

Webpack傻瓜式指南（一）



张轩 · 2 年前

Webpack最近很热，我一开始是想翻译一篇国外关于webpack的佳作，但是找来找去也没有一篇让我感觉到很满意的，好多都是一步到位，满屏幕都是React + Webpack，官方文档写的不太好，好多点都没有解释的详细，所以我参考了几篇文章，写一篇傻瓜式指南。本文适合第一次接触webpack的朋友，如果是老鸟，就不用看了。这是系列的第一篇，主要讲他最基本的用法。

比较

如果你熟悉原来一系列的构建工具，grunt或者gulp之类的，这里有一篇webpack和他们比较的文章可以读一读。 [Webpack Compared](#)

安装

先装好node和npm，因为webpack是一个基于node的项目。然后

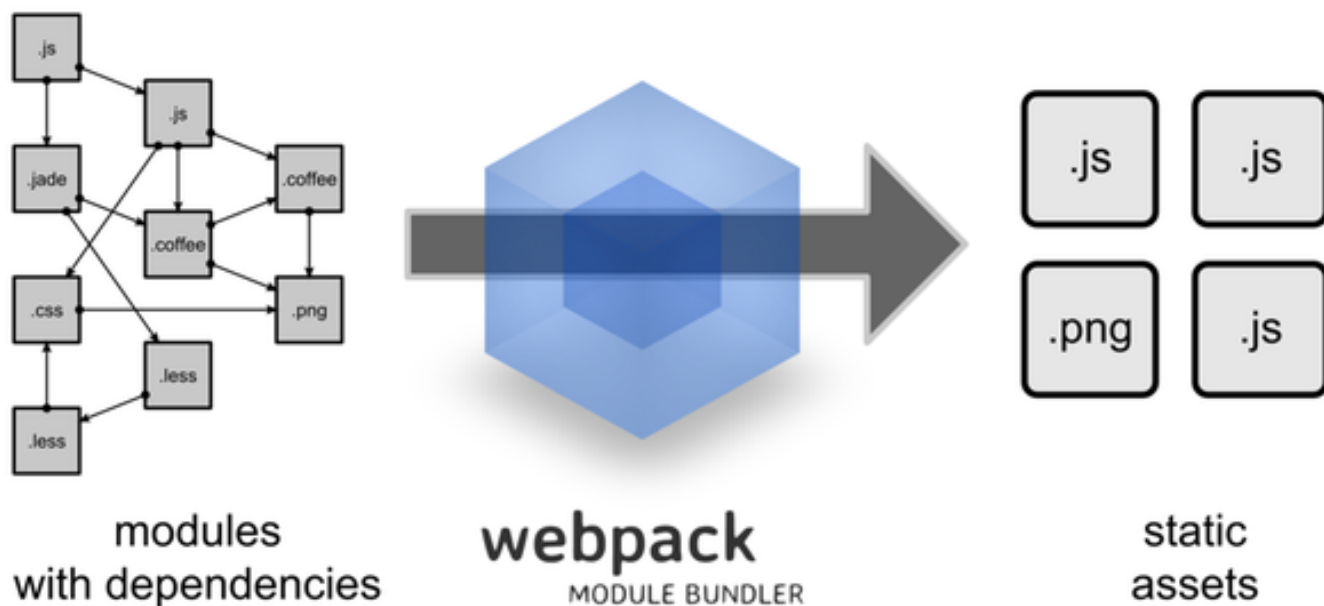
```
npm install -g webpack
```

知 首发于
前端外刊评论

写文章

登录

官网对webpack的定义是MODULE BUNDLER，他的目的就是把有依赖关系的各种文件打包成一系列的静态资源。请看下图



webpack简单点来说就就是一个配置文件，所有的魔力都是在这一个文件中发生的。这个配置文件主要分为三大块

- entry 入口文件 让webpack用哪个文件作为项目的入口
- output 出口 让webpack把处理完成的文件放在哪里
- module 模块 要用什么不同的模块来处理各种类型的文件

下面我们就一步一步来新建一个简单的项目

建立项目

建一个文件夹，然后新建一个package.json的文件在项目根目录下

```
mkdir webpack
cd webpack
npm init
# 一直点回车 如果懒得填一些信息
```

如果你使用git管理你的这个项目的話，建议你新建一个.gitignore文件，不要让git提交一些node依赖的模块，你也可以参考github的例子 [gitignore/Node.gitignore at master · github/gitignore · Git](#)

