HTTP

2020/21 COMP3322 Modern Technologies on WWW

Contents

Core Features of the Web

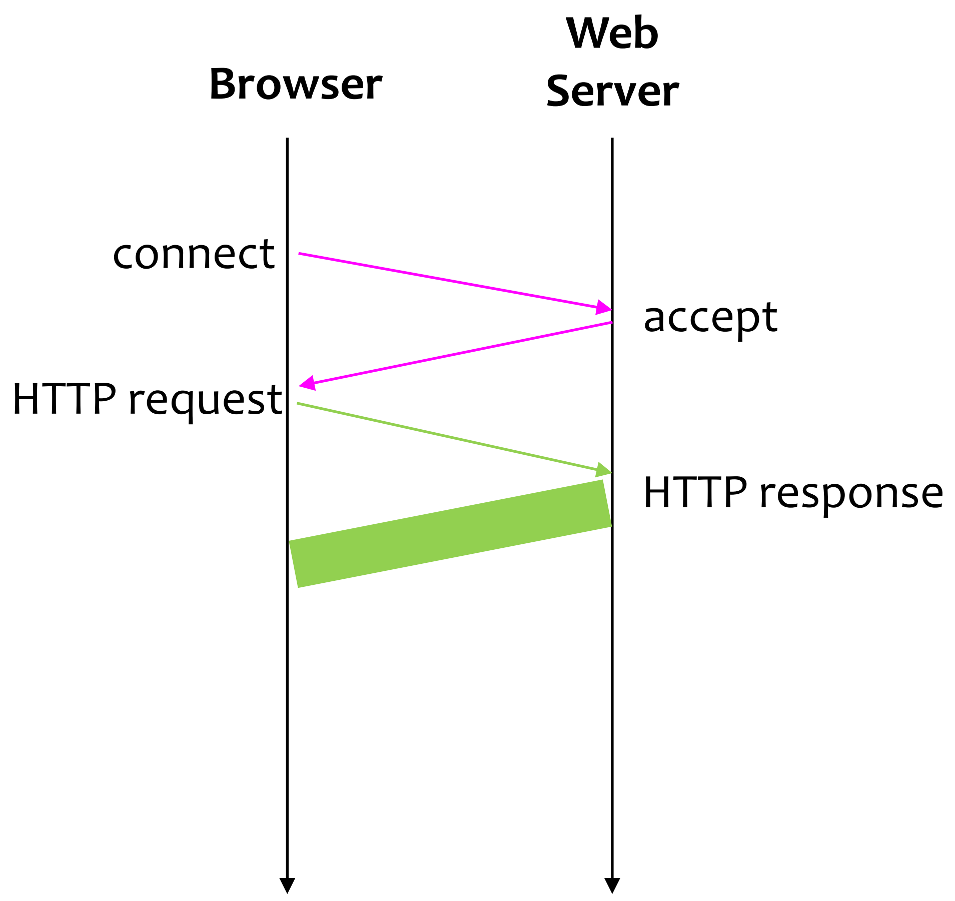
1. A **URL** to uniquely identify a resource on the WWW.
2. At two ends, there are two programs interacting by means of the **HTTP protocol**, which is one form of **client-server communication**.
   - The browser program which makes HTTP requests from URLs and that can display the HTML it receives.
   - The web server software program that responds to HTTP requests.
3. The HTTP protocol to describe how **requests and responses** operate.
4. **HTML** and **CSS** to publish documents.

HyperText Transfer Protocol (H T T P)

- HTTP is the foundation of data communication for the WWW.
- It is an **application layer** protocol that is sent over **TCP**.
- The protocol specifies **format and meaning** of messages exchanged between clients and servers.
- Each message has control information and message content presented in **plaintext format**, but it supports transmission of arbitrary binary data.
- Can download or upload data.

HyperText Transfer Protocol (H T T P)

- It follows a classical client-server communication model:
  - The client (a web browser) first **initiates a TCP connection** to the Web server.
  - After that, it sends a request to the connected server.
  - Then it waits for the response from the server.



Stateless Protocol

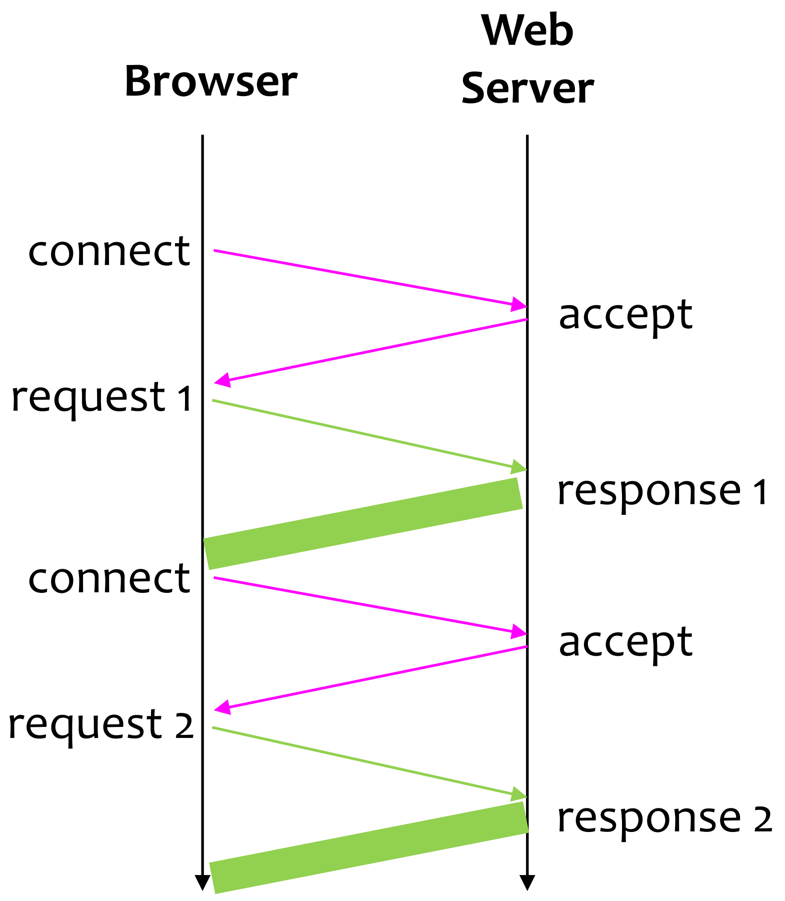
- HTTP is a stateless protocol.
  - This means that
    - each request is independent
    - every request is considered as a new request **without knowledge of previous activities**.
  - Server does not maintain information about the access history of the clients.
    - In the past, all web pages were static pages; this makes error recovery really simple.
- This creates difficulty for users attempting to interact with certain pages coherently, for example, using e-commerce shopping carts.

