IGNITING OUR APP

1.What is `NPM`?

NPM stands for NOTHING.NPM is used to manage packages in the Project. When we're creating a Web App in React. We need some functionalities and libraries for the app to perform better and to do that we need Package.

A package is a module which have JavaScript files in it. When we need a package in our project, we just get it in our project using package manager. One type of package manager is npm. we can use npm in our project by using "npm init".

2.What is Parcel/Webpack? Why do we need it?

Parcel and webpack are bundlers. Bundlers are the main catalyst in react. On its own react cannot do all the functionalities to give an web application super powers. It needs someone or some libraries to help it to provide all the required functionalities to the web app. These functionalities are provided by something known as bundler. These are the dependencies on which react majorly depends on.

Create react app is built on webpack. In production app the bundler used is webpack and Babel. Parcel is a zero configuration bundler. Apart from parcel and webpack VITE is one more popular bundler. Compared to webpack and VITE we use parcel because it is zero configuration , that doesn't mean webpack and VITE are not zero configuration their configuration is reduced in recent times too.

Apart from this parcel create a server, it minified our project ,manages development and production build, it is super fast build algorithm, it does caching while development ETC. Parcel take cares of all the production and development differences.

3. What is .parcel-cache?

parcel along with many functionalities does caching too. Caching is a technique that helps us to store a copy of a given resource in to our browser and servers and it backs us when requested. Parcel does all the functionalities with the help of this parcel-cache. All the required files and dependencies are stored inside this parcel-cache.

To ignite our app we use **npx parcel index.html** (entry point of our app).

This creates a server and parcel has ignited our app, now it runs in our local host. after the command hit it takes some time to be back and in in this time parcel has created development build and it also created parcel-cache.

4.What is npx?

We first used npx here **npx parcel index.html.**

The above command is use it to ignite our app for development build environment. here npx states execute using npm.

5. - What is difference between `dependencies` vs `devDependencies`?

devdependencies are stated with -D. This means we’re installing parcel as development dependency.

Dependency means all the packages that our project needs. Our project depends on something and parcel is one of the dependencies.

If we want to get our project on development environment and not on production we use

-D or –save-dev. We use -D to install packages in dev Environment.

Ex:- npm install -D parcel (to get parcel into our app)

There will be some packages we need in our global environment. Then we will install them in our global environment without using `- d`.

6. What is Tree Shaking?

Tree shaking is the term commonly used in JavaScript context which is used to describe the removal of dead code.

In modern JavaScript applications we use module bundlers (webpack, roll up) to automatically remove dead code when bundling multiple JavaScript files into single file. This is important for preparing code that is production ready which means with clear structures and minimal file size.

7. What is Hot Module Replacement?

When we add something in our files without any key reload or refresh the page gets reloaded instantly. This concept is known as hot module replacement. This is also known as HMR. This is a thing parcel is doing. This means parcel keeps track of all the files which we are updating if we just change any in our HTML or CSS or JS. It automatically shows up in our web page.

How does HMR knows that we have changed something in our file?

There is something known as file watcher algorithm. Parcel is doing this for us. Parcel uses file watcher algorithm Which is very fast and keep a track of all the files which are changing real time and tells the server to reload. This file watcher algorithm is written in C ++.

8. List down your favourite 5 superpowers of Parcel and describe any 3 of them in your own words?

There are many functionalities or superpowers parcel possesses in it. Sum of its superpowers is-

- Tree Shaking
- Clean our code
- Image optimization
- Compression
- CACHING while development
- Bundling
- compatible with older browser versions
- Uses consistent hashing algorithm
- HTTPS on development
- Hot module replacement
- minification
- Super-fast building algorithm
- parcel manages port number

- Chunking

HTTPS on development

As we are using local host HTTP which is not encrypted and in the development stage we cannot use encrypted https because browsers complaint that they don't trust our host. Https is useful when we are testing something, and it takes more build time than HTTP and moreover Parcel gives us a functionality that we can build our app on https by

npx parcel index.html --https

Clean our Code

Cleaning our code or tree shaking is removing unwanted code like console logs while code build or in development phase which is easily handled by parcel itself. We can find zero console logs in **dist** JavaScript file in our project. Parcel just do all these to make our project more efficient and proficient. But in order to get zero logs in our project we need to install a plugin which helps as a support and that plugin/package comes from Babel .

Babel Plugin Transform remove console when we search this on Google we'll get the options from npm and babel websites. We can pick any and paste the command

npm install babel-plugin-transform-remove-console –save-dev/-D

In our terminal and that's it. Babel plugin is in our files but it's not yet ready to do the job. So, we've to configure the plugin with. babelrc file and paste the Recommended usage in the file. After this we can see that no console. Logs/errors/warn in our files. Its's based on our requirement we've to select the options of console's.

