

——高性能PHP应用开发之

第13讲 NPM和Webpack

目录

CONTENTS

1 / 使用NPM

2 / 使用Webpack

3 / Laravel前端资源

目录

CONTENTS

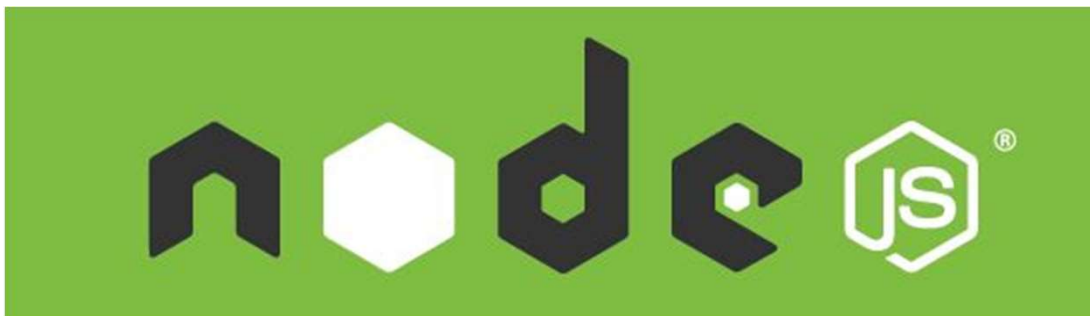
1 / 使用NPM

2 / 使用Webpack

3 / Laravel前端资源

Node.js和NPM

- Node.js 是运行在服务端的 JavaScript，它是一个基于 Chrome V8 引擎和事件驱动、非阻塞I/O模型的 JavaScript 运行环境；V8引擎执行JavaScript的速度非常快，性能很好。
- ✓ Node.js主要用来承担服务器端应用程序开发工作
- ✓ Node.js 的包管理器 **npm**，是全球最大的开源库生态系统，现在已经成为前端开发的标准配置，是前端包管理的最重要工具。



安装Node.js


■ node.js安装

- ✓ 下载地址: <https://nodejs.org/en/download/>
- ✓ 安装: 直接双击安装即可, 将会自动把 node和npm 工具写入命令行
- ✓ 通过 **node -v** 查看node.js版本, 以确认是否安装成功

LTS
Recommended For Most Users

Current
Latest Features


Windows Installer
node-v8.9.4-x86.msi


macOS Installer
node-v8.9.4.pkg


Source Code
node-v8.9.4.tar.gz

Windows Installer (.msi)
Windows Binary (.zip)
macOS Installer (.pkg)
macOS Binaries (.tar.gz)
Linux Binaries (x86/x64)
Linux Binaries (ARM)
Source Code

32-bit	64-bit	
32-bit	64-bit	
64-bit		
64-bit		
32-bit	64-bit	
ARMv6	ARMv7	ARMv8
node-v8.9.4.tar.gz		

使用NPM

- npm是前端依赖包管理工具，是一个命令行使用工具
- ✓ npm -v : 查看当前 npm 版本号
- ✓ **npm install 包名** : 安装指定的包到当前目录下
 - **-g 参数**: 全局安装
 - **--save 参数**: 写入到当前项目的 package.json 文件中
- ✓ npm uninstall 包名: 删除某个指定包
- ✓ npm update 包名: 更新某个指定包

使用NPM

- npm是前端依赖包管理工具，是一个命令行使用工具
- ✓ 安装淘宝镜像：npm install -g cnpm --registry=https://registry.npm.taobao.org
 - 后续**使用 cnpm 代替 npm**
- ✓ **使用 package.json 文件**：NPM包管理配置文件
 - 使用 **npm init 初始化** package.json 文件
 - ✓ dependencies属性：声明当前应用程序要使用的依赖库
 - ✓ npm install 命令 package.json 文件安装依赖关系

NPM包的使用

- npm下载的依赖包位于当前项目目录下的 node_modules 目录下
- ✓ 使用这些依赖包有两种使用方法：
- ✓ 直接引入 `<script src= "" ></script>`
- ✓ 使用 **webpack 打包工具** 打包使用



目录

CONTENTS

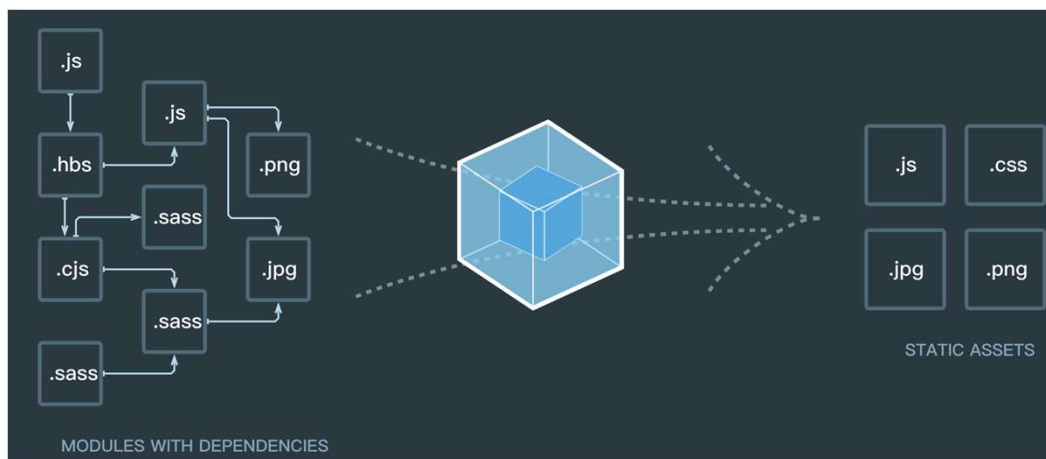
1 / 使用NPM

2 / 使用Webpack

3 / Laravel前端资源

安装Webpack

- webpack 是一个模块打包器，它的主要目标是将 JavaScript 文件打包在一起，打包后的文件用于在浏览器中使用，但它也能够胜任转换(transform)、打包(bundle)或包裹(package)任何资源(resource or asset)的工作。
- ✓ 使用 **cnpm install -g webpack webpack-cli** 全局安装 webpack
- ✓ 使用 **cnpm install --save-dev webpack** 在当前应用项目中安装webpack



