**NamasteDev**

Assignment 2 – Igniting our App

Q1: What is NPM?

NPM is the package manager for node.js. It is inappropriately expanded as Node Package Manager. NPM is world’s largest software registry with more than 800000 packages.

Q2: What is `Parcel/Webpack`? Why do we need it?Parcel, Webpack, Vite are some of the bundlers that are used to bundle up, compress the source code for preparing dev and prod builds. Bundler is the most important dependency for an app. These require zero to minimum configuration for setup. Parcel is a beast with many superpowers.

The reason why we need a bundler can be explained as:

Large apps require a lot of code structured into modules and files. If we send the files as it is, that is multiple .js, .html,.css files, the browser will have to make several network calls to fetch these files. This makes our app slower which is not optimum. This can be overcome by bundling up the code to generate minimal files, thereby making our app faster and lightweight.

Q3: What is .parcel-cache?

One of the superpowers of Parcel is to do caching. Parcel caches the code and place it in the .parcel-cache folder.

The .parcel-cache folder stores information about the project when parcel builds. When it re-builds, it doesn’t have to re-parse, re-analyze everything from scratch. This eventually results in subsequent faster builds thereby making our app respond faster.

Since this is a cache folder it must not to be committed to GitHub.

Q4: What is npx?

NPM by itself does not run any package. To do this via npm, the package needs to be installed. Then manually add this package to the “scripts” section in package.json file

{

"name": "whatever",

"version": "1.0.0",

"scripts": {

"some-package": "some-package here"

}

}

Now run the script using the command `npm run some-package`

NPX on the other hand is a command used to execute a node package. The command `npx parcel index.html ` will install the package locally before executing the package.

Q5: What is difference between `dependencies` vs `devDependencies`?

“dependencies”- packages required by your app in production. Installed using the command: npm install some-package –save-prod

“devDependencies” – packages required by app only in development and local testing.

npm install some-package –save-dev or npm install -D some-package

Parcel is installed as a dev dependency in React because this package compresses, bundles up the files in development phase. Bundling must be done before production.

Q6: What is Tree-shaking?

When we import and/or export modules in JS, most of the time there is unused code (referred to as dead code) floating around. This makes our app heavy weight. Tree shaking in JS context is the removal of dead code. It means that unused modules will not be included in the bundle during the build phase.

Tree shaking is supported for both static and dynamic imports, CommonJS and ES6 modules, and even languages with CSS modules.

Q7: What is Hot Module Replacement?

Hot Module Replacement (HMR) automatically updates the modules in the browser at runtime without the need for refreshing the page. In other words, HMR automatically refreshes the content. This means that the application state can be retained even when there is a very small change. Parcel uses a file watching algorithm to implement HMR.

Q8: List down your favourite 5 superpowers of Parcel and describe any 3 of them in your own words.Hot Module Replacement – automatic refresh of content without page refresh

Treeshaking – removal of dead code. Unused imports and exports

Caching - using .parcel-cache

Provides a local server to run the app

Differential bundling – When we want the app to be supported on different platforms and older versions.

Minification – Parcel will minify the code. The way minification works is by replacing long variables with short names.

Image optimization – Most expensive task in any app. Parcel does image compression .

Compression – Parcel supports compressing bundles using Gzip and Brotli

Diagnostics – provides useful log message to the developers. This can be a verbose message for an error, warning. The message can include the file being processed, error message, code frame, some suggestion to rectify the error, a useful link to the documentation.

Q9: What is `.gitignore`? What should we add and not add into it?

.gitignore file specifies intentionally untracked files that must be ignored by Git. Files already tracked by Git are unaffected.

node\_modules/

.parcel-cache/

/dist

Q10: What is `node\_modules` ? Is it a good idea to push that on git?

It is directory created by npm to keep a track of all the packages installed via package.json. We can say that it is like a database for all the dependencies along with its child dependencies.

Node-modules folder contains a lot of files and if there a new developer joining the team, they can install the dependencies using the package.json file. Therefore, it is unnecessary to push this folder to Git.

Q11: What is the difference between `package.json` and `package-lock.json`? Why should I not modify `package-lock.json`? Read about: ^ - caret and ~ - tilda

Package.json (PJ) is a tracked config file that primarily contains the list of dependencies for our project to run along with other meta information. Information like name ,version, description, author of the project, git repository, scripts, dev dependencies etc.

But there is an inherent problem. Look at the snippet below:

“dependencies”:{

“express”:”^2.1.3”

}

The key inside dependencies object is the package name (here express  
) and the value is a Version Range (^2.1.3). Note that this is not just a version number but denotes a range.

This range is known as the Semantic versioning. Universal way of versioning project releases.

Follows the format- MajorVersion.MinorVersion.Patch

^version- Compatible with version and accepts new minor and patch releases.

~version- Compatible with version and accepts only new patch

\*version – Compatible with version and accepts any new major, minor or patch.

Version – exact version

This versioning standard makes npm install indeterministic. How? Say, I run my code today and then 3 months later, I will not get the same node\_modules tree. What if there are new developers joining the team every month. Everyone will have a different structure of node\_modules. This will lead to inconsistencies in the dependencies installed. This can break the application.

This can be resolved using the package-lock.json(PLJ). PLJ is a lock file that contains information about the EXACT version of the installed dependency. This file is automatically generated(re-generated) when there is a change in the PJ or node\_modules. PLJ also has all information about the child dependencies (if any).

Note : Both PJ and PLJ must be pushed to GitHub. If I do a git clone and then install the dependencies using npm install, npm first looks for the PLJ only of the packages are in the same version range. If not, PJ is considered and PLJ is overwritten.

Q12: What is the `dist` folder?

The /dist stands for distributable. It is this folder that contains the minimized version of the source code. The code present in the /dist folder is the code which is used on production web applications. Should be ignored while doing git push.

Q13: What is `browserlists`?

Browserslist is a tool that allows specifying which browsers should be supported in your frontend app by specifying queries in a config file(package.json). Why do we need this?

As JS evolves, browsers won’t support new features at the same pace. Say some support only ES5. By using browserlist, bundlers know what browsers can be supported, so they can "group" browsers in different categories and generate separate bundles.

 "browserslist": [

    "last 2 versions"

  ]

Read about diff bundlers: vite, webpack, parcel

A table with text on it

Description automatically generated

Code splitting – In parcel, the bundled folder has a flat structure. Image, JS files, CSS everything is cramped up in one big folder. Whereas Webpack has a clean structure in the bundled folder where the image, js, css files can be easily identified.

Webpack 5 Vs. Parcel v2

Link <https://levelup.gitconnected.com/parcel-vs-webpack-2021-64c347bcb31>

● Read about Script types in html (MDN Docs)

Type indicates the type of script.

type=importmap -this indicates that the body of the element contains an import map

<script type="importmap">

// JSON object defining import

</script>

type=module – this indicates that the code is treated as a javascript module. The processing of the script contents is deferred (HTML parsing will be completed first while the Scripts are being downloded.Only then the scripts are executed)