

Nuxt 3

基于 Vue3 的一个服务端渲染 (SSR) 框架

模板渲染

1. 什么是模板渲染?

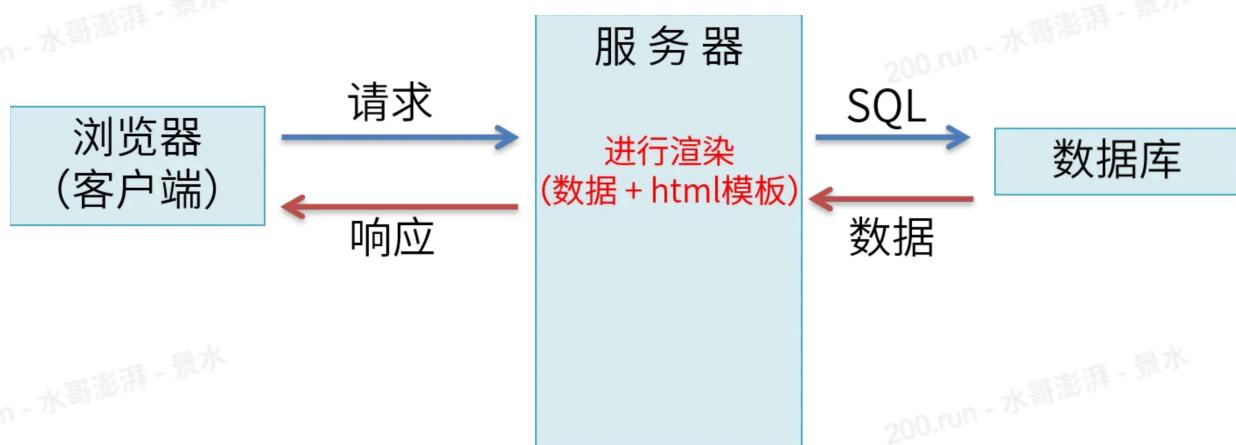
动态网站 = 数据 + 模板 => 拼到一起, 形成 html

2. 在哪里渲染?

- 传统服务端渲染
- 客户端渲染
- 同构渲染

传统的服务端渲染 - MVC

1. 传统的服务端渲染流程?



2. 特点

💡 快速上手

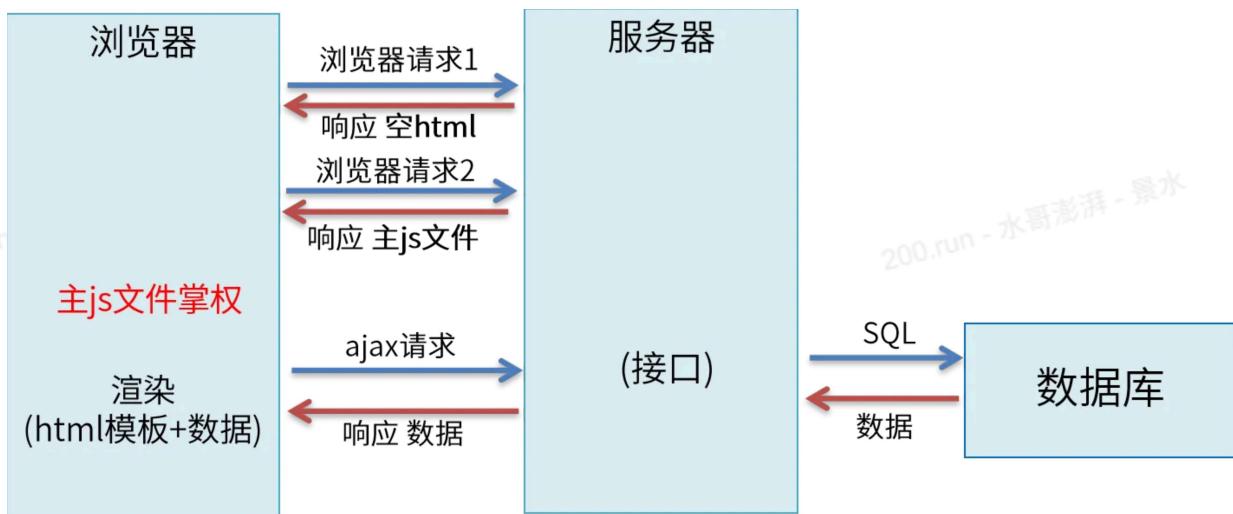
所有客户端请求的页面都有服务器渲染返回，服务器压力大

5457字

页面局部变化时，也需要服务器把整体页面重新传给客户端，浪费带宽。
(ajax可以适当的解决)

客户端渲染 - MVVM

1. 客户端渲染的流程？



2. 主要框架

Vue、React、Angular

3. 与传统的服务端渲染比较

优点：用户体验好，后续请求数据量少

缺点：首屏加载慢、不利于 SEO

4. 为什么首屏渲染慢？

5. 为什么不利于 SEO？

快速上手

同构渲染 - SSR

5457字

1. 同构渲染的流程?



