**旅行日程- Vue开发说明**

变更历史:

|  |  |  |
| --- | --- | --- |
| **版本号** | **日期** | **变更描述** |
| 1.0 | 2018/08/22 | 创建 |
|  |  |  |
|  |  |  |

# 开发前准备

由于node.js的出现，使JavaScript可以运行在服务端。由于运行在服务端，所以需要封装一些可重用的组件（像C#中dll和java中的jar），从而进一步促进了JavaScript项目的项目化而非以前的一个一个的js文件。所以，目前创建前端项目都需要通过下面的技术栈来实现项目化：

node + npm + webpack + vue

当然还有其他技术栈选择，但不是本文的重点，就不再介绍了。

## 什么是Node.js

Node.js是一个JavaScript在本地的运行环境，其在Chrome v8引擎的基础上进行了一层封装，使JavaScript可以在本地执行而不用再在浏览器中执行。

作用：在开发前端项目，由于我们使用Visual Stuido Code进行开发，需要快速验证一些js代码，此时就可以通过node.js直接在本地创建一个web server供浏览器访问相关页面进行验证从而节省了我们自己将js部署到IIS或tomcat的时间。

下载地址以及官方文档：

https://nodejs.org/en/

## 什么是npm

npm在开始之初是专门服务于Node的包管理器，由于node.js的出现，所以js可以用来编写服务端逻辑，从而出现需要管理可重用的包（模块）的场景，npm便应运而生。后面，由于通过npm进行包（模块）的发布、管理以及引用非常方便，所以npm已经从node中独立出来，成为一个独立的应用。npm基本上等同于Java的maven.

作用：在前端项目中，提供模块的引用、安装已经管理。

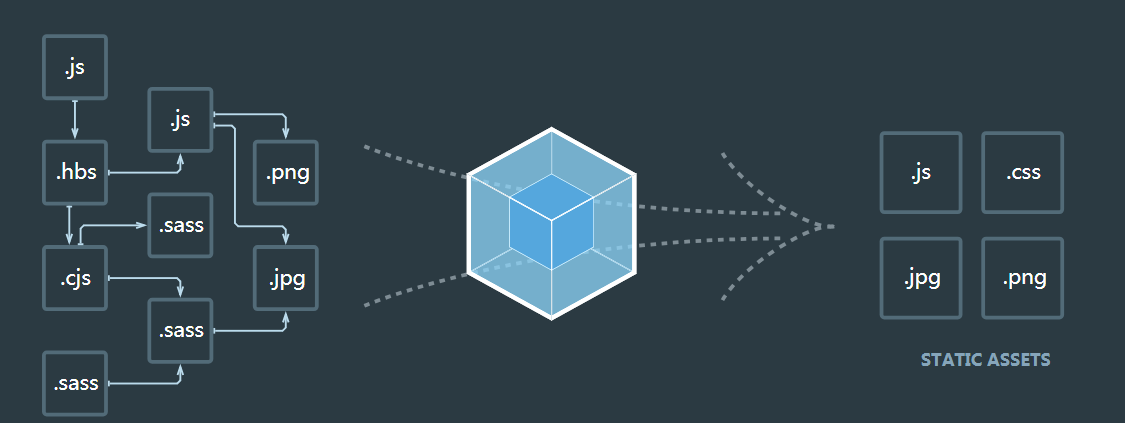
下载地址以及官方中文文档：

https://www.npmjs.com.cn/

## 什么是Webpack

随着技术的发展以及对生产力/性能的要求，我们在编写前端项目的代码时，可能并不会直接编写.js, .css, .html文件，取而代之的编写.vue, .less文件等。但浏览器只识别传统的.js, .css, .html文件，所以webpack会把我们编写的文件编译成相应的浏览器识别的静态文件，同时还提供压缩js/css，合并js/css，提供本地web server等等非常多的能力。

用官方的图片总结下webpack的功能：



下载安装：

npm install webpack

官方文档：

https://www.webpackjs.com/

## 什么是Vue.js

Vue是一套前端框架，不同于JQuery的是，JQuery仅仅提供了一套和DOM打交道的API，而Vue包含整套前端框架的解决方案，如下：

1. UI的编写
2. 数据和UI的双向绑定
3. 组件化
4. 前端路由（使用SPA应用）
5. 状态管理
6. 其他

所以，使用vue可以快速开发出高可用、可维护的前端代码。

官网：

<https://cn.vuejs.org/>

# 创建Vue项目

下文将会介绍如何从无到有的创建一个vue项目。

## 使用vue-cli来创建vue项目

vue-cli是一个vue的命令行工具，可供开发人员迅速创建一个vue的模板工程。

使用npm全局安装vue-cli工具，命令如下：

|  |
| --- |
| npm install vue-cli –g |

安装好后，使用cmd路由到希望存放代码的文件夹，然后使用如下命令创建一个vue项目：(注意，下面的project\_name是项目名，由开发人员随意填写)

