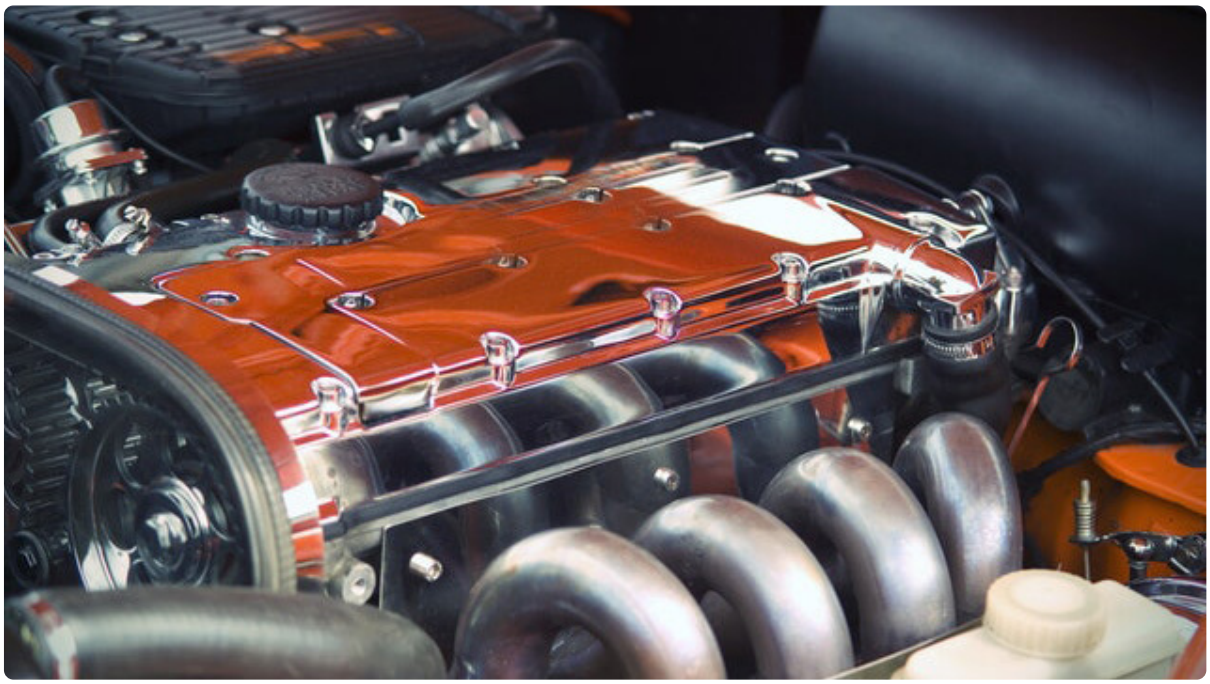
# 15 Webpack 中配置React和Vue开发环境

更新时间：2019-06-24 09:27:56



> 上天赋予的生命，就是要为人类的繁荣和平和幸福而奉献。
>
> ——松下幸之助

当下前端最火的两个框架就是了 React 和 Vue 了，本篇文章主要介绍手动配置 Webpack 来做 React 和 Vue 的开发。

> Facebook 官方推出的 create-react-app ，Vue 也有自己的 CLI 工具 Vue-CLI，两个工具都已经非常好用，但在实际项目中，我们仍然需要做一些修改才可以满足实际项目上线的需求，同时我们仍希望有更多所谓个性化设置来支持项目。所以掌握 Webpack 中 React 和 Vue 的配置还是很有必要的事情。

Webpack 的基本配置讲解部分已经接近尾声，下面我们完全按照一个新项目来创建 React 和 Vue 的 Webpack 配置，通过项目的配置帮助我们回忆下 Webpack 基本配置的一些知识点。

## Webpack 中配置 React 的开发

首先是创建目录、初始化 package.json、安装 React、安装 Webpack 和安装 Babel。

```
# 新建目录，并且进入
mkdir react-webpack && cd $_
# 创建 package.json
npm init -y
# 安装 react react-dom依赖
npm i react react-dom
# 安装 webpack 和 webpack-cli 开发依赖
npm i webpack webpack-cli -D
# 安装 babel
npm i babel-loader @babel/core @babel/preset-env -D
# 安装 babel preset-react
npm i @babel/preset-react -D
```

因为 React 中提供了一种 JavaScript 的扩展语法，React 将它们命名为 jsx，所以在这里我们需要给 jsx 文件添加 Babel 的编译支持，现在我们新建一个 webpack.config.js 的配置文件，内容和注释如下：

```
module.exports = {
  resolve: {
    extensions: ['.wasm', '.mjs', '.js', '.json', '.jsx']
  },
  module: {
    rules: [
      {
        test: /\.jsx?$/, // jsx/js文件的正则
        exclude: /node_modules/, // 排除 node_modules 文件夹
        use: {
          // loader 是 babel
          loader: 'babel-loader',
          options: {
            // babel 转义的配置选项
            babelrc: false,
            presets: [
              // 添加 preset-react
              require.resolve('@babel/preset-react'),
              [require.resolve('@babel/preset-env'), {modules: false}]
            ],
            cacheDirectory: true
          }
        }
      }
    ]
  }
};
```

