传输优化方法

**压缩Gzip**

主流方法。 Gzip压缩比比较高。

一般不压缩图片。

可以通过nginx进行设置，包括对Gzip压缩文件格式的设置以及大小阈值的设置。

**Keep-alive**

Connection:keep-alive

HTTP1.1。 对TCP连接进行复用，默认开启。

可以减少tcp连接建立所消耗的时间。

超时时间。超时会关掉。

Keep-alive\_timeout:65秒

如果设置为0则不会超时。

Keep-alive\_request:100

一个tcp连接所能发送的请求

**HTTP资源缓存**

提高重复访问时资源加载的速度。

强缓存

Cache-control/Expires

Cache-control:

max-age, s-maxage

public:中间/代理服务器可以做缓存。

Private: 缓存只能在浏览器上

No-store 不管文件有没有发生变化，始终和服务器要最新的文件。

No-cache 相当于强缓存失效，协商缓存有效（304可以触发）。

Status:304

Expires是http1.0

If-None-Match和If-Modified-Since是请求头部

E-tag和Last-modified 是response的头部

E-tag是文件的标识符

**Service Worker**

加速重复访问

离线支持

WorkboxWebpackPlugin

ManifestPlugin

有一个manifest决定什么文件要进行缓存。

React默认启用 service.register()

Service Worker位于客户端和服务器中间

只允许在localhost和https下使用

**HTTP2 性能提升**

只能建立在https下

Protocol h2

主要提升减少了http1.1长连接的数量，多路复用。 即使http1.1可以使用一个连接，但是也存在一个资源阻塞的问题，http1.1即使可以一股脑的发送请求，但是服务端返回response还是需要按照请求顺序，就会出现阻塞。 而多路复用可以同时传输多个资源，互不干扰。

服务端推送。可以直接把资源推到缓存里。

首部压缩。

二进制传输。

**服务端渲染**

加速首屏渲染

更好的SEO

客户端渲染拿到的是一个空的html，然后再进行js渲染

服务端渲染拿到的是一个有基本元素的html

Next.js可以渲染React，Nuxt.js进行服务端渲染。