# 服务端渲染（SSR）与同构开发

## 概要

- 纵观所有 Web 应用的前端渲染方式
- 五大页面渲染方式之间的对比（后端模板渲染、客户端渲染、Node.js 中间层、服务器端渲染、服务端预渲染（静态页面生成））
- SPA 应用的 SEO 问题
- React/Vue.js 中实现 SSR 与同构开发
- 服务端渲染的实现原理剖析

## 页面渲染方式

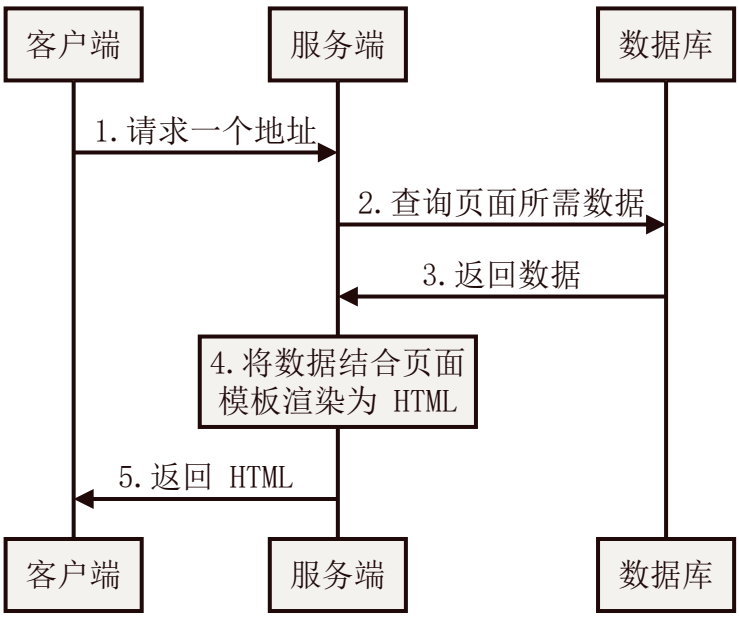
Web 应用的前端渲染方式大致经历的以下这几个阶段：

### 一、服务端渲染

注意这里的服务端渲染是指在后端通过模板渲染 HTML，并不是指后面要介绍的 SSR 方案。

最早期，Web 应用的前端渲染是在服务端完成的，即服务端运行过程中将所需的数据结合页面模板渲染为 HTML，响应给客户端浏览器。所以浏览器呈现出来的是直接包含内容的页面。

工作流程：




这也就是最早 Web 2.0 时代，动态网站的核心工作步骤。在这样的一个工作过程中，因为页面中的内容不是固定的，它有一些动态的内容，所以导致属于「前端」部分的 HTML，需要「后端」代码操作。

代表性技术有：ASP、PHP、JSP，再到后来的一些相对高级一点的服务端框架配合一些模板引擎。

在今天看来，这种前端渲染模式是不合理或者说不先进的。因为在当下这种前端越来越复杂的情况下，这种模式存在很多明显的不足：

- 应用的前后端部分完全耦合在一起，在前后端协同开发方面会有非常大的阻力；
- 前端没有足够的发挥空间，无法充分利用现在前端生态下的一些更优秀的方案；

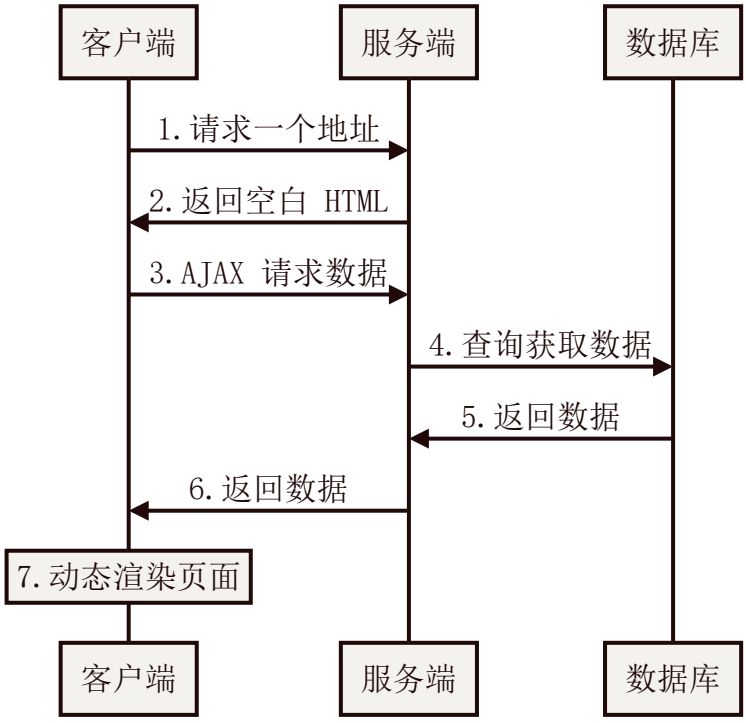
但是不得不说，在应用前端并不复杂的情况下，这种方式也是可取的。

## 二、客户端渲染

随着 AJAX 技术的普及，客户端动态获取数据变为可能，前后端分离的方式也越来越常见。现如今大多数 Web 应用都是采用前后端分离的方式协同开发完成的。

在这种模式下，用户所看到的页面内容一般都是由客户端渲染完成的，即服务端返回空的 HTML，客户端通过 JavaScript 操作 AJAX 请求数据，动态创建页面内容。

