Web and HTTP

- HTTP Request Message
- HTTP Response Message
- Cookies

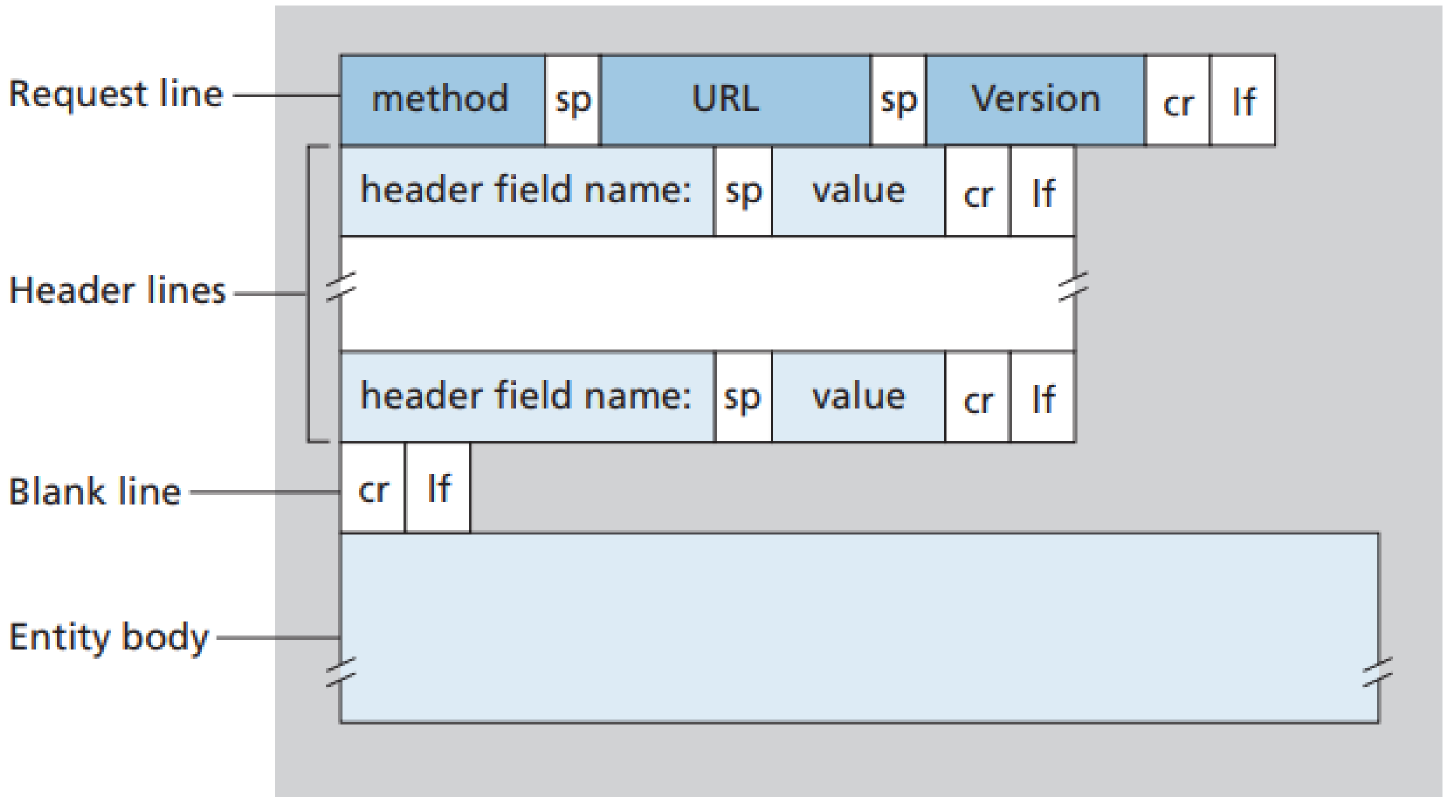
HTTP Requests and Responses

- Request and responses have two parts: headers and content
- If you type a URL into your browser, the browser creates an HTTP request and sends it to a server
- The server finds the requested file and sends it back in an HTTP response
- The response headers describe things like *the type of web server, the file type of the response, the length of the response and other info*
- The response content is the file data

