# Grunt

**\*** Grunt介绍  
 **\*** 中文主页 : **http://www.gruntjs.net/  
 \*** 是一套前端**\*\*自动化构建\*\***工具，一个基于nodeJs的命令行工具  
 **\*** 它是一个**\*\*任务运行器\*\***, 配合其丰富强大的**\*\*插件\*\*  
 \*** 常用功能:  
 **\* \*\*合并文件\*\***(js/css)  
 **\* \*\*压缩文件\*\***(js/css)  
 **\* \*\*语法检查\*\***(js)  
 **\* \*\*less/sass预编译处理\*\*   
 \*** 其它...  
**\*** 安装nodejs, 查看版本  
 ```  
 node -v  
 ```  
**\*** 创建一个简单的应用grunt\_test  
 ```  
 |- build----------构建生成的文件所在的文件夹  
 |- src------------源码文件夹   
 |- js---------------js源文件夹  
 |- css--------------css源文件夹  
 |- index.html-----页面文件  
 |- Gruntfile.js---grunt配置文件(注意首字母大写)  
 |- package.json---项目包配置文件  
 {  
 "name": "grunt\_test",  
 "version": "1.0.0"   
 }  
 ```   
**\*** 全局安装 grunt-cli  
 ```  
 npm install -g grunt-cli   
 ```  
**\*** 安装grunt  
 ```  
 npm install grunt --save-dev  
   
 ```  
**\*** 运行构建项目命令  
 ```  
 grunt //提示 Warning: Task "default" not found  
 ```  
**\*** 配置文件: Gruntfile.js  
 **\*** 此配置文件本质就是一个node函数类型模块  
 **\*** 配置编码包含3步:  
 **1.** 初始化插件配置  
 **2.** 加载插件任务  
 **3.** 注册构建任务  
 **\*** 基本编码:  
 ```  
 module.exports = function(grunt){  
 // 1. 初始化插件配置  
 grunt.initConfig({  
 //主要编码处  
 });  
 // 2. 加载插件任务  
 // grunt.loadNpmTasks('grunt-contrib-concat');  
 // 3. 注册构建任务  
 grunt.registerTask('default', []);  
 };  
 ```  
 **\*** 命令: grunt //提示成功, 但没有任何效果(还没有使用插件定义任务)  
**\*** Grunt插件介绍  
 **\*** grunt官网的插件列表页面 http://www.gruntjs.net/plugins   
 **\*** 插件分类:  
 **\*** grunt团队贡献的插件 : 插件名大都以contrib-开头  
 **\*** 第三方提供的插件 : 大都不以contrib-开头  
 **\*** 常用的插件:  
 **\*** grunt-contrib-clean——清除文件(打包处理生成的)  
 **\*** grunt-contrib-concat——合并多个文件的代码到一个文件中  
 **\*** grunt-contrib-uglify——压缩js文件  
 **\*** grunt-contrib-jshint——javascript语法错误检查；  
 **\*** grunt-contrib-cssmin——压缩/合并css文件  
 **\*** grunt-contrib-htmlmin——压缩html文件  
 **\*** grunt-contrib-imagemin——压缩图片文件(无损)  
 **\*** grunt-contrib-copy——复制文件、文件夹  
 **\*** grunt-contrib-watch——实时监控文件变化、调用相应的任务重新执行  
**\*** 合并js: 使用concat插件  
 **\*** 命令:  
 ```  
 npm install grunt-contrib-concat --save-dev  
 ```  
 **\*** 编码:  
 **\*** src/js/test1.js  
 ```  
 (function () {  
 function add(num1, num2) {  
 return num1 + num2;  
 }  
 console.log(add(10, 20));  
 })();  
 ```  
 **\*** src/js/test2.js  
 ```  
 (function () {  
 var arr = **[**2,3,4**]**.map(function (item, index) {  
 return item+1;  
 });  
 console.log(arr);  
 })();  
 ```  
 **\*** 配置: Gruntfile.js  
 **\*** 配置任务:  
 ```  
 concat: {  
 options: { //可选项配置  
 separator: ';' //使用;连接合并  
 },  
 build: { //此名称任意  
 src: ["src/js/***\*****.js"], //合并哪些js文件  
