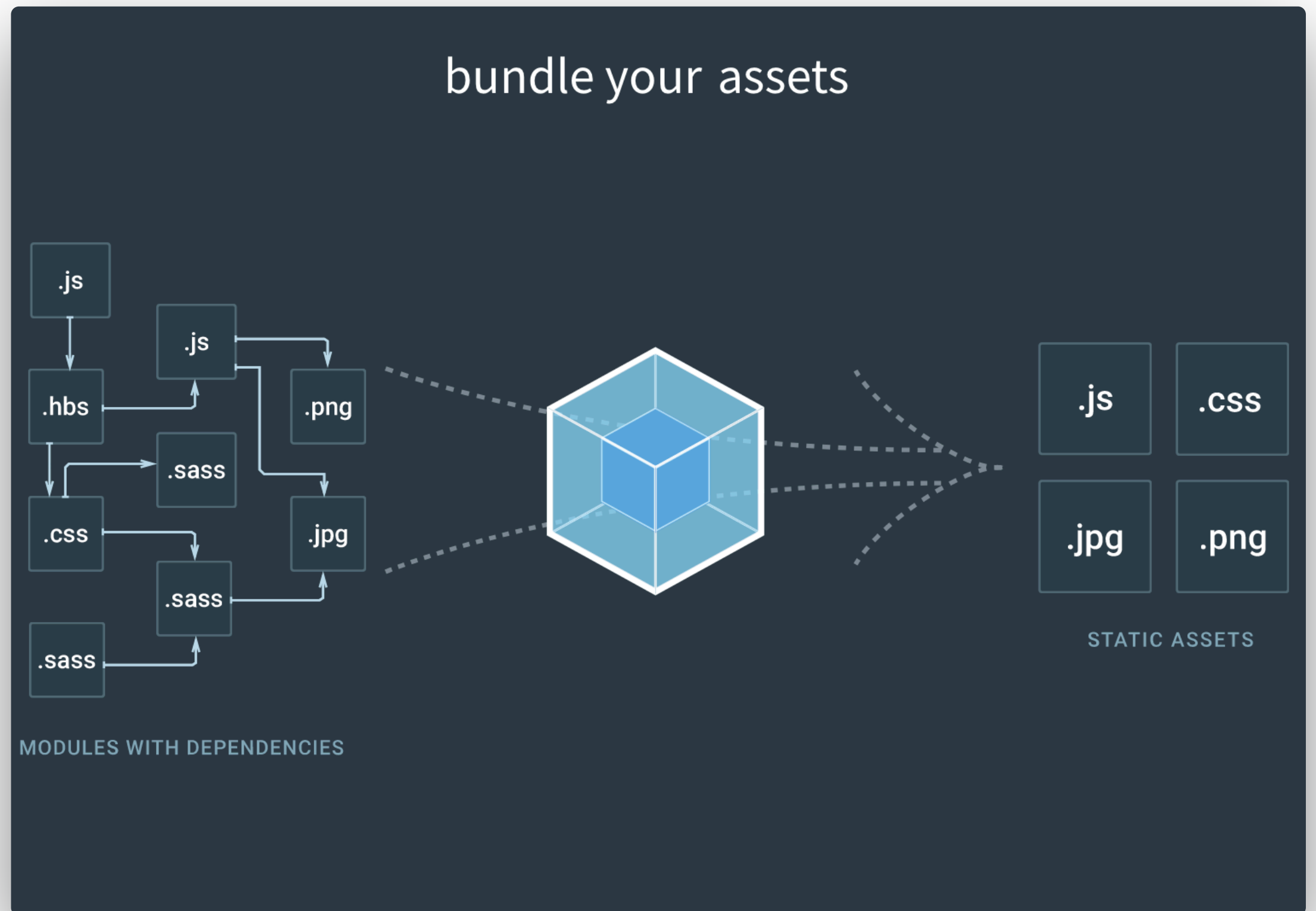




**webpack**

# – What is Webpack?

A static asset  
bundler



# – What's in a Bundle?

A JavaScript  
file built for  
browser  
compatibility



The screenshot shows a code editor window titled 'bundle.js' with a dark theme. The code is as follows:

```
1  var MODULES = {
2    sum: function () {
3      return function sum () {
4        return a + b
5      }
6    },
7
8    _entry: function () {
9      var sum = MODULES.sum()
10
11      console.log(sum(1, 2))
12    }
13  }
14
15  MODULES._entry()
16
```

