**Production ready application:**

* Need to be optimized
* Images need to be minified
* Whole application need to be bundled into small application size

Create react app 🡪 already the app is production ready and all the folders comes with create react app

Dive deep into how to create production ready react app

React itself do not make production ready app .. there are many libraries available to use with react and make it happen

What is npm? 🡪act as node package manager …all packages are hosted in node and npm manages It

* Create react app – already have npm with it
* Add node package manager

Npm init -> it create new package.json file which is configuration for npm ..why do we need?

Npm 🡪 node package manager

1.package.json -🡪 it is configuration for npm .. we can get packages or dependencies into application 🡪 it keeps track of all dependencies or packages installed in application

**2.important packages 🡪 bundler🡪whole code need to be compressed,cleaned,optimize it .. whole code need to be compressed and cleaned**

* **What is bundler? 🡪 webpack,parcel,vite are bundlers**

**It bundles the app to push it to production**

**Create react app 🡪 It uses webpack behind scenes**

we are going to use parcel ..

**1.get parcel to app**

**npm install parcel -d 🡪 what is -d**

there are 2 types of dependencies we can install 🡪 dev dependency or normal dependency .I want the package as dev dependency …so we are using -d to make it use only for dev envoinment

**dependencies are the packages your application needs to run in production, like React or Axios.  
devDependencies are only needed during development, like Webpack, ESLint, or testing libraries.  
These help reduce the final build size by excluding unnecessary packages from the production bundle.**

 "dependencies": {

    "parcel": "^2.14.4"

  }

}

^ -> if new upgrade comes to **parcel** it will install the minor upgrades

~-> it will install the major upgrade of parcel

**Package.lock.json 🡪 why?**

* It keeps track of exact version of package .. it locks the version of package ..it has exact version of package
* **What is integrity ..>** it keeps the hash of all the packages with exact versions to check whether the package in local and production are same ..i.e both have same version of package present
* **Node modules:**
  + It went to production and got all the code of parcel and put in node modules
  + Node modules is like database that has the original code of parcel
  + It fetches all the code of all the dependencies or package
  + There are many other folders present for parcel ..they are dependencies of parcel … and those dependencies will be dependent on other dependencies this is called transitive dependency ..pacel knows its dependencies through its package.json
* **We no need to push node modules to git why?**
  + we can install the dependencies of project using package.json and we can compare whether it is correct using package.lock.json .. we can recreate it with package.json

**what is npx?**

* We are executing a package using npx -> it will execute the package

Cdn is not preferred why?