Evolution of HTTP

- HTTP/0.9 – 1991
- HTTP/1.0 – 1996
- HTTP/1.1 – 1997
- HTTP/2 – 2015
- HTTP/3 is coming

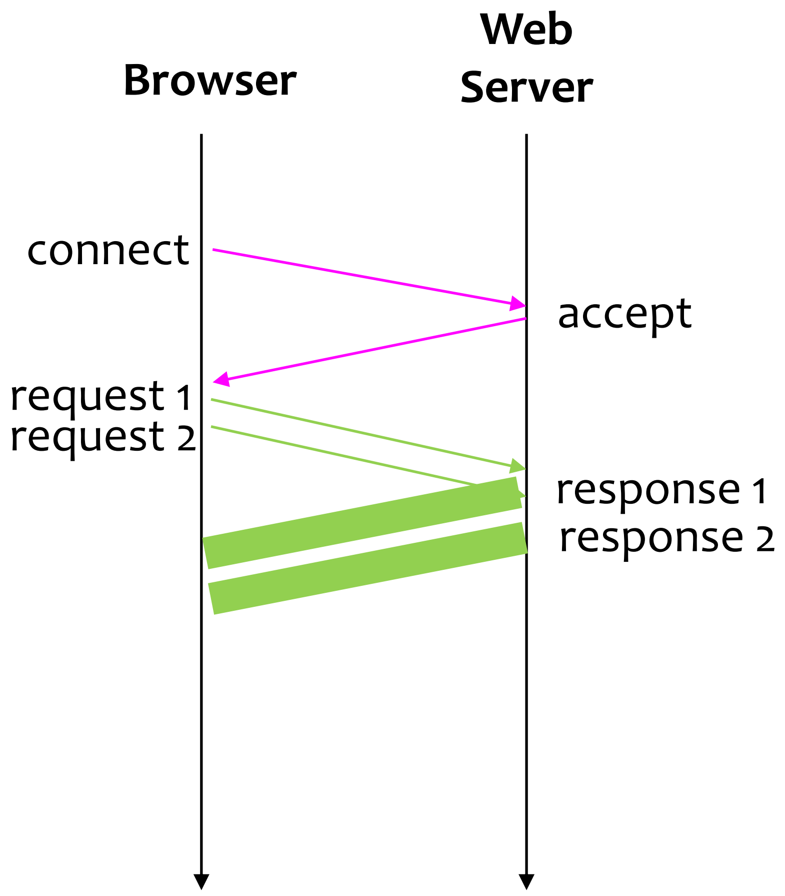
HTTP/1.0

- HTTP/0.9 was called the **one-line protocol** and only supported one type of request: the **GET** request.
- **HTTP headers** were introduced in HTTP/1.0
  - **New functionality** can even be introduced by adding a new header.
  - Made it **easy to extend** and experiment with new features.
- Added the **status code** info in the response messages.
- HTTP/1.0 has one major issue:
  - A new TCP connection is opened for each request/response exchange.
  - This affects the communication performance.



HTTP/1.1

- HTTP/1.1 is **still widely used** at the moment.
- To improve performance:
  - It introduced the concept of reusing the TCP connection (aka **Persistent Connection**) for **multiple** request/response exchanges.
  - It supports **pipelining requests**, i.e., sending a second request even before the first response is back.
  - Adding **caching mechanism** to reduce data traffic.
- Support **content negotiation** between the client and server.
  - Clients can specify the best suited representation of data for the user, e.g., language, encoding, image format, etc.

