

一. SSR

<https://ssr.vuejs.org/zh/guide/#%E5%AE%89%E8%A3%85>

```
1 // 第 1 步: 创建一个 Vue 实例
2 const Vue = require('vue')
3 const app = new Vue({
4   template: `<div>Hello World</div>`
5 })
6
7 // 第 2 步: 创建一个 renderer
8 const renderer = require('vue-server-renderer').createRenderer()
9
10 // 第 3 步: 将 Vue 实例渲染为 HTML
11 renderer.renderToString(app).then(html => {
12   console.log(html)
13 }).catch(err => {
14   console.error(err)
15 })
16
```

二. Nuxt.js

1. Nuxt.js介绍与安装

<https://zh.nuxtjs.org/guide>

npx create-nuxt-app <项目名>

服务端渲染， 解决首屏加载速度， 和 seo问题

```
1 //错误 HTMLElement is not define
2
3 修改nuxt.config.js 中plugins
4 plugins: [
5   // '@plugins/element-ui',
6   { src: '@plugins/element-ui', ssr: false}
7 ]
8 //不要复制 ， 格式有问题
```

2. Nuxt.js的配置

<https://zh.nuxtjs.org/guide/configuration>

3. 路由

Nuxt.js 依据 pages 目录结构自动生成 vue-router 模块的路由配置。

(1) 要在页面之间使用路由，我们建议使用<nuxt-link> 标签。 支持activeClass ,tag

(2)

1 pages/

```
2  --| user/
3  -----| index.vue
4  -----| one.vue
5  --| index.vue
6
7
```

8 那么，Nuxt.js 自动生成的路由配置如下：

```
9
10 router: {
11   routes: [
12     {
13       name: 'index',
14       path: '/',
15       component: 'pages/index.vue'
16     },
17     {
18       name: 'user',
19       path: '/user',
20       component: 'pages/user/index.vue'
21     },
22     {
23       name: 'user-one',
24       path: '/user/one',
25       component: 'pages/user/one.vue'
26     }
27   ]
28 }
```

(3) 嵌套路由

创建内嵌子路由，你需要添加一个 Vue 文件，同时添加一个与该文件同名的目录用来存放子视图组件。

Warning: 别忘了在父组件(.vue文件) 内增加 <nuxt-child/> 用于显示子视图内容。

```
1  pages/
2  --| film/
3  -----| nowplaying.vue
4  -----| comingsoon.vue
5  --| film.vue
```

(4) 重定向

a. nuxt.config.js

```
1  router:{
2    extendRoutes (routes, resolve) {
3      routes.push({
4        path: '/',
```

```

5         redirect: '/film'
6     })
7 }
8 }

```

b. 利用中间件来处理

```

1 // 中间件 middle/ redirect.js
2 export default function({ isHMR, app, store, route, params, error, redirect }) {
3     if (isHMR) return
4     // 页面均放在_lang文件夹下，即lang为动态路由参数
5     /*if (!params.lang) { //此写法会出现路由重定向次数过多的问题
6         return redirect('/') + defaultLocale + '/' + route.fullPath)
7     }
8     */
9     if(route.fullPath == '/film') {
10         return redirect('/film/nowplaying')
11     }
12 }
13 router: {
14     middleware: 'redirect' // 即每次路由跳转会调用该中间件
15     //多个中间件写法
16     // middleware: ['redirect']
17 }

```

(5) 动态路由

必须加下划线（文件夹也可以加下划线(多级嵌套)， 文件也可以加下划线）

```

1 pages/
2 --| detail/
3 -----| _id.vue
4
5
6 //程式跳转 this.$router.push("/detail");

```

(6) 获取动态路由参数

```

1 asyncData({params}){
2     console.log(params.id);
3 }

```

4. 视图

在layout 里面 写好default.vue 可以认为这是根组件的模板了，

所有的组件都加在里面， 但是有些页面 可能不一样，就可以使用 个性化定制页面。

举个例子 layouts/template.vue:

```
1 <template>
2   <div>
3     <div>这个页面不需要导航栏</div>
4   </div>
5 </template>
```

8 在 `pages/detail.vue` 里， 可以指定页面组件使用 `template` 布局。

```
10 <script>
11 export default {
12   layout: 'template'
13 }
14 </script>
```

5. 异步数据与资源文件

(1) 如果组件的数据不需要异步获取或处理，可以直接返回指定的字面对象作为组件的数据。

```
1 export default {
2   data () {
3     return { foo: 'bar' }
4   }
5 }
```

(2)使用 `req/res(request/response)` 对象

1 在服务器端调用`asyncData`时，您可以访问用户请求的`req`和`res`对象。

2 在当前页面刷新， 服务端执行此函数

3 从其他页面跳转过来， 客户端执行此函数

```
5 export default {
6   async asyncData ({ req, res }) {
7     // 请检查您是否在服务器端
8     // 使用 req 和 res
9     if (process.server) { //判断是否在服务器被调用
10       //process.client 判断是否在客户端被调用
11       return { host: req.headers.host }
12     }
13
14     return {}
15   }
16 }
```

(3)错误处理

Nuxt.js 在上下文对象`context`中提供了一个 `error(params)` 方法，

你可以通过调用该方法来显示错误信息页面。`params.statusCode` 可用于指定服务端返回的请求状态码。

以返回 Promise 的方式举个例子：

```
1 export default {
2   asyncData ({ params, error }) {
3     return axios.get(`https://my-api/posts/${params.id}`)
4       .then((res) => {
5         return { title: res.data.title }
6       })
7     .catch((e) => {
8       error({ statusCode: 404, message: 'Post not found' })
9     })
10  }
11 }
```

(4)反向代理的配置 （重启服务器）

```
1 npm i @nuxtjs/proxy -D
2 在 nuxt.config.js 配置文件中添加对应的模块，并设置代理
```

```
3
4  modules: [
5     '@nuxtjs/axios', //添加axios
6     '@nuxtjs/proxy'  //添加proxy模块
7  ],
8  axios: {
9    proxy: true
10 },
11 proxy: {
12   '/api': {
13     target: 'http://example.com',
14     pathRewrite: {
```

```
15     '^/api' : '/'
16   }
17 }
18 }
```

20 这样就配置好了webpack的反向代理。

21 为了在服务端和客户端都工作， 需要

22

```
23 axios.get((process.server?'https://h5.ele.me':'')+"/restapi/shop.....e&terminal=h5").then
24   console.log(res.data)
25 })
```

如果上线了， 需要在node中配置好 http-proxy-middleware 就工作了。

(5) 轮播，下拉刷新， 是必须在浏览器环境中运行

```
1 // plugins/vue-swipe.js
2 import { Swipe, SwipeItem } from 'vue-swipe'
3 import Vue from 'vue'
4 import 'vue-swipe/dist/vue-swipe.css'
5
6 Vue.component('swipe', Swipe);
7 Vue.component('swipe-item', SwipeItem);
8
9 // nuxt.config.js
10 module.exports = {
11   plugins: [
12     { src: '~/plugins/vue-swipe', ssr: false }
13   ]
14 }
15 直接限制这个插件只在浏览器环境下被引入。(但是会报个警告)
16
17 或者: 只要在mounted()中初始化,
18 因为服务端渲染只会执行beforeCreate 以及created两个生命周期。
```