|  |
| --- |
| vue init webpack project\_name |

使用该命令后会让用户选择一系列的选项，如：

项目的名称（注意，此处只能是小写）

项目的描述

项目的作者

项目是否包含编译时和运行时，还是只包含运行时？（这里通常选包含编译时+运行时）

是否安装vue-router

是否使用ESLint来规范代码，如果选择是的话，会让你选择规范类型：标准，Airbnb，无

是否建立单元测试

是否建立e2e 测试

是否使用NPM/Yarn/无来创建项目，通常选择NPM

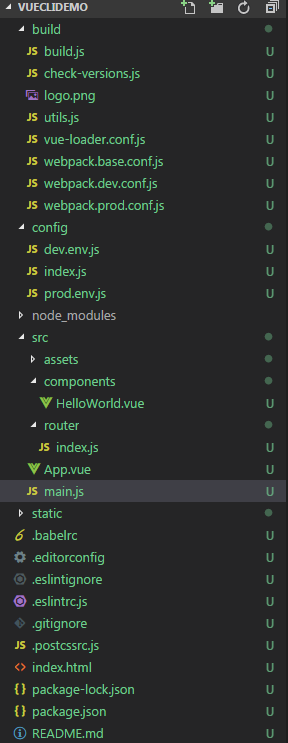
填写完上面选项后，等待下载各种组件，下载完成后，vue项目的模板工程就搭建完成了。

工程搭建好了后，使用cmd路由到代码所在的文件夹，然后运行如下命令，即可以运行该项目了：

|  |
| --- |
| npm run dev |

## 项目结构分析

项目创建好后，其项目结构如下：



下面我们将从上至下分析该项目结构中每个部分的作用以及介绍如何添加新的功能。

参考文章：

http://vuejs-templates.github.io/webpack/structure.html

https://www.cnblogs.com/amunamuna/p/8708102.html

### build文件夹

该文件夹包含webpack相关的配置文件，一般不需要进行修改，其中每个文件的作用如下：

|  |  |
| --- | --- |
| 文件名 | 用途 |
| build.js | webpack进行生产环境构建时依赖的配置文件 |
| check-version.js | 校验当前node的版本和npm的版本是否符合package.json中engines节点下指定的版本号 |
| logo.png | 使用npm run dev编译报错时在桌面通知中需要用到的icon，该icon被utils.js中的createNotifierCallback方法使用。 |
| utils.js | 提供一些供vue-loader.conf.js，webpack.base.conf.js，webpack.dev.conf.js，webpack.prod.conf.js使用的方法，方法如下：  assetsPath：获取资源路径  cssLoaders：生成各种样式文件的loader，转为vue-loader.conf.js文件使用  styleLoaders: 生成供webpack的配置文件使用的各种样式文件的loader  createNotifierCallback：在测试环境用npm run dev报错时，提供桌面通知回掉。 |
| vue-loader.conf.js | vue-loader依赖的配置文件 |
| webpack.base.conf.js | webpack依赖的配置文件，该配置文件会被下面两个配置文件继承：  webpack.dev.conf.js  webpack.prod.conf.js |
| webpack.dev.conf.js | 开发环境，webpack依赖的配置文件 |
| webpack.prod.conf.js | 生产环境，webpack依赖的配置文件 |

### config文件夹

该文件夹包含了build文件夹中webpack所依赖的配置文件中需要的配置项信息，其中每个文件的作用如下：

|  |  |
| --- | --- |
| 文件名 | 用途 |
| dev.env.js | build/webpack.dev.conf.js中使用的配置文件，用于控制编译为测试环境的代码 |
| index.js | 除环境变量配置以外的配置 |
| prod.env.js | build/webpack.prod.conf.js中使用的配置文件，用于控制编译为生产环境的代码 |

### node\_modules文件夹

项目依赖的包存放的位置。

### src文件夹

该文件夹包含该项目的源代码。其中每个文件夹/文件的作用如下：

|  |  |
| --- | --- |
| 文件夹名/文件名 | 用途 |
| assets/ | 存放项目依赖的需要被webpack处理静态资源文件，如图片，样式文件（css, less等）以及外部的js文件 |
| components/ | 存放项目的组件vue文件 |
| router/ | 存放项目的路由信息，路由信息会存放于router/index.js 文件中 |
| App.vue | vue的入口文件 |
| main.js | 项目的入口文件 |

