此文：本人撰写

首先，清楚browserify是什么：其实就是一个nodejs的工具，这个工具的运用，客户端（前端开发人员）只需要安装nodejs及browserify插件就可以使用。

第一：安装（假设nodejs已经安装了），然后再安装browserify

命令行敲入： npm install –g browserify 即可

查看： 如果敲入 browserify –v

出现版本号就表示安装成功！ 哈哈！下面可以开始browserify的应用了

其实很简单，就是我在不要任何js插件的情况下，对一堆js文件相互依赖，就是在一个a.js中写 var b= require(‘./b.js’);就是这样的。但是要注意一点，此时的b.js中应该有module.exports = sssss ; 丢出去（这个丢跟seajs或者requirejs一样的写法）。

具体下面剧烈说明：

如在我c:\users\wangduo\desktop\connector\module文件夹下面有三个js文件

：connector.js, longConnector.js, websocketConnector.js.

在connector.js中依赖了longConnector.js, websocketConnector.js两个模块，

Var longConnector = require(‘./ longConnector.js’); //此处的后缀’.js’可以不用写

Var websocketConnector = require(‘./ websocketConnector.js’);

此时，如果不做任何处理，直接在一个html中引入connector.js肯定是要报错的，因为里面的模块拿不到。

但是我们可以借助browerify将connector.js 运行一遍，将里面的依赖关系都进行实现，然后生成一个新的js文件，然后将这个新的js文件引入到html的页面即可。

具体： 命令行敲入：

进入到module文件夹

C:\users\wangduo\desktop\connector\module> browserify connector.js –o result.js

此时就会在这个module文件夹下面生成一个result.js文件，这个文件就是前面三个的综合。只要在html中引入result.js文件即可。简单吧！

另外：添加两个功能：

第一：可以讲result.js进一步压缩(用uglifyjs可以完成)

第二：可以watch… result.js。一旦result.js或者依赖的js改变后就让其同步更新

压缩们就用uglifyjs即可

现在说说watch

这个要依赖于grunt, grunt这个插件本人认为他是一个大的node的插件容器。里面有很多具体的功能插件，根据grunt里面的一个package.json文件中的devDependencies这个属性配置具体的下载项。为了简单起见，我以后不用去下载了，直接将我的一个node\_modules（这个文件在内外桌面，就是已经下载好的grunt，里面的所有合并，watch，压缩等插件都集成在里面了）文件夹拷贝到js文件的同级即可，这一步搞定之后，我只要配置一个Gruntfile.js文件才可以运行grunt,因为Gruntfile.js这个文件中有对grunt的运行配置，包括运行指令的配置等，下面我写一个简单的watch…监听每个子文件的改变，然后实时编译，编译之后重新用browserify引入到入口文件，生成一个结果文件。页面只需要引入结果文件即可，就如同上面的对connector.js, longConnector.js, websocketConnector.js这三个文件进行watch,而页面引入的是result.js。

具体的Gruntfile.js如下：（这个文件也是和js文件同级的）

在这里里面我需要声明一下（我用的browserify是grunt-browserify，而不是单独的browserify，功能与单独的browserify完全一样，只是因为grunt-browserify是在grunt集成的而且在node\_modules的文件下已经下载了，方便与Gruntfile.js中的引入（事件加载），（如果自己喜欢也可以用单独的browserify来手动每次引入））

// 这个文件其实是在指令中敲入grunt的时候载入的

Module.exports = function(grunt){

// 此处配置命令

Grunt.initConfig({

// 配置watch命令，用grunt-contrib-watch来执行的

Watch: {

File: [‘connect.js’], //文件的路径都是依据Gruntfile.js的

Tasks: [‘browserify’]

},

Browserify: {

‘result.js’: [‘connect.js’]

},

// 可以配置个压缩指令（用uglifyjs）

Uglify: {

// my\_target这个名字随便取得

My\_target:{

Files: {

‘result.min.js’: ‘result.js’// 将result.js进行压缩

}

}

}

});

// 下面定义下task的加载

Grunt.loadNpmTasks(‘grunt-contrib-uglify’);

Grunt.loadNpmTasks(‘grunt-contrib-watch’)

Grunt.loadNpmTasks(‘grunt-browserify’);

// 注册指令

Grunt.registerTask(‘default’, [‘watch’]);//这个是默认指令就是一旦在cmd中敲入grunt就执行watch

Grunt.registerTask(‘build’, [‘browserify’, ‘uglify’]); //这个就是在敲入grunt bulid的时候先执行browserify然后再压缩一次，生成最后的result.min.js文件。

}

就这些！哈哈哈其实很简单吧！

http://www.cnblogs.com/yexiaochai/p/3603389.html