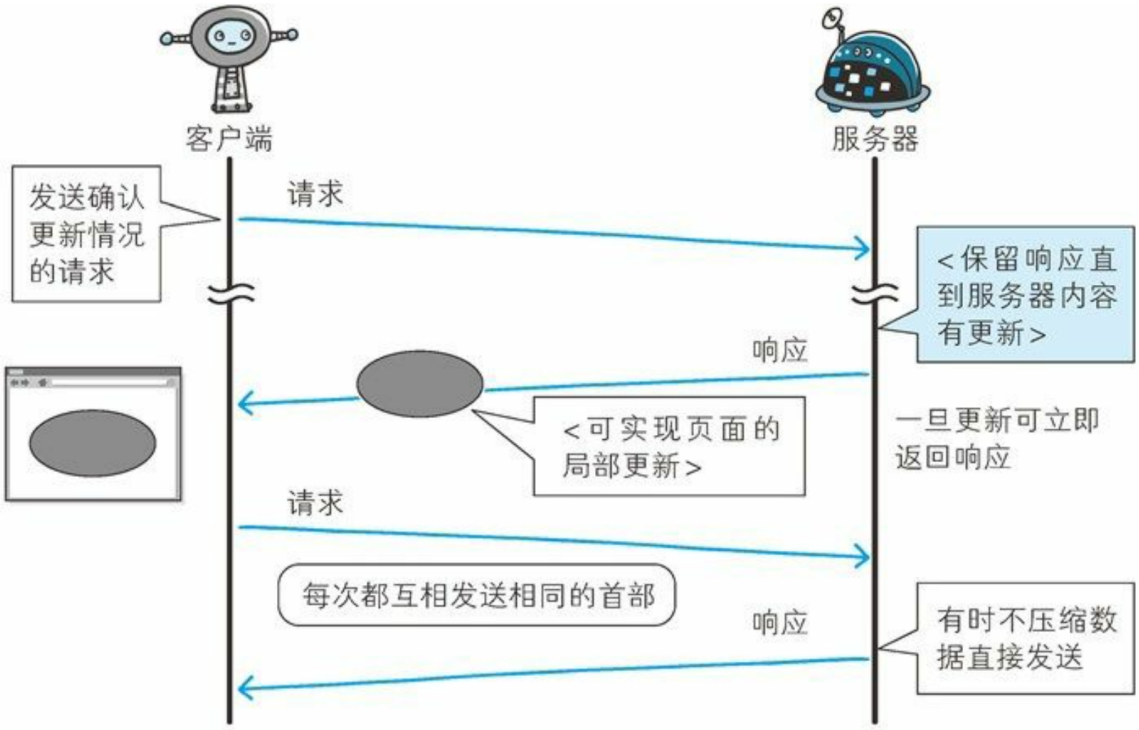
图解HTTP(9) 基于HTTP的功能追加协议1

1 使用HTTP协议探知服务器上是否有内容更新，就必须频繁地从客户端到服务端进行确认。
如果服务器上没有内容更新，那么 客户端就白白发送了一堆无用的请求。

2 HTTP标准存在瓶颈的可能原因
【首部压缩】请求 / 响应首部未经压缩就发送了。
【多路复用】一条连接上只能发一个请求。
【服务端推送？！！】请求只能从客户端开始。客户端不可以接收除响应以外的指令。
发送冗长的首部。每次互相发送相同的 首部 造成的浪费较多。
【数据压缩】可任意选择压缩格式。

3 Ajax：局部刷新【把要获取资源的 时机 交给了程序员】。
核心技术是 名为 XMLHttprequest 的API 【不应该是 类、对象吗？？怎么说是API？？】。
利用Ajax实时地从服务器获取内容，有可能 导致大量请求的产生。
而且 Ajax 仍未解决 HTTP协议 本身存在的问题【Ajax只是实现了 局部刷新！！】！！

