**build|build.js npm run build**

require('./check-versions')()

process.env.NODE\_ENV = 'production'

var ora = require('ora')

var rm = require('rimraf')

var path = require('path')

var chalk = require('chalk')

var webpack = require('webpack')

var config = require('../config')

var webpackConfig = require('./webpack.build.conf')

var spinner = ora('building for production...')

spinner.start()

rm(config.build.assetsRoot, err => {

if (err) throw err

webpack(webpackConfig, function (err, stats) {

spinner.stop()

if (err) throw err

process.stdout.write(stats.toString({

colors: true,

modules: false,

children: false,

chunks: false,

chunkModules: false

}) + '\n\n')

console.log(chalk.cyan(' Build complete.\n'))

console.log(chalk.yellow(

' Tip: built files are meant to be served over an HTTP server.\n' +

' Opening index.html over file:// won\'t work.\n'

))

})

})

**check-versions.js**

计算 versionRequirements ([]) ，warnings([])

在package.json中配置好所需要的引擎node,npm的版本；

Requires,引入package.json文件；

定义空数组versionRequirements 记录package.json中所需要的引擎版本；

1. **exec**('npm --version')

function **exec**(cmd) {

return require('child\_process').execSync(cmd).toString().trim()

}

1. semver.clean(process.version)

定义数组warnings跟踪版本不匹配问题

Warnings.length 长度大于0，控制台提示更新

var chalk = require('chalk')

var semver = require('semver')

var **packageConfig** = require('../package.json')

var shell = require('shelljs')

function **exec**(cmd) {

return require('child\_process').execSync(cmd).toString().trim()

}

var versionRequirements = [

{

name: 'node',

currentVersion: semver.clean(process.version),

versionRequirement: **packageConfig**.engines.node

},

]

if (shell.which('npm')) {

versionRequirements.push({

name: 'npm',

currentVersion: **exec**('npm --version'),

versionRequirement: **packageConfig**.engines.npm

})

}

module.exports = function () {

var warnings = []

for (var i = 0; i < versionRequirements.length; i++) {

var mod = versionRequirements[i]

if (!semver.satisfies(mod.currentVersion, mod.versionRequirement)) {

warnings.push(mod.name + ': ' +

chalk.red(mod.currentVersion) + ' should be ' +

chalk.green(mod.versionRequirement)

)

}

}

if (warnings.length) {

console.log('')

console.log(chalk.yellow('To use this template, you must update following to modules:'))

console.log()

for (var i = 0; i < warnings.length; i++) {

var warning = warnings[i]

console.log(' ' + warning)

}

console.log()

process.exit(1)

}

}

**npm run dev的实现过程线路**

1. 进入package.json 配置资源命令行执行node执行build/dev-server.js

"dev": "node --max\_old\_space\_size=4096 build/dev-server.js",

1. 进入dev-server.js

dev-server.js

|--require('./check-versions')()

|--if (!process.env.NODE\_ENV) {}

|--var config = require('../config')

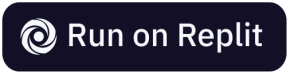
|--require Opn，path，express，webpack，proxyMiddleware

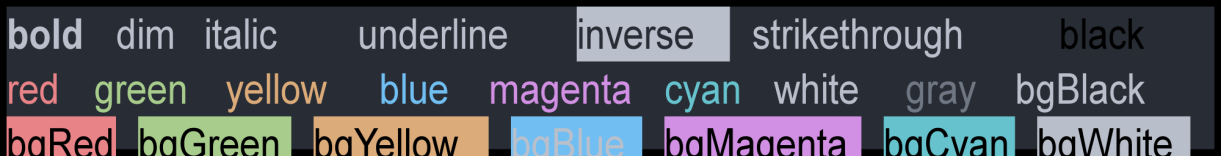
|--var webpackConfig = require('./webpack.dev.conf')

chalk

chalk <https://www.npmjs.com/package/chalk>

Terminal string styling done right

**[IMG_256](https://codecov.io/gh/chalk/chalk)** **[IMG_257](https://www.npmjs.com/package/chalk?activeTab=dependents)** **[IMG_258](https://www.npmjs.com/package/chalk)** **[](https://repl.it/github/chalk/chalk)** **[IMG_260](https://stakes.social/0x44d871aebF0126Bf646753E2C976Aa7e68A66c15)**

**[](https://camo.githubusercontent.com/6b0dc65e5c1d4f1f53aac072b8bb0ef92f7e9cd513bde35cdf609e10ab0d3226/68747470733a2f2f63646e2e6a7364656c6976722e6e65742f67682f6368616c6b2f616e73692d7374796c657340383236313639376339356266333462366337373637653263626539393431613835316435393338352f73637265656e73686f742e737667)**

### **Install**

npm install chalk

****IMPORTANT:**** Chalk 5 is ESM. If you want to use Chalk with TypeScript or a build tool, you will probably want to use Chalk 4 for now. **[Read more.](https://github.com/chalk/chalk/releases/tag/v5.0.0)**

**Usage**

import chalk from 'chalk';

console.log(chalk.blue('Hello world!'));

Chalk comes with an easy to use composable API where you just **chain and nest(**链轮链窝**)** the styles you want.

import chalk from 'chalk';

const log = console.log;

// Combine styled and normal stringslog(chalk.blue('Hello') + ' World' + chalk.red('!'));

// Compose multiple styles using the chainable APIlog(chalk.blue.bgRed.bold('Hello world!'));

// Pass in multiple argumentslog(chalk.blue('Hello', 'World!', 'Foo', 'bar', 'biz', 'baz'));

// Nest styleslog(chalk.red('Hello', chalk.underline.bgBlue('world') + '!'));

// Nest styles of the same type even (color, underline, background)log(chalk.green(

'I am a green line ' +

chalk.blue.underline.bold('with a blue substring') +

' that becomes green again!'));

// ES2015 template literallog(`CPU: ${chalk.red('90%')}RAM: ${chalk.green('40%')}DISK: ${chalk.yellow('70%')}`);

// Use RGB colors in terminal emulators that support it.log(chalk.rgb(123, 45, 67).underline('Underlined reddish color'));log(chalk.hex('#DEADED').bold('Bold gray!'));

Easily define your own themes:

import chalk from 'chalk';

const error = chalk.bold.red;const warning = chalk.hex('#FFA500'); // Orange color

console.log(error('Error!'));console.log(warning('Warning!'));

Take advantage of console.log **[string substitution](https://nodejs.org/docs/latest/api/console.html" \l "console_console_log_data_args)**:

import chalk from 'chalk';

const name = 'Sindre';console.log(chalk.green('Hello %s'), name);//=> 'Hello Sindre'

**Semver**

As a node module:

const semver = require('semver')

semver.valid('1.2.3') // '1.2.3'

semver.valid('a.b.c') // null

semver.clean(' =v1.2.3 ') // '1.2.3'

semver.satisfies('1.2.3', '1.x || >=2.5.0 || 5.0.0 - 7.2.3')

// true

semver.gt('1.2.3', '9.8.7') // false

semver.lt('1.2.3', '9.8.7') // true

semver.minVersion('>=1.0.0') // '1.0.0'

semver.valid(semver.coerce('v2')) // '2.0.0'

semver.valid(semver.coerce('42.6.7.9.3-alpha')) // '42.6.7'

You can also just load the module for the function that you care about, if you'd like to minimize your footprint.

ShellJS

npm官网中的shelljs <https://www.npmjs.com/package/shelljs>

ShellJS is a portable ****(Windows/Linux/OS X)**** implementation(**实施，执行**) of Unix shell commands on top of the Node.js API. You can use it to eliminate your shell script's dependency on Unix while still keeping its familiar and powerful commands. You can also install it globally so you can run it from outside Node projects - say goodbye to those gnarly(**粗糙的**) Bash scripts!

ShellJS is proudly tested on every node release since v4!

* The project is **[unit-tested](http://travis-ci.org/shelljs/shelljs)** and battle-tested(**久经沙场的**) in projects like:
* [Firebug](http://getfirebug.com/) - Firefox's infamous(**声名狼藉的,不名誉的**) debugger
* [JSHint](http://jshint.com/) & [ESLint](http://eslint.org/) - popular JavaScript linters
* [Zepto](http://zeptojs.com/) - jQuery-compatible JavaScript library for modern browsers
* [Yeoman](http://yeoman.io/) - Web application stack and development tool
* [Deployd.com](http://deployd.com/) - Open source PaaS for quick API backend generation(后端生成)
* And [many more](https://npmjs.org/browse/depended/shelljs).

If you have feedback, suggestions, or need help, feel free to post in our **[issue tracker](https://github.com/shelljs/shelljs/issues)(**问题跟踪器**)**.

Think ShellJS is cool? Check out some related projects in our **[Wiki page](https://github.com/shelljs/shelljs/wiki)**!

Upgrading from an older version? Check out our **[breaking changes](https://github.com/shelljs/shelljs/wiki/Breaking-Changes)(**不兼容的修改,突发的变化**)** page to see what changes to watch out for while upgrading.