Request and Response format

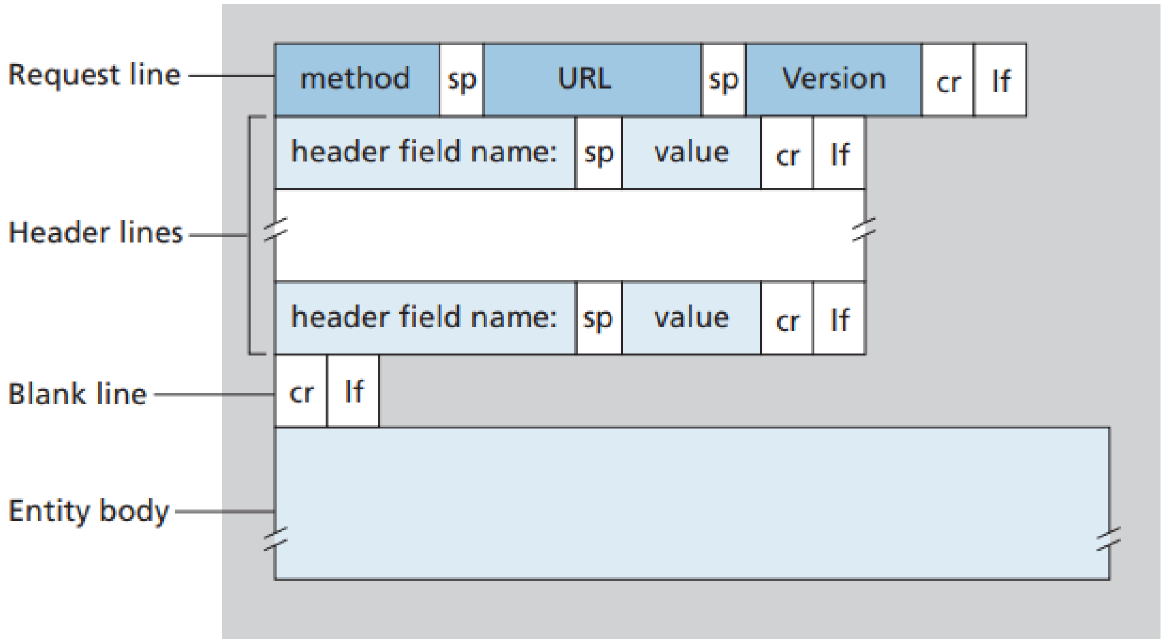
- The format of the request and response messages are similar and both kinds of messages consist of:
  - an initial line,
  - zero or more header lines,
  - a blank line (i.e. a CRLF by itself), and
  - an optional message body (e.g. a file, or query data, or query output).

HTTP request message: general format



HTTP request message: general format

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```



Initial Request Line

- A request line has three parts, separated by spaces:
  - a **method** name,
  - the **local path** of the requested resource
  - and the **version of HTTP** being used
- A typical request line is:
  - GET /path/to/file/index.html HTTP/1.0
- GET is the most common HTTP method
- The path is the part of the URL after the host name
- The HTTP version always takes the form **"HTTP/x.x"**, uppercase – current version is 3.0

HTTP Methods

- HTTP supports several different request commands, called HTTP methods
- Every HTTP request message has a method, which tells the server what action to perform, such as
  - fetch a web page,
  - run a gateway program
  - delete a file, etc.

HTTP Methods

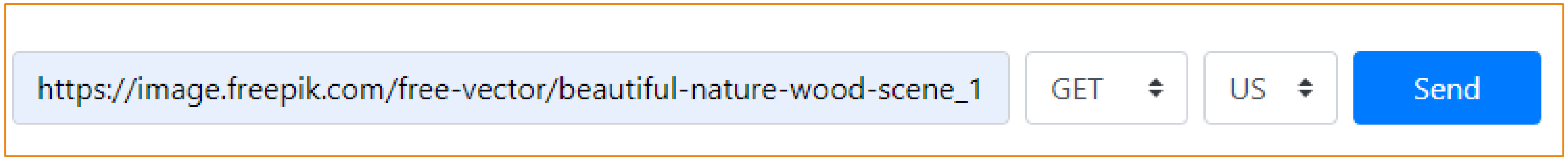
Most common methods

- GET: Used for getting information from a web server
- POST: Used for submitting data to the web server and potentially creating new records
- PUT: Used for submitting data to a web server to update information
- DELETE: Used for deleting information/records from a web server