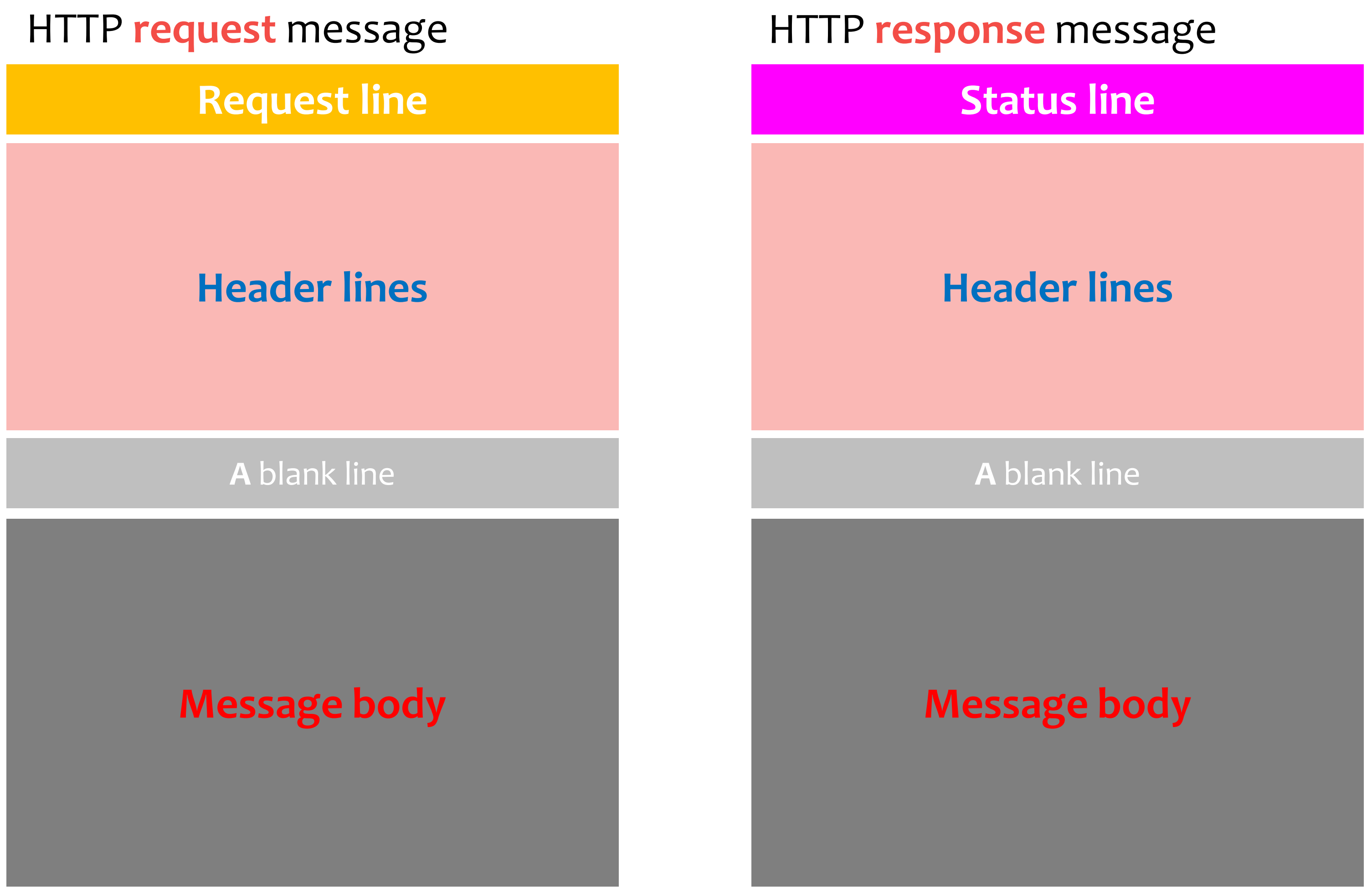


HTTP Messages

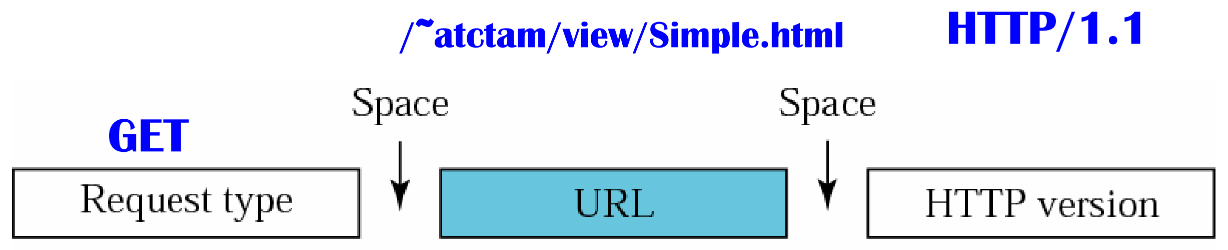
- There are two types of HTTP messages:
  - Request messages
  - Response messages
- More-or-less they are of the same message format.
  - HTTP messages are composed of textual information encoded in ASCII, and span over multiple lines.
  - Starts with a single **start-line**.
  - Follows by a set of **HTTP headers**.
  - **A blank line** indicating the end of header block.
  - An optional **message body** containing data associated with the request or the document associated with the response.

Ignore the binary format of HTTP/2, this is the view of a Web developer

HTTP Messages: General Format



HTTP Request



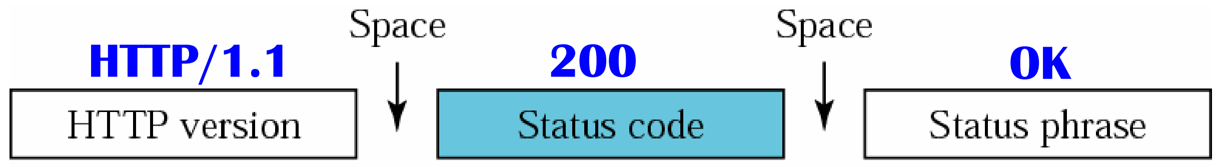
- Request line
  - Request type **defines the action** to be performed.
    - Common request types are: GET, POST, HEAD, PUT, and DELETE

GET	Request to read a Web page
HEAD	Request to read a Web page's header
POST	Append/add to a name resource (e.g., a Web page)
PUT	Request to store/update a Web page
DELETE	Request to remove a Web page/resource

  - URL gives the address of the request resource
    - Could be just the path, e.g., GET /~atctam/view/Simple.html HTTP/1.1
    - Could be the complete URL, e.g., GET http://i.cs.hku.hk/~atctam/view/Simple.html HTTP/1.1
      - Mostly used with GET when connected to proxy server.
  - HTTP version

HTTP Response

- Status line
  - HTTP version
  - Status code indicates success or failure of the request.
    - Common status code are 200, 302, 304 and 404.
  - Status phrase gives the textual description of the status code.