### static文件夹

该文件用于存放纯静态资源，这类资源不需要被webpack编译处理。

### 其他文件

项目中的其他文件用途如下：

|  |  |
| --- | --- |
| 文件名 | 用途 |
| .babelrc | babel编译参数 |
| .editorconfig | 编辑器的一些设置，如缩进，空格等设置 |
| .eslintignore | eslint忽略的文件 |
| .eslintrc.js | eslint的配置文件 |
| .gitignore | git的控制文件，指定哪些文件需要被git忽略 |
| .postcssrc.js | postcss的配置文件 |
| index.html | 入口html文件 |
| package.json | npm依赖的配置文件 |
| package-lock.json | 记录第一次使用npm install时实际安装的各个包的来源以及版本 |
| README.md | 介绍如何安装，启动该项目的文件 |

## src文件结构说明

由于我们只需要关心src文件夹中的内容，它也是我们管理源码的地方，所以其最好按照如下结构进行源码管理：

src

--asserts

--components（存放被所有页面共享的组件）

--lib

--router

----index.js

----views（存放代表页面的组件）

------home（每个页面级的组件有一个文件夹且名字须和页面组件的名字一致）

--------home.vue（页面级组件）

--------components（仅被该页面使用的组件）

----------xxx.vue

------zhihu

--------zhihu.vue

--------components

----------yyy.vue

App.vue

main.js

style.css

# 调试

开发前端的IDE（并不是真正的IDE不能和visual studio 或 intellij进行比较）或者叫编辑器比较多，较为流行是WebStorm和Visual Stuido Code.

## 使用Visual Studio Code调试

由于目前编写前端项目代码通常使用Visual Studio Code作为IDE，所以下文会以Visual Studio Code为例讲解如何调试Vue项目。

下文中的讲解基于Vue的官方调试文档以及使用VUE CLI创建的项目

https://vuejs.org/v2/cookbook/debugging-in-vscode.html

1. 安装Visual Studio Code，下载地址在[这里](https://code.visualstudio.com/)。该工具用来编写和调试前端代码。
2. 安装Chrome浏览器，用来显示前端页面。
3. 安装Debugger for Chrome 插件，下载地址在[这里](https://marketplace.visualstudio.com/items?itemName=msjsdiag.debugger-for-chrome)。当然也可以直接在VS Code的插件管理中搜索这个插件进行安装。该工具用来支持在VS Code中调试在Chrome中运行的JavaScript代码。
4. 调整项目中config/index.js中devtool属性的值为’source-map’使项目被编译后的代码能够映射到源代码
5. 点击VS Code左侧面板的debug按钮，点击下拉箭头选择“Add Configuration…”，然后在打开的launch.json文件中填入以下内容：

|  |
| --- |
| {  // Use IntelliSense to learn about possible attributes.  // Hover to view descriptions of existing attributes.  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387  "version": "0.2.0",  "configurations": [  {  "type": "chrome",  "request": "launch",  "name": "vuejs: chrome",  "url": "http://localhost:8080",  "webRoot": "${workspaceFolder}/src",  "breakOnLoad": true,  "sourceMapPathOverrides": {  "webpack:///src/\*": "${webRoot}/\*"  }  }  ]  } |

1. 打开cmd路由到项目的根目录，输入“npm start”. 注意，此步骤必须需要，由于调整了devtool属性值，需要使用该命令对项目进行重新编译。
2. 点击VS Code左侧面板的debug按钮，选择刚才添加的“vuejs: chrome”, 然后点击绿色箭头开启调试，此时便可以进入断点调试了。

# Q&A

## 文件中的import指令的寻址问题

在import指令中，会看到下面几种形式，其寻址的优先级各不相同：

|  |  |
| --- | --- |
| 形式 | 说明 |
| import Vue from ‘vue’ | 当from后面仅仅是包名时，那么其会从项目的‘node\_modules’文件夹下查找相应文件，如左边对应的文件如下：  node\_modules/vue/dist/vue.js |
| import App from './App' | 当from后面的字符串中包含‘/’时，那么其会按照对应的相对目录查找目标文件，’/’说明如下：  ‘./’: 当前目录下  ‘../’: 上一级目录  以左边为例，寻址的优先级如下：  ./App.vue  ./App.mjs  ./App.json  ./App.node  ./App/package.json中的main字段指定的文件  .App/index.vue  .App/index.mjs  .App/index.js  .App/index.json  .App/index.node |