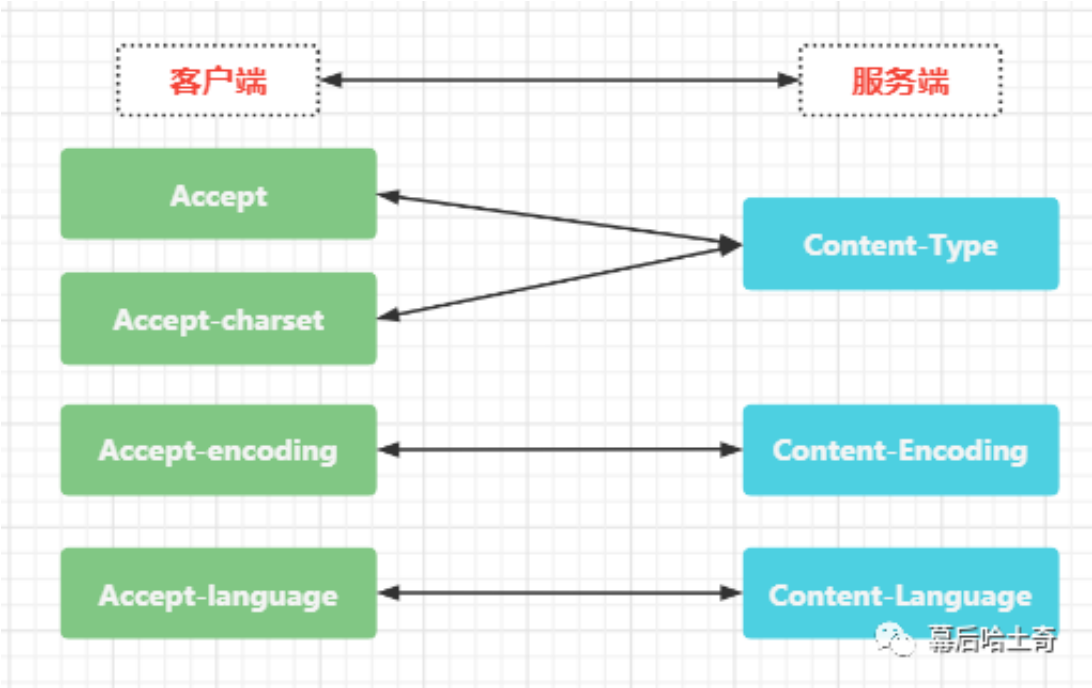
通信需要基于双方协商的协议，就像两个人约定都用同一种语言说话才能互相沟通和理解，我们人类的语言最主要的目的是传输自己的想法，在计算机网络中也是一样，最核心的目的是数据传输，不同于人类的语言，网络中的数据可能是普通文本数据，也可能是图像数据或音视频数据，或者其他特定格式的数据，这需要协议层面进行确定。

此外，由于网络资源十分宝贵，一般数据都会经过压缩后才会传输，这就涉及到压缩编码的协商；客户端和服务端通信的语言和字符集也同样需要明确。

在HTTP协议中，这些协商都是通过头部字段确定的，将涉及讨论如下七个首部字段：



让我们一个个来看看是什么含义和作用。

一、什么是MIME type

在浏览器中显示的内容有 HTML、有 XML、有 GIF、还有 Flash ……那么，浏览器是如何区分它们，决定什么内容用什么形式来显示呢？答案是 MIME Type。

MIME（Multipurpose Internet Mail Extensions），是描述消息内容类型的因特网标准，说白了也就是文件的媒体类型。浏览器可以根据它来区分文件，然后决定什么内容用什么形式来显示。

部分常见的MIME type：

媒体类型	MIME type
超文本标记语言文本 .html	text/html
普通文本 .txt	text/plain
GIF图形 .gif	image/gif
JPEG图形 .jpeg,.jpg	image/jpeg
json数据	application/json
mp3	audio/mpeg
mp4	video/mp4