Status Code Format	Meaning	Description
1yy	Informational Message	Provides general information; does not indicate success or failure of a request.
2yy	Success	The method was received, understood and accepted by the server.
3yy	Redirection	The request did not fail outright, but additional action is needed before it can be successfully completed.
4yy	Client Error	The request was invalid, contained bad syntax or could not be completed for some other reason that the server believes was the client's fault.
5yy	Server Error	The request was valid but the server was unable to complete it due to a problem of its own.

HTTP Headers

- The basic structure of a header:
  - A case-insensitive string followed by a colon ‘:’ and a value whose format depends upon the header.
- There are four types of headers:
  - General headers, which apply to the message as a whole.
  - Entity headers, which apply to the message body.
  - Request headers, which provide more information about the resource to be fetched or about the client itself.
  - Response headers, which give additional information about the response.

https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

HTTP Headers

Header	Type	Description
Connection	General	Shows whether the connection should be closed or not
Date	General	Date and time the message was sent
Cache-Control	General	Controls who can cache the response, under which conditions, and for how long.
Accept	Request	Shows the media format the client can accept
Host	Request	Shows the host and port number of the resource being requested
Referrer	Request	Specifies the URL of the linked document
User-agent	Request	Identifies the client program
Server	Response	Information about the server
Age	Response	Shows the age of the document
Location	Response	Redirects the recipient to a location other than the Request-URI
Content-Encoding	Entity	How the content is encoded (e.g., gzip)
Content-Length	Entity	Shows the length of the document
Content-Type	Entity	The page's MIME type
Last-Modified	Entity	Time and date the page was last changed

Message Body

- HTTP Requests
  - Most of the request types, like GET or HEAD, do not have a body.
  - Often seen in the case with POST – containing HTML form data.
- HTTP Responses
  - Carries the resource requested by the client.
  - Usually consists of a single file defined by the two headers: Content-Type and Content-Length.

