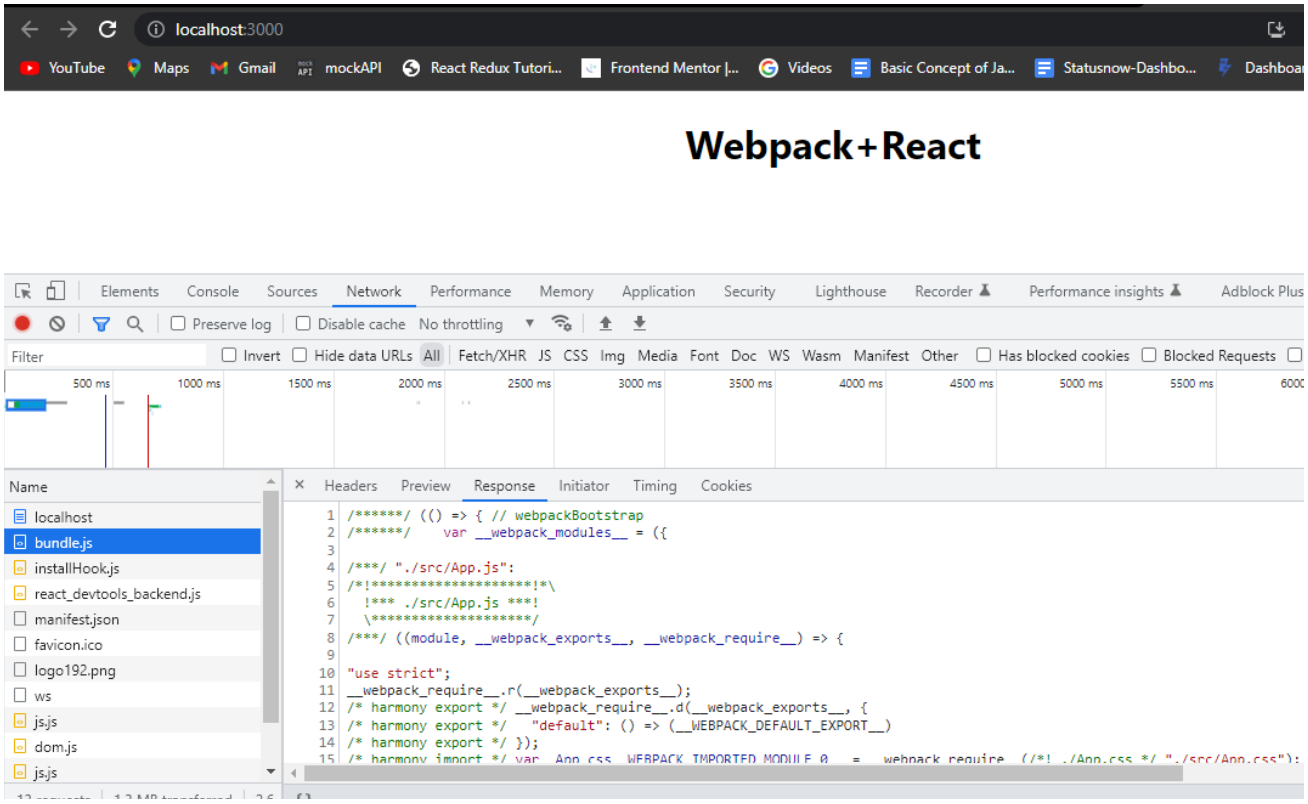
Webpack

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



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 .bablerc->>

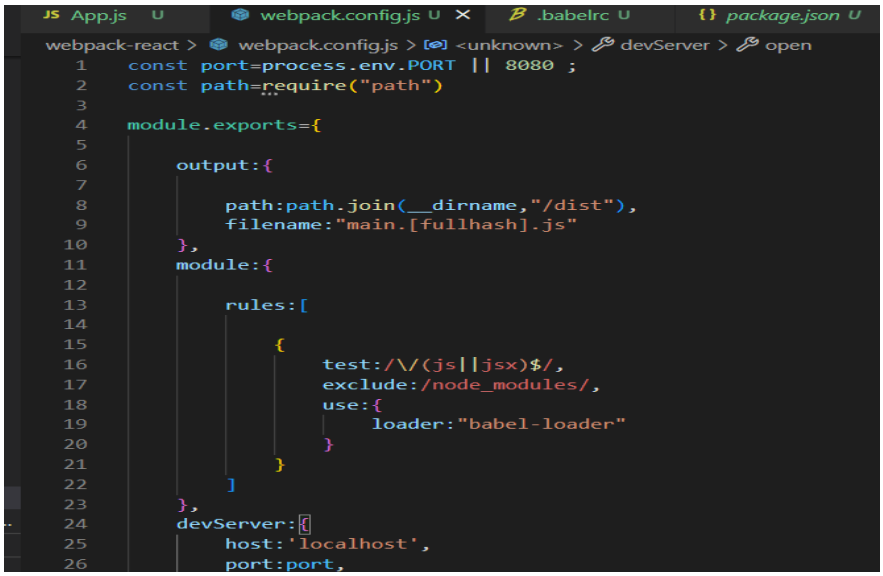
Babel presets can act as a shareable set of Babel plugins and/or config options.

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

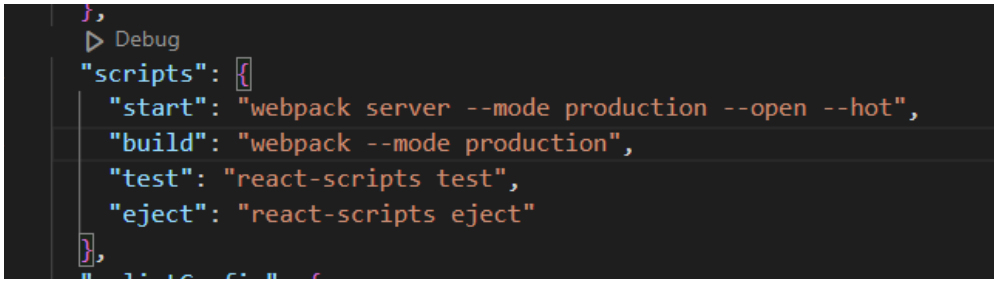
In webpack. config. js->



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

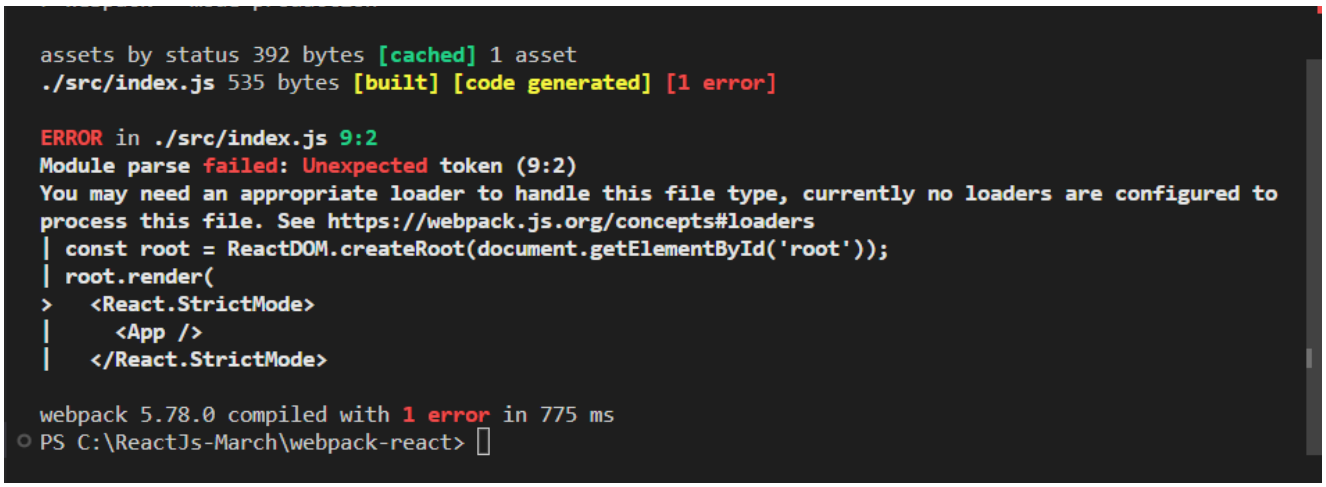
Filename:"main.[fullhash].js" → This creates a different name.