还有很多其他的，太多了，这里不一一列举了，可以发现，一般是以[type]/[subtype]的形式表示，前面的type有以下形式：

- **text**：即文本格式的可读数据，我们最熟悉的应该就是 text/html 了，表示超文本文档，此外还有纯文本 text/plain、样式表 text/css 等。；
- **image**：用于传输静态图片数据，有 image/gif、image/jpeg、image/png 等；
- **audio**：用于传输音频或者音声数据；
- **video**：用于传输视频数据；
- **application**：数据格式不固定，可能是文本也可能是二进制，必须由上层应用程序来解释。常见的有 application/json，application/javascript、application/pdf 等，另外，如果实在是不知道数据是什么类型，像刚才说的“黑盒”，就会是 application/octet-stream，即不透明的二进制数据；
- 等等...

subtype用于指定type的详细形式，感兴趣的话可以去百度获取完整的列表，总之，拥有了MIME type后，web客户端和服务端就可以根据类型进行相应地解析处理了。

二、什么是Encoding type

光有数据类型还不够，数据借助HTTP传输的时候，往往还会压缩数据以节约带宽。**为了明确压缩格式，双方也需要进行协商，以使对方能正确解压缩，还原出原始的数据。**

压缩类型一般有：

- gzip：GNU zip 压缩格式，也是互联网上最流行的压缩格式；
- deflate：zlib（deflate）压缩格式，流行程度仅次于 gzip；

- br：一种专门为 HTTP 优化的新压缩算法（Brotli）；
- 等等

三、HTTP之MIME type协商

上面说了MIME type和Encoding type两个概念，下面就要说明在HTTP协议中是如何具体协商的了，我们先来看下Accept字段。

Accept：接收的意思，客户端用 Accept 头告诉服务器它希望接收什么样类型的数据。

我们将图片解释下。

假设客户端的HTTP请求报文首部字段中Accept字段填的是：

```
Accept: text/html,application/xml,image/webp,image/png
```

这就是告诉服务器：我能够看懂 HTML、XML 的文本，还有 webp 和 png 的图片，请给我这四类格式的数据，其他格式我无法处理。

可以看出，为了让服务器有更多的选择余地，客户端的Accept字段的值可以是多个。

相应的，服务器会在响应报文里用头字段Content-Type告诉实体数据的真实类型。

假设服务器回复的是：

```
Content-Type: text/html
```

客户端看到响应报文首部中内容类型是text/html，说明这是一个HTML文件，会对其进行渲染变成我们经常访问到的网页。

假设服务器回复的是：

```
Content-Type: image/png
```

客户端则相应地显示为一张图片。

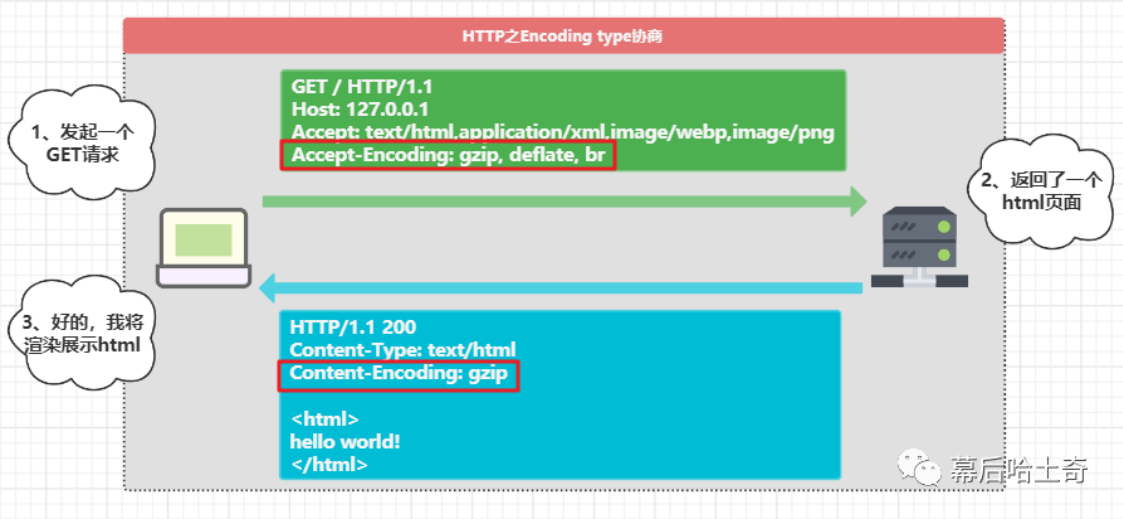
四、HTTP之Encoding type协商

Accept字段表示的是客户端希望接收什么样MIME type的数据，那同理容易想到，客户端应当也会发出它希望的压缩格式。

Accept-Encoding字段标记的是客户端支持的压缩格式，也可以列出多个来，例如：

```
Accept-Encoding: gzip, deflate, br
```

