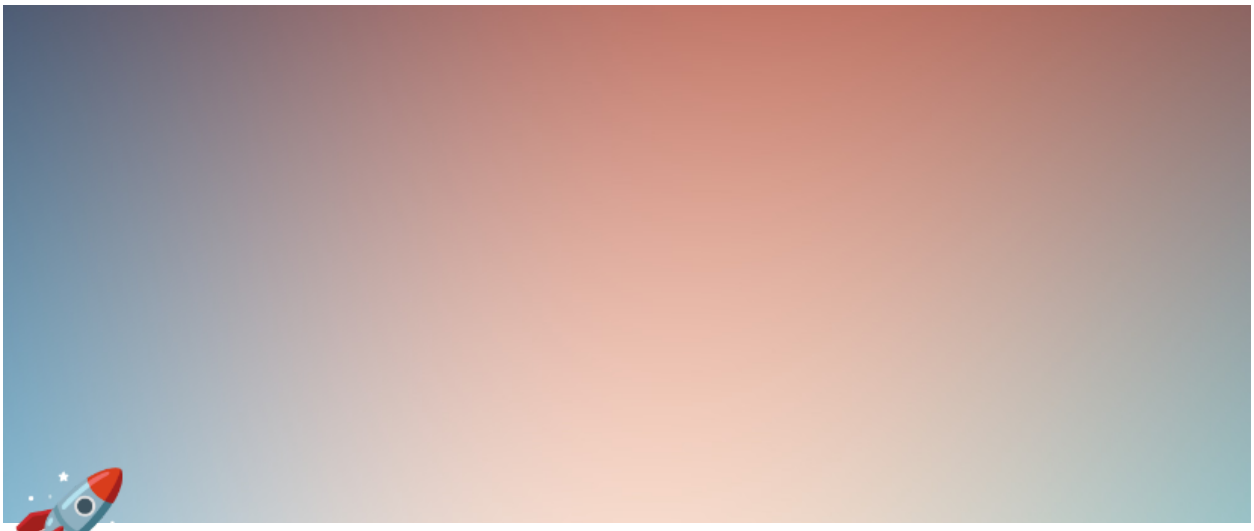


Igniting Our App! (Namaste-React)

💡 Please make sure to follow along with the whole ***"Namaste React"*** series, starting from Episode-1 and continuing through each subsequent episode. The notes are designed to provide detailed explanations of each concept along with examples to ensure thorough understanding. Each episode builds upon the knowledge gained from the previous ones, so starting from the beginning will give you a comprehensive understanding of React development.

💡 I've got a quick tip for you. To get the most out of these notes, it's a good idea to watch **Episode-2** first. Understanding what ***"Akshay"*** shares in the video will make these notes way easier to understand.

> **So far, here's what we have learned in the previous episode.**
>
> - We studied about Libraries, Frameworks, their differences.
> - We have also created Hello World! using HTML, JavaScript, and React.
> - We have also studies about what is Emmet, CORS (Cross Origin)

Igniting Our App.

Q) To make our app production ready what should we do?

- Minify our file (Remove console logs, bundle things up)
- Need a server to run things

📢 **NOTE: Minify –> Optimization –>Clean console –> Bundle**

* Bundlers:

- A bundler is a tool that bundles our app, packages our app so that it can be shipped to production.
- Examples of Bundlers:

- Webpack
- Vite
- Parcel

> 📢 **NOTE: In create-react-app, the bundler used is webpack.**

* Package Manager:

- Bundlers are packages. If we want to use a package in our code, we have to use a package manager.
- We use a package manager known as npm or yarn

* Configuring the Project:

```
npm init
```

- It creates a package.json file.
- Now to install parcel we will do:

```
npm install -D parcel
```

- Now we will get a package-lock.json file.

* package.json:

- Package.json file is a configuration for NPM. Whatever packages our project needs, we install those packages using npm install <packageName>.

- Once package installation is complete, their versions and configuration related information is stored as dependencies inside package.json file.

* package-lock.json:

- Package-lock.json locks the exact version of packages being used in the project.

Q) What is difference between package.json and package.lock.json?

- In package. json we have information about generic version of installed packages whereas in package.lock.json we have information about the specific or exact version of installed packages.

