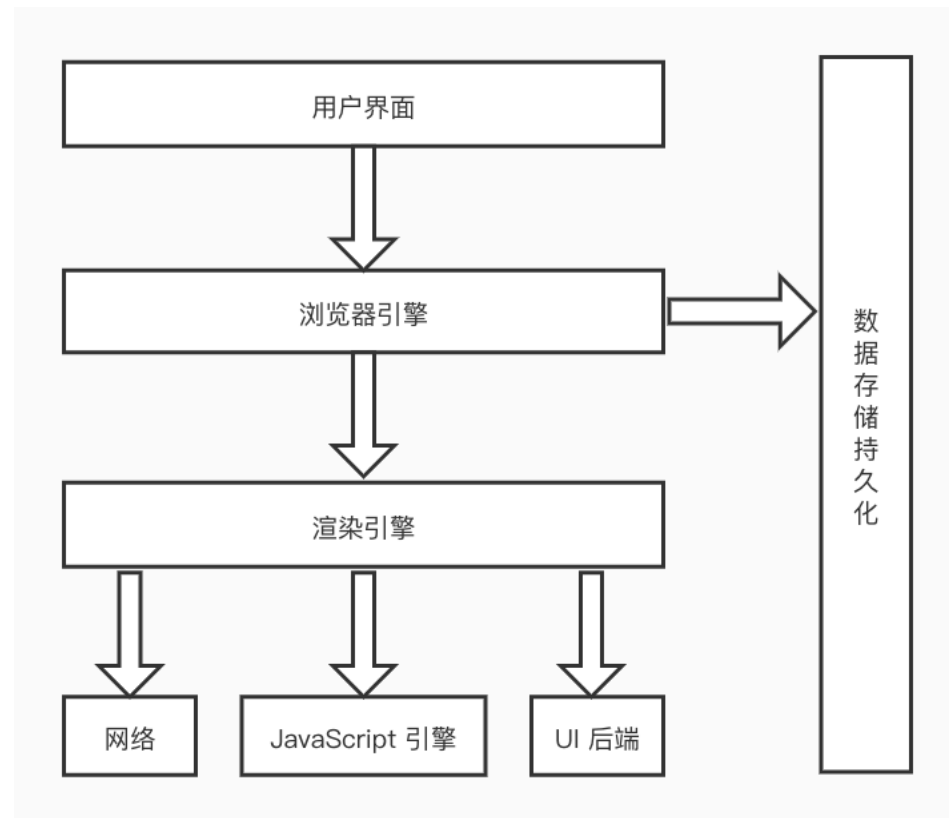


浏览器工作原理（一）

现代浏览器的组成部分

1. **用户界面**：浏览器中可见的地址栏、前进后退按钮、书签、历史记录等用户可操作的功能选项
2. **浏览器引擎**：在用户界面和渲染引擎之间传送指令，或在客户端本地缓存中读写数据等，浏览器中各部分相互通信的核心
3. **渲染引擎**：解析 DOM 文档和 CSS 规则并将内容排版到浏览器中显示有样式的界面
4. **网络**：开启网络线程发送请求或下载资源文件的模块
5. **JavaScript 引擎**：解释和执行 JS 脚本的部分
6. **UI 后端**：绘制基本的浏览器窗口内控件，比如组合选择框，按钮，输入框
5. **持久化数据存储**：涉及 Cookie, LocalStorage 等一些本地存储技术



