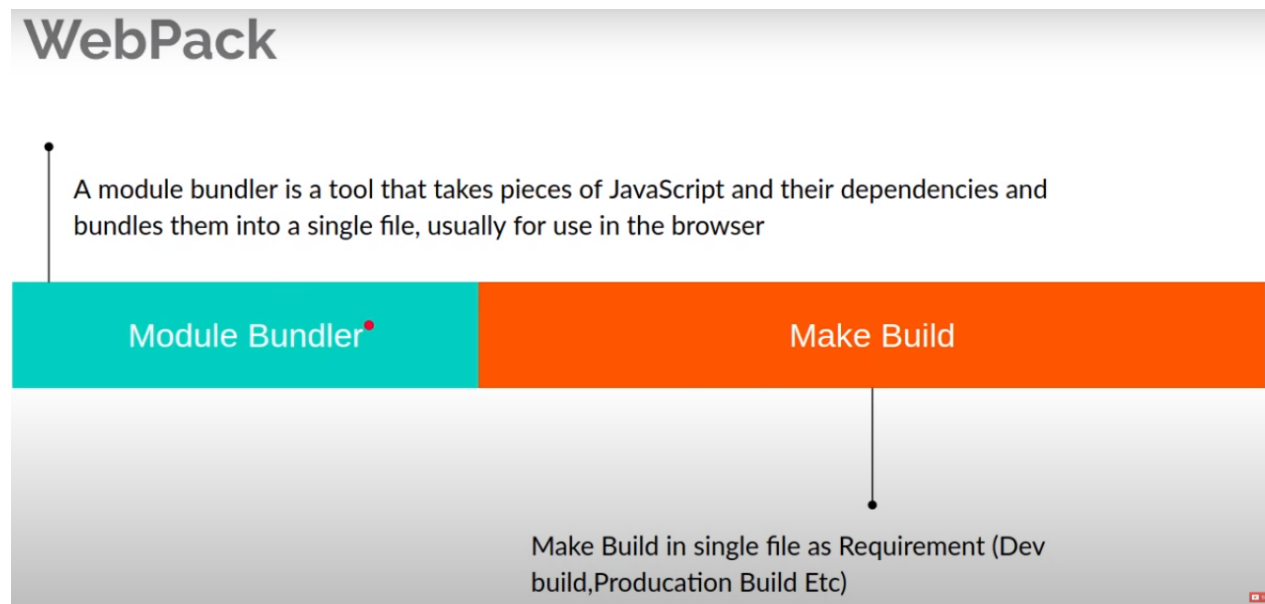
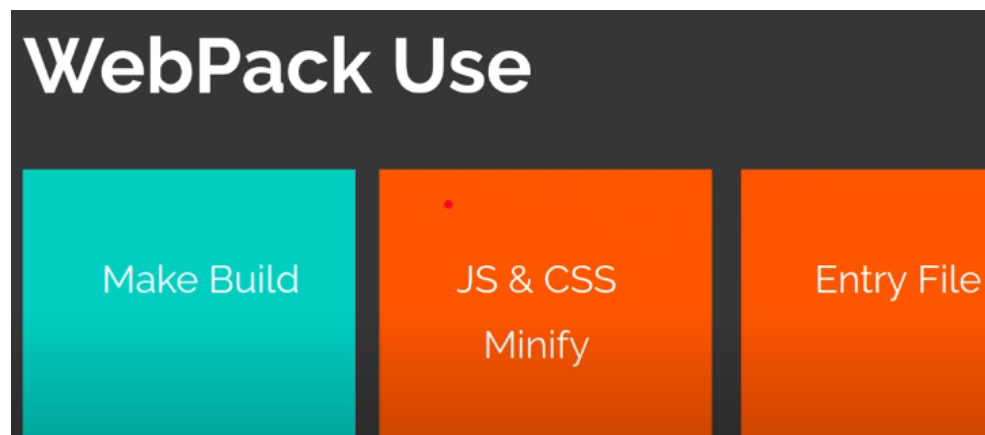
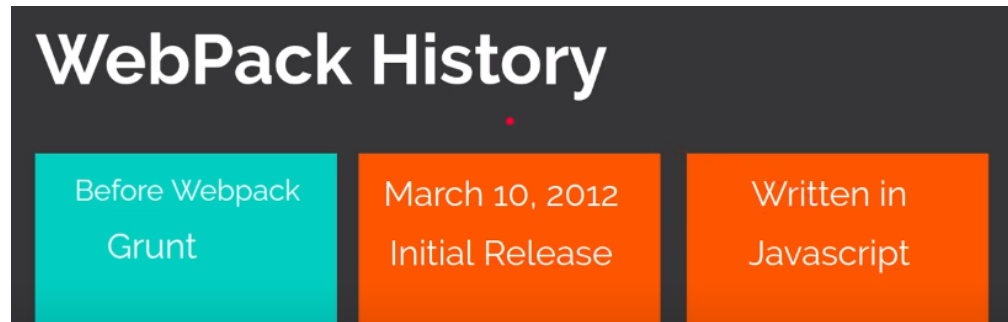


