# Introduction to Web Technologies and e-Services

ONE LOVE. ONE FUTURE.

1

## Contents

1. Internet, Web
2. HTTP
3. URL
4. Web Browser
5. Web Application
6. Web Application Architecture
7. Web Developer Roadmap



2

## Reasonable Questions

- What is the World Wide Web?
- Is it the same thing as the Internet?
- Who invented it?
- How old is it?
- How does it work?
- What kinds of things can it do?
- What does it have to do with programming?



## Web ≠ Internet

- Internet: a physical network connecting millions of computers using the same protocols for sharing/transmitting information (TCP/IP)
  - Internet is a network of smaller networks
- World Wide Web: a collection of interlinked multimedia documents that are stored on the Internet and accessed using a common protocol (HTTP)
- Key distinction: Internet is hardware; Web is software along with data, documents, and other media
- Many other Internet-based applications exist e.g., email, telnet, ftp, usenet, instant messaging services, file-sharing services, …

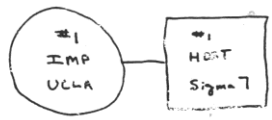
# (A Very Brief) History of the Internet

- The idea of a long-distance computer network traces back to early 60's
  - Joseph Licklider at M.I.T. (a “time-sharing network of computers”)
  - Paul Baran at Rand (tasked with designing a “survivable” communications system that could maintain communication between end points even after damage from a nuclear attack)
  - Donald Davies at National Physics Laboratory in U.K.
- In particular, the US Department of Defense was interested in the development of distributed, decentralized networks
  - survivability (i.e., network still functions despite a local attack)
  - fault-tolerance (i.e., network still functions despite local failure)
  - contrast with phone system, electrical system which are highly centralized services



# Nguồn gốc Internet: ARPANET

- Bắt đầu từ một thí nghiệm của dự án của Advanced Research Project Agency (ARPA)[1] – Bộ quốc phòng Mỹ
- Một liên kết giữa hai nút (IMP tại UCLA và IMP tại SRI) ➔ ARPANET
- Hợp tác giữa Bob Kahn[2] tại DARPA và Vint Cerf[3] tại đại học Stanford



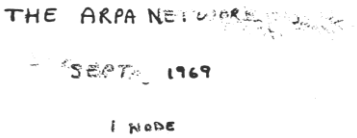



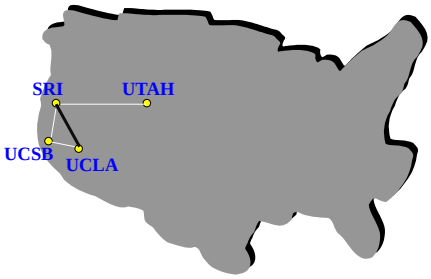

FIGURE 6.1 Drawing of September 1969 (Courtesy of Alex McKenzie)

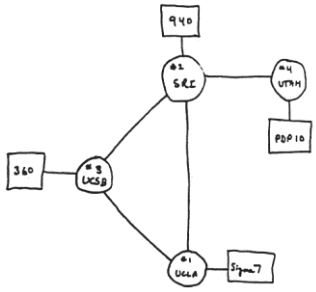
ARPA: Advanced Research Project Agency
UCLA: University California Los Angeles
SRI: Stanford Research Institute
IMP: Interface Message Processor

Nguồn: http://www.cybergeography.org/atlas/historical.html

[1] https://en.wikipedia.org/wiki/DARPA
[2] https://en.wikipedia.org/wiki/Bob_Kahn
[3] https://en.wikipedia.org/wiki/Vint_Cerf

# 3 tháng sau, 12/1969




THE ARPA NETWORK

DEC 1969

4 NODES

FIGURE 6.2 Drawing of 4 Node Network (Courtesy of Alex McKenzie)

Một mạng hoàn chỉnh với 4 nút, 56kbps

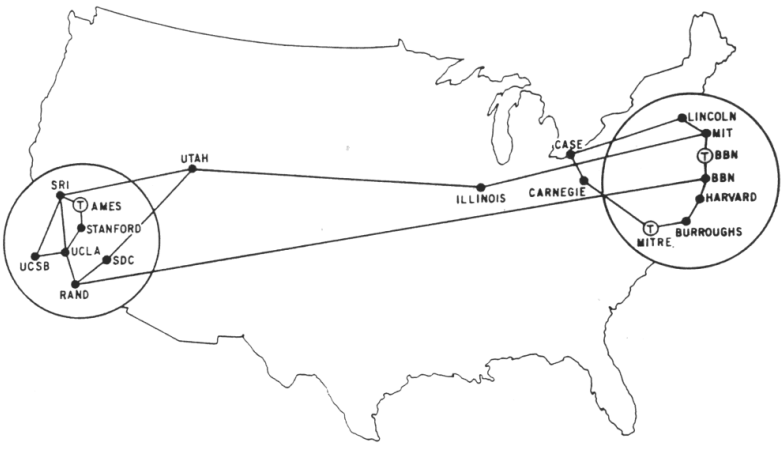
UCSB: University of California, Santa Barbara
UTAH: University of Utah

Nguồn: http://www.cybergeography.org/atlas/historical.html

7

7

# ARPANET thời kỳ đầu, 1971




MAP 4 September 1971

Mạng phát triển với tốc độ thêm mỗi nút một tháng

Source: http://www.cybergeography.org/atlas/historical.html

8

8

# Internet thập niên 80

- Mạng NSFNET & thay thế sứ mệnh ARPANET
- Unix & mạng USENET
- USENET chuyển từ ARPANET sang NSFNET
- Các giao thức mới & kết nối mạng mới
- NSFNET ➔ Internet backbone
- Chuẩn hóa Internet (bao gồm cả các dịch vụ trên Internet như là Web): IETF

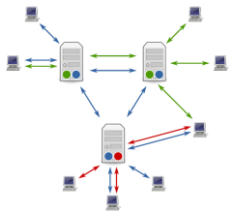
9

9

# Unix & mạng USENET

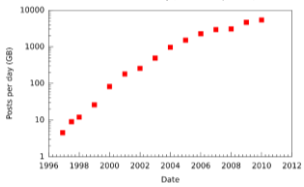
- Unix:
  - Hệ điều hành máy tính lâu đời nhất vẫn phát triển đến nay
  - Ra đời năm 1970 tại phòng thí nghiệm Bell[1]
  - Phát triển theo rất nhiều dòng sản phẩm, cài đặt trên các loại máy tính lớn mainframe, mini và cả các dòng máy cá nhân
  - Linux: hệ điều hành "clone" từ Unix
- USENET:
  - Ra đời năm 1980 và vẫn được sử dụng đến gần đây
  - Kết nối các máy tính chạy hệ điều hành Unix
  - Sử dụng ARPANET làm đường truyền (sau này dùng Internet)
  - Unix-to-Unix Copy (UUCP) network architecture
  - Hướng đến unix users với các dịch vụ unix như email, file transfer, telnet
  - Newsgroup: dịch vụ cung cấp thông tin rất phổ biết (trước khi có Web)

