed by Facebook, React simplifies the process of building dynamic & interactive web applications.

2. What are the features in React JS?

i) JSX :- JSX is a syntax extension that lets us write HTML-like code within Javascript, making it easier to visualize the UI structure.

ii) Components :- React builds UIs using components, which are reusable and self-contained pieces of the interface.

iii) One-way Data Binding :- Data flows in one direction from parent to child components, making the application easier to understand and debug.

iv) Virtual DOM :- React uses a virtual DOM to efficiently update and render changes, improving performance.

v) Performance :- React supports code performance to load components only when needed, improving the initial load time of applications

Now, Before going to deep dive into React we first know about "npm"

What is NPM?

npm short for Node Package Manager, is a tool that allows developers to manage & share reusable code packages. It is the default package manager for Node.js and helps in:

- Installing :- Downloading & Installing libraries and tools from the npm registry.
- Versioning :- Managing different version of code dependencies.
- Sharing :- Publishing & sharing our own packages with others.

Commands ⇒

* To check node version:

```
node -v ↵
```

* To check npm version:

```
npm -v ↵
```

* To run the latest updated npm:

```
npm install -g npm@latest ↵
```

N.B: npm version should be > 10.5.1

* The '-g' flag in npm stands for global it install the package globally on our system, rather than just in current project directory.

npm commands :-

1. Create a new React App:

```
npx create-react-app my-app ↵
```

or,

```
npm init react-app my-app ↵
```

2. Start the development Server:

```
npm start ↵
```

3. Build the app for production:

```
npm run build ↵
```

4. Run test :

```
npm test ↵
```

5. Install dependencies:

```
npm install package-name ↵
```

6. Uninstall dependencies:

```
npm uninstall package-name ↵
```

N.B :- code. command is used for open VS code from cmd.

What is DOM in React?

The DOM (Document Object Model) is a programming interface for web docx. It represents the structure of a docx as a tree of nodes. In the context of React, the DOM is crucial because, React interacts with the DOM, to render the UI.

What is JSX?

JSX is short form of Javascript with XML

and is a javascript extension. It is used to create React User Interfaces.

Explain about file folder structure React.

```
my-react-app/
├ node_modules/
├ public/
| ├ index.html
| ├ favicon.ico
| └ ...
├ src/
| ├ components/
| ├ Home.js
| ├ About.js
| ├ Contact.js
| ├ ProfileImage..js
| └ ...
| ├ App.js
| ├ App.css
| ├ index.js
| └ ...
├ .gitignore
├ package.json
├ README.md
└ yarn.lock (or package-lock.json)
```

node_modules:- This folder contains all the dependencies install using npm or yarn.

public:- This folder contains static assets like images, as fonts and index.html.

src:- This folder contains the source of application.

components:- This folder contains reusable react components.

container:- This folder contains higher order components that wrap around other components.

actions:- This folder contains action creators for Redux or other state management libraries.

reducers:- This folder contains reducers for Redux, or other state management libraries.

utils:- The folder contains utility functions and helpers.

index.js:- The file contains the main application component, the entry point application.

App.js:- Main Application component.

setupTest.js:- Setup file for tests.

package.json:- Metadata & dependencies for the project.

.gitignore:- Files and folders to ignore in Git version control.

package-lock.json:- Ensure consistent installation of dependencies across different environments.

Readme.md:- A markdown file with information about our project.

3. Why React JS called Single Page Web App?

Ans: React JS allows us to build applications, each have a single HTML pages that loads initially and then dynamic updates content as user interact with application without requiring full page reloads.

i) Single HTML page
ii) Dynamic content updates
iii) Client side rendering

* N.B:-
- To uninstall all node modules:
  rm -rf node_modules ↵
- For install nodes:
  npm install ↵

4.

JS	JSX
(i) JS is used traditional JavaScript syntax	(i) JSX uses HTML-like syntax
(ii) JS is for general programming language itself.	(ii) It is specifically for creating React components.
(iii) JS is executed directly.	(iii) JSX is transpiled

5. What is Virtual DOM?

Ans:- The Virtual DOM is a programming concept used to optimize the performance of web application. It is primarily associated with frameworks like React, which is used it to improve the efficiency of UI updates.