Start with this project webpack.

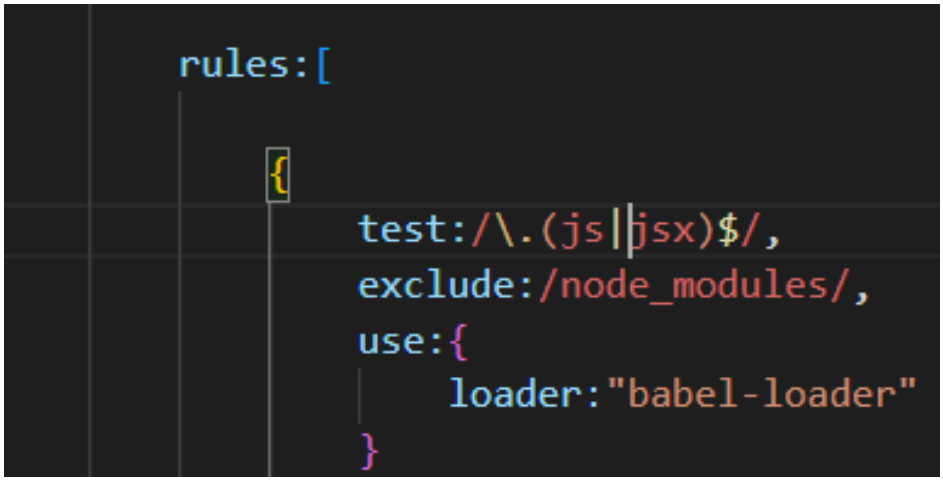


Start with cmd: $npm run build

If any error arise->



Solve this error->



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:

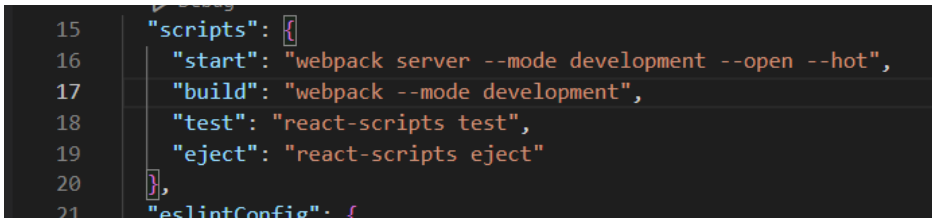
```
const port=process.env.PORT || 8080 ;
const path=require("path");
const HtmlWebpackPlugin = require('html-webpack-plugin');

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
    }
}
```

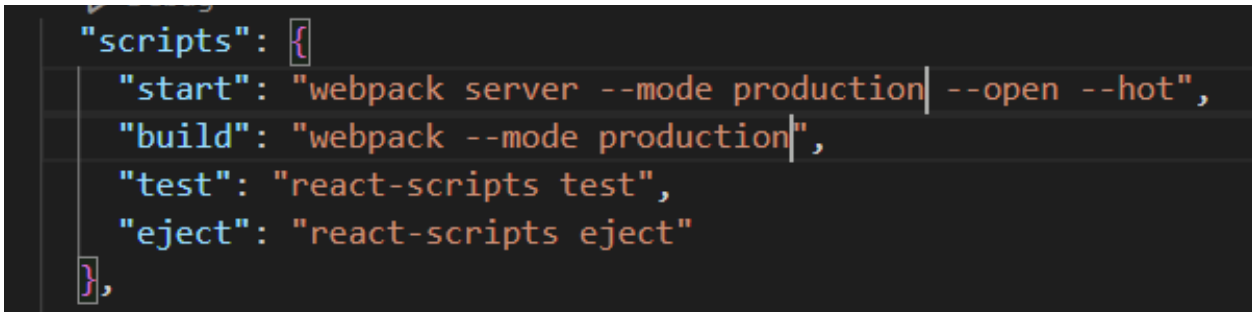
Entry point, Output & Mode

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

Development mode:



Production mode:

