validate-npm-package-name 检测 npm 包是否符合标准

这个包确实挺眼熟的，vue-cli中就有用到，代码如下

```
const result = validateProjectName(name)
  if (!result.validForNewPackages) {
    console.error(chalk.red(`Invalid project name: "${name}"`))
    result.errors && result.errors.forEach(err => {
      console.error(chalk.red.dim('Error: ' + err))
    })
    result.warnings && result.warnings.forEach(warn => {
      console.error(chalk.red.dim('Warning: ' + warn))
    })
    exit(1)
  }
```

就跟包的名字一样，作用是检验npm包是否符合标准，如果不符合输出错误信息并exit
走，直接去源码看看
短小精悍，算上空行也只有100行的代码

```
'use strict'

var scopedPackagePattern = new RegExp('^(?:@([^/]+?)[/])?([^/]+?)$')
var builtins = require('builtins')
var blacklist = [
  'node_modules',
  'favicon.ico'
]
/**
 * 传入的参数name，也就是包名
 * warrings与errors都是数组，一旦有不符合标准的则对应增加错误/警告信息
 */
var validate = module.exports = function (name) {
  var warnings = []
  var errors = []

  // 包名不能为空
  if (name === null) {
    errors.push('name cannot be null')
    return done(warnings, errors)
  }

  // 包名不能未定义
  if (name === undefined) {
    errors.push('name cannot be undefined')
    return done(warnings, errors)
  }

  // 包名必须为字符串
  if (typeof name !== 'string') {
```

```
  errors.push('name must be a string')
  return done(warnings, errors)
}

// 字符串长度不能为0
if (!name.length) {
  errors.push('name length must be greater than zero')
}

// 不能以 点开头
if (name.match(/^\./)) {
  errors.push('name cannot start with a period')
}

// 不能以 下划线开头
if (name.match(/^_/)) {
  errors.push('name cannot start with an underscore')
}

// 头尾不能有空格
if (name.trim() !== name) {
  errors.push('name cannot contain leading or trailing spaces')
}

// No funny business
/**
 * var blacklist = ['node_modules','favicon.ico']
 * 不能是黑名单中的名字
 */
blacklist.forEach(function (blacklistedName) {
  if (name.toLowerCase() === blacklistedName) {
    errors.push(blacklistedName + ' is a blacklisted name')
  }
})

// Generate warnings for stuff that used to be allowed

// 不能是核心模块的名字，builtins是一个json，这比黑名单好一点，只是个警告
builtins.forEach(function (builtin) {
  if (name.toLowerCase() === builtin) {
    warnings.push(builtin + ' is a core module name')
  }
})

// really-long-package-names-------------------------------such--length-----many---wow
// 包名不能太长，不能长于214，不然生成警告
if (name.length > 214) {
  warnings.push('name can no longer contain more than 214 characters')
}

// mIxeD CaSe nAMEs
// 如果有大写则生成警告
if (name.toLowerCase() !== name) {
  warnings.push('name can no longer contain capital letters')
}

// 不能包含 ~)('!* 任意一个字符串
if (/[~'!()*]/.test(name.split('/').slice(-1)[0])) {
  warnings.push('name can no longer contain special characters ("~\'!()*")')
}
```

```
  /**
   * 不能存在url不友好字符
   * encodeURIComponent() 函数可把字符串作为 URI 组件进行编码。
   * URL编码不会对ASCII字母和数字进行编码
   * 也不会对这些 ASCII 标点符号进行编码： - _ . ! ~ * ' ( )
   * 其他字符则把ASCII转成16进制
   * 也就是过滤了 除 字母，数组， - _ . ! ~ * ' ( ) 以外的字符
   * var scopedPackagePattern = new RegExp('^(?:@([^/]+?)[/])?([^/]+?)$')
   * 再用正则
   */
  if (encodeURIComponent(name) !== name) {
    // Maybe it's a scoped package name, like @user/package
    // 可能是scope package name，例如 @user/package
    var nameMatch = name.match(scopedPackagePattern)
    if (nameMatch) {
      var user = nameMatch[1]  // 得到 user
      var pkg = nameMatch[2]   // 得到 package
      // 重新URL编码，如果没问题就退出，否则添加错误信息
      if (encodeURIComponent(user) === user && encodeURIComponent(pkg) === pkg) {
        return done(warnings, errors)
      }
    }

    errors.push('name can only contain URL-friendly characters')
  }

  return done(warnings, errors)
}

validate.scopedPackagePattern = scopedPackagePattern

/**
 * 封装result对象
 * validForNewPackages：boolean  //  是否符合新包标准
 * validForOldPackages：boolean  //  是否符合旧包标准
 * warnings                      //  警告信息
 * errors                        //  错误信息
 *
 * @param {Array} warnings
 * @param {Array} errors
 * @returns
 */
var done = function (warnings, errors) {
  var result = {
    validForNewPackages: errors.length === 0 && warnings.length === 0,
    validForOldPackages: errors.length === 0,
    warnings: warnings,
    errors: errors
  }
  // 如果没有错误/警告信息，则删除错误/警告属性
  if (!result.warnings.length) delete result.warnings
  if (!result.errors.length) delete result.errors
  return result
}
```

总结

这次源码阅读还是很愉悦，很快就看完了。
但是，发现正则无疑是痛处了。
最开心的事情，之前经过一轮系统的网络复习后，一眼就看出了别人笔记中关于URL编码的错误。
果然只要你学习了就能有回报，加油。