**Transpilers:** A transpiler is a special piece of software that translates source code to another source code. It can parse (“read and understand”) modern code and rewrite it using older syntax constructs, so that it’ll also work in outdated engines.

**Polyfill: HELPER CODE** A script that updates/adds new functions is called “polyfill”. It “fills in” the gap and adds missing implementations.

[**https://javascript.info/polyfills**](https://javascript.info/polyfills)

**BABEL: TRANSPILER** is one of the most prominent transpilers out there.

**Webpack**: **MODULE BUNDLER AND REQUIRE NODE.JS TO USE**

Modern project build systems, such as webpack, provide a means to run a transpiler automatically on every code change, so it’s very easy to integrate into the development process.

Webpack is a great tool which can be easily used out of the box to automatically load and bundle CSS and other files and assets in addition to JS , with the help of an easy to set up config file. And at the same time, offers the users the freedom to perform much more complex tasks with its vast configurable options, and plugins.

<https://webpack.js.org/concepts/>

<https://medium.com/js-imaginea/webpack-why-and-what-4948433cc2d3>

**FRAGMENTS & WHY REACT NEEDS ELEMENT WRAPPED IN DIV**:

JSX looks like HTML, but under the hood it is transformed into plain JavaScript objects. You can’t return two objects from a function without wrapping them into an array. This explains why you also can’t return two JSX tags without wrapping them into another tag or a Fragment.

**THIS IS HOW JSX:**

import React from 'react'

function Greet() {

return React.createElement("h1", {}, "Hello, World!")

}

## Important JSX Rules

1. You can only return one top-level element from a given component. This is usually known as a parent element and is used to group the content **USE FRAGMENT**. Remember, JavaScript is the backbone of React, and in JavaScript a function can only return one value.
2. Some elements in HTML do not have a closing tag. In React JSX, every tag, including those with no closing tags, must be closed. If you have an element that doesn’t have a closing tag, you have to add a slash at the end (e.g., <hr/>).
3. A React component must be capitalized. Component names that do not begin with a capital letter are treated like built-in components, and it results in strings (“div”, “span”…). When the component name is capitalized, it is treated as an identifier instead of a string.
4. To include JavaScript expressions in JSX, you need to wrap them in curly braces. Content between the opening and closing curly braces will be evaluated as JavaScript.
5. The term “class” is a reserved keyword in JavaScript. In React, we must substitute className for class.

**Lecture # 1:**

**What is React JS?**

It is an open-source, client-side, **JS Front-End Library** developed by Facebook.

**Why we use React?**

It's used for **building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScrip.**

**It is used for building SPA.** just like Netflix, Instagram, whatsapp, Gmail, Google Maps, Pinterest and Paypal.(need defination)

It is a component-based front-end library responsible only for the view layer of a **Model View Controller (MVC)** architecture. React is used to create modular user interfaces and promotes the development of reusable UI components that display dynamic data.

**History of React?**

First Release on 29-may-2013.

**Why to Learn React?**

1. Maintained by FB that’s why so it will long term player in market.
2. Mobile app development React-Native.
3. Large community for support.

It is a component based model for the development of the website.

**Lecture # 2:**

We can use the React JS without installing anything like NPM or React. For this purpose we use CDN (content Delivery Network). Example is given in the lecture.

**Lecture # 3:**

Installation of the React with **UBUNTO**.

**Lecture # 4:**

Installation of the React with **WINDOWS**.

Install node: both NPM and NPX automatically installed with node

**Npm (**Node package manager**):** Node package manager

**Npx (**Node Package eXecute**):** It also automatically also installed with the NPM. Actually it is a package runner tool which allows us to run JavaScript packages without installing it and come with the 5.2 and above.

**Alternatives to NPM:**

1. Yarn (born in FB and Google)

2- Bit 3- MNPM 4- TURBO

<https://blog.bitsrc.io/4-npm-alternatives-best-js-package-managers-and-publishing-tools-fe6779937ee9>

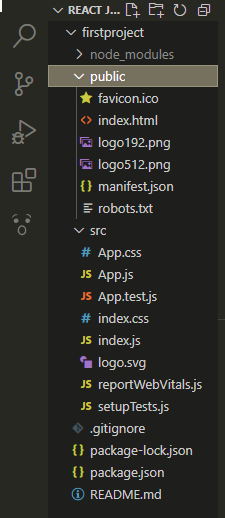
**How to build the project for sharing with others:**

**Lecture # 5:**

First new project and new JS file for REACT

**Lecture # 6:**

All the details about the folder structure.



