



# WWW (World Wide Web) Basics

---

BUPT/QMUL  
2021-5-6



北京邮电大学

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Electronic Engineering 



# Agenda

---

- Brief introduction to WWW
- WWW Components
- WWW Standards
- Summary

*Refer to Chapter 27, textbook*



---

# Brief Introduction To WWW



# What Is WWW?

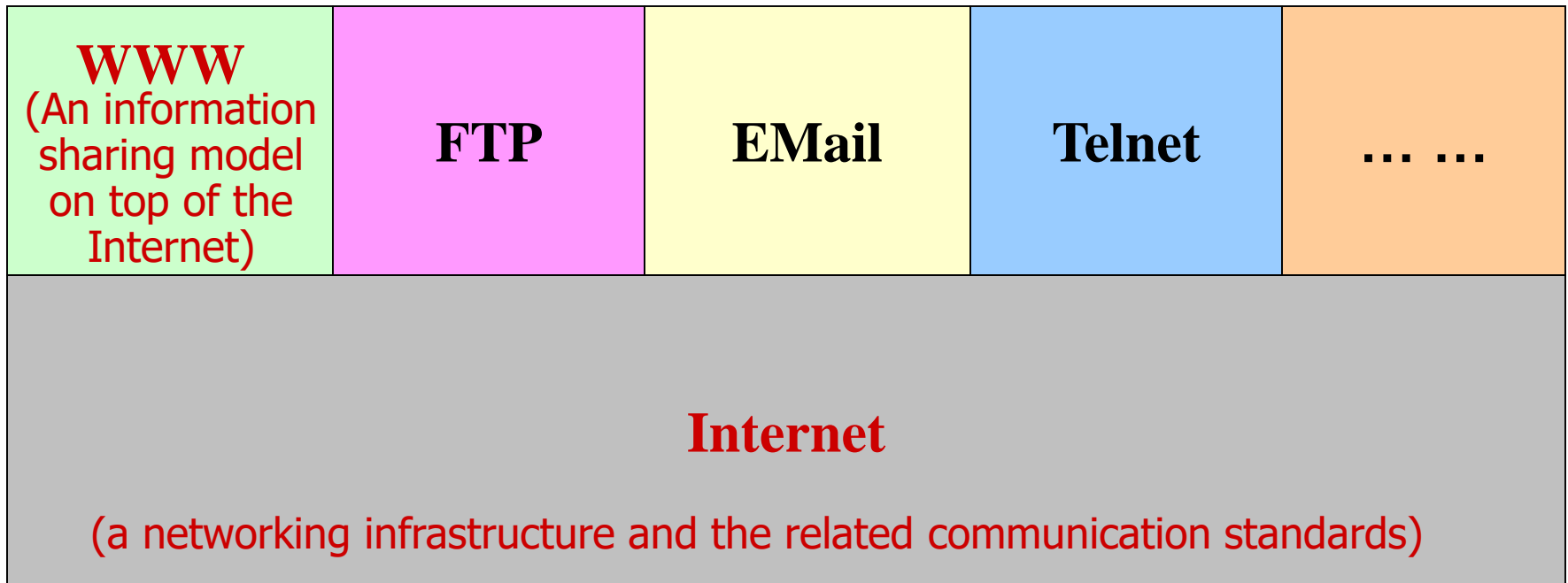
---

- World Wide Web
  - WWW, the Web, W3
- A technical definition
  - All the resources and users on the Internet that are using the Hypertext Transfer Protocol (HTTP).
  - A system of interlinked hypertext documents accessed via the Internet. -- *Wikipedia*
- A broader definition from W3C (World Wide Web Consortium)
  - The World Wide Web is the universe of network-accessible information, an embodiment of human knowledge.



# WWW vs. Internet

---





# History Of WWW

**CERN**

The world's largest  
particle physics laboratory

*... where the web was born!*

**1989-03, Tim Berners Lee**

proposed the idea of sharing information through  
**hypertext** in CERN

**1989-12 , Tim Berners Lee**

named his invention **WWW** (World Wide Web)

**1990-11**

The first (text-based) **prototype** was operational

**1991-12**

The first **public demonstration** was given at  
Hypertext '91 in San Antonio - Texas

**1993-02, Marc Andreessen**

The first **GUI browser** – Mosaic, at NCSA, Illinois

**1994-95, Netscape, Microsoft**

Netscape Navigator, Internet Explorer

**Other browsers**

Mozilla, Firefox, Opera, Chrome, Safari, ...

**Other technologies**

HTML, JAVA, VRML, Web 2.0, ...



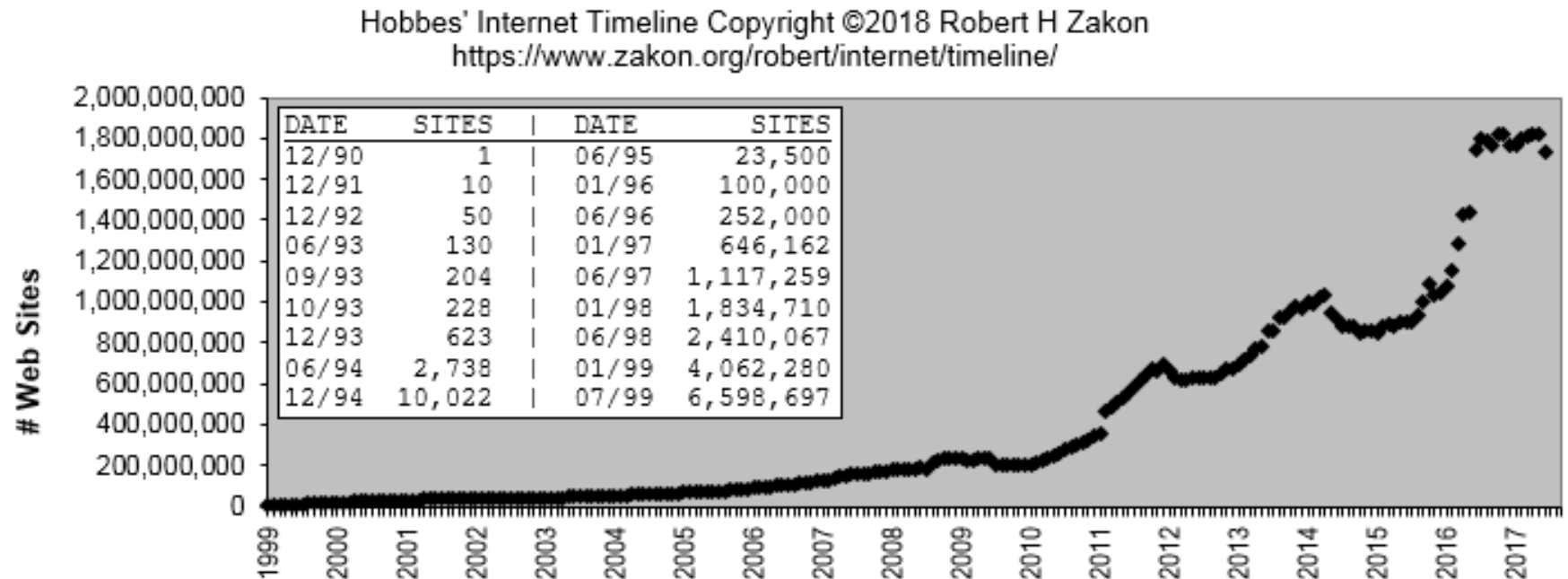
# Features of WWW

---

- Global
- Open
- Interactive
- Dynamic
- Platform-independent
- Multimedia
- ...

# WWW Growth

**Figure: WWW Growth**







# WWW Terminologies

---

- **The Web**

- Is a true information superhighway

- **URL** (Uniform Resource Locator)

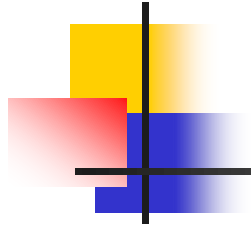
- Designates a specific **webpage** on a specific **webserver**

- **HTTP** (HyperText Transfer Protocol)

- An application-level transfer protocol standard

- **HTML** (HyperText Markup Language)

- A document format standard



# WWW Components



# WWW Components

---

## ■ Structural Components

- Clients/browsers – various implementations
- Servers – run on sophisticated hardware
- Caches – used to improve response time
- Internet – the global infrastructure which facilitates data transfer

