文本

描述已自动生成HTTP

URI（统一资源标识符）：

URI是一个用于标识资源的字符串序列。它可以是一个标准化的字符串，用于唯一地标识资源的位置、名称或者其他属性。包括URL和URN。

URL（统一资源定位符）：

URL是URI的一种特定形式，用于指定资源的位置（例如，资源的网络地址）。它提供了一个标准化的方式来指定互联网上的资源的位置。

URL通常由协议、主机名、端口号、路径和查询参数等组成，以便客户端能够准确地定位资源。

标准格式： 协议类型:[//服务器地址[:端口号]][/资源层级 UNIX 文件路径]文件名[?查询字符串][#片段标识符]

完整格式： 协议类型:[//[访问资源需要的凭证信息@]服务器地址[:端口号]][/资源层级 UNIX 文件路径]文件名[?查询字符串][#片段标识符]

表格

描述已自动生成参数介绍：

URN（统一资源名称）：

URN是URI的另一种形式，用于指定资源的名称，而不是其位置。URN的目的是提供一种持久的、全球唯一的资源标识符。

与URL不同，URN不指定资源的位置，而是指定资源的名称或其他属性。

表格

描述已自动生成方法：

GET 方法

基本介绍：

GET 是最常用的 HTTP 方法，常用于获取服务器上的某个资源。以下几种方式都会触发 GET 方法的请求

在浏览器中直接输入 URL 回车或点击浏览器收藏夹中的链接，此时浏览器就会发送出一个 GET 请求。

HTML 中的 link、img、script 等标签的属性中放的 URL，浏览器也会构造出 HTTP GET 请求

使用 Javascript 重点 ajax，也能构造出 HTTP GET 请求

各种编程语言（只要能够访问网络），就都能够构造出 HTTP GER 请求

GET 请求的特点：

首行里面的第一个部分就是 GET、URL 里面的 query string 可以为空，也可以不为空、GET 请求的 header 有若干个键值对结构、GET 请求的 body 一般是空的

POST 方法

基本介绍：

POST 方法也是一种常见的方法，多用于提交用户输入的数据给服务器（如登录页面）。以下几种方法都会触发 POST 方法的请求

通过 HTML 中的 form 标签可以构造 POST 请求

使用 JavaScript 的 ajax 可以构造 POST 请求

POST 请求的特点：

首行第一个部分就是 POST、URL 里面的 query string 一般是空的、POST 请求的 header 里面有若干个键值对、POST 请求的 body 一般不为空（body 的具体数据格式，由 header 中的 Content-Type 来描述；body 的具体数据长度，由 header 中的 Content-Length 来描述

HEAD方法与GET相比只返回响应头信息不返回实际主体；PUT方法与POST类似，但是在HTTP中，PUT被定义为具备[幂等性（idempotent）](https://blog.csdn.net/post/119)的方法，POST则不是。

PUT 方法用于上传或更新资源，是幂等的，通常用于创建或更新资源。

POST 方法用于提交数据，通常用于创建新的资源或执行某些操作，不是幂等的。

请求头重要信息如下：

HOST 的值表示服务器主机的地址和端口（地址可以是域名，也可以是 IP；端口号可以省略或者手动指定），Content-Length表示 body 的数据长度，长度单位是字节，Content-Type 表示 body 的数据格式，User-Agent 表示浏览器或者操作系统的属性，Referer 表示这个页面是从哪个页面跳转过来的。

文本

中度可信度描述已自动生成Response类型：

302 Move temporarily表示临时重定向

Cookie和Session：

Cookie：

Cookie是由服务器发送到客户端的小型文本文件，存储在客户端的浏览器中。

Cookie通常包含有关用户的一些信息，例如用户ID、偏好设置等。

客户端的每个请求都会将相应的Cookie发送回服务器，以便服务器可以根据Cookie中的信息进行响应。

Cookie有一些限制，例如存储容量的限制，以及安全性方面的考虑。

Session：

Session是服务器端维护的关于用户的会话信息，通常存储在服务器的内存或数据库中。

每个会话都有一个唯一的会话ID，通常通过Cookie或URL参数发送到客户端。

与Cookie不同，Session数据存储在服务器上，因此可以存储更多的信息，并且相对更安全。

Session通常用于存储用户的登录状态、购物车内容等重要信息。

比较：

Cookie存储在客户端，Session存储在服务器端。

Cookie相对不安全，容易被篡改，而Session存储在服务器端，相对更安全。

Cookie可以设置过期时间，而Session通常在一段时间内保持活动状态，或者在用户退出时过期。

HTTPS相比HTTP有三大优点：

数据加密（Data Encryption）：HTTPS使用SSL/TLS协议来加密数据传输，使得传输的数据在网络上传输过程中无法被窃听和窃取。这样可以保护用户的隐私和数据安全，防止敏感信息泄露。

