Igniting Our App

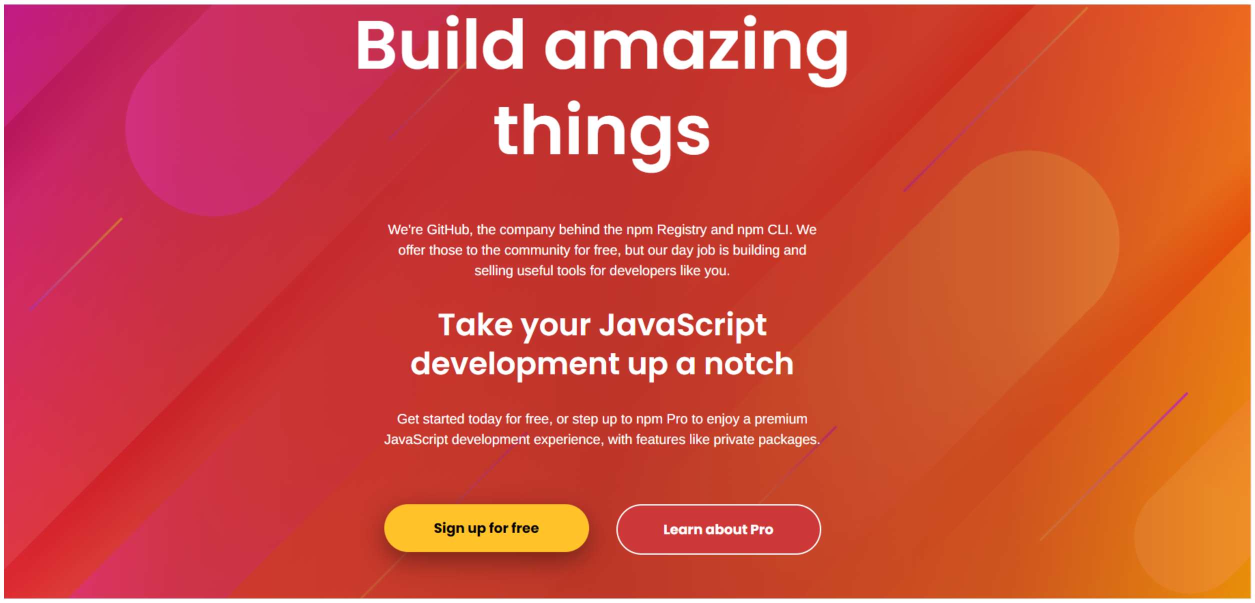
It is not just the React that make the website fast , there are different packages that help react making our App fast.

For getting installing these Packages we need to get familar with **"NPM".**

NPM : Node Package Manager. People say this is the actual full form of NPM but that's not true NPM does'nt stand for anything.

Suprised 😄!!

