# Visual Studio Code 下的 jsconfig.json

|  |
| --- |
|  |

# vue.config.js 配置详解

|  |
| --- |
| <https://www.yuque.com/szr190/yzzm82/ux40w7> |

# Git

## tortoiseGit和gitlab 使用SSH 加密访问

|  |
| --- |
| <https://hbiao68.iteye.com/blog/2400835>  <https://www.jianshu.com/p/5d89119bece7>  <https://www.jianshu.com/p/60465ff1ced3> |

## windows下生成ssh key并上传到gitlab

|  |
| --- |
| <https://blog.csdn.net/qq_28056641/article/details/53445832>  <https://blog.csdn.net/xyzchenxiaolin/article/details/51852333> |

## 集中式与分布式的区别

|  |
| --- |
| git svn 区别  svn是集中式版本管理，git是分布式版本管理  svn传输方式为文件，git元数据，存储快  git 严格的分支管理，有本地版本管理 |

## 配置

|  |
| --- |
| 配置用户名：git config --global user.name "xiaoming"  配置邮箱 ： git config --global user.email "xiaoming@shi.cn"  查看配置: git config --list |

## 初始化仓储/仓库

|  |
| --- |
| git init :创建一个仓库/仓储  本质就是在当前目录添加了一个.git的隐藏目录,里面存储项目中文件的变更信息 |

## 查看状态

|  |
| --- |
| git status 或 git status -s // 查看简要的状态的信息 |

## 添加文件

|  |
| --- |
| git add 文件名  git add . 一次性把所有修改过或新添加的文件添加的仓库门口，这里的.表示当前路径 |

## 提交文件

|  |
| --- |
| git commit -m "我写了一行代码"  git commit -m -c // -c表示不写日志,不推荐  把文件存储到仓库里面,仓库是有管理员的，需要通过-m告诉管理员,这次存储的东西有什么不同之处 |

## 合并操作

|  |
| --- |
| git commit -a -m "提交的信息"  是git add . 和 git commit -m 的合并操作  注意：需要配置user.name 与user.email |

## 对比与前一次提交文件的变化

|  |
| --- |
| git diff 用于项目中的代码与add后的代码的区别 |

## 版本回退

|  |
| --- |
| git log 查看以前提交的日志  git log --oneline 在一行内显示一次提交信息  commit ..... 后跟对应是的版本号，是唯一的  git reset --hard Head^  HEAD 指的是当前分支最新的一次提交（就是 git log 所能看到的第一个提交）  HEAD^ 表示最近一次提交的前一次提交  HEAD^^ 表示最近一次提交的前前一次提交  git reset --hard Head~100 回退到往前第100次提交时的代码,最近的第100次提交  git reset --hard 321fa 通过唯一标识来回退到指定的提交  git reflog 可以查看每一次的提交的版本号 |

## 忽略文件

|  |
| --- |
| 新建 .gitingore文件,在这个文件里写一些想要被git忽略的文件或者文件夹  不要省略./ 除非有必要  忽略某个目录: /node\_modules  忽略某个文件: /css/my.css  忽略某一类文件: /css/\*.css  忽略目录下所有文件: \*.\*  忽略所有名为node\_modules的目录: node\_modules  #号表示注释  # #号开头的行，会被当作注释信息  # 如果只是写目录名，会默认忽略所有名为该目录的文件  # 如果是想指定具体的目录，或者文件被忽略，需要使用绝对路径的形式  /node\_modules  # 指定一个具体的文件被忽略  /index.html |

## 创建/合并分支

|  |
| --- |
| git branch 分支名 功能写到一半，想要保存，又不想影响别人的代码,这时我们创建自己的分支去备份代码  git branch 列出所有分支,在列出所以分支中，当前分支前会有个\*号  git checkout 分支名 切换分支  git checkout master切换到主分支  git merge 分支名 合并分支,把指定分支名的分支合并到当前分支,如:把新创建的分支的合并到主分支中  git branch -d 分支名 删除指定的分支  git branch -d dev 表示删除dev分支,当合并分支后，如果不需要再使用dev分支，则可以直接删除  git checkout -b dev创建并切换分支 |

## 解决冲突

|  |
| --- |
| 在不同的分支中对同一文件，同一行代码(只要是使用后面的往发生改变),git就无法解决这种类型的冲突，因为git只是个工具，它不能够判断对我们来说，哪一部分代码是最有效.  这时就需要我们手动的去解决冲突(其实就是在文件中有选择的进行删减)  Updated upstream 和=====之间的内容就是pull下来的内容  ====和stashed changes之间的内容就是本地修改的内容  <<<<< Updated upstream  xxxxxxxxx  =======================  xxxxxxxxxxxxx  >>>>>>>stashed changes |