身份认证（Identity Authentication/Confidential Reality）：HTTPS使用SSL/TLS证书(CA)对服务器进行身份认证，确保客户端连接到正确的服务器，防止中间人攻击。这样可以保证用户与服务器之间的通信是安全可信的。

数据完整性（Data Integrity）：HTTPS使用SSL/TLS协议对数据进行加密和认证，防止数据在传输过程中被篡改或修改。这样可以保证数据的完整性，确保数据在传输过程中不被篡改。

TLS工作原理：

**握手阶段（Handshake）：**

客户端发送一个加密套件列表和一些其他信息给服务器，包括支持的TLS版本和随机数。

服务器从客户端提供的加密套件列表中选择一个加密算法，并发送自己的证书（可选）和一些其他信息给客户端，包括服务器的随机数和服务器的密钥交换参数。

客户端验证服务器的证书，验证完成后生成一个预主密钥，并使用服务器的公钥对其进行加密，并将其发送给服务器。

服务器使用自己的私钥解密预主密钥，并使用预主密钥生成会话密钥。

客户端和服务器都使用协商的会话密钥来生成密钥材料，用于加密和验证后续的通信。

**密钥交换阶段（Key Exchange）：**

客户端和服务器使用协商的会话密钥来生成用于加密和验证通信的对称密钥。

对称密钥用于对通信数据进行加密和解密，以确保数据在传输过程中的保密性和完整性。

**数据传输阶段（Data Transfer）：**

客户端和服务器使用协商的加密算法和对称密钥来加密和解密通信数据。

客户端和服务器之间的通信数据在传输过程中得到保护，防止被窃听和篡改。

**会话终止阶段（Session Termination）：**

通信结束后，客户端和服务器可以选择终止会话或者保持会话以供后续通信使用。

如果终止会话，对称密钥将被丢弃，保证会话的安全性。

**URI scheme**通常由一个或多个小写字母组成，它们用于表示协议、服务或者资源的类型。一些常见的URI scheme包括：

https：用于指定安全的超文本传输协议（HTTPS），通过加密通信保护Web页面和其他资源的URI。

ftp：用于指定文件传输协议（FTP），用于访问和传输文件的URI。

mailto：用于指定邮件地址，表示电子邮件的URI。

file：用于指定本地文件系统的URI，用于访问本地文件资源。

tel：用于指定电话号码，表示电话号码的URI。

urn：用于指定统一资源名称（Uniform Resource Name），表示永久性资源的URI，通常用于唯一标识资源而不关注其位置。

data：用于指定数据URI，允许在URI中直接嵌入数据，通常用于在Web页面中嵌入小型数据资源。

git：用于指定Git资源的URI，用于访问和管理Git版本控制系统中的代码库。

magnet：用于指定磁力链接，用于通过P2P协议获取资源，常用于下载种子文件。

图形用户界面, 文本, 应用程序

描述已自动生成**REST**

REST的主要特征和原则：

资源（Resources）：系统中的所有内容都被视为资源，每个资源都有一个唯一的标识符（URI）。资源可以是任何类型的数据，例如文本、图像、视频、数据库记录等。

表现层（Representation）：资源的表现形式是指资源的数据表示方式，通常是JSON或XML格式的数据。客户端通过HTTP请求获取资源的表现形式，并与之交互。

状态（State）：客户端和服务器之间的交互是无状态的，即每个请求都包含了足够的信息来处理该请求，而不需要依赖于之前的请求。服务器不会保存客户端的状态，客户端必须包含所有必要的信息。

统一接口（Uniform Interface）：REST使用统一的接口来对资源进行操作，包括GET（获取资源）、POST（创建资源）、PUT（更新资源）、DELETE（删除资源）等操作。这些操作通过HTTP方法来表示，并且在不同的资源上具有相同的语义。

无状态（Stateless）：REST是一种无状态协议，服务器不会保存客户端的状态，客户端必须包含所有必要的信息。这样可以使系统更具可扩展性和可靠性。

客户端-服务器分离（Client-Server Separation）：客户端和服务器之间的交互是通过标准的HTTP协议进行的，彼此之间是解耦的，这样可以使系统更具灵活性和可伸缩性。

**HTML**

第一行：<!DOCTYPE html>

在一个html里面，包含head和body两个部分，最外面是html标签

在head里面的元素：base; link; meta; noscript; script; style; template; title

Block：块级元素在页面中以块的形式展现，会另起一行。Inline：内联元素通常出现在块级元素中并环绕文档内容的一小部分。

<br />换行<hr />空一行

一个块级元素不会嵌套在一个内联元素里面，但它可能嵌套在另一个块级元素里面。

为元素添加属性：<p class=”style”>test<p/>元素p和属性之间要有空格，为一个元素添加多个属性，属性之间要有空格，以图片这个元素为例：

<img src=<https://raw.githubusercontent.com/mdn/beginner-html-site/gh-pages/images/firefox-icon.png> alt="The Firefox icon" width="300" height="300" />