**Node Modules :** it contains all the code of dependencies of the project. We don’t need to share it with other when we commit code because it node automatically generate on the basis of dependencies present in the Package.JSON file because it keeps the record of the all the dependencies.

**Public Folder:** Mainly it contains the **index.html** file which contains the div with **id:root**  that is targeted in **index.JS**  all the code written in React components basically run by this html file.

It contains some png’s for the react initial project.

**Manifest.js:** it keeps the meta file e.g. icons details.

**Robot.txt:** This file is used to guide crawlers and tell them what resources they shouldn't access. Means help to search engine to prevent it from irrelevant access.

**Index.JS**: it is our first **entry point** we enter in our website. It import the **app.JS** file and wrap it

**App.JS:** it is a file in which we write our first code.

**Package.JSON:** it contains all the details about the packages, dependencies, libraries and commands used in our react project. It keep the detail about our react application.

**Package-lock.JSON:** it contains complete details (History) about the installed packages in our react app. It keeps details about the node module file means all the packages details.

**Important file in React project ?**

Package.json is an important file than all other files

**Why we always ignore node modules and then how other developer get it?**

Node modules file takes huge space and it will take more time for updating that’s why we ignore this file during pushing.

**Then how other developers are able to get the node modules file?**

**29-Exploring The React Project**

* Index .js file is the first file which will be executed first. root is the entry point in the application.
* Index.html is the only file that is present in the project which contain nothing just a div with id=’ root‘ this is the reason why we call it SPA.
* App.js is the main component we can say it a root component and all other components that we will make are custom components. All customs components are import in the App component means in main component.

Custom component must start with the Capital Letter because when it imported in the App component React recognize it by its style**.**

**Lecture # 7: Package.json in details**

**npm install command:** it simply check the package.json file and check inside the project which library/dependency present it something is not present it will install according to the details present in the packge.json.

**Custom commands in react js:**

Npm start can be replace with npm run hamza.

**How many types of components in React.Js?**

**Lecture # 8:** Functional components in React JS.

For F-components we need to add the file.js and write function then export the file to app.js

This function is a valid React component because it accepts a single “props” (which stands for properties) object argument with data and returns a React element. We call such components “function components” because they are literally JavaScript functions.

**Types of component:**

* 1] Functional Components: In simple words, Functional components are javascript functions. ...
* 2] Class Components./ stateful Components/ Smart Components.
* 3] Higher-Order Components.
* 4] Presentational Components / stateless component/ Dumb Components.
* 5] Container components.

<https://articles.wesionary.team/types-of-react-components-you-should-know-251cceacd8ac>

**How many ways to export/import the component:**

export default Userabc; //1st way

export *const* Userabc = () *=>* { //2nd way}

but when we import them there is a different

import Userabc form ‘./Userabc’ //1st way

import {Userabc} from ‘./Userabc’ //2nd way

we must use this bracket if we are not using the export default because it maybe more then one component.

**Lecture # 9:**

**Is it possible to write a component inside of other component?**

Yes, it is possible because this is like to write a function inside of the other function. Which can be used inside of the scope of parent function.

**Class component:** in class component we can’t be able to console outside of the function.

import React, { Component } from "react";

export default *class* ClassComponent extends Component {

  render() {

    return (

      <div>

        <h1>This is class component</h1>

      </div>

    );

  }

}

**Class Components**

* Class components make use of ES6 class and extend the Component class in React.
* Sometimes called “smart” or “stateful” components as they tend to implement logic and state.
* React lifecycle methods can be used inside class components (for example, componentDidMount).
* You pass props down to class components and access them with this.props.

**Lecture # 10: JSX**

JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement()  and/or appendChild() methods. JSX converts HTML tags into react elements.

it is special (syntax) not a standard syntax which is enabled in React project which we create by command of **create-react-app**. Without jsx code will be more complicated just like we have done in App.js.Where we also clear that why we should wrap code in single div.

**Lecture # 11: Click Events**

**Lecture # 12: State in React JS**

**O.S**

State is a container like variable in js but in React we can’t be able to manage the component with variable React understand the state and get component get re-render with updated state not only itself but also update the children component.

