HTTP协议和AJAX

HTTP协议

HTTP协议简介

超文本传输协议（英文：HyperText Transfer Protocol，缩写：HTTP）是一种用于分布式、协作式和超媒体信息系统的应用层协议，是万维网的数据通信的基础。

HTTP协议基本格式：

```JavaScript
协议://服务器IP:[端口]/路径/[? 查询]
```

1999年6月公布的RFC 2616定义了HTTP协议中现今广泛使用的HTTP 1.1。HTTP/2标准于2015年5月以RFC 7540正式发表，取代HTTP 1.1成为HTTP的实现标准。

HTTP协议概述

HTTP是一个客户端与服务器端请求应答的标准（TCP）。可通过浏览器或者其它的工具，发起一个HTTP请求到服务器上指定端口（默认：80）。

HTTP协议广泛应用于TCP/IP协议之上，但并非必须使用TCP/IP协议。HTTP假定下层协议可靠传输，因此能够保证这一要求的协议都可被使用。

HTTP协议数据是明文传输，不安全，可基于HTTPS进行加密处理，通常使用SSL/TLS协议进行加密。

基本使用流程：客户端发起HTTP请求，创建到指定服务器（端口）的TCP连接。服务器监听对应端口（默认80端口），接收并处理请求，返回状态码（比如：“HTTP/1.1 200 OK”）、内容、错误消息或其他信息。

HTTP协议特性

HTTP是无状态的

HTTP协议是无状态(stateless)协议。每次请求都是相互独立的，不会对请求或响应做持久化处理。

好处：可以更快地处理大量事务、确保协议的可伸缩性。

针对业务需要，可引入了Cookie（HTTP 1.1）和Session技术，用于管理状态。

无连接

HTTP 1.0每次连接只处理一个请求。服务器处理完客户的请求，收到客户的应答后，即断开连接。目的是节省传输时间、提高并发性能。

HTTP 1.1 会等待一段时间，如无后续请求则断开，否则继续使用。目的是提高效率，减少短时间内建立连接的次数。

基于TCP协议

HTTP协议的目标是规定客户端和服务端数据传输的格式和数据交互行为，并不负责数据传输的细节。大多数底层是基于TCP实现。现在使用的版本当中是默认持久连接的，也就是多次HTTP请求使用一个TCP连接。

HTTP工作流程

客户端向服务器发送请求报文，请求报文包含请求的方法、URL、协议版本、请求头部和请求数据。服务器响应请求结果，响应内容包括协议的版本、成功或者错误代码、服务器信息、响应头部和响应数据。

