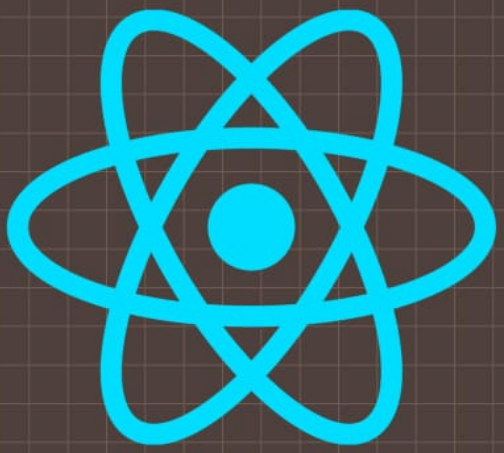
Note Taking

Namaste React



Khushi Johri Notes

Episode 1 - Inception

What is React?

React is an Open-source JavaScript library that is used for creating User Interfaces for Single Page Application (SPA). It is mainly used for handling view layer of any Web or Mobile Application.

Apps Used

Browser
- Chrome

Code Editor
- VS Code

Create an HTML file in VS Code

→ Drag-Drop a folder in VS code
→ Create a file index.html
→ Use Emmet to create a boiler plate :html5

Ways to Add React in our project

1) Using CDN links in Script Tag

CDN → Content Delivery Network, these are websites where react has been hosted and we are pulling react into our project

Why do we use CDN?
To reduce latency (Delay in network communication). It is that annoying delay you experience when trying to access a web page or video stream before it fully loads.

HW What is cross origin in this script tag?
A cross origin request is a request for a resource (eg: style sheets, iframes, images, font or script) from another domain.

Who has written this react code?
Some developers of Facebook

What happened when we got React into our Project?
It gives us alot of different important functions and methods to use in our project.

What are we using 2 CDN links?
First is react.development.js this is the core of react.

Second is the react-dom.development.js to modify the Dom (Document Object Model)

Why there are 2 files and not just 1?
React works on other devices using React Native in Mobile Phones, React 3D and more.

1st file is the core react and 2nd file is the react-dom file which make a bridge between react and browser.

Creating Elements in React

Create Element in React

```
const heading = React.createElement("h1", {}, "Hello World")
```

Element

{} → id: "heading" to give attribute to tag

"Hello World" → Content to display inside Element

```
const root = React.DOM.createRoot(document.getElementById("root"));
```

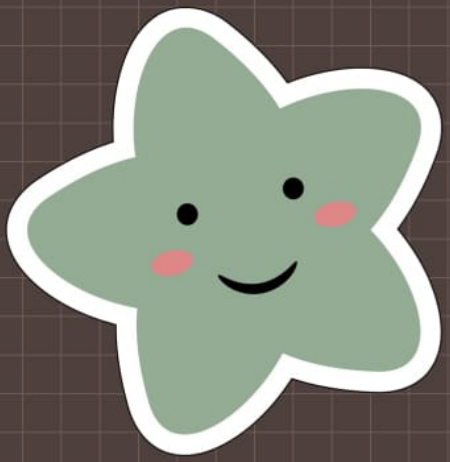
To put the element on the DOM we use React DOM library

```
root.render(heading)
```

React was built to manipulate DOM using JS

Most costly thing is change the dom tree on some update in the project

To optimize it React is used to do this using JS.



- Import JS in HTML
  `<script src="./App.js"></script>`
- Import CSS in HTML
  `<link rel="stylesheet" href="./index.css" />`

HW What is createElement? What does heading return?

```
const heading = React.createElement("h1",
                {id:"heading", xyz:"abc"},
                "Hello World from React!"
                );
console.log(heading)
```

{id:"heading", xyz:"abc"} → attribute

"Hello World from React!" → children

React Element is an object

Props are children + attributes we are passing

```
root.render(heading)
```

→ used to create the <h1> tag and display it in the root

render is used to convert object into tag

How to create nested elements in React ?

```
/*
   <div id="parent">
      <div id="child">
         <h1> I'm h1 tag </h1>
      </div>
   </div>
*/
const parent = React.createElement(
                    "div",
                    { id : "parent",
                    React.createElement(
                          "div"
                          { id : "child" },
                          React.createElement ("h1",
                          {},
                          "I'm h1 tag"
                       ));
root.render (parent);
```

ReactElement (Object) ⟹ HTML (Browser Understands)

How to create sibling elements in React ?