If you go to NPM website



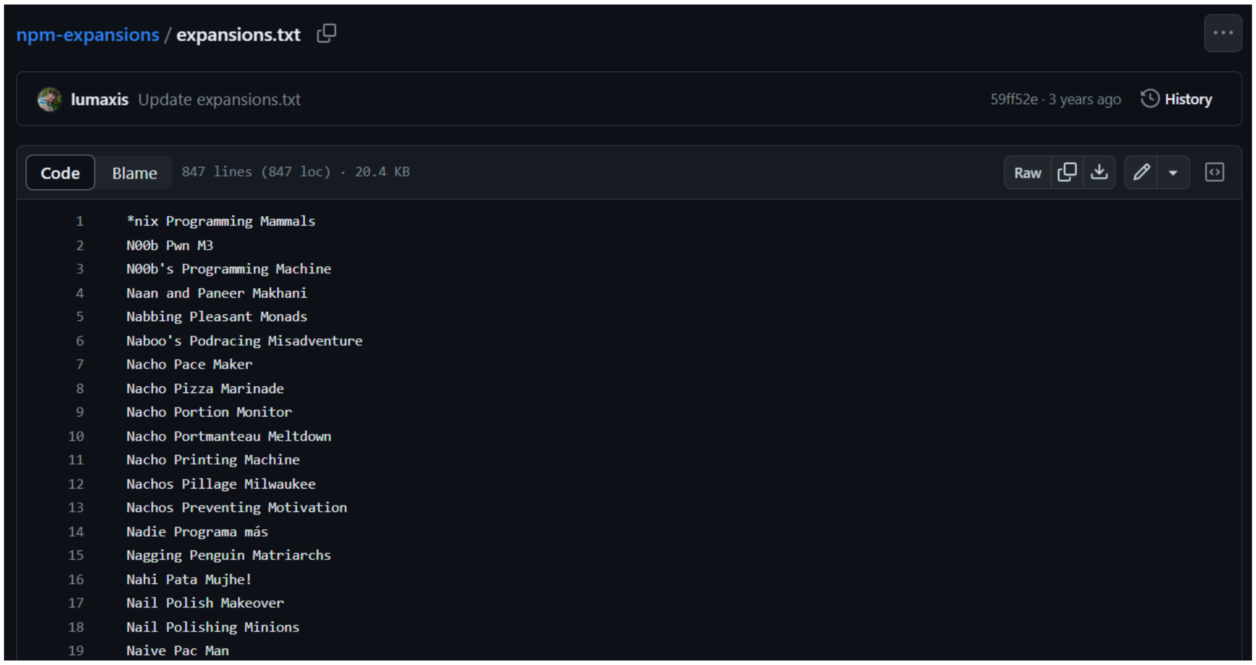
Build amazing things

We're GitHub, the company behind the npm Registry and npm CLI. We offer those to the community for free, but our day job is building and selling useful tools for developers like you.

Take your JavaScript development up a notch

Get started today for free, or step up to npm Pro to enjoy a premium JavaScript development experience, with features like private packages.

You will see there is no name for **NPM**, people has given different names to NPM. If you go to "npm expansion" you will find that people has given different names to NPM


npm-expansions / expansions.txt
lumaxis Update expansions.txt
59ff52e · 3 years ago
History
Code
Blame
847 lines (847 loc) · 20.4 KB
Raw

```
*nix Programming Mammals
N00b Pwn M3
N00b's Programming Machine
Naan and Paneer Makhani
Nabbing Pleasant Monads
Naboo's Podracing Misadventure
Nacho Pace Maker
Nacho Pizza Marinade
Nacho Portion Monitor
Nacho Portmanteau Meltdown
Nacho Printing Machine
Nachos Pillage Milwaukee
Nachos Preventing Motivation
Nadie Programa más
Nagging Penguin Matriarchs
Nahi Pata Mujhe!
Nail Polish Makeover
Nail Polishing Minions
Naive Pac Man
```

So, What is NPM ? 🤔

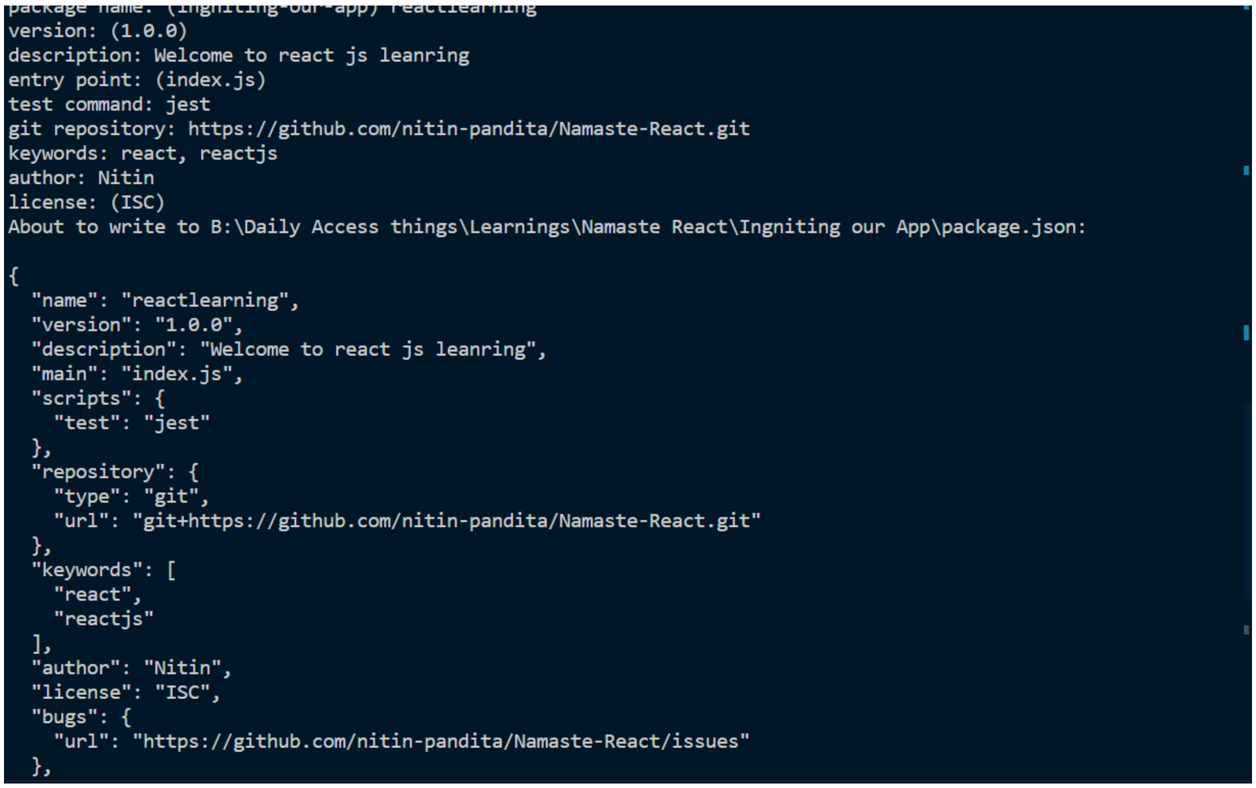
> npm is a package manager for the JavaScript programming language maintained by npm