**child\_process**

1.0.2 • Public • Published 6 years ago

* **[Readme](https://www.npmjs.com/package/child_process?activeTab=readme)**
* **[Explore BETA](https://www.npmjs.com/package/child_process?activeTab=explore)**
* **[0 Dependencies](https://www.npmjs.com/package/child_process?activeTab=dependencies)**
* **[2,068 Dependents](https://www.npmjs.com/package/child_process?activeTab=dependents)**
* **[1 Versions](https://www.npmjs.com/package/child_process?activeTab=versions)**

**Security holding package(**安全持有包?**)**

This package name is not currently in use, but was formerly occupied by another package. To avoid malicious use, npm is hanging on to the package name, but loosely, and we'll probably give it to you if you want it.

You may adopt this package by contacting **[support@npmjs.com](mailto:support@npmjs.com)** and requesting the name.

**Keywords**

none

# **process**

require('process'); just like any other module.

Works in node.js and browsers via the browser.js shim provided with the module.

## **browser implementation**

The goal of this module is not to be a full-fledged alternative to the builtin process module. This module mostly exists to provide the nextTick functionality and little more. We keep this module lean because it will often be included by default by tools like browserify when it detects a module has used the process global.

It also exposes a "browser" member (i.e. process.browser) which is true in this implementation but undefined in node. This can be used in isomorphic code that adjusts it's behavior depending on which environment it's running in.

If you are looking to provide other process methods, I suggest you monkey patch them onto the process global in your app. A list of user created patches is below.

* [hrtime](https://github.com/kumavis/browser-process-hrtime)
* [stdout](https://github.com/kumavis/browser-stdout)

## **package manager notes**

If you are writing a bundler to package modules for client side use, make sure you use the browser field hint in package.json.

See **<https://gist.github.com/4339901>** for details.

The **[browserify](https://github.com/substack/node-browserify)** module will properly handle this field when bundling your files.

# **http-proxy-middleware**

**[IMG_256](https://github.com/chimurai/http-proxy-middleware/actions?query=branch:master)** **[IMG_257](https://coveralls.io/r/chimurai/http-proxy-middleware)** **[IMG_258](https://snyk.io/test/npm/http-proxy-middleware)** **[IMG_259](https://www.npmjs.com/package/http-proxy-middleware)**

Node.js proxying made simple. Configure proxy middleware with ease for **[connect](https://github.com/senchalabs/connect)**, **[express](https://github.com/strongloop/express)**, **[browser-sync](https://github.com/BrowserSync/browser-sync)** and **[many more](https://www.npmjs.com/package/http-proxy-middleware" \l "compatible-servers)**.

Powered by the popular Nodejitsu **[http-proxy](https://github.com/nodejitsu/node-http-proxy)**. **[IMG_260](https://github.com/nodejitsu/node-http-proxy)**

## **⚠️ Note**

This page is showing documentation for version v2.x.x (**[release notes](https://github.com/chimurai/http-proxy-middleware/releases)**)

If you're looking for v0.x documentation. Go to: **[https://github.com/chimurai/http-proxy-middleware/tree/v0.21.0#readme](https://github.com/chimurai/http-proxy-middleware/tree/v0.21.0" \l "readme)**

## **TL;DR**

Proxy /api requests to http://www.example.org

// javascript

const express = require('express');

const { createProxyMiddleware } = require('http-proxy-middleware');

const app = express();

app.use('/api', createProxyMiddleware({ target: 'http://www.example.org', changeOrigin: true }));

app.listen(3000);

// http://localhost:3000/api/foo/bar -> http://www.example.org/api/foo/bar

// typescript

import \* as express from 'express';import { createProxyMiddleware, Filter, Options, RequestHandler } from 'http-proxy-middleware';

const app = express();

app.use('/api', createProxyMiddleware({ target: 'http://www.example.org', changeOrigin: true }));

app.listen(3000);

// http://localhost:3000/api/foo/bar -> http://www.example.org/api/foo/bar

All http-proxy **[options](https://github.com/nodejitsu/node-http-proxy" \l "options)** can be used, along with some extra http-proxy-middleware **[options](https://www.npmjs.com/package/http-proxy-middleware" \l "options)**.

****Tip:**** Set the option changeOrigin to true for **[name-based virtual hosted sites](http://en.wikipedia.org/wiki/Virtual_hosting" \l "Name-based)**.



# **webpack-dev-middleware**

<https://www.npmjs.com/package/webpack-dev-middleware>

An express-style development middleware for use with **[webpack](https://webpack.js.org/)** bundles and allows for serving of the files emitted from webpack. This should be used for ****development only****.

Some of the benefits of using this middleware include:

* No files are written to disk, rather it handles files in memory
* If files changed in watch mode, the middleware delays requests until compiling has completed.
* Supports hot module reload (HMR).

## **Getting Started**

First thing's first, install the module:

npm install webpack-dev-middleware --save-dev

Note: We do not recommend installing this module globally.

## **Usage**

const webpack = require("webpack");

const middleware = require("webpack-dev-middleware");

const compiler = webpack({ // webpack options});

const express = require("express");

const app = express();

app.use( middleware(compiler, { // webpack-dev-middleware options }));

app.listen(3000, () => console.log("Example app listening on port 3000!"));

See **[below](https://www.npmjs.com/package/webpack-dev-middleware" \l "other-servers)** for an example of use with fastify.

**webpack.tool.js**

var path = require('path')

function excludeFile() {

if(process.env.NODE\_ENV === 'production') {

return ["demo", "integrate", "../static/external/gojs"];

}

return [];

}

module.exports = {

build: {

includePages: [],

excludeDirs: [].concat(excludeFile()),

favicon: path.resolve(\_\_dirname, '../static/images/favicon.ico')

}

}

If no path segments path segments(路径片段) are passed, path.resolve() will return **the absolute path** of the current working directory.

path.resolve('/foo/bar', './baz');// Returns: '/foo/bar/baz'

path.resolve('/foo/bar', '/tmp/file/');// Returns: '/tmp/file'

path.resolve('wwwroot', 'static\_files/png/', '../gif/image.gif');

// If the current working directory is /home/myself/node,

// this returns '/home/myself/node/wwwroot/static\_files/gif/image.gif'

the current working directory

For example, on POSIX:

If **a zero-length string** is passed as from or to, **the current working directory** will be used instead of the zero-length strings.

**a zero-length string 是指from与to层级一样深吗**

path.relative('/data/orandea/test/aaa', '/data/orandea/impl/bbb');

// Returns: '../../impl/bbb'

On Windows:

path.relative('C:\\orandea\\test\\aaa', 'C:\\orandea\\impl\\bbb');

// Returns: '..\\..\\impl\\bbb'

// 获取入口页面实体

function getPageEnties (pageFiles) {

let pagesAttr = []

pageFiles.map(pageFile => {

// key为home/login、system/user/selectuser等

let key = path.dirname**[[1]](#endnote-0)**(

path.relative(

path.resolve(config.build.srcPath, 'pages'), pageFile)

).replace(/\\/g, '/')

if (!srcConfig.build.includePages.length || srcConfig.build.includePages.indexOf(key) !== -1) {

let sysName = key.split('/')[0]

if (srcConfig.build.excludeDirs.indexOf(sysName) === -1) {

let js = path.resolve(path.dirname(pageFile), 'main.js')

let html = require(pageFile)

pagesAttr.push({

key,

js,

html

})

}

}

})

return pagesAttr

}

dev-client.js-配合dev-server.js监听html文件改动也能够触发自动刷新

// 引入 webpack-hot-middleware/client

var hotClient = require('webpack-hot-middleware/client');

// 订阅事件，当 event.action === 'reload' 时执行页面刷新

hotClient.subscribe(function(event){

if(event.action === 'reload'){

window.location.reload();

}

})

**webpack.build.conf.js 实现 chunkhash 的稳定化**

plugins: [

// http://vuejs.github.io/vue-loader/en/workflow/production.html

new webpack.DefinePlugin({

'process.env': env

}),

// extract css into its own file

new ExtractTextPlugin({

filename: utils.assetsPath('css/[name].[contenthash:8].css')

}),

// 对模块路径进行MD5摘要，轻松地实现 chunkhash 的稳定化

new webpack.HashedModuleIdsPlugin(),

// copy custom static assets

new CopyWebpackPlugin([

{

from: path.resolve(\_\_dirname, '../src/static/external'),

to: config.build.assetsSubDirectory,

ignore: ['.\*']

},

// {

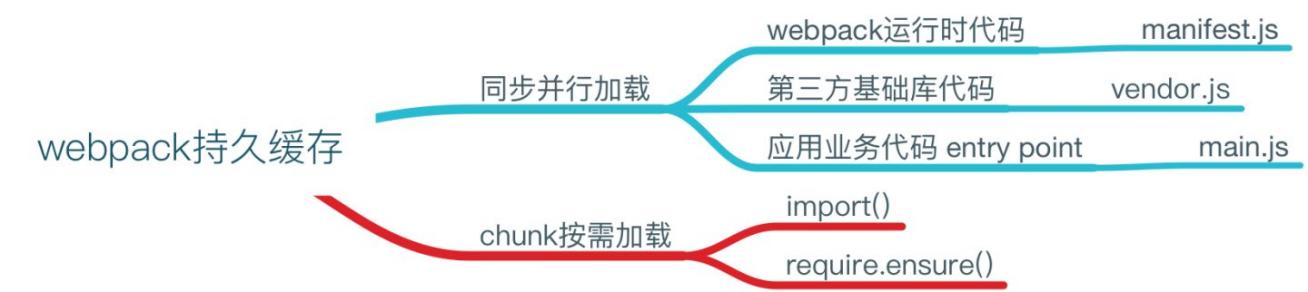
// from: path.resolve(\_\_dirname, `../src/static/config.${env.NODE\_ENV.replace(/"/g, "")}.js`),