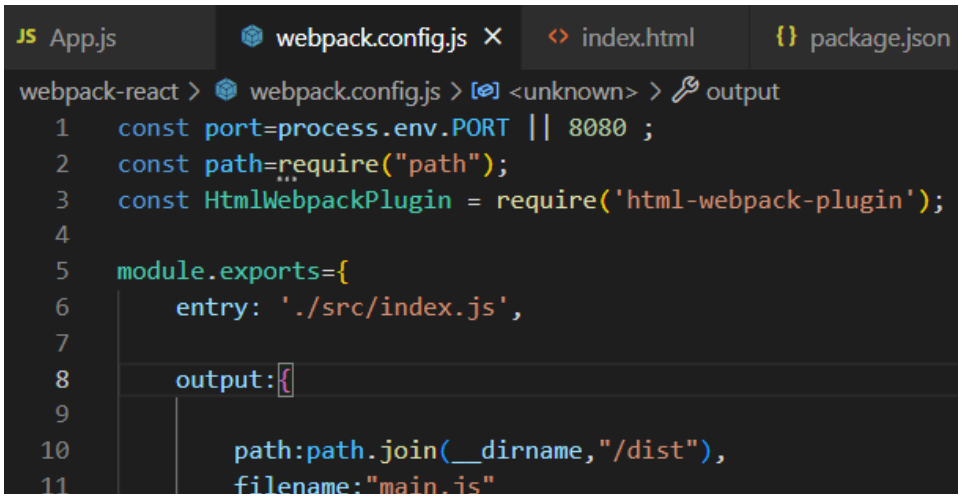


Run this two things->

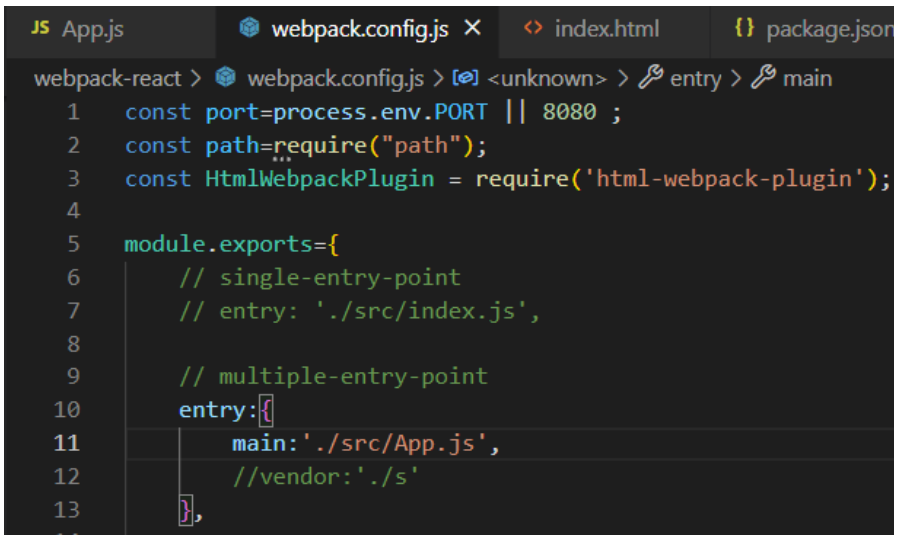
$npm run build

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

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



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



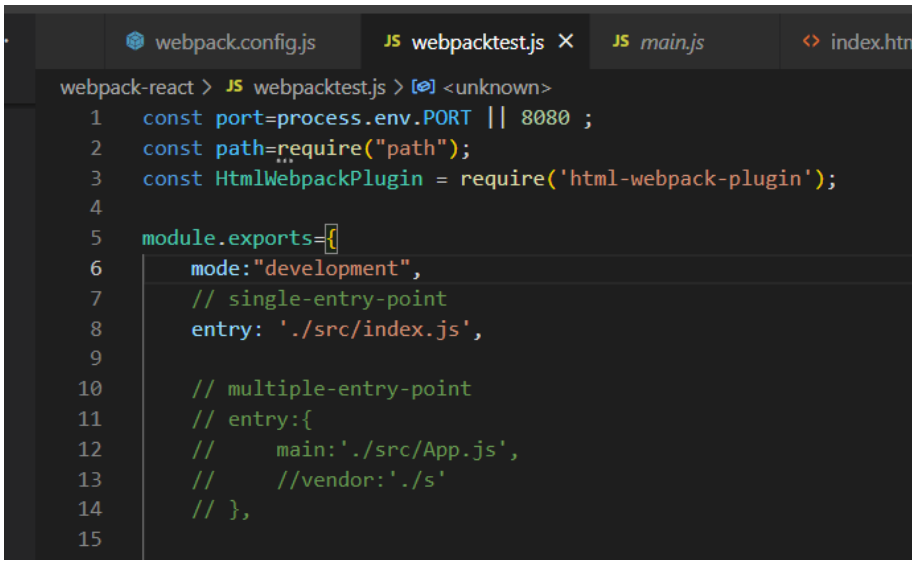
If you want to change entry point,you can change like this,

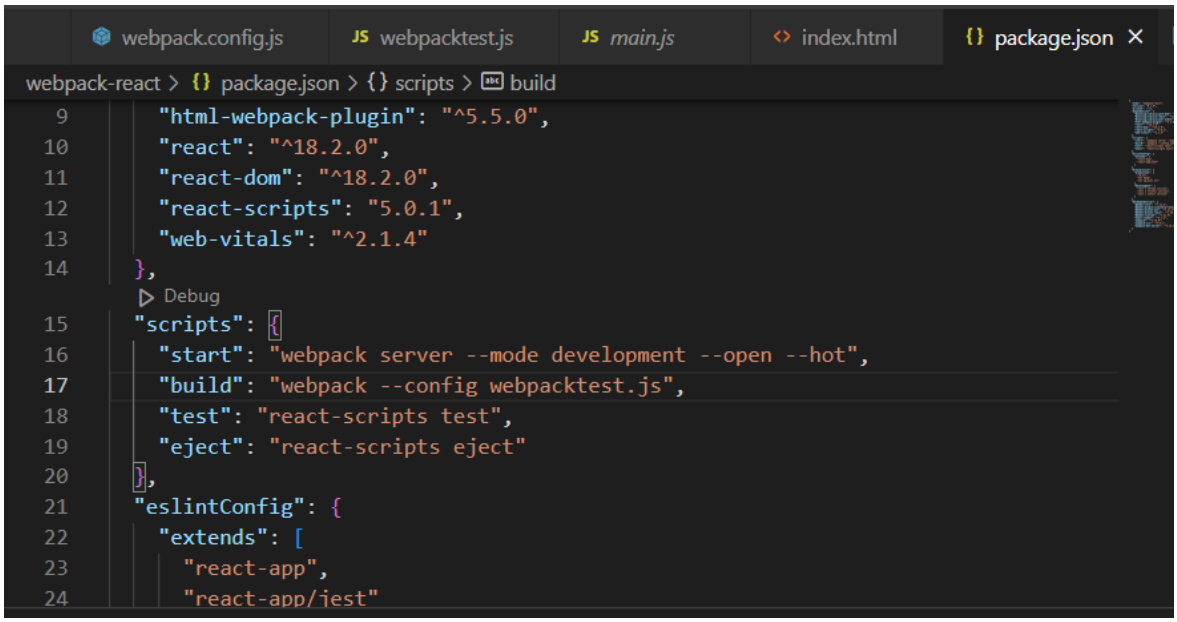
```
entry:{
       main:'./src/<file-name>',
       //vendor:'./s'
   },
```

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.