So, how can we initialized “npm” ?

We can use:

```
npm init
```

Once we have run this command in our terminal we will be asked some questions



```
package name: (ingniting-our-app) reactlearning
version: (1.0.0)
description: Welcome to react js leanring
entry point: (index.js)
test command: jest
git repository: https://github.com/nitin-pandita/Namaste-React.git
keywords: react, reactjs
author: Nitin
license: (ISC)
About to write to B:\Daily Access things\Learnings\Namaste React\Ingniting our App\package.json:

{
  "name": "reactlearning",
  "version": "1.0.0",
  "description": "Welcome to react js leanring",
  "main": "index.js",
  "scripts": {
    "test": "jest"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/nitin-pandita/Namaste-React.git"
  },
  "keywords": [
    "react",
    "reactjs"
  ],
  "author": "Nitin",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/nitin-pandita/Namaste-React/issues"
  },
```

After answering all the question we will be getting a “json” file

In my case it was:

```
{
  "name": "reactlearning",
  "version": "1.0.0",
  "description": "Welcome to react js leanring",
  "main": "index.js",
  "scripts": {
    "test": "jest"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/nitin-pandita/Namaste-React.
  },
  "keywords": [
    "react",
    "reactjs"
  ],
  "author": "Nitin",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/nitin-pandita/Namaste-React/issu
  },
  "homepage": "https://github.com/nitin-pandita/Namaste-React#r
}
```

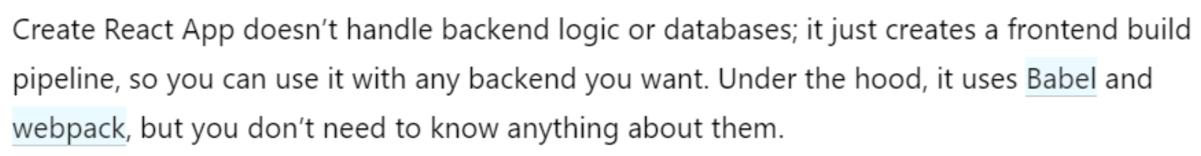
> Package.json is the configuration of NPM

NPM take care of:

- Version of the package with the help of Package.json

Q: What is bundler ?

Bundler is used to bundle the whole app. React uses webpack bundler to bundle the whole app.


Create React App doesn't handle backend logic or databases; it just creates a frontend build pipeline, so you can use it with any backend you want. Under the hood, it uses Babel and webpack, but you don't need to know anything about them.

