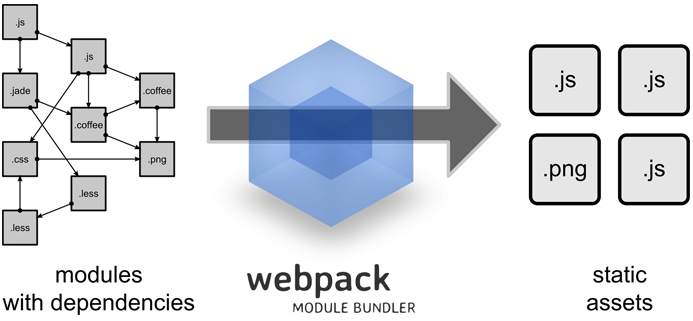
**webpack+ES6+less开发环境搭建（附带视频教程）**

 资料来源：https://cnodejs.org/topic/571f4e785a26c4a841ecbd7c

**webpack是什么**

Webpack 是一个模块打包器。它将根据模块的依赖关系进行静态分析，然后将这些模块按照指定的规则生成对应的静态资源。 

百度网盘视频下载：<http://pan.baidu.com/s/1jIPJucu> 百度网盘在线观看：[http://pan.baidu.com/share/link?shareid=1645979812&uk=2469628767&fid=988128759555711](http://pan.baidu.com/share/link?shareid=1645979812&uk=2469628767&fid=988128759555711" \t "_blank)

**基本命令**

webpack // 最基本的启动webpack的方法

webpack -w // 提供watch方法；实时进行打包更新

webpack -p // 对打包后的文件进行压缩

webpack -d // 提供source map，方便调式代码

**全局安装**

# npm install webpack -g

项目安装：

# 进入项目目录

# 确定已经有 package.json，没有就通过 npm init 创建

# 安装 webpack 依赖

# npm install webpack --save-dev

**使用ES6**

安装 babel-loader：

# npm install babel-loader --save-dev

安装转码规则：

# npm install babel-preset-es2015 --save-dev

ES7不同阶段语法提案的转码规则（共有4个阶段），选装一个

# npm install --save-dev babel-preset-stage-0

# npm install --save-dev babel-preset-stage-1

# npm install --save-dev babel-preset-stage-2

# npm install --save-dev babel-preset-stage-3

参考格式：

{

test: /\.js$/,

loader: 'babel?presets[]=es2015,presets[]=stage-0'

}

**编译css**

安装css-loader：

# npm install css-loader --save-dev

安装style-loader

# npm install style-loader --save-dev

参考格式：

{

test: /\.css$/,

loaders: ['style', 'css', 'autoprefixer']

}

**编译less**

# npm install less --save-dev

安装less-loader：

# npm install less-loader --save-dev

参考格式：

{

test: /\.less/,

loaders: ['style', 'css', 'autoprefixer', 'less'],

}

**使用autoprefixer自动补上浏览器兼容**

# npm install autoprefixer-loader --save-dev

参考格式：

{

test: /\.css$/,

loaders: ['style', 'css', 'autoprefixer']

}, {

test: /\.less/,

loaders: ['style', 'css', 'autoprefixer', 'less'],

}

**编译文件**

安装file-loader：

# npm install file-loader --save-dev

参考格式：

{

test: /\.(eot|woff|svg|ttf|woff2|gif)(\?|$)/,

loader: 'file-loader?name=[hash].[ext]'

}

**编译图片**

# npm install url-loader --save-dev

参考格式：

{

test: /\.(png|jpg)$/,

loader: 'url?limit=1200&name=[hash].[ext]'

}

**编译JSX**

# npm install jsx-loader --save-dev

# npm install babel-preset-react --save-dev

参考格式：

{

test: /\.jsx$/,

loaders: ['jsx', 'babel?presets[]=es2015,presets[]=stage-0,presets[]=react']

}

**示例源码**

在项目目录下，新建一个webpack.config.js文件，把下面代码复制进去，然后在新建一个app.js和index.js文件，写上console.log('你好世界');

执行命令：webpack 然后找到build目录就看到编译后的文件了

var webpack = require('webpack');

module.exports = {

entry: {

app: './app', //编译的入口文件

index: './index', //编译的入口文件

},

output: {

publicPath: '/build/', //服务器根路径

path: './build', //编译到当前目录

filename: '[name].js' //编译后的文件名字

},

module: {

loaders: [{

test: /\.js$/,

loader: 'babel?presets=es2015'

}, {

test: /\.css$/,

loaders: ['style', 'css', 'autoprefixer']

}, {

test: /\.less/,

loaders: ['style', 'css', 'autoprefixer', 'less'],

}, {

test: /\.(eot|woff|svg|ttf|woff2|gif)(\?|$)/,

loader: 'file-loader?name=[hash].[ext]'

}, {

test: /\.(png|jpg)$/,

loader: 'url?limit=1200&name=[hash].[ext]' //注意后面那个limit的参数，当你图片大小小于这个限制的时候，会自动启用base64编码图片

}

]

},

plugins: [

new webpack.optimize.CommonsChunkPlugin('common.js') //将公用模块，打包进common.js

],

resolve: {

extensions: ['', '.js', '.jsx'] //后缀名自动补全

}

};