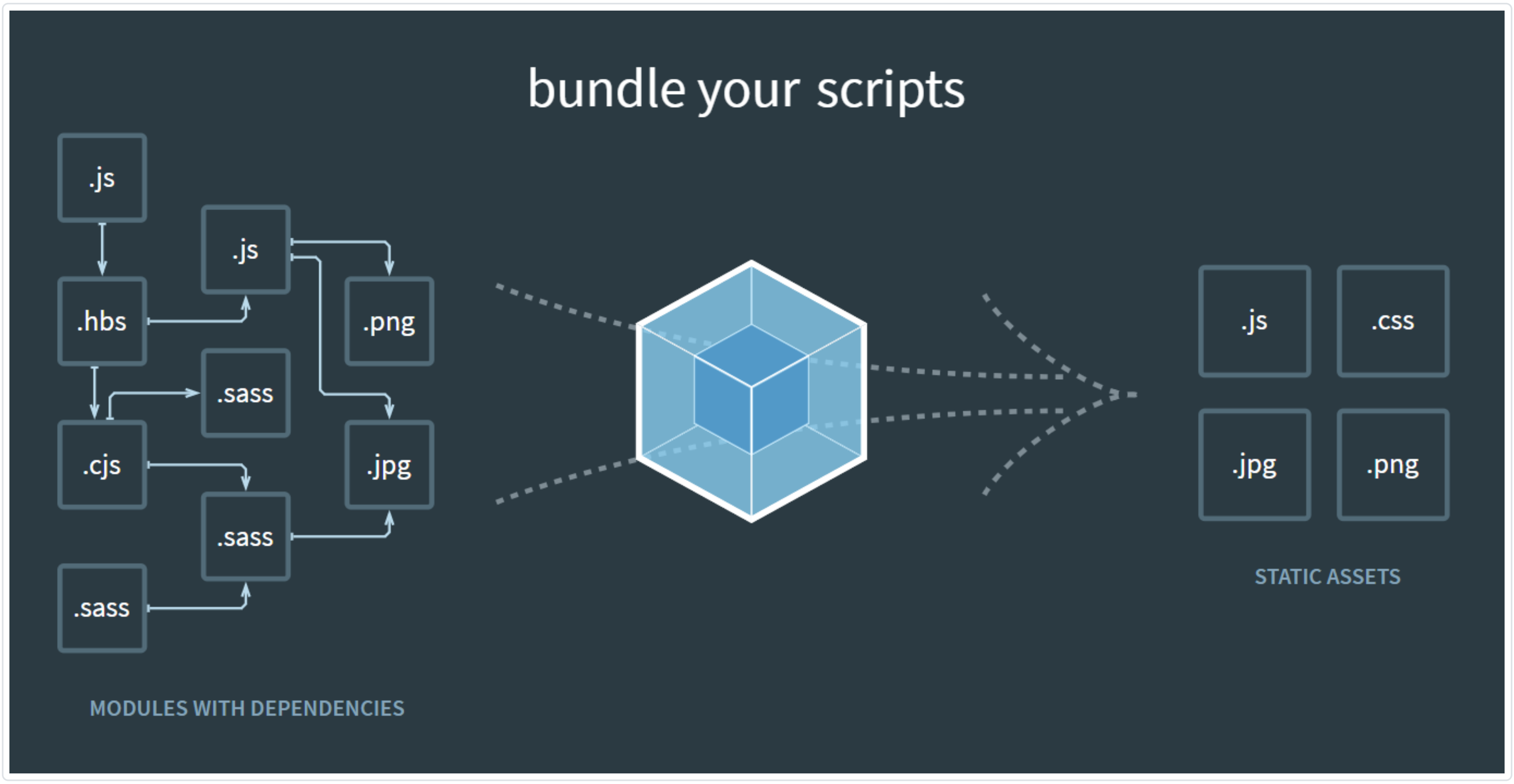
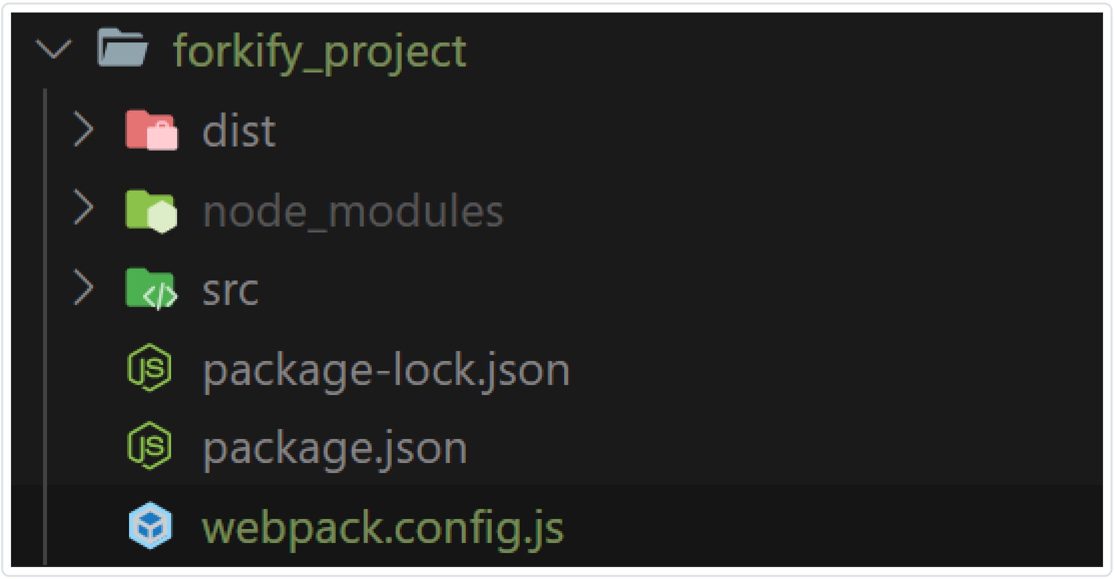
Modern JS: Webpack Basics

Open webpack.html for the markup/code | Images are taken from: JS Course by Jonas Schmedtmann

Webpack is the most commonly used asset bundler. Webpack doesn't only bundle JS files into bundles, but it bundles all kinds of assets like JS files, CSS, Images, etc, into a single file as shown below.

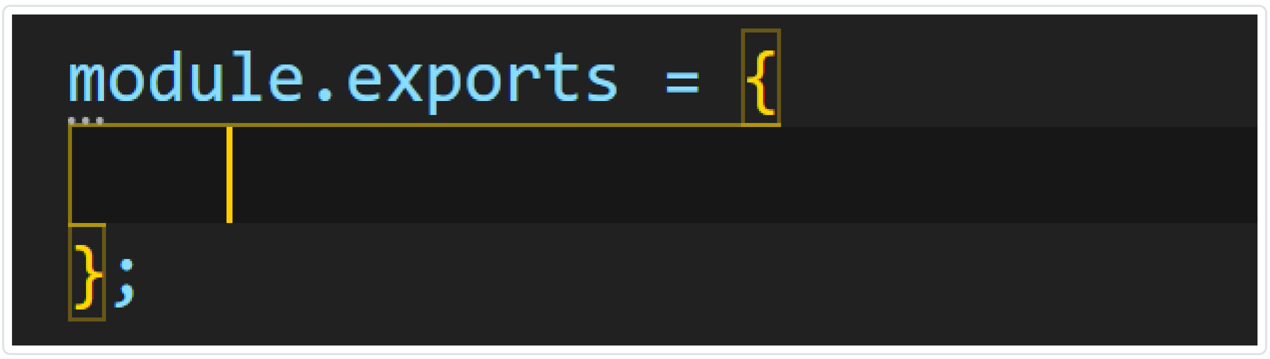


We will use Webpack v4.0 and above in which something known as **zero configuration**, where we don't even nee to write a configuration file. If we want to use zero configuration, we just need to have a **source** folder (aka `src`) in the **root** of the project and in there, a single `index.js` file. The webpack will automatically create a **distribution** folder (aka `dist`) and put the bundled file in there. But that is just for really small projects/apps. For bigger apps, we definitely have to write the **configuration** file ourselves. Now, in order to do that, we have to add a configuration file in our app/project root named as `webpack.config.js` as shown below.



