大麦项目—webpack使用指南

具体开始之前，我们先说明一下为什么需要前端构建，这里我简单列举几点前端构建的优势：

1. 解决JavaScript和CSS的依赖问题：

我们实际开发中经常发现CSS没起作用，JavaScript的某个变量和方法找不到，有很多情况都是因为引入JavaScript或者CSS的顺序不对造成，而使用构建工具可以大大减少此类问题。

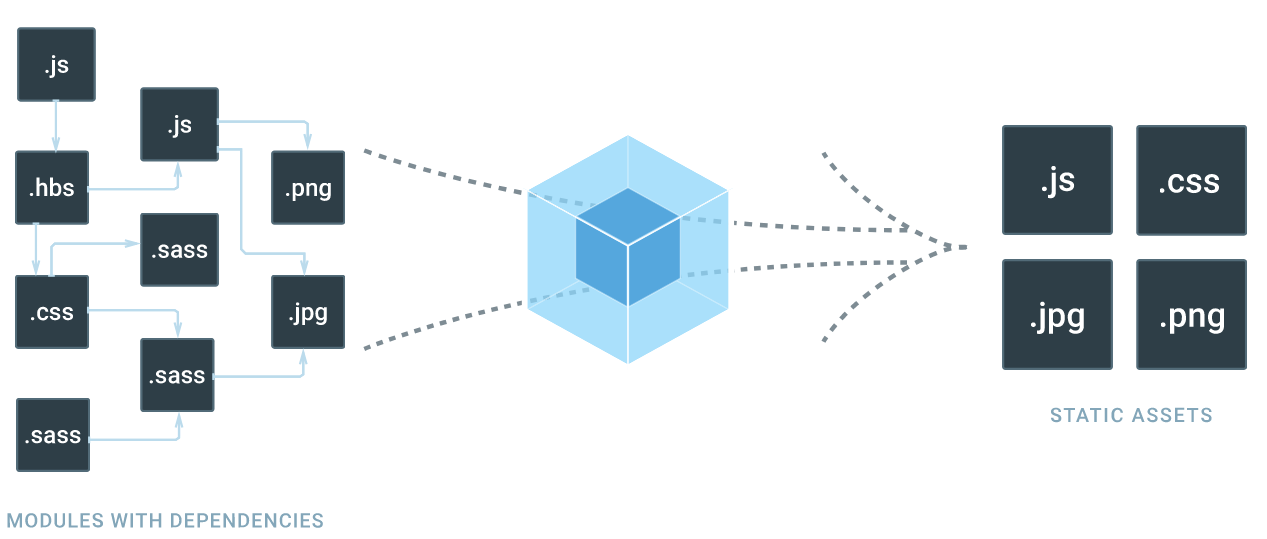
1. 性能优化

随着项目规模的增大，前端就会由很多JS文件构成，为了前端代码更清晰，结构更合理，我们就需要做两件事，文件合并和文件压缩，而构建工具就可以帮我们去完成这两件事情，当然构建所能做的事情远远不止于这两件事情。

1. 效率提升（添加css3前缀）

这里我们举例说明一下，在css3出现之初，各大浏览器对于css3的新属性兼容性并不是很好，需要加特定的浏览器前缀才能解决这个问题，但是人工去一个一个的添加，不光工作量大，而且有的时候也会遗漏，而且这个工作是没有什么技术含量的，当然如果可以自动添加最好，构建工具就可以帮我们完成。

我们大体了解了构建工具的优势之后，既然构建工具能满足前端的开发需求，减少不必要的人工投入，经过我们前端这块的市场调研，根据构建工具的使用具体占比情况，现在构建工具分类中，webpack占比最大，使用最为广泛，我们这里就选择使用webpack进行文件的打包和编译，使我们更方便的进行前端代码的开发和维护。Webpack可以将多种静态资源js、css、less转换成一个静态的文件（如图1），减少页面的请求，同时也减少了我们去转义less或ES6语法等工作，大大的提高了我们的开发效率；在学习webpack之前请先看《npm使用指南》，关于npm命令此文档不做过多的讲解；

关于webpack的使用我们简单的从以下几点进行讲解：安装webpack、使用webpack打包部署、webpack配置文件、本地环境调试、webpack配置文件的编写；  
 图1

