



Wear
Weiss Fi
Watch
(White Dial)
\$950.00

Camping
Hatchet
\$42.00
-30%



Home Decor
Handmade Picking Baskets
\$129.00



WEICHUANG
webpack

谢成

Mu,
Shaving
\$45.00

SHOP NOW



CONDITION

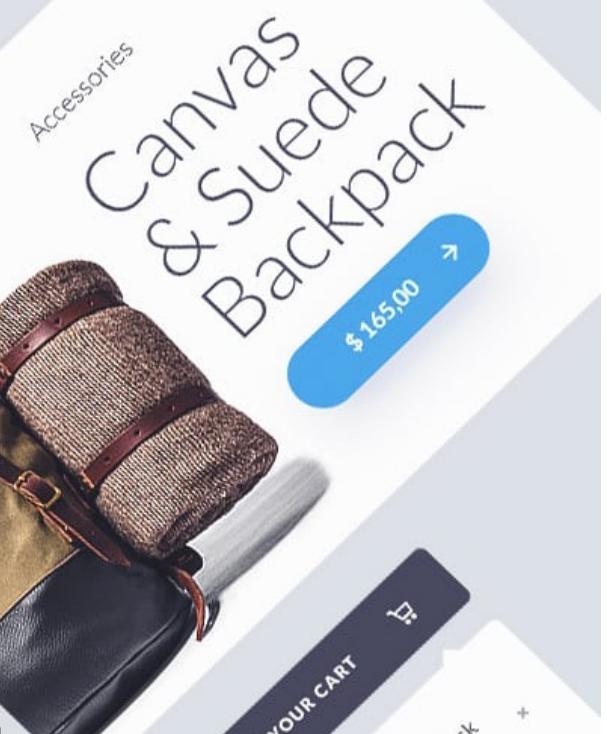
- New
- Manufacturer Refurbished
- User Refurbished
- Not Working



2 ITEMS IN YOUR CART
Marshall Pack
Qty: 1
\$359.00

EDC Kit 2
Qty: 1
\$229.00

DISCOUNT
TOT:



Accessories

Canvas
& Suede
Backpack
\$165.00

为什么要使用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

...

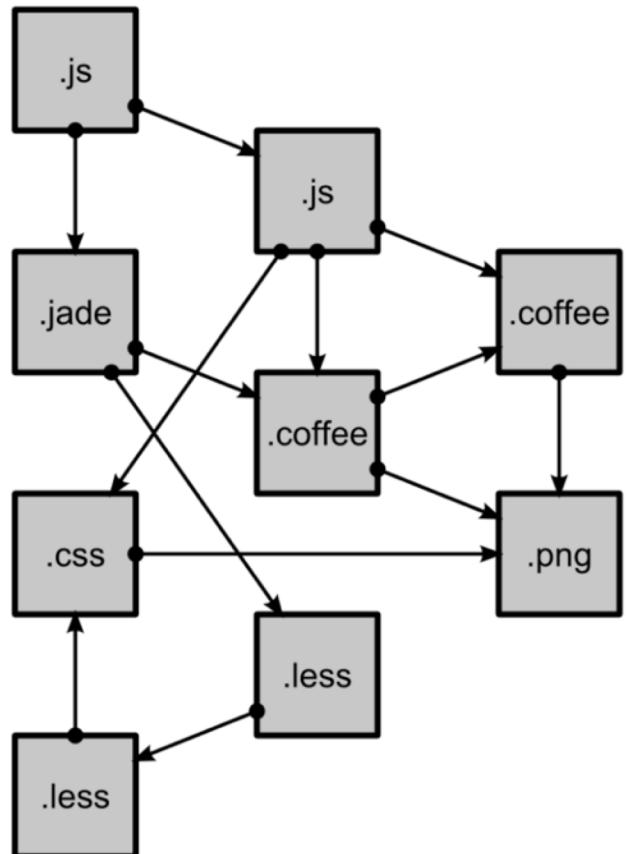
这些改进确实大大的提高了我们的开发效率，但是利用它们开发的文件往往需要进行额外的处理才能让浏览器识别，而手动处理又是非常繁琐的，这就为WebPack类的工具的出现提供了需求。