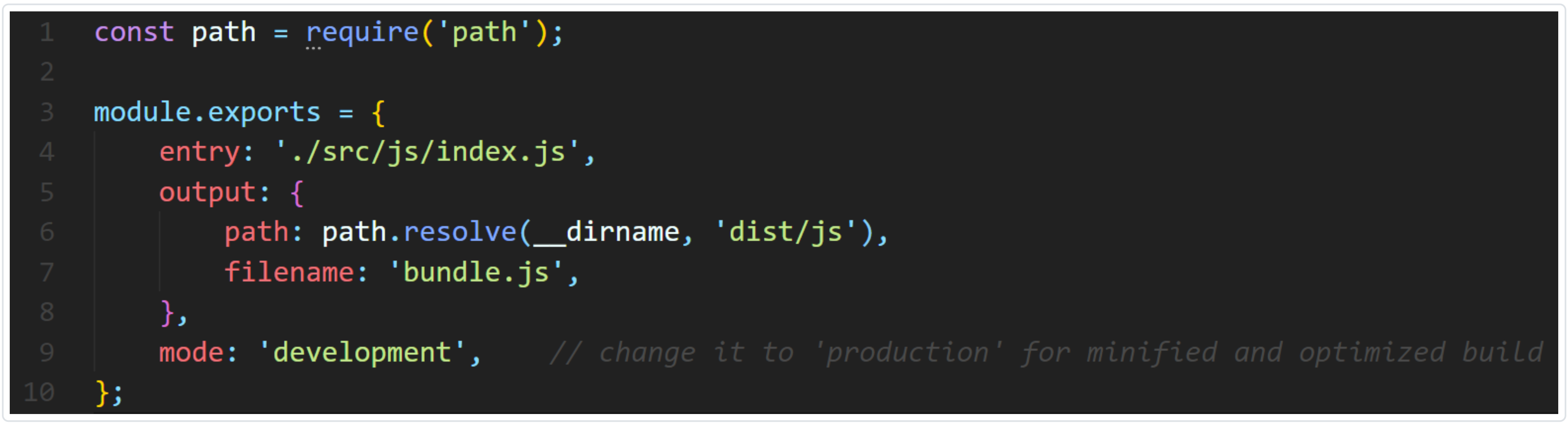
```
forkify_project
  dist
  node_modules
  src
  package-lock.json
  package.json
  webpack.config.js
```

In the `webpack.config.js` file, we have only one object in which we can specify our settings or configuration. We basically want to export this configuration object Node.js syntax. For that, we assign the object to `module.exports`. Now, `module.exports` is simply there to export the configuration object, so that webpack can take this object and do its magic.



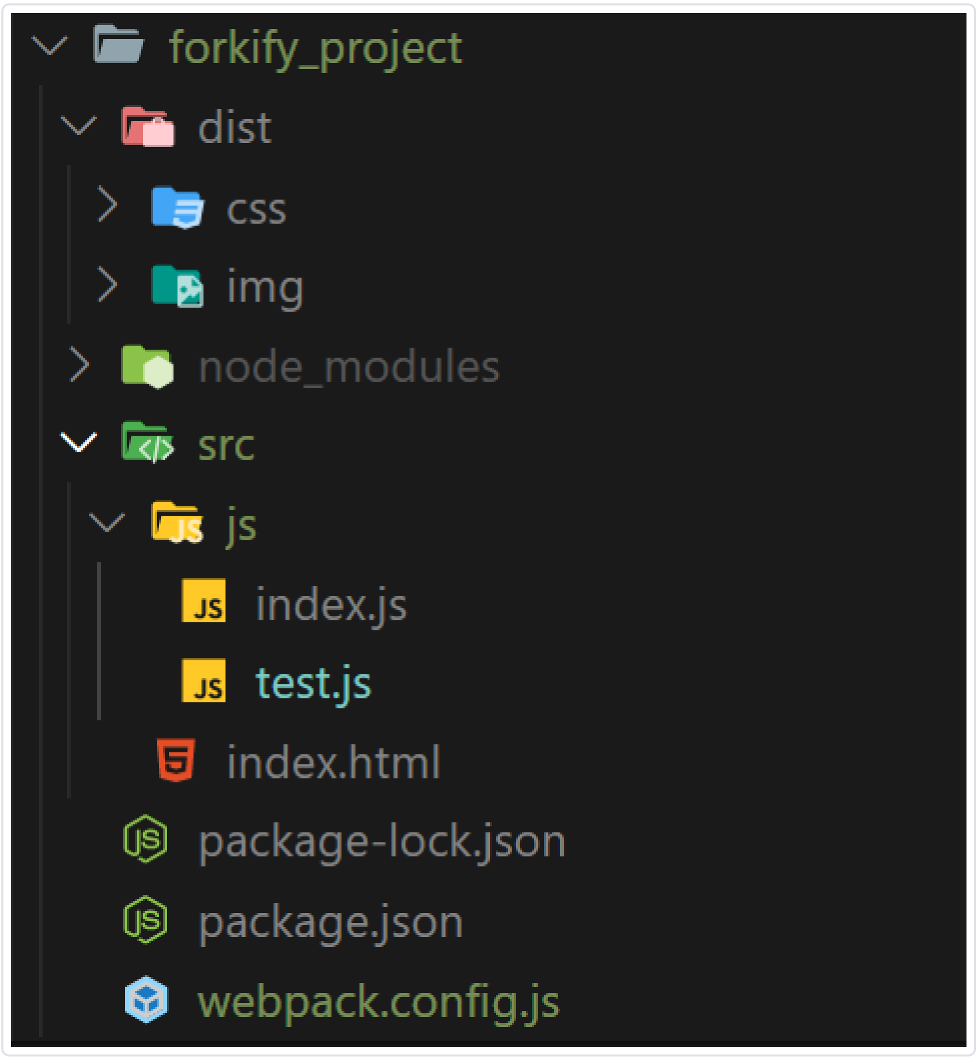
```
module.exports = {

};
```

To create a configuration, we need to know the 4 core concepts of webpack - The **entry point**, **output**, **loaders** and **plugins**. We will do a very simple configuration here to show how we usually configure. Starting with the entry point, it is a property in the `modules.export` object where, the webpack starts the bundling of files in our app/project. Basically, `entry` property needs to be mentioned with the file where all the dependencies can be looked upon, so that webpack can bundle it together. In the `entry` property, we can specify one or more entry files, but for now, we will keep it simple, and just give one entry file which is `index.js` as shown below. Next, we need to define the `output` property in the `modules.export` which will tell webpack, where to save the final bundled file. `output` property, is an object where we put a path in the `path` property, which is an absolute path to the directory where we want our final bundled file to be in. And finally, we give the name of the file (conventionally named as `bundle.js`) in the `filename` property. Now, to get the absolute path, we use a built-in node package, which we have to include in the `webpack.config.js` as `const path = require('path');`. Now, in the `path` property inside the `output` object, we will use the package we imported as `path.resolve(__dirname, <param2>)`, where `__dirname` is the **absolute path** and <param2> is the path to inside the absolute path as shown below. Now, we can go ahead and continue now, but to make things a little better, because in webpack v4 and above, we now have something called the **production** and the **development** mode. The development mode simply builds our bundle without minifying our code in order for our bundling to be as fast as possible. Whereas the production mode will automatically enable all kinds of optimization(s), like minifying and tree shaking in order to reduce the final bundle size. Our very first configuration file is as below (we will look into loaders and plugins later).