# 1.1 安装webpack

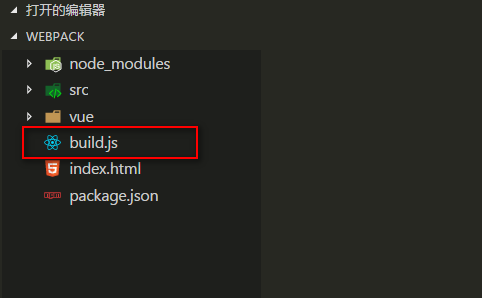
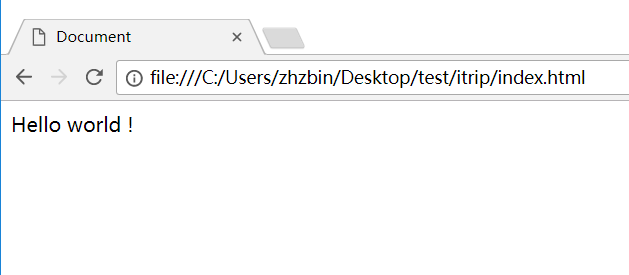
在安装webpack前需要先安装nodejs，由于npm安装比较慢我们可以使用淘宝的镜像cnpm（可参照《npm使用指南》）进行安装，这里简单介绍webpack是如何进行全局安装和项目安装：

* + 全局安装：cnpm install webpack -g
    - 安装完成之后可以打开cmd命令窗口输入webpack -v 会打印出版本号证明你已经安装成功；
  + 当前项目安装：cnpm install webpack --save-dev （此命令是针对于项目进行安装的）
    - 项目安装完成之后会在package.json文件devDependencies属性下有对应的信息"webpack": "^3.4.1+"
    - 添加后缀的具体意思：
      * --save 具体含义代表，我们在安装依赖包的时候，添加--save代表这时候下载的依赖包是我们项目在生产环境所依赖的包，这里有一个简写像是可以使用-S进行简写
      * --save-dev 具体含义代表，我们在安装依赖包的时候，添加--save-dev代表这时候下载的依赖包是我们项目在开发我们项目的阶段中所依赖的包 这里也有一个简写，可以使用-D来代替

# 1.2 使用webpack

安装完成webpack之后我们来讲一下关于webpack的使用，主要从以下几点进行讲解：打包部署、文件加载；

为方便我们在后期进行项目分组开发，使用VSCode打开dm文件夹，打开终端输入初始化命令npm init 进行初始化，初始化完成之后我们就可以开始安装对应的插件了。

* 一、在dm文件夹下建src文件夹，在此文件夹下新建一个index.js文件；
* 二、在index.js文件中添加一下代码document.write(“Hello world !”)
* 三、在dm下新建一个index.html文件，此文件的代码如下  
  <html>   
   <head>   
   <meta charset="utf-8">   
   </head>   
   <body>   
   <script type="text/javascript" src="build.js" charset="utf-8"></script>  
   </body>  
   </html>
* 四、在终端输入webpack ./src/index.js build.js命令后文件夹里会出现build.js文件（图2）。进入到dm文件夹下面，用浏览器打开index.html文件（图3），这就表明我们已经打包完成了；  
    
   图2  
    
   图3
* 五、现在我们可以添加第二个js文件了代码如下  
  export let world=()=>{  
   document.write(" This is index2.js.");  
  }  
  index.js文件代码如下  
   import {world} from './index2.js'  
   world();  
  代码修改完成之后输入 webpack ./src/index.js build.js命令之后，再打开页面显示出我们在index2.js文件中输入的内容；
* 六、引入CSS样式进行编译，在src文件夹下新建style.css文件，在文件夹里面添加：body{background-color:red},然后在index.js文件中添加 import ‘style.css’,然后使用 webpack ./src/index.js build.js命令打包会报错，提示编码错误，不能解析CSS样式，我们需要对引入的信息修改成：import "!style-loader!css-loader!./style.css"，再输入打包命令，发现还是报错，提示缺少css-loader style-loader这两个文件，那我们可以使用npm install css-loader style-loader --save-dev 将文件添加到package.json文件中，如果在后期打包部署时还是报文件缺失，可使用此命令添加文件；添加完成之后重新打包部署，页面显示为红色；