什么是webpack

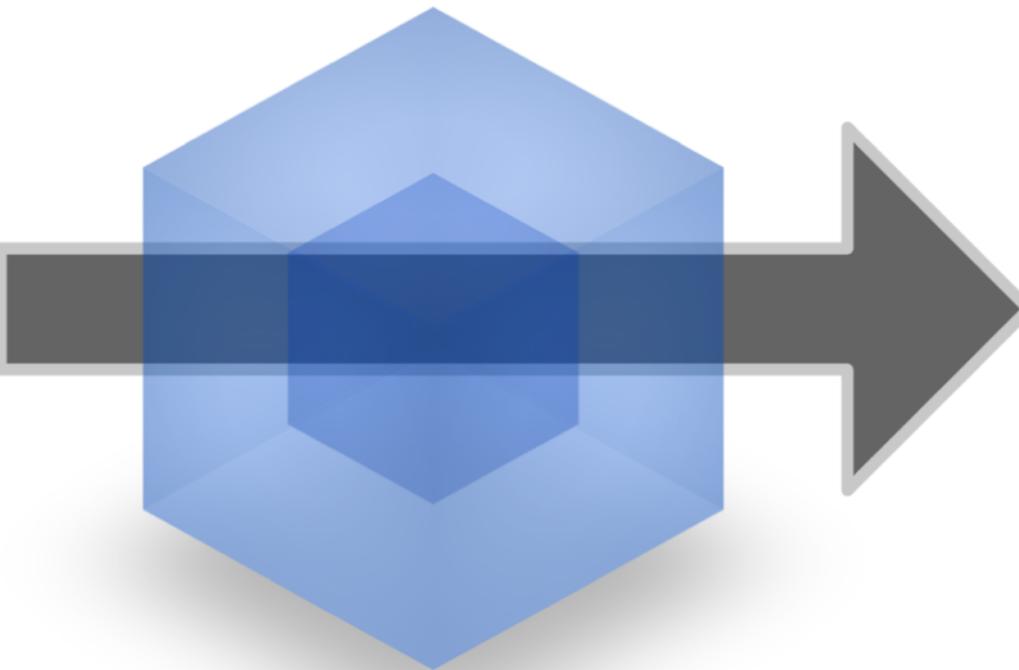
WebPack可以看做是模块打包机：它做的事情是，分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其转换和打包为合适的格式供浏览器使用。

