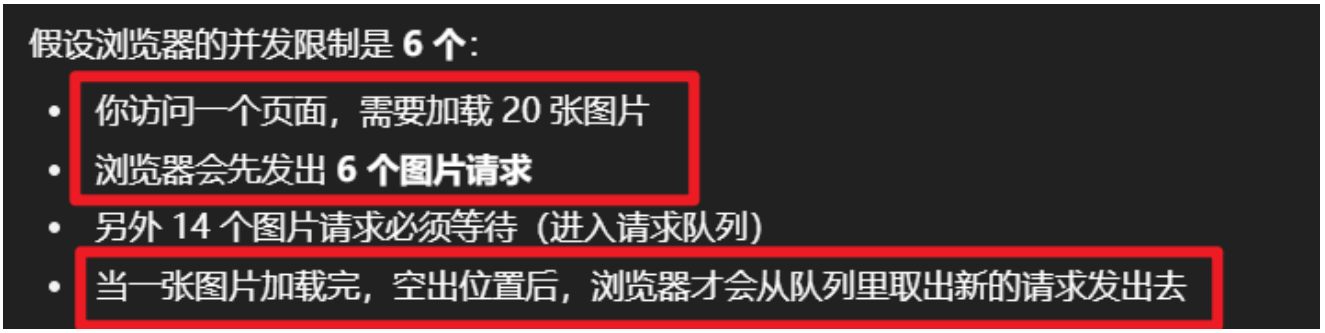
浏览器请求队列&并发限制

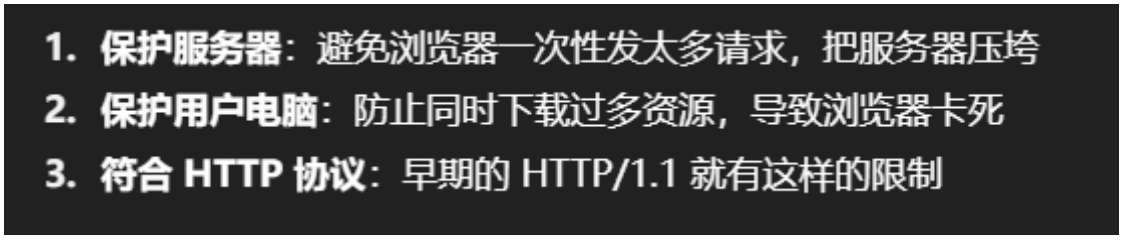
并发限制

在浏览器里：
👉 **并发限制**就是 **同一个域名下，浏览器能同时发起的请求数量的上限。**

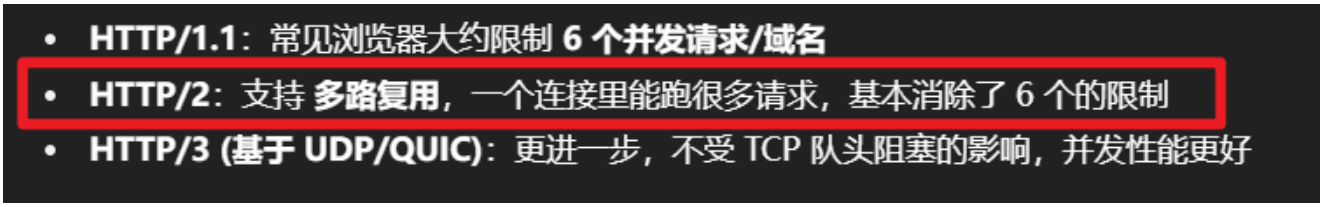
也就是说，**浏览器不会无限制地同时发送所有请求**，它会控制**最多只能“并行”多少个。**



