Assignment 2 - Igniting Our App

- **What is ‘npm’ ?**
- Npm is a package manager through which we can manage various dependencies in our app. It also supports version controlling of the added packages.

- **What is ‘Parcel/Webpack’? Why do we need it ?**
- Parcel is a bundler which is used for bundling all the code. Parcel is used to make our application production ready by optimizing our files by doing various operations such as code splitting, bundling, etc. and making our app ready for deployment on production.

- **What is ‘.parcel-cache’ ?**
- .parcel-cache is a file where parcel does caching while building the project for production or for dev. It caches the files and details and checks if any file is changed. If it finds some changes in a file then it reloads the page but the build time is very less compared to the first time because of the cache. It is generated automatically.

- **What is npx?**
- Npx is a command line tools which is used for executing packages which are installed in the system. It's often used to execute binaries from packages without the need to install them globally or clutter your project's dependencies.

- **What is difference between ‘dependencies’ and ‘devDependencies’?**
- The difference is that dependencies which are included in dependencies are required in the production build for performing the required operation. Whereas devDependencies are not needed in the production build and only required in development for debugging or error checking by developers.

- **What is Tree Shaking?**
- Tree shaking in Parcel refers to the process of analyzing and eliminating unused or dead code from your JavaScript bundles during the build process. The goal of tree shaking is to reduce the size of the final bundle by removing parts of the code that are never used, thus improving the overall performance and load times of your web application. Parcel, as a build tool, automatically performs tree shaking as part of its optimization process.

- **What is Hot Module Replacement?**

- Hot Module Replacement (HMR) is a feature in development tools that allows you to update parts of your application in real-time without requiring a full page reload. It enables developers to make changes to their code and immediately see the effects reflected in the browser, speeding up the development process and improving productivity. HMR works by swapping out the updated modules in the running application while preserving the application's state.

- **List down fourite 5 superpowers or Parcel and describe any 3?**
- 5 superpowers of Parcel
    - Hosts website on Local Server
    - Tree Shaking
    - Code Splitting
    - Faster Building using caching
    - Differential Bundling for supporting older browsers

- **Hosts website on Local Server** - Using this feature we can host our website on the local machine
- **Faster Building using caching** - Because of caching, the build times are very fast when we build the app again. It caches the previous build details through which the next build is faster.
- **Differential bundling for supporting older browsers** - It automatically provides the support for older browsers by generating nomodule for older browsers.

- **What is .gitignore? What should we add and not add into it ?**
- .gitignore is the file where we add the files and folder which are not supposed to be pushed into the git repo. The files which be generated again from other files (such as node_modules, dist, .parcel-cache, etc.) are not required to be pushed into the git repo. The files which are not automatically generated are not supposed to be added in .gitignore.

- **What is difference between ‘package.json’ and ‘package-lock.json’ ?**
- **package.json:** This file contains metadata about your project, including the project's name, version, dependencies, scripts, and other configuration settings. It's used to define the project's structure and dependencies but doesn't necessarily lock down exact versions.
- **package-lock.json:** This file is generated automatically when you install or update dependencies using npm. It records the exact versions of each package and its dependencies that were installed. This ensures

consistent and reproducible builds across different environments and prevents unintended version mismatches.

- In short, package.json outlines your project's configuration and dependencies, while package-lock.json specifies the exact versions of those dependencies for consistency.

- **What is difference between ‘package.json’ and ‘package-lock.json’ ?**
- Modifying the package-lock.json file is not recommended because it's automatically generated by npm to ensure consistent and reproducible builds. Editing it manually can lead to unintended version conflicts, dependency inconsistencies, and difficulties in maintaining the project's stability. It's best to let npm manage the package-lock.json file to ensure accurate dependency management and reliable builds.

- **What is ‘node_modules’ ? Is it good idea to push that to git ?**
- Node_modules is a folder which contains all the dependencies and their codes which are being used in the project. It acts as a database containing the source code and logic of each of the dependencies used in the project. **It is not a good idea to push node_modules in git** because it is automatically generated and its code may also change based on the dependency version.

- **What is ‘dist’ folder ?**
- In a React application built with Parcel, the "dist" folder stands for "distribution" and contains the final output files after the build process. These files are optimized, bundled, and ready to be deployed to a web server. The "dist" folder typically includes minified and concatenated versions of your JavaScript, CSS, and other assets, making it suitable for hosting and serving your application in a production environment.

- **What is ‘browserslists’ ?**
- Browserlists is an attribute in package.json in which we can configure that which browsers our app should support. Like ‘last 2 versions’, ‘last 2 chrome versions’, etc.

- **What is vite, webpack, parcel ?**
- Vite, webpack, and Parcel are all popular bundlers used in web development to package and optimize code for production. Each has its strengths:

- **Vite**: Vite is a modern bundler that emphasizes fast development and build times. It leverages ES modules to provide near-instantaneous hot module replacement (HMR) and efficient caching during development. Best suited for small to medium-sized projects where speed and developer experience are key. Great for building Vue.js applications and works well with other frameworks as well.

- **Webpack**: Webpack is a powerful and versatile bundler used widely in the industry. Offers extensive customization and features through plugins and loaders. Suitable for projects of all sizes, from small to large, with complex requirements. Requires more configuration but provides fine-grained control over the build process.

- **Parcel**: Parcel is an easy-to-use and zero-config bundler, ideal for quick setups. It handles many tasks automatically, like bundling, minification, and asset optimization. Best for simple projects or when you want a low barrier to entry. May have fewer customization options compared to Webpack.

- Which to Use:
    - **For Speed and Small Projects:** If you prioritize speed and are working on small to medium-sized projects, Vite might be a good choice due to its fast development experience.
    - **For Flexibility:** If you need extensive customization and have more complex requirements, Webpack offers high configurability.
    - **For Simplicity:** If you prefer simplicity and want to avoid extensive configuration, Parcel is straightforward and quick to set up.

- **Read About: ^ caret and ~ tilda?**
- In dependencies, ^ and ~ are used for the dependency versions. Caret (^) is used where we can automatically update the dependency if any minor updates are released. Tilda (~) is used where we automatically update the dependency even after a major version update (like 2.0.0 to 3.0.0). It is advised to use ^ as using ~ may break the code with major update.

- **Read About Script types in HTML ?**
- In HTML, the type attribute in the <script> tag specifies the content type of the script being included. Here are the commonly used type values:

- **text/javascript:** Specifies that the content is JavaScript code. This type is widely supported and used by default if the type attribute is omitted.

- **module:** Indicates that the script is a JavaScript module, allowing for modular code organization with features like import and export.
- **application/json:** Defines JSON data. Not commonly used for scripts but rather for embedding JSON data within the HTML document.
- **text/html:** Rarely used for scripts; typically used to embed HTML code as a script.
- **text/css:** Specifies that the content is CSS code. Rarely used in modern development due to the separation of CSS in separate files.
- **text/plain:** Specifies plain text, not treated as executable code.
- Remember that for JavaScript, the modern and recommended approach is to use the type="module" attribute for scripts that are organized as modules and the default type="text/javascript" for regular scripts.