**What is NPM?**

**NPM (Node Package Manager)** is a package manager for JavaScript and is used to manage packages and dependencies in Node.js projects. It helps developers to:

* **Install packages** from the npm registry.
* **Manage dependencies** and their versions.
* **Run scripts** to automate tasks (like building, testing, or starting a server).

**Common Commands:**

* npm install <package>: Install a package.
* npm update: Update packages.
* npm run <script>: Run a script defined in package.json.

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

**Parcel and Webpack** are module bundlers used to bundle JavaScript, CSS, HTML, and other assets for web applications.

* **Parcel:**
  + **Description:** A zero-config bundler that automatically handles various tasks such as module resolution, asset bundling, and live reloading.
  + **Why We Need It:** Simplifies the bundling process with minimal configuration and built-in features like hot module replacement and fast builds.
* **Webpack:**
  + **Description:** A powerful, configurable module bundler that can handle complex bundling scenarios.
  + **Why We Need It:** Provides advanced features and plugins for code splitting, minification, and optimization, making it suitable for large-scale applications.

**Comparison:**

* **Parcel:** Easier to set up, less configuration, faster development builds.
* **Webpack:** More flexible and powerful, suited for complex builds, requires more configuration.

**What is .parcel-cache?**

* **Description:** A directory created by Parcel to store cache files for faster rebuilds.
* **Purpose:** Improves build performance by storing intermediate build results and avoiding redundant processing.

**What is npx?**

**npx** is a package runner tool that comes with npm (v5.2.0 and higher). It allows you to:

* **Run packages** without installing them globally.
* **Execute binaries** from the node\_modules/.bin directory.

**Example:**

bash

Copy code

npx create-react-app my-app

**What is the difference between dependencies vs devDependencies?**

* **dependencies:** Packages required for your application to run in production. These are essential for the app’s functionality.
  + **Example:** react, express, axios
* **devDependencies:** Packages required only during development and testing. These are not needed in the production environment.
  + **Example:** eslint, webpack, jest

**What is Tree Shaking?**

**Tree Shaking** is a technique used in modern JavaScript bundlers to remove unused code from the final bundle.

* **Purpose:** Reduces the size of the JavaScript bundle, improving performance by eliminating dead code.

**How It Works:** Analyzes the code to determine which parts are not used and excludes them from the final build.

**What is Hot Module Replacement (HMR)?**

**Hot Module Replacement (HMR)** is a feature in bundlers like Webpack and Parcel that allows you to:

* **Update Modules Without a Full Refresh:** Automatically reload only the changed parts of the application without losing the state.
* **Improve Development Speed:** Speeds up development by applying changes in real-time.

**Favorite 5 Superpowers of Parcel and Description of 3**

1. **Zero Configuration**
2. **Fast Builds**
3. **Hot Module Replacement**
4. **Automatic Code Splitting**
5. **Built-in Support for Various File Types**

**Descriptions:**

1. **Zero Configuration:** Parcel requires minimal setup to get started. You can begin a project without extensive configuration files, which simplifies the initial setup process and lets you focus on coding.
2. **Fast Builds:** Parcel uses advanced caching techniques to speed up build times. The .parcel-cache directory stores intermediate build results, making subsequent builds faster.
3. **Hot Module Replacement:** This feature updates only the changed modules without refreshing the entire page. It helps in maintaining the application state and speeds up the development process.

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

**.gitignore** is a file used to specify which files and directories Git should ignore.

* **Add to .gitignore:**
  + **Build Artifacts:** node\_modules/, dist/
  + **Configuration Files:** .env (contains sensitive data)
  + **IDE Files:** .vscode/, .idea/
  + **Logs:** \*.log
* **Do Not Add to .gitignore:**
  + **Source Code:** All the files that are essential for the project’s codebase.
  + **Configuration Files:** Files needed for setting up the project (e.g., package.json).

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

* **package.json:**
  + **Purpose:** Defines the metadata, dependencies, and scripts for the project.
  + **Editable:** You can modify this file manually to update dependencies, add scripts, etc.
* **package-lock.json:**
  + **Purpose:** Locks the versions of dependencies to ensure consistent installs across environments.
  + **Not Editable:** Automatically generated and updated by npm; manually editing it can cause issues.

