mdn web docs

# A typical HTTP session

In client-server protocols, like HTTP, sessions consist of three phases:

1. The client establishes a TCP connection (or the appropriate connection if the transport layer is not TCP).
2. The client sends its request, and waits for the answer.
3. The server processes the request, sending back its answer, providing a status code and appropriate data.

As of HTTP/1.1, the connection is no longer closed after completing the third phase, and the client is now granted a further request: this means the second and third phases can now be performed any number of times.

## Establishing a connection

In client-server protocols, it is the client which establishes the connection. Opening a connection in HTTP means initiating a connection in the underlying transport layer, usually this is TCP.

With TCP the default port, for an HTTP server on a computer, is port 80. Other ports can also be used, like 8000 or 8080. The URL of a page to fetch contains both the domain name, and the port number, though the latter can be omitted if it is 80. See Identifying resources on the Web for more details.

> **Note:** The client-server model does not allow the server to send data to the client without an explicit request for it. However, various Web APIs enable

> this use case, including the Push API, Server-sent events, and the WebSockets API.

## Sending a client request

Once the connection is established, the user-agent can send the request (a user-agent is typically a web browser, but can be anything else, a crawler, for example). A client request consists of text directives, separated by CRLF (carriage return, followed by line feed), divided into three blocks:

1. The first line contains a request method followed by its parameters:
   - the path of the document, as an absolute URL without the protocol or domain name
   - the HTTP protocol version
2. Subsequent lines represent an HTTP header, giving the server information about what type of data is appropriate (for example, what language, what MIME types), or other data altering its behavior (for example, not sending an answer if it is already cached). These HTTP headers form a block which ends with an empty line.
3. The final block is an optional data block, which may contain further data mainly used by the POST method.

### Example requests

Fetching the root page of developer.mozilla.org, ( `https://developer.mozilla.org/` ), and telling the server that the user-agent would prefer the page in French, if possible:

HTTP

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

Observe that final empty line, this separates the data block from the header block. As there is no `Content-Length` provided in an HTTP header, this data block is presented empty, marking the end of the headers, allowing the server to process the request the moment it receives this empty line.

For example, sending the result of a form:

```http
POST /contact_form.php HTTP/1.1
Host: developer.mozilla.org
Content-Length: 64
Content-Type: application/x-www-form-urlencoded

name=Joe%20User&request=Send%20me%20one%20of%20your%20catalogue
```

### Request methods

HTTP defines a set of request methods indicating the desired action to be performed upon a resource. Although they can also be nouns, these requests methods are sometimes referred as HTTP verbs. The most common requests are `GET` and `POST`:

- The `GET` method requests a data representation of the specified resource. Requests using `GET` should only retrieve data.
- The `POST` method sends data to a server so it may change its state. This is the method often used for HTML Forms.

## Structure of a server response

After the connected agent has sent its request, the web server processes it, and ultimately returns a response. Similar to a client request, a server response is formed of text directives, separated by CRLF, though divided into three blocks:

1. The first line, the *status line*, consists of an acknowledgment of the HTTP version used, followed by a response status code (and its brief meaning in human-readable text).

2. Subsequent lines represent specific HTTP headers, giving the client information about the data sent (for example, type, data size, compression algorithm used, hints about caching). Similarly to the block of HTTP headers for a client request, these HTTP headers form a block ending with an empty line.
3. The final block is a data block, which contains the optional data.

## Example responses

Successful web page response:

HTTP

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 55743
Connection: keep-alive
Cache-Control: s-maxage=300, public, max-age=0
Content-Language: en-US
Date: Thu, 06 Dec 2018 17:37:18 GMT
ETag: "2e77ad1dc6ab0b53a2996dfd4653c1c3"
Server: meinheld/0.6.1
Strict-Transport-Security: max-age=63072000
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Vary: Accept-Encoding,Cookie
Age: 7

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>A simple webpage</title>
</head>
<body>
  <h1>Simple HTML webpage</h1>
  <p>Hello, world!</p>
</body>
</html>
```

Notification that the requested resource has permanently moved:

```http
HTTP/1.1 301 Moved Permanently
Server: Apache/2.4.37 (Red Hat)
Content-Type: text/html; charset=utf-8
Date: Thu, 06 Dec 2018 17:33:08 GMT
Location: https://developer.mozilla.org/ (this is the new link to the resource; it is expected that the user-agent will fetch it)
Keep-Alive: timeout=15, max=98
Accept-Ranges: bytes
Via: Moz-Cache-zlb05
Connection: Keep-Alive
Content-Length: 325 (the content contains a default page to display if the user-agent is not able to follow the link)

<!DOCTYPE html>… (contains a site-customized page helping the user to find the missing resource)
```

Notification that the requested resource doesn't exist:

```http
HTTP/1.1 404 Not Found
Content-Type: text/html; charset=utf-8
Content-Length: 38217
Connection: keep-alive
Cache-Control: no-cache, no-store, must-revalidate, max-age=0
Content-Language: en-US
Date: Thu, 06 Dec 2018 17:35:13 GMT
Expires: Thu, 06 Dec 2018 17:35:13 GMT
Server: meinheld/0.6.1
Strict-Transport-Security: max-age=63072000
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Vary: Accept-Encoding,Cookie
X-Cache: Error from cloudfront
```

```
<!DOCTYPE html>… (contains a site-customized page helping the user to find the missing resource)
```

### Response status codes

HTTP response status codes indicate if a specific HTTP request has been successfully completed. Responses are grouped into five classes: informational responses, successful responses, redirects, client errors, and server errors.

- `200` : OK. The request has succeeded.
- `301` : Moved Permanently. This response code means that the URI of requested resource has been changed.
- `404` : Not Found. The server cannot find the requested resource.

## See also

- Identifying resources on the Web
- HTTP headers
- HTTP request methods
- HTTP response status codes