Chapter 02 Assignment Theory

1. What is NPM ?
   NPM is a package manager (not *node package manager).* It is used when we want to include various package into our react application.

2. What is 'Parcel/Webpack'? Why do we need it?
   Parcel/Webpack/vite are module bundlers. It bundles the various modules and assets and generate static assets.

3. What is '.parcel-cache'
   To reduce the building time for subsequent builds, parcel will store project information into *.parcel-cache,* which it would have created when the project is built for the first time.

4. What is 'npx'
   Npx is node package executer. It is used to when you run a package using npx, it searches the local and global registry for the package and then executes it.

5. What is the difference between dependencies vs devDependencies

Dependencies	**DevDependencies**
These are the dependencies that are required in dev and prod	These are the dependencies which might be used only during development
Use *npm install --save*	Use *npm install --save-dev*
Probable example: React, Express, Axios	Probable example: ESLint, Mocha

6. What is tree shaking?
   It refers to removal of dead code. It is an important step in preparing code for production. Bundlers remove the dead code when bundling the modules into single files.

7. What is Hot Module Replacement

   It refers to the exchange, addition, or removal of modules while an application is running without a full reload. This speed up the development.

   The application asks the HMR runtime to check for updates > The runtime asynchronously downloads the updates and notifies the app. > The app then asks the runtime to apply the updates > Runtime synchronously applies the updates.

8. List down 5 superpowers of Parcel and describe any 3 of them.
   Few core features of Parcel are:
   1. Development: dev server, hot reloading, lazy mode, caching etc
   2. Code splitting: zero configuration code splitting
   3. Targets: Compile code for multiple targets
   4. Production: Minification, tree shaking, content hashing etc
   5. Bundle inlining: Inline the compiled contents of one bundle inside another

Code splitting: Parcel supports automatic code splitting. Meaning we can split our application into separate bundles. These bundles can be loaded as and when needed. This is a crucial step that makes the loading faster.

Targets: Parcel follows the dependencies in each resolved entry to build your source code for one or more targets. Targets specify the output directory or file path and also information about how the code should be compiled. By default the output is put into *dist* folder. Target also specifies the info about environments the code can run in. They tell Parcel what type of environment to build for (e.g. a browser or Node.js), as well as what versions of each engine you support. This influences how Parcel compiles your code, including what syntax to transpile. An example would look like this in package.json:

```
{
"modern": "dist/modern.js",
 "legacy": "dist/legacy.js",
 "targets": {
   "modern": {
     "engines": {
       "browsers": "Chrome 80"
     }
   },
   "legacy": {
     "engines": {
       "browsers": "> 0.5%, last 2 versions, not dead"
     }
   }
 }
}
```

Production: Parcel automatically bundles and optimizes your application for production. It applies size optimization, Development branach removal, image optimization, differential bundling, cache optimization etc.

9. What is `.gitignore`? What should we add and not add into it?
   It is a file that specifies the files that git should not worry about. We should not add any file or folder that can be generated on the fly. For example, *node_modules*

10. What is the difference between `package.json` and `package-lock.json`
    Package.json has the data about the project and the packages that the project needs. Package-lock.js has the information about versions of packages that the project is currently installing and using.

11. Why should I not modify `package-lock.json`?
    Since this files contains the exact versions of packages that are being used in the project, any change to this file might cause compatibility issues.

12. What is `node_modules` ? Is it a good idea to push that on git?
    It is a folder that gets generated when you run npm install. It houses the installed dependencies and transitive dependencies that are mentioned in package.json and package-lock.json. It should not be pushed to git because this takes up a huge space. Moreover, this can be regenerated based on package.json and package-lock.json

13. What is the `dist` folder?
    It is a distribution folder that contains minified version of our project including the compiled modules, to be used in prod.

14. 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.