

【前端构建】WebPack实例与前端性能优化

计划把微信的文章也搬一份上来。

这篇主要介绍一下我在玩Webpack过程中的心得。通过实例介绍WebPack的安装，插件使用及加载策略。感受构建工具给前端优化工作带来的便利。

壹 | Fisrt

```
<script src="util.js"></script>
<script src="dialog.js"></script>
<script>
  org.CoolSite.Dialog.init({ /* 传入配置 */ });
</script>
```

曾几何时我们的JS是这样引入页面的

曾几何时，我们是如上图的方式引入JS资源的，相信现在很少遇见了。近年来Web前端开发领域朝着规范开发的方向演进。体现在以下两点：

- MVC研发构架。多多益处（逻辑清晰，程序注重数据与表现分离，可读性强，利于规避和排查问题...）
- 构建工具层出不穷。多多益处（提升团队协作，以及工程运维，避免人工处理琐碎而重复的工作）
 - 模块化开发
 - 将前端性能优化理论落地，代码压缩，合并，缓存控制，提取公共代码等
 - 其他的还包括比如你可以用ES 6 或CoffeeScript写源码，然后构建出浏览器支持的ES5

所以，前端这么好玩，如果还有项目没有前后端分离的话，真的是守旧过头了。

主流构建工具

市面上有许多构建工具，包括Grunt、Gulp、browserify等，这些和WebPack都是打包工具。但WebPack同时也具备以下特点：

- 相比Grunt，WebPack除了具备丰富的插件外，同时带有一套加载（Loader）系统。使它支持多种规范的加载方式，包括ES6、CommonJS、AMD等方式，这是Grunt、Gulp所不具备的。
- 从代码混淆的角度来看，WebPack更加的极致
- 代码分片为处理单元（而不是文件），使得文件的分片更为灵活。

P.S.此处只做简单的比较，不论孰优孰劣。其实工具都能满足需求，关键是看怎么用，工具的使用背后是对前端性能优化的理解程度。

36

推荐

3

反对

返回顶部

贰 | Secor

WebPack安装与使用

WebPack运行在 NodeJS之下，并且它及其插件都是使用NPM（NodeJS的包管理工具）管理。

1. 安装Node及NPM。到NodeJS官网安装包，安装即可
2. 全局安装WebPack。联网情况下，执行命令行 \$npm install webpack -g 即可。

此至即可使用WebPack了，到WebPack官网去按着Get start（<http://webpack.github.io/docs/tutorials/getting-started/>）的步骤来，感受一个最简单的构建过程。