webpack 4.X版本

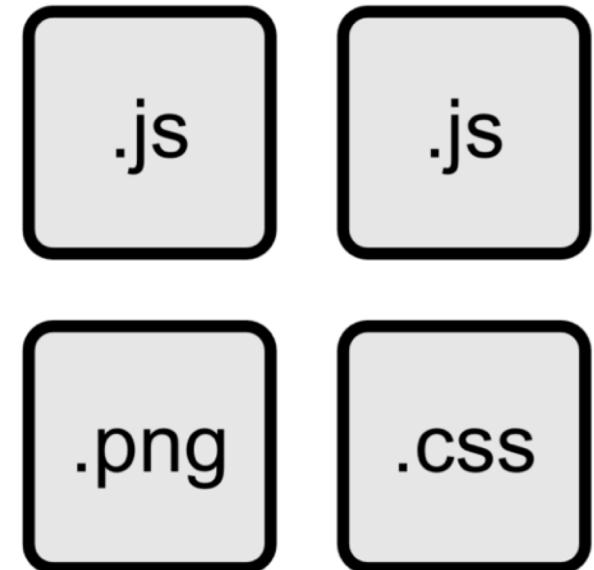
- 打包
- 转换（比如sass、less等）
- 优化



modules
with dependencies



webpack
MODULE BUNDLER



static
assets

英文: <https://webpack.js.org/>

中文: <https://doc.webpack-china.org/>

Github: <https://github.com/webpack/webpack>

安装webpack

安装node

查看版本: node -v

全局安装(不建议): npm install -g webpack

查看版本: webpack -v

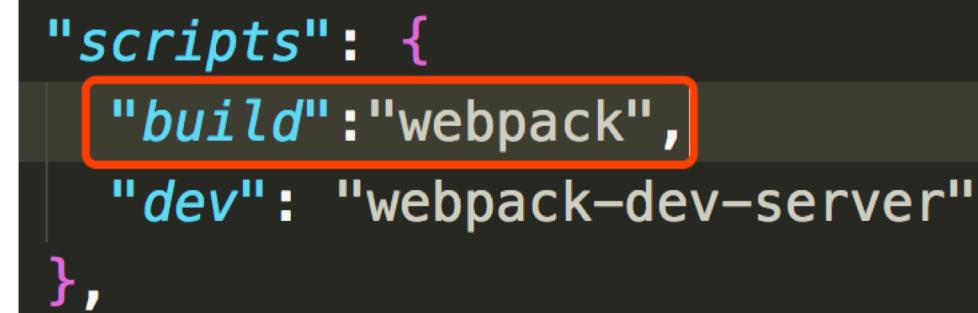
建议项目安装, 进入项目目录:

npm init

npm install webpack webpack-cli --save-dev

注意:

- ① Mac全局安装需要在命令前面加: sudo
- ② 如果npm安装比较慢, 可以使用yarn或者cnpm(<https://npm.taobao.org/>)



```
"scripts": {  
  "build": "webpack",  
  "dev": "webpack-dev-server"  
},
```

webpack.config.js

```
son  webpack.config.js x

const path = require('path');
module.exports = {
  mode: 'development',
  entry: {
    'index': './src/index.js',
    'index2': './src/index.js'
  },
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: '[name].js'
  }
};
```

mode:
development
production

入口与出口

webpack.config.js

```
1 const path = require('path');
2 module.exports = {
3   //入口文件的配置项
4   entry: {
5     entry: './src/entry.js',
6     entry2: './src/entry2.js'
7   },
8   //出口文件的配置项
9   output: {
10     path: path.resolve(__dirname, 'dist'),
11     filename: '[name].js'
12   },
13   //模块：例如解读CSS，图片如何转换，压缩
14   module: {},
15   //插件，用于生产模版和各项功能
16   plugins: [],
17   //配置webpack开发服务功能
18   devServer: {}
19 }
```

- **entry:** 配置入口文件的地址，可以是单一入口，也可以是多入口。
- **output:** 配置出口文件的地址，在 webpack2.X版本后，支持多出口配置。
- **module:** 配置模块，主要是解析 CSS和图片转换压缩等功能。
- **plugins:** 配置插件，根据你的需要配置不同功能的插件。
- **devServer:** 配置开发服务功能

pm run build命令打包

webpack.config.js

```
17 //配置webpack开发服务功能
18 devServer: {
19     //设置基本目录结构
20     contentBase: path.resolve(__dirname, 'dist'),
21     //服务器的IP地址，可以使用IP也可以使用localhost
22     host: 'localhost',
23     //服务端压缩是否开启
24     compress: true,
25     //配置服务端口号
26     port: 8081
27 }
```

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

webpack-dev-server (原因：服务装到了node_modules里面，找不到)

```
"scripts": {
  "build": "webpack",
  "dev": "webpack-dev-server"
},
```

webpack.config.js

```
const webpack = require('webpack');
```

```
new webpack.HotModuleReplacementPlugin()  
],  
devServer: {  
    contentBase: path.resolve(__dirname, 'dist'),  
    host: '127.0.0.1',  
    port: '8081',  
    compress: true,  
    open: true,  
    hot: true  
}
```

HTML文件打包

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

```
3 const HtmlWebpackPlugin= require('html-webpack-plugin');
```

```
24 //插件，用于生产模版和各项功能
25 plugins:[
26     // new UglifyJsPlugin(),
27     new HtmlWebpackPlugin({
28         minify:{
29             removeAttributeQuotes:true
30         },
31         hash:true,
32         template:'./src/index.html'
33     })
34 ];
```

