**旅行日程- Vue开发说明**

变更历史:

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **日期** | **人员** | **变更描述** |
| 1.0 | 2018/08/22 | 卫望 | 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。

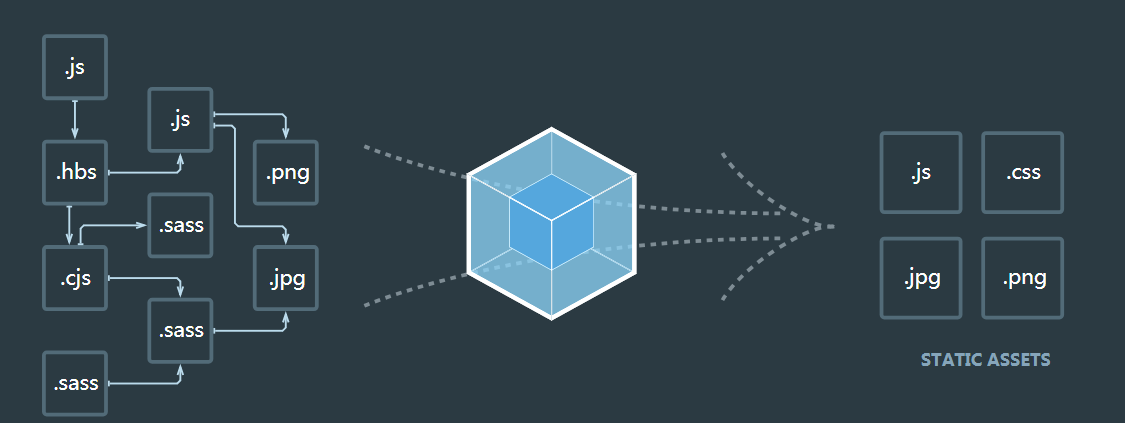
下载地址以及官方中文文档：

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/>

### 生命周期

引用VUE官方站点的图片：



各生命周期事件说明如下：

|  |  |
| --- | --- |
| 生命周期事件 | 用途 |
| beforeCreated | vue实例未创建完成，仅是一个空壳 |
| created | vue实例创建完成，即data、method等等已经可以访问，但尚未挂载指定的dom元素。需要初始化处理一些数据时通常会用到该事件 |
| beforeMount | 将template属性或者待挂载点的OuterHtml编译成虚拟DOM，但此时并未将vue实例的数据反映到$el属性上 |
| mounted | 将实例的值反映到虚拟DOM上，然后将虚拟DOM生成真实的DOM，最后append到页面上的挂载点下 |
| beforeUpdate | vue实例有数据变化时，会重新生成虚拟DOM，然后和上一次的虚拟DOM进行比较，找到发生变化的元素 |
| updated | 将变化的元素反馈到挂载点下 |
| beforeDestroy | vue实例销毁之前调用，主要用于解绑一些事件 |
| destroy | 销毁vue实例 |

### vue指令介绍

vue指令是Vue.js模板最为常用的功能，指令名均以”v-“作为前缀，后文会对每个指令进行介绍。

#### v-model

该指令用于在input元素上做数据的双向绑定，即input控件值的变化会导致底层关联数据的变化，底层数据的变化会导致关联input控件值的变化。

如：

|  |
| --- |
| <input type=”text” v-model=”name”> |

将input控件输入的值与name属性进行双向绑定

#### v-if/v-else-if/v-else

这类指令用于控制在符合条件时才在DOM中加入该html元素，否则则不创建或从DOM中销毁该html元素。

如：

|  |
| --- |
| <p v-if=”status===1”>status等于1</p>  <p v-else-if=“status===2”>status等于2</p>  <p v-else>status等于其他值</p> |

若show属性的值为true，则p元素插入DOM；反之，p元素不插入DOM

#### v-show

该指令用于控制html元素是否显示，该指令内部操作html元素的display属性。

如：

|  |
| --- |
| <p v-show=”status===1”>status等于1</p> |

#### v-for

该指令用于将一个数组类型的属性/计算属性进行遍历渲染。

如：

|  |
| --- |
| <ul>  <li v-for=”book in books”>{{ book.name }}</li>  <ul>  <script>  var app = new Vue({  data: {  books: [  { name: ‘name1’},  { name: ‘name2’}  ]  }  })  </script> |

