node.js

++

path.resolve()

1. 当path.resolve参数中不存在开头带'/'的参数时 将return 绝对路径

2. 否则 将return '/' + 最后一个带'/'文件文件名 + 剩下文件夹

`path.resolve('foo/bar', '/tmp/file/', '..', 'a/../subfile')`

相当于

```
cd foo/bar
cd /tmp/file/
cd ..
cd a/../subfile
pwd
```

例子：

```
path.resolve('/foo/bar', './baz')
// 输出结果为
'/foo/bar/baz'
path.resolve('/foo/bar', '/tmp/file/')
// 输出结果为
'/tmp/file'

path.resolve('wwwroot', 'static_files/png/', '../gif/image.gif')
// 当前的工作路径是 /home/itbilu/node, 则输出结果为
'/home/itbilu/node/wwwroot/static_files/gif/image.gif'
```

path.join与path.resolve区别

1. path.resolve使用返回绝对路径而path.join只是简单的拼接

```
> path.resolve('./')
'/Users/yunshan/WORK/xiaoer/purchase-operate/pomp-web/src/main/webapp/web'
> path.join('./')
'./'
>
```

2. path.resolve相当于对于每个路径进行cd操作 并返回最后的绝对路径 而path.join只是只是简单的拼接

```
> path.join('./','/a','/b')
'a/b'
> path.resolve('./','/a','/b')
'/b'
>
```

++

配置文档中出现了 path.posix的用法 查了一下相关用法

默认情况下node会根据不同的操作系统进行兼容性处理、力保输出结果在不同平台一致 但是某些情况还是不能完美兼容所有情况的 所以node提供了win32和posix两种api

除了目录结构有区别外，路径也是有区别的。windows是用反斜杠 \ 分割目录或者文件的，而在类Unix的系统中是用的 / 。

```
windows的路径:  C:\temp\myfile.html
类Unix的路径:   /tmp/myfile.html
```

path.win32 （想要任何操作系统处理Window文件路径获得一致效果）

path.posix （ 想要任何操作体统处理posix文件路径时获得一致效果）

详情：http://nodejs.cn/api/path.html#path_path_posix

++

配置中出现了使用webpack-merge 的情况， 如图

```
2  const utils = require('./utils')
3  const webpack = require('webpack')
4  const config = require('../config')
5  const merge = require('webpack-merge')
6  const path = require('path')
7  const baseWebpackConfig = require('./webpack.base.conf')
8  const CopyWebpackPlugin = require('copy-webpack-plugin')
9  const HtmlWebpackPlugin = require('html-webpack-plugin')
10 const FriendlyErrorsPlugin = require('friendly-errors-webpack-plugin')
11 const portfinder = require('portfinder')
12
13 const HOST = process.env.HOST
14 const PORT = process.env.PORT && Number(process.env.PORT)
15
16 const devWebpackConfig = merge(baseWebpackConfig, {
17   module: {
18     rules: utils.styleLoaders({ sourceMap: config.dev.cssSourceMap, usePostCSS:
```

webpack-merge提供了一个合并函数，用于连接数组并合并创建新对象的对象。 如果遇到函数，它将执行它们，通过算法运行结果，然后再次将返回的值包装在函数中。

这种行为在配置webpack时特别有用，尽管它有超出它的用途。 无论何时需要合并配置对象，webpack-merge都可以派上用场。

使用详情：https://www.npmjs.com/package/webpack-merge
发现一个和项目中接口相似的配置文件 并附有注释：
https://blog.csdn.net/itkingone/article/details/70331783
++