> Tips： 注意这里将 Babel 的配置直接放到 webpack.config.js 中，没有单独放到 .babelrc，这是一个推荐写法，因为在某些打包（上线构建）机器，对于 . 开头的 dotFile 支持并不好，所以建议 .babelrc 这类 dotFile 在 webpack.config.js 中显性声明，显性声明的配置还能第一时间在 Webpack 配置中被找到。

然后在 src 文件夹下创建一个 App.jsx 的文件：

```
// App.jsx
import React from 'react';
import ReactDOM from 'react-dom';
const App = () => {
  return (
    <div>
      <h1>Hello React and Webpack</h1>
    </div>
  );
};
export default App;
ReactDOM.render(<App />, document.getElementById('app'));
```

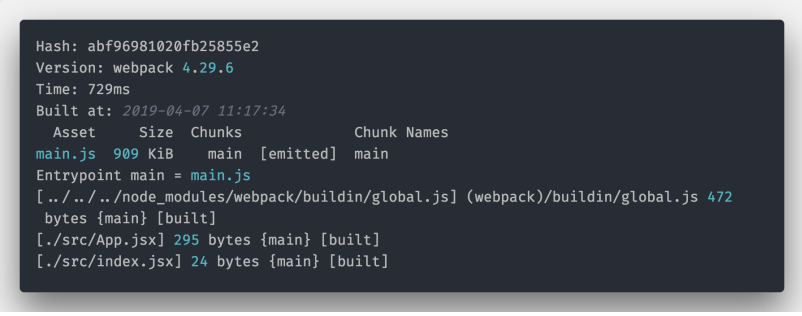
在创建一个 index.jsx 文件：

```
// index.jsx
import App from './App'; // 这里可以省略.jsx
```

在 webpack.config.js 中添加 entry：

```
module.exports = {
  entry: './src/index.jsx'
  // ...
};
```

这时候可以执行下 npx webpack --mode development 看下，文件已经打出来了。




接下来我们创建一个 HTML 文件（ src/index.html ），作为项目的模板，内容如下：

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="UTF-8" />
    <title>Hello React Webpack</title>
  </head>
  <body>
    <div id="app"></div>
  </body>
</html>
```

我们需要使用 html-webpack-plugin 插件来复制 index.html 到 dist 文件夹下：

```
# 首先是安装 html-webpack-plugin
npm i html-webpack-plugin -D
```

修改 webpack.config.js

```
//webpack.config.js
const HtmlWebPackPlugin = require('html-webpack-plugin');
module.exports = {
  // ...
  plugins: [
    new HtmlWebPackPlugin({
      template: 'src/index.html',
      filename: 'index.html'
    })
  ]
};
```

完成之后，执行 npx webpack --mode development 就可以看到打包结果了：

好了，现在可以将命令放到 npm 的 scripts 了：

```
{
  "scripts": {
    "build": "webpack --mode production"
  }
}
```

**React 的开发阶段配置**

## 配置 React 开发的 webpack-dev-server

首先使用 webpack-dev-server 搭建一个本地开发服务，安装 webpack-dev-server 依赖：

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

在 package.json 增加 start 的命令：

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

为了方便开发，我们将新建了一个 webpack.config.dev.js 的配置文件：

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

const HtmlWebPackPlugin = require('html-webpack-plugin');

module.exports = {
   mode: 'development',
   devtool: 'cheap-module-source-map',
   devServer: {
      contentBase: path.join(__dirname, './src/'),
      publicPath: '/',
      host: '127.0.0.1',
      port: 3000,
      stats: {
         colors: true
      }
   },
   entry: './src/index.jsx',
   // 将 jsx 添加到默认扩展名中，省略 jsx
   resolve: {
      extensions: ['.wasm', '.mjs', '.js', '.json', '.jsx']
   },
   module: {
      rules: [
         {
            test: /\.jsx?$/, // jsx文件的正则
            exclude: /node_modules/, // 排除 node_modules 文件夹
            use: {
               // loader 是 babel
               loader: 'babel-loader',
               options: {
                  // babel 转义的配置选项
                  babelrc: false,
                  presets: [
                     // 添加 preset-react
                     require.resolve('@babel/preset-react'),
                     [require.resolve('@babel/preset-env'), {modules: false}]
                  ],
                  cacheDirectory: true
               }
            }
         }
      ]
   },
   plugins: [
      new HtmlWebPackPlugin({
         template: 'src/index.html',
         filename: 'index.html',
         inject: true
      })
   ]
};
```

对应的 package.json 的 scripts 也修改下，start 添加 config 文件路径：

```
{
   "scripts": {
      "build": "webpack --mode production",
      "start": "webpack-dev-server --config './webpack.config.dev.js' --open "
   }
}
```

到这里，可以执行下 npm start 看下效果了！

> Tips：当然还可以将 webpack.config.js 继续拆分，将公共部分放到 webpack.config.base.js 部分，然后使用 webpack-merge 来合并配置项，这里不再展开，参考[TODO](dev。md 相关的文件链接)。