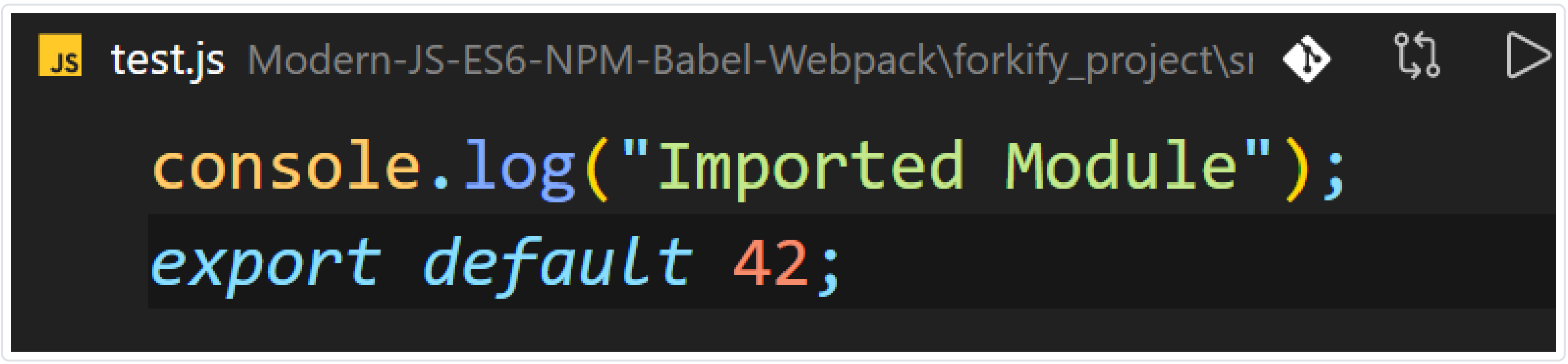


```
const path = require('path');

module.exports = {
    entry: './src/js/index.js',
    output: {
        path: path.resolve(__dirname, 'dist/js'),
        filename: 'bundle.js',
    },
    mode: 'development',    // change it to 'production' for minified and optimized build
};
```

Now, to test how webpack bundles our files, we need to create a a new JavaScript file, say `test.js`. Now, we know that in the webpack configuration file, we have mentioned the entry point to be `index.js` and therefore, if we want to include a new module inside the entry point in order to test if the other module gets included and if they're bundled together or not. So just for testing purposes, let's add a new file in the **src** folder named `test.js` as shown below.

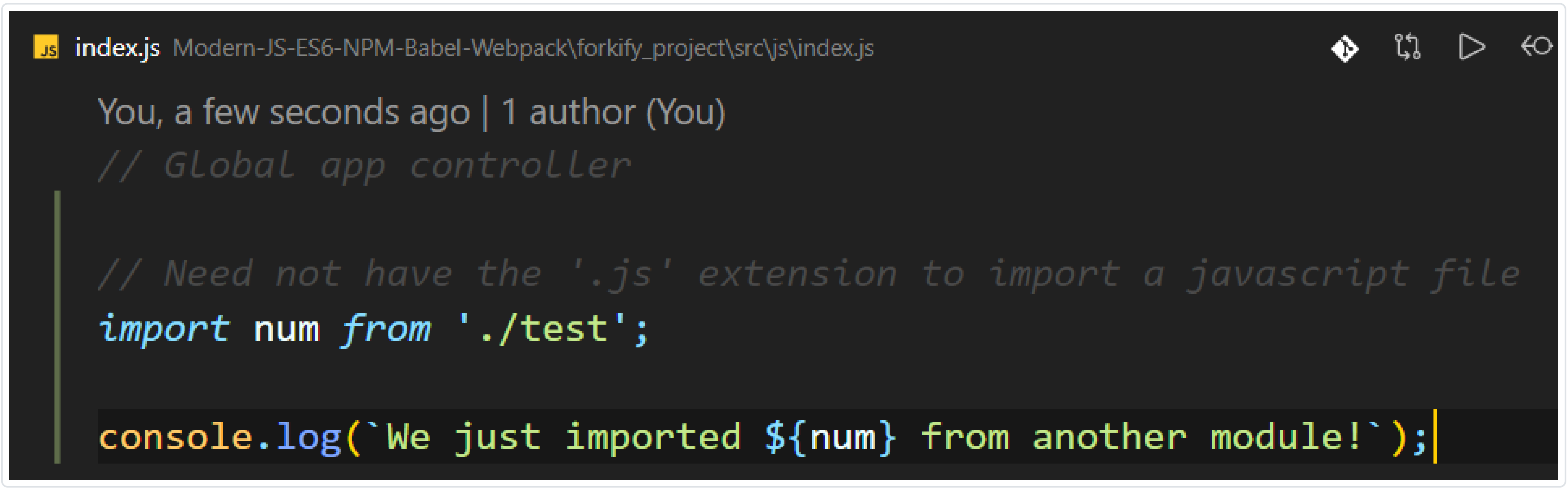

```
forkify_project
  dist
    css
    img
  node_modules
  src
    js
      index.js
      test.js
    index.html
  package-lock.json
  package.json
  webpack.config.js
```

And in `test.js` we log something to the console, just for testing purposes. And then, we can also export something from `test.js`. To export something, we use the `export` keyword, followed by a `default` export or a `named` export. And so, we can write the code as shown below (we will know more about `default` and `named` exports later).


test.js Modern-JS-ES6-NPM-Babel-Webpack\forkify_project\s

```javascript
console.log("Imported Module");
export default 42;
```