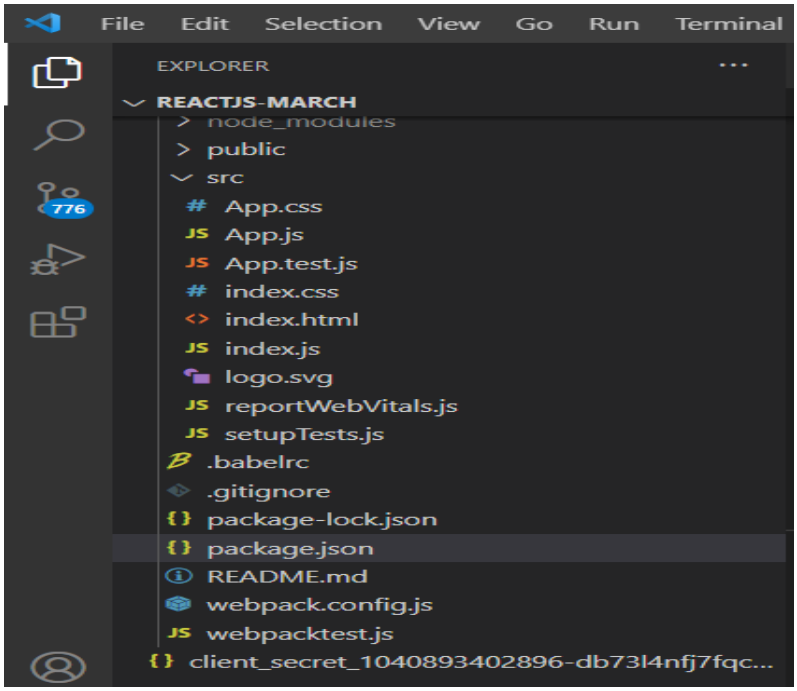




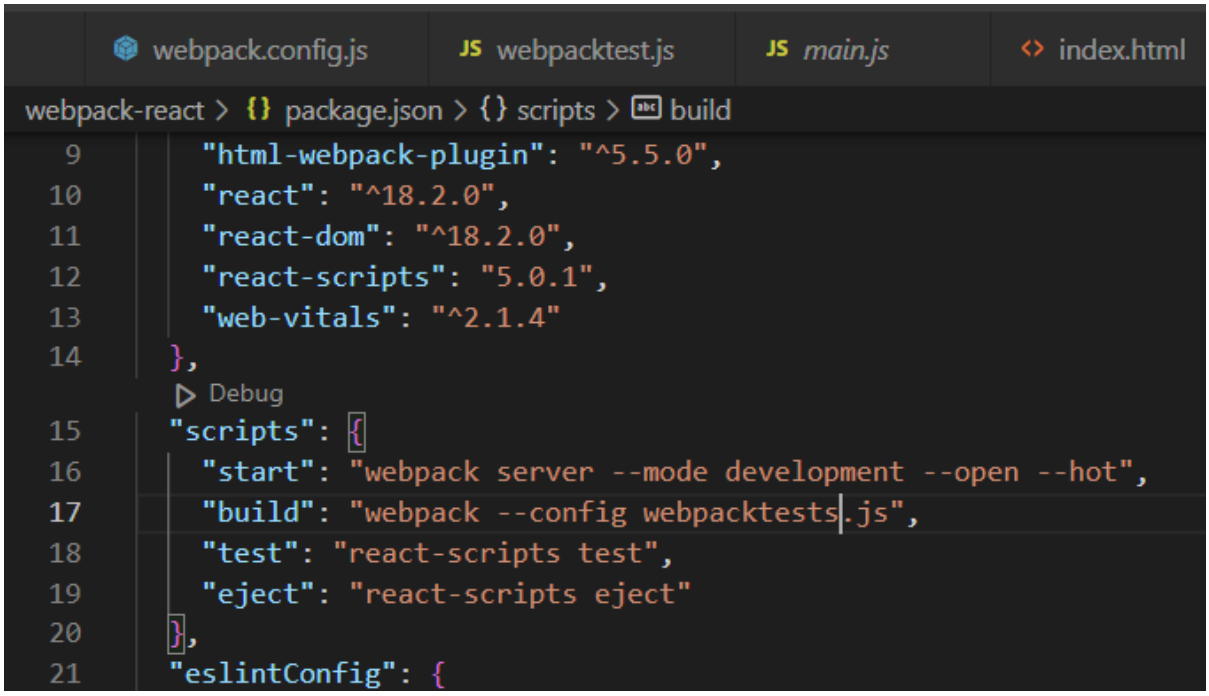

```
"build": "webpack --config <path of the folder>"
```

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

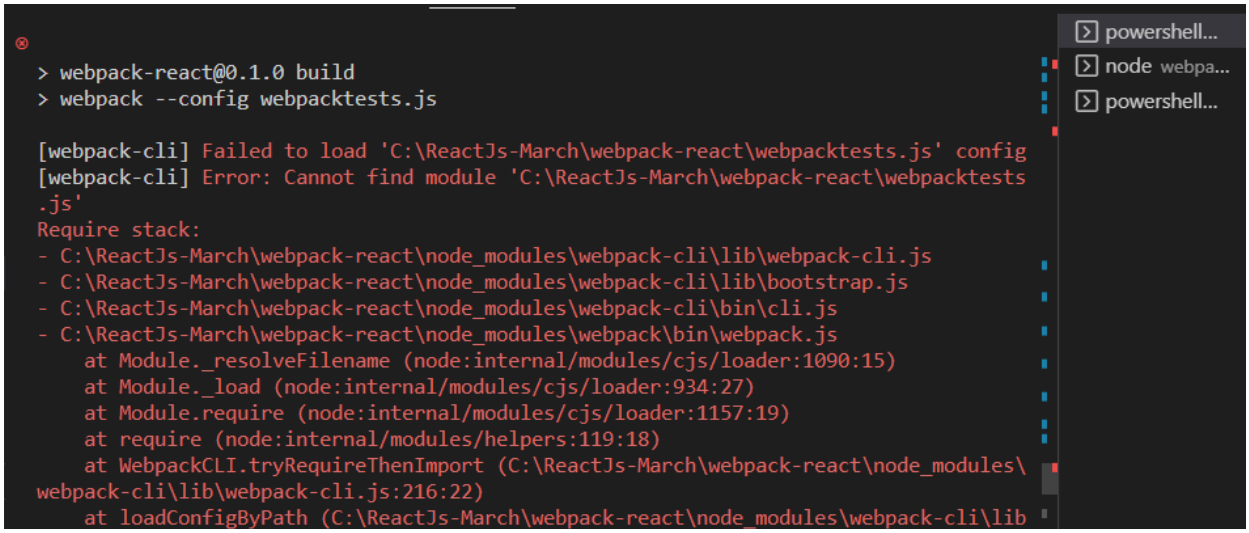
Folder name: webpacktest.js



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



Error->



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

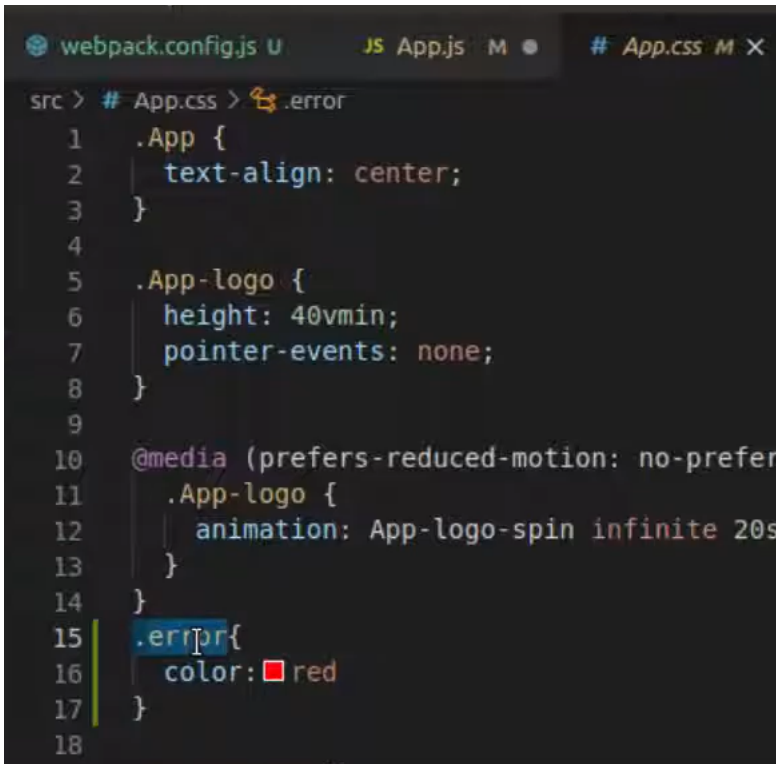
First you need to install file-loader,html-loader.

$npm i file-loader html-loader.

Want to do that color of any html text.

Step to follow->

App.css->



App.js->

webpack.config.js U | App.js M | App.css M | Untitled-1 | package.json

src > App.js > App

development

```
import React from 'react'
import './App.css'
function App() {

  return (
    <div >
      <h1 className="error"><center>Webpack + React</center></h1>

    </div>
  );
}

export default App;
```

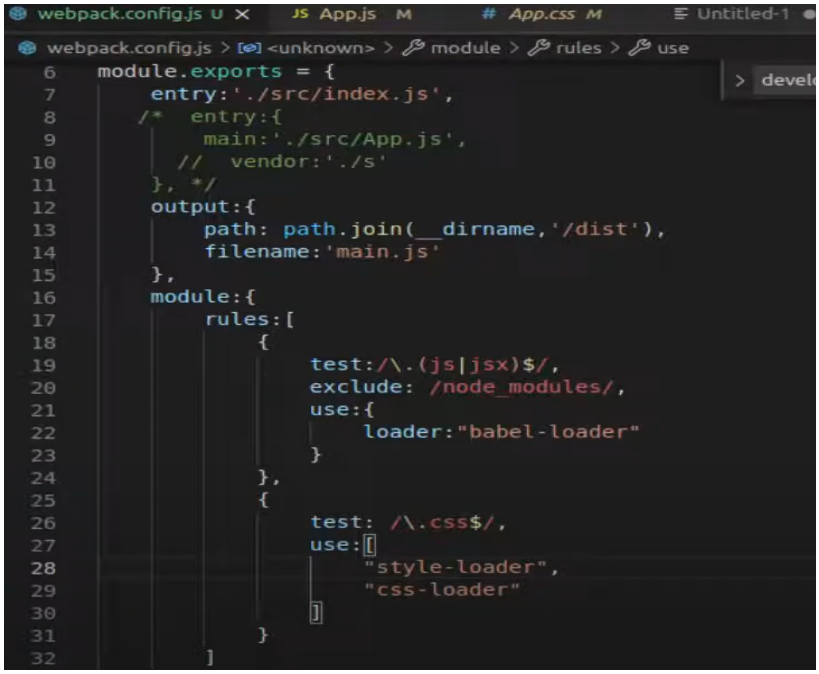
In **webpack.config.js** file(use css loader)