4 Comet【“模拟 HTTP/2.0中的 服务端推送功能？？！“ 延迟应答？？！ 那如果一直保持连接，人家知道了，可能短时间内发大量请求、服务器可能就瘫痪了 DDos攻击？？！ 把要获取资源的 时机 交给了 服务器？？！】
一旦服务器有内容更新了，Comet不会让请求等待，而是直接 给客户端返回响应。
这是通过 延迟应答 ，模拟 服务端向客户端 推送的 功能。
原理：
服务端接收到请求，为了实现 推送功能 ， Comet【接收请求，但先不应答，等到又内容更新才应答回去！！】会将响应挂起，服务端一旦有了内容的更新，才会真的响应回去！！
缺点：
为了保留响应，一次连接【维持连接需要消耗更多的资源】的时间变长了。期间，为了维持连接需要消耗更多的资源。
而且 Comet只是 “模拟了服务端推送”， 但未解决 HTTP协议本身存在的问题！！



5 SPDY【在协议级别上改善 HTTP协议，不像 Ajax、Comet 那样“优化很局限”！！】
出现的背景：
陆续出现的 Ajax 和 Comet 等提高易用性的技术，一定程度上使 HTTP得到了改善，但 HTTP协议本身的一些限制仍束手无策！！
处于持续开发中的 SPDY协议， 正是 为了 在协议级别上 消除HTTP所遭遇的瓶颈！！！

SPDY的设计【为了安全 在表示层上 使用了SSL， SPDY在会话层，更高一点的层】与功能：
SPDY没有完全改写HTTP协议，只是在 TCP/IP 的应用层 和 运输层 之间通过新加会话的形式运作！！
考虑安全性，SPDY规定使用了 SSL 。
SPDY还是采用 HTTP建立连接【可照常使用HTTP的各种方法、东西等！！】，只是 以会话形式 加入、控制对数据的流动。
故，可照常使用 HTTP的GET、POST等方法、Cookie、HTTP报文等。

原先 HTTPS 就是 HTTP 和 TCP 多加了一层SSL。
现在的 SPDY在此基础上，再在 HTTP 和 SSL 之间加了一层 SPDY【所以它算是协议】 ?!!

功能【“怎么感觉这些功能像极了 HTTP / 2.0 ？ ？ ！ “】：
多路复用：所有请求可在同一条TCP上完成。

赋予请求优先级：可 无限制！！ 的并发处理请求，还可以给请求 逐个分配优先级顺序。 主要为了在发送多个请求时，解决因带宽而导致响应变慢的问题。

【HTTP首部】首部压缩：压缩HTTP的 请求 / 响应 首部。通信产生的数据包数量 和 发送的字节数 就更少了。

推送功能：服务器可直接发送数据，不必等待客户端的请求【原理呢？ ？ ！ 类似HTTP/2.0 中的在 nginx配置 http2_push字段的值去推送额外的资源】

服务器提示功能：服务器主动提示客户端请求所需的资源。由于在客户端发现资源之前就可获知资源的存在，因此资源已经缓存等情况下，可避免发送不必要的请求【原理？ ？ ！ 提示了、主动发现资源存在？ ？ ！ 不必要的请求？ ？ 】。

6 SPDY消除web瓶颈了吗?
使用SPDY，内容端【即服务端？？】不必做什么特别改动，但 B、web服务器应该要改动。
web服务器也进行了 实验性质的 应用，但把 该技术导入实际的 web网站缺 进展不太好！！
SPDY只是将 单个域名【IP地址，不是可以虚拟物理主机？？ 域名不等于 IP吧？？！】的通信多路复用！！

7 全双工通信的 —— WebSocket【“应该是 HTML5技术 的东西”一套 新协议 以及 API。可用于 聊天室的 实现】
WebSocket出现的背景：
Ajax【局部刷新】、Comet【"伪服务端推送，原理就是 将响应挂起，有资源真正更新了才响应回去"】可以提升 web的浏览速度。
SPDY虽有较大幅度提升，但是它不是新协议，还是借助了 HTTP协议， 所以提升也有限！！
但是它们都使用了 HTTP协议，就无法彻底解决 瓶颈问题 。
websocket正是为了解决这些问题的 一套新协议 及 API。

websocket【web浏览器 和 web服务器 之间的全双工通信标准。】的设计与功能【目的是为了 解决 Ajax 和 Comet里XMLHttpRequest附带的缺陷所引起的问题。 websocket通信过程可互相发送 JSON、XML、HTML、图片等 任意格式的 数据？？ 微前端可以使用？？ 发送 HTML？？】：

