React is a js library to build user interfaces.

Inception:

Adding react :

1. Via CDN – it hosts react library which we can import/use in our code to use react. Add these cdn links in body tag at the end.

Cdn – **content delivery network** – a network of geographically distributed servers that work together to deliver content to user faster and more efficiently, by caching data on servers closer to users, reducing the distance that data needs to be travelled and improving load times.

Create ele in react takes 3 arguments – tag, {} (empty obj), “data/value”

const heading = React.createElement(“tag”,{},”data is entered here”);

created an ele, base react work, so **React**.**createElement**(“tag”,{},”content”) is used

any attributes like id/class are taken inside{}

heading here is a js obj created by react but not h1 tag

this obj has props {} and “----” where {} is the attributes and “---” is the children.

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

creating a root and which connects browser and the dev data, so **ReactDOM** is used.

            root.render(heading);

react.createElement (by babel) 🡪 js obj 🡪 html ele

like how we append child tags to the root element we **render** the child created to the root elements using render()

render converts the js obj into html and returns the html elements.

An array is used to create sibling tags, either inside a tag or in the main root.

Rendering replaces the original content in root ele (ele where rendering is performed)

Igniting a react app:

Npm – anything but node package manager (it has many random abbreviations)

It is actually a package manager. It’s a stnd package for all repositories/ libraries/ utilities

using npm –

**Npm init** – asks few questions answer them and it will give u a **package.json** (package manager- a configuration of npm == dependencies) file. This file has the info of the details answered

The imp package is a **bundler** – it bundles all the code files to be able to pushed to production.

Webpack, parcel, v etc are a few examples for bundlers.

Dependencies: **dev dependencies** – used for development and **normal dep** – used for production

**Npm install -D parcel** -> install parcel as a dev dependency package and also adds **package-lock.json** file -> keeps exact track of version and keeps the record of the dependencies and node modules folder – it has all the packages in the npm (collection of dependencies ~ db of packages)

parcel": "^2.14.4" **^** -> updates the parcel version automatically when used later

integrity has a “sha512----” it looks that the same version of the react app is deployed for the production. So that the app that works on local also works in the external server.

Transitive dependency – dependency depending on other dependency and it again depends on the other dependencies. That’s y node modules downloads many dependency package folders.

Package and package-lock should be pushed to git/ for production but node modules is not required as with the p&p-l we can recreate the node modules but p pl have the main info about the dependencies so it is very imp to have them for production.

Therefore node modules can be mode to a new file .gitignore (mention .\node\_modules) so that git will ignore that file.

Npx parcel index.html -> runs the react app in localhost:1234

Npm -> for installing a package

Npx -> executing a package -> goes to the source and builds a dev for the app and hosts it in local host

Cdn is not recommended as It is available in npm packages(nide mods) which makes it easier to work

And have to change/update the url for future versions.

1. Using npm to get react –
2. **npm install react**
3. **npm i/install react-dom** in terminal
4. **import React from “react”;**
5. **import ReactDOM from “react-dom”;** in app.js
6. **type=”module”** in script tag in index.html – to specify that the script now is a react file.
7. **Npx parcel index.html** to run the app in localhost/browser or **npx parcel build index.html -** it will give *dist folder* that contains a minified index and a few other files and thus takes a lil longer time than nrml npx execution.
8. **Remove main: App.js** from *package.json* before running the file via npx

ReactDOM.createELement() doesn’t work, use **import {createElement} from “react-dom/client”**

and directly use **createElement(document.getElementbyId(“id”))**

**parcel** – advantages (read *parceljs.org*)

creates a developer build

hosts the app in local server

automatically refresher the browser as soon as the file is save 🡪 **HMR** -> **Hot Module Replacement (hot reloading)**

File Watching Algorithm – written in c++

Reliable data **Caches** -> so builds faster

Image optimization

**Minfication** of file -> bundling => optimization

**Compressing** -> removes white spaces and reduces the size of file

**content hashing** - a tech used (in distributing systems) to distribute data across nodes while minimizing the amt of data that needs to moved when servers are added or removed.

Code splitting

Differential bundling – to support older/ different browsers

Diagnostic – beautifies the error execution

**Error handling** – gives errors in a better way

**hosts in https** aswell – npx parcel index.html --https

**tree shaking algorithm** – removes unused code

diff dev and prod bundles -> **npx parcel index.html** and **npx parcel build index.html**

any code automatically generated while hosting need not be pushed to git. So can be included in gitignore

u can manipulate the app to work on diff browsers by adding “browserslist” : [ “ last n versions”, “last 10 Chrome versions” , …….];

foundation to react: (read *babeljs.io)*

scripts in package.json should be added start: index.html and build: parcel build index.html so that next time u don’t need to write npx parcel index.html instead u can write **npm start** or **npm run start** or to run in build mode **npm run build.**

Creating a react app in the core way is more complex. So jsx is used.

In jsx we merge js and html. Jsx is only a syntax similar to html or xml

Js engine understands es6. So html tags written in jsx is not understood by the browser. But the code is executed bcoz parcel, the bundler will **transpile** the code that the browser/react can understand.

Parcel itself doesn’t do but it gives this transpilation task to **babel** (converts jsx to code that react will understand).

Jsx 🡪 react.createElement (by babel) 🡪 js obj 🡪 html ele

Attributes in jsx are given in camelCase.

The jsx code should be wrapped in () if >1 line is written

React Components – components are the building blocks of react.

2 types of comp – **class** based comps and **functional based comps** uses js class and funcs respectively.

React functional comp – PascalCase is used to name comps. - A js func that returns a jsx.

Comp composition – rendering a comp inside another comp

Any js code can be written in react comp using {} var/ ele can be adding in the same way

This can be included inside a comp {var} {23+35}

Jsx escape the malicious data bcoz of {}. It helps in cross siting scripting. {} Doesn’t allow api attackers to take charge over your browser.

diff ways to call a comp inside another comp :

        {HeadingComp()}

        <HeadingComp/>

        <HeadingComp></HeadingComp>

React code is readable as it is written in jsx