然而要把WebPack用好，只是跑起来是远远不够的。

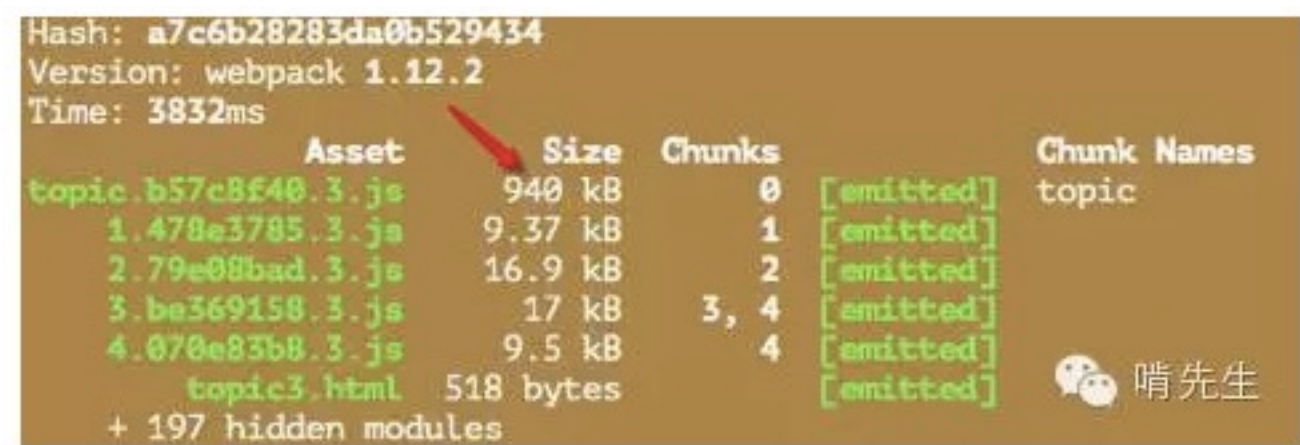
叁 | Third

WebPack插件

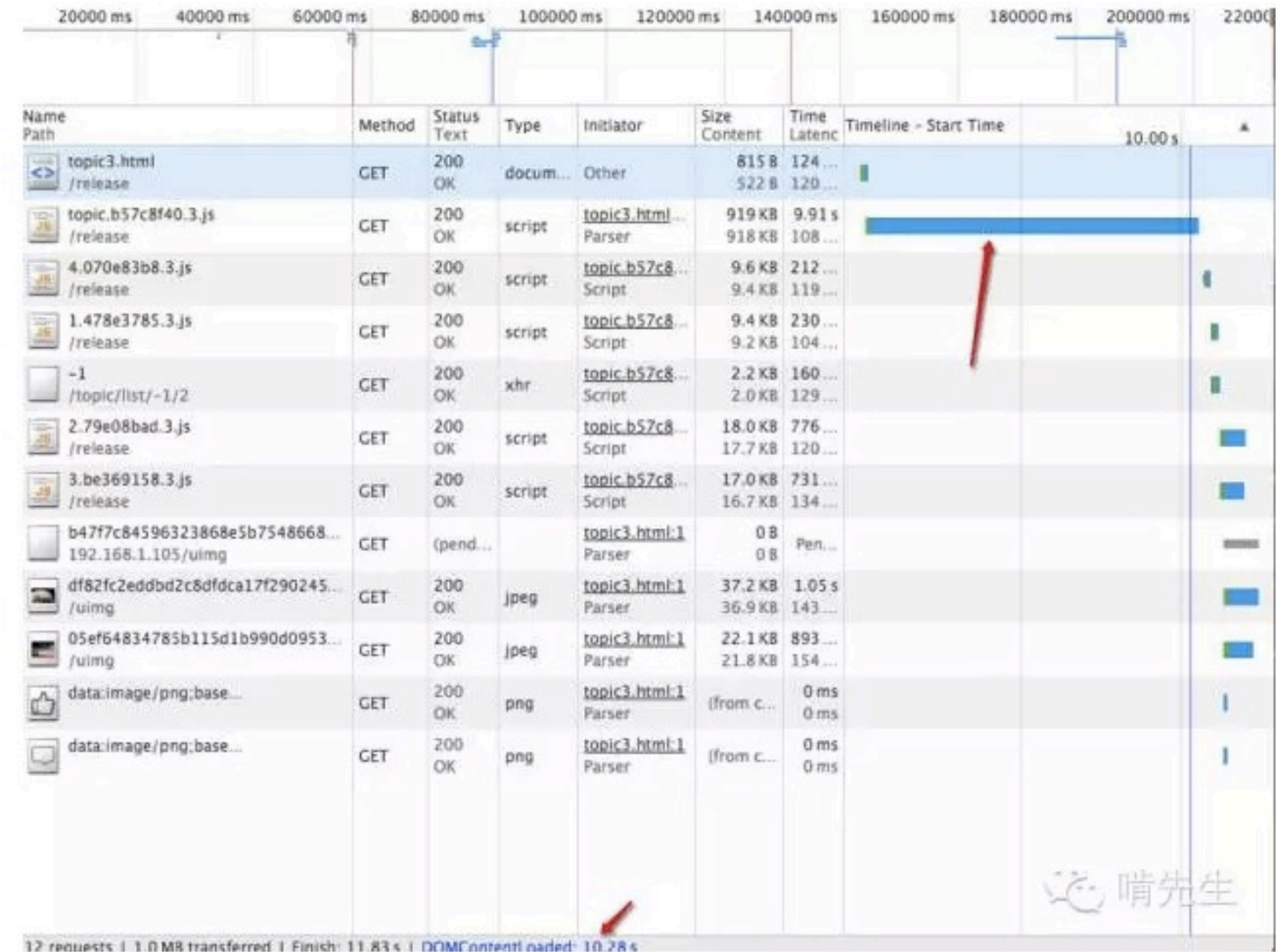
花较大篇幅介绍插件的使用，以下通过在一个DemoApp的构建过程中思考的一些问题（这些问题基本会在每个项目中遇到），让这些插件逐一登场。

