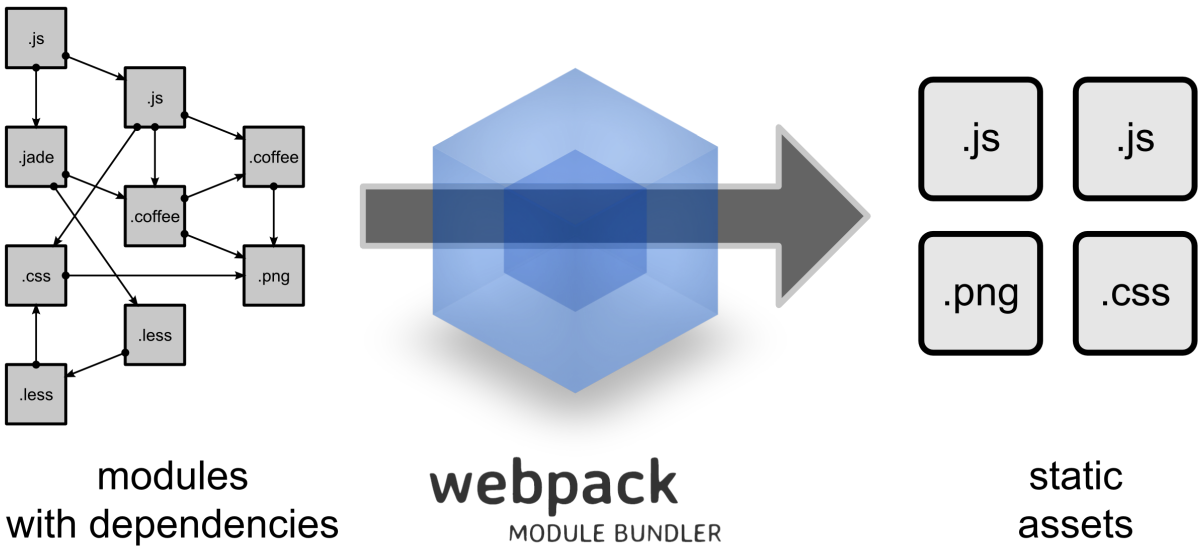
webpack

[webpack中文文档]

第一章 入门

简介



webpack是一个module bundler（模块打包工具），所谓的模块就是在平时的前端开发中，用到一些静态资源，如JavaScript、CSS、图片等文件，webpack就将这些静态资源文件称之为模块。

webpack支持CommonJS、AMD和ES6模块系统，并且兼容多种JS书写规范，可以处理模块间的依赖关系，所以具有更强大的JS模块化的功能，它能打包图片，对CSS、js文件进行编译打包，代码压缩，语法检查。

第一节 安装

> 1、webpack-cli必须要全局安装，否则不能使用webpack指令； 2、webpack也必须要全局安装，否则也不能使用webpack指令。 3、webpack4.x中webpack.config.js这样的配置文件不是必须的。 4、默认入口文件是./src/index.js，默认输出文件./dist/main.js。

1. 新建项目文件夹
2. 在`cmd`命令行窗口，进入项目文件夹
3. 在当前目录下，运行如下命令：

```
//初始化npm ,生成package.json文件
npm init -y

//安装全局webpack
npm install webpack -g

//安装全局webpack-cli
npm install webpack-cli -g

//本地依赖安装
npm install webpack webpack-cli --save-dev
```

第二节 hello world

SPA（single page web application）单页应用程序，是webpack打包的典型应用，一个典型的SPA应用，主要由以下几个部分组成：

文件	说明
index.html	主文件
JS文件	可能有多个JS文件，可通过webpack合并打包为一个文件
CSS文件	可能有多个CSS文件，可通过webpack合并打包为一个文件
图片	可通过webpack压缩优化

1. **新建src文件夹**

   该文件夹存放开发用的文件，通常命名为：src 、dev、app

```
//a.js
var run=function(){
    alert(123);
};
//node CommonJS模块
//module.exports.run=run;

//ES6语法
export default {
    run
};
//b.js-----------------------------------------------
var play=function(arg){
    alert(arg);
};
//node CommonJS模块
//module.exports.play=play;

//ES6语法
export default {
```

```
    play
};

//main.js-----------------------------------------------
//node CommonJS 引入js模块
//var a=require("./js/a.js");
//var b=require("./js/b.js");

//ES6 引入js模块
import a from "./js/a.js";
import b from "./js/b.js";

var txt = "hello world";
a.run();
b.play(txt);
```

2. **新建dist文件夹**

该文件夹存放打包后的文件，可以先不创建，打包时可以自动创建，通常命名为：dis 、dist、bulit

在此文件夹下新建index.html文件

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <!--该文件暂时不存在，打包后自动创建该文件-->
        <script src="./main.js"></script>
    </head>
    <body>
          <h1>hxsd</h1>
    </body>
</html>
```

3. **打包**

