

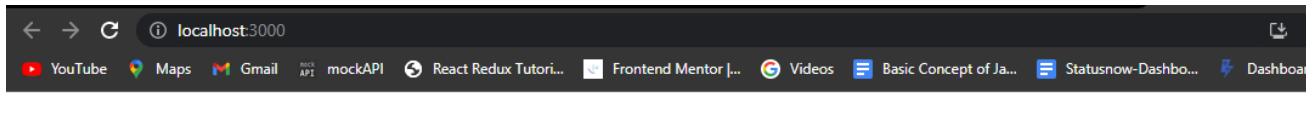
## Webpack

Webpack is a module bundle. Webpack is an open-source module bundle, minify js,css.

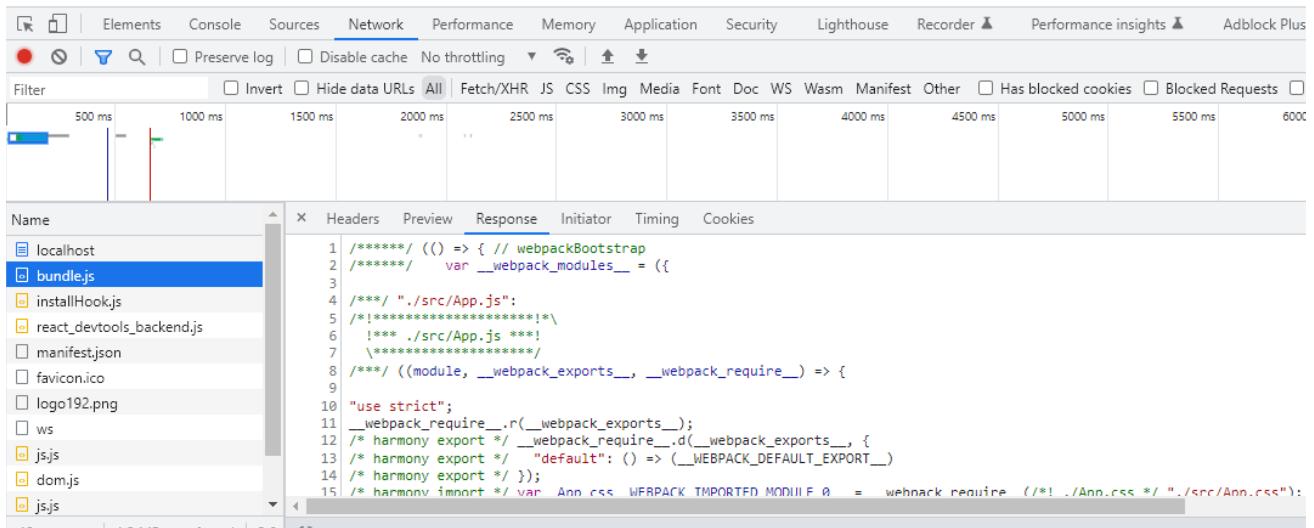
For an example->

Create React app, go to network and refresh the page, you can see bundle.js and main.js (all contains are there css,js)

### bundle.js



### Webpack+React



If you create home.js , the about.js component will also create a bundle, if you go to the home page and refresh only that page. So application of that page will increase.

When using a bundle, there is loader, css, scss. And scss is converted into css. Then use loader(plugin) for css or scss.

Create a file into react application, name is dist where all bundle of home.js, about.js page are there.

All unnecessary files, space are stored in bundle.js.

## Webpack Install, Webpack-CLI & Configure

Install->

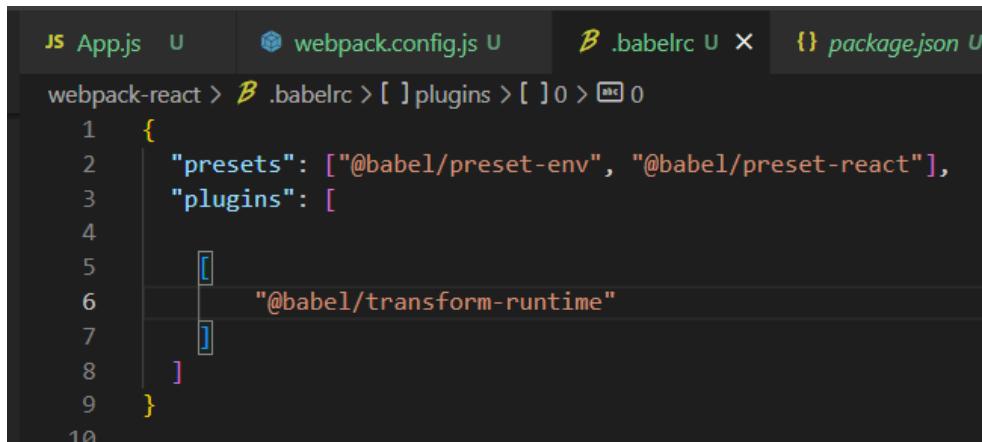
```
1) npm i --save-dev webpack webpack-dev-server webpack-cli
2) npm i --save-dev @babel/core babel-loader @babel/preset-env @babel/preset-react
3) npm install --save @babel/runtime
4) npm install --save-dev @babel/plugin-transform-runtime
```

After all package installation, create two files, one is **webpack.config.js** and another is **.babelrc**.

**babelrc** file is your local configuration for your code in your project. Generally you would put it in the root of your application repo.

The webpack configuration file **webpack.config.js** is the file that contains all the configuration, plugins, loaders, etc. to build the JavaScript part of the NativeScript application.

In .babelrc->>



```
js App.js U webpack.config.js U .babelrc U package.json U
webpack-react > .babelrc > [ ] plugins > [ ] 0 > 0
1  {
2    "presets": ["@babel/preset-env", "@babel/preset-react"],
3    "plugins": [
4      [
5        "@babel/transform-runtime"
6      ]
7    ]
8  }
9
10
```

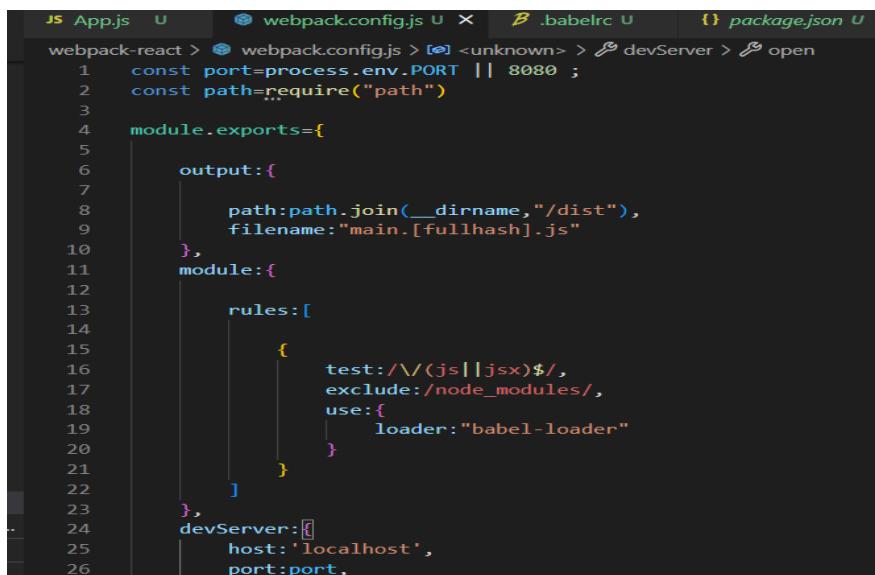
Babel presets can act as a shareable set of Babel plugins and/or config options.

