

Episode-02 | Igniting Our Application

Theory Assignment:

1. What is NPM?

NPM (Node Package Manager) is a package manager for JavaScript programming language. It is used to install, share, and manage dependencies in a JavaScript project. NPM comes bundled with Node.js and allows developers to easily manage libraries, tools, and frameworks needed for their projects.

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

Parcel and Webpack are both module bundlers commonly used in web development. They take various assets (JavaScript, CSS, images, etc.) and bundle them together in a way that's optimized for deployment.

We need bundlers like Parcel or Webpack because they:

- **Bundle Assets:** Combine multiple files into a single file for optimized loading.
- **Code Splitting:** Allow for splitting code into smaller chunks for efficient loading.
- **Module Loading:** Enable using modules and dependencies in a modular way.
- **Optimizations:** Perform optimizations like minification and compression for better performance.

3. What is .parcel-cache?

The .parcel-cache folder is created by Parcel to store cached data. It helps to speed up the build process by reusing previously processed information, reducing the need to recreate everything from scratch.

4. What is npx?

npx is a package runner tool that comes with NPM. It is used to execute Node.js packages without the need to install them globally. It helps run packages as if they were installed, making it convenient for one-time use or executing specific package versions.

5. What is the difference between dependencies vs devDependencies?

- **Dependencies:** These are packages required for the application to run in a production environment. They are crucial for the application to work as intended.
- **devDependencies:** These are packages required for development and testing purposes. They include tools, testing frameworks, and other utilities that are not needed in the production environment.

6. What is Tree Shaking?

Tree shaking is a technique used by bundlers like Webpack and Parcel to eliminate dead (unused) code during the build process. It helps in reducing the size of the final bundled file by removing unused parts of the code.

7. What is Hot Module Replacement?

Hot Module Replacement (HMR) is a feature provided by bundlers like Webpack. It allows developers to update modules in the application without a full page refresh. This significantly speeds up the development process by applying changes on-the-fly.

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

1. **Zero Configuration:** Parcel requires minimal to no configuration, making it easy for developers to get started without spending time on setup.
2. **Automatic Asset Optimization:** Parcel automatically optimizes assets, including minification, compression, and other performance improvements.
3. **Blazing Fast:** Parcel is known for its fast build times, making the development workflow smoother and more efficient.

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

.gitignore is a file used to specify files and directories that should be ignored by Git. It helps in avoiding the unnecessary addition of files to version control. Common entries include:

- **node_modules/**
- **.parcel-cache/**
- **dist/**

Files like configuration files containing sensitive information (e.g., API keys) should not be added to version control.

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

- **package.json:** Contains metadata about the project, dependencies, scripts, and other configuration. It is used for high-level configuration.
- **package-lock.json:** Locks down the version of each package's dependencies. It ensures that every developer working on the project uses the same dependency versions, reducing potential issues with different environments.

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

package-lock.json should not be manually modified because it is generated automatically by NPM to lock down dependency versions. Modifying it manually can lead to inconsistencies and conflicts between developers' environments.

12. What is `node_modules`? Is it a good idea to push that on git?

`node_modules` is a directory where NPM installs project dependencies. It contains the actual code of the packages required by the project. It is not recommended to push `node_modules` to Git, as it can result in a large repository size, and dependencies can be easily installed using the `npm install` command on the developer's machine.

13. What is the `dist` folder?

The `dist` (distribution) folder is where the bundled and optimized files for production are stored. It contains the final version of the application that can be deployed.

14. What is `browserslist`?

`browserslist` is a configuration file or entry in `package.json` specifying which browsers and their versions your project should support. It is used by tools like Babel and Autoprefixer to determine the set of browsers for which the code should be transpiled or prefixed.

Coding Assignment for Episode 2 of Namaste React

In your existing project:

1. Initialize npm into your repo:

npm init

```
ameya Episode-2-Namaste-React 00:59 npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (episode-2-namaste-react)
version: (1.0.0)
description: coding tutorial for namaste react episode 2
entry point: (index.js)
test command: jest
git repository:
keywords: parcel bundler webpack
author: Ameya Belvalkar
license: (ISC)
About to write to C:\Users\belva\OneDrive\Desktop\Episode-2-Namaste-React\package.json:
{
  "name": "episode-2-namaste-react",
  "version": "1.0.0",
  "description": "coding tutorial for namaste react episode 2",
  "main": "index.js",
  "scripts": {
    "test": "jest"
  },
  "keywords": [
    "parcel",
    "bundler",
    "webpack"
  ],
  "author": "Ameya Belvalkar",
  "license": "ISC"
}

Is this OK? (yes)
```

2. Install react and react-dom:

npm install react react-dom

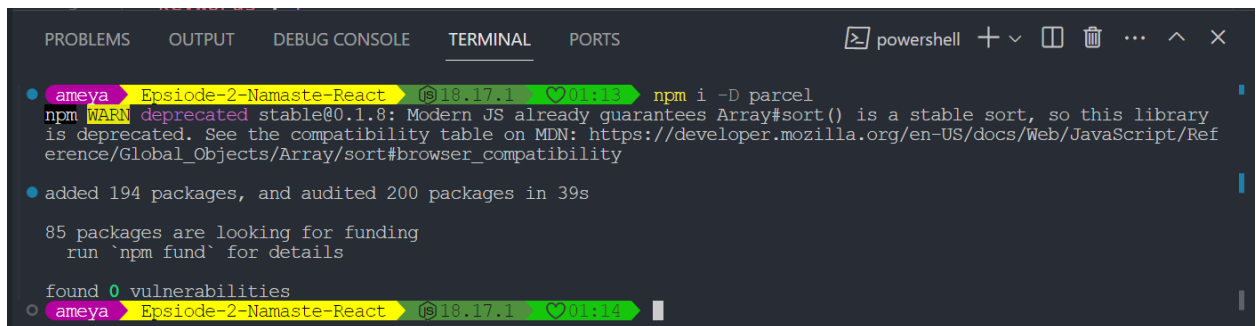
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + - [] ... ^ x
ameya Episode-2-Namaste-React 18.17.1 01:11 npm install react react-dom
```

3. **Remove CDN links of React:** Replace any CDN links of React in your HTML files with the following import statements:

```
<script type="module">
  import React from 'react';
  import ReactDOM from 'react-dom';
  // Your other scripts...
</script>
```

4. **Install Parcel:**

```
npm install -D parcel
```



```
ameya@Epsiode-2-Namaste-React: ~$ npm i -D parcel
npm WARN deprecated stable@0.1.8: Modern JS already guarantees Array#sort() is a stable sort, so this library
is deprecated. See the compatibility table on MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Ref
erence/Global_Objects/Array/sort#browser_compatibility

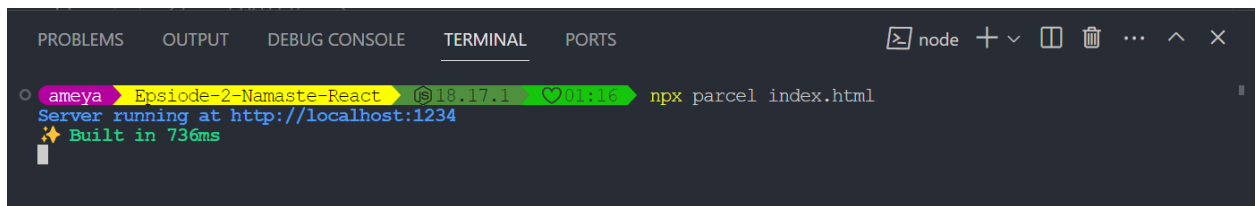
added 194 packages, and audited 200 packages in 39s

85 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

5. **Ignite your app with Parcel:** Create an entry file (e.g., index.html or index.js) and run:

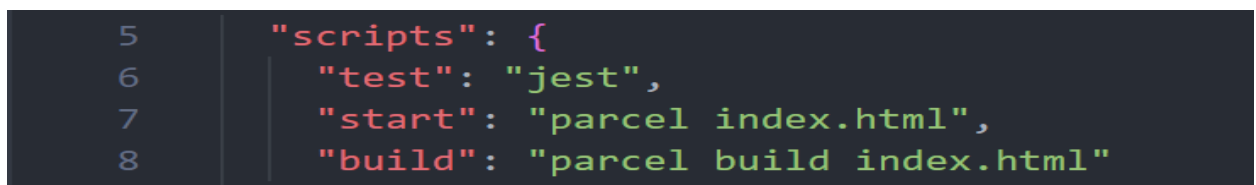
```
npx parcel index.html
```



```
ameya@Epsiode-2-Namaste-React: ~$ npx parcel index.html
Server running at http://localhost:1234
Built in 736ms
```

6. **Add scripts for “start” and “build” with Parcel commands:** Update your package.json file with the following scripts:

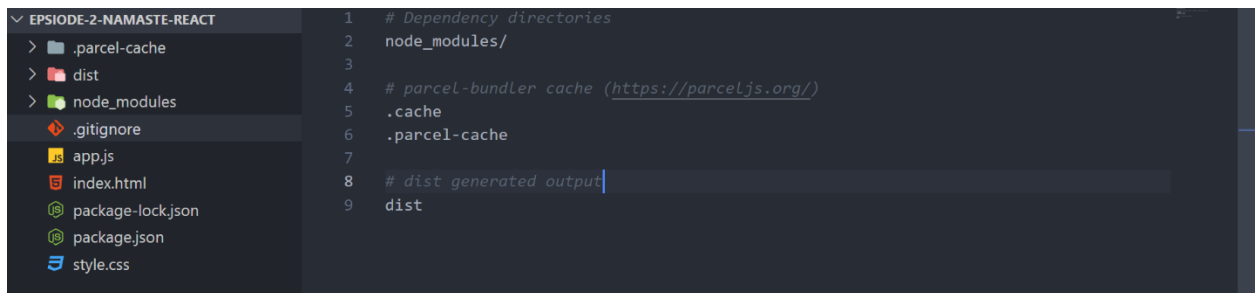
```
"scripts": {
  "test": "jest",
  "start": "parcel index.html",
  "build": "parcel build index.html"
}
```



```
5  "scripts": {
6    "test": "jest",
7    "start": "parcel index.html",
8    "build": "parcel build index.html"
```

7. **Add .gitignore file:** Create a .gitignore file in the root of your project and include the following:

```
node_modules/  
.parcel-cache/  
dist/
```

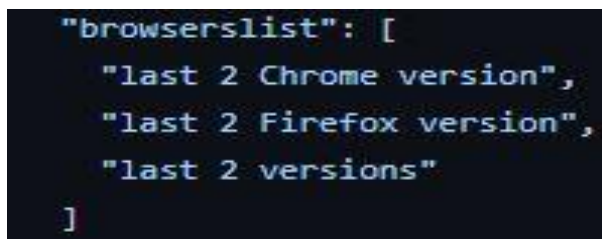


8. **Add browserlists:** Create a .browserslistrc file in the root of your project and specify the browsers you want to support. For example:

```
last 2 versions
```

```
last 2 Chrome versions
```

```
last 2 Firefox versions
```



9. **Build a production version of your code using parcel build:**

```
npm run build
```