2. 有哪些常见框架?

Nuxt.js: 基于 Vue 的 SSR

Next.js: 基于 React 的 SSR

3. 首屏和seo 是否真的需要?

后台管理系统

准备 - Nuxt.js

官网: <https://nuxtjs.com/> <<https://nuxtjs.com>>

环境

node -v ≥ 14.16 或 ≥ 16.11

npm -v

💡 快速上手

VS Code 安装 volar TypeScript Vue Plugin(Volar)

创建新项目方式1 - 官网

npx nuxi init project-name (若过程中下载github模板总是报错, 采用其它方式)

创建新项目方式2 - 克隆方式

1. 直接下载或克隆代码

```
git clone -b v3 https://github.com/nuxt/starter.git  
<https://github.com/nuxt/starter.git>  
git clone -b v3 https://gitee.com/2xx/starter.git  
<https://gitee.com/2xx/starter.git> (github不行, 可使用这个地址)
```

2. 进入目录, 安装依赖

```
cd starter  
npm i
```

3. 查看 package.json 中的项目启动命令

```
code .  
npm run dev
```

创建新项目方式3 - 手动

1. 创建目录, 进入目录, 初始化, 安装依赖, 用编辑打开

```
mkdir my-nuxt  
cd my-nuxt  
npm init -y  
npm i nuxt  
code .
```

2. 配置文件 - 创建 nuxt.config.ts

```
▼ nuxt.config.ts  
1 import { defineNuxtConfig } from "nuxt/config"  
2  
3 export default defineNuxtConfig({})
```

TypeScript | ⏪ 运行代码 ⏪ 复制代码

3. 配置文件 - 创建 tsconfig.json

```
▼ tsconfig.json  
1 {  
2   "extends": "./.nuxt/tsconfig.json"  
3 }
```

JSON | ⏪ 复制代码

4. 添加启动开发命令 - 编辑 package.json

```
▼  
1 {  
2   ...  
3   "scripts": {  
4     "dev": "nuxi dev"  
5   },  
6   ...  
7 }
```

JSON | ⏪ 复制代码

5. 启动项目

npm run dev (会生成 .nuxt 目录)

根据项目需要安装其它依赖

...

路由

💡 快速上手

1. 项目根组件

编辑 app.vue - 查看效果需要重启项目

```
▼ app.vue Vue | 复制代码
1  
2    <div>
3      <h1>项目根组件</h1>
4      <nuxt-page /> <!-- 挖个洞,留个显示区域, 用于显示路由组件内容 -->
5    </div>
6  </template>
```

2. 路由组件

创建 pages / about.vue - 对应路由地址为 **/about** - 需要重启

```
▼ pages / about.vue Vue | 复制代码
1  
2    <h1>关于2023年前端发展方向的研讨</h1>
3  </template>
```

3. 带目录的路由组件

创建 pages / users / create-or-edit.vue - 对应地址为 **/users/create-or-edit**

```
▼ pages/users/create-or-edit.vue Vue | 复制代码
1  
2    <h1>创建或编辑用户信息</h1>
3  </template>
```

4. 默认路由组件

创建 pages / index.vue - 对应地址为 **/**

```
1  
2    <h1>路由组件1</h1>
3  </template>
```

创建 pages / users / index.vue 对应地址为 **/users**

快速上手

```
1  
2      <h1>用户列表信息</h1>
3  </template>
```

5. 父子路由

1. 创建 pages / roles.vue - 对应路由地址为 **/roles** - 需要重启

```
1  
2      <div>
3          <h1>角色信息汇总</h1>
4      </div>
5  </template>
```

2. 给子路由组件留显示区域 - 编辑 roles.vue

```
1  
2      <div>
3          <h1>角色信息汇总</h1>
4          <nuxt-page />    <!-- 挖洞 -->
5      </div>
6  </template>
```

3. 创建 pages / roles / 目录, 以及 pages / roles / admin.vue

```
1  
2      <div>
3          <h1>角色-管理员</h1>
4      </div>
5  </template>
```

4. 创建 pages / roles / normal.vue

💡 快速上手

Vue | 复制代码

```

1 <template>
2   <div>
3     <h1>角色-普通用户</h1>
4   </div>
5 </template>

```

(测试查看效果)

路由导航跳转

1. 编辑 app.vue - 设置跳转链接

```

▼ App.vue
Vue | 复制代码

1 <template>
2   <div>
3     <h1>我是根组件</h1>
4     <nuxt-link to="/users">用户-列表</nuxt-link>
5     <nuxt-link to="/users/create-or-edit">用户-添加或更新</nuxt-link>
6     <nuxt-link to="/about">关于</nuxt-link>
7     <nuxt-link to="/roles/admin">角色-管理员</nuxt-link>
8     <nuxt-link to="/roles/normal">角色-普通用户</nuxt-link>
9
10    <!--
11      nuxt-link 会被编译成 a 标签，但是不推荐直接使用 a 标签
12      直接用 a 标签，点击会刷新跳转，相当于一次全新的向服务端发起的请求
13      使用 nuxt-link 是浏览器端本地切换页面，即SPA
14    -->
15    <nuxt-page />
16  </div>
17 </template>
18
19 <script setup>
20 </script>
21
22 <style>
23   /* nuxt-link 会被编译成 a 标签 */
24   a { margin: 20px; }
25 </style>