**Usenet servers and clients:**



Source: Wikipedia

**Usenet Traffic Per Day:**



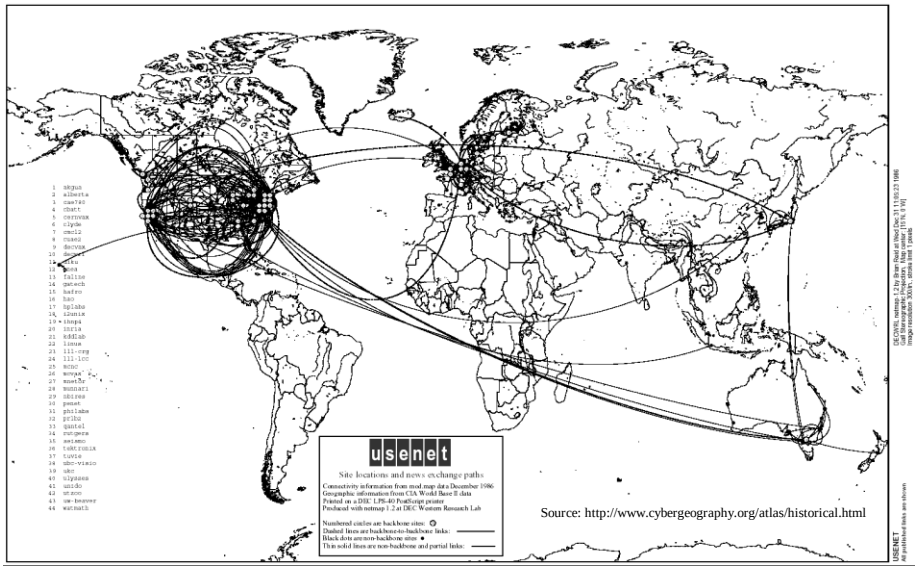

Source: Wikipedia

[1] https://en.wikipedia.org/wiki/Bell_Labs
[2] https://en.wikipedia.org/wiki/National_Science_Foundation_Network

10

10

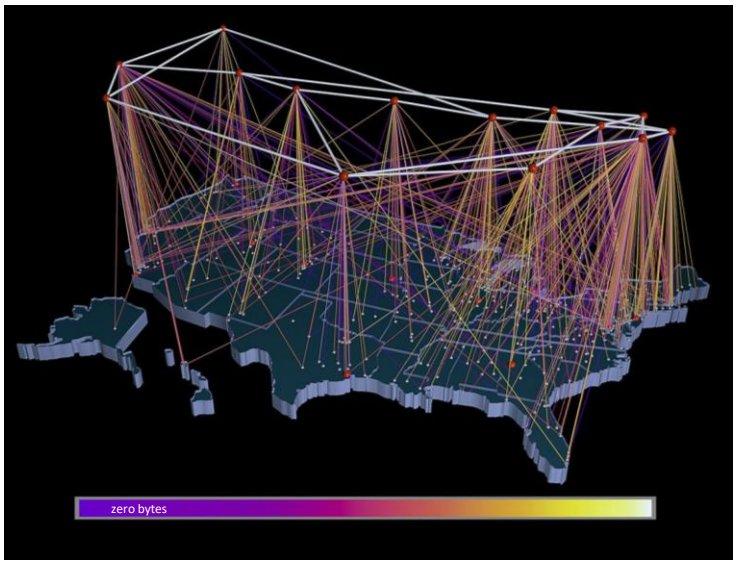
# 1986: Nối kết USENET & NSFNET




11



# Internet thập niên 90



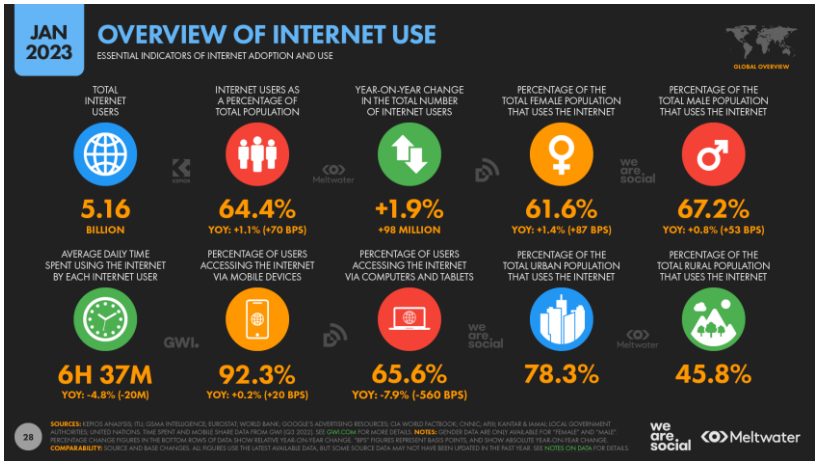

- Kết nối Internet năm 1991: Phổ lưu lượng giao thông trên mạng backbone NSFNET năm 1991 (nguồn: wikipedia)

- Thương mại hóa Internet
- Web & Web & Web

12

# Internet Growth (cont.)

- Internet has exhibited exponential growth, doubling in size every 1-2 years (stats from Internet Software Consortium)
- Internet những ngày đầu tại Việt Nam: https://vi.wikipedia.org/wiki/Internet_t%E1%BA%A1i_Vi%E1%BB%87t_Nam




13




## World Wide Web

- History & Web growth
- Web 1.0, 2.0 and 3.0
- Browser war
- Web IT engineering market

14

## Thủa chưa có Web





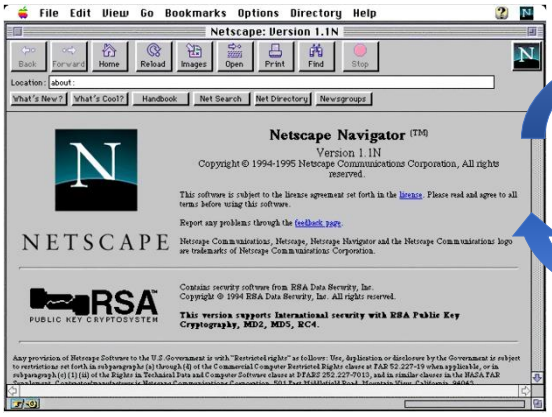
## (A Very Brief) History of the Web

- The idea of hypertext (cross-linked and inter-linked documents) traces back to Vannevar Bush in the 1940's
  - Online hypertext systems began to be developed in 1960's
  - In 1987, Apple introduced hypercard (a hypermedia system that predated the WWW)
- In 1989, Tim Berners-lee at the European particle physics laboratory (CERN) designed a hypertext system for linking documents over the internet
  - Designed a (non-wysiwyg) language for specifying document content => Evolved into hypertext markup language (HTML)
  - Designed a protocol for downloading documents and interpreting the content => Evolved into hypertext transfer protocol (HTTP)
  - Implemented the first browser -- text-based, no embedded media