一、文件过大

DemoApp最初的构建结果如下：



这里生成了一个topic.xxx.js，这个文件本来很小，估计只有10Kb左右，但构建的结果居然快1Mb了。在3G网络（750Kb/s）下的加载瀑布流如下图：



这张图（体现前端文件加载过程）曝露了几个问题：

1. 上面箭头所指的很蓝色柱子，说明了大部分时间消耗在加载**topic.xxxx.js**上。
2. 所有**JS**文件相关的柱子是一根结束了另一根才开始，说明不合理的文件合并策略，导致文件串行加载。

3. 结果就是如底部的箭头所看到的，页面的加载时间太长了（3G网络，10+秒）。

观察构建的文件，发现原来构建工具把React、jQuery等代码库合并到了topic.xxxx.js，造成此文件过大。如果再加一个activity模块呢？很明显，activity.xxx.js得到类似的结果，都太大了，并且React、jQuery等代码库重复被合并到activity和topic里，如下图。如果再加模块也会得到同样的结果，模块越多重复加载的情况越严重。

Hash: 1601b244f958411df6a2
Version: webpack 1.12.2
Time: 4351ms

Asset	Size	Chunks		Chunk Names
activity.1723d126.3.js	928 kB	0	[emitted]	activity
1.478e3785.3.js	9.37 kB	1	[emitted]	
2.752421bb.3.js	9.5 kB	2	[emitted]	
topic.e000dd57.3.js	940 kB	3	[emitted]	topic
4.18a581dc.3.js	16.9 kB	4	[emitted]	
5.e590f4ad.3.js	17 kB	5, 2	[emitted]	
topic3.html	518 bytes		[emitted]	
activity3.html	521 bytes		[emitted]	

啃先生

可见，提取公共代码，情况可以得到改善，另外，压缩代码无疑是可以使文件变小的。

1. 提取React、jQuery等库文件。它们很少变化，并且到处被复用，应该被提取出来，并且得到长时间的缓存。

此处使用插件：WebPack.optimize.CommonsChunkPlugin（WebPack内建插件）

```
entry: {
  topic: './src/compoents/app/topic.js',
  activity: './src/compoents/app/activity.js',
  react: ['react'],
  jquery:['jquery']
},

plugins: [
  new CommonsChunkPlugin({
    name: ['jquery','react'], // 将公共模块提取
    minChunks: Infinity // 提取所有entry共同依赖的模块
  }),
```

