ChinaEdu
弘 成 教 育

# Webpack 入门指南

大数据应用组
唐莉

创新 · 分享 · 进步

# 目录

# 一、什么是 Webpack



## 1.1 出现原因

随着前端技术的发展，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：

1.模块化。（ES6 export/import）

2.类似于 TypeScript 这种在JavaScript基础上拓展的开发语言

3.Scss，less等CSS预处理器

4.其他：如浏览器兼容性处理

# 一、什么是 Webpack



## 1.1 出现原因

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**

Webpack类的工具应运而生

Grunt / Gulp

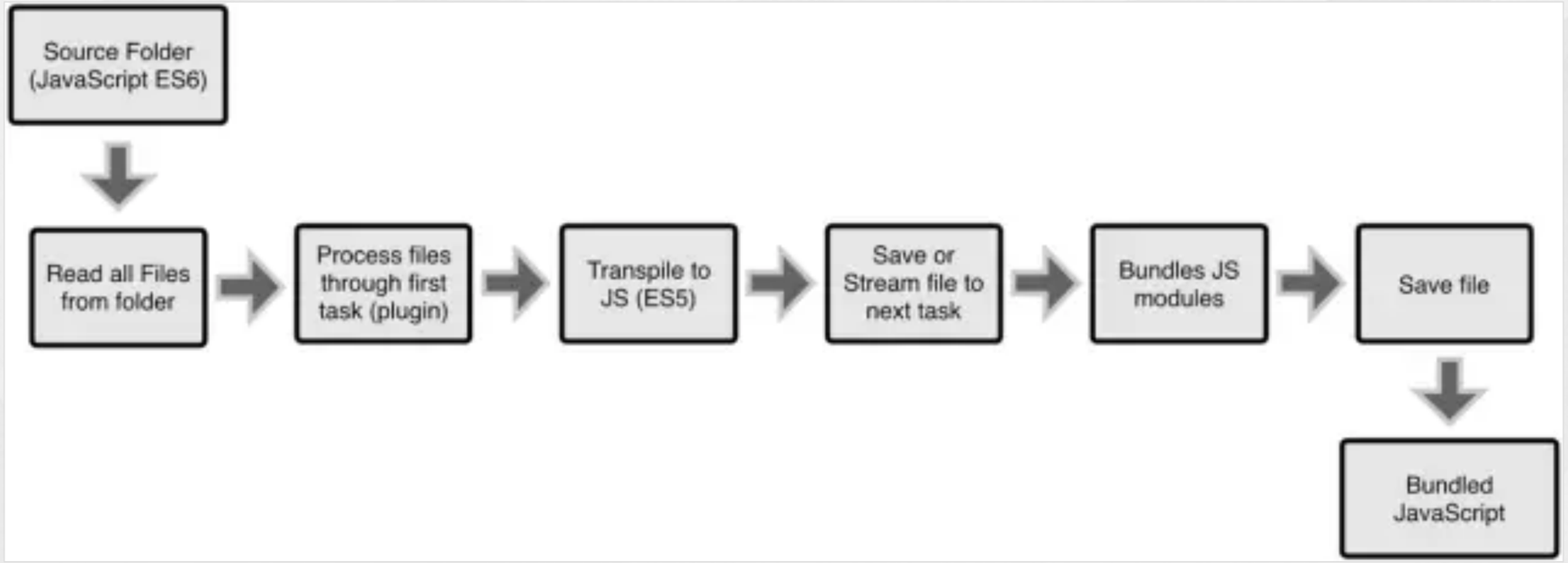
一种优化前端的开发流程的工具

Webpack

一种模块化的解决方案