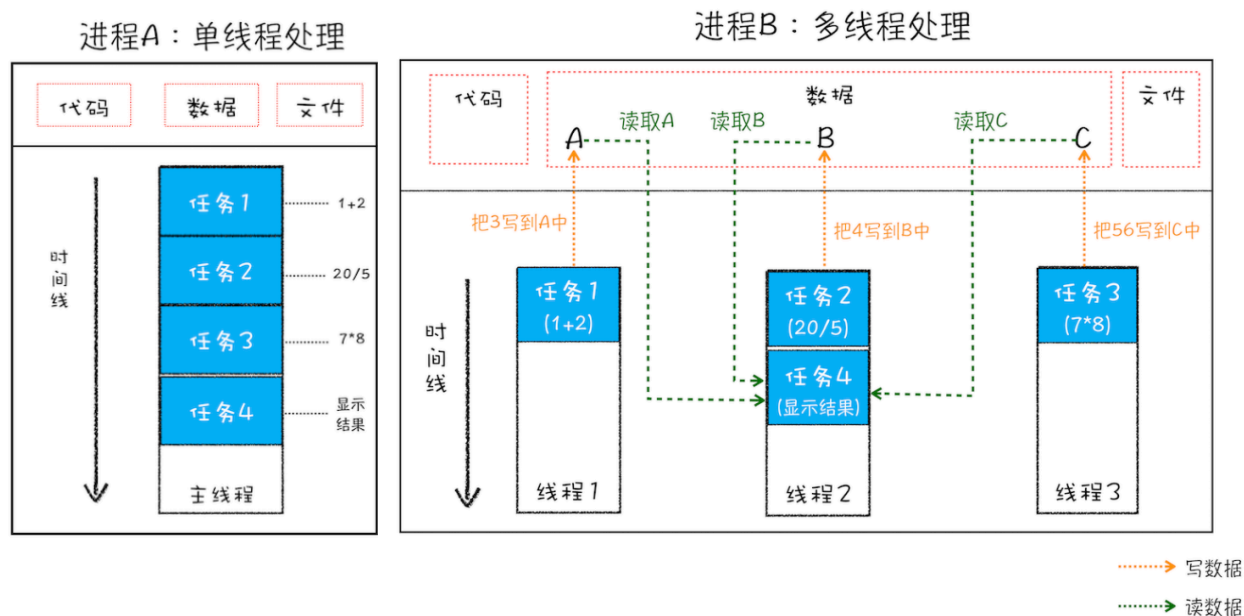
现代浏览器内核

1. Trident: 从1996年开始, IE的呈现引擎就是Trident, 这也是大家俗称的IE内核, 国内的大多数浏览器都有使用IE内核, 或者是IE和Chrome双内核这样的形式来提高性能。
2. Gecko: 原网景公司的人员创办了Mozilla基金会, 这是一个非盈利组织, 2004年推出了自己的浏览器Firefox, 并且以之前的Mosaic内核为基础, 开发了Gecko引擎
3. Webkit: 从Safari推出之时起, 它的渲染引擎就是Webkit, 而 2008 年谷歌公司发布 chrome 浏览器, 采用的 chromium 内核是 fork 了 Webkit。
4. Blink: 内置于 Chrome 浏览器之中, 其实Blink引擎就是是Webkit的分支