An Example Messages Exchange

```
GET /~atctam/view/Simple.html HTTP/1.1
Host: i.cs.hku.hk
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:79.0) Gecko/20100101 Firefox/79.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

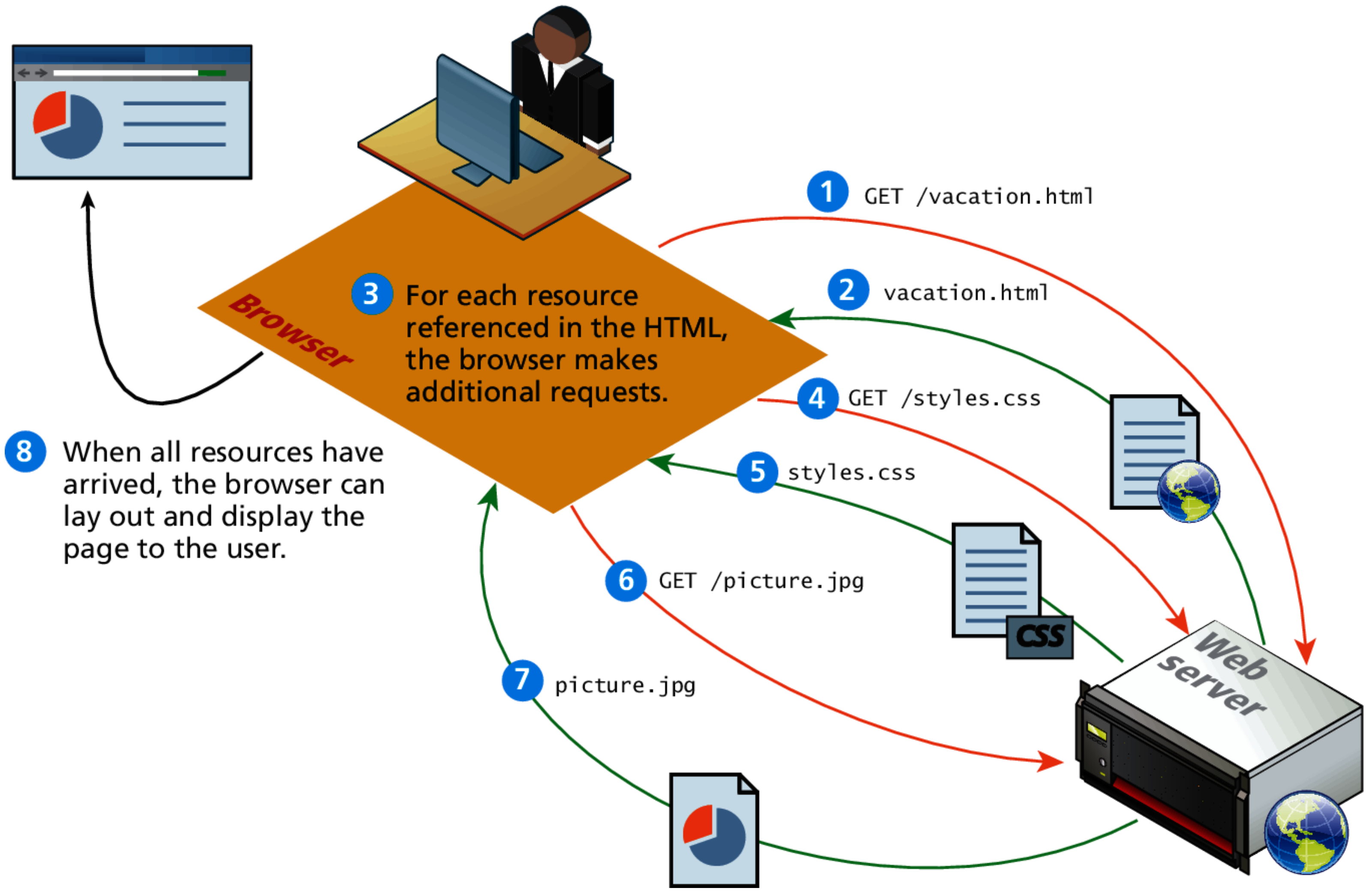
```
HTTP/1.1 200 OK
Date: Sat, 05 Sep 2020 03:00:56 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Wed, 18 Jul 2018 06:47:40 GMT
ETag: "a9-571407054bb07-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 136
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>My Sample HTML Page</title>
        </head>
        <body>
                <h1>This is an HTML Page</h1>
        </body>
</html>
```

Web Requests

- Most of the web users might be tempted to think of an entire page being returned in a single HTTP response, this is not in fact what happens.
- In reality the experience of seeing a single web page is facilitated by the client's browser which **requests the initial HTML page**, then parses the returned HTML to **find all the resources referenced** from within it, like images, style sheets and scripts, and further requests the server(s) for the files/resources.
- Only when all the files/resources have been retrieved is the page fully loaded for the user.

Browser parsing HTML and making subsequent requests



1 GET /vacation.html
2 vacation.html
Browser
3 For each resource referenced in the HTML, the browser makes additional requests.
4 GET /styles.css
5 styles.css
6 GET /picture.jpg
7 picture.jpg
8 When all resources have arrived, the browser can lay out and display the page to the user.
CSS
Web server

Web Caching

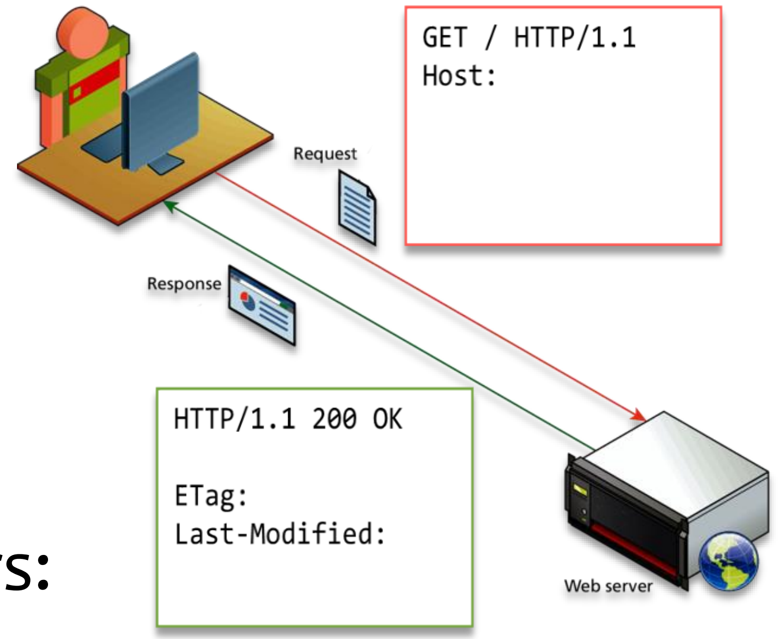
- Web caching is a technique that stores a copy of a given resource and serves it back when being requested in the future.
- Advantages:
  - Cache is "close" to client (e.g., in same network), thus **reduces the response time**.
  - **Decreases network traffic** to distant servers.
- Two kinds of caches:
  - Private browser caches
  - Proxy servers set up by the company/organization/network service provider.
    - Client sends all HTTP requests to proxy server
      - If the requested object is in web cache: proxy returns object to client
      - else proxy requests object from origin server, then returns object to client and caches the object.
    - Proxy acts as both client and server

Caching

- Cached contents may become stale.
- How HTTP determines that a resource is still “fresh”, not considered stale?
- There are a few response headers related to caching.
  - **Cache-Control** header (e.g., Cache-Control: max-age=300)
    - Specifies the number of seconds (e.g., 300) that this resource is considered to be fresh relative to the time of the request
  - **Expires** (e.g., Expires: Wed, 15 Dec 2019 07:30:00 GMT)
    - Specifies the date/time after which the response is considered stale.
  - After the resource is considered stale, **perform validation**.

Validation

- HTTP uses two response headers and a request header to perform validation.
- The origin server includes two response headers:
  - Etag - is an identifier for a specific version of a resource and is generated by the server whenever the resource is updated.
    - ETag: "33a64df551425fcc55e4d42a148795d9f25f89d4"
  - Last-Modified header - contains the date and time at which the origin server believes the resource was last modified.
    - Last-Modified: Wed, 21 Oct 2015 07:28:00 GMT

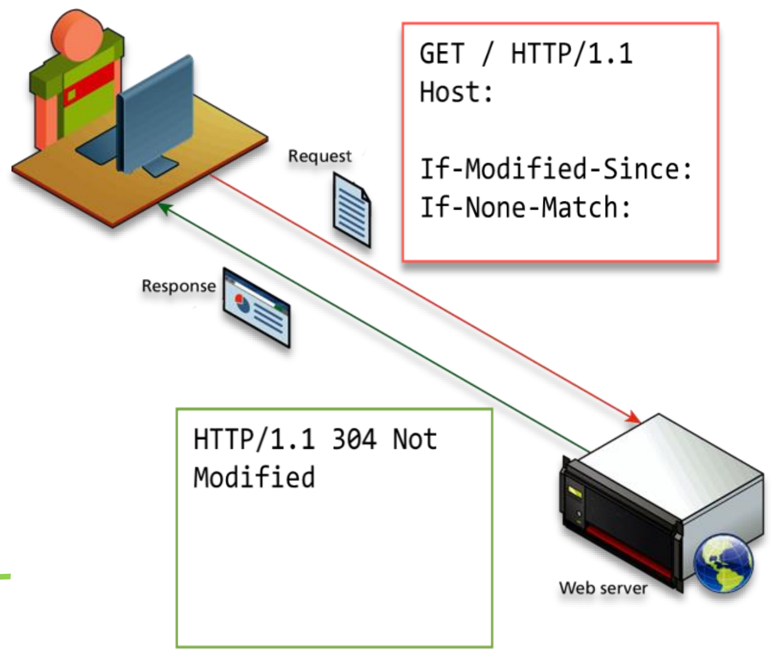


