Assignment 2 - Igniting our app

1. **What is 'NPM'?**

   Npm is a node package manager but it doesn't stand for node package manger.It provides a vast repository of ready-to-use packages, allowing developers to easily install, update, and remove dependencies with simple commands.
   Npm uses package.json to track the project details and dependencies.
   The command-line interface (CLI) of npm offers a user-friendly way to interact with the package manager and perform various tasks.

2. **What is Parcel/Webpack? Why do we need it?**

   Parcel and Webpack are both popular bundlers used in modern web development to bundle and optimize web assets such as JavaScript, CSS, images, and more. They serve similar purposes, but there are some differences between the two.

   Parcel requires minimal configuration to get started and is known for its ease of use and simplicity, making it an excellent choice for small to medium-sized projects.

   Webpack is a powerful and highly configurable bundler that offers extensive control over how the assets are processed and bundled. Due to its flexibility and robustness, Webpack is often preferred for larger and more complex projects.

   Bundlers are essential in modern web development for the following reasons:

   - Bundle Assets: They bundle and optimize various assets, like JavaScript and CSS, into smaller and more efficient files, reducing the size and improving the loading speed of web pages. They also perform image optimization.

   - Dependency Management: They manage dependencies between different assets, ensuring that everything needed for the application to run is included in the final bundle.

   - Code Transformation: They can transform modern JavaScript features and other cutting-edge technologies into compatible code for older browsers, enabling broader browser support.

- Code Splitting: They support code splitting, allowing the application to load only the necessary parts on demand, leading to faster initial page loads.

- Development Workflow: They enhance the development workflow by enabling features like hot module replacement, which automatically updates the application in the browser during development without a full page refresh.

- Optimizations: They apply various optimizations, like minification and compression, to reduce the size of the final bundle, which is crucial for improving website performance.

3. **What is '.parcel-cache' ?**

   .parcel-cache is a directory created by Parcel during the bundling process. When we run Parcel to bundle our web assets (JavaScript, CSS, images, etc.), it analyzes the project's dependencies and generates optimized bundles. To speed up the bundling process for subsequent runs, Parcel caches certain data and intermediate files in the .parcel-cache directory.
   Build time reduces due to this caching.

4. **What is 'npx' ?**

   npx is a command-line tool that comes bundled with npm starting from version 5.2.0. It stands for "Node Package eXecute" and is used to execute Node.js packages directly from the npm registry or from a local file system.
   The primary purpose of npx is to make it easy to run command-line tools and executables that are included in npm packages without having to install them globally or include them in the package.json as dependencies.

   For example, we can run a package named my-package with npx as follows:

   npx my-package
   This command will check if my-package is installed in the local node_modules, and if not, it will install it on-the-fly and then run it.

5. **What is the difference between 'dependencies' vs 'devDependencies' ?**

"**Dependencies**" are packages that are required for the application to run correctly in the production environment. These packages are essential for the application's core functionality.
Packages listed under "dependencies" are used in the production version of the application. They are crucial for the application's proper functioning and must be included when deploying the application to a production server.
When you run **npm install** or **npm ci (clean install)**, npm will install all the packages listed under **"dependencies"** in the **package.json** file.

"**devDependencies**" are packages that are only required during development and are not necessary for the application to run in the production environment.
When you run **npm install -D**, npm will install all the packages listed under "**devDependencies**" in the **package.json** file.
Packages listed under "devDependencies" are typically used for development and testing purposes. They include tools like testing frameworks, code linters, build tools, and development servers. These packages help developers with tasks like running the development server, running tests, or performing code linting, but they are not required to be installed in the production environment where they won't be used.

6. **What is Tree Shaking ?**

Tree shaking is an optimization technique used in modern JavaScript development to describe the removal of dead code. It helps reduce the size of JavaScript bundles, leading to faster load times and improved performance for web applications.

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 Rollup) 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.

7. **What is Hot Module Replacement?**

Hot Module Replacement (HMR) is a feature in modern front-end development tools that allows developers to update parts of a web application in real-time without

requiring a full page reload. It significantly speeds up the development process by making iterative changes faster and more efficient.

HMR allows you to see the changes you've made immediately, without losing the application's current state. It works as follows:

- Detecting Changes: The development server keeps track of the files in your project and watches for any changes to the source code, stylesheets, or other assets.