现代浏览器与进程

1. **线程**：单线程串行处理任务，多线程并行处理任务
2. **进程**：线程是不能单独存在的，它是由进程来启动和管理的，一个进程就是一个程序的运行实例
3. **进程与线程的关系**：
 - a. 进程中的任意一线程执行出错，都会导致整个进程的崩溃
 - b. 线程之间共享进程中的数据
 - c. 当一个进程关闭之后，操作系统会回收进程所占用的内存
 - d. 进程之间的内容相互隔离

```
1 A = 1+2
2 B = 20/5
3 C = 7*8
```

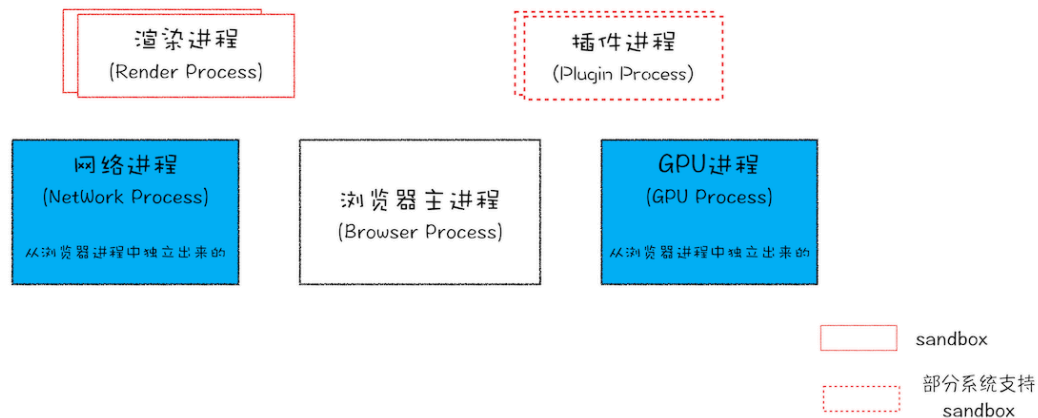
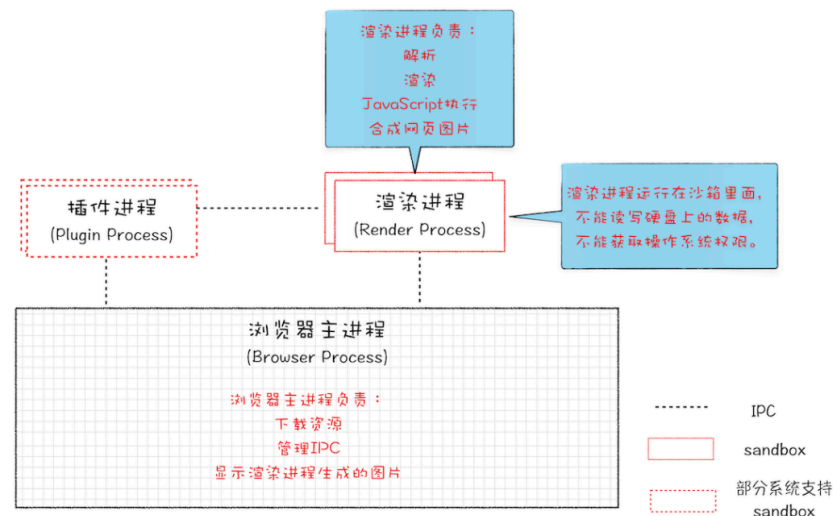
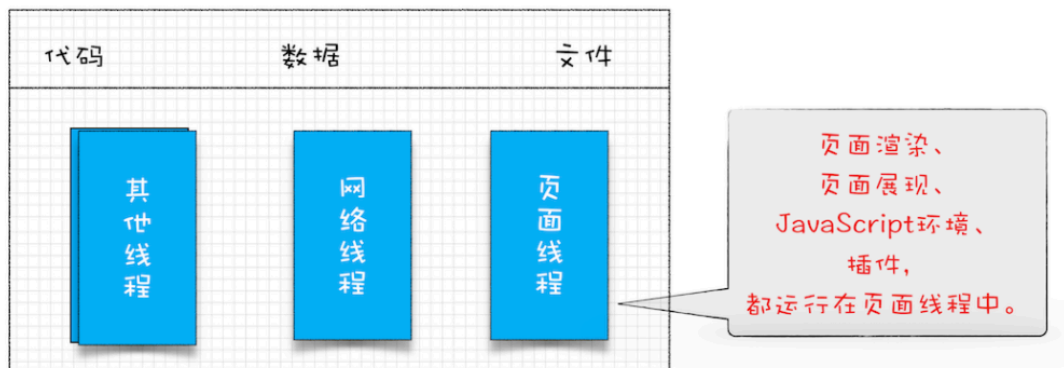