**What Is ‘State’ in ReactJS?**

The state is a **built-in React object** that is used to contain data or information about the [component.](https://www.simplilearn.com/tutorials/reactjs-tutorial/reactjs-components) A component’s state can change over time; whenever it changes, this object instructs the component to re-renders. The change in state can happen as a response to user action or system-generated events and these changes determine the behavior of the component and how it will render.

**Why state not simple variable?**

1. A state can be modified based on user action or network changes
2. Every time the state of an object changes, React re-renders the component to the browser
3. The state object is initialized in the constructor
4. The state object can store multiple properties
5. this.setState() is used to change the value of the state object
6. setState() function performs a shallow merge between the new and the previous state

**The setState () Method**

State can be updated in response to event handlers, server responses, or prop changes. This is done using the **setState()** method. The setState() method enqueues all of the updates made to the component state and instructs React to re-render the component and its children with the updated state.

Always use the [setState() method](https://css-tricks.com/understanding-react-setstate/" \o "setState() method" \t "_blank) to change the state object, since it will ensure that the component knows it’s been updated and calls the render() method.

<https://www.simplilearn.com/tutorials/reactjs-tutorial/reactjs-state>

function component

import { useState } from "react";

import React from "react";

*const* FunctionComponent = () *=>* {

*const* [data, setData] = useState("initial Data for heading");

*function* funClick() {

    setData("state settled by the useState Hook");

  }

  return (

    <div>

      <h1>{data}</h1>

      <button onClick={funClick}>click me</button>

    </div>

  );

};

export default FunctionComponent;

**Is it possible to use the state of the component into another component?**

Authentically not but we have ways to use but react restrict to use out state object.

**Is state public or private?**

It is public

**Lecture # 13: State in Class component**

Class component

export default *class* Bike extends Component {

*constructor*(*props*) {

    super(*props*);

    this.state = {

      make: "Yamaha",

      model: "R15",

      color: "blue",

    };

  }

  changeBikeColor = () *=>* {

    this.setState({ color: "black" });

  };

  render() {

    return (

      <div>

        <h2>My {this.state.make}</h2>

        <p>

          It is a {this.state.color}

          {this.state.model}.

        </p>

        <button type="button" onClick={this.changeBikeColor}>

          Change color

        </button>

      </div>

    );

  }

}

**Is it possible to use Hooks in class component?**

No

**Lecture # 14: Props in function component**

Props are used to store and access data coming from the App.js into our customized component.

React Props are like functions arguments in JavaScript and attributes in HTML.

To send props into a component, use the same syntax as HTML attributes and to receive props in the custom component as JS function parameter.

We must have data in component from where we are calling the other component like in our case<App.js is main and ExpenseItem is custom component>

**Is it possible to change props inside the receiver component?**

No, its not possible props only change inside the major/sender/App.js component both in class and function component.

**Is it possible to pass a function as a props? Lifting up state**

Yes I have done

**Lecture # 15: Props in Class component**

**Lecture # 16: Get the value of input box.**

**Lecture # 17: Toggle button && same with 2 button**

**Lecture # 18: form input**

**Lecture # 19: conditional rendering**

**Lecture # 20: Form validation**

**Lecture # 21: Pass function as a props**

This is actually lifting up state because in this way we will share children data to parent.

**Lecture # 22: Life cycle method**

There are three phase and many methods.

**Phases:**

* Mounting
* Update
* Unmounting.

**Methods:** Many Methods (1st- constructor)

**When life cycle methods run in program??**

React lifecycle methods is the series of events that happen from the birth of a React component to its death.

[**https://programmingwithmosh.com/javascript/react-lifecycle-methods/**](https://programmingwithmosh.com/javascript/react-lifecycle-methods/)

1-when component load.

2-when state and props update.

3- when component remove.

**Lecture # 23: constructor Life cycle method**

1. it will run first of all other method
2. when it run props also present here
3. in classes it must call the super() method to take access to parent properties
4. super() is a JS method not a react method ()
5. it calls only once in the life cycle method when component mount

**getDerivedStateFromProps() :** [**https://www.geeksforgeeks.org/react-js-static-getderivedstatefromprops/**](https://www.geeksforgeeks.org/react-js-static-getderivedstatefromprops/)

**Lecture # 24: render Life cycle method**

1. it is a life cycle method
2. in class component everything will be written inside of the render method
3. it call when component mount and **whenever** states and props get updated.