2. 代码压缩。React由700+ Kb压缩成100+ Kb

此处使用插件：WebPack.optimize.UglifyJsPlugin（WebPack内建插件）

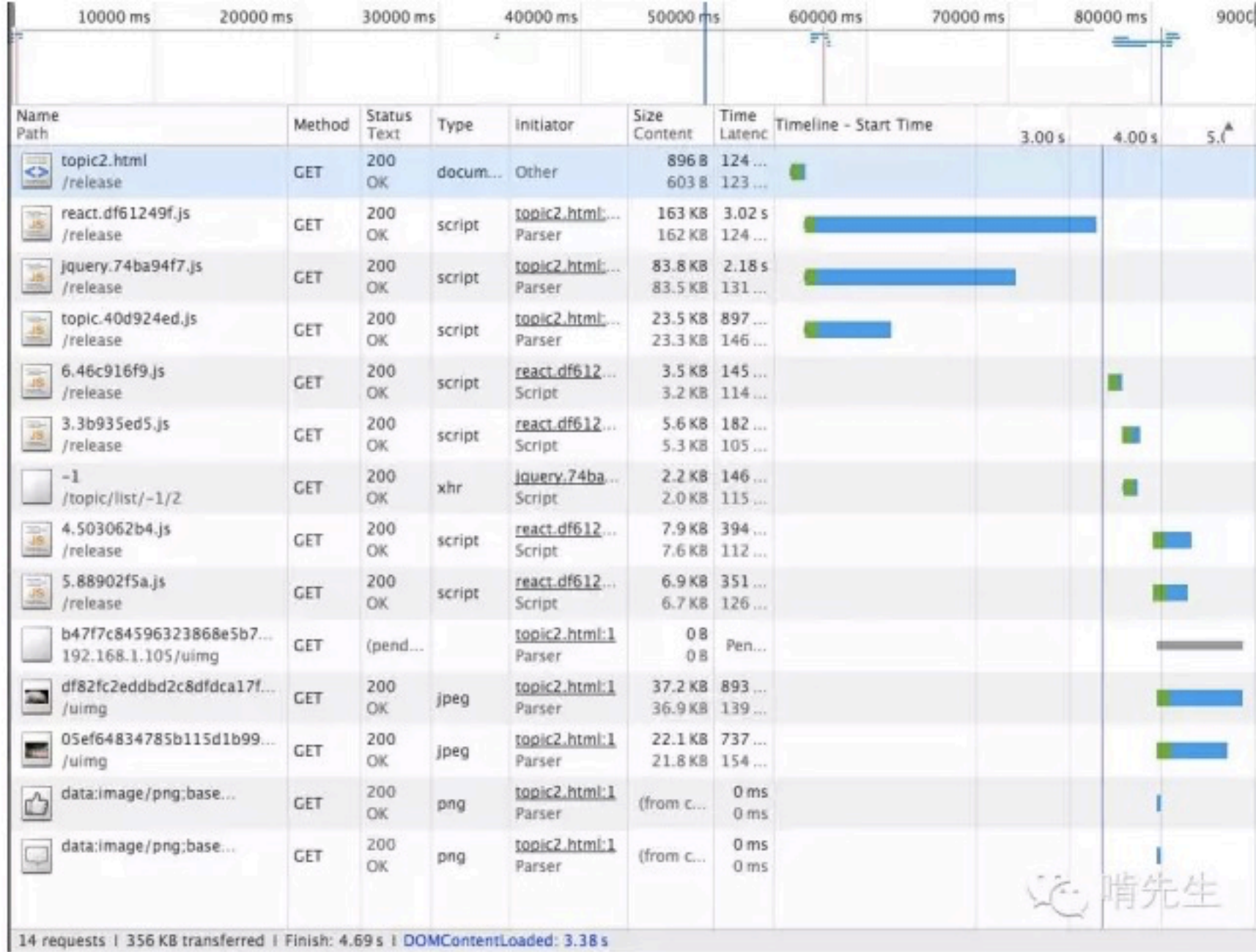
```
new webpack.optimize.UglifyJsPlugin({
  compress: {
    warnings: false
  }
}),
```

处理后topic.xxx.js和activity.xxx.js都很小了，并且多了jquery.xxx.js和react.xxx.js

Asset	Size	Chunks		Chunk Names
activity.dd99d006.js	15.9 kB	0	[emitted]	activity
1.19ce34aa.js	5.42 kB	1	[emitted]	
2.006a5327.js	3.27 kB	2	[emitted]	
jquery.78d098a6.js	85.5 kB	3	[emitted]	jquery
react.8a419004.js	166 kB	4	[emitted]	react
topic.bd5b67a7.js	23.7 kB	5	[emitted]	topic
6.36a590b9.js	7.58 kB	6	[emitted]	
7.465dd2eb.js	6.76 kB	7, 2	[emitted]	
topic2.html	599 bytes		[emitted]	
activity2.html	602 bytes		[emitted]	
[0] multi jquery	28 bytes	{3}	[built]	
[0] multi react	28 bytes	{4}	[built]	
+ 205 hidden modules				

