目录

[HTTP版本与HTTP请求方法 1](#_Toc56621321)

[如何减少http请求 2](#_Toc56621322)

[跨域如何解决 2](#_Toc56621323)

[TCP 的拥塞控制 3](#_Toc56621324)

[https://blog.csdn.net/qq\_18425655/article/details/51955674 3](#_Toc56621325)

[谈谈你做过什么数据处理相关的操作 3](#_Toc56621326)

[讲一下Cookie是Http哪个字段 4](#_Toc56621327)

[断点续传 4](#_Toc56621328)

[数据库的索引 4](#_Toc56621329)

[前端缓存 4](#_Toc56621330)

[三、缓存常用字段 5](#_Toc56621331)

[Redux 6](#_Toc56621332)

[http请求报文 7](#_Toc56621333)

[二面：ajax、axios 等和后台交互的问题 7](#_Toc56621334)

[二分查找、快速排序、二叉树 7](#_Toc56621335)

[写一个正则匹配不同的文件类型 7](#_Toc56621336)

[一个搜索组件，输入框搜索的时候从后台加载选项然后显示 7](#_Toc56621337)

# HTTP版本与HTTP请求方法

超文本传输协议（HTTP, HyperText Transfer Protocol）是一种无状态的协议，它位于[OSI七层模型](http://itbilu.com/nodejs/core/VkcdcFq9.html" \l "OSI" \t "https://itbilu.com/other/relate/_blank)的传输层。HTTP客户端会根据需要构建合适的HTTP请求方法，而HTTP服务器会根据不同的HTTP请求方法做出不同的响应。

1. HTTP版本与HTTP请求方法

在HTTP的发展过程中，出现了很多HTTP版本，其中的大部分协议都是向下兼容的。在进行HTTP请求时，客户端在请求时会告诉服务器它采用的协议版本号，而服务器则会在使用相同或者更早的协议版本进行响应。

HTTP/0.9

现已过时。只有GET一种请求方法，在HTTP通讯也没有指定版本号，也不支持请求头信息。客户端向服务器传递信息的能力非常有限。HTTP/0.9的请求只有如下一行：

HTTP/1.0

这个版本是第一个在HTTP通讯中指定版本号的协议版本，HTTP/1.0至今仍被广泛采用，特别是在代理服务器中。

支持：GET、POST、HEAD三种HTTP请求方法。

HTTP/1.1 8种方法

HTTP/1.1是当前正在使用的版本。该版本默认采用持久连接，并能很好地配合代理服务器工作。还支持以管道方式同时发送多个请求，以便降低线路负载，提高传输速度。

HTTP/1.1新增了：OPTIONS、PUT、DELETE、TRACE、CONNECT五种HTTP请求方法。

HTTP/2

HTTP/2通过支持请求与相应的多路重用来减少延迟，通过压缩HTTP头字段将协议开销降到最低，同时增加了对请求优先级和服务器端推送的支持。新一代的 HTTP/2 协议的支持需以 HTTPS 为基础。

HTTP/1.1之后增加的方法

在HTTP/1.1标准制定之后，又陆续扩展了一些方法。其中使用中较多的是 PATCH 方法：

PATCH

PATCH方法出现的较晚，它在2010年的[RFC 5789](http://tools.ietf.org/html/rfc5789" \t "https://itbilu.com/other/relate/_blank)标准中被定义。PATCH请求与PUT请求类似，同样用于资源的更新。二者有以下两点不同：

但PATCH一般用于资源的部分更新，而PUT一般用于资源的整体更新。

当资源不存在时，PATCH会创建一个新的资源，而PUT只会对已在资源进行更新。

详细介绍一下某个请求

get是从服务器上获取数据，post是向服务器传送数据。

（1）在客户端，Get方式通过URL提交数据，数据在URL中可以看到；POST方式，数据放置在HTML HEADER内提交。

（2）对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。

（2）GET方式提交的数据最多只能有1024字节，而POST则没有此限制。

（3）安全性问题。使用 Get 的时候，参数会显示在地址栏上，，如果这些数据是中文数据而且是非敏感数据，那么使用 get；如果用户输入的数据不是中文字符而且包含敏感数据，那么还是使用 post为好。

# 如何减少http请求

# 跨域如何解决

<https://segmentfault.com/a/1190000011145364>

跨域是指一个域下的文档或脚本试图去请求另一个域下的资源，这里跨域是广义的。

广义的跨域：

1.) 资源跳转： A链接、重定向、表单提交

2.) 资源嵌入： <link>、<script>、<img>、<frame>等dom标签，还有样式中background:url()、@font-face()等文件外链

3.) 脚本请求： js发起的ajax请求、dom和js对象的跨域操作等

通常所说的跨域是狭义的，是由浏览器相似策略限制的一类请求场景。

**什么是同源策略？**  
同源策略/ SOP（Same origin policy）是一种约定，由Netscape公司1995年发布。，它是浏览器最核心也最基本的安全功能，如果有了同源策略，所谓相似是指“协议+域名+端口”三者相同，甚至两个不同的域名指向同一个IP地址，也非同源。