使用Webpack

■ webpack 打包程序的作用：

- ✓ 把多个 js 脚本（或其它格式文件），打包成一个js文件
- ✓ 处理 es6 语法或其它浏览器不能理解的js代码（如 typescript）
- ✓ 打包命令：**webpack 入口文件 出口文件**
 - 入口文件：一般采用单一入口文件形式
 - 出口文件：打包后的js文件，需要在浏览器HTML中加载该文件

Webpack配置文件

- 使用webpack配置文件打包文件
- ✓ 在应用程序根目录创建 webpack.config.js 配置文件
 - **entry**: 入口文件, 一般使用单一入口
 - **output**: 出口文件, 由 path 和 filename 选项构成
 - **module.rules**: 调用外部的脚本或工具, 实现对不同格式的文件的处理
 - ✓ test: 待处理的文件格式 (扩展名)
 - ✓ loader: 外部工具名称
 - resolve.alias: 配置如何解析模块, 创建 import 或 require 的别名, 来确保模块引入变得更简单。

Webpack配置文件

- 使用webpack配置文件打包文件
- ✓ 在应用程序根目录创建 webpack.config.js 配置文件
- ✓ 直接使用 **npx webpack** 命令打包应用程序
- ✓ 注意，webpack4+版本中，要在 webpack.config.js 配置文件中添加 mode 选项
mode: "development"

```
module.exports = {  
  entry: __dirname + '/src/main.js',  
  output: {  
    path: __dirname + '/public',  
    filename: 'bundle.js'  
  },  
  module: {  
    rules: [{  
      test: /\.vue$/,  
      loader: 'vue-loader'  
    }]  
  },  
  resolve: {  
    alias: {  
      'vue$': 'vue/dist/vue.esm.js'  
    }  
  }  
};
```

Webpack打包

- 使用webpack配置文件打包文件
- ✓ 可以直接在 `package.json` 配置文件（npm配置文件），在 `scripts` 段，添加 webpack 快速执行命令
- ✓ 此时，直接在应用程序根目录 **执行 `npm start`** 即可实现 webpack 程序的打包功能

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

webpack-dev-server

- webpack-dev-server开启本地调试服务器
 - ✓ 全局（或指定项目中）**安装 webpack-dev-server**，实现本地服务器调试开发
 - `cnpm install -g webpack-dev-server` 全局安装
 - ✓ 在 `webpack.config.json` 文件中，使用 **devServer** 段实现 调试服务器的配置
 - 默认端口号为 8080 端口
 - ✓ 在 `npm`配置文件中添加 **server**段 `"server": "webpack-dev-server --open"`

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 <code>true</code> ，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 <code>true</code> ，所有的跳转将指向index.html

使用Bootstrap

■ Bootstrap版本

- ✓ Bootstrap4+：可以直接被webpack打包处理
- ✓ Bootstrap3.3.7：需要添加一系列 loader 来处理

■ 使用Bootstrap3.3.7

- ✓ 安装指定的 loader：css-loader、style-loader、url-loader
- ✓ 修改webpack.config.js配置文件，添加loader解析

```
module: {  
  rules: [  
    {  
      test: /\.css$/,  
      use: ['style-loader', 'css-loader']  
    }, {  
      test: /\.?(eot|woff|woff2|ttf|svg)$/,  
      loader: ['url-loader']  
    }  
  ]  
}
```


NPM和Webpack参考文档

■ NPM参考文档

- ✓ node.js文档: <https://nodejs.org/en/docs/guides/>
- ✓ npm文档: <https://www.npmjs.com.cn/>

■ Webpack参考文档

- ✓ 官方文档: <https://doc.webpack-china.org/guides/>
- ✓ 入门教程: <https://www.jianshu.com/p/42e11515c10f>
- Es6模块教程: <http://es6.ruanyifeng.com/#docs/module>

目录

CONTENTS

1 / 使用NPM

2 / 使用Webpack

3 / Laravel前端资源

Laravel前端资源

■ 加载静态资源：

- ✓ 静态资源根目录为 /public 目录，只有该目录下的静态资源才会被Laravel加载。
- ✓ 使用 **asset()** 辅助函数，获取前端静态资源的路径。
- ✓ 注意：图片上传时，需要修改图片的上传目录为 public 目录下的子目录，而不能使用默认的 resources 目录，否则图片不能正确显示。

```
<link type="text/css" rel="stylesheet" href="{{ asset('css/app.css') }}" />>  
<script type="text/javascript" src="{{ asset('js/app.js') }}"></script>
```

Laravel前端资源

- 使用Webpack编译的前端资源：
 - ✓ 修改 package.json 文件，添加静态库依赖关系（类似于 composer.json ）
 - ✓ 执行 npm install 命令，下载静态资源库
 - ✓ 修改 webpack.mix.js 文件，设置静态资源编译配置
 - ✓ 编译：npm run dev （或 其它段）
 - ✓ 加载 /public 目录下的静态资源文件

感谢聆听！

THANK YOU FOR YOUR ATTENTION