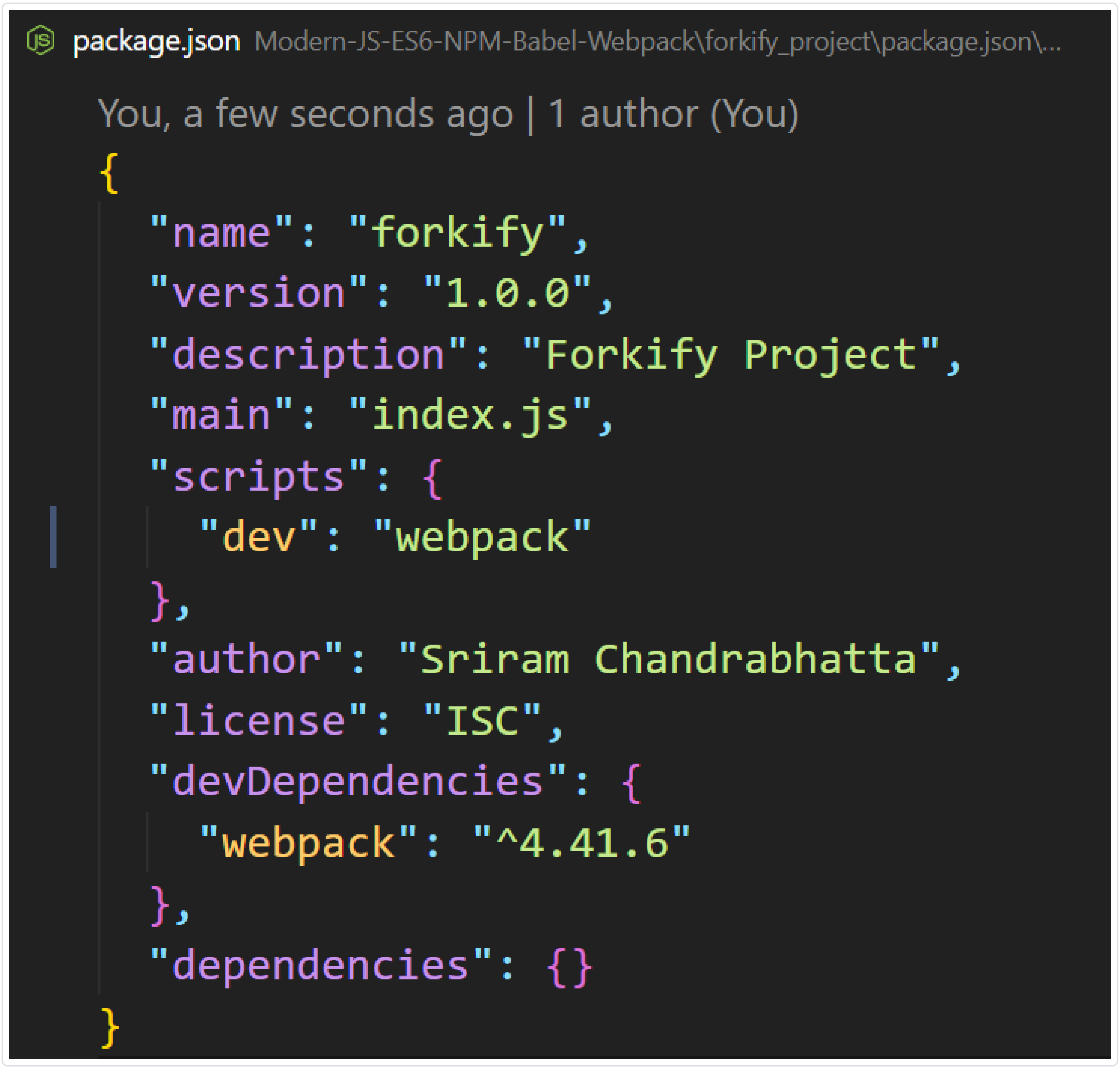
Now, in the entry point (which is the `index.js` file), we can go ahead and import the value that we exported in `test.js` by also naming it. The way we import the value, is given below.


index.js Modern-JS-ES6-NPM-Babel-Webpack\forkify_project\src\js\index.js

```javascript
You, a few seconds ago | 1 author (You)
// Global app controller

// Need not have the '.js' extension to import a javascript file
import num from './test';

console.log(`We just imported ${num} from another module!`);
```

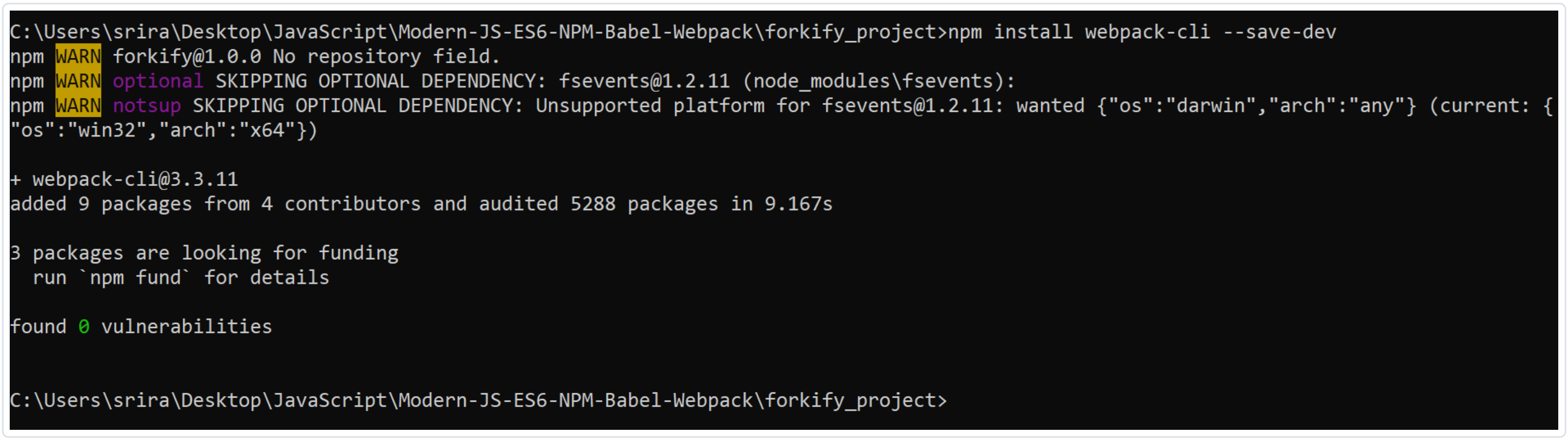
Now, the pieces of code that we wrote in `test.js` and `index.js` where we exported and imported values respectively, won't run in the browser if we do not use webpack or some kind of a bundler and that is exactly why we are using webpack.

We now have to update our `package.json` and add an **npm script**. We use npm scripts because that's the only well known and efficient way to launch our locally installed dev dependencies. We can see the `package.json` file below.



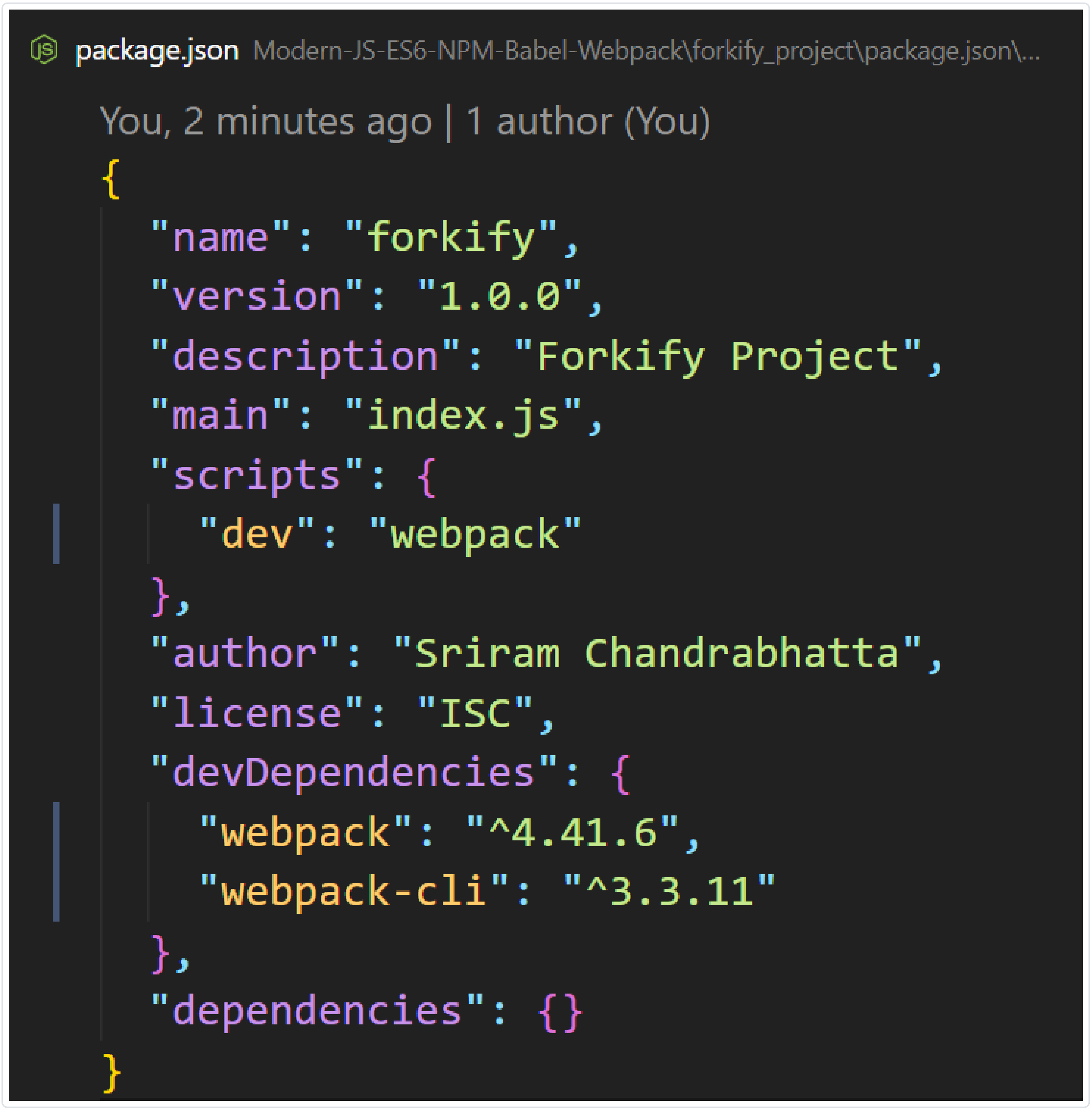
```
package.json Modern-JS-ES6-NPM-Babel-Webpack\forkify_project\package.json\...

You, a few seconds ago | 1 author (You)
{
  "name": "forkify",
  "version": "1.0.0",
  "description": "Forkify Project",
  "main": "index.js",
  "scripts": {
    "dev": "webpack"
  },
  "author": "Sriram Chandrabhatta",
  "license": "ISC",
  "devDependencies": {
    "webpack": "^4.41.6"
  },
  "dependencies": {}
}
```