# Where It All Started



# Web at the beginning days




*Netscape Navigator (borowser) in 1994*

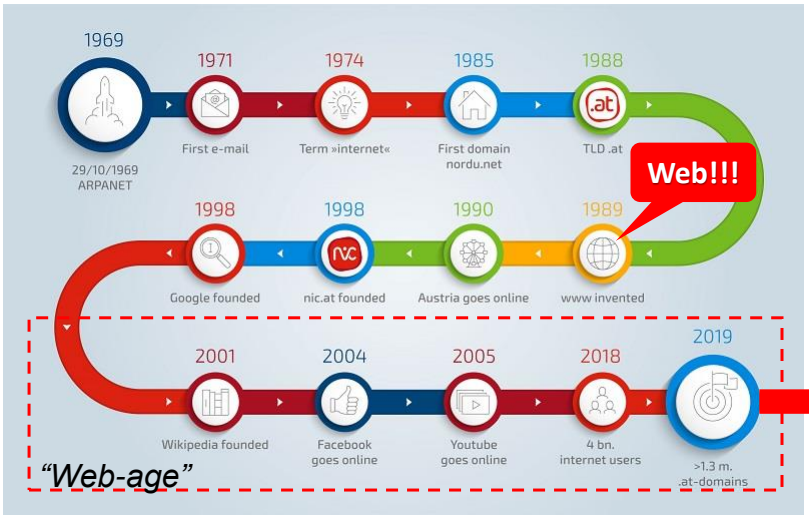

*Old servers to provide online Internet services*

# History of the Web (cont.)

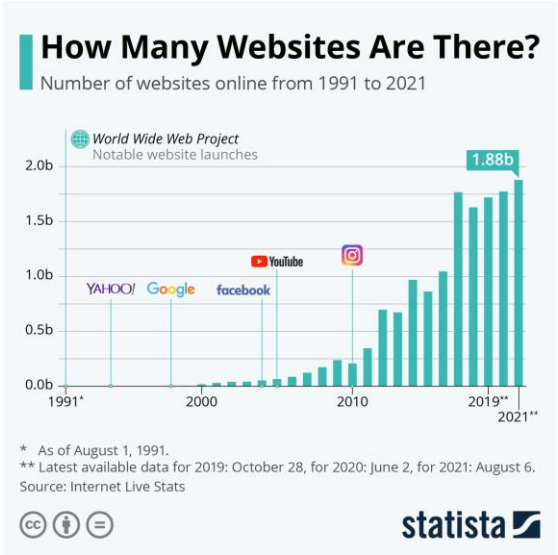
- The Web was an obscure, European research tool until 1993
- In 1993, Marc Andreessen and Eric Bina (at the National Center for Supercomputing Applications, a unit of the University of Illinois) developed Mosaic, one of the early graphical Web browsers that popularized the WWW for the general public
- Andreessen left NCSA to found Netscape in 1994
  - Cheap/free browser  further popularized the Web (75% market share in 1996)
- In 1995, Microsoft came out with Internet Explorer
- Opera web browser released in 1996
  - Firefox web browser, version 1.0, released in 2004
  - Google Chrome released in 2008
- Today, the Web is the most visible aspect of the Internet



# Internet milestones & Web-age



# Web Growth




| | Year | Computers on the Internet | Web Servers on the Internet |
|---|---|---|---|
| | 2011 | ~605,000,000 | ~250,000,000 |
| Google Chrome → | 2008 | | 172,338,726 |
| | 2006 | 439,286,364 | 85,541,228 |
| Firefox → | 2004 | 285,139,107 | 56,923,737 |
| Safari → | 2002 | 162,128,493 | 33,082,657 |
| | 2000 | 93,047,785 | 18,169,498 |
| | 1998 | 36,739,000 | 4,279,000 |
| | 1996 | 12,881,000 | 300,000 |
| IE, Opera → | 1994 | 3,212,000 | 3,000 |
| Netscape → | | | |
| Mosaic → | 1992 | 992,000 | 50 |

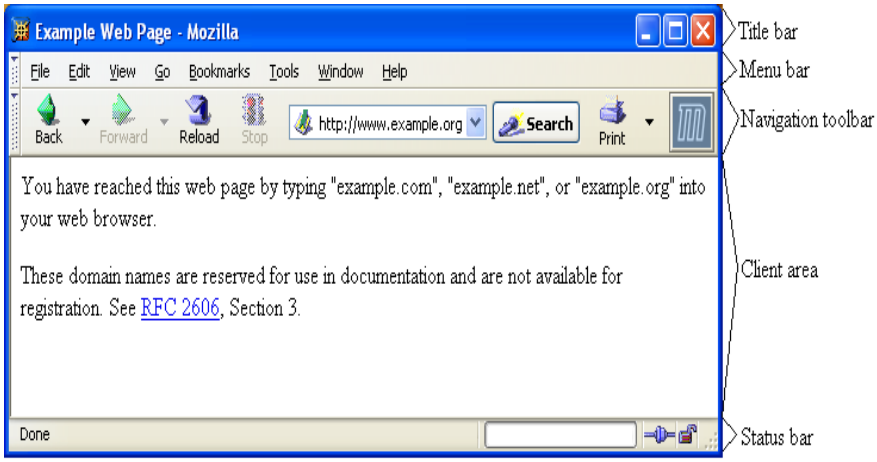


# Web growth (cont.)

- Internet addresses are used to identify computers on the internet.
- Internet Protocol version 4 (IPv4) was first defined in 1981 and is still in use today, but this uses a 32-bit number to specify addresses.
- IPv4 provides around 4.29 billion addresses that are in use (or reserved).
- IPv6 had been deployed since the mid-2000s and uses 128 bit addresses, but also redesigned to allow more efficient routing, network aggregation, and ease of network reconfiguration.

# Web Browsers

- Convert web addresses (URL's) to HTTP requests
- Communicate with web servers via HTTP
- Render (appropriately display) documents returned by a server






# Web Browsers - History

- 1990. WordWideWeb, Tim Berners-Lee
- 1993. Mosaic 1.0
- 1994. Netscape Navigator 1.0
- 1995. Microsoft Internet Explorer 1.0
- 1996. Opera 2.0
- 2002. Mozilla Phoenix 0.1
- 2003. Apple Safari Public Beta
- 2004. Mozilla Firefox 1.0
- 2008. Google Chrome Beta

# Browser WAR

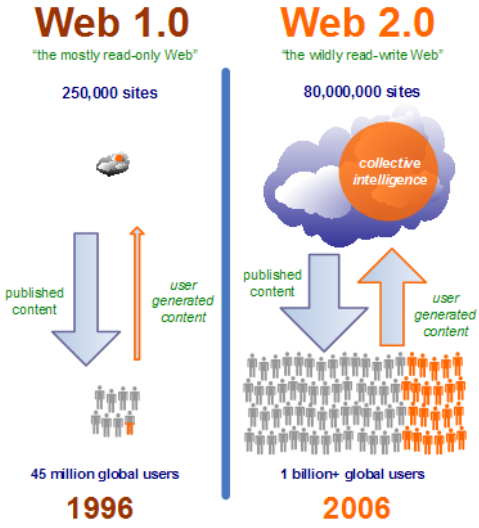
- https://en.wikipedia.org/wiki/Browser_wars
- The "first browser war" (1995–2001) consisted of Internet Explorer and Netscape Navigator: Unlike Netscape Navigator, Internet Explorer 1.0 was available to all Windows (from Win95) users free of charge, including commercial companies.
- Finally: Netscape died!!! But Javascript win ➔ Web 2.0 started its time

- The "second browser war" (2004-2017) between Internet Explorer, Firefox, and Google Chrome
- Other companies later followed suit and released their browsers free of charge.[14] Netscape Navigator and competitor products like InternetWorks, Quarterdeck Browser, InterAp, and WinTapestry were bundled with other applications to full Internet suites.