DOC Link- <https://babeljs.io/docs/presets>

Plugin-> Babel's code transformations are enabled by applying plugins (or presets) to your configuration file.

DOC Link-> <https://babeljs.io/docs/plugins>

In webpack.config.js->



```
js App.js U webpack.config.js U .babelrc U package.json U
webpack-react > webpack.config.js > <unknown> > devServer > open
1 const port=process.env.PORT || 8080 ;
2 const path=require("path")
3
4 module.exports={
5   output:{
6     path:path.join(__dirname,"/dist"),
7     filename:"main.[fullhash].js"
8   },
9   module:{}
10
11   rules:[
12     {
13       test:/\.(js|jsx)$/,
14       exclude:/node_modules/,
15       use:{
16         loader:"babel-loader"
17       }
18     }
19   ],
20   devServer:[
21     host:'localhost',
22     port:port,
23   ]
24 }
25
26
```

```
23     },
24     devServer: [
25       host:'localhost',
26       port:port,
27       historyApiFall:true,
28       open:true
29     ]
30 }
```

Filename:"main.[fullhash].js" → This creates a different name.

Start with this project webpack.

```
},
  ▷ Debug
  "scripts": [
    "start": "webpack server --mode production --open --hot",
    "build": "webpack --mode production",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  ],
  "eslintConfig": {}
```

Start with cmd: \$npm run build

If any error arise->

```
assets by status 392 bytes [cached] 1 asset
./src/index.js 535 bytes [built] [code generated] [1 error]

ERROR in ./src/index.js 9:2
Module parse failed: Unexpected token (9:2)
You may need an appropriate loader to handle this file type, currently no loaders are configured to
process this file. See https://webpack.js.org/concepts#loaders
| const root = ReactDOM.createRoot(document.getElementById('root'));
| root.render(
>   <React.StrictMode>
|     <App />
|   </React.StrictMode>

webpack 5.78.0 compiled with 1 error in 775 ms
PS C:\ReactJs-March\webpack-react> []
```

Solve this error->

```
rules:[
  {
    test:/\.(js|jsx)$/,
    exclude:/node_modules/,
    use:{
      loader:"babel-loader"
    }
  }
]
```

Then run the build.

Comment in `app.css`, `index.css`.

webpack.config.js->

```
const port=process.env.PORT || 8080 ;
const path=require("path")

module.exports={
  entry: './src/index.js',

  output:{

    path:path.join(__dirname,"/dist"),
    filename:"main.[fullhash].js"
  },
  module:{

    rules:[
      {
        test:/\.(js|css|jsx)$/,
        exclude:/node_modules/,
        use:{
          loader:"babel-loader"
        }
      }
    ]
  },
}
```

```
devServer:{  
    host:'localhost',  
    port:port,  
    open:true  
}  
}
```

Create Index.html file.

When use template, use plugin into webpack.config.js, install it->  
`$npm i html-webpack-plugin`

#### Webpack.config.js:

```
const port=process.env.PORT || 8080 ;  
const path=require("path");  
const HtmlWebpackPlugin = require('html-webpack-plugin');  
  
module.exports={  
    entry: './src/index.js',  
  
    output:{  
  
        path:path.join(__dirname,"/dist"),  
        filename:"main.[fullhash].js"  
    },  
    module:{  
  
        rules:[  
  
            {  
                test:/\.(js|css|jsx)$/,  
                exclude:/node_modules/,  
                use:{  
                    loader:"babel-loader"  
                }  
            }  
        ]  
    }  
};
```

```
plugins: [new HtmlWebpackPlugin({
  template:'src/index.html'
}),
{
  devServer:{
    host:'localhost',
    port:port,
    open:true
  }
}]
```

## Entry point, Output & Mode

When it is in production mode, main.js file change is compact. When it is development mode , it is different. (`package.json`)

Development mode:

```
15  "scripts": [
16    "start": "webpack server --mode development --open --hot",
17    "build": "webpack --mode development",
18    "test": "react-scripts test",
19    "eject": "react-scripts eject"
20  ],
21  "eslintConfig": {
```

Production mode:

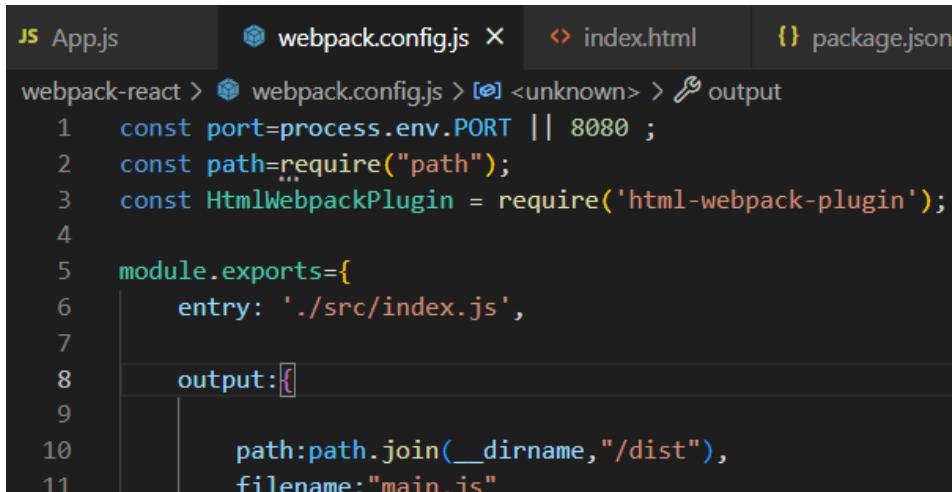
```
"scripts": [
  "start": "webpack server --mode production --open --hot",
  "build": "webpack --mode production",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
],
```

Run this two things->

\$npm run build

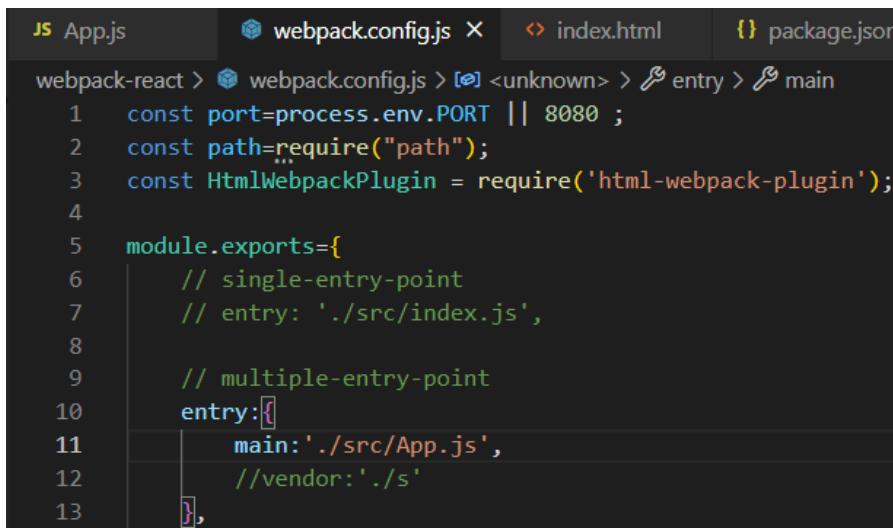
You can also define mode into `webpack.config.json`.