即客户端支持的压缩格式有三种，你服务器告诉我选择哪一种进行压缩的，只要在这个三种内，我就可以正常解压缩还原出原始的数据。



服务器通过Content-Encoding字段进行响应，比如：

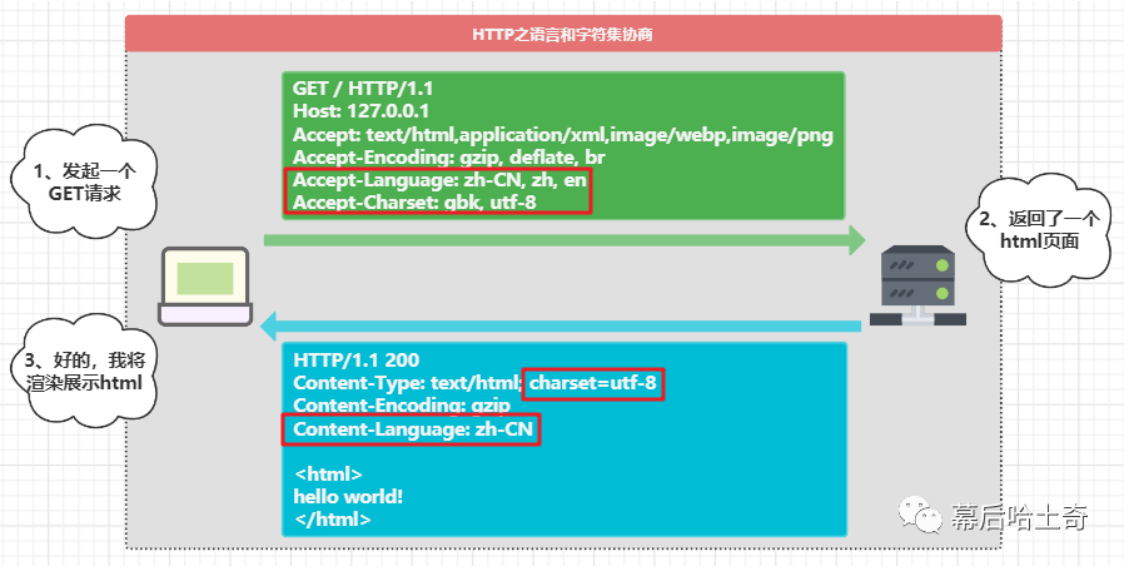
```
Content-Encoding: gzip
```

它们之间的数据传输即可利用gzip压缩格式进行压缩和解压缩了。

此外，需要注意的是，**这两个字段是可以省略的，当请求报文中没有 Accept-Encoding 字段，就表示客户端不支持压缩数据；如果响应报文里没有 Content-Encoding 字段，就表示响应数据没有被压缩。**

五、HTTP之语言和字符集协商

理解了上面的机制，下面就很简单了，这里所要介绍的是客户端和服务器就语言与字符集进行“内容协商”。



Accept-Language字段标记了客户端可理解的自然语言，也允许用“,”做分隔符列出多个类型，例如：

```
Accept-Language: zh-CN, zh, en
```

这个请求头会告诉服务器：“最好给我 zh-CN 的汉语文字，如果没有就用其他的汉语方言，如果还没有就给英文”。

相应的，服务器应该在响应报文里用头字段Content-Language告诉客户端实体数据使用的实际语言类型。

```
Content-Language: zh-CN
```

最后再说下字符集协商问题。

客户端使用Accept-Charset来表示自己支持的字符集，例如：

```
Accept-Charset: gbk, utf-8
```

需要注意，服务器响应头中并没有对应的Content-Charset字段，而是在Content-Type字段的数据类型后面用“charset=xxx”来表示，例如：

```
Content-Type: text/html; charset=utf-8
```

