## 为什么需要一款模块加载器？

现有的模块加载器，不能很好适配大型项目（大型单页面应用程序）开发。开发这样一款加载器最大原因，就是为了代码分离以及静态资源模块化无缝接合。

## 开发这一款加载器的目标

* 分离现有依赖树，按需加载
* 高效保证第一次加载
* 静态资源模块化
* 第三方库模块化加载
* 实现加载器几乎所有环节可配置性
* 适配大项目开发

## Webpack特性功能？

### 代码分离

Webpack有两种依赖声明方式：同步与异步。异步方式，将依赖分割成多个节点，然后每个节点形成一个新的文件块。经过优化后的文件块树，会以一个个文件形式分发出去（仅仅打包成一个大文件形式是很低效的，详见）。

### 加载器插件

原生的Webpack只能处理JS文件，使用加载器插件，可以将其他资源转为JS资源。通过这种方式来加载，每一种资源都可以被Webpack看作是一个模块来加载。

## 智能模块解析

Webpack内置一个智能加载模块，可以用于处理几乎所有的第三方库。它甚至可以解析依赖声明的表达式，比如 require("./templates" + name + ".jade")。Webpack会处理最常见的JS模块标准：CommonJS 和 AMD。

## 插件系统

Webpack的最大特点，就是配套了非常丰富的插件系统。大部分内置特性功能都是基于这套插件系统。它可以让你根据需要自定义Webpack，将一般插件作为开源项目发布出去。

## Webpack的配置主要为了这几大项目：

* entry：js入口源文件
* output：生成文件
* module：进行字符串的处理
* resolve：文件路径的指向
* plugins：插件，比loader更强大，能使用更多webpack的api

## 常用Loaders介绍

* 处理样式，转成css，如：less-loader, sass-loader
* 图片处理，如: url-loader, file-loader。两个都必须用上。否则超过大小限制的图片无法生成到目标文件夹中
* 处理js，将es6或更高级的代码转成es5的代码。如： babel-loader，babel-preset-es2015，babel-preset-react
* 将js模块暴露到全局，如果expose-loader

## 常用Plugins介绍

* 代码热替换, HotModuleReplacementPlugin
* 生成html文件，HtmlWebpackPlugin
* 将css成生文件，而非内联，ExtractTextPlugin
* 报错但不退出webpack进程，NoErrorsPlugin
* 代码丑化，UglifyJsPlugin，开发过程中不建议打开
* 多个 html共用一个js文件(chunk)，可用CommonsChunkPlugin
* 清理文件夹，Clean
* 调用模块的别名ProvidePlugin，例如想在js中用$，如果通过webpack加载，需要将$与jQuery对应起来