**Is it possible to write more than one render method or render inside the render?**

**Lecture # 25: componentDidMount()**

1. it will call after all the process of the mounting.
2. One of its usages is API call (because it is not updates everytime that’s why we call api in it
3. Compare to render method It will call after the render
4. Compare to didComponentUpdate it will before didComponentUpdate
5. It has no effect of state and props updation because didComponentMount and render effected by the states and props updation.

**Question is pending if we update the state inside this what will happen?**

**Lecture # 26: componentDidUpdate()**

1. It will call every time when the state/props will get updated
2. It **preserve** the previous state and props
3. We can update the state and prop in it but it will go infinite whenever state update it will run again and again.(but with condition it is possible to stop)
4. It takes three parameter **(preProps,preState,snapShot)** to preserve the previous state and third one is snapshot. Snapshot will give you undefined until unless you will not implement it.

**Is it possible to stop the componentDidUpdate()?**

**Yes with shouldComponentUpdate**

**Should we call api in it??**

**Yes, but if conditional**

**Lecture # 27: shouldComponentUpdate()**

1. It ask the question that should I render the component or not after updating the state???
2. byDefault behavior of this is to return false means it will not render the component

**which will update first componentDid or shouldComponent?**

**Lecture # 28: componentWillUnmount()**

**Lecture # 29: Functional Components-🡪Hooks**

1. class components contain in built features like(states, life cycle method and pure component etc) but function components are not have these states so we have to use **HOOKS.**

**Lecture # 30: Hooks useEffect()**

**Lecture # 31: useEffect() with states and props**

**Lecture # 32: styles 3way and moduler have high priority**

**Lecture # 33: Bootstrap**

**Lecture # 34: list the array with map**

Difference between Map and For loop

**Lecture # 35: list the array with BOOTSTRAP and resolve unique key error**

Install bootstrap and make table

**Lecture # 36:**

**Lecture # 36:**

**Lecture # 36:**

**Lecture # 36:**

**Lecture # 40:Pure component in class components**

How to control the re rendring when state/props get updated with the use of Pure component.

**Lecture # 41: useMemo() It is a hook alternate to pure component**

Used to improve the performance of the application. useMemo takes a call back function and in second parameter it takes the condition at which the function should be called.

**Lecture # 42: ref in class component**

-it should be used lesser/ at minimum.

- it is used to manipulate the DOM that’s why it should be used minimum as much less as possible.

-In form validation /access0 it should be used.

**Lecture # 43: useRef()**

In function component it is used to manipulate the DOM.

**Lecture # 44: forwardRef()**

It is a wrapper which is used in the child component whose element or DOM is need to be manipulated through the parent component.

**Lecture # 45: Controlled Component**

Functional component in which we manage the input fields with the use of states and props

Components in which input fields are not controlled by states but controlled by the ref/document.getElement….(directly JS) means in which directly DOM is acquired for the manipulation.

**Lecture # 46: Uncontrolled Component**

**Lecture # 47: HOC**

**Lecture # 48: React Router**

It is used to make the router for different pages.

**Lecture # 49: React Router**

How to create <link tag>, disadvantages of using anchor tag.

**Lecture # 50: React Router Page 404**

How make page 404 with “/\*” and how to use navigate to the by-default set page.

**Lecture # 50: React Router Params**

Use single component for the data of multiple users coming from database.

**Lecture # 51: Styling of Navbar and Difference between Link and Navlink**

We should not add class with Link but can be use with the Navlink.

**Lecture # 52: how to make link active**

Method #1: by using active class.

Method #2: by using inline style.

**Lecture # 53: searchParam()/setSearchParam**

**Lecture # 54: searchParam()/setSearchParam**

setSearchParams: To give parameters to the url manually

searchParams: is used to get the manually setteled parameter of the url.

**Lecture # 55: useNavigate()/Dynamically Routing / Navigation on Click**

It is used to make the button as link.

**Lecture # 56: Nested Routing**

-----------------------done-----------------

**Lecture # 57: useLocation()**

It gives all the details about the route

State (with state we can share the data of the of component state)

Path

Hash

Search

**Lecture # 58: Protected Routing**

**Lecture # 63: API** Application Programming interface

Almost data present in api in the from of json.

Get: for fetching

Post: for adding

Put: for updating

Delete: for deleting

**Lecture # 69: previous state**

How can we preserve the previous state after updation.

**Lecture # 70: provious Props**

Handle with the use of the useRef.

**Lecture # 71: if initial value is object in case of useState??**

**After context API I am starting interview question**

**Lecture #101: Interview Question**