WEBPACK

Config File

Sreekanth M. E.

Freelance Trainer & Consultant

http://www.SreekanthME.com

Webpack has a lot of configuration options.

So using command line is not practical.

Hence there is an option to use a config file named: webpack.config.js

webpack.config.js uses commonJS format to import/export modules.

webpack.config.js basically exports a config object.

Webpack is highly configurable, and has four Core Concepts:

- Entry
- Output
- Loaders
- Plugins

Each of these are keys of the config object.

ENTRY

The starting point of the dependency graph is known as an entry point.

```
entry: './path/to/my/entry/file.js'
```

is short form of:

```
entry: {
  main: './path/to/my/entry/file.js'
}
```

Multiple entry points are possible:

```
entry: {
    app: './src/app.js',
    vendors: './src/vendors.js'
}
```

OUTPUT

The output property tells webpack how to treat bundled code.

At the minimum, it must have following settings:

- A filename to use for the output file(s).
- An absolute path to output directory.

```
output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'my-first-webpack.bundle.js’
}
```

Node’s in-built path module is used to get absolute path.

The term emitted or emit is used in webpack documentation to mean 'produced' or 'discharged’ or "created“.

LOADERS

Loaders are transformers that act on the source code of a module during the bundling process.

They allow pre-processing files as they are imported or loaded.

Loaders can perform actions like linting, transpiling, reading CSS, compression etc.

List of loaders can be found at:

https://webpack.js.org/loaders/

Loaders can be chained.

When chained, they will be applied from right to left (last to first configured).

Loaders run in Node.js and can do everything that’s possible there.

Loaders can also be configured with an options option.

These options are passed to the loader as parameters.

module.rules key of the config object is used to configure loaders.

```
module: {rules: [
    { test: /\.css$/, use: 'css-loader' },
    { test: /\.ts$/, use: 'ts-loader' }
  ]  }
```

test: takes a regexp to match files.

/\.css$/ → matches all .css files.

use: takes list of loaders

PLUGINS

While Loaders only execute transforms on a per-file basis, plugins are most commonly used to perform actions and custom functionality on "compilations" or "chunks" of the bundled modules

In order to use a plugin, an object of the plugin has to be created and added to plugins array of config object's plugins key.

Most plugins are customizable via options.

Plugins can be used multiple times in a config object for different purposes

List of OOTB plugins:

https://webpack.js.org/concepts/plugins/

3rd party plugins:

https://github.com/webpack-contrib/awesome-webpack#webpack-plugins

END OF CHAPTER

APPENDIX