```

会有一些小问题：**警告** - 推荐组件有一个根标签, 加一个就行了

 快速上手

App.vue

Vue | 复制代码

```

1  
2    <div>
3      <h1>我是根组件</h1>
4      ...
5
6      <nuxt-link to="/roles/normal">角色-普通用户</nuxt-link>
7
8      <a href="/roles/normal">角色-普通用户</a>
9
10     <nuxt-page />
11   </div>
12 </template>
13
14 <script setup>
15 </script>
16
17 <style></style>

```

(首屏，不一定是首页，任何一个url地址，通过刷新请求，都算是首屏)

动态路由参数

创建 pages / course / [id].vue - 对应路由为 **/course/:id**

<template>

Vue | 复制代码

```

1  <h1>课程编号:{{ route.params.id }}</h1>
2  </template>
3
4
5  <script setup>
6    const route = useRoute()
7    console.log(route) // 刷新时，这个打印应该出现在服务端，为了调试 nuxt 使其同时
8    // 也可以看到 route 中的 query path name hash matched
9  </script>
10
11 <style lang="scss"></style>
12

```

自定义路由

<https://nuxt.com/docs/api/configuration/nuxt-config#router>

5457字 <<https://nuxt.com/docs/api/configuration/nuxt-config#routerules-1>>

▼ app/router.options.ts (TS 版)

TypeScript | ⚡运行代码 ⏚ 复制代码

```

1 import type { RouterConfig } from '@nuxt/schema'
2
3 // https://router.vuejs.org/api/interfaces/routeroptions.html
4 export default <RouterConfig> {
5   routes: (_routes) => [
6     ..._routes,
7     {
8       name: 'home',
9       path: '/',
10      component: () => import '~/pages/home.vue'
11    }
12  ],
13}
14

```

▼ app/router.options.js (JS 版)

JavaScript | ⚡运行代码 ⏚ 复制代码

```

1 const customRoutes = [
2   {
3     name: 'home',
4     path: '/xxx',
5     component: () => import('../pages/about.vue')
6   }
7 ]
8 export default {
9   routes: (_routes) => _routes.concat(customRoutes)
10 }

```

路由匹配失败等错误，可以在项目根目录下创建 error.vue

路由中间件

匿名（或内联）中间件，直接在使用它们的页面中定义

命名中间件：放在 middleware 目录，哪个页面使用，就在哪个页面有效。名称建议为 some-middleware 形式

全局中间件：放在 middleware 目录，some-middleware.global.ts，每次路由变化都会执行

举例：auth 中间件，保护 /dashboard 页面

定义 - 中间件

 快速上手

▼ middleware/auth.ts

TypeScript | ⚡ 运行代码 ⌂ 复制代码

```

1 export default defineNuxtRouteMiddleware(({ to, from }) => {
2   if (isAuthenticated() === false) {
3     return navigateTo('/login')
4   }
5 })

```

使用 - 中间件

▼ pages/dashboard.vue

Vue | ⌂ 复制代码

```

1 <script setup>
2 definePageMeta({ middleware: 'auth' })
3 </script>

```

路由验证

validate 属性接受 route 参数，可以返回 布尔值，决定是否想要的路由，返回 false 或没匹配到路由时，报 404，也可以返回一个对象 状态码和状态信息

```

1 <script setup>
2 definePageMeta({
3   validate: async (route) => {
4     const nuxtApp = useNuxtApp()
5     return /^\d+$/ .test(route.params.id)
6   }
7 })
8 </script>

```

视图

使用子组件

1. 在项目根目录中，创建 components / users / header.vue - 创建组件文

```

1 <template>
2   <h1>用户头部-子组件</h1>
3 </template>

```

Vue | ⌂ 复制代码

💡 快速上手

2. 使用子组件 - 编辑 pages / users / index.vue

```

1  
2    <div>
3      <h1>用户列表信息</h1>
4      <UsersHeader />    <!-- 目录加文件名称 -->
5    </div>
6  </template>
7
8  <script setup>
9
10 </script>
11
12 <style lang="scss"></style>

```

Vue | 复制代码

布局

1. 创建 头部布局组件 - components / AppHeader.vue

2. 创建 侧边栏布局组件 - components / AppAside.vue

3. 创建 底部布局组件 - components / AppFooter.vue

4. 在项目根目录中，创建默认布局组件 layouts / default.vue - 因为有「洞」，所以需要有一个根标签

```

1  <template>
2
3    <app-header />
4    <app-aside />
5
6    <slot />    <!-- 挖洞，显示不同子路由组件的内容 -->
7
8    <app-footer />
9
10 </template>

