WEBTEK SYSTEM & TECHNOLOGIES

* Internet (Internetwork) – 1969
* massive global network of networks
* to the global (worldwide) information system

\*system – components that are organized

* Network
* logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons
* is able to support communications using Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible Protocol
* provides, uses, or makes accessible, either publicly or privately, high level services layered on the communications and related infrastructure
* interconnection of devices
* nodes - devices that are interconnected
* interconnection technology
* wired
* wireless
* protocols (standards)
* IP address
* identifies each and every node
* IPv4 – 32 bit IP address
* IPv6 – 64 bit IP address
* 1972 – email
* 1989 – world wide web
* created by Sir Tim Berners-Lee, who worked at CERN
* also developed HTTP, HTML, URL
* created first web server and first web client
* Before www
* WAIS (Wide Area Information Servers)
* contacts, stores index, creates WAIS client (like a browser)
* one document at a time
* Gopher (protocol)
* hierarchy of information (vs WAIS)
* one document at a time (like WAIS)
* search engines are named after Archie comic characters (Archie, Veronica, Jughead, etc)
* Usenet
* post, comment, attachments
* Web Server
* program that hosts websites
* websites – web documents
* HTML – web document linkers
* URL – web server address for web documents
* Web Client
* web browsers
* HTTP – requests and sends to and from web server
* HyperText Transfer Protocol (HTTP)
* application layer communications protocol used to access resources (hypertext/hypermedia) on the world wide web
* invented by Sir Tim Berners-Lee
* jointly developed by W3C and IETF
* version history:
* HTTP 0.9 (1991)

\*Connection -> Request -> Response -> Disconnection

* HTTP 1.0 (RFC 1945, May 1996)
* HTTP 1.1 (RFC 2068 Jan 1997, RFC 2616 June 1999, RFC 7230-7235) (June 2014)
* HTTP 2 (RFC 7540 May 2015)
* backwards compatible

\*SPDY ( Google)

* Fundamentals
* HTTP runs on top of TCP/IP using TCP port 80 by default, or TCP port 443 for HTTPS

(HTTPS over SSL/TLS) \*encrypted

* SSL : Secure Socket Layer
* TLS : Transfer Layer Socket
* HTTP is based on a client-server architectute
* clients (User Agents – UA)
* web browsers, web crawlers/spiders, other end user tools and applications
* servers
* origin servers - physical
* proxy servers, gateways, tunnels – intermediary nodes/hosts
* HTTP uses a request-response standard protocol
* server processes the request and replies with an HTTP response message
* HTTP is a stateless communications protocol
* servers do not keep information about clients in-between requests ( serve and forget)
* HTTP provides support for other functionalities such as:
* cache control
* content media type (MiME – Multipurpose internet Mail Extensions) specification
* language and character set specifications
* content/transfer codings
* content negotiation
* client-server protocol negotiations
* persistent connections
* request pipelinng
* authentication/authorization
* HTTP Resource addressing
* resources are identified using URIs (Uniform Resource Identifier; RFC 3986) or more specifically, HTTPS URLs (Uniform Resource Locator)
* scheme
* HTTP or HTTPS
* authority
* user information or authentication credentials (deprecated)
* host
* domain name (resolved to an IP address using DNS) of the server where the resource resides (or will be created)
* port number
* optional
* HTTP (port 80) or HTTPS (port 223)
* path to resource
* resolved relative to the document root on the server
* may refer to a static or dynamic resource
* query
* typically provided as key=value pairs, with ampersand (&) separators between key/value pairs
* may be URL encoded
* fragment identifier
* starts with a pound sign (#)
* only on client side

