关于vue项目的seo问题

我们不可避免会考虑到的是seo问题，这直接关系到我们网站的排名，很多人说用vue搭建的网站不能做优化，那我们真的要放弃vue，放弃前后端分离开发么?
首先，可以肯定的是前后端分离不利于SEO，为什么呢?
1.搜索引擎的基础爬虫的原理就是抓取你的url，然后获取你的html源代码并解析。 而你的页面通常用了vue等js的数据绑定机制来展示页面数据，爬虫获取到的html是你的模型页面而不是最终数据的渲染页面，所以说用js来渲染数据对seo并不友好。
2.seo 本质是一个服务器向另一个服务器发起请求，解析请求内容。但一般来说搜索引擎是不回去执行请求到的js的。也就是说，如果一个单页应用，html在服务器端还没有渲染部分数据数据，在浏览器才渲染出数据，而搜索引擎请求到的html是没有渲染数据的。 这样就很不利于内容被搜索引擎搜索到。 所以服务端渲染就是尽量在服务器发送到浏览器前页面上就是有数据的。
3.一般的数据逻辑操作是放在后端的。排序这个如果仅仅是几条数据，前后端排序开起来是一样的，如果是有1000条数据，前端要排序就要都请求过来。这样显然是不合理的。

常用的解决方案有三种:
1. 页面预渲染
2. 服务端渲染
3. 路由采用h5 history模式

https://zhuanlan.zhihu.com/p/29148760
（设置vue 单页面meta info信息，如果需要单页面SEO，可以和prerender-spa-plugin形成更优的配合）

npm install prerender-spa-plugin --save；
`npm i vue-meta-info --save`

1：修改router下index.js：hash模式改为history模式，路径加一层base
{
下两种模式:

- hash —— 即地址栏 URL 中的 # 符号（此 hash 不是密码学里的

散列运算）。比如这个 URL：http://www.abc.com/#/hello，hash 的值为 #/hello。它的特点在于：hash 虽然出现在 URL 中，但不会被包括在 HTTP 请求中，对后端完全没有影响，因此改变 hash 不会重新加载页面。

- history ——利用了 HTML5 History Interface 中新增的 pushState() 和 replaceState() 方法。（需要特定浏览器支持）这两个方法应用于浏览器的历史记录栈，在当前已有的 back、forward、go 的基础之上，它们提供了对历史记录进行修改的功能。只是当它们执行修改时，虽然改变了当前的 URL，但浏览器不会立即向后端发送请求。
- 因此可以说，hash 模式和 history 模式都属于浏览器自身的特性，Vue-Router 只是利用了这两个特性（通过调用浏览器提供的接口）来实现前端路由.
- 但是hash模式下url需要带“#”符号，不仅看起来不舒服，而且有些场景下是会破坏路由中的"#"（微信分享页面就会把"#"后边的内容处理掉，需要将带参的url作为一个参数传给后台，后台取不到#后面的东西）在后面需要开发微信支付、分享,授权登录等就暴露出了问题，所以就需要使用history模式。

- 接着说一下微信支付，也是因为**凡是涉及充值的页面，不能做成带hash路由的的页面！，于是也要用history模式**

/**此处是改成history模式后，但是不能刷新，一刷新，然后就找不到页面，公司要求是要有当前页面的刷新功能，还要有直接通过链接跳转到指定页面的功能，显然history模式是不符合要求的，所以还得用hash模式，改回hash模式后，支付页面竟然可以支付了，应该是微信公众号配置的路径域名后加上了dist,vue项目router文件夹下的index.js加入**base: '/dist'的缘故****/**

}

2: webpack.prod.conf.js

```
  const PrerenderSpaPlugin = require('prerender-spa-plugin')

const Renderer = PrerenderSpaPlugin.PuppeteerRenderer

    new PrerenderSpaPlugin({
```

```
      // 生成文件的路径，也可以与webpakc打包的一致。
      // 下面这句话非常重要！！！
      // 这个目录只能有一级，如果目录层次大于一级，在生成的时候不会有任何错误提示，在预渲染的时候只会卡着不动。
      staticDir: path.join(__dirname, '../dist'),
      // 对应自己的路由文件，比如index有参数，就需要写成 /index/param1。
      routes: ['/', '/about', '/my'],
      // 这个很重要，如果没有配置这段，也不会进行预编译
      renderer: new Renderer({
        inject: {
          foo: 'bar'
        },
        headless: false,
        // 在 main.js 中 document.dispatchEvent(new Event('render-event'))，两者的事件名称要对应上。
        renderAfterDocumentEvent: 'render-event'
      })
    })
```

3: 开发目录main.js

```
 mounted () {
    document.dispatchEvent(new Event('render-event'))
  }
```

4: 结合管理头部标签插件vue-meta-info

main.js中加

```
import MetaInfo from 'vue-meta-info'
Vue.use(MetaInfo)

export default {
  metaInfo: {
    title: 'We Inc',
    meta: [
      {
        name: 'keywords',
        content: '关键字1,关键字2,关键字3'
      },
      {
```

```
        name: 'description',
        content: '这是一段网页的描述'
      }
    ]
  }
}
```

就可以将关键字预渲染到html的页面中去

```
================================================================================
```