To pass siblings we need to pass it in an array in the children

```
<h1>.... </h1>
<h2>..... </h2>
```

```
[React.createElement("h1", {}, "I'm an h1 tag"),
 React.createElement("h2", {}, "I'm an h2 tag"),
]
```

Will the order of files in HTML file matter?

It will throw an error. The order needs to be in sequence.

What happen if something is already inside

```
<div id="root">
   <h1> Khushi </h1>
</div>
```

When we try to do root.render(....) and if something is already present inside the div tag then root.render will replace the child tag inside the div.

It is loaded but within few milliseconds replaced by root.render(parent)

What is the difference between a library and a framework ?

Library can work in a small part of the app too unlike a framework.

Framework need to be used in a fully fleged app to use them. They can't be used in already existing project.

But react can be used inside your existing project as well.

Episode 2 - Ignite Our App

1) Create a Repo on GitHub

2) In project add these commands

- git init
- git add README.md
- git commit -M main
- git remote add origin ....
- git push -u origin main

Can only React make our fast?
React can only make an app fast to an extend

NPM => It does not have a full form
It stands for anything but no node package manager.

It manages packages.
All the packages and utilities that we need in a project come from npm.

How to include npm in project?
In terminal
- npm init
  package name: (namaste-react)
  version: (1.0.0)
  description: This is Namaste React
  entry point: (App.js)
  test command: jest
  git repository:
  Keywords: react, namaste react, Js
  author: Khushi Johri
  license: (ISC)

package.json has been added to the project

package.json file is a configuration for our npm

packages are also known as dependencies

npm will take care of the version of the package

What important package/dependancies do we need to install?

Bundler → Our whole code needs to be bundled together, it needs to be minified, cached, compressed, cleaned before we send it to production

Webpack, parcel, vite are some Example of bundler

It packages your app so that it can be shift to production

How to install Parcel?
Run the following command:

npm install -D parcel

dev dependencies → It is generally requires during development

normal dependancies → It is used production also.

What is package-lock.json?
It keeps the track of exact version of the dependency

Package.json keeps approx version & package-lock.js keeps exact version.

What is node-modules?
All the code we fetch from npm. It's like database for our dependency.
Dependencies required for a dependency is called Transitive dependencies. Every dependency will have package.json with its own dev & normal dependencies

Should I put all these node modules to git, production?

No, if I have package.json & package-lock.json
It can regenerate our node modules

What is .gitignore?

When we don't want anything to not go on github or production, we use .gitignore

Go to .gitignore
and type /node_modules

Should I put package-lock.json & package.json on git?

Yes, maintains the note of what all dependencies our project needs

Ignite the app

npx parcel index.html

parcel has created a server
npm → command of npm
npx → executing a package

Another way of getting react in app

1. npm
2. CDN links → not prefered way of using react in a project

→ We can already have React in our node module we don't have to make a network called externally through CDN

→ Using CDN will not update it to the newest version automatically

In terminal,
npm install react / npm i react
npm install react-dom / npm i react-dom

How to use react in file after modules installation?

```
import React from "react"
import ReactDOM from "react-dom"
```

Error:
Browser Scripts can't have imports and Exports

<Script src = "./App.js"> </Script>
↓
Browser treats it like normal JS file

Normal JS files can't have imports & exports

We have to tell the browser that its a module

<script type = "module" src = "./App.js">
</script>

Now react is coming from module not from CDN links

import ReactDOM from "react-dom/client"
To remove warning

What does Parcel do?

- Dev Build
- Local Server
- HMR → Hot Module Replacement
- File Watching Algo → written in C++
  To keep a track of every change
- Caching → Faster Builds present in .parcel-cache
- Image Optimization
- Minification
- Bundling
- Compress files
- Consistent Hashing
- Code Splitting
- Differential Bundling → To support older & newer browsers along with devices
- Diagnostic
- Error Handling
- HTTPs
- Tree shaking Algo → Remove unused code
- Different dev and prod. bundlers

How to create production build?