 dest: "build/js/built.js" //输出的js文件  
 }  
 }* ```  
 *\* 加载插件:* ```  
 *grunt.loadNpmTasks('grunt-contrib-concat');* ```  
 *\* 注册任务:* ```  
 *grunt.registerTask('default',* **[***'concat'***]***);* ```  
 *\* 命令:* ```  
 *grunt //会在build下生成一个built.js* ```  
*\* 压缩js: 使用uglify插件  
 \* 下载* ```  
 *npm install grunt-contrib-uglify --save-dev* ```  
 *\* 配置: Gruntfile.js  
 \* 配置任务:* ```  
 *pkg : grunt.file.readJSON('package.json'),  
 uglify : {  
 options: { //不是必须的  
 banner: '/****\****! <%= pkg.name %> - v<%= pkg.version %> - ' +  
 '<%= grunt.template.today("yyyy-mm-dd") %> \*/'  
 },  
 build: {  
 files: {  
 'build/js/built-<%=pkg.name%>-<%=pkg.version%>.min.js': **[**'build/js/built.js'**]**  }  
 }  
 }  
 ```  
 **\*** 加载任务:  
 ```  
 grunt.loadNpmTasks('grunt-contrib-uglify');  
 ```  
 **\*** 注册任务:  
 ```  
 grunt.registerTask('default', **[**'concat', 'uglify'**]**);  
 ```  
 **\*** 命令:   
 ```  
 grunt //会在build下生成一个压缩的js文件  
 ```  
**\*** js语法检查: 使用jshint插件  
 **\*** 命令:   
 ```  
 npm install grunt-contrib-jshint --save-dev  
 ```  
 **\*** 编码: .jshintrc  
 ```  
 {  
 "curly": true,  
 "eqeqeq": true,  
 "eqnull": true,  
 "expr" : true,  
 "immed": true,  
 "newcap": true,  
 "noempty": true,  
 "noarg": true,  
 "regexp": true,  
 "browser": true,  
 "devel": true,  
 "node": true,  
 "boss": false,  
   
 //不能使用未定义的变量  
 "undef": true,  
 //语句后面必须有分号  
 "asi": false,  
 //预定义不检查的全局变量  
 "predef": [ "define", "BMap", "angular", "BMAP\_STATUS\_SUCCESS"]  
 }  
 ```  
 **\*** 修改src/js/test1.js  
 ```  
 (function () {  
 function add(num1, num2) {  
 num1 = num1 + num3  
 return num1 + num2;  
 }  
 console.log(add(10, 20));  
 })();  
 ```  
 **\*** 配置 : Gruntfile.js  
 **\*** 配置任务:  
 ```  
 jshint : {  
 options: {  
 jshintrc : '.jshintrc' //指定配置文件  
 },  
 build : ['Gruntfile.js', 'src/js/\*.js'] //指定检查的文件  
 }  
 ```  
 **\*** 加载任务:  
 ```  
 grunt.loadNpmTasks('grunt-contrib-jshint');  
 ```  
 **\*** 注册任务:  
 ```  
 grunt.registerTask('default', ['concat', 'uglify', 'jshint']);  
 ```  
 **\*** 命令:   
 ```  
 grunt //提示变量未定义和语句后未加分号 -->修改后重新编译  
 ```  
**\*** 使用cssmin插件  
 **\*** 安装:  
 ```  
 npm install grunt-contrib-cssmin --save-dev  
 ```  
 **\*** 编码:   
 **\*** test1.css  
 ```  
 #box1 {  
 width: 100px;  
 height: 100px;  
 background: red;  
 }  
 ```  
 **\*** test2.css  
 ```  
 #box2 {  
 width: 200px;  
 height: 200px;  
 background: blue;  
 }  
 ```  
 **\*** index.html  
 ```  
 <link rel="stylesheet" href="build/css/output.min.css">  
 <div id="box1"></div>  
 <div id="box2"></div>  
 ```  
   
