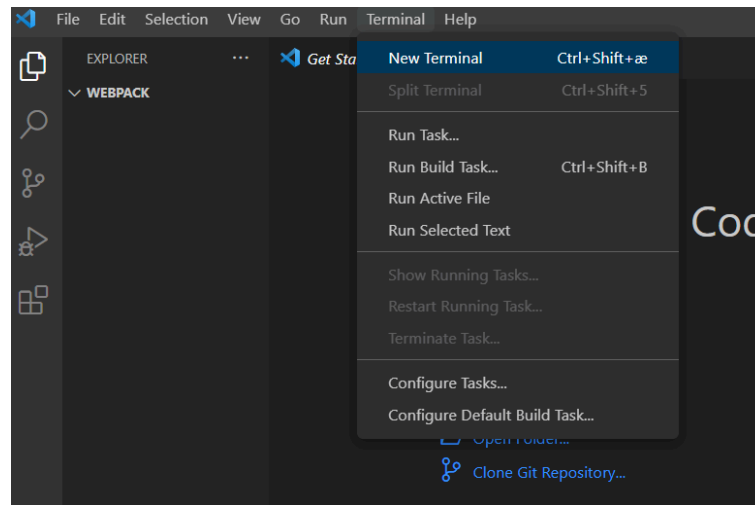


Installation

- To install webpack, you start by opening a new terminal by navigating to Terminal and select New Terminal.



- Open Git Bash, which as a default has access to npm and npx (npm is a Node Package Manager and npx is a Node Package Executor)

Konfiguration

- start by making a folder called webpack-styles, you can do that by typing **"mkdir webpack-styles"** in the terminal
- then you have to navigate to your folder, you can do that by typing **"cd webpack-styles"** in the terminal

```
U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skrivebord/Hjemmeside opgave 2.0/Hovedforløb/Automatisering/Uge 3/webpack
$ mkdir webpack-styles

U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skrivebord/Hjemmeside opgave 2.0/Hovedforløb/Automatisering/Uge 3/webpack
$ cd webpack-styles
```

- now you have to make a package, you can do that by typing **"npm init -y"** in the terminal
-y means yes so you don't have to type it after

```
U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skrivebord/Hjemmeside opgave 2.0/Hovedforløb/Automatisering/Uge 3/webpack/webpack-styles
$ npm init -y
Wrote to C:\Users\U87960\OneDrive - Roskilde Tekniske Skole\Skrivebord\Hjemmeside opgave 2.0\Hovedforløb\Automatisering\Uge 3\webpack\webpack-styles\package.json:

{
  "name": "webpack-styles",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

- now you have to add some modules, you can do that by typing “**npm i -D webpack webpack-cli**” in the terminal
-D stands for devDependencies.

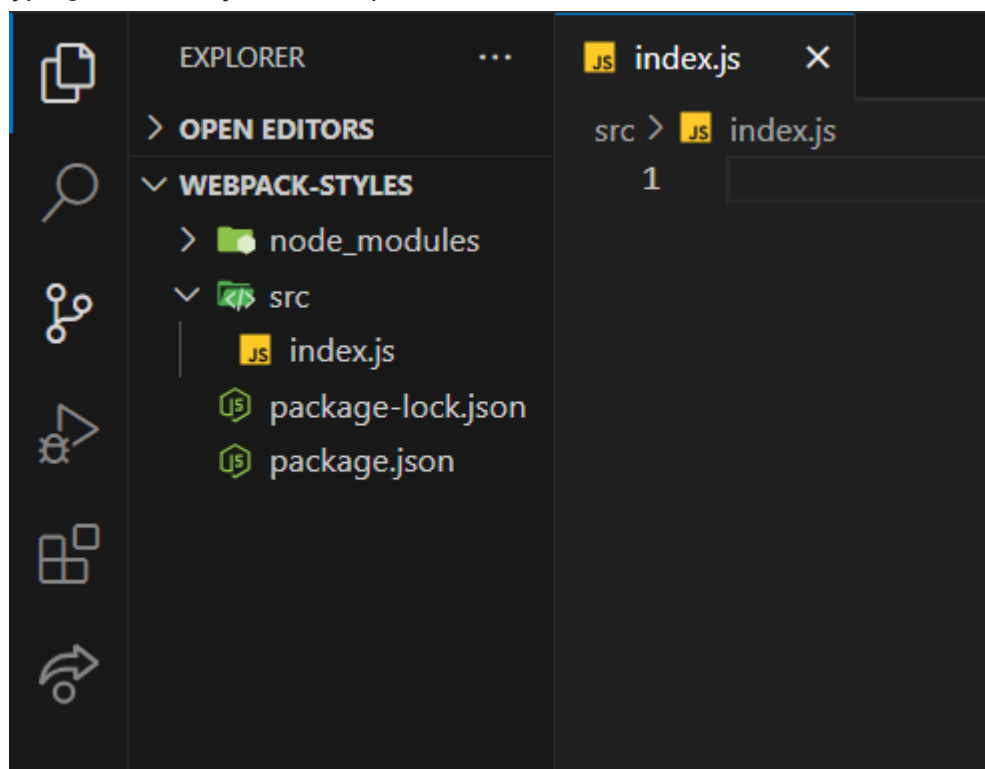
```
U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skrivebord/Hjemmeside opgave 2.0/Hovedforløb/Automatisering/Uge 3/webpack/webpack-styles
$ npm i -D webpack webpack-cli

