npm – used to manage JavaScript packages but it is not abbreviated as node package manager.

Standard repository where many packages hosted

npm is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well.

Package.json is a configurstion file for npm

A bundler in react. js is a tool which allows to package code into a single file or bundle. Due to this size of code will reduce and so performance of application will improve. Common bundlers used with React are Webpack and Parcel.

There are two type of dependencies

* Dev dependemcy – required in development phase
* Normal dependency – used in production also

Parcel is done in development phase

So we use npm install -D parcel where D denotes development

Package-lock.json keeps track on versions of all the dependencies that are being installed

“It works on my local but not working on my production” – to avoid that package-lock.json keeps a hash to verify whatever there is on my machine is the same version when deployed to the production

node\_modules is the database which contains the actual data of the dependencies - contains all the code fetched from npm

Our project needs parcel depencency, parcel as a project has its own dependencies and those dependencies has its own dependencies , this is known as transitive dependencies

Parcel is a beast, it does lot of thing for our project.

If I have package.json and package-lock.json , I can recreate all the node\_modules whenever required

Just give npm install -it will install all the required dependencies and recreates the node\_modules

The command use to execute packages

npx parcel index.html

this will host the html page to the local server localhost:1234 – which hosts a development buid to local host1234

One way to add react to our app is by using cdn script commands

*<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>*

*But using cdn links is not a good way*

Another way is by using npm

npm install react or npm i react

// here we didn’t use -D because we install it as normal dependency

npm install react-dom or npm I react-dom

// React.createElement() is used to create a React element

// React.createElement() => returns a React element as an object => once rendered, it is converted to HTML by the browser

*const heading = React.createElement('h1',*

*{ id: "heading", xyz: "abc" }, // attributes*

*'Hello World from React!');*

*console.log(heading); //returns object*

*const root = ReactDOM.createRoot(document.getElementById('root'));*

## Parcel a powerful dependency

* It creates a dev build
* It hosts on local server
* HMR – Hot Modul Replacement is performed when changes are made
* It uses File watching algorithms to display the changes immediately once the changes are made to code. These algorithms are written in c++;
* It do caching - which results in faster builds
* Image Optimization
* Bundling
* Compressing files
* Consistent Hashing
* Code Splitting
* Differential Building – it helps to support old browsers
* Diagnostic
* Error Handling
* Https
* Tree Shaking – removes unused code
* Different dev and prod bundles

By adding, in package.json

"scripts": {

    "start": "parcel index.html",

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

We can easily start or build our app

* To start npm start / npm run start
* To build npm run build / here npm build wont work

// JSX - JavaScript XML

// JSX is a syntax extension for JavaScript, written to be used with React

// HTML like syntax in JavaScript

// JSX is not a necessity to use React, but it makes the code more readable

// JSX ultimately transpiles to React.createElement() calls which browser understands

// transpiling is done by parcel's babel

// // babel is a JavaScript compiler, used to convert code from one form to another

// JSX is not a string, it is compiled to JavaScript

// JSX => React.createElement() => React element (object) => HTML

*const jsxElement = <h1 id="heading">Hello World from JSX!</h1>; //*

*console.log(jsxElement); //returns object*

*const root2 = ReactDOM.createRoot(document.getElementById('root2'));*

*root2.render(jsxElement);*

to add a class to jsx use className attribute

*const jsxElement = <h1 id="heading" className="new" tabIndex="5">Hello World from JSX!</h1>;*

attributes in jsx are camelCase

// React Components

// Components are independent and reusable code blocks

// two types of components - functional and class based components

// class based components is old method

// functional components is new method

// react functional component is a normal JavaScript function that returns a React element

const HeadingComponent = () => {

    return <h1>Hello World from HeadingComponent!</h1>;

}

//also can be written as

const TitleComponent = () => <h1>Hello World I'm Title form Title functional component!😍</h1>;

const HeadingComponent2 = () => (

    <div id="heading">

        <TitleComponent/> // nested component as known as component composition

        <h1>Hello World from HeadingComponent!</h1>

    </div>

); // used commonly

Rendered using

const root3 = ReactDOM.createRoot(document.getElementById('root3'));

root3.render(<HeadingComponent2/>);

Javascript expression or statements can be injected into jsx code using {}

const element = <h1>I'm an element</h1>; //react object

 <div id="heading">

            <TitleComponent />  {/\*nested component as known as component composition\*/}

            {element} {/\*javascript expression\*/}

            <h1>Hello World from HeadingComponent!</h1>

        </div>