Single entry point for file, you can define like this,



```
webpack-react > webpack.config.js > [o] <unknown> > output
1 const port=process.env.PORT || 8080 ;
2 const path=require("path");
3 const HtmlWebpackPlugin = require('html-webpack-plugin');
4
5 module.exports={
6   entry: './src/index.js',
7
8   output:[
9     {
10       path:path.join(__dirname,"/dist"),
11       filename:"main.js"
12     }
13   ]
14 }
```

Multiple entry point for file, you can define like this,



```
webpack-react > webpack.config.js > [o] <unknown> > entry > main
1 const port=process.env.PORT || 8080 ;
2 const path=require("path");
3 const HtmlWebpackPlugin = require('html-webpack-plugin');
4
5 module.exports={
6   // single-entry-point
7   // entry: './src/index.js',
8
9   // multiple-entry-point
10  entry:[
11    {
12      main:'./src/App.js',
13      //vendor:'./s'
14    }
15  ],
16 }
```

If you want to change entry point, you can change like this,

```
entry:{
  main:'./src/<file-name>',
  //vendor:'./s'
},
```

How can code identify the webpack config file?  
If webpack config file change the name,

Webpack.config.js to webpacktest.js.

Change in package.json, also change in changeable files, like webpacktest.js.

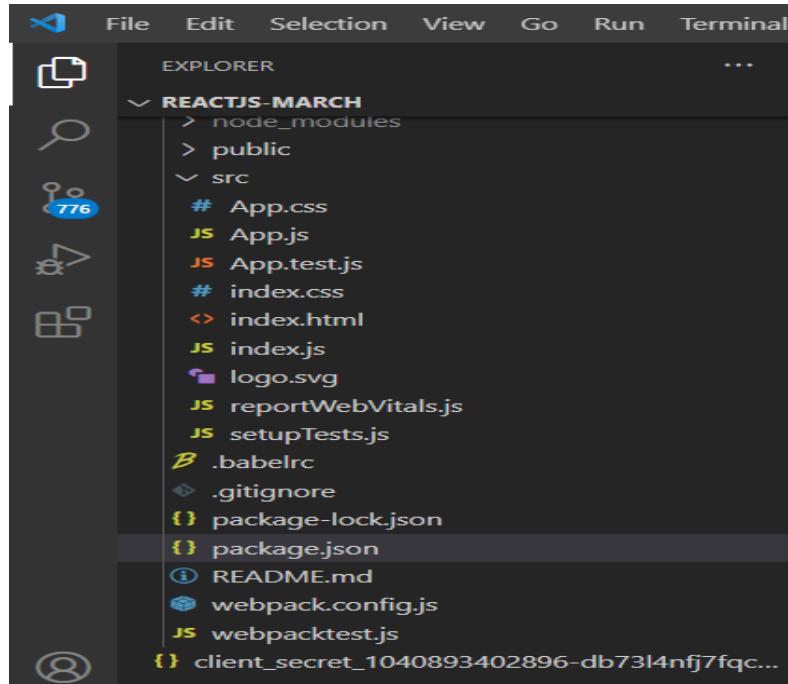
```
webpack-react > JS webpacktest.js > [?] <unknown>
1 const port=process.env.PORT || 8080 ;
2 const path=require("path");
3 const HtmlWebpackPlugin = require('html-webpack-plugin');
4
5 module.exports=[
6   mode:"development",
7   // single-entry-point
8   entry: './src/index.js',
9
10  // multiple-entry-point
11  // entry:{
12  //   main:'./src/App.js',
13  //   //vendor:'./s'
14  // },
15]
```

```
webpack-react > {} package.json > {} scripts > build
9   "html-webpack-plugin": "^5.5.0",
10  "react": "^18.2.0",
11  "react-dom": "^18.2.0",
12  "react-scripts": "5.0.1",
13  "web-vitals": "^2.1.4"
14 },
15   > Debug
16   "scripts": [
17     "start": "webpack server --mode development --open --hot",
18     "build": "webpack --config webpacktest.js",
19     "test": "react-scripts test",
20     "eject": "react-scripts eject"
21   ],
22   "eslintConfig": {
23     "extends": [
24       "react-app",
25       "react-app/jest"
26     ]
27   }
28 }
```

"build": "webpack --config <path of the folder>"

If the path name is different from file name, then you get an error message. Like this,