added 117 packages, and audited 118 packages in 7s

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

found 0 vulnerabilities
```

- change your root folder, by typing “**code .**” in the terminal
- create a folder with a javascript called index.js a shortcut you can use is by typing “src/index.js” in the explorer



- Now create a variable on index.js can be const, let or var, give the variable a name such as **headline** and a string with some text for example “**Welcome to The Webpage**”.

```
index.js  main.js

src > index.js > ...
1  const headline = "Welcome to The Webpage"
```

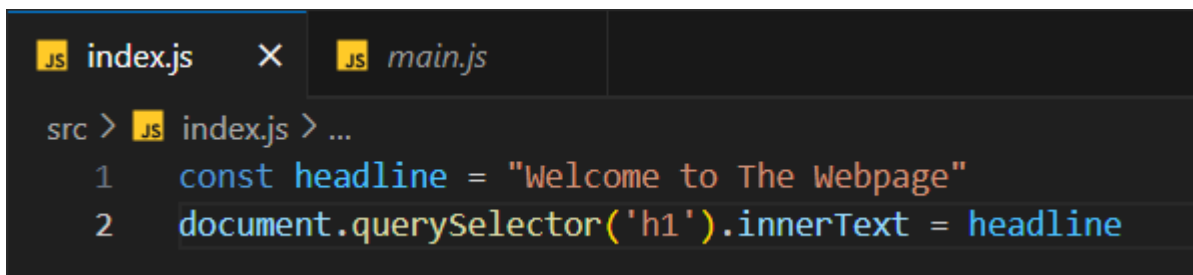
- Then type in **"npx webpack"** into the terminal to get the newest version of the webpack. It will also create a **dist** with a **main.js** inside.

```
U8796@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skrivebord/Hjemmeside opgave 2.0/Hovedforløb/Automatisering/Uge 3/webpack/webpack-styles
$ npx webpack
asset main.js 0 bytes [emitted] [minimized] (name: main)
./src/index.js 1 bytes [built] [code generated]

WARNING in configuration
The 'mode' option has not been set, webpack will fallback to 'production' for this value.
Set 'mode' option to 'development' or 'production' to enable defaults for each environment.
You can also set it to 'none' to disable any default behavior. Learn more: https://webpack.js.org/configuration/mode/

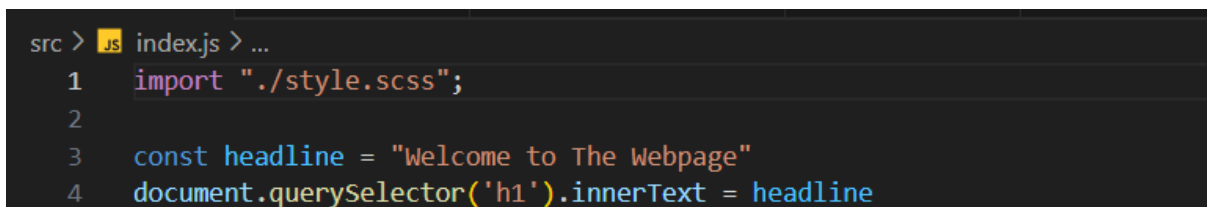
webpack 5.95.0 compiled with 1 warning in 156 ms
```

- In the **index.js** write **"document.querySelector('h1').innerText = headline"** we use the **headline** variable from before to write the text in the document



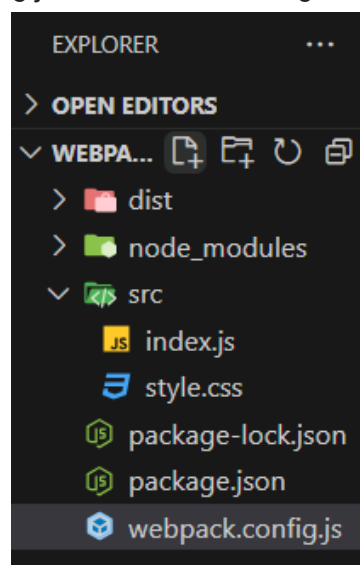
```
src > JS index.js > ...
1  const headline = "Welcome to The Webpage"
2  document.querySelector('h1').innerText = headline
```

- in the **"src"** folder Make a **"style.css"** with some styling elements inside.
- Try making styling to the **"body"** and some styling to your **"h1"**.
- After you have made some styling, import your **"style.css"** into **"index.js"** like the example below



