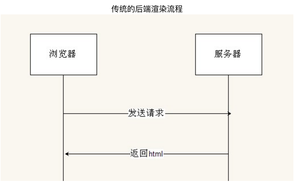
基于HTTP/1.1 chunked编码的不局限语言的前后端整合技术方案



1. 浏览器发送一个HTTP请求到Web服务器。

2. Web服务器解析请求，然后读取数据存储层，生成HTML，并用一个HTTP响应把它发送到客户端。

3. HTTP响应通过互联网传送到浏览器。

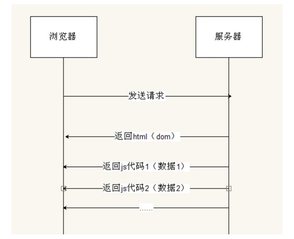
4. 生成DOM，下载并解析CSS，JS。

串型的阻塞模型：

BigPipe的渲染流程

- 渲染页面布局框架（无数据）
- 后端持续性的数据输出（HTTP/1.1 支持的chunked编码）
- 前端通过js来渲染

BigPipe的本质实际上是客户端，服务端并行来提升速度



BigPipe原理：

Transfer-Encoding: chunked（HTTP/1.1 分块传输编码）：

通常，HTTP应答消息中发送的数据是整个发送的，Content-Length消息头字段表示数据的长度。数据的长度很重要，因为客户端需要知道哪里是应答消息的结束，以及后续应答消息的开始。然而，使用分块传输编码，数据分解成一系列数据块，并以一个或多个块发送，这样服务器可以发送数据而不需要预先知道发送内容的总大小。

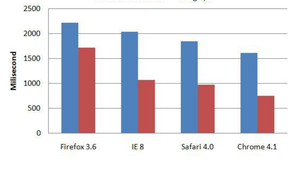
分块加载js（为何不是分块加载html？）：

模块很多的时候，在html前面的模块必须先加载完，后面的才能加载，如果采用js的形式，就可以先把大体的结构先返还，加载若干个loading，然后利用js的方式把数据返回

BigPipe与Ajax对比

Ajax	BigPipe
ajax 核心是XMLHttpRequest	不需要客户端发送请求，当访问量大时可以减少服务器负载，带来性能提升
串型执行	并行执行
seo不友好	seo不影响

Facebook用BigPipe之后的提升



局部刷新

传统网页方式

- 页面跳转等待时间过长，体验太差
- 静态重叠部分还是要被加载一次
- 消耗大多的网络带宽（任何的交互都需要返回一个完整的HTML）

ajax方式局刷

优点：

- 使用hash局刷，增强了体验
- 静态资源可以只加载一次，减少了这一部分的资源

缺点：

- 前端路由对seo不友好
- 大量冗余数据在json中，无形之中浪费了带宽
- 使用前端模板来进行渲染，当用户设备不太行的时候容易造成浏览器崩溃。

pjax 局刷：

原理：

- pushState + ajax
- 不需要前端路由，每个可点击的链接都是真实存在的地址
- 将链接自动转成ajax调用，然后获取到后端渲染好的html回填到页面里

优点：

- 对seo友好
- 减轻浏览器压力

缺点

- 需要支持HTML5 （ie8给跪）
- pushState有时候会整崩浏览器

```
<html>
<body>
<script>
    var total = "";
    for(var i = 0; i< 1000000; i++) {
        total = total + i.toString();
        history.pushState(0, 0, total);
    }
</script>
</body>
</html>
```