HTTP 请求/响应的基本步骤：

步骤一：客户端（比如浏览器）连接到Web服务器（默认80端口），并建立TCP连接。

步骤二：基于TCP，发起HTTP请求；

步骤三：服务接受请求并返回响应报文；

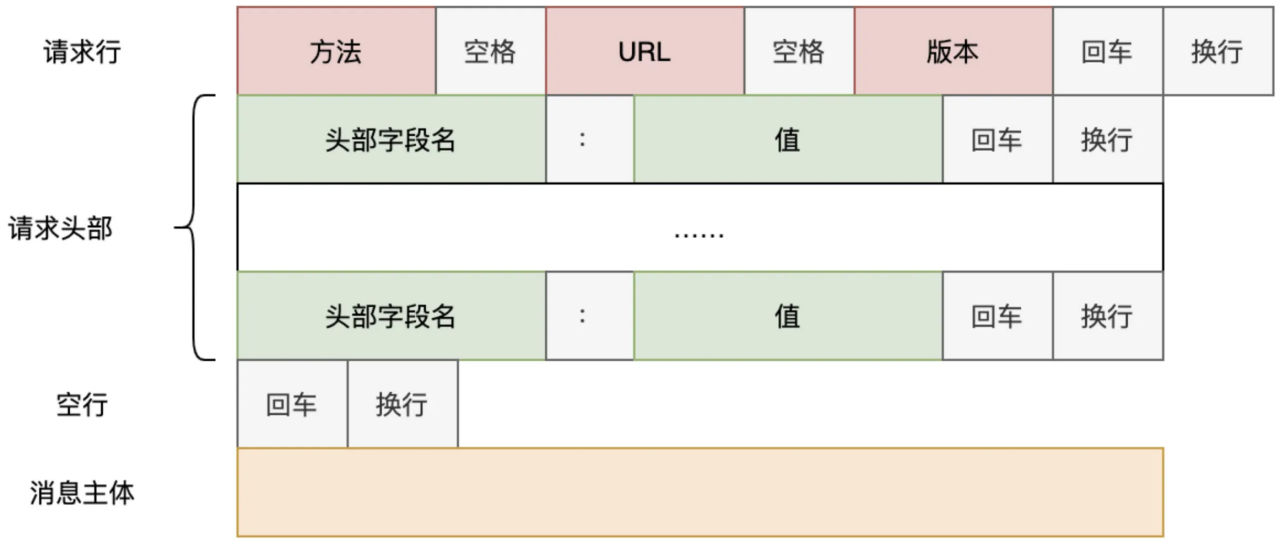
步骤四：释放连接TCP连接；

步骤五：客户端（浏览器）解析HTML内容并呈现；

如果在浏览器输入URL地址，地址为域名，则还需先向DNS服务器请求解析域名对应的IP，然后在基于IP和端口建立TCP连接。

HTTP请求报文

HTTP请求包含四个部分，分别是请求行（请求方法）、请求头（消息报头）、空行和请求正文。




```HTTP
# 请求行
POST /index.html HTTP/1.1
# 请求头
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:67.0)
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: zh-CN,zh;
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/index.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Connection: close
Cookie: security=impossible; PHPSESSID=8vv0n11btuol45hqcm5recmfp7
Upgrade-Insecure-Requests: 1

# 请求正文
username=admin&password=admin
```

需要注意的每一行末尾都有回车和换行，在内容实体和请求头之间有一个空行。

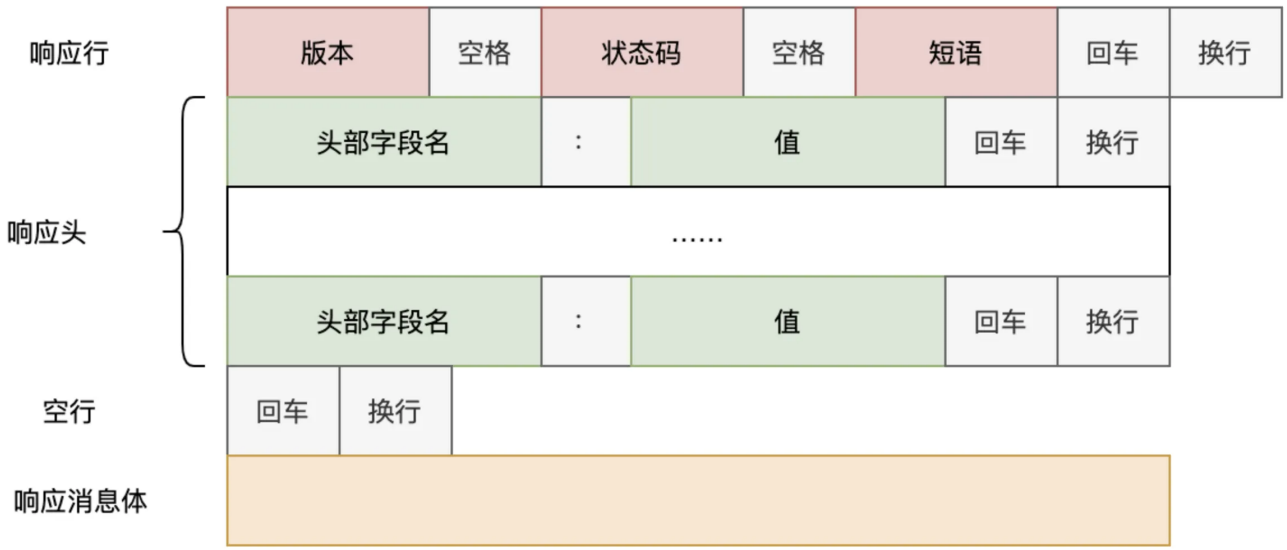
HTTP 请求头（HTTP Request Headers）是客户端发送给服务器的额外信息，用于描述客户端请求的各种属性和设置。它们在客户端和服务器之间的通信中扮演着重要的角色，影响着服务器如何处理请求以及客户端如何接收响应。

