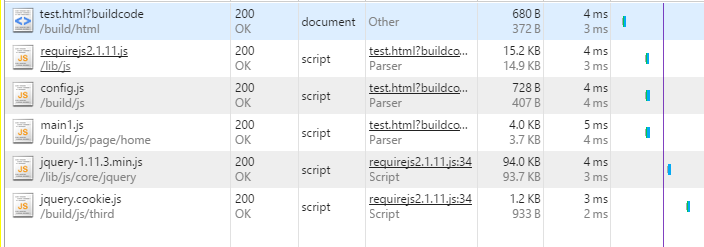
# 什么是优化？

## 1.1优化前



## 1.2优化后



## 1.3对比

|  |  |  |
| --- | --- | --- |
| 指标 | 优化前 | 优化后 |
| 资源请求数 | 15 | 6 |
| 资源大小 | 196KB | 116KB |
| 资源加载完成时间 | 185ms | 153ms |
| domReady时间 | 120ms | 101ms |
| window.onload时间 | 120ms | 101ms |

特别说明：此demo非常简单，没有多少代码量，所以除了“资源请求数”指标，其他指标对比差异不是特别明显。

## 1.4总结

requirejs优化包括以下2点：

1. 根据模块依赖合并代码，减少网络请求；
2. 压缩代码，减少文件体积；

# 前期准备

在了解requirejs的优化前，先得了解requirejs的基础知识。

requires官网：<http://www.requirejs.cn/>

请打开[requirejs优化资源库](https://github.com/zmrdlb/grunt-requirejs-Intelligent-optimization)，下载代码，本例子中以此代码的工程目录结构为例。更多具体的介绍也可以看github里的wiki。

# 官方优化

requirejs优化官方说明：<http://www.requirejs.cn/docs/optimization.html>

r.js下载：<http://www.requirejs.cn/docs/download.html#rjs>

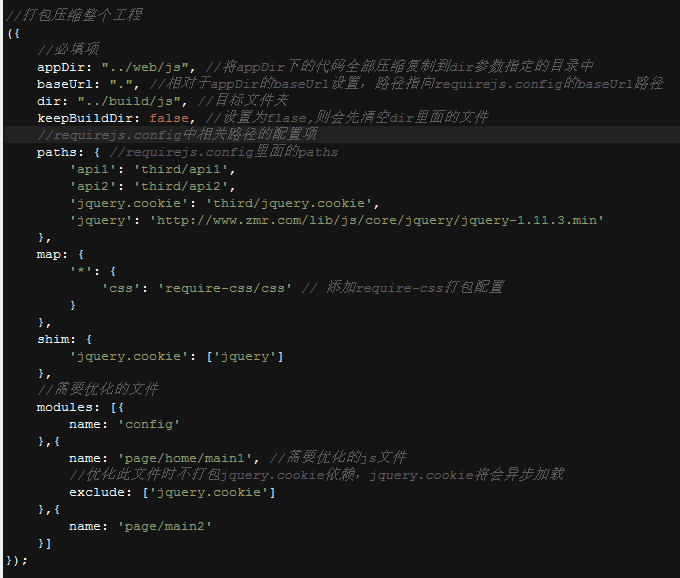
build.js说明：<https://github.com/jrburke/r.js/blob/master/build/example.build.js>

requirejs的优化脚本可以选择多种环境来执行，本文选择nodejs环境，因为nodejs类似于我们浏览器的js脚本，便于理解。

## 3.1使用说明

1) 下载r.js到项目工程目录里，本例中在tools文件夹里；

2) 在tools里建立build.js文件，里面编写优化配置项。下面只给出一些常用的配置例子：



1. 运行

打开node.js命令对话框，指定路径到tools里

C:\Users\mingrui\AppData\Roaming\Tencent\Users\592044573\QQ\WinTemp\RichOle\7UFH~4WQV5$D07}E9L}V8ME.png

执行优化：

C:\Users\mingrui\AppData\Roaming\Tencent\Users\592044573\QQ\WinTemp\RichOle\0P(@(R2YO_SPKM7TLH7Z[4O.png

可以看到运行过程：



可以看出已经优化了page/home/main1.js和page/main2.js

## 3.2使用grunt来优化

grunt也是基于nodejs平台的，它是一个轻量级的自动化工具，帮你快速完成复杂的工作。

快速安装：<http://www.gruntjs.net/getting-started>

为何使用grunt？

1. 基于nodejs平台，对于前端人员容易上手；
2. 发展迅速，功能健全；
3. 一个工程在上线前不可能只对Js进行优化，还包括css，image文件复制，多个文件合并等操作，这些功能在grunt上面都有。所以，多需求的自动化工具可以统一使用一个grunt。

grunt上面有requirejs的优化工具：<https://www.npmjs.com/package/grunt-contrib-requirejs>

使用的就是requirejs官方提供的优化脚本r.js。配置项和官方说明build.js一样，只不过放在grunt中。

如何使用grunt在此先不讲解。

# 问题

requirejs优化我们可以灵活配置，但是我们如果要优化100个js文件，那就得在modules里写100项，这个是不是很头疼？

# 5.reuqirejs智能打包工具

对于上面提出的问题，我对requirejs的优化进行了一次封装，智能读取待优化的文件夹里的文件，提高工作效率。在此提出了“[requirejs智能打包工具](https://github.com/zmrdlb/grunt-requirejs-Intelligent-optimization/wiki/1.-%E8%83%8C%E6%99%AF%E7%9F%A5%E8%AF%86)”

具体说明可见里面的wiki说明及资源里的tools/Gruntfile.js

# 6.选择性

现在requirejs的优化方式大概有两种：

1. 官方方式，可直接node运行或grunt运行；
2. 本文提出的requirejs智能打包工具；

到底选择哪个，我给出以下对比，大家可以自行选择。

|  |  |  |
| --- | --- | --- |
|  | 优点 | 缺点 |
| 官方方式 | 直接使用未封装的底层配置，可扩展性强，容易阅读； | 待优化的文件需手动一一配置，对于大项目，这种方式很痛苦 |
| requirejs智能打包工具 | 有一定的学习成本；因为对官方方式进行了封装，所以可扩展性有一定的限制； | 智能读取待优化文件夹里的文件，无需手动一一编写 |