

WEBPACK

Webpack goes through the package and creates what it calls a dependency graph which consists of various modules which our web app would require to function as expected. Then, depending on this graph, it creates a new package which consists of the very bare minimum number of files required, often just a single bundle.js file which can be plugged in to the html file easily and used for the application.

Steps for creating webpack:

Initiating the project:

This will create a starter package and add a package. Json file. This is where all the dependencies required to build this application will be mentioned.

```
C:\Users\ssree\OneDrive\Desktop\webpack>npm init -y
Wrote to C:\Users\ssree\OneDrive\Desktop\webpack\package.json:

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

Creating index.html:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
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>Document</title>
8   </head>
9   <body></body>
10 </html>
```

Creating script.js:

```
1 function component() {
2     const element = document.createElement('div');
3
4     // Lodash, currently included via a script, is required for this line to work
5     element.innerHTML = _.join(['Hello', 'webpack'], ' ');
6
7     return element;
8 }
9
10 document.body.appendChild(component());
```

INSTALLING LODASH:

Lodash makes JavaScript easier by taking the hassle out of working with arrays, numbers, objects, strings, etc. Lodash's modular methods are great for:

- Iterating arrays, objects, & strings
- Manipulating & testing values
- Creating composite functions

```
C:\Users\ssree\OneDrive\Desktop\webpack>npm install lodash
added 1 package, and audited 2 packages in 13s
found 0 vulnerabilities
```

Updating index.html

```
<> index.html > ...
2 <html lang="en">
3   <head>
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 Example</title>
8     <script src="node_modules/lodash/lodash.min.js"></script>
9     <script src="src/script.js" defer></script>
10  </head>
11  <body></body>
12 </html>
13
```

Output:

```
Hello webpack
```

Installing Webpack:

Assuming we have initialized a new project with npm, there are two packages we need to install to use webpack, webpack and webpack-cli.

```
C:\Users\ssree\OneDrive\Desktop\webpack>npm install webpack webpack-cli --save-dev
added 119 packages, and audited 121 packages in 31s

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

found 0 vulnerabilities
```

The whole point of webpack is to "examine all of the modules, (optionally) transform them, then intelligently put all of them together into one or more bundle(s)". To do that, webpack needs to know three things.

1. The entry point of the application
2. Which transformations, if any, to make on your code
3. The location to put the newly formed bundle(s)

Webpack.config.js:



```
1 const path = require("path");
2
3 module.exports = {
4   mode: "none",
5   entry: "./src/script.js",
6   output: {
7     filename: "main.js",
8     path: path.resolve(__dirname, "dist"),
9   },
10 };
```

1. webpack grabs the entry point located at ./src./script.js
2. It examines all of our import and require statements and creates a dependency graph.
3. webpack starts creating a bundle, whenever it comes across a path we have a loader for, it transforms the code according to that loader then adds it to the bundle.
4. It takes the final bundle and outputs it at dist directory.

Run Webpack:

```
C:\Users\ssree\OneDrive\Desktop\webpack>npx webpack
asset main.js 382 bytes [emitted] (name: main)
./src/script.js 298 bytes [built] [code generated]
webpack 5.89.0 compiled successfully in 271 ms
```

Webpack Plugins:

Unlike loaders, plugins allow to execute certain tasks after the bundle has been created. Because of this, these tasks can be on the bundle itself, or just to the codebase.

Next, we add a **plugins** property which is an array to our webpack config.

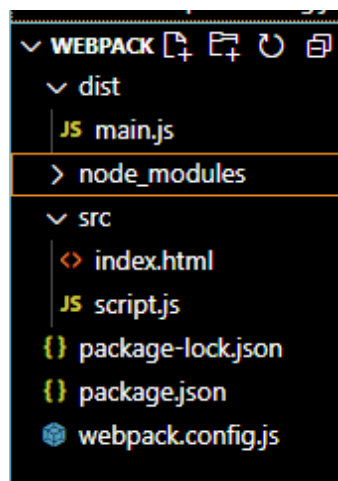
Html Webpack Plugin:

```
C:\Users\ssree\OneDrive\Desktop\webpack>npm install html-webpack-plugin --save-dev
added 30 packages, and audited 151 packages in 15s

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

found 0 vulnerabilities
```