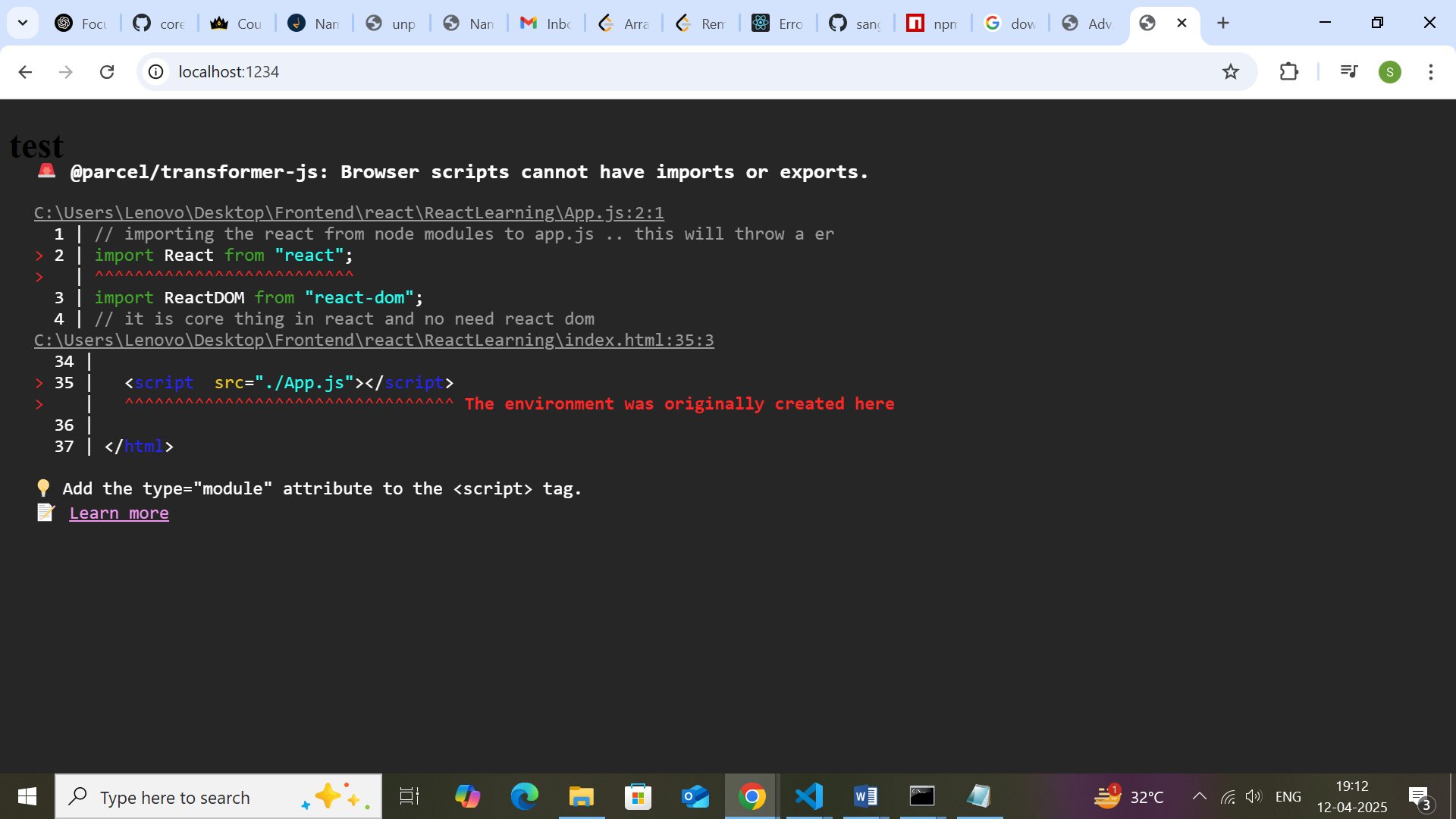
1. I already have react in node modules it is easy for us to have in package.json

Npm install react

When we import the react from node modules inside app.js file

  <script src="./App.js"></script>

It throws below error



The error is due to the script is loaded as browser scripts i.e app.js is added to html as script that run in browser

App.js is not javascript … normal js dosnt have import .so we need to tell browse app.js is not script it is module which can be done using module

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

**#parcel**

#it is setting up server for us

#it is automatically loading changes to ui --> it is doing hmr -->hot module replacement

#how it do ?? ..it is done using file watching algorithm --> written with c++

**#parcel is cacheing -->it caches the files in parcel-cache ..> it watches the code change and changes only the file that got changed**

#during first time it takes more time and on subsequent builds it takes cache of code and runs the code faster

#pracel will do image optimisation

#when we make production build it will minify the file .. it will do bundle

#compress your file-->remove all empty space everything

#react alone is not making application fast ..it is because of bundler like vite,parcel etc to do it and optimise it

**#parcel features in short**

1.dev build

2.hmr

3.local server

4.file watching algorithm

5.image optimisation

6.minifcation

7.compression

8.bundling

9.consistant hashing --> what it is?