webpack.config.js U | App.js M | App.css M | Untitled-1

webpack.config.js > <unknown> > module > rules > use

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

Use **style-loader**->



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->

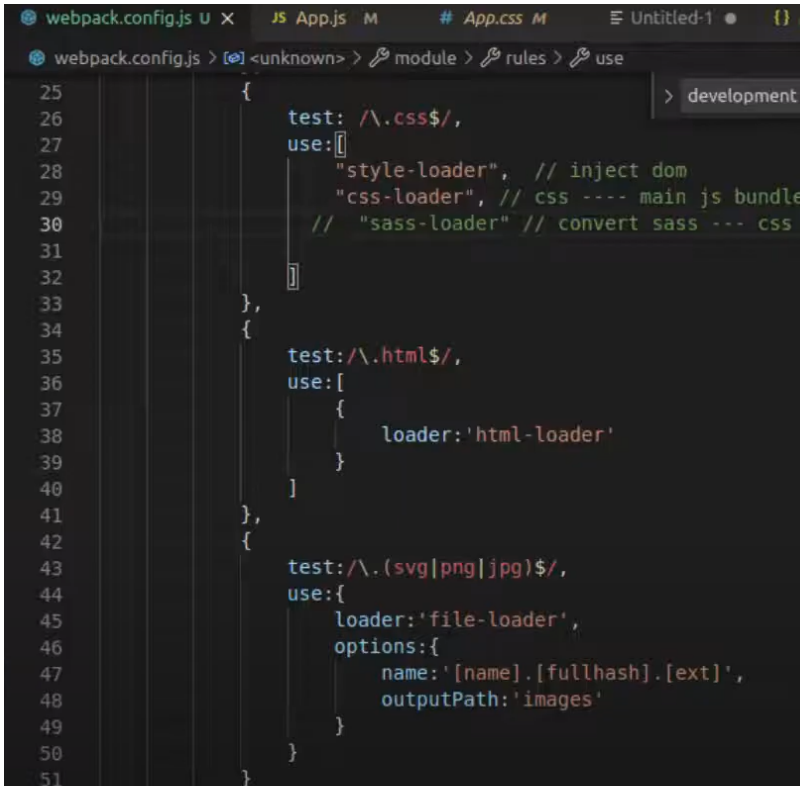
App.js

```
erminal  Help
webpack.config.js U   JS App.js M ×   # App.css M   Untitled-1 ●   {} package.js
src > JS App.js > App
 1  import React from 'react'
 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
14      </div>
15    );
16  }
17
18  export default App;
19
                                                          > development
```

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
17          rules:[                                                  > development
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",  // inject dom
29                      "css-loader", // css ---- main js bundle
30                      "sass-loader" // convert sass --- css
31
32                  ]
33              },
34              {
35                  test:/\.html$/,
36                  use:[
37                      {
38                          loader:'html-loader'
39                      }
40                  ]
41              },
```

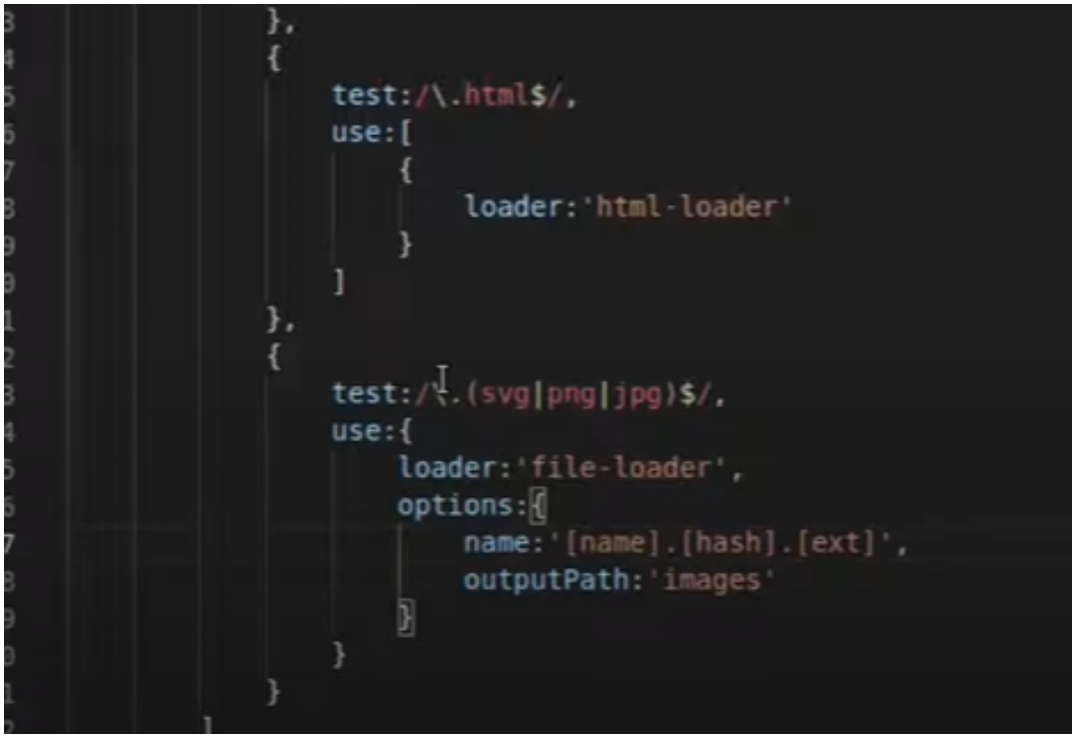


Now need to start->

$npm run build

$npm start

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



```
        },
        {
            test:/\.html$/,
            use:[
                {
                    loader:'html-loader'
                }
            ]
        },
        {
            test:/\.(svg|png|jpg)$/,
            use:{
                loader:'file-loader',
                options:{
                    name:'[name].[hash].[ext]',
                    outputPath:'images'
                }
            }
        }
```

```
$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.