## ■ Semantic Components

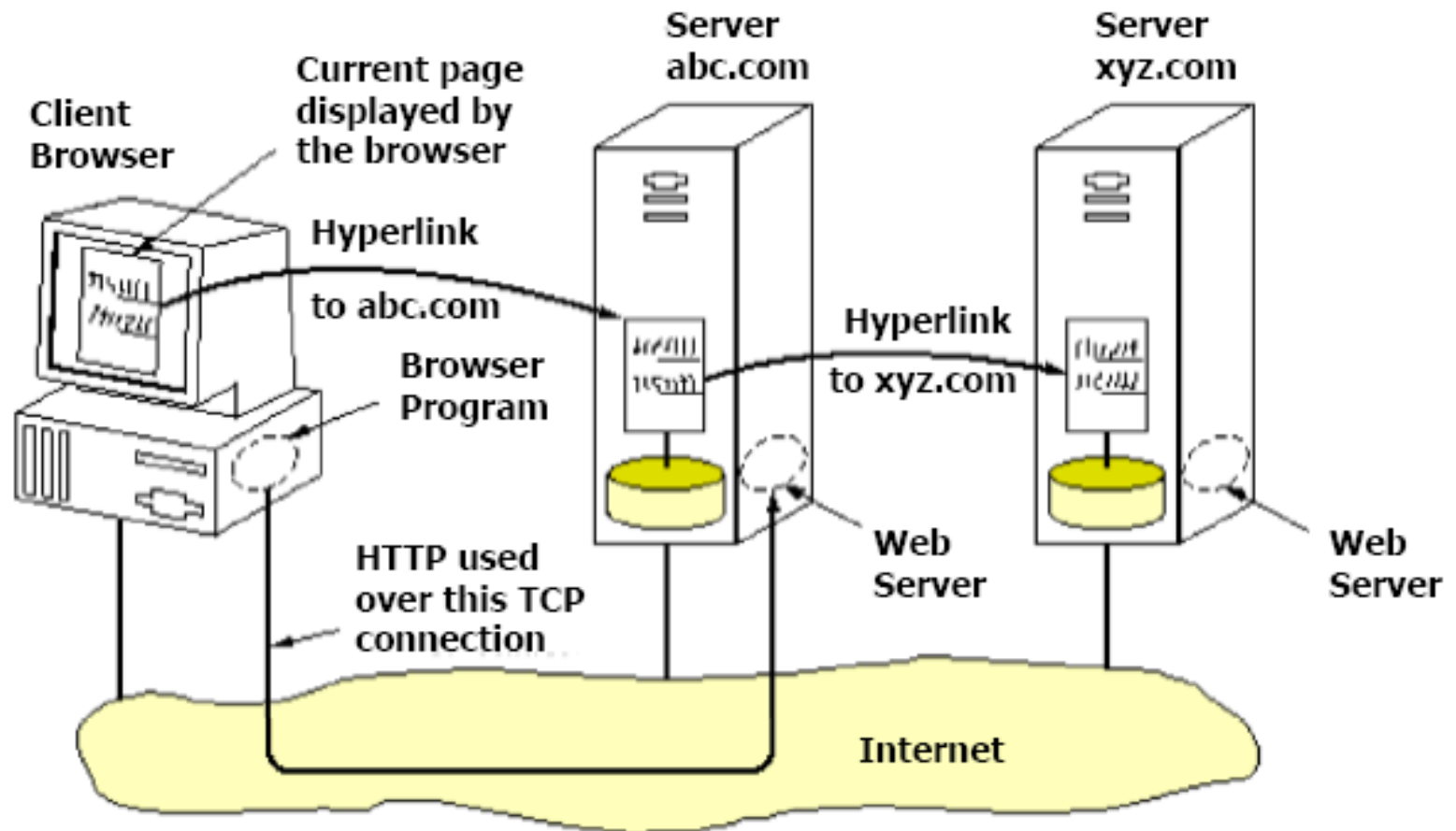
- Hyper Text Transfer Protocol (HTTP)
- Hyper Text Markup Language (HTML)
  - eXtensible Markup Language (XML)
- Uniform Resource Locators (URLs)

# The Web



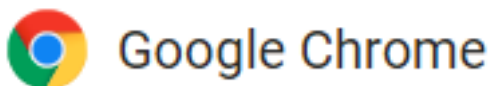
- The Web is actually an information superhighway
- The Web is a collection of electronic documents that are linked together like a spider web
- The Web is basically an information system that links data from many different Internet services under one set of protocols
- **Web clients**, also called **browsers**, **interpret HTML** delivered from **Web servers**
- These documents use **hypertext links** to connect different **documents** and information **resources** together; click on a link and the client software retrieves the linked document or jumps to a specific position in the current document
- **HTTP** is easily modified to incorporate **new data formats** and uses
- The Web model successfully unites the diverse Internet resources under a **single** system, relying on servers and Web-browsers to “**negotiate**” or handle data compatibility

# The Web Access Model



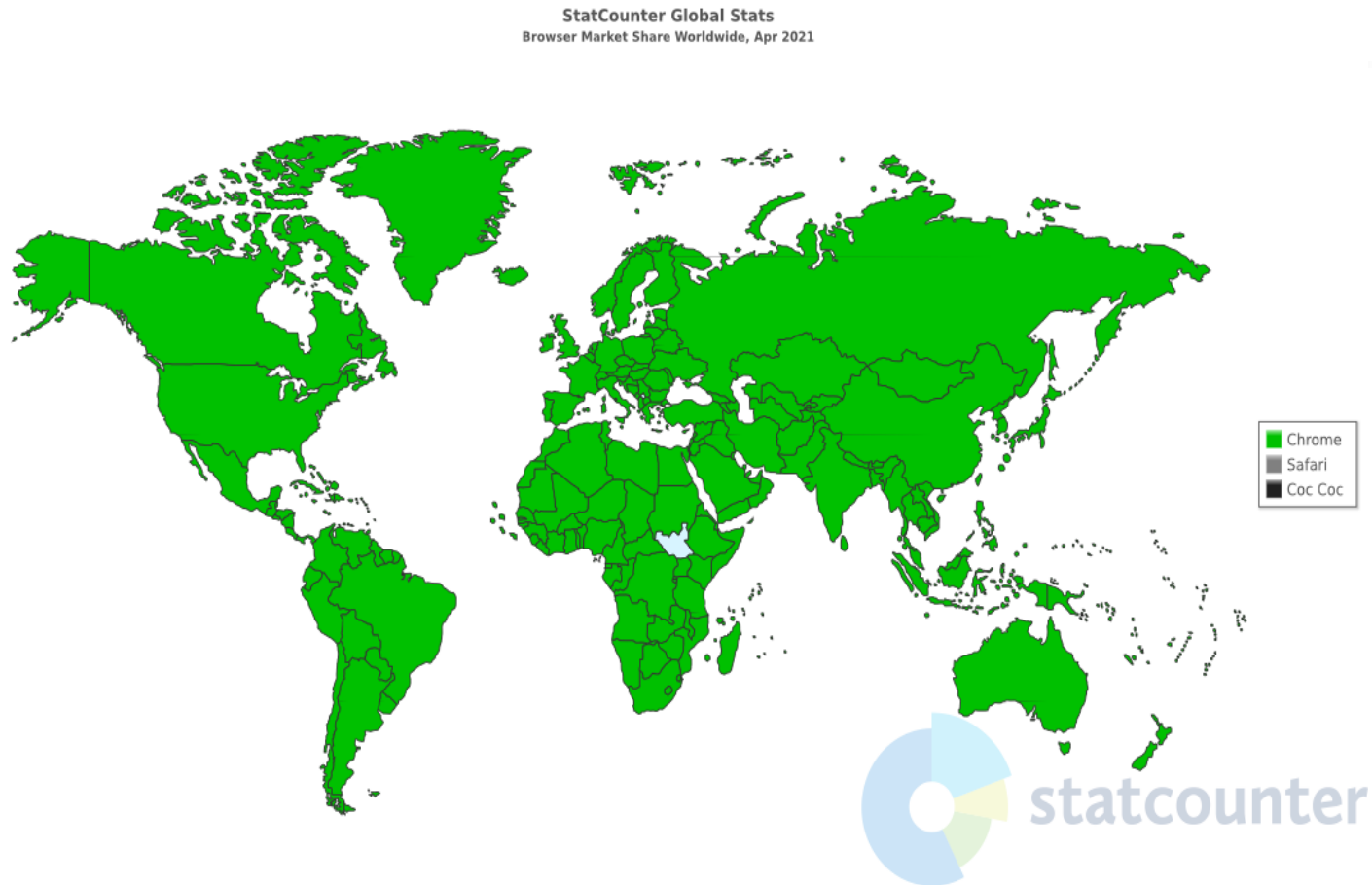
# WWW Clients

- The Web is designed like all the client/server applications
  - The **client** is called a “**browser**”
  - The **server** is where the **data is stored** and it is software that runs on **well known port (80)** ... **usually**
- The browser and server talk using a protocol – **HTTP**
- We already know from past experience that this architecture gives us client options
  - Netscape, Internet Explorer, Maxthon, Mozilla, Firefox, Lynx, ...