In terminal,

npx parcel build index.html

// Error

Because here we are giving entry point as index.html but in package.json entry point is ./App.js

This conflict will lead to error

→ So "main": "App.js" needs to be removed from package.js

After creating production ready code, it will put it in dist folder in form of development build

Don't need to put dist & .parcel-cache on github as they can be created again

How to make a project compatible for older versions of browsers?

By using Browsers list in node-modules

Add this in package

Eg

```
"browserlist": {
    "last 2 Chrome version",
    "last 2 Firefox version"
}
```

You can search it on browserlist website

```
"browserlist": [
    "last 2 versions"
]
```

Episode 3 - Jaying The Foundation

Create script to start build of npx parcel index.html

In package.json, in "script" write

"start": "parcel index.html",
"build": "parcel build index.html"

In Dev Mode — Production Mode

To start it:
In terminal,
→ npm run start / npm start

To build it:
In terminal
→ npm run build

Remove React code from App.js

React.createElement is an object
render() will convert Object
into HTML

What is JSX?
It is a JavaScript syntax which is easier to create React elements.
It makes developer life easy.

** JSX is not a part of React, is not HTML inside JS. JSX is HTML like Syntax or XML like syntax
JS Engine will not understand JSX cuz it understands ECMA Script.

Eg Using JSX

```
const jsxHeading = <h1> Namaste React </h1>
```

jsxHeading → React Element
<h1> Namaste React </h1> → JSX

This is not pure HTML or Pure JS. This is JSX.

Eg Using Pure React

```
const heading = React.createElement("h1",
                {id: "heading"},
                "Namaste React
                );
```

** But still this is running on the screen. This is done by Parcel.
Parcel transpiler all this code before it goes to JS Engine. Then JS Engine recieves the code that browsers can understands

Transpile → Converted such that browser can understand

Parcel gives the responsiblity of transpilation to babel.

Babel is a package which is compiler / transpiler of JS. It takes JSX and convert it into the code which browsers understands. It is a library not created by Facebook.

JSX ⇒ React.createElement ⇒ ReactElement ⇒ JS Object ⇒ HTML Element (render)
Transpiled by Babel

render() will replace everything present in HTML file (if any)

If writing JSX in multiple lines then () are mandatory. To tell Babel where is JSX starting & ending.

```
// Code
import React from "react";
import ReactDOM from "react-dom/client"

//React Element
const jsxHeading = (
        <h1 className="head">
            Namaste React
        </h1>

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

In JSX, we use ClassName instead of class, we use camelCase, we don't use hyphen '-'

What is React functional Component?
Just a JavaScript function with return some JSX

```
Eg  const HeadingComp = () => (
        <h1> Namaste</h1>

    const HeadingComponent = () => {
        returning <h1> Namaste React </h1>
    };
```

Both are Same

A functional component returns a React Element

What are Component Composition?
A component inside a component

Inside React Component when {} is present we can write any JavaScript expression inside it

If you put one component inside other and other inside one then it will go into infinite loop.

Attacks

If someone gets access to your. Js code and sends some malacious data which will get displayed on the screen that attack is called cross-side scripting

It can read cookies, local storage, session storage, get cookie & read data, get info about laptop

Jsx takes care of your data

If some api passes some malacious data Jsx will escape it. It prevents cross-site scripting. It sanitizes the data before running.

```
<Title />
<Title></Title>
{Title()}
```

All are same

Components can be called like this in side Jsx

React Fragments

It allows you to retuen multiple elements from a React Component by allowing you to group a list of children without adding extra nodes to the DOM.

It behaves like an empty tag