现代浏览器与进程

1. 单进程浏览器：单进程浏览器是指浏览器的所有功能模块都是运行在同一个进程里，导致不稳定、不流畅和不安全的问题产生

2. 多进程浏览器：多进程浏览器是指浏览器的所有功能模块都是运行在同多个进程里，解决不稳定、不流畅和不安全的问题

单进程浏览器



Chrome 进程架构与功能

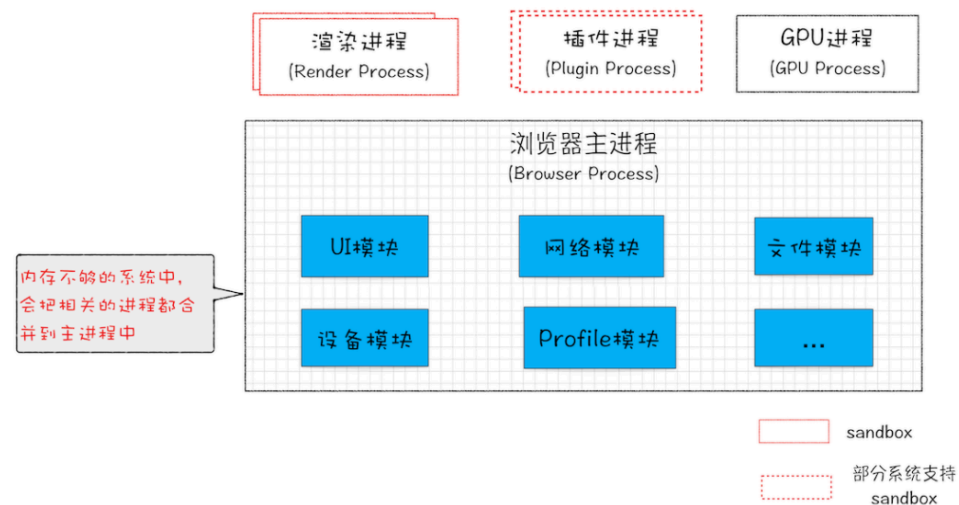
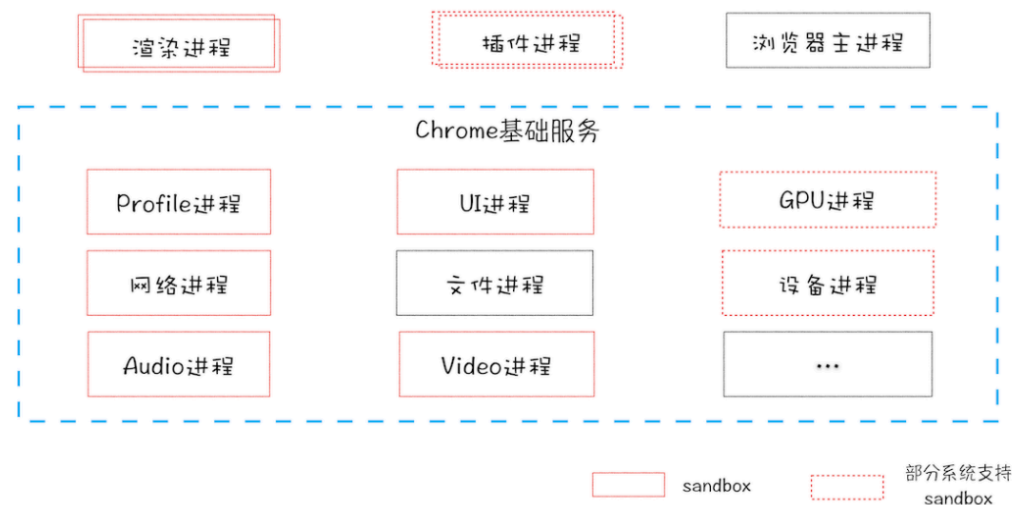
1. **浏览器进程**：主要负责界面显示、用户交互、子进程管理，同时提供存储等功能。
2. **渲染进程**：核心任务是将 HTML、CSS 和 JavaScript 转换为用户可以与之交互的网页，渲染引擎 Blink 和 JavaScript 引擎 V8 都是运行在该进程中，默认情况下，Chrome 会为每个 Tab 标签创建一个渲染进程。出于安全考虑，渲染进程都是运行在沙箱模式下。
3. **GPU进程**：GPU 的使用初衷最早是为了实现 3D CSS 的效果，只是随后网页、Chrome 的 UI 界面都选择采用 GPU 来绘制，这使得 GPU 成为浏览器普遍的需求。
4. **网络进程**：主要负责页面的网络资源加载，之前是作为一个模块运行在浏览器进程里面的，直至最近才独立出来，成为一个单独的进程。
5. **插件进程**：主要是负责插件的运行，因插件易崩溃，所以需要通过插件进程来隔离，以保证插件进程崩溃不会对浏览器和页面造成影响。