✳ App.js, In component folder create hom Page, about Page and contact etc.....

home.jsx:

```
const HomePage = () => {
    return (
        <div>
            <h3> Welcome Everybody into
                Home Page </h3>
        </div>
    );
};
function HomeTab1 () {
    return (
        <div>
            <h2> Hello HomeTab1 </h2>
        </div>
    );
}
function HomeTab2 () {
    return (
        <div>
            <h2> Hello Home Tab2 </h2>
        </div>
    );
} export default HomePage;
```

```
export { HomeTab1, HomeTab2 };
```

about.jsx:

```
const AboutPage = () => {
    return (
        <div>
            <img src = "assets/img.jpg"
                alt = "img_Name" height {300}
            />
            <h3> About Section </h3>
        </div>
    );
};
export default AboutPage;
```

app.js:

```
import './App.css';
import HomePage, { HomeTab1, HomeTab2 }
                    from './components/home';
import AboutPage from './components/
                        about';

function App(){
    return (
        <div className = "App">
            <p> <b> Welcome to our Webski-
                _tter Acadamy </b><p>
            <h1> i am a loyal employee </h1>
            <HomePage/>
            <HomeTab1/>
            <HomeTab2/>
```

```
        <AboutPage/>
    </div>
    );
}
export default App;
```

Output:

Welcome to our Webskitters Academy
i am a loyal employee
Welcome everybody into Home Page
Hello Home Tab1
Hello Home Tab2
About Section

* Passing values from Parent class to Child cl

Parent.jsx:

```
import React from "react"
import child from "./child"
const Parent =()=>{
    let n1="raj";
      n2=[1,2,3,4];
    return(
      <div>
      <child aaa={n1} aa2={n2} />
      </div>
    );
};
export default Parent;
```

child.jsx:

```
import React from "react";
const child = (props) => {
   return (
      <div>
        <p>
          {props.aaa} {props.aa2}
        </p>
      </div>
   );
};
export default child;
```

Output: raj 1234

* Passing objects values in one parent Class:-

```
import React from "react";
// import child from "./child";
  const Parent = () => {
      const obej ={
        firstname: "Achintya"
        lastname: "Maiti",
        age: 12,
      };
      return(
        <div>
          <p>
            Age: {obej.age},
            First Name: {obej.firstname}
          </p>
        </div>
```

```
    );
};
export default Parent;
```

Output: Age: 12, First Name: Achintya

⊛ By using Array:-

```
let arr = ["Sonu", "Arya", "Chayan"];
return (
    <div>
      { arr.map ((x) => (
          <p> {x} </p>
      ))
      }
      </>
```

Output: Sonu
Arya
Chayan

By using Array of Object:-

```
const title = [{fname: "SSD", lname: "Konar",
                age: 23},
               {fname: "Htr", lname: "Ali",
                age: 75}];

return (
    <div>
     {
       title.map ((x) => (
          <p> {x.fname} </p>
       ))}
```

Output:- SSD
Htr

Problems:-

Ex-1 Show the color and create their id by passing array or,
How do I pass arrays and objects as props from one component to another component and how can you access and render those props within the child component?

```
import React from "react";
import componentB from "./componentB";

const ComponentA = () => {
    let colors = ["red", "green", "blue",
                  "yellow"];
    let emp = [
    {
        id:1, name: "Happy"},
    { id:2, name: "John"},
    {id:3, name: "Rony"}
    return (
      <div>
          <componentB colors={colors}
                      emp={emp} />

      </div>

    );

};
export default ComponentA;
```

```
import React from "react";
const componentB = (props) => {
    let {colors, emp} = props;
    return(
      <div>
```

```
{colors.map((clr, index) => (
    <p key={index}> clr </p>
))}
{emp.map((x) => (
    <div key={x.id}>
        <p> id: {x.id} </p>
    ))} </div>
    </div>
);
};
export defaut component B
```

6. What is key?

Ans:- In React, the key is a special attribute that is used when rendering list of elemen It helps React efficiently identify which items have changed, been added or remov enabaling it to optimize rendering perfor -nce during updates.

e.g:
```
const items = ["Apple", "Banana", "Orange"];

const Itemlists = () => {
    <div>
    <ul>
    {items.map((item, index) => (
        <li key={index}> {item}
        </li>
    ))}
    </ul>
);
}
```

7. What is props? Define with example.

Ans:- In React, the props (properties) are used to pass data from one computer to another, typically from a parent component to a child component. Props are read only meaning they cann't be modified by the receiving component.

e.g:
```
import React from 'react';
import Greeting from './greeting';
function App(){
    return(
        <div>
            <Greeting name="John" />
            <Greeting name="Smith" />
        </div>
    );
}
export default App;
```

This is the example of parent component. Now, the child component below:

```
import React from 'react';
function Greetings (props){
    return <h1> Hello {props.name}! </h1>
}
export default Greeting;
```

O.P: <h1> Hello John! </h1>
Hello Smith!