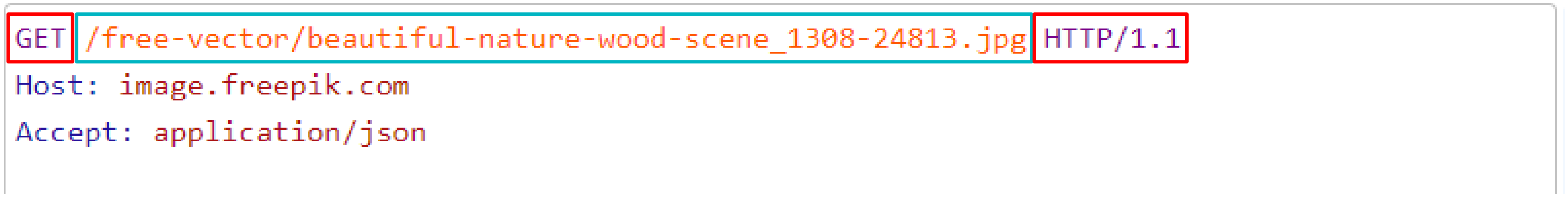
More details: https://www.w3schools.com/tags/ref_httpmethods.asp

HTTP Request Example

https://reqbin.com/

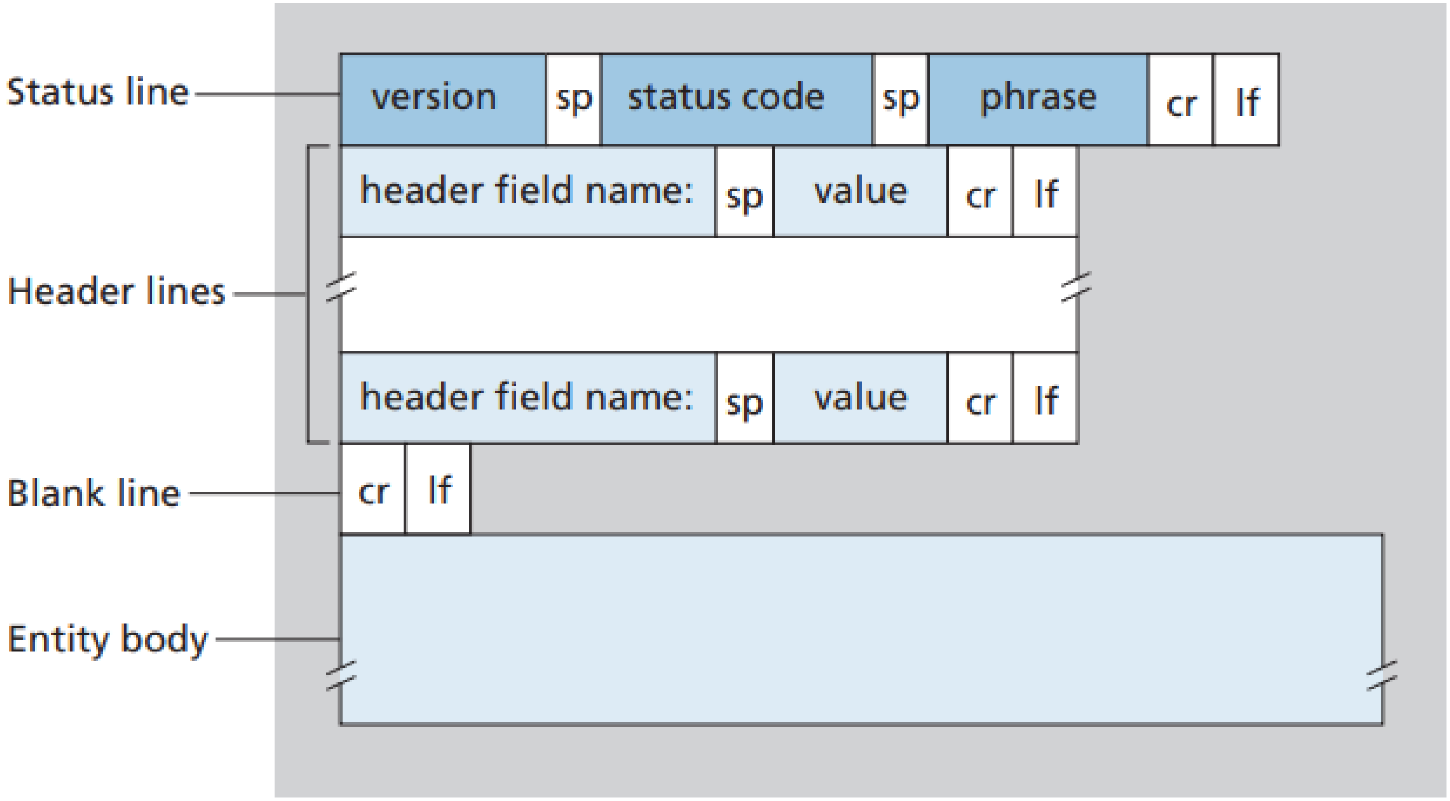

https://image.freepik.com/free-vector/beautiful-nature-wood-scene_1
GET
US
Send

Request Header



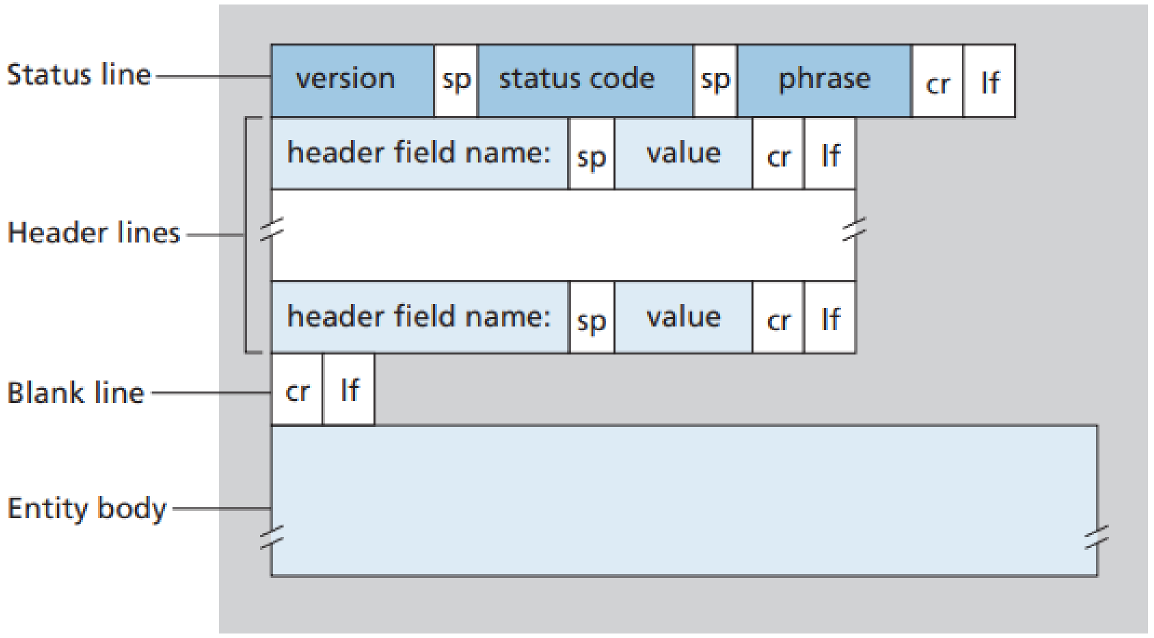
```
GET /free-vector/beautiful-nature-wood-scene_1308-24813.jpg HTTP/1.1
Host: image.freepik.com
Accept: application/json
```

HTTP response message: general format



HTTP response message: general format

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```



Initial Response Line

- The initial response line, called the status line, also has three parts separated by spaces:
  - the **HTTP version**,
  - a **response status code** that gives the result of the request,
  - and an English **reason phrase** describing the status code
- Examples:
  - HTTP/1.1 200 OK
  - HTTP/1.1 404 Not Found
- The HTTP version is in the same format as in the request line, **"HTTP/x.x"**

HTTP Status Codes

- Every HTTP response message comes back with a status code, a **three-digit number code** that tells the client
  - If the request succeeded, or
  - If other actions are required
- HTTP also sends an explanatory text "reason phrase" followed by each status code
- The phrase is included only for descriptive purposes; the numeric code is used for all processing