浏览器多进程架构带来的问题

1. **更高的资源占用：** 因为每个进程都会包含公共基础结构的副本（如 JavaScript 运行环境），这就意味着浏览器会消耗更多的内存资源。
2. **更复杂的体系架构：** 浏览器各模块之间耦合性高、扩展性差等问题，会导致现在的架构已经很难适应新的需求了。

面向服务的架构

向现代操作系统所采用的“面向服务的架构”方向发展，原来的各种模块会被重构成独立的服务（Service），每个服务（Service）都可以在独立的进程中运行，访问服务（Service）必须使用定义好的接口，通过 IPC 来通信，从而构建一个更内聚、松耦合、易于维护和扩展的系统。



浏览器与网络协议

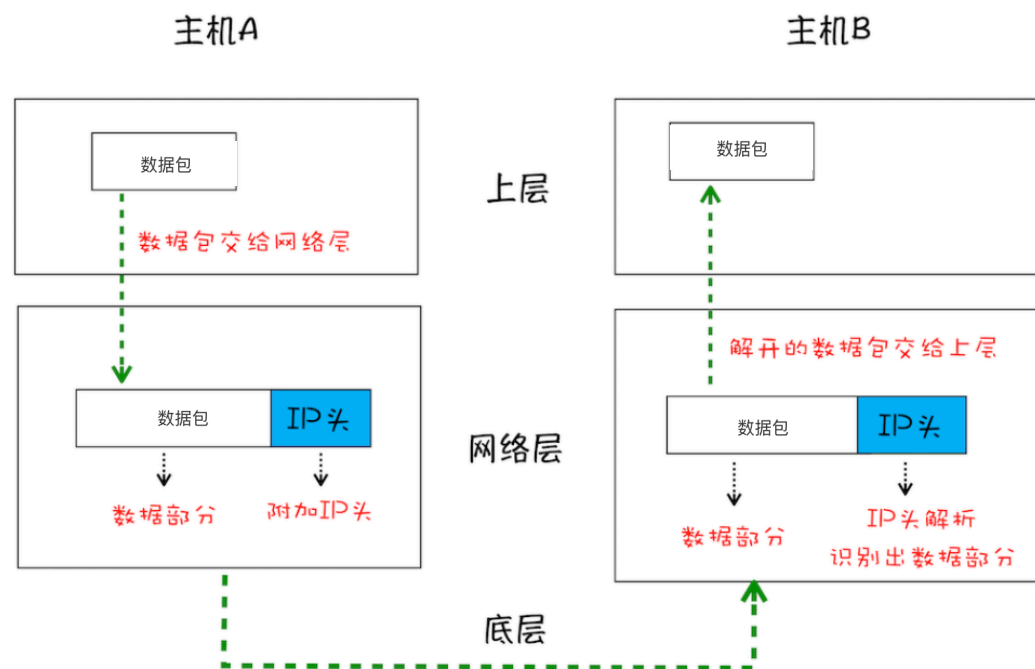
如何保证页面文件能被完整地送达浏览器呢？（TCP/IP）

互联网中的数据是通过数据包来传输的。

1. 数据包如何送达主机。
2. 主机如何将数据包转交给应用
3. 数据如何被完整地送达应用程序

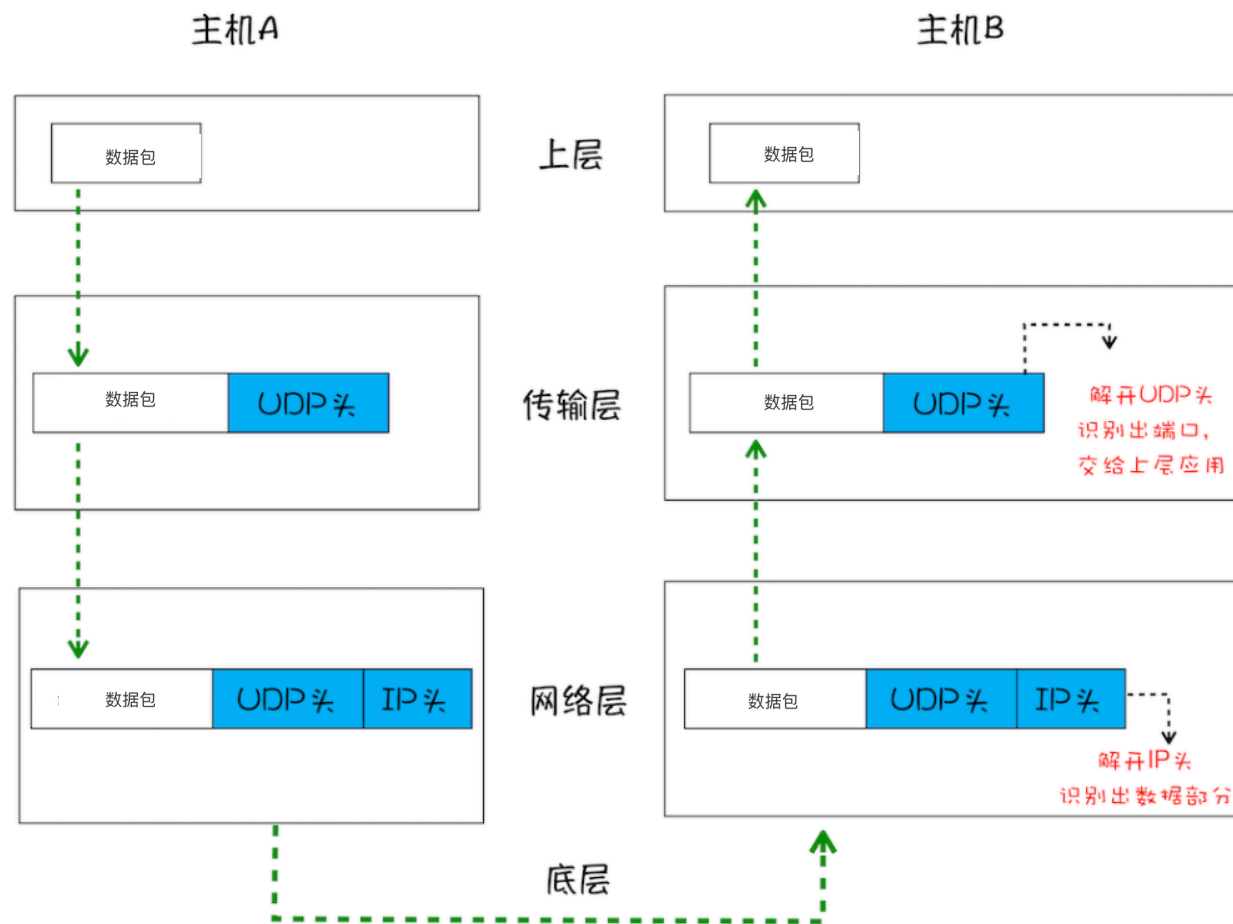
浏览器与网络协议 IP

IP：把数据包送达目的主机。



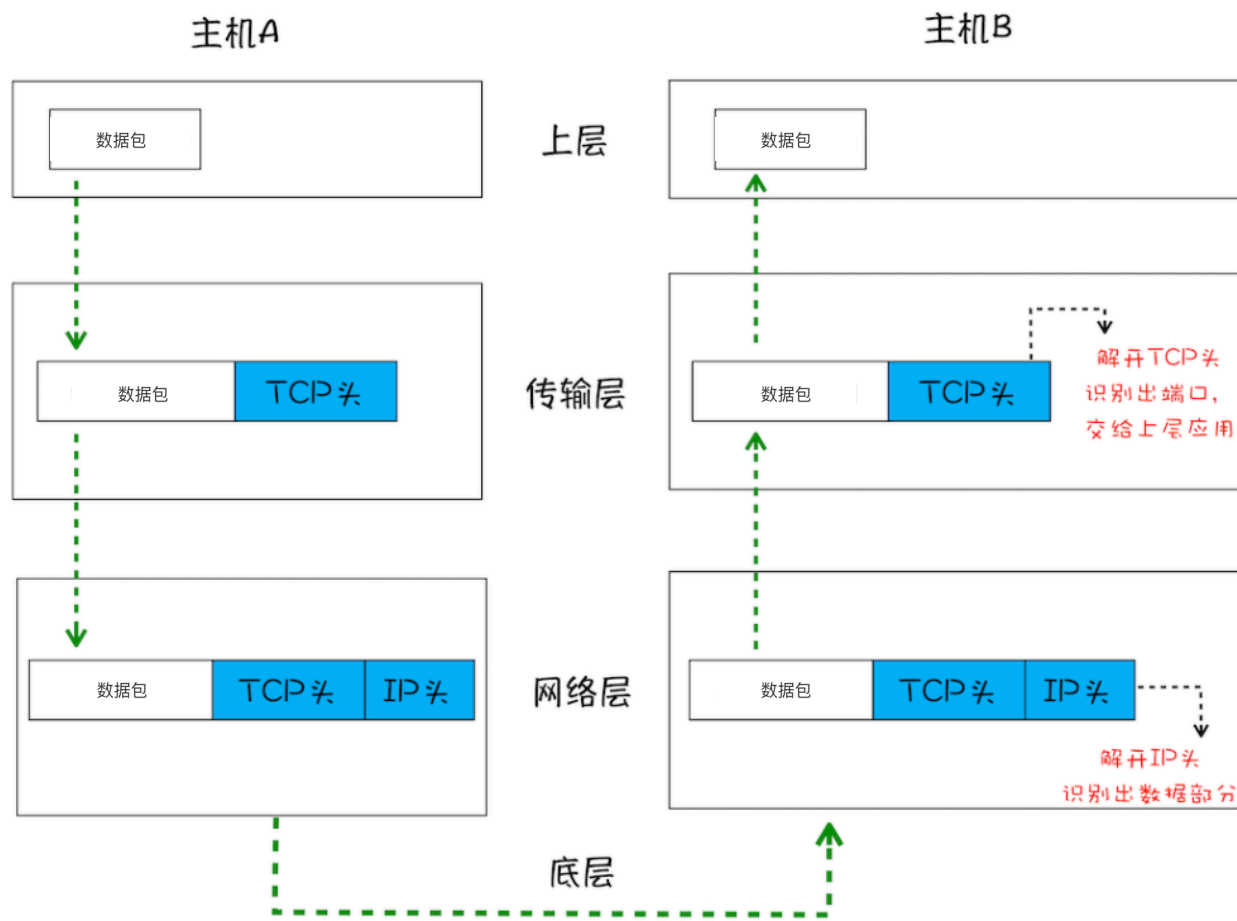
浏览器与网络协议 UDP

UDP：把数据包送达应用程序。



浏览器与网络协议 TCP

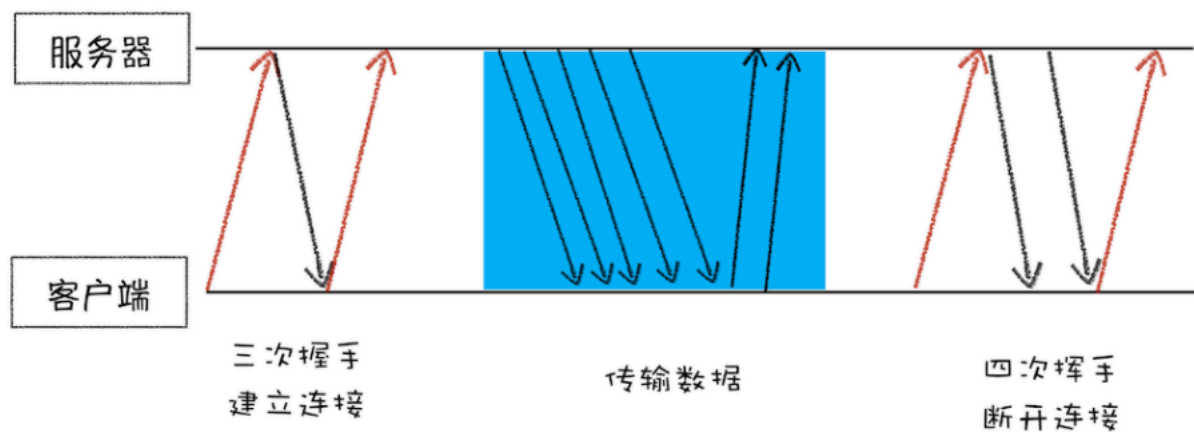