```
GET / HTTP/1.1
Host:
```

```
HTTP/1.1 200 OK

ETag:
Last-Modified:
```

Validation



```
GET / HTTP/1.1
Host:

If-Modified-Since:
If-None-Match:
```

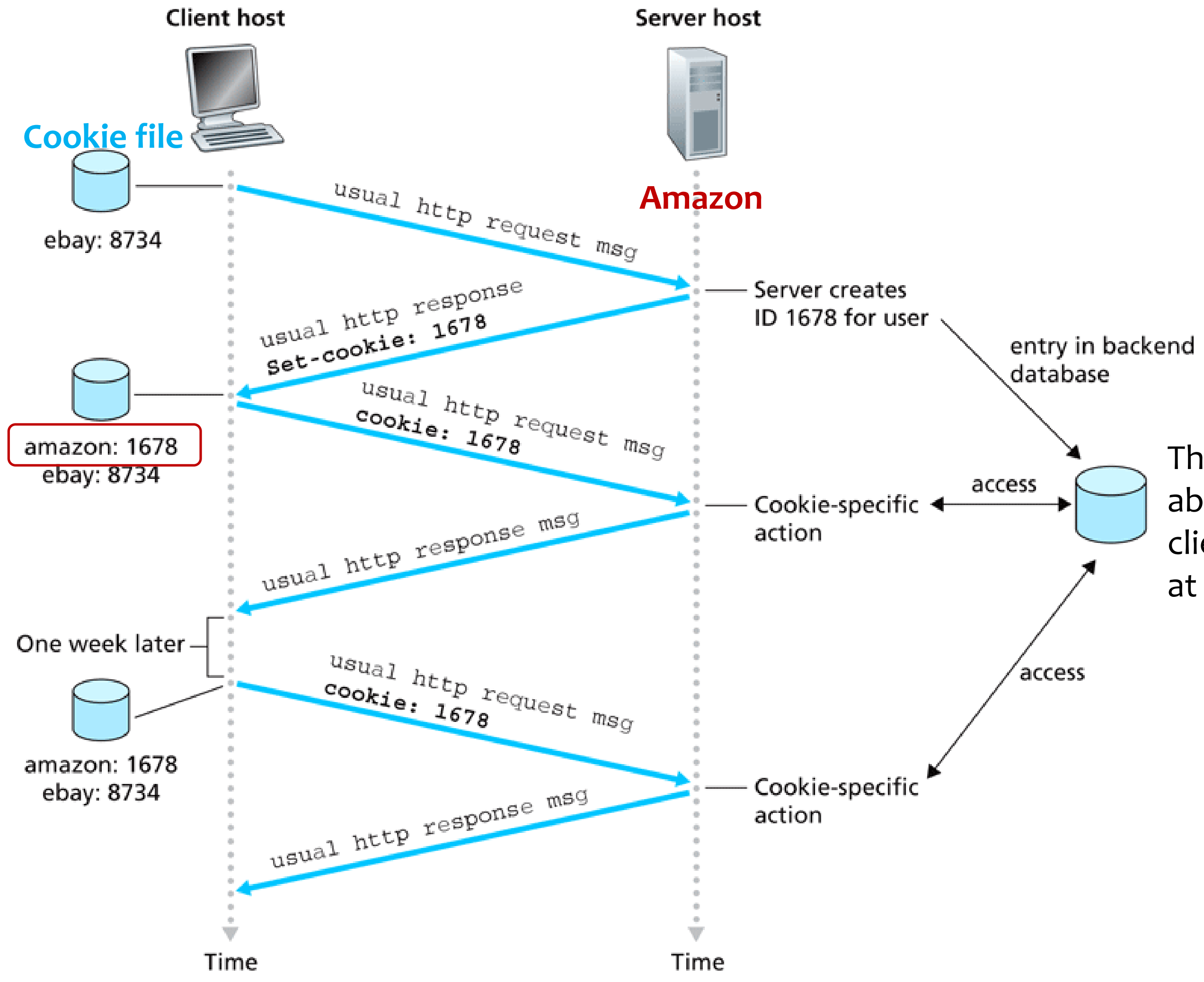
```
HTTP/1.1 304 Not
Modified
```

- Later requests, proxy server or browser sends a **condition GET** request to origin server.
  - Normal GET with the **If-None-Match** header and/or **If-Modified-Since** header
    - If-None-Match: "33a64df551425fcc55e4d42a148795d9f25f89d4"
    - If-Modified-Since: Wed, 21 Oct 2015 07:28:00 GMT
- Suppose the object is not modified. Origin server responses:
  - **HTTP/1.1 304 Not Modified** and other headers.
  - Not including the object in the response.

Using Cookies

- By using cookies, we can perform:
  - Session management
  - Personalization
  - Tracking
- This was not possible with the stateless HTTP protocol.
- Cookie is a small piece of data that a server sends to the browser with the **Set-Cookie** header.
  - E.g., Set-Cookie: user_id=1678
- The browser may store it and **send it back with the next request** to the same server with the **Cookie** header.
  - E.g., Cookie: user_id=1678

Cookies: Keeping States



The server is able to track client's activity at the web site.

Reading

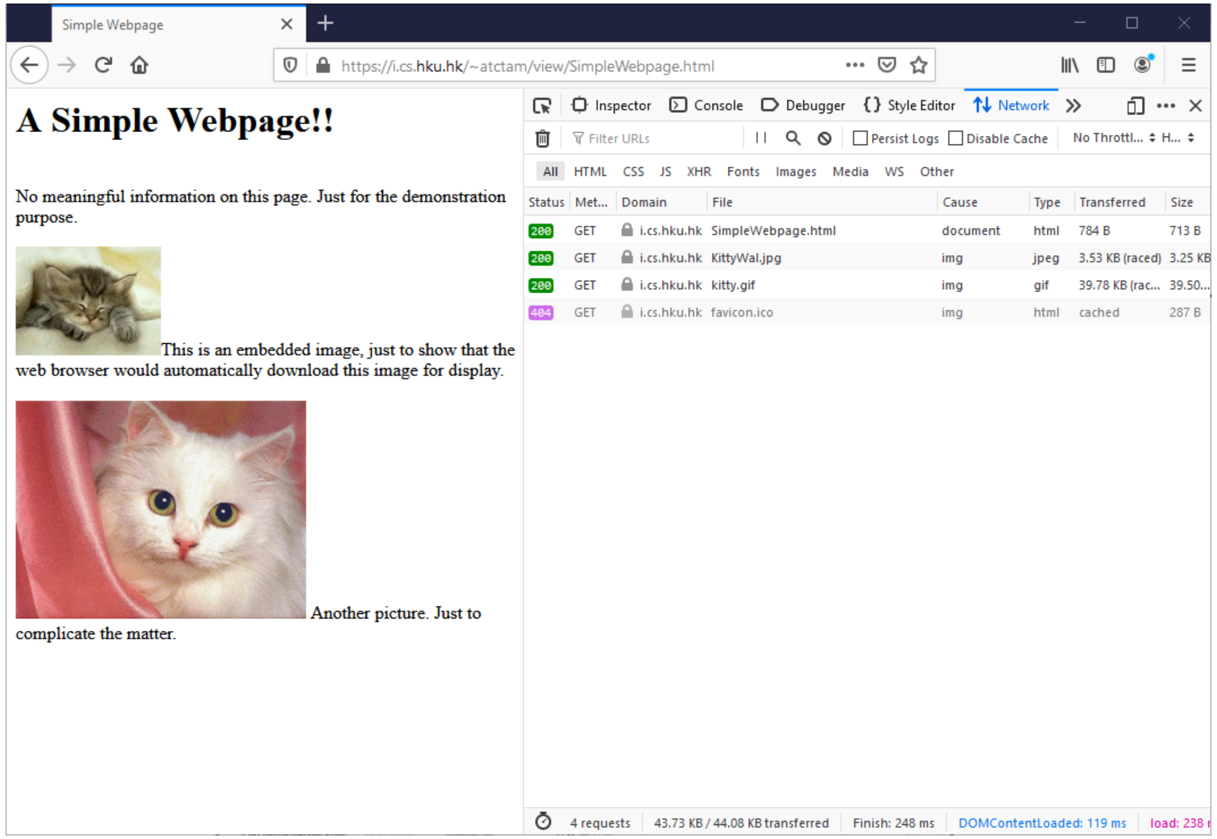
- MDN Web Docs
    - An overview of HTTP
        - https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Other References

- The TCP/IP Guide
  - Hypertext Transfer Protocol
    - http://www.tcpipguide.com/free/t_TCPIPHypertextTransferProtocolHTTP.htm

Browser Tools for HTTP/HTTPS

- Modern browsers provide the developer tools that help us understand the HTTP traffic for a given web page.