```
src > JS index.js > ...
1  import "./style.scss";
2
3  const headline = "Welcome to The Webpage"
4  document.querySelector('h1').innerText = headline
```

- create a **webpack.config.js** which is for making rules to the modules



- and in the `webpack.config` write out as depicted below

```
webpack-styles > webpack.config.js > <unknown> > module > rules
1  module.exports = {
2    module: {
3      rules: [
4        {
5          test: /\.css$/i,
6          use: ['style-loader', 'css-loader'],
7        },
8      ],
9    },
10  };
```

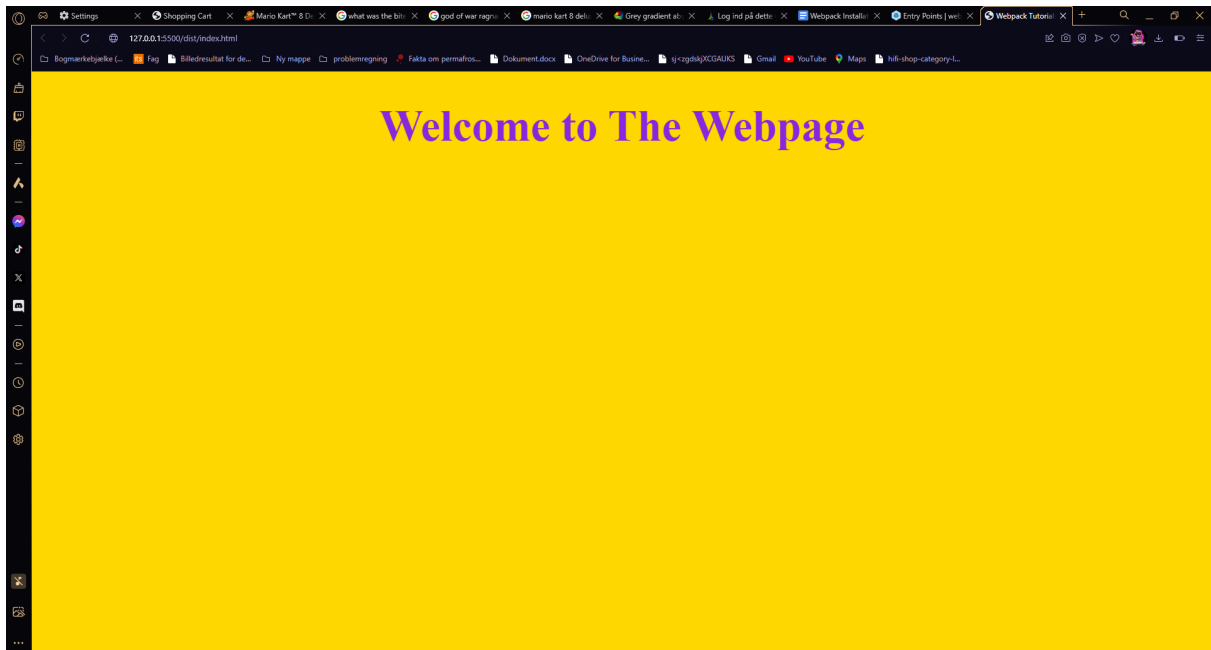
- afterwards type `npm install --save-dev style-loader css-loader` to be able to get the modules to run. The `"main.js"` is now used as a compiler for all source material