## 工作区与暂存区

|  |
| --- |
| 工作区指的是我们写代码的目录(不包括.git隐藏目录)  暂存区是我们执行`git add xxx` 之后文件存放的地方 |

## 克隆分支

|  |
| --- |
| git clone 路径 空项目时用 没有.git目录时用 |

## 提交分支到远程仓储(push)

|  |
| --- |
| git remote add origin [远程仓储地址] git remote ---查看: 添加一个远程仓储的地址，取名为origin  git push -u origin [远程的某个分支名]: 将本地当前分支提交到远程的某个分支  -u参数表示 将当前分支与远程分支关联起来，下次拉取时直接git pull origin就可以了 |

## 从远程仓储拉取分支(pull)

|  |
| --- |
| git pull [远程仓储地址] master:master (提交代码前先pull代码 看看是否有冲突问题 会处理合并分支问题 手动修改)  将远程仓储的某个分支拉取到本地的某个分支并合并  :冒号及后面不写表示与当前分支合并  相当于两个命令:  git fetch [远程仓储地址]  fetch的取来的意思  git merge [远程仓储地址]/master  上面两个命令的意思是取远程仓储的master分支并与当前分支合并  git init 后可以 pull 因为有主分支  提交前 要 pull 下 查看是否有冲突 手动修改 |

## git与gitHub的关系

|  |
| --- |
| git版本管理工具,  gitHub就是一个网站 |

## 注意问题

|  |
| --- |
| 1.不推荐使用window右键新建文本文件,编码问题  2.在cmd命令行中进行git操作，字符串值使用双引号  add commit pull push |

# npm

## 简介

|  |
| --- |
| 官网 <https://www.npmjs.com/>  package.json <http://www.runoob.com/nodejs/nodejs-npm.html>  npm 全局安装的路径 C:\Users\AppData\Roaming\npm  npm(node package managen):node包管理器,主要是管理node包，包括：安装、卸载、更新、查看、搜索、发布等。  记录了每个包的信息，包括作者、版本、依赖、授权信息等。  作用：将开发者从繁琐的包管理工作（版本、依赖等）中解放出来，更加专注于功能的开发。  npm不仅仅能够安装前端的依赖包,还可以安装一些前端的工具(gulp)  npm 3 以前的版本文件依赖是层级依赖  npm 3 以后依赖关系改为平行依赖（windows上路径过长问题） |

## 常用命令

|  |
| --- |
| npm config set registry https://registry.npm.taobao.org  npm cache clean --force  npm install -g cnpm --registry=https://registry.npm.taobao.org 淘宝 NPM 镜像  npm -v： 查看npm版本  npm -h ： 查看所有命令  npm install -h ：查看install命令  npm init ：初始化package.json  npm init -y：初始化package.json，使用默认配置  npm ls <包名> ： 查看具体包  npm install <包名> -g  npm install -g @vue/cli 全局安装  npm uninstall -g vue-cli 全局删除  npm bin ：全局安装包路径  npm install <包名>@<version> 下载指定版本  npm install jquery --save-dev 局部安装  npm install jquery --save局部安装  npm install jquery -S 局部安装  npm install jquery -D 局部安装  npm uninstall jquery 局部删除  npm remove <包名> 卸载  npm remove <包名> --save  npm adduser: 添加用户名 密码 邮箱  npm publish: 发布  npm uppulish 撤销发布  npm run buld 运行定义好的脚本  {  "name": "shipeng",  "version": "1.0.0",  "description": "",  "main": "index.js",  "keywork": "xm",  "scripts": {  "test": "echo \"Error: no test specified\" && exit 1"  },  "author": "",  "license": "ISC",  "devDependencies": {},  "dependencies": {}  } |

# yarn

|  |
| --- |
| 安装 yarn scoop install yarn  yarn global bin 全局安装包路径  yarn global remove vue-cli 全局删除  yarn global add @vue/cli 全局安装  Yarn cache clena 清除缓存  yarn --version  yarn init  yarn 安装  yarn install 安装  yarn add [package] --dev/ --peer / --optional  yarn upgrade [pakage] 升级  yarn remove [package] 删除  yarn run buld 运行定义好的脚本  yarn publish 发布到包管理器  yarn check  yarn login |