推送功能：服务器可直接发送数据，不必等 客户端的请求。

减少通信量：基本一直保持连接状态【啥原理保持连接消 就没有 Comet高了？？】。且websocket的首部信息很少【？？ 因为基本上通信的数据类型等也就那些、相对比较规范化了，不需要太多首部了？？！】

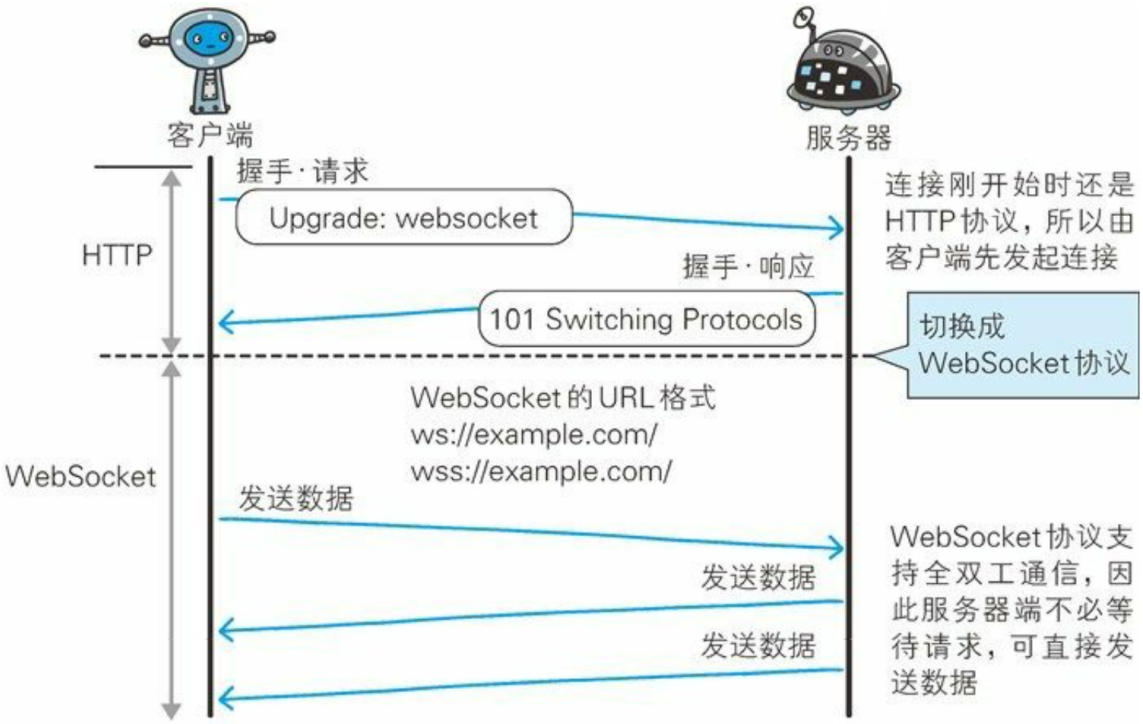
.2 为了实现 WebSocket通信，在HTTP连接建立之后，需要完成一次 “握手” 的步骤：
握手·请求【Upgrade: websocket、Connection: Upgrade。告知服务器通信协议发生了变化，不再是 HTTP通信了。】

握手·响应【对于之前的请求，返回状态码 101 交换协议 的响应】

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

成功握手之前， 还是HTTP通信，所以还由 客户端发起连接。
握手成功之后、WebSocket也连接上了，通信 不再使用HTTP 的数据帧，而采用

WebSocket独立的数据帧！！



.2 WebSocket API的使用【每隔50ms 发送一次数据的实例】

```
var socket = new WebSocket('ws://game.example.com:12010/
socket.onopen = function () {
  setInterval(function() {
    if (socket.bufferedAmount == 0)
      socket.send(getUpdateData());
  }, 50);
};
```