Webpack

Webpack is used to compile JavaScript modules.



- Build for basic JavaScript App:

- Create a folder.

```
$mkdir <folder name>
```

- Create Package.json in CLI

```
$npm init -y
```

When you are the default config of webpack, you need two folders, one is dist and another is src.

In the dist folder, create a file index.html.

In the src, create a file index.js.

How can you bundle two files together, Install webpack dependency.

```
$npm i webpack webpack-cli - -save
```

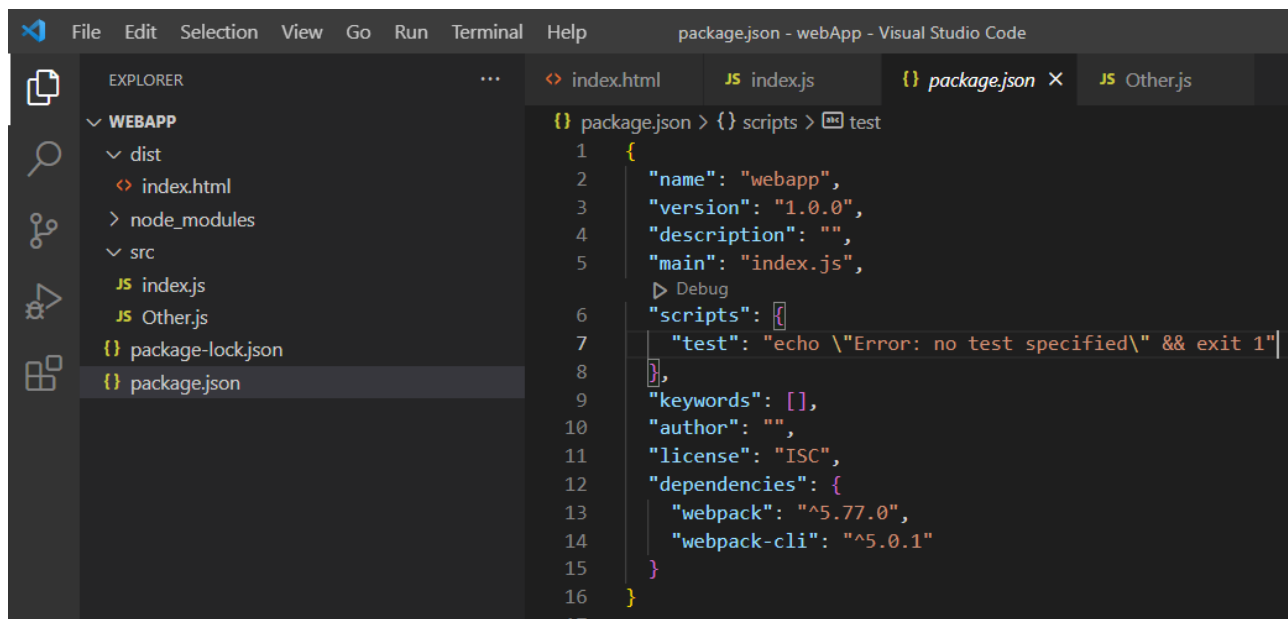
And a package. Json, in script, write "build"

```
"Build": "webpack"
```