 **\*** 配置 : Gruntfile.js  
 **\*** 配置任务:  
 ```  
 cssmin:{  
 options: {  
 shorthandCompacting: false,  
 roundingPrecision: -1  
 },  
 build: {  
 files: {  
 'build/css/output.min.css': ['src/css/\*.css']  
 }  
 }  
 }  
 ```  
 **\*** 加载任务:  
 ```  
 grunt.loadNpmTasks('grunt-contrib-cssmin');  
 ```  
 **\*** 注册任务:  
 ```  
 grunt.registerTask('default', ['concat', 'uglify', 'jshint', 'cssmin']);  
 ```  
 **\*** 命令:   
 ```  
 grunt //在build/css/下生成output.min.css  
 ```  
**\*** 使用watch插件（真正实现自动化）   
 **\*** 命令: npm install grunt-contrib-watch --save-dev  
 **\*** 配置 : Gruntfile.js  
   
 **\*** 配置任务:  
 ```  
 watch : {  
 scripts : {  
 files : ['src/js/\*.js', 'src/css/\*.css'],  
 tasks : ['concat', 'jshint', 'uglify', 'cssmin'],  
 options : {spawn : false}   
 }  
 }  
 ```  
 **\*** 加载任务:  
 ```  
 grunt.loadNpmTasks('grunt-contrib-watch');  
 ```  
 **\*** 注册任务:  
 ```  
 grunt.registerTask('default', ['concat', 'uglify', 'jshint', 'watch']);  
 改进：grunt.registerTask('myWatch', ['default','watch']);  
 ```  
 **\*** 命令:   
 ```  
 grunt //控制台提示watch已经开始监听, 修改保存后自动编译处理  
 ```