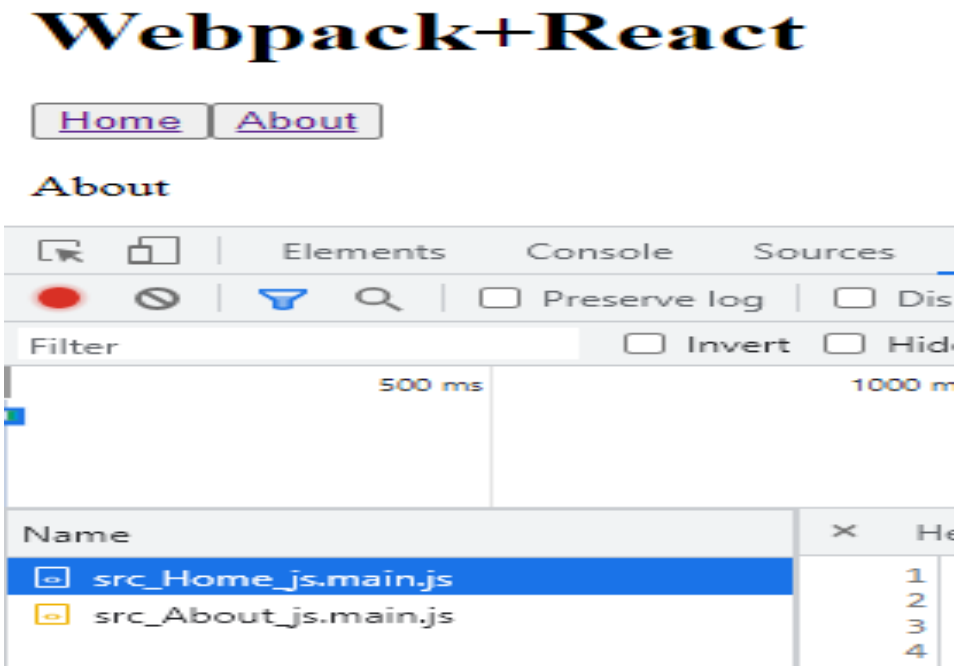
Step to Follow:

1.First,you install ->

$npm i react-loadable

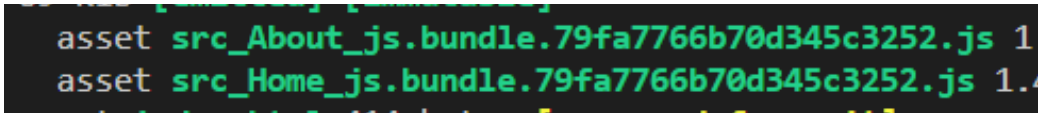
2.Import react-loadable into App.js (main routing component)

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.



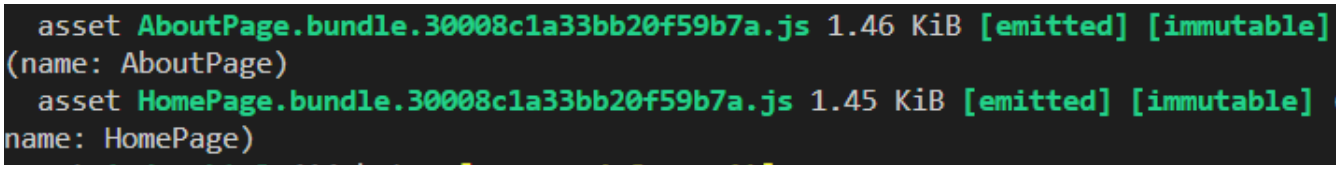
If you want to change,chunk(This webpack-specific term is used internally to manage the bundling process.) name,

You can see the bundle name with hash,



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

Output:Name(ChunkName)

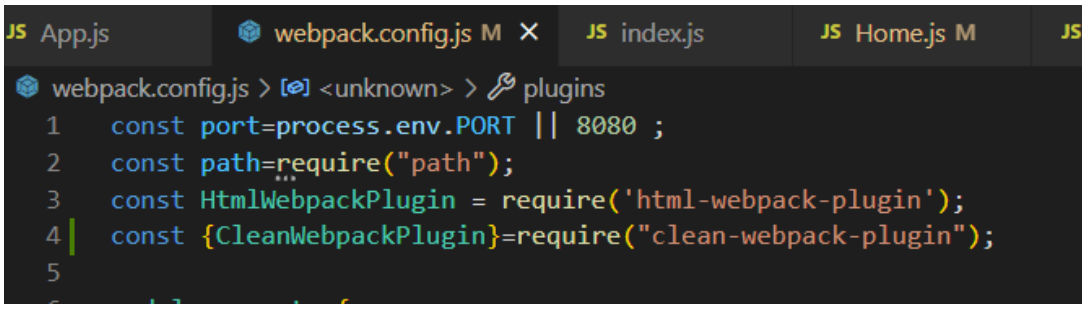


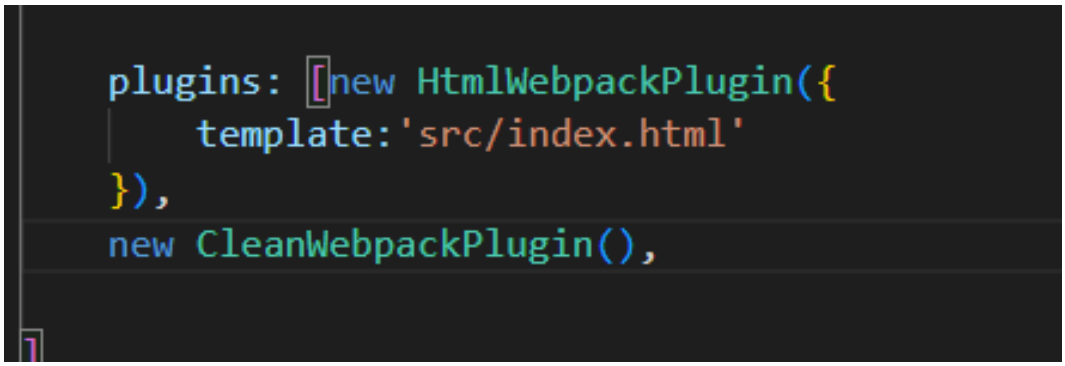
Clean Webpack

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.





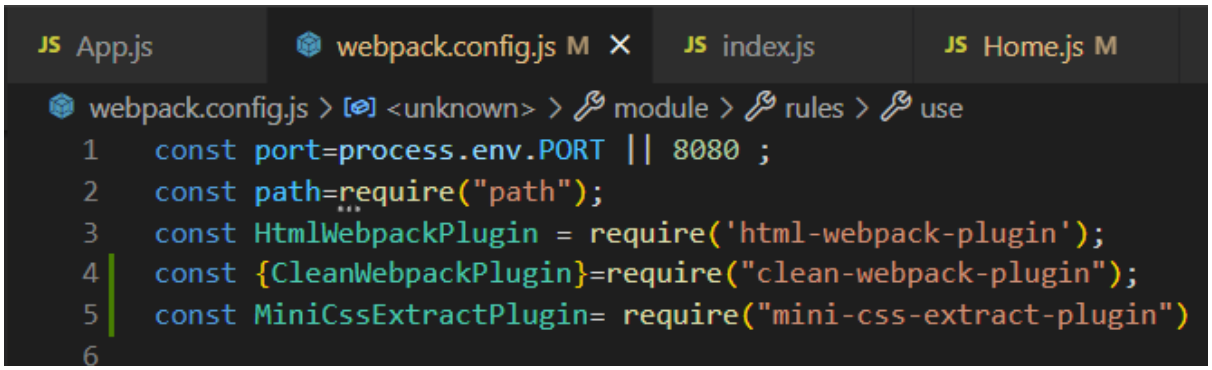
Step3: $npm run build

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,



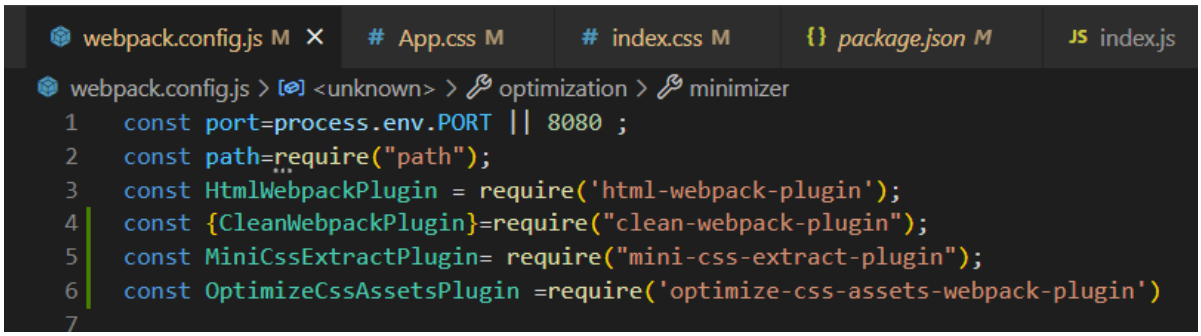
Step3: $npm run build

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

- Minimize the css file.