工作流程：




这种模式下，很容易的解决了传统动态网站方式下前后端的耦合问题，实现了比较理想的前后端分离。

「后端」负责提供数据接口，「前端」负责将数据接口提供的数据渲染到页面中。

这样一来，「前端」更为独立，也不再受限制于「后端」，它可以选择任意的技术方案或者框架。

但是这种模式下，也会存在一些明显的不足：

- 首屏渲染问题：因为 HTML 中没有内容，必须等到 JavaScript 加载并执行完成才能呈现页面内容。
- SEO 问题：同样因为 HTML 中没有内容，所以对于目前的搜索引擎爬虫来说，页面中没有任何有用的信息，自然无法提取关键词，进行索引了。

## 解决方案

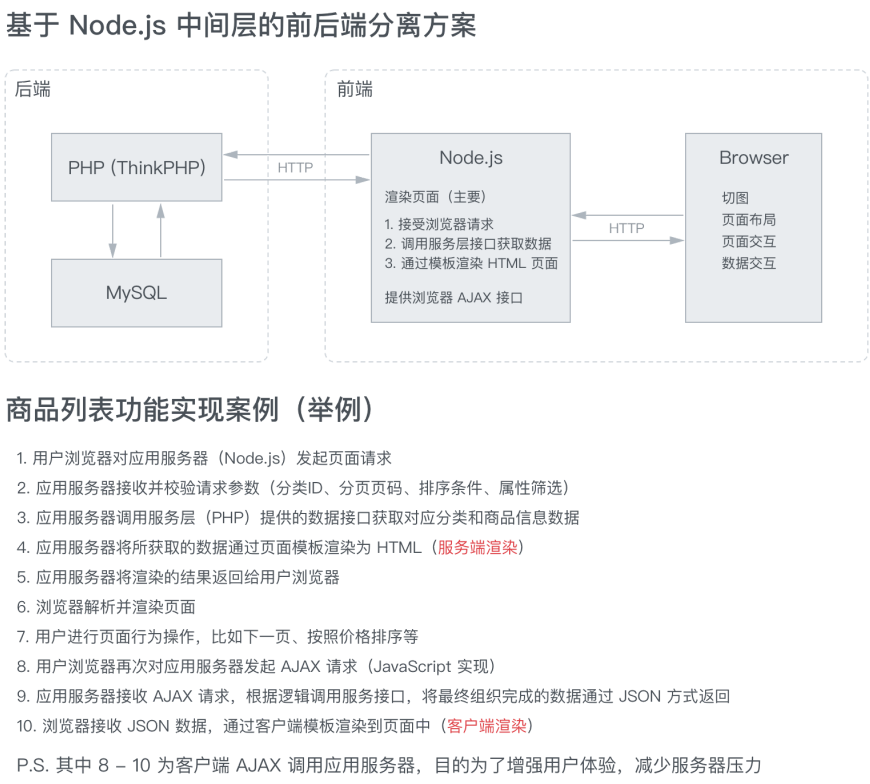
以上两种 Web 应用中的页面渲染方式都存在各自的问题，下面来介绍一些主流的解决办法。

### 一、中间层方案

中间层方案，顾名思义，就是在服务端和客户端中间添加一个中间环节。这个中间环节从职能上属于「前端」范畴，也就是归前端管，从物理角度，这个中间层实际上是一个简单的服务端，也就是我们经常听到的 BFF（Backend for Frontend），它就是一个专门为前端服务的服务端。

下面这张图清楚的描述了这种中间层方案的落地，其中的 Node.js 部分就属于我们这里所提到的 BFF：

颗粒




这种中间层的作用有很多（包括但不限于此）：

- HTML 渲染
- 接口聚合
- 部分功能、WebSocket
- Bigpipe

## 二、服务端渲染

大部分情况下，服务器端渲染（SSR）与 Node.js 中间层是同一个概念。

服务器端渲染（SSR）一般特指，在上文讲到的 Node.js 中间层基础上，加上前端组件化技术在服务器上的渲染，特别是 React 和 Vue.js。

React、Vue.js、Angular 等框架的出现，让前端组件化技术深入人心，但在一些需要首屏快速加载与 SEO 友好的页面就陷入了两难的境地了。

因为前端组件化技术天生就是给客户端渲染用的，而在服务器端需要被渲染成 `html` 文本，这确实不是一件很容易的事，所以服务器端渲染（SSR）就是为了解决这个问题。

好在社区一直在不断的探索中，让前端组件化能够在服务器端渲染，比如 Next.js、Nuxt.js、razzle、react-server、beidou 等。

一般这些框架都会有一些目录结构、书写方式、组件集成、项目构建的要求，自定义属性可能不是很强。

以 Next.js 为例，整个应用中是没有 `html` 文件的，所有的响应 `html` 都是 Node.js 动态渲染的，包括里面的元信息、 `css,js` 路径等。渲染过程中，`Next.js` 会根据路由，将首页所有的组件渲染成 `html`，余下的页面保留原生组件的格式，在客户端渲染。

## 三、静态站点（预渲染）

SSG（）

JAMstack：Gatsby、Gridsome

# 具体实践

## React

## Vue.js

### 实现原理

Virtual DOM

```
var dom = {
    type: 'div',
    attrs: {
```

```
        id: 'root',
        className: 'container'
    },
    children: [
        {
            type: 'h1',
        },
        {
            type: 'h1',
        }
    ]
}

function renderToString (dom) {
    // 返回 dom 对象所对应的 html 字符
}
```

## 注意事项

同构问题

## License

MIT © 汪磊