8 HTTP/2.0

.1 HTTP/2.0出现的背景

目前主流的 1.1标准，自 1999年 发布的RFC2616之后再未进行过修订。SPDY【仍使用了HTTP协议】 和 WebSocket【与HTTP协议并列，是H5的新协议以及API！！】等技术纷纷出现，很难断言 1.1 仍适用于 当下的web协议！！

.2 2.0的目标：

改善用户 在使用web时 的速度体验。

.3 【实现方法】由于基本上会先通过HTTP/1.1 与TCP连接，现在我们以下面的协议为基

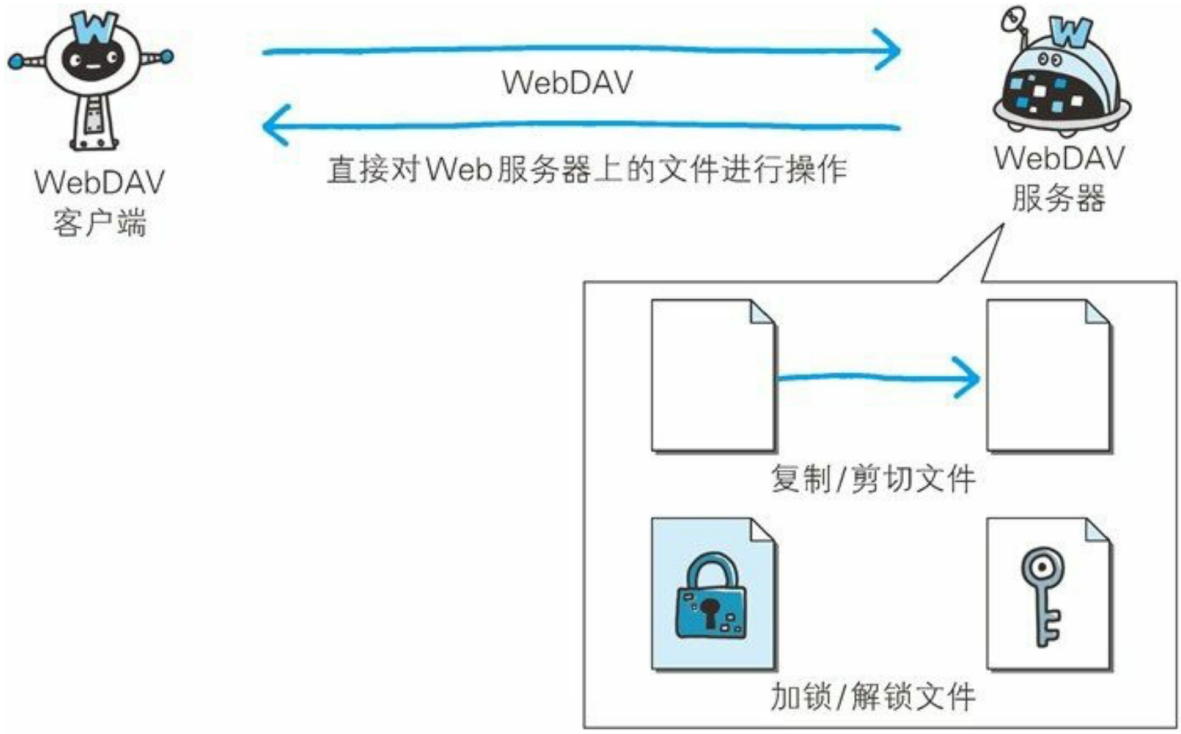
础，探讨一下它们的实现方法：
SPDY
HTTP Speed + Molibity
Network-Friendly HTTP Upgrade

.7 2.0主要围绕下面的7项技术进行讨论【首部压缩、多路复用、客户端拉拽 / 服务器推送、流量控制、协商、WebSocket、？？ TLS义务化？】

压缩	**SPDY、Friendly**
多路复用	SPDY
TLS 义务化	Speed＋ Mobility
协商	Speed＋ Mobility，Friendly
客户端拉曳（Client Pull）/服务器推送（Server Push）	Speed＋ Mobility
流量控制	SPDY
WebSocket	Speed＋ Mobility

