浅谈服务端渲染(SSR)

首屏服务端渲染（SSR：server side render）

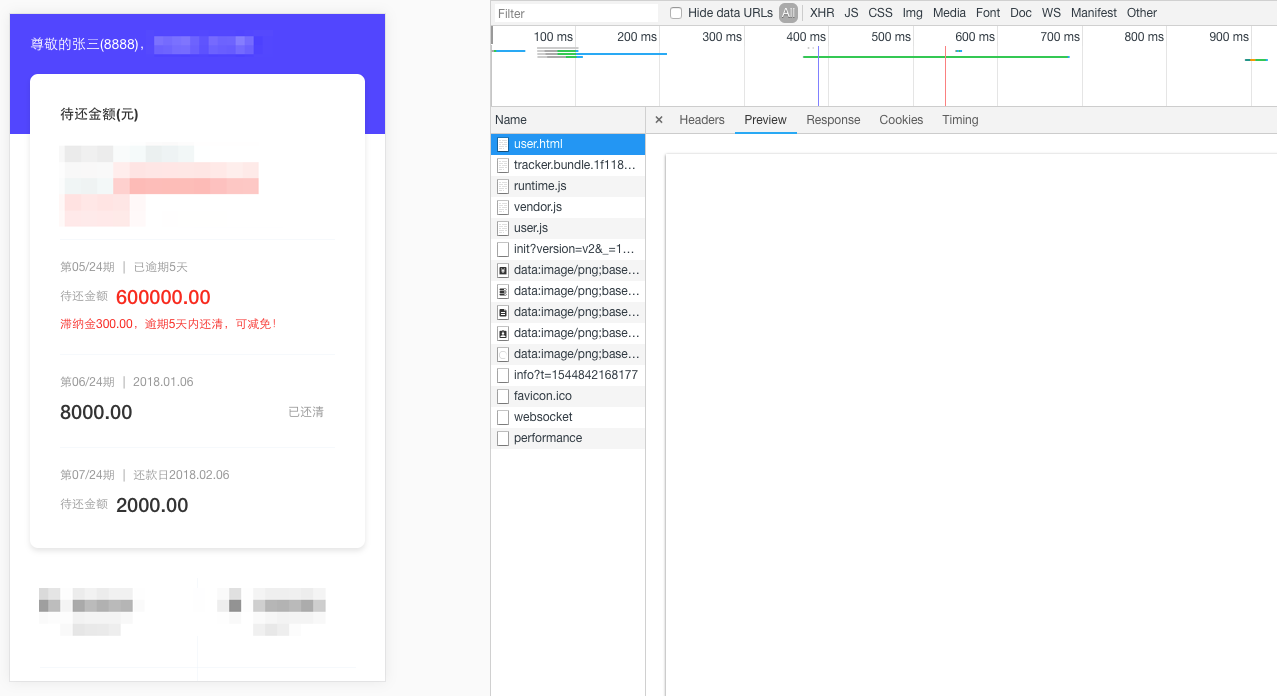
项目用到的技术： React 、webpack、koa2、webpack

对于重构成SSR，redux不是必须的，所以没用redux

一、 什么是服务端渲染

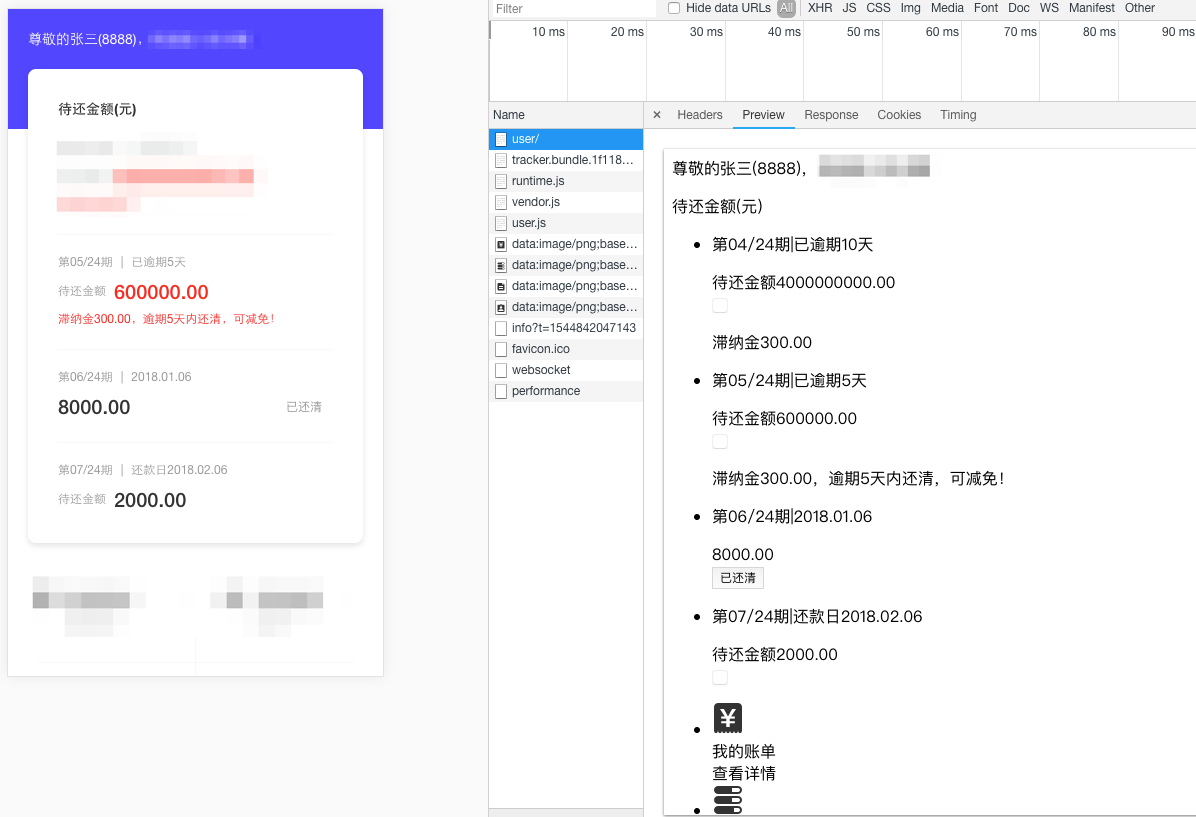
将组件或页面通过服务器生成html字符串，再发送到浏览器，最后将静态标记"混合"为客户端上完全交互的应用程序

如下图所示，  
页面没使用服务渲染，当请求user页面时，返回的body里为空，之后执行js将html结构注入到body里，结合css显示出来；



(图一)

下面使用了服务端渲染，当请求user页面时，返回的body里已经有了首屏的html结构，之后结合css显示出来



二、SSR的优势

1. 更利于SEO。

不同爬虫工作原理类似，只会爬取源码，不会执行网站的任何脚本（Google除外，据说Googlebot可以运行javaScript）。使用了React或者其它MVVM框架，页面的大多数DOM元素都是在客户端根据js动态生成，可供爬虫抓取分析的内容大大减少(如图一)。浏览器爬虫不会等待我们的数据完成之后再去抓取我们的页面数据。服务端渲染返回给客户端的是已经获取了异步数据并执行JavaScript代码的最终HTML，网络爬中就可以抓取到完整页面的信息。

2. 更利于首屏渲染

首屏的渲染是node发送过来的html字符串，并不依赖于js文件了，这就会使用户更快的看到页面的内容。尤其是针对大型单页应用，打包后文件体积比较大，普通客户端渲染加载所有所需文件时间较长，首页就会有一个很长的白屏等待时间。

三、SSR的局限

服务端压力较大

本来是通过客户端完成渲染，现在统一到服务端node服务去做。尤其是在高并发访问的情况时，会大量占用服务端CPU资源。

开发条件受限

在服务端渲染中，只执行到componentDidMount之前的生命周期钩子函数，因此：项目引用到的第三方的库也不可用其它生命周期钩子，这对引用库的选择产生了很大的限制；