```

Vue | 复制代码

5. 使用布局组件 - 编辑 App.vue

💡 快速上手

Vue | 复制代码

```
1  
2      <h1>我是根组件</h1>
3      <nuxt-link to="/users">用户列表</nuxt-link>
4      <nuxt-link to="/users/create-or-edit">添加或更新用户</nuxt-link>
5      <nuxt-link to="/about">关于</nuxt-link>
6
7      <!--
8          使用默认布局组件. 这里它只是有一个插槽, 表示空的, 可以添东西
9          而里面的 nuxt-page 是把路由对应的页面组件替换到这里
10     -->
11    <NuxtLayout>
12        <nuxt-page></nuxt-page>
13    </NuxtLayout>
14
15  </template>
16
17  <script lang="ts" setup>
18
19  </script>
20
21  <style>
22  a{
23      margin: 20px;
24  }
25  </style>
```

SEO

1. 统一设置 - 编辑 nuxt.config.ts

 快速上手

▼ nuxt.config.ts

[TypeScript](#) | [运行代码](#) [复制代码](#)

```
1  export default defineNuxtConfig({  
2    app: {  
3      head: {  
4        charset: 'utf-8', // 默认值，可不写  
5        viewport: 'width=device-width, initial-scale=1', // 默认值，可不写  
6        title: '水哥澎湃',  
7        meta: [  
8          // <meta name="description" content="一个适合初学者的网站">  
9          { name: 'description', content: '我的破网站' },  
10         { name: 'keywords', content: '编程,IT,前端,后端,培训,教程,视频' }  
11       ],  
12       link: [  
13         // <link rel="stylesheet" href="https://myawesome-lib.css">  
14         { rel: 'stylesheet', href: 'https://awesome-lib.css' }  
15       ],  
16       style: [  
17         // <style type="text/css"> .xxx { color: red }</style>  
18         { children: '.xxx { color: red }', type: 'text/css' }  
19       ],  
20       script: [  
21         // <script src="https://myawesome-lib.js"></script>  
22         { src: 'https://awesome-lib.js', body: true }  
23       ],  
24       noscript: [  
25         // <noscript>Javascript is required</noscript>  
26         { children: 'Javascript is required' }  
27       ]  
28     }  
29   }  
30 })
```

2. 独立设置 - useHead()

 快速上手

Vue | 复制代码

```

1  
```

▼ pages/some-page.vue

Vue | 复制代码

```

1 <script setup>
2 definePageMeta({
3   title: 'Some Page'
4 })
5 </script>

```

▼ layouts/default.vue

Vue | 复制代码

```

1 <script setup>
2 const route = useRoute()
3 useHead({
4   meta: [{ name: 'og:title', content: `App Name - ${route.meta.title}` }]
5 })
6 </script>

```

静态资源

public => /

磁盘路径 public / img / 1.jpg

网站路径

assets

磁盘路径 assets / img / 1.jpg

网站路径

更多配置说明：

<https://v3.nuxtjs.org/api/configuration/nuxt-config/>

[<https://v3.nuxtjs.org/api/configuration/nuxt-config/>](https://v3.nuxtjs.org/api/configuration/nuxt-config/)

获取数据

与 axios 对比

💡 快速上手

1. 创建并启动一个服务器, 用于统计请求次数

5457字

▼ server.js

JavaScript | ⚡运行代码 ⌂ 复制代码

```

1  const http = require('http')
2
3  // 1. 创建 web服务器
4  let n = 0
5  const server = http.createServer((req, res) => {
6      console.log(n++)
7      res.setHeader('Content-Type', 'text/html;charset=utf-8')
8      res.end('拉勾教育')
9  })
10 // 2. 设置 web服务器 监听3000端口
11 server.listen(80, () => {
12     console.log('服务器运行了')
13 })

```

node server.js

2. 编辑 pages / user / index.vue

▼

Vue | ⌂ 复制代码

```

1  <template>
2      <h1>用户首页</h1>
3  </template>
4
5  <script setup>
6  // 方式1：使用 axios 请求
7  import axios from 'axios'
8  axios.get('http://localhost').then(res => {
9      console.log(res)
10 })
11 // 方式2：使用 内置函数 请求
12 /*
13  useFetch('http://localhost').then(res => {
14      console.log(res)
15  })
16 */
17 </script>
18
19 <style lang="scss"></style>