## 1.2 二者异同 —— Grunt / Gulp

Grunt和Gulp的工作方式是：在一个配置文件中，指明对某些文件进行类似编译，组合，压缩等任务的具体步骤，工具之后可以自动替你完成这些任务。

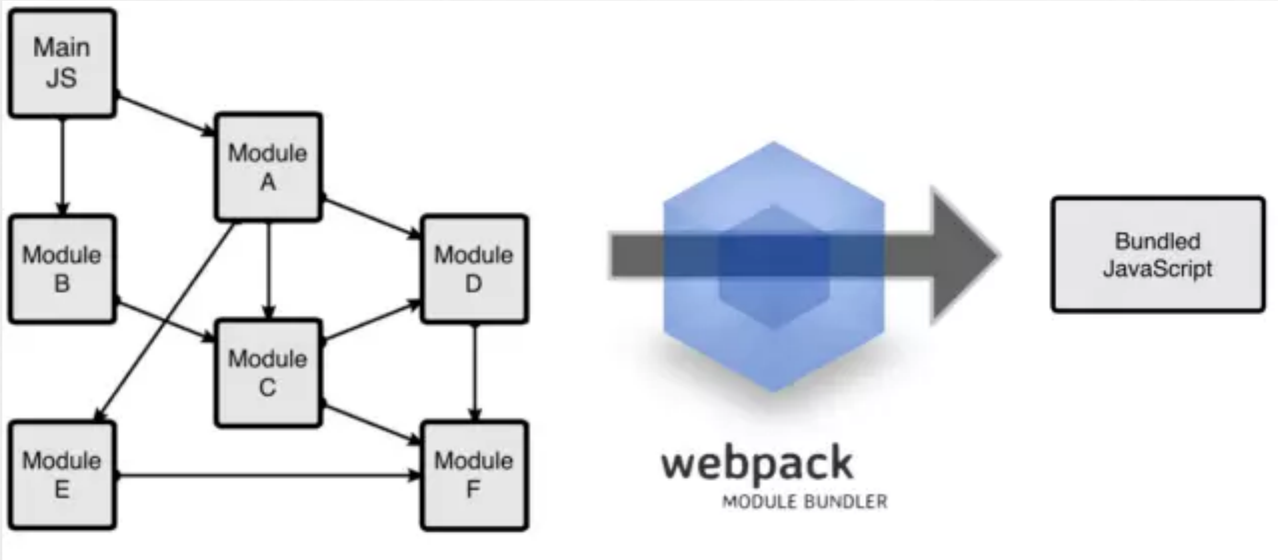


# 一、什么是 Webpack



## 1.2 二者异同 —— Webpack

Webpack的工作方式是：把你的项目当做一个整体，通过一个给定的主文件，Webpack将从这个文件开始找到你的项目的所有依赖文件，使用loaders处理它们，最后打包为一个（或多个）浏览器可识别的JavaScript文件。



## 简言之：Webpack 就是一个模块打包机

Webpack的处理速度更快更直接，能打包更多**不同类型**的文件。

| | |
|---|---|
| file-loader url-loader | 图片 字体 |
| style-loader css-loader | CSS 文件 |
| bable-loader | js/jsx 文件 |
| csv-loader xml-loader | CSV / TSV / XML 等数据文件 |