Alt是图片加载不出来时显示的文字

HTML 实体引用(HTML entity reference)是一种用于在 HTML 中表示特殊字符的方法。在 HTML 中，有些字符具有特殊的含义，比如小于号（<）、大于号（>）、引号（"）、和符号（&），如果直接在 HTML 中使用这些字符，可能会被解析器误解为标签或其他语法结构，导致页面显示不正确。

为了避免这种问题，可以使用实体引用来表示这些特殊字符。实体引用是以 & 开头、以 ; 结尾的字符序列，它们代表了特定的字符。例如，&lt; 表示小于号（<），&gt; 表示大于号（>），&quot; 表示引号（"），&amp; 表示和符号（&）等等。

以下是一些常用的 HTML 实体引用示例：

&lt; 表示小于号（<） &gt; 表示大于号（>） &quot; 表示引号（"） &apos; 表示引号（‘） &amp; 表示和符号（&）

&copy; 表示版权符号（©） &reg; 表示注册符号（®） &nbsp; 表示空格符（non-breaking space）

<p>HTML 中用 <p> 来定义段落元素。</p> <-错误，第二个<p>会起到换行作用

<p>HTML 中用 &lt;p&gt; 来定义段落元素</p> <-正确

HTML注释：开头：<!-- ，结尾： --> <!--这就是注释-->

HTML语义标签（semantic tag）及其作用：

<header>：定义页面或节的页眉。 <footer>：定义页面或节的页脚。 <nav>：定义导航链接的容器。 <main>：定义文档的主要内容区域。

<article>：定义独立的内容块，如博客文章或新闻报道。 <section>：定义文档中的一个区域或节。 <aside>：定义与页面内容相关但不是主要内容的部分，如侧边栏或广告。

<figure>：定义一组媒体内容和它们的标题。 <figcaption>：定义<figure>元素的标题。 <time>：定义日期和时间，有助于机器和搜索引擎识别日期和时间信息。

<details>：定义可展开的详细信息。 <summary>：定义<details>元素的摘要或标题。 <address>：定义联系信息。 <blockquote>：定义长的引用。

<cite>：定义引用的作者或标题。 <abbr>：定义缩写或首字母缩略语。 <mark>：定义需要突出显示的文本。 <strong>：定义强调文本。

<em>：定义强调文本的语气。 <progress>：定义任务的进度。

<q> 标签用于定义短的引用或内联引用。通常浏览器会自动为引用添加引号。 <var> 标签用于定义变量，表示在数学或逻辑上的变量。

部分语义标签如下：

表格

描述已自动生成

Placeholder标签：

<section> 标签：（会换行）

<section> 标签用于定义文档的独立节（section），通常表示文档中的一个主题或区域。

它用于组织和分隔页面内容，通常包含一个标题（通常是 <h1> 到 <h6> 标签）来描述该节的主题。

<div> 标签：(会换行)

<div> 标签是一个通用的容器元素，用于组织和布局页面的内容，它本身没有特定的语义。

它可以用于将相关内容组合在一起，并为这些内容应用样式或脚本。

<div> 通常被用于创建网页的布局结构，或者将内容分组为不同的部分以便于样式化或脚本操作。

<span> 标签：

<span> 标签用于对文本中的一部分进行分组，并为其应用样式或脚本。

它本身没有特定的语义，通常用于对行内元素进行包裹和操作。

文本

描述已自动生成<span> 可以用于对文本中的一部分进行高亮、加粗、斜体等操作。

文本

描述已自动生成文本, 信件

描述已自动生成有序列表ol和无序列表ul：

超链接link：

通过将文本或其他内容包裹在 <a> 元素内，并给它一个包含网址的 href 属性（也称为超文本引用或目标，它将包含一个网址）来创建一个基本链接。

图片包含 文本

描述已自动生成

如果有需要作为链接的图片，使用 [<a>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/a) 元素来包裹要引用图片的 [<img>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/img) 元素。

文本

中度可信度描述已自动生成

也可以在连接内部添加title来给超链接增加补充信息，信息将在鼠标放在超链接处显示

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 应用程序

中度可信度描述已自动生成

以上表明当前目录是bristol.ac.uk/students,第一个/courses是指同级的courses目录，第二个是指当前目录的courses子目录，第三个是指上一级目录的中的courses子目录，即当前同级的courses目录

超链接除了可以链接到文档外，也可以链接到 HTML 文档的特定部分（被称为文档片段）。要做到这一点，你必须首先给要链接到的元素分配一个 id 属性。通常情况下，链接到一个特定的标题是有意义的，这看起来就像下面这样：<h2 id="Mailing\_address">邮寄地址</h2>，为了链接到那个特定的 id，要将它放在 URL 的末尾，并在前面包含井号（#），例如：