# gulp

## 常用指令

|  |
| --- |
| <http://javascript.ruanyifeng.com/tool/gulp.html>  gulp.src(['js/\*\*/\*.js', '!js/\*\*/\*.min.js']) 指定的文件会转换成数据流  gulp.dest()将管道的输出写入文件  gulp.task() 用于定义具体的任务  gulp.watch() 指定需要监视的文件  gulp.task('js', function () {  gulp.src('./client/templates/\*.jade')  .pipe(jade())  .pipe(gulp.dest('./build/templates'))  .pipe(minify())  .pipe(gulp.dest('./build/minified\_templates'));  }  gulp-concat  gulp-uglify  gulp-less/scss  gulp-notify  gulp-cssnano  gulp-cleam brawser-sync  gulp-replace  常用gulp插件：  编译 Less：gulp-less <https://www.npmjs.com/package/gulp-less>  创建本地服务器：gulp-connect <https://www.npmjs.com/package/gulp-connect>  合并文件：gulp-concat <https://www.npmjs.com/package/gulp-concat>  最小化 js 文件：gulp-uglify <https://www.npmjs.com/package/gulp-uglify>  重命名文件：gulp-rename <https://www.npmjs.com/package/gulp-rename>  最小化 css 文件：gulp-minify-css <https://www.npmjs.com/package/gulp-minify-css>  压缩html文件 gulp-minify-html <https://www.npmjs.com/package/gulp-minify-html>  压缩html文件 gulp-htmlmin <https://www.npmjs.com/package/gulp-htmlmin>  最小化图像：gulp-imagemin <https://www.npmjs.com/package/gulp-imagemin> |