Ex;- **IN .babelrc paste**

//without options

```
{

"Plugins": ["transform-remove-console"]

}
```

//with options

```
{

"plugins": [ ["transform-remove-console", { "exclude" : [ "error", "warn"] } ] ]

}
```

Image Optimization

Most of the heavy thing on website happen because of media i.e images videos. So partial does image optimization and JS optimization. Parcel minifies it and compress uncompressed the files. The media which are in our project is taken care by parcel. with optimization the website takes, or application takes less time to display.

Zero Configuration

parcel offers fast performance utilising the multi core processing and requires zero configuration. parcel does many tasks for our development process like minifying our files so that their size will be reduced in order to make our application faster and easy to deploy.

Minification

minification improve loading time of the web page. It also decreases the size of your file and it can be used for any interpreted language. Ordering matters in case of bundling but it doesn't matter in case of minification. Hey for minification of a JavaScript file we do the following steps:

1. remove the white spaces from file

2. remove line breaks to shorten file size

3. remove final semicolon

4. renaming local variable name and providing it a short name

8. What is `.gitignore`? What should we add and not add into it?

The git ignore file specifies the files which git should ignore. These are untracked files which shouldn't be on git. Anything which we can regenerate on a server can be put in gitignore.

For example we don't push node_modules to git because we can generate node modules by doing npm install.

9. What is the difference between `package.json` and `package-lock.json`?

When we initialise our node application we will see 3 files installed that is node_modules, package.json, package-lock.json.

When we initialise npm in our app with **npm init** . we'll get package.json to our Project.

Package.json is a versioning file used to install multiple packages in our project. It contains basic information about the project.It is mandatory for every project. It records important metadata about the project. It contains information such as name,description,author,script,keywords,license and dependencies.

Package-lock.json is added in our project when we get parcel. We can get parcel with

npm install -D parcel

package-lock.json tells us exactly what version of library we're using on production.

package-lock.json is a very important file.It locks the version. We never have to keep it in .gitignore.It cause us many issues if we keep the file in .gitignore.

We’ve to put package-lock.json in our git project. Because Our Machine is a simulated machine and our project runs on a random server.

Package-lock.json maintains the exact version parcel. It maintain a `hash` of it as well. This “hash” ensures what exact version of parcel is running in our system is same on the production or not. It allows future developers to install the same dependencies in the project. It contains the name, dependencies, and locked version of the project.

If exactly same on the production as well it maintains the integrity. That’s why it has the hash.

10. - Why should I not modify `package-lock.json`?

Because it is the file that locks the version and stores a copy of all the dependencies. And with its hash property only we will know if the parcel version on production is exactly same with our build or not. It’s as important as our other files, that’s the reason we keep in our project git repo and not into .gitignore. It contains all the copies of our dependencies.

11. What is `node_modules` ? Is it a good idea to push that on git?

Node_modules are like a Database for NPM. When we install parcel, along with package-lock.json , node_modules are also installed in our project.

Yes, we didn’t installed them but they’re in our project file.Because whenever we install something(like react and react-dom) they get into node_modules and we use our required files from node_modules whenever we needed in our project. So, It’s like a Database for NPM.

Never push node_modules into the git repo. Node_modules are the heaviest thing with full of all the dependencies and functionalities files and getting them into our git is not good.

We can re-generate everything in node_modules .package.json has sufficient information to recreate node_modules.With package-lock.json and package.json we can generate them. With package-lock.json we can re-generate our node_modules exactly the same how much heavy they’re.

How can we do that?

As, node-modules keep a tract of all the dependencies (in it), we will be able to do all the stuff the same manner.

12. What is the `dist` folder?

`dist` folder is formed when we ignite our app(for dev build) with the command

npx parcel index.html

npx – execute using npm

index.html- entry pint to our app

The above creates a local server and dist folder along with it.

`dist` folder basically keeps the files minified for us.

`dist` is a development build.

For production build

npx parcel build index.html

When we say this, it creates a lot of things. Minifies the file and Parcel will build all the production files in the ``dist`` folder.

13. What is `browserlists`?

Browserslist helps you keep the right balance between browser compatibility and bundle size. With Browserslist, we'll cover wider audience and have smaller bundle size.

We can find target browsers automatically if we add the following to our **package.json**

"browserslist": ["defaults and supports es6-module", "maintained node versions"]

Defaults – if we're building a web app for the global audience

Node 18 – if we're building a node.js app, Ex:- server side rendering

For detailed info:- https://browsersl.ist/