Folder name: webpacktest.js

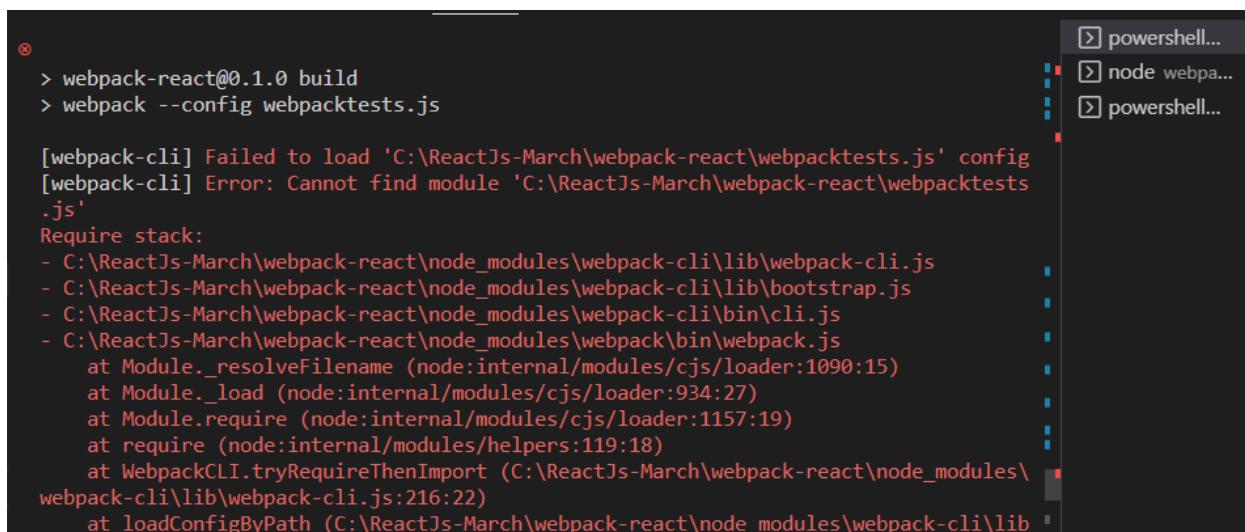


Package.json: "build": "webpack --config webpacktests.js",

```
webpack-react > {} package.json > {} scripts > build
  9   "html-webpack-plugin": "^5.5.0",
10   "react": "^18.2.0",
11   "react-dom": "^18.2.0",
12   "react-scripts": "5.0.1",
13   "web-vitals": "^2.1.4"
14 },
  ▶ Debug
15 "scripts": [
16   "start": "webpack server --mode development --open --hot",
17   "build": "webpack --config webpacktests.js",
18   "test": "react-scripts test",
19   "eject": "react-scripts eject"
20 ],
  "eslintConfig": {
```

The screenshot shows the VS Code editor with the 'package.json' file open. The 'scripts' section is highlighted, showing the 'build' command: "webpack --config webpacktests.js".

## Error->



```
✖ > webpack-react@0.1.0 build
> webpack --config webpacktests.js

[webpack-cli] Failed to load 'C:\ReactJs-March\webpack-react\webpacktests.js' config
[webpack-cli] Error: Cannot find module 'C:\ReactJs-March\webpack-react\webpacktests
.js'
Require stack:
- C:\ReactJs-March\webpack-react\node_modules\webpack-cli\lib\webpack-cli.js
- C:\ReactJs-March\webpack-react\node_modules\webpack-cli\lib\bootstrap.js
- C:\ReactJs-March\webpack-react\node_modules\webpack-cli\bin\cli.js
- C:\ReactJs-March\webpack-react\node_modules\webpack\bin\webpack.js
  at Module._resolveFilename (node:internal/modules/cjs/loader:1090:15)
  at Module._load (node:internal/modules/cjs/loader:934:27)
  at Module.require (node:internal/modules/cjs/loader:1157:19)
  at require (node:internal/modules/helpers:119:18)
  at WebpackCLI.tryRequireThenImport (C:\ReactJs-March\webpack-react\node_modules\
webpack-cli\lib\webpack-cli.js:216:22)
  at loadConfigByPath (C:\ReactJs-March\webpack-react\node_modules\webpack-cli\lib
```

Then check the folder, change its name, and solve it.

Run-> \$npm run build

You can change mode in production and development mode in the webpack config file.

In package.json,

```
change ,
"start": "webpack server --mode development --open
--hot",
"build": "webpack --config webpacktest.js"
```

To

```
"start": "webpack server --mode development --open --hot",
"build": "webpack --mode development"
```

Run-> \$npm run build

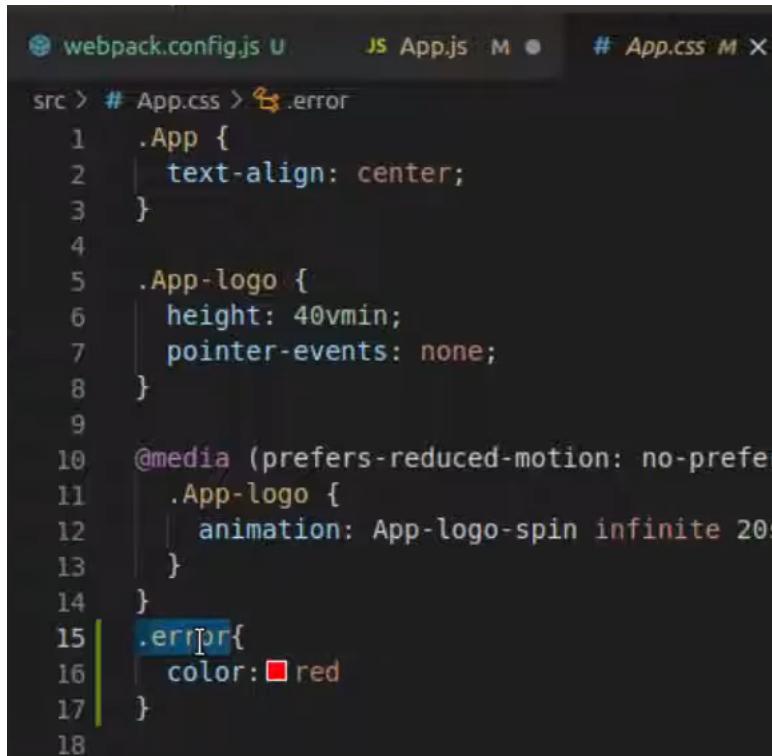
## Loader, Html Loader & File Loader

First you need to install file-loader,html-loader.  
\$npm i file-loader html-loader.

Want to do that color of any html text.

**Step to follow->**

**App.css->**



```
webpack.config.js  JS App.js  # App.css X
src > # App.css > error
1  .App {
2    text-align: center;
3  }
4
5  .App-logo {
6    height: 40vmin;
7    pointer-events: none;
8  }
9
10 @media (prefers-reduced-motion: no-prefer
11   .App-logo {
12     animation: App-logo-spin infinite 20s
13   }
14 }
15 .error{
16   color:red
17 }
```

**App.js->**

A screenshot of a code editor showing the file `App.js`. The code defines a `App` component that returns a `<div>` containing a centered `<h1>` with the text "Webpack + React". The code is annotated with a tooltip "development" pointing to the right side of the editor.

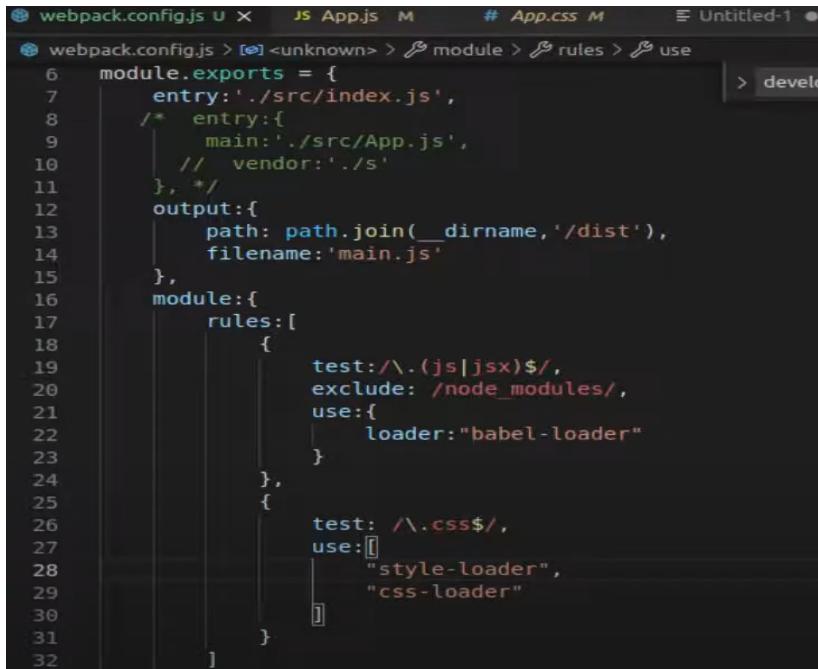
```
src > JS App.js > App
1 import React from 'react'
2 import './App.css'
3 function App() {
4
5     return (
6         <div>
7             <h1 className="error"><center>Webpack + React</center></h1>
8         </div>
9     );
10 }
11
12 export default App;
```