#### v-bind

该指令用于将数据和html元素的属性（如id, class, href等等）进行绑定，使相应元素的属性能够实现动态变化。

如：

|  |
| --- |
| <img v-bind:src=”imgUrl” /> |

简写：

v-bind可以直接用”:”代替，如上面的示例可以采用下面这种写法：

|  |
| --- |
| <img :src=”imgUrl” /> |

#### v-on

该指令用于绑定事件监听，将vue实例中的methods里的方法与html元素的事件关联，当然也可以直接在事件监听后面绑定表达式。

如：

|  |
| --- |
| <button v-on:click=”handleClose”>按钮</button>  或  <button v-on:click=”show=false”>按钮</button> |

简写：

v-on可以直接用”@”代替，如上面示例可以采用下面的写法：

|  |
| --- |
| <button @click=”handleClose”>按钮</button> |

#### v-cloak

该指令不需要任何表达式，没有任何作用，只是会在vue实例结束编译时从绑定的html元素上自动移除。

利用这个特性，所以该指令经常会和css的display:none结合使用。因为有时vue创建实例、编译模板较慢，用户会看到一个中间状态，所以结合使用可以隐藏掉中间状态。

如通过下面代码可以隐藏掉因为vue实例还未编译完成时导致用户看到{{message}}这个文案的中间状态。

|  |
| --- |
| <div id=”app” v-cloak>  {{ message }}  </div>  <script>  var app = new Vue({  el: ‘#app’,  data: {  message: ‘一段文本’  }  })  </script>  <style>  [v-cloak] {  display: none;  }  </style> |

### Vue实例配置说明

总体来讲，vue实例的配置选项参数中支持如下配置：

|  |  |
| --- | --- |
| 配置名 | 用途 |
| el | vue实例需要挂载的html元素 |
| data | vue实例的属性 |
| methods | 方法，vue实例中可以重用的方法 |
| computed | 计算属性，类似于方法，与方法的区别在于其会缓存数据。仅当计算属性中依赖的任意一个属性发生变化时，该计算属性值才会重新计算 |

## 什么是Nuxt.js

Nuxt框架是一个基于VUE的通用框架，用于快速方便的完成基于SSR（Server Side Render）的前端web站点的搭建。它只是一个服务端渲染框架。

### 页面组件配置项介绍

由于Nuxt对vue进行了较好的封装，所以为了能够快速开发，Nuxt对页面级别的vue组件提供了一些额外的方法，如下：

|  |  |
| --- | --- |
| 方法 | 说明 |
| asyncData(context) | 该方法会在页面组件每次加载之前被调用，即可以再服务端或路由更新之前被调用，该方法返回的数据将和dat1·a方法返回的数据合并之后一起返回给组件。  需要注意的是：该方法仅在页面通过url打开时才会在服务端执行，通过nuxt-link跳转的方式跳转到页面时，该方法仅会在客户端（浏览器执行）  详细介绍可见[这里](https://zh.nuxtjs.org/guide/async-data/)。 |
| fetch(context) | 该方法会在页面组件每次加载之前被调用，即可以在服务端或路由更新之前被调用，与asyncData类似，但与asyncData不同的是，该方法不会设置组件的数据。通常使用该方法来填充应用的状态树。  需要注意的是：该方法同asyncData一样，仅在页面通过url打开时才会在服务端执行，通过nuxt-link跳转的方式跳转到页面时，该方法仅会在客户端（浏览器执行）  详细介绍可见[这里](https://zh.nuxtjs.org/api/pages-fetch/)。 |
| head() | 设置页面的头部标签，如title，meta等内容。  详细介绍可见[这里](https://zh.nuxtjs.org/api/pages-head/)。 |
| layout layout(context) | 指定页面的布局文件。  详细介绍可见[这里](https://zh.nuxtjs.org/api/pages-layout/)。 |
| loading | 控制页面跳转时的进度条显示  详细介绍可见[这里](https://zh.nuxtjs.org/api/configuration-loading)。 |
| transition | 定义页面切换的动效  详细介绍可见[这里](https://zh.nuxtjs.org/api/pages-transition/)。 |
| scrollToTop | 渲染页面前是否需要将当前页面滚动至顶部 |
| validate | 校验动态路由参数  详细介绍可见[这里](https://zh.nuxtjs.org/guide/routing/#%E8%B7%AF%E7%94%B1%E5%8F%82%E6%95%B0%E6%A0%A1%E9%AA%8C)。 |
| middleware | 指定页面的中间件，如权限验证等等。中间件会在改变时被调用。  详细介绍可见[这里](https://zh.nuxtjs.org/guide/routing/#%E4%B8%AD%E9%97%B4%E4%BB%B6)。 |

