**Chapter 2: Igniting Our App**

**✅ 1. What is NPM?**

* **NPM (Node Package Manager)** is a tool that comes with Node.js.
* It lets you **download, install, and manage packages/libraries** for your project.

📌 Example: npm install react

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

* **Parcel/Webpack** are **bundlers**.
* They take all your JS, CSS, images, etc. → optimize them → bundle into small files for the browser.

📌 Why?

* Faster loading
* Minification
* Compression
* Handle modern JS (ES6+) for older browsers

**✅ 3. What is .parcel-cache?**

* A hidden folder created by **Parcel**.
* Stores cached build files → so next build is **faster**.

**✅ 4. What is npx?**

* **npx** = Node Package eXecute.
* Runs a package **without installing globally**.

📌 Example:  
npx create-react-app myApp → runs directly without needing permanent install.

**✅ 5. Difference between dependencies vs devDependencies**

* **dependencies** → Packages needed **in production** (e.g., React, Lodash).
* 📌 Example in React:

react → To build UI

react-dom → To render UI to browser

axios → To call APIs

👉 If you remove these, the **app will break in production**.

* **devDependencies** → Packages needed **only in development** (e.g., Parcel, Jest).
* 📌 Example in React:

parcel or webpack → To bundle your code

jest → For testing

👉 If you remove these, the **app still works in production**, but you **can’t develop or build it properly**.

**✅ 6. What is Tree Shaking?**

* A process where **unused code is removed** during bundling → keeps bundle small.

📌 Example: If you import only one function from Lodash, unused ones won’t be bundled.

**✅ 7. What is Hot Module Replacement (HMR)?**

* A feature of bundlers like Parcel.
* Updates only the changed code in your app **without full reload**.

📌 Example: You change CSS → browser updates instantly without refreshing.

**✅ 8. Favorite 5 Superpowers of Parcel**

* 🚀 HMR (Hot Module Replacement)
* 📦 Bundling
* ⚡ Caching (faster builds with .parcel-cache)
* 🌍 Image & file optimization
* 🔀 Tree shaking

**Describe any 3:**

1. **HMR** → Instantly updates changes without full reload → faster dev cycle.
2. **Caching** → Uses .parcel-cache to make rebuilds faster.
3. **Tree Shaking** → Removes unused code, keeping bundles small.

**✅ 9. What is .gitignore? What should we add & not add?**

* .gitignore tells Git **which files/folders NOT to track**.

📌 We add:

* node\_modules/
* dist/
* .parcel-cache/
* .env (sensitive info like API keys)

📌 We don’t add: which are important and can’t generate again.

* Actual source code
* Config files (package.json, etc.)

**✅ 10. Difference between package.json and package-lock.json**

* **package.json** → Lists project details + dependencies (with version ranges).
* **package-lock.json** → Exact versions installed (ensures everyone has same setup).

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

* It’s **auto-generated**.
* Manually changing it may cause **version mismatches** → errors when others run the project.

**✅ 12. What is node\_modules? Should we push it to Git?**

* **node\_modules** is a folder where NPM stores all downloaded packages.
* ❌ Don’t push it to Git → It’s very heavy, and anyone can regenerate it using npm install.

**✅ 13. What is the dist folder?**

* dist = **Distribution folder**.
* Contains the **final bundled and optimized code** ready for deployment.

**✅ 14. What is Browserslist?**

* A config in package.json that defines **which browsers your app should support**.
* Bundlers use this to add/remove polyfills.

📌 Example:

A screen shot of a computer

AI-generated content may be incorrect.

**✅ 15. Different Bundlers**

* **Vite** → Super-fast (uses ESBuild).
* **Webpack** → Most popular, highly configurable.
* **Parcel** → Zero-config, beginner-friendly.

**✅ 16. ^ (caret) vs ~ (tilde) in package.json**

* ^1.2.3 → Updates **minor & patch versions** (1.x.x).
* ~1.2.3 → Updates **only patch versions** (1.2.x).

**✅ 17. Script Types in HTML (<script>)**

* **Default** → Blocks HTML parsing until script loads.
* **async** → Loads in parallel, executes immediately after load.
* **defer** → Loads in parallel, executes after HTML parsing (recommended for modern JS).

**18.** **✅ What is the dist/ folder?**

* dist stands for **distribution**.
* It is the folder that contains the **final, production-ready version of your app**.
* Created when you run a **build command** (npm run build).

**🔹 What happens during build?**

When you run the build, tools like **Parcel/Webpack/Vite**:

1. **Bundle** → Combine all your JS, CSS, images into fewer files.
2. **Minify** → Remove spaces, shorten variable names → smaller file size.
3. **Optimize** → Compress images, apply tree shaking (remove unused code).
4. **Polyfills** → Add extra code for older browsers if needed.
5. Save the result inside **dist/** folder.

**🔹 Why do we need dist/?**

Because **users don’t need your messy dev files (src/)**.  
They only need **fast, optimized files** to load in the browser.

👉 In interviews, the common question is:  
**“What is dist/? Should you push it to Git?”**

* Answer: dist/ is the production build folder, and ❌ **no, you should not push it** — it can always be generated again with npm run build.

**🔸 Analogy**

Think of src/ as **raw ingredients** 🥦🍅🍚 and dist/ as the **final cooked dish** 🍲 that you serve to users.

**19.** **✅ Difference between dist/ and node\_modules/**

**🔹 dist/**

* Stands for **distribution**.
* Contains the **final bundled and optimized code** (production build).
* Generated when you run:

npm run build

* Files are **small, minified, optimized** → ready to be deployed to a server.
* ❌ Should not be pushed to Git (because it can be regenerated anytime).

📌 Example contents:

dist/

index.html

main.ab12c.js

style.34dff.css

**🔹 node\_modules/**

* A folder where **all installed npm packages** are stored.
* Created when you run:

npm install

* Contains all libraries like **React, Lodash, Parcel, Axios**, and their dependencies.
* Usually **very large** (can be hundreds of MB).
* ❌ Should not be pushed to Git (because it can also be regenerated using npm install).

📌 Example contents:

node\_modules/

react/

react-dom/

parcel/

lodash/

**🔸 Simple Analogy**

* **node\_modules/** = **ingredients store** 🛒 (all raw ingredients/libraries you need).
* **src/** = **your recipe book + raw cooking** (your code).
* **dist/** = **final cooked dish** 🍲 (optimized app served to users).

👉 Interview Tip:  
If they ask **“Which folders should be in .gitignore?” - Which should not push to GIT**

* **node\_modules/**
* **dist/**
* **.parcel-cache/**
* **.env** (secrets like API keys)