- **minify:**
`removeAttributeQuotes:true`去掉属性的双引号。
`collapseWhitespace: true`折叠空白区域
- **hash:** 为了开发中js有缓存效果，所以加入hash，这样可以有效避免缓存JS
- **template:** 是要打包的html模版路径和文件名称

多页面

```
new HtmlWebpackPlugin({
  filename: 'a.html',
  title: 'index1-title',
  chunks: ['index11'],
  minify: {
    removeAttributeQuotes: true
  },
  hash: true,
  template: './src/index.html'
}),
new HtmlWebpackPlugin({
  filename: 'b.html',
  title: 'index2-title',
  chunks: ['index22'],
  minify: {
    removeAttributeQuotes: true
  },
  hash: true,
  template: './src/index2.html'
})
```

CSS文件打包

在入口js文件中： `import css from './css/index.css';`

style-loader: 用来处理css文件中的url()等， url挂在到js中

css-loader: 用来将css插入到页面的style标签

安装**style-loader**: `npm install --save-dev style-loader`

安装**css-loader**: `npm install --save-dev css-loader`

```
13 //模块：例如解读CSS,图片如何转换, 压缩
14 module: {
15     rules: [
16         {
17             test: /\.css$/,
18             use: ['style-loader', 'css-loader']
19         }
20     ]
21 },
```

CSS分离(mini-css-extract-plugin)

<https://github.com/webpack-contrib/extract-text-webpack-plugin>

安装: `npm install --save-dev extract-text-webpack-plugin@next`

```
4 const ExtractTextPlugin = require("extract-text-webpack-plugin");
```

```
module: {
  rules: [
    {
      test: /\.css$/,
      // use: ['style-loader', 'css-loader']
      use: ExtractTextPlugin.extract({
        fallback: "style-loader",
        use: "css-loader"
      })
    },
  ],
}
```

```
plugins: [
  // new UglifyJsPlugin(),
  new HtmlWebpackPlugin({
    minify: {
      removeAttributeQuotes: true
    },
    hash: true,
    template: './src/index.html'
  }),
  new ExtractTextPlugin("css/index.css")
],
```

webpack.config.js

```
2 const UglifyJsPlugin = require('uglifyjs-webpack-plugin');  
23 //插件，用于生产模版和各项功能  
24 plugins: [  
25     new UglifyJsPlugin()  
26 ],
```

**Webpack4.X:
mode:production**

CSS中引用图片

```
background-image: url(..../images/logo.png);
```

npm install --save-dev file-loader url-loader

```
19 module: {
20   rules: [
21     {
22       test: /\.css$/,
23       // use: ['style-loader', 'css-loader']
24       use: ExtractTextPlugin.extract({
25         fallback: "style-loader",
26         use: "css-loader"
27       })
28     }, {
29       test: /\.(png|jpg|gif)/,
30       use: [
31         {
32           loader: 'url-loader',
33           options: {
34             limit: 500,
35             outputPath: 'images/'
36           }
37         }
38       ]
39     },
40   ],
41 }
```

- **test: \.(png|jpg|gif)** 是匹配图片文件后缀名称。
- **use:** 是指定使用的loader和loader的配置参数。
- **limit:** 是把小于**500000B**的文件打成Base64的格式，写入css。

```
12 //出口文件的配置项
13 output: {
14   path: path.resolve(__dirname, 'dist'),
15   filename: '[name].js',
16   publicPath: 'http://localhost:8081/' // 这里是重点
17 }
```

HTML中的图片打包

```
<div id="div3">  
    
</div>
```

<https://github.com/wzsxyz/html-withimg>

```
}, {  
  test: /\.(\htm|html)$/i,  
  loader: 'html-withimg-loader'  
}
```

Sass打包和分离

npm install --save-dev node-sass sass-loader

```
}, {  
  test: /\.scss/,  
  use: [  
    {  
      loader: 'style-loader'  
    },  
    {  
      loader: 'css-loader'  
    },  
    {  
      loader: 'sass-loader'  
    }  
  ]  
}
```