**Firefox**

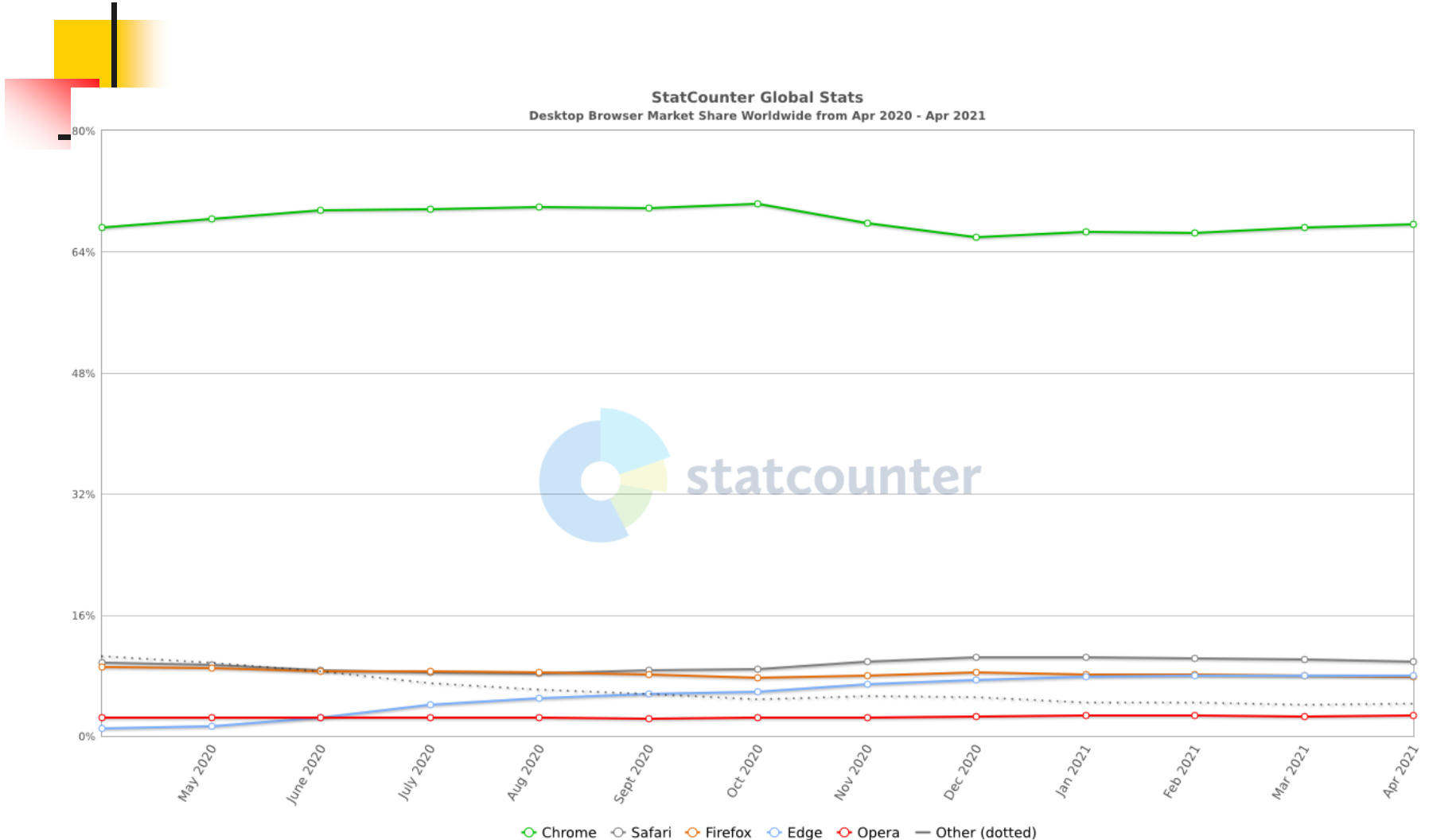
# Web Browsers Statistics(1)



Most used web browsers in country or dependency as of  
April 2021, according to [Statcounter](https://gs.statcounter.com/browser-market-share#monthly-202104-202104-map)