# Gulp

**\*** Gulp介绍  
 **\*** 中文主页: **http://www.gulpjs.com.cn/  
 \*** gulp是与grunt功能类似的**\*\*前端项目构建\*\***工具, 也是基于Nodejs的自动**\*\*任务运行器\*\*  
 \*** 能自动化地完成 javascript/coffee/sass/less/html/image/css 等文件的  
 合并、压缩、检查、监听文件变化、浏览器自动刷新、测试等任务  
 **\*** gulp更高效(异步多任务), 更易于使用, 插件高质量  
**\*** 安装 nodejs, 查看版本: node -v  
**\*** 创建一个简单的应用gulp\_test  
 ```  
 |- dist  
 |- src  
 |- js  
 |- css  
 |- less  
 |- index.html  
 |- gulpfile.js-----gulp配置文件  
 |- package.json  
 {  
 "name": "gulp\_test",  
 "version": "1.0.0"  
 }   
 ```  
**\*** 安装gulp:  
 **\*** 全局安装gulp  
 ```  
 npm install gulp -g  
 ```  
 **\*** 局部安装gulp  
 ```  
 npm install gulp --save-dev  
 ```  
 **\*** 配置编码: gulpfile.js  
 ```  
 //引入gulp模块  
 var gulp = require('gulp');  
 //定义默认任务  
 gulp.task('任务名', function() {  
 // 将你的任务的任务代码放在这  
 });  
 gulp.task('default', ['任务'])//异步执行  
 ```  
 **\*** 构建命令:   
 ```  
 gulp  
 ```  
**\*** 使用gulp插件  
 **\*** 相关插件:  
 **\*** gulp-concat : 合并文件(js/css)  
 **\*** gulp-uglify : 压缩js文件  
 **\*** gulp-rename : 文件重命名  
 **\*** gulp-less : 编译less  
 **\*** gulp-clean-css : 压缩css  
 **\*** gulp-livereload : 实时自动编译刷新  
 **\*** 重要API  
 **\*** gulp.src(filePath/pathArr) :   
 **\*** 指向指定路径的所有文件, 返回文件流对象  
 **\*** 用于读取文件  
 **\*** gulp.dest(dirPath/pathArr)  
 **\*** 指向指定的所有文件夹  
 **\*** 用于向文件夹中输出文件  
 **\*** gulp.task(name, **[**deps**]**, fn)   
 **\*** 定义一个任务  
 **\*** gulp.watch()   
 **\*** 监视文件的变化  
 **\*** 处理js  
 **\*** 创建js文件  
 **\*** src/js/test1.js  
 ```  
 (function () {  
 function add(num1, num2) {  
 var num3 = 0;  
 num1 = num2 + num3;  
 return num1 + num2;  
 }  
 console.log(add(10, 30));  
 })();  
 ```  
 **\*** src/js/test2.js  
 ```  
 (function () {  
 var arr = [2,3,4].map(function (item, index) {  
 return item+1;  
 });  
 console.log(arr);  
 })();  
 ```  
 **\*** 下载插件:  
 ```  
 npm install gulp-concat gulp-uglify gulp-rename --save-dev  
 ```  
 **\*** 配置编码  
 ```  
 var concat = require('gulp-concat');  
 var uglify = require('gulp-uglify');  
 var rename = require('gulp-rename');  
   
 gulp.task('minifyjs', function() {  
 return gulp.src('src/js/\*.js') //操作的源文件  
 .pipe(concat('built.js')) //合并到临时文件   
 .pipe(gulp.dest('dist/js')) //生成到目标文件夹  
 .pipe(rename({suffix: '.min'})) //重命名   
 .pipe(uglify()) //压缩  
 .pipe(gulp.dest('dist/js'));  
 });  
   
 gulp.task('default', **[**'minifyjs'**]**);  
 ```  
 **\*** 页面引入js浏览测试 : index.html  
 ```  
 <script type="text/javascript" src="dist/js/built.min.js"></script>  
 ```  
 **\*** 打包测试: gulp  
 **\*** 处理css  
 **\*** 创建less/css文件  
 **\*** src/css/test1.css  
 ```  
 #div1 {  
 width: 100px;  
 height: 100px;  
 background: green;  
 }  
 ```  
 **\*** src/css/test2.css  
 ```  
 #div2 {  
 width: 200px;  
 height: 200px;  
 background: blue;  
 }  
 ```  
 **\*** src/less/test3.less  
 ```  
 @base: yellow;  
 .index1 { color: @base; }  
 .index2 { color: green; }  
 ```  
 **\*** 下载插件:  
 ```  
 npm install gulp-less gulp-clean-css --save-dev   
 ```  
 **\*** 配置编码  
 ```  
 var less = require('gulp-less');  
 var cleanCSS = require('gulp-clean-css');  
   
 //less处理任务  
 gulp.task('lessTask', function () {  
 return gulp.src('src/less/\*.less')  
 .pipe(less())   
   
 .pipe(gulp.dest('src/css'));  
 })  
 //css处理任务, 指定依赖的任务  
 gulp.task('cssTask',**[**'lessTask'**]**, function () {  
   
 return gulp.src('src/css/\*.css')  
 .pipe(concat('built.css'))  
 .pipe(gulp.dest('dist/css'))  
 .pipe(rename({suffix: '.min'}))  
 .pipe(cleanCSS({compatibility: 'ie8'}))  
 .pipe(gulp.dest('dist/css'));  
 });  
   
 gulp.task('default', **[**'minifyjs', 'cssTask'**]**);  
 ```  
 **\*** 页面引入css浏览测试 : index.html  
 ```  
 <link rel="stylesheet" href="dist/css/built.min.css">  
 <div id="div1" class="index1">div1111111</div>  
 <div id="div2" class="index2">div2222222</div>  
 ```  
 **\*** 打包测试: gulp  
 **\*** 处理html  
 **\*** 下载插件:  
 ```  
 npm install gulp-htmlmin --save-dev  
 ```  
 **\*** 配置编码  
 ```  
 var htmlmin = require('gulp-htmlmin');  
 //压缩html任务  
 gulp.task('htmlMinify', function() {  
 return gulp.src('index.html')  
 .pipe(htmlmin({collapseWhitespace: true}))  
 .pipe(gulp.dest('dist'));  
 });  
 gulp.task('default', ['minifyjs', 'cssTask', 'htmlMinify']);  
 ```  
 **\*** 修改页面引入  
 ```  
 <link rel="stylesheet" href="css/built.min.css">  
 <script type="text/javascript" src="js/built.min.js"></script>  
 ```  
 **\*** 打包测试: gulp   
 **\*** 自动编译  
 **\*** 下载插件  
 ```  
 npm install gulp-livereload --save-dev  
 ```  
 **\*** 配置编码:  
 ```  
 var livereload = require('gulp-livereload');  
   
 //所有的pipe  
 .pipe(livereload());  
   
 gulp.task('watch', **[**'default'**]**, function () {   
 //开启监视  
 livereload.listen();  
 //监视指定的文件, 并指定对应的处理任务  
 gulp.watch('src/js/\*.js', **[**'minifyjs'**]**)  
 gulp.watch(**[**'src/css/***\*****.css','src/less/****\****.less'**]**, **[**'cssTask','lessTask'**]**);  
 });  
 ```  
   