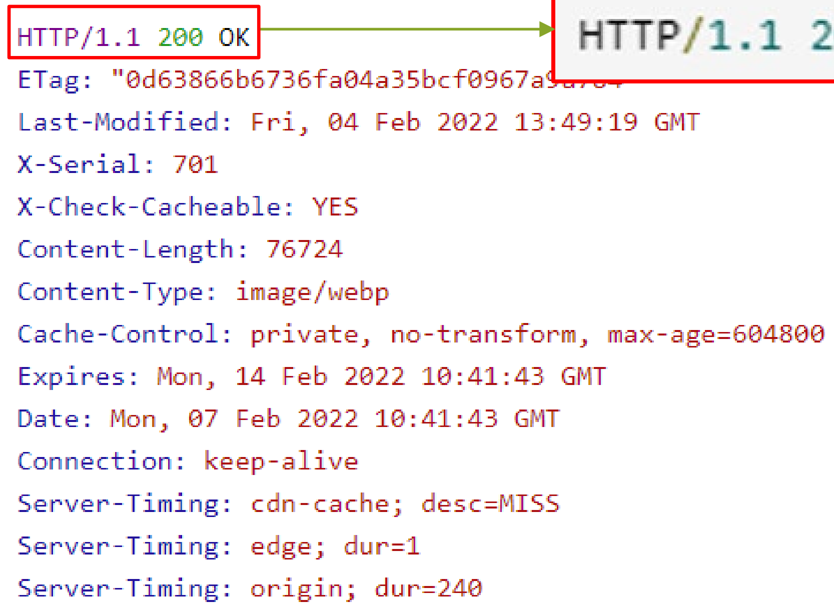
Code	Meaning	Examples
1XX	Information	100 = server agrees to handle client's request
2XX	Success	200 = request succeeded; 204 = no content present
3XX	Redirection	301 = page moved; 304 = cached page still valid
4XX	Client error	403 =forbidden page; 404 = page not found
5XX	Server error	500 = internal server error; 503 = try again later

More details: https://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.1.1

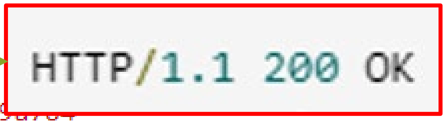
HTTP Response Example

https://reqbin.com/

Response Header

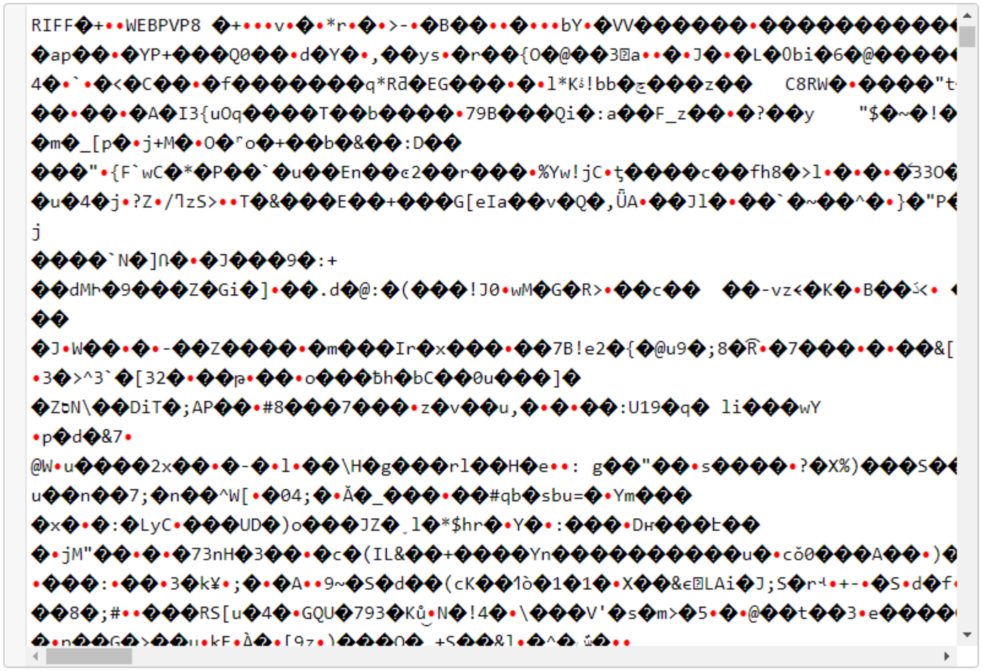


```
HTTP/1.1 200 OK
ETag: "0d63866b6736fa04a35bcf0967a
Last-Modified: Fri, 04 Feb 2022 13:49:19 GMT
X-Serial: 701
X-Check-Cacheable: YES
Content-Length: 76724
Content-Type: image/webp
Cache-Control: private, no-transform, max-age=604800
Expires: Mon, 14 Feb 2022 10:41:43 GMT
Date: Mon, 07 Feb 2022 10:41:43 GMT
Connection: keep-alive
Server-Timing: cdn-cache; desc=MISS
Server-Timing: edge; dur=1
Server-Timing: origin; dur=240
```



```
HTTP/1.1 200 OK
```

