Namaste React Course by Akshay Saini

Chapter 02 - Igniting our App

Q: What is `NPM`?

A: It is a tool used for package management and the default package manager for Node projects. NPM is installed when NodeJS is installed on a machine. It comes with a command-line interface (CLI) used to interact with the online database of NPM. This database is called the NPM Registry, and it hosts public and private 'packages.' To add or update packages, we use the NPM CLI to interact with this database.

- `npm` alternative is `yarn`

How to initialize `npm`?

```
npm init
```

`npm init -y` can be used to skip the setup step, `npm` takes care of it and creates the `package.json` json file automatically , but without configurations.

Q: What is `Parcel/Webpack`? Why do we need it?

A: Parcel/Webpack is type of a web application bundler used for development and productions purposes or power our application with different type functionalities and features.
It offers blazing fast performance utilizing multicore processing, and requires zero configuration. Parcel can take any type of file as an entry point, but an HTML or JavaScript file is a good place to start.
Parcel/Webpack are type of bundlers that we use to power our application with different type functionalities and features.

Parcel Features:

- HMR (Hot Module Replacement) - parcel keeps track of file changes via file watcher algorithm and renders the changes in the files
- File watcher algorithm - made with C++
- Minification
- Cleaning our code
- DEV and production Build
- Super fast building algorithm
- Image optimization
- Caching while development
- Compresses
- Compatible with older version of browser
- HTTPS in dev
- Port Number
- Consistent hashing algorithm
- Zero Configuration
- Automatic code splitting

installation commands:

- Install:

```
npm install -D parcel
```

`-D` is used for development and as a development dependency.

- Parcel Commands :
  - For development build:

  ```
  npx parcel <entry_point>
  ```

  - For production build :

  ```
  npx parcel build <entry_point>
  ```

Q: What is `.parcel-cache`

A: `.parcel-cache` is used by parcel(bundler) to reduce the building time.
It stores information about your project when parcel builds it, so that when it rebuilds, it doesn't have to re-parse and re-analyze everything from scratch. It's a key reason why parcel can be so fast in development mode.

Q: What is `npx`?

A: npx is a tool that is used to execute the packages. It comes with the npm, when you installed npm above 5.2.0 version then automatically npx will installed. It is an npm package runner that can execute any package that you want from the npm registry without even installing that package.

Q: What is difference between `dependencies` vs `devDependencies`?

A: Dependencies should contain library and framework in which your app is built on, needs to function effectively. such as Vue, React, Angular, Express, JQuery and etc.
DevDependencies should contain modules/packages a developer needs during development. such as, parcel, webpack, vite, mocha.
These packages are necessary only while you are developing your project, not necessary on production.
To save a dependency as a devDependency on installation we need to do,

```
npm install --save-dev
```

instead of just,

```
npm install --save
```

Q: What is Tree Shaking?

A: Tree shaking is 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.

Q: What is Hot Module Replacement?

A: Hot Module Replacement (HMR) exchanges, adds, or removes modules while an application is running, without a full reload. This can significantly speed up development in a few ways: Retain application state which is lost during a full reload.

Q: List down your favorite 5 superpowers of Parcel and describe any 3 of them in your own words.

A: 5 superpowers of Parcel:

- HMR (Hot Module Replacement) - adds, or removes modules while an application is running, without a full reload.
- File watcher algorithm - File Watchers monitor directories on the file system and perform specific actions when desired files appear.
- Minification - Minification is the process of minimizing code and markup in your web pages and script files.
- Image optimization
- Caching while development

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

A: The .gitignore file is a text file that tells Git which files or folders to ignore in a project during commit to the repository.
The types of files you should consider adding to a .gitignore file are any files that do not need to get committed. for example, For security, the security key files and API keys should get added to the gitignore.
package-lock.json should not add into your .gitignore file.

The entries in this file can also follow a matching pattern.

```
* is used as a wildcard match
/ is used to ignore pathnames relative to the .gitignore file
# is used to add comments to a .gitignore file
```

This is an example of what the .gitignore file could look like:

```
# Ignore Mac system files
.DS_store

# Ignore node_modules folder
node_modules

# Ignore all text files
*.txt

# Ignore files related to API keys
.env

# Ignore SASS config files
.sass-cache
```

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

A: `package.json`:

- this file is mandatory for every project
- It contains basic information about the project
- Application name/version/scripts

`package-lock.json`:

- This file is automatically generated for those operations where npm modifies either the node_module tree or package-json.
- It is generated after an npm install
- It allows future devs & automated systems to download the same dependencies as the project.
- it also allows to go back to the past version of the dependencies without actual ‘committing the node_modules folder.
- It records the same version of the installed packages which allows to reinstall them. Future installs wll be capable of building identical description tree.

~ or ^ in `package.json` file :
These are used with the versions of the package installed.

For example in `package.json` file:

```
"dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  }
```

- ~ : “Approximately equivalent to version”, will update you to all future patch versions, without incrementing the minor version.
- ^ : “Compatible with version”, will update you to all future minor/patch versions, without incrementing the major version.

If none of them is present , that means only the version specified in `package.json` file is used in the development.

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

A: `package-lock.json` file contains the information about the dependencies and their versions used in the project. Deleting it would cause dependencies issues in the production environment. So don't modify it, It's being handled automatically by NPM.

Q: What is `node_modules` ? Is it a good idea to push that on git?

A: `node_modules` folder like a cache for the external modules that your project depends upon. When you npm install them, they are downloaded from the web and copied into the node_modules folder and Nodejs is trained to look for them there when you import them (without a specific path).
Don't push `node_modules`in github because it contains lots of files(more than 100 MB), it will cost you memory space.

Q: What is the `dist` folder?

A: The /dist folder contains the minimized version of the source code. The code present in the /dist folder is actually the code which is used on production web applications. 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.

Q: What is `browserslist`?

A: Browserslist is a tool that allows specifying which browsers should be supported in your frontend app by specifying "queries" in a config file. It's used by frameworks/libraries such as React, Angular and Vue, but it's not limited to them.