10.it will do code spliting

11.differnetial bundling ---> app can be opened in any browser old or new .. parcel will give different bundles for different type of older or new browser

12.diganoies -> gives better error suggestions

13.host app in https

14.tree shaking --> in whole code if we are using only 5 functions then it will remove unused remaining code

16.different dev and production build

#parcel --> it done all above with many packages --> parcel is like manager for all other packages

**What happen when we run npx parcel index.html?**

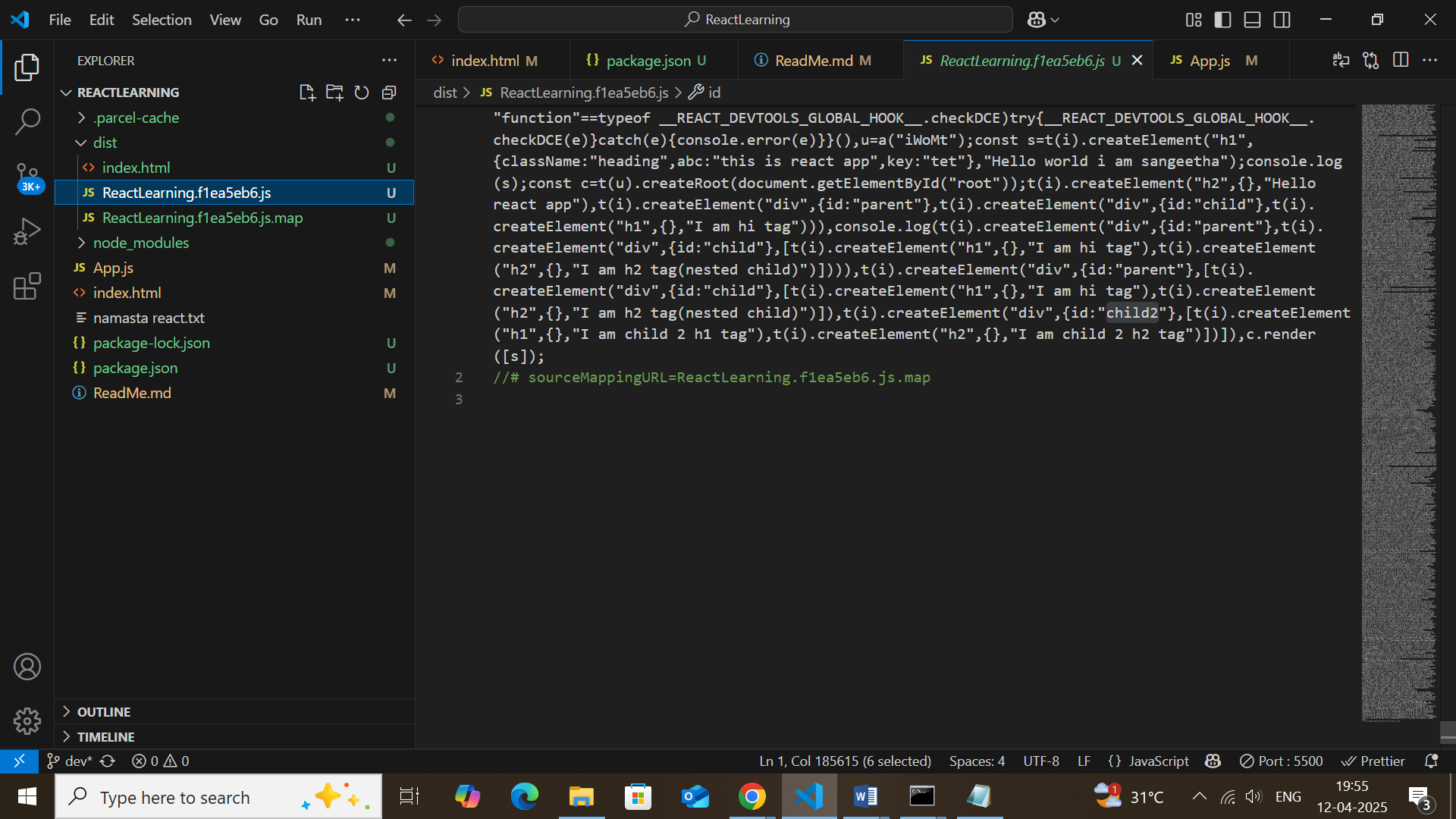
1.it generate the dev build and put into dist folder and when we save file it uses the parcel and dist to do hot module replacement and save changes