<https://gs.statcounter.com/browser-market-share#monthly-202104-202104-map>

# Web Browsers Statistics(2)-Desktop

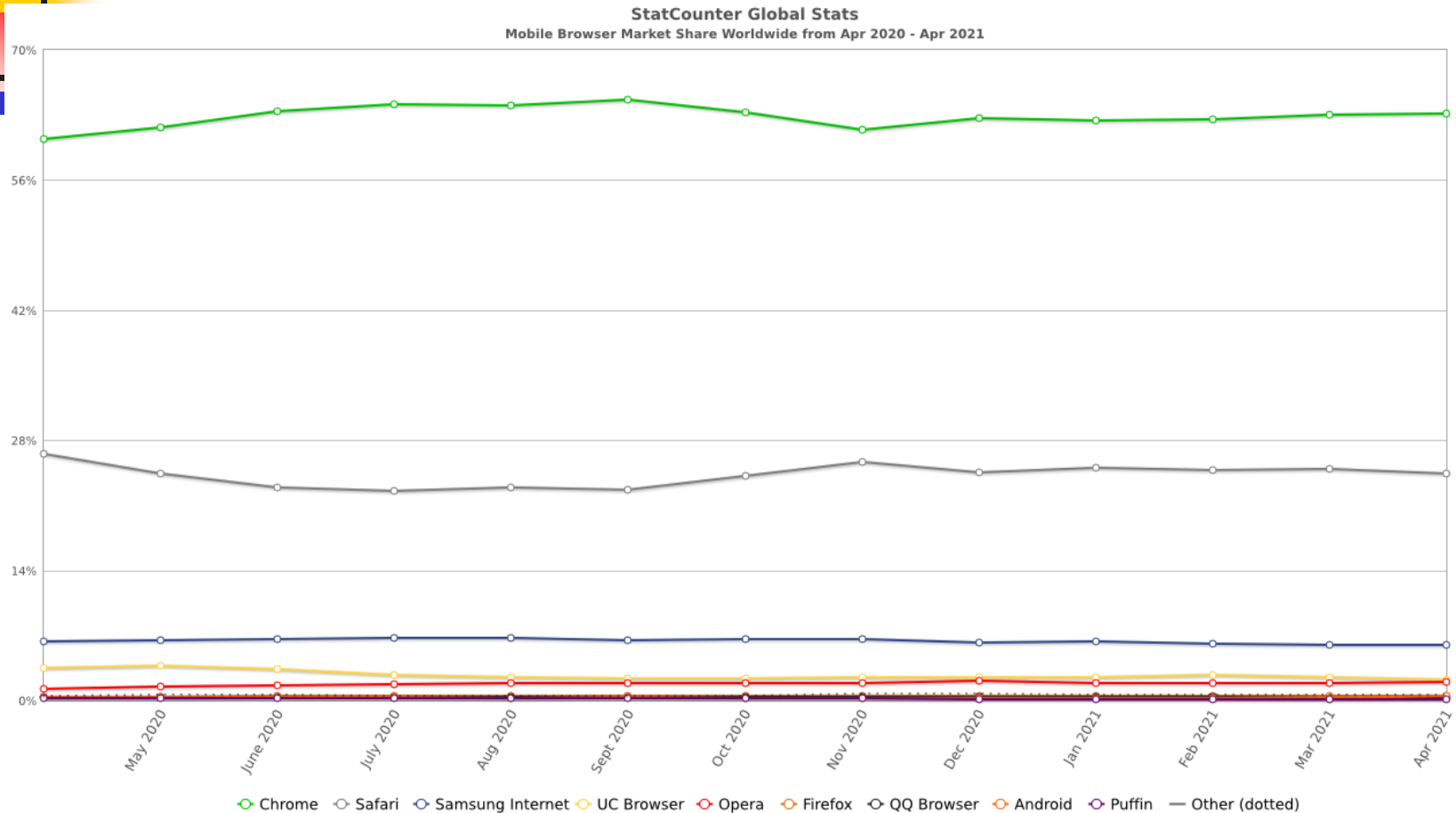


202004-202104, Desktop Browser Market Share

<http://gs.statcounter.com/browser-market-share/desktop/worldwide>



# Web Browsers Statistics(3)-Mobile



202004-202104, Mobile Browser Market Share

<http://gs.statcounter.com/browser-market-share/mobile/worldwide>



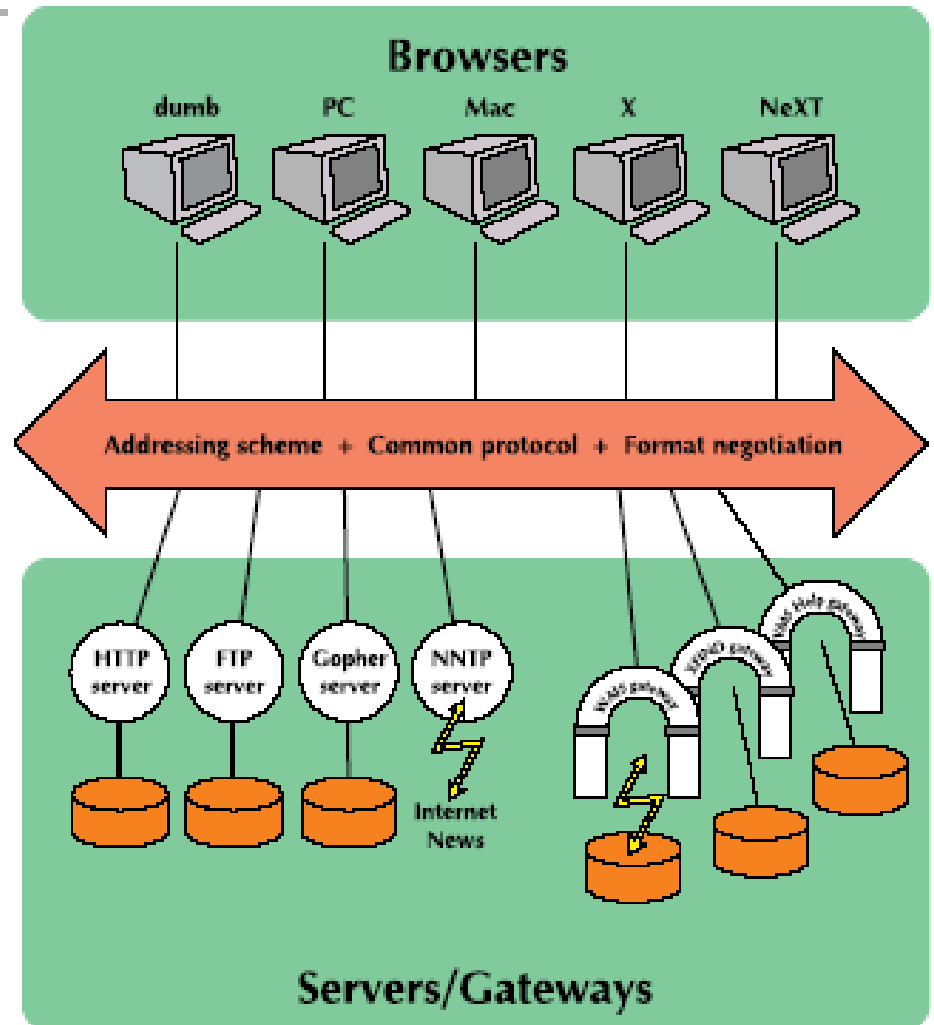
# Basic Client Properties (1)

---

- All the different browsers show us the **same information** but they **display it differently** (depending on their **capabilities**)
- In front of each Web address there is an **http://** to indicate to the browser that it is talking **HTTP**, the protocol of the Web
- A user on a client machine uses a browser to download a Web page by either entering a **URL** or clicking on a **HyperLink**

# Basic Client Properties (2)

- Web browsers are often called **Universal Clients** because most can talk other protocols besides HTTP
  - ftp://home.domain: to use our Web browser as an FTP client
  - telnet://home.domain
  - ...
- The Web is capable of accessing data on many different Internet services:
  - Web pages, FTP, Email service, file directories, Telnet services, HTML, plain ASCII, etc.





# WWW Servers (1)

---

- The server is **software** that is running on a remote location. Its job is to **make “pages” available** to the client - so when a client requests a page the server responds appropriately
- Web servers are typically on Unix or Windows NT boxes rather than on individual PCs
- Popular Web Servers:
  - On Unix - Apache, On Windows NT - IIS (Internet Information Server), Both - Netscape's Web Server



# WWW Servers (2)

---

- Every Web site has a server process listening to **TCP** port **80** for incoming connections from clients – normally browsers
- After a connection has been established, the client sends one **request** and the server sends one **response**
- Then the **connection is released**
- The protocol that defines the legal request and response is **HTTP**
- The operation is **Stateless**



# URLs (Uniform Resource Locators)

---

- The global address of a Web page is described by its URL
- URLs identify
  - the **protocol** you want to talk
  - the **site (domain name or IP Address)** you want to go to
  - possible the **item** you want to see
- They have the form:
  - **protocol://hostname [:port]/directory/item-you-want**

**Resources can be dynamically generated on server upon query  
(Dynamic documents)**



# Structure Of URLs

---

- A URL consists of three parts:
  - The protocol – for example **http** or **ftp**
  - The DNS name of the host
  - The directory and file name



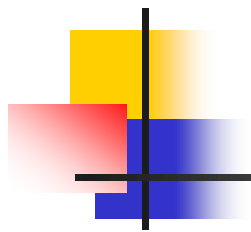
- Protocol: `http` by default
- Port: `80` by default
- `Index.html`, `index.htm`, `default.htm`, `default.asp` etc. are assumed if no file-name given



# Some URLs Examples

Protocol	Use	Example
http	Web pages	http://www.elec.qmul.ac.uk
ftp	File transfer	ftp://elec.qmul.ac.uk/pub/info.doc
file	Local files	file:///D:/src/multim/filter.txt
mailto	E-mail	mailto:cip@elec.qmul.ac.uk
telnet	Remote login	telnet://www.elec.qmul.ac.uk





# WWW Standards



# WWW Standards

---

- URL
  - RFC 1630, RFC 1738
  - Many RFCs define the URL used for telnet, gopher, mailto, POP, IMAP, etc.
- HTML
  - RFC 2854
- HTTP
  - RFC 2616: defines HTTP/1.1
  - RFC 2617: defines HTTP Authentication (Basic and Digest Access Authentication)



# HTML – HTML standards

---

- HTML is the agreed upon markup language for the Web
- Currently several versions are available
  - HTML 1.0 - most basic tags
  - HTML 2.0 - forms support
  - HTML 3.0 - vendor specific tags crept in
  - HTML 3.2 - current standard, scaled-back 3.0
  - HTML 4.01 - current recommended
  - HTML 5.0 – newest standard
  - XHTML (eXtensible HyperText Markup Language) 1.0/1.1/2.0 – XML based, more extensible, more flexible
- Depending on the **browser** you use and what **version** you use, pages can look different because different browsers support different HTML versions
- Differences between HTML and XHTML
  - <http://www.w3.org/MarkUp/2004/xhtml-faq>