**Why should I not modify package-lock.json?**

* **Consistency:** It ensures consistent dependency versions across different environments. Manual modifications can lead to discrepancies between different installations.

**What is node\_modules? Is it a good idea to push that on Git?**

* **node\_modules:** A directory where npm installs project dependencies.
* **Do Not Push to Git:** It’s not necessary to push node\_modules to a repository. Instead, you should use package.json and package-lock.json to manage dependencies. Pushing node\_modules increases the repository size and causes conflicts.

**What is the dist folder?**

* **Description:** The dist (distribution) folder contains the final build artifacts ready for deployment. It includes minified and optimized versions of your source code and assets.

**What is browserslist?**

**browserslist** is a configuration that specifies which browsers to support for your project. It is used by tools like Babel and Autoprefixer to compile code and add CSS prefixes accordingly.

**Example:**

json

Copy code

"browserslist": "> 1%, last 2 versions, Firefox ESR"

* **Purpose:** Ensures that your application is compatible with specified browser versions and helps in generating appropriate polyfills and prefixes.

**Read About Different Bundlers**

* [**Vite**](https://vitejs.dev/)**:** A modern, fast build tool that focuses on fast development server start times and fast hot module replacement.
* **Webpack:** A powerful and highly configurable module bundler for JavaScript applications.
* [**Parcel**](https://parceljs.org/)**:** A zero-config, fast bundler with built-in features for a simpler development experience.

**Read About: ^ - Caret and ~ - Tilde**

* **^ (Caret):** Allows updates to minor and patch versions but not major versions.
  + **Example:** "react": "^17.0.2" allows updates to 17.x.x but not 18.x.x.
* **~ (Tilde):** Allows updates to only the patch version.
  + **Example:** "react": "~17.0.2" allows updates to 17.0.x but not 17.1.0.

**Read About: Script Types in HTML (MDN Docs)**

* **<script> tag types**:
  + **text/javascript**: Default type for JavaScript files.
  + **module**: Used for ES Modules.
  + **text/babel**: For scripts using Babel’s JSX syntax (not recommended for production).

**MDN Docs:** [Script Types](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script#attr-type)

These explanations should give you a solid foundation for understanding these concepts and how they fit into web development with React and other technologies.

**Summary Table**

| **Term** | **Description** |
| --- | --- |
| **NPM** | Package manager for JavaScript, used for managing packages and running scripts. |
| **Parcel/Webpack** | Module bundlers for managing JavaScript, CSS, and other assets. Parcel is zero-config; Webpack is highly configurable. |
| **.parcel-cache** | Directory for caching build results in Parcel to improve build performance. |
| **npx** | Tool for running Node.js packages and binaries without global installation. |
| **dependencies vs devDependencies** | dependencies are for production; devDependencies are for development. |
| **Tree Shaking** | Optimization technique to remove unused code from the final bundle. |
| **Hot Module Replacement** | Updates only the changed parts of the application without a full page reload. |
| **Parcel Superpowers** | **Zero Configuration:** Minimal setup required. **Fast Builds:** Improved build times. **Hot Module Replacement:** Real-time updates during development. |
| **.gitignore** | Specifies files and directories Git should ignore. Include node\_modules, dist, .env. |
| **package.json vs package-lock.json** | package.json defines project metadata and dependencies; package-lock.json locks dependency versions for consistency. |
| **node\_modules** | Directory containing installed project dependencies. Do not push to Git. |
| **dist folder** | Contains the final build artifacts for deployment. |
| **browserslist** | Configuration for specifying which browsers to support in the project. |
| **Vite, Webpack, Parcel** | Vite is fast and modern. Webpack is highly configurable. Parcel is zero-config and fast. |
| **Caret (^) and Tilde (~)** | ^ allows minor and patch updates; ~ allows only patch updates. |
| **Script Types** | text/javascript for JavaScript, module for ES Modules, text/babel for JSX (not for production). |