九 Web服务器管理文件 的 WebDAV
1 webDAV（基于万维网的 分布式创作 和 版本控制）是一个 可对web服务器上的内容直接进行文件复制、编辑等操作 的分布式文件系统。
除了创建、删除文件等基本功能，它还具备文件创建者管理、文件比那几过程中禁止其他用户内容覆盖的加锁功能，以及对文件内容修改的 版本控制 功能。

【感觉 webDAV自带 PUT、DELETE方法（安全性！！） 和 “FTP协议”，直接能够 操作服务器上的文件； 且有 “版本控制”功能。】
Tip: 使用 HTTP/1.1 中的 PUT、DELETE方法，就可以对服务器上的文件进行创建和删除操作，但出于安全性、便捷性等考虑一般不用他们。



2 webDV扩展的概念
集合：是一种统一管理多个资源的概念。以 集合为单位 进行各种操作【？？？】
资源【文件 或 集合】：把 文件 或 集合 称为资源。
属性：定义资源的属性【定义以“名称=值”的格式执行】
锁：把文件 设为无法编辑状态。多人同时编辑时，可以防止同一时间内进行内容的写入。

3 webDV内新增的方法 和 状态码。
webDV为了实现 远程文件管理， 向 HTTP/1.1 中追加了一下这些方法。
PROPFIND：获取属性【find】
PROPPATCH:修改属性【patch】
MKCOL：创建集合【mk】
COPY：复制资源【文件 或 集合】及属性
MOVE：移动资源
LOCK：资源加锁
UNLOCK：资源解锁

新增的状态码【在原有HTTP状态码上进行的增加】
102 Procssing: 可正常处理请求，但目前是 处理中的状态。
207 Multi-Status: 存在多种状态【？？？】
422 Unprocessible Entity: 格式正确，内容有误
423 Lockd: 资源已被加锁。

424 Failed Dependency: 处理与某请求关联的请求失败，因此不在维持依赖关系。
507 Insufficient Storage： 保存空间不足

PROPFIND请求：

```
PROPFIND /file HTTP/1.1
Host: www.example.com
Content-Type: application/xml; charset="utf-8"
Content-Length: 219

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop xmlns:R="http://ns.example.com/boxschema/">
    <R:bigbox/>
    <R:author/>
    <R:DingALing/>
    <R:Random/>
  </D:prop>
</D:propfind>
```

PROPFIND响应：

```
HTTP/1.1 207 Multi-Status
Content-Type: application/xml; charset="utf-8"
Content-Length: 831

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response xmlns:R="http://ns.example.com/boxschema/"
    <D:href>http://www.example.com/file</D:href>
    <D:propstat>
      <D:prop>
        <R:bigbox>
        <R:BoxType>Box type A</R:BoxType>
        </R:bigbox>
```

```
        <R:author>
        <R:Name>J.J. Johnson</R:Name>
        </R:author>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
    <D:propstat>
      <D:prop><R:DingALing/><R:Random/></D:prop>
      <D:status>HTTP/1.1 403 Forbidden</D:status>
      <D:responsedescription> The user does not have acc
      </D:responsedescription>
    </D:propstat>
  </D:response>
  <D:responsedescription> There has been an access viola
  </D:responsedescription>
</D:multistatus>
```

十 为啥HTTP协议手中如此广泛？？

1【这与防火墙的设置有莫大联系】防火墙的基本功能： 禁止非指定 协议 和 端口号 的数据包通过。

因此使用写协议 或 端口号则必须修改防火墙的设置！！

2 互联网上，使用率最高的当属 web 。

不管是否具备访问 FTP 和 SSH的权限，一般公司都会开放对 web的访问。

web是基于HTTP协议运作的，因此在构建 web服务器 或 访问web站点时，须事先设置防火墙 HTTP【80/tcp】 HTTPS【443/tcp】的权限。

HTTP简单导入的优势，许多公司、组织已设定权限将 HTTP 作为通信环境，因此无需再修改防火墙的设定。

3 作为HTTP客户端的浏览器已相当普遍，HTTP服务器的数量也很多了。

且 HTTP本身 就是优秀的应用【HTTP是出于应用层的】。

完