# Web 2.0

- "Web 2.0" refers to the second generation of web development and web design that facilities
  - information sharing,
  - interoperability,
  - user-centered design
  - collaboration on web.
- Web 2.0 does not have any technical update specifications
- Web 2.0 refers to cumulative changes in the ways software developers and end-users utilize the Web
  - User create web contents
  - Users can own the data and exercise control over that data
- Ex: social-networking sites, video-sharing sites, wikis, blogs, google maps...



# Web 2.0 vs. Web 1.0

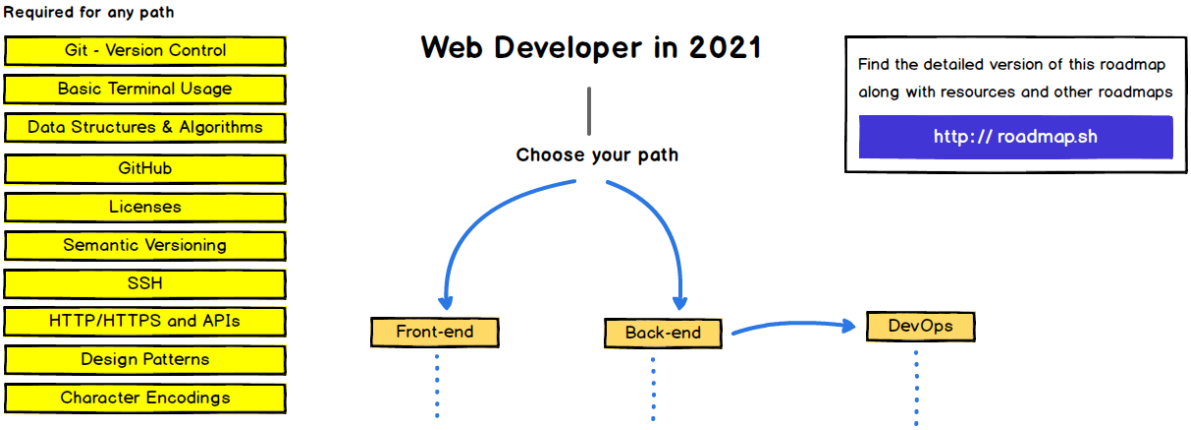
| Web 1.0 | Web 2.0 |
|---|---|
| collective | dispersed in many places |
| for individuals | for society, collective wisdom |
| provide content | provide services and APIs |
| readable | writable |
| communication between systems | synchronization between systems |
| the system includes structure, the content generated is pre-calculated | auto-generate and auto-suggest |



# and… perhaps, Web 3.0!!!

- Web 1.0
  - Website publish information, user read it
  - Ex:
- Web 2.0
  - User create content: post information, modify, delete
  - Ex: YouTube, Flick
- Web 3.0, next web generation
  - Semantic web (or the meaning of data), personalization (e.g. iGoogle), intelligent search based on behavioral of users.
    - Search for information for user with a request in nature form (a complex sentence)
    - different users obtain deferent search result
  - Ex:iGoogle
- Web 3.0 is defined as the creation of high-quality content and services produced by gifted individuals using Web 2.0 technology as an enabling platform.


# Web engineering market

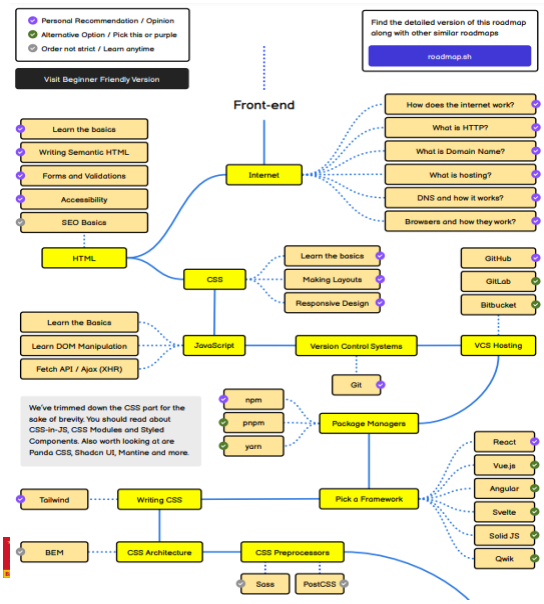


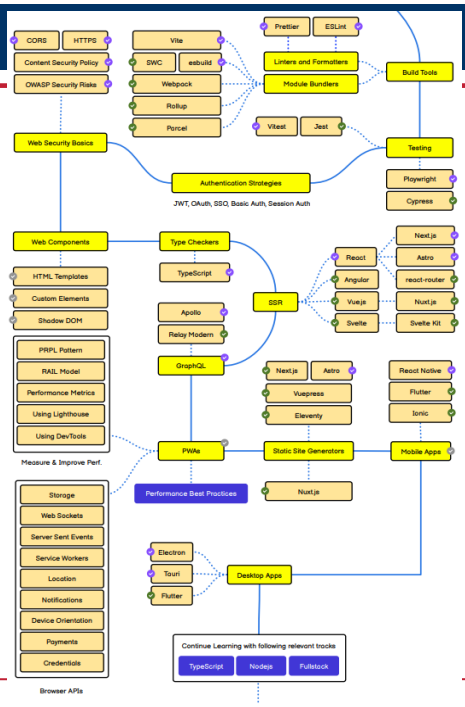





# Front-end

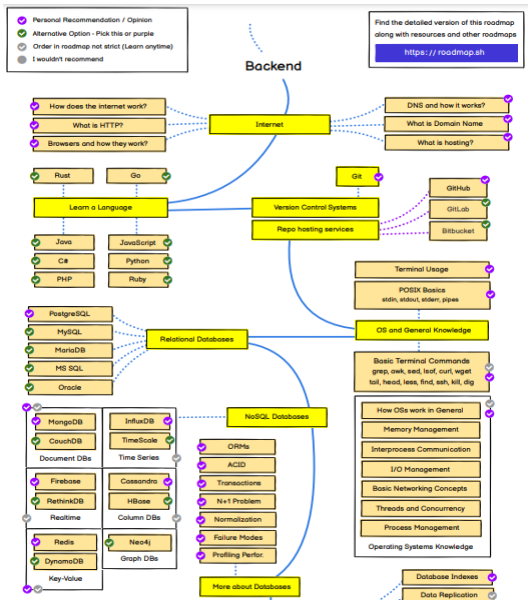
2024: https://roadmap.sh/pdfs/roadmaps/frontend.pdf