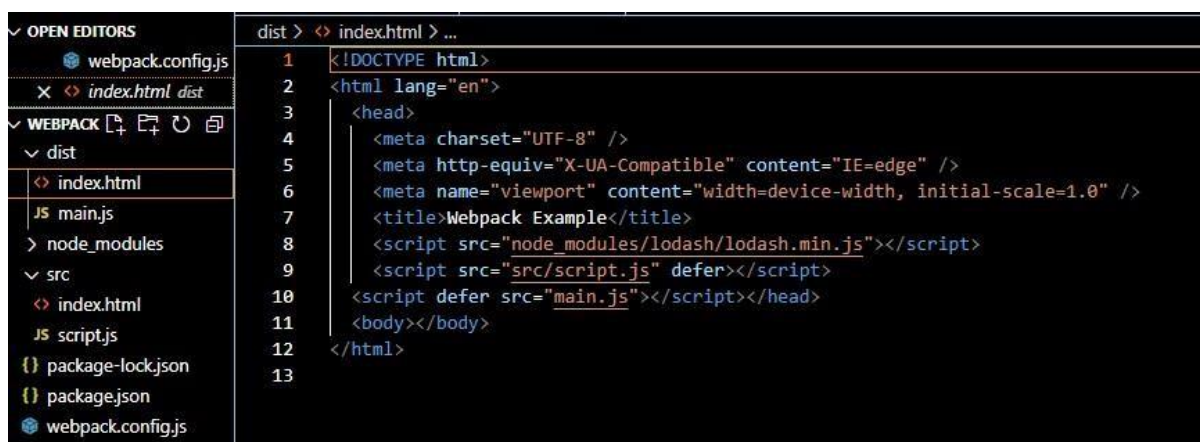
View of the project directories:



Running Webpack after adding plugin:

```
C:\Users\ssree\OneDrive\Desktop\webpack>npx webpack
asset index.html 436 bytes [emitted]
asset main.js 382 bytes [compared for emit] (name: main)
./src/script.js 298 bytes [built] [code generated]
webpack 5.89.0 compiled successfully in 558 ms
```

Output Index.html:

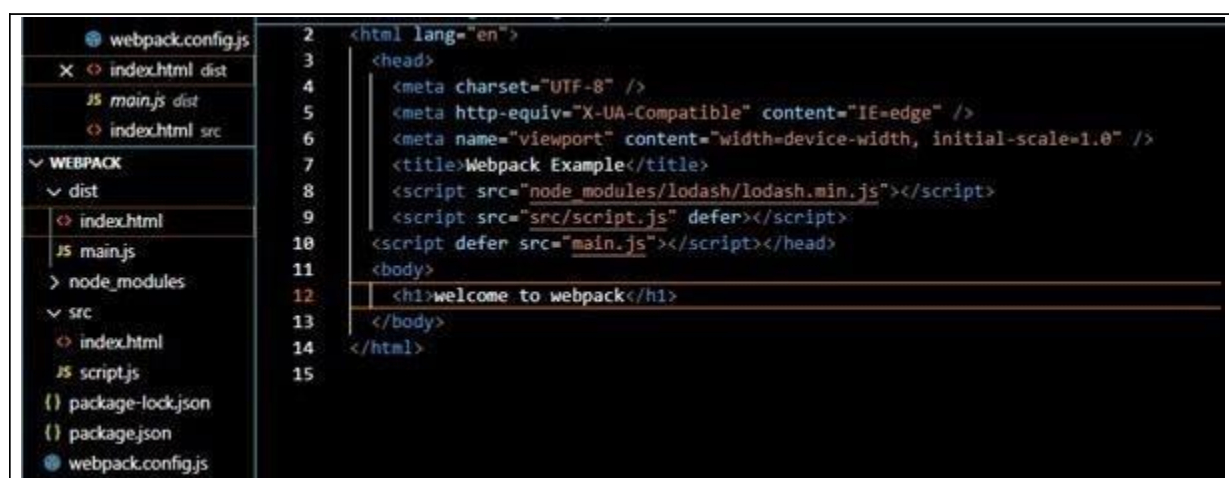


The screenshot shows the VS Code interface with the 'dist' directory expanded in the Explorer sidebar. The 'index.html' file is selected, and its content is displayed in the editor. The code is a standard HTML document with a head section containing meta tags for charset, compatibility, and viewport, a title 'Webpack Example', and two script tags. One script tag is for 'node_modules/lodash/lodash.min.js' and the other is for 'src/script.js' with the 'defer' attribute. The body section is currently empty.