啃先生

再看看文件加载的瀑布流，柱子所占比例短了，同时资源并行加载。



到此为止，这个问题算比较好地解决了，但还不够，随着项目越来越大，还有一个重要因素会导致文件很大。这部分内容放到本文的最后介绍。

二、如何缓存

缓存控制要做到两件事情，提到缓存命中率

1. 对于没有修改的文件，从缓存中获取文件
2. 对于已经修改的文件，不要从缓存中获取

围绕这两点，演绎出了很多方案，此处列两种：

- 不处理，等待用户浏览器缓存过期，自动更新。这是最偷懒的，命中率低一些，同时可能会出现部分文件没有更新，导致报错的情况。
- Http头对文件设置很大的max-age，例如1年。同时，给每个文件命名上带上该文件的版本号，例如把文件的hash值做为版本号，topic.ef8bed6c.js。即是让文件很长时间不过期。
 - 当文件没有更新时，使用缓存的文件自然不会出错；
 - 当文件已经有更新时，其hash值必然改变，此时文件名变了，自然不存在此文件的缓存，于是浏览器会去加载最新的文件。

从上面的截图可以看出来，通过WebPack是可以很轻松做到第二点的——只需要给文件名配置上[chunkhash:8]即可，其中8是指hash长度为8，默认是16。

```
output: {
  path: __dirname + '/release/',
  filename: "[chunkhash:8].[name].js"
  chunkFilename: "[name].[chunkhash:8].js"
},
```

P.S.这样的处理效果已经很好了，但同样有劣处，即浏览器给这种缓存方式的缓存容量太少了，只有12Mb，且不分Host。所以更极致的做法是以文件名为Key，文件内容为value，缓存在localStorage里，命中则从缓存中取，不命中则去服务器取，虽然缓存容量也只有5Mb，但是每个Host是独享这5Mb的。

三、自动生成页面

文件名带上版本号后，每一次文件变化，都需要Html文件里手动修改引用的文件名，这种重复工作很琐碎且容易出错。

```
<link href="topic.16bcd89b.css" rel="stylesheet">
</head>

<body>
  <script src="react.64cf4451.js"></script>
  <script src="jquery.64cf4451.js"></script>
  <script src="topic.64cf4451.js"></script>
</body>
</html>
```



容易出错的部分

使用 **HtmlWebpackPlugin** 和 **ExtractTextPlugin** 插件可以解决此问题。

- 生成带JS的页面

```
new HtmlWebpackPlugin({
  filename: 'topic2.html',
  template: __dirname + '/src/app.html',
  inject:true,
  chunks:['react','jquery','topic'],

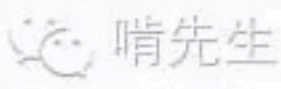
  // 排序
  chunksSortMode:function (a, b) {
    var index = {'topic':1, 'react':3, 'jquery': 2},
        aI = index[a.origins[0].name],
        bI = index[b.origins[0].name];

    return aI&&bI? bI - aI : -1;
  }
}),

new HtmlWebpackPlugin({
  filename: 'activity2.html',
  template: __dirname + '/src/app.html',
  inject:true,
  chunks:['react','jquery','activity'],

  // 排序
  chunksSortMode:function (a, b) {
    var index = {'activity':1, 'react':3, 'jquery': 2},
        aI = index[a.origins[0].name],
        bI = index[b.origins[0].name];

    return aI&&bI? bI - aI : -1;
  }
}),
```



- 生成带css的页面

```
new ExtractTextPlugin("comm.[contenthash:9].css")
```

插件介绍到此为止，然而，还有一个关于同步加载和异步加载的问题，否则入口文件还是会很臃肿。

肆 | Fourth

关于同步加载和异步加载

使用WebPack打包，最爽的事情莫过于可以像服务器编程那样直接require文件，看起来是同步地从服务器上取得文件

直接就使用了。如下面的代码一样，没有任何异步逻辑，代码很干净。

```
var React = require('react');
var TopicItem = require('../topic/topicitem');

var Topic = React.createClass({
  render: function() {
    var topicItems = this.props.data.map(function (topic) {
      return (
        <TopicItem data={topic} />
      );
    });

    return (
      <div className="topic-list">
        {topicItems}
      </div>
    );
  }
});

module.exports = Topic;
```

晴先生

然而，这种爽是有代价的，对于直接require模块，WebPack的做法是把依赖的文件都打包在一起，造成文件很臃肿。所以在写代码的时候要注意适度同步加载，同步的代码会被合成并且打包在一起；异步加载的代码会被分片成一个个**chunk**，在需要该模块时再加载，即按需加载，这个度是要开发者自己把握的，同步加载过多代码会造成文件过大影响加载速度，异步过多则文件太碎，造成过多的Http请求，同样影响加载速度。