六、权重问题

最后补充一个小点。

在 HTTP 协议里用 Accept、Accept-Encoding、Accept-Language 等请求头字段进行内容协商的时候，还可以用一种特殊的“q”参数表示权重来设定优先级。

例如下面的 Accept 字段：

```
Accept: text/html,application/xml;q=0.9,/;q=0.8
```

权重的最大值是 1，最小值是 0.01，默认值是 1，如果值是 0 就表示拒绝。

它表示浏览器最希望使用的是 HTML 文件，权重是 1，其次是 XML 文件，权重是 0.9，最后是任意数据类型，权重是0.8。服务器收到请求头后，就会计算权重，再根据自己的实际情况优先输出 HTML 或者 XML。

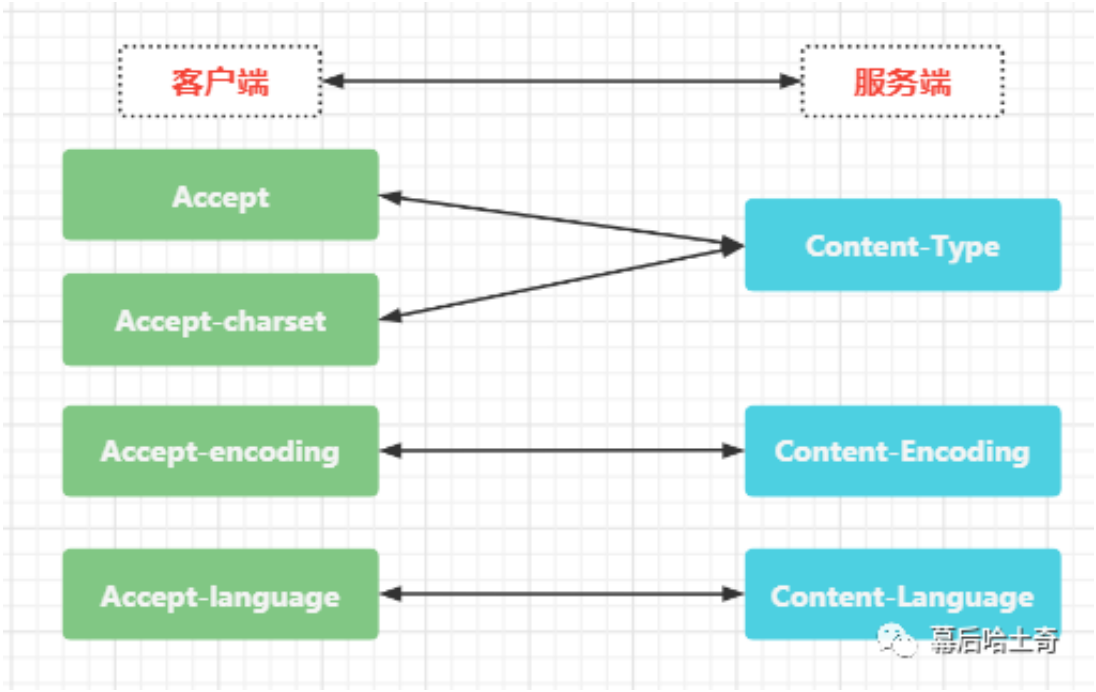
这里要提醒的是“;”的用法，在大多数编程语言里“;”的断句语气要强于“,”，而在 HTTP 的内容协商里却恰好反了过来，“;”的意义是小于“,”的。

七、总结

本篇文章所介绍的是数据协商问题，具体点是在HTTP协议中，如何来协商客户端和服务端的数据类型、数据压缩格式、语言以及编码问题。

一共讨论到了七个字段，包括客户端请求的四个字段：Accept、Accept-encoding、Accept-language、Accept-charset，服务端响应的3个字段：Content-Type、Content-Encoding、Content-Language。

它们的对应关系为：



- 数据类型表示实体数据的内容是什么，使用的是 MIME type，相关的头字段是 Accept和 Content-Type；
- 数据编码表示实体数据的压缩方式，相关的头字段是 Accept-Encoding 和 Content-Encoding；
- 语言类型表示实体数据的自然语言，相关的头字段是 Accept-Language 和 Content-Language；
- 字符集表示实体数据的编码方式，相关的头字段是 Accept-Charset和 Content-Type；

本文完，下篇见。