# 1.3 webpack配置文件

在上面的使用过程中，我们需要对不同的文件打包的时候引入对应的解析插件，但是在整个项目的开发过程中，使用这种方式非常不利于企业级开发，而且也比较浪费时间，这里我们使用webpack.config.js文件进行部署；

我们在dm文件夹下新建一个webpack.config.js文件，在文件里面添加一下文件信息：  
module.exports = {

entry: "./src/index.js",

output: {

path: \_\_dirname,

filename: "build.js"

},

module: {

loaders: [

{

test: /\.css$/,

loader: "style-loader!css-loader"

}

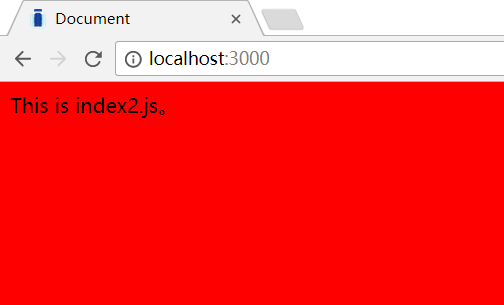
]

}

};

将index.js文件中的’ !style-loader!css-loader!’删除，直接运行webpack即可，webpack会自动查找webpack.config.js文件；关于webpack.config.js文件的其他配置属性请参照：[https://webpack.js.org/concepts/loaders/#example](https://webpack.js.org/concepts/loaders/" \l "example)地址

# 1.4 本地调试

我们在开发过程中发现，每次修改代码都要手动的build一下，页面再刷新一次才能看到后面修改的界面效果，那是否有别的方式来解决我们这个问题那？答案是有的，我们需要安装使用命令进行安装：cnpm install webpack-dev-server -g（全局安装）安装完成之后输入一下命令进行启动：webpack-dev-server --inline –-hot 在浏览器打开 http://localhost:8080/ 可以访问页面，也可以通过 --port 参数修改端口，参数后面加端口名称例如：webpack-dev-server --inline –-hot --port 3000 在页面访问的地址变成：http://localhost:3000/，界面显示效果如下，图4；  
  
 图4

为方便我们在后期开发时减少命令的输入我们在package.json文件中增加以下代码，再输入时可使用 ：npm start进行启动操作，打包部署使用 npm run build 进行打包操作，添加的代码如下：

"scripts": {

"start": "webpack-dev-server --inline --hot --port 3000",

"build": "webpack "

}

整个package.json文件的代码如下：  
{

"name": "dm",

"version": "1.0.0",

"description": "",

"main": "build.js",

"dependencies": {},

"devDependencies": {

"color-convert": "^1.9.0",

"css-loader": "^0.28.4",

"cssesc": "^1.0.0",

"emojis-list": "^2.1.0",

"escape-string-regexp": "^1.0.5",

"fastparse": "^1.1.1",

"has-ansi": "^3.0.0",

"has-flag": "^2.0.0",

"js-base64": "^2.1.9",

"json-stable-stringify": "^1.0.1",

"json5": "^0.5.1",

"loader-utils": "^1.1.0",

"regexpu-core": "^4.1.1",

"schema-utils": "^0.3.0",

"style-loader": "^0.18.2",

"webpack": "^3.4.1",

"webpack-dev-server": "^2.6.1"

},

"scripts": {

"start": "webpack-dev-server --progress --colors --port 3000",

"build": "webpack "

},

"author": "",

"license": "ISC"

}

如果在使用webpack进行打包处理或者本地调试提示报错，可以将以上代码粘贴至package.json文件中，在终端输入 npm install 添加需要的插件，如果还报某某插件未安装输入：npm install \*\*\*\* --save-dev 其中‘\*\*\*\*’为插件名称，这里的处理方式，是来针对某一个插件进行安装的。

总结：

这里给大家介绍的webpack构建构建内容只是我们比较常用的一些部分，至于更多的webpack的内容大家可以去它的官方网站上进行学习，这里给大家提供一个学习的网址。

地址：<https://webpack.js.org/>