Run this in terminal:

```
$npm run <script Name>
```

```
$npm run build
```



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows a project named 'WEBAPP' with a 'src' folder containing 'index.js' and 'Other.js'. The 'package.json' file is selected. The main editor shows the content of 'package.json', which includes fields for 'name', 'version', 'description', 'main', 'scripts', 'keywords', and 'author'. The 'scripts' field has a 'build' script set to 'webpack'. Below the editor, the TERMINAL panel shows the output of the command 'npm run build'. The output indicates that 15 packages are looking for funding, 0 vulnerabilities were found, and the build process completed successfully, generating a bundle for 'main.js'.

```
{
  "name": "webapp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "build": "webpack"
  },
  "keywords": [],
  "author": ""
}
```

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

found 0 vulnerabilities
PS D:\webApp> npm run build

> webapp@1.0.0 build
> webpack

asset main.js 53 bytes [emitted] [minimized] (name: main)
orphan modules 59 bytes [orphan] 1 module
./src/index.js + 1 modules 110 bytes [built] [code generated]

WARNING in configuration
```

Then you see that one file is created,
Main.js

This file is known as a bundle of javascript.

Make Config file

Make a default file for config file, Name of the default file is that **webpack.config.js** (In root folder).

The screenshot shows the VS Code editor with the 'webpack.config.js' file open. The file contains a single line of code: 'module.exports = { mode: "development" }'. The 'mode' property is set to 'development'.

```
module.exports = {
  mode: "development"
}
```

Run this->> `$npm run build`

When it is in `development` and `production` mode, the `main.js` file will change to a different pattern.

Mode: development

```
// webpack.development.config.js
module.exports = {
  mode: 'development',
};
```

Mode: production

```
// webpack.production.config.js
module.exports = {
  mode: 'production',
};
```

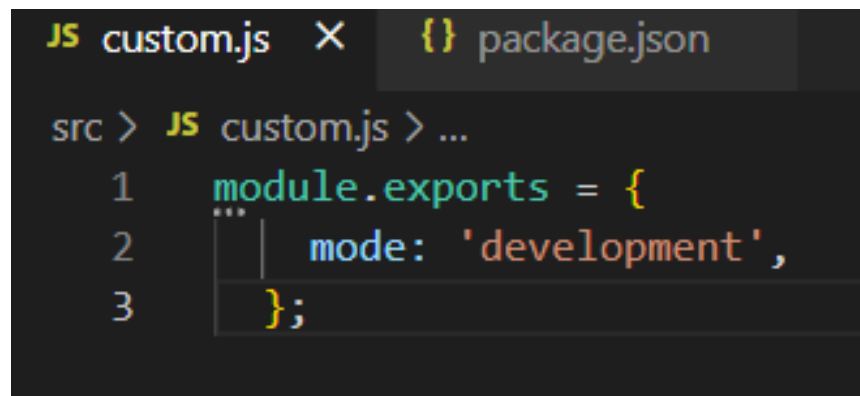
Mode: none

```
// webpack.custom.config.js
module.exports = {
  mode: 'none',
};
```

If you want to change the config name.

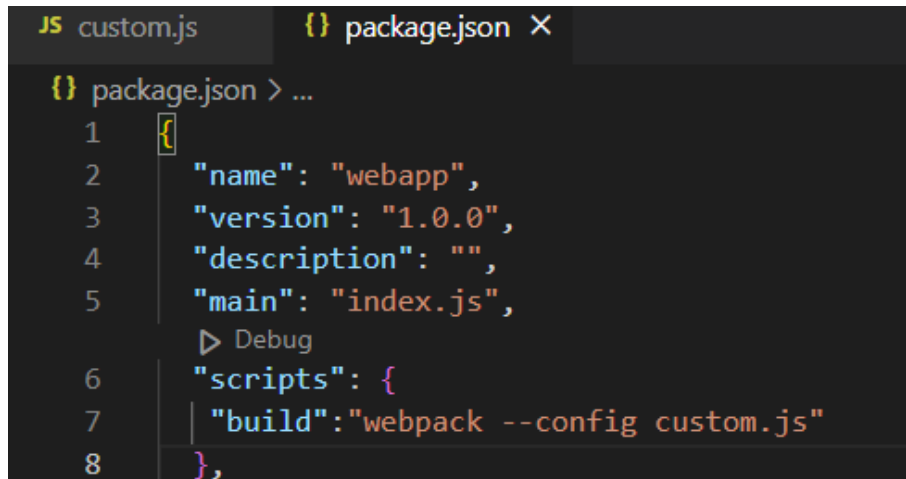
For an example->>>> `custom.js`

Then you write, in a package. Json file change in scripts,



The screenshot shows a code editor with a dark background. At the top, there are two tabs: 'JS custom.js' and '{} package.json'. The 'JS custom.js' tab is active. Below the tabs, the text 'src > JS custom.js > ...' is visible. The main content of the editor shows a JavaScript code snippet:

```
1  module.exports = {
2    ...
3    mode: 'development',
4  };
```



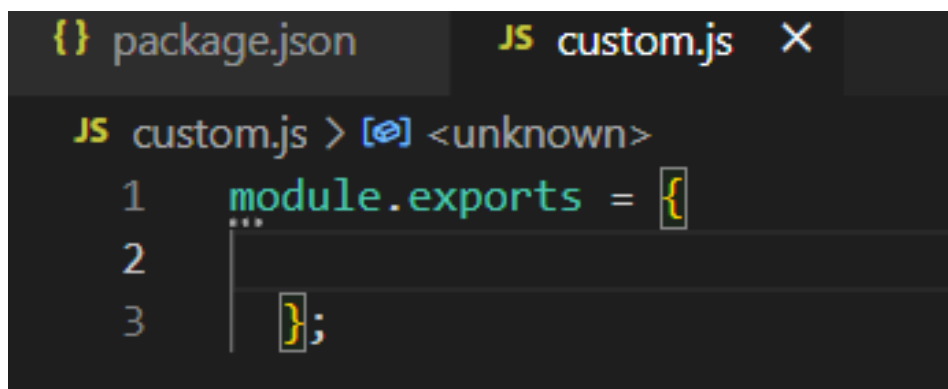
```
JS custom.js {} package.json X
{} package.json > ...
1 {
2   "name": "webapp",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "build": "webpack --config custom.js"
8   },
```

Name syntax->>>

Build: `"webpack --config <file Name>`

NOTE: Always remember that webpack.config.js file is created in the root folder.

If your webpackmode is empty, then it will be in production mode.



```
{} package.json JS custom.js X
JS custom.js > [?] <unknown>
1 module.exports = {
2   ...
3 };
```

Then you `$npm run build`

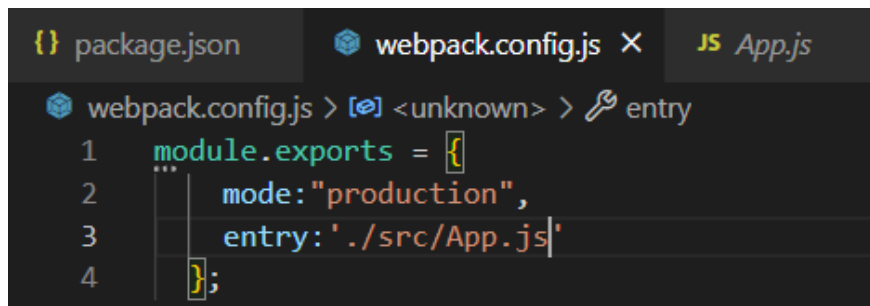
Entry and Output file

By default, the entry file is `index.js`. And the output file is `main.js`, which is build file.

If you want to change input and output file name, For an example, `index.js` to `app.js`, `main.js` to `<file name>`

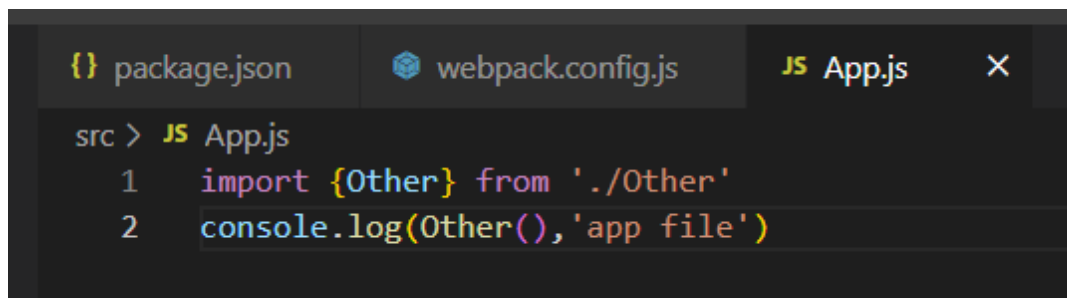
Define entry point-->>

`entry: 'path name'`

A screenshot of a code editor showing the webpack.config.js file. The file has tabs for package.json, webpack.config.js, and JS App.js. The webpack.config.js tab is active, showing the following code:

```
1 module.exports = {  
2   mode: "production",  
3   entry: './src/App.js'  
4 };
```

Entry file-->>

A screenshot of a code editor showing the App.js file. The file has tabs for package.json, webpack.config.js, and JS App.js. The JS App.js tab is active, showing the following code:

```
src > JS App.js  
1 import {Other} from './Other'  
2 console.log(Other(), 'app file')
```

Now, Change the output file, (`webpack.config.js`)
Import path, this path imports from node package.
In `module.exports`, define output with two things, one is `path` and another is `file name`.

```
{} package.json  webpack.config.js X JS App.js
webpack.config.js > [?] <unknown> > [?] output > [?] filename
1  //import path for change the output file
2  const path=require('path')
3  module.exports = {
4      mode:"production",
5      entry: './src/App.js',
6      output:{
7
8          path:path.resolve(__dirname,'dist'),
9          filename:"output.js"
10     }
11 };
```

Run this-->> `$npm run build`

These output files want to run, then also change in index.html's script.

```
{} package.json  webpack.config.js  <> index.html X JS App.js
dist > <> index.html > [?] html > [?] body > [?] script
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Webpack</title>
8  </head>
9  <body>
10     <script src="output.js"></script>
11 </body>
12 </html>
```

Run this output in the browser.

Dev Server

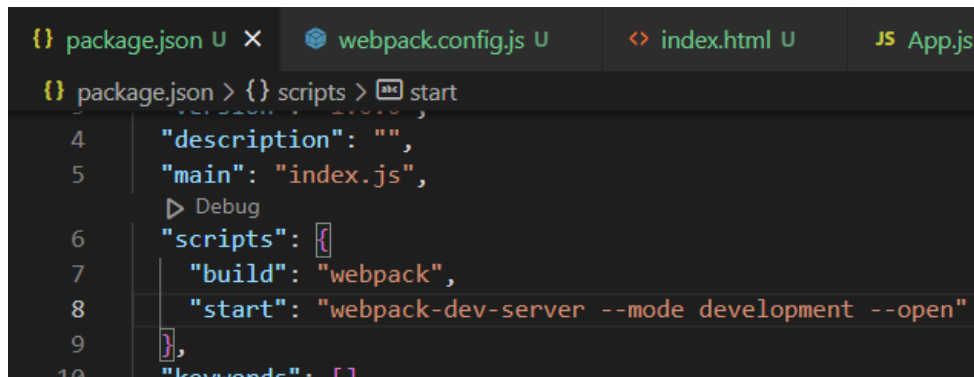
We want to It is automatically run.

Install webpack server or webpack dev server.