### 服务端生命周期

\* Incoming Request

\* nuxtServerInit

Store action

\* middleware

1. nuxt.config.js

2. matching layout

3. matching page & children

\* validate()

Pages & children

\* asyncData() & fetch()

Pages & children

\* Render

\* Navigate

<nuxt-link>

### 哪些Vue的原生方法可以在服务端执行呢？

除了Nuxt在页面组件中新增的方法（见[这里](#_页面组件配置项介绍)）会在服务端执行外，如下两个组件中的方法不仅会在客户端调用，也会在服务端调用。

beforeCreate

created

### 同时在客户端与服务端执行的方法

Nuxt扩展的如下方法会在不同场景下分别在客户端/服务端执行：

|  |  |  |
| --- | --- | --- |
| 方法名 | 客户端执行 | 服务端执行 |
| asyncData | 通过nuxt-link进行跳转时，即page不会reload时，都仅在客户端执行 | 直接通过浏览器地址输入url跳转到该页面，即page发生reload时，会在服务端执行 |
| fetch | 同上 | 同上 |

## 什么是pm2

pm2是一个**node**进程管理工具，使用pm2来管理进程可以获得如下好处：

1. 可以轻松的查看进程的cpu, 内存使用情况
2. 可以轻松的进行集群管理以及负载均衡
3. 可以轻松的查看应用产生的日志
4. 可以轻松的开启/关闭应用
5. 可以自动重启应用
6. 可以监听应用的变化从而重新加载
7. 可以在不中断应用的情况下重启应用（reload）

官方文档：

<https://pm2.io/doc/en/runtime/overview/>

### 常用命令介绍

|  |  |
| --- | --- |
| 命令 | 功能 |
| pm2 start xxx | 启动应用，参数说明见后文详细介绍。 |
| pm2 stop 应用名/应用id | 关闭应用，关闭指定进程，但并不为将应用信息从pm2的管理列表中移除 |
| pm2 list | 列出所有被pm2管理 |
| pm2 delete args | args为all时:  清空pm2的管理列表  args为应用名或应用id时：  将指定应用从pm2的管理列表中移除 |
| pm2 logs应用名/应用id | 查看指定应用产生的日志 |
| pm2 show 应用名/应用id | 展示应用的相关信息，如：启动参数，运行时间，node版本等等 |

### pm2 start 参数介绍

通过pm2启动应用有如下两种方式：

1. 直接在bash里面输入”pm2 start 各种参数”来启动node应用
2. 提供一个pm2.config.js的配置文件，通过”pm2 start pm2.config.js”的方式来启动

在启动参数较多的情况下，通过第二种方式提供一个pm2的配置文件会更加方便一点，后文以第二种方式启动来讲解。

pm2.config.js的内容示例如下：

|  |
| --- |
| module.exports = {  apps: [  {  name: 'infoweb',  script: 'nuxt',  args: 'start -H "10.2.56.38"',  max\_restarts: 0,  cwd: './build',  env: {  NODE\_ENV: 'development'  },  env\_production: {  NODE\_ENV: 'production'  }  }  ]  } |

配置文件中的内容介绍如下：

|  |  |
| --- | --- |
| 配置参数 | 说明 |
| name | 应用名称，后面通过该名称管理应用 |
| script | 启动应用的命令，如npm, nuxt等等 |
| args | 配合启动命令的参数，如配合npm，我们通常会在该项配置”run dev”; 配合nuxt, 我们通常会在该项配置”start” |
| cwd | 应用的工作目录，通常是指待启动应用的package.json所在目录 |
| exec\_mode | cluster\_mode或fork模式，默认fork模式  fork模式：单实例单进程，占用端口不能被共享，可以用于启动其他非node类型的应用  cluster\_mode模式：多实例多进程，多进程间可以共享同一个端口，仅能用于启动node类型的应用 |
| watch | 是否启用监控模式 |

完整参数列表可以查看这里：

http://pm2.keymetrics.io/docs/usage/application-declaration/#attributes-available

