Assignment 02 - Igniting Our App

Owner	Pankaj Kumar

1: What is NPM?

- It is a tool used for package management and the default package manager for Node projects. NPM is installed when NodeJS is installed on a machine. It comes with a command-line interface (CLI) used to interact with the online database of NPM. This database is called the NPM Registry, and it hosts public and private 'packages.' To add or update packages, we use the NPM CLI to interact with this database.
- We use NPM because we want a lot of packages in our Projects/React App.

Note : npm does not stand for node package manager but everything else.

- `npm` alternative is `yarn`

How to initialize `npm`?

```
npm init
```

`npm init -y` can be used to skip the setup step, `npm` takes care of it and creates the `package.json` json file automatically , but without configurations.

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

- Parcel/Webpack is type of a web application bundler used for development and productions purposes or power our application with different type functionalities and features.

- Parcel and webpack are the bundlers used mostly for JavaScript or Typescript code that helps you to minify, clean, and make your code compact so that it becomes easier to send a request or receive the response from the server when it usually takes you to transfer multiple files without using any bundler for loading the page of your application.
- Both of these bundlers substantially reduce the time it takes for the transfer of data and files to the server from the application.
- Along with that both bundlers parcel and webpack remove the unnecessary comments, new lines, any kind of block delimiters, and white spaces while the functionality of the code remains unchanged.
- It offers blazing fast performance utilizing multicore processing, and requires zero configuration. Parcel can take any type of file as an entry point, but an HTML or JavaScript file is a good place to start.

Parcel Features:

- HMR (Hot Module Replacement) —> parcel keeps track of file changes via file watcher algorithm and renders the changes in the files
- File watcher algorithm (written in C++)
- Minification
- Cleaning our code
- DEV and production Build
- Super fast building algorithm
- Image optimization
- Caching while development
- Compresses
- Compatible with older version of browser
- HTTPS in dev
- Port Number
- Consistent hashing algorithm

- Zero Configuration
- Automatic code splitting
- Tree Shaking —> Removed unwanted/unused code

installation commands:

```
npm install -D parcel
```

- `D` is used for development and as a development dependency.
- Parcel Commands :
  - For development build:

    ```
    npx parcel <entry_point>
    ```

  - For production build :

    ```
    npx parcel build <entry_point>
    ```

3: What is `.parcel-cache`

- `.parcel-cache` is used by parcel(bundler) to reduce the building time.
- It stores information about your project when parcel builds it, so that when it rebuilds, it doesn't have to re-parse and re-analyze everything from scratch. It's a key reason why parcel can be so fast in development mode.
- The dist folder contains the output of Parcel and the content of that folder is served by the web server.

4: What is `npx`?

- The `npx` stands for Node Package Execute. It is a tool that is used to execute the packages. It comes with the npm, when you installed npm above 5.2.0 version then automatically npx will be installed. It is an npm package runner that can execute any package that you want from the npm registry without even installing that package. npx is pre-bundled with npm.

- If the package is only to be used once or twice, rather than every time the project runs, it is preferable to utilize NPX, which will execute the package without installing it.
- NPM is used to install packages, which we should do if our project requires dependencies or packages.

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

- Dependencies should contain library and framework in which your app is built on, needs to function effectively. such as Vue, React, Angular, Express, JQuery and etc.
- DevDependencies should contain modules/packages a developer needs during development.
  such as, parcel, webpack, vite, mocha etc. These packages are necessary only while you are developing your project, not necessary on production.
- To save a dependency as a devDependency on installation we need to do,

```
npm install --save-dev
```

instead of just,

```
npm install --save
```

6: What is Tree Shaking?

- Tree shaking is a term to describe the removal of dead/unused code.
- “You can imagine your application as a tree. The source code and libraries you actually use represent the green, living leaves of the tree. Dead code represents the brown, dead leaves of the tree that are consumed by autumn. In order to get rid of the dead leaves, you have to shake the tree, causing them to fall.”

- In JavaScript land, tree-shaking has been possible since the ECMAScript module (ESM) specification in ES2015, previously known as ES6. Since then, tree-shaking has been enabled by default in most bundlers because they reduce output size without changing the program’s behaviour.

7: What is Hot Module Replacement?

- Hot Module Replacement (HMR) exchanges, adds, or removes modules while an application is running, without a full reload.
- This can significantly speed up development in a few ways: Retain application state which is lost during a full reload.
- HMR is enabled for the dev server. When you change your files and save them, the HMR will automatically change your contents without recompile and reload whole project.
- HMR is the same as Live Reload with the difference that it only replaces the modules that have been modified, hence the word Replacement.
- The advantage of this is that it doesn't lose your app state e.g. your inputs on your form fields, your currently selected tab etc.

8: Superpowers of Parcel and describe any 3 of them in your own words.

- Zero config: Parcel supports many languages and file types out of the box, from web technologies like HTML, CSS, and JavaScript, to assets like images, fonts, videos, and more. It has a built-in dev server with hot reloading, beautiful error diagnostics, and much more. No configuration needed!
- HMR (Hot Module Replacement): Adds/removes/Updates modules while an application is running, without a full reload.
- File watcher algorithm: File Watchers monitor directories on the file system and perform specific actions when desired files appear.
- Automatic production optimization: Parcel optimizes your whole app for production automatically. This includes tree-shaking and minifying your JavaScript, CSS, and

HTML, resizing and optimizing images, content hashing, automatic code splitting, and much more.