In `webpack.config.js` file (use css loader)

A screenshot of a code editor showing the `webpack.config.js` file. The configuration includes an `entry` point at `./src/index.js`, an `output` path to `/dist/main.js`, and a `module.rules` section. The `rules` section contains two entries: one for `js|jsx` files using the `babel-loader`, and another for `.css` files using the `css-loader`.

```
module.exports = {
  entry: './src/index.js',
  output: {
    path: path.join(__dirname, '/dist'),
    filename: 'main.js'
  },
  module: {
    rules: [
      {
        test: /\.js|jsx$/,
        exclude: /node_modules/,
        use: [
          {
            loader: "babel-loader"
          }
        ],
        [
          {
            test: /\.css$/,
            use: [
              {
                loader: "css-loader"
              }
            ]
          }
        ]
      },
      plugins: [
        new HtmlWebpackPlugin({
          template: 'index.html'
        })
      ]
    ]
  }
};
```

Use **style-loader**->



```
webpack.config.js 6  JS App.js M # App.css M Untitled-1
webpack.config.js > <unknown> > module > rules > use
6   module.exports = {
7     entry: './src/index.js',
8     /* entry: {
9       main: './src/App.js',
10      // vendor: './s'
11    }, */
12    output: {
13      path: path.join(__dirname, '/dist'),
14      filename: 'main.js'
15    },
16    module: {
17      rules: [
18        {
19          test: /\.js|jsx$/,
20          exclude: /node_modules/,
21          use: {
22            loader: "babel-loader"
23          }
24        },
25        {
26          test: /\.css$/,
27          use: [
28            "style-loader",
29            "css-loader"
30          ]
31        }
32      ]
33    }
34  }
```

Now, start it-> \$npm start

Style-loader is injected into DOM, css-loader is injected into main.js bundle.

If you are a scss loader then 1st scss is converted into css.

If you are use image into component,

Step to follow->

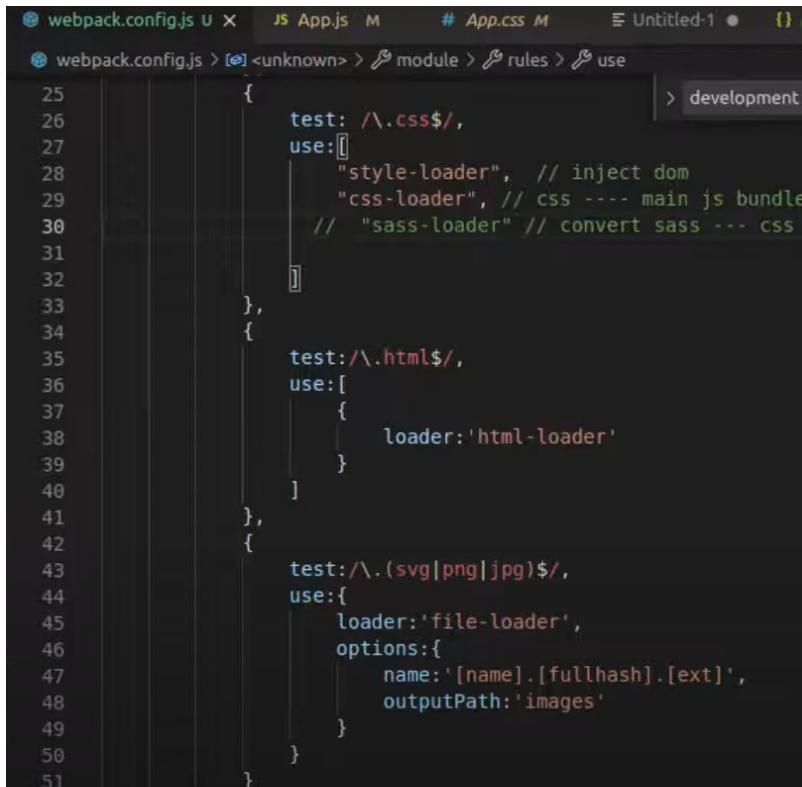
App.js

```
terminal Help
webpack.config.js u JS App.js M X # App.css M Untitled-1 ● {} package.js

src > JS App.js > App
1 import React from 'react' > development
2 import './App.css'
3 import logo from './logo.svg'
4 function App() {
5
6   return (
7     <div>
8       <h1 className="error"><center>Webpack + React</center></h1>
9
10      <div className="App-header">
11        <img src={logo} className="App-logo" alt="logo" />
12      </div>
13    </div>
14  );
15
16}
17
18 export default App;
19
```

If you start it, it is throwing an error  
Then use html loader into webpack.config.js file,

```
webpack.config.js u ● JS App.js M # App.css M Untitled-1 ● {} pa
webpack.config.js > [●] <unknown> > module > rules > use > options
rules:[
  {
    test: /\.js|jsx$/,
    exclude: /node_modules/,
    use: [
      {
        loader: "babel-loader"
      }
    ],
    {
      test: /\.css$/,
      use: [
        "style-loader", // inject dom
        "css-loader", // css ---- main js bundle
        "sass-loader" // convert sass --- css
      ]
    },
    {
      test: /\.html$/,
      use: [
        {
          loader: 'html-loader'
        }
      ]
    }
  ],
  f
```



A screenshot of a code editor showing the `webpack.config.js` file. The code defines rules for different file types:

```
25      {
26        test: /\.css$/,
27        use:[{"style-loader", // inject dom
28              "css-loader", // css ---- main js bundle
29              // "sass-loader" // convert sass --- css
30            ],
31        },
32      },
33      {
34        test:/\.(html$|,
35        use:[
36          {
37            loader:'html-loader'
38          }
39        ],
40      },
41      {
42        test:/\.(svg|png|jpg)$|,
43        use:{
44          loader:'file-loader',
45          options:{name:'[name].[fullhash].[ext]', outputPath:'images'}
46        }
47      }
48    }
49  }
50 }
51 }
```

Now need to start->

\$npm run build

\$npm start

If image not come ,change the path of  
webpack.config.js -> full hash to hash

```
3 },
4 {
5     test:/\.\html$/,
6     use:[
7         {
8             loader:'html-loader'
9         }
10    ]
11 },
12 {
13     test:/\.(svg|png|jpg)$/,
14     use:{
15         loader:'file-loader',
16         options:[
17             name:'[name].[hash].[ext]',
18             outputPath:'images'
19         ]
20     }
21 }
```

\$npm run build

\$npm start

## Code Split & Webpack Magic Comments

All bundles are stored into one folder. For example , home page, about page , admin page are into a project, for this each and every component has a different bundle to create.