我们这里就简单一点

```
node_modules
```

项目结构

现在项目里面就有一个package.json，我们多加一点东西，慢慢丰富它的内容。

- /app
 - index.js
 - sub.js
- package.json
- webpack.config.js

添加了两个js文件，添加了最重要的webpack的配置文件，我们还是从非常简单的hello world开始玩起，webpack原生直接支持AMD和CommonJS两种格式，如果你想使用ES6的风格，这点以后再提。

JS代码

sub.js

```
//我们这里使用CommonJS的风格
function generateText() {
  var element = document.createElement('h2');
  element.innerHTML = "Hello h2 world";
  return element;
}
```

```
module.exports = generateText;
```

index.js

```
var sub = require('./sub');
var app = document.createElement('div');
app.innerHTML = '<h1>Hello World</h1>';
app.appendChild(sub());
```

代码写完了，完成一个很简单的功能，新建一个单独的module，并且在另外一个module里面引用他，最后会在页面里面输出两个标题。

配置Webpack

现在开始配置webpack，目标是把这两个js文件合并成一个文件。我们可以自己在build文件夹里面手动建一个index.html文件夹，然后再把合并以后的js引用在里面，但是这样有些麻烦，所以我们这里安装一个plugin，可以自动快速的帮我们生成HTML。

```
npm install html-webpack-plugin --save-dev
```

好 有了这个插件 开始写config文件

```
var path = require('path');
var HtmlWebpackPlugin = require('html-webpack-plugin');
//定义了一些文件夹的路径
var ROOT_PATH = path.resolve(__dirname);
var APP_PATH = path.resolve(ROOT_PATH, 'app');
var BUILD_PATH = path.resolve(ROOT_PATH, 'build');

module.exports = {
  //项目的文件夹 可以直接用文件夹名称 默认会找index.js 也可以确定是哪个文件名字
  entry: APP_PATH,
  //输出的文件名 合并以后的js会命名为bundle.js
  output: {
    path: BUILD_PATH,
    filename: 'bundle.js'
  },
  //添加我们的插件 会自动生成一个html文件
  plugins: [
    new HtmlWebpackPlugin({
      title: 'Hello World app'
    })
  ]
};
```

然后在项目根目录运行

```
webpack
```

终端显示一堆信息，然后告诉你成功了。

你会发现多出来一个build文件夹，直接点开里面的html文件，你会发现我们可爱的“hello world”已经插入到页面了。我们的任务完成了，成功生成html，合并js，html引入了js，js被执行了。

配置webpack-dev-server

上面任务虽然完成了，但是我们要不断运行程序然后查看页面，所以最好新建一个开发服务器，可以serve我们pack以后的代码，并且当代码更新的时候自动刷新浏览器。

安装webpack-dev-server

```
npm install webpack-dev-server --save-dev
```

安装完毕后 在config中添加配置

```
module.exports = {  
  ....  
  devServer: {  
    historyApiFallback: true,  
    hot: true,  
    inline: true,  
    progress: true,  
  },  
  ...  
}
```

然后再package.json里面配置一下运行的命令,npm支持自定义一些命令

```
...  
"scripts": {  
  "start": "webpack-dev-server --hot --inline"  
},  
...
```

好了，万事具备了，在项目根目录下输入npm start,一堆花花绿绿的信息后server已经起来了，在浏览器里面输入localhost:8080 发现伟大的hello world出现了，在js里面随便修改一些输出然后保存, boom!浏览器自动刷新，新的结果出现了。

拓展阅读 如果你的服务器端使用的是express框架，你还可以直接安装express的middleware，webpack配合express，很好用。

```
npm install webpack-dev-middleware --save-dev
```

Using React with Webpack Tutorial

添加CSS样式

现在来添加一些样式，webpack使用loader的方式来处理各种各样的资源，比如说样式文件，我们需要两种loader，css-loader 和 style-loader，css-loader会遍历css文件，找到所有的url(...)并且处理。style-loader会把所有的样式插入到你页面的一个style tag中。

安装我们的loader

```
npm install css-loader style-loader --save-dev
```

配置loader，在webpack.config.js中

```
devServer: {
  historyApiFallback: true,
  hot: true,
  inline: true,
  progress: true,
},
...
module: {
  loaders: [
    {
      test: /\.css$/,
      loaders: ['style', 'css'],
      include: APP_PATH
    }
  ]
},
...
plugins: [
  new HtmlwebpackPlugin({
    title: 'Hello World app'
  })
]
```

看loaders的书写方式，test里面包含一个正则，包含需要匹配的文件，loaders是一个数组，包含要处理这些程序的loaders，这里我们用了css和style，注意loaders的处理顺序是从右到左的，这里就是先运行css-loader然后是style-loader。

... ..

