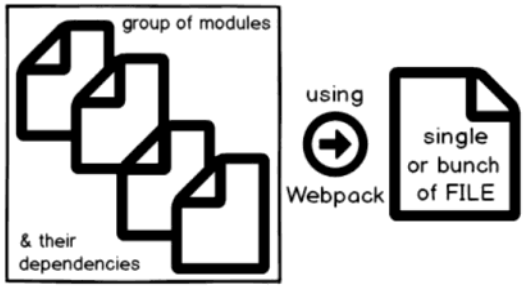
WEBPACK, m*odule bundler*

25 December 2020 18:17



Webpack is a **module bundler** and has a broader definition of what a module is, specifically, for webpack, modules are:

- Common JS modules
- AMD modules
- CSS import
- Images url
- ES modules

The ultimate goal of webpack is to unify all these different sources and module types in a way that's possible to import everything in your JavaScript code, and finally produce a shippable output.

Webpack goes through your package and creates what it calls a **dependency graph** (Any time one file depends on another, webpack treats this as a *dependency*. It takes non-code assets, such as images or web fonts, and also provide them as *dependencies* for your application.)
which consists of various **modules** which your webapp 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.

To get started you only need to understand its **Core Concepts**:

- Entry
- Output
- Loaders
- Plugins
- Mode
- Browser Compatibility

ENTRY:

An entry point indicates which module webpack should use to begin building out its internal dependency graph. webpack will figure out which other modules and libraries that entry point depends on (directly and indirectly).

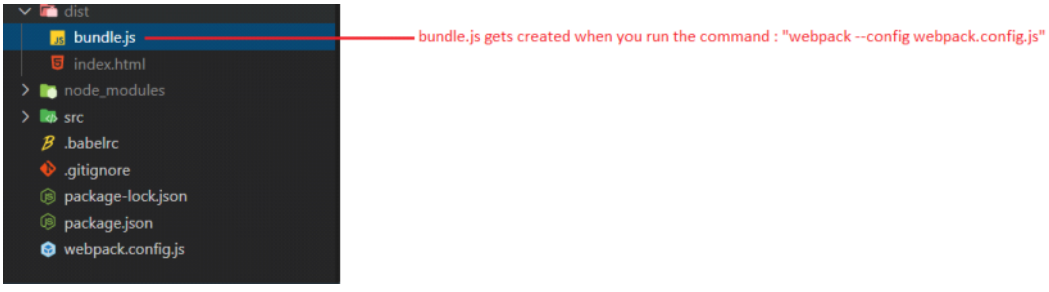
By default its value is `./src/index.js`, but you can specify a different (or multiple entry points) by setting an `entry` property in the webpack configuration. For example:

```
module.exports = {
  entry: './path/to/my/entry/file.js'
};
```

OUTPUT:

The `output` property tells webpack where to emit the *bundles* it creates and how to name these files.
It defaults to `./dist/main.js` for the main output file and to the `./dist` folder for any other generated file. For e.g. when you configure the below in the webpack configuration file (webpack.config.js)

```
// The output property specifies where the bundled file should be generated and what the name of the bundled file would be.
// This is done by the output.path and output.filename properties.
output: {
    path: path.join(__dirname, '/dist'),
    filename: 'bundle.js'
},
```



LOADERS:

Out of the box, webpack only understands JavaScript and JSON files. Loaders allow webpack to process other types of files and convert them into valid modules that can be consumed by your application and added to the dependency graph.
At a high level, loaders have two properties in your webpack configuration:

- The **test** property identifies which file or files should be transformed.
- The **use** property indicates which loader should be used to do the transforming.

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

module.exports = {
  output: {
    filename: 'my-first-webpack.bundle.js'
  },
  module: {
    rules: [
      { test: /\.txt$/, use: 'raw-loader' }
    ]
  }
};
```

The configuration above has defined a `rules` property for a single module with two required properties: `test` and `use`. This tells webpack's compiler the following:

> *"Hey webpack compiler, when you come across a path that resolves to a '.txt' file inside of a `require()` / `import` statement, use the `raw-loader` to transform it before you add it to the bundle."*

PLUGINS:

Plugins perform a wider range of tasks like bundle optimization, asset management and injection of environment variables.
In order to use a plugin, you need to require() it and add it to the plugins array.

```
You, an hour ago | 1 author (You)
// start by requiring the default path module to access the file location and make changes to the file location.
const path = require('path');
// the HTMLWebpackPlugin to generate an HTML file to be used for serving bundled JavaScript file/files.
const HtmlWebpackPlugin = require('html-webpack-plugin');
```

```
// Setup plugin to use a HTML file for serving bundled js files
plugins: [
  // The plugin will generate an HTML5 file for you that includes all your webpack bundles in the body using script tags
  new HtmlWebpackPlugin({
    template: './src/index.html'      You, 3 hours ago • webpack-demo
  })
]
```

MODE:

By setting the mode parameter to either development, production or none, you can enable webpack's built-in optimizations that correspond to each environment. The default value is production.

```
// Default mode for Webpack is production.
// Depending on mode Webpack will apply different things
// on final bundle. For now we don't need production's JavaScript
// minifying and other thing so let's set mode to development
mode: 'development',
```

BROWSER COMPATIBILITY:

webpack supports all browsers that are ES5-compliant (IE8 and below are not supported). webpack needs Promise for import() and require.ensure(). If you want to supportolder browsers, you will need to load a polyfill before using these expressions.

Sources:
https://www.valentinog.com/blog/webpack/
https://webpack.js.org/