```
U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skrivebord/Hjemmeside opgave 2.0/Hovedforløb/Automatisering/Uge 3/webpack-tutorial-webpack-tutorial-andreas
-og-magnus/webpack-styles (Andreas)
$ npm install --save-dev style-loader css-loader
```

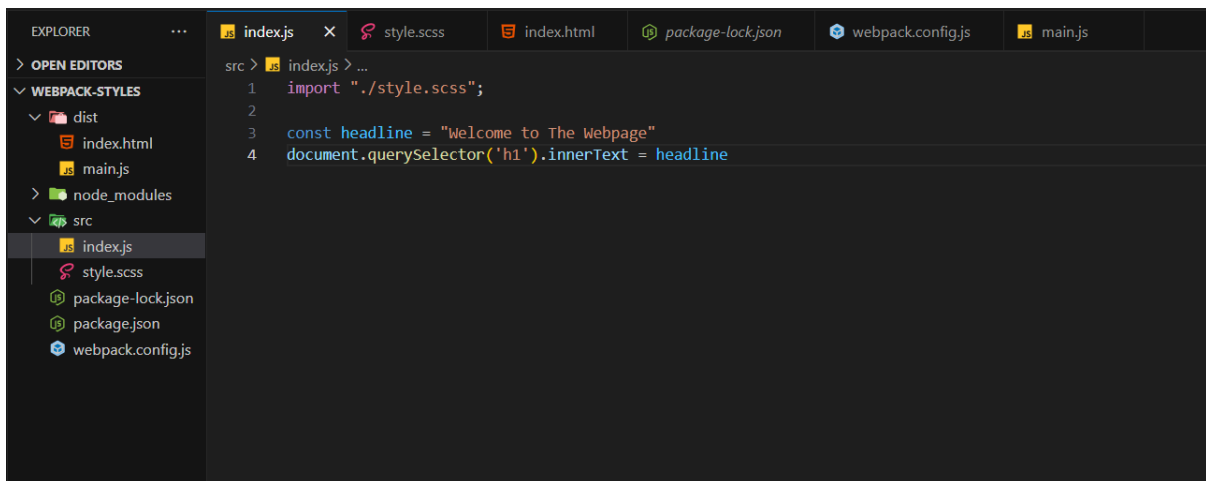
- Make an `"index.html"` in the `dist` folder where you link the javascript and make a `"div"` with a `"class"` so that we can style it after and inside the `"div"` make a `"h1"` with a bunch of randoms letter doesn't matter what you type in