## gulpfile.js

|  |
| --- |
| 'use strict';  // gulp 自动化构建工具:用来机械化的完成重复的工作,将重复的工作抽象成一个一个的任务  /\*\*  \* 1.Less编译 合并 压缩  \* 2.js 合并，压缩，混淆  \* 3.css 压缩 ，合并  \* 3.img 复制  \* 4.html 压缩  \*/  // 全局安装gulp是工具  // 项目下安装的是包  *var* gulp = require('gulp');  // 压缩html页面javascript、css，去除页面空格、注释，删除多余属性等操作  // http://www.ydcss.com/archives/20  *var* htmlmin = require('gulp-htmlmin');  // ============================================  // 合并javascript文件，减少网络请求  // http://www.ydcss.com/archives/83  *var* concat = require('gulp-concat');  // 引入用于压缩的  *var* uglify = require('gulp-uglify');  // ============================================  // 把less文件转换为css  // http://www.ydcss.com/archives/34  *var* less = require('gulp-less');  // 当less有各种引入关系时，编译后不容易找到对应less文件，所以需要生成sourcemap文件，方便修改  // http://www.ydcss.com/archives/34  *var* sourcemaps = require('gulp-sourcemaps');  //当发生异常时提示错误  *var* notify = require('gulp-notify');  // var plumber = require('gulp-plumber');  // 压缩css  *var* cssnano = require('gulp-cssnano');  // 或  *var* cssmin = require('gulp-minify-css');  // ===========================================  *var* imagemin = require('gulp-imagemin');  *var* pngquant = require('imagemin-pngquant');  *var* cache = require('gulp-cache');  // ================================================  // http://www.browsersync.cn/docs/gulp/  *var* browserSync = require('browser-sync');  // =========================  //确保已本地安装gulp-make-css-url-version [cnpm install gulp-make-css-url-version --save-dev]  *var* cssver = require('gulp-make-css-url-version');  *var* autoprefixer = require('gulp-autoprefixer');  *var* rename = require('gulp-rename'); // 文件重命名  *var* clean = require('gulp-clean'); // 文件清理  // ============================================  gulp.task('html', *function* () {  *var* options = {  removeComments: true,//清除HTML注释  collapseWhitespace: true,//压缩HTML 移除html中的空白  collapseBooleanAttributes: true,//省略布尔属性的值 <input checked="true"/> ==> <input />  removeEmptyAttributes: true,//删除所有空格作属性值 <input id="" /> ==> <input />  removeScriptTypeAttributes: true,//删除<script>的type="text/javascript"  removeStyleLinkTypeAttributes: true,//删除<style>和<link>的type="text/css"  minifyJS: true,//压缩页面JS  minifyCSS: true//压缩页面CSS  };  gulp.src('src/\*.html')  .pipe(htmlmin(options))  .pipe(gulp.dest('dist'))  .pipe(browserSync.reload({ // 不会对当前任务有影响,只是通知浏览器刷新  stream: true  }))  })  gulp.task('script', *function* () {  // gulp.src('src/\*.js')  gulp.src(['src/js/app1.js', 'src/js/app2.js', '!src/js/\*\*/{app3, app4}.js'])  .pipe(concat('all.js')) //合并后的文件名  .pipe(uglify())  /\* .pipe(uglify({  mangle: true,//类型：Boolean 默认：true 是否修改变量名  compress: true,//类型：Boolean 默认：true 是否完全压缩  preserveComments: 'all' //保留所有注释  }))\*/  .pipe(gulp.dest('dist/js'))  .pipe(browserSync.reload({ // 不会对当前任务有影响,只是通知浏览器刷新  stream: true  }))  })  gulp.task('less', *function* () {  gulp.src(['./src/less/\*.less'])  //.pipe(plumber({errorHandler: notify.onError('Error: <%= error.message %>')}))  .pipe(less())   // 把less 转换为css，一般预处理css都可以自动导包  .pipe(sourcemaps.init()) // 执行sourcemaps  .pipe(sourcemaps.write('maps')) // 地图输出路径（存放位置）  .pipe(cssnano()) // 压缩css文件  .pipe(gulp.dest('dist/css'))// 输出文件到指定目录  .pipe(browserSync.reload({ // 不会对当前任务有影响,只是通知浏览器刷新  stream: true  }))  // gulp.watch('./src/\*\*/\*.less', ['less'])  })  //图片压缩  gulp.task('imagemin', *function* () {  gulp.src('./src/images/\*\*/\*.{png,jpg,gif,ico}')  .pipe(cache(imagemin({  progressive: true,  svgoPlugins: [{ removeViewBox: false }],  use: [pngquant()]  })))  .pipe(gulp.dest('./src/images'));  });  //css压缩  gulp.task('cssmin', *function* () {  gulp.src('src/css/\*.css')  .pipe(cssver()) //给css文件里引用文件加版本号（文件MD5）  .pipe(cssmin())  .pipe(gulp.dest('dist/css'));  })  //autoprefixer css浏览器前奏  gulp.task('autoFx', *function* () {  gulp.src('src/css/index.css')  .pipe(autoprefixer({  browsers: ['last 2 versions', 'Android >= 4.0'],  cascade: true, //是否美化属性值 默认：true 像这样：  //-webkit-transform: rotate(45deg);  // transform: rotate(45deg);  remove: true //是否去掉不必要的前缀 默认：true  }))  .pipe(gulp.dest('dist/css'));  });  //重命名  gulp.task('rename', *function* () {  return gulp.src(['dist/js/\*.js', 'dist/js/!\*.min.js']) // 指明源文件路径、并进行文件匹配  .pipe(sourcemaps.init()) // 执行sourcemaps  .pipe(rename({ suffix: '.min' })) // 重命名  .pipe(uglify({ preserveComments: 'some' })) // 使用uglify进行压缩，并保留部分注释  .pipe(sourcemaps.write('maps')) // 地图输出路径（存放位置）  .pipe(gulp.dest('dist/js')); // 输出路径  });  //文件清理  gulp.task('clean', *function* () {  return gulp.src(['dist/css/maps', 'dist/js/maps'], { read: false })  .pipe(clean());  });  // 复制文件  gulp.task('copy', *function* () {  // 如果有多个html可以传入'./src/\*.html'  gulp.src('./src/index.html')  .pipe(gulp.dest('./dist/'))  })  // 图片复制  gulp.task('image', *function* () {  gulp.src('src/images/\*.\*')  .pipe(gulp.dest('dist/images'))// 指定输出路径  .pipe(browserSync.reload({ // 不会对当前任务有影响,只是通知浏览器刷新  stream: true  }))  })  // serve任务仅仅是监测到dist目录下文件有变化就刷新浏览器  gulp.task('serve', *function* () {  browserSync({ // browserSync.init  server: {  baseDir: './dist' //指定服务器启动根目录  },  // server:true,  files: ['dist/\*.html', 'dist/css/\*.css', 'dist/js/\*.js'],  port: 8089  }, *function* (*err*, *bs*) {  *console*.log('正在运行..');  });  //监听任何文件变化，实时刷新页面  //gulp.watch("./src/\*.html").on('change', browserSync.reload);  //gulp.watch("./src/\*\*/\*.\*").on('change', browserSync.reload);  //gulp.watch("./src/\*\*/\*.less").on('change', browserSync.reload);  })  /\*  gulp.task('serve',function(){  // 使用browserSync的init方法，传入一个json对象，来进行一些初始化操作  browserSync.init({  // --server --files "sss"  server:'dist',// 这是指定一下启动的http服务的根目录  files:["src/\*.html","src/css/\*.css","src/js/\*.js"]// 指定的是想要监视的文件  });  gulp.watch('src/index.html',["html"]);  })  \*/  // 监视文件的变化,自动调用响应的任务  gulp.task('watch', *function* () {  gulp.watch('./src/\*.html', ['html'])  gulp.watch('./src/\*\*/\*.js', ['script'])  gulp.watch('./src/\*\*/\*.less', ['less'])  gulp.watch('./src/images/\*.\*', ['images']);  })  // 运行要同时执行 gulp watch 和 gulp serve  // 异步并发执行任务  // gulp.task('build', ['coffee', 'sass', 'image', 'html'])  gulp.task('default', ['serve', 'watch']); //定义默认任务 |