That is why page loading is increasing.

Previously I saw that all bundle into main.js, but now I'm going to create separate bundle for separate components.

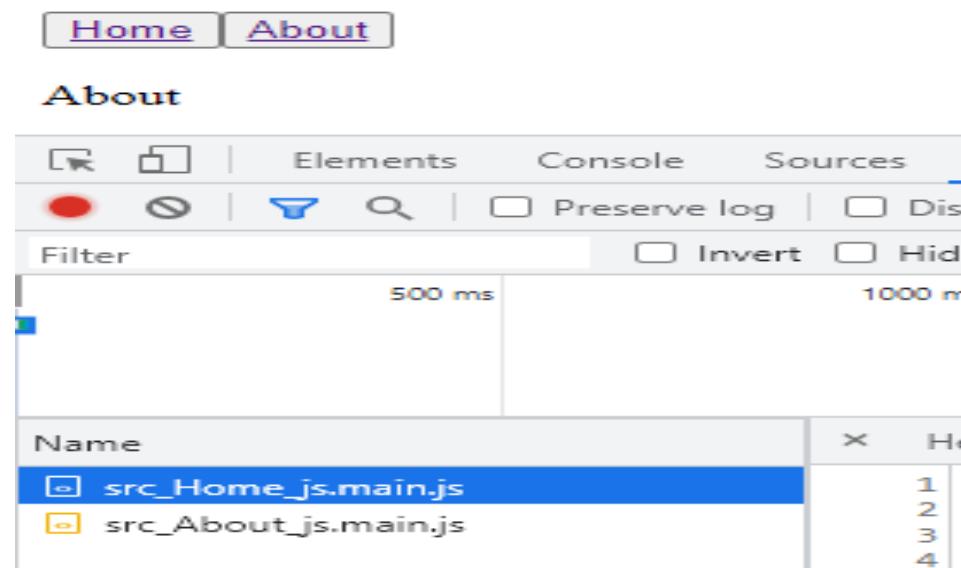
Now, create, Home.js, About.js component,  
Import it into App.js and BrowserRouter in  
index.js. And In App.js , Routing part is there. Now

want to do all components bundle is created separately not in main.js.

#### Step to Follow:

1. First, you install ->  
\$npm i react-loadable
2. Import react-loadable into App.js (main routing component)
3. Go Network , click the button and you can see the home with main.js bundle as well as about.js with main.js bundle.

## Webpack+React



If you want to change, chunk( This webpack-specific term is used internally to manage the bundling process.) name,

The screenshot shows a code editor with several tabs at the top: App.js, webpack.config.js, index.js, and Home.js. The webpack.config.js tab is active, displaying the following code:

```
17     output:{  
18         path:path.join(__dirname,"/dist"),  
19         filename:"main.js",  
20         chunkFilename:'[name].bundle.[fullhash].js',  
21     },  
22  
23 }
```

You can see the bundle name with hash,

```
asset src_About_js.bundle.79fa7766b70d345c3252.js 1  
asset src_Home_js.bundle.79fa7766b70d345c3252.js 1..
```

We want a chunk name , same as file name, So use a magic component. And In App.js, use it,

```
const Home=loadable({  
    loader:()=>import /*webpackChunkName:'HomePage'*/ './Home',  
    loading>LoadingComponent  
})  
  
//For About,if it is loading show the message  
const About=loadable({  
    loader:()=>import /*webpackChunkName:'AboutPage'*/ './About',  
    loading>LoadingComponent  
})
```

Output: Name (ChunkName)

```
asset AboutPage.bundle.30008c1a33bb20f59b7a.js 1.46 KiB [emitted] [immutable]  
(name: AboutPage)  
asset HomePage.bundle.30008c1a33bb20f59b7a.js 1.45 KiB [emitted] [immutable]  
(name: HomePage)
```

[Clean Webpack](#)

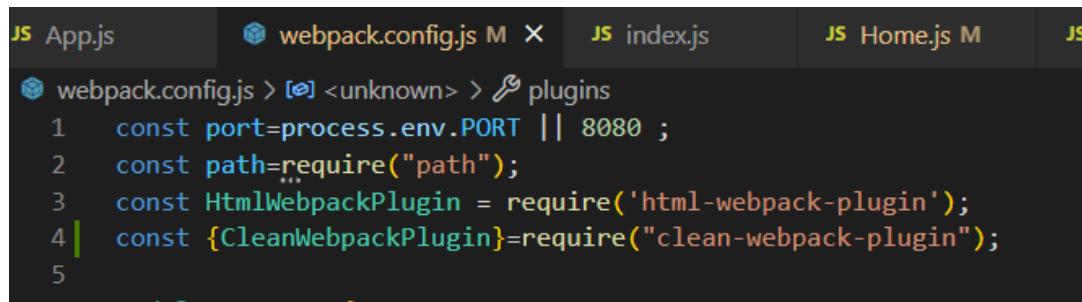
For, any component has comment in code, for this unnecessary webpack create, that's why clear webpack.

When you update anything, webpack creates each and every time. So, want to do only recent files.

```
$npm i clean-webpack-plugin
```

Step1: npm i clean-webpack-plugin

Step2: require it and use it in a plugin into webpack config.



```
js App.js      webpack.config.js M X  JS index.js      JS Home.js M  JS
webpack.config.js > [o] <unknown> > ⚡ plugins
1 const port=process.env.PORT || 8080 ;
2 const path=require("path");
3 const HtmlWebpackPlugin = require('html-webpack-plugin');
4 | const {CleanWebpackPlugin}=require("clean-webpack-plugin");
5
6
7
```

```
plugins: [new HtmlWebpackPlugin({
  template:'src/index.html'
}),
new CleanWebpackPlugin(),
```

Step3: \$npm run build

Then you can see that unnecessary files will be deleted.

- All css files, want to create different webpack.