- Incremental Compilation: When you save a file, the tool recompiles only the modified part, not the entire project. This is possible because of the modular nature of modern JavaScript frameworks and build tools.

- Sending Updates: Once the incremental compilation is complete, the tool sends the updated modules to the browser.

- Applying Changes: The browser receives the updates and uses HMR to replace the outdated modules with the new ones, thus updating the application's view in real-time.

The result is that you can immediately see the changes you made without having to manually refresh the page and lose the current application state. HMR makes the development process more interactive and fluid, allowing developers to quickly iterate and experiment with their code.

It's worth noting that while HMR is beneficial during development, it is typically not used in production builds.

8. **List down your favourite 5 superpowers of Parcel and describe any 3 of them in your own words.**

Below are my top 5 superpowers of Parcel:

- Code Splitting
- HMR(Hot Module Replacement)
- Differential Bundling
- Tree Shaking
- Caching

Code Splitting: Code splitting is a technique that allows you to split your JavaScript code into multiple smaller bundles, which can be loaded on demand. This can improve the performance of your app, as users only have to download the code they actually need, rather than downloading your entire app upfront. Parcel makes code splitting easy by automatically taking care of it for us. It uses dynamic import() statements to split our code into separate bundles, which are loaded on demand when they are needed.

Differential Bundling: “Differential bundling” is the idea of shipping multiple versions of your code for different targets, and allowing the browser to choose the most optimal one to download. When we use a <script type="module"> element in an HTML file, and some of the browsers specified by the environment do not support ES modules natively, Parcel will automatically generate a <script nomodule> fallback as well.

HMR: HMR improves the development experience by updating modules in the browser at runtime without needing a whole page refresh. This means that application state can be retained as we change small things in our code.

9. **What is ‘.gitignore’? What should we add and not add into it?**

.gitignore is a special file used in Git version control to specify which files and directories should be ignored and not tracked by Git. When you're working on a project, you might have certain files or directories that you don't want to include in the Git repository, such as temporary files, build outputs, sensitive data, **files which can be re-generated** or editor-specific files. The .gitignore file allows you to list these items, preventing them from being accidentally committed to the repository. Some example:
- Temporary Files
- Build Outputs and Compiled Code
- User-Specific or Editor-Specific Files
- Sensitive Information (such as configuration files with sensitive data)
- Logs and Error Reports
- Generated Documentation
- Dependency Lock Files

10. **What is the difference between ‘package.json’ and ‘package-lock.json’ ?**

package.json is a manually maintained file that provides information about the project and its dependencies, while package-lock.json is automatically generated by npm to lock the versions of installed packages, ensuring consistency across different environments.

We can use '**npm ci**' to install the dependencies in the package-lock.json file'

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

package-lock.json file is automatically generated and is meant to ensure dependency consistency and reproducibility across different environments. Modifying it manually can introduce problems, conflicts, and security risks. Instead, we should manage your project's dependencies through the package.json file, and rely on npm install or npm ci to handle the generation of the lock file and the installation of dependencies.

12. What is 'node_modules'? Is it a good idea to push that to git?

node_modules is a directory that is created in a Node.js project when we use npm to install project dependencies. It contains all the packages and their dependencies that our project needs to run.
When we run npm install or npm ci, npm reads the package.json file in the project, fetches the specified packages from the npm registry, and store them into the node_modules directory so that our project can use them.
It's not a good idea to push node_modules to git as it can be very large and it can be regenerated easily using the package.json and package-lock.json files.

13. What is the 'dist' folder ?

The 'dist' folder is a directory that contains the production-ready version of your code. It is generated by a build process and contains optimized files that are ready to be deployed and used by others.

14. What is 'browserslist' ?

Browserslist is a tool that allows you to share target browsers and Node.js versions between different front-end tools. It is used by tools such as Autoprefixer, Babel, and others to ensure that your code is properly transpiled and optimized for the browsers you support. You can configure Browserslist by adding a browserslist property to your package.json file or by creating a .browserslistrc config file in the root of your project. This allows you to specify the browsers and Node.js versions that your project supports using queries like last 2 versions or > 0.5% in my stats.
Popular Transpilers: Babel, TypeScript.