// to: config.build.assetsRoot + "/config.js"

// }

])

]



# webpack持久化缓存优化策略

*作者：淼森*

## 问题背景

webpack 是目前最流行的前端依赖打包工具，日常项目开发当中利用webpack在编译打包生成静态资源文件时，通过相应的配置以及引入某些插件，我们给静态资源js、css等添加chunkhash，从而可以根据代码文件的变动，自动生成相应的文件chunkhash，最终可以充分利用浏览器的静态资源缓存能力。

通常在我们会使用CommonsChunkPlugin来将所有依赖的第三方包打包到一个名为vender的chunk中。与此同时，为了避免每次更改项目代码时导致vender chunk的chunkHash改变，我们还会单独生成一个manifest chunk。

但在实际项目当中经常遇到一些chunkhash失效的问题，即只在业务模块中新增了一个新的依赖模块，重新编译打包，就会发现导致抽离出的**公共基础库模块** vendor.js的chunkhash发生变化，而这绝对不是我们所期望的情况，因为从模块依赖关系上来看业务应用模块中的新增模块与vendor.js并没有直接的依赖关系。

而这就要从webpack的持久化缓存说起。。。

## 持久化缓存

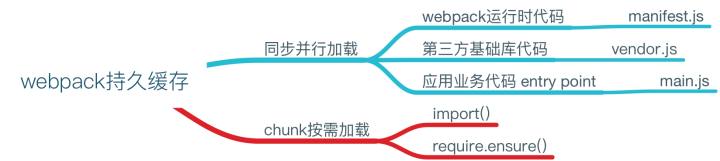
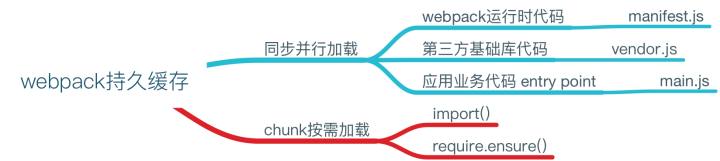
持久缓存，也就是说在相关代码内容没有变化的时候，尽可能的使浏览器利用缓存而不是发送静态资源请求。

常用的方法：

* 将很少变动代码提取到单独文件，使得这部分代码几乎总是被缓存，从而提高整体静态资源缓存利用率
* 使用[数据摘要要算法](https://link.zhihu.com/?target=https://baike.baidu.com/item/%E6%91%98%E8%A6%81%E7%AE%97%E6%B3%95" \t "_blank)对文件求摘要信息，摘要信息与文件内容一一对应，将摘要信息作为文件名（的一部分）或者版本号，同时设置最大缓存失效期

## webpack实现持久化缓存

## 代码分离 code splitting



## webpack基础配置

* 提取**webpack运行库**和**第三方基础库代码**

// 将基础库代码提取到单独文件中

new webpack.optimize.CommonsChunkPlugin({

name: 'vendor',

minChunks: function(module, count){

return (

module.resource &&

/\.js$/.test(module.resource) &&

module.resource.indexOf(

path.join(\_\_dirname, '../node\_modules')

) === 0

)

}

}),

new webpack.optimize.CommonsChunkPlugin({

name: 'manifest',

chunks: ['vendor']

}),

* 应用入口**entry chunk**提取

module.exports = {

entry: {

main: './src/entry/newPage.jsx',

},

output: {

path: './dist',

filename: '[name].js'

},

* css文件提取

module: {

rules: [{

test: /\.css?$/,

use: ExtractTextPlugin.extract({

fallback: 'style-loader',

use: 'css-loader'

})

}]

}

## webpack文件hash

文件的hash指纹通常作为前端静态资源实现持久化缓存的方案之一，Webpack提供了两个配置项可供使用：hash和chunkhash。那么两者有何区别呢？

## 区分 **[hash]** 和 **[chunkhash]**

* hash代表的是compilation的hash值
  + [compilation](https://link.zhihu.com/?target=https://webpack.js.org/development/how-to-write-a-plugin/%23compiler-and-compilation" \t "_blank)对象代表某个版本的资源对应的编译进程，任何一个文件改动后就会被重新创建，然后webpack计算新的compilation的hash值
  + **所有的文件名都会使用相同的hash指纹**
* chunkhash代表的是chunk的hash值
  + 根据具体chunk模块文件的内容计算所得的hash值，所以某个文件的改动只会影响它本身的hash指纹，不会影响其他文件

## webpack配置使用 [chunkhash]

* output文件名js chunkhash配置

output: {

path: './dist/',

//为了减少提交文件数，采用 ?\_v=[chunkhash:8]的文件命名规则

filename: utils.assetsPath('js/[name].js?\_v=[chunkhash:8]'),

chunkFilename: utils.assetsPath('js/[id].chunk.js?\_v=[chunkhash:8]'),

// filename: utils.assetsPath('js/[name].[chunkhash:8].js'),

// chunkFilename: utils.assetsPath('js/[id].[chunkhash:8].js')

}

filename （[官方文档](https://link.zhihu.com/?target=https://webpack.js.org/configuration/output/%23output-filename" \t "_blank)）对应于entry chunk生成的文件名。单个入口filename 会是一个静态名称，多个入口起点使用[name]来使用对应的配置名。

chunkFilename ([官方文档](https://link.zhihu.com/?target=https://webpack.js.org/configuration/output/%23output-chunkfilename" \t "_blank))对应于非入口(non-entry) chunk 文件，需要被单独打包出来的文件命名配置。比如按需加载（异步）模块的时候，这样的文件是没有被列在entry中的使用CommonJS的方式异步加载模块

* 提取css文件chunkhash配置

const ExtractTextPlugin = require('extract-text-webpack-plugin');

new ExtractTextPlugin({

filename: utils.assetsPath('css/[name].css?\_v=[contenthash:8]')

}),

## 不稳定的chunkhash

不过，只是计算 chunk MD5 摘要并修改 chunk 资源文件名是不够的。**Chunk 的生成还涉及到依赖解析和模块 ID 分配，这是无法稳定实质上没有变化的 chunk 文件的 chunkhash 变动问题的本源**

对于复杂项目的构建，由于模块间互相依赖，可能只改动了一个小模块，但在构建后，会发现所有与之直接或间接相关的 chunk 及其 chunkhash 都被更新了。

## 模块依赖导致chunk更新示例

以赤兔平台的代码为例：

* app.jsx 是平台的业务模块入口文件（entry chunk）所依赖的模块

import React from 'react';

import {connect} from 'react-redux';

import EditModule from './EditModule';

import BaseComponent from '../components/BaseComponent';

import PreviewPage from './Layout/PreviewPage';

import EditPage from './Layout/EditPage';

import {editMode, MODAL\_TYPE, RIGHT\_LAYOUT} from '../common/util';

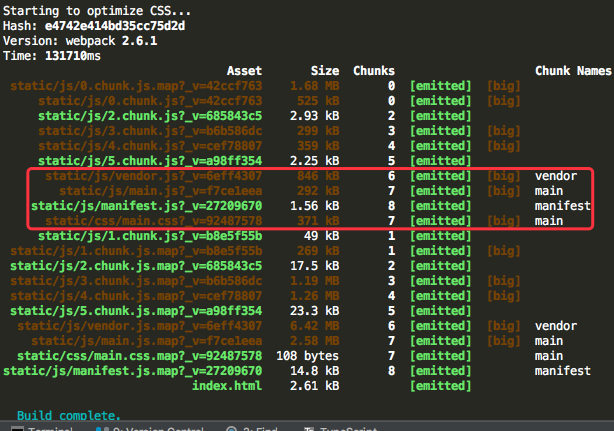
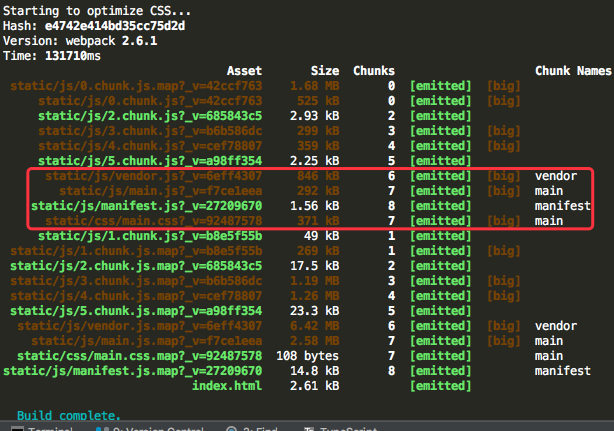
import '../common/lib';

import './App.scss';

class App extends BaseComponent {

}

* 第一次编译构建：



* 增加一个新的依赖模块 ../common/util2后重新构建编译：

import {a} from '../common/util2';

import React from 'react';

import {connect} from 'react-redux';

import EditModule from './EditModule';

import BaseComponent from '../components/BaseComponent';

