Date: 15 Aug 24

EPISODE-2

→ Optimization → Clean console → "Bundle"!

↳ We need to "minify", remove unwanted comments and optimize our code to make it production ready.

→ When we use 'Create React app', we already get a small ready to production react app & have npm in it already.

→ There are a lot of different packages which comes with react which makes our app fast, not just react alone.

Que: What is npm?

→ Npm doesn't have a full form. {It doesn't stand for node package manager}

→ Basically, npm manages the packages.

→ Npm is the standard repository for all the packages (It's the biggest package manager).

* packages — are pre written ~~co~~ tested code for a specific task into our projects. It is a collection of million of packages.

↳ Millions of package ensist and there is at least one package for our task.

→ To use npm into our project.

• We use npm init : It is like a box for us into which we can arrange all our packages.

→ It basically creates a package.json file where we get our packages informations. (like name, version etc.) organized at a single place.

→ Package.json — It is configuration for npm file.

the way in which parts of something are arranged.

↳ In package.json we can see all the installed packages and their versions in dependencies section.

↳ When we write npm init, it asks us some questions like package name, version etc. So it is also like we are creating a new package.

→ Packages & Dependencies are same.

→ In React, to get external functionalities, we use "BUNDLERS".

Que: What is Bundler?

Ans- It is a node package which basically bundles our project and make it production ready.

"BUNDLERS" :-

a) Webpack is a bundler
b) vite
c) parcel

} These are alternatives

↳ Job of a bundler -

Bundler is basically a package which bundles our app properly. So that it can be shift to production.

In create-react-app, the bundler used is "webpack"

→ If we want to use a package in our code, we have to use a 'package-manager'.

→ We use a 'package-manager' known as npm or yarn

→ We use npm because we want a lot of packages in our project

[npm init -y] → This command will skip lot of options. {so we will not use it}

[npm install -D parcel] → parcel is one of the dependency

Dev dependency - because we want it in our developer machine.

→ There are two type of dependency which a app can have

① Dev Dependency — It is required in development phase.

② Normal Dependency — They can also be used in production ~~also~~.

→ As bundler combines different files & minifies it, and we do it during production that's why we are installing it as an development dependency. {Then, we will get package-lock.json}

→ [Caret & ~~Tilde~~ Tilde Sign]

Caret (^) :- Our project will automatically update if we use caret sign [i.e minor version upgrade]

e.g. '^1.4.2' : Allows updates to version '1.4.3' through '1.9.x' but not '2.0.0'

```
"devDependencies": {
    parcel : "^1.4.2"
}
```

Major Minor Patch

Tilde (~) :- It allows you to automatically update to any patch version within the same minor version.

e.g. '~1.4.2' : Allows updates to patch versions '1.4.3' through '1.4.x' but not '1.5.0'.

→ '~1.4.2' : Updates to minor & patch versions like '1.5.0' or '1.6.0' but not '2.0.0'.

→ If we does not have any sign nor tilde (~) nor caret (^), if any updates of that dependencies comes it will not get installed.

→ It is recommended to have caret (^) & not tilde (~) because with major updates a lot of things might break into our app. So always have caret (^) because minor update will not have changes much more than previous one.

Que :- Difference b/w package-lock.json & package.json ?

↳ Package-lock.json :- It keeps the track / record of exact version which is installed

↳ It keeps the record of exact version of any package that is being installed into our project.

↳ Package.json :- It keeps an approx version of package

e.g. Suppose in package.json we have ^1.8.0 & new update comes as 1.8.5 then our package.json will remain same & 1.8.5 will be reflected into package-lock.json.

* You may have faced a situation where you say that my project is working on my localhost/my machine but not works in production. Why?

→ It can happen because on our local host & on production we have different versions of packages/dependencies.

→ To solve this package-lock.json have a hash ~~code~~ known as integrity.

Que: What is integrity inside package-lock.json?

The 'integrity' field in 'package-lock.json' is a hash that ensures the package installed on your machine matches exactly with the version being deployed on the production.

→ It helps verify that the package hasn't been altered or corrupted.

- **Node-modules:-** It is the database or we can say that they are the files of actual codes of all the dependencies or packages which we are using in our project.

- **Transitive Dependency:-** As we install parcel, in node modules we can see a file/folder named as parcel. It will have all the code of parcel. But along with it we can see alot of additional folder they are the dependencies of parcel.

→ Parcel can have its own dependencies & those dependencies can have their own dependencies this is known as transitive dependency.

Que: How npm will know that parcel have its own dependencies $ it should be installed also.

Every package which we install have their own package.json and in their package.json their own dependencies/packages are listed. Through which npm will know about the other dependencies $ it will automatically install all of them.

Que: We don't put 'node-modules' ~~inside~~ into git, why?

"No" we should not put 'node-modules' into git ~~because~~ as the filesize is very large.

→ Best practice is to put the node-modules files inside git-ignore.

Reason: — Because, our package-lock.json file have sufficient information to recreate node-modules.

→ Package-lock.json file keep $ maintain the version of everything in node-modules.

→ 'node-modules' in our machine can be ~~generated~~ re-generated in server using the package-lock.json file.

Que: Should we put package.json $ package-lock.json into git? Why?

