# Episode 1 and 2 (igniting our APP)

What is CDN?  
 what is croosorigin?

 <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

What is React.createElement ?

What is ReactDOM.createRoot()?  
  
  
 const heading =  React.createElement("h1", {} , "Hello World from React");

In this empty object {} we give attribute like id “ ritik” so it will be inside that

React.createElement("h1", {id=”ritik”} , "Hello World from React");  
so it will look something like this in html

<h1 id = “ritik”> Hello world From React <h1> so we give this attribute in this object

Q. what is npm?

Npm is a package manger it has all the package we need for our Project

I ran some command on my folder terminal in vs code

Npm init

Descriptin: can fill anything

Test: jest

After that I got a JSON package in my folder

Package. JSON file is basically a configuration to my npm

For example I am building a project for project I need some packages the version of that package name of it will be taken care by the package.json

Its also called dependencies like our project is depend on this packages

So I am using parcel package here these package will wrap up my html css jaavascript in one bundle and will push to the production code

We can install that dependencies through npm and terminal commands

npm install -D , what is the -D

there are basically two types of dependencies

1. Dev dependencies

2. Normal dependencies

-D is telling we need dev dependencies

"parcel": "^2.14.4"

  This upside arrow is caret and it will auto update the new version it is for minior update 2.14.5

If we put tilde ~ then it will give a major update 3.14.4 something if its available

Package.json – keeps track or configure that what version of package have installed in our system so tomorrow if there is new version it will update acc to the caret or tilde  
in package.json it can change to 2.14.4 to => 2.14.5 ,,,,,,, or 3.14.7

Package-lock.json – will keep the track of the exact version that is being installed in the system , but it will still keep the exact version and lock that were installed but it will keep

Track the exact 2.14.4 in lock

In package-lock.json there is a file name as integrity which has (SHA-512 or SHA-1) hash thing which

Will ensure that whatever version u have worked and created will be deployed to same version in production – or u have heard the most famous phrase its working on my local machine but not on the production ( its working on my computer but not on other ) to ensure that thing , this hash is very important file

It keep tracks of all the version all of the dependencies

That is the main difference between package.json and package-lock.json

When we created those package also there is another file is created which is node module

The node module contains all the code that is being fetch from npm

---when we did npm.parcel what happened – it took some time that time it was fetch all the code from parcel and putting into the node module ,,,,,you can also see in node module there is a parcel folder ------that is a actual parcel code that is beinf fetched

If I simply explain that package.json contains configuration of package

And node module contains actual data of that package that our Project actual needs

Now there is a question I just installed package so why there so many other packages in nord\_modules folder --- answer is – our project has dependencies parcel and parcel as a project also its dependencies and thoses dependencies has their own dependencies

This is known as transative dependencies , that’s why parcel install other libraries

None of them is garbage if u think , parcel needs all of that

Q. now with parcel we have other packages have their own seperate package.json configuration , for browser lit , for babel and if you open this package you will see

They have their own dev dependencies and normal dependencies

Like we installed parcel for our project dependencies the parcel also dependencis on other packages other respositery and it is maintaining its own package.json

NODE MODULES is a collection of dependencies  
  
now do we want to upload that mich big node module into our git hub repo its 317mb big

Do we want o push this big in our production the answer is NO  
so what will we do we put this in a newfile named as .gitiginore it will not push this file into git hub  
  
but do we want to upload the package.json, package-lock.json on our git hub the answer is yes

But why don’t the node module because the package.json and package-lock.json has the configuration of the those version so we can re-generate my node\_modules

Even though I delete my node\_module , as long as I have my package-lock.json and package.json I can still re-generate

Lets delete our node module and re-generate

After that in the terminal type npm install

That’s why its not required to push the node module in the git hub …

Whatever u can regernate don’t put it on git

now we are going to ignite our App for that we are going to

we will use parcel to ignite

we will type this command npx parcel index.html (index.html is my source file )

before we dd--npm--command which means installing the package

now we are doing npx –command which means executing our package  
  
Q. before how did we inject the react in our system through using CDN links

<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 this is not a good practice because we have to fetch from networks

