Namaste React Assignment2:

1. What is NPM?

Ans: NPM is used for managing the packages, it is the standard repository for all the packages, if you want to include any package in your application then you can use the NPM.

Example:

Npm install -D parcel (Installing parcel package as a dev-dependency)

Note: NPM does not stand for Node Package Manager, officially there is no proper fullform for NPM.

1. What is Parcel/Webpack why do we need them?

Ans: Parcel/Webpack are the examples of bundlers, bundlers will bundle our code into a single file (Compressed, minified and clean) and makes it ready for the production.

1. What is .parcel-cache?

Ans: Parcel also uses the cache for faster buidling.When you do the build for the first time using npx parcel Index.html, then parcel will create the .parcel-cache folder and use it for the faster builds, next time when you again build the code, using the cache the build will be faster.

1. What is npx?

Ans: NPX stands for “Node Package Execute”, it is used to execute a package. NPX is a package runner tool that comes with npm (Automatically installed with NPM). It is used to execute packages without the need to install them globally.

Example: npx create-react-app swiggyapp

This command uses npx to run the ‘create-react-app’ without having to install it. It fetches the latest version of ‘create-react-app’ and initializes a new React application named swiggyapp.

1. What is the difference between dependencies and dev dependencies?

Ans: dev dependencies are required only during the development time and dependencies are required during the development and also during the production time.

Installing the dependencies:

Npm install axios

Installing the dev dependencies:

Npm install -D parcel

1. What is tree shaking?

Ans: Tree shaking means removal of unsued or dead code. Parcel supports the feature of tree shaking as it removes the unused or dead code during the code optimization.

1. What is hot module replacement?

Ans: HMR means whenever you make any code chnages in your editor the code will be build automatically and it refreshes our React page in the browser and we can see the changes. Parcel supports the HMR by using the “file watch algorithm” which is written in C++.

1. List down your favorite 5 superpowers of Parcel and describe any 3 of them in your own words.

Ans: 1) Parcel supports HMR: Whenever you make some code changes parcel automatically build your code and this makes our React page refresh in the browser. For this parcel uses the “file watch algorithm” which is written in C++.

2) Parcel supports Tree shaking: Tree shaking means it removes the unused or dead code from your project during the code optimization process.

3) Parcel uses the cache for faster build: Whenever you first time build your project using the parcel, then it will create the folder called .parcel-cache, and during the next build parcel will make use of this cache for building your code much faster.

4) Parcel can create different bundles for production and development:

* Development build: npx parcel Index.html
* Production build: npx parcel build Index.html

5) Parcel supports the lazy loading: Lazy loading means loading the resources/ content on the webpage only when needed rather than loading all at once. Lazy loading improves the performance of a webpage.

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

Ans: In .gitignore we add files that do not need to be pushed to git or production. Mostly we place all the files in the .gitignore that can be regenerated.

Example:

1. Node\_modules: It can be generated using the “npm install” and its size is huge, so we put this in .gitgnore.
2. Dist: Dist folder contains the development or production build and in case of parcel it can be re generated using the build command: npx parcel Index.html (dev build) npx parcel build Index.html.
3. .parcel-cache: It can be regenerated using the dev build or production build commands.

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

Ans: package.json maintains the approximate version of the packages whereas package-lock.json maintains the exact version of the package. Package-lock.json ensures everyone working on the sam eproject gets the same version of the packages.

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

Ans: We should never directly modify the package-lock.json.

The primary purpose of package-lock.json is to ensure consistency of dependency versions across different installations (npm install) (ensure that everyone working on the project uses the same versions of dependencies) , if we manually modify the package-lock.json then there might be some inconsistencies so we should not modify the package-lock.json, just npm will update it whenever it is required.

The package-lock.json file includes checksums for each downloaded package. Modifying the file might compromise the integrity checks, potentially loading to corrupted or compromised packages.

So, always let npm change it never manually modify it.

1. What is `node\_modules` ? Is it a good idea to push that on git?

Ans: node\_modules is a kind of database where all our packages will present, all the code that we fetch from the npm will be present in the node\_modules. For example when we do the “npm install parcel” parcel will be fetched from npm and placed inside in the node\_modules folder along with parcel’s all transitive dependencies.

It is not a good idea to push node\_modules folder onto the git, because the size of node\_modules is very huge and it can be recreated using the “npm install” command. so to save space in github and production we need to put the node\_modules inside the .gitignore file, so that it will not be pushed to github and production.

1. What is the `dist` folder?

Ans: dist folder contains the development or production build, whenever we build our code, then the dist folder will be created.

Dist folder mainly contains three files they are: one html file, one css file and one js file. All our application code will be present inside these 3 files only. There will some other map files also present inside this dist folder.

Development build example: npm parcel Index.html

Production byuild example: npm Parcel build Index.html

1. What is `browserlists`?

Ans: It contains the configuratiuon, which tells on what type of browsers and versions our application will run.

