Webpack开发文档

1. 安装

打开一个目录

Cnpm install webpack --global

1. 创建一个初始化文件 npm init
2. 创建文件

Name.js

Var name=”gp02-webpack”;

Module.exports = name;

Entry.js

document.getElementById("app").textContent = "Hello";

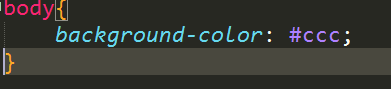
1. 创建一个html文件index.html,写一个id为app的div

引入bundle.js

1. 执行打包命令 webpack entry.js bundle.js
2. 打开浏览器查看页面
3. 通过webpack引入css模块化

Cnpm install css-loader style-loader --save-dev

1. 创建一个style.css



1. 在entry.js中引入css

require("style!css!./style.css");

1. 执行编译命令，打开网页查看，这个时候发现添加了一个灰色的背景
2. 编写webpack配置文件，可以节省很多编译命令

创建一个webpack.config.js

Module.exports = {

entry:”./entry.js”,

Output:{

Path:\_\_dirname,

Filename:”bundle.js”

},

Module:{

Loaders:[

{test:/\.css$/,loader:”style!css”}

]

}

此时我们已经不需要再在entry.js中那样引入css了

require("./style.css");

命令行编译 输入webpack即可编译

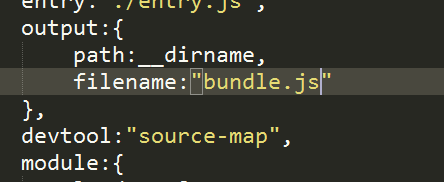
1. 调试代码，通过source-map调试

查看控制台sources

执行命令webpack --devtool source-map

在配置文件中配置直接生成source-map

添加devtool:”source-map”



1. webpack结合babel处理es2015(es6)

安装 cnpm install babel-loader babel-core babel-preset-es2015 --save-dev

在创建一个文件.babelrc的配置文件，是一个数组

{

“preset”:[“es2015”]

}

在配置文件中配置

添加一个{test:/\.js$/,loader:”babel”}

创建一个test.js,写一些标准的es6的语法

let name = “gp02-webpack”;

Es6导出方法

Export default name;

在entry.js中导入

Import name from “./name”;

Import “./style.css”

命令行执行命令webpack

1. 服务热替换

Cnpm install webpack-dev-server --global

Cnpm install webpack-dev-server --save-dev

启动命令 webpack-dev-server --inline --hot

更改一下样式表，就可以看到实时效果了

Npm install gulp-imagemin –save-dev