Now, in order for us to bundle everything in our project into `bundle.js`, we need to install one more thing, which is Webpack Command Line Interface. We install Webpack CLI using NPM (as our development dependencies) in the terminal using the command shown below.



```
C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project>npm install webpack-cli --save-dev
npm WARN forkify@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.11 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.11: wanted {"os":"darwin","arch":"any"} (current: {
"os":"win32","arch":"x64"})

+ webpack-cli@3.3.11
added 9 packages from 4 contributors and audited 5288 packages in 9.167s

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

found 0 vulnerabilities


C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project>
```

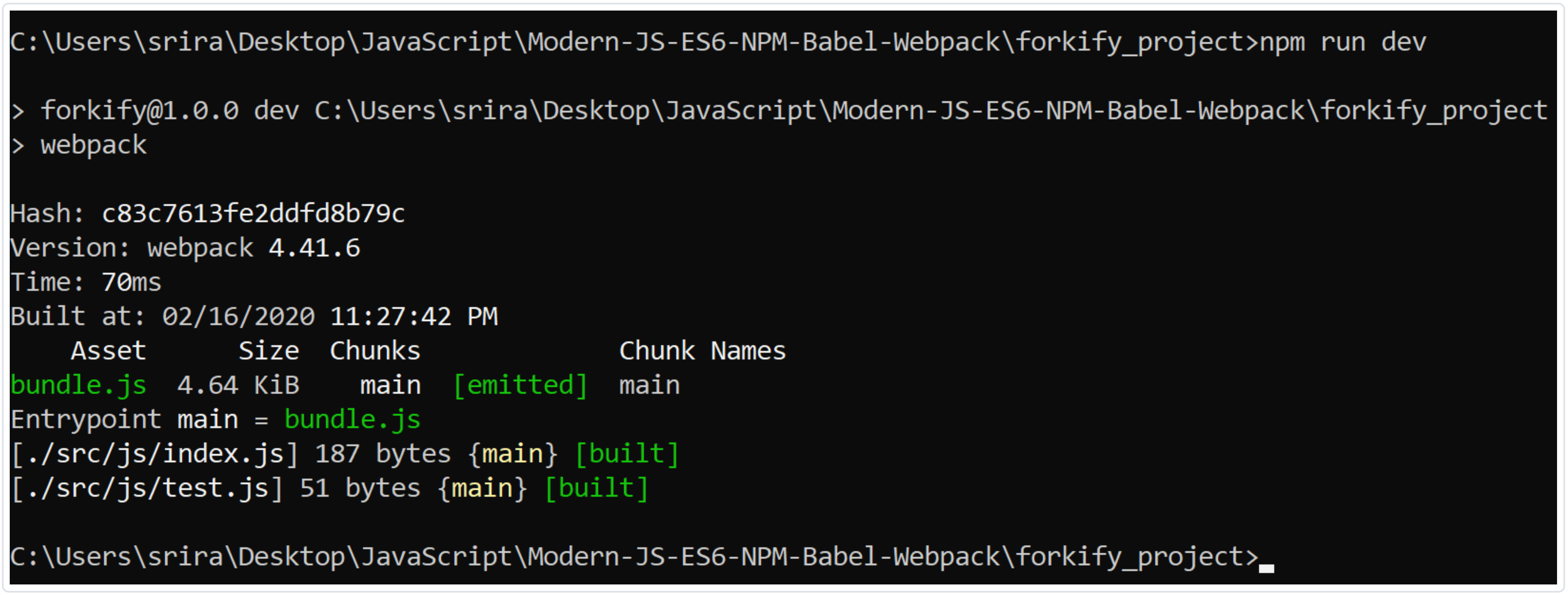
We can look into our `package.json` file for the reflected changes, specifically, we need to look into `"devDependencies"` field where we can see that `"webpack-cli"` is added along with its version, as shown below.


package.json Modern-JS-ES6-NPM-Babel-Webpack\forkify_project\package.json\...

You, 2 minutes ago | 1 author (You)

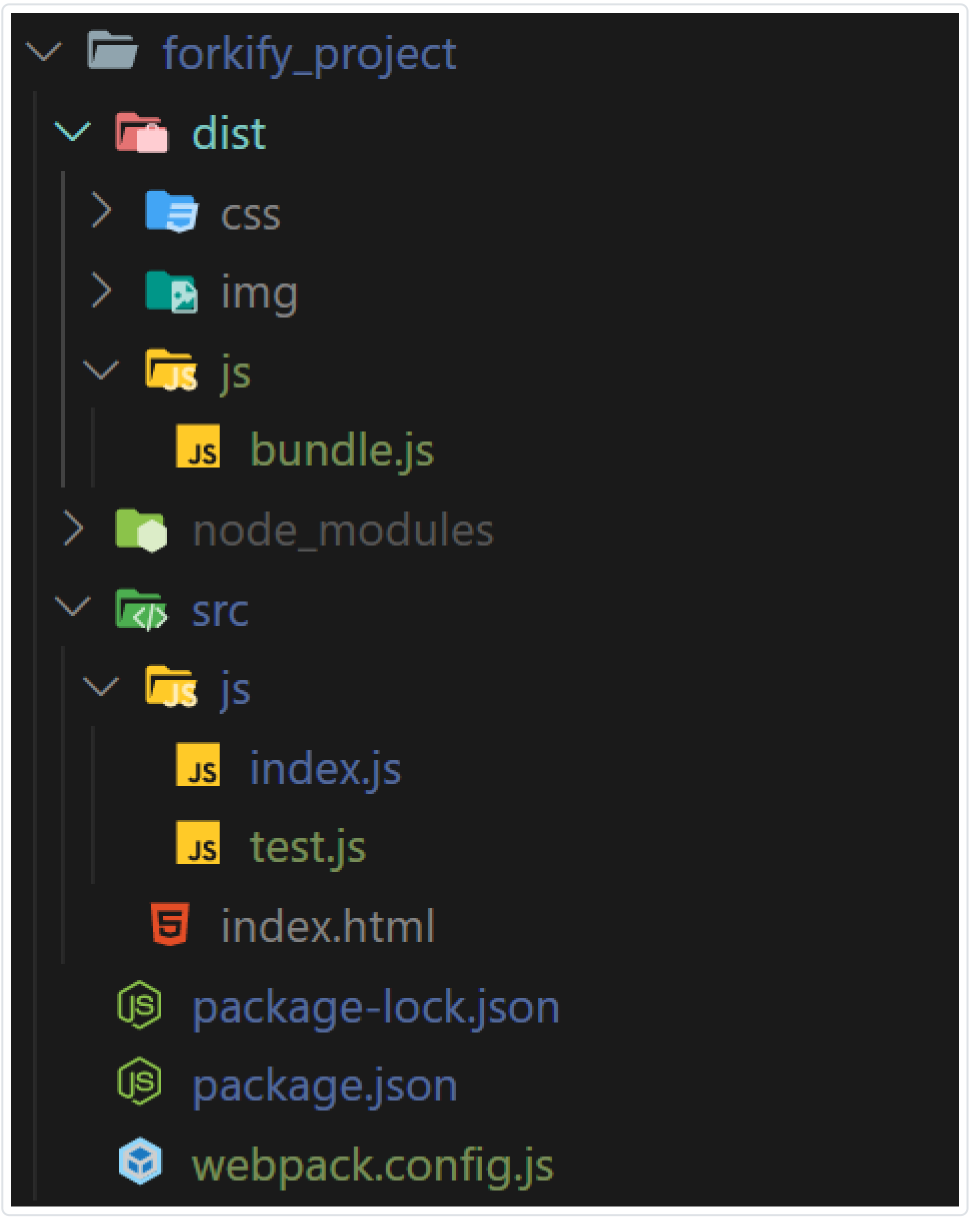
```json
{
  "name": "forkify",
  "version": "1.0.0",
  "description": "Forkify Project",
  "main": "index.js",
  "scripts": {
    "dev": "webpack"
  },
  "author": "Sriram Chandrabhatta",
  "license": "ISC",
  "devDependencies": {
    "webpack": "^4.41.6",
    "webpack-cli": "^3.3.11"
  },
  "dependencies": {}
}
```