# webpack

Webpack可以做的事情

代码转换、文件优化、代码分割、自动刷新、代码校验、自动发布

深入浅出Webpack <http://webpack.wuhaolin.cn/>

## webpack的核心

|  |
| --- |
| 入口、出口、加载器、插件... |

## webpack的一些plugin，怎么使用webpack对项目进行优化

|  |
| --- |
| webpack插件归纳总结 <https://segmentfault.com/a/1190000016816813>  构建优化  1、减少编译体积 ContextReplacementPugin、IgnorePlugin、babel-plugin-import、babel-plugin-transform-runtime。  2、并行编译 happypack、thread-loader、uglifyjsWebpackPlugin开启并行  3、缓存 cache-loader、hard-source-webpack-plugin、uglifyjsWebpackPlugin开启缓存、babel-loader开启缓存  4、预编译 dllWebpackPlugin && DllReferencePlugin、auto-dll-webapck-plugin  性能优化  1、减少编译体积 Tree-shaking、Scope Hositing。  2、hash缓存 webpack-md5-plugin  3、拆包 splitChunksPlugin、import()、require.ensure |

## gulp和webpack的区别？(模块与流，CommonChunks抽出公共模块)

|  |
| --- |
|  |

## webpack打包原理

|  |
| --- |
|  |

## webpack打包插件

|  |
| --- |
|  |

## webpack优化构建速度

|  |
| --- |
|  |

## loader编写

|  |
| --- |
| loader就是一个node模块，它输出了一个函数。当某种资源需要用这个loader转换时，这个函数会被调用。并且，这个函数可以通过提供给它的this上下文访问Loader API。  reverse-txt-loader  定义  *module*.*exports* = *function*(*src*) {  //src是原文件内容（abcde），下面对内容进行处理，这里是反转  *var* result = src.split('').reverse().join('');  //返回JavaScript源码，必须是String或者Buffer  return `module.exports = '${result}'`;  }  使用  {    test: /\.txt$/,    use: [      {        './path/reverse-txt-loader'      }    ]  }, |

## Plugins

|  |
| --- |
| 使用范围更广，通常只需要require()然后添加到plugins数组中，且需要new一个 |

## 打包体积 优化思路

|  |
| --- |
| 1.提取第三方库或通过引用外部文件的方式引入第三方库  2.代码压缩插件UglifyJsPlugin  3.服务器启用gzip压缩  4.按需加载资源文件 require.ensure  5.优化devtool中的source-map  6.剥离css文件，单独打包  7.去除不必要插件，通常就是开发环境与生产环境用同一套配置文件导致  8.开发环境采用增量构建，启用热更新  9.开发环境不做无意义的工作如提取css计算文件hash等  10.配置devtool  11.选择合适的loader  12.个别loader开启cache 如babel-loader  13.第三方库采用引入方式  14.提取公共代码  15.优化构建时的搜索路径 指明需要构建目录及不需要构建目录  16.模块化引入需要的部分 |

## babel原理

|  |
| --- |
| ES6、7代码输入 -> babylon进行解析 -> 得到AST（抽象语法树）-> plugin用babel-traverse对AST树进行遍历转译 ->得到新的AST树->用babel-generator通过AST树生成ES5代码、  ====================  babylon 将 ES6/ES7 代码解析成 AST  babel-traverse 对 AST 进行遍历转译，得到新的 AST  新 AST 通过 babel-generator 转换成 ES5 |