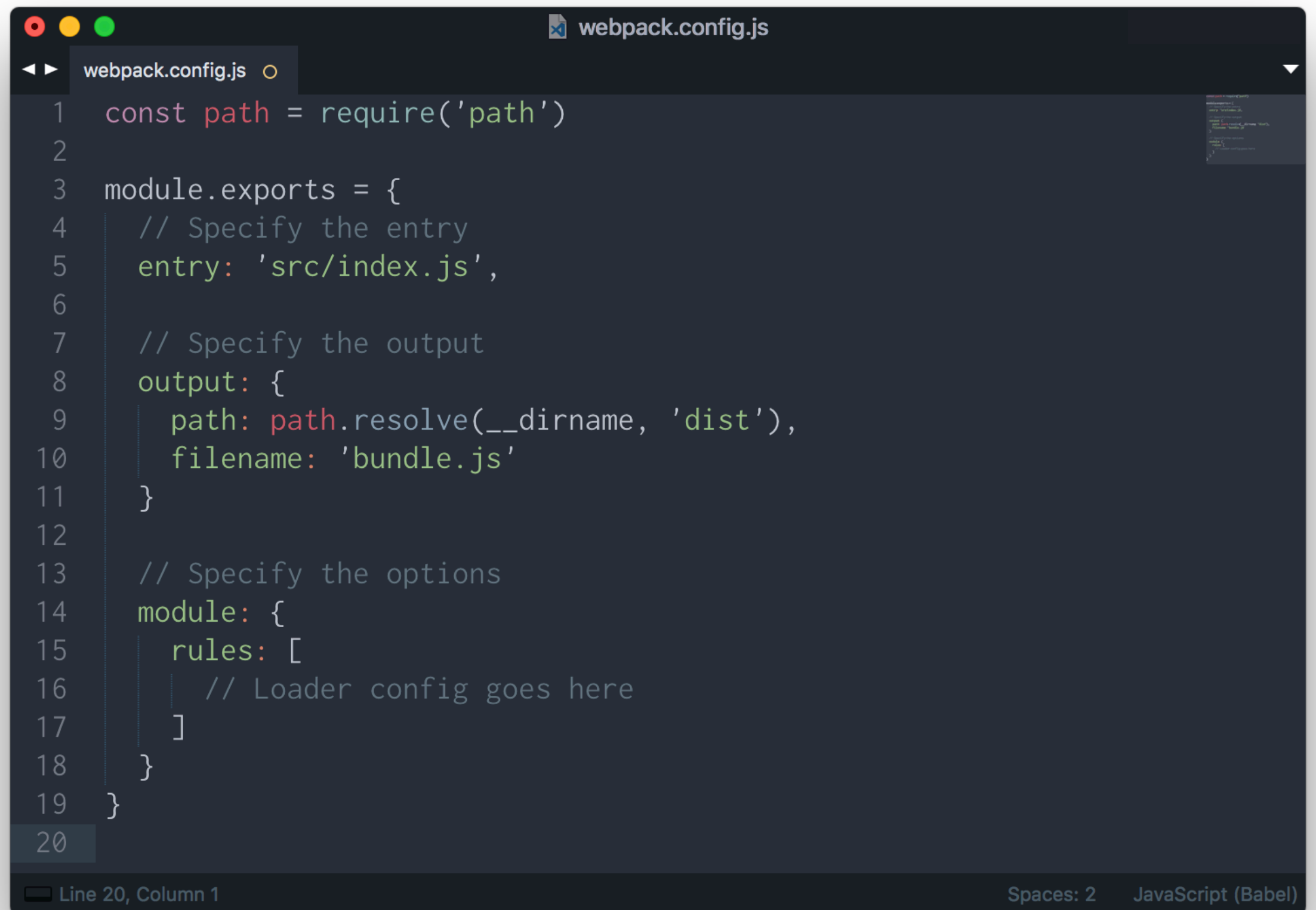
The editor interface includes a tab for 'bundle.js', a status bar at the bottom left showing 'Line 16, Column 1', and a bottom right area showing 'Spaces: 2' and 'JavaScript (Babel)'. The text 'UNREGISTERED' is visible in the top right corner of the editor window.

A simplified bundle.js file



# – The Configuration File

webpack.config.js  
is used to  
configure the  
input, output,  
and rules for  
creating the  
bundle

A screenshot of a code editor window titled 'webpack.config.js'. The editor shows a JavaScript configuration file for Webpack. The code is as follows:

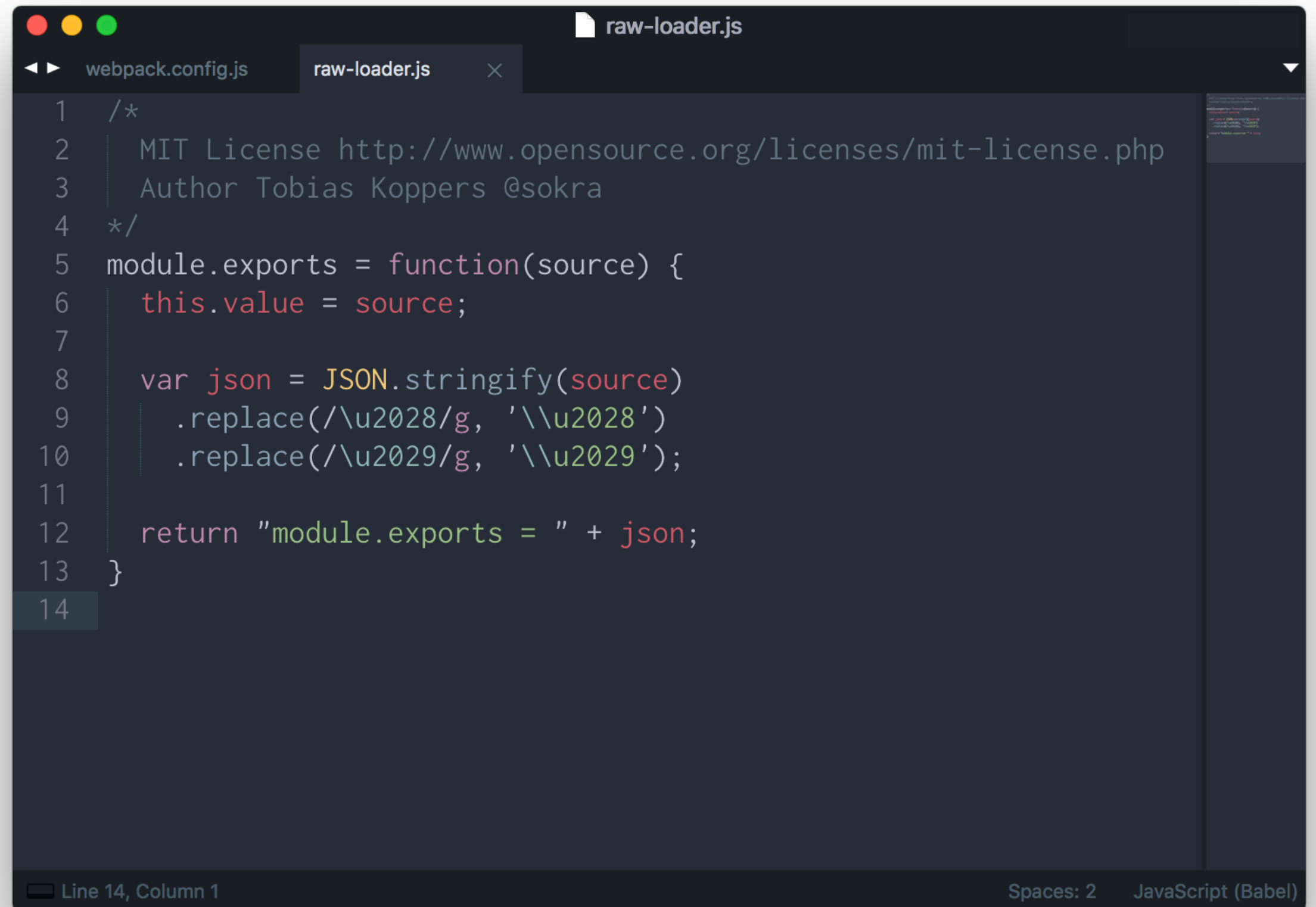
```
1  const path = require('path')
2
3  module.exports = {
4    // Specify the entry
5    entry: 'src/index.js',
6
7    // Specify the output
8    output: {
9      path: path.resolve(__dirname, 'dist'),
10     filename: 'bundle.js'
11   }
12
13   // Specify the options
14   module: {
15     rules: [
16       // Loader config goes here
17     ]
18   }
19 }
20
```

The editor has a dark theme. The status bar at the bottom indicates 'Line 20, Column 1', 'Spaces: 2', and 'JavaScript (Babel)'. There is a small preview window on the right side of the editor.

A basic configuration file

# – Loaders

Functions to transform the contents of a file



```
1  /*
2   MIT License http://www.opensource.org/licenses/mit-license.php
3   Author Tobias Koppers @sokra
4  */
5  module.exports = function(source) {
6    this.value = source;
7
8    var json = JSON.stringify(source)
9      .replace(/\u2028/g, '\\u2028')
10     .replace(/\u2029/g, '\\u2029');
11
12    return "module.exports = " + json;
13  }
14
```

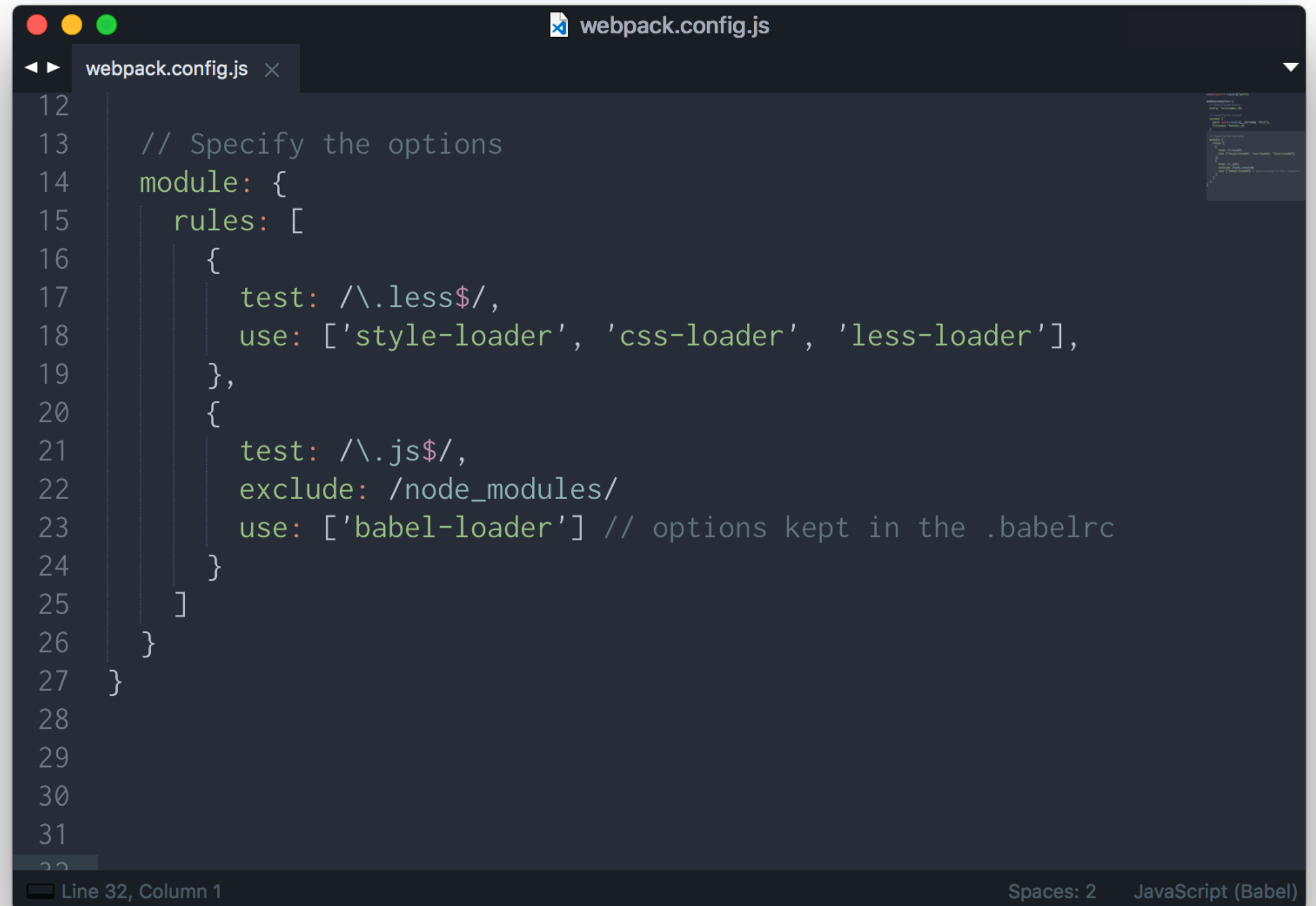
Line 14, Column 1      Spaces: 2      JavaScript (Babel)