Installing Dependencies :

For now we will be using "parcel"

```
npm install -D parcel
```

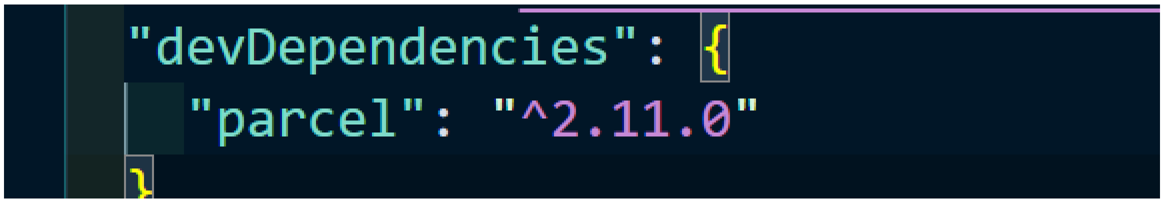
npm : package manager

install : to install the package

-D : There are two types of dependency : Dev Dependency and Normal Dependency

parcel : Is a bundler that we are installing and we are installing it as a Dev dependency by saying -D

After running this command we will see a change in **Package.json** file.



```
"devDependencies": {
  "parcel": "^2.11.0"
}
```

Now you might be wondering what is this “ ^” used for:

^ : is called carat

we can also have “~2.11.0” is some cases

Q: What is the difference between ^ (carat) and ~ (tilda) ?

Ans: ^ : **Approximately equivalent to version. If tomorrow a new version will come, and if we have used carat, the parcel will automatically update its version. It will install minor versions, which is safe.**

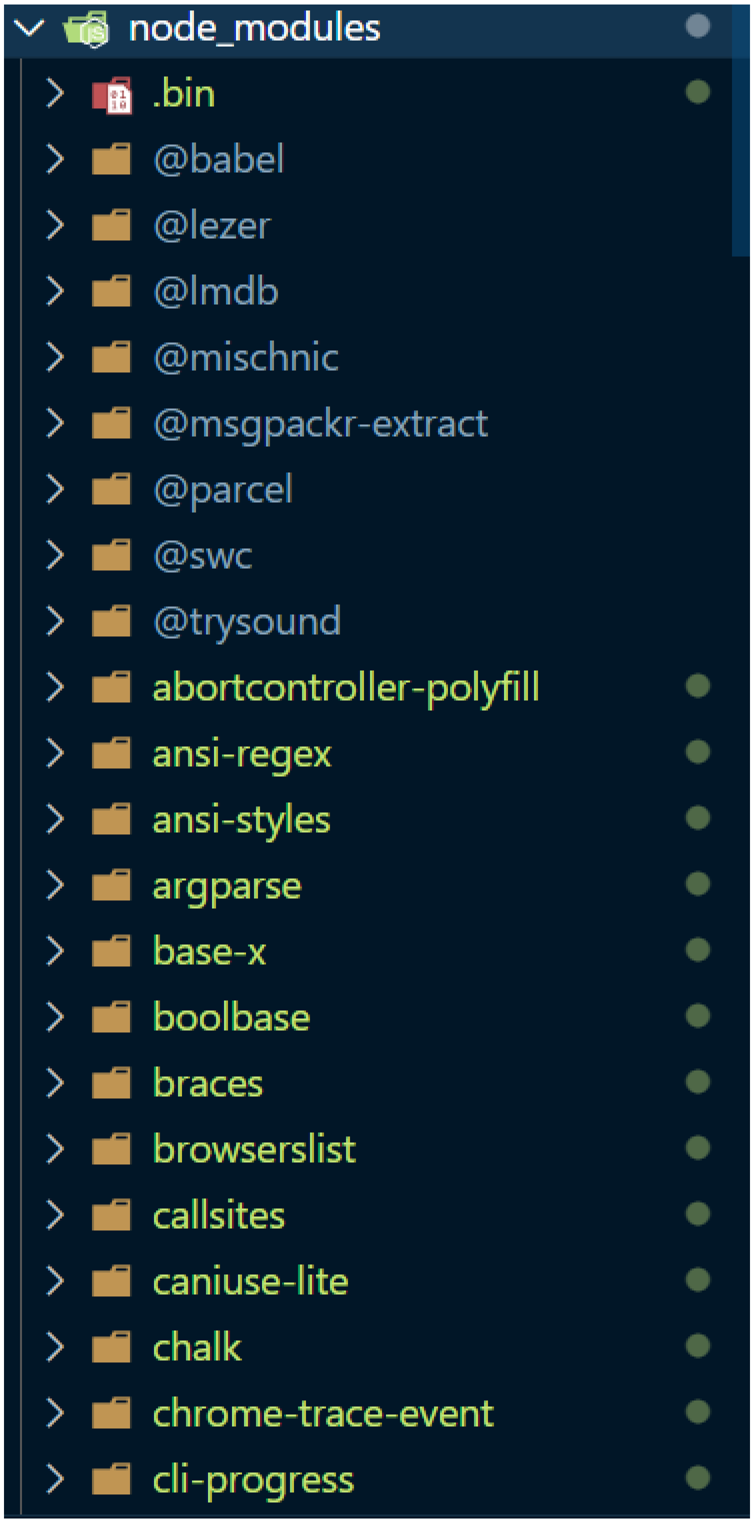
~ : **Compatible with version** . Now if we have used “**Tilda” it will install the major version automatically**