```
//基本命令
webpack src/main.js --output dist/main.js --mode development

//mode
//development(开发模式):非压缩打包 main.js
//production(生产模式):压缩打包 main.js
```

经过打包后，dist/main.js已经根据a.js b.js 的依赖关系，将三个文件打包合并为一个main.js文件。

第三章 配置文件入门

打包的参数有很多，需要定义配置文件进行复杂操作。

第一节 核心概念

一个配置文件的基本结构如下：

```
//需要依赖的模块
const webpack = require('webpack');

//配置项
module.exports={
    //入口配置
    entry:{ ..... },

    //输出配置
    output:{ ......},

    //模块
    module: { ......},

    //插件（数组）
    plugins:[ ......  ],

    //webpack-dev-server
    devServer:{ ...... }

    //打包模式
    mode:'development'
};
```

核心	说明
entry	入口 指示 webpack 应该使用哪个模块，来作为构建其内部*依赖图*的开始。 进入入口起点后，webpack 会找出有哪些模块和库是入口起点（直接和间接）依赖的。
output	输出 webpack 在哪里输出它所创建的 *bundles*，以及如何命名这些文件。
resolve	resolve属性中的extensions数组中，定义了可以省略哪些后缀名
module	模块识别loader 让 webpack 能够去处理那些非 JavaScript 文件（webpack 自身只理解 JavaScript）。
plugins	插件 用于执行范围更广的任务。插件的范围包括，从打包优化和压缩，一直到重新定义环境中的变量。插件接口功能极其强大，可以用来处理各种各样的任务。

第二节 简单的配置文件

- **创建配置文件**

在根目录下**webpack.config.js**（默认名称，不要修改），webpack运行时，会自动到项目根目录查找webpack.config.js并执行。

hello world案例，可以通过设置配置文件自动完成：

```
module.exports={  //是exports 不是 export
    //入口配置
    entry:'./src/index.js',

    //出口配置
    output:{
        path:__dirname + '/dist', //输出目录 __dirname:本文件所在硬盘路径（node全局变量）
        filename:'main.js' //文件名称（可以有子目录，例如： /js/main.js）
    }
};
```

- **修改webpack.json文件**

  在webpack.json中的"scripts"下增加：

```
"scripts": {
    "dev": "webpack --mode development",
    "build": "webpack --mode production"
  },
```

- **执行打包**

```
//命令行窗口执行
npm run dev
```

webpack.json中"script"下的dev命令会被执行，相当执行了webpack --mode development

第三章 配置文件进阶

webpack打包功能非常强大，可以一次性打包多个文件，下面详细讲解webpack.config.js配置文件的各个配置项

第一节 entry 入口配置

entry是指页面中的入口文件。也就是打包从哪个文件开始。

```
//可以是多个入口文件
entry: {
    index: './src/main.js',
    a: './src/a.js'
},
```

第二节 output 出口配置

是指生成的文件输出到哪个地方去，主要有以下属性：

属性	说明
path	输出路径
filename	输出文件名

第三节module

module项中定义了不同文件的所要使用的loader。

webpack除了可以打包js文件，还可以打包其他文件，hello world案例中，可以引入css文件：

```
//main.js-----------------------------------------------
//node CommonJS 引入js模块
//var a=require("./js/a.js");
//var b=require("./js/b.js");

//ES6 引入js模块
import a from "./js/a.js";
import b from "./js/b.js";

//希望同时打包css文件
import from "style.css";

var txt = "hello world";
a.run();
b.play(txt);
```

但是，webpack 自身只理解 JavaScript，无法识别css格式文件。

loader 可以将所有类型的文件转换为 webpack 能够识别、处理的有效模块，然后就可以利用 webpack对它们进行打包处理，常用loader有：

loader	说明
css-loader	解析css语句
style-loader	将css-loader解析后的文本，添加<style>标签
Babel-loader	编译器可以将jsx文件转成js文件（ES6转低版本）语法，提高兼容性
url-loader	url-loader对未设置或者小于limit设置的图片进行转换， 以base64的格式被img的src所使用， 而对于大于limit byte的图片用file-loader进行解析
file-loader	解析项目中的url引入（包括img的src和background的url） 修改打包后文件引用路径，使之指向正确的文件
less-loader	less编译器
sass-loader	sass编译器

loader也需要安装：