"YES" Because with the help of package.json $ package-lock.json we can create our node modules.

[By just writing "npm install"] into our terminal.

Git: A version control system which keeps a track of all the changes which we have made into our project.

Git Hub:- A web based hosting services for git repositories.

> **Note:-** All things which we can re-generate do not upload it into git.

Previously we used CDN links to get react into our app. This is not a good way because through CDN we are making another n/w connection with those cdn links. Instead we can have it in our node-modules by writing

`npm install React` & `npm install React-dom`

- npm i is a short form of npm install.

→ Due to version changes in react, we need to update the CDN links. That's why we use npm install React, npm install React-dom

To ignite our app we use

`npx parcel index.html`

- npx → It means executing the package
- parcel → package name
- index.html → source file of our project

Que: Diff b/w npm & npx ?

npm: for installing package we use npm.

npx: for executing package we use npx.

After writing command when we Click 'Enter'

↳ Then a mini server is created for us
- like localhost 1234

- Parcel given a server to us.
- "Parcel" ignited our app.

↳ As we removed CDN links, ~~we don't have~~ and install react $ react-dom, but still our browser will not understand what is react $ react-dom.

↳ So, we want to import it into our app. For that we use the keyword "import".

Example:—

```
const heading = React.createElement("h1", {}, "Hi");
const root = ReactDOM.createRoot(docm----);
```

Our browser will not understand these keywords and show error. So we import it i.e.

```
import React from "react";
import ReactDOM from "react-dom/client";
```

This is the react which is inside node-modules

Similarly this react dom is inside node-modules.

- As we got an error we need to specify to the browser that 'we are not using a normal script tag, but a module'.

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

Note:- We cannot import & export scripts inside a tag.
Modules can be import & export.

* Hot Module Reload (HMR)

→ means that parcel will keep a track of all the files which you've updating.

Que:
↳ How HMR works?

- There is File watching Algorithm (written in C++). It keeps track of all the files which are changing in realtime & it tells the server to reload.

↳ These all are done by "PARCEL".

* There will be a folder called .parcel-cache which will be there automatically.

- In our project, parcel needs some space. So, it creates .parcel-cache. { we push it inside git.ignore becoz it can be regenerated }

* 'dist' folder keeps the files minified for us.

↳ When we run command

```
npx parcel index.html
```

This will creates a faster development version of our project & serves it on the server.

→ When I tell parcel to make a production build:

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

It creates a lot of things minify your file. And "parcel will build

all the production files to the dist folder".

Que: What takes a lot of time to load in a website?

- Media - Images
- Parcel does image optimization also.

↳ Parcel also does Caching while devlopment.

* Parcel also takes care of your olden version of browsers.

'Compatible with olden version of browsers'

- Sometimes we need to test our app on https rather than http bcoz something only works on https.

↳ Parcel gives us a functionality that we can just build our app on https: on dev machine.

```
npx parcel index.html --https.
```

* Parcel features in a glance :-

→ HMR - Hot Module Reloading
→ File watching Algorithm (in C++)
→ Bundling
→ Minify code
→ Cleaning our code
→ Dev & production build
→ Super fast build algorithm.
→ Image Optimization.
→ Caching while devlopment
→ Compression
→ Compatible with olden browser version.
→ HTTPs on dev.
→ port number
→ Consistent Hashing Algorithm.
→ Zero Config.
→ Tree Shaking.

Que: How do I make our app compatible with older browsers?

— There is a package called 'browserslist' & parcel automatically gives it to us. Browserslist makes our code compatible for a lot of browsers.

go to 'browserslist.dev'

In package.json file, do:—

```
"browserslist": [
  "last-2 versions"
]
```

→ Support 74%

feeded with some configuration.

means parcel will make sure that my app works in last 2 versions of all the browsers available.

If you don't care about other browsers, except chrome:

```
"browserslist": [
  "last 2 Chrome versions"
]
```

→ Support 16%

↳ Tree Shaking

— It removes the unwanted code for us.

E.g:— Suppose your App is importing a library which has a lot of functions (say, 20 helper function). Then, all those 20 functions will come into your code. But in my App, I want to use only 1 or 2 out of it. Here parcel will ignore all the unused ~~app~~ code.

create-react-app: uses 'webpack' along with 'babel'.

* **Parse:-** Parsing means analyzing & converting a program into an internal format that a runtime environment can actually run, for example JS engine inside a browser.

↳ The browser parse HTML into a DOM tree.

↳ In script tag when we don't have async or defer attribute blocks rendering & pause parsing of HTML.

↳ **Rendering:-** It is the process of combining all our code according to some instruction and translate it into HTML code, that our browser understands and displays it into the screen.

↳ **Script Types in HTML:-**

① **Async:-** A boolean attribute which makes sure that the script will be downloaded in parallel to the parsing of the page and it will be executed as soon as it completes the download. It does not block HTML DOM construction during downloading process.

② **defer:-** Similar to async, only difference is that, it will be executed only after completing the parsing.

③ **blocking:-** This attribute indicates that certain operations should be blocked on the fetching of the script, The operations to be block can be mentioned as

```
<script blocking="render" async="abc.js"> </script>
```

"Module script defer by-default"