```
dist > index.html > html > head > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Webpack Tutorials</title>
7    <script src="./main.js" defer></script>
8  </head>
9  <body>
10   <div class="container">
11     <h1>asædljas</h1>
12   </div>
13
14 </body>
15 </html>
```

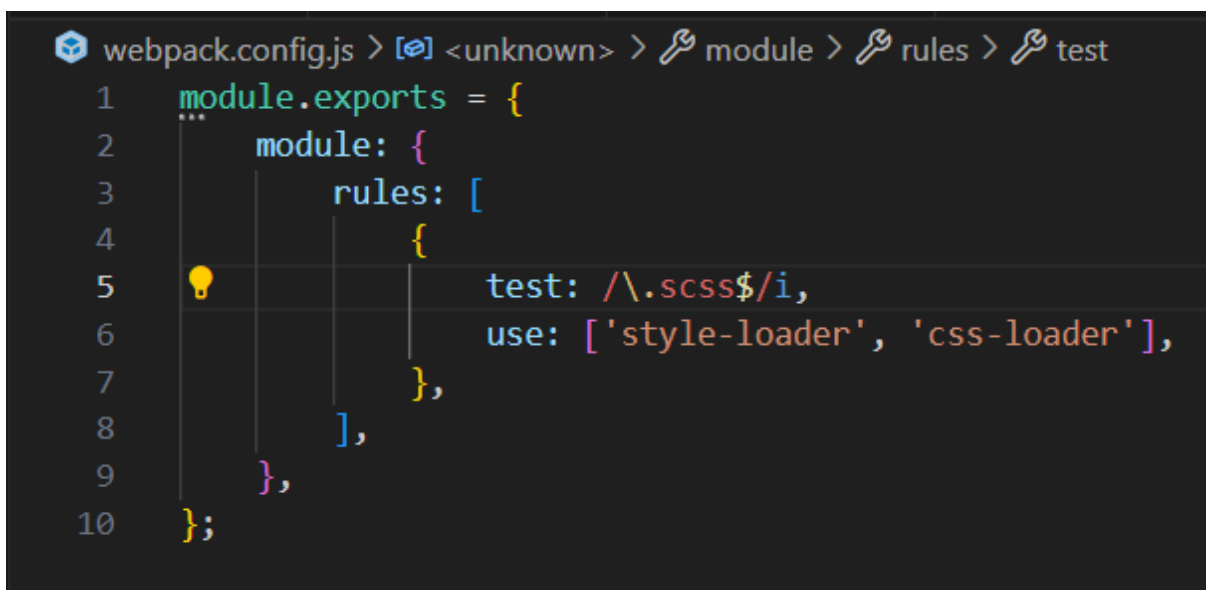
if done correctly it should replace the current text you have with the headline made earlier like so.



- *now we are going to make it compatible with multiple types of stylesheets and we start by changing all css to scss.*

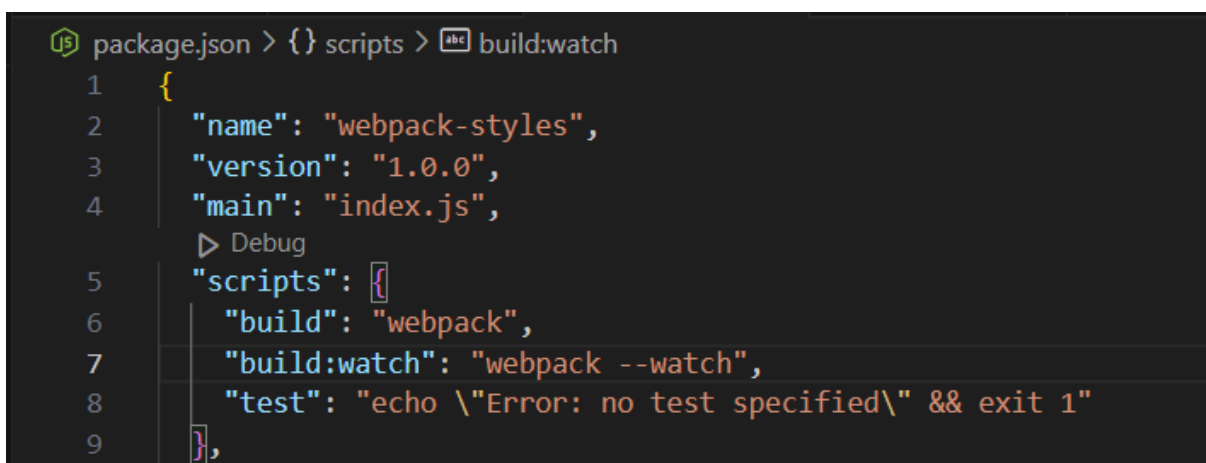


```
src > index.js > ...
1  import './style.scss';
2
3  const headline = "Welcome to The Webpage"
4  document.querySelector('h1').innerText = headline
```



```
webpack.config.js > [?] <unknown> > [?] module > [?] rules > [?] test
1  module.exports = {
2    ...
3    module: {
4      rules: [
5        {
6          test: /\.scss$/i,
7          use: ['style-loader', 'css-loader'],
8        },
9      ],
10 };
```

- go to "package.json" and type in " "build:" "webpack", "build:watch": "webpack --watch",



```
package.json > {} scripts > [?] build:watch
1  {
2    "name": "webpack-styles",
3    "version": "1.0.0",
4    "main": "index.js",
5    "scripts": {
6      "build": "webpack",
7      "build:watch": "webpack --watch",
8      "test": "echo \"Error: no test specified\" && exit 1"
9    },
10 }
```

this is so because "scss" has an option to "Watch Scss" and what we do is we make sure that the watch sass run automatically and we do that by putting it into scripts in the "package.json"

- try to adding your **"background-color"** in the scss as a **"`:root`"** to use it as a separate variable and the same with text-color just outside the **"`:root`"**

```
src > style.scss > ...
1  √ :root{
2    |    --MainBg: ■gold;
3    |  }
4
5    $light-text-color: ■blueviolet;
6
7  √ body{
8    |    background-color: var(--mainBg);
9    |    color: $light-text-color;
10   |  }
11  √ h1{
12    |    font-size: 70px;
13    |    text-align: center;
14    |  }
```

- now add a **"sass-loader"** in **"webpack.config.js"** to be able to run the **"scss"**

