**概念：**

本质上，*webpack* 是一个现代 JavaScript 应用程序的*静态模块打包器(module bundler)*。当 webpack 处理应用程序时，它会递归地构建一个*依赖关系图(dependency graph)*，其中包含应用程序需要的每个模块，然后将所有这些模块打包成一个或多个 *bundle*。

**四个核心概念：**

入口（entry）

输出(output)

loader

插件(plugins)

**(1)入口（entry）**

指示webpack应该使用哪个模块来作为构建其内部以来图的开始，

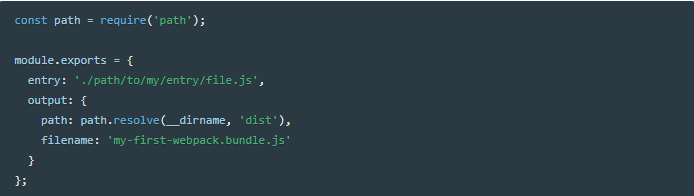
webpack.config.js



**（2）输出(output)**

output 属性告诉 webpack 在哪里输出它所创建的 *bundles*，以及如何命名这些文件，默认值为 ./dist

webpack.config.js



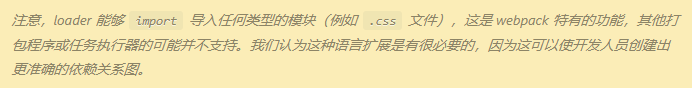
output.filename告诉我们webpack bundle的名称，output.path告诉我们想要bundle生成的路径（就是在哪里生成）。

const path = require('path')这句话是用的node中的path模块。

**（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://webpack.docschina.org/concepts/modules)，然后你就可以利用 webpack 的打包能力，对它们进行处理。

通俗的说，loader就是让webpack不能识别的转换为可以识别的。

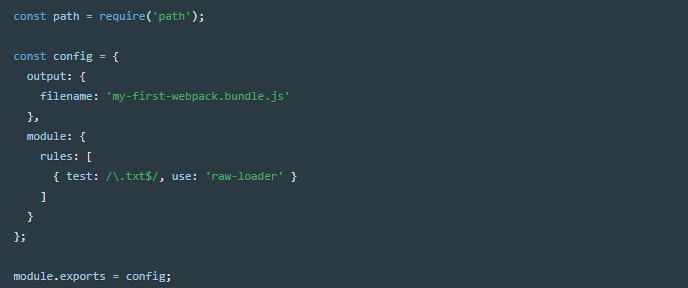


在更高层面，在 webpack 的配置中 loader 有两个目标：

①：test属性，用于表示出应该呗对应的loader进行转换的某个或者某些文件

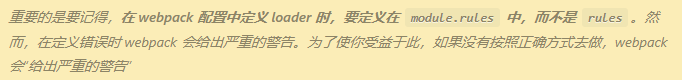
②：use属性，表示进行转换时，应该使用哪个loader。

webpack.config.js



以上配置中，对一个单独的module对象定义了一个rules属性，里面必须包含两个属性：test和use。这段话的意思就是：

webpack编译器，当遇到「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。



**（4）插件(plugins)**

相比于loader，插件被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。插件的功能极其强大，可以用来处理各种各样的任务。

想要用一个插件，要先require它，然后添加到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。

webpack.config.js



**模式:**

通过选择development(开发)和production（生产）之中的一个，来设置mode参数，你可以启用相应模式下的webpack内置的优化。