<p>要提供意见和建议，请将信件邮寄至<a href="contacts.html#Mailing\_address">我们的地址</a></p>

<a> 标签是 HTML 中用于创建超链接的元素，它具有以下常用的属性：

href：指定链接的目标地址。可以是一个 URL、文件路径或页面内的锚点。

target：指定链接的打开方式。常见取值包括：

\_self：在当前窗口中打开链接（默认）。

\_blank：在新的浏览器标签页中打开链接。

\_parent：在父级窗口中打开链接。

\_top：在顶级窗口中打开链接。

download：表示链接指向的资源将被下载而不是在浏览器中打开，可选值为要保存的文件名。

rel：指定链接与当前文档的关系。常见取值包括：

noopener：确保链接在新窗口中打开时没有对原始页面的访问权限。

noreferrer：阻止浏览器发送 Referer 头部到目标页面。

title：提供关于链接的附加信息，通常显示为工具提示。

Form表单：action是处理该表单的url地址，method是处理输入数据的方法

文本

描述已自动生成

将一个 <label> 和一个 <input> 元素匹配在一起，你需要给 <input> 一个 id 属性。而 <label> 需要一个 for 属性，其值和 <input> 的 id 一样。

Form的input中的name就是表单的名字，不同表单的名字必须不同，input中的required表示必须填写，input 中的placeholder是未输入数据时的提示词

<form action="/submit" method="post" autocomplete="off">

<label for="password">密码：</label>

<input type="password" id="password" name="password" autocomplete="off" />

<input type="submit" value="提交" />

</form>

注意表单中的autocomplete属性，如果是涉及密码一般用off，填写重复内容或者公共内容可以on。表单中的input类型：

表格

描述已自动生成

单选用radio，多选用checkbox， disabled属性表示禁用，禁止与用户交互

<select>元素表示一个提供选项菜单的控件

图形用户界面, 文本, 应用程序

描述已自动生成

Multiple表示多选，size表示一次显示4个，disabled表示该选项不可选

**Table表格**

Caption是table的子元素，用于为table添加标题，每个table只能有一个

图形用户界面, 文本, 应用程序

描述已自动生成

Thead是表头，th是表头的具体内容会加粗显示（也可以放在数据行中充当数据项），tbody是表体，tr是每一行，td是每行的数据项，

Th中的scope用于指定表头单元格（<th>）的作用范围，可以是col，row，colgroup，rowgroup, colspan表示占几列，在th中有效

HTML结构：

<main> is for content unique to this page. Use <main> only once per page, and put it directly inside <body>. Ideally this shouldn't be nested within other elements.

<article> encloses a block of related content that makes sense on its own without the rest of the page (e.g., a single blog post).

<section> is similar to <article>, but it is more for grouping together a single part of the page that constitutes one single piece of functionality (e.g., a mini map, or a set of article headlines and summaries), or a theme. It's considered best practice to begin each section with a heading; also note that you can break <article>s up into different <section>s, or <section>s up into different <article>s, depending on the context.

<aside> contains content that is not directly related to the main content but can provide additional information indirectly related to it (glossary entries, author biography, related links, etc.).

<header> represents a group of introductory content. If it is a child of <body> it defines the global header of a webpage, but if it's a child of an <article> or <section> it defines a specific header for that section (try not to confuse this with titles and headings).

<nav> contains the main navigation functionality for the page. Secondary links, etc., would not go in the navigation.

<footer> represents a group of end content for a page.

<main> 是用于页面独有内容的。每个页面只应该使用一次 <main>，并直接放在 <body> 内部。理想情况下，不应将其嵌套在其他元素内部。

<article> 包含了一个独立的、完整的内容块，该内容块可以独立于页面的其余部分存在（例如，一个单独的博客文章）。

<section> 类似于 <article>，但更多用于将页面的一个单独部分组合在一起，构成一个单一的功能（例如，一个迷你地图，或一组文章标题和摘要），或一个主题。最佳实践是在每个 section 开始时都添加一个标题；此外请注意，根据上下文，可以将 <article> 分成不同的 <section>，或将 <section> 分成不同的 <article>。

<aside> 包含的内容与主要内容无直接关联，但可以提供间接相关的额外信息（例如，词汇表条目、作者简介、相关链接等）。

<header> 表示一组引导性内容。如果它是 <body> 的子元素，则定义网页的全局页眉，但如果它是 <article> 或 <section> 的子元素，则定义该部分的特定页眉（请注意，不要与标题混淆）。

<nav> 包含页面的主导航功能。次要链接等不应放在导航中。

<footer> 表示页面的末尾内容组。