The entirety of raw-loader

# – Rules

Combinations of loaders and options for the files that they affect



```
12
13 // Specify the options
14 module: {
15   rules: [
16     {
17       test: /\.less$/,
18       use: ['style-loader', 'css-loader', 'less-loader'],
19     },
20     {
21       test: /\.js$/,
22       exclude: /node_modules/,
23       use: ['babel-loader'] // options kept in the .babelrc
24     }
25   ]
26 }
27 }
```

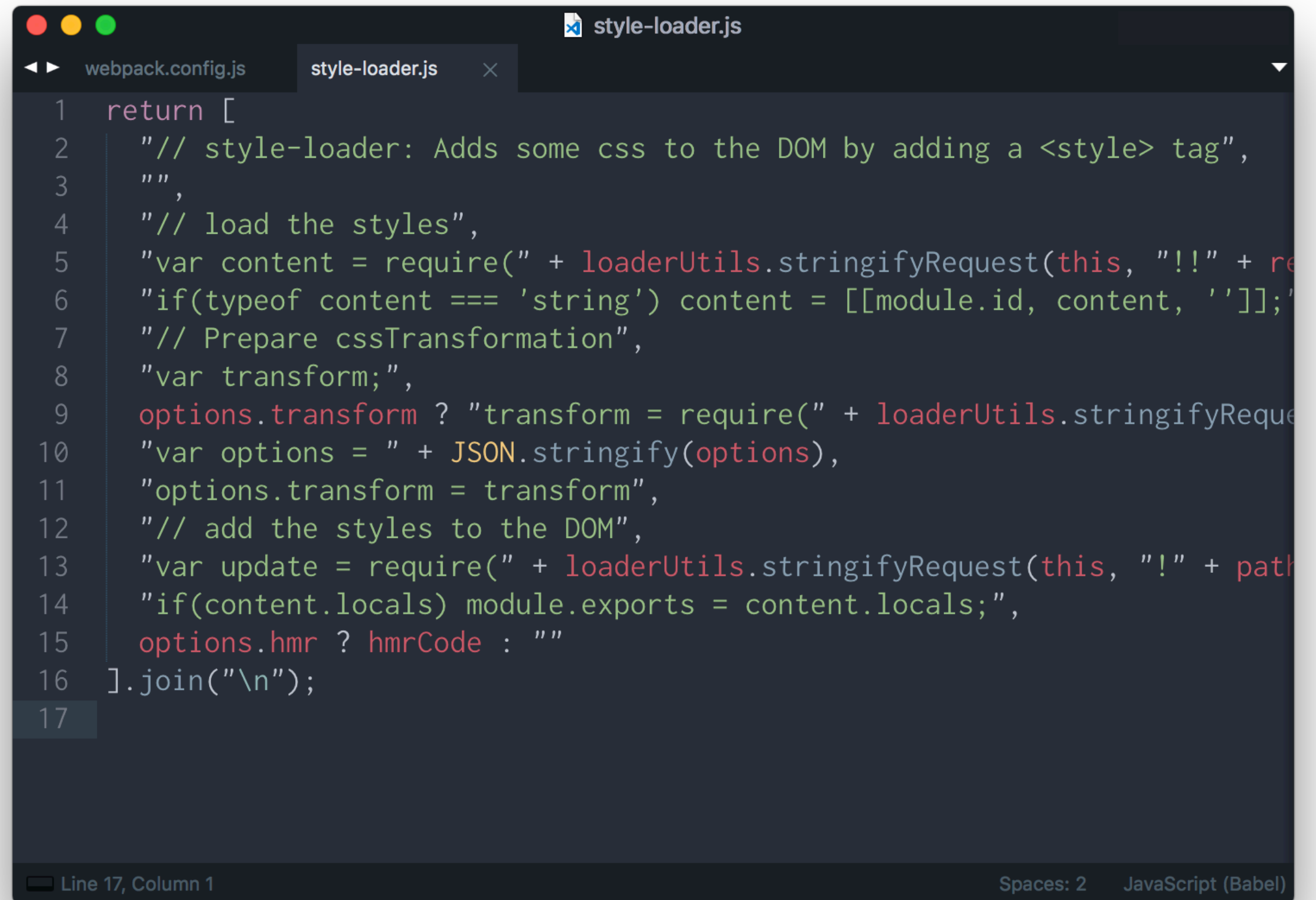
Line 32, Column 1      Spaces: 2      JavaScript (Babel)