A Simple Webpage!!

No meaningful information on this page. Just for the demonstration purpose.

This is an embedded image, just to show that the web browser would automatically download this image for display.

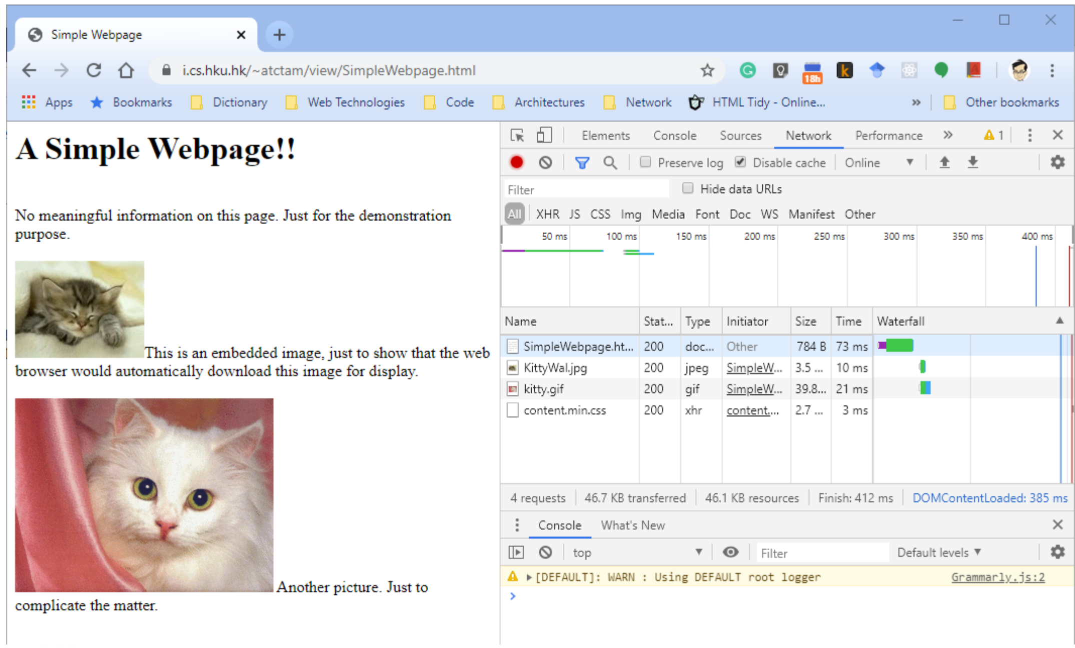
Another picture. Just to complicate the matter.

Status	Met...	Domain	File	Cause	Type	Transferred	Size
200	GET	i.cs.hku.hk	SimpleWebpage.html	document	html	784 B	713 B
200	GET	i.cs.hku.hk	KittyWal.jpg	img	jpeg	3.53 KB (raced)	3.25 KB
200	GET	i.cs.hku.hk	kitty.gif	img	gif	39.78 KB (rac...	39.50...
404	GET	i.cs.hku.hk	favicon.ico	img	html	cached	287 B

4 requests | 43.73 KB / 44.08 KB transferred | Finish: 248 ms | DOMContentLoaded: 119 ms | load: 238

Browser Tools for HTTP/

- To use the developer tools
  - Chrome and Firefox
    - Press "F12" or "Control+Shift+I"
  - Edge
    - Press "F12" or "Control+Shift+I"
  - Safari
    - To enable the Develop menu in the menu bar, choose Safari > Preferences, click Advanced, then select “Show Develop menu in menu bar”
    - Press "Option-Command-I"



A Simple Webpage!!

No meaningful information on this page. Just for the demonstration purpose.

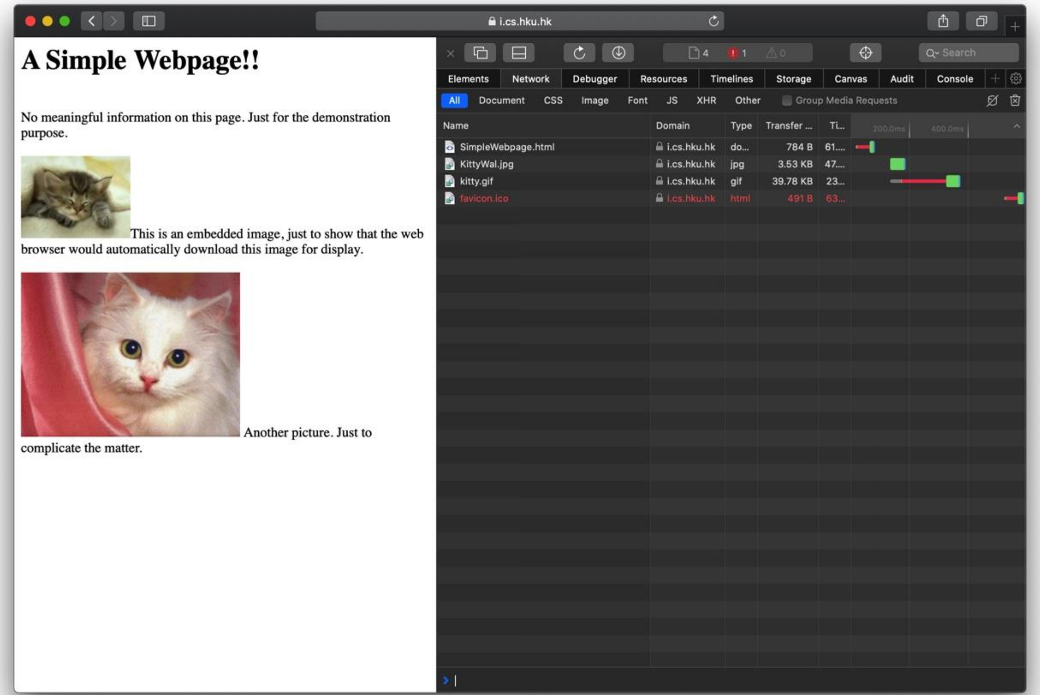
This is an embedded image, just to show that the web browser would automatically download this image for display.

Another picture. Just to complicate the matter.

Name	Stat...	Type	Initiator	Size	Time
SimpleWebpage.ht...	200	doc...	Other	784 B	73 ms
KittyWal.jpg	200	jpeg	SimpleW...	3.5 ...	10 ms
kitty.gif	200	gif	SimpleW...	39.8...	21 ms
content.min.css	200	xhr	content....	2.7 ...	3 ms

4 requests | 46.7 KB transferred | 46.1 KB resources | Finish: 412 ms | DOMContentLoaded: 385 ms