```
$npm i webpack-dev-server --save-dev
```

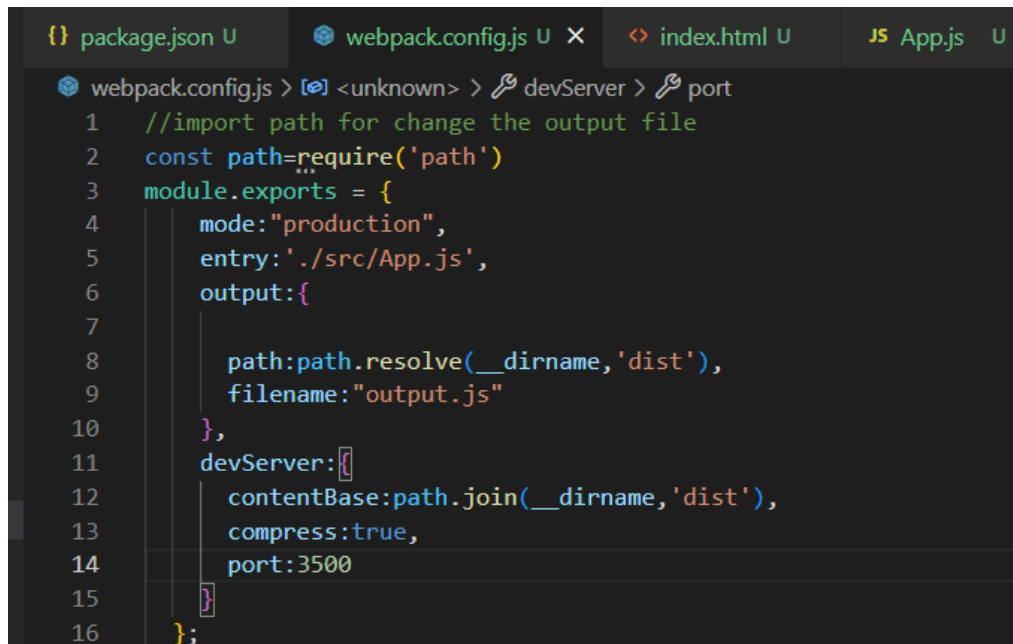
Save-dev for dev dependencies.

Change the `package.json`, scripts->> starts

A screenshot of a code editor with four tabs: package.json, webpack.config.js, index.html, and JS App.js. The package.json tab is active, showing a JSON object with a 'scripts' property. The 'scripts' object contains 'build' set to 'webpack' and 'start' set to 'webpack-dev-server --mode development --open'.

```
{  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "build": "webpack",  
    "start": "webpack-dev-server --mode development --open"  
  },  
  "keywords": []  
}
```

Change also `webpack.config.js`.

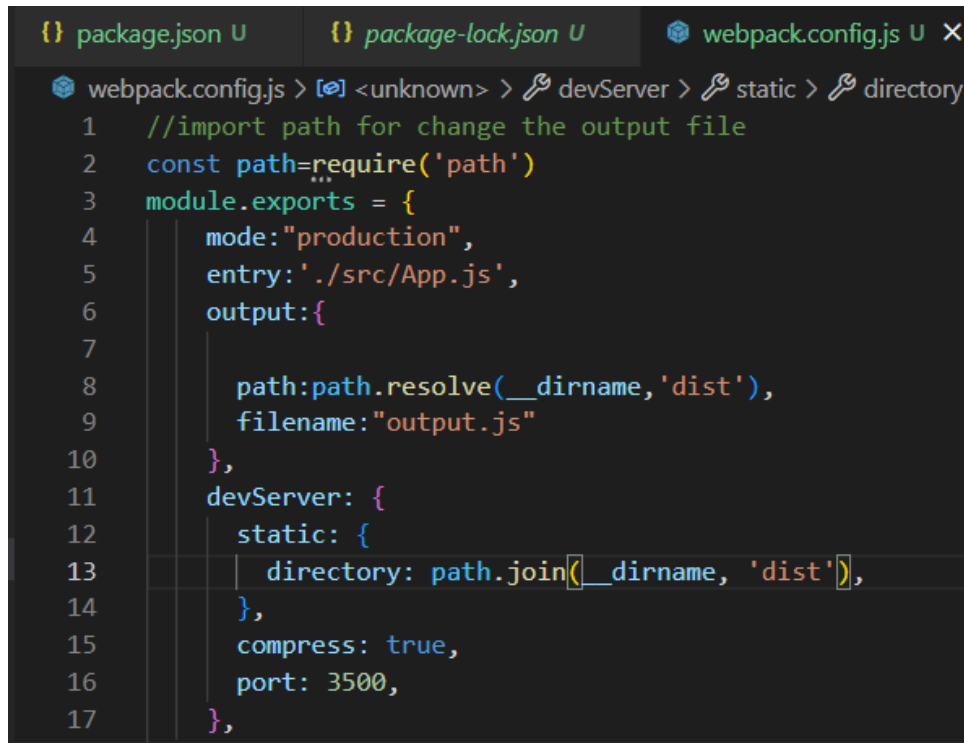
A screenshot of a code editor with four tabs: package.json, webpack.config.js, index.html, and JS App.js. The webpack.config.js tab is active, showing a JavaScript file. It imports 'path' and defines a configuration object for Webpack Dev Server. The 'devServer' object includes 'contentBase' (path to 'dist'), 'compress' (true), and 'port' (3500).