为什么有并发限制？



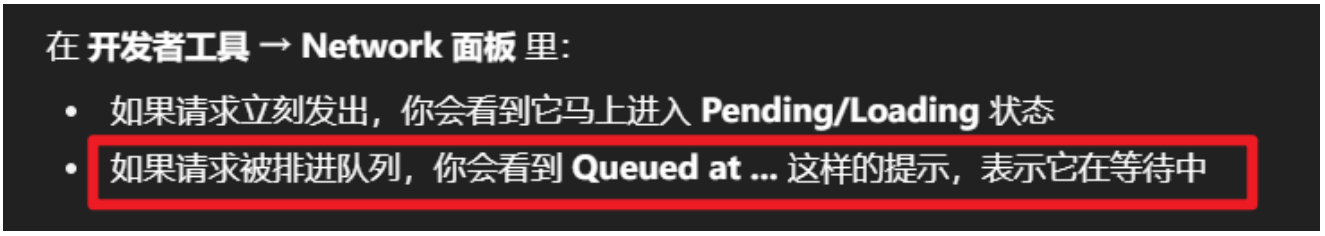
不同协议下的并发限制



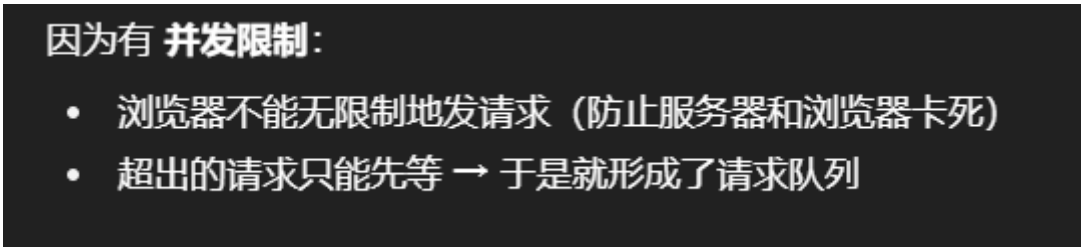
请求队列

请求队列指的是：
当浏览器发出的请求 **超过了并发限制** 时，那些暂时没法发出去的请求，**就会被放到一个“等待区”**，这个等待区就是 **请求队列。**

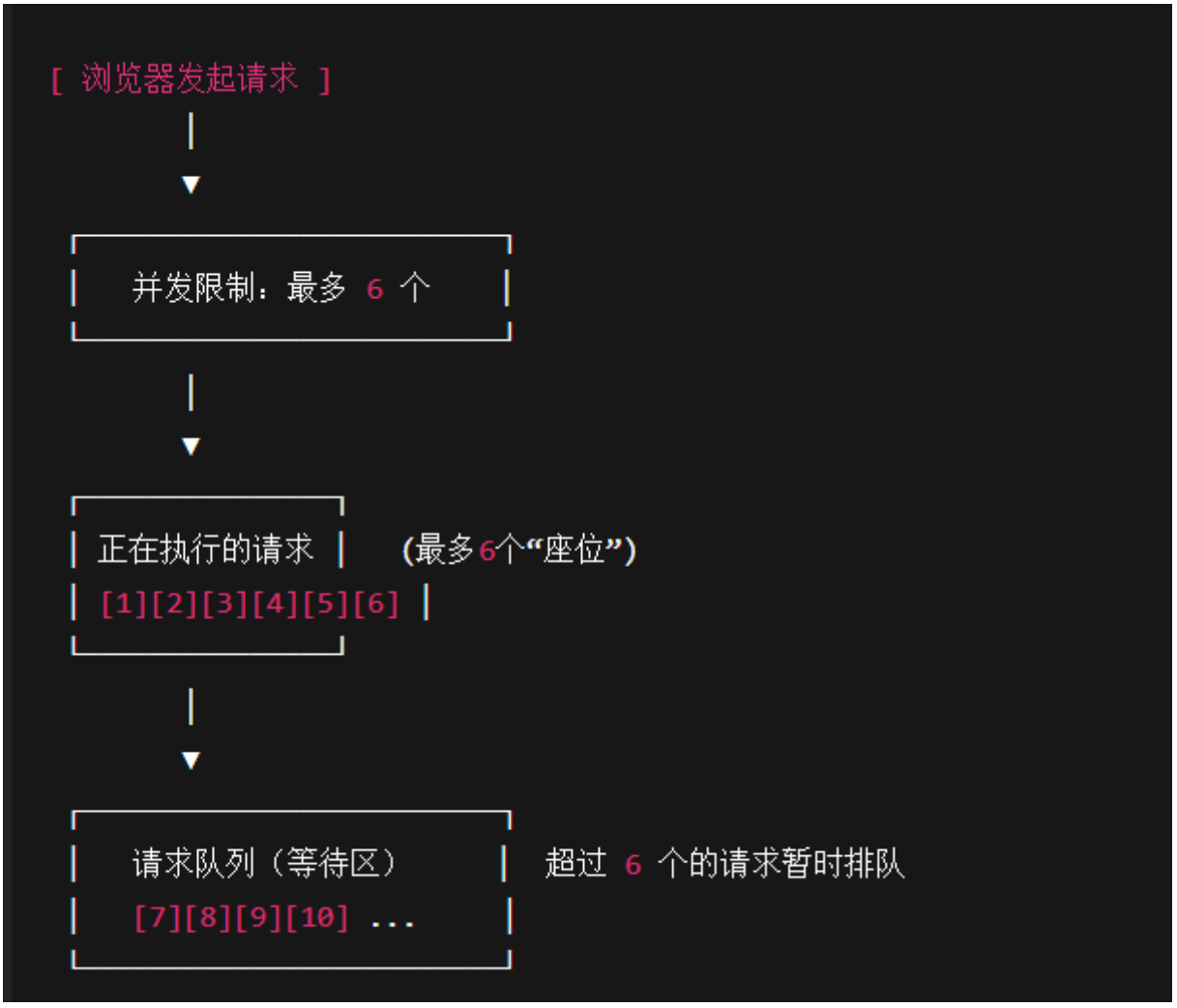
👉 请求队列就是 **排队等待执行的请求集合。**



为什么有请求队列?



示意图



- **[1]~[6]**：前 6 个请求，直接发送（正在下载图片/JS 等资源）
- **[7][8][9]...**：其余请求，因为超过了并发限制，被放进 **请求队列**
- 当 **一个请求完成** → 队列里的第一个请求补上

协议演化

1. HTTP/1.0

- 特点：
  - **无持久连接**：每请求一个资源（HTML、CSS、图片...）都要新建一个 TCP 连接，用完就关掉
- 并发情况：
  - 浏览器没法在一个 TCP 连接里并发多个请求
  - 为了并行加载，浏览器只能开很多 TCP 连接（比如同时开 2~4 个）
- 问题：
  - 连接频繁建立和关闭 → 成本高，速度慢

---

2. HTTP/1.1

- 特点：
  - **持久连接 (Keep-Alive)**：一个 TCP 连接可以复用，多次请求/响应
  - **流水线 (Pipeline)**：理论上可以在一个连接里连续发多个请求，但响应必须按顺序返回
- 并发限制：
  - 浏览器厂商规定：对 **同一域名**，最多同时发 **6~8 个 TCP 连接**
  - 超出的请求就会进入 **请求队列**
- 问题：
  - **队头阻塞 (Head-of-Line Blocking)**：如果前面一个请求卡住，后面所有请求都得等
  - 所以"请求队列"在 HTTP/1.1 里很明显

3. HTTP/2

- 特点：
  - **多路复用 (Multiplexing)**：多个请求可以在一个 TCP 连接里交错传输，不用排队等顺序
  - **二进制分帧**：把数据切成帧，带上流 ID，能区分不同请求的数据
- 并发限制：
  - 不再依赖"6 个 TCP 连接"的限制，一个 TCP 连接就能处理很多并发请求
- 问题：
  - 仍然依赖 **TCP** → 如果底层丢包，整个连接都受影响，还是有 **TCP 队头阻塞**