A couple of example rules



# – Loading Non-JS Assets

Some loaders will add things like style tags to the page with JS



```
1  return [
2    "// style-loader: Adds some css to the DOM by adding a <style> tag",
3    "",
4    "// load the styles",
5    "var content = require(" + loaderUtils.stringifyRequest(this, "!!" + re
6    "if(typeof content === 'string') content = [[module.id, content, '']];"
7    "// Prepare cssTransformation",
8    "var transform;",
9    options.transform ? "transform = require(" + loaderUtils.stringifyReque
10   "var options = " + JSON.stringify(options),
11   "options.transform = transform",
12   "// add the styles to the DOM",
13   "var update = require(" + loaderUtils.stringifyRequest(this, "!" + path
14   "if(content.locals) module.exports = content.locals;",
15   options.hmr ? hmrCode : ""
16 ].join("\n");
17
```

The relevant section of style-loader



# – Multiple Entry Points

The output filenames are specified with a special syntax



```
webpack.config.js
1  const path = require('path')
2
3  module.exports = {
4    entry: {
5      bundle1: 'src/app1.js',
6      bundle2: 'src/app2.js'
7    },
8
9    output: {
10     path: path.join(__dirname, 'dist'),
11     filename: '[name].[chunkhash].js'
12   }
13 }
14
```

Line 14, Column 1      Spaces: 2      JavaScript (Babel)