```
//import path for change the output file  
const path=require('path')  
module.exports = {  
  mode:"production",  
  entry: './src/App.js',  
  output:{  
    path:path.resolve(__dirname,'dist'),  
    filename:"output.js"  
  },  
  devServer:{  
    contentBase:path.join(__dirname,'dist'),  
    compress:true,  
    port:3500  
  }  
};
```

In devServer: 3 key value pair added- ->>
`contentBase,compress,port`.

Then this is run - ->

\$npm start



```
webpack.config.js > <unknown> > devServer > static > directory
1 //import path for change the output file
2 const path=require('path')
3 module.exports = {
4   mode:"production",
5   entry: './src/App.js',
6   output:{
7
8     path:path.resolve(__dirname,'dist'),
9     filename:"output.js"
10  },
11  devServer: {
12    static: {
13      directory: path.join(__dirname, 'dist'),
14    },
15    compress: true,
16    port: 3500,
17  },
```

Link->>>

<https://webpack.js.org/configuration/dev-server/#devserver>

CSS and style Loader

Use css and style with webpack.

Create file in src for styling.

Import it into app.js, it is throwing an parse error.

```
package-lock.json U  webpack.config.js U  # style.css U  <> index.html U  JS App.js U X

src > JS App.js
1  import {Other} from './Other'
2  import './style.css'
3  console.log(Other(), 'app file')

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  node + - []

asset main.65b62e3fa765cec500b4.hot-update.json 28 bytes [emitted] [immutable] [hmr]
Entrypoint main 264 KiB = output.js 262 KiB main.65b62e3fa765cec500b4.hot-update.js 2.04
cached modules 173 KiB [cached] 27 modules
runtime modules 27.3 KiB 12 modules
javascript modules 114 bytes
./src/App.js 84 bytes [built] [code generated]
./src/style.css 30 bytes [built] [code generated] [1 error]

ERROR in ./src/style.css 1:0
Module parse failed: Unexpected token (1:0)
You may need an appropriate loader to handle this file type, currently no loaders are co
process this file. See https://webpack.js.org/concepts#loaders
> .hello{
|
|   color: aqua;
| }
```

How to resolve this error?

->> For this install two packages for this,
One is a style-loader and another css-loader.

```
package-lock.json U  webpack.config.js U  # style.css U

src > JS App.js
1  import {Other} from './Other'
2  import './style.css'
3  console.log(Other(), 'app file')

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\webApp> npm i --save-dev style-loader css-loader
```

Some code written into `webpack.config.js`.

Run this css-> `$npm start`

Different config file environments

How to create files for different mode, production and development environments.

For an example--> create two files, like `config.dev.js` and `config.prod.js`.

The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the project structure. The main editor area shows the `webpack.config.js` file. The configuration is for development mode, with the output file set to `dist/index.html`. The `devServer` is configured to serve the application from the `dist` directory.

```
1 //import path for change the output file
2 const path = require('path')
3 module.exports = {
4   mode: "production",
5   entry: './src/App.js',
6
7   //loader
8   module: {
9
10    rules: [
11      {
12        test: /\.css$/,
13        use: [
14          'style-loader',
15          'css-loader'
16        ]
17      }
18    ]
19  },
20  //loader
21  devServer: {
22    static: {
23      directory: path.join(__dirname, 'dist'),
24    },
25    compress: true,
```

The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the project structure. The main editor area shows the `webpack.config.js` file. The configuration is for production mode, with the output file set to `dist/output.js`. The `devServer` is configured to serve the application from the `dist` directory.

```
1 //import path for change the output file
2 const path = require('path')
3 module.exports = {
4   mode: "production",
5   entry: './src/App.js',
6   output: {
7
8     path: path.resolve(__dirname, 'dist'),
9     filename: "output.js"
10  },
11  //loader
12  module: {
13
14    rules: [
15      {
16        test: /\.css$/,
17        use: [
18          'style-loader',
19          'css-loader'
20        ]
21      }
22    ]
23  }
```

Start development server->>>

```
$npm start
```

Start production server->>>

```
$npm run build
```