Response Body



RIFF�+•••WEBPVP8 �+•••v•�•*r•�•>-•�B��••�•••bY•�VV������•����������
�ap��•�YP+���Q0��•d�Y�•,��ys•�r��{O�@��3a••�•J�•�L�0bi�6�@����
4�•`•�<�C��•�f�������q*R�EG���•�•1*K!bb����z��   C8RW�•����"t
��•��•�A�I3{uOq����T��b����•79B���Qi�:a��F_z��•�?��y   "$�~�!�
�m�_[p�•j+M�•O�o�+��b&��:D��
���"•{F`wC�*�P��`�u��En��2��r���•%Yw!jC•ţ����c��fh8�>l•�•�•�330�
�u�4�j•?Z•/7zS>••T�&���E��+���G[eIa��v�Q�,ŨA•��Jl�•��`�~��^�•}�"P�
j
����`N�]n�•�J���9�:+
��dMh�9���Z�Gi�]•��.d�@:�(���!J0•wM�G�R>•��c�� ��-vz<�K�•B��<• �
��
�J•W��•�•-��Z����•�m���Ir�x���•��7B!e2�{�@u9�;8�R•�7���•�•��&[
•3�>^3`�[32�•��p•��•o���bh�bC��0u���]�
�ZbN\��DiT�;AP��•#8���7���•z�v��u,�•�•��:U19�q� li���wY
•p�d�&7•
@W•u����2x��•�-�•l•��\H�g���rl��H�e••: g��"��•s����•?�X%)���S�
u��n��7;�n��^W[•�04;�•Ă�_���•��#qb�sbu=�•Ym���
�x�•�:�LyC•���UD�)o���JZ�_l�*$hr�•Y�•:���•DH���t��
�•jM"��•�•�73nH�3��•�c�(IL&��+����Yn����������u�•cŏ0���A��•)�
•���:•��•3�k¥•;�•�A••9~�S�d��(cK��1ò�1�1�•X��&€LAi�J;S�r••+-•�S•d�f
��8�;#••���RS[u�4�•GQU�793�Kû•N�!4�•\���V'�s�m>�5•�•@��t��3•e����

Header Lines

- Zero or more header field follow the start line
- Header lines provide information about the request or response, or about the object sent in the message body
- Each header field consists of name and a value, separated by colon (:) for easy parsing
- The header name is not case-sensitive
- The headers end with a blank line (end in CRLF)
- Header lines beginning with space or tab are actually part of the previous header line, folded into multiple lines for easy reading
- Examples: following two headers are equivalent

