**React Notes**

**Igniting our App**

**Bundlers:**

* Parcel, vite, webpack
* Parcel comes as a node package
* npm install -D parcel
* dev dependency used only during development.
* normal dependency used during production.

**Caret and tilde:**

* ^ - caret and ~ - tilde
* ^ - parcel will automatically upgrade minor version.
* ~ - parcel will automatically upgrade major version.

**Difference between package and package-lock.json:**

* package.json – configuration for npm, keeps track of what version is of a package is installed in your system. If there is ^ sign install any version of the package and if there is a minor upgrade for that package, install that version as well.
* packag-lock.json – keeps a track of exact version, that is installed. Suppose today a 2.8.5 version is released, package.json can stay as it is ( will show 2.8.3) , but package-lock.json, locks the version and keeps record of it.
* Integrity: it is basically a hash. Hash is present to verify whatever version in my machine is deployed to production.
* Package.json and package-lock.json should be pushed to git.

**Node modules:**

* Contains all the code that we fetch from npm.
* Fetches all the codes of all the system into your system.
* Node modules is a collection of dependencies.
* No need to put node modules to git. You can include it in .gitignore, because we have package.json and package-lock.json we can recreate the node modules folder via npm install.

**Transitive dependencies: Dependency Tree**

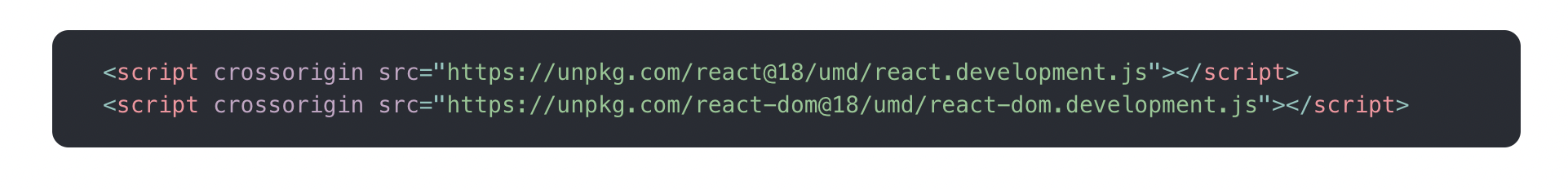
* When you install a package in Node. js using npm or Yarn, that package might depend on other packages. These other packages are called transitive dependencies.

**Ignite Our App:**

* npx parcel index.html
* npx means executing a package
* npm means install a package
* parcel builds our app.
* parcel has created a server for us ( <http://localhost:1234>)

**React and ReactDOM:**

* CDN links are not a good way to bring react and reactDOM into our application.



* Because fetching react from CDN is a costly operation. It will make a network call to unpkg and it will get react from unpkg, If we have react in node modules, we can easily use it.
* When version changes, we will have to change the CDN url accordingly. npm install will usually install latest version of react.
* **npm install react**
* **npm install react-dom or npm I react-dom**

**@parcel/transformer-js: Browser scripts cannot have imports or exports.**

* Treats App.js in index.html as normal javascript file.
* App.js has import specified for react and reat-Dom.
* Normal javascript do not understand/can not have import.
* We need to specify that it is a module.

**<script type="module" src="./App.js"></script>**

**Parcel:**

* Parcel is a beast.
* Dev build
* Local server
* HMR (Hot Module Replacement)
* File watching Algorithm - written in C++
* Caching - Faster Builds
* Image Optimization
* Minification
* Bundling
* Compress
* Consistent Hashing
* Code splitting
* Deferential Bundling – to support older browser – ex. If we have type module, parcel will generate a nomodule fallback for older browsers.
* Diagnostic
* Better Error Handling.
* https support – npx parcel index.html –https
* Tree Shaking – remove unused code
* Different dev and prod bundles.
* Prod build command:   
  
* This will throw error, because package.json has main as App.js, Remove that line.
* Production build takes more time.
* Parcel-cache and dist folder can be regenerated. No need to push to git
* Refer Parcel documentation – parclejs.org

Browserslist:

* To support different browsers and browser versions.

**Laying the foundation**

Scripts:

* You can write scripts in package.json to run the commands.
* Eg:   
  “scritps”: {

“start”: “parcel index.html”

“build”: “parcel build index.html”

}

* npm run { name of th script } : will execute the respective script
* npm run start / npm run build