## 配置 React 的 HMR

热更新是开发阶段很好用的一个功能，配置了热更新就可以让我们在开发过程中，将修改后代码整页面无刷新且保持原有 state 的情况下直接反应到页面，下面我们继续修改 webpack.config.dev.js 并在 App.jsx 增加内容：

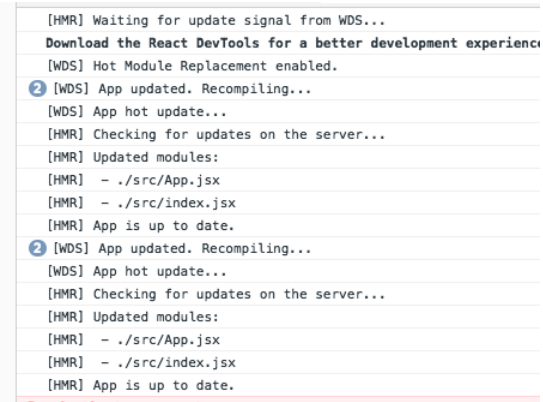
```
// webpack.config.dev.js
+ const webpack = require('webpack');

module.exports = {
   devServer: {
   ...
+      hot: true,
   ...
   },
   ...
   plugins: [
+      new webpack.HotModuleReplacementPlugin(),
   ...
   ]
   ...
}
```

还需要给对应的文件添加 HMR 对应的代码，打开 index.jsx 在末尾添加：

```
if (module.hot) {
   module.hot.accept(err => {
      if (err) {
         console.error('Cannot apply HMR update.', err);
      }
   });
}
```

最后，执行 npm start，会启动 webpack-dev-server，修改下 App.jsx 的内容，控制台被重新编译触发了，同时浏览器的热更新已经生效了：




上面给 index.jsx 末尾添加的代码，是触发 HMR 的，这部分代码应该不属于业务代码，直接写在文件内对我们代码的侵入性实在是太大，所以建议将它们存入一个 dev.js 文件中，然后修改 webpack.config.dev.js 的 entry，将 index.jsx 和 dev.js 合并在一起。

```
// webpack.config.dev.js
module.exports = {
   //...
   entry: ['./src/index.jsx', './src/dev.js']
   //...
};
```

## Webpack 中配置 Vue 的开发

上面 React 部分已经详细讲解了一些重复的步骤，这里就简单说下操作步骤和代码。

首先也是创建目录、初始化 package.json、安装 Vue、安装 Webpack 和安装 Babel。

```
# 新建目录，并且进入
mkdir vue-webpack && cd $_
# 创建 package.json
npm init -y
# 安装 vue 依赖
npm i vue
# 安装 webpack 和 webpack-cli 开发依赖
npm i webpack webpack-cli -D
# 安装 babel
npm i babel-loader @babel/core @babel/preset-env -D
# 安装 loader
npm i vue-loader vue-template-compiler -D
# 安装 html-webpack-plugin
npm i html-webpack-plugin -D
```

然后在 src 文件夹下新建 App.vue 和 index.js 两个文件，内容如下：

```
// app.js
import Vue from 'vue';
import App from './app.vue';

Vue.config.productionTip = false;

new Vue({
  render: h => h(App)
}).$mount('#app');
```

```
<template>
  <div id="app">
    Hello Vue & Webpack
  </div>
</template>

<script>
  export default {};
</script>
```

然后创建一个 HTML 文件 index.html：

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="UTF-8" />
    <title>Webpack Vue Demo</title>
  </head>
  <body>
    <div id="app"></div>
  </body>
</html>
```

最后配置 webpack.config.js，内容如下：

```
const HtmlWebpackPlugin = require('html-webpack-plugin');
const VueLoaderPlugin = require('vue-loader/lib/plugin');

module.exports = {
  resolve: {
    alias: {
      vue$: 'vue/dist/vue.esm.js'
    },
    extensions: ['*', '.js', '.vue', '.json']
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: {
          loader: 'babel-loader'
        }
      },
      {
        test: /\.vue$/,
        loader: 'vue-loader'
      }
    ]
  },
  plugins: [
    new VueLoaderPlugin(),
    new HtmlWebpackPlugin({
      template: './src/index.html',
      filename: 'index.html'
    })
  ]
};
```

Vue 的配置文件跟 React 最大区别是，React 是直接扩展了 Babel 的语法，而 Vue 语法的模板还需要使用 vue-loader 来处理。

完成上面配置后，执行下 npx webpack 看下 dist 产出吧。开发相关的配置跟 React 部分基本一致，这里不再重复介绍了。

小结

本文从一个项目搭建开始介绍了 React 和 Vue 两个时下最流行的 JavaScript 库的 Webpack 配置。由于 React 和 Vue 自身语法跟普通的 ES6 语法不同，所以需要配置对应的 loader 或者 Babel 插件。

14 Webpack Dev Server 本地开发服务

16 Webpack 环境相关配置及配置文件拆分

**精选留言 0**

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论