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

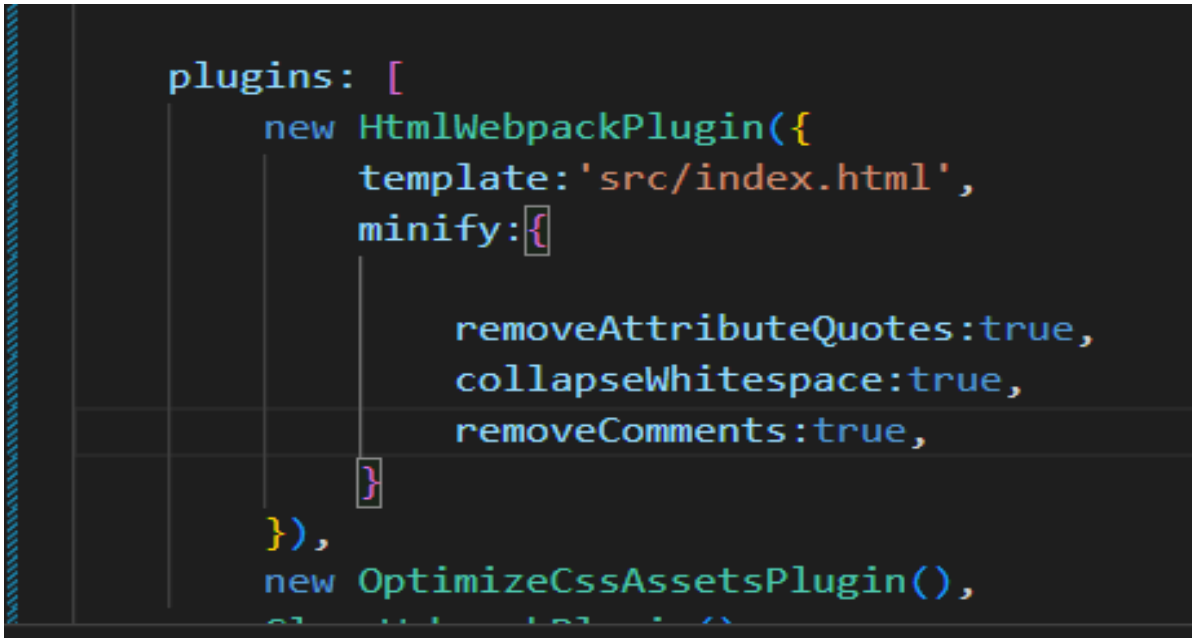
Step2:Require it and use it before plugin into webpack config.