8. Difference between key & props.

Ans:-

key	props
(i) Used to uniquely identify elements in a list.	(i) Used to pass data fro parent to child compon
(ii) It's scope only relevant within the context of lists.	(ii) It's scope is used to pass data and event handlers.
(iii) It is not accessible in child component.	(iii) It is accessible in child component.
(iv) Data type must be a unique string or number.	(iv) Can be any data types.
(v) `<li key {item.id}>`	(v) `<childComp propName =value/>`

Ex:- Using key:

```
const keyitems = ['Apple', 'Banana', 'Cherry']
return (
  <ul>
    {items.map((item, index) => (
      <li key={index}>{item}</li>
        // 'key' uniquely identifies
          each item in the list
    ))}
  </ul>
);
```

Using Props:

```
function Item(props) {
    return <li>{props.name}</li>
        //'props' is used to pass data to
                the component.
}
const items = ['Apple', 'Banana', 'Cherry'];
return (
    <ul>
        {items.map((item, index) => (
            <Item key={index} name =
                {item} />
            //'name' is passed as a prop
        ))}
    </ul>
);
```

9. What is props drilling?

Ans:- Props Drilling in React refers to the process of passing data from a parent component to deeply nested child components, often through intermediate components that don't use the data themselves.

Ex:

```
import React from 'react';
function Parent(){
    const user = {name: John, age: 30}
                //Data to be passed.
    return (
        <div>
```

```
        <child1 user={user}/>
      </div>
    );
  }
  function Child1 ({user}){
    return <child2 user={user}/>;
  }
  function child2 ({user}){
    return <child3 user={user}/>;
  }
  function child3 ({user}){
    return <div> {`Name: ${user.
      name}, Age: ${user.age}`}
      </div>
  }
  export default Parent;
```

10. Problems with Propsdrilling:-

(i) Difficulty in Maintenance: As our com-ponent tree grows, we might have to pass props through multiple levels of components that don't use data, leading to unnecessary complexity & boilerplate code.

(ii) Unnecessary Re-renders: Intermedia components may re-render un-necessa-ily when the props changed, even thou they don't use the data.

(iii) Harder to Manage: When the structure of our app grows larger, managing data flow and ensuring that props are passed correctly can become cubersome.

11. Soln to avoid Props Drilling:- The context API is a way to share values accross the entire component tree without having to explicitly pass props at every level.

e.g:

```
import React, {createContext, useContext}
                    from 'react';
const UserContext = createContext();
function Parent(){
    const user ={name: 'Johnaton', age:
                                    30};
    return (
        <UserContext.Provider value=
                        {user}>
            <Child1/>
        <UserContext.Provider>
    );
}
function Child1(){
    return <Child2/>;
}
function Child2(){
    return <Child3/>;
```

```
}
function Child3(){
    const user = useContext(UserConte
    return <div>{`Name: ${user.nam
            Age: ${user.age}`}</div>;
}
export default Parent;
```

12. Difference between Real DOM & Virtual DOM:-

Real DOM	Virtual DOM
(i) The actual HTML document structure in the browser.	(i) A lightweight JS representation of the Real DOM.
(ii) Slower because it updates the entire tree.	(ii) Faster because it updates only the cha-nged elements.
(iii) Directly updates the entire DOM causing performance issues.	(iii) Using Diffing Algori-thm to update only the necessary parts.
(iv) Requires more memory as the entire DOM is updated.	(iv) Only consumes less memory since change are handled in a virtu environment first.
(v) Every change trig-gers a full page re-render.	(v) Only the componen with change get re-rendered.

13. How does Diffing Algorithm works in React?

Ans:- React Diffing Algorithm is a smart way to update the Real DoM efficiently by only applying minimum necessary changes. This makes React super fast.

Steps:-

(i) Create the Virtual DOM:- React first creates light copy of the Real DOM in memory.

(ii) Render New Virtual DOM :- When state or props change, React creates a new Virtual DOM.

(iii) Diffing Process :- React compares the new Virtual DOM with the previous one to find differences.

(iv) Find changes :- React identifies only the changed elements insted of re-rendering the whole page.

(v) Reconsiliztion :- React updates only the changed parts in the Real DOM for better performance.

14. What is Reconciliation?

Ans:- Reconciliation is the process React uses to efficiently update the Real DOM when a component's state or props change.

E.g :-
```
<div>
    <h1> Hello Johnathon! </h1>
    <p> Welcome to React </p>
</div>
```

Update Virtual DOM :-

```
<div>
    <h1> Hello, Johnathon </h1>
    <p> Welcome to React JS </p>
        <!----- Only this text
                        change--->>
</div>
```

15. What is Higher Order Component?

Ans:- A higher order component is a functio
that takes a component as input and retu
a new enhance component with additiona
functionality.

Syntax:

```
const withExtraInfo =
        (WrappedComponent) => {
    return (props) => {
        <div> style ={{
            border: "2 px, solid, blue"
            padding: 10 px" }}>
            <WrappedComponent {... prop
        </div>
    );
};
```

- WrappedComponent is the original compo
- ent.
- It returns a new component with additional features.