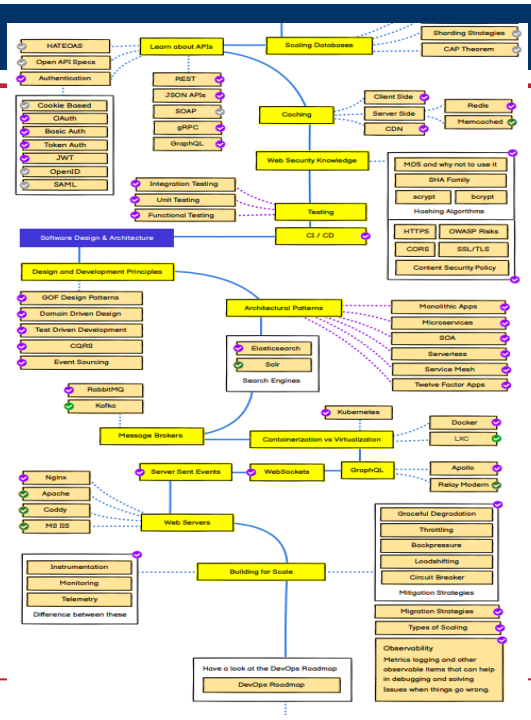


# Back-end

2024: https://roadmap.sh/pdfs/roadmaps/backend.pdf



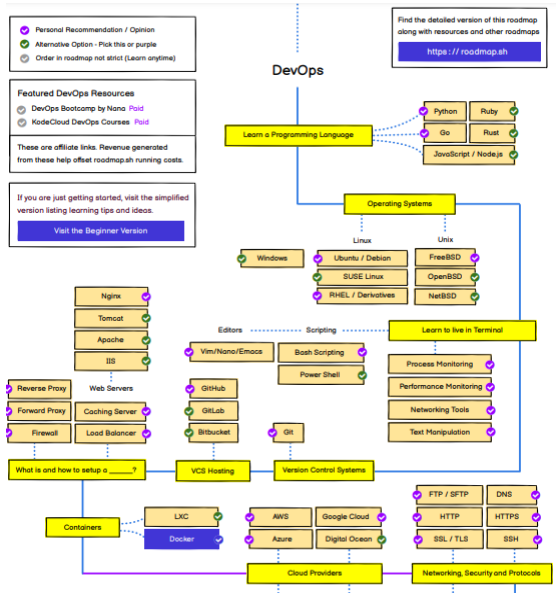



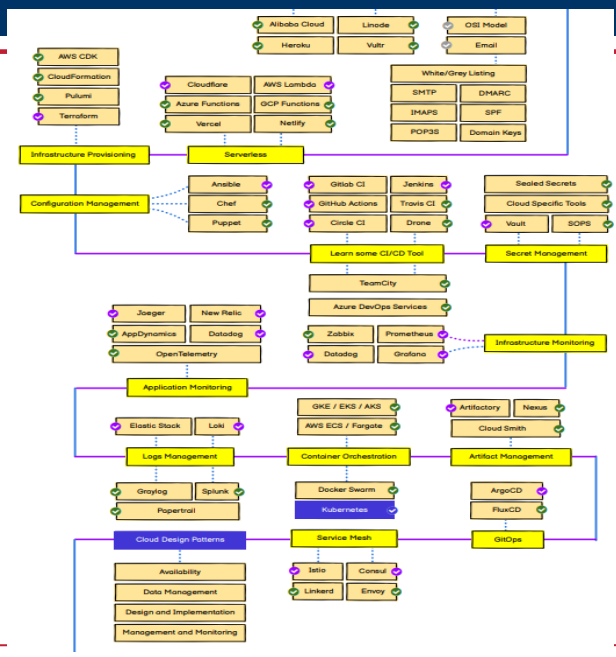

31

31

# DevOps

2024: https://roadmap.sh/pdfs/roadmaps/devops.pdf




32

32


## HTTP Protocol

- Practical environment preparation
- HTTP overview
- Request & Response message
- HTTP methods & custom method
- Persistance and pipeline connection
- Classless model and usage scenarios
- Client authentication & cookie
- Data caching
- Security with HTTPS





## Practical environment

- Apache server: https://httpd.apache.org/download.cgi
- Any Web browser
- Curl (https://curl.se/download.html) or Postman (https://www.postman.com/downloads)
- tcpdump (Linux) or Whiteshark (https://www.wireshark.org/download.html)

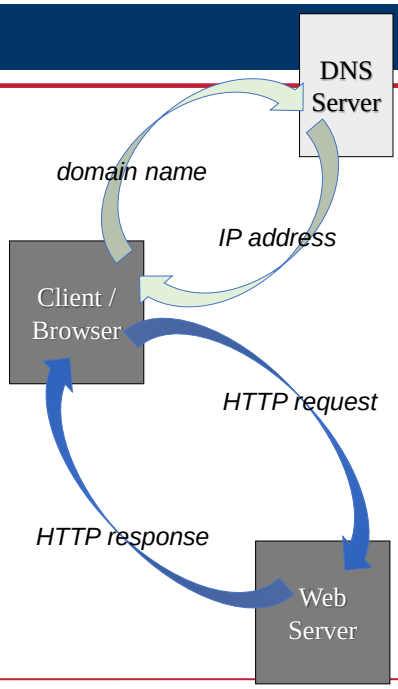
# Hypertext Transport Protocol (HTTP)

- Initiated by Tim Berners-Lee in 1989, described in a standard document as version 1.0 and then, 1.1 (RFC9112 – the most popular now)
- HTTP is based on the request-response communication model:
  - Client sends a request
  - Server sends a response
  - HTTP is a stateless protocol: The protocol does not require the server to remember anything about the client between requests.
- HTTP nowadays:
  - Widely used (Internet killer), not only in Web but other application/service models
  - Not only text but multimedia document and video streaming



# HTTP & DNS

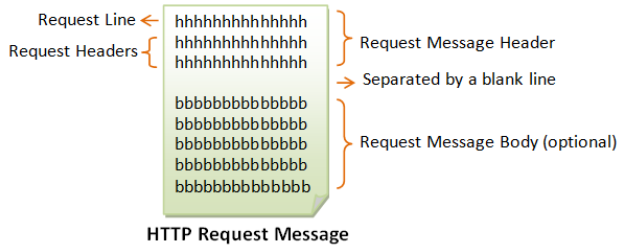
- Run over a TCP connection (welknown port 80)
- Typical browser-server interaction:
  - User enters Web address in browser
  - Browser uses DNS to locate IP address
  - Browser opens TCP connection to server
  - Browser sends HTTP request over connection
  - Server sends HTTP response to browser over connection
  - Browser displays body of response in the client area of the browser window



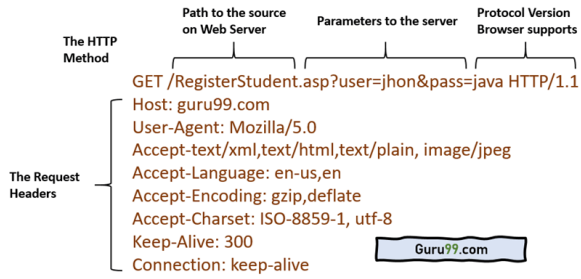
# HTTP Request

- Structure:
  - request line
    - Method
    - URI
    - HTTP version
  - request headers
  - blank line
  - optional request body

**GET** /test.html HTTP/1.1
**POST** /index.html HTTP/1.1




HTTP Request Message



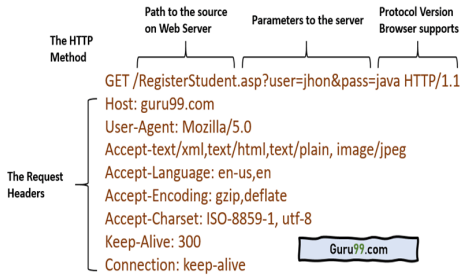



# HTTP Methods *(RFC9112: HTTP 1.1)*

- HTTP Methods: *"The method token indicates the request method to be performed on the target resource. The request method is case-sensitive"*
- The most popular methods:
  - GET: *"Transfer a current representation of the target resource"*
  - POST: *"Perform resource-specific processing on the request content."* (practical use: similar to GET, but parameters are put in message body, not in URI)
- Other methods:
  - HEAD: Same as GET, but do not transfer the response content.
  - PUT: Replace current representations of target resource with the request content.
  - DELETE: Remove all current representations of the target resource.
  - CONNECT: Establish a tunnel to the server identified by the target resource.
  - OPTIONS: Describe the communication options for the target resource.
  - TRACE: Perform a message loop-back test along the path to the target resource.

