目录

[基础 1](#_Toc530558663)

[概念 2](#_Toc530558664)

[chunk 2](#_Toc530558665)

[entry 2](#_Toc530558666)

[output 2](#_Toc530558667)

[module 3](#_Toc530558668)

[loader规则 3](#_Toc530558669)

[rule 3](#_Toc530558670)

[css 3](#_Toc530558671)

[postcss-loader 3](#_Toc530558672)

[plugins 4](#_Toc530558673)

[webpack4 4](#_Toc530558674)

[默认配置 4](#_Toc530558675)

[mode 4](#_Toc530558676)

[更新 4](#_Toc530558677)

[webpack-dev-server 4](#_Toc530558678)

[调用 5](#_Toc530558679)

[webpack-dev-server命令行 5](#_Toc530558680)

[nodeapi 5](#_Toc530558681)

process.env.NODE\_ENV,这里的process.env.NODE\_ENV不是node中的环境变量,而是webpack.DefinePlugin中定义的全局变量,允许你根据不同的环境执行不同的代码.

# 基础

最好用\_\_dirname或者process.cwd控制目录，因为有些配置的路径是相对工作目录，比如插件，entry等。有些又是相对配置文件所在的目录，比如自己引入的文件。

## 概念

### chunk

webpack当中配置的入口文件（entry）是chunk，可以理解为entry chunk

入口文件以及它的依赖文件通过code split（代码分割）出来的也是chunk，可以理解为children chunk

通过CommonsChunkPlugin创建出来的文件也是chunk，可以理解为commons chunk

## entry

数组可以用来引入一些前置文件：比如polyfill。

相对目录是当前工作目录(调用命令的目录)，即使配置文件是在子目录里。（所以一般用\_\_dirname确定文件所在目录）

## output

必须有filename

path是绝对路径

chunkFilename和filename同时设置时，结果为 chunkFilename，filename无效。

devtoolModuleFilenameTemplate用于调试时资源映射。默认只是js文件名（可能重名），可以把路径放进字符串里。

devtoolModuleFilenameTemplate: info =>

path.resolve(info.absoluteResourcePath).replace(/\\/g, '/'),

## module

### loader规则

！？

### rule

Rule.test 是 Rule.resource.test 的简写。

use必须跟一个数组。字符串的情况可以直接用loader。也可以use:['style-loader']。

test: RegExp | string | function | [(recursive)] | object 字符串必须是一个绝对路径。

## css

### postcss-loader

options: {

// Necessary for external CSS imports to work

// https://github.com/facebookincubator/create-react-app/issues/2677

ident: 'postcss',

plugins: () => [

require('postcss-flexbugs-fixes'),

autoprefixer({

browsers: [

'>1%',

'last 4 versions',

'Firefox ESR',

'not ie < 9', // React doesn't support IE8 anyway

],

flexbox: 'no-2009',

}),

],

},

## 功能

## 热更新

live reload: 保存后自动刷新浏览器。

hmr: 无刷新更新代码

以chunk为单位进行更新。

# plugins

一般需要装： 生成html的，注入变量的，拆分chunk的

生产环境： 压缩，分离css，

# webpack4

## 默认配置

entry: 'src/index.js'

output: dist/main.js

## mode

## 更新

SplitChunkPlugin,取代CommonsChunkPlugin

# webpack-dev-server

调用这个命令时，也会查找webpack.config.js命令。

## 调用

### webpack-dev-server命令行

执行命令行时，会读webpack配置文件的devServer配置。

3.x增加了一个log参数。

底层通过express实现。

### nodeapi

如果你通过 Node.js API 来使用 dev-server， devServer 中的选项将被忽略。将选项作为第二个参数传入： new WebpackDevServer(compiler, {...})。

const compiler = webpack(webpackConfig)

const serverConfig = {

contentBase: config.distPath,

compress: true,

port: 9000,

hot: true,

}

const server = new webpackDevServer(compiler, serverConfig);

server.listen(9000);