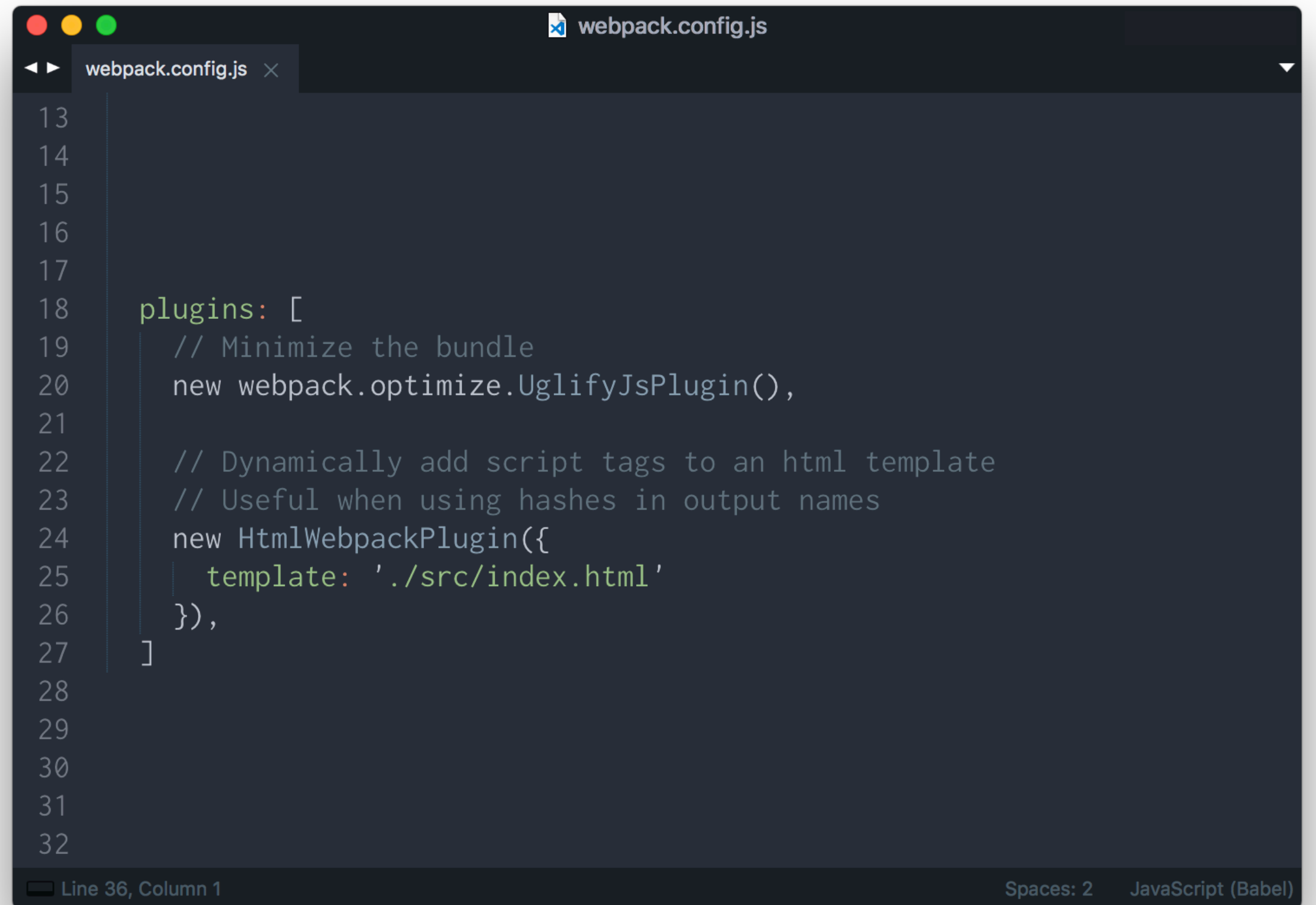


Creating two bundles from two entry points



# – Plugins

Functions that modify Webpack's output. More flexible than loaders

A screenshot of a code editor window titled 'webpack.config.js'. The editor shows a JavaScript configuration file with a 'plugins' array. The array contains two plugin instances: 'UglifyJsPlugin' for minification and 'HtmlWebpackPlugin' for adding HTML templates. The code is syntax-highlighted, with comments in grey and function names in blue. Line numbers 13 through 32 are visible on the left side of the editor. The status bar at the bottom indicates 'Line 36, Column 1', 'Spaces: 2', and 'JavaScript (Babel)'.

```
13
14
15
16
17
18   plugins: [
19     // Minimize the bundle
20     new webpack.optimize.UglifyJsPlugin(),
21
22     // Dynamically add script tags to an html template
23     // Useful when using hashes in output names
24     new HtmlWebpackPlugin({
25       template: './src/index.html'
26     }),
27   ]
28
29
30
31
32
```

A few plugins in webpack.config.js



# – Using CommonsChunkPlugin

Library code  
will change  
less frequently  
than app code  
and can be  
cached



```
webpack.config.js
1  const path = require('path')
2
3  module.exports = {
4    entry: {
5      commons: ['moment', 'lodash'],
6      bundle: 'src/index.js'
7    },
8
9    output: {
10     path: '[name].[chunkhash].js'
11   },
12
13   plugins: [
14     // Minimize the bundle
15     new webpack.optimize.CommonsChunkPlugin({
16       name: 'commons'
17     })
18   ]
19 }
20
```

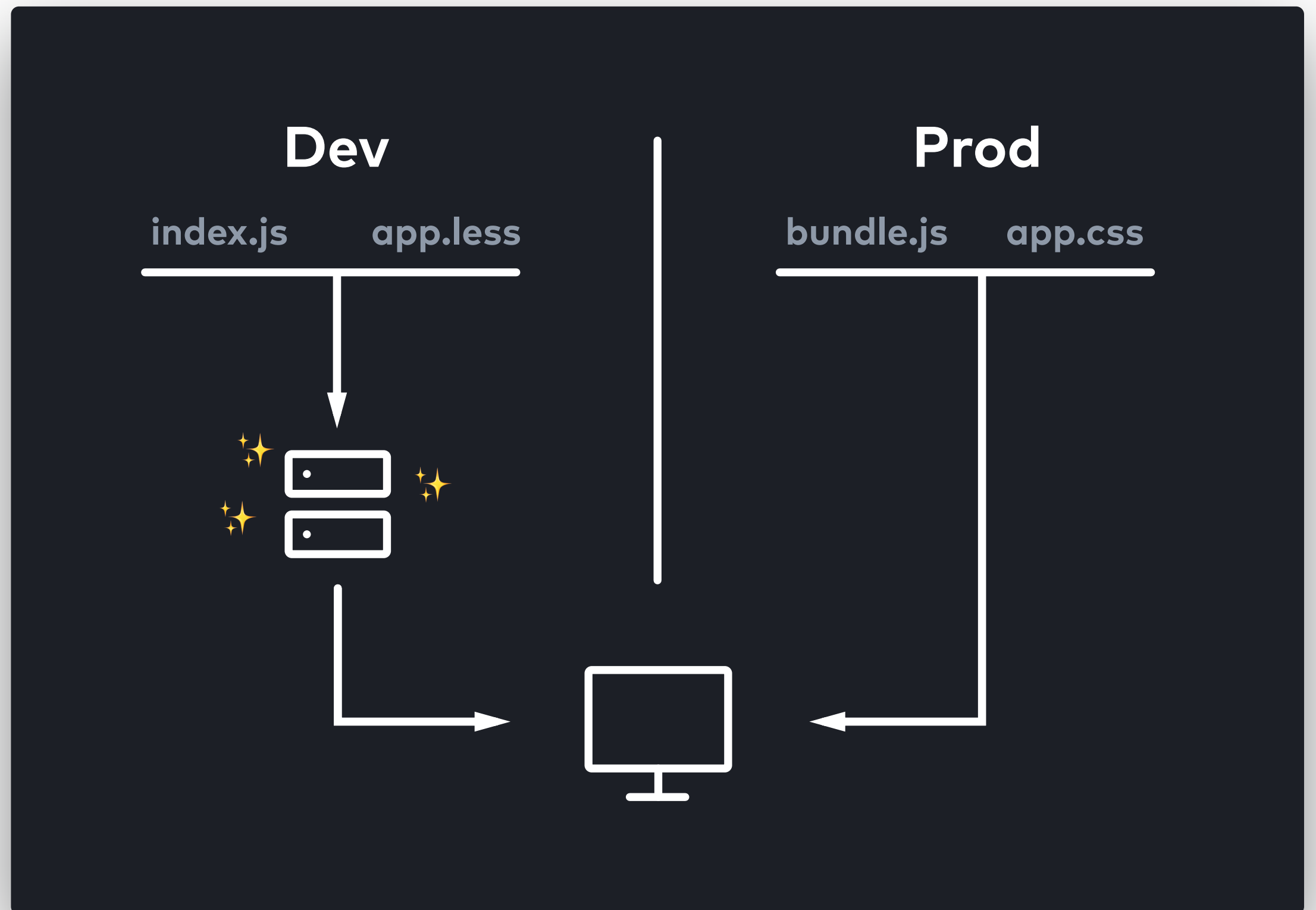
Line 20, Column 1      Spaces: 2    JavaScript (Babel)