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
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 Example</title>
8     <script src="node_modules/lodash/lodash.min.js"></script>
9     <script src="src/script.js" defer></script>
10    <script defer src="main.js"></script></head>
11    <body></body>
12  </html>
```

Any changes made in source directory (index.html) the output will be in the dist directory.

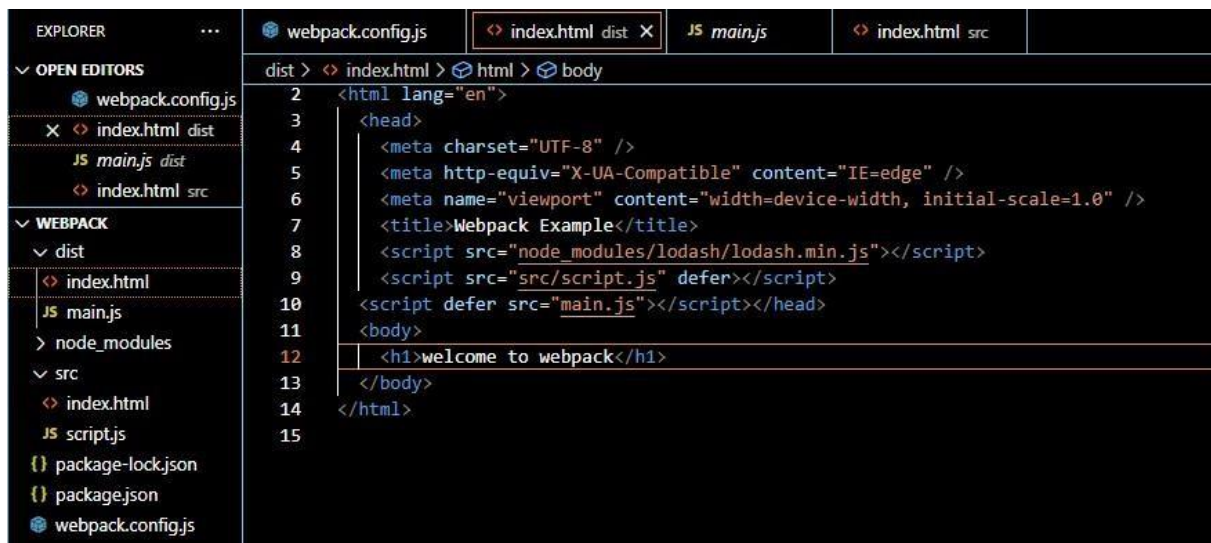
Src/index.html:



The screenshot shows the VS Code interface with the 'src' directory expanded in the Explorer sidebar. The 'index.html' file is selected, and its content is displayed in the editor. The code is a standard HTML document with a head section containing meta tags for charset, compatibility, and viewport, a title 'Webpack Example', and two script tags. One script tag is for 'node_modules/lodash/lodash.min.js' and the other is for 'src/script.js' with the 'defer' attribute. The body section contains a single heading element: 'welcome to webpack'.

```
2 <html lang="en">
3   <head>
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 Example</title>
8     <script src="node_modules/lodash/lodash.min.js"></script>
9     <script src="src/script.js" defer></script>
10    <script defer src="main.js"></script></head>
11    <body>
12      <h1>welcome to webpack</h1>
13    </body>
14  </html>
```

Dist/index.html:



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the file structure with 'dist' expanded, showing 'index.html'. The Open Editors sidebar shows 'index.html dist' as the active file. The main editor area displays the content of 'dist/index.html' with the following HTML code:

```
1  <html lang="en">
2
3  <head>
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 Example</title>
8    <script src="node_modules/lodash/lodash.min.js"></script>
9    <script src="src/script.js" defer></script>
10   <script defer src="main.js"></script></head>
11   <body>
12     <h1>welcome to webpack</h1>
13   </body>
14 </html>
15
```