Igniting our App

Q1> What is `NPM`?

Ans> ‘node package manager’, is both a registry for open-source software and a command line client allowing developers to interact with npm. A package manager is a piece of software that allows you to download and install software.

npm comes packaged with NodeJS, so if you have NodeJS downloaded, you should have npm as well.

Q2> What is `Parcel/Webpack`? Why do we need it?

Ans> Parcel is a web application bundler, differentiated by its developer experience. It offers blazing fast performance utilizing multicore processing and requires zero configuration. Webpack is detailed as "A bundler for JavaScript and friends". A bundler for JavaScript and friends.

Today web developers which writes code won’t understand by browser as it’s high-level codes hence a transformation is made by the bundler. Apart from it has other benefits like hot module replacement, code splitting. Parcel does all this FAST with zero configuration.

Q3> What is `.parcel-cache`?

Ans> The .cache folder (or .parcel-cache in parcel v2) 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.

Q4> What is `npx`?

Ans> The npx stands for **Node Package Execute** and it comes with the npm when you installed npm above 5.2.0 version then automatically npx will installed.

Packages used by npx are not installed globally. You don’t have to worry about for pollution in the long term.

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

Ans>

"dependencies": Packages required by your application in production.

"devDependencies": Packages that are only needed for local development and testing.

Q6> - What is Tree Shaking?

Ans> Tree shaking is a term commonly used within a JavaScript context to describe the removal of dead code.

It relies on the import and export statements to detect if code modules are exported and imported for use between JavaScript files.

In modern JavaScript applications, we use module bundlers (e.g., webpack or parcel) to automatically remove dead code when bundling multiple JavaScript files into single files. This is important for preparing code that is production ready, for example with clean structures and minimal file size.

Q7> What is HOT Module Replacement?

Ans> 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.
* Save valuable development time by only updating what's changed.
* Instantly update the browser when modifications are made to CSS/JS in the source code, which is almost comparable to changing styles directly in the browser's dev tools.

The following steps allow modules to be swapped in and out of an application:

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

Q8> List down your favorite 5 superpowers of Parcel and describe any 3 of them in your

own words.

Ans> Parcel is a web application bundler which offers a blazingly fast performance utilizing multicore processing and requires zero configuration. It has many features, few of them are mentioned below:

* **Code splitting** using dynamic imports
* Assets handling for any type of file, but of course for HTML, CSS and JavaScript
* **Hot Module Replacement** to update elements without a page refresh during development
* Mistakes in the code are **highlighted** when they are logged, making them easy to locate and correct
* Environment variables to easily distinguish between local and production development
* A **“Production Mode”** that speeds up the build by preventing unnecessary build steps
* Image Optimization
* Caching while development
* Compression of build files.
* Compatible with older version of browsers similar like Polyfills.
* HTTPS on Development
* Port number
* Zero configuration

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

Ans> The .gitignore file is a text file that tells Git which files or folders to ignore in a project.

The types of files you should consider adding to a .gitignore file are any files that do not need to get committed.

Q10> What is the difference between `package.json` and `package-lock.json`?

Ans>

| **package.json** | **package.lock.json** |
| --- | --- |
| It contains basic information about the project. | It describes the exact tree that was generated to allow subsequent installs to have the identical tree. |
| It is mandatory for every project. | It is automatically generated for those operations where npm modifies either node\_modules tree or package.json. |
| It records important metadata about the project. | It allows future devs to install the same dependencies in the project. |
| It contains information such as name, description, author, script, and dependencies. | It contains the name, dependencies, and locked version of the project. |

Q11> Why should I not modify `package-lock.json`?

Ans> package-lock.json file tells which version of library is using in PRODUCTION.

Without package.lock.json, there might be some differences in installed versions in different environments. To overcome this problem, package.lock.json is created to have the same results in every environment. It should be in source control with the package.json file because if any other user will clone the project and install dependencies then it will install the exact same dependencies as in package.lock.json to avoid differences.

Q12> What is `node\_modules` ? Is it a good idea to push that on git?

Ans> It is like a database for NPM. Whenever any package install with npm, it’s get installed into Node\_modules folder. Node\_modules folder is created when “Npm init” invoke. It contains all the folders & files mentioned in package.json file. It’s a huge folder and it should not be push into git or any central repository.

Q13> What is the `dist` folder?

Ans> When we build our code for deployment, it generates dist folder. It contains our production build code files & folders.

**Command: npx parcel build index.html**

Q14> What is `browserlists`?

Ans> Browserlist talks about supportive browsers for our application. We can exclusively mention about which browser and its version supports our application.

Q15>