## wabpack学习

理解四个核心概念：1.入口，2.输出，3.loader,4.插件

入口：指示webpack应该使用哪个模块，来构建其内部依赖图的开始。进入入口起点后，webpack会找到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，最后输出到称为bundles的文件中，可以通过在webpack配置中配置entry属性，指定一个入口起点，默认值为./src。

出口：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。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。

**Loader**：让webpack能够去处理那些非Javascript文件（webpacker自身只理解Javascript）。

Loader可以将所以类型的文件转换为webpack能够处理的有效模块，然后利用webpack的打包能力，对它们进行处理。本质上，webpack loader将所以类型的文件，转换为应用程序的依赖图可以直接引用的模块。

*注意，loader 能够 import 导入任何类型的模块（例如 .css 文件），这是 webpack 特有的功能，其他打包程序或任务执行器的可能并不支持。我们认为这种语言扩展是有很必要的，因为这可以使开发人员创建出更准确的依赖关系图。*

插件：loader是被用于转换某种类型的模块，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的变量。插件借口功能极其强大，可以用来处理各种各样的任务。

想要使用一个插件，你需要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，则需要通过使用new操作符来创建他的一个实例。

**入口起点 ：1.单个入口语法**

1. **对象语法**
2. **分离应用程序（app）和第三方库（vendor）入口**
3. **多页面应用程序**

**输出 ：1.用法**

1. **多个入口起点**
2. **高级进阶**

**模式：1.用法**

1. **mode:development**
2. **mode:production**

**Loader:1.示例**

1. **使用Loader**
2. **配置[Configuration]**
3. **内联**
4. **CLI**
5. **Loader特性**
6. **解析loader**

Webpack的工作方式：把你的项目当作一个整体，通过一个给定的主文件，Webpack将从这个文件开始找你的项目的所有依赖文件，使用Loaders处理他们，最后打包为一个浏览器可识别的JavaScript文件。