```
webpack.config.js > ...
1  module.exports = {
2    |   module: {
3    |     |   rules: [
4    |     |     |   {
5    |     |     |     |   test: /\.scss$/i,
6    |     |     |     |   use: ['style-loader', 'css-loader', 'sass-loader'],
7    |     |     |     | }
8    |     |     |   ],
9    |     |   },
10  |   };

```

- you also need to install the sass-loader by running **"npm i -D sass sass-loader"** in terminal

```
U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skrivebord/Hjemmeside
-og-magnus/webpack-styles (Andreas)
$ npm i -D sass sass-loader

added 17 packages, and audited 150 packages in 3s

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

- then "npm run build" to be able to build on your server

```
U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skrivebord/Hjemmeside
-og-magnus/webpack-styles (Andreas)
$ npm run build

> webpack-styles@1.0.0 build
> webpack

asset main.js 4.21 KiB [emitted] [minimized] (name: main)
```

- in the terminal run "npm run build:watch" for the "scss" to work

```
U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skrivebord/Hjemmeside opgave 2.0/Hovedforl
-og-magnus/webpack-styles (Andreas)
$ npm run build:watch

> webpack-styles@1.0.0 build:watch
> webpack --watch
```

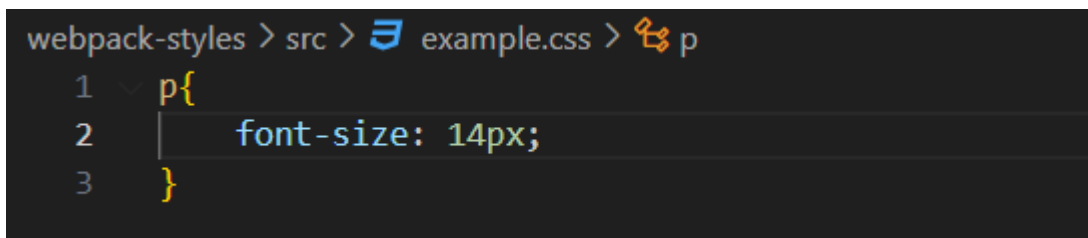
- Create a "css" and give an name and remember to to import it into "index.js"



The screenshot shows the VS Code editor with the file explorer on the left and the editor window on the right. The file explorer shows the project structure with files like index.html, main.js, and example.css. The editor window shows the content of index.js, which includes imports for style.scss and example.css, and a JavaScript snippet that sets the innerText of an h1 element to 'Welcome to The Webpage'. Red arrows point to the import statements and the example.css file in the file explorer.

```
webpack-styles > src > index.js > ...
1 import './style.scss';
2 import './example.css';
3
4 const headline = "Welcome to The Webpage"
5 document.querySelector('h1').innerText = headline
```

and put something in the css doesn't matter what it is just to test



The screenshot shows the VS Code editor with the file explorer on the left and the editor window on the right. The file explorer shows the project structure with files like index.html, main.js, and example.css. The editor window shows the content of example.css, which includes a CSS rule for the p element that sets the font-size to 14px. A red arrow points to the example.css file in the file explorer.

```
webpack-styles > src > example.css > p
1 p{
2   font-size: 14px;
3 }
```


- and to support multiple style sheet types, we start by going to “webpack.config.js” and replace the current rule “test” contents and replace it with “/\.(s[ac]|c)ss\$/i” .

```

webpack-styles > webpack.config.js > <unknown> > module > rules > test
1  module.exports = {
2    module: {
3      rules: [
4        {
5          test: /\.(s[ac]|c)ss$/i,
6          use: ['style-loader', 'css-loader', 'sass-loader'],
7        },
8      ],
9    },
10   };

```

which is so it can read both sass, scss and css.

- After that we are going make plugin and to do that we need to build the plugin by “npm i -D mini-css-extract-plugin” the name of the plugin doesn’t just make sure it makes sense for you and or the people you work with.

```

U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske
-og-magnus/webpack-styles (Andreas)
● $ npm i -D mini-css-extract-plugin

added 10 packages, and audited 160 packages in 3s

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

```

- now we're going to add the plugin into **"webpack.config.js"** and to do that we start by adding a **"const"** you can call whatever you just remember to give a good name so it's easier to re-use. After giving the **"const"** a name we need to give it a required statement

```
webpack-styles > webpack.config.js > ...
1 | const MiniCssExtractPlugin = require("mini-css-extract-plugin")
2 | module.exports = {
3 |   module: {
4 |     rules: [
5 |       {
6 |         test: /\.([a-z]|c)ss$/i,
7 |         use: ['style-loader', 'css-loader', 'sass-loader'],
8 |       },
9 |     ],
10 |   },
11 | };