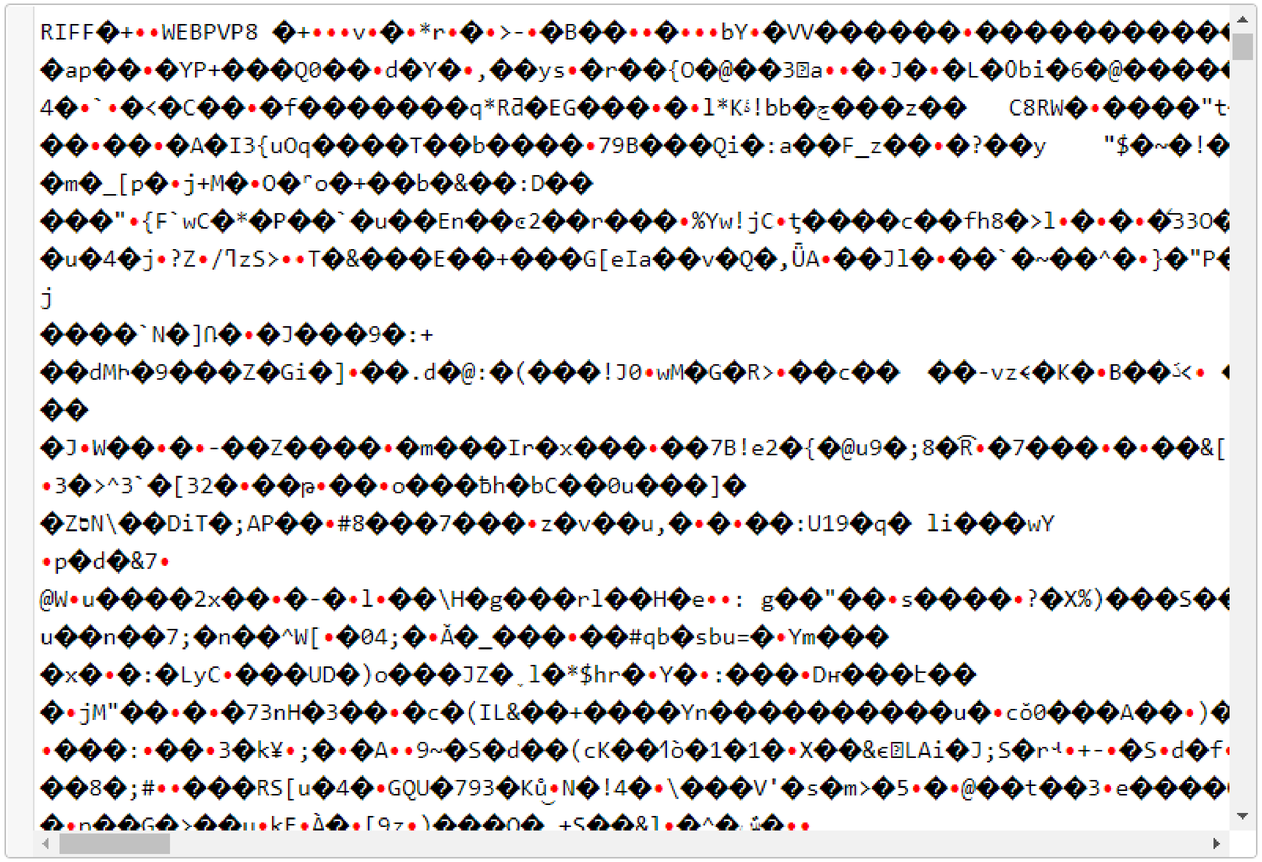
```
Header1: some-long-value-1a, some-long-value-1b

HEADER1: some-long-value-1a,
      some-long-value-1b
```

Message Body

- An HTTP message may have a body of data sent after the header lines
- In a **response**: the requested resource returned to the client
    - or perhaps explanatory text if there's an error
- In a **request**: user-entered data or uploaded files are sent to the server
- If an HTTP message includes a body, there are usually header lines in the message that describe the body:
    - The Content-Type: header gives the **MIME-type** of the data in the body, such as text/html or image/gif
    - The Content-Length: header gives the number of bytes in the body

Message Body: Example

Message Body/Content



```
RIFF�+••WEBPVP8 �+•••v•�•*r•�•>-•�B��••�•••bY•�VV������•����������
�ap��•�YP+���Q0��•d�Y�•,��ys•�r��{O�@��3⊡a••�•J�•�L�0bi�6�@�����
4�•`•�<�C��•�f�������q*Rd�EG���•�•l*Kأ!bb�ꞓ���z��   C8RW�•����"t
��•��•�A�I3{uOq����T��b����•79B���Qi�:a��F_z��•�?��y    "$�~�!�
�m�_[p�•j+M�•O�ᒥo�+��b�&��:D��
���"•{F`wC�*�P��`�u��En��ꞓ2��r���•%Yw!jC•ţ����c��fh8�>l•�•�•�330
�u�4�j•?Z•/ˡzS>••T�&���E��+���G[eIa��v�Q�,ŰA•��J1�•��`�~��^�•}�"P
j
����`N�]ꞃ�•�J����9�:+
��dMh�9���Z�Gi�]•��.d�@:�(���!J0•wM�G�R>•��c��  ��-vz<�K�•B��ݝ<•
��
�J•W��•�•-��Z����•�m���Ir�x���•��7B!e2�{�@u9�;8�R̂•�7���•�•��&[
•3�>^3`�[32�•��ꝑ•��•o���ђh�bC��0u���]�
�ZꝺN\��DiT�;AP��•#8���7���•z�v��u,�•�•��:U19�q� li���wY
•p�d�&7•
@W•u����2x��•�-�•l•��\H�g���rl��H�e••: g��"��•s����•?�X%)���S�
u��n��7;�n��^W[•�04;�•Ă�_���•��#qb�sbu=�•Ym���
�x�•�:�LyC•���UD�)o���JZ�ˌl�*$hr�•Y�•:���•DH���Ŀ��
�•jM"��•�•�73nH�3��•�c�(IL&��+����Yn���������u�•cŏ0���A��•)�
•���:•��•3�k¥•;�•�A••9~�S�d��(cK��1ò�1�1�•X��&ɛ⊡LAi�J;S�rᆨ•+-•�S•d�f
��8�;#••���RS[u�4�•GQU�793�Kű•N�!4�•\���V'�s�m>�5•�•@��t��3•e����
```