Now, to bundle the modules together, we simply go to the terminal and type in `npm run <script_name>` where our <script_name> in this case is "dev", as shown below.

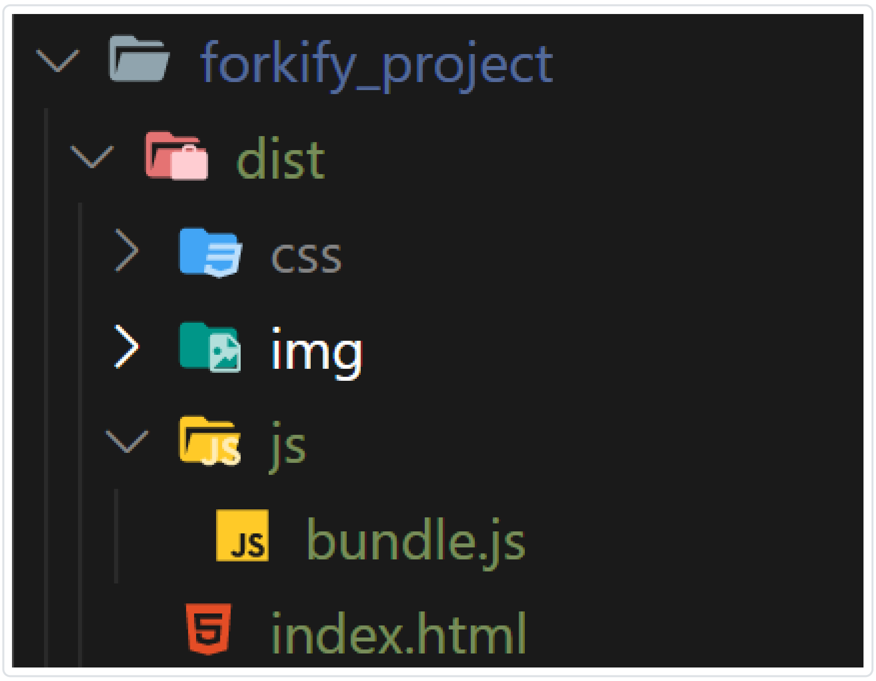


```
C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project>npm run dev

> forkify@1.0.0 dev C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project
> webpack

Hash: c83c7613fe2ddfd8b79c
Version: webpack 4.41.6
Time: 70ms
Built at: 02/16/2020 11:27:42 PM
    Asset      Size  Chunks             Chunk Names
bundle.js  4.64 KiB    main  [emitted]  main
Entrypoint main = bundle.js
[./src/js/index.js] 187 bytes {main} [built]
[./src/js/test.js] 51 bytes {main} [built]

C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project>
```

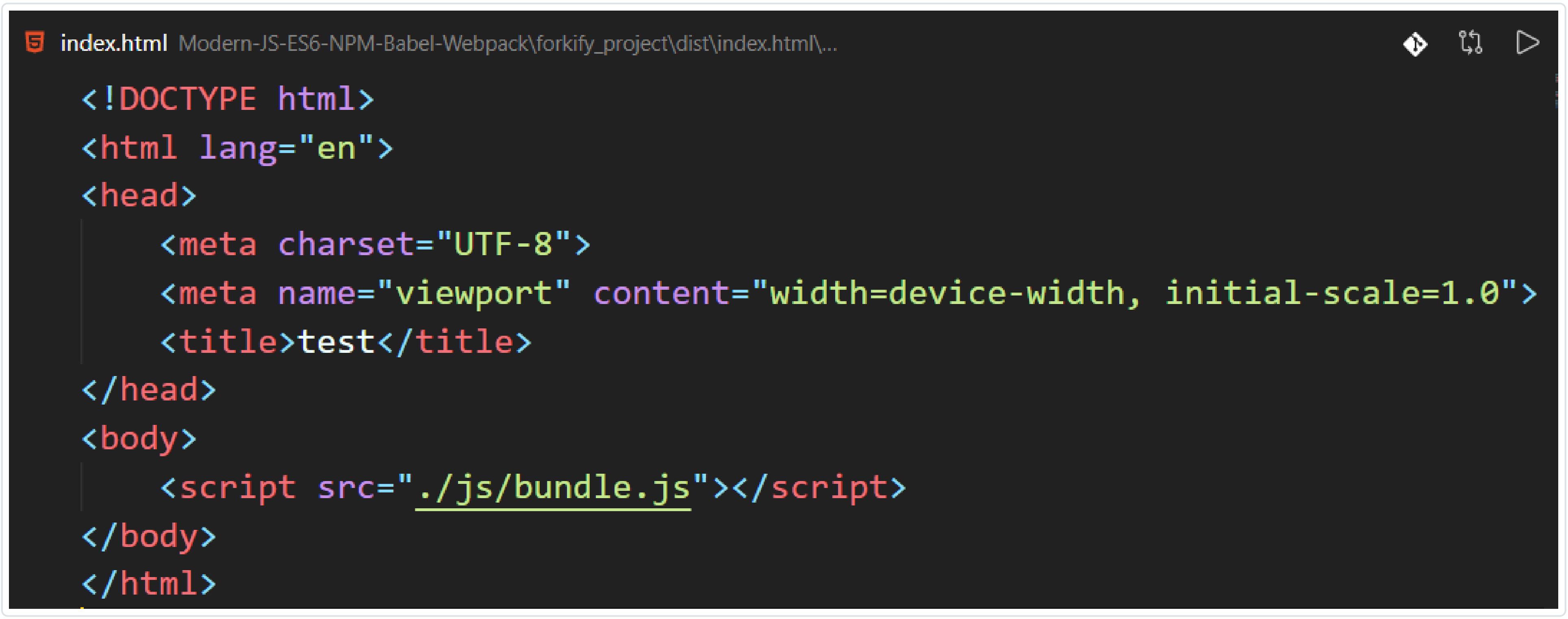
We can see that `bundle.js` has been created inside the `dist/js/` as shown below.


```
forkify_project
  dist
    css
    img
    js
      bundle.js
  node_modules
  src
    js
      index.js
      test.js
    index.html
  package-lock.json
  package.json
  webpack.config.js
```

Finally, we make a dummy html file, say `index.html` inside `/dist/` as shown below.