知

首发于
前端外刊评论

写文章

登录

```
h1 {  
  color: red;  
}
```

记得在入口文件index.js中引用

```
require('./main.css');
```

然后发现标题变成红色的了，webpack的理念是基于项目处理的，把对应的文件格式给对应的loader处理，然后你就不用管了，它会决定怎么压缩，编译。

那现在想使用一些有爱的css预编译程序，来点sass吧。你可能已经想到了，再来个loader就行啦，确实是这样简单。

```
npm install sass-loader --save-dev
```

稍微修改一下config，删掉我们先前添加的css规则，加上下面的loader

```
{  
  test: /\.scss$/,  
  loaders: ['style', 'css', 'sass'],  
  include: APP_PATH  
},
```

添加两个sass文件，variables.scss和main.scss

variables.scss

```
$red: red;
```

main.scss

```
@import "./variables.scss";  
  
h1 {  
  color: $red;  
}
```

在index.js中引用

```
require('./main.scss');
```

然后发现标题如愿变红，相当简单吧。

处理图片和其他静态文件

这个和其他一样，也许你也已经会玩了。安装loader，处理文件。诸如图片，字体等等，不过有个神奇的地方它可以根据你的需求将一些图片自动转成base64编码的，为你减轻很多的网络请求。

安装url-loader

```
npm install url-loader --save-dev
```

配置config文件

```
{
  test: /\. (png|jpg)$/,
  loader: 'url?limit=40000'
}
```

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。

下面举个栗子。

新建一个imgs文件夹，往里面添加一张崔叔的照片。在scss文件中添加如下的东西。

```
@import "./variables.scss";

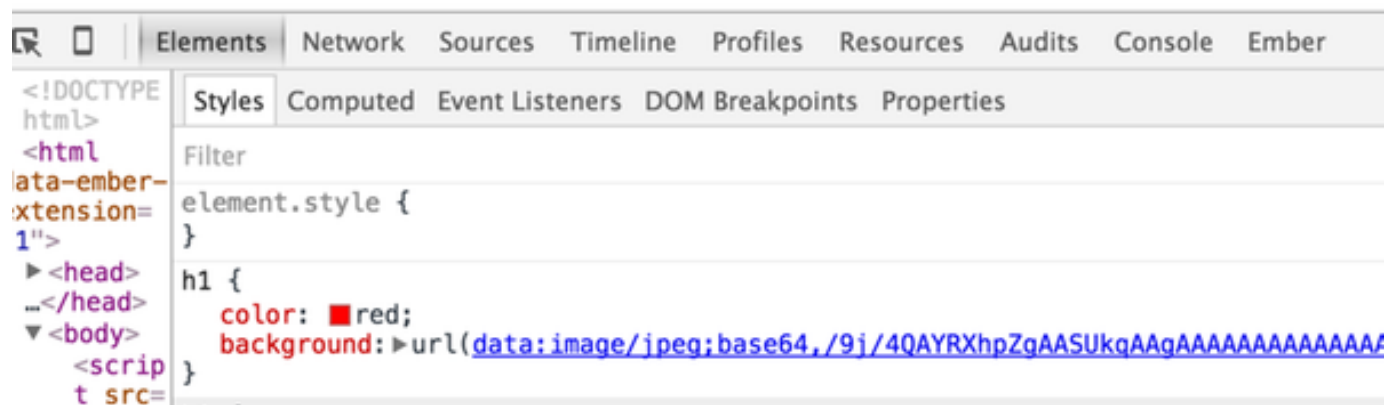
h1 {
  color: $red;
  background: url('./imgs/avatar.jpg');
}
```

npm start, 然后查看图片的url，发现神奇。

Hello World it

Hello h2 world hahaha

look at me! now is 2015-11-17T14:21:53+08:00



添加第三方库

有的时候还想来点jquery, moment, underscore之类的库, webpack可以非常容易的做到这一点, 有谣言说Bower即将停止开发了, 作者推荐都使用npm来管理依赖。那么我们现在安装在我们的app中添加jquery和moment的支持。

```
npm install jquery moment --save-dev
```

在js中引用

```
var sub = require('./sub');
var $ = require('jquery');
var moment = require('moment');
var app = document.createElement('div');
app.innerHTML = '<h1>Hello World it</h1>';
document.body.appendChild(app);
app.appendChild(sub());
$('body').append('<p>look at me! now is ' + moment().format() + '</p>');
```

看看浏览器, 成功! jquery和moment现在都起作用了!

