# 环境配置管理

## config-lite

更具根目录下的config文件夹中的文件 ，对应环境变量 NOOE\_ENV, 获取匹配文件中的配置（覆盖默认配置default.js中的配置）

# 命令行

## shelljs

js调用shell命令

用法：

直接在全局引用全局变量

require('shelljs/global')

rm('-rf', assetsPath)

mkdir('-p', assetsPath)

cp('-R', 'static/\*', assetsPath)

## ora （命令行加载动画）

const ora = require('ora');

const spinner = ora('Loading unicorns').start();

setTimeout(() => {

spinner.color = 'yellow';

spinner.text = 'Loading rainbows';

}, 1000);

## rimraf

由于window不支持rm命令，使用这个模块代替rm命令

# 依赖管理

## lerna

由于依赖的复杂关系，一旦一个被依赖的包更新了，这个包的版本更新了，所有依赖这个包的版本也因该更新。这就需要这个了，自动更新所有依赖。

# 压缩

## Content-Encoding 对应的主体内容压缩

const CompressionPlugin = require("compression-webpack-plugin")

module.exports = {

plugins: [

new CompressionPlugin(...options)

]}

# 代理中间件

## http-proxy-middleware

服务器中间层代理请求，可以

switch(process.env.NODE\_ENV){

case 'local': var proxypath = 'http://localhost:8001'; break;

case 'online': var proxypath = 'http://cangdu.org:8001'; break;

default: var proxypath = config.dev.proxypath;

}

var options = {

target: proxypath,

changeOrigin: true,

}

if (context.length) {

server.use(proxyMiddleware(context, options))

}

# bable

babel-loader

webpack babel loader

babel-plugin-transform-object-rest-spread

babel实现对象解构插件

var obj1 = {

name: "james",

age: 27

}

var obj2 = {...obj1};

# webpack

babel-loader

webpack-merge

合并配置