作用：

- **提供客户端信息：** 告知服务器客户端的软件类型（浏览器、爬虫等）、操作系统、语言偏好等。例如，`User-Agent` 头部字段。
- **指定请求内容：** 描述请求体的数据类型、编码方式、长度等。例如，`Content-Type` 、`Content-Length` 头部字段。
- **控制缓存：** 控制客户端和服务器端的缓存行为，例如，`Cache-Control` 、`If-Modified-Since` 头部字段。
- **设置 Cookie：** 用于客户端和服务器之间传递 Cookie 信息，例如，`Cookie` 头部字段。
- **进行身份验证和授权：** 传递身份验证和授权信息，例如，`Authorization` 头部字段。
- **实现条件请求：** 根据服务器资源的最新状态决定是否发送完整的响应，例如，`If-Match` 、`If-None-Match` 头部字段。
- **其他控制信息：** 例如，设置请求的优先级、指定可接受的响应类型等。`Accept` 、`Accept-Language` 、`Accept-Encoding` 等。

HTTP响应报文

HTTP响应由四部分组成，分别是响应行、响应头（消息报头）、空行和响应正文（消息主题）。



```HTTP
# 响应行
HTTP/1.1 200 OK
# 响应头
Date: Tue, 10 Aug 2021 09:09:09 GMT
//...省略...
Content-Length: 5185
Connection: close
Content-Type: text/html;charset=gb2312

# 响应正文
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
```

其中，Content-Length表示响应正文中内容的长度。

响应头（Response Header）是在HTTP协议中用于描述服务器响应的元数据。它是服务器在响应客户端请求时发送给客户端的一部分响应信息,包含了服务器的相关配置和响应内容的描述。响应头通常包含以下几个部分：

1.状态码（Status Code）： 表示服务器对请求的处理结果的状态码,如200表示成功、404表示未找到资源等。

2.状态消息（Status Message）： 对状态码的文本描述,如200的状态消息是"OK"。

3.协议版本（Protocol Version）： 指定了服务器使用的HTTP协议版本。

4.Content-Type（内容类型）： 指定了服务器返回的响应内容的类型。

5.Content-Length（内容长度）： 指定了响应内容的长度。

6.ETag（实体标记）： 用于实体的缓存控制,客户端可以通过该标记判断资源是否改变。

7.Location（重定向地址）： 用于重定向响应,客户端可以根据该字段的值重新发送请求。

8.Set-Cookie（设置Cookie）： 用于在响应中设置Cookie,服务器可以通过该字段在客户端保存会话信息。

响应头的作用是让服务器能够向客户端传递一些额外的信息,以控制或者说明响应的方式、内容和缓存等。客户端根据响应头中的信息进行相应的处理,如解析响应内容、处理重定向、保存Cookie等。响应头的具体内容可以根据实际需求进行定制和扩展。

什么是ajax?

在当今数字化快速发展的世界里，Web应用的响应速度和用户体验成为了竞争的关键。想象一下，当你打开一个网页，信息瞬间呈现，交互流畅无阻——这一切的背后，离不开JavaScript强大的数据接口请求技术。

Ajax（Asynchronous JavaScript and XML）是一种在Web开发中使用的技术，它允许网页在不重新加载整个页面的情况下从服务器请求和接收数据，并动态更新页面的一部分。这提供了更好的用户体验，因为用户无需等待整个页面的刷新即可看到更新的内容。

例如：

- 传统方式：使用 `XMLHttpRequest` 对象发起HTTP请求，获取服务器端的数据并在不刷新页面的情况下更新DOM。
- 现代方式：使用 `fetch` API，它们提供了更简洁、更强大的API来进行异步数据请求。这两种方法都基于Promise，支持流式处理和其他高级特性，使得开发更加高效。

`XMLHttpRequest`：传统但可靠

`XMLHttpRequest`（通常简称为XHR）是最早的用于浏览器异步请求的技术之一。它允许网页在不重新加载整个页面的情况下向服务器发送请求，并根据返回的数据更新部分页面内容。尽管它的API相对复杂，但它仍然是一个强大且可靠的工具。

