

## Assignment Ep-2

### 1. What is NPM?

NPM (Full form is not Node Package Manager) is the default package manager for Node.js. It provides a vast **online repository** of open-source JavaScript packages and tools. It helps you **install, manage, and version control** dependencies in your project using simple commands like `npm install`. It also manages package metadata via `package.json`.

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

Parcel and Webpack are **JavaScript bundlers**. They take all your JS, CSS, HTML, and asset files and bundle them into optimized files for production. They help make your React app fast and efficient by handling:

- **Minification** (removing unnecessary code),
- **Code splitting** and **lazy loading** (chunking),
- **Tree shaking** (removing unused code),
- **Caching** and **Hot Module Replacement (HMR)** for better development experience.
- Parcel is known for its **zero-config setup** and built-in support for HMR, differential bundling, and faster builds.

### 3. What is .parcel-cache?

`.parcel-cache` is a folder used by Parcel to **store intermediate build results**. When you make changes to your project, Parcel rebuilds **only the changed modules**, while reusing the rest from the cache. This speeds up the development build process significantly.

### 4. What is `npx` ?

`npx` is a command-line tool that comes with `npm` (version 5.2 and above) and is used to execute Node.js packages without installing them globally. It runs packages directly from the `node_modules/.bin` folder if they are installed locally, or downloads and runs them temporarily if they are not. For example, after installing Parcel using `npm install parcel`, you can run it using `npx parcel <file-name>` without needing to install it globally. This makes it convenient for running tools and scripts in a project-specific context.

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

**Dependencies** are packages required for your project to run in **production**, such as React, Axios, or Lodash. These are essential for the app's core functionality and are listed

under the "dependencies" section in package.json. In contrast, **devDependencies** are packages needed **only during development**, such as testing libraries, bundlers like Parcel or Webpack, linters, and formatters. They are listed under the "devDependencies" section and are not included in the production build.

## 6. What is Tree Shaking?

**Tree shaking** is a process used by bundlers like Parcel and Webpack to eliminate **unused code** from the final production build. It analyzes your application's **imported modules and functions**, and if any code is **not used anywhere**, it automatically removes (or "shakes") them off during the build process. This results in a **smaller, cleaner, and more optimized bundle**, improving load times and deployment efficiency.

## 7. What is Hot Module Replacement?

**Hot Module Replacement (HMR)** is a process used by bundlers like Parcel to update only the modules that have changed without refreshing the entire app. Bundlers continuously watch for file changes in your project, and when a module is modified, they rebuild just that module and inject the updated code into the running app. This selective replacement helps preserve the application's state by avoiding a full page reload, making development faster and smoother.

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

Tree Shaking, Hot Module Replacement (HMR), Dev Server Creation During Development, Minifying, Bundling

- **Dev Server Creation During Development:** Parcel automatically starts a local development server when you run your project. This server serves your app in the browser and refreshes it instantly when you make changes, providing a smooth development workflow without extra setup.
- **Minifying:** This process reduces the size of your JavaScript and CSS files by removing unnecessary white spaces, comments, and shortening variable names. This results in faster load times and better performance in production.
- **Bundling:** Parcel combines all your project files—JavaScript, CSS, images, etc.—into one or more optimized bundles. This makes it easier to load your app efficiently in browsers by reducing the number of requests and organizing your code smartly.

## 9. Here's a polished version of your explanation in one paragraph:

**.gitignore** is a file where you list files or folders that you want Git to ignore, meaning they won't be committed to your local repository or pushed to GitHub. Typically, files that can be regenerated—such as `node_modules`, `dist`, or `.parcel-cache`—should be included in `.gitignore` to keep the repository clean and avoid unnecessary bloat. Only

essential source code and configuration files should be tracked and pushed to GitHub, while everything else that can be recreated should be excluded using this file.

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

**package.json** is a configuration file that lists all the dependencies and packages required for a project, along with their approximate version ranges. It serves as the blueprint to install the necessary packages and generate the `node_modules` folder.

In contrast, **package-lock.json** records the exact versions of all dependencies and their sub-dependencies (transitive dependencies) that were installed, ensuring consistent installs across different environments. This file is crucial for project stability and should never be manually edited.

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

You should not modify `package-lock.json` because it precisely records the exact versions of all dependencies and their sub-dependencies used in your project to ensure consistent and reproducible builds. Manually changing it can cause version mismatches, break dependencies, or introduce bugs that are hard to track. Instead, let package managers like npm or yarn update it automatically when you install or update packages.

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

**node\_modules** is a folder that contains all the packages and dependencies installed via npm, based on the specifications in your `package.json` file. It is not advisable to push this folder to GitHub because it is often very large and can be easily regenerated using `npm install`. Instead, tracking `package.json` and `package-lock.json` is sufficient for others to install the exact same dependencies.

13. What is the `dist` folder?

The **dist** (short for "distribution") folder contains the final build output of your application after running a bundler like Parcel using a command such as `npx parcel build`. It includes the optimized, minified, and bundled code that is ready to be deployed to a production environment or served by a development server. This folder represents the actual code that runs in the browser.

14. What is `browserslist`?

`browserslist` is a configuration key typically found in the `package.json` file (or in a separate `.browserslistrc` file) where developers define the list of browsers and their versions that their app should support. Tools like Babel, Autoprefixer, and Parcel use this configuration to transpile code and add necessary polyfills to ensure compatibility across the specified browsers.

15. Read about: webpack, vite and parcel.

### Vite

- **Dev Server Speed:** Extremely fast due to native ES modules and on-demand file serving.
  - **Build Tool:** Uses Rollup under the hood for production builds.
  - **HMR:** Lightning-fast Hot Module Replacement.
  - **Zero Config:** Minimal setup needed; great DX (Developer Experience).
  - **Best For:** Modern frontend apps with fast iteration needs (like React, Vue, Svelte).
- 


### Parcel

- **Bundling:** Zero-config bundler with out-of-the-box support for many file types.
  - **Features:** Built-in HMR, caching, tree shaking, code splitting.
  - **Parallel Processing:** Uses worker threads to speed up bundling.
  - **Best For:** Beginners and quick prototypes, or when you want powerful features without complex config.
- 

### Webpack

- **Highly Configurable:** Can be tailored to any project setup with loaders and plugins.
  - **Ecosystem:** Massive plugin ecosystem and community.
  - **Mature:** Very stable, used in many large-scale production apps.
  - **Best For:** Complex applications that require fine-grained control.
- 

### Summary

Feature	Vite	Parcel	Webpack
Config	Minimal	Zero-config	Highly configurable
Dev Speed	 Fast (ESM-based)	Fast (caching, HMR)	Slower (static builds)

Feature	Vite	Parcel	Webpack
Learning Curve	Easy	Very Easy	Steeper
Customization	Moderate	Low	High
Build Tool	Rollup (prod)	Built-in bundler	Built-in bundler

#### 16. Read about: ^ - caret and ~ - tilde

In package.json, the caret (^) and tilde (~) symbols specify how version updates are handled. The caret ^ allows automatic updates to newer minor and patch versions that are backwards-compatible—for example, ^1.2.3 will update to any version below 2.0.0. The tilde ~ allows updates only to newer patch versions—for example, ~1.2.3 will update up to but not including 1.3.0. Using ^ is common for receiving safe feature updates, while ~ is preferred for stricter stability with minimal risk of breaking changes.