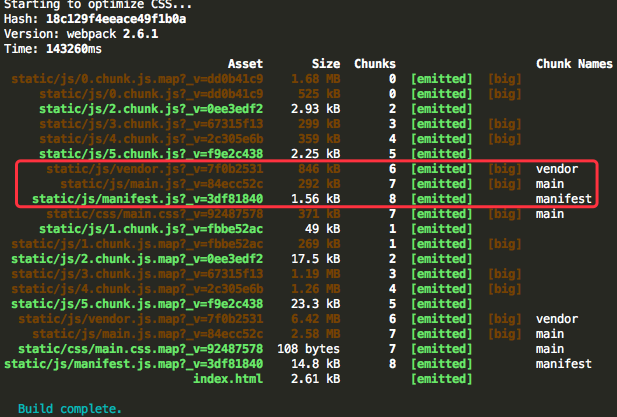
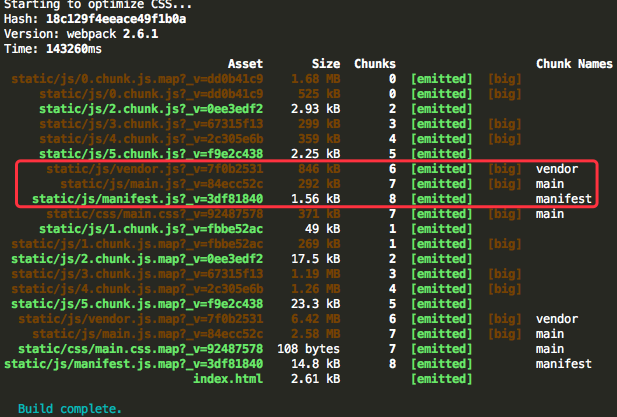
import PreviewPage from './Layout/PreviewPage';

import EditPage from './Layout/EditPage';

import {editMode, MODAL\_TYPE, RIGHT\_LAYOUT} from '../common/util';

import '../common/lib';

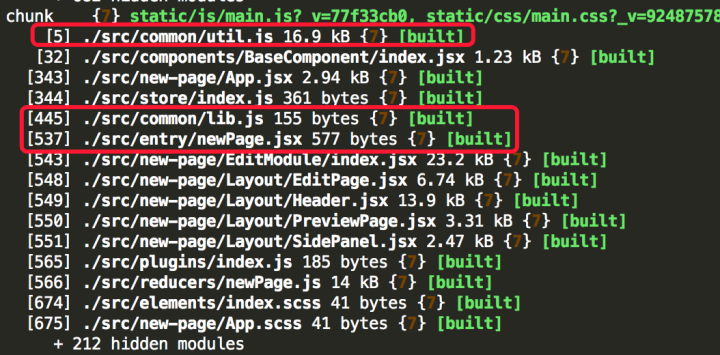
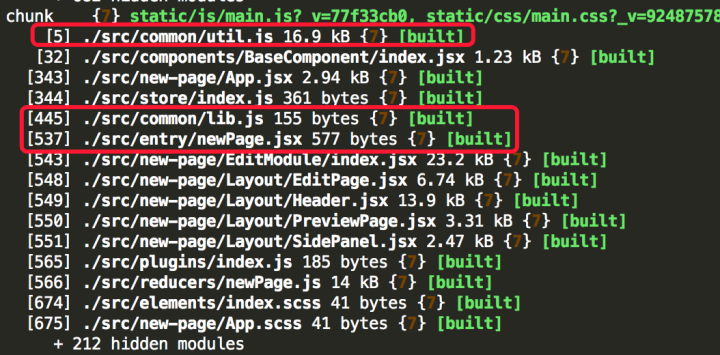
import './App.scss';



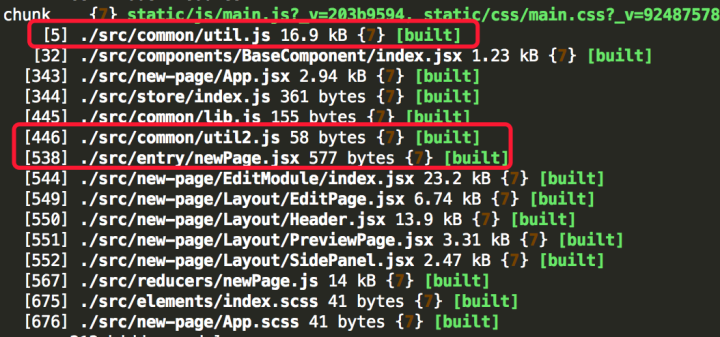
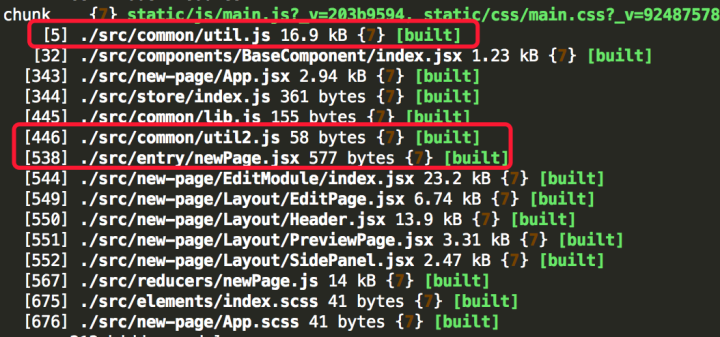
我们并没有修改依赖的第三方包，但是vender chunk的chunkHash也发生了更改。

导致这个结果的原因在于，由于引入了一个新模块，使得打包过程中部分模块的模块ID发生了改变。而模块ID的改变，直接导致了包含这些模块的chunk内容改变，进而导致chunkHash的改变。

* 引入**util2.js**前



* 引入 **util2.js**后



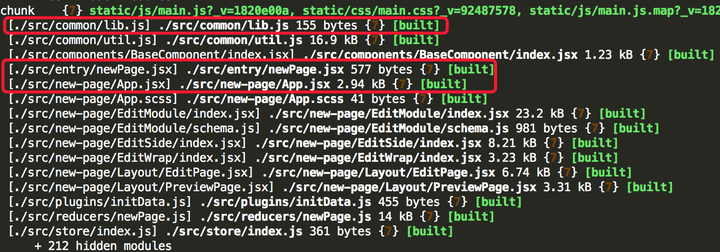
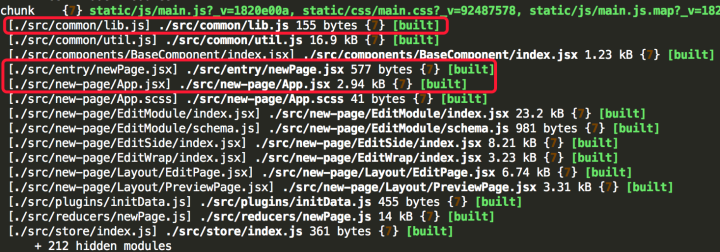
## 稳定的模块chunkhash

前面我们既然找到了问题的原因，那么解决方案也就很明了了。那就是找到一种和顺序无关的模块ID命名方式。最容易想到的就是基于文件名或者文件内容的哈希值这两种方案了。其实也就是今天要说的webpack内置的两个pluginNamedModulesPlugin与HashedModuleIdsPlugin的功能。

## 启用NamedModulesPlugin

这个模块可以将依赖模块的**正整数 ID** 替换为**相对路径**（如：将 4 替换为 ./node\_modules/es6-promise/dist/es6-promise.js）

* 模块相对路径



* **优点**

1. **开发模式**，可以让 **webpack-dev-server** 和 **HMR** 进行热更新时在控制台输出模块名字而不是纯数字
2. **生产构建环境**，可以避免因修改内容导致的 ID 变化，从而实现持久化缓存

* **缺点**

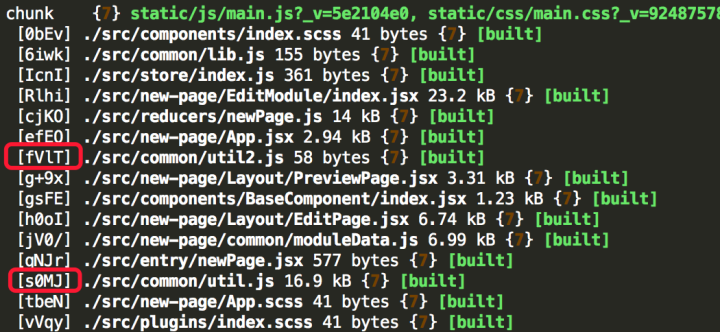
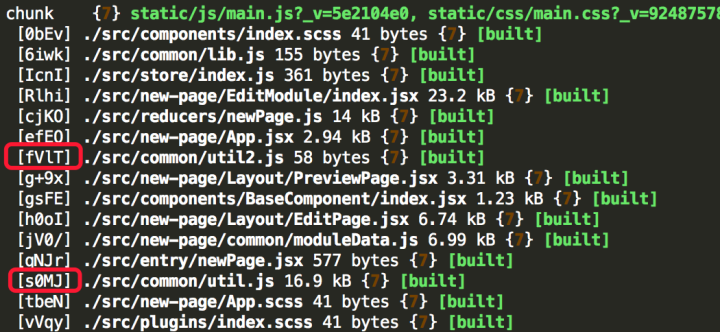
1. 递增 ID 替换为模块相对路径，构建出来的 chunk 会充满各种路径，使文件增大
2. 模块（npm 和自己的模块）路径会泄露，可能导致安全问题