```
[DEFAULT]: WARN : Using DEFAULT root logger        Grammarly.js:2
```



A Simple Webpage!!

No meaningful information on this page. Just for the demonstration purpose.

This is an embedded image, just to show that the web browser would automatically download this image for display.

Another picture. Just to complicate the matter.

Name	Domain	Type	Transfer ...	Ti...
SimpleWebpage.html	i.cs.hku.hk	do...	784 B	61....
KittyWal.jpg	i.cs.hku.hk	jpg	3.53 KB	47....
kitty.gif	i.cs.hku.hk	gif	39.78 KB	23...
favicon.ico	i.cs.hku.hk	html	491 B	63...

Quick Practice

Assume this is the first time you access the page. Otherwise, clear the cache before performing this experiment.
Menu >> Options >> Privacy & Security >> Cookies and Site Data >> select Cached Web Content only >> Clear Data

- For the course, both Firefox and Chrome are two important platforms for our learning.
- Turn on the developer tools of Firefox and then access this web page:
  - https://i.cs.hku.hk/~atctam/view/SimpleWebpage.html
  - Find the following information
    - How many GET requests? What are the status codes of the responses?
    - What is the IP address of the server? Which type of OS the web server is running on?
    - What is the size of each returned entity?
    - Find the headers that are related to Cache Control.
  - Click the reload button to download the page again. Can you see any differences?

Quick Practice

- Perform the same experiment by using Chrome.
  - Again, clear the cache before the first download.
    - Menu >> Settings >> Privacy and security >> Clear browsing data >> select Cached images and files only >> Clear data
  - Find the following information
    - How many GET requests? What are the status codes of the responses?
    - Any differences with what you have seen via Firefox?
  - Chick the reload button to download the page again. Can you see any differences?