添加ES6的支持

首先 装各种loader

```
npm install babel-loader babel-preset-es2015 --save-dev
```

配置我们的config文件

```
...  
  {  
    test: /\.jsx?$/,  
    loader: 'babel',  
    include: APP_PATH,  
    query: {  
      presets: ['es2015']  
    }  
  },  
...
```

es2015这个参数是babel的plugin，可以支持各种最新的es6的特性，具体的情况看这个链接。[Babel es2015 plugin](#)

现在我们可以改掉CommonJS风格的文件了。

sub.js

```
export default function() {  
  var element = document.createElement('h2');  
  element.innerHTML = "Hello h2 world hahaha";  
  return element;  
}
```

index.js

```
import './main.scss';  
import generateText from './sub';  
import $ from 'jquery';  
import moment from 'moment';  
  
let app = document.createElement('div');  
const myPromise = Promise.resolve(42);  
myPromise.then((number) => {  
  $('body').append('<p>promise result is ' + number + ' now is ' + moment().format() + '</p>');  
});  
app.innerHTML = '<h1>Hello World it</h1>';
```

我们上面测试了import, export , const , let , promise等一系列es6的特性。

最后完美的输出界面。

结语

第一部分到这里结束，经过一系列例子，你应该能够了解webpack最基本的用法了吧。是否已经喜欢上这种简洁的配置了？下一部分我们会继续讨论一些webpack更高级的用法。

参考文章

[SurviveJS - Table of Contents](#)

[Introduction](#)

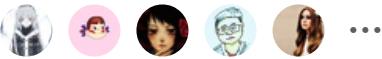
把这一节的代码做了一个repo，想看所有源码的可以clone下 [vikingmute/webpack-basic-starter · GitHub](#)

同时这一系列关于Webpack的文章也可以在github找到 [vikingmute/webpack-for-fools · GitHub](#)

微博关注：[前端外刊评论](#)

☆ 收藏 ↗ 分享 ⚠ 举报

 620



241 条评论

写下你的评论...




Galen

— · · ·

知

首发于
前端外刊评论

 写文章

[登录](#)



邱桐城

赞

2 年前



AnnatarHe

挺好的。我前几天刚开一个webpack + react的项目。因为babel升级到了6，出了各种问题。后来还是查文档才解决的。
给楼主点赞~

2 年前

2 赞



Awee

vue+webpack不能再赞

2 年前

3 赞



vagusX

点赞

2 年前



杜晗

很接地气， 赞一个....赞一个

2 年前



it Bee

不错

2 年前



小前风

先赞！

2 年前



首发于
前端外刊评论



写文章

登录

install webpack-dev-server --save-dev 回车后一直报错 Illegal instruction: 4;请问有其他人遇到过吗？

2 年前

1 赞



寸志 回复 代笔人与抒情体

查看对话

不要加 sudo

2 年前

1 赞

下一页

文章被以下专栏收录

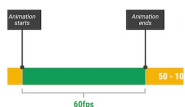


前端外刊评论

关注前端前沿技术，探寻业界深邃思想 qianduan.guru

进入专栏

推荐阅读



开启你的动画

原文地址：FLIP Your Animations网页上的动画应该要达到 60fps 的帧率，这个目标并不是那么... 查看全文 >

kmokidd · 2 年前 · 发表于 前端外刊评论

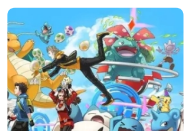


OSTHTML

HTML 处理利器 PostHTML 入门教程

看到 PostHTML，你应该自然就会想起 PostCSS，后者作为新一代的 CSS 处理器，一扫 SASS LESS... 查看全文 >

寸志 · 2 年前 · 发表于 前端外刊评论



《Pokemon Go》一周年：活跃用户流失90%，抓精灵风潮就这样逝去了？

知

首发于
前端外刊评论

写文章

登录

sykong · 18 天前 · 编辑精选

U23第14轮观察：二次转会U21球员将成焦点

2018版U23新政推出之后，中超真的开始变了模样。二次转会的窗口已经开启，却没有像往常一样出现外援的大规模更换，各队都仍在静观其变。但不管外援有何变化，在新政的强硬指导下，年轻球员... [查看全文](#) >

创冰DATA · 1 个月前 · 编辑精选 · 发表于 创冰DATA