* [参考示例](https://link.zhihu.com/?target=https://gw.alipayobjects.com/zos/rmsportal/bEkONHBGqynQSOnBFNjY.png" \t "_blank)

## 启用 HashedModuleIdsPlugin

NamedModulesPlugin 的进阶模块，它在其基础上**对模块路径进行 MD5 摘要**，不仅可以实现持久化缓存，同时还避免了它引起的两个问题（文件增大，路径泄露）。因此可以轻松地实现 chunkhash 的稳定化！

* 模块路径md5化



* [官方文档](https://link.zhihu.com/?target=https://webpack.js.org/plugins/hashed-module-ids-plugin/" \t "_blank)
* 配置方法：

plugins: [

// 将基础库代码提取到单独文件中

new webpack.optimize.CommonsChunkPlugin({

name: 'vendor',

minChunks: function (module, count) {

// any required modules inside node\_modules are extracted to vendor

return (

module.resource &&

/\.js$/.test(module.resource) &&

module.resource.indexOf(

path.join(\_\_dirname, '../node\_modules')

) === 0

)

}

}),

new webpack.optimize.CommonsChunkPlugin({

name: 'manifest',

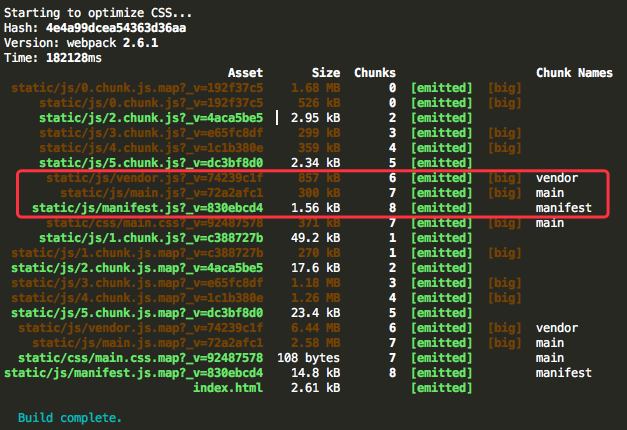
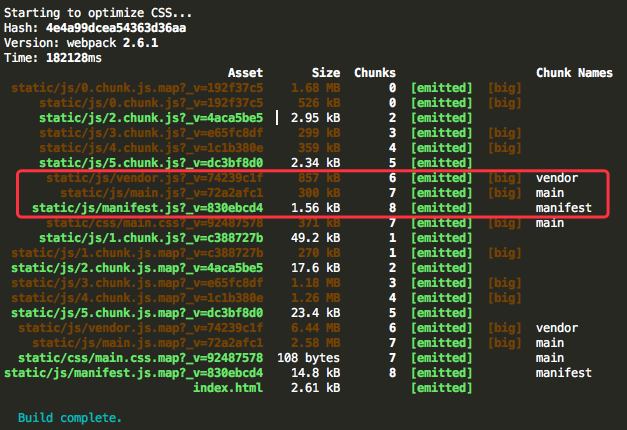
chunks: ['vendor']

}),

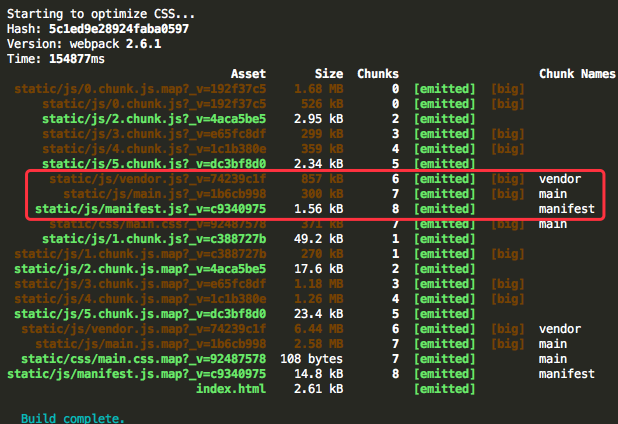
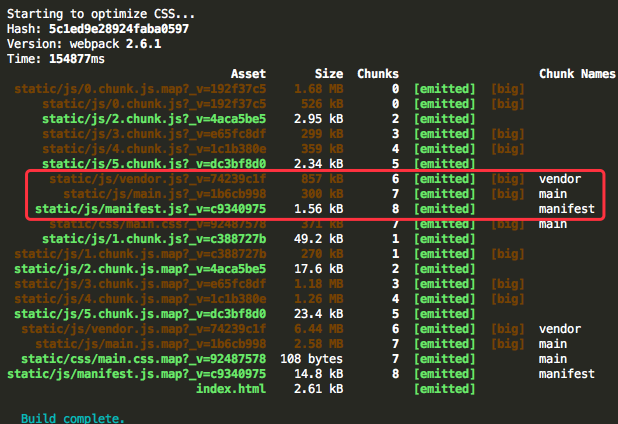
new webpack.HashedModuleIdsPlugin(),

]

构建编辑结果为：



* 增加依赖模块 ../common/util2后重新构建编译：



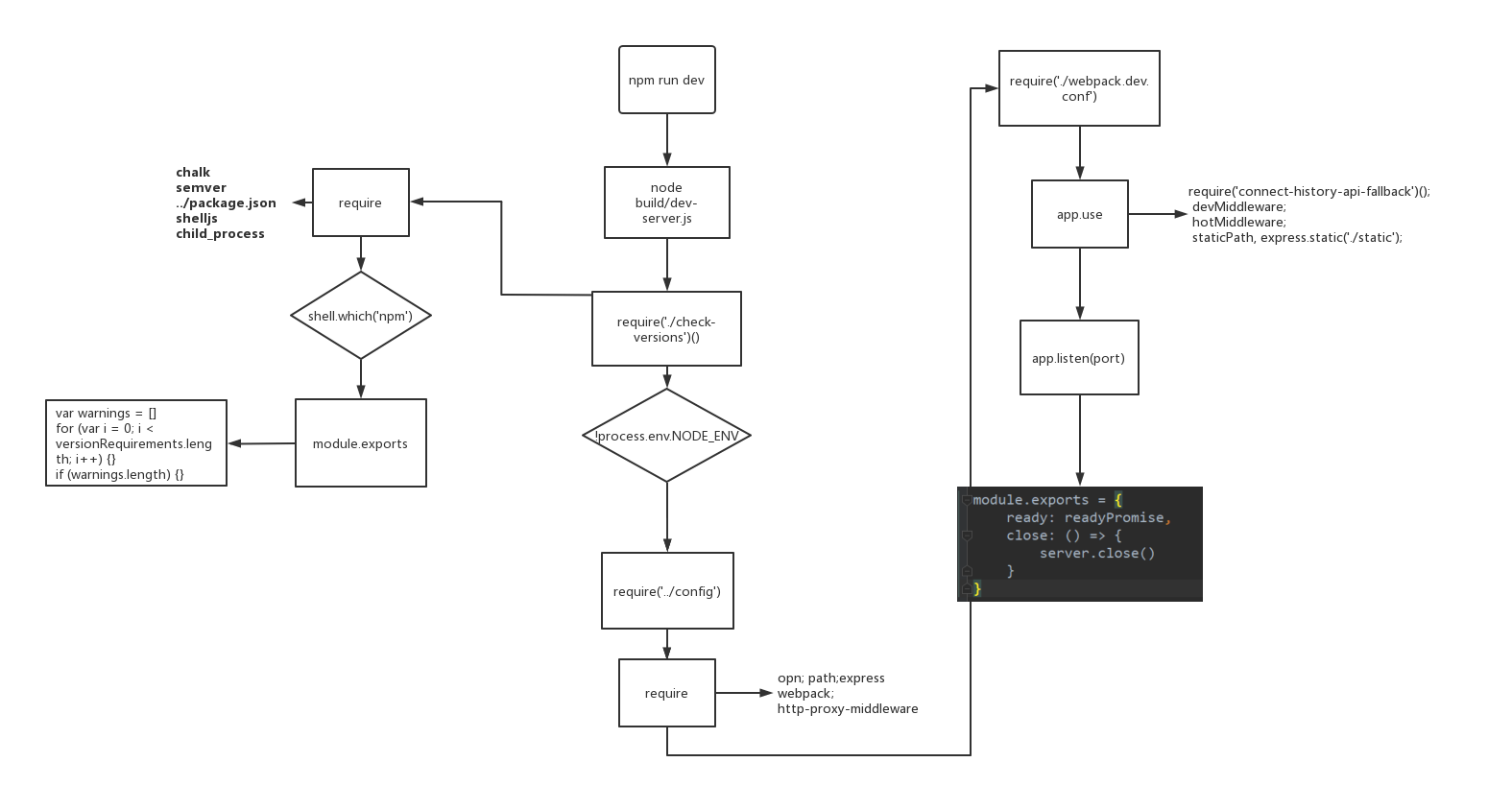
可以看到启用 HashedModuleIdsPlugin后，改变 **App.jsx**的依赖模块后**公共基础模块vendor.js**的 chunkhash保持一致，从而也就能够避免持久化缓存的失效。