```
//可以一次性全部安装上：
npm install babel-loader babel babel-core css-loader style-loader url-loader file-loader less-loader less --save-dev
```

> vuejs 是一个入门简单的框架，具有使用简单，扩展方便的特点。随着webpack的流行，vuejs也推出了自己的vue-loader，可以方便的打包 .vue文件 的代码。在vue-cli（快速构建单页应用的脚手架）中得到应用。

第四节 plugins插件

hello world案例中，只对三个js文件进行了打包压缩，对index.html并没有进行任何处理，对html文件的打包压缩，要比js 和css文件要复杂，webpack本身并不具备这种复杂的处理能力，webpack提供了plugins（插件）功能，可以编写插件包，用于完成一些 loader 不能完成的工作。webpack 自带一些插件 官方插件列表，你可以通过 npm 安装插件。

例：HtmlWebpackPlugin

HtmlWebpackPlugin简化了HTML文件的创建，可以通过模板文件，生成一个HTML文件，安装：

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

第四章 webpack-dev-server

webpack-dev-server是一个小型的web服务器，可以自动监视项目文件的变化，自动刷新浏览器，其热模块替换(HMR：Hot Module Replacement)方式只替换更新的部分，而不是重载页面，大大提高了刷新效率。

安装

在当前的项目目录下，打开cmd窗口，安装webpack-dev-server，安装后，当前目录就是webserver的根目录

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

配置文件

配置	说明	设定值
contentBase	指定了服务器资源的根目录， 如果不写入contentBase的值，那么contentBase默认是项目的目录	"./"
historyApiFallback	它使用的是**HTML5 History Api**，任意的跳转或404响应可以指向index.html 页面；	true
inline	用来支持dev-server自动刷新的配置	true
hot	启动webpack热模块替换特性 需要安装HotModuleReplacementPlugin的插件	true
host	主机地址	
port	端口	默认8080
overlay	编译出错的时候，在浏览器页面上显示错误	false
stats	用来控制编译的时候shell上的输出内容 stats: "errors-only" 只打印错误 还有"minimal"，"normal"，"verbose"	
compress	当它被设置为true的时候对所有的服务器资源采用gzip压缩	false

inline和hot

inline自动刷新

当我们对业务代码做了一些修改然后保存后（ctrl+s），页面会自动刷新，我们所做的修改会直接同步到页面上，而不需要我们刷新页面，或重新开启服务

hot——模块热替换

在热替换（HMR：HotModuleReplacement）机制里，不是重载整个页面，HMR程序会只加载被更新的那一部分模块，然后将其注入到运行中的APP中

需要在配置项的plugins中增加插件：

```
plugins:[
    ....其他插件
    new webpack.HotModuleReplacementPlugin()//热替换插件
],
```

完整案例

一个具有完整基本功能的webpack.config.js配置文件：

```
let HtmlWebpackPlugin=require('html-webpack-plugin');
let webpack=require("webpack");

module.exports = {
    //输入
    entry:{
        main:__dirname + "/src/main.js",//已多次提及的唯一入口文件
    },
    //输出
    output: {
        path: __dirname + "/dist",//打包后的文件存放的地方
        filename:"main.js"//打包后输出的文件名
    },
    module: {
        rules: [
            {
                test: /.css$/, //解析css，并把css添加到html的style标签里
                use: ['style-loader', 'css-loader']
            },
        ]
    },
    plugins:[
        new HtmlWebpackPlugin({
            filename:"index.html", //生成的新文件
            template:__dirname+"/src/index_temp.html",  //模板文件
            minify:{ //压缩
                removeComments:true,  //删除注释
                collapseWhitespace:true  //合并空格
            },
        }),
        new webpack.HotModuleReplacementPlugin()//热替换插件
    ],
    devServer: {
        inline:true,
        hot:true
    }
}
```

```
package.js中增加命令
-----------------------------------------------------------------------
"scripts": {
    "dev": "webpack --mode development",
    "build": "webpack --mode production",
    "start":"node_modules/.bin/webpack-dev-server --mode development --port 8080"
  },
```

配置完成后，打开浏览器，进入网站，在console窗口会有如下显示：

```
[HMR] Waiting for update signal from WDS...
我是a.js
我是b.js
我是adsasindex.js
[WDS] Hot Module Replacement enabled.
>
```

> HMR：模块热替换(Hot Module Replacement)
>
> WDS：webpack dev server

注意：

webpack-dev-server运行后，浏览器中输出的页面，都是运行在内存中的，只有build以后，才会在dist目录中得到最终的结果文件。

附录：

课程所需模块列表（以下命令请手动输入，从pdf文件直接copy会产生错误）：

```
常用工具模块
--------------------------------------------------------------------------------
• 自动刷新页面工具
npm install live-server -g

• 自动重启http服务工具
npm install supervisor -g


vue-cli工具
--------------------------------------------------------------------------------
• 全局安装vue-cli
npm install vue-cli -g


webpack工具
--------------------------------------------------------------------------------
• 安装全局webpack
npm install webpack -g

• 安装全局webpack-cli
npm install webpack-cli -g

• 本地依赖安装
npm install webpack webpack-cli --save-dev

• html-webpack-plugin插件
npm install html-webpack-plugin --save-dev

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

选装模块（以下模块，不是非必须安装）

- loader 这些loader文件在vue-cli初始化时，会被默认安装

```
npm install babel-loader babel babel-core css-loader style-loader url-loader file-loader less-loader less --save-dev
```

- element-ui

```
npm install element-ui --save-dev
```