* node_modules:

- Which gets installed is like a database for the npm.
- Every dependency in node_module will have its package.json.
- Node modules are very heavy so we should always put this in git ignore.

📢 **NOTE: Never touch node_modules and package-lock.json**

* To ignite our app:

```
npx parcel index.html
```

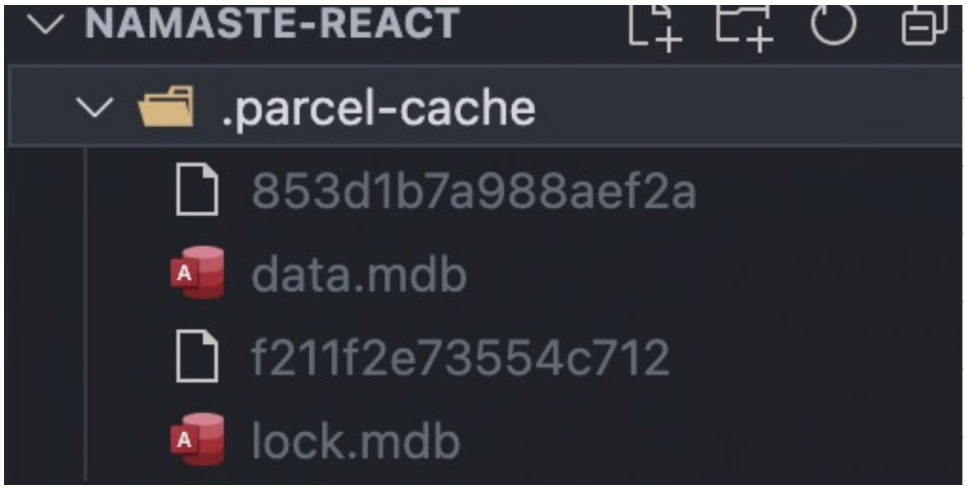
- npx means ‘execute using npm’
- index.html is the entry point

* Hot Module Replacement (HMR):

- It means that parcel will keep a track of all the files which you are updating.
- There is File Watcher Algorithm (written in C++). It keeps track of all the files which are changing realtime and it tells the server to reload.
- These are all done by PARCEL

* parcel-cache:

- Parcel caches code all the time.
- When we run the application, a build is created which takes some time in ms.
- If we make any code changes and save the application, another build will be triggered which might take even less time than the previous build.
- This reduction of time is due to parcel cache.
- Parcel immediately loads the code from the cache every time there is a subsequent build.
- On the very first build parcel creates a folder .parcel-cache where it stores the caches in binary codeformat.
- Parcel gives faster build, faster developer experience because of caching.

- 

* dist:

- It keeps the files minified for us.
- When bundler builds the app, the build goes into a folder called dist.
- The `/dist` folder contains the minimized and optimised version the source code.
- Along with the minified code, the /dist folder also comprises of all the compiled modules that may or may not be used with other systems.
- When we run command:

```
npx parcel index.html
```

- This will create a faster development version of our project and serves it on the server.
- When I tell parcel to make a production build:

```
npx parcel build index.html
```

- It creates a lot of things, minify your file.
- And the parcel will build all the production files to the dist folder.

* Parcel features at a glance:

- Hot Module Replacement (HMR)
- File Watcher Algorithm - C++
- Bundling
- Minify Code
- Cleaning our code
- Dev and production build

- Super fast build algorithm
- Image Optimization
- Caching while development
- Compression
- Compatible with older browser versions
- Https on dev
- Image Optimization
- Port No
- Consistency Hashing Algorithm
- Zero Config
- Tree Shaking

* Transitive Dependencies :

- We have our package manager which takes care of our transitive dependencies of our code.
- If we’ve to build a production ready app which uses all optimisations (like minify, bundling, compression, etc), we need to do all these.
- But we can’t do this alone, we need some dependencies on it. Those dependencies are also dependent on other dependencies.