* [参考示例](https://link.zhihu.com/?target=https://github.com/loveky/webpack-module-ids/tree/master/withHashedModulesPlugin" \t "_blank)

## 参考资料

* [用 webpack 实现持久化缓存](https://link.zhihu.com/?target=https://sebastianblade.com/using-webpack-to-achieve-long-term-cache/" \t "_blank)
* [Webpack2中的NamedModulesPlugin与HashedModuleIdsPlugin](https://link.zhihu.com/?target=https://loveky.github.io/2017/03/29/webpack-module-ids/" \t "_blank)
* [webpack缓存](https://link.zhihu.com/?target=https://doc.webpack-china.org/guides/caching/" \t "_blank)
* [webpack-module-ids](https://link.zhihu.com/?target=https://github.com/loveky/webpack-module-ids" \t "_blank)

发布于 2017-07-31 10:55



config/index.js

// 获取页面配置

function getPagesConfig () {

try {

let pageFiles=**glob**.sync(`${config.build.srcPath}/pages/\*\*/page.js`)

} catch (err) {

console.log('\\007') // Beep

console.error(err)

}

}

参考：<https://www.npmjs.com/package/glob>

# **glob.sync(pattern, [options])**

* pattern {String} Pattern to be matched
* options {Object}
* return: {Array<String>} filenames found matching the pattern

Perform a synchronous glob search.

let sysDirs = glob.sync(`${config.build.srcPath}/pages/\*/static/js`)

let pagesAttr = getPageEnties(pageFiles)

## **path**

0.12.7 • Public • Published 7 years ago

* **[Readme](https://www.npmjs.com/package/path?activeTab=readme)**
* **[Explore BETA](https://www.npmjs.com/package/path?activeTab=explore)**
* **[2 Dependencies](https://www.npmjs.com/package/path?activeTab=dependencies)**
* **[9,878 Dependents](https://www.npmjs.com/package/path?activeTab=dependents)**
* **[4 Versions](https://www.npmjs.com/package/path?activeTab=versions)**

# **path**

This is an exact copy of the NodeJS ’path’ module published to the NPM registry.

**[Documentation](http://nodejs.org/docs/latest/api/path.html)**

## **Install**

$ npm install --save path

## **License**

MIT

## **Keywords**

* **[ender](https://www.npmjs.com/search?q=keywords:ender)**
* **[path](https://www.npmjs.com/search?q=keywords:path)**

https://www.npmjs.com/package/path

<https://nodejs.org/docs/latest/api/path.html>

Table of contents

* [Path](https://nodejs.org/docs/latest/api/path.html" \l "path)
  + [Windows vs. POSIX](https://nodejs.org/docs/latest/api/path.html" \l "windows-vs-posix)
  + [path.basename(path[, ext])](https://nodejs.org/docs/latest/api/path.html" \l "pathbasenamepath-ext)
  + [path.delimiter](https://nodejs.org/docs/latest/api/path.html" \l "pathdelimiter)
  + [path.dirname(path)](https://nodejs.org/docs/latest/api/path.html" \l "pathdirnamepath)
  + [path.extname(path)](https://nodejs.org/docs/latest/api/path.html" \l "pathextnamepath)
  + [path.format(pathObject)](https://nodejs.org/docs/latest/api/path.html" \l "pathformatpathobject)
  + [path.isAbsolute(path)](https://nodejs.org/docs/latest/api/path.html" \l "pathisabsolutepath)
  + [path.join([...paths])](https://nodejs.org/docs/latest/api/path.html" \l "pathjoinpaths)
  + [path.normalize(path)](https://nodejs.org/docs/latest/api/path.html" \l "pathnormalizepath)
  + [path.parse(path)](https://nodejs.org/docs/latest/api/path.html" \l "pathparsepath)
  + [path.posix](https://nodejs.org/docs/latest/api/path.html" \l "pathposix)
  + [path.relative(from, to)](https://nodejs.org/docs/latest/api/path.html" \l "pathrelativefrom-to)
  + [path.resolve([...paths])](https://nodejs.org/docs/latest/api/path.html" \l "pathresolvepaths)
  + [path.sep](https://nodejs.org/docs/latest/api/path.html" \l "pathsep)
  + [path.toNamespacedPath(path)](https://nodejs.org/docs/latest/api/path.html" \l "pathtonamespacedpathpath)
  + [path.win32](https://nodejs.org/docs/latest/api/path.html" \l "pathwin32)

# **Path[#](https://nodejs.org/docs/latest/api/path.html" \l "path)**

[Stability: 2](https://nodejs.org/docs/latest/api/documentation.html" \l "stability-index) - Stable

****Source Code:**** [lib/path.js](https://github.com/nodejs/node/blob/v18.4.0/lib/path.js)

The node:path module provides utilities for working with file and directory paths. It can be accessed using:

const path = require('node:path');

## **Windows vs. POSIX[#](https://nodejs.org/docs/latest/api/path.html" \l "windows-vs-posix)**

The default operation of the node:path module varies based on the operating system on which a Node.js application is running. Specifically, when running on a Windows operating system, the node:path module will assume that Windows-style paths are being used.

So using path.basename() might yield different results on POSIX and Windows:

On POSIX:

path.basename('C:\\temp\\myfile.html');// Returns: 'C:\\temp\\myfile.html'

On Windows:

path.basename('C:\\temp\\myfile.html');// Returns: 'myfile.html'

To achieve consistent results when working with Windows file paths on any operating system, use [path.win32](https://nodejs.org/docs/latest/api/path.html" \l "pathwin32):

On POSIX and Windows:

path.win32.basename('C:\\temp\\myfile.html');// Returns: 'myfile.html'

To achieve consistent results when working with POSIX file paths on any operating system, use [path.posix](https://nodejs.org/docs/latest/api/path.html" \l "pathposix):

On POSIX and Windows:

path.posix.basename('/tmp/myfile.html');// Returns: 'myfile.html'

On Windows Node.js follows the concept of per-drive working directory. This behavior can be observed when using a drive path without a backslash. For example, path.resolve('C:\\') can potentially return a different result than path.resolve('C:'). For more information, see [this MSDN page](https://docs.microsoft.com/en-us/windows/desktop/FileIO/naming-a-file" \l "fully-qualified-vs-relative-paths).

## **path.basename(path[, ext])[#](https://nodejs.org/docs/latest/api/path.html" \l "pathbasenamepath-ext)**

History

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

* path [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* ext [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type) An optional file extension
* Returns: [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

The path.basename() method returns the last portion of a path, similar to the Unix basename command. Trailing directory separators are ignored, see [path.sep](https://nodejs.org/docs/latest/api/path.html" \l "pathsep).

path.basename('/foo/bar/baz/asdf/quux.html');// Returns: 'quux.html'

path.basename('/foo/bar/baz/asdf/quux.html', '.html');// Returns: 'quux'

Although Windows usually treats file names, including file extensions, in a case-insensitive manner, this function does not. For example, C:\\foo.html and C:\\foo.HTML refer to the same file, but basename treats the extension as a case-sensitive string:

path.win32.basename('C:\\foo.html', '.html');// Returns: 'foo'

path.win32.basename('C:\\foo.HTML', '.html');// Returns: 'foo.HTML'

A [TypeError](https://nodejs.org/docs/latest/api/errors.html" \l "class-typeerror) is thrown if path is not a string or if ext is given and is not a string.

## **path.delimiter[#](https://nodejs.org/docs/latest/api/path.html" \l "pathdelimiter)**

Added in: v0.9.3

* [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

Provides the platform-specific path delimiter:

* ; for Windows
* : for POSIX

For example, on POSIX:

console.log(process.env.PATH);// Prints: '/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin'

process.env.PATH.split(path.delimiter);// Returns: ['/usr/bin', '/bin', '/usr/sbin', '/sbin', '/usr/local/bin']

On Windows:

console.log(process.env.PATH);// Prints: 'C:\Windows\system32;C:\Windows;C:\Program Files\node\'

process.env.PATH.split(path.delimiter);// Returns ['C:\\Windows\\system32', 'C:\\Windows', 'C:\\Program Files\\node\\']

## **path.dirname(path)[#](https://nodejs.org/docs/latest/api/path.html" \l "pathdirnamepath)**

History

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

* path [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* Returns: [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

The path.dirname() method returns the directory name of a path, similar to the Unix dirname command. Trailing directory separators are ignored, see [path.sep](https://nodejs.org/docs/latest/api/path.html" \l "pathsep).

path.dirname('/foo/bar/baz/asdf/quux');// Returns: '/foo/bar/baz/asdf'

A [TypeError](https://nodejs.org/docs/latest/api/errors.html" \l "class-typeerror) is thrown if path is not a string.