can be substituted with the IP address

<http://usr:pwd@server.org:81/info/profile.php?id=1234#addr>

scheme

query

path

authority

* REQUEST MESSAGE
* Request Line (CRLF – terminated line consisting of three space-separated values)
* method
* request URI
* HTTP protocol version
* message headers (general, request, and/or entity headers)
* HTTP 1.1 requires at least the host request header to be provided
* Empty Line (CRLF)
* Message Body, aka payload (optional)
* RESPONSE MESSAGE
* Status Line (CRLF – terminated line consisting of three space-separated values)
* HTTP protocol version
* Status Codes
* Information (1xx)
* Success (2xx)
* Redirection (3xx)
* Client Error (4xx)
* Server Error (5xx)
* Reason phrase
* Message headers (general, response, and/or entity headers)
* Empty Line (CRLF)
* Message Body (optional)
* HTTP Request Methods
* GET
* transfer a current selected representation of resource identified by the request URI, the retrieved resource is returned in the message body of the response as an entity
* most commonly used HTTP method
* must be supported by all compliant general purpose servers
* HEAD
* same as GET except that the entity is not included in the response (i.e. returns only the statues line and headers returned by a GET request, without the message body)
* used to retrieve metadata about the entity implied by the request without the transferring the entity itself (e.g. to test the link solidity or resource modification)
* like GET, must be supported by all compliant general purpose servers
* POST
* perform resource-specific processing of the entities enclosed in the message body by the resource identified by the request URI
* typically used to get --- HTML from data
* PUT
* store the enclosed entity in the message body under the specific request URI (i.e. the resource identified by the request URI is either created or replaced, using the enclosed entity)
* DELETE
* remove the resource associated with the request URI
* OPTIONS
* request information about the communication …
* TRACE
* request a loop-back of the request message (i.e. request the server to echo back to the client the received request message)
* typically used for testing/diagnostics of the request/response chain
* CONNECT
* request the establishment of a tunnel to the destination origin server, and if successful, thereafter restrict its behavior to blind forwarding of packets, in both directions, until the tunnel is closed

\* Extension Methods

- e.g WebDAV (RFC 4918) : PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK

\* Safe Methods

- does not modify resources (GET, HEAD, OPTIONS, TRACE)

\* Idempotent Methods

- always has the same results however many times it is used ( GET, HEAD, OPTIONS, TRACE, CONNECT, PUT, DELETE)

* Message Headers
* General Header fields
* Cache-control
* Connection
* Date
* Pragma
* Trailer
* Transfer-Encoding
* Upgrade
* Via
* Request Header fields
* Accept
* Accept-Charset
* Accept-Encoding
* Accept-Language
* Authorization
* Host
* If-Match
* If-Modified-Since
* If-None-Match
* If-Range
* If-Unmodified-Since
* Max-Forwards
* Proxy-Authorization
* Range
* Referer
* TE
* User-Agent
* Response Header fields
* Accept-Ranges
* Age
* ETag
* Location
* Proxy-Authenticate
* Retry-After
* Server
* Vary
* WWW-Authenticatie
* Entity Header fields
* Allow
* Content-Encoding
* Content-Language
* Content-Length
* Content-Location
* Content-MD5
* Content-Range
* Content-Type
* Expires
* Lost-Modified
* HTTP Status Codes
* Information (1xx)
* 100 Continue
* 101 Switching Protocols
* Success (2xx)
* 200 OK
* 201 Created
* 202 Accepted
* 203 Non-Authorative Information
* 204 No Content
* 205 Reset Content
* 206 Partial Content
* Redirect (3xx)
* 300 Multiple Choices
* 301 Moved Permanently
* 302 Found
* 303 See Other
* 304 Not Modified
* 305 Use Proxy
* 306 (unused)
* 307 Temporary Redirect
* Client Error (4xx)
* 400 Bad Request
* 401 Unauthorized
* 402 Payment Required
* 403 Forbidden
* 404 Not Found
* 405 Method Not Allowed
* 406 Not Acceptable
* 407 Proxy Authentication Required
* 408 Request Time-out
* 409 Conflict
* 410 Gone
* 411 Length Required
* 412 Precondition Failed
* 413 Request Entity Too Large
* 414 Request-URI Too Large
* 415 Unsupported Media Type
* 416 Requested Range Not Satisfied
* 417 Expectation Failed
* 426 Upgrade Required
* Server Error (5xx)
* 500 Internal Server Error
* 501 Not Implemented
* 502 Bad Gateway
* 503 Service Unavailable
* 504 Gateway Time-out
* 505 HTTP Version Not Supported