TCP：把数据完整地送达应用程序。



浏览器与网络协议 TCP

完整的 TCP 连接过程

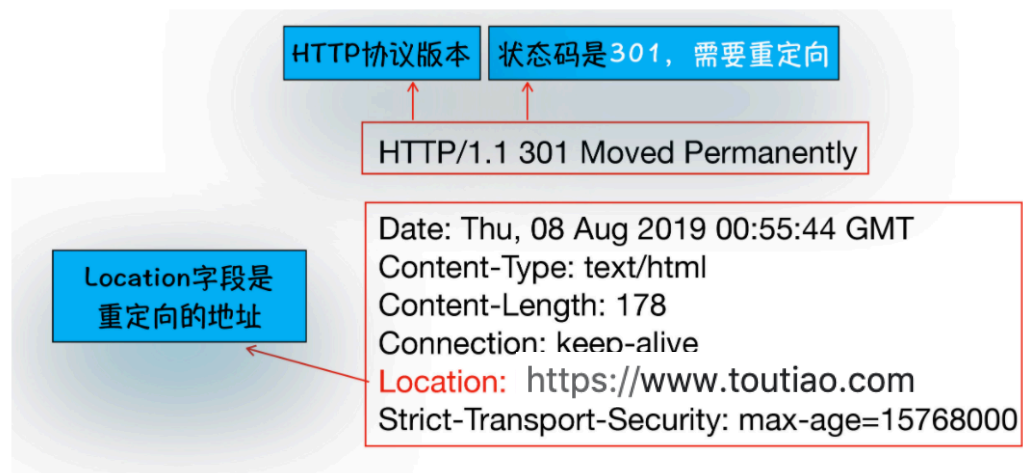
1. 建立连接
2. 传输数据
3. 断开连接



浏览器与网络协议 HTTP

浏览器发起 HTTP 请求流程

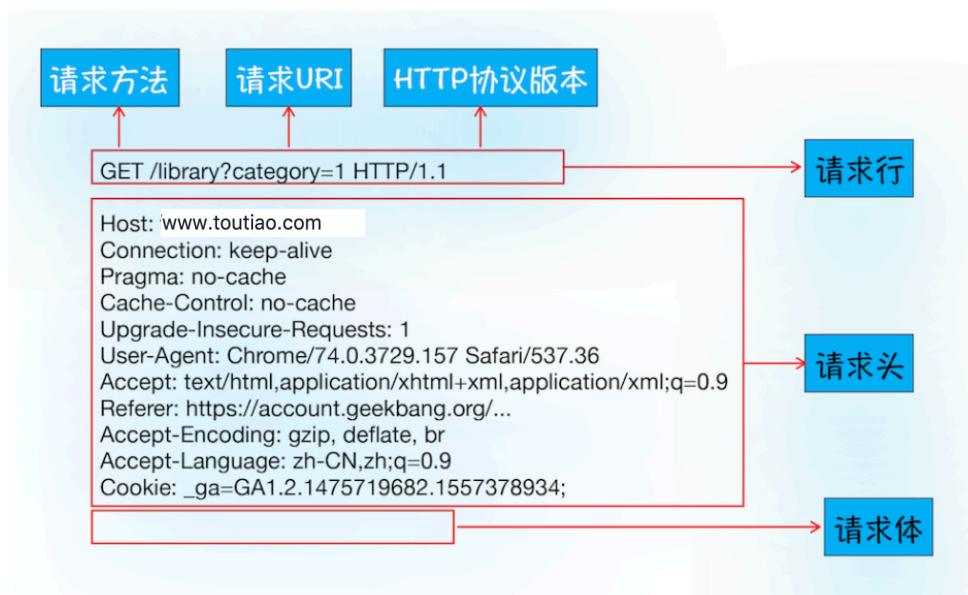
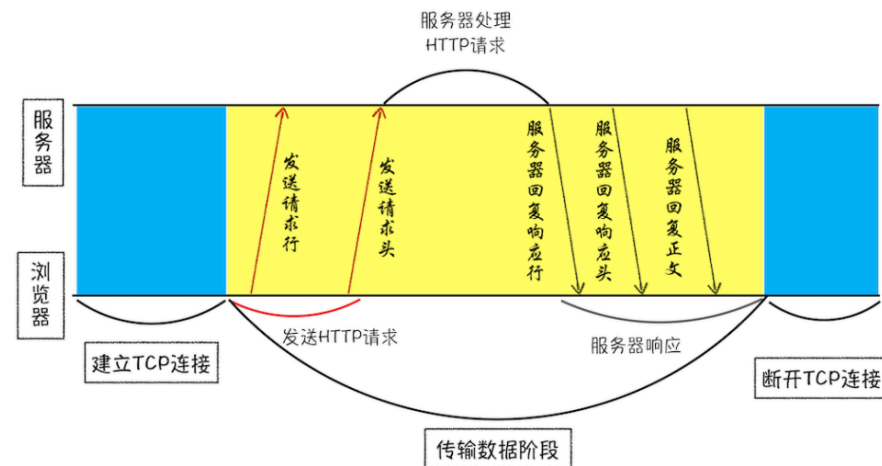
1. 构建请求
2. 查找缓存
3. 准备 IP 地址和端口
4. 等待 TCP 队列
5. 建立 TCP 连接
6. 发送 HTTP 请求



