**Webpack介绍及使用**

**目录**

1. 了解webpack
2. 核心概念
3. 常用配置项
4. 常用loader及plugin
5. CLI常用命令
6. 配置文件编写
7. 代码分离
8. 缓存管理

**一、了解webpack**

1. webpack是什么？

webpack 是一个js静态模块打包器。

1. webpack是怎么工作的？

当 webpack 处理应用程序时，它会递归地构建一个依赖关系图，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。

1. webpack支持的模块语法有哪些？

Es6

Commonjs

Amd

1. webpack和gulp的区别？

Webpack可以通过众多loader和plugin完成许多gulp上的功能，但Webpack本质在于模块打包，而gulp的本质在于执行任务；使用webpack-stream包可以实现gulp和webpack的搭配使用

**二、核心概念**

1．入口：指定使用哪个模块来作为构建内部依赖图的开始，随后打包成bundle

2．输出：定义输出生产的bundle的位置及文件名；

3．loader：让webpack可以处理非js模块；loader 可以将所有类型的文件转换为webpack 能够处理的有效模块。

4. 插件：执行的任务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，目的在于完成一些loader无法完成的任务。

**三、常用配置项**

1. mode： ‘development’|| ‘production’ || ‘none’
2. entry: array || obj
3. output: {

path: path.resolve(\_\_dirname, "dist"), // 输出文件的目标路径

filename:’bundle.js’,// 打包后的文件名

publicPath: ‘static’// 输出解析文件目录，url相对于html

}

1. module: {

rules: [

{

test: /\.js$/, // 指定使用此loader的文件

use: ‘babel-loader’,

options: { },

include: /src/, // 匹配项

exclude: /node\_modules/ // 非匹配项

}

]

}

1. plugins: […instance]
2. devServer: {

host: ‘localhost’,

port: ‘8080’,

contentBase: Boolean|| string || array, // 服务器从哪里获取内容

hot：boolean, // 是否开启webpack模块热替换

open: Boolean, // 是否自动打开浏览器

proxy: { path: targetUrl}, // 代理

…others

}

1. resolve: {

alias: {

test: url

},

extensions: [.js, .json]

}

1. devtool: string || false

可选值：

cheap-eval-source-map

eval-source-map

cheap-source-map

source-map

cheap-module-eval-source-map

…others

**四、常用loader及plugin**

1. Loaders：

Css-loader：解析css中资源路径；

Sass-loader：sass转码为css

Postcss-loader：使webpack可以用postcss处理css

Px2rem-loader: px转换为rem

Style-loader：将css以style标签的形式插入到dom中

Babel-loader：转码js以提高兼容性

url-loader：图片转base64

file-loader：将文件输出到输出目录并返回文件路径

html-loader：处理html中资源路径

eslint-loader：使js支持eslint

1. plugins:

html-webpack-plugin：根据模板导出html文件，并注入bundle

DinePlugin: 创建编译时可以进行配置的全局变量。

HashedModuleIdsPlugin: 根据模块的相对路径生成一个四位数的hash作为模块id

extract-text-webpack-plugin：分离css

mini-css-extract-plugin：分离css

uglifyjs-webpack-plugin：压缩js

optimize-css-assets-webpack-plugin：压缩css

clean-webpack-plugin: 清空某个目录

**五、cli常用命令**

--config 使用指定的配置文件

--env.key=value 指定环境变量

--mode=production 指定模式

--progress 打印编译进度的百分比

--display-error-details 展示错误细节

--define 定义环境变量

-w 观察文件变化，变化后重新执行构建流程

--hot 开启模块热替换

--json > stats.json 将编译结果的各种信息输出为json文件

--profile 捕获编译时每个步骤的时间信息

1. **配置文件编写**

**前端常见需求：**

1. Js压缩
2. Css压缩
3. Css提取
4. 小图片转base64
5. Px转rem
6. Css前缀添加
7. Sass转码
8. Babel转码
9. 文件名加hash
10. 文件编码格式转换（常见utf-8与gbk互转）
11. 代码分离
12. Eslint
13. 模块热替换
14. 配置编译过程的全局变量
15. 打包后的代码注入html中

**配置前需要额外了解的相关知识点：**

1. **常用模块变量**

\_\_dirname:当前文件夹路径

1. **常用占位符**

[hash]:模块标识符的hash（后拼接：‘:n’可指定长度）

[chunkhash]: chunk内容的hash（后拼接：‘:n’可指定长度）

[name]:模块名称

[id]: 模块标识符

1. **配置文件使用commonjs编写的，配置类型有下面几种：**

1) 导出一个配置对象

2) 导出一个函数

3) 导出一个promise

**七、代码分离**

该功能可以把代码分离到不同的 bundle 中，然后可以按需加载或并行加载这些文件。代码分离可以用于获取更小的 bundle，以及控制资源加载优先级，如果使用合理，会极大影响加载时间。

代码分离的方式:

1. 手动配置多个入口文件
2. 使用webpack自带的splitChunks功能
3. 使用import（）动态导入

**八、缓存管理**

客户端获取服务器的资源，是比较耗费时间的，但是可以通过合理的配置使请求命中缓存，以降低网络流量，使网站加载速度更快。这就需要采用特定的方法控制采用缓存还是重新加载新资源。