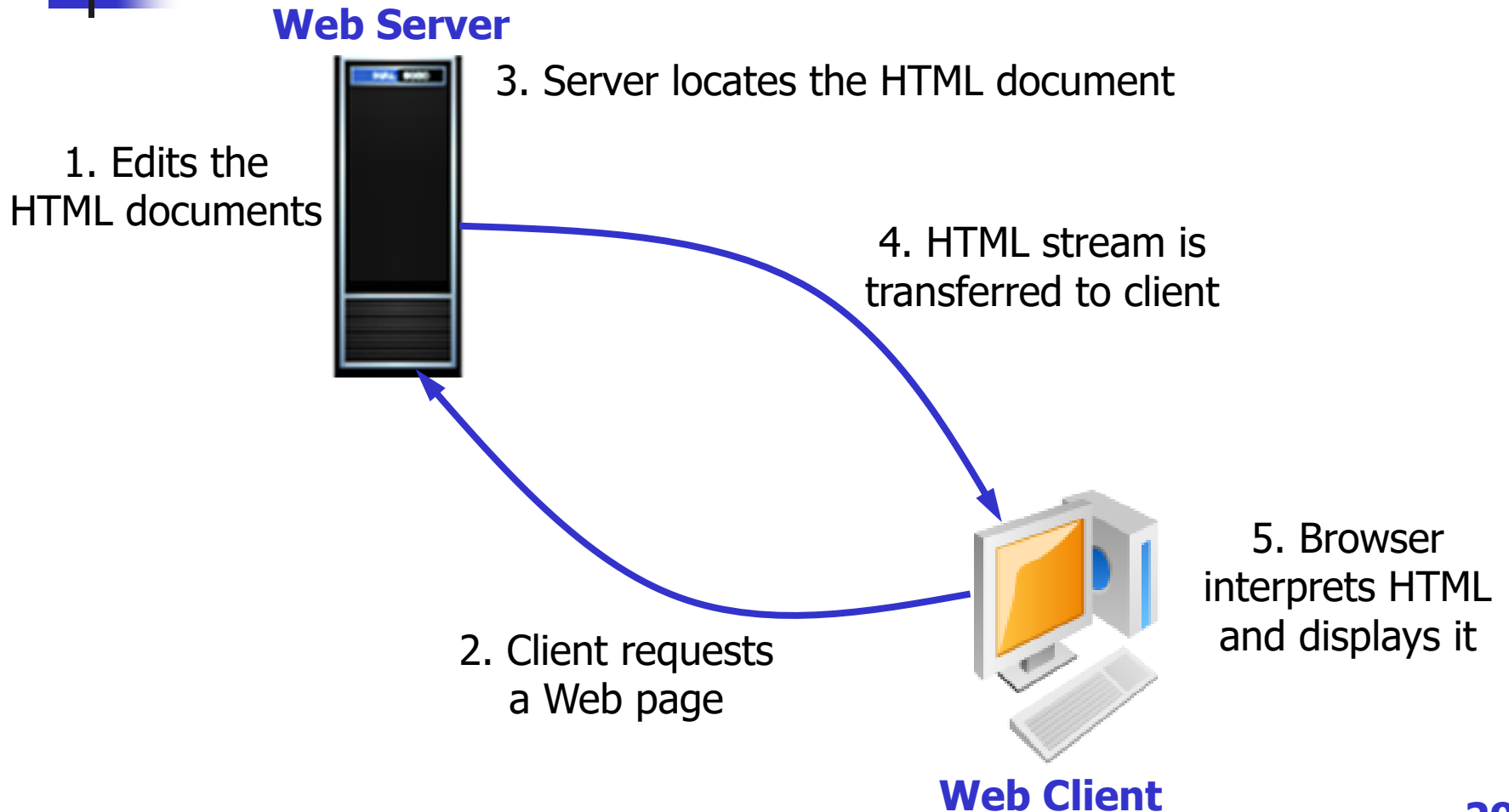


# Static vs. Dynamic

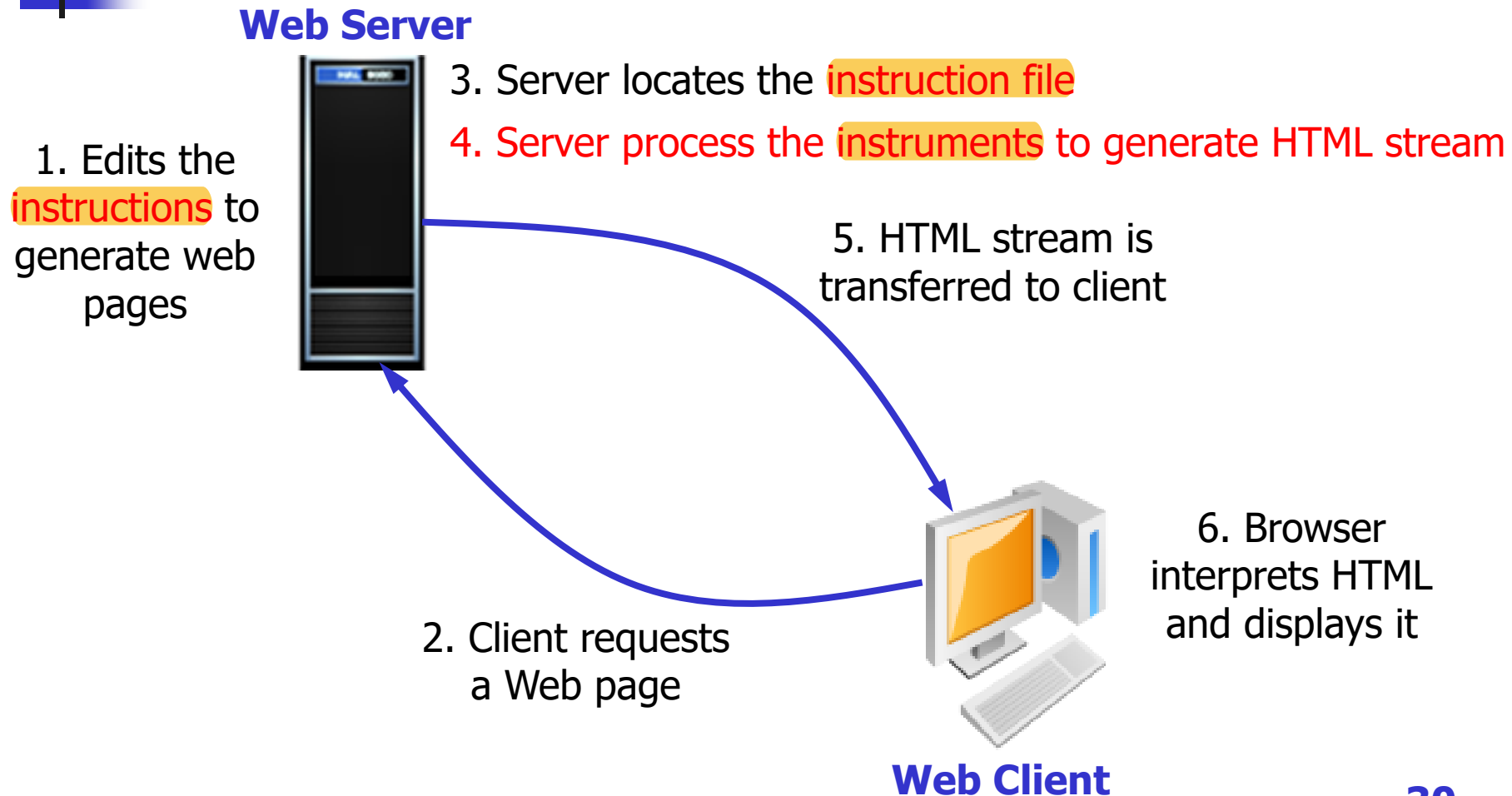
---

- At the beginning, WWW was made up of **static documents**
  - Each URL corresponded to a single file stored on some hard disk
  - Edit in HTML format
  - .html, .htm
- Today - many of WWW documents are **built at request time**
  - The URL doesn't correspond to a single file
  - Examples: website access counter, WWW based date-time server, BBS, ...
  - Generated dynamically by ASP, JSP, VB Script, PHP, CGI or other programs
  - .asp, .shtm, .php, .cgi etc.
- Why dynamic documents?
  - automation of web site maintenance
  - customized advertising
  - database access
  - shopping carts
  - date and time service
  - jobs for ElecEng students

