什么是WebPack，为什么要使用它？

1. 为什要使用WebPack

现今的很多网页其实可以看做是功能丰富的应用，它们拥有着复杂的JavaScript代码和一大堆依赖包。为了简化开发的复杂度，前端社区涌现出了很多好的实践方法。

模块化，让我们可以把复杂的程序细化为小的文件;

类似于TypeScript这种在JavaScript基础上拓展的开发语言：使我们能够实现目前版本的JavaScript不能直接使用的特性，并且之后还能转换为JavaScript文件使浏览器可以识别；

Scss，less等CSS预处理器

...

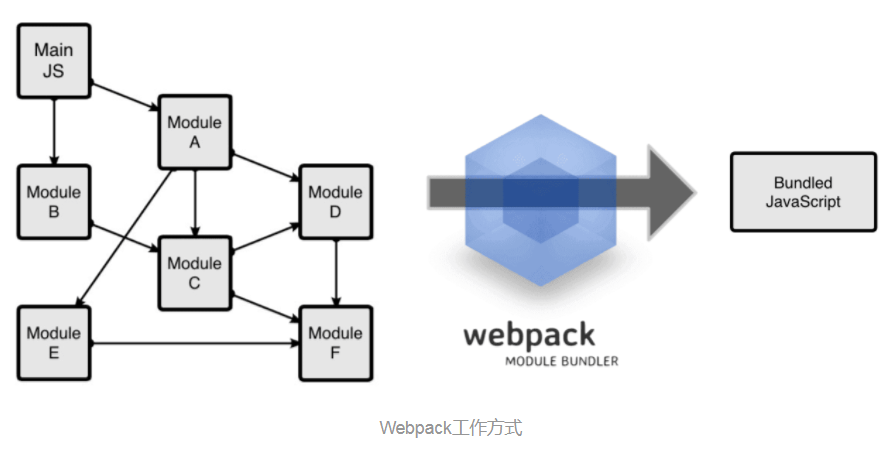
这些改进确实大大的提高了我们的开发效率，但是利用它们开发的文件往往需要进行额外的处理才能让浏览器识别,而手动处理又是非常繁琐的，这就为WebPack类的工具的出现提供了需求。

2.什么是Webpack

Webpack 是德国开发者 Tobias Koppers 开发的模块加载器兼打包工具，在webpack中，它能把各种资源，例如JS（含JSX）、coffee、样式（含less/sass）、图片等都作为模块来使用和处理。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比如vue用的是vue-loader当然这是后话，在后面我们再来说。

Webpack工作方式



1. 安装Webpack

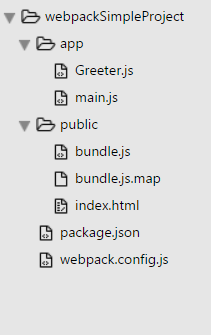
全局安装： npm install webpack -g;

安装到项目目录： npm install webpack --save-dev

1. 使用Webpack

一个简单的Webpack工程打包。

项目目录如下：



首先创建一个项目文件夹，然后使用npm init初始化npm项目，会自动生成package.json文件，然后在项目目录下创建app和public文件夹，app下存放项目源代码，public用于存放打包后的代码。

Greeter.js

var greet = document.createElement('div');

greet.textContent = "Hi there and greetings!";

export {greet}

main.js

import {greet} from './Greeter.js'

console.log(greet);

document.querySelector("#root").appendChild(greet);

Index.html

<!-- index.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Webpack Sample Project</title>

</head>

<body>

<div id='root'>

</div>

<script src="bundle.js"></script>

</body>

</html>

#### 5.通过配置文件来使用Webpack

Webpack拥有很多其它的比较高级的功能（比如说本文后面会介绍的loaders和plugins），这些功能其实都可以通过命令行模式实现，这样不太方便且容易出错的，更好的办法是定义一个配置文件，这个配置文件其实也是一个简单的JavaScript模块，我们可以把所有的与打包相关的信息放在里面。

继续上面的例子来说明如何写这个配置文件，在当前练习文件夹的根目录下新建一个名为webpack.config.js的文件，我们在其中写入如下所示的简单配置代码，目前的配置主要涉及到的内容是入口文件路径和打包后文件的存放路径。

webpack.config.js

module.exports = {

entry: \_\_dirname + "/app/main.js",//已多次提及的唯一入口文件

output: {

path: \_\_dirname + "/public",//打包后的文件存放的地方

filename: "bundle.js"//打包后输出文件的文件名

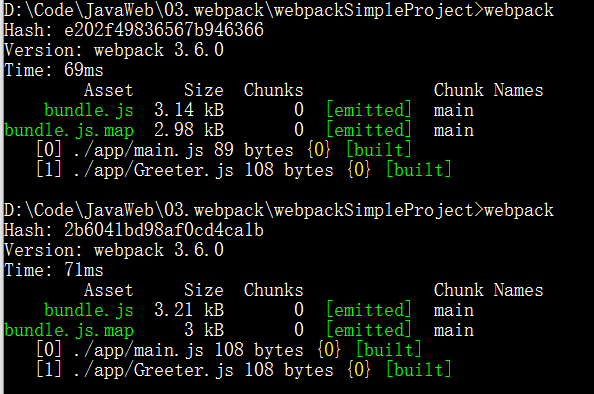
},

devtool: "source-map"

}

注：“\_\_dirname”是node.js中的一个全局变量，它指向当前执行脚本所在的目录。

有了webpack.config.js文件之后，只需在跟webpack打包配置文件同级目录下运行webpack命令，webpack就会自动读取改配置文件，根据配置完成打包。



至此，就完成了webpack的简单使用。

6.Webpack的强大功能

生成Source Maps（使调试更容易）

开发总是离不开调试，方便的调试能极大的提高开发效率，不过有时候通过打包后的文件，你是不容易找到出错了的地方，对应的你写的代码的位置的，Source Maps就是来帮我们解决这个问题的。

通过简单的配置，webpack就可以在打包时为我们生成的source maps，这为我们提供了一种对应编译文件和源文件的方法，使得编译后的代码可读性更高，也更容易调试。

在webpack的配置文件中配置source maps，需要配置devtool，它有以下四种不同的配置选项，各具优缺点，描述如下：



正如上表所述，上述选项由上到下打包速度越来越快，不过同时也具有越来越多的负面作用，较快的打包速度的后果就是对打包后的文件的的执行有一定影响。

对小到中型的项目中，eval-source-map是一个很好的选项，再次强调你只应该开发阶段使用它，我们继续对上文新建的webpack.config.js，进行如下配置:



cheap-module-eval-source-map方法构建速度更快，但是不利于调试，推荐在大型项目考虑时间成本时使用。

7.使用webpack构建本地服务器

想不想让你的浏览器监听你的代码的修改，并自动刷新显示修改后的结果，其实Webpack提供一个可选的本地开发服务器，这个本地服务器基于node.js构建，可以实现你想要的这些功能，不过它是一个单独的组件，在webpack中进行配置之前需要单独安装它作为项目依赖

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

devserver作为webpack配置选项中的一项，以下是它的一些配置选项，更多配置可参考这里。



把这些命令加到webpack的配置文件中，现在的配置文件webpack.config.js如下所示。