同源策略限制以下几种行为：

1.) Cookie、LocalStorage 和 IndexDB 无法读取

2.) DOM 和 Js对象无法获得

3.) AJAX 请求不能发送

1. JSONP：通过动态创建script，再请求一个带参网址实现跨域通信。
2. document.domain + iframe跨域：两个页面都通过js强制设置document.domain为基础主域，就实现了同域。
3. location.hash + iframe跨域：a欲与b跨域相互通信，通过中间页c来实现。 三个页面，不同域之间利用iframe的location.hash传值，相同域之间直接js访问来通信。
4. window.name + iframe跨域：通过iframe的src属性由外域转向本地域，跨域数据即由iframe的window.name从外域传递到本地域。
5. postMessage跨域：可以跨域操作的window属性之一。（HTML5）
6. CORS：服务端设置Access-Control-Allow-Origin即可，前端无须设置，若要带cookie请求，前后端都需要设置。
7. 代理跨域：启一个代理服务器，实现数据的转发
8. WebSocket协议跨域

# TCP 的拥塞控制

# <https://blog.csdn.net/qq_18425655/article/details/51955674>

一个混this和原型的程序 问输出

流动布局实现三个div自适应

手写正则表达式判断电话号码

如何实现渲染后台传来的html片段

vue3.0新增了什么

手写快排（有延伸）

手写斐波那契 递归 动态规划（有延伸）

# 谈谈你做过什么数据处理相关的操作

算法 找数组里总和大于等于目标值的连续数字合和

算法 promise封装原生ajax

算法 找二叉树从根节点到叶节点所有路径的总连续和

问项目难点

# 讲一下Cookie是Http哪个字段

# 断点续传

# 数据库的索引

<https://juejin.im/post/5e6509fd518825490d1267eb>

Hash、B+

**1、**：

三、缓存常用字段

1、http1.0时期的缓存方案

注意：

（1）如果使用了Pragma: 'no-cache'的话，再设置Expires或者Cache-Control，就没有用了，说明Pragma的权值比后两者高。

 （2）如果设置了Expires之后，客户端在需要请求数据的时候，首先会对比当前系统时间和这个Expires时间，如果没有超过Expires时间，则直接读取本地磁盘中的缓存数据，不发送请求。

2、http1.1 时期的缓存方案

2.1、Cache-Control 字段

2.1.1、Cache-Control 作为请求头字段

（1）Cache-Control: no-cache

使用no-cache指令的目的是为了防止从缓存中返回过期的资源。 客户端发送的请求中如果包含 no-cache 指令，则表示客户端将不会接收缓存的资源。每次请求都是从服务器获取资源，返回304。

（2）Cache-Control: no-store

使用no-store 指令表示请求的资源不会被缓存，下次任何其它请求获取该资源，还是会从服务器获取，返回 200，即资源本身。

2.1.2、Cache-Control 作为响应头字段

Cache-Control: public

当指定使用 public 指令时，则明确表明其他用户也可利用缓存。

 Cache-Control: private

当指定 private 指令后，响应只以特定的用户作为对象，这与 public 指令的行为相反。 缓存服务器会对该特定用户提供资源缓存的服务，对于其他用户发送 过来的请求，代理服务器则不会返回缓存。

 Cache-Control: no-cache

如果服务器返回的响应中包含 no-cache 指令，每次客户端请求，必需先向服务器确认其有效性，如果资源没有更改，则返回304.

Cache-Control: no-store

不对响应的资源进行缓存，即用户下次请求还是返回 200，返回资源本身。

Cache-Control: max-age=604800（单位：秒）

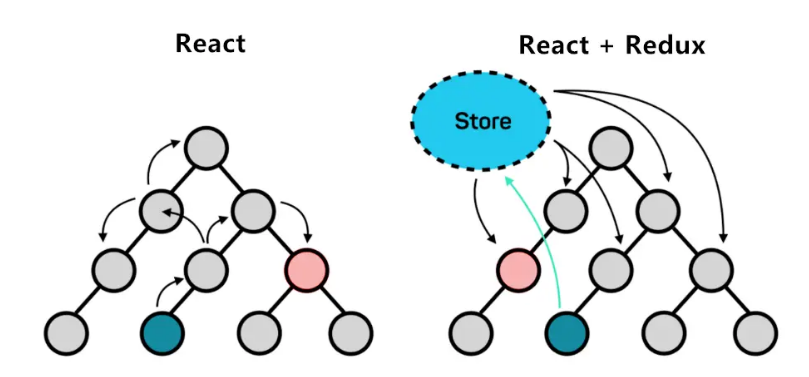
资源缓存在本地浏览器的时间，如果超过该时间，则重新向服务器获取。

# Redux

问题来了，React 中一个组件里面维护数据只需要 state 和 setState 就可以轻松搞定。假如多个组件都需要维护这一份数据怎么办呢？

Redux 有一个非常核心的部分就是 Store，Store 中管理的数据独立于 React 组件之外，如果 React 某个组件中的某个数据在某个时刻改变了（可以称之为状态改变了），就可以直接更改这个 Store 中管理的数据，这样其他组件想要拿到此时的数据直接拿就行了，不需要传来传去。

Store 通常要和 Reducer 来配合使用，Store 存数据，Reducer 是个纯函数，它接收并更新数据。



webpack用过么？原理明白吗？为什么 可以 import css文件

ES6、7、9

react开发用的是hook么？ 不是，那说一下生命周期

react-router如何工作？如果输入 '/a/b/c/index.html'应该怎么处理

# ****http请求报文****

# 二面：ajax、axios 等和后台交互的问题

# 二分查找、快速排序、二叉树

# 写一个正则匹配不同的文件类型

# 一个搜索组件，输入框搜索的时候从后台加载选项然后显示