*Read only*

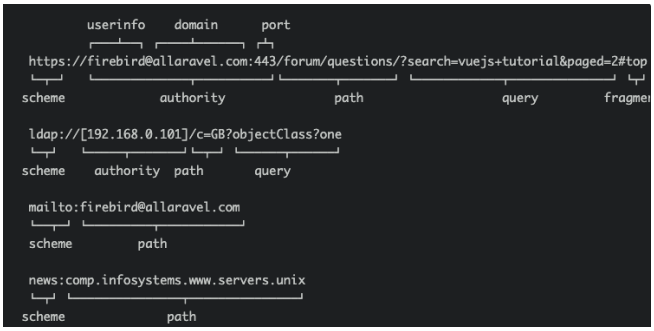
*Write*




*HTTP methods are considered safe if they do not alter the server state (read-only operations). HTTP RFC defines GET, HEAD, OPTIONS and TRACE to be safe*

# URI

- Uniform Resource Identifier
- Uniform Resource Name (URN): Can be used to identify resources with unique names, such as books (which have unique ISBN's)
- Uniform Resource Locator (URL): Specifies location at which a resource can be found In addition to http, some other URL schemes are https, ftp, mailto, and file

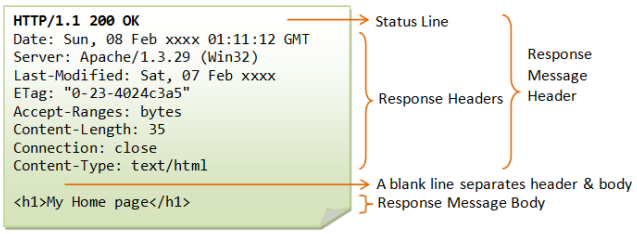



- URI = scheme:[//authority]path[?query][#fragment]



# HTTP Response

- Structure:
  - status line
    - HTTP version
    - status code
    - reason phrase (for human use)
  - header field(s)
  - blank line
  - optional body
- Status code:
  - Three-digit number
  - First digit is class of the status code:
    - 1=Informational
    - 2=Success
    - 3=Redirection (alternate URL is supplied)
    - 4=Client Error
    - 5=Server Error
  - Other two digits provide additional information



# Common header fields in HTTP Response

- Connection, Content-Type, Content-Length
- Date: date and time at which response was generated (required)
- Location: alternate URI if status is redirection
- Last-Modified: date and time the requested resource was last modified on the server
- Expires: date and time after which the client's copy of the resource will be out-of-date
- ETag: a unique identifier for this version of the requested resource (changes if resource changes)

```
curl.exe  -iX GET "http://localhost"
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2024 04:21:49 GMT
Server: Apache/2.4.62 (Win64)
Last-Modified: Thu, 19 Sep 2024 03:08:02 GMT
ETag: "2e-622703b0974a9"
Accept-Ranges: bytes
Content-Length: 46
Content-Type: text/html

<html><body><h1>It works!</h1></body></html>

curl.exe  -iX POST "http://localhost"
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2024 04:23:35 GMT
Server: Apache/2.4.62 (Win64)
. . . .

curl.exe  -iX DELETE "http://localhost"
HTTP/1.1 405 Method Not Allowed
Date: Thu, 19 Sep 2024 04:24:27 GMT
Server: Apache/2.4.62 (Win64)
Allow: GET,POST,OPTIONS,HEAD,TRACE
Content-Length: 223
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>405 Method Not Allowed</title>
</head><body>
<h1>Method Not Allowed</h1>
<p>The requested method DELETE is not allowed for this URL.</p>
</body></html>
```



# HTTP Custom Methods

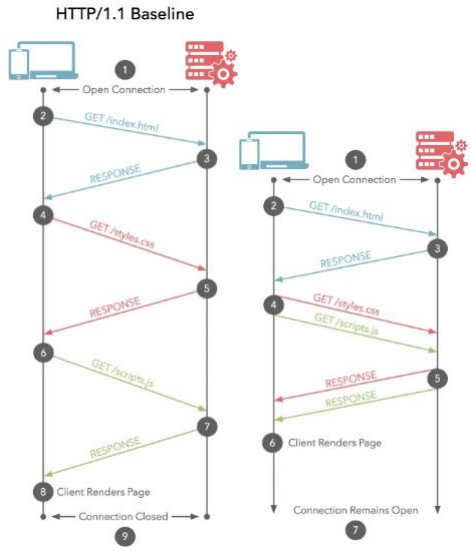
- HTTP protocol (RFC9112) specifies how messages are conveyed over TCP/IP, not the semantic of HTTP methods (RFC9110)
- HTTP methods are now practically used by different standard semantic
- Backend API services may also want to create their own methods with particular semantic
- HTTP custom methods can be considered as an "extension of RFC9110 for a specific service", running on RFC9112
- *Curl* supports custom method: accept any custom methods in command line, transfer them in HTTP request and get the result from HTTP server
- HTTP application servers (such as Apache) need to understand new HTTP methods and do the appropriate behaviour

```
curl.exe  -iX CustomMethod "http://localhost"

HTTP/1.1 501 Not Implemented
Date: Thu, 19 Sep 2024 06:38:18 GMT
Server: Apache/2.4.62 (Win64)
Allow: GET,POST,OPTIONS,HEAD,TRACE
Content-Length: 211
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>501 Not Implemented</title>
</head><body>
<h1>Not Implemented</h1>
<p>CustomMethod not supported for current URL.<br />
</p>
</body></html>
```

(reference) Defining Custom HTTP Methods
https://ericjmritz.wordpress.com/2015/10/22/defining-custom-http-methods

# HTTP Persistent vs Non-Persistent

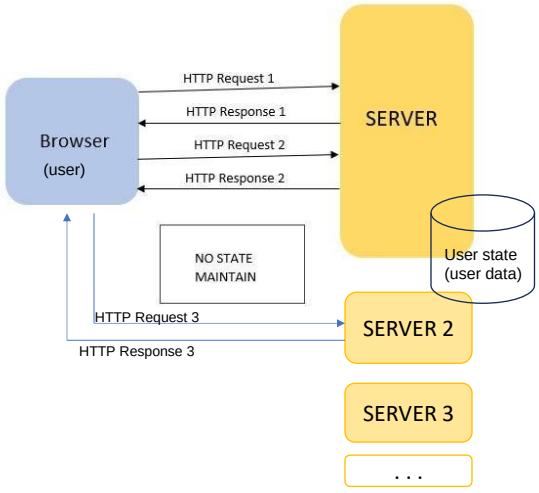
- HTTP runs over TCP ➔ need to open (& destroy) TCP connection
- HTTP 1.0
  - Non-persistent connection, a new connection is established for each request/response cycle.
  - High overhead, increased latency, and slow performance, especially for large or complex web pages
- HTTP 1.1
  - Persistent connection, allowed multiple requests and responses to be sent over a single connection
  - Improve performance by pipeline (parallel) requests
  - When and how to close the connection?
- CONNECTION header line
  - *"keep-alive"*: the connection is persistent and not closed
  - *"close"*: client or the server would like to close the connection. This is the default on HTTP/1.0 requests.





