**Internet**

- A series of computer networks connected globally and communicate using the standard Internet Protocol Suite (TCP/IP). Any communication between two computers worldwide is possible when connected to the Internet.

- Created and tested in ARPANET (1964) with four computers connecting to each other.

- Email was introduced (1972).

**World Wide Web**

- A way of retrieving and reading information shared throughout the Internet, using the HTTP/ HTTPS protocol to store and share information. Web browsers are used to view documents and files are stored in Web readable formats such as HTML and CSS.

- Created by Tim Berners-Lee in his research at Cern (1989).

**Intranet**

- Limited network connection within a certain region or area, within a boundary.

**HTTP**

- Layered approach

- Plain Text Data

- Application layer communications protocol used to access resources (hypertext/hypermedia) on the World Wide Web

- Created by Tim Berners-Lee

- Jointly developed by the W3C and the IETF

- Version history

* HTTP 0.9 (1991)
* HTTP 1.0 (RFC 1945, May 1996)

HTTP 1.1 (RFC 2068 January 1997, RFC 2616 June 1999), RFC 7230-7235 (June 2014)

* HTTP 2 (RFC 7540 May 2015, patterned after SPDY)

**HTTP Fundamentals**

- Runs on top of TCP/IP, using TCP port 80 by default, or TCP port 443 for HTTPS (HTTP over SSL/TLS)

- HTTP is based on a client-server, architecture

* Clients (user agents UA):

- Web browsers, web crawlers/spiders, other end user tools and applications

* Servers:

- Origin servers

- Proxy servers, gateways, tunnels

- HTTP uses a request-response standard protocol

* The client sends an HTTP request message to the server
* The 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

- HTTP provides support for other functionalities, such as:

* Cache Control – to cache or not to cache (Locality of Reference – only use when needed).
* Content Media Type (MIME - Multipurpose Internet Mail Extensions) specification – no reliance on file name extension and can be opened regardless.
* Language and character set specification (Unicode/ASCII)
* Content/Transfer Coding (byte by byte stream, compression up to 80-90%)
* Content Negotiation – agreement between handling of content by the server and the client (file type, language, compression)
* Client-server Protocol Negotiations (HTTP 1.0, 1.1 or 1.2)
* Persistent Connections – client retains connection to server after request and response have been sent and received.
* Request Pipelining – one requests at a time (wait for response).
* Authentication/authorization
* etc.

- Pre HTTP2 servers won’t volunteer data to clients. Client must request for the data before being given said data.

- HTTP 2 follows the “push” protocol which allows servers to volunteer info based on client requests.

**HTTP Resource Addressing**

- Uniform Resource Name (URN) – does not matter where location is; check data availability.

- Uniform Resource Locator (URL) – same as URN but also points out the location of the data.

- HTTP Resources are identified using URIs (RFC 3986) or more specifically, HTTP URLs

* Scheme (HTTP or HTTPS protocols)
  + Ex. *http:// or https://*
* Authority
* User information or authentication credentials (deprecated/optional).
* Host
* domain name (resolved to an IP address using DNS) of the server where the resource resides.
* Port number (optional)
* Ex. *serverhere.com:8080*
* Path to resource (resolved relative to the document root on the server)
* May refer to a static or dynamic resource.
* Ex. */where/here/there.pdf*
* Query
* Typically provided as key=value pairs, with ampersand (&) separators between key/value pairs.
* May be URL encoded.
* Fragment identifier
  + Denoted via a pound sign.

Complete URL example:

*http://serverhere.com:8080/where/here/there.pdf?here=school&stud=dudette#heading*

Scheme Host/Port Path Query Fragment Id.

**HTTP Request Message**

CRLF – Carrier Return and Line Field (used to note the termination of a line)

* Request Line (CRLF - terminated line)
* Method
* Request URI
* HTTP Protocol Versions
* Message Headers (general, request, and/or entity leaders)
* HTTP 1.1 requires at least the Host request header to be provided
* Empty Line (CRLF) – done with headers
* Message Body (payload, optional)

**HTTP Response Message**

* Status Line (CRLF - terminated line consisting of three space-separated values)
* HTTP Protocol Version
* 3-Digit Status Code
* Reason Phrase
* Message Headers (general, response, and/or entity headers)
* Empty Line (CRLF)
* Message Body (optional)

**HTTP Request Methods**

* Standard Methods

GET

* Transfer a current selected representation of the resource identified by the Request URI; the retrieved resource is returned in the message 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 (returns only the status line and headers returned by a GET request, without the message body).
* Used to retrieve metadata about the entity implied by the request without transferring the entity itself (tests for link validity or resource modification).
* Like GET, must be supported by all 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 in submitting HTML form data.
* Has a payload.

PUT

* Store the enclosed entity in the message body under the specified Request URI (resource identified by the Request URI is either created or replaced, using the enclosed entity).

DELETE

* Deletes the stored enclosed entity in the message body under the specified Request URI (resource identified by the Request URI is deleted).

OPTIONS

* + Request information about the communication.
  + Asks the server, “What can I do?”

TRACE

* + Request a loop-back of the request message (requests the server to echo back to the client the received request message).
  + Typically used for teaching/diagnostics of the request/response shown.

CONNECT

* + Request the establishment of a tunnel to the destination origin server, and if successful, thereafter restricts its behaviour to blind-forwarding of packet, in both directions, until the tunnel is closed.
  + Commonly used to create an end-to-end virtual connection.
* Safe Methods, Idempotent Methods, Cacheable Methods
* Extension Methods
  + Web DAV (RFC 4918)
  + PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK

**HTTP Request Method Types**

* Safe – no modification on server side after a request (GET, HEAD, OPTIONS, TRACE).
* Idempotent – results stay the same after multiple requests (GET HEAD, OPTIONS, TRACE).
  + PUT – still expects new resources.
  + DELETE – no more resources to expect (remove).
* Cacheable – results are dependent on message headers.