**Episode 2: Igniting our App**

**Theory:**

1. What is ‘NPM’?

Npm is the World’s largest Software Registry. This registry contains over 8,00,000 code packages. Open-source developers use npm to share software. Many organizations also use npm to manage private development.

These modules are configurable to support a wide variety of use cases. Most commonly, it is used to publish, install, discover and develop node programs.

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

ParcelJS describes itself as a blazingly fast, zero configuration web application bundler. Parcel is based on the premise that building web applications with the help of Javascript bundlers could be easier. Webpack provides us greater configurability but there may be several use cases where we don’t require that level of configuration.

Parcel automatically tracks all the files, configurations, dependencies, devDependencies etc that are involved in your build and granularly indicates the cache when something changes. It integrates with low level operating system APIs to determine what files have changed in milliseconds, no matter the size of the project.

Webpack in react is a module bundler that is commonly used with React to bundle and manage dependencies. It takes all of the individual Javascript files and other assets in a project such as Images and CSS and combines them into a single bundle that can be loaded by the browser.

3. What is ‘.parcel-cache’

‘.parcel-cache’ is a directory generated by Parcel Bundler, serving as a cache for storing intermediate build results. It enhances the development workflow by speeding up the subsequent builds and optimizing resource usage.

4. What is ‘npx’?

‘npx’ stands for ‘Node Package Execute’ and it is included with npm. If you install above 5.2.0, then npx will be installed automatically. It is a npm package runner which can execute any package from the npm registry without ever installing it.

We can basically say that, NPX is a Tool for executing Node packages.

5. What is difference between ‘dependencies’ vs ‘devDependencies’

‘dependencies’: Packages required by your application in production

‘devDependencies’: Packages that are only needed for local development and testing.

‘peerDependencies’: They are used to specify that our package is compatible with a specific version of npm package. To add a peer Dependency, we will manually need to modify the package.json file of our Project.

‘dependencies’ are installed Transitively. if A requires B, and B requires C, then C gets installed, otherwise, B could not work, and neither would A.

‘devDependencies’ is not installed Transitively. E.g. we don't need to test B to test A, so B's testing dependencies can be left out.

6. What is Tree Shaking?

Tree Shaking is a commonly used term within Javascript context to describe the removal of dead node. If basically relies on the import and export statements to detect if code modules are exported and imported for use between Javascript files.

The dead code or unused modules from a bundle are eliminated during the build process. It is basically a part of the bundling process in Javascript based development.

7. What is Hot Module Replacement?

Hot Module Replacement (HMR) is a powerful feature designed to enhance the development process by providing real-time updates to modules within a running application.

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

My 5 favourite superpowers of Parcel are as follows:

1. Local Server

2. File Watching Algorithm

Parcel uses a very fast watcher written in C++ that integrates with low level file watching of each operating system. Based on the events and metadata from these files, Parcel determines which files need to be rebuilt.

3. Caching – Faster Builds

Parcel caches everything it builds in disk. If we restart the dev server, Parcel will only build files that have changed since the time it last ran. Parcel automatically tracks all of the files, configuration, plugins, dev dependencies that are involved in our build and indicates the cache when something changes. The cache is stored in .parcel-cache folder.

4. Diagnostic

Parcel displays us beautiful diagnostics in our terminal and in the browser when we make any errors in the code or configuration. Every error includes a syntax highlighted of exact location where the error occurred, along with the hints about how to fix them.

5. Image Optimization

9. What is ‘.gitignore’? What should we add and not add into it?

A .gitignore file specifies intentionally untracked files that Git should ignore. Files which are already tracked by Git are not affected.

We can say that .gitignore file tells Git which files to ignore when committing our project to GitHub repository. .gitignore is located in the root directory of our project. .gitignore is itself a plain text document.

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

The package.json focuses on project metadata and specifying the desired version of dependencies, while the package-lock.json file ensures deterministic installations by locking the exact versions of dependencies and their dependencies.

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

Lock files should not be modified unless we are actively updating the packages required. It seems that sometimes, npm decides to update the package-lock.json file when we run an npm install. It does this to seemingly keep dependencies up-to-date without having to run npm update separately.

Package-lock.json is automatically generated for any operations where npm modifies either the node\_modules tree or package.json. It describes the exact tree that was generated, such that subsequent installs are able to generate identical trees, regardless of intermediate dependency updates.

12. What is ‘node\_modules?’ Is it good idea to push that on git?

The node\_modules contains every installed dependency for our Project. In most cases, we should not commit this folder to our version controlled repository. As we install more dependencies, the size of this folder will keep on increasing.

13. What is ‘dist’ folder?

If we are using something that bundles our code, then this is where our code gets bundled to for use in its ‘production’ environment. This is most often used for UI apps, but with things like typescript and es6 being used with webpack, this can be used on nodejs as well.

It basically stands for distribution and in its minified or concatenated version it has been used on production sites.

/dist stands for distributable.

Ideally it is considered a good practice, to clean the /dist folder before each build.

