# Webpack

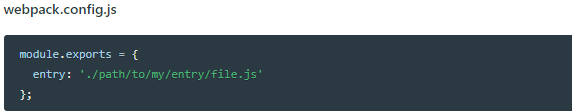
## 介绍：

<https://segmentfault.com/a/1190000017777256>

## 1、4个核心概念

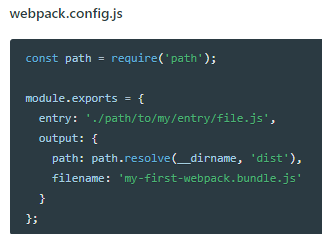
### 1、入口

**入口起点(entry point)**指示 webpack 应该使用哪个模块，来作为构建其内部依赖图的开始。进入入口起点后，webpack 会找出有哪些模块和库是入口起点（直接和间接）依赖的。每个依赖项随即被处理，最后输出到称之为 bundles 的文件中。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）。**默认值为 ./src**。



### 2、出口

**output** 属性告诉 webpack 在哪里输出它所创建的 bundles（就是输出依赖所创建的文件），以及如何命名这些文件，**默认值为 ./dist**。基本上，整个应用程序结构，都会被编译到你指定的输出路径的文件夹中。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：



注意：\_\_dirame—表示当前文件所在的目录

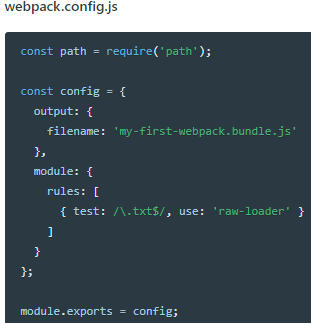
### 3、loader

loader 让 webpack 能够去处理那些非 JavaScript 文件（webpack 自身只理解 JavaScript）。loader 可以将所有类型的文件转换为 webpack 能够处理的有效[模块](https://www.webpackjs.com/concepts/modules)（使用loader将非javaScript文件处理为webpack可以认识的javaScript文件），然后你就可以利用 webpack 的打包能力，对它们进行处理。

*注意，loader 能够 import 导入任何类型的模块（例如 .css 文件）。*

在 webpack 的配置中 **loader** 有两个很重要的属性：

1. test：指定需要被转换的文件类型
2. use：指定转换时使用的loader



以上配置中，对一个单独的 module 对象定义了 rules 属性，里面包含两个必须属性：test 和 use。这告诉 webpack 编译器(compiler) 如下信息：碰到「在 require()/import 语句中引入的.txt文件」时，在对它打包之前，先**使用** raw-loader 转换一下。

***在 webpack 配置中定义 loader 时，要定义在 module.rules 中，而不是 rules****。然而，在定义错误时 webpack 会给出严重的警告。为了使你受益于此，如果没有按照正确方式去做，webpack 会“给出严重的警告”*

### 4、插件（plugins）

loader 被用于转换某些类型的模块，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。[插件接口](https://www.webpackjs.com/api/plugins)功能极其强大，可以用来处理各种各样的任务（**插件可以用来处理各种各样的任务**，功能比loader更加强大）。

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。多数插件可以通过选项(option)自定义。你也可以在一个配置文件中因为不同目的而多次使用同一个插件，这时需要通过使用 new 操作符来创建它的一个实例。



### 5、模式

通过选择development或production，来设置mode参数，可以启用相应模式下的webpack内置的优化。

## 输出

配置 output 选项可以控制 webpack 如何向硬盘写入编译文件。注意，即使可以存在多个入口起点，但只指定一个输出配置。

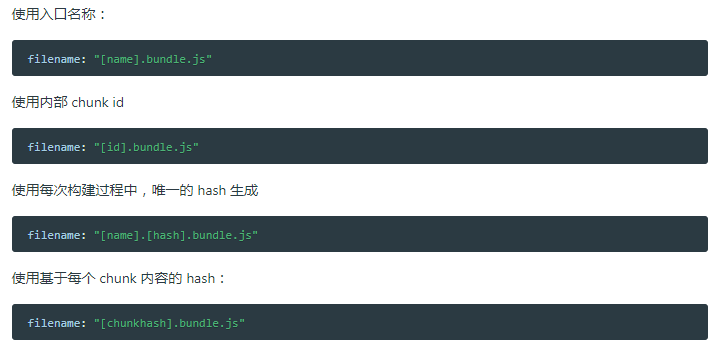


### 多个入口起点，确保对应的输出文件都有一个对应的名字

对于单个入口起点，filename会是一个静态名称---》bundle.js。



然而，当通过多个入口起点(entry point)、代码拆分(code splitting)或各种插件(plugin)创建多个 bundle，应该使用以下一种替换方式，来赋予每个 bundle 一个唯一的名称……



还有一个chunk待了解，读文档时暂时没搞明白什么意思。

**关于bundle、chunk的理解请看文档《Webpack的几个变量理解》**

### publicPath(公共路径)

webpack为项目中所有的资源（css、js、img等静态资源）提供一个基础路径，这个基础路径就是publicPath。实际上打包后，静态资源的访问路径为：**静态资源最终访问路径 = output.publicPath + 资源loader或插件等配置路径。**

关于使用webpack和webpack-dev-server打包时的publicPath区别：https://www.cnblogs.com/SamWeb/p/8353367.html