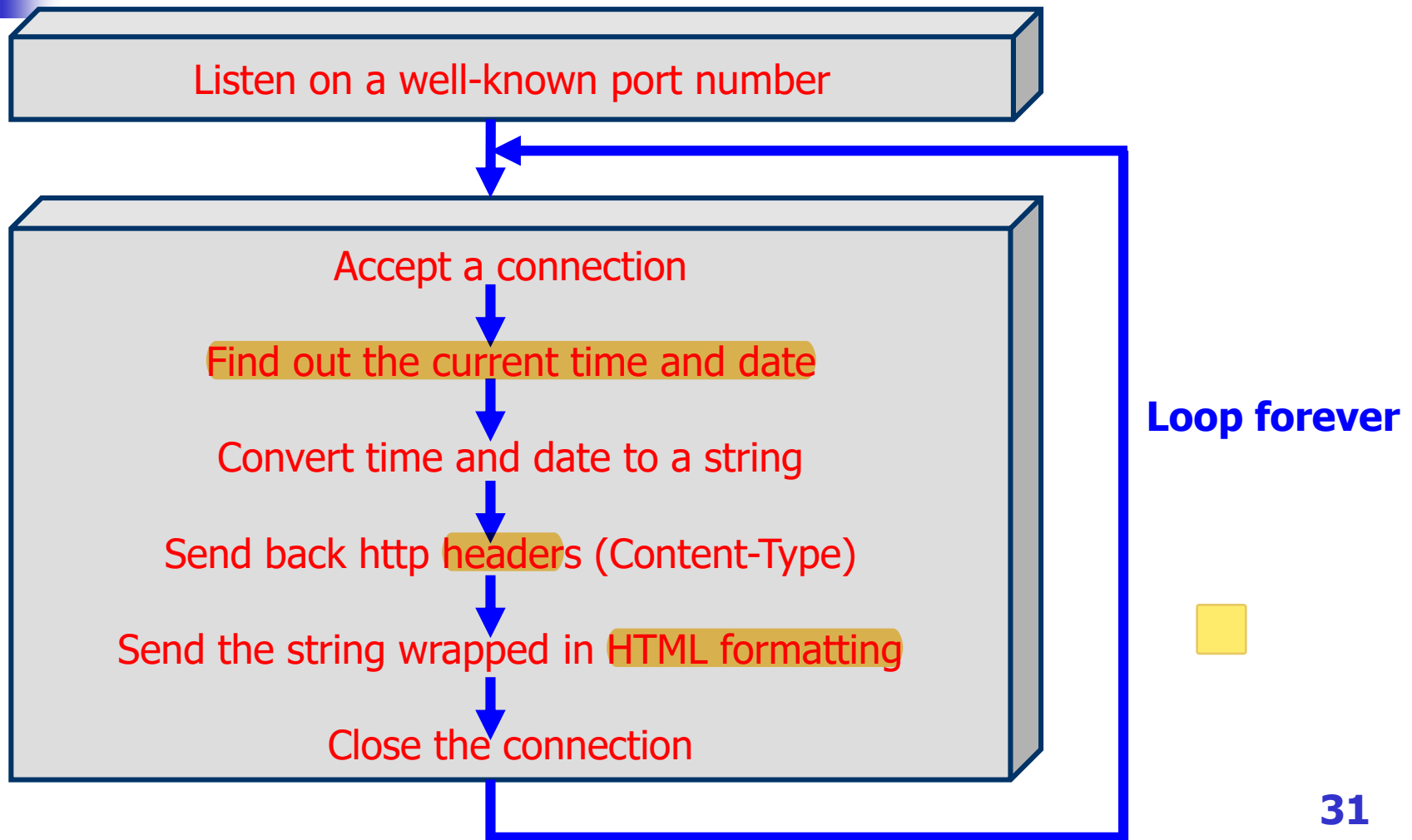
# Procedure Of Static Pages



# Procedure Of Server-based Dynamic Pages




# Example: WWW based time and date server





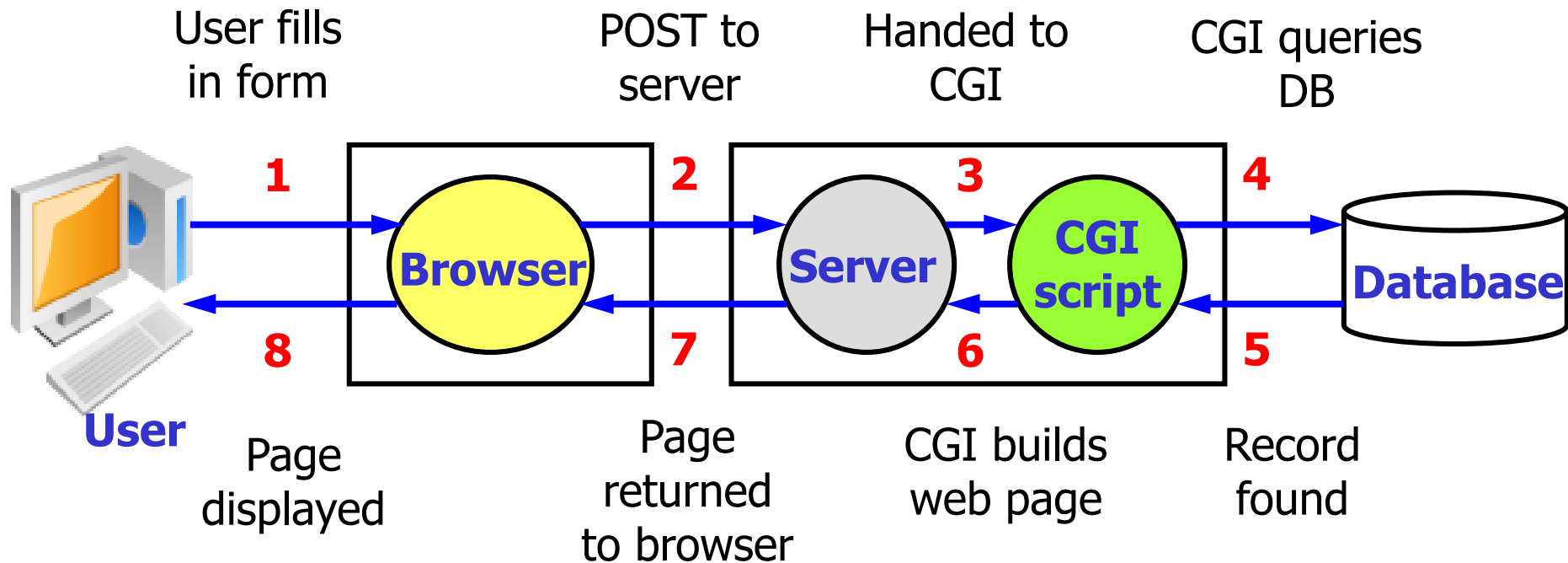
# CGI (Common Gateway Interface)

---

- The Common Gateway Interface (CGI) is a standard for **interfacing external applications with information servers**, such as HTTP or Web servers.
- A plain HTML document that the Web client retrieves is static, which means it exists in a constant state: a text file that doesn't change. A CGI program, on the other hand, is **executed in realtime**, so that it can **output dynamic information** for the server. 
- The Web server executes a CGI program to transmit information to the database engine, receive the results and display them to the client. This is an example of a **gateway**. Currently version is 1.1.
- A CGI program is basically the equivalent of letting the world run a program on your system. For safety, security precautions are taken.



# CGI – Procedure





# HTTP – Basics

---

- The heart of the Web
- Features
  - Application layer protocol for client/server communication
  - Request/response based
  - Stateless ☐
  - Bi-directional transfer
  - Capability negotiation
  - Support for cache ☐
  - Support for intermediaries: HTTP proxy ☐



# HTTP – HTTP Transaction

Establish connection

- TCP connection set up
- uses a port number as application reference
- usually port 80

Client request

- HTTP message sent with a request line
- request-line = **method** **URL** **HTTP version**

Server response

- server sends HTTP message and optionally requested data
- resp-message = **HTTP version** **status code**  
reason-phrase [optional stuff]

Connection terminated

- usually the server
- sometimes the client “stops” it
- anything else, whoever notices terminates



# HTTP – Status Codes

---

- 1xx – for information only
- 2xx – action successful
- 3xx – further action needed (redirect) ■
- 4xx – client request error ■
- 5xx – server error



# HTTP – Getting Remote Web Pages

---

- The browser determines the URL
- Browser asks DNS for the IP address of web-page being referred to
- DNS returns the IP address to the browser
- The browser makes a TCP connection to port 80 at the web-page IP address
- The browser sends a get request, eg.
  - GET /dir/FileName.html HTTP/1.0
- The remote server sends the file FileName.html
- The TCP connection is released
- The browser displays all the text in FileName.html
- The browser fetches and displays all the images in FileName.html



# HTTP – HTTP Methods

Method	Description
GET	retrieve document specified by URL
PUT	store specified document under given URL
HEAD	identical to GET except that the server <b>MUST NOT</b> return a <b>message-body in the response</b>
OPTIONS	retrieve information about available options
POST	give information (eg. annotation) to the server
DELETE	remove document specified by URL
TRACE	loopback request message
<b>CONNECT</b>	<b>reserved for use with a proxy</b>

# HTTP Request and Response Examples

Request Line

**method** **URL** **HTTP version**

GET /chn/yxsx/index.htm HTTP/1.1

Headers

**Header name: header value**

Host: www.tsinghua.edu.cn

Connection: close

User-Agent: Mozilla/5.0

Accept-Language: cn

Blank Line

[blank line]

Entity body

**Often not used in request**

Response Line

HTTP version **status code** phrase

HTTP/1.1 404 Not Found

Headers

**Header name: header value**

HTTP/1.1 301 Moved Permanently

Location: <http://www.xyz.com/index.html>

Blank Line

HTTP/1.1 200 Ok

Entity body

**used in some messages**

[blank line]

<data>

# HTTP – An ASCII/MIME protocol

- Because HTTP is an **ASCII / MIME protocol**, it is simple for a user at a terminal to communicate directly to a Web server
  - ASCII: defined in RFC **5322**
  - MIME: **Multipurpose** Internet Mail Extension
- Each interaction consists of one **ASCII request**, followed by one **RFC5322 / MIME-like response**
  - e.g. Content-type: text/html
  - Data type/subtype
    - text/html
    - text/plain
    - image/gif
    - video/mpeg
    - application/msword
    - etc.





# HTTP – An ASCII/MIME protocol

## ■ Content types and subtypes defined by MIME

Type	Subtype	Description
Text	Plain	Unformatted text
	Richtext	Text including simple formatting commands
Image	Gif	Still picture in GIF format
	Jpeg	Still picture in JPEG format
Audio	Basic	Audible sound
Video	Mpeg	Movie in MPEG format
Application	octet-stream	An uninterpreted byte sequence
	Postscript	A printable document in PostScript
Message	RFC5322	A MIME RFC 5322 message
	Partial	Message has been split for transmission
	External-body	Message itself must be fetched over the net
Multipart	Mixed	Independent parts in the specified order
	Alternative	Same message in different formats
	Parallel	Parts must be viewed simultaneously
	Digest	Each part is a complete RFC 2822 message



# HTTP/1.1 Performance Enhancements

---

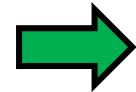
- HTTP/1.0 is a “stop and wait” protocol
  - Separate TCP connection for each file
    - Connect setup and tear down is incurred for each file
    - Inefficient use of packets
    - Server must maintain many connections
- HTTP/1.1 specification focus on performance enhancements
  - Persistent connections
  - Pipelining
  - Enhanced caching options
  - Support for compression