MIME types

- HTTP tags the object being transported with a data format label called a MIME types
- **MIME (Multipurpose Internet Mail Extensions)**
  - Originally designed to solve problems in moving multimedia message between different email systems
  - MIME worked so well for email that HTTP adopted it to describe and label its own multimedia content
- An **Internet media type** (originally **MIME**) is a two-part identifier for file formats on the Internet
  - A media type is composed of at least two parts: a *type*, a *subtype*, and one or more optional parameters
  - Example: `Content-Type:text/html; charset=UTF-8`

More details: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types

MIME types

- Web servers attach a MIME type to all HTTP object data
- When a web browser get an object back from a server, it display/play the object according to the associated MIME types
  - display image files,
  - parse and format HTML files,
  - play audio files,
  - launch external plug-in software,
  - Or launch external helping software

Statelessness and Cookies

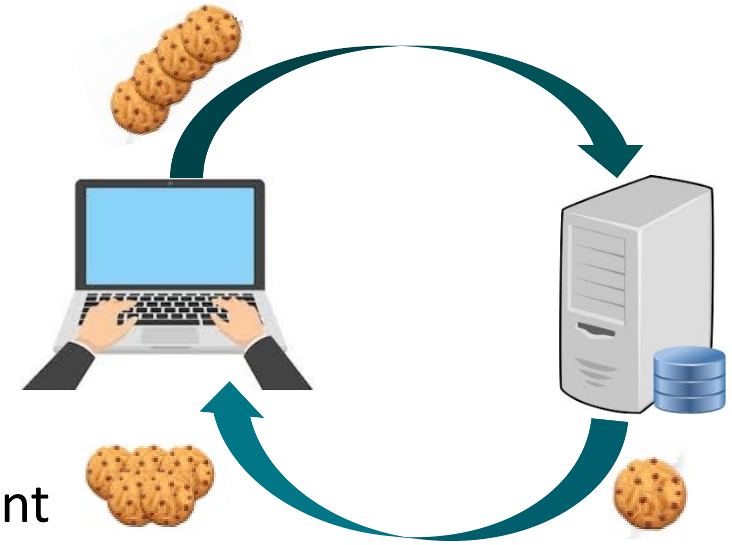
- **HTTP** is stateless
  - Servers would not remember from where the requests are coming
  - Requests from same user do not necessarily come in adjacent requests
  - There may be several other requests in the middle
  - Remembering states would help to provide customised responses

Statelessness and Cookies

- Storing state somewhere
  - **Server side:**
    - makes servers really complicated;
    - state per client!
  - **Client side:**
    - Server puts little notes on the client side;
    - When client submits the next request, it also (unknowingly) submits these little notes;
    - Server reads the notes, remembers who the client is – or obtain the info to generate a page that logically follows

Cookies

- Add state management
- Information saved by the browser on the client
- Sent back to the server inside the http data
- Cookies can be client side or server side
- Typical Uses of Cookies – storing information client side
    - Identifying a user during an e-commerce session
    - Avoiding username and password
    - Customising a site
    - Focusing advertising
- Positive side: Site will remember who you are



Some problem with Cookies

- **The problem is privacy, not security**
  - Servers can remember your previous actions
  - If you give out personal information, servers can link that information to your previous actions
  - Servers can share cookie information through use of a cooperating third party like doubleclick.net
  - **Poorly designed** sites **store sensitive information** like credit card numbers directly in cookie
  - JavaScript bugs let hostile sites steal cookies (old browsers)

Acknowledgements and References

- James F. Kurose and Keith W. Ross. Computer Networking: A Top-Down Approach, 8th edition, Pearson, 2021.

- Some parts of the content are adapted from:
  - Professor F. Ricci’s lecture on HTTP