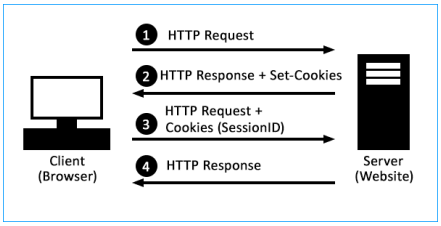

# HTTP Stateless

- Web state:
  - User data stored at server represents user state
  - Client sends HTTP request and server back HTTP response to client with current state of user
  - No state is maintained between HTTP connections (request/response round trip)
- Stateless Protocols: does not remember previous request or response. HTTP is stateless.
- Why HTTP is not stateful?
  - Simplicity: easier to implement and maintain
  - Scalability: do not require any synchronization or coordination between clients and servers ➔ can handle more concurrent requests and responses
  - Reliability: do not depend on any previous or next message ➔ can handle any failure or interruption in the communication without affecting the outcome of the request or response.



# HTTP Stateful by Cookie

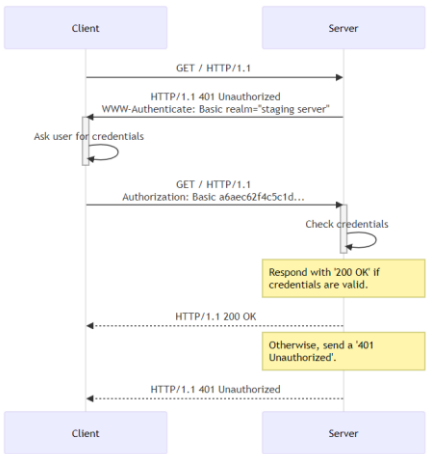
- Stateless is not good
  - Lack of context: do not have any context between clients and servers ➔ each request or response has to contain all the necessary information as the result in previous connections (such as to complete the transaction, client authentication, session data, etc..) ➔ increase the size and complexity of the messages and reduce the performance.
  - Lack of personalization: do not have any memory or knowledge of the communication between clients and servers ➔ each request or response is treated equally and generically without any customization or adaptation based on the user's behavior, preferences, history, etc.. ➔ reduce the user experience and satisfaction of the communication.
- Make HTTP stateful by Cookie
  - Cookies are small pieces of data that are stored on the client's device by the server and implicitly sent back to the server with each request.
  - Cookies can contain various information about the user's identity, preferences, history, etc.
  - Cookies can make HTTP stateful by allowing the server to recognize and remember the user across multiple requests and responses.



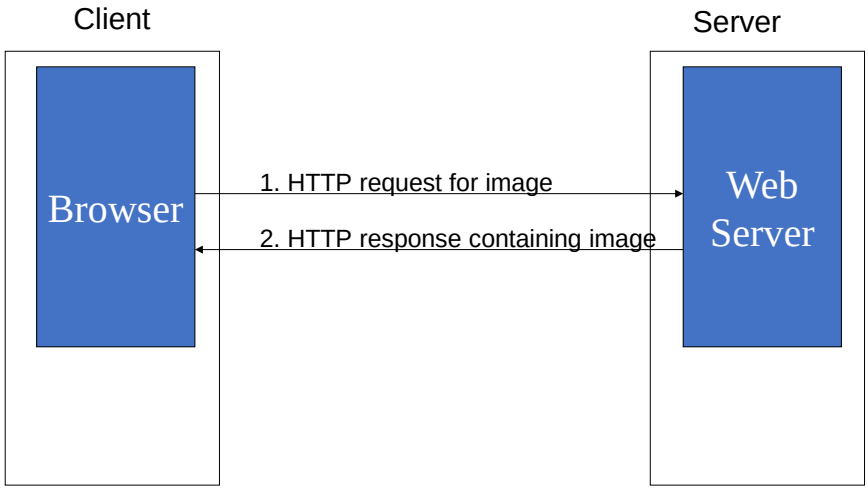



# Client Authentication

- Basic scenario:
  - Client requests access to a resource that need authorization
  - Server responds 401 (Unauthorized) and provides some information on how to authorize
  - Client need credentials from user
  - Client authenticates itself with the server by including an Authorization request header with the credentials.
  - Server get user credentials from HTTP request, successful check and response 200 OK to client.
- Authentication schemes
  - Basic: RFC7617, base64-encoded credentials
  - Bearer: RFC6750, bearer tokens to access OAuth 2.0
  - Digest: RFC7616. Firefox 93 and later support the SHA-256 algorithm
  - HOBA: RFC7486, digital-signature-based
  - . . .
- Support document: Google Authen and Gmail client

