1. **What is `NPM`?**

   NPM is a powerful tool for managing and installing JavaScript packages and dependencies.
   It's the default package manager for the Node.js environment.It is the Standard Repository for all the packages.

   Features of NPM:

   - **Package Installation:**
     Allows you to install and manage reusable JavaScript packages from the npm registry.

     These packages can include libraries, frameworks, tools, and more.

   - **Dependency Management:**
     Automatically handles the installation and updates of dependencies for your project.

     Ensures that your project uses the correct versions of packages and their dependencies.

   - **Command Line Tool:**
     The npm command-line tool is used to interact with the registry, install libraries, and manage dependencies.

   - **Package Publishing:**
     Enables you to share your own JavaScript packages with the community.

     You can publish your code to the npm registry, making it accessible to other developers.

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

   Parcel and Webpack are both powerful tools known as module bundlers. They play a crucial role in modern web development by taking various assets like JavaScript, CSS, images, and more, and bundling them into optimised files that browsers can understand.

   Need of Parcel or Webpack:

   Module Bundling:
   Combines multiple files into a smaller number of bundles, reducing the number of HTTP requests and improving load times.

   Code Splitting:

Breaks down large bundles into smaller chunks that can be loaded on demand, further optimising performance.

Asset Optimization:

Minifies and compresses assets like JavaScript, CSS, and images to reduce file sizes.

Transpilation:

Converts modern JavaScript syntax (e.g., ES6, TypeScript) into older, more widely supported syntax.

Hot Module Replacement (HMR):

Allows for live updates to your application without full page reloads, significantly speeding up development workflows.

3. **What is `.parcel-cache`**

.parcel-cache is a directory created by Parcel to store cached information about your project.

The .parcel-cache directory is a **cache folder** created by **Parcel** during the build process.

This cache significantly speeds up subsequent builds by allowing Parcel to reuse previously processed data.

The `.parcel-cache` directory is specific to your local development environment and should not be committed to version control systems like Git. This is because it contains machine-specific information that can vary across different setups.

4. **What is `npx` ?**

`npx` (Node Package eXecute) is a **command-line tool** that comes with **Node.js**. It is bundled with npm and is used to **execute JavaScript packages** directly from the **npm registry**, without installing them globally on your system.

npx automatically installs the specified version of the package, preventing version conflicts.

You can use packages for one-off tasks without permanently installing them. You can directly run scripts from GitHub repositories using npx.

5. **What is difference between `dependencies` vs `devDependencies`**

In a `package.json` file, you'll often encounter two key sections: `dependencies` and `devDependencies`. These sections are used to specify the packages your project relies on.

Dependencies

- **Production Dependencies:** These are packages that are essential for your application to run in production.
- **Included in the Final Bundle:** They are bundled with your application and deployed to the production environment.
- **Examples:** React, Vue, Angular, Express, and other core frameworks and libraries.

devDependencies

- **Development Dependencies:** These are packages used during development but are not required for the application to run in production.
- **Not Included in the Final Bundle:** They are not bundled with your application and are only used locally.
- **Examples:** Webpack, Babel, ESLint, Jest, and other build tools, linters, and testing frameworks.

Why the Distinction?

- **Performance:** By excluding unnecessary packages from the production bundle, you can reduce the size of your application and improve load times.
- **Security:** Minimising the number of dependencies in production reduces the attack surface, potentially improving security.
- **Consistency:** Keeping development dependencies separate ensures that your production environment remains clean and predictable.

6. **What is Tree Shaking?**
   Tree Shaking is a technique used to optimise JavaScript code by removing unused code.
   The bundler (like Webpack or Rollup) analyses your code to identify which parts are actually used.
   It removes any unused code, such as functions, variables, and modules, from the final bundle.
   By eliminating unused code, the final bundle size is significantly reduced.

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

   Hot Module Replacement (HMR) is a feature provided by modern JavaScript bundlers like **Webpack**, **Vite**, and **Parcel** that allows you to update modules in a running application **without requiring a full page reload**. It's primarily used during development to improve developer experience.

   How it works:

   1. **Module Detection:** When you make changes to a module, the development server detects the change.

2. **Module Update:** The changed module is recompiled and sent to the browser.
3. **Module Replacement:** The browser replaces the old module with the new one without reloading the entire page.

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

1. Create a local server.
2. Auto Refresh the page (Hot Module Replacement).
3. Caching
4. Tree Shaking
5. Minification

1. Caching
   Parcel caches the results of previous builds, which can significantly speed up subsequent builds, especially during development. This is particularly beneficial when working on large projects or making frequent changes.

2. Tree Shaking
   Tree Shaking is a technique used to optimise JavaScript code by removing unused code.

3. Minification
   Parcel automatically minifies your code, removing unnecessary characters and whitespace to reduce file size and improve performance.

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

.gitignore is a file in which we can add those files we don't want to put into production. Whatever we can regenerate doesn't need to be put in on git.

What to Add: node_modules, dist, .parcel- cache

What Not to Add: package.json, package-lock.json

10. What is the difference between `package.json` and `package-lock.json`

package.json:

- This file is intended to be edited by developers.
- It lists the packages your project depends on, along with their versions.
- Includes information like the project name, version, description, author, and script
- When you run `npm install`, npm uses the `package.json` file to install the specified dependencies.

package-lock.json:

- This file is automatically generated by npm.
- It records the exact versions of all installed dependencies and their transitive dependencies.
- This file guarantees that the same dependencies are installed across different environments
- While it's technically possible to edit this file, it's generally not recommended.

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

- The `package-lock.json` file is automatically generated by npm to lock down specific versions of dependencies.
- By manually editing it, you risk introducing inconsistencies between different environments (e.g., development, production, CI/CD).
- This can lead to unexpected behaviour and errors, especially when deploying to different environments.
- npm uses the `package-lock.json` file to ensure that the same dependencies are installed across different environments.
- Manually editing it can interfere with npm's ability to manage dependencies effectively.

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

Node_modules is a collection of dependencies.
It is like a database for dependencies we need in our project.

No, it's generally not a good idea to push the `node_modules` directory to Git.

13. What is the `dist` folder?

The dist folder, short for "distribution", is a directory that contains the final, optimised, and ready-to-deploy version of a project.

It typically includes:

- **Minified and bundled JavaScript and CSS files:** These files are compressed to reduce their size and improve loading times.
- **Optimised images:** Images are compressed to reduce file size without compromising quality.
- **HTML files:** These are the final HTML files that will be served to users.

14. What is `browserlists`

`browserlist` is used to specify the target browsers and their versions for which you want to compile and optimise your code.

```
{
  "name": "my-project",
  "version": "1.0.0",
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not dead",
    "not ie 11"
  ]
}
```