 **\*** 热加载(实时加载)  
 **\*** 下载插件：gulp-connect  
 ```  
 1、 npm install gulp-connect --save-dev  
 2、 注册 热加载的任务 server，注意依赖build任务   
 3、注册热加载的任务  
 //配置加载的选项  
 connect.server({  
 root : 'dist/',//提供服务的根路径  
 livereload : true,//是否实时刷新  
 port : 5000//开启端口号  
 });  
 // 自动开启链接  
 open('http://localhost:5000');//npm install open --save-dev  
 // 监视目标文件  
 gulp.watch('src/js/\*.js', ['js']);  
 gulp.watch(['src/css/\*.css', 'src/css/\*.less'], ['cssMin', 'less']);  
 ```  
   
 **\*** 扩展  
 **\*** 打包加载gulp插件  
 **\*** 前提：将插件下载好。  
 **\*** 下载打包插件： gulp-load-plugins  
 **\*** npm install gulp-load-plugins --save-dev  
 **\*** 引入： var $ = require('gulp-load-plugins')();！！！引入的插件是个方法，必须记住调用。  
 **\*** 神来之笔：其他的插件不用再引入了  
 **\*** 使用方法：  
 ```  
 **\*** 所有的插件用 $ 引出，其他插件的方法名统一为插件的功能名字(即插件名字的最后一部分)：如：concat,connect,cssmin...  
 gulp.task('lib', function() {  
 gulp.src('bower\_components/\*\*/\*.js')  
 .pipe(gulp.dest(app.devPath + 'vendor'))  
 .pipe(gulp.dest(app.prdPath + 'vendor'))  
 .pipe($.connect.reload());  
 });  
  
 ```

# webpack

**## webpack快速入门教程  
###1、了解Webpack相关** \* 什么是webpack  
 \* Webpack是一个模块打包器(bundler)。  
 \* 在Webpack看来, 前端的所有资源文件(js/json/css/img/less/...)都会作为模块处理  
 \* 它将根据模块的依赖关系进行静态分析，生成对应的静态资源  
 \* 理解Loader  
 \* Webpack 本身只能加载JS/JSON模块，如果要加载其他类型的文件(模块)，就需要使用对应的loader 进行转换/加载  
 \* Loader 本身也是运行在 node.js 环境中的 JavaScript 模块  
 \* 它本身是一个函数，接受源文件作为参数，返回转换的结果  
 \* loader 一般以 xxx-loader 的方式命名，xxx 代表了这个 loader 要做的转换功能，比如 json-loader。  
 \* 配置文件(默认)  
 \* webpack.config.js : 是一个node模块，返回一个 json 格式的配置信息对象  
 \* 插件  
 \* 插件件可以完成一些loader不能完成的功能。  
 \* 插件的使用一般是在 webpack 的配置信息 plugins 选项中指定。  
 \* CleanWebpackPlugin: 自动清除指定文件夹资源  
 \* HtmlWebpackPlugin: 自动生成HTML文件并  
 \* UglifyJSPlugin: 压缩js文件  
   
**###2、学习文档 :   
 \*** webpack官网: **http://webpack.github.io/  
 \*** webpack3文档(英文): **https://webpack.js.org/  
 \*** webpack3文档(中文): **https://doc.webpack-china.org/  
###3、开启项目  
 \*** 初始化项目：  
 **\*** 生成package.json文件  
 **\*** ```   
 {  
 "name": "webpack\_test",  
 "version": "1.0.0"  
 }   
 ```  
 **\*** 安装webpack  
 **-** npm install webpack -g //全局安装  
 **-** npm install webpack --save-dev //局部安装  
**###4、编译打包应用  
 \*** 创建入口src/js/ : entry.js  
 **-** document.write("entry.js is work");  
 **\*** 创建主页面: dist/index.html  
 **-** <script type="text/javascript" src="bundle.js"></script>  
 **\*** 编译js  
 **-** webpack src/js/entry.js dist/bundle.js   
 **\*** 查看页面效果  
**###5、添加js/json文件  
 \*** 创建第二个js: src/js/math.js  
 ```   
 export function square(x) {  
 return x \* x;  
 }  
   
 export function cube(x) {  
 return x \* x \* x;  
 }  
 ```  
 **\*** 创建json文件: src/json/data.json  
 ```  
 {  
 "name": "Tom",  
 "age": 12  
 }  
 ```  
 **\*** 更新入口js : entry.js  
 ```  
 import {cube} from './math'  
 import data from '../json/data.json'  
 //注意data会自动被转换为原生的js对象或者数组  
 document.write("entry.js is work <br/>");  
 document.write(cube(2) + '<br/>');  
 document.write(JSON.stringify(data) + '<br/>')  
 ```  
 **\*** 编译js:  
 ```  
 webpack src/js/entry.js dist/bundle.js  
 ```  
 **\*** 查看页面效果  
**###6、使用webpack配置文件  
 \*** 创建webpack.config.js  
 ```  
 const path = require('path'); //path内置的模块，用来设置路径。  
   