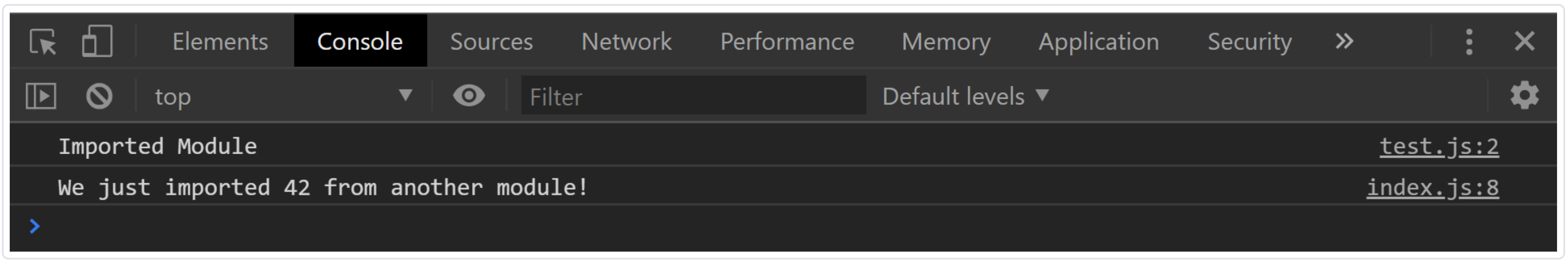

```
forkify_project
  dist
    css
    img
    js
      bundle.js
    index.html
```

And then import `bundle.js` in `index.js` as shown below.


index.html Modern-JS-ES6-NPM-Babel-Webpack\forkify_project\dist\index.html\...

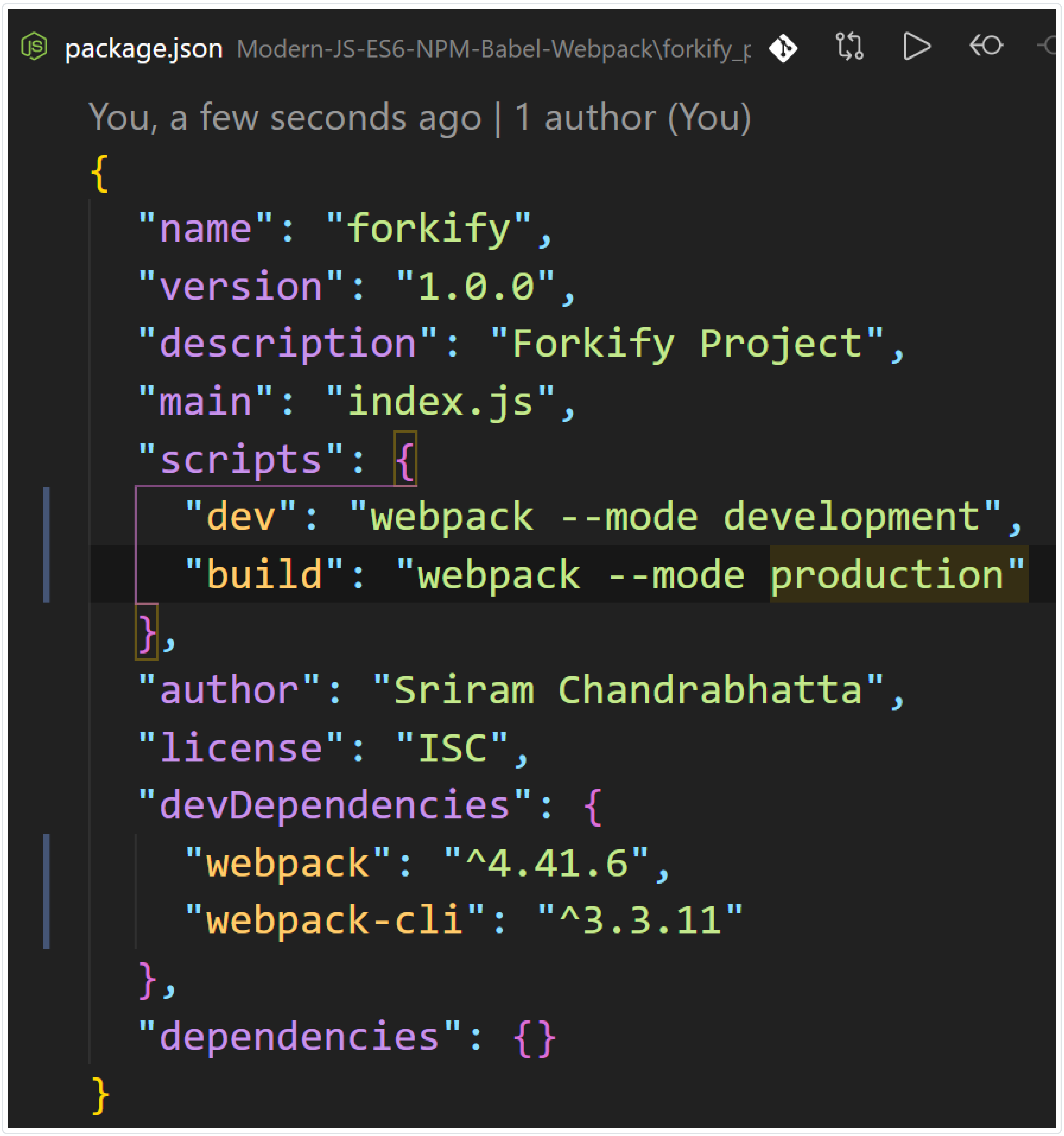
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>test</title>
</head>
<body>
    <script src="./js/bundle.js"></script>
</body>
</html>
```

We open `/dist/index.html` and then check the developer console for the output, and we should see the output shown below.



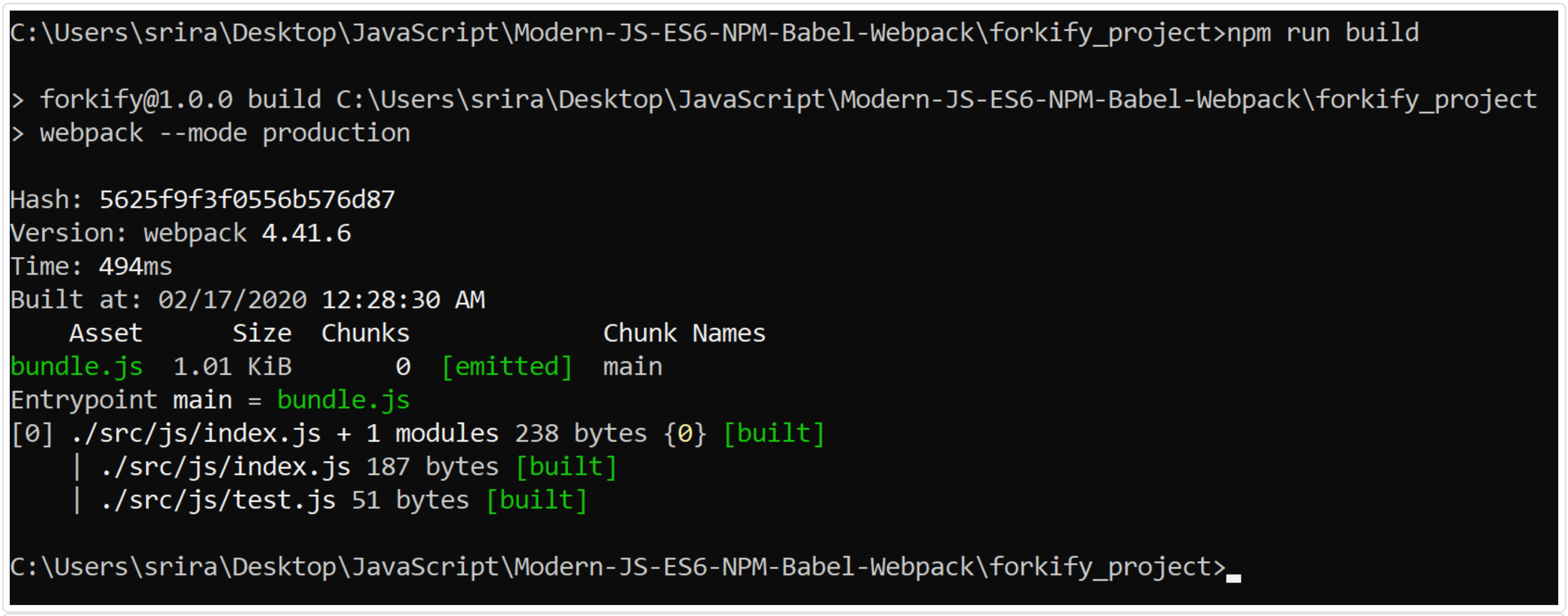
```
Imported Module                                   test.js:2
We just imported 42 from another module!          index.js:8
```

Now, we bundled our code on the basis that the `mode` in the `webpack.config.js` file was `"development"`. If we change the `mode` to `"production"`, the bundled file generated from webpack will be extremely small, as it will be minified and completely optimized. But, if we want to produce production level bundled file a lot of times, we can just manually go into the `webpack.config.js` file and change the `mode` field every time. But that would be counter-productive. So, we generally would write **npm scripts** for such things. Inside the `package.json` file, we will write an npm script inside the `"scripts"` object, where we write a script to be run for `"dev"` and we write another script for `"build"` as shown below (Note that `"build"` script should be run when we want to bundle the project for production).

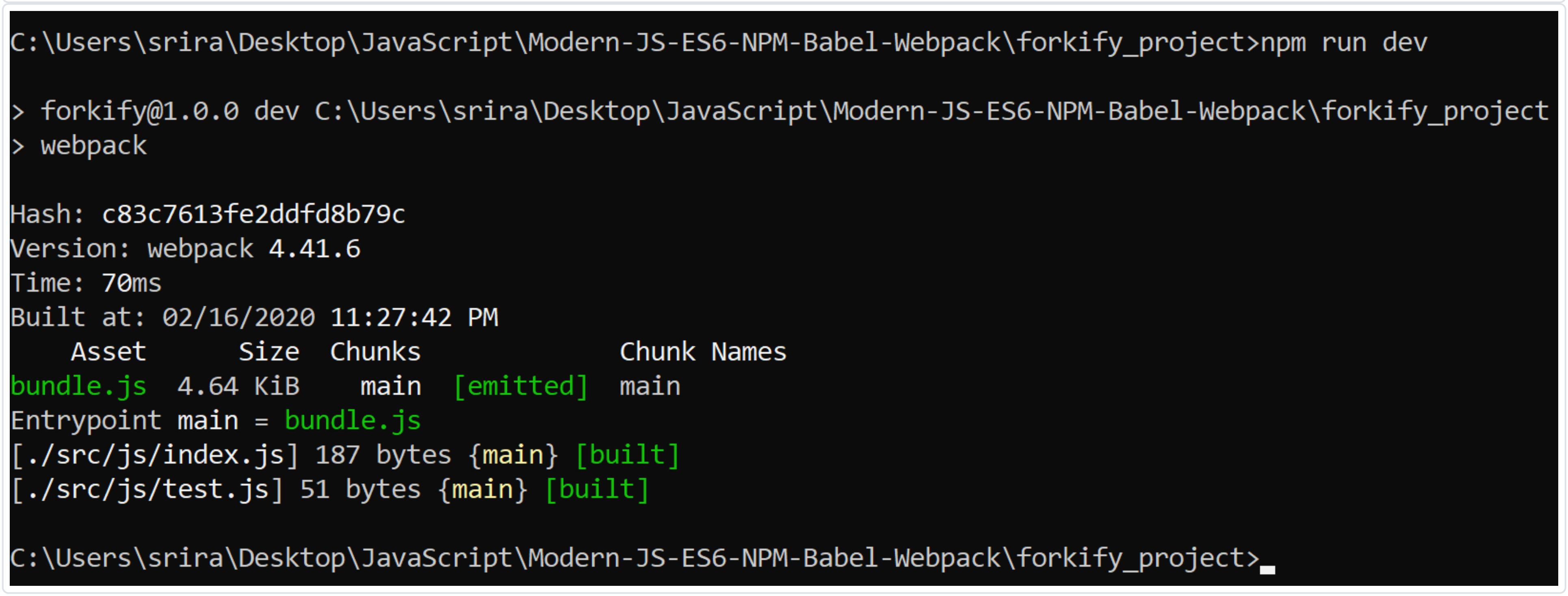

package.json Modern-JS-ES6-NPM-Babel-Webpack\forkify_p
You, a few seconds ago | 1 author (You)