# 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

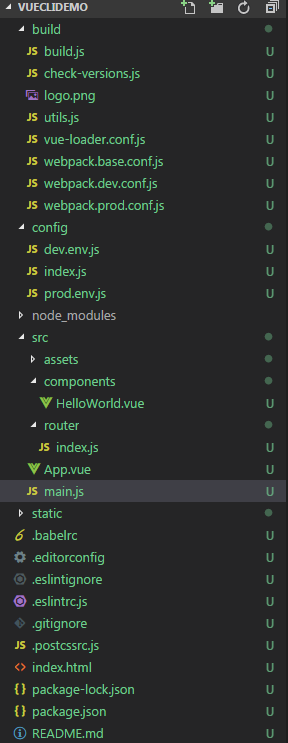
填写完上面选项后，等待下载各种组件，下载完成后，vue项目的模板工程就搭建完成了。

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。

# 创建Nuxt项目

下文会介绍如何创建一个nuxt项目。

## 创建nuxt项目

1. 创建存放源码的文件夹，在文件夹中创建package.json文件，内容如下：

|  |
| --- |
| {  "name": "my-app",  "scripts": {  "dev": "nuxt"  }  } |

1. 创建完package.json后，打开cmd，cd到该文件夹，然后通过如下命令创建对nuxt的依赖：

|  |
| --- |
| npm install –save nuxt |

1. 创建pages/index.vue, 输入如下内容：

|  |
| --- |
| <template>  <h1>Hello world!</h1>  </template> |

1. 打开cmd，运行如下命令，即可启动nuxt应用

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

## 项目结构分析

由于nuxt采用了约定大于配置的设定，所以每个目录都有其特殊的用途，介绍如下：

|  |  |
| --- | --- |
| 目录 | 用途 |
| .nuxt | 包含nuxt的框架代码，一般我们不需要调整该文件夹的代码 |
| asset | 包含了未编译的静态资源文件如LESS,SASS等等 |
| layouts | 包含页面依赖的布局组件 |
| middleware | 包含中间件 |
| pages | 包含了页面级的组件 |
| plugins | 包含了需要在vue实例运行之前的js插件 |
| static | 静态文件目录，该目录不会被webpack编译 |
| store | 存放组织应用的vuex状态树 |
| nuxt.config.js | 用于覆盖nuxtjs的默认配置 |
| package.json | 描述应用的依赖信息 |

## 调试

### 使用Visual Studio Code调试

#### 调试服务端代码

注意：后文的描述均针对windows平台，linux平台会有一点区别。

vs code中的launch.json文件设置为如下：

|  |
| --- |
| {  // 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": "node",  "request": "launch",  "name": "launch nuxt",  "protocol": "inspector",  "program": "${workspaceRoot}/node\_modules/nuxt/bin/nuxt",  "stopOnEntry": false,  "args": ["dev"],  "cwd": "${workspaceRoot}",  "sourceMaps": true,  "env": {  "NODE\_ENV": "development",  "DEBUG": "nuxt:\*,app"  }  }  ]  } |

在项目中的查找nuxt.config.json文件，若没有则添加一个，在其中添加/输入如下内容：

|  |
| --- |
| module.exports = {  build: {  extend (config, {isClient}){  if (isClient)  config.devtool = 'eval-source-map'  else  config.devtool = "inline-source-map"  }  }  } |

最后在vs code中启动调试即可进入服务端的断点。

最后需要说的是，即便按照如上的步骤可以调试服务端的代码了，但也并不友好，比如有些断点打不了，有些断点触发后并不是自己之前设置的断点位置。所以整体来讲，目前调试服务端还是通过console.log的方式会比较好。

## 部署

下文主要以在linux服务器上部署服务端渲染项目为例，进行讲解。

### 在linux服务器上安装nodejs

由于nuxtjs应用本质是一个nodejs应用，所以若要在linux上运行该应用，首先需要在linux上安装nodejs环境。

1. 通过如下命令查看linux系统是32位的还是64位的

|  |
| --- |
| uname –a |

若输出的结果中有”x86\_64”，则说明是64位的；若有“i686”则表示32位系统。

1. 进入如下地址查找匹配的安装包，最终应该会得到一个\*\*\*\*.tar.xz的安装压缩包

<https://nodejs.org/en/download/>

1. 将下载的安装上传到linux服务器上，通过如下命令进行解压

|  |
| --- |
| 解压xz文件得到tar文件  xz –d \*\*\*.tar.xz  解压tar文件得到安装文件目录  tar –xvf \*\*\*.tar |

1. 使用如下命令，建立node和npm的软连接，使这node和npm命令变为全局命令：

|  |
| --- |
| sudo ln –s 解压tar得到的解压目录/bin/node usr/local/bin/  sudo ln –s 解压tar得到的解压目录/bin/npm usr/local/bin/ |