# 二、Webpack 的安装



通过 npm 进行安装, npm 在安装 node.js 时自带。也可用淘宝镜像 cnpm

全局安装：

```
npm install -g webpack
```

```
sudo npm install -g webpack
```

项目内安装：

```
npm install --save-dev webpack
```

# 三、Webpack 的使用



**mkdir learn-webpack**

创建项目目录

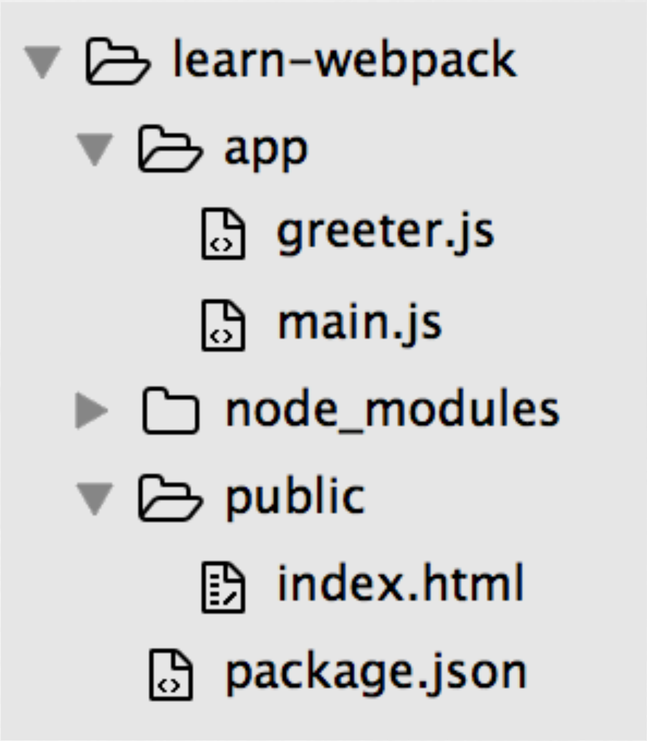
**npm init**

得到 package.json 文件

**npm install --save-dev webpack**

项目内安装 Webpack

创建文件得到如下目录



# 三、Webpack 的使用



## 3.1 正式使用Webpack

命令行使用 **webpack app/index public/bundle**

配置文件使用 根目录下建立文件 webpack.config.js **webpack**

```
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
  }
}
```

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。

# 三、Webpack 的使用



## 3.2 更快捷的执行打包任务

在 package.json 文件中设置 快捷命令 npm start

**npm start**

```
{
  "name": "learn-webpack",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "webpack"
  },
  "author": "tangli",
  "license": "ISC",
  "devDependencies": {
    "webpack": "^3.5.4"
  }
}
```

注：package.json 中的 script 会安装一定顺序寻找命令对应位置，本地的 node_modules/.bin 路径就在这个寻找清单中

## 4.1 调试 —— Source Maps

开发离不开调试，不过有时候通过打包后的文件，不易找到 bug。
**Source Maps** 就是来帮我们解决这个问题的。

为我们提供了一种对应编译文件和源文件的方法，使编译后的代码可读性更高

中小型项目： eval-source-map

大型项目： cheap-module-eval-source-map

构建速度更快，但是不利于调试，考虑时间成本时可使用。

# 四、Webpack 的更多功能



## 4.1 调试 —— Source Maps

配置 **Source Maps**，需要配置**devtool**，它有以下四种不同的配置选项

| devtool选项 | 配置结果 |
| --- | --- |
| `source-map` | 在一个单独的文件中产生一个完整且功能完全的文件。这个文件具有最好的`source map`，但是它会减慢打包速度； |
| `cheap-module-source-map` | 在一个单独的文件中生成一个不带列映射的`map`，不带列映射提高了打包速度，但是也使得浏览器开发者工具只能对应到具体的行，不能对应到具体的列（符号），会对调试造成不便； |
| `eval-source-map` | 使用`eval`打包源文件模块，在同一个文件中生成干净的完整的`source map`。这个选项可以在不影响构建速度的前提下生成完整的`sourcemap`，但是对打包后输出的JS文件的执行具有性能和安全的隐患。在开发阶段这是一个非常好的选项，在生产阶段则一定不要启用这个选项； |
| `cheap-module-eval-source-map` | 这是在打包文件时最快的生成`source map`的方法，生成的`Source Map`会和打包后的`JavaScript`文件同行显示，没有列映射，和`eval-source-map`选项具有相似的缺点； |

# 四、Webpack 的更多功能



## 4.2 构建本地服务器

npm 安装 webpack-dev-server

```
npm i webpack-dev-server -D
```

| devserver的配置选项 | 功能描述 |
| --- | --- |
| contentBase | 默认webpack-dev-server会为根文件夹提供本地服务器，如果想为另外一个目录下的文件提供本地服务器，应该在这里设置其所在目录（本例设置到“public"目录） |
| port | 设置默认监听端口，如果省略，默认为”8080“ |
| inline | 设置为`true`，当源文件改变时会自动刷新页面 |
| historyApiFallback | 在开发单页应用时非常有用，它依赖于HTML5 history API，如果设置为`true`，所有的跳转将指向index.html |