```

⚠ \$fetch · Nuxt Utils[<https://nuxt.com/docs/api/utils/dollarfetch>](https://nuxt.com/docs/api/utils/dollarfetch)**首选方式****\$fetch是在Nuxt 3中进行HTTP调用的****💡 快速上手**

useFetch、useAsyncData、useLazyFetch、useLazyAsyncData

user / index.vue

Vue | 复制代码

```

1  <template>
2      <h1>用户首页</h1>
3  </template>
4
5  <script setup>
6  useFetch('/posts', {
7      method: 'GET',
8      baseURL: 'https://jsonplaceholder.typicode.com'
9  }).then(res => {
10     console.log(res)
11 })
12
13 /*
14     useAsyncData('获取文章', () => $fetch('/posts', {
15         method: 'GET',
16         baseURL: 'https://jsonplaceholder.typicode.com'
17     })).then(res => {
18         console.log(res.data)
19     })
20 */
21 </script>
22
23 <style lang="scss"></style>
24

```

返回值

data: 请求的结果

pending: 一个布尔值，指示是否仍在获取数据

refresh: 可用于刷新处理程序函数返回的数据的函数

error: 如果数据获取失败，则返回错误对象

Lazy

查看效果

0. 使用 useFetch 请求数据

1. 设置 Network / Slow 3G

2. 从其它地址，点击切换到 /user (现象：先卡住不动，等请求完毕后，切换)

3. 改为 useLazyFetch 请求数据

4. 从其它地址, 点击切换到 /user (现象: 先切换, 显示静态内容, 请求完毕后渲染数据)

更多配置项

 useFetch · Nuxt Composables

配置说明

<<https://nuxt.com/docs/api/composables/use-fetch>>

头信息

请求拦截 / 响应拦截

同构的 cookie 处理、token 处理

 Data Fetching · Get Started with Nuxt

<<https://nuxt.com/docs/getting-started/data-fetching#isomorphic-fetch-and-fetch>>

同构渲染中, 刷新页面, 向接口请求数据的动作, 将在服务器端发生, 如果请求时需要 cookie、token, 服务器端是无法提供的, 此时可以从客户端请求头获取相应信息

useRequestHeaders() 获取所有请求头信息

useRequestHeaders(['cookie']) 获取指定的请求头信息

例1:

例2:

在浏览器环境时, useRequestHeaders 将返回一个空对象。

具体看官方文档 API 部分

如果是请求需要写回给客户端

 快速上手

useRequestEvent()

浏览器环境中, useRequestEvent 将返回 `undefined`.

console.log(process.server, process.client)

Server API

自动扫描相关目录:

`~ / server / api /`
`~ / server / routes /`
`~ / server / middleware /`

目录中, 每个文件都应该导出一个用 `defineEventHandler ()` 定义的默认函数

处理程序可以直接返回JSON数据、Promise或使用`event.node.res.end ()` 发送响应。

有 /api 前缀

将文件存放到 `server / api` 目录下

1. 创建 `server / api / aaaa.ts`

2. 编辑 `pages / user / index.vue` - 请求接口

3. 通过从 `/user/create-or-edit` 切到 `/user` 查看控制台请求信息

无 /api 前缀

将文件放在 `server / routes` 目录下

1. 创建文件

💡 快速上手

2. 在组件页面中使用

获取路由参数

server / api / cccc / [name].ts

TypeScript | ⏪ 运行代码 ⌂ 复制代码

```
1 export default defineEventHandler((event) => `Hello, ${event.context.params.name}`)
```

获取查询字符串

server / api / cccc / [name].js

TypeScript | ⏪ 运行代码 ⌂ 复制代码

```
1 // http://localhost:3000/api/cccc/xxxx?a=10&b=20&c=30
2 export default defineEventHandler(event => {
3   const query = getQuery(event)
4   return query
5 })
```

请求方式

可以以.get、.post、.put、.delete...作为后缀，以匹配请求的HTTP方法

请求 body

server / utils 目录

 h3@1.0.2 - jsDocs.io

[<https://www.jsdocs.io/package/h3#package-index-functions>](https://www.jsdocs.io/package/h3#package-index-functions)

Catch-all Route

server / middleware 目录

中间件 接口验证

💡 快速上手

中间件处理程序 将在任何其他服务器路由之前对每个请求运行，以添加或检查标头、日志请求或扩展事件的请求对象。

中间件处理程序 不应返回任何内容（也不应关闭或响应请求），只应检查或扩展请求上下文或抛出错误。

▼ server / middleware / log.ts

TypeScript | ⚡ 运行代码 ⌂ 复制代码

```
1 export default defineEventHandler((event) => {  
2   // console.log(event.node.req)  
3   console.log('----服务中间件----')  
4 })
```

文档中其它

请求代理

状态管理

1. 定义需要-共享的数据 - 创建 composables / use-counter.ts

▼

TypeScript | ⚡ 运行代码 ⌂ 复制代码

```
1 export const useCounter = () => useState('counter', () => 0)
```

2. 操作改变-共享的数据 - 编辑 user / index.vue

▼ pages / user / index.vue

Vue | 复制代码

```

1  
2    <h1>用户首页</h1>
3    <h2>{{ counter }}</h2>
4    <button @click="counter++">加1</button>
5    <button @click="addCounter">加2</button>
6  </template>
7
8  
```