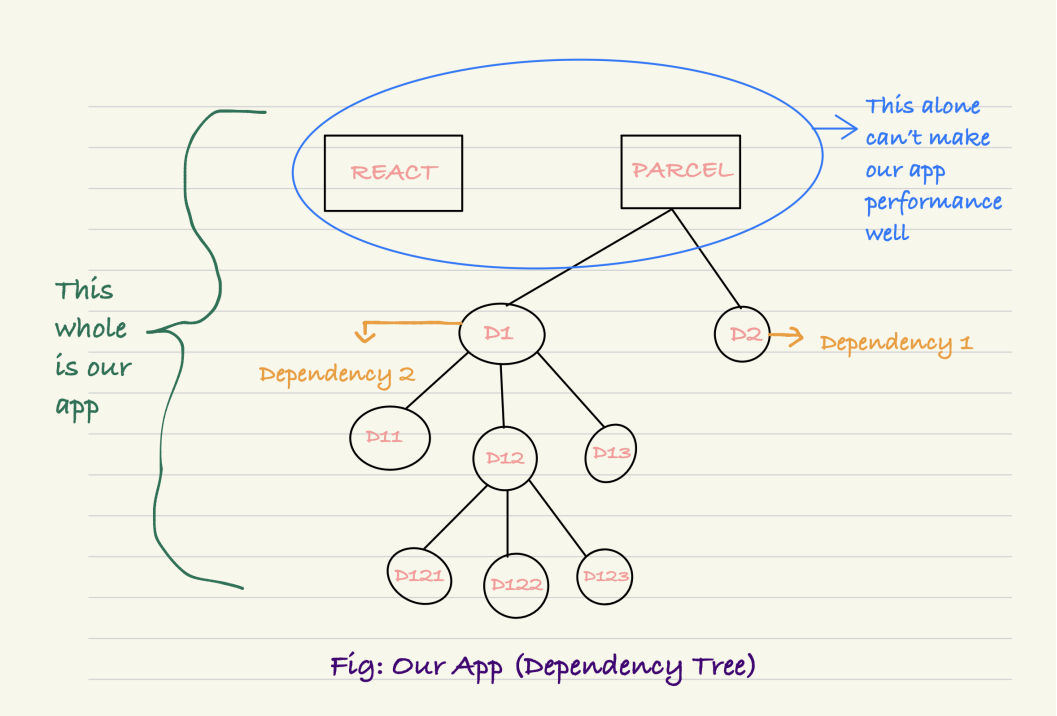
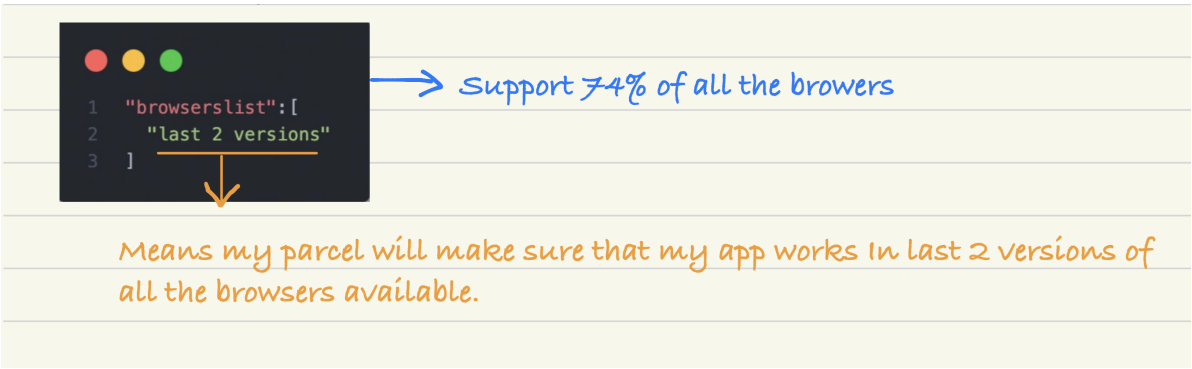


Fig: Our App (Dependency Tree)

* Browserslist:

- Browserslist is a tool that specifies which browsers should be supported/compatible in your frontend app.
- It makes our code compatible for a lot of browsers.
- In package.json file do:



* Tree Shaking:

- Tree shaking is a process of removing the unwanted code that we do not use while developing the application.
- In computing, tree shaking is a dead code elimination technique that is applied when optimizing code.