- 同步加载的写法，如：

```
var TopicItem = require('../topic/topicitem');
```

- 异步加载的写法，如：

```
require.ensure([], function(){
  dialog = require('../widget/dialog');
  $ = require('jquery');
  io = require('../utils/io');
});
```

晴先生

一个原则是：首屏需要的同步加载，首屏过后才需要的则按需加载（异步）

结语

以上是WebPack构建工具比较好的实践，可见，要用好还是很考验前端性能优化的功力的，比较什么时候同步，什么时候异步，如果做缓存等等。



如果觉得文章有用，顺手点击下方的推荐

标签: javascript , WebPack , 前端构建

标签: javascript , WebPack , 前端构建

好文要顶

关注我

收藏该文







啃先生

关注 - 9

粉丝 - 66

+加关注


« 上一篇: [【移动适配】移动Web怎么做屏幕适配（一）](#)

» 下一篇: [【移动适配】一个像素的border怎么实现](#)

posted @ 2016-03-03 00:32 啃先生 阅读(27048) 评论(21) 编辑 收藏

努力加载评论中...

[刷新评论](#) [刷新页面](#)

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)， [访问](#) 网站首页。

- 【推荐】超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
- 【活动】京东云服务器_云主机低于1折，低价高性能产品备战双11
- 【推荐】天翼云双十一提前开抢，1核1G云主机3个月仅需59元
- 【推荐】阿里云双11冰点钜惠，热门产品低至一折等你来抢！
- 【福利】个推四大热门移动开发SDK全部免费用一年，限时抢！

- 相关博文:
- webpack实例与前端性能优化
 - 关于web前端性能优化问题
 - 前端性能优化
 - 关于前端性能优化
 - 前端性能优化
 - » 更多推荐...

- 最新 IT 新闻:
- 华为与四十多家企业成立江苏鲲鹏计算产业联盟
 - 科学家发现低质量黑洞组成的双星系统
 - CPU市场已然天翻地覆！AMD稳稳占据78%
 - 宇宙的一切，都源自这个“扭曲”的过程
 - 苹果宣布拿出25亿美元应对美国加州住房危机
 - » 更多新闻...

公告



前腾讯前端开发工程师，后来
历参与创业，目前在券商。当
前端开发经验，创业故事，互
考。 @深圳

昵称： 啃先生
园龄： 8年6个月
粉丝： 66
关注： 9
[+加关注](#)

<	2019年11月				
日	一	二	三	四	五
27	28	29	30	31	
3	4	5	6	7	8
10	11	12	13	14	15
17	18	19	20	21	22
24	25	26	27	28	29
1	2	3	4	5	6

搜索

常用链接

- [我的随笔](#)
- [我的评论](#)
- [我的参与](#)
- [最新评论](#)
- [我的标签](#)

我的标签

- [javascript](#)(8)
- [H5](#)(3)
- [移动Web](#)(3)
- [CentOS](#)(1)
- [JavaScript](#)[JavaScript](#)(1)
- [NodeJs](#)(1)
- [WebPack](#)(1)
- [前端构建](#)(1)

随笔档案 (9)

- [2016年3月](#)(5)
- [2011年11月](#)(2)
- [2011年6月](#)(2)

1. Re: [【前端构建】Webpack 4 使用指南](#)

[前端性能优化](#)

GOOD

2. Re: [【前端构建】Webpack 4 使用指南](#)

[前端性能优化](#)

好文章，收藏了

webpack中文文档：

3. Re: [【移动适配】移动端 H5 页面适配](#)

[屏幕适配（一）](#)

如果原生的html中的style是在head中的，我该如何设置js计算呢

4. Re: [【移动适配】一个css3的border-radius实现](#)

[border怎么实现](#)

请问，使用transform scale为1.5，border-radius啊?不太明白。

5. Re: [【移动适配】移动端 H5 页面适配](#)

[屏幕适配（一）](#)

我的为什么会报错?

Uncaught TypeError: Cannot read property 'firstElementChild' of null