4. HTTP/3 (基于 QUIC/UDP)

- 特点:
  - **基于 UDP**，不用再受 TCP 顺序传输的限制
  - **内建多路复用**：即使一个请求丢包，其他请求也不会受影响
  - 建立连接更快（TLS + 连接合并）
- 并发限制:
  - 理论上几乎没有"传统的并发限制"问题
  - 一个 QUIC 连接就能处理成百上千个请求
- 优点:
  - 彻底解决了 **队头阻塞**
  - 让请求队列几乎消失

协议	并发处理方式	并发限制	队头阻塞问题
HTTP/1.0	每请求新建 TCP	2~4 连接	每个请求独立，不复用
HTTP/1.1	持久连接 + 流水线	~6/域名	有队头阻塞（排队明显）
HTTP/2	多路复用 (1 个 TCP)	几乎不限	有 TCP 层队头阻塞
HTTP/3	基于 QUIC/UDP 多路复用	几乎不限	无队头阻塞

- **HTTP/1.1** → 受限于 6 个并发，导致请求队列明显
- **HTTP/2** → 用多路复用缓解了问题，但还受 TCP 队头阻塞影响
- **HTTP/3** → 基于 UDP/QUIC，真正解决并发限制和队头阻塞问题

注意点

2. HTTP/2 / HTTP/3 说“没有并发限制”，其实是怎么做的?

这里有个误区：

👉 “没有并发限制” ≠ “无限制乱发请求”

它们的处理方式是：

HTTP/2

- 一个 TCP 连接里可以开很多“流（stream）”
- 每个流都有 ID，请求和响应互不干扰
- 但是！HTTP/2 **协议本身还是允许服务器设置并发上限**（比如 100 或 200 个流）
- 所以实际上还是有“限制”，只是比 HTTP/1.1 的“6 个”要宽松得多

HTTP/3 (QUIC/UDP)

- QUIC 协议允许真正的多路复用，一个包丢了不会卡死所有请求
- 但是！服务器和浏览器之间还是会“协商”一个最大并发流数
- 也就是说，不是无限制，而是**更智能的控制**

因为 **协议和服务器本身会做“流量控制”**：

- **HTTP/2**：
  - 客户端和服务器通过 **SETTINGS 帧** 来协商最大并发流数量
  - 服务端可以告诉浏览器：“别超过 100 个请求”
- **HTTP/3/QUIC**：
  - 内置了 **拥塞控制、流量控制**（借鉴 TCP 的经验）
  - 会根据网络情况动态调整能发多少请求

小结

- **HTTP/1.1**：浏览器硬性限制 6 个 → 避免资源抢占
- **HTTP/2**：允许更多并发（几十甚至上百），但服务器会“协商上限”
- **HTTP/3**：基于 UDP，更自由，但仍然有 **拥塞控制**，不会让服务器爆掉

“没有并发限制”更准确的说法是：
不再是浏览器强制的 6 个，而是浏览器和服务器之间动态协商 + 协议本身的流控机制来决定。

缓存与队列

如果是使用缓存 （强缓存）也需要进入到请求队列吗?

好问题 👍 你问到 **缓存和请求队列的关系**了，这个细节很关键。
答案是：**如果命中强缓存，就不需要进入请求队列。**

1. 强缓存的流程

当浏览器准备加载一个资源时，它会先检查 **本地缓存**：

- **强缓存（200 from disk cache / memory cache）**
  - 浏览器发现缓存未过期
  - 直接从本地拿结果，**不会发 HTTP 请求**
- **协商缓存（304 Not Modified）**
  - 浏览器还是会发请求给服务器问：“资源有更新吗？”
  - 如果没更新，服务器返回 304，浏览器用本地的缓存

2. 请求队列的作用点

请求队列只会限制“需要真正发到服务器的请求”：

- 如果是 **强缓存命中**：
  - 根本不需要发网络请求
  - 所以不会进入请求队列
- 如果是 **协商缓存**：
  - 还是要发 HTTP 请求（哪怕只是 HEAD/GET）
  - 所以要受并发限制，如果超过了，还得进入请求队列