## 阅读本系列教程，你需要对ES6的import，export有了解

本系列教程1-8为webpack概念说明文档，其余为实际应用教程

本教程转自Webpack中文文档：<https://www.webpackjs.com/concepts/>，经过个人修改而成

笔者不生产水，我只是大自然的搬运工

**Webpack做了什么？**

如我们在src文件夹下面有a.js，b.js，c.js，其中a.js import了b.js，而b.js又import了c.js，而webpack可以将者3个文件打包成一个文件，其中a,b,c文件称为chunk（组块），打包出来的文件称为bundle（包）

从这可以看出Webpack主要功能是将我们的多个文打包为一个文件

## **入口(entry)**

可以通过在 [webpack 配置](https://www.webpackjs.com/configuration)中配置 entry 属性，来指定一个入口起点（或多个入口起点）如下指定我们的入口文件为file.js

webpack.config.js

module.exports = {

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

};

## **出口(output)**

output 属性告诉 webpack 在哪里输出它所创建的 *bundles（包，打包了的文件）*，以及如何命名这些文件，默认值为 ./dist。你可以通过在配置中指定一个 output 字段，来配置这些处理过程：

****webpack.config.js****

const path = require('path');

module.exports = {

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

output: {

path: path.resolve(\_\_dirname, 'dist'),

filename: 'my-first-webpack.bundle.js'

}};

Path为输出路径，filename为输出文件名称

## **loader**

webpack 无法处理非 JavaScript 文件（如：import img from ‘a.jpg’），这就需要一个loader将文件转为webpack可以处理的格式

在 webpack 的配置中 ****loader**** 有两个属性：

1. test 属性，用于匹配需要使用loader转换处理的文件。
2. use 属性，表示进行转换时，应该使用哪个 loader。

****webpack.config.js****

const path = require('path');

const config = {

output: {

filename: 'my-first-webpack.bundle.js'

},

module: {

rules: [

{ test: /\.txt$/, use: 'raw-loader' }

]

}};

module.exports = config;

以上配置中，告诉 webpack 编译器(compiler) 如下信息：

嘿，webpack 编译器，当你碰到「在 require()/import 语句中被解析为 '.txt' 的路径」时，在你对它打包之前，先****使用**** raw-loader 转换一下。

## **插件(plugins)**

loader 被用于转换某些类型的模块（也就是一个文件），而插件则可以用于执行范围更广的任务。插件的范围包括，从打包优化和压缩，一直到重新定义环境中的变量。

想要使用一个插件，你只需要 require() 它，然后把它添加到 plugins 数组中。

****webpack.config.js****

const HtmlWebpackPlugin = require('html-webpack-plugin'); // 通过 npm 安装

const config = {

module: {

rules: [

{ test: /\.txt$/, use: 'raw-loader' }

]

},

plugins: [

new HtmlWebpackPlugin({template: './src/index.html'})

]};

module.exports = config;

## **模式**

就是当前环境（如：开发环境，生成环境等），通过选择 development 或 production 之中的一个，来设置 mode 参数

module.exports = {

mode: 'production'

};