在js文件中导入scss文件

```
2 import sass from './css/common.scss';
```

```
use: ExtractTextPlugin.extract({  
  use: [  
    {  
      loader: "css-loader"  
    }, {  
      loader: "sass-loader"  
    }],  
  fallback: "style-loader"  
})
```

<https://github.com/postcss/postcss-loader>

新建文件： postcss.config.js

postcss.config.js x

```
1 module.exports = {  
2   plugins: [  
3     require('autoprefixer')  
4   ]  
5};
```

```
module: {  
  rules: [  
    {  
      test: /\.css$/,
// use: ['style-loader', 'css-loader']  
      use: ExtractTextPlugin.extract({  
        fallback: "style-loader",  
        use: [  
          loader: "css-loader",  
          options: {importLoaders: 1}  
        ], 'postcss-loader'  
      })  
    }, {
```

清除未使用的css

<https://github.com/webpack-contrib/purifycss-webpack>

npm install --save-dev purifycss-webpack purify-css

```
2 const glob = require('glob');
```

```
6 const PurifyCSSPlugin = require("purifycss-webpack");
```

```
new PurifyCSSPlugin({
  paths: glob.sync(path.join(__dirname, 'src/*.html')),
})
```

babel

npm install --save-dev babel-core babel-loader babel-preset-es2015
babel-preset-react

```
}, {  
  test: /\.jsx|js$/,
  use: {
    loader: 'babel-loader',
    options: {
      presets: [
        "es2015", "react"
      ]
    }
  },
  exclude: /node_modules/
}
```

babel-preset-env

打包注释

```
new webpack.BannerPlugin('成哥所有，翻版必究!')
```

模块化配置

```
▲ └── webpack_config
      └── JS entry_webpack.js
```

```
1 const entry = {
2   entry: './src/entry.js'
3 };
4 module.exports = entry;
```

```
8 const entry = require("./webpack_config/entry_webpack.js");
```

```
module.exports = {
  //入口文件的配置项
  entry: entry,
  //出口文件的配置项
};
```

开发环境与生产环境

`devDependencies` 存放测试代码依赖的包或构建工具的包

`dependencies` 存放项目或组件代码中依赖到的

安装全部项目依赖包: `npm install`

安装生产环境依赖包: `npm install --production`

`npm install jquery --save`

打包第三方类库1

npm install --save jquery

3 import \$ from 'jquery';

打包第三方类库2

```
9 const webpack = require("webpack");
```

```
new webpack.ProvidePlugin({  
    $: "jquery"  
})
```

打包第三方类库3(4.x取消了CommonsChunkPlugin)

JS entry_webpack.js x

```
1 const entry = {  
2   entry: './src/entry.js',  
3   jquery: 'jquery'  
4 };  
5 module.exports = entry;
```

```
new webpack.optimize.CommonsChunkPlugin({  
  //name对应入口文件中的名字  
  name: 'jquery',  
  //把文件打包到哪里, 是一个路径  
  filename: "assets/js/jquery.js",  
  //最小打包的文件模块数, 这里直接写2就好  
  minChunks: 2  
})
```

```
new webpack.optimize.CommonsChunkPlugin({  
  //name对应入口文件中的名字  
  name: ['jquery', 'vue'],  
  //把文件打包到哪里, 是一个路径  
  filename: "assets/js/[name].js",  
  //最小打包的文件模块数, 这里直接写2就好  
  minChunks: 2  
})
```

资源拷贝

npm install --save-dev copy-webpack-plugin

```
9 var CopyWebpackPlugin = require('copy-webpack-plugin');
```

```
new CopyWebpackPlugin([
  {
    from: __dirname + '/src/public',
    to: './public'
}])
```

Json文件

根目录下创建json文件：

```
{ } author.json ×  
1 {  
2   "name": "xiecheng",  
3   "age": 30,  
4   "company": "weichuang"  
5 }
```

```
JS entry.js ×  
12 let json = require('../author.json');  
13 document.querySelector('#json').innerHTML = `作者: ${json.name}, 年龄: ${json.age}, 公司: ${json.company}`;
```



Thank you

谢

谢

观

看