```
{
  "name": "forkify",
  "version": "1.0.0",
  "description": "Forkify Project",
  "main": "index.js",
  "scripts": {
    "dev": "webpack --mode development",
    "build": "webpack --mode production"
  },
  "author": "Sriram Chandrabhatta",
  "license": "ISC",
  "devDependencies": {
    "webpack": "^4.41.6",
    "webpack-cli": "^3.3.11"
  },
  "dependencies": {}
}
```

And we will test whether the scripts are working properly or not, by typing in the commands shown below.



```
C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project>npm run build

> forkify@1.0.0 build C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project
> webpack --mode production

Hash: 5625f9f3f0556b576d87
Version: webpack 4.41.6
Time: 494ms
Built at: 02/17/2020 12:28:30 AM
    Asset      Size  Chunks             Chunk Names
bundle.js  1.01 KiB       0  [emitted]  main
Entrypoint main = bundle.js
[0] ./src/js/index.js + 1 modules 238 bytes {0} [built]
    | ./src/js/index.js 187 bytes [built]
    | ./src/js/test.js 51 bytes [built]

C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project>
```



```
C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project>npm run dev

> forkify@1.0.0 dev C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project
> webpack

Hash: c83c7613fe2ddfd8b79c
Version: webpack 4.41.6
Time: 70ms
Built at: 02/16/2020 11:27:42 PM
    Asset      Size  Chunks             Chunk Names
bundle.js  4.64 KiB    main  [emitted]  main
Entrypoint main = bundle.js
[./src/js/index.js] 187 bytes {main} [built]
[./src/js/test.js] 51 bytes {main} [built]

C:\Users\srira\Desktop\JavaScript\Modern-JS-ES6-NPM-Babel-Webpack\forkify_project>
```

We can clearly see that the `build.js` generated after running the command: `npm run dev` had a size of 4.64 KB, whereas `bundle.js` generated from the command: `npm run build` had a size of 1.01 KB, which is very less compared to 4.64 KB, as the it is a minified and optimized bundle.

Now running these **npm scripts** over and over again whenever we want to see the results of the code that we wrote can be a hassle. Therefore, we can make our lives easier by installing the **Webpack Dev Server**, which will automate a lot of tasks related to npm scripts. We will see that later.