- Ship for any target: Parcel automatically transforms your code for your target environments. From modern and legacy browser support, to zero config JSX and TypeScript compilation, Parcel makes it easy to build for any target or many!
- Scalable: Parcel requires zero configuration to get started. But as your application grows and your build requirements become more complex, it's possible to extend Parcel in just about every way. A simple configuration format and powerful plugin system that's designed from the ground up for performance means Parcel can support projects of any size.

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

- A gitignore is a text file where each line contains a pattern for files or directories to ignore. It is usually placed at the root of the project folder.
- A gitignore file specifies intentionally untracked files that Git should ignore. Files already tracked by Git are not affected. The .gitignore file is a text file that tells Git which files or folders to ignore in a project during commit to the repository.
- The types of files you should consider adding to a .gitignore file are any files that do not need to get committed. for example, For security, the security key files and API keys should get added to the gitignore.
  Note: package-lock.json should not add into your .gitignore file.
- You should not commit these four types of files into your Git repository.
  - Files that don't belong to the project
  - Files that are automatically generated
  - Libraries (depends on the situation)
  - Credentials
- The entries in this file can also follow a matching pattern.

```
* is used as a wildcard match
/ is used to ignore pathnames relative to the .gitignore file
```

```
# is used to add comments to a .gitignore file
```

This is an example of what the .gitignore file could look like:

```
# Ignore Mac system files
.DS_store

# Ignore node_modules folder
node_modules

# Ignore all text files
*.txt

# Ignore files related to API keys
.env

# Ignore SASS config files
.sass-cache
```

10: What is the difference between `package.json` and `package-lock.json`

- `package.json`:
  - The package. json file is the heart of any Node project.
  - It records important metadata about a project which is required before publishing to NPM, and also defines functional attributes of a project that npm uses to install dependencies, run scripts, and identify the entry point to our package.
  - The file resides in the root directory of every Node. js package and appears after running the npm init command.
  - The package. json file contains descriptive and functional metadata about a project, such as a name, version, and dependencies.
- `package-lock.json`:
  - This file is automatically generated for those operations where npm modifies either the node_module tree or package-json.
  - The “package. json” file defines the rules required to run your application and install dependencies. On the other hand, the “package-lock. json” file holds

detailed information on all the dependencies installed based on the package.

  - It is generated after an npm install and not designed to be manually edited and we should not delete it either.
  - As name suggests, lock. json is created for locking the dependency with the installed version. It will install the exact latest version of that package in your application and save it in package.
  - It allows future devs & automated systems to download the same dependencies as the project.
  - it also allows to go back to the past version of the dependencies without actual committing the node_modules folder.
  - It records the same version of the installed packages which allows to reinstall them. Future installs wll be capable of building identical description tree.

~ or **^** in `package.json` file :
These are used with the versions of the package installed.

For example in `package.json` file:

```
"dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  }
```

- **~** : “*Approximately equivalent to version*”, will update you to all future patch versions, without incrementing the minor version.
  eg. [major, minor, patch] is a tuple of versions where ~1.2.3 will use releases from 1.2.3 to <1.3.0.
- **^** : “*Compatible with version*”, will update you to all future minor/patch versions, without incrementing the major version.
  eg. ^2.3.4 will use releases from 2.3.4 to <3.0.0.

> If none of them is present , that means only the version specified in package.json file is used in the development.

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

`package-lock.json` file contains the information about the dependencies and their versions installed in the project. Deleting them would cause dependency issues in the production environment. So you should not modify it as it's being handled automatically by NPM.

12: What is `node_modules` ? Is it a good idea to push that on git?

- `node_modules` folder like a cache for the external modules that your project depends upon.
- When you npm install them, they are downloaded from the web and copied into the node_modules folder and Nodejs is trained to look for them there when you import them (without a specific path).
- we should not push `node_modules` in github because it contains lots of files(more than 100 MB), it will cost you memory space.

13: What is the `dist` folder?

- The /dist stands for distributable.
- The /dist folder contains the minimized version of the source code.
- The code present in the /dist folder is actually the code which is used on production web applications.
- Along with the minified code, the /dist folder also comprises of all the compiled modules that may or may not be used with other systems.
- It is easier to add files to the /dist folder as it is an automatic process. All the files are automatically copied to the dist folder on save.
- The /dist folder also contains all those files which are required to run/build a module for use with other platforms- either directly in the browser, or in an AMD system (eg. require.js).
- Ideally, it is considered a good practice to clean the /dist folder before each build.

14: What is `browserslist`?

- There is a package called ‘browserlist’ & parcel automatically gives to us.
- Browserslist makes our code compactible for a lot of browers.
- Browserslist is a tool that allows specifying which browsers should be supported in your frontend app by specifying "queries" in a config file.
- In package.json file, do:

```
"browserslist": [
    "last 3 versions"
  ]
This means my parcel will make sure that my app works in last 3 versions of
all the browsers available.
```

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