Step1: npm i mini-css-extract-plugin

Step2:Require it and use it in a plugin into webpack config.And use it module,

```
JS App.js          ⚡ webpack.config.js M X  JS index.js          JS Home.js M
webpack.config.js > [o] <unknown> > ⚡ module > ⚡ rules > ⚡ use
1  const port=process.env.PORT || 8080 ;
2  const path=require("path");
3  const HtmlWebpackPlugin = require('html-webpack-plugin');
4  const {CleanWebpackPlugin}=require("clean-webpack-plugin");
5  const MiniCssExtractPlugin= require("mini-css-extract-plugin")
6
```

```
plugins: [new HtmlWebpackPlugin({
  template:'src/index.html'
}),
new CleanWebpackPlugin(),
new MiniCssExtractPlugin({filename:"[name].[fullhash].css"})]
```

```
module:{  
  rules:[  
    {  
      test:/\.(js|css|jsx)$/,  
      exclude:/node_modules/,  
      use:{  
        loader:"babel-loader"  
      }  
    },  
    {  
      test:/\.css$/,  
      use:[  
        MiniCssExtractPlugin.loader,  
        // "style-loader",  
        "css-loader",  
      ]  
    },  
  ]  
},
```

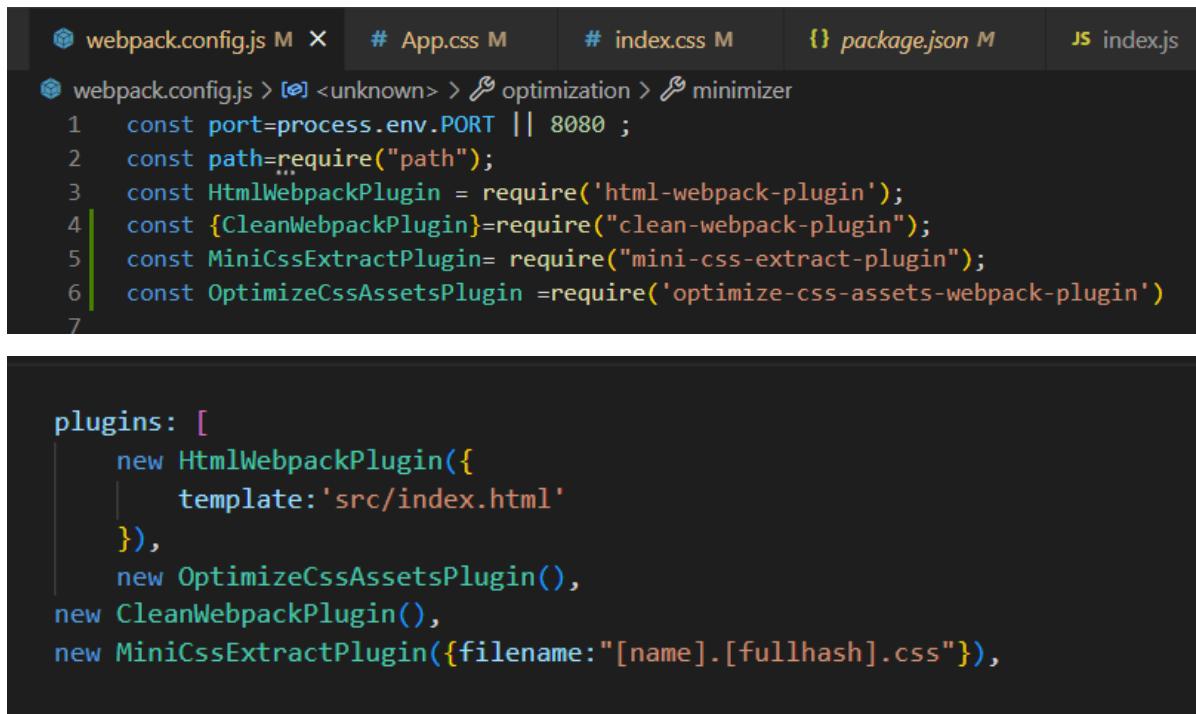
Step3: \$npm run build

See: create a file into dist (main.hash).

- Minimize the css file.

Step1: npm i optimize-css-assets-webpack-plugin

Step2: Require it and use it before plugin into webpack config.



The screenshot shows a code editor with two tabs: 'webpack.config.js' and 'package.json'. The 'webpack.config.js' tab is active, displaying the following code:

```
webpack.config.js M X # App.css M # index.css M {} package.json M JS index.js

webpack.config.js > [o] <unknown> > optimization > minimizer
1 const port=process.env.PORT || 8080 ;
2 const path=require("path");
3 const HtmlWebpackPlugin = require('html-webpack-plugin');
4 const {CleanWebpackPlugin}=require("clean-webpack-plugin");
5 const MiniCssExtractPlugin= require("mini-css-extract-plugin");
6 const OptimizeCssAssetsPlugin =require('optimize-css-assets-webpack-plugin')
7
```

The 'package.json' tab is also visible, showing the following configuration under the 'scripts' section:

```
scripts: {
  "start": "node index.js",
  "build": "webpack --config webpack.config.js"
}
```

Step3: \$npm run build

See that there is a bundle for optimizing css.  
Now, want to do all comment parts are remove, then use minify key,

```
plugins: [
  new HtmlWebpackPlugin({
    template: 'src/index.html',
    minify: {
      removeAttributeQuotes: true,
      collapseWhitespace: true,
      removeComments: true,
    },
  }),
  new OptimizeCssAssetsPlugin(),
]
```

## Webpack Merge

Goal is Splitting dev & production.

Now, create two different files one is a dev file and another is prod file.

I want that when creating a bundle into dev ,it's name as file name but when create bundle into prod ,it will change.

Step1: npm install --save-dev webpack-merge

Step2: Want to call common webpack into dev config and prod config,

```
webpack.prod.js  webpack.dev.js  webpack.prod.js X
o.js  webpack.config.js  webpack.dev.js U  webpack.prod.js U X
webpack.prod.js > [o] <unknown> > ↗ output > ↗ chunkFilename
const common=...require('./webpack.config');
const {merge}=require('webpack-merge')
module.exports=merge(common,{

  output:[
    path:path.join(__dirname,"/dist"),
    filename:"main.js",
    // chunkFilename:'[name].bundle.[fullhash].js', //with hash name
    chunkFilename:'[name].bundle.[fullhash].js', //without hash name
  ],
})

})
```

```
JS App.js  webpack.config.js  webpack.dev.js U X  webpack.prod.js U
webpack.dev.js > [o] <unknown> > ↗ output
1 const common=...require('./webpack.config');
2 const {merge}=require('webpack-merge')
3 module.exports=merge(common,{

4   output:[
5     path:path.join(__dirname,"/dist"),
6     filename:"main.js",
7     // chunkFilename:'[name].bundle.[fullhash].js', //with hash name
8     chunkFilename:'[name].bundle.js', //without hash name
9   ],
10
11
12 },
13
14 })
```

### Step3: Change in package.json

```
19  },
20  ▷ Debug
21  "scripts": {
22    "start": "webpack server --config webpack.dev.js --mode development --open --h
23    "build": "webpack --config webpack.prod.js --mode development",
24    "test": "react-scripts test",
25    "eject": "react-scripts eject"
26  },
27  "eslintConfig": {
```