While installing the “parasl” you will incounter that we have another file called “Package.lock.json”

Q: Why this file was installed , What is the need of this file, Where it is used ?

Ans: “**Package.lock.json**” file keeps a track of exact verison. Suppose if we have a update in parsal, but my **Pakage.lock.json** will remain the same and there will be no effect in the functionality. **Package.lock.json** will lock the version and keeps the record of exact version.

There is anothe folder created called “node_modules” , when we run the command it installed all the code for “parcel”.



```
node_modules
  .bin
  @babel
  @lezer
  @lmdb
  @mischnic
  @msgpackr-extract
  @parcel
  @swc
  @trysound
  abortcontroller-polyfill
  ansi-regex
  ansi-styles
  argparse
  base-x
  boolbase
  braces
  browserslist
  callsites
  caniuse-lite
  chalk
  chrome-trace-event
  cli-progress
```

Now you might thinking that we have just installed “parcel what are the other files for” actually “parcel” is a project itself and it has its own dependencies, and those dependencies have their dependencies so this is called **Transitive Dependencies.**

Important Questions !!

Q: How will NPM know which dependencies are of Parcel ?

Q: How many Package.json or Package.lock.json files do we have in our project ?

Ans: We have many package.json files in a project , for example when we installed parcel , it has its own Package.json file and it’s dependencies have its own Package.json file, which help the npm to keep track of the dependencies.

> We should never push the node_module folder into our production or github.

Now you might be wonder, Nitin you said we should not push the node_module to production or Github, how can we do that ?

It’s very easy, just create .gitignore file and inside the .gitignore file type :

```
/node_module
```

- If we have Package.json and Package.lock.json we can recreate the node_module even if we delete the node_module.

> To regenerate the node_module again we have to run a command

```
npm install
```

Igniting our App:

We need to run a command :

```
npx parcel index.html
```

Now what is “ npx parcel index.html "

Q: What is npx ?

Ans: npx : are used for executing a Package.

At the starting of the react learning we imported react using the cdn link but that is not the good way, beacause when we use CDN link it goes to the unpkg , and get the react functionality, but what if we have react installed in our node_module wound’nt that be easy and fast way to use it.

Now we will be installing react into our node_module :

```
npm install react
```

now we will be also installing react-dom into our node_module

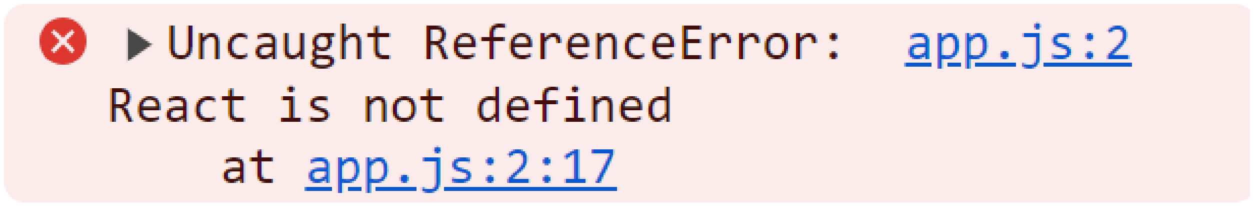
```
npm install react-dom
```

Now we can remove the CDN links from the index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
    <title>Document</title>
    <link rel="stylesheet" href="index.css" />
  </head>
  <body>
    <div id="root">
      <h1>Nitin Is Here</h1>
    </div>

    <script src="app.js"></script>
  </body>
</html>
```

It will work fine but we will have an error in our console :



