Episode 2: Igniting our App

* Making production ready create-react-app from scratch.

- some commands like npx create-react-app gives us production ready setup.

- we can also make our production ready setup without using existing ones like npx create-react-app.

[] When we have to build fast production ready app react itself can't do it

There are lot's of other packages required to build large scale applications

There is lot of other things need to make our app fast.

• How to get those packages in our app -

↳ Using npm

• It manages the packages

• It does not stand for node package manager

• It is the standard repository for all the packages.

- Any package you need to include in project you can use npm.

- All the packages are hosted over npm.

↳ Add npm in our project

- npm init : create package.json (with some info filled by us).

- package.json :

  - configuration for our npm
  - Packages is also known as dependancies
  - Packages we are going to use in our project, the version of it will take care by npm in this package.json file

↳ Add packages in project using npm.

- Installing dependancies :

Note: most important package in our project is bundler.

- Bundler :

  - when we have HTML, CSS, JS files in project the whole code needs to be bundle togethen, minified, cached, compressed, cleaned together before send to production.

- Bundler helps to do all this things.
- for e.g, webpack, parcel, vite etc
- what is the job of bundler?
  It bundles or packages our app to shift to the production.

New to know:
create-react-app → It uses webpack Bundler and bebble

• Parcel Bundler (package)

- parcel basically comes as node package
- parcel is easy to configure
- To install any package into project we always use common command,
  npm install (package_name)
- Install parcel
  npm install parcel
  or
  npm install -D parcel
- There are two types of dependancies

  1) Dev Dependancies -
  Required for development phase on when we are developing our app.

apsara

Date

2] Normal Dependancies -
Used in production also.

- In package.json, there is something

```
"devDependancies": {
  "parcel" : "^2.8.3";
}
```

^ caret — parcel auto upgrade to new minor upgrade

2.8.3 — It says parcel 2.8.3 version is install.

Note: There is also something like ~ (Tilde) we can use over ^(Caret).
It is used for major version auto upgrade

- package-lock.json

  - keeps the track of exact version of packages or dependancies.

  - where, package.json have approx version of packages.

Note: "integrity" : "sha512-5rMB....."
In package-lock.json.

package-lock.json keeps the hash to verify that whatever in my machine is the same version which is being deployed on to the production

- node_modules folder

  - contains all the code we fetch from the npm.

  - It is like database for dependancies we need in our project

  - Transistive Dependancies :
    Our project dependant on parcel, parcel itself has it own dependancies, that dependancies also have their own dependancies.
    This is known as Transistive dependancies

Note: Parcel Itself can be dependant on lot of things. It can't do all this thing on it own. It needs help of a lot of other packages.

Note: Every dependancy or project will have its own package.json

  - node_modules is a collection of dependancies.

  - Should we put this node_module to production or github?

    No need to put node_modules to product or github.

- gitignore file - If you want some files to not go into production just put it inside git ignore

- should we put package.json & package-lock.json file?

Yes, need to put on github.

- Why we don't need to put node_modules on github?

1. package.json and package-lock.json maintains a note of all dependancies that our project needs.

2. with the help of package and package-lock.json we can regenerate node_modules

3. To regenrate just run npm install command in the repository.

Note: whatever we can regenerate don't put it on git

* Building our app using parcel

↳ npx parcel index.html
(source file)

- It will create localhost server to run our app or host our app.

- To install package : npm
  To execute package : npx

- Here parcel package is executed with npx with main source file ie index.html

↳ To get react into our app via npm
(This is the 2nd way to get react)

- React is javascript library or package that is hosted on npm also.

- why do we install react in node_modules

1. Easy access of react compare to CDN Links.
2. Because we don't have to make request call for react to another domain.
3. React is already present as an package into our app or project.

4. If there is another version came, then we have to change manually CDN Links.

- Install react and react-dom
  npm i react
  npm i react-dom

↳ Using react & react-dom in project

- we use import keyword to use react & react-dom in project

  eg : import React from 'react';
  Import ReactDOM from 'react-dom';

- How we are using react & react-dom from the package we are Installed in the node_modules

Error - Browser's script can't have import or exports

1. In our index.html we are injecting 'App.js' file

2. Index.html file treats this script as a browser's script or normal JS file

3. Normal JS file does not have import

4. Need to tell html file it is module not normal JS file.

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

- 'react-dom/client' : new reactDOM file
- Parcel refresh the browser automatically when we save file

* What Parcel DO ?

1. Create DevBuild
2. Create Local Server
3. Auto refresh the page
   - HMR (Hot Module Replacment)
   - It is done using file watching algorithm. and It is written in c++.
4. Caching
   - Gives faster build experience
   - Caching In .parcel-cache
5. Image Optimization
   - Expensive thing In browser to load Images on page
6. When we Build production app parcel do
   - minification of code also
   - Bundle the code
   - Compress file
7. Consistent hashing
8. Code splitting
9. differential Bundling (Our app run on older & new browsers as well)

10. Error Handling
11. Diagnostic
12. Support HTTPs also.
13. Tree shaking : remove unused code
14. Different dev and prod bundles

Parcel Documentation : parceljs.org.

↳ Creating production build

- npx parcel build index.html

Error: Try changing the file extension of 'main' in package.json

Remove "main" : "App.js" from package.json

- when above command, builds up production ready app, where does that builds goes up?

1. After all the minification, bundling the files goes into 'dist folder.

2. The code we look on the page is coming from 'dist folder. Code get compressed.

- Production build is very highly optimised build which is used to serve on production for the users.

- .parcel-cache
  dist
  Put in .gitignore

↳ making our app compatible for older versions of the browser

- use browserslist
- node_module their is browserslist (npm package)
- we can tell browserslist which browser is compatible to run our app
- give some configuration,

1. In package.json, add

```
"browserslist": [
"last 2 versions",
"last 2 chrome versions"
];
```