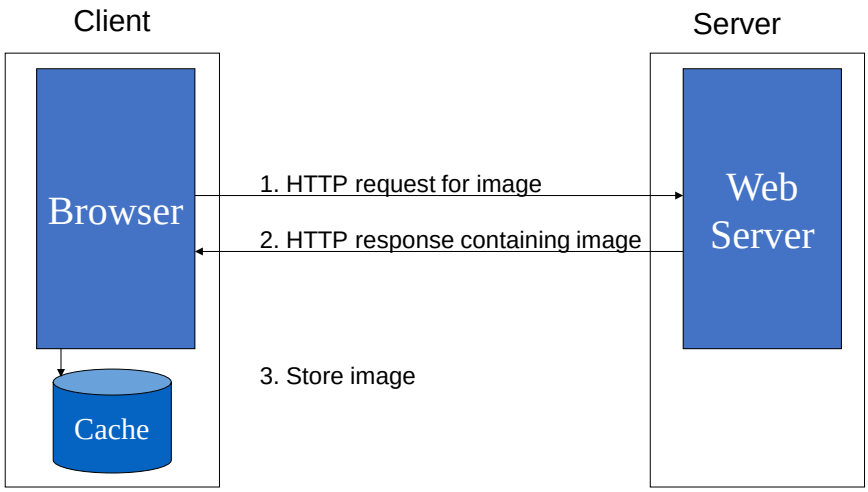


# HTTP caching: basic scenario

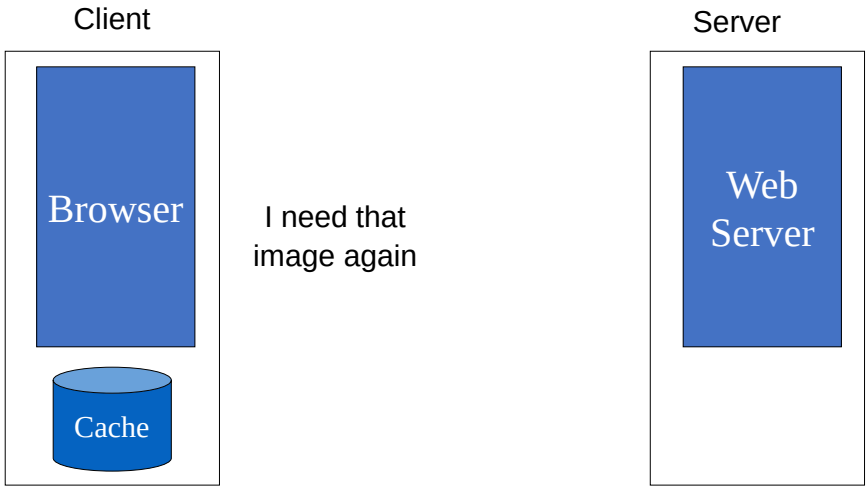





# HTTP caching: basic scenario (2)

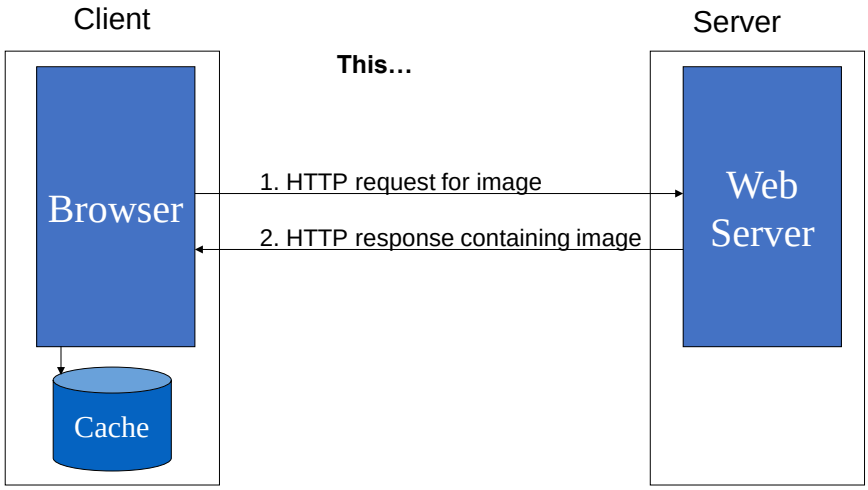


# HTTP caching: basic scenario (3)

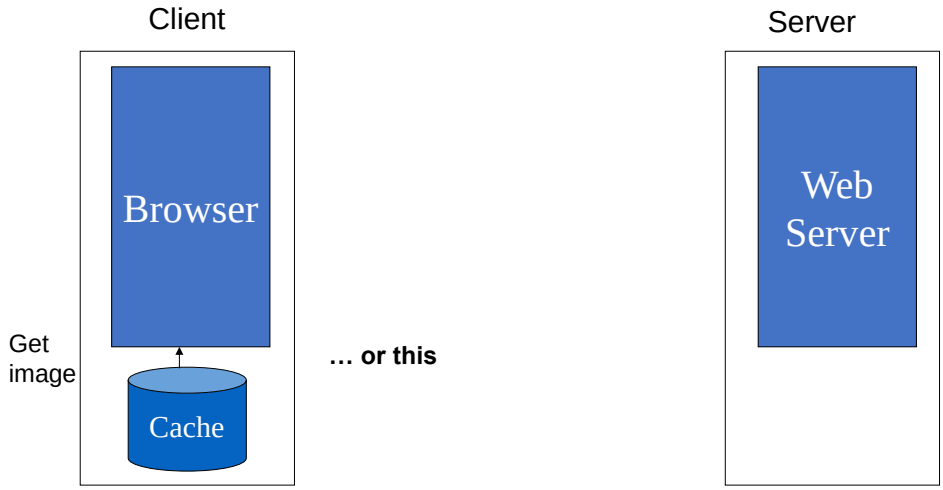





# HTTP caching: basic scenario (4)



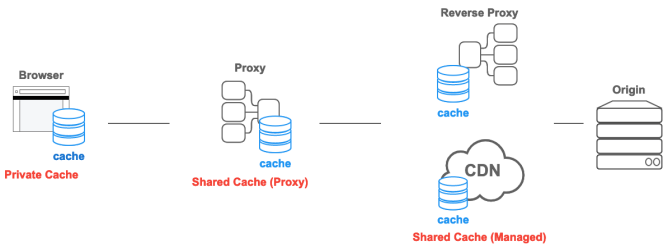
# HTTP caching: basic scenario (5)






# HTTP Caching is the most sophisticated !!!

- Not only in Web but all over computing: DNS, Memory, Youtube, etc…
- Cache locations in Web architecture
  - Local (private)
  - Proxy (shared cache)
  - Network (CDN, such as Youtube)
- HTTP support cache
  - RFC9111: HTTP Caching[1]
  - *"Cache-Control"* header field.
  - no-store, no-cache, max-age=0, must-revalidate, proxy-revalidate
- Heuristic caching
  - HTTP is designed to cache as much as possible, so even if no Cache-Control is given, responses will get stored and reused if certain conditions are met. This is called heuristic caching.
  - It is heuristically known that content which has not been updated for more than 3 years will not be updated for some time after that




```
curl.exe  -iX GET "http://. . . "
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1024
Date: Fri, 20 Sep 2024 2:2:2 GMT
Last-Modified: Tue, 22 Feb 2021 22:22:22 GMT

<!doctype html>
```

[1] https://httpwg.org/specs/rfc9111.html

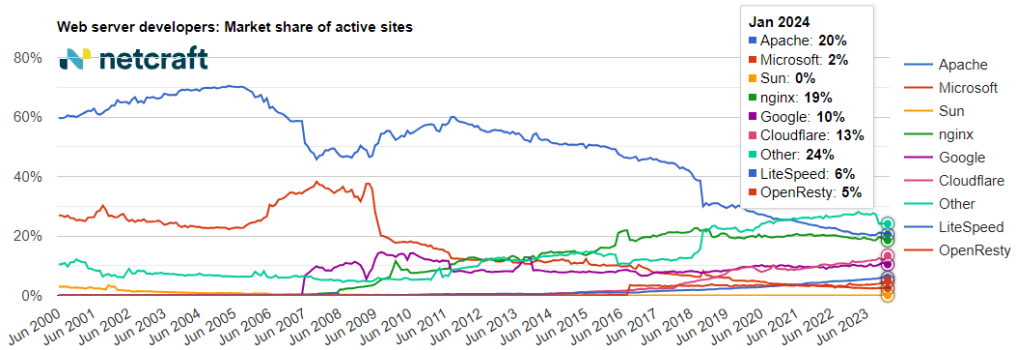
# Some cache solutions for Web development

- Redis:
  - https://redis.io
  - In-memory data store used by millions of developers as a cache, vector database, document database, streaming engine, and message broker. Redis has built-in replication and different levels of on-disk persistence. It supports complex data types (for example, strings, hashes, lists, sets, sorted sets, and JSON), with atomic operations defined on those data types.
- Memcache:
  - https://memcached.org
  - Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering.



# HTTP Servers

- https://www.netcraft.com/blog/january-2024-web-server-survey



# HTTPS: Web Security Solution

- HTTP conveys messages over TCP/IP by clear text, that is easily captured
- Login name and password, credit card number, or even personal phone number should not be disclosed over Internet
- HTTPS provides mechanism to encrypt data before transfer over Internet
- Security guarantee by SSL/TSL, based on PKI mechanism
- Support document: HTTPS in Apache
- See more: https://www.youtube.com/watch?app=desktop&v=j9QmMEWmcfo