```

- Add the const into a **"plugins"** in **"module.exports"** and also replace **"style-loader"** with your previous made const

```
webpack-styles > webpack.config.js > <unknown>
1 | const MiniCssExtractPlugin = require("mini-css-extract-plugin");
2 | module.exports = {
3 |   plugins: [MiniCssExtractPlugin()],
4 |
5 |   module: {
6 |     rules: [
7 |       {
8 |         test: /\.([a-z]|c)ss$/i,
9 |         use: [MiniCssExtractPlugin.loader, 'css-loader', 'sass-loader'],
10 |       },
11 |     ],
12 |   },
13 | };

```

to be able to extract the plugin we just made.

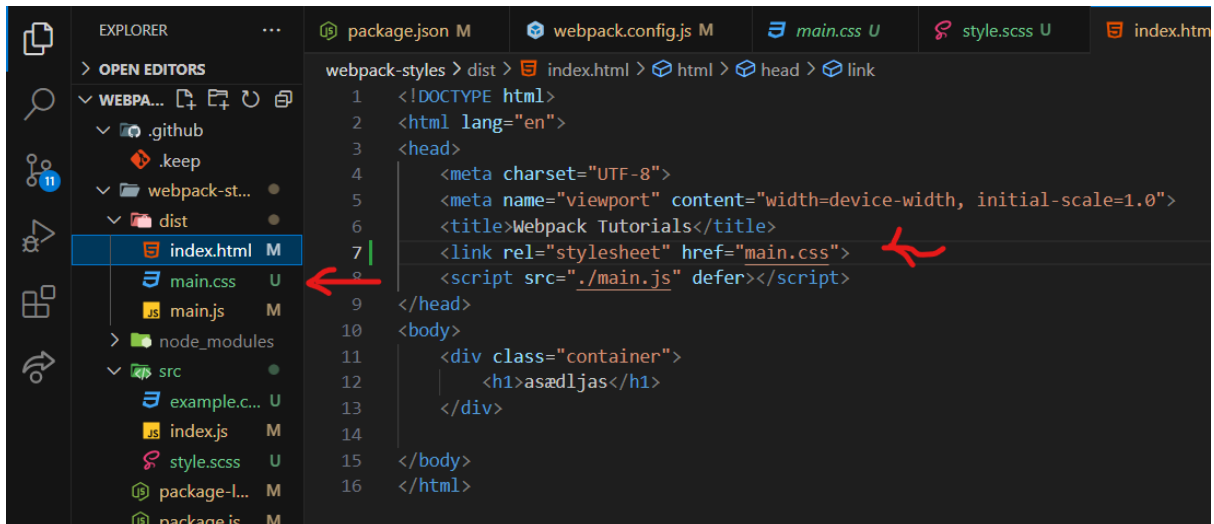
- We're now going to run build again by typing in, **"npm run build"** into the terminal

```
U87960@DXS4134 MINGW64 ~/OneDrive -
-og-magnus/webpack-styles (Andreas)
$ npm run build

> webpack-styles@1.0.0 build
> webpack

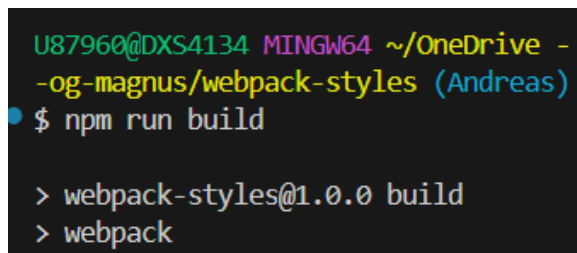
```

- now it is going to add a **"main.css"** and you have to link that to **"index.html"** remember to put defer when linking for it to load as the last thing



Sourcemap

- to make source-map we start by going to **"webpack.config.js"**, and there we are going to make a developer tool by typing **"devtool"** outside of module, and in **"devtool"** we make a string called **"source-map"** and after that run the **"npm run build"** in the terminal once again. Which is going to generate a sourcemap