If we already have in our node modules in our machine it makes things much easy

In this package is using react@ what if tomorrow a new react came in so we have to change this link again and again what in other npm case its easy to manage all our dependencies

Now we will install react package in our module the same way through npm install react

But here I did not do npm install -D parcel , I don’t put any -D here  
  
after installing we have react folder inside our node module

Now we will install React DOM for that command is – npm install react-dom

Some people also write npm i react-dom i means insatll

Now we no longer neend cdn so we can remove

So now if want to see if our server is working or not for that we have to execute our file

The command is npx parcel index.html   
  
if run this on web we will get an error that react is not defined coz earlier we were using the cdn link for react but now we don’t have any link now we have nodee module

React is in node module so weill using a keyword import

Import React from “ react”; it means I am importing react from react folder in node module  
now I will need React dom it will com from node module

Import ReactDom from “react-dom”;

Now will again face an error that error will perfectly display in the console

Problem is <script src=”our file”> <script>

This is more normal javascript browser only understands this as a normal script

But now our javascript is not normal noramal script don’t have import

Now we have to tell browser that its not a normal javascript file it’s a module

For that we will do < script type=”module”> </script>

After that its showing an error again in which we have to import dom from

Import ReacDom from react-dom/client;  
After if change anything in java Script it will reload and update in the webpage very fast

That is the power of Parcel package  
**# parcel**

- Dev build

- local server // hosting our app into the server

- HMR = Hot Module Replacement //  its reloading into web faster

- file watching algorithim -- written in c++

Caching – faster build

Whenever I am saving in my file we can in terminal bulit is take time like 3ms save it again 8ms save it 5ms how?

Because of caching , when we installed parcel package it went into our project and took some space and created a parecel cache folder

Image Optimization – the Parcel also do image optimization the most hard part is loading your image in web – browser parcel helps in loading that easily

That was development build

If we do production build – the parcel will minimize or minification of our file also

Bundling--- parcel will also bundle our file  
Compressing – it will also compress your code it will minified and pack all of your javscript files and ship it onto production

Consistent hashing -   
code splitting -  
Differntial **bundling- that our web app will run on older version of browser too it creaete different types or version of bundling  
  
Diagnostic-**

**Error Handling-  
  
parcel is also displaying errors on the screen better error handling**

**HTTPS- it is also giving us a https so we can host our app some app will only work in https**

**Tree shaking – remove unused code ( suppose If you have 100 function in your file and you are only using the 5 of so tree shaking will remove and make more optimize  
  
- Differnt dev and production bundle—production build will take little bit more time than building dev bundle , some optimization is more in production build than the dev bulid  
  
now we will see how to create Production bulid\_\_\_---- index.html is my file name  
command line is – npx parcel build index.html – this line means entery point of app  
error will pop up  
remove the main from package. Jason cause it will give u an error**

**when we do npx parcel index.html it will create our dev build of our file and host on the server 1234 example  
and it will put those dev build into “dist” folder which you will see below node module folder right in this folder is your dev build files is stored so the things we are seeing in the web page or hosted server is coming from dist folder and whenever we are saving the anything in file it is using dist folder and cache folder with help of HMR to showcase in web server**

**Production build**

**but when we will do npx parcel build index.html so when we will build production build**

**it will come inside “dist” folder  
  
now when we were doing developer build it was taking time in milliseconds**

**now in production build they are taking time in seconds generally more time**

**if we will open our production build file example html file from diat folder**

**you will see the code has compressed in one line removed unnecessary spaces and unused code**

**for example – if we have 20 files it will compess all the 20 files into 2,3 files these 2,3 files will contain all the code we will write**

**and that production build is very highly optimized which you for app production and serve to user, it will be fast , performant and optimized**

**Cache and “dist” folder are temporary folder if we delete and put dev build from npx parcel index.html**

**It will recreate this folder , even Node\_module can be regernate so things we can regernate we will not push to the git hub**

**Now lets make our app compatiable for older version of browser for that we need browser\_list package from node\_module**