## HTTP/1.1 Persistent Connections and Pipelining

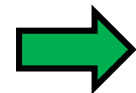
---

- Persistent connections



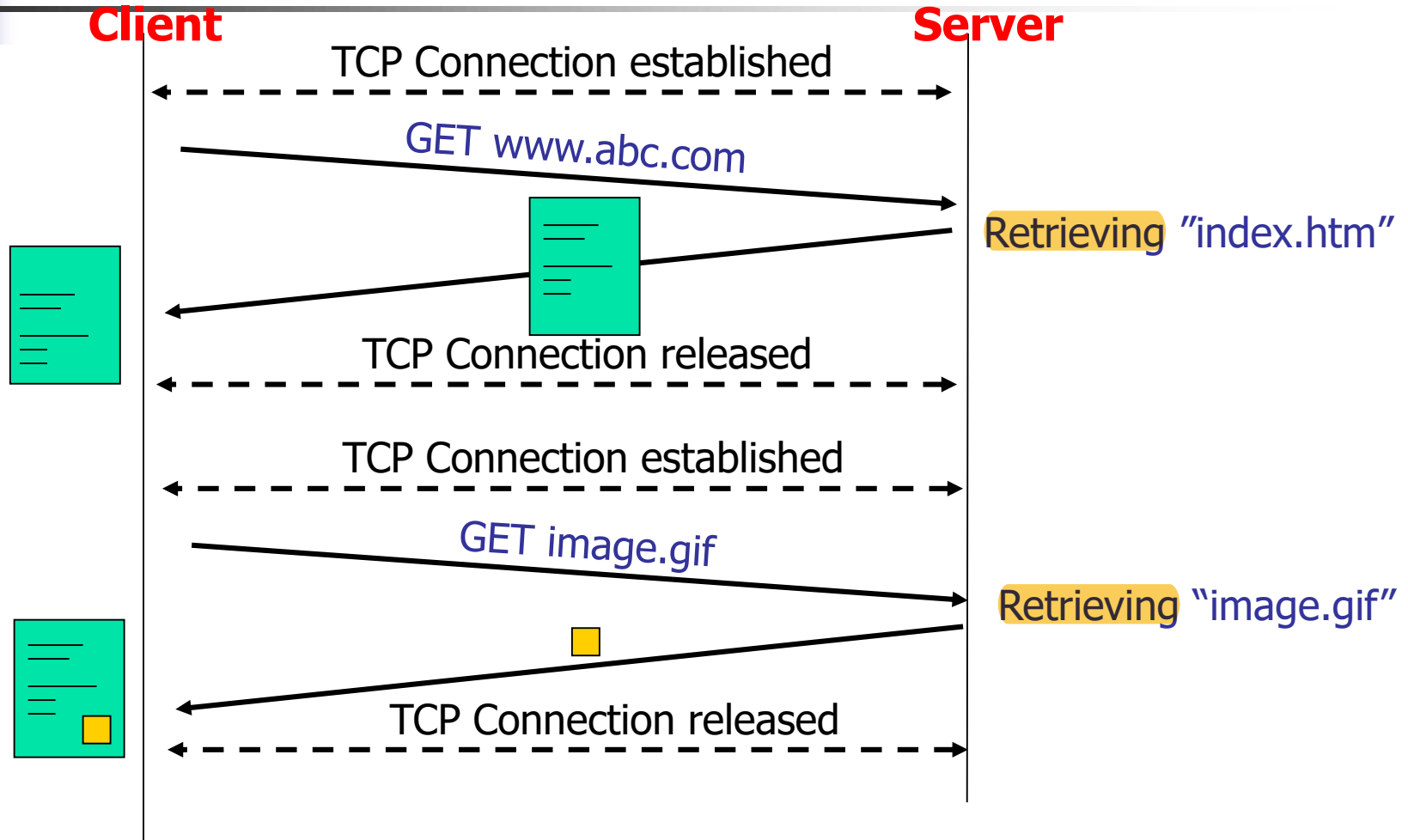
- Use the same TCP connection(s) for transfer of multiple files
- Reduces packet traffic significantly

- Pipelining

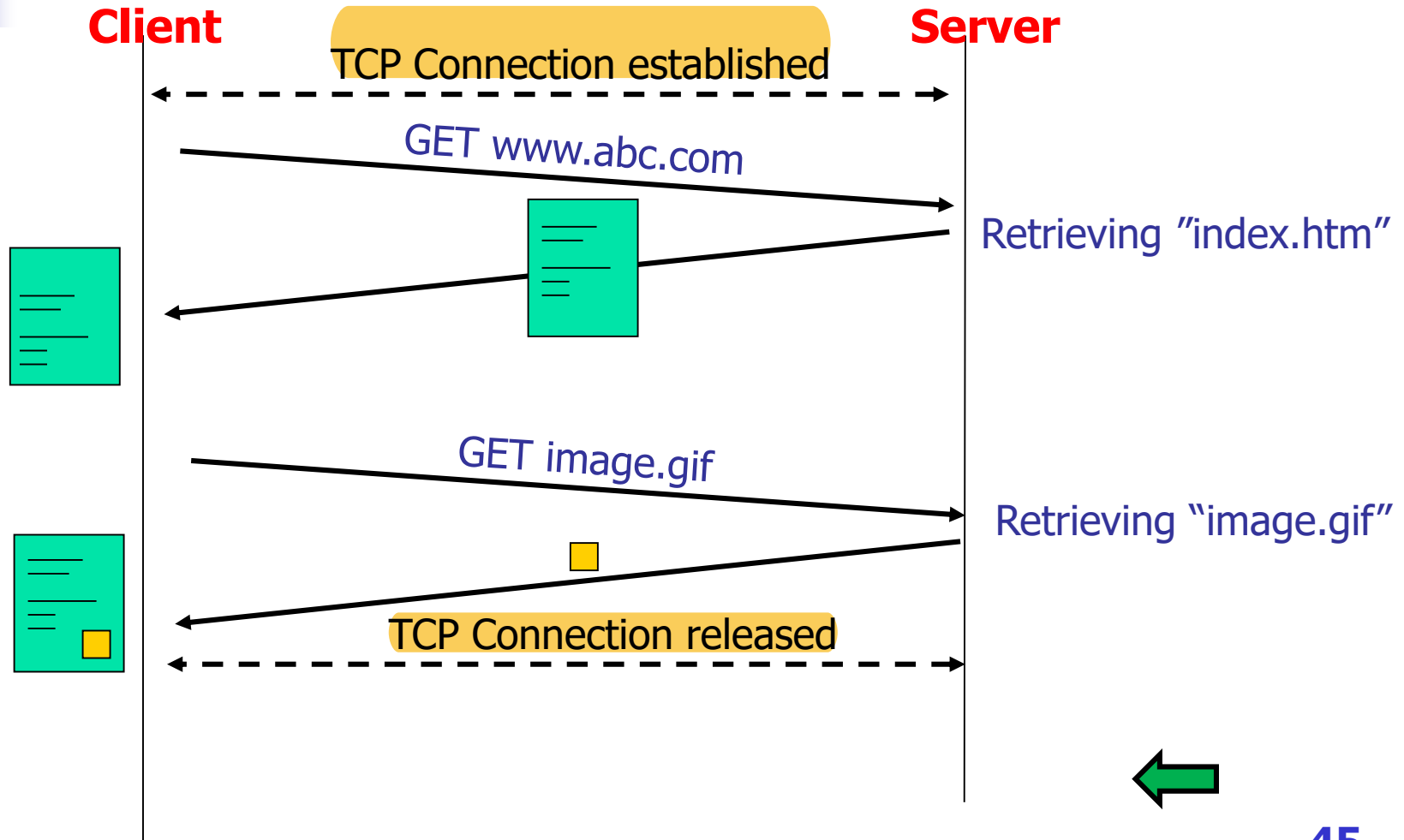


- **Multiple HTTP requests** can be written out to a socket together without waiting for the corresponding responses.
- Pack several HTTP requests into one TCP/IP packet

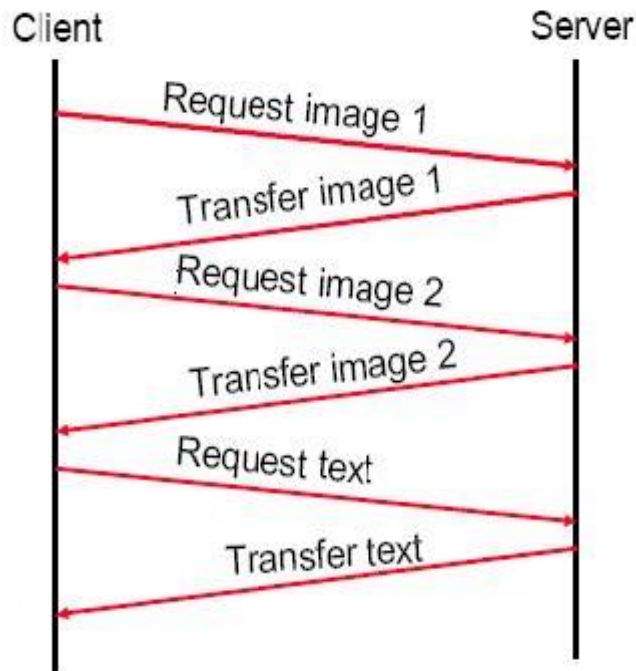
# Example of Non-persistent connections



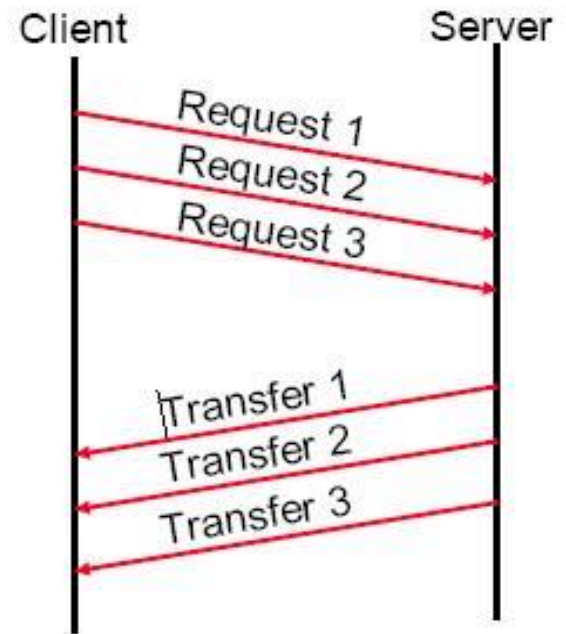
# Example of Persistent Connections



# Example of Pipelining



Non-pipelining



Pipelining



# User-server state: cookies

---

- Many major Web sites use cookies

## Example:

- Susan always access Internet from PC
- She visits specific e-commerce site for first time
- when initial HTTP request arrives at site, site creates:
  - unique ID
  - entry in backend database for ID