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

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,



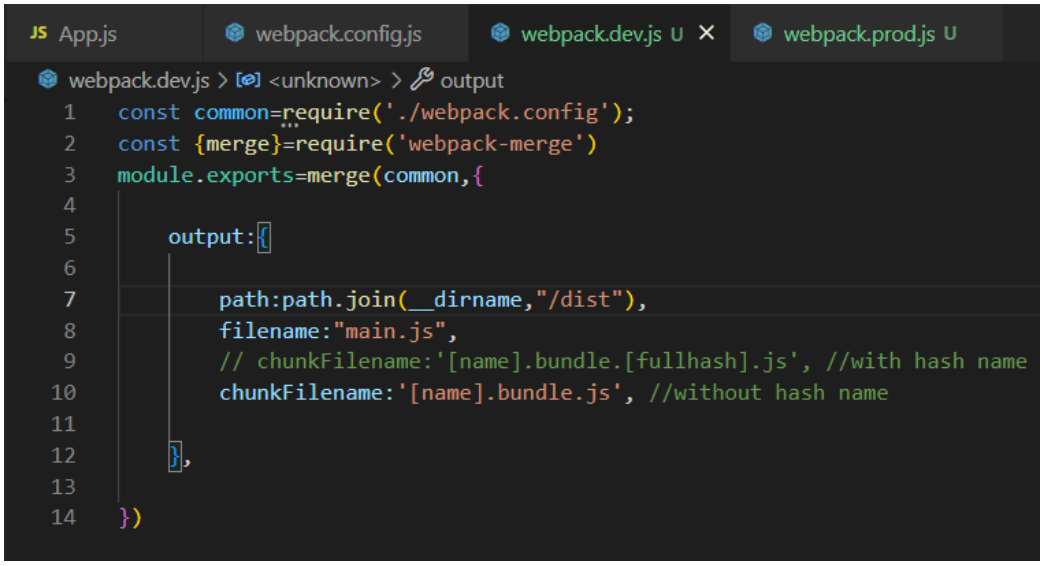
Webpack Merge

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,

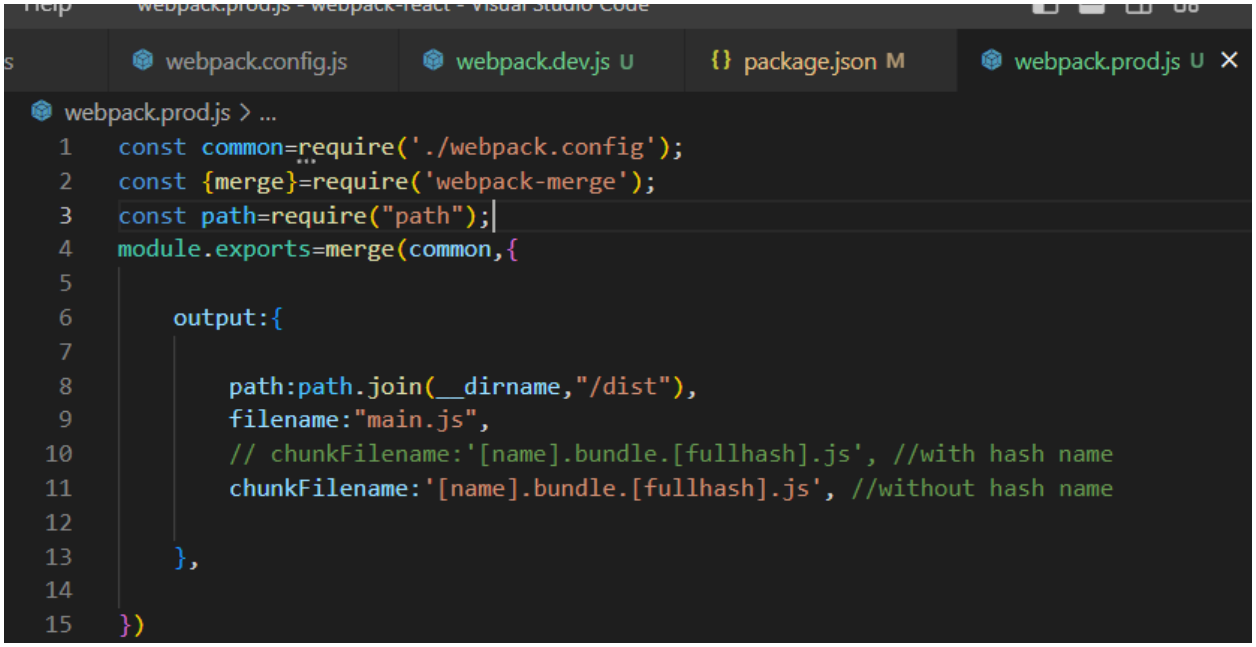


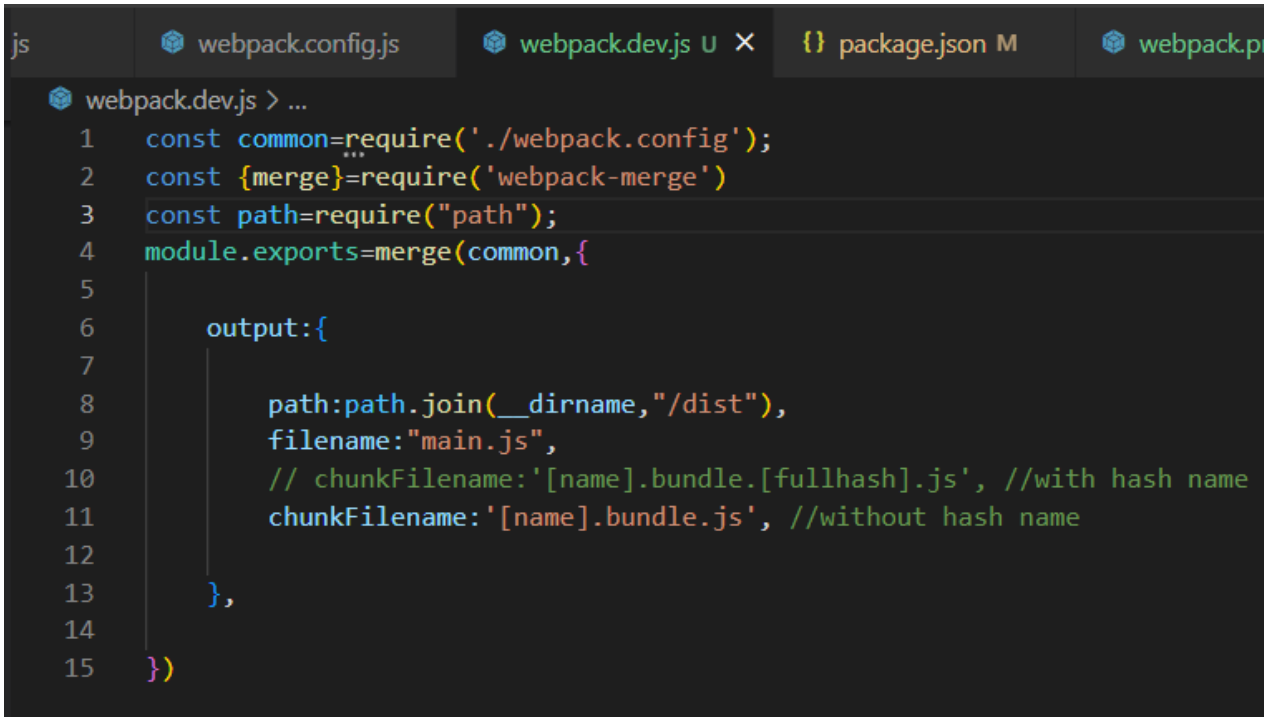
Step3: Change in package.json

Step4: $npm start

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

```
const path=require("path");
```






```
$npm start
$npm run build
```

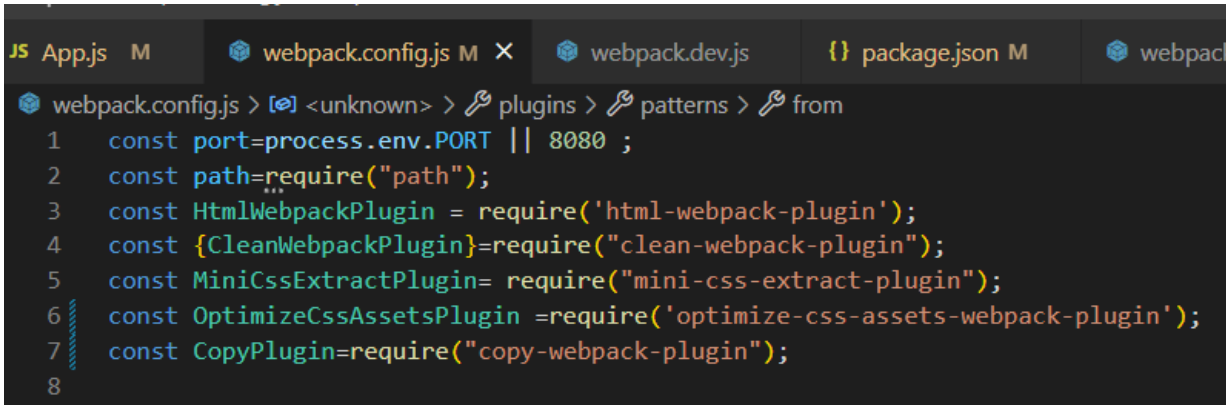
Copy Webpack Plugin

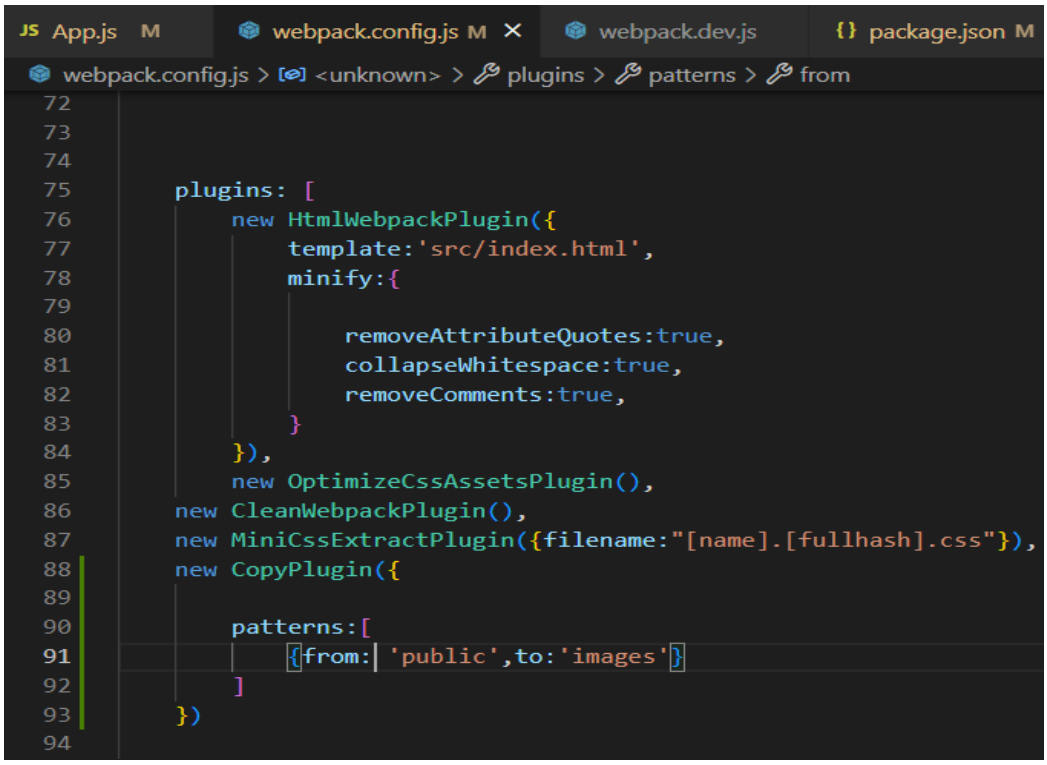
If another image file, create bundle for image.But when use path into component,then image not loaded.

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.





Step: $npm run build