## 4.2 构建本地服务器 (及热更新)

配置 webpack.config.js

```
module.exports = {
    devtool: "evel-source-map", //  中小型项目常用
    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
    },
    devServer: {
        contentBase: "./public", //本地服务器所加载的页面所在的目录
        historyApiFallback: true, //不跳转
        inline: true //实时刷新
    }

}
```

## 4.2 构建本地服务器

在 package.json 文件中设置快捷命令 **npm run server**

```
{
  "name": "learn-webpack",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "webpack",
    "server": "webpack-dev-server --open"
  },
  "author": "tangli",
  "license": "ISC",
  "devDependencies": {
    "webpack": "^3.5.4",
    "webpack-dev-server": "^2.7.1"
  }
}
```

## 4.3 管理资源

通过使用不同的 Loader，Webpack 可以实现对不同格式的文件的处理。

比如说：

1. 分析转换 scss 为 css
2. 把下一代的JS文件（ES6，ES7) 转换为现代浏览器兼容的JS文件
3. 对React的开发而言，JSX文件转换为JS文件。

注意: Loaders需要通过 npm 单独安装

并且需要在 webpack.config.js 中的 modules关键字下进行配置

# 四、Webpack 的更多功能

ChinaEdu 弘成教育

## 4.3 管理资源 —— Bable > js/jsx

Bable：一个编译JavaScript的平台.　　Babel 官网　　http://babeljs.io

使用：　Babel其实是几个模块化的包，其核心功能位于称为 bable-core 的npm包中，webpack可以把其不同的包整合在一起使用,按需安装使用即可。

（用得最多的是解析Es6的babel-preset-es2015包和解析JSX的babel-preset-react包）。

安装：　**npm i  -D babel-core babel-loader**

**npm i  -D babel-preset-es2015 babel-preset-react**

## 4.3 管理资源 —— Bable > js/jsx

配置： webapck.config.js 中配置：

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

# 四、Webpack 的更多功能

ChinaEdu 弘成教育

## 4.3 管理资源 —— Bable > js/jsx

配置：　把babel的配置选项放在一个单独的名为 ".babelrc" 的配置文件中

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

webpack.config.js

```
{
  "presets": ["react", "es2015"]
}
```

.babelrc

```
module: {
    rules: [{
        test: /(\.jsx|\.js)$/,
        use: {
            loader: "babel-loader"
        },
        exclude: /node_modules/
    }]
}
```

webpack.config.js

# 四、Webpack 的更多功能

ChinaEdu 弘成教育

## 4.3 管理资源 —— CSS

### CSS 常用处理器

- **css-loader** 能使类似@import 和 url(...)的方法实现 require()的功能
- **style-loader** 将所有的计算后的样式加入页面中

### CSS 预处理器

- `Less Loader / Sass Loader`
- `Stylus Loader`
- `PostCSS Loader` 为CSS代码自动添加适应不同浏览器的CSS前缀

安装：

```
npm i  -D css-loader style-loader
npm i  -D postcss-loader autoprefixer
```

# 四、Webpack 的更多功能



## 4.3 管理资源 —— CSS

在 webpack.config.js 中配置

```
module: {
    rules: [{
        test: /(\.jsx|\.js)$/,
        use: {
            loader: "babel-loader"
        },
        exclude: /node_modules/
    }, {
        test: /\.css$/,
        use: [
            "style-loader",
            "css-loader",
            "postcss-loader"
        ]
    }]
}
```

```
// postcss.config.js
module.exports = {
    plugins: [
        require('autoprefixer')
    ]
}
```

在 postcss.config.js 中配置

## 4.4 插件 plugins —— html-webpack-plugin

**用于拓展Webpack功能，它们会在整个构建过程中生效，执行相关的任务。**

作用: 依据一个简单的模板，生成一个自动引用打包后的JS文件的新index.html。这在每次生成的js文件名称不同时非常有用（比如添加了 hash 值）

```
plugins: [
    new HtmlWebpackPlugin({
        template: __dirname + "/app/index.tmpl.html" //new 一个这个插件的实例
    })
],
```

ChinaEdu
弘 成 教 育

# THANKS