注意：使用上述命令前需要回到根目录(/)

### 安装PM2

由于我们要部署的nuxt应用就是node应用，所以通过PM2来进行node进程的管理工具。

如果linux服务器可以访问外网，那可以用如下命令即可安装PM2:

|  |
| --- |
| npm install pm2@latest -g |

如果linux服务无法访问外网，只能通过如下步骤进行手工安装PM2了：

1. 找一台安装好npm的可以联网的机器，然后使用如下命令安装PM2：

|  |
| --- |
| npm install pm2@lateset -g |

1. 使用如下命令安装npm-bundle，该工具可以将待打包的bundle及该bundle的依赖项一起打入同一个包中

|  |
| --- |
| npm install npm-bundle -g |

1. 使用如下命令将PM2打包，打包成功后会生成一个pm2-x.x.x.tgz的压缩包

|  |
| --- |
| npm-bundle pm2 |

注意，这个打包过程较慢，需要耐心等待。

1. 上传pm2-x.x.x.tgz的压缩包到linux服务器上
2. 在linux服务器上找到上传的pm2-x.x.x.tgz，使用如下命令安装该工具：

|  |
| --- |
| npm install pm2-x.x.x.tgz |

注意，这个安装过程较慢，需要耐心等待。

1. 使用如下命令为pm2建立软连接，使其成为全局命令：

|  |
| --- |
| sudo ln –s node\_module的安装目录pm2/bin/pm2 usr/local/bin/  例如：  sudo ln –s /home/lsqiu/SmartTrip/Software/node-v10.13.0-linux-x64/lib/node\_modules/pm2/bin usr/local/bin/ |

注意：使用上述命令前需要回到根目录(/)

### 安装Nuxtjs

如果linux服务器可以访问外网，那可以用如下命令即可安装Nuxtjs:

|  |
| --- |
| npm install nuxt@2.2.0 -g |

如果linux服务无法访问外网，只能通过类似安装PM2的步骤进行手工安装Nuxtjs了：

1. 找一台安装好npm的可以联网的机器，然后使用如下命令安装Nuxtjs：

|  |
| --- |
| npm install nuxt@2.2.0 -g |

1. 使用如下命令将Nuxtjs打包，打包成功后会生成一个nuxt-2.2.0.tgz的压缩包

|  |
| --- |
| npm-bundle nuxt |

注意，这个打包过程较慢，需要耐心等待。

1. 上传nuxt-2.2.0.tgz的压缩包到linux服务器上
2. 在linux服务器上找到上传的nuxt-2.2.0.tgz，使用如下命令安装该工具：

|  |
| --- |
| npm install nuxt-2.2.0.tgz |

注意，这个安装过程较慢，需要耐心等待。

1. 使用如下命令为nuxt建立软连接，使其成为全局命令：

|  |
| --- |
| sudo ln –s 解压tar得到的解压目录/bin/node usr/local/bin/ |

注意：使用上述命令前需要回到根目录(/)

### 在服务器上运行nuxtjs应用

1. 使用nuxt build对项目进行编译，编译完成后，会在.nuxt/ 目录下生成一个新的dist的文件夹，该文件夹即包含了编译后的源代码。
2. 将如下文件/文件夹拷贝至服务器：

|  |
| --- |
| .nuxt整个文件夹  node\_modules整个文件夹  static整个文件夹  nuxt.config.js文件  package.json文件 |

1. 定位到(cd命令)服务器上存放上述文件的文件夹，使用如下命令启动应用即可：

|  |
| --- |
| nuxt start -H “域名/服务器IP” |

注意，一定需要添加-H参数，否则会以localhost作为域名启动从而导致外部无法访问。

1. 上面第3步是通过直接使用nuxt start命令来启动的方式来启动进程的，但由于NuxtJs应用本质上是node应用且上述启动方式不便于我们管理应用。所以可以通过pm2来进行应用的启动，从而方便我们对进程的监控以及启动。定位到(cd命令)服务器上存放上述文件的文件夹，使用如下命令即可通过pm2来进行应用的启动

|  |
| --- |
| pm2 start nuxt --name 应用名称 -- start -H “10.2.56.38”  如：  pm2 start nuxt --name infoweb -- start -H “10.2.56.38” |

1. 通过pm2关闭应用可以使用如下命令：

|  |
| --- |
| pm2 stop 应用名称  如：  pm2 stop infoweb |

# 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 |