Example:

browserslist: [“last 10 Chrome version”]: It means our application for sure will run on the last 10 versions of the google chrome and it may or may not run on other versions and other browsers.

browserslist: [“last 2 versions”]: It means our application will run on last 2 versions of all the browsers.

1. Different types of bundlers vite, webpack and parcel?

Ans: Webpack: Webapck is released on March10th, 2012 and it is written in JavaScript.

* Using webpack we can build our code, it minify our code and using webpack we can configure the entry file (which file to be executed first in our project)
* Installing webpack: npm install webpack webpack-cli –save

After running the above command we can see webpack and webpack-cli in the package.json under the dependencies.

Steps for building basic JavaScript App:

1. Create a folder and open it in the visual studio.
2. Do npm init it will create the package.json
3. In the project folder create the dist folder and add index.html file to it
4. In the project folder create the src folder and add index.js (By default webpack considers entry as index.js) and some other js files.
5. Install webpack: npm install webpack webpack-cli –save
6. Go to the scripts of package.json and add this script: “build”: “webpack”
7. Now run this command in terminal: npm run build, thats it our build is sucessful, in the dist folder one new file called “main.js” will be created it contains our minified code.

Steps for creating the config file in webpack:

* In the above steps we created the build successful and it is stored inside the main.js and all the minified code is present in a single line.

1. We will create a file called “webpack.config.js” file in our project folder for creating the config.
2. Add this code inside the “webpack.config.js” file:

module.exports = {

“mode”: “development”

}

1. Now run the build command: “npm run build”, now our main.js file inside the dist folder contains the minified code but not in a single line, it is splitted into mulyiple lines.

It is not mandatory to use filename as “webpack.config.js”, we can also use other name like “custom.js” but for this we need to make change in scripts of package.json:

Replace “build”: “webpack” with “build”: “webpack --config custom.js”

Note: By default mode is production (Build file will conatin code in a single line).

* By default entry file for webpack is index.js and output file is main.js (build file in dist).
* We can change the entry and output files:
* Entry file:

Example:

module.exports = {

mode: “developement”,

entry: “Path of the file”

}

* Output file: For changing the output file we need to import the path.

const path = require(‘path’);

module.exports = {

mode: “development”,

entry: “Path of the file”,

output:{

path: path.resolve(\_\_dirname, “name of the folder (usually we give dist)”)

filename: “name of the outpit file”

}

}

* Dev server in wepack:

First install the “npm install webpack-dev-server –save-dev”

Add this script in package.json:

“start” : “webpack-dev-server –mode development --open”

In the config file add this:

const path = require(‘path’);

module.exports = {

mode: “development”,

entry: “Path of the file”,

output:{

path: path.resolve(\_\_dirname, “name of the folder (usually we give dist)”)

filename: “name of the outpit file”

},

devServer: {

static: path.join(\_\_dirname, “dist”),

compress: true,

port: 3500

}

}

Now when you run the “npm run start” command your application will be run the localhost port number 3500.

Webpack also supports the HMR: Hot Module Replacement, when you make some changes in your code, it refreshes the page automatically in the browser.

* Applying CSS and style loader with webpack:

npm install –save-dev style-loader css-loader.

Create one style.css file in src:

.hello{color: orange}

Import that style file into js file.

Now go to config file and add this code:

module: {

rules:[

{

test: /\.css$/,

use: :[

‘style-loader’,

‘css-loader’

]

}

]

}

* How to create different config files for different environments like developmnent, production etc?

Ans: create two file namely: config.dev.js and config.prod.js. Now simply copy paste the previous webpack.config.js code in in this both files and from the config.dev.js remove the output file and from the config.prod.js remove the deverver.

Now go to scripts of package.json and replace:

“build”: “webpack”,

“start” : “webpack-dev-server –mode development --open”

With:

“build”: “webpack –config config.prod.js”,

“start” : “webpack-dev-server –mode developemt –open –config config.dev.js”

1. What are ^ caret and ~ tilde?

Ans: Normally the package version will be divided into 3 parts: example: 16.7.3

Here 16 is the major release, 7 is the minor release and 3 is the patch release.

The meaning of ^16.7.3 is, when you have ^16.7.3 in the package.json and you pushed your code into github and someone clones and do the npm install or if you do npm start then the npm will automatically update the minor and patch releases if they are available.

The meaninf of ~16.7.3 is, when you have ~16.7.3 in the package.json and you pushed your code into github and someone clones and do the npm install or if you do npm start then the npm will automatically update the patch release if it is available.

1. Script types in html?

Ans: The script element in the JavaScript is usually used to refer the JavaScript code.

<script src=”App.js”> </script>

The type attribute of the script element refers to the type of content it is referring.

We can omit the type attribute if we are referring to the JavaScript code, as we have done above in this question.

type=”text/javascript”

type=”text/ecmascript”

type=”application/javascript”

type=”application/ecmascript”