EPISODE-02 | Igniting our App

npm:

Everyone's npm means node package manager, but it is not node package manager, but it is everything.

⇒ npm behind seens looks as a node package manager but it didn't stand for it.

To get npm.

```
npm init
```

⇒ After 'npm init' process we get package.json

package.json is configuration for npm.

⇒ Bundle

The whole code to be, minified, compressed, cleaned, so that Bundles will do this.

1. Web pack
2. Vite
3. parcel.

What is work of Bundler?

Bundles basically bundles your app ~~and~~, so it can be shipped it to production.

parcel is best package

⇒ npm install -D parcel

what is -D ?

A: These are ② types dependencies that we can install

① dev dependencies

② ~~Production dependencies~~ (or) Normal depencies.

⇒ After installation we get parcel : "^2.8.3" in package.json

↓

what is these means

⇒ '^' and '~' caarat & tilde these are two which we can infront of some version.

→ If we have ^ in front of version, it will install minor version automatically, it updated it self if there is small change in version.

'^2.8.3' ⟶ '2.8.4

→ If we have ~ infront of version, it will install major version automatically, it updated upto a big changes in version.

'~2.8.3' ⟶ '~3.0.0'

⇒ package.json → contains the approximate version of new version.

⇒ package-lock.json → contains their exact old version of all dependencies.

***

In package-lock.json it has a integrity, it is most imp because, we hear that it works in local but it doesn't ~~of~~ work in production. To avoid these the package-lock.json plays a important role [integrety].

Node_modules:-

Node_modules are very heavy

→ It fetch all the dependencies into system, it kind of a data base where all dependencies exist.

⇒ pasrel have dependencies, those dependencies have their dependencies know as "Transetive dependencies"

• gitignore:-

The files which we don't want to push for production and also github, we placed it files in it.

These files can be regenerate easily

•gitignore

/node_modules

igniting our app:

```
npx parcel index.html
```

⇒ npm is calling where npx means execute the package.

Upto now we are Using CDN Links for reacts, it not a good one, so by these we get slow type process.

⇒ Bcz link should be called from CDN and again react have to call it it time taking process,

→ If react is in our modules, it is easy process and we can get it very fastly.

```
npm install react
```

```
npm install react-dom
```

we can use i (or) install

⇒ After this if we start server, we get o/p as well as an errors,

bcz we deleted our CDN Links so .js didn't from where the react is coming, for that we use import.

import React from "react"; → present in node_modules

import ReactDOM from "react-dom";

=> We are importing from node modules to our interface

=> We get error as "which looks very beautiful" means a beautiful error.

→ Because Browser script cannot have imports and Exports

→ The solution for these we have to keep type="module" for script which we are linked.

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

What is parcel doing for you?

- Dev Build → it created Dev build
- Local server " " local server.
- parcel is refreshing for you called 'HMR'.

HMR - Hot Module Replacement

↳ it has file watching Algorithm - written C++

→ caching - faster Builds

→ Imagine optimization

→ Bundle

→ compress

→ Minification.

→ Differential Bundling - support older Browsers

→ Tree shaking - remove Unused code.

To create prod build [production build]

```
npx parcel build index.html
```

⇒ We get error bez, in package.json we have "main": "App.js", so index.html and App.js may involve into conflict and throw an error.
We have to remove that line "main": "App.js"

***

The O/p does not come from the our code, the output will come from "dist" it contain a minified, compressed, optimized code of our code

⇒ Add,

```
"browserslist": [
    "last 2 chrome version",
    "last 2 firefox version",
]
```

in package.json

⇒ It works upto 2 old version of chrome and firefox versions.