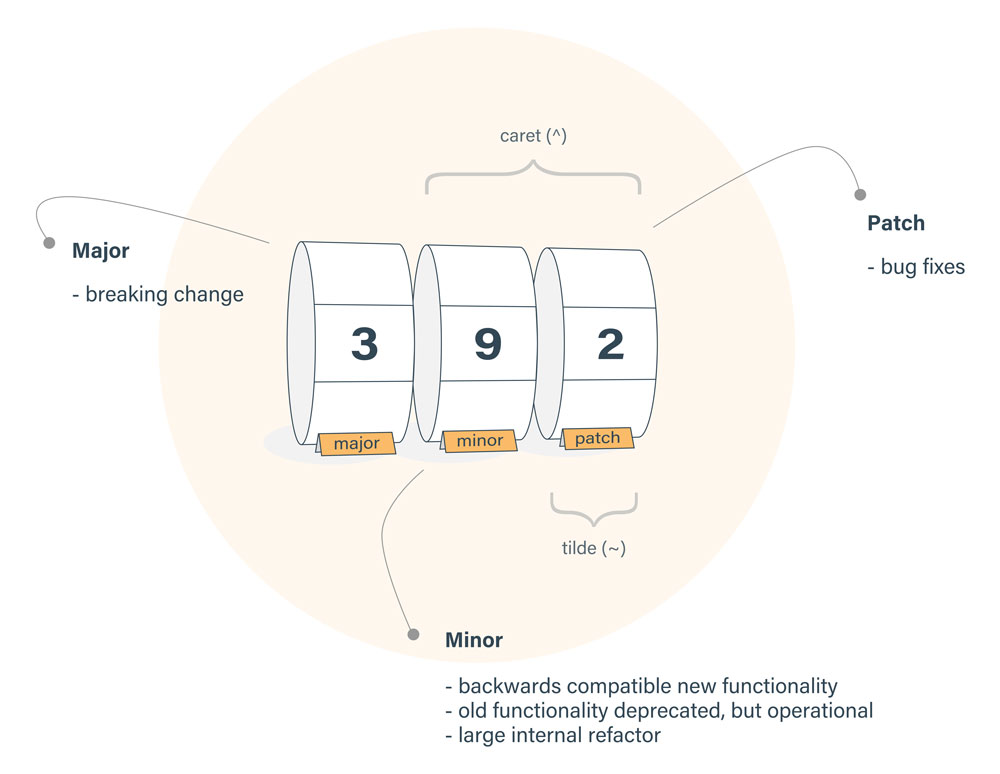
14. What is ‘browserlists’. Read about diff bundlers: vite, webpack, parcel

Browserlists is a configuration file that determines which browsers your project should support. It is used by many tools like Autoprefixer, Babel and Stylelint, to automatically add vendor prefixes and polyfills to our CSS and Javascript code.

Browserlists can specify which browsers our applications can run in. It provides a configuration for specifying browsers range. Browserlists has become a standard in the Industry and it is used by libraries like Autoprefixer, Babel, ESLint, PostCSS, SWC and Webpack.

15. Read about: ^ caret and ~ tilda

Caret(^) is less safer to use than Tilde(~) for production application, because here minor feature will also update automatically. Caret(^) is mainly used to automatic update in package for minor and bug fixes both.



~version: Approximately equal to version, it will update all future patch versions without incrementing the minor versions. ~1.2.3 will use releases from 1.2.3 to < 1.3.0.

^version: Compatible with version, it will update all the future minor/patch versions without incrementing the major version. ^1.2.3 will use releases from 1.2.3 to <2.0.0.

16. Read about Script types in html (MDN docs)

The type attribute specifies the type of the script. The type attribute identifies the content between <script> and </script> tags. The value of type will be one among the following:

Attribute is not set by default. It basically is an empty string or a Javascript MIME type. It indicates that the script is a “classic script” containing Javascript code. We should basically omit the attribute if the script refers to Javascript code rather than specify a Javascript MIME type.

1. Importmap

This value indicates that the body of the element contains an import map. The import map is a JSON object that developers can use to control how the browsers receives module specifiers when importing Javascript modules.

2. Module

This value causes the code to be treated as a Javascript module. The processing of the script contents is deferred. The charset and defer attribute have no effect. So basically unlike “classic scripts”, module scripts require the use of CORS protocol for cross-origin fetching.

3. speculationrules

This value indicates that the body of the element includes speculation rules. Speculation rules take the form of a JSON object that determine what resources should be prefetched or prerendered by the browser. This is part of the Speculation Rules API.

**Code:**

In your existing Project

1. Initialize ‘npm’ into your repo

2. Install ‘react’ and ‘react-dom’

3. Remove CDN links of react

4. Install Parcel

5. Ignite your app with parcel

6. Add scripts for “start” and “build” with parcel commands

7. Add .gitignore file

8. Add ‘browserlists’

9. Build a production version of your code using ‘parcel build’

**References:**

1. Creating your own create-react-app

2. Parcel Documentation

3. Parcel on Production

4. Browserlists: <https://browserslist.dev/>