Q.1 What is 'NPM'?

- It manages our packages.
- Though its official name is not Node Package Manager.
- A React app is a huge app that just does not work on React, instead needs a lot of superpowers which we need, and those super powers come from different packages, and those packages are handled by NPM.
- Command: **npm init** - This command will ask you some questions to generate a package.json file in your project route that describes all the dependencies of your project. This file will be updated when adding further dependencies during the development process.

Q.2 What is 'parcel/ Webpack'? Why do we need it?

- Parcel is a bundler.
- It's a fast bundles tool with a zero configuration - all you need to do is just point it at the entry point of your application.
- It do alot of work for us like -
    - **Bundling -** Bundling is the process of following imported files and merging them into a single file: "A Bundle".This bundle can then be included on a webpage to load an entire app at once. Parcel has a fast bundling time.
    - **Minify our app**
    - **Automatic Transforms:** All our code is automatically transformed using Babel, Post CSS, and PostHTML.
    - **Code Splitting:** Parcel allows you to split the output bundle by using the dynamic import() syntax.
    - **HOT Module Reloading(HMR):** Parcel automatically updates modules in the browser as you make changes during development, no configuration needed.
    - **Dev Server:** Parcel builtin dev server is automatically started when you run the default parcel command, which is a shortcut for parcel serve. It starts a server at localhost at 1234. If port 1234 is in use, then a fallback port will be used.
    - **Lazy Mode:** In development, it can be frustrating to wait for the whole app to build before the dev server starts up. Instead we can use –lazy CLI flag to tell parcel to defer

building files until they are requested in the browser, which can significantly reduce dev build times.

- **Caching:** Parcel Caches everything it builds to disk. If you restart the dev server, Parcel will only rebuild files that have changed since the last time it ran. Parcel automatically tracks all the files, configuration, plugins and dev-dependencies that are involved in your build, and granularly invalidates the cache when something changes. Ex: If you change a config file, all of the source files that rely on the config will be rebuilt.

  By default, the cache is stored in the.parcel-cache folder inside your project. You should add this folder to your .gitignore so that it is not committed in your repo. You can also override the location of the cache using the –cache-dir CIL option.

  It can also be disabled using –no-cache flag. But this only disables reading from the cache -a .parcel-cache folder will still be created.
- **HTTPS:** Sometimes, we may need to use HTTPS during development.To use an automatically generated self-signed certificate, use the –https CLI Flag.
- **API Proxy:**
- **File Watcher:** To support an optimal caching and development experience Parcel utilizes a very fast watcher written in C++ that integrates with low-level file watching functionality of each operating system. Using the watcher Parcel watches every file in project root(including node modules). Based on events and metadata and events from these files, Parcel determined which files need to be rebuilt.
- **Auto Install:**When you use a language or plugin that isn't included by default, Parcel will automatically install the necessary dependencies into your project. Example, if we include a .sass file, parcel will install the @parcel/transformer/sass.
- **Development Target:** When using the dev-server, only a single target can be built at once. By default,Parcel uses a development target that supports modern browsers.

- If you need to test in older browsers, you need to provide the –target CLI option to choose which of your targets to build.
  File Name: package.json
  Ex:
  “Browserlist”: [“last 2 versions of Chrome”]

Q.3 What is ‘parcel-cache’?
Parcel Caches everything it builds to disk. If you restart the dev server, Parcel will only rebuild files that have changed since the last time it ran. Parcel automatically tracks all the files, configuration, plugins and dev-dependencies that are involved in your build, and granularly invalidates the cache when something changes. Ex: If you change a config file, all of the source files that rely on the config will be rebuilt.

By default, the cache is stored in the.parcel-cache folder inside your project. You should add this folder to your .gitignore so that it is not committed in your repo. You can also override the location of the cache using the –cache-dir CIL option.

It can also be disabled using –no-cache flag. But this only disables reading from the cache -a .parcel-cache folder will still be created.

Q.4 What is ‘npx’?
The npx stands for Node Package execute and it comes with the npm, when you install npm above 5.2.0v then automatically npx will be installed. It is a npm package runner that can execute any package that you want from the npm registry without even installing the package.

Q.5 What is the difference between ‘dependency’ and ‘dev-dependency’?
In every web application project, we have a file called package.json. This file contains all the relevant data regarding the project i.e., metadata. Starting from all the dependencies used to all version numbers are present in the file. In this way, there are three types of dependencies - Dependency, Dev Dependency and Peer Dependency.

- **Dependency:** In package.json file , there is an object called dependencies and it consists of all the packages that are used in the project with its version number.Son, whenever we install any library that is required in your project that library we can find in dependencies object.
- **Dev Dependency:** In the package.json file, there is an object called dev dependencies, and it consists of all the packages that are used in the project in its development phase and not in the production or testing environment with its version number. So, whenever we want to install any library that is required only in your development phase then we can find it in the dev dependency object.

Q.6 What is Tree Shaking?

Tree Shaking is a process that bundlers like webpack and rollup go through to remove unused code. Tree shaking means that only components from our library used in the app are included in the app bundle.

Q.7 What is 'HMR'?

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 days.Retain application state which is lost during a full reload.

Q.8 What is '.gitignore'?

Such files that we want to ignore before committing for example, the files that are to do with our user settings or any utility settings, private files like passwords and API keys. These files are not of use to anyone else and we do not want to clutter our git. We do this with the help of '.gitignore'.

'.gitignore' is an auto-generated file inside the project folder that ignores files to get committed to the local and remote repositories.

Anything we can generate on our server, we keep in '.gitignore'.We should put parcel.cache in .gitignore.

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

The package.json is used for more than dependencies like defining project properties, description, author and license information, scripts etc.
The package-lock.json is solely used to lock dependencies to a specific version number.

Q.10 Why should I not modify 'package-lock.json'?
It is a generated file and is not designed to be manually edited. Its purpose is to track the entire tree of dependencies and the exact version of each dependency.
We should commit package-lock.json to ourcode repository.

Q.11 What are 'node modules'? Is it a good idea to push that on git?
The node_module folder contains every installed dependency for your project. We should not commit this folder into our version controlled repository. As you install more dependencies, the size of this folder will quickly grow.

Q.12 What is the 'dist' folder?
The dist folder contains the minimized version of the source code. The code present in the /dist folder is the code which is used on production.

npx parcel (root file name)
Creates a development build and host on a server.

npx parcel build (root file name)
Builds on production and creates all files in the dist folder.

Q.13 What is 'browserlists'?
Browserlist is a tool that allows specifying which browsers should be supported in our frontend app by specifying "queries" in a config file.

Q.14 What is (tilde) and ^(caret)?
Npm uses the (tilde) and ^(caret) to designate which patch and minor version to use respectively. So, if we see (tilde)1.0.2 it means to install version 1.0.2 or the latest patch version such as 1.0.4 .