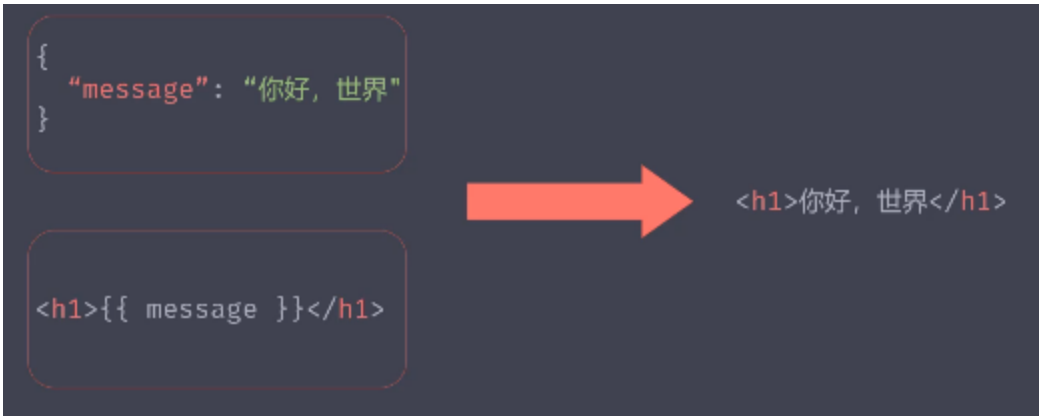
8.1 渲染基础

1.什么是渲染

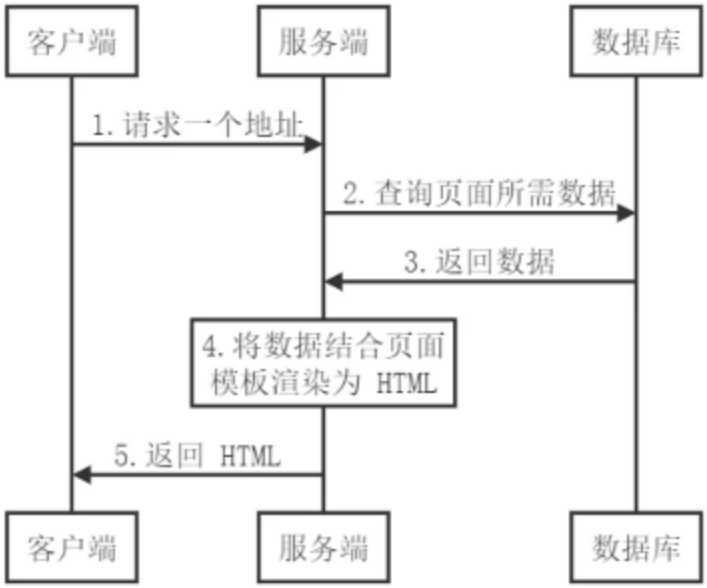
渲染：把数据 + 模板拼接到一起



我们关注的不是怎么渲染,而是在哪里渲染

2. 传统的服务端渲染

- 早起的web页面渲染都是在服务端进行的
- 服务端运行过程中,把所需要的的数据结合模板生成html,相应给浏览器
- 大概的流程如下

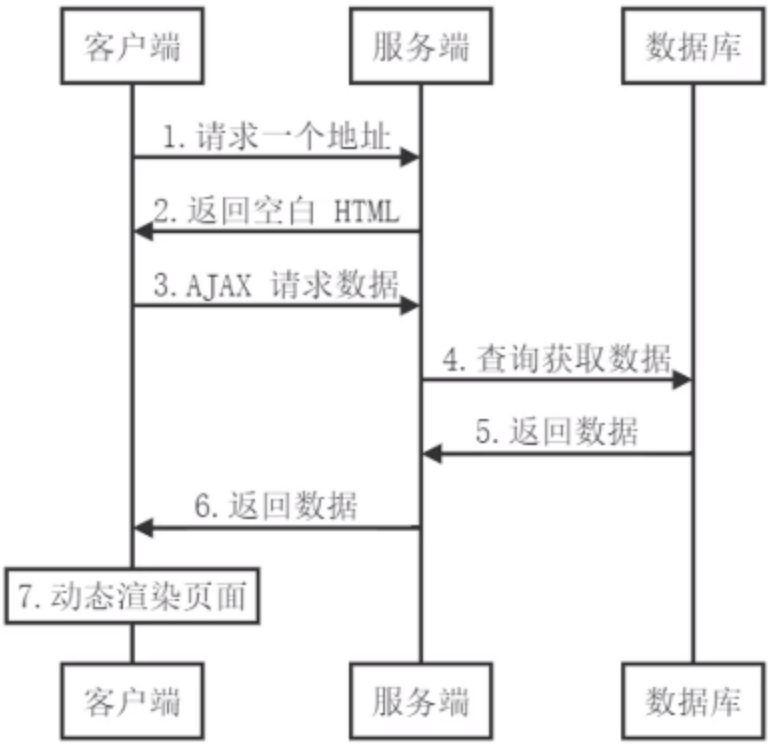
- 

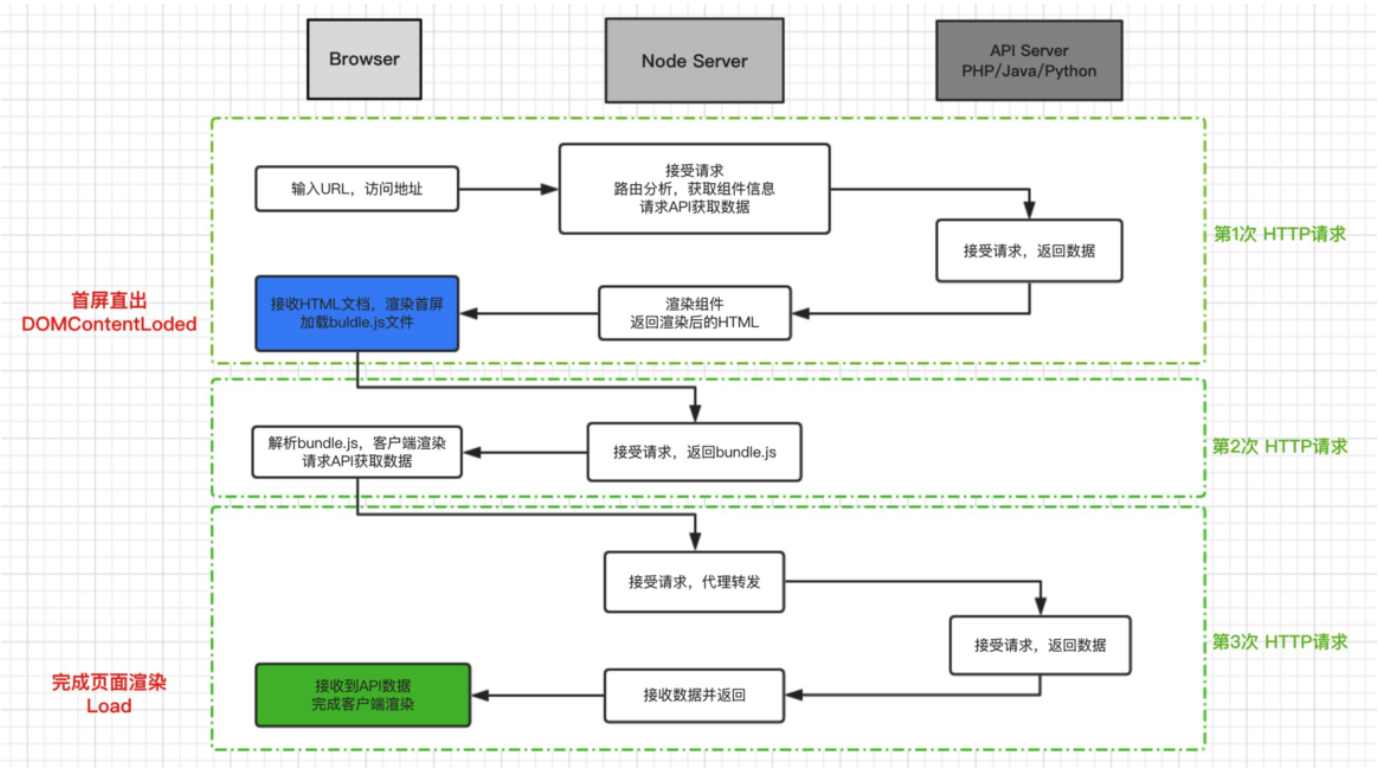
缺点:

- 前后端代码完全耦合在一起,不利于维护
- 前端没有足够的发挥空间,
- 服务端压力大
- 用户体验一遍,查看其它页面都要刷新浏览器

3. 客户端渲染

- 服务端渲染的缺点随着ajax的诞生普及得到有效的解决
- Ajax使客户端动态获取数据成为可能
- 服务端渲染工作 => 客户端渲染
- 客户端渲染流程

- 



缺点:

- 首屏渲染慢
- 不利于seo

4.为什么客户端渲染首屏慢

客户端渲染的TTFP（Time To First Page）时间比较长，一般起码需要3个HTTP请求周期： 加载HTML文档 -> 加载JS文件 -> API请求数据 -> 根据数据渲染页面 也就是初始化页面会出现白屏，性能上通过Node直出，将传统的三次串行http请求简化成一次http请求，降低首屏渲染时间