▼ nuxt.config.ts

TypeScript | ⏪运行代码 ⏪ 复制代码

```

1  export default defineNuxtConfig({
2    runtimeConfig: {
3      aaaa: '123',          // 只能在 服务端 使用
4      isServer: true,
5    public: {              // 服务端和客户端都可以使用
6      apiBase: 'https://200.run'
7    }
8  }
9 })

```

在其它地方 - 使用变量

▼

Vue | ⏪ 复制代码

```

1 <script setup>
2   const runtimeConfig = useRuntimeConfig()
3 </script>

```

在 .env 中定义的数据可覆盖 nuxt.config.ts 中数据

▼ .env

Plain Text | ⏪ 复制代码

```

1 # This will override the value of apiSecret
2 NUXT_AAAA=200
3 NUXT_PUBLIC_API_BASE='http://www.jd.com'

```

只在服务器端执行

▼ nuxt.config.ts

Vue | ⏪ 复制代码

```

1 export default defineNuxtConfig({
2   runtimeConfig: {
3     isServer: true,
4   public: {
5     apiBase: 'http://www.jd.com'
6   }
7 }
8 })

```

Vue | 复制代码

```

1 <script setup>
2 const runtimeConfig = useRuntimeConfig()
3 if (runtimeConfig.isServer) {
4   console.log('只在服务器端执行，因为客户端访问isServer时，会得到 undefined')
5 }
6 </script>

```

app.config.ts

在配置文件中 - 设置变量

```

▼ app.config.ts
TypeScript | ⚡运行代码 | 复制代码

1 export default defineAppConfig({
2   title: 'Hello Nuxt',
3   theme: {
4     dark: true,
5     colors: {
6       primary: '#ff0000'
7     }
8   }
9 })

```

在其它地方 - 使用变量

```

▼
Vue | 复制代码

1 <script setup>
2 const appConfig = useAppConfig()
3 </script>

```

部署 - Node.js Server

需要一个独立的 node 服务器, 来满足同构渲染的需要. 也包括项目自有 API 实现的需要(如果有的话)

 快速上手

项目构建

5457字

https://www.yuque.com/_h2o/vmv5lo/zfhleo#P3ofR

25/37

npm run build // 需要清空之前的 .nuxt 和 .output 目录

直接运行

node .output/server/index.mjs

端口和地址可以通过 process.env 配置, 但没必要, 不要纠结

使用 PM2

可以用于生产环境的Nodejs的进程管理工具, 并且它内置一个负载均衡。它不仅可以保证服务不会中断一直在线, 并且提供0秒reload功能, 还有其他一系列进程管理、监控功能

更多参考:

<https://www.jianshu.com/p/bab31fac7655>

<<https://www.jianshu.com/p/bab31fac7655>>

<https://www.cnblogs.com/kunmomo/p/14990703.html>

<<https://www.cnblogs.com/kunmomo/p/14990703.html>>

PM2 官网: <https://pm2.keymetrics.io/> <<https://pm2.keymetrics.io/>>

1. 安装 pm2

`npm i pm2 -g // 可能需要管理员权限`

`pm2 -v`

2. 创建并编辑 运行配置文件 - ecosystem.config.js

 快速上手

▼ ecosystem.config.js

JavaScript | ⚡运行代码 ⌂ 复制代码

```

1  module.exports = {
2    "apps": [
3      {"script": "./bin/www",
4       "args": ["-p", "3001"], //node的args参数 等同于 node ./bin/www -p
5       "node_args": "--harmony", //node harmony模式启动
6       "merge_logs": true,
7       "cwd": "./",
8       "log_file": "./log/combined.outerr.log",
9       "out_file": "./log/out.log",
10      "error_file": "./log/err.log",
11      "name": "my-nuxt3",
12      "exec_mode": 'cluster',
13      "instances": 'max',
14      "script": './output/server/index.mjs',
15      "env": { //node的 env参数，可以通过 process.env.xxx获取
16        NODE_ENV: "production",
17        PORT: 8080
18      }
19    }]
20  }

```

3. 基于配置文件 - 运行项目

pm2 start

4. 其它 pm2 命令 (xx 可以是 id , 也可以是名称)

pm2 stop xx / pm2 stop all

pm2 delete xx / pm2 delete all

pm2 start xx / pm2 start all

pm2 ls

渲染策略

文档位置: nuxt.com -> Started -> Guide / Rendering Modes / Route Rules

💡 快速上手

nuxt.config.js

JavaScript | ⚡运行代码 ⌂ 复制代码

```

1  export default defineNuxtConfig({
2    routeRules: {
3      '/blog/**': { swr: true }, // Static page generated on-demand, revalidat
4      '/articles/**': { static: true }, // Static page generated on-demand on
5      '/_nuxt/**': { headers: { 'cache-control': 's-maxage=0' } }, // 设置响应
6      '/admin/**': { ssr: false }, // Render these routes with SPA
7      // Add cors headers
8      '/api/v1/**': { cors: true },
9      // Add redirect headers
10     '/old-page': { redirect: '/new-page' },
11     '/old-page2': { redirect: { to: '/new-page', statusCode: 302 } }
12   }
13 })
14