## **path.extname(path)[#](https://nodejs.org/docs/latest/api/path.html" \l "pathextnamepath)**

History

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

* path [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* Returns: [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

The path.extname() method returns the extension of the path, from the last occurrence of the . (period) character to end of string in the last portion of the path. If there is no . in the last portion of the path, or if there are no . characters other than the first character of the basename of path (see path.basename()) , an empty string is returned.

path.extname('index.html');// Returns: '.html'

path.extname('index.coffee.md');// Returns: '.md'

path.extname('index.');// Returns: '.'

path.extname('index');// Returns: ''

path.extname('.index');// Returns: ''

path.extname('.index.md');// Returns: '.md'

A [TypeError](https://nodejs.org/docs/latest/api/errors.html" \l "class-typeerror) is thrown if path is not a string.

## **path.format(pathObject)[#](https://nodejs.org/docs/latest/api/path.html" \l "pathformatpathobject)**

Added in: v0.11.15

* pathObject [<Object>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) Any JavaScript object having the following properties:
  + dir [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
  + root [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
  + base [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
  + name [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
  + ext [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* Returns: [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

The path.format() method returns a path string from an object. This is the opposite of [path.parse()](https://nodejs.org/docs/latest/api/path.html" \l "pathparsepath).

When providing properties to the pathObject remember that there are combinations where one property has priority over another:

* pathObject.root is ignored if pathObject.dir is provided
* pathObject.ext and pathObject.name are ignored if pathObject.base exists

For example, on POSIX:

// If `dir`, `root` and `base` are provided,// `${dir}${path.sep}${base}`// will be returned. `root` is ignored.

path.format({

root: '/ignored',

dir: '/home/user/dir',

base: 'file.txt'

});// Returns: '/home/user/dir/file.txt'

// `root` will be used if `dir` is not specified.// If only `root` is provided or `dir` is equal to `root` then the// platform separator will not be included. `ext` will be ignored.

path.format({

root: '/',

base: 'file.txt',

ext: 'ignored'

});// Returns: '/file.txt'

// `name` + `ext` will be used if `base` is not specified.

path.format({

root: '/',

name: 'file',

ext: '.txt'

});// Returns: '/file.txt'

On Windows:

path.format({

dir: 'C:\\path\\dir',

base: 'file.txt'

});// Returns: 'C:\\path\\dir\\file.txt'

## **path.isAbsolute(path)[#](https://nodejs.org/docs/latest/api/path.html" \l "pathisabsolutepath)**

Added in: v0.11.2

* path [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* Returns: [<boolean>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "Boolean_type)

The path.isAbsolute() method determines if path is an absolute path.

If the given path is a zero-length string, false will be returned.

For example, on POSIX:

path.isAbsolute('/foo/bar'); // true

path.isAbsolute('/baz/..'); // true

path.isAbsolute('qux/'); // false

path.isAbsolute('.'); // false

On Windows:

path.isAbsolute('//server'); // true

path.isAbsolute('\\\\server'); // true

path.isAbsolute('C:/foo/..'); // true

path.isAbsolute('C:\\foo\\..'); // true

path.isAbsolute('bar\\baz'); // false

path.isAbsolute('bar/baz'); // false

path.isAbsolute('.'); // false

A [TypeError](https://nodejs.org/docs/latest/api/errors.html" \l "class-typeerror) is thrown if path is not a string.

## **path.join([...paths])[#](https://nodejs.org/docs/latest/api/path.html" \l "pathjoinpaths)**

Added in: v0.1.16

* ...paths [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type) A sequence of path segments
* Returns: [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

The path.join() method joins all given path segments together using the platform-specific separator as a delimiter, then normalizes the resulting path.

Zero-length path segments are ignored. If the joined path string is a zero-length string then '.' will be returned, representing the current working directory.

path.join('/foo', 'bar', 'baz/asdf', 'quux', '..');// Returns: '/foo/bar/baz/asdf'

path.join('foo', {}, 'bar');// Throws 'TypeError: Path must be a string. Received {}'

A [TypeError](https://nodejs.org/docs/latest/api/errors.html" \l "class-typeerror) is thrown if any of the path segments is not a string.

## **path.normalize(path)[#](https://nodejs.org/docs/latest/api/path.html" \l "pathnormalizepath)**

Added in: v0.1.23

* path [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* Returns: [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

The path.normalize() method normalizes the given path, resolving '..' and '.' segments.

When multiple, sequential path segment separation characters are found (e.g. / on POSIX and either \ or / on Windows), they are replaced by a single instance of the platform-specific path segment separator (/ on POSIX and \ on Windows). Trailing separators are preserved.

If the path is a zero-length string, '.' is returned, representing the current working directory.

For example, on POSIX:

path.normalize('/foo/bar//baz/asdf/quux/..');// Returns: '/foo/bar/baz/asdf'

On Windows:

path.normalize('C:\\temp\\\\foo\\bar\\..\\');// Returns: 'C:\\temp\\foo\\'

Since Windows recognizes multiple path separators, both separators will be replaced by instances of the Windows preferred separator (\):

path.win32.normalize('C:////temp\\\\/\\/\\/foo/bar');// Returns: 'C:\\temp\\foo\\bar'

A [TypeError](https://nodejs.org/docs/latest/api/errors.html" \l "class-typeerror) is thrown if path is not a string.

## **path.parse(path)[#](https://nodejs.org/docs/latest/api/path.html" \l "pathparsepath)**

Added in: v0.11.15

* path [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* Returns: [<Object>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object)

The path.parse() method returns an object whose properties represent significant elements of the path. Trailing directory separators are ignored, see [path.sep](https://nodejs.org/docs/latest/api/path.html" \l "pathsep).

The returned object will have the following properties:

* dir [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* root [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* base [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* name [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* ext [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

For example, on POSIX:

path.parse('/home/user/dir/file.txt');// Returns:// { root: '/',// dir: '/home/user/dir',// base: 'file.txt',// ext: '.txt',// name: 'file' }

┌─────────────────────┬────────────┐

│ dir │ base │

├──────┬ ├──────┬─────┤

│ root │ │ name │ ext │

" / home/user/dir / file .txt "

└──────┴──────────────┴──────┴─────┘

(All spaces in the "" line should be ignored. They are purely for formatting.)

On Windows:

path.parse('C:\\path\\dir\\file.txt');// Returns:// { root: 'C:\\',// dir: 'C:\\path\\dir',// base: 'file.txt',// ext: '.txt',// name: 'file' }

┌─────────────────────┬────────────┐

│ dir │ base │

├──────┬ ├──────┬─────┤

│ root │ │ name │ ext │

" C:\ path\dir \ file .txt "

└──────┴──────────────┴──────┴─────┘

(All spaces in the "" line should be ignored. They are purely for formatting.)

A [TypeError](https://nodejs.org/docs/latest/api/errors.html" \l "class-typeerror) is thrown if path is not a string.

## **path.posix[#](https://nodejs.org/docs/latest/api/path.html" \l "pathposix)**

History

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

* [<Object>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object)

The path.posix property provides access to POSIX specific implementations of the path methods.

The API is accessible via require('node:path').posix or require('node:path/posix').

## **path.relative(from, to)[#](https://nodejs.org/docs/latest/api/path.html" \l "pathrelativefrom-to)**

History

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

* from [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* to [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* Returns: [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

The path.relative() method returns the relative path from from to to based on the current working directory. If from and to each resolve to the same path (after calling path.resolve() on each), a zero-length string is returned.

If a zero-length string is passed as from or to, the current working directory will be used instead of the zero-length strings.

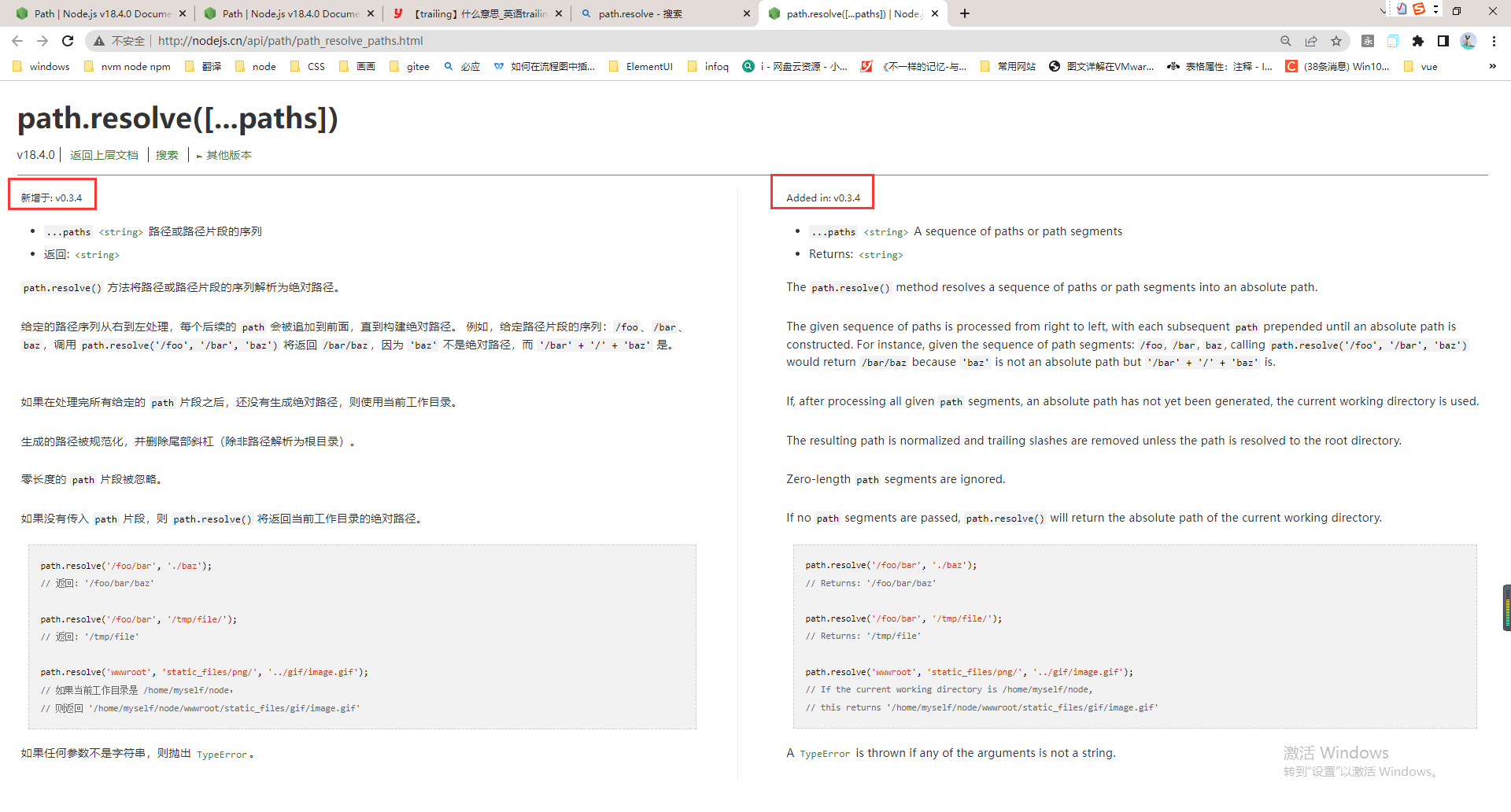
For example, on POSIX:

path.relative('/data/orandea/test/aaa', '/data/orandea/impl/bbb');// Returns: '../../impl/bbb'

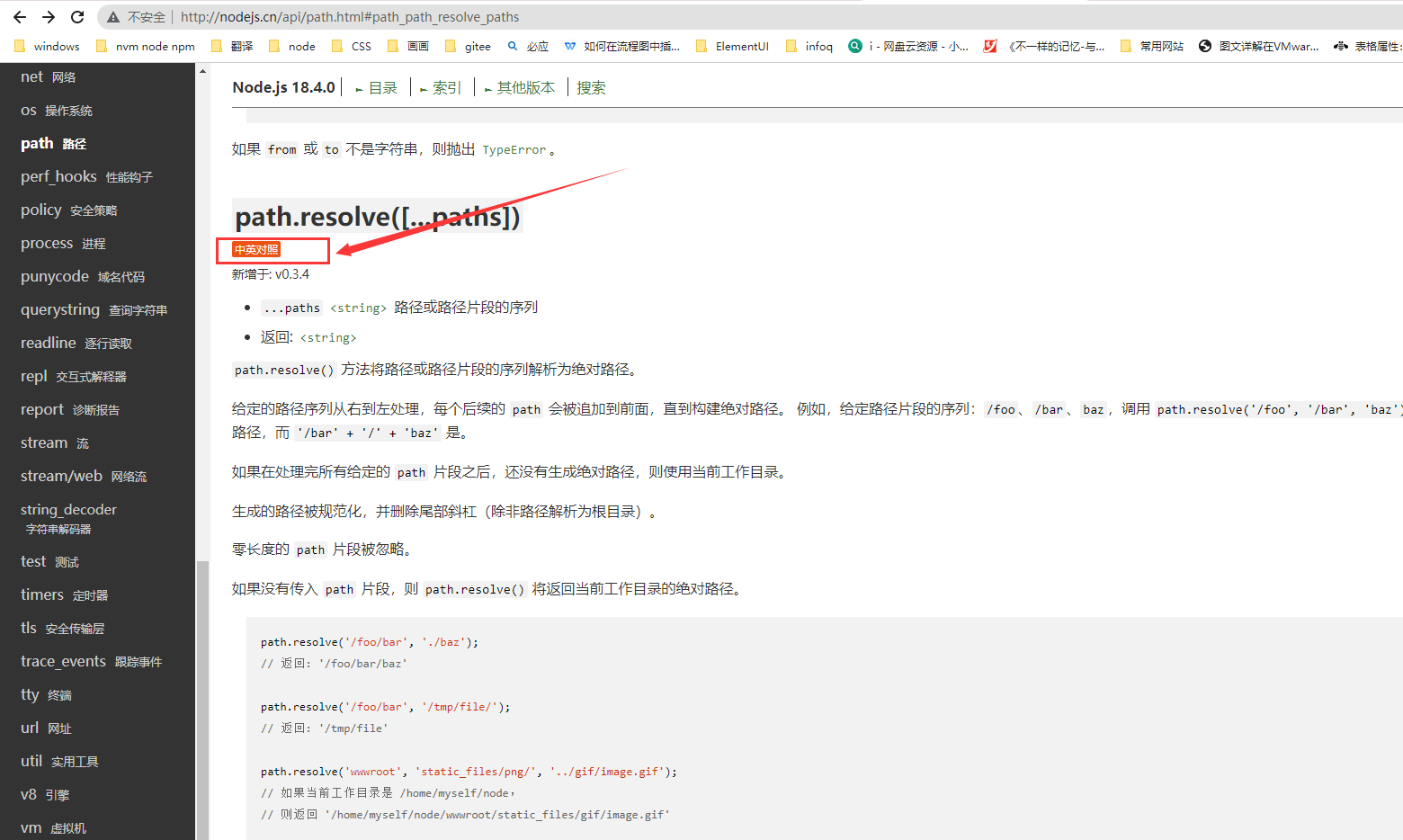
On Windows:

path.relative('C:\\orandea\\test\\aaa', 'C:\\orandea\\impl\\bbb');// Returns: '..\\..\\impl\\bbb'

A [TypeError](https://nodejs.org/docs/latest/api/errors.html" \l "class-typeerror) is thrown if either from or to is not a string.



**中英对照功能 <http://nodejs.cn/api/documentation.html> ！！！！！**



## **path.resolve([...paths])[#](https://nodejs.org/docs/latest/api/path.html" \l "pathresolvepaths)**

Added in: v0.3.4

* ...paths [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type) A sequence of paths or path segments
* Returns: [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

The path.resolve() method resolves a sequence of paths or path segments into an absolute path.

The given sequence of paths is processed from right to left, with each subsequent path prepended **until an absolute path is constructed.**

**For instance,** given the sequence of path segments: /foo, /bar, baz, calling path.resolve('/foo', '/bar', 'baz') would return /bar/baz because 'baz' is not an absolute path but '/bar' + '/' + 'baz' is.

If, after processing all given path segments, an absolute path has not yet been generated, the current working directory is used.

The resulting path is normalized and **trailing slashes** (尾部斜杠)are removed unless the path is resolved to the root directory.

Zero-length path segments are ignored.

If no path segments are passed, path.resolve() will return the absolute path of the current working directory.

path.resolve('/foo/bar', './baz');// Returns: '/foo/bar/baz'

path.resolve('/foo/bar', '/tmp/file/');// Returns: '/tmp/file'

path.resolve('wwwroot', 'static\_files/png/', '../gif/image.gif');// If the current working directory is /home/myself/node,// this returns '/home/myself/node/wwwroot/static\_files/gif/image.gif'

A [TypeError](https://nodejs.org/docs/latest/api/errors.html" \l "class-typeerror) is thrown if any of the arguments is not a string.

## **path.sep[#](https://nodejs.org/docs/latest/api/path.html" \l "pathsep)**

Added in: v0.7.9

* [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

Provides the platform-specific path segment separator:

* \ on Windows
* / on POSIX

For example, on POSIX:

'foo/bar/baz'.split(path.sep);// Returns: ['foo', 'bar', 'baz']

On Windows:

'foo\\bar\\baz'.split(path.sep);// Returns: ['foo', 'bar', 'baz']

On Windows, both the forward slash (/) and backward slash (\) are accepted as path segment separators; however, the path methods only add backward slashes (\).

## **path.toNamespacedPath(path)[#](https://nodejs.org/docs/latest/api/path.html" \l "pathtonamespacedpathpath)**

Added in: v9.0.0

* path [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)
* Returns: [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures" \l "String_type)

On Windows systems only, returns an equivalent [namespace-prefixed path](https://docs.microsoft.com/en-us/windows/desktop/FileIO/naming-a-file" \l "namespaces) for the given path. If path is not a string, path will be returned without modifications.

This method is meaningful only on Windows systems. On POSIX systems, the method is non-operational and always returns path without modifications.

## **path.win32[#](https://nodejs.org/docs/latest/api/path.html" \l "pathwin32)**

History

* [<Object>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object)

The path.win32 property provides access to Windows-specific implementations of the path methods.

The API is accessible via require('node:path').win32 or require('node:path/win32').

1. path.dirname('/foo/bar/baz/asdf/quux');// Returns: '/foo/bar/baz/asdf' [↑](#endnote-ref-0)