 module.exports = {  
 entry: './src/js/entry.js', // 入口文件  
 output: { // 输出配置  
 filename: 'bundle.js', // 输出文件名  
 path: path.resolve(\_\_dirname, 'dist') //输出文件路径配置  
 }  
 };  
 ```  
 **\*** 配置npm命令: package.json  
 ```  
 "scripts": {  
 "build": "webpack"  
 },  
 ```  
 **\*** 打包应用  
 ```  
 npm run build  
 ```  
**###7、打包css和图片文件  
 \*** 安装样式的loader  
 ```  
 npm install css-loader style-loader --save-dev  
 npm install file-loader url-loader --save-dev  
 补充：url-loader是对象file-loader的上层封装，使用时需配合file-loader使用。  
 ```  
 **\*** 配置loader  
 ```  
 module: {  
 rules: [  
 {  
 test: /\.css$/,  
 use: [  
 'style-loader',  
 'css-loader'  
 ]  
 },  
 {  
 test: /\.(png|jpg|gif)$/,  
 use: [  
 {  
 loader: 'url-loader',  
 options: {  
 limit: 8192   
 }  
 }  
 ]  
 }  
 ]  
 }  
 ```  
 **\*** 向应用中添加2张图片:  
 **\*** 小图: img/logo.png  
 **\*** 大图: img/big.jpg  
   
 **\*** 创建样式文件: src/css/test.css  
 ```  
 body {  
 background: url('../img/logo.jpg')  
 }  
 ```  
 **\*** 更新入口js : entry.js  
 **-** import '../css/test.css'  
 **\*** 添加css样式  
  
  *#box1{  
 width: 300px;  
 height: 300px;  
 background-image: url("../image/logo.jpg");  
 }  
 #box2{  
 width: 300px;  
 height: 300px;  
 background-image: url("../image/big.jpg");  
 }* **\*** index.html添加元素  
   
  *<div id="box1"></div>  
 <div id="box2"></div>* **\*** 执行打包命令：  
 ```  
 npm run build  
 ```  
 **\*** 发现问题：  
 **\*** 大图无法打包到entry.js文件中，index.html不在生成资源目录下。  
 **\*** 页面加载图片会在所在目录位置查找，导致页面加载图片时候大图路径无法找到  
 **\*** 解决办法：  
 **\*** 使用publicPath : 'dist/js/' //设置为index.html提供资源的路径,设置完后找所有的资源都会去当前目录下找。  
 **\*** 将index.html放在dist/js/也可以解决。  
**###8、自动编译打包  
 \*** 利用webpack开发服务器工具: webpack-dev-server  
 **\*** 下载  
 **-** npm install --save-dev webpack-dev-server  
 **\*** webpack配置  
 devServer: {  
 contentBase: './dist'  
 },  
 **\*** package配置  
 **-** "start": "webpack-dev-server --open"  
 **\*** 编译打包应用并运行  
 **-** npm start  
**###9、使用webpack插件  
 \*** 常用的插件  
 **\*** 使用html-webpack-plugin根据模板html生成引入script的页面  
 **\*** 使用clean-webpack-plugin清除dist文件夹  
   
 **\*** 下载  
 ```  
 npm install --save-dev html-webpack-plugin clean-webpack-plugin  
 ```  
 **\*** webpack配置  
  
  *const HtmlWebpackPlugin = require('html-webpack-plugin'); //自动生成html文件的插件* const CleanWebpackPlugin = require('clean-webpack-plugin'); //清除之前打包的文件   
 plugins: **[** new HtmlWebpackPlugin({template: './index.html'}),  
 new CleanWebpackPlugin(**[**'dist'**]**),  
 **]  
  
 \*** 创建页面: index.html  
  
  *<!DOCTYPE html>  
 <html lang="en">  
 <head>  
 <meta charset="UTF-8">  
 <title>webpack test</title>  
 </head>  
 <body>  
 <div id="app"></div>  
 <!--打包文件将自动通过script标签注入到此处-->  
 </body>  
 </html>* **\*** 打包运行项目  
 ```  
 npm run build  
 npm start  
 ```