# NuxtJS

## 简介

Nuxt.js简单的说是Vue.js的通用框架，最常用的就是用来作SSR（服务器端渲染）。再直白点说，就是Vue.js原来是开发SPA（单页应用）的，但是随着技术的普及，很多人想用Vue开发多页应用，并在服务端完成渲染。这时候就出现了Nuxt.js这个框架，她简化了SSR的开发难度。还可以直接用命令把我们制作的vue项目生成为静态html。

### 服务器端渲染到底有什么好处

最主要的原因时SPA（单页应用）不利于搜索引擎的SEO操作。比如你作一个新闻网站，流量的一个主要来源是通过百度、谷歌、bing这些搜索引擎，但是它们对SPA的抓取并不好，特别是百度根本没法抓取到SPA的内容页面，所以我们必须把我们的应用在服务端渲染成适合搜索引擎抓取的页面，再下载到客户端。那Nuxt.js适合作新闻、博客、电影、咨询这样的需要搜索引擎提供流量的项目。如果你要作移动端的项目，就没必要使用这个框架了。（其实Nuxt.js个人觉的是一个解决方案）

### 什么是SSR

SSR，即服务器渲染，就是在服务器端将对Vue页面进行渲染生成html文件，将html页面传递给浏览器。

SSR两个优点：

SEO 不同于SPA的HTML只有一个无实际内容的HTML和一个app.js，SSR生成的HTML是有内容的，这让搜索引擎能够索引到页面内容。

更快内容到达时间 传统的SPA应用是将bundle.js从服务器获取，然后在客户端解析并挂载到dom。而SSR直接将HTML字符串传递给浏览器。大大加快了首屏加载时间。

Nuxt.js的官方网站是这样介绍的：

Nuxt.js 是一个基于 Vue.js 的通用应用框架。

通过对客户端/服务端基础架构的抽象组织，Nuxt.js 主要关注的是应用的 UI渲染。

Nuxt.js是特点（优点）：

基于 Vue.js

自动代码分层

服务端渲染

强大的路由功能，支持异步数据

静态文件服务

ES6/ES7 语法支持

打包和压缩 JS 和 CSS

HTML头部标签管理

本地开发支持热加载

集成ESLint

支持各种样式预处理器： SASS、LESS、 Stylus等等

## NuxtJS安装

1.用npm来安装vue-cli这个框架，如果你已经安装过了，可以省略这步。

npm install vue-cli -g

这个根据你的网络环境不同，安装的速度不仅相同，如果你的网络环境实在不好，可以考虑使用cnpm来进行安装。（在实际开发中我会尽量避免使用cnpm来进行安装，因为会出现一些未知的错误。）

安装完成后可以使用vue -V 来测试是否安装成功。（注意：这里要使用大写的V，小写无效）。

2.使用vue安装 nuxt

安装好vue-cli后，就可以使用init命令来初始化Nuxt.js项目。

vue init nuxt/starter

这时候他会在github上下载模版，然后会询问你项目的名称叫什么，作者什么的，这些完全可以根据自己的爱好填写。

3.使用npm install安装依赖包

npm install

这个过程是要等一会的，如果你这个过程安装失败，也不要慌张，你可以直接诶删除项目中的node\_modules文件夹后，重新npm install进行安装。

4.使用npm run dev 启动服务

npm run dev

1. 在浏览器输入 localhost:3000,可以看到结果。

## NuxtJS目录结构

Nuxt自动生产了项目目录，我们先来一个一个介绍一下。

|-- .nuxt // Nuxt自动生成，临时的用于编辑的文件，build

|-- assets // 用于组织未编译的静态资源入LESS、SASS 或 JavaScript

|-- components // 用于自己编写的Vue组件，比如滚动组件，日历组件，分页组件

|-- layouts // 布局目录，用于组织应用的布局组件，不可更改。

|-- middleware // 用于存放中间件

|-- pages // 用于存放写的页面，我们主要的工作区域

|-- plugins // 用于存放JavaScript插件的地方

|-- static // 用于存放静态资源文件，比如图片

|-- store // 用于组织应用的Vuex 状态管理。

|-- .editorconfig // 开发工具格式配置

|-- .eslintrc.js // ESLint的配置文件，用于检查代码格式

|-- .gitignore // 配置git不上传的文件

|-- nuxt.config.json // 用于组织Nuxt.js应用的个性化配置，已覆盖默认配置

|-- package-lock.json //用于帮助package的统一性设置的，yarn也有相同的操作

|-- package-lock.json // 用于帮助package的统一性设置的，yarn也有相同的操作

|-- package.json // npm包管理配置文件

## Nuxt常用配置项

配置IP和端口：

开发中经常会遇到端口被占用或者指定IP的情况。我们需要在根目录下的package.json里对config项进行配置。比如现在我们想把IP配置成127.0.0.1，端口设置1818。

/package.json

"config":{

"nuxt":{

"host":"127.0.0.1",

"port":"1818"

}

},

配置好后，我们在终端中输入npm run dev，然后你会看到服务地址改为了127.0.0.1:1818.

配置全局CSS

在开发多页项目时，都会定义一个全局的CSS来初始化我们的页面渲染，比如把padding和margin设置成0，网上也有非常出名的开源css文件normailze.css。要定义这些配置，需要在nuxt.config.js里进行操作。

比如现在我们要把页面字体设置为红色，就可以在assets/css/normailze.css文件，然后把字体设置为红色。

/assets/css/normailze.css

html{

color:red;

}

/nuxt.config.js

css:['~assets/css/normailze.css'],

设置好后，在终端输入npm run dev 。然后你会发现字体已经变成了红色。

配置webpack的loader

在nuxt.config.js里是可以对webpack的基本配置进行覆盖的，比如现在我们要配置一个url-loader来进行小图片的64位打包。就可以在nuxt.config.js的build选项里进行配置。

build: {

loaders:[

{

test:/\.(png|jpe?g|gif|svg)$/,

loader:"url-loader",

query:{

limit:10000,

name:'img/[name].[hash].[ext]'

}

}

],

/\*

\*\* Run ESLint on save

\*/

extend (config, { isDev, isClient }) {

if (isDev && isClient) {

config.module.rules.push({

enforce: 'pre',

test: /\.(js|vue)$/,

loader: 'eslint-loader',

exclude: /(node\_modules)/

})

}

}

}