PostCSS

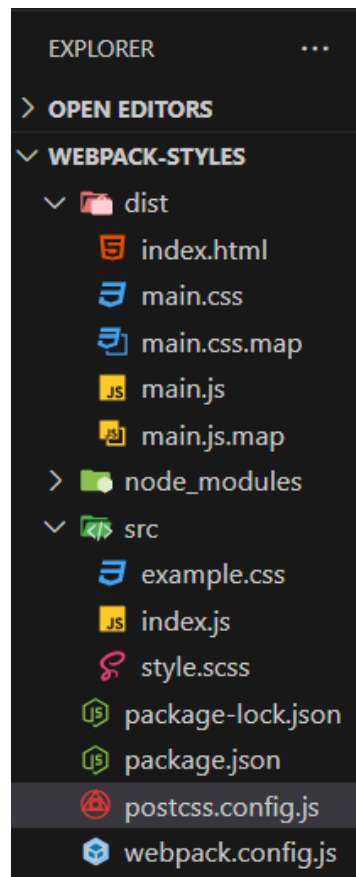
- first we are going to install postcss via a terminal and we do that by typing in
"npm i -D postcss postcss-loader postcss-preset-env"

```
U87960@DXS4134 MINGW64 ~/OneDrive - Roskilde Tekniske Skole/Skri  
-og-magnus/webpack-styles (Andreas)  
$ npm i -D postcss postcss-loader postcss-preset-env
```

- and now add the postcss-loader into "webpack.config.js" so that it can be found when running program again

```
webpack.config.js > [?] <unknown> > module > rules > use  
2 module.exports = {  
3   plugins: [ new MiniCssExtractPlugin()],  
4  
5   module: {  
6     rules: [  
7       {  
8         test: /\.([a-z])css$/i,  
9         use: [MiniCssExtractPlugin.loader, 'css-loader', 'sass-loader', 'postcss-loader'],  
10      },  
11     ],  
12   },  
13  
14   devtool: "source-map",  
15 };
```

- create a new file called "postcss.config.js" to configure the postcss



- in postcss add **"module.exports"** statement that has a **"plugin"** with **"require"** inside of the require there gonna be the **"postcss-preset-env"**

```
postcss.config.js > [?] <unknown>
1  module.exports = {
2    plugins: [require("postcss-preset-env")]
3  }
```

- and we can test postcss by adding some extra styles to **"style.scss"** we can start by putting our **"h1"** into the **"container"** class made when making **"index"** and also add some other style for example **"padding"**, **"display: flex"** and **"linear gradient"**.

```

6
7  body{
8    background-color: var(--MainBg);
9    color: $light-text-color;
10
11  }
12  .container{
13    padding: 2rem;
14    display: flex;
15    background-color: linear-gradient(to top, orange, firebrick );
16
17    h1{
18      font-size: 70px;
19      text-align: center;
20    }
21  }
22 }
23
```

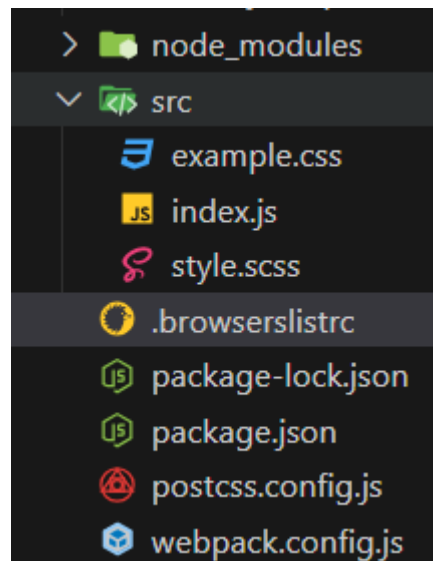
- we are now going to add **"mode: "development" "** to **"webpack.config.js"**

```
webpack.config.js > [?] <unknown> > mode
1  const MiniCssExtractPlugin = require("mini-css-extract-plugin");
2  module.exports = {
3    mode: "development",
4    plugins: [ new MiniCssExtractPlugin()],
5
6    module: {
7      rules: [
8        {
9          test: /\.s[ac]c?$/,
10         use: [MiniCssExtractPlugin.loader, 'css-loader', 'sass-loader', 'postcss-loader'],
11       },
12     ],
13   },
14
15   devtool: "source-map",
16 };

```

and then run build

- afterwards add a new file called **".browserslistrc"** to add browser fallback,



- inside the file we add **"last 2 version > 0.5% IE 10"**

```
.browserslistrc
1  last 2 version
2  > 0.5%
3  IE 10
```

and then run build once more