Creating a commons bundle

# – Webpack Dev Server

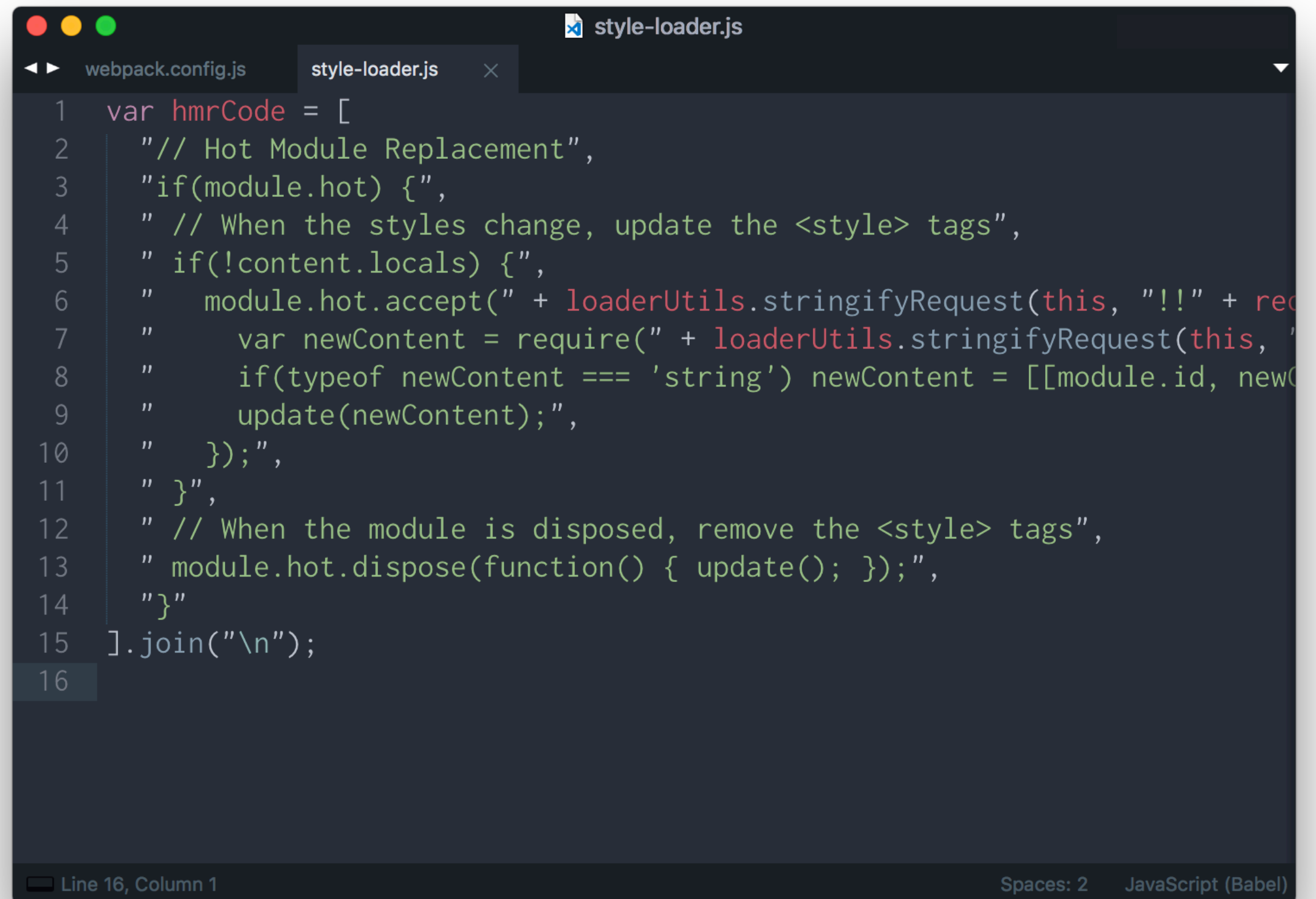
Re-bundles  
and swaps out  
assets on the  
page