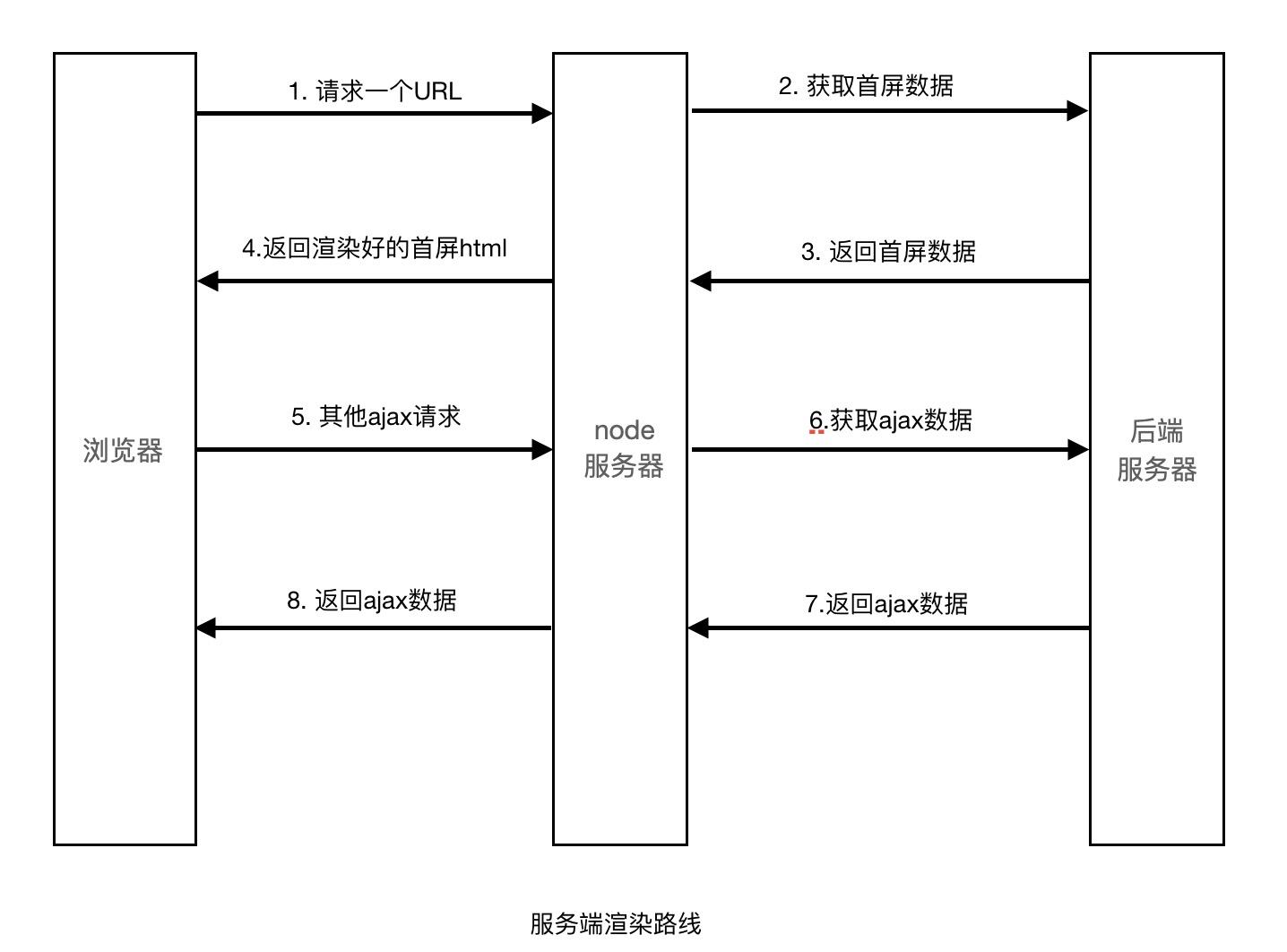
学习成本相对较高

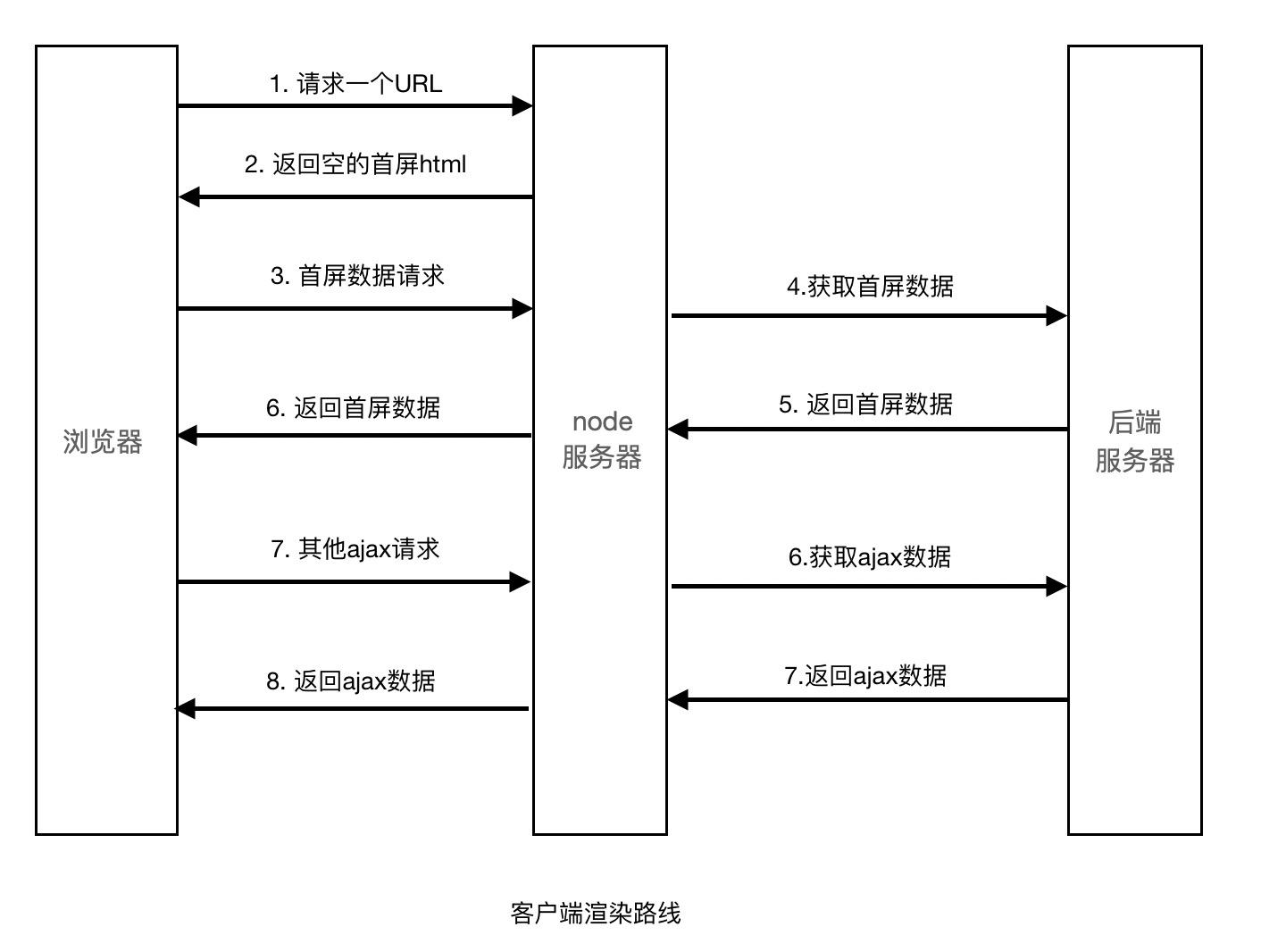
除了对webpack、React要熟悉，还需要掌握node、Koa2等相关技术。相对于客户端渲染，项目构建、部署过程更加复杂。

三、 时间耗时比较

数据请求

由服务端请求首屏数据，而不是由客户端请求首屏数据，这是“快”的一个主要原因。服务端在内网进行请求，数据响应速度快。客户端在不同网络环境，进行数据请求，且外网http请求开销大，导致时间长。 下图为服务端渲染数据的请求路线和客户端渲染数据的请求路线图





html渲染  
服务端渲染是先向后端服务器请求数据，然后生成完整首屏html返回给浏览器；而客户端渲染是等js代码下载、加载、解析完成后再请求数据渲染，等待的过程页面是什么都没有的，就是用户看到的白屏。就是服务端渲染不需要等待js代码下载完成就已经请求数据了，返回一个已有完整数据的首屏页面。

具体流程可参考下面两张图