### Step4: \$npm start

If any **error** arise due to path, then select the path into dev and prod.js,

```
const path=require("path");  
webpack.prod.js - webpack-react - Visual Studio Code  
Help webpack.config.js webpack.dev.js package.json webpack.prod.js  
webpack.prod.js > ...  
1 const common=...  
2 const {merge}=require('webpack-merge');  
3 const path=require("path");|  
4 module.exports=merge(common,{  
5  
6   output:{  
7  
8     path:path.join(__dirname,"/dist"),  
9     filename:"main.js",  
10    // chunkFilename:'[name].bundle.[fullhash].js', //with hash name  
11    chunkFilename:'[name].bundle.[fullhash].js', //without hash name  
12  
13  },  
14}  
15 })
```

```
webpack.dev.js > ...  
1 const common=...  
2 const {merge}=require('webpack-merge');  
3 const path=require("path");  
4 module.exports=merge(common,{  
5  
6   output:{  
7  
8     path:path.join(__dirname,"/dist"),  
9     filename:"main.js",  
10    // chunkFilename:'[name].bundle.[fullhash].js', //with hash name  
11    chunkFilename:'[name].bundle.js', //without hash name  
12  
13  },  
14}  
15 })
```

\$npm start

\$npm run build

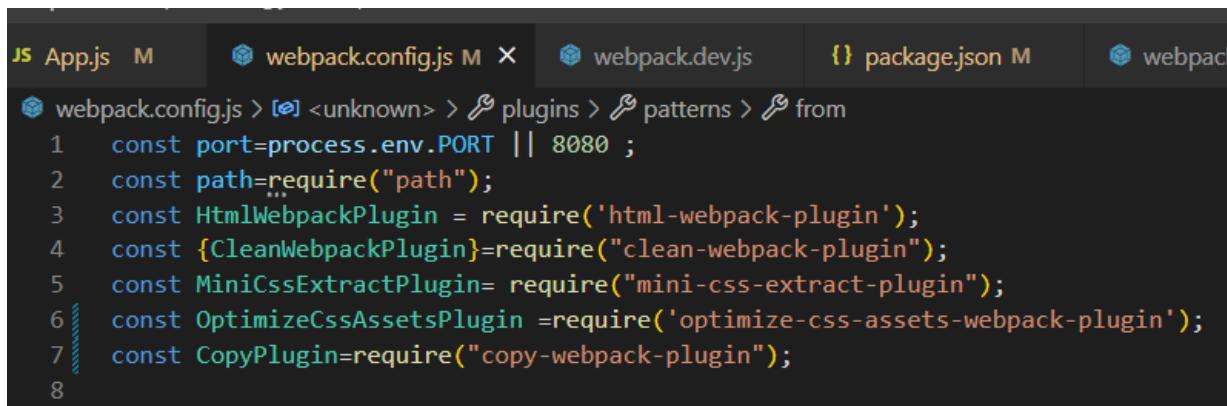
## Copy Webpack Plugin

If another image file, create bundle for image. But when use path into component, then image not loaded.

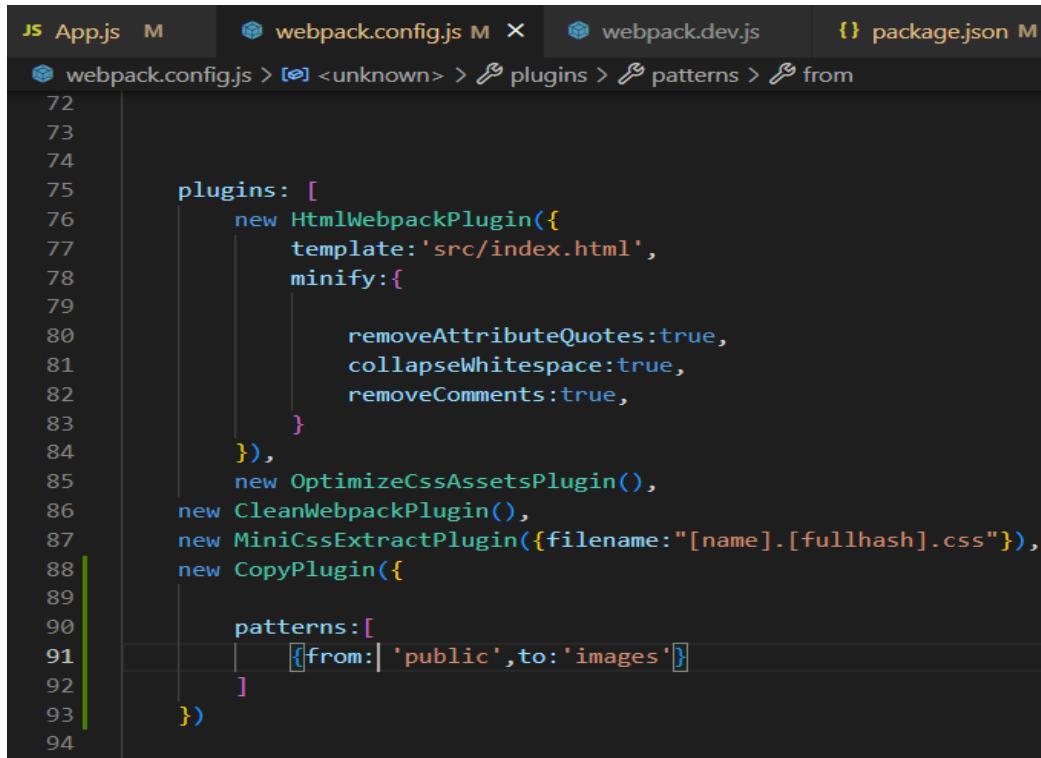
Step1:\$npm i copy-webpack-plugin

For an example, If I want to put any folder into dist.

Step2: Require a copy-webpack-plugin and use it in the plugin.



```
JS App.js M webpack.config.js M X webpack.dev.js {} package.json M webpack
webpack.config.js > [ ] <unknown> > ⚡ plugins > ⚡ patterns > ⚡ from
1 const port=process.env.PORT || 8080 ;
2 const path=require("path");
3 const HtmlWebpackPlugin = require('html-webpack-plugin');
4 const {CleanWebpackPlugin}=require("clean-webpack-plugin");
5 const MiniCssExtractPlugin= require("mini-css-extract-plugin");
6 const OptimizeCssAssetsPlugin =require('optimize-css-assets-webpack-plugin');
7 const CopyPlugin=require("copy-webpack-plugin");
8
```



```
JS App.js M webpack.config.js M X webpack.dev.js {} package.json M
webpack.config.js > [ ] <unknown> > ⚡ plugins > ⚡ patterns > ⚡ from
72
73
74
75     plugins: [
76         new HtmlWebpackPlugin({
77             template:'src/index.html',
78             minify:{
79
80                 removeAttributeQuotes:true,
81                 collapseWhitespace:true,
82                 removeComments:true,
83             }
84         }),
85         new OptimizeCssAssetsPlugin(),
86         new CleanWebpackPlugin(),
87         new MiniCssExtractPlugin({filename:"[name].[fullhash].css"}),
88         new CopyPlugin({
89
90             patterns:[
91                 {from:'public',to:'images'}
92             ]
93         })
94     ]
```

Step: \$npm run build