```
<React.Fragment>
</React.Fragment>
```

or

```
<>
</>
```

Assignment 2 - Ignite Our App

Q1 What is NPM?

⇒ NPM is a package manager for the JavaScript programming language maintained by npm, Inc.
NPM will take care of the version of the packages. It consists of a command line client, also called npm and an online database of public and paid-for private packages, called the npm registry.

Q2 What is `Parcel/Webpack`? Why do we need it?

⇒ Parcel/Webpack are the bundlers used mostly for JavaScript or TypeScript code that helps you to minify, clean and make your code compact so that it becomes easier to send a request or recieve the response from the server when it usually take you to transfer multiple files without using any bundler for loading the page of your application

Both of these bundlers sustantially reduce the time it takes for the transfer of data and files to the server from the application. Along with that both bundlers parcel and webpack removes the unnecessary comments, new lines, any kind of block delimiters, and white spaces while the functionality of the code remains unchanged

Q3 What is `.parcel-cache`?

⇒ Cache folders stores information about your project when parcel builds it, so that when it rebuilds, it doesn't have to re-parse and re-analyze everything from scratch. It's a key reason why parcel can be so fast in development mode.

Q4 What is npx?

⇒ NPX stands for Node Package Execute. It is simply an NPM package runner. It allows developers to execute any JavaScript Package available on the NPM registry without Package available on the NPM registry without even installing it.

Q5 What is the difference between dependencies and devDependencies?

⇒ Dependencies are used for production or in testing environment. Whereas devDependencies are used for project development purposes only.

Q6 What is Tree Shaking?

⇒ Tree Shaking is used within a Javascript context to describe the removal of dead code. By using tree shaking and code splitting together, developers can create smaller, faster, and more efficient React application. These technique can help to eliminate unused code and split large application into manageable chunk, improving the performance and user experience of an application

Q7 What is Hot Module Replacement?

⇒ Hot Module Replacement is a feature that enables you to see code changes in the browser without having to refresh it, allowing you to preserve the state of your frontend application
It is used to retain application state which is used to retain application state which is lost during a full reload.

Q8 List down your favourite 5 superpowers of Parcel and describe any 3 of them in your words

⇒ Tree-shaking → It removes unwanted code like comments, white spaces while sending it to the production for speeding the application

Hot Module Replacement → It updates or removes or adds a module without refreshing the whole page. It make app fast

Caching → It creates caches of files which are built once and reuses the cache file so that rebuild of that file doesn't happen to faster the app.

Bundling

Minification

Q9 What is 'gitignore'? What should we add and not add into it?

⇒ When we don't want anything to not go on Github or production, we use .gitignore

Go to .gitignore
and type /node-modules

Q10 What is the difference between 'package.json' and 'package-lock.json'?

⇒ Package-lock.json keeps the track of exact version of the dependency.
While package.json keeps approx version

Q11 Why should I not modify 'package-lock.json'?

⇒ As it contains exact version of dependencies rather than approximation like package.json. It will automatically regenerate new version when package-json change. It has nested dependencies along with exact version unlike package.json.

Q12 What is 'node-module'? Is it a good idea to push that on Git?

⇒ All the code we fetch from npm. Its like database for our dependency. Dependencies required for a dependency is called Transitive dependencies.

Every dependency will have package.json with its own dev & normal dependencies

Q13 What is the 'dist' folder?

⇒ It is where the compiled code is stored. This is the code that is ready to be deployed to production

Q14 What is 'browserlists'?

⇒ It defines & shares the list of target browsers between various frontend build tools. It is used by autoprefixer, Babel. It is a tool that allows specifying which browser should be supported in your frontend app.

Assignment 3 - Laying the Foundation

Q1 What is Jsx?

⇒ JSX is a syntax extension for JavaScript that lets you write HTML-like markup inside a JavaScript file keeping rendering logic and content in the same place

Q2 What are the superpowers / benefits of JSX?

⇒ 1) JSX prevents from cross-site scripting
2) JSX makes it easier to write code as we are no longer creating elements using React.createElement
3) Makes code simple & elegant
4) Show more useful error and warning message
5) JSX prevents from code injections (attacks)

Q3 What is the Role of 'type' attribute in script tag ? What options can I use there?

⇒ The type attribute specifies the type of the script. The type of the script. The type attribute identifies the content between the <script> and </script> tag.
It specifies the media type of the script. Media type indicates the format and nature of file.
By default is "application/javascript"

Script types

text/javascript
image/apng
image/avif
image/gif
image/jpeg
image/png
image/svg+xml
image/webg

Audio & video files

multipart/form-data

Q4 What is Babel?

⇒ Babel is JavaScript compiler which is used to convert ECMAScript 2015+ code into a backward compatible version of Js in current and older browsers

Powers of Babel:-

1) Transform syntax
2) Source code transformation
3) Polyfill

Q5 React without JsX

⇒ JSX is not required for using React. Each JSX element is just syntactic sugar for calling React.createElement(component, props, ...children). So anything you can do with Jsx can also be done with just plain JavaScript

Q6 What is a React Component?

⇒ Components let you split the UI into independent, reusable pieces and think about each piece in isolation. Components are like javascript functions.