WebApp实战应用 12章：webpack应用实战

作者： 徐礼文 2016/8/20 15:41:53

前端革命，革了再革：WebPack

Webpack: 为Web开发而生的模块管理器

> Webpack 是德国开发者 Tobias Koppers 开发的模块加载器 Instagram 工程师认为这个方案很棒, 还把作者招过去了 在 Webpack 当中, 所有的资源都被当作是模块： js, css, 图片等等.. 因此, Webpack 当中 js 可以引用 css, css 中可以嵌入图片 dataUrl。
>
> **对应各种不同文件类型的资源, Webpack 有对应的模块 loader**

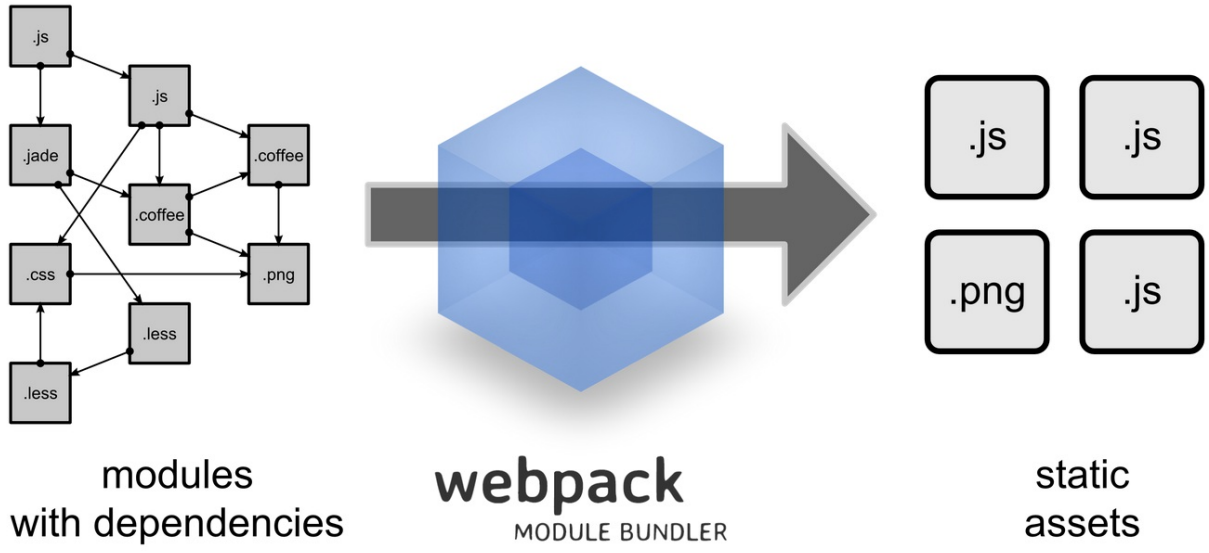
一、 webpack说明

> CommonJS和AMD是用于JavaScript模块管理的两大规范，前者定义的是模块的同步加载，主要用于NodeJS；而后者则是异步加载，通过requirejs等工具适用于前端。随着npm成为主流的JavaScript组件发布平台，越来越多的前端项目也依赖于npm上的项目，或者自身就会发布到npm平台。
>
> 因此，让前端项目更方便的使用npm上的资源成为一大需求。于是诞生了类似browserify这样的工具(**Browserify,让Node模块跑在浏览器里**)，Browserify让代码中可以使用require函数, 直接以同步语法形式引入npm模块，打包后再由浏览器执行。
>
> web开发中常用到的静态资源主要有JavaScript、CSS、图片、Jade等文件，webpack中将静态资源文件称之为模块。webpack是一个module bundler(模块打包工具)，其可以兼容多种js书写规范，且可以处理模块间的依赖关系，具有更强大的js模块化的功能。Webpack对它们进行统一的管理以及打包发布。
>
> Webpack其实有点类似browserify，出自Facebook的Instagram团队，但功能比browserify更为强大。

webpack 的优势

1. webpack 是以 commonJS 的形式来书写脚本的, 但对 AMD/CMD 的支持也很全面，方便旧项目进行代码迁移；
2. 串联式模块加载器以及插件机制，让其具有更好的灵活性和扩展性，例如提供对CoffeeScript、ES6的支持；
3. 能被模块化的不仅仅是JS了，还可以对js、css、图片等资源文件都支持打包， 可以基于配置或者智能分析打包成多个文件，实现公共模块或者按需加载；
4. 开发便捷，能替代部分 grunt/gulp 的工作，比如打包、压缩混淆、图片转base64等。
5. 开发时在内存中完成打包，性能更快，完全可以支持开发过程的实时打包需求； 对sourcemap有很好的支持，易于调试。