浏览器与网络协议 HTTP

服务器端处理 HTTP 请求流程

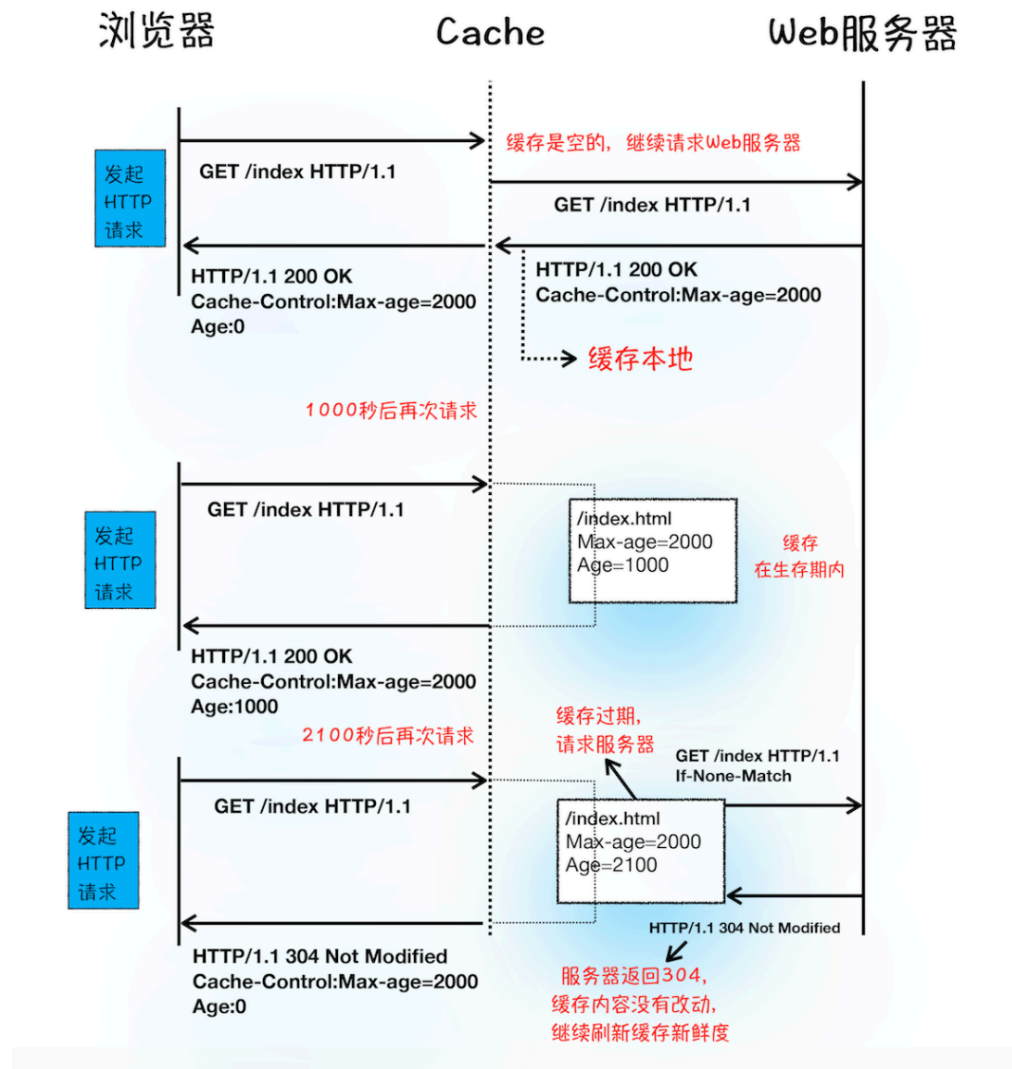
1. 返回请求
2. 断开连接
3. 重定向



浏览器与网络协议 HTTP

为什么很多站点第二次打开速度会很快？

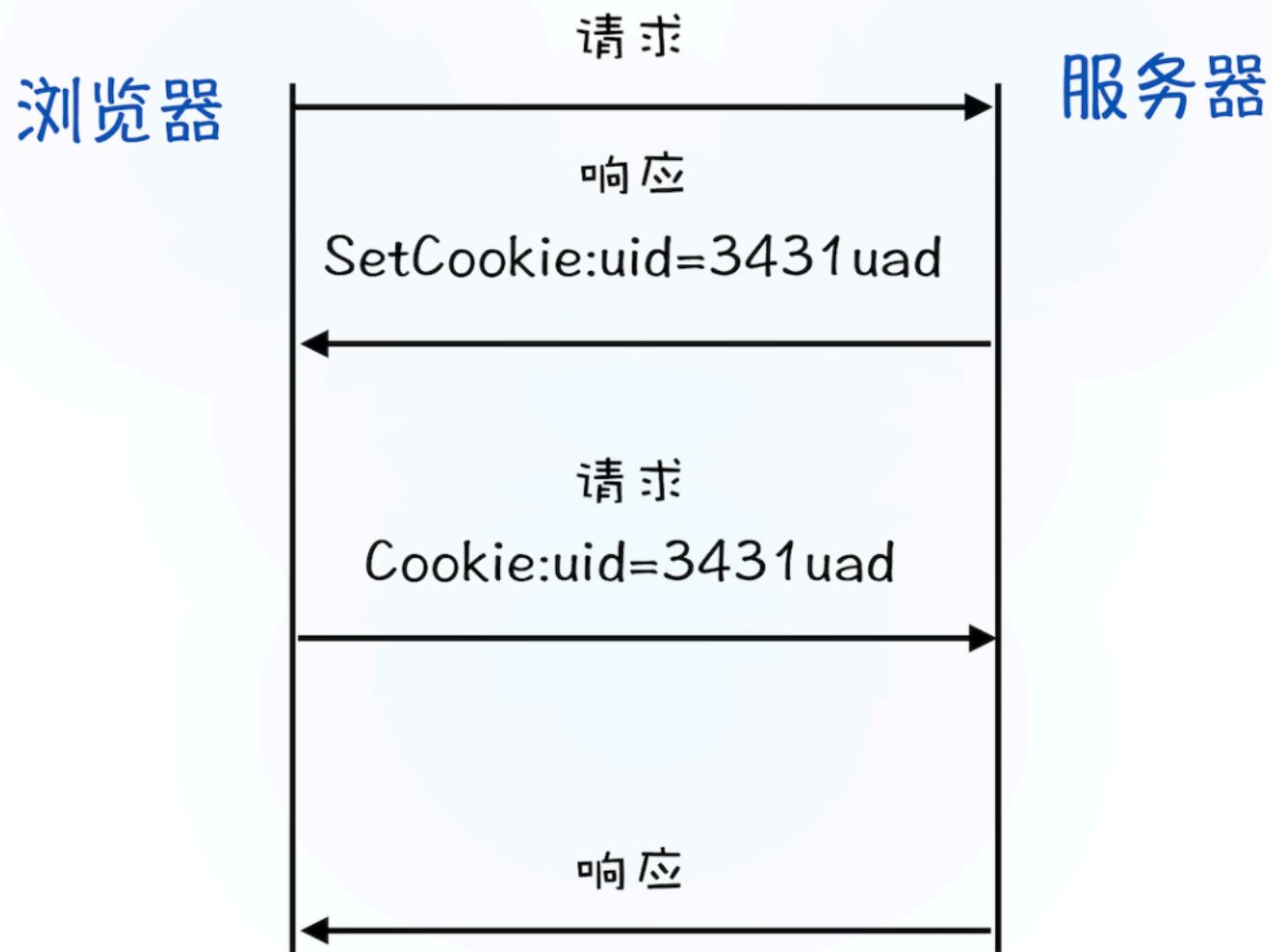
DNS 缓存和页面资源缓存



浏览器与网络协议 HTTP

登录状态是如何保持的？

Cookie



浏览器与网络协议 HTTP