## webpack优化

|  |
| --- |
| - 分离第三方库(依赖),比如引入`dll`  - 引入多进程编译,比如`happypack`  - 提取公共的依赖模块,比如`commonChunkPlugin`  - 资源混淆和压缩:比如`UglifyJS`  - 分离样式这些,减小`bundle chunk`的大小,比如`ExtractTextPlugin`  - GZIP 压缩,在打包的时候对资源对齐压缩,只要部署的服务器能解析即可..减少请求的大小  - 还有按需加载这些,一般主流的框架都有对应的模块懒加载方式.  - 至于`tree shaking`目前webpack3/4已经默认集成 |

## 组件封装

|  |
| --- |
| 目的：为了重用，提高开发效率和代码质量 注意：低耦合，单一职责，可复用性，可维护性 常用操作：  1.分析布局  2.初步开发  3.化繁为简  4.组件抽象 |

# scss less区别

|  |
| --- |
| SASS提供四个编译风格的选项：  　　\* nested：嵌套缩进的css代码，它是默认值。  　　\* expanded：没有缩进的、扩展的css代码。  　　\* compact：简洁格式的css代码。  　　\* compressed：压缩后的css代码。  scss 是在ruby环境处理，less引入lees.js处理，  scss支持@if @else @ for(){} @while() @each循坏，less不支持  scss @function() {} less不支持自定义函数 |

# less

|  |
| --- |
| 变量 @num: 10px;  混合（Mixins）  定义：.bordered（@num: 10px）{}  使用  #menu a {  color: #111;  .bordered;  }  .mixin (@a, @rest...) {  // @rest 表示 @a 之后的参数  // @arguments 表示所有参数  }  嵌套：选择器的嵌套  运算  @import  @{name} 插值 |

# sass

|  |
| --- |
| sass、less有哪些特性？  答：都是用动态语言编写样式  变量(scss为$, less为@，scss有局部作用域而less没有)、  计算：margin: (14px/2) // 7px  嵌套(选择器的嵌套、属性嵌套)、  混入@mixin name($num: 10px){} 使用@include name(20px)  选择器继承 %block {} @extend %block; // 代码重用 #或 .被替换成了 % 占位符选择器 %  #{name} 插值  & 引用父选择器  @import "path/filename.scss";  继承选择样式  @if 1 + 1 == 2 { border: 1px solid;}    @else {background-color: #fff;}  // for 循坏  @for $i from 1 to 10 {font-size: #{$i}px;}  // while 循坏  $i: 6;  @while $i > 0 {  .item-#{$i} { width: 2em \* $i;}  }  // each循坏  @each $member in a, b, c, d {  　　.#{$member} {  　　　　background-image: url("/image/#{$member}.jpg");  　　}  }  // 函数  @function double($n) {  　　@return $n \* 2;  }  width: double(5px);  .sidebar {  width: 300px;  @media screen and (orientation: landscape) {  width: 500px; }   }  颜色： rgba() reb() green(#002233) blue() min($color1, $color2)  HSL”所表示的是“H:色相”，“S：饱和度”，“L：亮度”  alpha($color) /opacity($color)：获取颜色透明度值  字符串函数  unquote($string)：删除字符串中的引号；  quote($string)：给字符串添加引号。  数字函数  percentage($value)：将一个不带单位的数转换成百分比值；  round($value)：将数值四舍五入，转换成一个最接近的整数；  ceil($value)：将大于自己的小数转换成下一位整数；  floor($value)：将一个数去除他的小数部分；  abs($value)：返回一个数的绝对值；  min($numbers…)：找出几个数值之间的最小值；  max($numbers…)：找出几个数值之间的最大值。  List函数  length($list)：返回一个列表的长度值；  nth($list, $n)：返回一个列表中指定的某个标签值  join($list1, $list2, [$separator])：将两个列给连接在一起，变成一个列表；  append($list1, $val, [$separator])：将某个值放在列表的最后；  zip($lists…)：将几个列表结合成一个多维的列表；  index($list, $value)：返回一个值在列表中的位置值。  Introspection函数  type-of($value)：返回一个值的类型  unit($number)：返回一个值的单位；  unitless($number)：判断一个值是否带有带位  comparable($number-1, $number-2)：判断两个值是否可以做加、减和合并 |