插件 / plugin

定义

是webpack的支柱功能

webpack自身也构建在一个webpack配置中用到的相同插件系统上

解决loader无法实现的其他事

Tip：

使用webpack-sources的包的时候，通过require("webpack").source进行引用，避免持久缓存的版本冲突

解析

插件是一个JavaScript对象 / 类

具有apply方法，会被webpack compiler调用

调用apply方法的时候会传入compiler对象，以便可以使用它的各种hooks，在整个编译声明周期都可以访问compiler

compiler hook的tap方法的第一个参数应该是驼峰式命名的插件名称

建议使用一个常量保存，如pluginName, 这样可以在所有hooks中重复使用

如下在compile过程中进行输出的插件

```
const pluginName = "ConsoleLogOnBuildWebpackPlugin";

class ConsoleLogOnBuildWebpackPlugin {
   apply(compiler) {
      compiler.hooks.run.tap(pluginName,(compilation) => {
         console.log("webpack 构建正在启动")
      })
   }
}

module.exports = ConsoleLogOnBuildWebpackPlugin;
```

用法

插件可以在创建实例对象的时候携带参数 / 选项

在webpack配置中，如果需要重复使用，并且携带参数，向plugins传入的就必须是一个new出来的实例

如下

```
const HtmlWebpackPlugin = require("html-webpack-plugin");
const webpack = require("webpack"); // 用于访问内置的插件
const path = require("path"); // 用于操作路径

module.exports = {
    entry : "./src/app.js",
    output: {
        filename: [name].js,
        path: path.resolve(__dirname, "dist"),
    },
    module: {
        rules: [
            {
                test: /\.(js|jsx)$/,
                use: "babel-loader",
            }
        ]
    }
    plugins:[
        new webpack.ProgressPlugin(),
        new HtmlWebpackPlugin({ template: "./src/index.html" }),
    ]
}
```

ProgressPlugin用于自定义编译过程中的进度报告

HtmlWebpackPlugin用于单页应用，将会生成一个html文件，并且自动将output的所有bundle都在其中引入，在这里是app.js

Node API方式

可以使用Node API，直接访问webpack运行时，然后传入配置，得到compiler对象，然后就可以直接调用插件的apply方法了

```
const webpack = require("webpack"); // 访问webpack运行时
const configuration = require("./webpack.config.js");

let compiler = webpack(configuration); // 将配置文件传入运行时

new webpack.ProgressPlugin().apply(compiler); // 自定义编译过程中的进度报告

compiler.run(function (err,stats){
    // ...
})
```