```
Uncaught ReferenceError:   app.js:2
React is not defined
      at app.js:2:17
```

Our app.js does'nt know where the React has came from, to solve this problem we can import React from 'react'

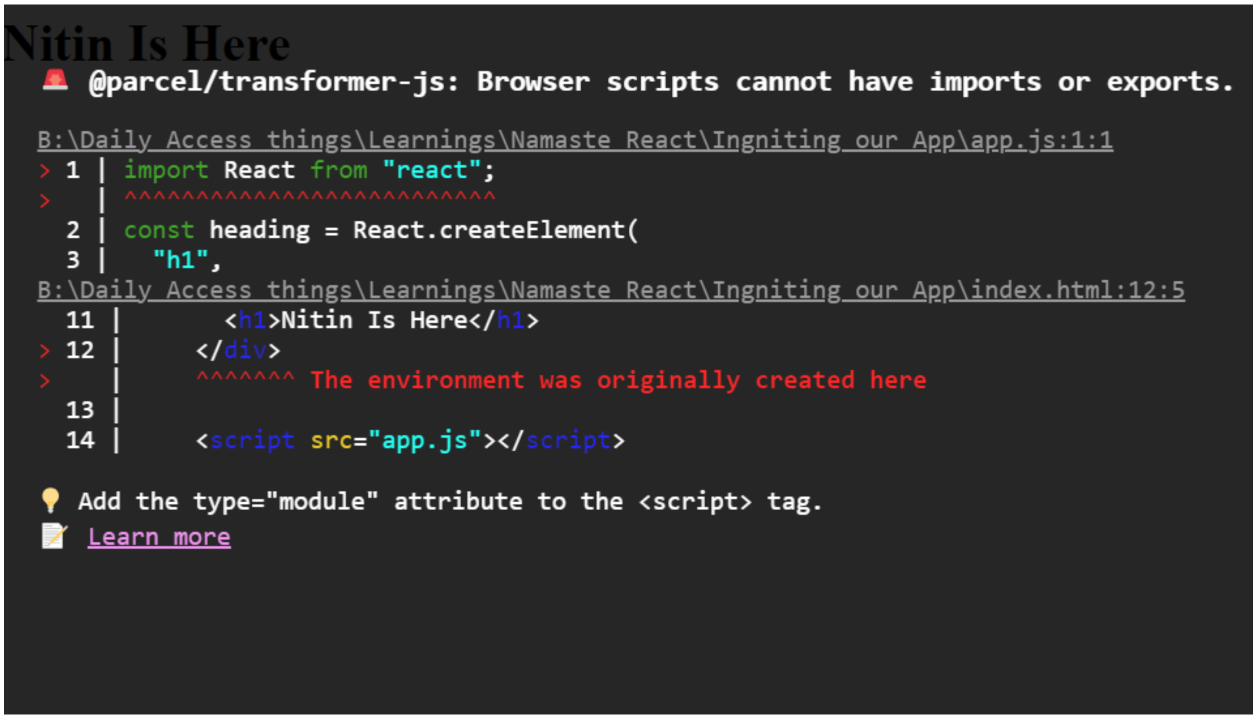
```
import React from "react";
const heading = React.createElement(
  "h1",
  { class: "Kartik" },
  "Hello World ! From React"
);

const parent = React.createElement(
  "div",
  { id: "parent" },
  React.createElement("div", { id: "child" }, [
    React.createElement("h1", {}, "Hey I am a H1 Tag"),
    React.createElement("h1", {}, "Hey there I am a second h1 ta
  ])
);
console.log(parent);

const root = ReactDOM.createRoot(document.getElementById("root"

root.render(parent);
```

Will it works ?



```
Nitin Is Here
🚨 @parcel/transformer-js: Browser scripts cannot have imports or exports.

B:\Daily Access things\Learnings\Namaste React\Ingniting our App\app.js:1:1
> 1 | import React from "react";
>   | ^^^^^^^^^^^^^^^^^^^^^^^^^^
  2 | const heading = React.createElement(
  3 |   "h1",
B:\Daily Access things\Learnings\Namaste React\Ingniting our App\index.html:12:5
  11 |       <h1>Nitin Is Here</h1>
> 12 |     </div>
>    |     ^^^^^^^ The environment was originally created here
  13 |
  14 |     <script src="app.js"></script>

💡 Add the type="module" attribute to the <script> tag.
📝 Learn more
```

We will have an error that we can’t import or export the script file.

This is beacuse the script tag we have imported in the index.html is treated as simple js file, and the normal javascript file does’nt have an import in it, so it is throwing us an error.

So to fix it we can use type= “module” in the script tag :

```html
<script type="module" src="app.js"></script>
```