Dev server doing it's magic

# – Loaders & Webpack Dev Server

Loaders can implement the HMR interface to support use with the dev server



```
1  var hmrCode = [  
2    "// Hot Module Replacement",  
3    "if(module.hot) {",  
4    "  // When the styles change, update the <style> tags",  
5    "  if(!content.locals) {",  
6    "    module.hot.accept(" + loaderUtils.stringifyRequest(this, "!!" + req  
7    "    var newContent = require(" + loaderUtils.stringifyRequest(this, "  
8    "    if(typeof newContent === 'string') newContent = [[module.id, newC  
9    "    update(newContent);",  
10   "  });",  
11   "  }",  
12   "  // When the module is disposed, remove the <style> tags",  
13   "  module.hot.dispose(function() { update(); });",  
14   "}"  
15 ].join("\n");  
16
```

Line 16, Column 1      Spaces: 2      JavaScript (Babel)



The HMR code from style-loader

## – Built on Top of Webpack

Several static  
site generators  
have adopted  
Webpack as a  
base



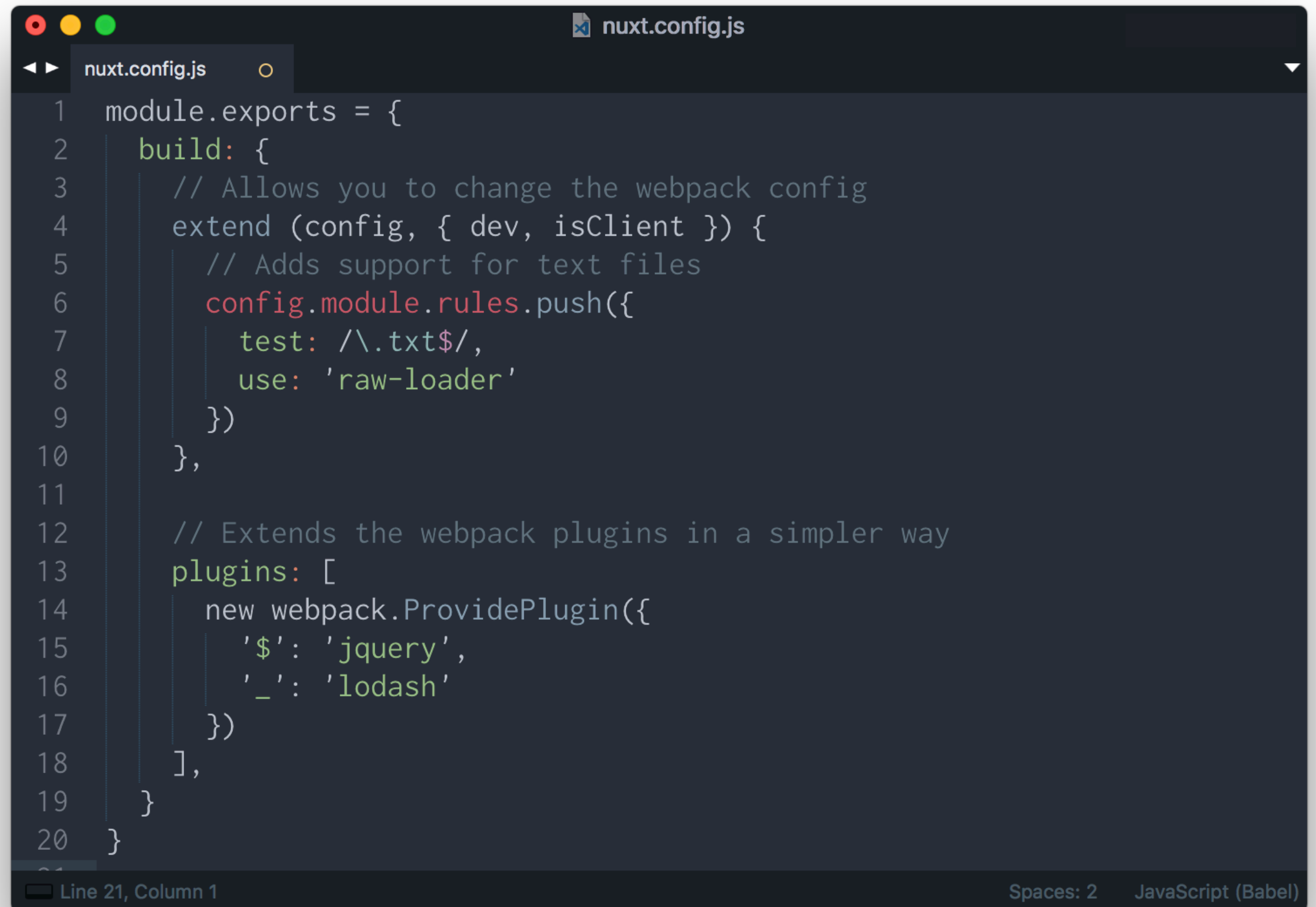
`create-react-app`

`create-vue-app`  
(through poi)



# – Extending Webpack Configs

Each tool has  
its own syntax  
for extending  
the config



```
1  module.exports = {
2    build: {
3      // Allows you to change the webpack config
4      extend (config, { dev, isClient }) {
5        // Adds support for text files
6        config.module.rules.push({
7          test: /\.txt$/,
8          use: 'raw-loader'
9        })
10     },
11
12     // Extends the webpack plugins in a simpler way
13     plugins: [
14       new webpack.ProvidePlugin({
15         '$': 'jquery',
16         '_': 'lodash'
17       })
18     ],
19   }
20 }
```

Line 21, Column 1      Spaces: 2    JavaScript (Babel)