**To that we will add in package.json file a browser list**

**And we will give an array if I have to just support chrome last 2 version I will do**

**“browserlist”[**

**“last 2 chrome version”**

**]**

**I can tell my app that it should be working till last 2 version I can also write last 10 version**

**I can also write last 2 Firefox version**

**"browserlist":[**

**"last 2 chrome version ",**

**" last 2 Firefox version"**

**]**

**Now it will work on both chrome and firefox , now there is a question arises**

**Will it work only for chrome and firefox the answer is no**

**Defining like this means it will surely work on this last chrome and firefox version**

**It might or might not work for other browser version but it will surley work on this mentioned version of browser**

**Question arises how do I know what to write inside the browserlist just go to browserlist website**

**There more old verion we will add mode bundle it will create and more code it will write**

**And our app will become slow so better approach to cover till 90% of version or u can do last 10 version it will cover till 90%  
  
NOW finally we have created a create-React-app   
 when we create a create- react-app this is inside the app reside  
  
Episode 2 Ends here**

# Episode -3-LAYING THE FOUNDATION

Now we will ignite our app npx parcel index.html , now we will create a script so we don’t have to give this command for build

So it’s a npm script so we have to create this in our in package.json there is a script {}

There we can create different types of script like for a dev build for production build different kinds of build  
lets create for dev build   
"scripts": {

    "start": "parcel index.html" ,// like there is a start we can name anything here

“build” : “ parcel build index.html”

}

Its industry standerd that we have to use script for start our app

Suppose if we are in company and we don’t know the command for starting the app

We have to just go to the their package.json file search for script tag there will be a our start command

No we wont be doing --- command npx parcel index.html after writing the script

We will do just--- npm run start ---coz we named start in the script

And for build a have also put a script in package,json

I will do--- npm run build  
  
npm also give us a shortcut instead of doing npm run start we can also do

npm start // and its only for start it wont work on build if u try npm build it wont work

start is a keyword reserved by npm from now on we will use npm start bye bye npx parcel command

npm start = npm run start

npm run start behind the scene execute the package parcel index.html behind the scene because we have configured this on our package.json

NOW in episode one we were creating some react object by writing something like this

const parent = React.createElement(

  "div",

  { id: "parent" },

  React.createElement(

    "div",

    { id: "child" },

    [

      React.createElement("h1", {key: "h1-1"}, "Namaste React🚀"),

      React.createElement("h2", {key: "h1-2"}, "I am the best developer"),

    ]

  ),

  React.createElement(

    "div",

    { id: "child2" },

    [

      React.createElement("h1", {key: "h1-1"}, "I am H3 tag"),

      React.createElement("h2", {key: "h1-2"}, "I am H4 tag"),

    ]

  )

);

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

root.render(parent);

but its too complex just to write an html tag through react so facebook developer created

JSX is a JavaScript syntax to create React Element .

People think JSX is a part of React but not JSX is different React is different , we can make big app without even using JSX but JSX make our life easy  
  
JSX is convactional where we merge our HTML and JavaScript together

Don’t get confuse JSX is nor HTML ,, its HTML like syntax

const JsxHeading =<h1 id=”heading”>Hello World!</h1>

<h1>Hello World!</h1>

                    // this peice of code is JSX = React Element

JSX is a syntax The upper code is React element

This JSX is not a javascipt so how this is running in a js engine because of the parcel

Even before the the JSX code goes into browser the parcel is trans piled (convert) it befoe it goes to JS engine then jS engine receive code that browser can understand  
  
the parcel is a package manger in this parcel all of this work is done by babel

JSX is converting into React elemant its converting into JS element and convert into html

JSX => React.createElemnt =>ReactElemnt-JS. object => HTML(render)

What is converting behind the scene the , thing is babel

Babel is a javascript compiler  
there is a little difference in jsx

In html we write like this

<div class =“Rtik”> helloe</div>

In jsx const heading = <div className =”Ritik”> hello </div>

Its like if we writing anything two word together first word first letter will be small second word first letter will be big example – className  
  
new thing if we are writing jsx code in one line is is valid code

<h1 className="Ritik">Hello</h1>

But if we have to break the Lines then w have to add brackets around it coz babel need to understand where is JSX starting point and ending point

const JsxHeading = (<h1 className="Ritik">Hello World!🚀</h1>);