目录

[HTTP版本与HTTP请求方法 1](#_Toc45062613)

[前端安全 2](#_Toc45062614)

[如何减少http请求 3](#_Toc45062615)

[跨域如何解决 3](#_Toc45062616)

[TCP 的拥塞控制 4](#_Toc45062617)

[https://blog.csdn.net/qq\_18425655/article/details/51955674 4](#_Toc45062618)

[谈谈你做过什么数据处理相关的操作 4](#_Toc45062619)

[讲一下Cookie是Http哪个字段 5](#_Toc45062620)

[断点续传 5](#_Toc45062621)

[数据库的索引 5](#_Toc45062622)

[前端缓存 5](#_Toc45062623)

[三、缓存常用字段 6](#_Toc45062624)

[Redux 7](#_Toc45062625)

[http请求报文 8](#_Toc45062626)

[二面：ajax、axios 等和后台交互的问题 8](#_Toc45062627)

[二分查找、快速排序、二叉树、数组扁平化、深度/广度优先遍历 8](#_Toc45062628)

[写一个正则匹配不同的文件类型 8](#_Toc45062629)

[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 8](#_Toc45062630)

# 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。

# 前端安全

<https://juejin.im/entry/598d6eb46fb9a03c3a25d2c1>

一、XSS攻击与防御

跨站脚本攻击，是说攻击者通过注入恶意的脚本，在用户浏览网页的时候进行攻击，比如获取cookie，或者其他用户身份信息，可以分为存储型和反射型，存储型是攻击者输入一些数据并且存储到了数据库中，其他浏览者看到的时候进行攻击，反射型的话不存储在数据库中，往往表现为将攻击代码放在url地址的请求参数中，防御的话为cookie设置httpOnly属性，对用户的输入进行检查，进行特殊字符过滤

二、CSRF攻击

跨站请求伪造，可以理解为攻击者盗用了用户的身份，以用户的名义发送了恶意请求，比如用户登录了一个网站后，立刻在另一个ｔａｂ页面访问量攻击者用来制造攻击的网站，这个网站要求访问刚刚登陆的网站，并发送了一个恶意请求，这时候CSRF就产生了，比如这个制造攻击的网站使用一张图片，但是这种图片的链接却是可以修改数据库的，这时候攻击者就可以以用户的名义操作这个数据库，防御方式的话：使用验证码，检查https头部的refer，使用token

防御CSRF 攻击主要有三种策略：验证 HTTP Referer 字段；在请求地址中添加 token 并验证；在 HTTP 头中自定义属性并验证。

三、HTTP劫持与对策

当我们访问页面的时候，运营商在页面的HTML代码中，插入弹窗、广告等HTML代码，来获取相应的利益。

针对这种情况，最好的解决方式也就是使用HTTPS，加密过后，他们就没法插入广告代码了。

那么对于还没有升级的情况，我们可以努力让影响降到最低。

四、界面操作劫持

五、防御手段

**上面列举的例子都不具备实际攻击作用**，因为浏览器厂商，W3C等已经做了很多安全工作，让我们的页面可以安稳的运行起来。但道高一尺魔高一丈，我们要合理运用防护手段，才能让页面不被攻击。

1、HTTP响应头，在响应头可以通过这些字段来提高安全性

* X-Frame-Options 禁止页面被加载进iframe中
* X-XSS-Protection 对于反射型XSS进行一些防御
* X-Content-Security-Policy 这个就比较复杂了，可选项很多，用来设置允许的的资源来源以及对脚本执行环境的控制等。

2、使用HTTPS、使用HTTP ONLY的cookie。cookie的secure字段设置为true

3、GET请求与POST请求，要严格遵守规范，不要混用，不要将一些危险的提交使用JSONP完成。

# 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

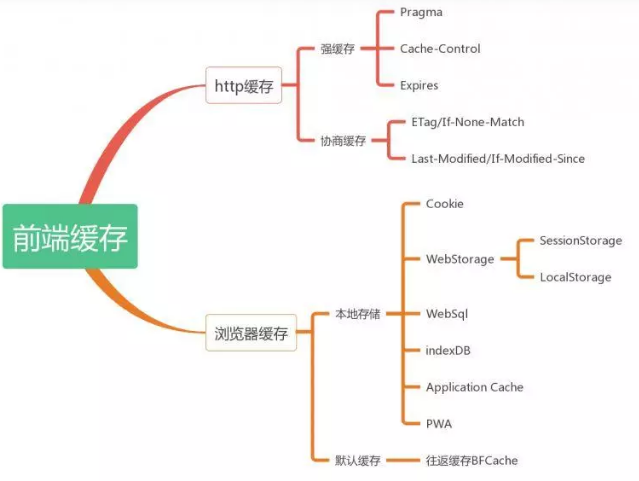
# 数据库的索引

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

Hash、B+

# 前端缓存

前端缓存可分为两大类：http缓存和浏览器缓存



#### 一、什么是HTTP缓存 ？

http缓存指的是: 当客户端向服务器请求资源时，会先抵达浏览器缓存，如果浏览器有“要请求资源”的副本，就可以直接从浏览器缓存中提取而不是从原始服务器中提取这个资源。

常见的http缓存只能缓存get请求响应的资源，对于其他类型的响应则无能为力，所以后续说的请求缓存都是指GET请求。

http缓存都是从第二次请求开始的。第一次请求资源时，服务器返回资源，并在respone header头中回传资源的缓存参数；第二次请求时，浏览器判断这些请求参数，命中强缓存就直接200，否则就把请求参数加到request header头中传给服务器，看是否命中协商缓存，命中则返回304，否则服务器会返回新的资源。

**1、http缓存的分类：**  
根据是否需要重新向服务器发起请求来分类，可分为(强制缓存，协商缓存) 根据是否可以被单个或者多个用户使用来分类，可分为(私有缓存，共享缓存) 强制缓存如果生效，不需要再和服务器发生交互，而协商缓存不管是否生效，都需要与服务端发生交互。下面是强制缓存和协商缓存的一些对比：

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）

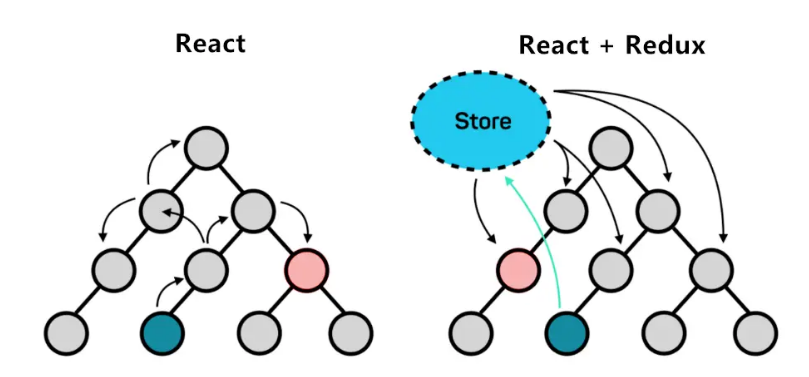
资源缓存在本地浏览器的时间，如果超过该时间，则重新向服务器获取。

# 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。



js有哪些数据类型，举几个object

 JavaScript中有6种数据类型：number、string、boolean、undefined、null、Object。 其中对象类型包括：数组（Array）、函数（Function）、还有两个特殊的对象：正则（RegExp）和日期（Date）

webpack用过么？原理明白吗？为什么 可以 import css文件 --瞎猜

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、数组扁平化、深度/广度优先遍历

# 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