特点

- **历史最悠久**：作为最早的异步请求技术之一，它在所有浏览器中都有很好的支持。
- **复杂API**：相比现代的替代品，它的API较为冗长且复杂，需要更多的代码来完成同样的任务。
- **适用于**：如果你需要向后兼容非常老的浏览器版本，或者项目中已经大量使用了 `XMLHttpRequest` 并且没有迁移计划。

```jsx
var xhr = new XMLHttpRequest();
xhr.open("GET", "https://netease-cloud-music-api-five-roan-88.vercel.app")
xhr.onreadystatechange = function (e) {
  console.log(e.target === this); // true
  console.log(this === xhr); // true
  if(this.status === 200 && this.readyState === 4) {
    console.log(JSON.parse(this.response))
  }
}
xhr.send();
```

代码分析：

1. 创建并配置XHR对象：

- `new XMLHttpRequest()`：创建一个新的XHR实例。
- `xhr.open('GET', 'https://api.github.com/orgs/lemoncode/members', false)`：初始化一个HTTP GET请求，指向GitHub的API端点。这里第三个参数设置为`false`，表示这是一个同步请求。
  - 请注意，当你使用同步请求会阻塞浏览器，直到请求完成，这会导致用户体验不佳，因此不推荐在现代Web开发中使用。

2. 处理服务器响应：

- `xhr.onreadystatechange`：设置一个回调函数，当服务器响应状态改变时触发，
  - 当`readyState`属性发生变化时触发。通过检查`readyState`和`status`，我们可以确定请求的状态，并根据结果采取相应的行动。
  - `if (xhr.status === 200 && xhr.readyState === 4)`：检查请求是否成功完成（HTTP状态码200表示成功，`readyState`为4表示请求已完成）。
  - `JSON.parse(xhr.responseText)`：将返回的JSON字符串解析为JavaScript对象。

3. 发送请求：

- `xhr.send()`：发送请求给服务器。
- 请求在最后一步是原因：
  - 初始化和配置请求：在发送请求之前，必须先创建并配置`XMLHttpRequest`对象。这包括指定请求方法（如GET、POST等）、目标URL以及是否为异步请求。

- 设置事件监听器：为了正确处理服务器的响应，你需要提前设置好回调函数或事件监听器来捕获各种状态变化和错误情况。这些监听器应该在请求被发送之前就准备好，以确保不会错过任何重要的响应信息。
- 遵循逻辑顺序：从逻辑上讲，只有当一切都准备就绪时，你才应该发起请求。如果过早地发送请求，可能会导致未定义的行为，因为某些关键设置还没有完成。

XMLHttpRequest对象中的readyState和status用于监控异步 HTTP 请求的状态。

它们分别表示请求的当前阶段和服务器的响应状态。

readyState 属性:

readyState 属性表示请求的当前状态，取值范围从 0 到 4。每个值的含义如下：

0: UNSENT 请求未初始化，尚未调用 open() 方法。

1: OPENED 请求已被初始化，调用了 open() 方法，但尚未调用 send() 方法。

2: HEADERS_RECEIVED 请求已发送，服务器响应头已被接收，但响应体尚未接收。

3: LOADING 响应体正在接收中。可以接收到部分数据。

4: DONE 请求已完成，响应已完全接收。此时可以通过 responseText 或 responseXML 属性访问响应数据。

status 属性:

