**1、学习内容：常用配置（entry、output、loader、plugins……）**

**2、作业**

1）理解entry、output作用及配置，举例说明；

2）了解loader实现原理，实现一个模板解析解析器；

功能：解析 .tpl.html 文件的模板

3）了解plugins实现原理，实现一个自定义插件；

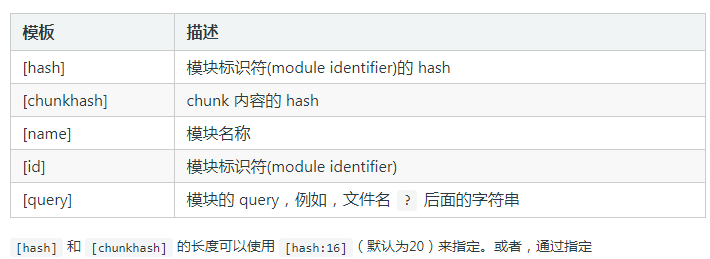
功能：实现一个创建README.md插件

4）区分loader和plugin的差异；

**1）理解entry、output作用及配置，举例说明；**

Entry 入口 ，输入 默认值为: ./src

Output 打包出来的文件路径



Contenthash 内容hash

<https://www.cnblogs.com/giggle/p/9583940.html>

**2) 了解loader实现原理，实现一个模板解析解析器；**

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)，然后你就可以利用 webpack 的打包能力，对它们进行处理。

功能：解析 .tpl.html 文件的模板

3）了解plugins实现原理，实现一个自定义插件；

webpack **插件**是一个具有 [apply](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/apply) 属性的 JavaScript 对象。apply 属性会被 webpack compiler 调用，并且 compiler 对象可在**整个**编译生命周期访问。

4）区分loader和plugin的差异；

<https://www.cnblogs.com/tangjiao/p/10429645.html>

【Loader】：用于对模块源码的转换，loader描述了webpack如何处理非javascript模块，并且在buld中引入这些依赖。loader可以将文件从不同的语言（如TypeScript）转换为JavaScript，或者将内联图像转换为data URL。比如说：CSS-Loader，Style-Loader等。

loader的使用很简单：

在webpack.config.js中指定loader。module.rules可以指定多个loader，对项目中的各个loader有个全局概览。

loader是运行在NodeJS中，可以用options对象进行配置。plugin可以为loader带来更多特性。loader可以进行压缩，打包，语言翻译等等。

loader从模板路径解析，npm install node\_modules。也可以自定义loader，命名XXX-loader。

语言类的处理器loader：CoffeeScript，TypeScript，ESNext（Bable）,Sass,Less,Stylus。任何开发技术栈都可以使用webpack。

【Plugin】：目的在于解决loader无法实现的其他事，从打包优化和压缩，到重新定义环境变量，功能强大到可以用来处理各种各样的任务。webpack提供了很多开箱即用的插件：CommonChunkPlugin主要用于提取第三方库和公共模块，避免首屏加载的bundle文件，或者按需加载的bundle文件体积过大，导致加载时间过长，是一把优化的利器。而在多页面应用中，更是能够为每个页面间的应用程序共享代码创建bundle。

webpack功能强大，难点在于它的配置文件，webpack4默认不需要配置文件，可以通过mode选项为webpack指定了一些默认的配置，mode分为：development/production，默认是production。