```

另一个参考

nuxt.config.js

TypeScript | ⚡运行代码 ⌂ 复制代码

```

1  export default defineNuxtConfig({
2    routes: {
3      '/': { prerender: true }, // 每一次构建时，都重新预渲染页面（透过 Buil
4      '/blog/*': { static: true }, // 接收到一个请求时，页面依照需求重新渲染页面
5      '/stats/*': { swr: '10 min' }, // 接收到一个请求时，10 分钟的快取缓存过期后
6      '/admin/*': { ssr: false }, // 仅在客户端渲染
7      '/react/*': { redirect: '/vue' }, // 路由重定向
8    }
9  })

```

为视频准备的示例

nuxt.config.js

JavaScript | ⚡运行代码 ⌂ 复制代码

```

1  export default defineNuxtConfig({
2    routeRules: {
3      '/xxx/yyy/zzz': { prerender: true },
4      '/admin/**': { ssr: false },
5      '/old-page': { redirect: '/user/create-or-edit' },
6      '/user/**': { static: true, headers: { 'cache-control': 's-maxage=123456' } }
7      '/api/aaaa': { cors: true }
8    }
9  })
10

```

快速上手

纯 SPA

不使用同构渲染

▼ nuxt.config.js

JavaScript | ⚡ 运行代码 ⌂ 复制代码

```
1 export default defineNuxtConfig({
2   ssr: false
3 })
```

部署 - Static Hosting

静态部署. 不需要独立的 Node 服务器. 意味着不能实现同构渲染(SSR)和项目自己的API接口

纯静态 SSG

npm run generate

动态路由生成静态页面

1. 编辑 nuxt.config.ts

```
1 let routes = [] // 保存拼接好的 动态路由
2 for(let i = 1; i <= 100; i++) {
3   routes.push(`/article/${i}`)
4 }
5 export default defineNuxtConfig({
6   // ssr: false,
7   generate: {
8     routes: routes
9   }
10 })
11
```

2. 执行命令, 生成静态页面

npm run generate

 快速上手

自动导入

data fetching useNuxtApp() useState() useRuntimeConfig()
ref computed 生命周期函数 助手函数(useXxx...)
components composables utils
自动导入使用，无须手工导入。子目录中的，想自动导入怎么办? export {}
from
明确导入

TypeScript | ⚡运行代码 ⌂ 复制代码

```
1 <script setup>
2   import { ref, computed } from '#imports'
3
4   const count = ref(1)
5   const double = computed(() => count.value * 2)
6 </script>
7
```

禁止自动导入

TypeScript | ⚡运行代码 ⌂ 复制代码

```
1 - export default defineNuxtConfig({
2 -   imports: {
3     autoImport: false
4   }
5 })
6
```

添加自动导入的目录. 不会覆盖已有的

TypeScript | ⚡运行代码 ⌂ 复制代码

```
1 - imports: {
2   // Auto-import pinia stores defined in `~/stores`
3   dirs: ['stores']
4 }
5
```

💡 快速上手

useNuxtApp()

获取实例

具体看官方文档 API 部分

useRequestHeaders()

```
1 // Get all request headers
2 const headers = useRequestHeaders()
3
4 // Get only cookie request header
5 const headers = useRequestHeaders(['cookie'])
6
```

Element-Plus for Nuxt

推荐方式

github 说明: <https://github.com/element-plus/element-plus-nuxt>

[<https://github.com/element-plus/element-plus-nuxt>](https://github.com/element-plus/element-plus-nuxt)

1. 安装依赖

npm i @element-plus/nuxt

2. 进行配置

 快速上手

nuxt.config.ts

TypeScript | ⏪运行代码 ⏪ 复制代码

```

1  export default defineNuxtConfig({
2    modules: [
3      '@element-plus/nuxt'
4    ],
5    elementPlus: { /* Options */ }
6  })

```

3. 测试使用

Vue | ⏪ 复制代码

```

1 <template>
2   <h1>用户列表</h1>
3
4   <el-button @click="handleClick" type="primary">哈哈哈哈</el-button>
5
6 </template>
7
8 <script lang="ts" setup>
9 ...
10 const handleClick = () => {
11   ElMessage.success('成功啦!')
12 }
13 ...
14 </script>
15
16 <style lang="scss"></style>

```

传统方式

element-plus 官网: <https://element-plus.gitee.io/> <<https://element-plus.gitee.io/>>

1. 安装依赖

```
npm i element-plus
```

```
npm i -D unplugin-vue-components unplugin-auto-import // 自动按需导入插件
```