> Webpack将项目中用到的一切静态资源都视之为模块，模块之间可以互相依赖。Webpack对它们进行统一的管理以及打包发布，其官方主页用下面这张图来说明Webpack的作用：



可以看到Webpack的目标就是对项目中的静态资源进行统一管理，为产品的最终发布提供最优的打包部署方案。

webpack中的Loader

> loaders 用于转换应用程序的资源文件，他们是运行在nodejs下的函数 使用参数来获取一个资源的来源并且返回一个新的来源(资源的位置)，例如：你可以使用loader来告诉webpack去加载一个coffeescript或者jsx。
>
> 在webpack中JavaScript，CSS，LESS，TypeScript，JSX，CoffeeScript，图片等静态文件都是模块，不同模块的加载是通过模块加载器（webpack-loader）来统一管理的。loaders之间是可以串联的，一个加载器的输出可以作为下一个加载器的输入，最终返回到JavaScript上：
>
> ！用来定义loader的串联关系，"-loader"是可以省略不写的，多个loader之间用“!”连接起来，但所有的加载器都需要通过 npm 来加载。

loader 特性

1. loaders可以串联，他们应用于管道资源，最后的loader将返回javascript，其它的可返回任意格式（传递给下一个loader）
2. loaders 可以同步也可以异步
3. loaders在nodejs下运行并且可以做一切可能的事
4. loader接受参数，可用于配置里
5. loaders可以绑定到extension/RegExps 配置
6. loaders 可以通过npm发布和安装
7. loaders 可以访问配置
8. 插件可以给loaders更多的特性
9. loaders可以释放任意额外的文件

如果你对loader的例子感兴趣可以去看下现有的loader列表：

> http://webpack.github.io/docs/list-of-loaders.html

二、 webpack安装与配置

全局安装

```
npm install webpack -g
```

项目本地安装

```
npm init     //创建package.json
npm install webpack --save-dev
```

安装plugin

```
npm install html-webpack-plugin --save-dev  //自动快速的帮我们生成HTML
```

Webpack开发服务器

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

webpack-dev-server --inline --hot //启动模块热替换

//或者采用配置的方式
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"
},
...

再执行 npm start    http://localhost:8080 (默认)
```

安装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
//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
npm install css-loader style-loader --save-dev

//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。
npm install url-loader --save-dev

npm install sass-loader --save-dev


Loader列表查询： http://webpack.github.io/docs/list-of-loaders.html
```

生成source-map

```
webpack --devtool source-map
```

创建项目配置文件 webpack.config.js

> 每个项目下都必须配置有一个 webpack.config.js ， 它的作用如同常规的 gulpfile.js/Gruntfile.js ， 就是一个配置项，告诉 webpack 它需要做什么。

```
var path = require("path");
var htmlplugin = 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 = {

//entry:{
//    page1:"./modules/test",
//    //支持数组形式，将加载数组中的所有模块，但以最后一个模块作为输出
//    page2:["./modules2/test2","./modules2/test3"]
//},

//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 htmlplugin({
        title:"hello"
    })
],
//开发服务器,当代码更新的时候自动刷新浏览器
devServer:{
    historyApiFallback: true,
    hot: true,
    inline: true,
    progress: true,
},
module:{
    loaders:[
        //对.scss的文件,应用 style-loader,css-loader,sass-loader来处理
        {test:/\.scss$/,loaders:["style","css","sass"],include:APP_PATH}
    ]
 }
}
```

webpack命令行

1. webpack --display-error-details
2. webpack --config XXX.js //使用另一份配置文件（比如webpack.config2.js）来打包
3. webpack --watch //监听变动并自动打包
4. webpack -p //压缩混淆脚本，这个非常非常重要！
5. webpack -d //生成map映射文件，告知哪些模块被最终打包到哪里了