## Four components:

- 1) cookie header line of HTTP *response* message
- 2) cookie header line in HTTP *request* message
- 3) cookie file kept on user's host, managed by user's browser
- 4) back-end database at Web site

# Cookies: keeping "state"

client

server



cookie file



usual http request msg

usual http response  
**Set-cookie: 1678**

usual http request msg  
**cookie: 1678**

usual http response msg

one week later:



usual http request msg  
**cookie: 1678**

usual http response msg

Amazon server  
creates ID  
1678 for user

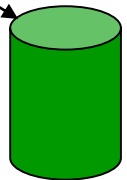
create  
entry

cookie-  
specific  
action

cookie-  
specific  
action

access

access

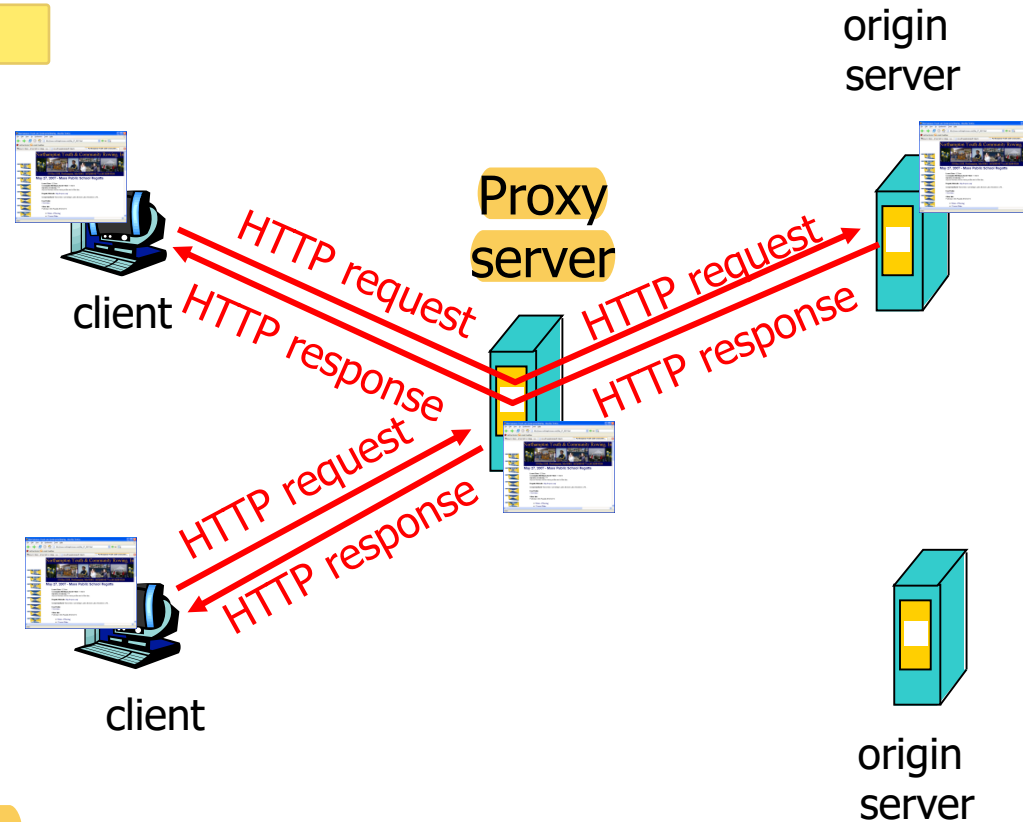


backend  
database



# Web Caches (Proxy Server)

- Motivation: satisfy client request without involving origin server
- User sets browser: Web accesses via a proxy server
- Browser sends all HTTP requests to proxy server
  - If requested file in cache: proxy server returns file
  - else proxy requests file from origin server, then forwards to client



# Conditional get

- Server does not send required files if **cache has up-to-date cached version**

- **cache**: specify **date** of **cached copy** in HTTP request

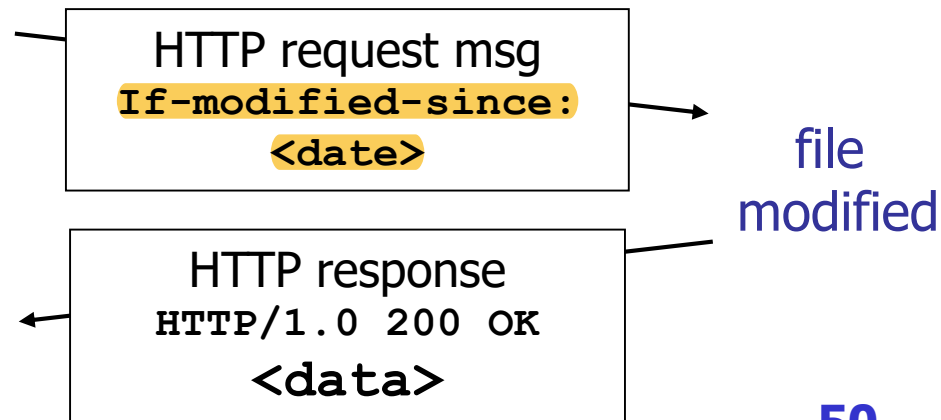
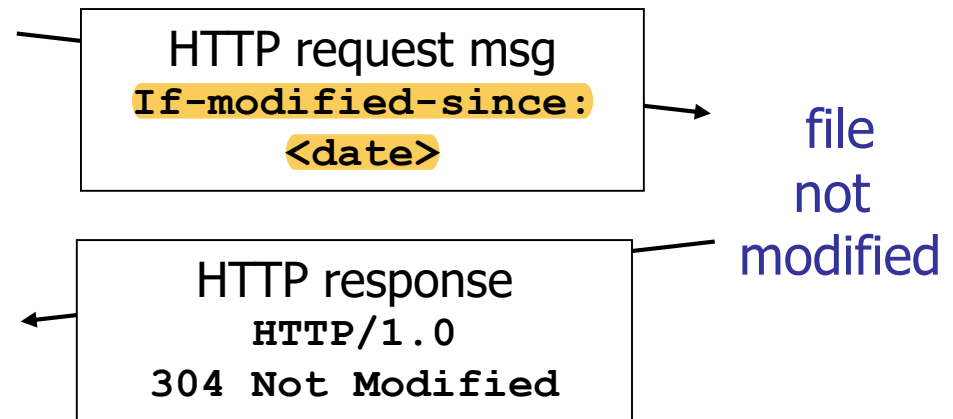
**If-modified-since: <date>**

- **server**: response contains no object if cached copy is **up-to-date**:

**HTTP/1.0 304 Not Modified**

cache

server

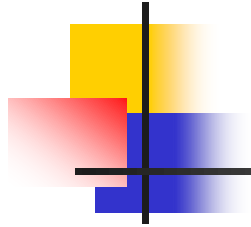




# HTTPS

---

- Hypertext Transfer Protocol Secure (HTTPS)
- HTTP on top of the SSL(Secure Sockets Layer)/TLS (Transport Layer Security) protocol
- A communication protocol for secure communication over a computer network, with especially wide deployment on the Internet



# Summary



# Summary

---

- Terminologies
  - WWW, the Web, W3
  - URL
  - HTML
  - HTTP
- WWW components
  - Client/browser
  - Web server
- URL
  - Structure
  - Used for different services
- HTML
  - Basic web page structure
  - Static vs. dynamic
  - CGI
- HTTP
  - Features
  - Transaction
  - Methods and responses
  - Performance enhancement of HTTP 1.1
  - Idea of cookies
  - HTTP Proxy



# Questions

---

- Does a web address equal to a domain name?
- What are the disadvantages of the stateless feature of HTTP?
- How is the procedure of client-based dynamic pages?
- What is the cookies?
- How does the HTTP proxy work?



# Useful URLs

---

- W3C
  - <http://www.w3.org/>
- HTML
  - <https://www.w3.org/TR/html50/>
  - <http://www.jmarshall.com/easy/html/>
- HTTP
  - <http://www.jmarshall.com/easy/http/>
- A detailed description of Internet history
  - <http://www.zakon.org/robert/internet/timeline/>



# Abbreviations

---

<b>CGI</b>	Common Gateway Interface
<b>HTML</b>	HyperText Markup Language
<b>HTTP</b>	HyperText Transfer Protocol
<b>MIME</b>	Multi-purpose Internet Mail Extension
<b>URL</b>	Uniform Resource Locator
<b>WWW</b>	World Wide Web