5.为什么客户端渲染不利于SEO

单页应用的SEO能力几乎为零 SPA首次加载的HTML文档没有内容，而目前大多数搜索引擎主要识别的内容还是 HTML，对 JavaScript 文件内容的识别都还比较弱，所以如果公司对SEO有需求（或者将来需要），那么SPA就不太适合了

6.同构渲染

nuxt基于vue框架，客户端渲染和服务器端渲染的结合，在服务器端执行一次，用于实现服务器端渲染（首屏直出），在客户端再执行一次，用于接管页面交互，核心解决SEO和首屏渲染慢的问题。

7. SSR的缺点

- **1、理论上，SSR（包括传统的服务端渲染）最大的瓶颈就是服务端的性能**如果用户规模大，SPA本身就是一个大型分布式系统，充分利用用户的设备去运行JS的运算，SSR则是把这些工作包揽到自己的服务器上。所以对于需要大量计算（图表特别多）而且用户量巨大的页面，并不太适合，但SSR非常适合于大部分的内容展示页面
- **2、项目复杂度增加，需要前端团队有较高的技术素养**为了同构要处处兼容 Node.js 不同的执行环境，不能有浏览器相关的原生代码在服务端执行，前端代码使用的 window 在 node 环境是不存在的，所以要 mock window，其中最重要的是 cookie，userAgent，location