## 自动化打包工具webpack

引言：随着前端的单页应用（ Webapp 模式）的兴起，那些所见即所得的IDE工具(例如Dreamweaver），就渐渐地退出了前端的主流。一个应用，通常只有一个页面，甚至body里面只有几个div。这时有大量的css和javascript代码需要编写。如何组织他们，就是现在前端所面临和要解决的问题。 一些很好的前端框架(像angularjs,React,Vue)可以帮我们如何合理的组织代码，保持代码的可维护性和扩展性。这在开发阶段是很有用的，可是要把应用发布到线上的时候，需要对代码进行合并压缩，以减小代码体积，和文件数量，人为的对代码进行丑化。于是就有了grunt,gulp,webpack等前端工程化打包工具。

### 1、大致介绍

它可以将许多松散的模块按照依赖和规则打包成符合生产环境部署的前端资源。还可以将按需加载的模块进行代码分隔，等到实际需要的时候再异步加载。通过 loader 的转换，任何形式的资源都可以视作模块，比如 CommonJs 模块、 AMD 模块、 ES6 模块、CSS、图片、 JSON、Coffeescript、 LESS 等。

依赖和规则

webpack

前端资源

打包

* CommonJs 模块：使用require 方法来同步加载（不适用在浏览器端）所要依赖的其他模块，然后通过 exports 或 module.exports 来导出需要暴露的接口。



* AMD 模块：依赖前置，使用回调函数(异步加载） CMD模块相似



### 2、模块的加载和传输

请求次数过多

1、每个文件单独请求

启动速度慢

流量浪费

1. 所有模块打包成一个文件然后只请求一次

初始化过程慢

**总结**：分块传输，按需进行懒加载，在实际用到某些模块的时候再增量更新，才是较为合理的模块加载方案。要实现模块的按需加载，就需要一个对整个代码库中的模块进行静态分析、编译打包的过程。

将依赖树拆分成按需加载的块

初始化加载的耗时尽量少

各种静态资源都可以视作模块

衡量标准

将第三方库整合成模块的能力

可以自定义打包逻辑的能力

适合大项目，无论是单页还是多页的 Web 应用

### Webpack的特点

同步

异步（异步依赖作为分割点，形成一个新的块。在优化了依赖树后，每一个异步区块都作为一个文件被打包）

代码拆分

loader转换器（将任何资源都可以转换为javascript模块）

有一个智能解析器，几乎可以处理任何第三方库

功能丰富的插件系统

快速运行

### webpack操作

安装nodejs -----> 安装npm ------> 安装cnpm

* 全局安装webpack打包工具： cnpm install webpack -g
* 安装webpack依赖：cnpm install webpack --save-dev
* 使用时直接使用webpack命令

例如：webpack entry.js bulid.js