**To build production build we use**

**Npx parcel build index.html**

1.This will build minified version of files in dist folder

2.it will give only 3 files and it minified all the lines inside html or js files



We don’t want to put dist and node modules inside git as we can generate it using package.json and npx parcel build index.html

**Make react code compatible with older browser ..how?? – by browserlist package and array config in package.json**

We need to tell the app what are browsers that it should support .. how?

Through package.json using browserlist

<https://browserslist.dev/?q=bGFzdCAyIGNocm9tZSB2ZXJzaW9u> – this gives list of browser version

 "browsersList": [

    "last 2 versions"

  ]

This code in package.json tells the app to work in all latest 2 version of browsers

**🛠 What Bundlers Like Vite/Parcel Do**

When you run vite build or parcel build, here’s what happens:

| **Step** | **What It Does** |
| --- | --- |
| ✅ Tree-shaking | Removes unused code (dead code elimination) |
| ✅ Minification | Compresses JS/CSS by removing spaces, shortening variable names |
| ✅ Code Splitting | Breaks the app into smaller chunks to load only what’s needed |
| ✅ Asset Compression | Images/fonts can be compressed or optimized |
| ✅ Module Bundling | Combines modules into fewer files to reduce HTTP requests |
| ✅ Caching Headers | Helps browser cache files effectively |

➡️ All of this ends up in the dist/ folder as optimized, ready-to-deploy files.

**⚡ From Build to Browser: What Happens?**

1. **First Load**:
   * HTML loads.
   * Browser fetches minified JS/CSS from /dist.
   * React app runs and takes over the page (single-page app behavior).
2. **Subsequent Loads**:
   * Browser uses **cached JS/CSS files** (thanks to cache headers).
   * Page loads very fast since network isn’t hit again unless files changed.

So yes — you’re right: bundlers make your app faster by optimizing and **preparing it for the browser**, and React makes it feel fast on the page by updating the UI efficiently.

What is Tree Shaking?

Remove unwanted code which are not used in module from file.

If you have this utility file:

js

CopyEdit

// utils.js

export function add(a, b) { return a + b; }

export function multiply(a, b) { return a \* b; }

And in your app you only use:

js

CopyEdit

import { add } from './utils';

Then during bundling, **multiply() will be removed** — because it’s never used. ✅

What is HMR?

Hot Module Replacement (HMR) is a feature provided by modern bundlers (like Webpack, Vite, Parcel) that allows modules (files) to be updated in the browser without a full page reload.

### 🛠️ How It Works:

* The bundler **watches your files** for changes.
* When a file is updated (e.g., you change a React component), only **that specific module** is **recompiled**.
* The browser **injects the new module** and **updates the UI instantly** — **without reloading the whole page** or losing app state.

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

* Package.json 🡪 manages the list of packages or dependies with aprox version either with ^ or ~
* Package-lock.json 🡪locks the version of packages strictly and used to compare it with production build of package.lock.json

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

**package-lock.json** is an auto-generated file that **locks the versions of all dependencies** in your project, ensuring **consistent installs** across environments. It is **not intended** to be manually modified for the following reasons:

What is `node\_modules` ? Is it a good idea to push that on git?

Node moduels contains the original code of package with source code of all other dependent packages . we no need ot push to git as it can be generated back with package.json list

- What is the `dist` folder?

It is folder that stores all build files from bundlers are minified version from which each time the files are loaded to application

**What is `browserlists`**

It tells app to which browser version the app should be compatable with .. it is package that helps to determine in application

**browserslist** is a tool that allows you to specify which browsers and environments you want to **support** for your web project. It helps in determining the compatibility of the **CSS** and **JavaScript** you write, ensuring it works across the browsers and versions you care about.