**1 解析一个完整URL中的各个组成部分**

城市中的建筑使用标准格式的地址来描述，如“北京市海淀区北三环西路18号”；类似的，网络中的资源使用标准格式的URL地址来描述。一个URL的完整地址如下所示：

<scheme>://<user>:<pwd>@<host>:<port>/<path>;<params>?<query>#<frag>。

例如，在某台Web服务器中有一个记录小说内容的页面，其完整URL格式形如：

http://pub:pub@terena.com@192.168.101.55:80/archive/xiyou.html;sid=Fe0511d9e?lang=zh&pc=1085#chapter2。

请使用正则表达式，解析出上述URL中的各个组成部分，输出在命令行中。

**参考答案**

实现此案例需要按照如下步骤进行。

**步骤一：创建页面**

首先创建一个纯文本文件，并修改名称为 exercise01\_03.html，然后在文本编辑工具中打开此文件，为其添加 HTML 代码，以创建一个标准结构的 HTML 文档（使用过渡型文档类型声明），并设置文档的标题以及编码格式。代码如下所示：

1. **<!**DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"**>**
2. **<**html xmlns**=**"http://www.w3.org/1999/xhtml"**>**
3. **<**head**>**
4. **<**meta http**-**equiv**=**"Content-Type" content**=**"text/html; charset=utf-8" **/>**
5. **<**title**>**解析一个完整URL中的各个组成部分**</**title**>**
6. **</**head**>**
7. **<**body**>**
8. **</**body**>**
9. **</**html**>**

**步骤二：添加提示文字，以及嵌入在BODY中执行的JS代码块**

1. **<!**DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"**>**
2. **<**html xmlns**=**"http://www.w3.org/1999/xhtml"**>**
3. **<**head**>**
4. **<**meta http**-**equiv**=**"Content-Type" content**=**"text/html; charset=utf-8" **/>**
5. **<**title**>**解析一个完整URL中的各个组成部分**</**title**>**
6. **</**head**>**
7. **<**body**>**
8. **<**h2**>**解析一个完整URL中的各个组成部分**</**h2**>**
9. **<**h3**>**http**:**//pub:pub@terena.com@192.168.101.55:80/archive/xiyou.html;sid=Fe0511d9e?lang=zh&pc=1085#chapter2</h3>
11. **<**script type**=**"text/javascript"**>**
13. **</**script**>**
14. **</**body**>**
15. **</**html**>**

**步骤三：创建URL变量和正则表达式，进行正则匹配**

1. **<!**DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"**>**
2. **<**html xmlns**=**"http://www.w3.org/1999/xhtml"**>**
3. **<**head**>**
4. **<**meta http**-**equiv**=**"Content-Type" content**=**"text/html; charset=utf-8" **/>**
5. **<**title**>**解析一个完整URL中的各个组成部分**</**title**>**
6. **</**head**>**
7. **<**body**>**
8. **<**h2**>**解析一个完整URL中的各个组成部分**</**h2**>**
9. **<**h3**>**http**:**//pub:pub@terena.com@192.168.101.55:80/archive/xiyou.html;sid=Fe0511d9e?lang=zh&pc=1085#chapter2</h3>
11. **<**script type**=**"text/javascript"**>**
12. **var** url **=** 'http://pub:pub@terena.com@192.168.101.55:80/archive/xiyou.html;sid=Fe0511d9e?lang=zh&pc=1085#chapter2'**;**
13. **var** regexp **=** /^(\w+):\/\/(\w+):(\S+)@(\S+):(\d+)\/(\S+);(\S+)\?(\S+)#(\S+)$/ig**;**
14. **if(**regexp**.**test**(**url**)){**
15. console**.**log**(**'scheme:**\t**'**+**RegExp**.**$**1);**
16. console**.**log**(**'user:**\t**'**+**RegExp**.**$**2);**
17. console**.**log**(**'pwd:**\t**'**+**RegExp**.**$**3);**
18. console**.**log**(**'host:**\t**'**+**RegExp**.**$**4);**
19. console**.**log**(**'port:**\t**'**+**RegExp**.**$**5);**
20. console**.**log**(**'path:**\t**'**+**RegExp**.**$**6);**
21. console**.**log**(**'params:**\t**'**+**RegExp**.**$**7);**
22. console**.**log**(**'query:**\t**'**+**RegExp**.**$**8);**
23. console**.**log**(**'frag:**\t**'**+**RegExp**.**$**9);**
24. **}else{**
25. console**.**log**(**'不是完整的URL地址：'**+**url**);**
26. **}**
27. **</**script**>**
28. **</**body**>**
29. **</**html**>**

**2 浏览器如何发起各种请求方法**

HTTP协议中规定：在请求消息的起始行中，必须定义如下请求方法之一：

(1)GET

(2)POST

(3)PUT

(4)DELETE

(5)HEAD

(6)TRACE

(7)OPTIONS

(8)CONNECT

描述通常情况下，浏览器中如何发起上述请求。

**参考答案**

一般情况下，浏览器能够发起的请求消息最常用的请求方法是GET和POST，二者都用于向服务器请求获取某资源；不同之处在于：GET请求消息中，不能向服务器提交请求正文，而POST请求消息中一般都存在请求正文。

浏览器下述默认发起的都是GET 请求：

(1)地址栏中直接输入URL；

(2)点击超链接，进行页面跳转；

(3)使用JavaScript进行页面跳转，如location.href="b.html";

(4)使用frame或iframe进行页面包含；

(5)使用GET方式提交表单

浏览器中默认只有POST方式提交表达发起的是POST请求。

**3 列举常见响应状态码及含义**

**参考答案**

服务器返回给客户端的响应消息中，第一行为起始行，其中需要声明响应状态码；HTTP协议定义了如下响应状态码：

1xx：提示消息，用于通知客户端请求已被服务器接收，并继续处理；

2xx：响应成功，需要的数据就在响应主体中；

3xx：重定向，需要客户端继续请求其它URL；

4xx：客户端错误，请求格式或内容有误，服务器无法根据无效的请求给出响应；

5xx：服务器端错误，请求没有无误，但服务器无法给出有效的响应。

**4 可以从哪些方面提高HTTP请求与响应的效率**

**参考答案**

网站性能优化最主要的就是要减少HTTP请求及每次响应中内容的长度。可以从连接过程中的下列方面加以考虑：

(1)域名解析

尽可能减少域名解析次数——少用外部链接文件(CSS/JS)

(2)创建连接

努力减少连接创建次数——使用Keep-Alive避免重复连接

(3)发送请求

尽力减少请求次数——合理设置Expires时间、资源合并

(4)等待响应

提高服务器端运行速度——提高数据运算及查询速度

(5)接收响应

尽可能减小响应数据长度——启用压缩