2. 配置 vite - Nuxt 中配置 Vite 的方式

 快速上手

1) Vite 按需导入配置

5457字

<https://element-plus.gitee.io/zh-CN/guide/quickstart.html#%E6%8C%89%E9%9C%80%E5%AF%BC%E5%85%A5> <<https://element-plus.gitee.io/zh-CN/guide/quickstart.html#%E6%8C%89%E9%9C%80%E5%AF%BC%E5%85%A5>>

2) Nuxt 中对 Vite 进行配置

<https://nuxt.com/docs/getting-started/configuration>
<<https://nuxt.com/docs/getting-started/configuration>>

3. 测试使用

Nuxt3 beta 使用问题总结

<https://ainyi.com/126> <<https://ainyi.com/126>>

增量

<https://devpress.csdn.net/vue/62f8fc417e6682346618b37b.html>
<<https://devpress.csdn.net/vue/62f8fc417e6682346618b37b.html>>

generate 参考 (不一定有效)

💡 快速上手



JavaScript

①运行代码 ②复制代码

```

1 module.exports = {
2   generate: {
3     dir: 'generateDist', // 我们生成的静态文件的目录
4     concurrency: 10, // 我们在批量生成文件的时候一次产生的文件数量
5     interval: 100, // 批量生成文件间隔
6     crawler: false, // 比如我们的页面中有a链接的话，会生成a链接指向的页面的
7     routes() {} // 我们想要生成静态页面对应的vue文件路由，它返回一个数
8   }
9 }
10

```

nuxt静态

[<https://www.bufeishi.cn/81311.html>](https://www.bufeishi.cn/81311.html)

Node.js 获取请求中的 IP 地址



TypeScript

①运行代码 ②复制代码

```

1 function getClientIp (req: any) {
2   return req.headers['x-forwarded-for'] ||
3     req.connection.remoteAddress ||
4     req.socket.remoteAddress ||
5     req.connection.socket.remoteAddress
6   // if (req.headers['x-forwarded-for']) {
7   //   console.log('x-forwarded-for', req.headers['x-forwarded-for'])
8   // } else if (req.connection.remoteAddress) {
9   //   console.log('req.connection.remoteAddress')
10  // } else if (req.socket.remoteAddress) {
11  //   console.log('req.socket.remoteAddress')
12  // } else if (req.connection.socket.remoteAddress) {
13  //   console.log('req.connection.socket.remoteAddress')
14  // }
15 }

```

参考

Nuxt3项目搭建（Nuxt3+element-plus+scss详细步骤）

快速上手

https://blog.csdn.net/m0_48489737/article/details/127325786
<https://blog.csdn.net/m0_48489737/article/details/127325786>

Nuxt3 rc.11 填坑 配置 element-plus

<https://blog.csdn.net/u011866450/article/details/122128790>
<<https://blog.csdn.net/u011866450/article/details/122128790>>

Nuxt3 应用创建与配置

<https://zhuanlan.zhihu.com/p/535230945>
<<https://zhuanlan.zhihu.com/p/535230945>>

组件懒加载 组件名前面加上 Lazy 前缀，则可以按需加载该组件，可用于优化打包尺寸

<LazyMountainsList>

更深层次的思考

<https://github.com/unjs/ohmyfetch#%EF%B8%8F-interceptors>
<<https://github.com/unjs/ohmyfetch#%EF%B8%8F-interceptors>>

 快速上手

▼

Vue | 复制代码

```
1 ▼ <slot name="xxx" />
2   <slot name="yyy" />
```

▼

Vue | 复制代码

```
1 ▼ <template #xxxx>
2   </template>
3
4 ▼ <template #yyy>
5   </template>
```

nuxt-app/composables/useCustomFetch.ts

TypeScript | 运行代码 | 复制代码

```
1 // 总地址: https://github.com/nuxt/framework/discussions
2 // 具体讨论: https://github.com/nuxt/framework/discussions/5370
3 ▼ export const useCustomFetch = (url: string, options?: FetchOptions) => {
4   return useFetch(url, {
5     ...options,
6     async onResponse({ request, response, options }) {
7       console.log('[fetch response]')
8     },
9     async onResponseError({ request, response, options }) {
10       console.log('[fetch response error]')
11     },
12
13     async onRequest({ request, options }) {
14       console.log('[fetch request]')
15     },
16     async onRequestError({ request, options, error }) {
17       console.log('[fetch request error]')
18     }
19   })
20 }
```

▼

JavaScript | ⚡运行代码 ⌂ 复制代码

```
1  @nuxtjs/tailwindcss
2  @headlessui/tailwindcss
3  @headlessui/vue
4  @pinia/nuxt
5  @tailwindcss/forms
6  @vee-validate/rules
7  dayjs
8  joi
9  mongoose
10 nuxt-icon
11 vue3-easy-data-table
12 vue-toastification
13 vee-validate
14 pinia
```

..@澎湃·深水

..@澎湃·深水

💡 快速上手