101 Switching Protocols 服务器已理解客户端的请求，并将其协议更改为客户端所请求的协议。

200 OK 请求成功，服务器返回所请求的数据。

201 Created 请求成功并创建了新的资源，通常用于 POST 请求。

204 No Content 在更新一个资源后，如果不需要向客户端返回任何新的数据请求成功，常用于 DELETE 请求。

301 Moved Permanently 请求的资源已被永久移动到新 URI，未来的请求应使用新 URI。

304 Not Modified 资源未修改，可以使用缓存的版本。常用于条件请求。

400 Bad Request 请求无效，服务器无法理解。

401 Unauthorized 请求未授权，需提供身份验证。

403 Forbidden 服务器理解请求，但拒绝执行，用户无权访问请求的资源。

404 Not Found 请求的资源未找到。

405 Method Not Allowed 请求方法不被允许，资源不支持该方法。

500 Internal Server Error 服务器发生意外错误，无法完成请求。

502 Bad Gateway 服务器作为网关或代理时收到无效响应。

GET请求

GET请求的参数是通过在请求路径后面追加查询字符串（querystring）的方式向服务端传递参数。

查询参数的格式为：请求地址***?参数1=值1&参数2=值2...***

```JSX
var xhr = new XMLHttpRequest();
var baseUrl = "https://netease-cloud-music-api-five-roan-88.vercel.app";
xhr.open("GET", `${baseUrl}/search?keywords=海阔天空`)
xhr.onreadystatechange = function (e) {
  console.log(e.target === this); // true
  console.log(this === xhr); // true
  if(this.status === 200 && this.readyState === 4) {
    console.log(JSON.parse(this.response))
  }
}
xhr.send();
```

POST请求

~~POST请求的参数也可以通过在请求路径后面追加查询字符串（querystring）的方式向服务端传递参数。~~

一般情况下POST请求使用请求体向服务端提交数据（json/表单数据/二进制数据）。

> *application/json: 用于发送 JSON 数据。*
> *application/x-www-form-urlencoded: 用于发送表单数据，数据以 key=value 的形式编码。multipart/form-data：用于发送二进制数据如：图片、视频、文件等*

发送JSON数据

如果需要使用JSON数据发送到服务端一定要设置请求头**Content-Type**为application/json来告知服务端程序如何解析你传递的数据。

```JSX
var xhr = new XMLHttpRequest();
var baseUrl = "https://netease-cloud-music-api-five-roan-88.vercel.app";
xhr.open("GET", `${baseUrl}/search`);
xhr.setRequestHeader("Content-Type","application/json")
xhr.onreadystatechange = function (e) {
  console.log(e.target === this); // true
  console.log(this === xhr); // true
  if(this.status === 200 && this.readyState === 4) {
    console.log(JSON.parse(this.response))
  }
}
xhr.send({"keywords": "海阔天空"});
```

发送表单数据

如果需要使用表单数据发送到服务端一定要设置请求头**Content-Type**为application/x-www-form-urlencoded来告知服务端程序如何解析你传递的数据。

```JSX
var xhr = new XMLHttpRequest();
var baseUrl = "https://netease-cloud-music-api-five-roan-88.vercel.app";
xhr.open("GET", `${baseUrl}/search`);
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded")
xhr.onreadystatechange = function (e) {
  console.log(e.target === this); // true
  console.log(this === xhr); // true
  if(this.status === 200 && this.readyState === 4) {
    console.log(JSON.parse(this.response))
  }
}
xhr.send("keywords=海阔天空");
```

发送二进制数据

如果需要使用二进制数据发送到服务端一定要设置请求头**Content-Type**为multipart/form-data来告知服务端程序如何解析你传递的数据。

```JSX
var xhr = new XMLHttpRequest();
var baseUrl = "https://netease-cloud-music-api-five-roan-88.vercel.app";
xhr.open("GET", `${baseUrl}/search`);
xhr.setRequestHeader("Content-Type","multipart/form-data")
xhr.onreadystatechange = function (e) {
  console.log(e.target === this); // true
  console.log(this === xhr); // true
  if(this.status === 200 && this.readyState === 4) {
    console.log(JSON.parse(this.response))
  }
}
var fd = new FormData();
fd.append("name": "avatar");
fd.append("file": file);
fd.append("filename": "abc.png")
xhr.send(fd);
```

通过监听xhr.upload.onprogress事件，来获取文件上传进度

```JSX
// 监听xhr.upload的onprogress事件
xhr.upload.onprogress = function(e) {
  // e.lengthComputable表示当前上传资源是否具有可计算长度，是一个布尔值
  if (e.lengthComputable) {
    // e.loaded已传输的字节
    // e.total 需传输的总字节
    var percentComplete = Math.ceil((e.loaded / e.total) * 100)
  }
}
```

监听上传完成的事件

```JSX
  xhr.upload.onload = function(e) {
      console.log('上传成功')
  }

```