服务器端渲染和客户端渲染

什么是服务器端渲染和客户端渲染？

　互联网早期，用户使用浏览器浏览的都是一些没有复杂逻辑的、简单的页面，这些页面都是在后端将html拼接好的然后将之返回给前端完整的html文件，浏览器拿到这个html文件之后就可以直接解析展示了，而这也就是所谓的服务器端渲染了。而随着前端页面的复杂性提高，前端就不仅仅是普通的页面展示了，而可能添加了更多功能性的组件，复杂性更大，另外，彼时ajax的兴起，使得业界就开始推崇前后端分离的开发模式，即后端不提供完整的html页面，而是提供一些api使得前端可以获取到json数据，然后前端拿到json数据之后再在前端进行html页面的拼接，然后展示在浏览器上，这就是所谓的客户端渲染了，这样前端就可以专注UI的开发，后端专注于逻辑的开发。

两者本质的区别是什么？

　　客户端渲染和服务器端渲染的最重要的区别就是究竟是谁来完成html文件的完整拼接，如果是在服务器端完成的，然后返回给客户端，就是服务器端渲染，而如果是前端做了更多的工作完成了html的拼接，则就是客户端渲染。

服务器端渲染的优缺点是怎样的？

优点：

前端耗时少。因为后端拼接完了html，浏览器只需要直接渲染出来。

有利于SEO。因为在后端有完整的html页面，所以爬虫更容易爬取获得信息，更有利于seo。

无需占用客户端资源。即解析模板的工作完全交由后端来做，客户端只要解析标准的html页面即可，这样对于客户端的资源占用更少，尤其是移动端，也可以更省电。

后端生成静态化文件。即生成缓存片段，这样就可以减少数据库查询浪费的时间了，且对于数据变化不大的页面非常高效 。

缺点：

不利于前后端分离，开发效率低。使用服务器端渲染，则无法进行分工合作，则对于前端复杂度高的项目，不利于项目高效开发。另外，如果是服务器端渲染，则前端一般就是写一个静态html文件，然后后端再修改为模板，这样是非常低效的，并且还常常需要前后端共同完成修改的动作； 或者是前端直接完成html模板，然后交由后端。另外，如果后端改了模板，前端还需要根据改动的模板再调节css，这样使得前后端联调的时间增加。

占用服务器端资源。即服务器端完成html模板的解析，如果请求较多，会对服务器造成一定的访问压力。而如果使用前端渲染，就是把这些解析的压力分摊了前端，而这里确实完全交给了一个服务器。

客户端渲染的优缺点是怎样的？

优点：

前后端分离。前端专注于前端UI，后端专注于api开发，且前端有更多的选择性，而不需要遵循后端特定的模板。

体验更好。比如，我们将网站做成SPA或者部分内容做成SPA，这样，尤其是移动端，可以使体验更接近于原生app。

缺点：

前端响应较慢。如果是客户端渲染，前端还要进行拼接字符串的过程，需要耗费额外的时间，不如服务器端渲染速度快。

不利于SEO。目前比如百度、谷歌的爬虫对于SPA都是不认的，只是记录了一个页面，所以SEO很差。因为服务器端可能没有保存完整的html，而是前端通过js进行dom的拼接，那么爬虫无法爬取信息。 除非搜索引擎的seo可以增加对于JavaScript的爬取能力，这才能保证seo。

使用服务器端渲染还是客户端渲染？

　　不谈业务场景而盲目选择使用何种渲染方式都是耍流氓。比如企业级网站，主要功能是展示而没有复杂的交互，并且需要良好的SEO，则这时我们就需要使用服务器端渲染；而类似后台管理页面，交互性比较强，不需要seo的考虑，那么就可以使用客户端渲染。

　　另外，具体使用何种渲染方法并不是绝对的，比如现在一些网站采用了首屏服务器端渲染，即对于用户最开始打开的那个页面采用的是服务器端渲染，这样就保证了渲染速度，而其他的页面采用客户端渲染，这样就完成了前后端分离。

对于前后端分离，如果进行seo优化？

　　如果进行了前后端分离，那么前端就是通过js来修改dom使得html拼接完全，然后再显示，或者是使用SPA，这样，seo几乎没有。那么这种情况下如何做seo优化呢？

　　我们可以自行提交sitemap，让蜘蛛主动去爬取，但是遇到了sitemap中的url，达到指定页面之后只有元js怎么办呢？这是我们可以使用<noscript>标签来进行简单的优化，比如打印出当前页面信息的一些关键的信息点，但是正常用户并不需要这些，会造成额外的负担，且前端可以判断是否支持JavaScript，而后段不行，只好根据百度的spider做UA判断，使用phantomjs或者nginx代理，来对spider访问的页面进行特殊的处理，达到被收录的效果。但这种效果还是不好。。。

　　而目前的react和vue都提供了SSR，即服务器端渲染，这也就是提供seo不好的解决方式了。

究竟如何理解前后端分离？

　　实际上，时至今日，前后端分离一定是必然或者趋势，因为早期在web1.0时代的网页就是简单的网页，而如今的网页越来越朝向app前进，而前后端分离就是实现app的必然的结果。所以，我们可以认为html、css、JavaScript组成了这个app，然后浏览器作为虚拟机来运行这些程序，即浏览器成为了app的运行环境，成了客户端，总的来说就是当前的前端越来越朝向桌面应用或者说是手机上的app发展了，而比如说电脑上的qq可以服务器端渲染吗？肯定不能！所以前后端分离也就成了必然。而我们目前接触额前端工程化、编译（转译）、各种MVC/MVVM框架、依赖工具、npm、bable、webpack等等看似很新鲜、创新的东西实际上都是传动桌面开发所形成的概念，只是近年来前端发展较快而借鉴过来的，本质上就是开源社区东平西凑做出来的一个visual studio。